doi:10. 3772/j. issn. 1006-6748. 2023. 04. 010

Neural network hyperparameter optimization based on improved particle swarm optimization[①]

XIE Xiaoyan(谢晓燕)[*], HE Wanqi[②][*], ZHU Yun[**], YU Jinhao[*]
([*] School of Computer, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)
([**] School of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)

Abstract

Hyperparameter optimization is considered as one of the most challenges in deep learning and dominates the precision of model in a certain. Recent proposals tried to solve this issue through the particle swarm optimization (PSO), but its native defect may result in the local optima trapped and convergence difficulty. In this paper, the genetic operations are introduced to the PSO, which makes the best hyperparameter combination scheme for specific network architecture be located easier. Specifically, to prevent the troubles caused by the different data types and value scopes, a mixed coding method is used to ensure the effectiveness of particles. Moreover, the crossover and mutation operations are added to the process of particles updating, to increase the diversity of particles and avoid local optima in searching. Verified with three benchmark datasets, MNIST, Fashion-MNIST, and CIFAR10, it is demonstrated that the proposed scheme can achieve accuracies of 99.58%, 93.39%, and 78.96%, respectively, improving the accuracy by about 0.1%, 0.5%, and 2%, respectively, compared with that of the PSO.

Key words: hyperparameter optimization, particle swarm optimization (PSO) algorithm, neural network

0 Introduction

With the rapid growth of the artificial intelligence (AI) technologies, deep learning has achieved significant results in complex regression and classification problems, involving computer vision (CV), natural language processing (NLP), object detection[1], and so on. a neural network model needs to be trained to drive a satisfied accuracy. But there are a large number of hyperparameters which need to be configured during the training processing, such as learning rate, batch size, and optimizer. How to select appropriate hyperparameters to help training and explore the best neural network model has become a focus and a difficulty.

In the early studies, grid search was used to exhaustive search of parameter space. Later, an improved algorithm was implemented based on this called random search. Experiments have shown that, with the same number of search iterations, random search tries more parameter values compared with grid search, and reduces search time while ensuring model accuracy[2]. However, these search methods cannot solve the parameter optimization problem of neural network very well. Ref. [3] mentioned that hyperparameter optimization is an NP-hard optimization problem, and most current approaches solve it by adopting metaheuristic algorithms.

Ref. [4] successfully used genetic algorithm (GA) for hyperparameter tuning. It is found that more and more metaheuristic and modified algorithms could be used to optimize neural networks, and some researchers choose to extend different exploration[3]. The development of hybrid models can improve performance and the ability of complex optimization. Ref. [5] proposed an improved hybrid algorithm based on bat echo location behavior, combining local search to optimize weights, architectures, and active neurons. Ref. [6] introduced a combination of genetic and gradient descent to train networks. The proposed HGADD-NN method achieved good results on several benchmark problems.

Among most common algorithms, the particle

① Supported by the National Key Research and Development Program of China (No. 2022ZD0119003) and the National Natural Science Foundation of China (No. 61834005).
② To whom correspondence should be addressed. E-mail: eewgdg@yeah.net.
Received on Sep. 21, 2022

swarm optimization (PSO)[7-8] is a more popular choice, compared with GA[9], grey wolf optimization algorithm (AWO)[10] and ant colony optimization (ACO), because of its less nodal parameters, efficient global search and easily concurrent processing. So, it is introduced into parameters selection of convolutional neural network (CNN). In Ref. [7], PSO is used for parameters decision of CNN and verified successfully in classification of CIFAR. This indicates that PSO is a feasible scheme for parameters optimization of CNN. To handle the different of value scopes and datatypes of hyperparameters, independent candidate particles were defined for different parameter in Ref. [7]. But the initial position selected randomly would result in local optima trapped and the weakly local searching ability would entail a long convergent stage. Such a weakness may lead to a biased model even with the cost of additional calculations. Therefore, the improved PSO or integrating other heuristic algorithms become a striking field. Ref. [11] proposed a distributed PSO (DPSO) mechanism. The particles were updated and allocated in the master, and the fitness of different particles was calculated using multiple slaves. This strategy automatically and globally searches for optimal hyperparameters and dramatically reduces training time compared with traditional PSO, thanks to the distributed training method. Unfortunately, this idea only uses a distribution way to tackle long training time in the parameter configuration process. The cost of calculation is ignored.

The key problem to this issue lies in balance of the diversity of particle swarms and cost of convergence. In this work, an improved solution of PSO is proposed to aid in locating the best hyperparameter combination of specific network architecture easier. The following specific contributions are made.

(1) To response to the troubles caused by inconsistent data types and widely different value scopes of neural networks, the interval mapping method is adopted to data coding. The motivation of such design is to ensure the effectiveness of particles through a normalized strategy and to avoid the local evolution of swarm stem from randomly position selecting of original particles.

(2) By introducing mutation and crossover operations to increase the diversity of particles, the proposed algorithm solves the problems of the raw PSO, such as being easily trapped in local optima and low convergence accuracy during hyperparameter searching.

The rest of this paper is organized as follows. Section 1 discusses related work. Section 2 analyzes the motivation and describes the implementation of proposed algorithms in detail. Section 3 discusses the experimental environment and implementation scheme on different network structures and datasets. Section 4 summarizes the content of this paper.

1 Related work

In PSO[12], the optima seeking is converted to a process of traversing a n-dimensional function with particles of a swarm. Each a potential solution to a given problem is viewed as a particle. PSO obtains the best solution from interaction of particles. When mapping the different value scope of hyperparameters into a n-dimensional function, selecting of the best parameter combination can be figured out by PSO.

If there are N hyperparameters in the specific CNN, the value of each parameter ranges from low to up. A particle can be denoted as $\{x_1, x_2, \cdots, x_{N-1}, x_N\}$, and the performance is evaluated by a fitness function (often the loss function in CNN). The swarm is constructed originally with particles generated randomly by predefined value. The fitness value of each particle is calculated iteratively until reaching its best position or meeting the pre-set termination condition. The prior personal best position (g_{best}) and the global best position (p_{best}) is characterized as the intermediate results. Updating of particles can be given by Eq. (1) and Eq. (2).

$$v_{id} = wv_{id} + c_1 r_1 (p_{\text{best}_{id}} - x_{id}) + c_2 r_2 (g_{\text{best}_{id}} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

where, v represents the velocity vector; w is the inertia weight utilized to balance the local exploitation and global exploration; r_1 and r_2 are random vectors uniformly distributed within the range of $[low, up]$; c_1 and c_2 are acceleration coefficients, which are positive constants.

Above scheme, mentioned in Ref. [7] had been demonstrated be helpful to parameters tuning. However, there are many kinds of hyperparameter included in CNN, their datatype is also inconsistent (the number of convolutional kernels is recorded as integer, the learning rate is float, etc.). Moreover, the value scope of parameters in different layers show an enormous difference. The uniform format in coding will merge the effective attribute of particles and results a calculation error. However, differentiated coding format will increase the complexity of the swarm. Additional, when solving the problem with a large number of dimensions or complex and large datasets, PSO shows poor-quality results, usually falling into local optima trap.

To cope with this tricky situation, a series of solutions are proposed in this paper. The composite variable-type particles are mapped into an interval to ensure the effectiveness of particles update and to prevent the poor results due to significant differences in data sampling intervals. Then, the selection, mutation, and crossover operations of GA are introduced to add diversity of particles and avoid getting stuck in local optima during the parameter search process. Finally, an improved particle swarm optimization (IPSO) algorithm is proposed based on the above two improvement measures.

2 Methods

2.1 Hybrid particle coding

The parameters of CNN to be optimized selected in this paper include the number of convolution kernels, learning rate and batch size, etc., as shown in Table 1.

Table 1 Parameters to be optimized

Variable	Scope	Parameter
Parm1	[1,6]	The size of the first layer convolution kernel
Parm2	[1,16]	The size of the second layer convolution kernel
Parm3	[1,84]	The number of fully connected layers in the first layer
Parm4	[1,120]	The number of fully connected layers in the second layer
Parm5	[4,256]	Batch Size
Parm6	[0.0001,0.1]	Learning rate

It can be seen from Table 1 that the scope of parameters of int-type ranges from 6 ×(Parm1) to 120 ×(Parm4). In addition, a float-type parameter (Parm6) is also involved. Such a variety of value maybe be leads to bias direction in updating procedure and unable to converge in severe cases. As a result, there needs an effective manner to remove this difference. To this end, the PSO advocate[7-8] can only independently characterizes particles for each parameter. As a result, initial value of each particle only occurs randomly in self-defined scope. Although the difference of parameter scope needs not to be considered, such a treatment may probably generate an unreachable position during searching. For example, in a two-dimensional space composed of Parm2 and Parm3, the value of Parm2 could not reach [1,3] and [8,11], marked as '☞' in Fig. 1, due to the limiting of random function. Subsequently, particles generated will not overall coverage to initial swarm, while it would trap in local optima in iteration.

To avoid such incidents, a random sampling function conformed to uniform distribution is introduced in our scheme, which uniforms the coding of overall parameters. Meanwhile, Eq. (3) is used to mapping scopes of parameters to a normalized uniform.

$$value = lowArea + (upArea - lowArea) \times value \tag{3}$$

where the lower bound and upper bound of a parameter are denoted as *lowArea* and *upArea* respectively. The distribution of initial particles is presented as '☞' in Fig. 1. It is obvious that particles scattered steadily in overall interval. The global coverage of search space and completeness of swarm are guaranteed very well.

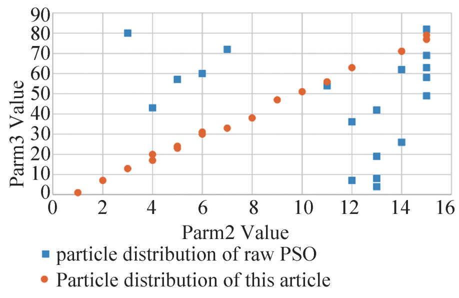


Fig. 1 Schematic diagram of particle distribution before and after optimization

2.2 Genetic manipulation

In this paper, in order to increase the diversity of particles during the updating, the selection, crossover, and mutation operations of GA are added to the IPSO. The cross process of particles is shown in Fig. 2. Suppose there are N particles in the swarm, which are denoted as $\{X_1, X_2, X_3, \cdots, X_{N-1}, X_N\}$. After the calculation, the fitness values of each particle are marked as $\{F(X_1), F(X_2), F(X_3), \cdots, F(X_{N-1}), F(X_N)\}$. Firstly, the first two particles with higher fitness values are selected and recorded as X_i and X_j. The inter-section point P is calculated using Eq. (4). Taking $N = 6$ as an example, the selection result is shown in Fig. 2 (a). Then, using the inter-section point P as the boundary, the particles X_i and X_j are divided into four parts: $X_{i-\mathrm{front}}$, $X_{i-\mathrm{after}}$, $X_{j-\mathrm{front}}$, and $X_{j-\mathrm{after}}$, as shown in Fig. 2 (b). Finally, $X_{i-\mathrm{front}}$ is spliced with $X_{j-\mathrm{after}}$, and $X_{j-\mathrm{front}}$ is spliced with $X_{i-\mathrm{after}}$, then two new particles $X_{i-\mathrm{new}}$ and $X_{j-\mathrm{new}}$ are formed, as shown in Fig. 2 (c).

$$p = \lceil N / 2 \rceil \tag{4}$$

The schematic diagram of the particle mutation is presented in Fig. 3, taking $N = 6$ as an example. Firstly,

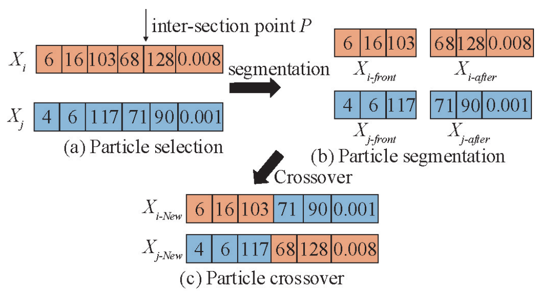

inter-section point P
X_i 6 16 103 68 128 0.008
X_j 4 6 117 71 90 0.001
(a) Particle selection
segmentation
6 16 103 | 68 128 0.008
$X_{i\text{-front}}$ $X_{i\text{-after}}$
4 6 117 | 71 90 0.001
$X_{j\text{-front}}$ $X_{j\text{-after}}$
(b) Particle segmentation
Crossover
$X_{i\text{-New}}$ 6 16 103 71 90 0.001
$X_{j\text{-New}}$ 4 6 117 68 128 0.008
(c) Particle crossover

Fig. 2 An example illustration of the crossover process

the complete information of the initial particle would be obtained. Then, an integer value in [1, N] will be randomly generated, N denotes the total number of optimized parameters. Finally, a randomly generated number within the corresponding parameter scope of that position will replace the original value, and the mutation operation will be done.

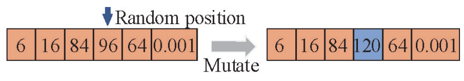


Fig. 3 Schematic diagram of mutation process

For the raw PSO method, particle position participating in the next iteration is determined by the optimal solution obtained from this searching round and the historical optimal solution. The update domain of raw PSO can be described by the zone, shown as dashed lines in Fig. 4. The updating domain of raw PSO is limited to such a domain. It is only stopped in the optima of such local domain. If the global optima are escaped out of this interval, the final result will be deviated to the correct one.

The GA operations will help to jump out of this limitation by involving new particle. It can enhance the diversity of particles in iteration of swarm, and make the global optima be located easier.

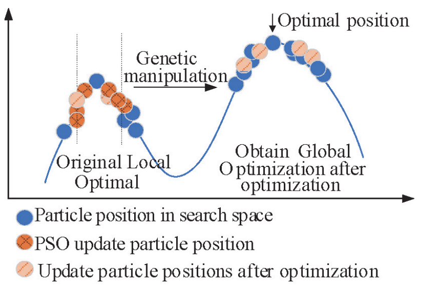

Optimal position
Genetic manipulation
Original Local Optimal
Obtain Global Optimization after optimization
● Particle position in search space
● PSO update particle position
● Update particle positions after optimization

Fig. 4 Comparison of particle positions before and after optimization

2.3 Overall scheme

The flow chart of hyperparameter configuration based on the proposed IPSO is shown in Fig. 5, there are 3 steps as follows.

Step 1 Initialization of swarm and its particles. The learning factors c_1, c_2, weight coefficient w, number of particles, and maximum particle speed need to be initialized at beginning. At the same time, it opens up a space to store local and global optimal values and randomly generates a specified number N of particles to form a swarm according to the number and scope of parameters to be optimized.

Step 2 Calculate the fitness value of the particles. The values of each particle are sequentially sent to the designated neural network for training and testing, and the accuracy rate obtained will be selected as the fitness value of the particle and recorded.

Step 3 Particles updating. According to Algorithm 1, the particles in the current swarm are updated to complete the subsequent iteration.

Algorithm 1 Particle updating

```
Input: Primitive swarm $D(X)$, Fitness function $F(X)$
Output: The updated swarm $D(X)$
1: function update ($D(X)$)
2: Initialization $max_1$ and $max_2$
3 for: $i$ in length ( $D(X)$ )
4: Evaluate $F(X_i)$
5: set $max_1$ and $max_2$
6: end for
7: update $g_{best}$ and $p_{best}$
8: $new_1$ , $new_2$ = crossover ( $max_1$ , $max_2$ )
9: D(X) = mutate ( $new_1$ , $new_2$ )
10: return $D(X)$
11: end function
```

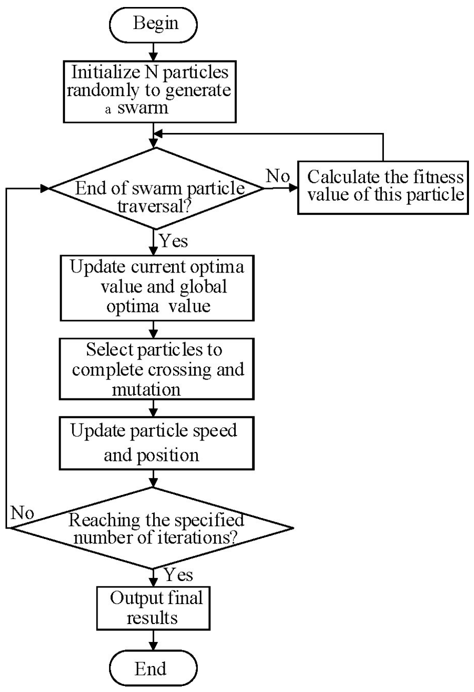

Begin
Initialize N particles randomly to generate a swarm
End of swarm particle traversal?
No
Calculate the fitness value of this particle
Yes
Update current optima value and global optima value
Select particles to complete crossing and mutation
Update particle speed and position
No
Reaching the specified number of iterations?
Yes
Output final results
End

Fig. 5 Flow chart of the IPSO

3 Experiments and results analysis

3.1 Implementation details

The initialization of the algorithm is completed according to the parameters shown in Table 2. The proposed algorithm is verified under different network structures and data sets. The experimental environment of this paper is shown in Table 3.

Table 2 IPSO algorithm related parameters

Name	Value	Meaning
particleNum	20	The number of particles in the population
epochMax	40	Maximum number of search iterations
c_1, c_2	2	learning factor
w	0.5	inertia factor
r_1, r_2	np. random. uniform (0,1)	random number

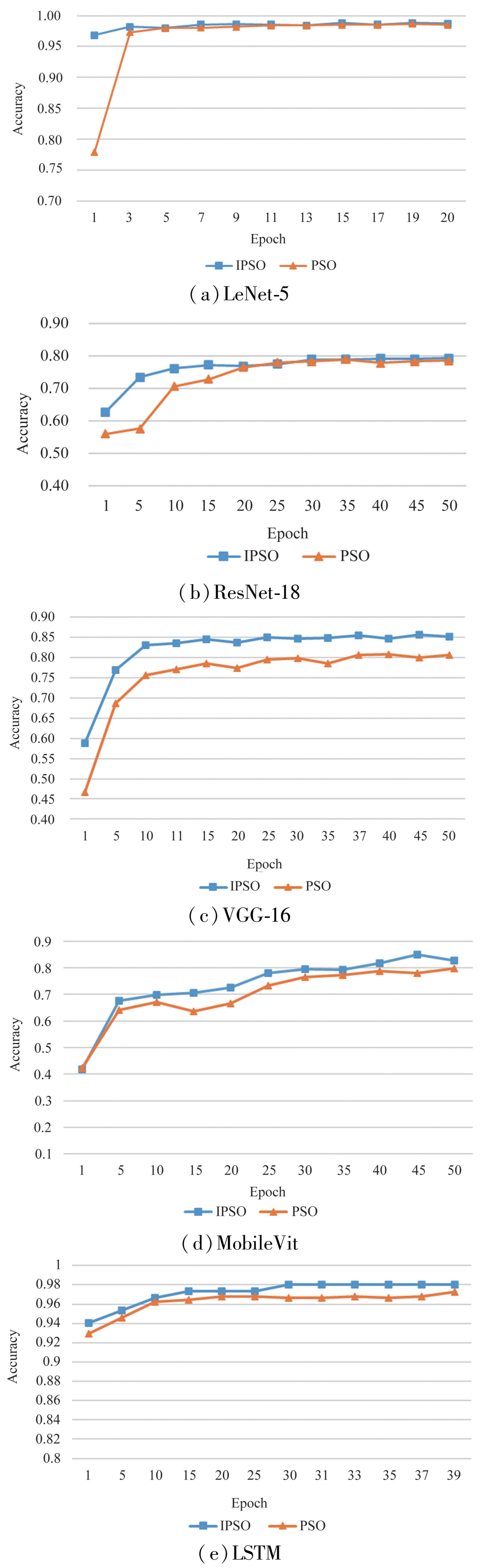
Table 3 Experimental environment setting

Parameter	Parameter details
System	Ubuntu 18.04
Processor	NVIDIA GeForce RTX 3090
Memory	64 GB
Data storage	2 TB SSD

3.2 Experimental results and analysis

3.2.1 Under different network structures

In this work, LeNet-5, ResNet-18, VGG-16, MobileViT, and Long short-term memory (LSTM) are selected as the test objects. The MNIST was used on LeNet-5 and LSTM, the Flowers-Recognition was used on MobileViT, and the remaining networks tested with the CIFAR10. This presearch uses the parameters searched by the IPSO and PSO to complete the training task. The accuracy rate changes with epoch during the training are shown in Fig. 6. It can be seen that for the same training epoch, no matter which neural network is used, the parameter configuration searched by the IPSO can make the neural network converge faster than the PSO. In addition, Fig. 6 (c) and Fig. 6(d) show more apparent difference in accuracy.



(a) LeNet-5
(b) ResNet-18
(c) VGG-16
(d) MobileVit
(e) LSTM

Fig. 6 Accuracy rate change with Epoch

Using the IPSO and the parameters searched by the PSO to complete the training, the final accuracy rate is compared in Table 4. It is clear that, after structure tuned by IPSO, the CNN model can finally obtain higher accuracy, compared with tuned by PSO, among which the accuracy rates are increased by about 0. 4% , 1. 9% , and 4. 5% , respectively. On the ViT model, the network built by the IPSO can finally complete the image classification tasks with an accuracy rate of 66. 21% , about 13% higher than that by the raw PSO. IPSO and PSO have achieved similar accuracy on the recurrent neural network. On the ViT model experiment, IPSO and PSO only have a difference of 1 in parameter 1, but the final network structure gains a significant difference due to the layer-by-layer accumulation. Meanwhile, although the data number of Flowers-Recognition used in this paper is small, different parameter configurations will still bring obvious difference of performance. LSTM is more inclined to the application of NLP. At the same time, this paper only considers two network-related parameters, the hidden state and the number of recurrent network layers. Although the difference between them will still change the number of parameters, the extraction effect of changing data feature is not obvious, which can be to negligible.

Table 4 Comparison of the accuracy of different algorithms on different network structures

Model	LeNet-5	ResNet-18	VGG-16	Mobile-ViT	LSTM
PSO	98. 52%	77. 02%	80. 67%	53. 63%	99. 48%
IPSO(this paper)	98. 98%	78. 96%	85. 13%	66. 21%	99. 58%

3. 2. 2 Under different datasets

To get the effectiveness of the IPSO proposed in this paper on the hyperparameter seeking, three datasets of MNIST, Fashion-MNIST and CIFAR10 are selected for verification.

(1) The MNIST dataset includes 70 000 handwritten digital images. The training set contains 60 000 samples, while the test set contains 10 000 samples. The dataset is categorized into 10 classes related to 10 numerals.

(2) The Fashion-MNIST dataset contains 10 categories of images: T-shirt, trouser, puller, dress, coat, sandy, shirt, sneaker, bag, and ankle boot. The training dataset has a total of 60 000 samples, and the test dataset has a total of 10 000 samples.

(3) The CIFAR-10 dataset includes 60 000 color images from 10 categories, with 6000 images per category. These 10 different categories represent aircraft, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks. The training set contains 50 000 samples (83% of the original dataset), while the test set contains 10 000 samples (17% of the original dataset).

The parameters searched by the IPSO on the above three data sets can have 99. 58% , 93. 39% , and 78. 96% accuracy when training the neural network. Compared with PSO, the accuracy has increased by about 0. 1% , 0. 5% , 2% , respectively. Compared with the SSO[13], the same accuracy has been achieved on the MNIST dataset meanwhile. On Fashion-MNIST and CIFAR10, the accuracy has increased by 0. 36% and 5. 83% respectively. Compared with the DPSO[11], the model accuracy obtained using IPSO on MNIST and Fashion MNIST is slightly higher, and the final results are shown in Table 5. In addition, in order to compare with the WOA[10], the network's optimization parameters are adjusted to be consistent with those in GWO, including batch size, epochs, and optimizer. The results are shown in Table 6. From the table, it can be seen that the IPSO algorithm's search for parameter combinations completed training on the Fashion-MNIST dataset has a higher accuracy on the test set than GWO algorithm.

Table 5 Comparison of the accuracy of different algorithms on different network structures

	MNIST	Fashion-MNIST	CIFAR10
PSO[7]	99. 48%	91. 92%	77. 02%
SSO[13]	99. 58%	92. 75%	73. 13%
DPSO[11]	99. 20%	92. 34%	-
IPSO(this paper)	99. 58%	93. 39%	78. 96%

Table 6 Accuracy comparison on the Fashion-MNIST dataset

	Batch Size	Epoch	Optimizer	Accuracy
Random Search[10]	128	40	Adamax	89. 41%
Grid Search[10]	16	45	Adamax	89. 88%
GWO[10]	512	30	Adamax	89. 75%
IPSO(this paper)	50	40	Adam	90. 28%

4 Conclusion

This paper proposes an IPSO algorithm that integrates GA to address the issues of traditional PSO, such as easily falling into local optima and low convergence accuracy in the process of seeking. Finally, the performance is verified by using different types of neural network models and datasets. Experimental results show that the proposed IPSO achieves higher accuracy than traditional PSO on CNNs and ViT models, tested with Fashion-MNIST and CIFAR10 datasets. Moreover, with optimized parameter configurations, the model is more stable and converges faster during training. However, this paper only considers a limited number of parameters to be optimized, while other parameters affecting the neural network structure, such as the depth and number of convolutional layers, can also be searched for the optimal solution. Therefore, in the future, it is worth considering the fusion of parameters at different levels to find the optimal network structure and model parameters for better model performance.

References

[1] POUYANFAR S, SADIQ S, YAN Y, et al. A survey on deep learning: algorithms, techniques, and applications [J]. ACM Computing Surveys, 2018, 51(5):1-36.

[2] BERGSTRA J, BENGIO Y. Random search for hyper-parameter optimization[J]. Journal of Machine Learning Research, 2012, 13(1):281-305.

[3] KAVEH M, MESGARI M S. Application of meta-heuristic algorithms for training neural networks and deep learning architectures: a comprehensive review [J]. Neural Processing Letters, 2022, 55: 4519-4622.

[4] YOUNG S R, ROSE D C, KARNOWSKI T P, et al. Optimizing deep learning hyper-parameters through an evolutionary algorithm [C]//Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. New York: Association for Computing Machinery, 2015: 1-5.

[5] JADDI N S, ABDULLAH S, HAMDAN A R. Optimization of neural network model using modified bat-inspired algorithm[J]. Applied Soft Computing, 2015, 37:71-86.

[6] SOHEIL G AND MORTEZA T. Training qubit neural network with hybrid genetic algorithm and gradient descent for indirect adaptive controller design[J]. Engineering Applications of Artificial Intelligence, 2017, 65: 346-360.

[7] SINHA T, HAIDAR A, VERMA B. Particle swarm optimization based approach for finding optimal values of convolutional neural network parameters[C]//2018 IEEE Congress on Evolutionary Computation. Rio de Janeiro: IEEE, 2018: 1-6.

[8] MIHAI A C, IANCU D T. Optimizing a convolutional neural network using particle swarm optimization[C]// 2022 14th International Conference on Electronics, Computers and Artificial Intelligence. Ploiesti: IEEE, 2022: 1-7.

[9] HAN J H, CHOI D J, PARK S U, et al. Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network[J]. Journal of Electrical Engineering and Technology, 2020, 15: 721-726.

[10] BRODZICKI A, PIEKARSKI M, JAWOREK-KORJAKOWSKA J. The whale optimization algorithm approach for deep neural networks[J]. Sensors, 2021, 21(23): 8003.

[11] GUO Y, LI J Y, ZHAN Z H. Efficient hyperparameter optimization for convolution neural networks in deep learning: a distributed particle swarm optimization approach[J]. Cybernetics and Systems,2020, 52(1): 36-57.

[12] GAD A G. Particle swarm optimization algorithm and its applications: a systematic review[J]. Archives of Computational Methods in Engineering, 2022, 29(5): 2531-2561.

[13] YEH W C, LIN Y P, LIANG Y C, et al. Simplified swarm optimization for hyperparameters of convolutional neural networks[J]. Computers and Industrial Engineering, 2023,177: 1-11.

XIE Xiaoyan, born in 1972. She is currently a professor in Xi'an University of Posts & Telecommunications. She received the B. S. degree in Computer Communication from Nanjing University of Science and Technology in 1995. She received her M. S. degree in Computer Science Department of Xidian University in 2002. She is now concentrating on reconfigurable computing and computer vision.