International Journal of Computer Applications (0975 - 8887)
Volume 182 - No.44, March 2019

Improved PSO Algorithm for Training of Neural Network in Co-design Architecture

Tuan Linh Dang
Hanoi University of Science and Technology
Hanoi City
100000, Vietnam

Yukinobu Hoshino
Kochi University of Technology
Tosayamada, Kami City
Kochi 782-8502, JAPAN

ABSTRACT

This paper proposes a new version of the standard particle swarm optimization (SPSO) algorithm to train a neural network (NN). The improved PSO, called the wPSOd_CV algorithm, is the improved version of the PSOd_CV algorithm presented in a previous study. The wPSOd_CV algorithm is introduced to solve the issue of premature convergence of the SPSO algorithm. The proposed wPSOd_CV algorithm is used in a co-design architecture. Experimental results confirmed the effectiveness of the NN trained by the wPSOd_CV algorithm when compared with the NN trained by the SPSO algorithm and the PSOd_CV algorithm concerning the minimum learning error and the recognition rates.

General Terms

Neural network, Particle swarm optimization, co-design architecture

Keywords

Neural network, Particle swarm optimization, FPGA, ARM, co-design architecture

1. INTRODUCTION

Today, a neural network (NN) has become a hot topic in the research. Many studies focus on the using of NN in different aspects [1, 2].
The NN is invented to represent a human brain. In order to be used in the testing phase, the NN need to be trained in the training phase [3–5]. The back-propagation (BP) algorithm has investigated in previous studies. However, the PSO algorithm has shown the advantages in the training of the NN compared with the using of BP to train the NN. The NN trained by the PSO algorithm had obtained higher accuracy concerning the learning error and the recognition rate than the NN trained by conventional BP algorithm [6–9].
The PSO is the algorithm based on the social behavior of a swarm. During the operation, when one particle p in the swarm found a better location than existing locations, all particles in the swarm will follow this particle p [10, 11].
The previous has investigated the NN trained by the PSO algorithm. However, normally the previous studies have used only the standard PSO (SPSO) algorithm that could stick to a local minimum to train the NN. In this case, the NN will have a low recognition rate in the testing phase and higher learning error in the training phase.
The classification tasks of the NN trained by PSO (NN-PSO) have been investigated in previous studies. However, these studies used only standard PSO (SPSO) that may easily stick to a local minimum during the training phase [12–14]. In this situation, the training phase will be stopped immediately. This leads to a very high learning error and a low recognition rate. Therefore, it is necessary to have a PSO algorithm that solves the premature convergence of the SPSO algorithm.
In the previous research, a new architecture for co-design between hardware and software was proposed. This architecture had not only the advantages of the software side but also the advantage of the hardware side. In this architecture, the PSO is implemented in software. Thus, in the training phase, it is very easy to modify the parameters of the PSO algorithm. In addition, the PSO is not needed in the testing phase. So, the operating speed of the testing phase is not affected by the software side. On the other hand, the NN is implemented in FPGA to increase the operation speed. The NN is implemented in FPGA so that the hardware speed could be maintained [15–17].
In the previous study, an improved version of the SPSO was proposed to solve the problem of the local minimum called the PSOd_CV algorithm. Experimental results demonstrated that the PSOd_CV algorithm achieved better accuracy than the SPSO algorithm. This algorithm has a big jump to help the particle can be moved out of the local minimum. However, the inertia weight in the PSOd_CV algorithm is fixed. The inertia weight is the trade-off between the exploration and the exploitation. With the high value of inertia weight, the algorithm will focus on the exploitation task to search in the global area. On the other hand, with the lower value of the inertia weight, the exploration task will be conducted to focus on the local area [16, 18].
The main contribution of this paper is to propose an improved version of the PSOd_CV algorithm called the wPSOd_CV algorithm to solve the premature convergence of the SPSO algorithm. All three algorithms (SPSO, PSOd_CV, and wPSOd_CV) will be investigated in the experiments with several publicly recognized databases.
The paper is presented as follows. Sections 2 presents the NN-wPSOd_CV system which has a neural network trained by particle swarm optimization algorithms. The wPSOd_CV is also proposed in this section. The NN-wPSOd_CV is based on the NN-PSO framework presented in the previous papers [15–17]. Section

3 details the experiments in NN-wPSOd_CV along with results. Section 4 concludes this paper.

2. NN-wPSOd-CV

2.1 Neural network

The neural network was invented to represent the human brain. Normally, the NN has three different of layers called the input layer, the output layer, and the hidden layer [3–5]. The D number of weights and biases of the NN can be seen in Eq. (1).
The outputs the input layer will become the input of the hidden layer, and the output of the hidden layer will become the input of the output layer through the activation function. The activation function will fire or the output has the value of one if the input if the input is higher than threshold [3–5]. The conventional activation function is the Sigmoid function as can be seen in Eq. (2).

$$D = (N_I+1)\times N_H+(N_H+1)\times N_H\times N_L+(N_H+1)\times N_O \quad (1)$$

where N_I, N_H, and N_O are the numbers of the nodes in the input layer, the hidden layer, and the output layer. N_L is the number of hidden layers as presented in Fig. 1.

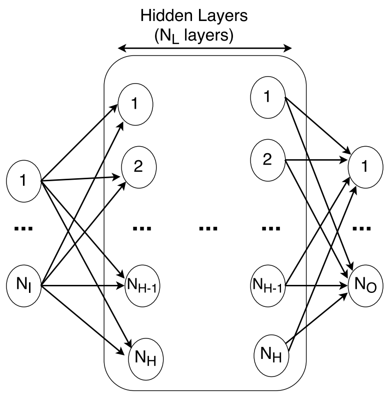


Fig. 1: Neural network

$$S(data) = \frac{1}{1+e^{-data}} \quad (2)$$

where $data$ is the input data of the Sigmoid function, and $S(data)$ is the output of the activation function.

2.2 Neural network trained by standard PSO

To be used, the NN needs to be trained. A famous training method is the back-propagation (BP) algorithm. However, the previous paper mentioned the accuracy advantage of the NN trained by PSO when compared with the NN trained by BP algorithm [6–9]. The previous paper proposed a framework for the co-design of the NN trained by PSO algorithm [15–17].
The original of the SPSO is from social behaviors. When one particle in the swarm found a better position than the current best position, the new best position is updated. In addition, other particles in the swarm have the tendency to move to the position of new best position [10, 11].
The operation of the NN trained by the SPSO (NN-PSO) algorithm is based on the operations of the SPSO [10, 11]. The NN-PSO can be detailed as follows.

(1) The weights and the biases of the neural network are encoded into the PSO particles. The dimension of one particle equals the number of weights and biases of the neural network. In the initial phase at time t, the positions of all particles are randomly generated. Thus it has:
—Position of particle p is $\boldsymbol{x}_p(t)$
—Velocity of particle p is $\boldsymbol{v}_p(t)$
—Best personal position of particle p is $\boldsymbol{x\_Pbest}_p(t)$
—Best global position of all particles in the swarm is $\boldsymbol{x\_Gbest}(t)$
—Fitness value for the best personal position of particle p is $Pbest_p(t)$
—Fitness value for the best global position of all particles is $Gbest(t)$

(2) At next time $t+1$, calculate the new velocity according to Eq. (3)

$$\boldsymbol{v}_p(t+1) = w\times \boldsymbol{v}_p(t) + c_1\times r_1(\boldsymbol{x\_Pbest}_p(t) - \boldsymbol{x}_p(t)) + c_2\times r_2(\boldsymbol{x\_Gbest}(t) - \boldsymbol{x}_p(t)) \quad (3)$$

where w is inertia weight, c_1 and c_2 are coefficients.

(3) The new position at time $t+1$ is also evaluated based on Eq. (4). Each position of a particle is considered as one set of potential parameters for the NN. When the particle has a new position, the corresponding weights and the biases of the NN are also changed. Each particle has D dimensions. D is also the size of the NN that can be calculated by Eq. (1).

$$\boldsymbol{x}_p(t+1) = \boldsymbol{x}_p(t) + \boldsymbol{v}_p(t+1) \quad (4)$$

(4) The new fitness values $Pbest_p(t+1)$ and $Gbest(t+1)$ at time $t+1$ are calculated. The calculation of new fitness values uses the output data from the NN. The parameters of the NN come from the position of the PSO particles. The output data from NN will be evaluated with the labeled data by mean squared error function as seen in Eq. (5).
The minimum learning error of each particle ($Pbest$) and the global minimum learning error ($Gbest$) of all particles are calculated

$$f_i = \frac{1}{T}\sum_{j=1}^{T}(labeled_j(k) - output_{ij}(k))^2 \quad (5)$$

where T is the number of training samples, $labeled(k)$ and $output(k)$ are the k^{th} component of the particle i in the labeled data and the output data of the NN.

(5) the stopping condition is checked.
—If the condition is satisfied, the training phase of the NN trained by PSO is stopped, the final $Gbest$ is found. Hence, the final position $\boldsymbol{x\_Gbest}(t)$ corresponds to the final $Gbest$ is also found. This final position corresponds to the weights and biases of the NN after the training phase.
—If the condition is not satisfied. A new iteration of the training will be conducted. The training phase return to step 2.

2.3 Hardware implementation of the NN-SPSO

The framework for the hardware implementation of the NN-SPSO is based on the previous studies. This is co-design architecture. One part of the system is on the hardware side, another part is on software side [15–17]. The system can be seen in Fig. 2.

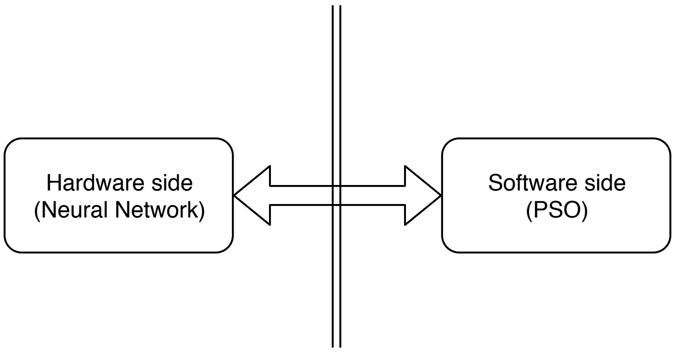


Fig. 2: NN-PSO system

The PSO algorithm is implemented on the software side. The PSO is only used in the training phase to determine the weights and the biases of the NN. In addition, during the training, the software-based PSO may easy to change the parameter such as the inertia weight w, the coefficients c_1, c_2, or even the new PSO algorithm can be used with the need to redesign or rebuild the FPGA side.
The NN is coded on the hardware side using SystemVerilog programming language to take the speed advantage of the hardware-based program when compared with the conventional software-based program. The hardware-based normally has a higher operating speed thanks to the parallelism. The NN is the only component used in the testing phase, therefore the speed of the software-based algorithm does not affect the system.
In the training phase, the D weights and biases of the NN are encoded in the PSO particles. Each PSO particle is a D-dimensional vector. As presented, D is calculated according to Eq. (1). The position of each particle corresponds to a potential set of weights and biases for the NN. The PSO algorithm is conducted to find the $Gbest$ fitness value and also the corresponding position of the $Gbest$. The found position of the particle is the trained weights and biases of the NN. These trained parameters are sent to the hardware side (NN module). The NN keeps the trained weights and biases and uses these parameters in the testing phase.

2.4 Neural network trained by improved particle swarm optimization

2.4.1 Linearly decreasing inertia weight. The linearly decreasing inertia weight was presented to improve the performance of the standard PSO algorithm by using a strategy for weight control. This algorithm reduces the weights by iterations as can be shown in equation 6. The linearly decreasing inertia weight strategy has two tasks called the exploration and the exploitation, respectively. The particles in the swarm do a global search at the beginning (the exploitation). When inertia weight w is small, the particles conduct the exploration. This inertia weight control has the possibility to search and find the local solutions [19].

$$w = w_{max} - \frac{w_{max} - w_{min}}{N_{iteration}} \times iteration \qquad (6)$$

where $N_{iteration}$ is the number of iterations, w is the inertia weight.

2.4.2 Particle swarm optimization with control of velocity and inertia weight. As described in the previous section, the PSOd_CV algorithm always has the big jumps to explore a wide area in the jumping phase. Therefore, this algorithm has the possibility to skip local solutions in high dimensional problems. In the PSOd_CV algorithm, the value of the inertia weight w is fixed. However, the control of this weight has an impact on the PSOd_CV algorithm. The inertia weight control is the trade-off between the exploration and the exploitation. With a significant value of w, the PSO algorithm tends to search the global area. On the other hand, with a small value of w, the PSO algorithm tends to focus on the local area. A famous technique for weight control is the linear decreasing strategy. This strategy focuses on the exploitation task at the beginning of the algorithm. After that, the exploration job is addressed to search in the local area [19]. This strategy has the chances to find local solutions.
This paper proposes the PSO with control of velocity and inertia weight algorithm (wPSOd_CV algorithm). This algorithm is the combination of the PSOd_CV algorithm and the inertia strategy. Thus, this algorithm may overcome the disadvantage of the PSOd_CV algorithm. The wPSOd_CV algorithm is given in equation (7).

$$\begin{aligned} \boldsymbol{v}_p(t+1) = w_{max} - \frac{w_{max} - w_{min}}{N_{ite}} \times ite \times \boldsymbol{v}_p(t) \\ +c_1 \times (\boldsymbol{x\_Pbest}_p(t) - \boldsymbol{x}_p(t)) \qquad (7) \\ +c_2 \times (\boldsymbol{x\_Gbest}(t) - \boldsymbol{x}_p(t)) + \frac{c_3 \times r}{(\boldsymbol{v}_p(t))^2} \end{aligned}$$

where N_{ite} is the number of iterations, ite is the current iteration, c_1, c_2, c_3 are the coefficients, r is the random number.

3. EXPERIMENTS

The experiments were conducted with the DE1-SoC development board provided by Altera [21]. The Hardware based NN is coded in SystemVerilog programming language while the PSO algorithms were implemented in ARM processor attached to the DE1-SoC board. The connection between the hardware side and the software side was done by direct memory access (DMA) mechanism.
The experiments investigated three different PSO algorithms which were SPSO, PSOd_CV presented in the previous research, and the proposed wPSOd_CV algorithm.
Based on the tests, a suitable set of parameters for all three algorithms was as follows.

(1) w = 0.7, $c1$ = 0.5, $c2$ = 0.3 in the standard PSO algorithm.
(2) w = 0.7, $c1$ = 0.5, $c2$ = 0.3, $c3$ = 0.00001 in the PSOd_CV algorithm.
(3) w_{max} = 0.7, w_{min} = 0.3, $c1$ = 0.5, $c2$ = 0.3, $c3$ = 0.00001 in the wPSOd_CV algorithm.

Four different databases were used in the experiments called XOR problem, Iris dataset, Balance-scale dataset, and Credit approval dataset.

3.1 XOR problem

The first experiment was conducted with the XOR problem, a non-linear problem. The configurations of the NN were two input nodes, six hidden nodes, and four output nodes. The parameters of this experiment were 50 particles, 150 iterations.
The measurement of three algorithms (standard PSO, PSOd_CV, and wPSOd_CV) was the $Gbest$, the global minimum value of the learning error. Fig. 3 shows the reduction of the $Gbest$. In all three algorithms, the $Gbest$ had a rapid reduction period. In the next period, the $Gbest$ declined slowly. The decreased value of $Gbest$ may

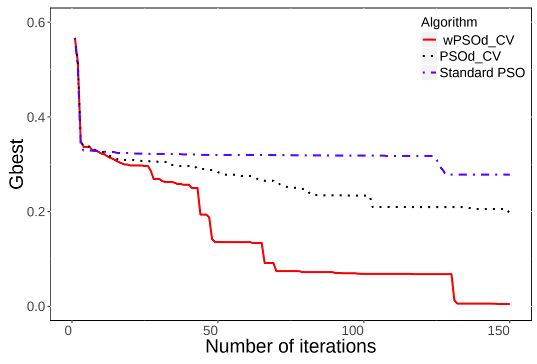
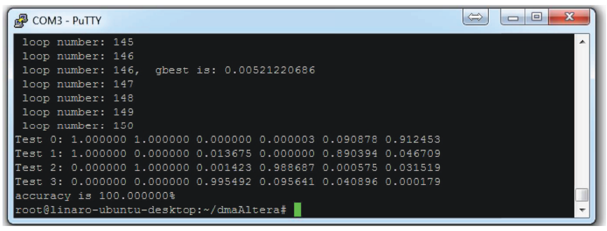


Fig. 3: The reduction of *Gbest*

be subtle. For example, the *Gbest* of the standard PSO was 0.323 in iteration 17 and this number reduced to 0.321 in iteration 38. The final *Gbest* after 150 iterations of the standard PSO algorithm was 0.278, the PSOd_CV algorithm was 0.195, and the wPSOd_CV algorithm was 0.005. The wPSOd_CV algorithm obtained better performance than the other two algorithms in this experiment.
The results of the co-design program were observed in the PuTTY software. For example, the results of the NN trained by the wPSOd_CV algorithm with the XOR problem were shown in Fig. 4. For reference, Table I shows the training data.



```
 loop number: 145
 loop number: 146
 loop number: 146,  gbest is: 0.00521220686
 loop number: 147
 loop number: 148
 loop number: 149
 loop number: 150
Test 0: 1.000000 1.000000 0.000000 0.000003 0.090878 0.912453
Test 1: 1.000000 0.000000 0.013675 0.000000 0.890394 0.046709
Test 2: 0.000000 1.000000 0.001423 0.988687 0.000575 0.031519
Test 3: 0.000000 0.000000 0.995492 0.095641 0.040896 0.000179
accuracy is 100.000000%
root@linaro-ubuntu-desktop:~/dmaAltera#
```

Fig. 4: The output in PuTTY software

Table I. : The training data of the XOR problem

Input	Output1	Ouput2	Outpu3	Output4
1 1	0	0	0	1
1 0	0	0	1	0
0 1	0	1	0	0
0 0	1	0	0	0

3.2 Iris dataset

In the previous experiment, the NN was capable of solving the XOR problem. It is needed to investigate the operation of the proposed co-design with a classification dataset comes from daily life. However, this research uses Cyclone V which has the lowest number of resources in the Altera device family [20]. Thus, the Iris dataset was chosen. This dataset has three different classes called Setosa, Versicolour, and Virginica. Each class has 50 samples. Each sample has four different attributes (the sepal length, the sepal width, the petal length, and the petal width) [22].

The NN in this experiment had four input nodes (for four attributes), three output nodes (for three classes), ten hidden nodes, and two hidden layers. The configuration was 4-10-10-3.
The Iris dataset was separated into two sets. The bigger set had 105 samples (each class had 35 samples). On the other hand, the smaller set had 45 samples (each class had 15 samples).
Several experiments were conducted to investigate the operation of three algorithms in different situations. Therefore, various values for the number of training data, the number of iterations and the number of particles were chosen.
In the first scenario, the bigger set was used as the training data and the other set was considered as the testing data. The number of particles was modified from a small value to a big value.
Fig. 5 and Fig. 6 describe the reduction of *Gbest*, the global minimum value of learning error. In these experiments, the number of particles was 20, and the numbers of iterations were 100 and 340. In both experiments, the final *Gbest* of the wPSOd_CV algorithm decreased to the lowest value (0.1774 in the case of 100 iterations and 0.0510 in the case of 340 iterations). The final *Gbest* of the PSOd_CV algorithm was 0.1944 (100 iterations) and 0.0883 (340 iterations). The final *Gbest* of the standard PSO algorithm was 0.2012 in both scenarios. Concerning the recognition rate, the wPSOd_CV also obtained a higher recognition rate (77.78% in 100 iterations and 93.33% in 340 iterations) than the PSOd_CV algorithm (66.67% in 100 iterations and 93.33% in 340 iterations) and the standard PSO algorithm (only 66.67%). Therefore, the wPSOd_CV algorithm may obtain a low *Gbest* and a high recognition rate even with the small number of particles.

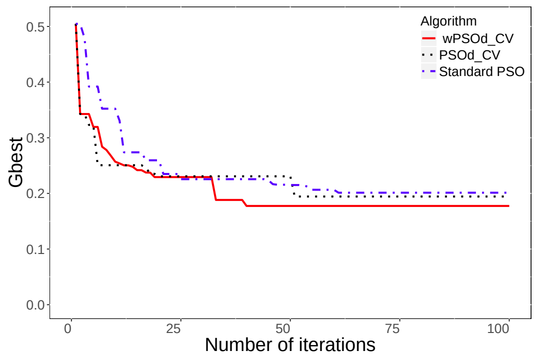


Fig. 5: The reduction of *Gbest* when $P = 20$ particles, 100 iterations

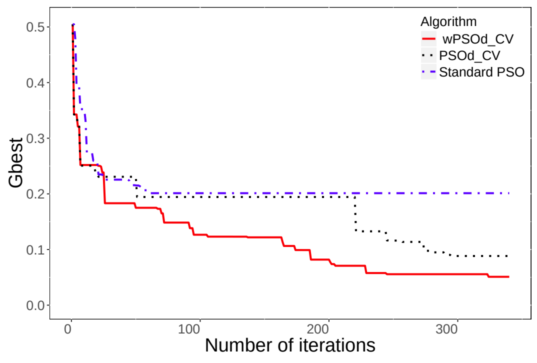


Fig. 6: The reduction of *Gbest* when $P = 20$ particles, 340 iterations

In the next experiments, the number of particles was changed to increase the recognition rate and to reduce the final *Gbest*. Fig. 7 presents the *Gbest* in all three algorithms when the number of particles $P = 42$. Table II shows the final *Gbest* and the recognition rate when $P = 100$. As seen in Table II and Fig. 7, the wPSOd_CV algorithm had a lower *Gbest* and also had a better recognition rate than the PSOd_CV algorithm and the standard PSO algorithm.

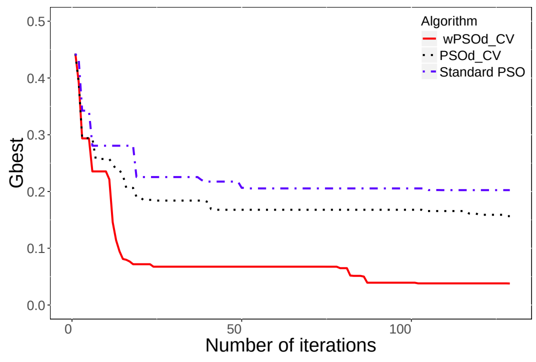


Fig. 7: The reduction of *Gbest* when $P = 42$ particles

Table II. : The results of three different PSO algorithms when $P = 100$ particles

Number of iterations	Algorithm	*Gbest*	Recognition rate
100	Standard PSO	0.0669	93.33%
	PSOd_CV	0.0541	95.56%
	wPSOd_CV	0.0457	100.00%
230	Standard PSO	0.0577	97.78%
	PSOd_CV	0.0368	100.00%
	wPSOd_CV	0.0210	100.00%

Another experiment was conducted to investigate the operation of the proposed system with a small number of training data. In this experiment, the smaller set (45 samples) was used as the training samples. The number of particles and the number of iterations were chosen randomly ($P = 80$ particles, 40 iterations).
Fig. 8 illustrates the *Gbest* of three algorithms. The wPSOd_CV algorithm still produced better results than the PSOd_CV algorithm and the standard PSO algorithm (the final *Gbest* and the recognition rate of wPSOd_CV were 0.0277 and 95.28%, respectively). The results of the PSOd_CV algorithm were *Gbest* = 0.0284, recognition rate = 94.29%. The results of the standard PSO algorithm were *Gbest* = 0.1698, the recognition rate = 76.19%.

3.3 Balance-scale dataset

Another dataset, called Balance-scale dataset, was also tested in order to investigate the operation of the NN trained by three different PSO algorithms in different situations. This dataset has the same number of classes and attributes with the Iris dataset. Thus, the configuration of the NN in this experiment was similar to the configuration of the NN in the Iris experiment (4-10-10-3). Three attributes of this dataset are “right state”, “left state”, and “balance state”, respectively. The names of four attributes are “left weight”,

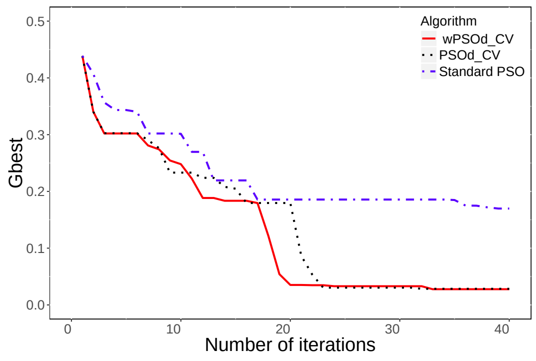


Fig. 8: The reduction of *Gbest* with 45 samples as training data

“left distance”, “right weight”, “right distance”. The values of the attributes are from one to five.
The Balance-scale dataset was also divided randomly into two sets. The first set which had 200 samples was considered as the training set. On the other hand, the second set that contains 60 samples was selected as the testing set.
The configurations for the PSO were 200 iterations, 50 particles.

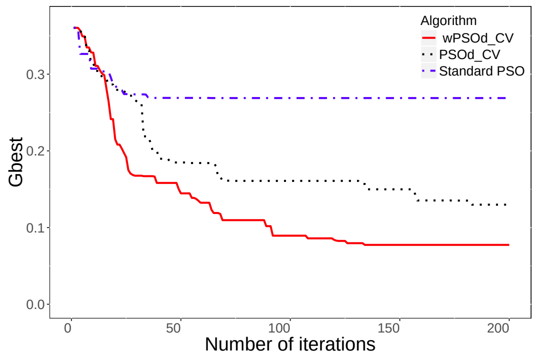


Fig. 9: The reduction of *Gbest* in the Balance-scale dataset experiment

Table III. : The results of the experiment with the Balance-scale dataset

Algorithm	*Gbest*	Recognition rate
Standard PSO	0.2689	81.67%
PSOd_CV	0.1298	85.00%
wPSOd_CV	0.0077	90.00%

Fig. 9 and Table III demonstrate the accuracy results of the experiment with 50 particles, 200 iterations. The wPSOd_CV algorithm also obtained the best recognition rate among the three algorithms.

3.4 Credit approval dataset

To investigate a different situation, the one hidden layer NN was also tested. Thus, the NN can be used with a bigger dataset that has bigger attributes. The Credit approval dataset was chosen. This dataset that is concerning the applications for credit cards has fourteen attributes. To protect the confidentiality of the data, the names of the attributes are not disclosed [22]. The configuration of the NN was 14-24-2.

The first experiment investigated the situation when the numbers of samples from each class were equal. For example, 175 samples from class 1 and 175 samples from class 2 were selected randomly as 350 training samples. The testing samples 125 data from set 1 and 125 data from set 2.

To investigate different cases, the number of iterations and particles were modified from 10 particles, 50 iterations in scenario 1 to 50 particles, 100 iterations in scenario 2.

The reductions of the minimum learning error $Gbest$ in the first scenario and second scenario are shown in Fig. 10, and Fig. 11, respectively. The results which are given in table IV demonstrated the performance of the proposed wPSOd_CV algorithm for training the FPGA-based NN in the co-design architecture.

Table IV. : Credit approval dataset, 350 training samples, 250 testing samples

Parameters	Algorithm	$Gbest$	Recognition rate
25 particles 50 iterations	Standard PSO	0.1267	91.20%
	PSOd_CV	0.1161	94.00%
	wPSOd_CV	0.1155	94.40%
50 particles 100 iterations	Standard PSO	0.1164	92.00%
	PSOd_CV	0.1089	94.80%
	wPSOd_CV	0.0971	96.40%

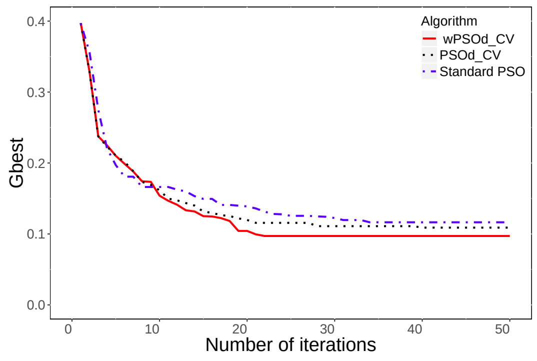


Fig. 10: The reduction of $Gbest$ with 350 training samples, 25 particles, 50 iterations

The second experiment used the training set and testing set that did not have an equal number of samples from each class. For example, 272 samples of set 1 and 229 samples of set 2 were randomly included in the training set. The testing set had 190 samples. The number of particles was 50, and the number of iterations was 100. The reduction of $Gbest$ is presented in Fig. 12. The final $Gbest$ of the wPSOd_CV algorithm decreased to the lowest value (0.1324) while the final $Gbest$ of the PSOd_CV algorithm was 0.1332, and the final $Gbest$ of the standard PSO algorithm was 0.1421. The recognition rates of the NN trained by these three algorithms are illustrated in table V. The wPSOd_CV algorithm also obtained the highest recognition rate (88.95%).

The experimental results show that the proposed wPSOd_CV algorithm obtained the best performance among the three algorithms (wPSOd_CV, PSOd_CV, and standard PSO).

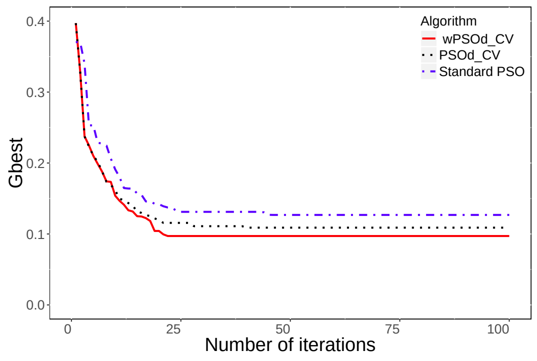


Fig. 11: The reduction of $Gbest$ with 350 training samples, 50 particles, 100 iterations

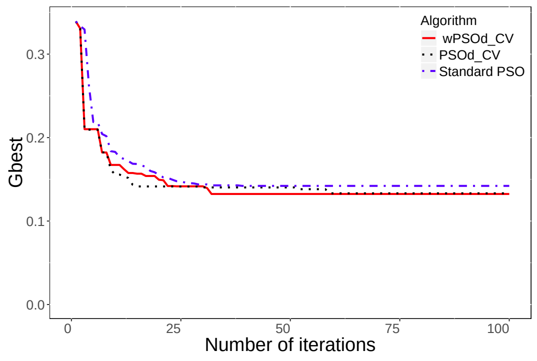


Fig. 12: The reduction of $Gbest$ with 500 training samples, 50 particles, 100 iterations

Table V. : Credit approval dataset, 500 training samples, 190 testing samples

Algorithm	$Gbest$	Recognition rate
Standard PSO	0.1421	83.65%
PSOd_CV	0.1332	86.84%
wPSOd_CV	0.1324	88.95%

4. CONCLUSION

This paper presents the new PSO algorithm called the wPSOd_CV algorithm which is the improved version of the PSOd_CV algorithm presented in the previous paper. In light of the evidence, experimental results confirmed that the neural network trained by PSO algorithms (SPSO, PSOd_CV, wPSOd_CV) was successfully developed. The results also stated that the proposed wPSOd_CV algorithm obtained higher accuracy concerning the learning errors $Gbest$ and the recognition rates than two other PSO algorithms with different datasets and different PSO parameters (number of iterations, number of particles). The proposed PSO algorithm could help the training phase of the NN out of the local minimum to increase the training efficiency.

A possible avenue for future research is to investigate more complex dataset with a bigger size of the NN. Another future scope of this study is to improve the wPSOd_CV to increase the recognition rate and to reduce the learning error. The future research will also investigate the using of the proposed NN-wPSOd_CV in real-life applications.

5. REFERENCES

[1] P.M. Ravdin, G. M. Clark GM, A practical application of neural network analysis for predicting outcome of individual breast cancer patients, *Breast Cancer Research and Treatment*, vol. 22, no. 3, pp. 285-293, 1992

[2] A. E. Celik, Y. Karatepe, Evaluating and forecasting banking crises through neural network models: An application for Turkish banking sector, *Expert Systems with Applications* vol. 33, no. 4, pp. 809-815, 2007

[3] S. Haykin, *Neural networks and learning machines*, 3rd edn, Prentice Hall, 2008

[4] R. H. Nielsen, Theory of the backpropagation neural network, *In processing of the international conference on neural networks*, pp. 693-605, 1989

[5] R. Rojas, *Neural networks - a systematic introduction*, Springer-Verlag, 1996

[6] J. R. Zhang, J. Zhang, T. M. Lok. M. R. Lyu, A hybrid particle swarm optimizationback-propagation algorithm for feedforward neural network training, *Applied mathematics and computation*, vol. 185, pp. 10261037, 2007

[7] Z.A. Bashir, M.E. El-Hawary, Applying Wavelets to Short-Term Load Forecasting Using PSO-Based Neural Networks, *IEEE transactions on power systems*, vol. 46, pp. 268-275, 2016

[8] A. Suresh, K. V. Harish, N. Radhika, Particle Swarm Optimization over Back Propagation Neural Network for Length of Stay Prediction, *In processing of the international conference on information and communication technologies*, vol. 24, no.1, pp. 20-27, 2009

[9] V. G. Gudise, G. K. Venayagamoorthy, Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks, *In processing of 2003 IEEE swarm intelligence symposium*, pp. 110-117, 2003

[10] J. Kennedy, R. Eberhart, Particle swarm optimization, *In processing of the IEEE international conference on neural networks*, vol. 4, pp.1942-1948, 1995

[11] R. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, *In processing of the 2001 IEEE international conference on congress on evolutionary computation*, vol. 1, pp. 81-86, 2001

[12] R. Mendes, et al., Particle swarms for feedforward neural network training, *In processing of the IEEE international joint conference on neural networks*, vol.2, pp.1895-1899, 2002

[13] K. W. Chau, Application of a PSO-based neural network in analysis of outcomes of construction claims. *Automation in construction*, vol. 16, no. 5, 642-646, 2007

[14] G. Montavon, G. B. Orr, K. R. Muller *Neural networks: tricks of the trade*, 2nd edn, Springer, 2012

[15] T. L. Dang, Y. Hoshino, Hardware/Software Co-design for a Neural Network Trained by Particle Swarm Optimization Algorithm, *Neural Processing Letters*, pp. 1-25, 2018

[16] T. L. Dang, C. Thang, Y. Hoshino, Hybrid hardware-software architecture for neural networks trained by improved pso algorithm, *ICIC Expree Letters*, pp. 565-574, 2017

[17] T. L. Dang, Y. Hoshino, An-FPGA based classification system by using a neural network and an improved particle swarm optimization algorithm, *In processing of the 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems*, pp.97-102, 2016

[18] Y. Hoshino, H. Takimoto, PSO training of the neural network application for a controller of the line tracing car, *In: Proceedings of the IEEE International Conference on Fuzzy Systems*, pp.1-8, 2012

[19] Y. Shi. R.Eberhart R, A modified particle swarm optimizer, *In Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, pp 69-73, 1998

[20] Altera company, SoC product brochure. https://www.altera.com/products/soc/overview.html, accessed 07 February 2019

[21] Terasic company, DE1-SoC user manual, http://de1-soc.terasic.com, accessed 07 February 2019

[22] M Lichman, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, accessed 07 February 2019