Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 257085, 13 pages
http://dx.doi.org/10.1155/2013/257085

Research Article

Improved SpikeProp for Using Particle Swarm Optimization

Falah Y. H. Ahmed, Siti Mariyam Shamsuddin, and Siti Zaiton Mohd Hashim

Soft Computing Research Group, Faculty of Computer Science & Information Systems, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

Correspondence should be addressed to Siti Mariyam Shamsuddin; mariyam@utm.my

Received 27 March 2013; Accepted 1 July 2013

Academic Editor: Praveen Agarwal

A spiking neurons network encodes information in the timing of individual spike times. A novel supervised learning rule for SpikeProp is derived to overcome the discontinuities introduced by the spiking thresholding. This algorithm is based on an error-backpropagation learning rule suited for supervised learning of spiking neurons that use exact spike time coding. The SpikeProp is able to demonstrate the spiking neurons that can perform complex nonlinear classification in fast temporal coding. This study proposes enhancements of SpikeProp learning algorithm for supervised training of spiking networks which can deal with complex patterns. The proposed methods include the SpikeProp particle swarm optimization (PSO) and angle driven dependency learning rate. These methods are presented to SpikeProp network for multilayer learning enhancement and weights optimization. Input and output patterns are encoded as spike trains of precisely timed spikes, and the network learns to transform the input trains into target output trains. With these enhancements, our proposed methods outperformed other conventional neural network architectures.

1. Introduction

For all three generations of neural networks, the output signals can be continuously altered by variation in synaptic weights (synaptic plasticity). Synaptic plasticity is the basis for learning in all ANNs. As long as there is nonvariant activation function, accurate classification based on certain vector input values can be implemented with the help of a BP learning algorithm like gradient descent [1, 2].

Spiking neural network (SNN) utilises individual spikes in time domain to communicate and to perform computation in a manner like what the real neurons actually do [3, 4]. This method of sending and receiving individual pulses is called pulse coding where information which is transmitted is carried by the pulse rate. Hence, this type of coding permits multiplexing of data [2].

For instance, analysis of visual input in humans requires less than 100 ms for facial recognition. Yet, facial recognition was performed by [5] by using SNN with a minimum of 10 synaptic steps on the retina at the temporal lobe, allowing nearly 10 ms for the neurons to process. Processing time is short, but it is sufficient to permit an averaging procedure which is required by pulse coding [2, 5, 6]. In fact, pulse coding technique is preferred when speed of computation is the issue [5].

2. Page Background and Related Works

2.1. Spiking Neural Networks (SNNs). Neural networks which perform artificial information processing are built using processing units composed of linear or nonlinear processing elements (a sigmoid function is widely used) [6–9]. SNN had remained unexplored for many years because it was considered too complex and too difficult to analyze. Apart from that, biological cortical neurons have long time constants. Inhibition speed can be of the order of several milliseconds, while excitation speed can reach several hundreds of milliseconds. These dynamics can considerably constrain applications that need fine temporal processing [10, 11].

Little is known about how information is encoded in time for SNNs. Although it is known that neurons receive and emit spikes, whether neurons encode information using spike rate or precise spike time is still unclear [12]. For those

supporting the theory of spike rate coding, it is reasonable to approximate the average number of spikes in a neuron with continuous values and consequently process them with traditional processing units (sigmoid, for instance). Therefore, it is not necessary to perform simulations with spikes, as the computation with continuous values is simpler to implement and evaluate [13].

An important landmark study by Maass [14] has shown that SNN can be used as universal approximations of continuous functions. Maass proposed a three-layer SNN (consisting of the input layer, the generalization layer, and the selection layer) to perform unsupervised pattern analysis. Reference [15] applied spiking neural networks to several benchmark datasets (which include internet traffic data, EEG data, XOR problems, 3-bit parity problems, and iris dataset) and performed function approximation and supervised pattern recognition [16].

One of the ongoing issues in SNN research is how the networks can be trained. Much research has been done on biologically inspired local learning rules [13, 17], but these rules can only carry out supervised learning for which the networks cannot be trained to perform a given task. Classical neural network research became famous because of the error-backpropagation learning rule. Due to this, a neural network can be trained to solve a problem which is specified by a representative set of examples. Spiking neural networks use a learning rule called SpikeProp which operates on networks of spiking neurons and uses the exact spike time temporal coding [18]. This means that the exact spike time of input and output spikes encodes the input and output values.

2.2. Learning in Networks of Spiking Neurons (SpikeProp). Learning in the perceptron networks is usually performed by a gradient descent method [19] by using the backpropagation algorithm [20], which explicitly evaluates the gradient of an error function. The same approach has been employed in the *SpikeProp* gradient learning algorithm [18] which learns the desired firing times of the output neurons by adapting the weight parameters in the Spike Response Model SRM0 [21]. Several experiments have been carried out on *SpikeProp* to clarify several burning issues, for example, the role of the parameter initialization and negative weights [22]. The performance of the original algorithm can be improved by adding the momentum term [23]. *SpikeProp* can be further enhanced with additional learning rules for synaptic delays, thresholds, and time constants [24], which will normally result in faster convergence and smaller network sizes for the given learning tasks. An essential speedup was achieved by approximating the firing time function using the logistic sigmoid [25]. Implementation of *SpikeProp* algorithm on recurrent network architectures has shown promising results [26].

SpikeProp does not usually allow more than one spike per neuron, which makes it suitable only for "time-to-first-spike" coding scheme [5]. Its adaptation mechanism fails for the weights of neurons that do not emit spikes. These difficulties are due to the fact that spike creation or its removal due to weight updates is very discontinuous. *ASNA-Prop* has been proposed [24] to solve this problem by emulating the feed forward networks of spiking neurons with the discrete-time analog sigmoid networks with local feedback, which is then used for deriving the gradient learning rule. It is possible to estimate the gradient by measuring the fluctuations in the error function in response to the dynamic neuron parameter perturbation [27].

SpikeProp adopts error backpropagation procedures which have been used widely in the training of analog neural networks to perform supervised learning [28]. SpikeProp does have weaknesses. The first weakness concerns sensitivity to parameter initialization values, which means that if the neuron is still inactive after initialization, the SpikeProp will not perform training for these weights which will not produce any spike. The second weakness is that SpikeProp is only suitable in cases where there is latency-based coding. The third weakness is that SpikeProp works only for SNNs where neurons spike only once in the simulation time. Finally, SpikeProp algorithm has been designed for training the weights only. To address these weaknesses, several improvements to SpikeProp algorithms have been suggested [29].

3. Methodology

We introduce new learning rules and enhancement architecture for improved SpikeProp.

3.1. Enhancement of SpikeProp Architecture by PSO (PSO-SpikeProp) (Model 1). In this proposed method, the SpikeProp was accelerated using four basic parameters of PSO; these parameters are the acceleration constant for g_{best}, the acceleration constants for p_{best}, the time interval (Δt), depending on the time of SpikeProp, and the number of particles used in SpikeProp.

The acceleration constants are used in the simulation to specify swarm behavior of particles. p_{best} and g_{best} positions are formed by the constants c_1 and c_2. The global best solution over the particle depends on the pulse time of SpikeProp (this is given by the constant c_1). The individual personal best solution over the particle depends on the pulse time of the SpikeProp (this is given by the constant c_2). If c_1 is more than c_2, the swarm moves around the global best solution. On the other hand, when c_2 is greater than c_1, the swarm moves around the individual personal best.

The time Δt of the parameters in SpikeProp defines the time in each pulse interval over each movement that occurs in the solution space in the pools processes. Reducing these parameters in SpikeProp leads to higher granularity movement within the solution space and a higher d time value of the highest pulse in SpikeProp on the node. The higher numbers of particles in the swarm or simulation in SpikeProp form the greater amount of the space that is covered in the problem; hence, the optimization will be fewer.

Depending on the node that has the higher pulse time in SpikeProp solution space, the parameters can be adapted to achieve better optimization. These basic parameters in PSO are used by SpikeProp. There are subparameters which depend on the dataset of the problem (like particle dimension, number of particles, and the stopping condition).

Table 1: PSO parameters used in SpikeProp.

Parameters	Values
c_1 (g_{best})	1.0
c_2 (p_{best})	1.0
Δt (time interval)	0.1
Number of particles	20
Problem dimension	Based on dataset SpikeProp architecture
Stop condition	SpikeProp minimum error or maximum number of iterations
Range of particles	From −1 to 1 [−1, 1]

The number of particles in SpikeProp using PSO significantly affects the execution time. There is a tradeoff between the size of the practical swarms and the execution time. PSO with a well-selected parameter set can perform well under all circumstances [30].

In this study, the parameters that have been used are summarized in Table 1, while particle position (weight and bias values) is initialized randomly with initial position velocity value set at 0.

Each particle position of the swarm is represented by a set of the weights for the current iteration. The dimension of the practical swarm determines the weight number of the network. In order to minimize the learning error, the particle should move within the weight space. Updating the weight of the network means changing the position in order to reduce the number of iterations. For each iteration, a new velocity calculation takes place to determine the new particle position movement. A set of new weights is used to obtain the new error, thus a new position. For PSO, the new weights are registered even if there is no noticeable improvement. This process applies for all particles. The global best particle position is the one with the least number of errors. The training process stops when the target minimum error is reached or the numbers of computational processes exceed the number of iterations allowable. When the training is complete, the weights are used to compute the classification error for the training patterns. The same patterns are used to test the network by using the same set of weights.

No researcher has yet used SpikeProp on PSO. p_{best} value and g_{best} value are applied to solve problems associated with the learning error. The SpikeProp weight and SpikeProp bias are achieved by adding the calculated velocity value as shown in (1) and (2). A new set of positions is used to produce the new learning error. In this proposed method, the classification dataset output has been written in a minimum number of iterations with the lowest error. The summary on PSO-SpikeProp learning process is shown in Figure 1.

$$w_{ij}\text{new} = w_{ij}\text{old} + \Delta w_{ij} * \Delta t\Big|_{t=t_j^d}^{(t)}, \tag{1}$$

$$\Delta w_{ij}(n) = c_1 * \left(p_{\text{best},n} - w_{ij}(n)\right) + c_2 * \left(g_{\text{best},n} - w_{ij}(n)\right)\Big|_{t=t_j^d}^{(t)}. \tag{2}$$

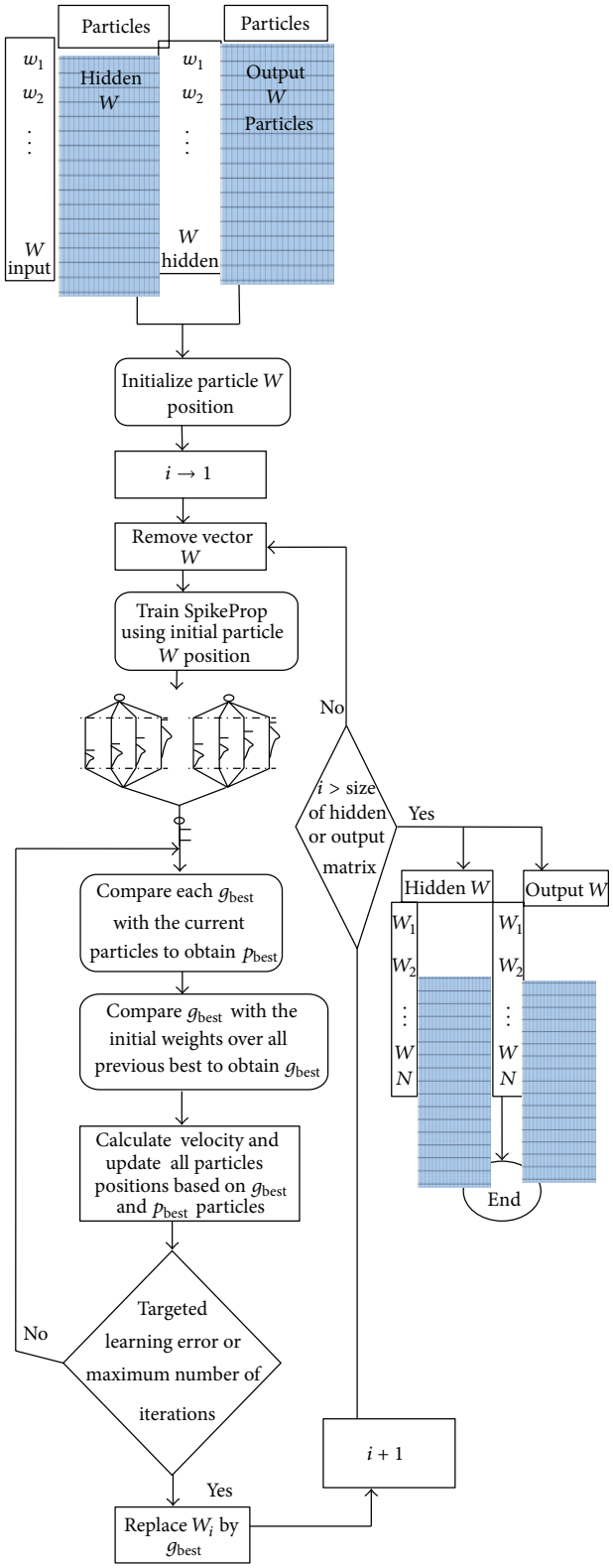


Figure 1: PSO-SpikeProp learning process.

In this proposed technique, the PSO is applied to SpikeProp algorithm (refer to Figure 1). Equation (1), the new weight (w_{ij}new), is performed for each element of the positions

of p_{best} and g_{best}. In (2), $\Delta w_{ij}(n)$ subtracts the dimensional weight of the element from the dimension from the best vector and then multiplies this by a random number (between 0.0 and 1.0) and an acceleration constant (c_1 and c_2). A number of particles have been used to solve 8 different dataset problems. The objectives of the study are to reduce errors, enhance the learning rate of SpikeProp, and to speed up the algorithm process. Figure 1 shows that particle swarms with initial random rates have different mean squared error (MSE). During the learning process, all particles move together to get p_{best} and g_{best}. g_{best} fit is an optimum solution.

3.2. Proposed μ Angle Driven Dependency Learning Rate (Model 2). The proposed μ angle driven dependency learning rate is an extension of Chan's [31] adapted learning rate and momentum during the training used in BP. This proposed method enhances SpikeProp learning according to the angle calculation between $\Delta E(n)$ and $\Delta w(n-1)$. The adaptation adjusts the angle at 90 degrees as per the Pythagoras formula method to get the square of the hypotenuse that equals to the sum of the squares of the other two sides. If the angle is less than 90 degrees, the learning rate is increased inversely, but if the angle is larger than 90 degrees, the learning rate is decreased. These mathematical methods have been applied to enhance SpikeProp, as shown next.

(1) The angle θ between the change of errors and the change of the weights can be calculated by

$$\cos\theta(n) = \left.\frac{\Delta W(n-1)}{\sqrt{\Delta E(n)^2 + \Delta W(n-1)^2}}\right|_{t=t_j^d}^{(t)}. \tag{3}$$

(2) The adaption learning rate cab be calculated by using the flowing formula:

$$\mu(n) = \mu(n-1) * (1 + 0.5 * \cos\theta(n)). \tag{4}$$

(3) Adaption of the momentum can be acquired as follows:

$$\alpha(n) = \alpha(0) * \left.\frac{\|\Delta E(n)\|}{\|\Delta W(n-1)\|}\right|_{t=t_j^d}^{(t)}. \tag{5}$$

(4) As mentioned previously, the weights can be changed as follows:

$$\Delta W_{ij}(n) = \mu(n) * \left(\frac{\partial E}{\partial W_{ij}}\right) + \alpha(n) * \Delta W_{ij}(n-1)\Big|_{t=t_j^d}^{t}. \tag{6}$$

Fortunately, the learning rate is adapted much faster when we are using the modified adaptation rule:

$$\mu(n) = \mu(n-1) * (1 + 0.1 * \cos\theta(n)). \tag{7}$$

Moreover, a backtracking strategy has been used, which reruns learning steps taking more than half learning rate time if total error increases. From this it can be concluded that the learning rate gets improved in Spikeprop with a higher rate than the standard SpikeProp.

3.3. Merging Model 1 with Model 2 for Enhancing SpikeProp (Model 3). In order to get better performance and enhance the operation of Spikeprop, a merging process between Model 1 and Model 2 (resulting in Model 3) has been carried out. Model 3 has an architecture which is partly PSO and partly angle driven dependency learning rate system. The flowchart in Figure 2 shows the general working of Model 3.

3.4. Error Measurement Functions. The target of the SpikeProp algorithm is to learn a set of target firing times, denoted by $\{t_j^d\}$, at the output neurons $j \in J$ for a given set of input patterns $\{P[t_j \cdots t_h]\}$, where $P[t_j \cdots t_h]$ defines a single input pattern described by single spike times for each neuron $h \in H$. Given the desired spike times $\{t_j^a\}$ and actual firing times $\{t_j^a\}$, this error function is defined by

$$\text{(a)}\quad \text{MSE} = \frac{1}{n}\sum_{2-1}^{n}\left(t_{j_2}^a - t_{j_2}^d\right)^2. \tag{8}$$

This thesis uses other error functions such as RMSE (9), MAPE (10) and MAD (11) to get more validation to evaluate the accuracy of SpikeProp (SNN) that has been enhanced and proposed (Models 1, 2, and 3).

Consider

$$\text{(b)}\quad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{2-1}^{n}\left(t_{j_2}^a - t_{j_2}^d\right)^2}, \tag{9}$$

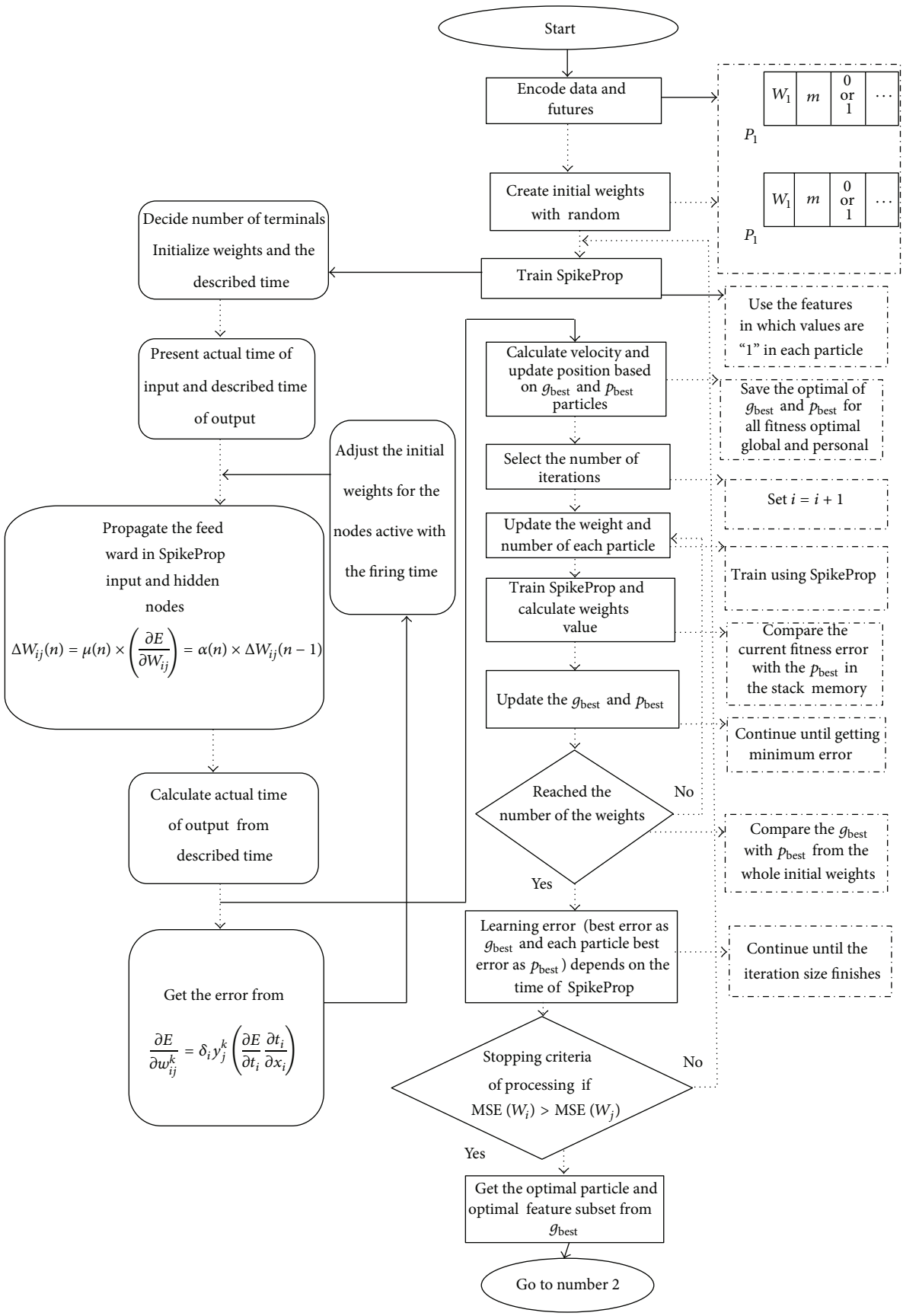
$$\text{(c)}\quad \text{MAPE} = \sum_{2-1}^{n}\left|\frac{t_{j_2}^a - t_{j_2}^d}{t_{j_2}^d}\right| * \frac{100}{n}, \tag{10}$$

$$\text{(d)}\quad \text{MAD} = \sum_{2-1}^{n}\frac{\left|t_{j_2}^a - t_{j_2}^d\right|}{n}. \tag{11}$$

4. Results and Discussion

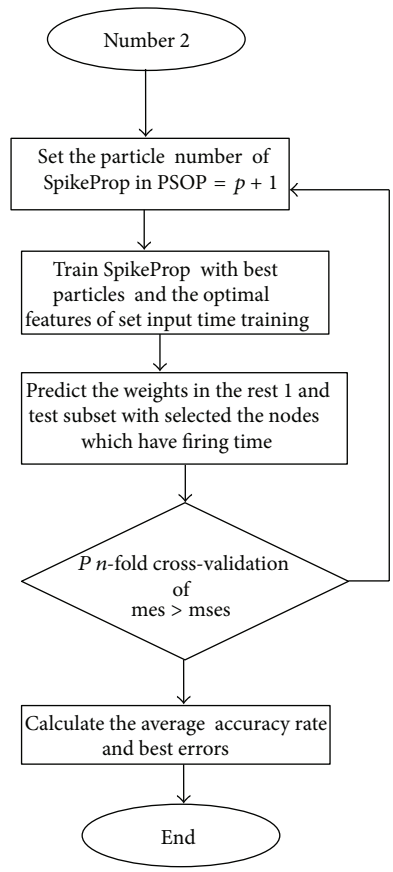
This section presents the results of study on learning of SpikeProp network based on the proposed method of improved SpikeProp. The results for all datasets involved are analyzed based on the convergence to MSE, RMSE, MAPE, and MAD with their classification performance. The results of the proposed methods for each dataset are analyzed based on performance (accuracy). For analysis purposes, methods of improved SpikeProp are used to train and optimize the networks, comparing different measurements of error. The results of SpikeProp based on the proposed method of improved SpikeProp are presented in the following subsections.

4.1. Results and Analysis of Standard SpikeProp. This section presents the result of standard SpikeProp for all datasets. The results are analyzed based on the convergence to MSE, RMSE, MAPE and MAD findings with their classification performance as shown in Table 2. All experiments for standard SpikeProp are based on ten runs. From Table 2, RMSE is better than MSE for the training datasets of BTX, Wine,



(a)

Figure 2: Continued.



(b)

Figure 2: Merge Model 1 with Model 2 learning process.

Heart, Iris, Diabetes, Breast Cancer, Liver, Hepatitis, and XOR, respectively. The RMSE is also better than MAPE, and MAD error measurements for all datasets. These show that the SpikeProp algorithm also demonstrates in a direct way that networks of spiking neurons can carry out complex, nonlinear tasks in a temporal code. As the experiments indicate, the SpikeProp algorithm is able to perform correct classification on nonlinearly separable datasets with all types of errors measurement compared to traditional sigmoidal networks (BP) as shown in Table 3. Table 2 illustrates a comparison of the measurements between the training and testing of SpikeProp algorithm. From this table, we can see that the SpikeProp converges with less errors than standard BP.

4.2. Results and Analysis of Standard BP. The analyses of standard BP for all datasets are validated based on MSE, RMSE, MAPE, and MAD as shown in Table 3. From the training part in Table 3, the RMSE is better than MSE for the training datasets of Diabetes, Heart, Breast Cancer, Liver, Hepatitis, Wine, Iris, BTX, and XOR, respectively. The MSE is also better than MAPE and MAD for all datasets except BTX, which shows that MAD values are less compared to MSE. This traditional BP network is widely used by other researchers for classification problems. Bohte et al. [18] designed SpikeProp for BP learning strategy; the comparisons between BP and SpikeProp are given in Tables 2 and 3. It shows that BP generates higher errors compared to SpikeProp.

4.3. Analysis of the Proposed Model 1: PSO-Spikeprop. Table 4 reveals the MSE generalization from the training part. MSE gives better results on Diabetes, Heart, Wine, and Breast Cancer datasets and least competitive for Liver, Hepatitis, Iris, BTX, and XOR datasets. For RMSE, the proposed PSO-SpikeProp is better for Wine, BTX, Diabetes, Iris, and Heart and less competitive in Liver, Breast Cancer, Hepatitis, and XOR, respectively. For MAD, the finding is better in Breast Cancer, Diabetes, and Heart and worse in Wine, Liver, Iris Hepatitis, BTX, and XOR, respectively. On the other hand, MAPE for PSO-SpikeProp gives higher error but still better than standard SpikeProp algorithm.

The PSO-SpikeProp gives the smallest error in MAPE compared to standard Spikeprop. However, PSO-SpikeProp stands out to be better if RMSE is being compared. Since the errors are squared before they are averaged, the RMSE gives a relatively low error rates, although the MSE has close values to RMSE as shown in Table 4. Therefore, Model 1: PSO-SpikeProp has its own good characteristics in generating

Table 2: Analysis for standard SpikeProp algorithm.

	Training SpikeProp				Testing SpikeProp			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.549	0.496	43.305	0.992	0.571	0.511	44.419	1.023
BTX	1.713	0.225	36.294	1.575	2.190	0.269	44.451	1.888
Diabetes	0.426	0.386	31.556	0.773	0.492	0.420	34.348	0.840
Heart	0.442	0.374	32.058	0.749	0.367	0.335	29.820	0.671
Hepatitis	0.690	0.551	47.732	1.102	0.702	0.558	48.509	1.117
Iris	0.814	0.384	40.992	1.153	0.122	0.345	19.124	1.036
Liver	0.684	0.525	41.984	1.051	0.721	0.542	43.184	1.085
Wine	0.525	0.293	34.152	0.879	0.514	0.287	33.553	0.863

Table 3: Analysis for standard BP algorithm.

	Training BP				Testing BP			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.672	0.525	54.759	1.278	0.816	0.638	54.443	1.277
BTX	1.982	0.796	88.018	2.575	2.707	0.798	88.344	2.588
Diabetes	0.510	0.429	37.576	1.271	0.792	0.627	59.208	1.255
Heart	0.608	0.477	41.902	1.270	0.805	0.633	55.724	1.267
Hepatitis	0.837	0.646	55.588	1.293	0.837	0.647	55.607	1.294
Iris	0.874	0.763	69.176	2.290	0.865	0.759	68.660	2.278
Liver	0.826	0.641	56.315	1.283	0.831	0.643	56.459	1.287
Wine	0.749	0.419	72.471	1.119	0.753	0.408	72.649	1.125

Table 4: Analysis for Model 1.

	Training Model 1				Testing Model 1			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.356	0.361	35.261	0.723	0.367	0.372	36.268	0.745
BTX	1.270	0.199	35.596	1.398	1.763	0.247	43.785	1.730
Diabetes	0.241	0.245	22.650	0.490	0.284	0.277	25.778	0.555
Heart	0.2828	0.281	26.813	0.563	0.272	0.272	25.997	0.545
Hepatitis	0.483	0.448	42.640	0.896	0.494	0.457	43.609	0.915
Iris	0.351	0.251	33.270	0.754	0.112	0.147	18.38	0.443
Liver	0.360	0.340	31.854	0.680	0.383	0.357	33.332	0.714
Wine	0.312	0.196	25.033	0.590	0.305	0.196	25.269	0.589

the smallest error during the implementation. For this model we conclude that error can be reduced to reach the minimum value.

4.4. Analysis of the Proposed Model 2: SpikeProp with Angle Driven Dependency (μ). Similar to previous experiments, the results of Model 2 of SpikeProp with angle driven dependency are computed based on MSE, RMSE, MAPE, and MAD error measurements. Table 5 shows the findings of the model, which are based on ten independent runs on both training and testing datasets, respectively. The average testing errors are being calculated along with the standard deviations for all datasets.

As can be seen from Table 5, it is interesting to see the small standard deviations for all error rates on the training set and the testing set in all datasets. The results of this proposed Model 2 have demonstrated that the generalization of MSE is better for Diabetes, Heart, and Wine datasets, while the results for Liver, BTX, Hepatitis, Iris, Breast Cancer, and XOR datasets are the least competitive. The results also have shown that the RMSE is better on all datasets and less competitive for other error measurements except for the Diabetes dataset which has better MSE values compared to RMSE. In general, the diversity of errors rates (MSE, RMSE, MAPE, and MAD) for this proposed Model 2 is considered better for all datasets. According to this model, we can find that the error is less than BP and SpikeProp standard.

4.5. Analysis of the Proposed Model 3: Hybridization of Model 1 and Model 2. Just as we get good strain by cross-breeding

Table 5: Analysis for Model 2.

	Training Model 2				Testing Model 2			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.472	0.451	40.922	0.902	0.489	0.465	37.977	0.930
BTX	1.256	0.191	32.779	1.338	1.899	0.253	44.841	1.774
Diabetes	0.248	0.251	23.103	0.503	0.290	0.283	25.455	0.566
Heart	0.360	0.326	29.344	0.653	0.327	0.307	26.355	0.614
Hepatitis	0.523	0.500	42.233	1.001	0.452	0.435	43.929	0.872
Iris	0.438	0.278	34.668	0.835	0.117	0.377	18.335	1.131
Liver	0.419	0.379	34.285	0.759	0.444	0.397	34.289	0.795
Wine	0.365	0.229	28.329	0.688	0.358	0.229	28.679	0.689

Table 6: Analysis for Model 3.

	Training Model 3				Testing Model 3			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.350	0.355	34.812	0.711	0.361	0.367	35.863	0.734
BTX	0.972	0.162	27.401	1.138	1.592	0.226	39.155	1.585
Diabetes	0.211	0.212	20.218	0.425	0.251	0.245	23.496	0.491
Heart	0.252	0.250	24.543	0.501	0.241	0.240	23.634	0.481
Hepatitis	0.437	0.421	41.179	0.843	0.446	0.431	42.194	0.863
Iris	0.330	0.244	32.747	0.732	0.105	0.153	17.576	0.461
Liver	0.359	0.339	31.778	0.678	0.380	0.356	33.296	0.712
Wine	0.263	0.168	22.186	0.506	0.273	0.178	23.637	0.535

two good genes, it may be possible to get good SpikeProp algorithm by merging (hybridizing) two good techniques. Therefore, this paper is concerned about the merging implementation of Model 1 and Model 2 (to get Model 3) as maintained in Figure 2.

In this section, we hybridize Model 1: PSO-SpikeProp and Model 2: SpikeProp with angle driven dependency to obtain a better performance for error rates (MSE, RMSE, MAPE, and MAD). We notice that the performance measurement is better than in the previous proposed method when the hybridization takes place as shown in Table 6. The experiments are based on 10 independent runs for the training and testing for all datasets, respectively. The results have revealed that generalization of error rates in RMSE is better for BTX, Wine, Diabetes, Iris, and Heart datasets and the least competitive for Breast Cancer, Hepatitis, Liver, and XOR datasets. Similarly, the result of MSE error rate has been demonstrated to be less better for Diabetes, Heart, Wine, Iris, Breast Cancer, Liver, Hepatitis, and BTX datasets, respectively. To get better performance, we hybridized Model 1 and Model 2, to get Model 3, to minimize the error to reach the optimum error value.

4.6. Analysis and Result for Proposed Methods Based on Error. In this section, the spiking neural network and SpikeProp use the encoding by depending on the timing of spike, where the first spike has a higher weight than the last one. Since a biological neuron uses few milliseconds to process information data, only few spikes are required and emitted, however the few first spikes with highest value information can contribute to all process learning, as we used Gaussian function for encoding. Figures 3–10 show the convergence of the error and the number of iterations of the Spikeprop standard and the proposed methods for the improved SpikeProp in the classification Liver dataset, Breast Cancer, BTX, Diabetes, Heart, Hepatitis, Iris, and Wine data problems. The SpikeProp standard configuration had a much slower rate of convergence, as can be clearly seen in the plot. Although its rate of progress gradually slow down from the beginning till last iteration but kept in high error in the MSE, despite this slowdown in all data problems as shown in Figures 3–10.

4.6.1. Analysis Error and Iterations of Model 1 (PSO-SpikeProp). We also see from the convergence that the proposed method for improved SpikeProp is much better than standard SpikeProp as PSO-SpikeProp (Model 1) had dramatic slowdown from the first 10 iterations down to iteration number 40, and afterwards the plot got a slight slowdown in Liver as shown in Figure 3. In Breast Cancer data problem in Figure 4 the curve steps down at the start in error near to 0.5 until 0.38 exactly and then it continued descending to the last iteration at error 0.35 gradually. PSO-SpikeProp (model 1) is the first model from the proposed methods; the error turned down quickly in the first 10 iterations at 1.78 and continued to drop down till the last iteration in the error 1.27 in the BTX data problem as seen in Figure 5. Also in Diabetes data problem in Figure 6 the curve for the error steps down so fast in the first 10 iterations from 0.7 till 0.39 error and then continues to drop in a steady way until iteration number 86 of error 0.24. From the iteration number 87 till the last iteration

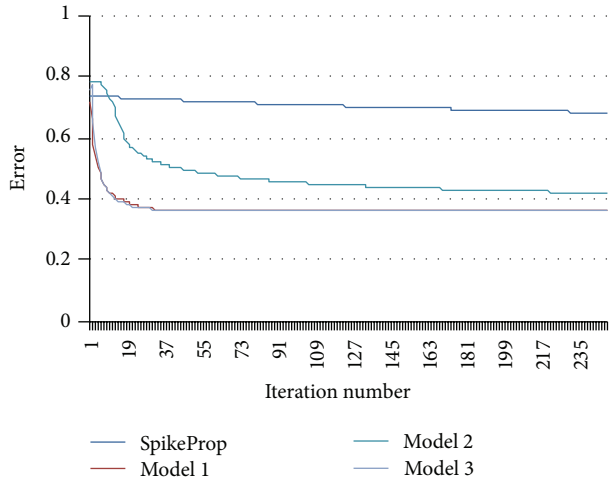


Figure 3: Liver.

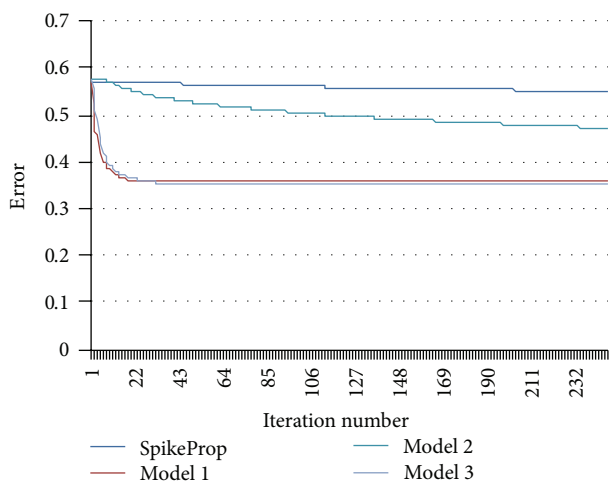


Figure 4: Breast cancer.

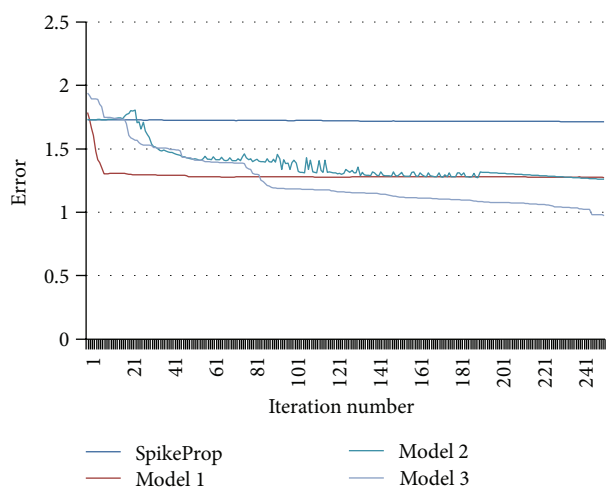


Figure 5: BTX.

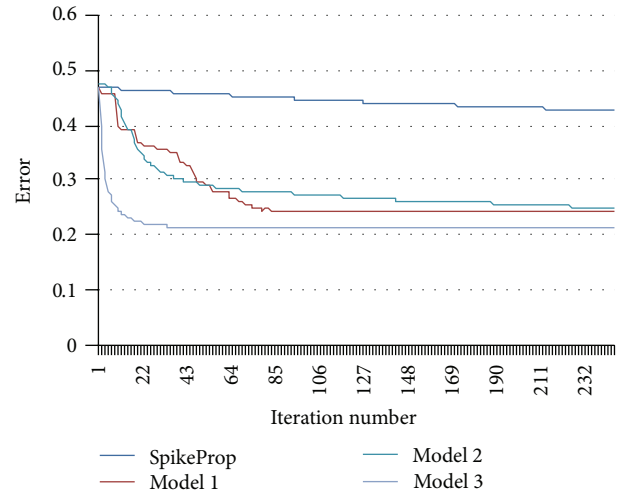


Figure 6: Diabetes.

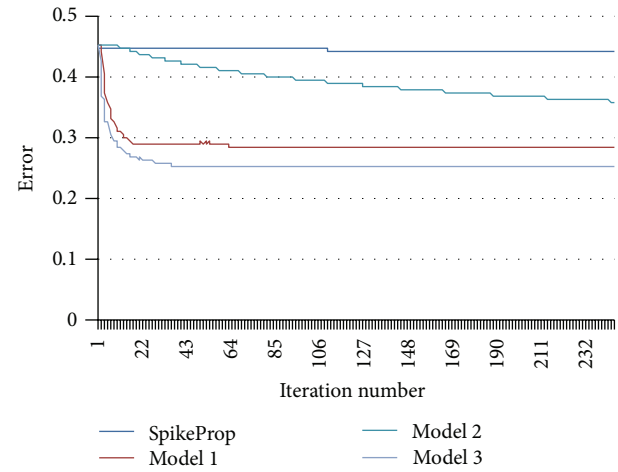


Figure 7: Heart.

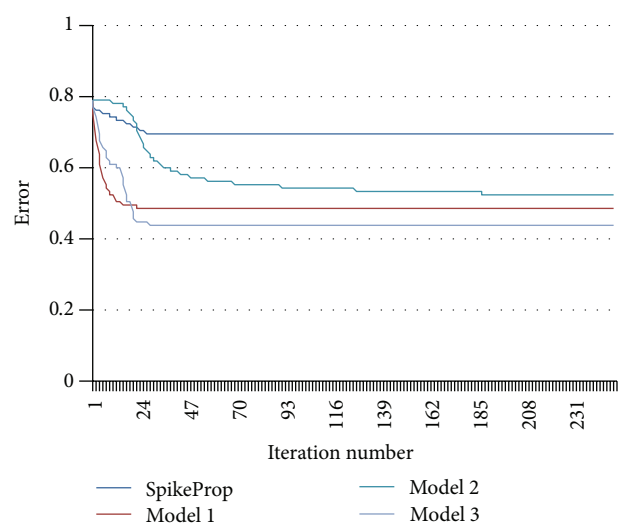


Figure 8: Hepatitis.

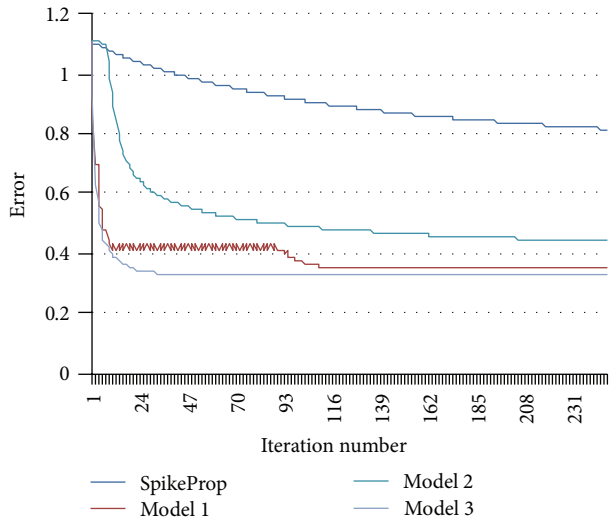


Figure 9: Iris.

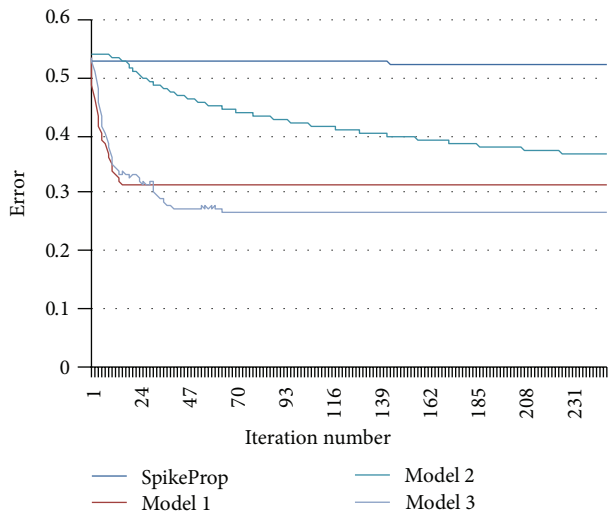


Figure 10: Wine.

it was a little sloped and stopped at the last iteration in error 0.24.

As we can see from Figure 7 in Heart problem the curve in error has dropped in a dramatic manner in the first 15 iterations in error from 0.45 to 0.28, and then the dropping is slowed down till iteration number 67 in error 0.28 and continued steadily till the last iteration for the same error. From the same proposed model in Hepatitis data problem in Figure 8 we can see that the slope of the curve starts at the first 15 iterations in a fast way in error 0.76 to 0.49, and then it slows down till iteration number 38 in error 0.48 and it stays likely stable till the last iteration. In Iris data problem in this model the plot from Figure 9 is stepped down in very fast drop from the first 10 iterations in error 1.105, and then it is zigzagged in error range 0.433 to 0.410 till iteration number 90 and then immediately fell down till the iteration number 112 in error 0.351 and it stayed on it till last iteration. Finally the plot of the error is stepped down in a high manner in the first 15 iterations of error 0.534 to 0.312, and then it stabilized in this error till the last iteration in Wine data problem as shown in Figure 10. From this model we have seen from Figures 3–10 how to obtain the list error from list iteration number from the whole data sets.

4.6.2. Analysis Error and Iterations of Model 2. In the second improvement for SpikeProp through the learning rate angle driven dependency (Model 2), we can also notice in Liver data problem as shown in Figure 3 that it has a slight dropping for the first 10 iterations in error around 0.75 and got a huge fall down till iteration number 50 for the error of nearly 0.55 and then it had a slight dropping till last iteration for the error 0.4. In the same model in Breast Cancer problem in Figure 4, the plot started to fall down on the first 20 iterations in error 0.57 until 0.47 gradually. Also the curve in BTX data problem as shown in Figure 5 start dropping down from the first 5 iterations in error 1.73 to 1.72, and it is fluctuated between the 6th iteration and iteration number 197 of the error range 1.73 to 1.31. Afterwards it has a steady drop till iteration 250 at error interval 1.30 to 1.25. From the result in Diabetes problem at the same Model 2 on the plot in Figure 6, the error in first seven iterations has dropped down slowly at error 0.47; after that it stepped down quickly till iteration number 20 in error 0.35 and then continued steadily until the last iteration for error 0.24. In the Heart data problem from Figure 7, the plot is stepped down from first iteration in error 0.45 until last iteration in error 0.36, and in some iterations, it is stable and then it continues descending. Also in Hepatitis data problem as shown in Figure 8, it has a stable error of 0.786 for the first 5 iterations, and then it start to fall down till iteration number 80 for error 0.549; after that it has a slower dropping till last iteration on error 0.523 in a sequence stepping down.

From Figure 9 as we can see clearly the curve is dropped down slowly in first 10 iterations in error 1.111 till 1.040; then it accelerates in dropping till iteration number 55 of error 0.53 and the drop slows down till last iteration in 0.438 error in the Iris data problem. Finally from Figure 10 in Wine data problem the plot shows Model 2 (learning rate angle driven dependency) is almost steady in first 15 iterations in error 0.538 to 0.526, and then the dropping got accelerated continuously till last iteration for error 0.366.

4.6.3. Analysis Error and Iterations of Model 3 (SpikeProp with Angle Driven Dependency (μ)). (PSOSpikeProp and Learning Rate Angle Driven Dependency) or (model 3), in Liver dataset as it is clear from Figure 3 the plot it provides the best enhancement result for the convergence of error and number of iteration. The slant starts from the first iteration in the error a bit less than 0.8 till iteration number 20 impressively, and then a gradual descending has been done till last iteration of error close to 0.354. The curve from Figure 5 witnessed a dramatic drop in first 10 iterations starting from error 1.95 and the drop decreases till last iteration at error 0.97. We can notice from the curves in Figure 4 that the last model is much better from the SpikeProp standard and the previous methods of Breast Cancer dataset problem. The error steps down in a steady and fast manner from

Table 7: Result of training in terms of accuracy.

	Training				
	SpikeProp	Model 1	Model 2	Model 3	BP
Breast cancer	50.36	63.82	54.88	60.9	33.68
BTX	73.75	76.69	75.7	77.01	42.19
Diabetes	22.68	50.97	49.64	52.3	14.81
Heart	25.04	43.69	34.69	42.24	13.17
Hepatitis	8.21	21.38	17.57	22.23	2.44
Iris	57.68	72.27	68.22	73.22	32.79
Liver	4.86	31.97	24.01	29.3	2.14
Wine	56.03	70.47	67.57	70.36	35.78

Table 8: Result of testing in terms of accuracy.

	Testing				
	SpikeProp	Model 1	Model 2	Model 5	BP
Breast cancer	48.82	62.7	53.46	63.28	32.21
BTX	60.52	71.15	70.42	73.57	41.77
Diabetes	25.99	44.45	43.31	50.8	18
Heart	32.81	45.4	38.56	51.81	14.64
Hepatitis	8.78	18.42	16.5	23.65	4.33
Iris	77.66	84.66	80.73	96.03	50.03
Liver	8.55	28.55	20.48	28.79	2.8
Wine	56.84	70.53	67.53	73.24	40.13

the first iteration of error 1.93 to last iteration of error 0.97. From this we can see that most impact on SpikeProp standard is from PSO-SpikeProp and the leaning rate methods, and other proposed methods have less impact on BTX data problem as shown in Figure 5.

From the plot in Figure 6, this model started to drop down in a very fast way in first 10 iterations for error 0.7 to error 0.24, and then it started to slow down till iteration number 43 in error 0.21, and then it became almost stopped in error 0.21 till last iteration; we conclude from the comparison of the previous result that the model 3 gives the best and the least error from all other methods in Diabetes dataset. We can see from the plot that the curve fell down quickly in first 10 iterations for error range 0.45 to 0.29, and then the dropping started to slow down till iteration number 30 in error 0.258, then it got almost steady till last iteration in error 0.252. From the previous it is obvious that this model is the best among all the other models for Heart data problem as seen in Figure 7. This merge (Model 3) starts to drop down quickly from the first iteration for the error 0.787 until iteration number 30 for error 0.438, and then it stays stable until last iteration, as it is obvious from Figure 8 and the results that the third model has the most impact on SpikeProp compared to the previous models for Hepatitis data problem. Finally from Figure 9 it can be seen that the plot in this merge model started to step down quickly in first 30 iterations in error 1.107 to 0.332 and then it got steady for the same error till last iteration. This can show that the third model of improving SpikeProp gives the best result compared to other previous methods in Iris data problem.

Lastly, Model 3 is merging between Model 1 and Model 2 (PSO-SpikeProp and learning rate angle driven dependency); this merging model has a high impact on enhancing SpikeProp as it is seen in the curve of Figure 10 that in Wine data problem the slope is dropped quickly starting from first iteration of error 0.533 till iteration number 70 of error 0.263 and then it becomes almost stable till last iteration.

4.7. Result and Analysis Comparison of the Proposed Methods in Terms of Accuracy. This section displays the result of SpikeProp standard besides the proposed methods for enhanced SpikeProp measured in terms of accuracy. The experiments are run 10 s, 10 dependent runs on training and testing for all datasets, respectively (refer to Tables 7 and 8 and Figures 11 and 12). As it is shown in Table 7 for training, the first proposed method PSO-SpikeProp (Model 1) is evaluated in terms of accuracy; we can see that we got the value in Breast Cancer better than SpikeProp standard and proposed methods except Model 5. Regarding the BTX dataset problem, it is also better than SpikeProp standard and other proposed methods except Model 3. The generalization of accuracy for the proposed method PSOSpikeProp is better than SpikeProp standard and learning rate angle driven dependency (Model 2) in all datasets. Learning rate angle driven dependency is our proposed method; it is better than SpikeProp standard in all datasets. Finally, Model 3 is merging model (PSO-SpikeProp and Learning Rate Angle Driven Dependency) as illustrated in Figure 11 and Table 7 that it is better in accuracy generalization from all proposed methods and SpikeProp standard in all datasets.

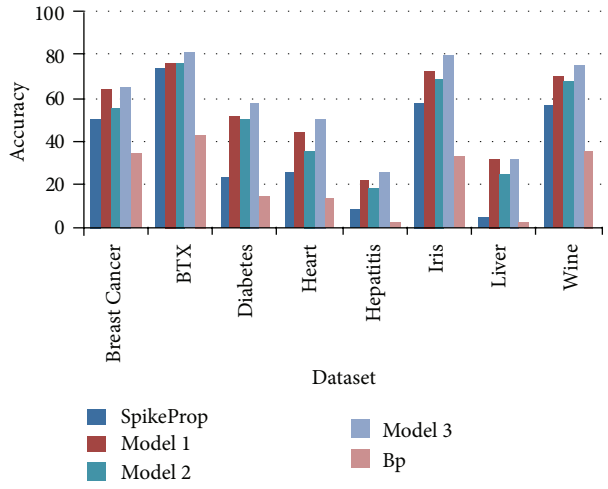


Figure 11: Results in training of the proposed methods in terms of accuracy.

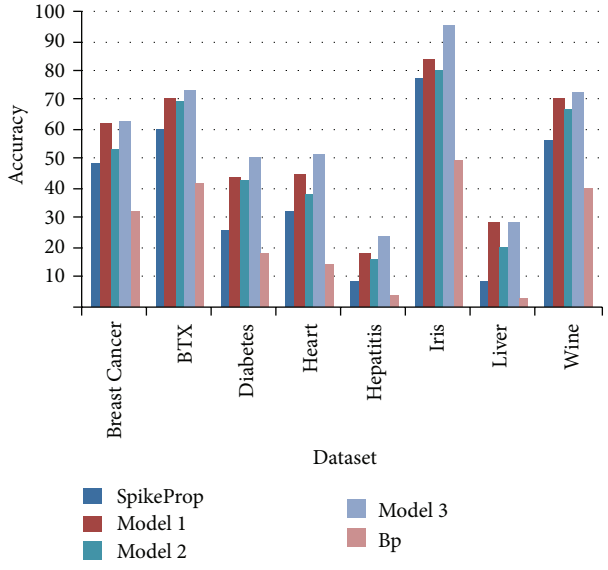


Figure 12: Results in testing of the proposed methods in terms of accuracy.

5. Conclusions

We introduced several extensions to the SpikeProp learning algorithm that make it possible to learn not only the weights, but also the delays and synaptic time constants of the connections and the thresholds of the neurons. Due to these enhancements, smaller network architecture can be used. This is mainly due to the fact that delays can now be trained and need not be enumerated. The simple 8 data sets could be solved with the same precision as the original SpikeProp algorithm, less errors (making the simulation and learning phase of the network much faster), and an increased learning convergence. There are several proposed models needed to improve the performance of SpikeProp further; hybridization of two or more good architectures is carried out (for instance the hybridization of Model 1 and Model 2 to obtain Model 3). The purpose of hybridization is to leverage the best function from each component of the hybrid. As an example, Model 3 is the hybridization of Model 1 which is PSO-SpikeProp (enhancement Spikeprop architecture by PSO) with Model 2 which is SpikeProp enhancement using angle driven dependency learning rate. For Model 3, when the position of search is far from the optimum, PSO is used to directly move the point of search close to the optimum. When the search point is close to the optimum, Model 3 switches over to the system where there is SpikeProp enhancement using angle driven dependency learning rate to reach the optimum position. Also a thorough analysis of the weight initialization problem is required. The convergence rate seems to be pretty sensitive to this. Several techniques used in classic neural networks to speed up backpropagation learning could be added to SpikeProp to further speed up learning.

References

[1] Y. J. Jin, B. X. Shen, R. F. Ren, L. Yang, J. Sui, and J. G. Zhao, "Prediction of the styrene butadiene rubber performance by emulsion polymerization using backpropagation neural network," *Journal of Engineering*, vol. 2013, Article ID 515704, 6 pages, 2013.

[2] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, *Hebbian Learning of Pulse Timing in the Barn Owl Auditory System in Maass*, 1999.

[3] A. Belatreche and R. Paul, "Dynamic cluster formation using populations of spiking neurons," in *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN '12)*, pp. 1–6, June 2012.

[4] D. Ferster and N. Spruston, "Cracking the neuronal code," *Science*, vol. 270, no. 5237, pp. 756–757, 1995.

[5] S. Thorpe, A. Delorme, and R. van Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6-7, pp. 715–725, 2001.

[6] N. Kasabov, "Evolving, probabilistic spiking neural networks and neurogenetic systems for spatio- and spectro-temporal data modelling and pattern recognition," in *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI '12)*, vol. 7311 of *Lecture Notes in Computer Science*, pp. 234–260, 2012.

[7] C. M. Bishop, *Neural Networks for Pattern Recognition*, 2000.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, 1998.

[9] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, Addison Wesley, 2002.

[10] N. Kasabov, "Integrative connectionist learning systems inspired by nature: current models, future trends and challenges," *Natural Computing*, vol. 8, no. 2, pp. 199–218, 2009.

[11] Gewaltig, *Evolution of Synchronous Spike Volleys in Cortical Networks: Network Simulations and Continuous Probabilistic Models*, vol. 1822 of *Lecture Notes in Artificial Intelligence*, Springer, 2000.

[12] S. Thorpe and J. Gaustrais, "Rank order coding," in *Computational Neuroscience: Trends in Research*, pp. 113–118, Plenum Press, New York, NY, USA, 1998.

[13] N. Kasabov, "To spike or not to spike: probabilistic spiking neuron model," *Neural Networks*, vol. 23, no. 1, pp. 16–19, 2010.

[14] W. Maass, "Computing with spiking neurons," in *Pulsed Neural Networks*, W. Maass and C. Bishop, Eds., vol. 2, pp. 55–85, MIT Press, Cambridge, Mass, USA, 2001.

[15] D. Mishra, A. Yadav, A. Dwivedi, and P. K. Kalra, "A neural network using single multiplicative spiking neuron for function approximation and classification," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '06)*, pp. 396–403, July 2006.

[16] A. Grüning and I. Sporea, "Supervised learning of logical operations in layered spiking neural networks with spike train encoding," *Neural Processing Letters*, vol. 36, no. 2, pp. 117–134, 2012.

[17] W. Gerstner and W. M. Kistler, *Spiking Neuron Models*, Cambridge University Press, 2002.

[18] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1–4, pp. 17–37, 2002.

[19] J. Šıma, "Gradient learning in networks of smoothly spiking neurons," Tech. Rep. 1045, 2009.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[21] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: An Introduction*, Cambridge University Press, 2002.

[22] S. C. Moore, *Back-propagation in spiking neural networks [M.S. thesis]*, Department of Computer Science, University of Bath, Bath, UK, 2002.

[23] J. Xin and M. J. Embrechts, "Supervised learning with spiking neural networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, pp. 1772–1777, July 2001.

[24] B. Schrauwen and J. V. Campenhout, "Parallel hardware implementation of a broad class of spiking neurons using serial arithmetic," in *Proceedings of the 14th European Symposium on Artificial Neural Networks*, M. Verleysen, Ed., pp. 623–628, 2007.

[25] K. Berkovec, *Learning in networks of spiking neurons [M.S. thesis]*, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, 2006.

[26] P. Tiňo and A. J. S. Mills, "Learning beyond finite memory in recurrent networks of spiking neurons," *Neural Computation*, vol. 18, no. 3, pp. 591–613, 2006.

[27] I. R. Fiete and H. S. Seung, "Gradient learning in spiking neural networks by dynamic perturbation of conductances," *Physical Review Letters*, vol. 97, no. 4, Article ID 048104, 2006.

[28] K. Mergenthaler and R. Engbert, "Modeling the control of fixational eye movements with neurophysiological delays," *Physical Review Letters*, vol. 98, no. 13, Article ID 138104, 2007.

[29] K. Nakazawa, M. C. Quirk, R. A. Chitwood et al., "Requirement for hippocampal CA3 NMDA receptors in associative memory recall," *Science*, vol. 297, no. 5579, pp. 211–218, 2002.

[30] Y. Shi, *Particle Swarm Optimization*, IEEE neural Network Society, 2004.

[31] L. W. Chan and F. Fallside, "An adaptive training algorithm for back propagation networks," *Computer Speech and Language*, vol. 2, no. 3-4, pp. 205–218, 1987.