Self-regulated Learning Algorithm for Distributed Coding Based Spiking Neural Classifier

Self-regulated Learning Algorithm for Distributed Coding Based Spiking Neural Classifier

Pranav Machingal
Dept. of Aerospace Engg.
Indian Institute of Science
Bengaluru, India
pranav.machingal6@gmail.com

Mohammed Thousif
Dept. of Aerospace Engg.
Indian Institute of Science
Bengaluru, India
pagalat@iisc.ac.in

Shirin Dora
Intelligent Systems Research Centre
Ulster University
Londonderry, United Kingdom
s.dora@ulster.ac.uk

Suresh Sundaram
Dept. of Aerospace Engg.
Indian Institute of Science
Bengaluru, India
vssuresh@iisc.ac.in

***Abstract*—This paper proposes a Distributed Coding Spiking Neural Network (DC-SNN) with a self-regulated learning algorithm to deal with pattern classification problems. DC-SNN employs two hidden layers. First hidden layer has receptive field neurons that convert the real-valued input features to spike patterns and the second hidden layer employs LIF neurons with inhibitory interconnections. The second hidden layer has been termed as the distributed coding layer in the rest of the paper. The inhibitory interconnections in distributed coding layer will ensure that each neuron in this layer learns a distinct spike pattern from input feature space. The synaptic weights between layers and the weights of lateral inhibitory connections are learned using a self-regulated learning algorithm. Self-regulation identifies neurons for updating in the output layer and distributed coding layer and also adapts the learning rate based on the temporal separation between spikes in the output layer. It also skips learning from samples which are correctly classified with higher temporal separation and hence prevents over-training. The detailed performance comparisons of DC-SNN with other algorithms for SNNs in the literature using six benchmark data set from the UCI machine learning repository has been presented. Further, the performance of DC-SNN is evaluated on a real-world brain computer interface problem for classification of electroencephalogram (EEG) signals recorded during motor-imagery tasks. The results clearly indicate that the proposed DC-SNN architecture provides slightly better generalization ability and is suitable for deep spiking networks.**

***Index Terms*—spiking neural networks, pattern classification, lateral inhibition, self-regulation**

I. Introduction

Spiking Neural Networks (SNN) have attracted a lot of researcher interest in recent times due to their proximity to biological neurons and lower energy footprint. Further, to approximate a given function, SNNs require fewer units than a network with Sigmoidal neurons [1] which highlights their superior computational capacities.

A SNN consists of spiking neurons which process and transmit information using discrete events in time which are termed as spikes. As a result, SNNs naturally require lesser power for realization in comparison to artificial neural networks which use continuous real values to represent information. This makes SNNs an effective alternative to artificial neural networks for applications where minimizing power usage is an important requirement. However, developing effective learning algorithms for SNNs is difficult due to the discontinuous nature of spikes and the 'silent neuron' problem which may arise when neurons stop generating spikes after updating weights.

Due to the closeness between spiking neurons and biological neurons, researchers have often used phenomena observed in the brain as an inspiration for the development of learning algorithms for SNNs. Spike Timing Dependent Plasticity (STDP) is a well established plasticity mechanism in the brain [2], [3] that is the foundation of learning algorithms like Synaptic Weight Association Training (SWAT) [4], ReSuMe [5], Synaptic Efficacy Function-based leaky integrate-and-fire neuRON (SEFRON) [6]. All of these algorithms employ network architectures with a single layer of spiking neurons because of the lack of an effective mechanism to propagate error in network response across multiple layers. This makes these approaches unsuitable for training SNNs with several layers.

To overcome this issue, many researchers employ layer-wise training in a multilayer SNN like [7], [8]. In these approaches each layer in a network is trained independently and the output of neurons in a given well-trained layer serve as the input for the successive layer. These approaches hinge on the idea that by processing the input through multiple such layers, the output of the final layer may be suitable for performing a particular task. The approaches based on this idea can be further divided into algorithms that use class information for training each layer in the network and algorithms that use class information only for training the last layer in the network. In [7], a two stage learning algorithm is developed which uses class information to train each layer in a SNN. However, this method is specifically developed for training a SNN classifier with three layers only.

In [8], a deep convolutional SNN consisting of non-leaky integrate-and-fire spiking neurons is trained in a layer-wise manner using STDP and the output of the final layer in the network is used to train a support vector machine classifier. All neurons in the network, including the input neurons, are allowed to generate a single spike only implying that the information is encoded by the precise time of generated spikes. To improve upon the biological plausibility of this approach, a STDP-based learning approach is proposed for training deep convolutional networks with leaky integrate-and-fire neurons

in [9]. The input to the network was presented using Poisson-distributed spike encoding and information is embedded in the average firing rate of spiking neurons.

Significant performance improvements have been achieved by layer-wise training approaches for SNNs but their biological realization relies on a complex neural circuitry. Further, in layer-wise training approaches only last layer utilizes the class information for learning as there is no mechanism for using feedback from the last layer to drive learning in other layers. Other layers in the network learn feature representations in an unsupervised manner. As a result, the activity of neurons in the intermediate layers may lead to lower classification performance. Recently, a STDP-based learning scheme for a deep SNN has been proposed for simultaneously learning weights in all layers of the network [10]. However, the approach employs a biologically implausible neuron model with separate integration processes for training and testing.

While STDP has been employed in the development of several learning algorithms for SNNs, the canonical form of STDP may results in a network consisting of neurons that have learned to encode overlapping representations of inputs from different classes [11]. To overcome these issues, we propose a SNN architecture which consists of a distributed coding layer that helps the network to learn non-overlapping representations for input spike patterns. The architecture has been named Distributed Coding SNN (DC-SNN). The distributed coding layer in SR-STDP consists of neurons that are connected to other neurons in the layer via inhibitory connections. As a result, the first neuron that fires in this layer suppresses the other neurons from firing.

To train DCSNN using biologically plausible mechanisms, we propose a Self-regulated Competitive STDP (SR-STDP) based learning algorithm. SR-STDP utilizes only the locally accessible information on a synapse to update its weight. Depending on the spike patterns that are generated by the output neurons, SR-STDP self-regulates its learning strategy for each neuron in the network. For output neurons, the weights of the output neuron associated with the same class as the class label of the current sample is updated using STDP whereas weights for the output neurons associated with other classes are updated using anti-STDP. Furthermore, SR-STDP also takes into account the temporal separation between the spike times of these output neurons while updating their weights. A smaller temporal separation leads to bigger weight updates and vice-versa. This helps SR-STDP achieve better generalization performance. Similarly, while updating the weights of distributed coding neurons, SR-STDP identifies a winner neuron based on their contribution to the membrane potential of output neurons. A single winner neuron from the distributed coding layer is identified for each output neuron. The weights of the winner neuron corresponding to the correct class output neuron are updated using STDP and weights of the winner neuron corresponding to other class are updated using anti-STDP.

The performance of SR-STDP is evaluated using multiple binary and multiclass benchmark problems from the UCI machine learning repository. Furthermore, the results of performance evaluation have been compared with other well-known learning approaches for SNNs, namely SpikeProp [12], SWAT [4], Self-Regulating Evolving Spiking Neural (SRESN) classifier [13], Two stage Margin Maximization SNN (TMM-SNN) [7] and SEFRON. The comparison clearly indicates that the performance of SR-STDP is similar to that of TMM-SNN and is better than other algorithms used for comparison. Furthermore, SR-STDP and TMM-SNN employ compact network architectures in comparison to other algorithms. Additionally, to study the effectiveness of SR-STDP , we also evaluated its performance on the real-world problem of motor-imagery classification using electroencephalogram (EEG) signal. It can be clearly observed from the results of performance evaluation that SR-STDP performs better than or similar to the other well-known learning algorithms for this problem.

The remaining sections in this paper are organized as follows: Section II describes the architecture of DCSNN and its SR-STDP based learning algorithm. Section III presents the results of performance evaluation of SR-STDP on benchmark problems and a real-world problem. Section IV summarizes the conclusions from this study.

II. Self-regulated learning algorithm for distributed coding based SNN

This sections presents the architecture of proposed distributed coding based SNN and its self-regulated learning algorithm. First, we present the proposed SNN architecture.

A. Distributed coding based SNN

The architecture of the proposed distributed coding based multi-layer spiking neural network which is succinctly termed as DC-SNN is shown in the Figure 1. DC-SNN employs a four layers architecture to approximate the functional relationship between m-dimensional feature space ($\mathbf{x} = [x_1, \cdots, x_i, \cdots, x_m]$) and coded class label ($\mathbf{y} = [y_1, y_2, \cdots, y_n]$). The first layer consist of receptive field neurons which transform the real-valued features into a spike pattern using a population encoding mechanism [13]. Each feature x_i is coded into spikes using 6 receptive field neuron. Note that the receptive field layer does not have any parameters for learning. The response of j^{th} receptive field neuron connected to the i^{th} feature denotes the spike time for a given input features and is given as

$$\phi_{ij} = \left(1 - exp\left(-\frac{(x_i - \mu_{ij})^2}{2\sigma^2}\right)\right) \times T, \tag{1}$$

where $\mu_{ij} = x_i^{min} + \frac{(2h-3)}{2} \times \frac{x_i^{max} - x_i^{min}}{4}$ is the center of j^{th} receptive field neuron and $\sigma = \alpha \frac{x_i^{max} - x_i^{min}}{4}$ is the width of the receptive field neuron. α is termed as the overlapping factor. The encoding interval T is set to 300ms in population encoding.

The second layer employs H_K Leaky-Integrate-Fire (LIF) neuron with lateral inhibition in the structure. The lateral inhibitory connections are set between the neurons such that they sent a strong inhibitory post-synaptic potential to its

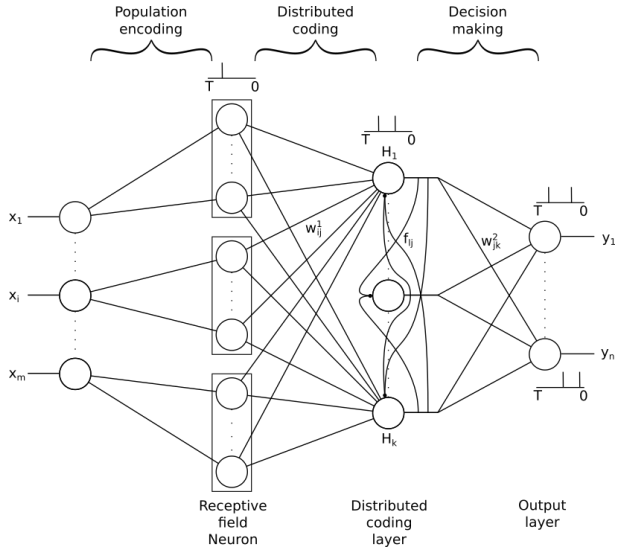

Fig. 1. A Distributed Coding based Multi-layer Spiking Neural Network Architecture.

neighbour. This forces these neurons to learn different spike patterns present in the data. Thereby, this layer achieves a distributed coding of the input spike pattern received from the receptive field neurons. The response of j^{th} LIF neuron in the distributed coding layer is determined based on the membrane potential contributed by the pre-synaptic spike patterns ($\bar{\phi}$) and inhibitory membrane potential resulting from the inhibitory connections between LIF neurons in the same layer. The post-synaptic potential in k^{th} neuron in the distributed coding layer is

$$v_k^d(t) = \sum_{i,j} w_{ijk}^1 \epsilon\left(t - \phi_{ij}\right) - \sum_{l,l\neq k} f_{lk}\epsilon\left(t - t_l\right) \tag{2}$$

where t_l denotes the spike time of neurons in the distributed coding layer for $l \in \{1, \cdots, H_K; l \neq k\}$. w_{ijk}^1 is the synaptic weight between the j^{th} receptive fields neurons encoding x_i and k^{th} neuron in the distributed coding layer. f_{lk} is the synaptic weight of the inhibitory connection between l^{th} and k^{th} neuron in the distributed coding layer. $\epsilon(\cdot)$ is the spike response function [14]. The k^{th} LIF neuron fires if the $v^d(t)$ crosses the threshold θ. The response of the k^{th} neuron in distributed coding layer is $\left[t_k^1, t_k^2, \cdots t_k^{g_k}\right]$ where g_k is the number of spikes generated by a neuron in the distributed computing layer.

The output layer consist of n LIF neurons without any inhibitory connections. The post-synaptic potential of the h^{th} neuron in the output layer at time t is

$$v_h^o(t) = \sum_k w_{kh}^2 \sum_j \epsilon\left(t - t_k^j\right) \tag{3}$$

where w_{kh}^2 is the synaptic weight connecting k^{th} neuron in the distributed coding layer to the h^{th} output neuron. The response of the h^{th} output neuron with r_h spikes is $y_h = \left[t_h^1, t_h^2, \cdots, t_h^{r_h}\right]$ where r_h is the number of spikes generated by a neuron in the output layer.

The first spike of the output neuron is used to determine the predicted class label. The predicted class label ($\hat{c}$) is

$$\hat{c} = \arg \min_h t_h^1, \qquad h = 1, 2, \cdots, n \tag{4}$$

The aim of the learning algorithm for DC-SNN is to determine the synaptic weights ($\mathbf{W^1}$ and $\mathbf{W^2}$) such that the predicted class-label is same as the actual class-label. In the next section, we will describe the learning algorithm for DC-SNN.

B. Self-regulated Learning Algorithm

In this section, we describe the new self-regulated learning algorithm which employs locally accessible information on a synapse along with the membrane potential of a post-synaptic neuron for learning. The self-regulated algorithm uses competitive learning approach to select the neurons for update. In the output layer, learning selects a 'correct class neuron' (CCN) neuron based on the target class and the next winner from the other classes based on minimum spike time (OCN).

$$OCN = \arg \min_{h,h\neq c} t_h^1 \tag{5}$$

where c is the actual class label.

The CCN and OCN neuron uses STDP and anti-STDP for updating the synaptic weights, respectively.

$$w_{k,CCN}^2 = w_{k,ccn}^2 + \eta \Delta w_{k,CCN}^2\left(t_{CCN}^1\right) \tag{6}$$

$$w_{k,OCN}^2 = w_{k,OCN}^2 - \eta \Delta w_{k,OCN}^2\left(t_{OCN}^1\right) \tag{7}$$

where η is the adaptive learning rate. The change in weight $\Delta w(.)$ is computed using the spike-time dependent plasticity [2], [3].

Similarly, a pre-synaptic spike ($t_{k^*}^{g^*}$) generated by a neuron from distributed coding layer is identified which contributed maximum postsynaptic potential of CCN.

$$g^*, k^* = \arg\max_{g,k} w_{k,CCN}\epsilon\left(t_{CCN}^1 - t_k^g\right) \tag{8}$$

For this selected spike ($t_{k^*}^{g^*}$), STDP has been applied to update the synaptic weights between the receptive field neurons and neuron k^* in the distributed coding layer.

$$w_{i,j,k^*}^1 = w_{i,j,k^*}^1 + \eta \Delta w_{i,j,k^*}^1\left(t_{k^*}^{g^*}\right) \tag{9}$$

Also, the inhibitory connections are updated using STDP.

$$f_{l,k^*} = f_{l,k^*} - \eta \Delta f_{l,k^*}\left(t_{k^*}^{g^*}\right) \tag{10}$$

In order to maximize the temporal separation between first spikes generated by CCN and OCN, a similar procedure as described above is also followed for OCN. In this case, a pre-synaptic spike ($t_{s^*}^{o^*}$) generated by a neuron in the distributed coding layer is identified which contributed maximum post-synaptic potential of OCN. This is be mathematically represented as

$$s^*, o^* = \arg\max_{s,o} w_{o,OCN}\epsilon\left(t_{OCN}^1 - t_o^s\right) \tag{11}$$

TABLE I
DESCRIPTION OF BINARY AND MULTI-CLASS DATA SETS USED FOR EVALUATION

Data set	# Features	# Classes	# Samples	
			Training	Testing
Breast Cancer	9	2	350	333
Liver	6	2	170	175
PIMA	8	2	384	384
Ionosphere	33	2	175	176
Iris	4	3	75	75
Wine	13	3	60	118

For this selected spike ($t_{s^*}^{o^*}$), anti-STDP has been applied to update the synaptic weights between the receptive field neurons and o^*th distributed coding layer neuron. The weight update due to anti-STDP is given by

$$w_{i,j,s^*}^1 = w_{i,j,s^*}^1 - \eta \Delta w_{i,j,s^*}^1 \left(t_{s^*}^{o^*}\right) \tag{12}$$

Also, the inhibitory connections are updated using anti-STDP.

$$f_{l,s^*} = f_{l,s^*} - \eta \Delta f_{l,s^*} \left(t_{s^*}^{o^*}\right) \tag{13}$$

It has been shown in the literature [15]–[18] that self-regulation in learning helps in better convergence and improves the generalization ability. It is also shown in [13] that self-regulation helps in online evolving SNN architecture. Since the proposed learning algorithm is based on batch learning, the participation of sample in particular epoch is decided based on the following criterion:

$$t_{OCN}^1 - t_{CCN}^1 < T_d$$

where T_d is the expected time difference between the first spikes of CCN and OCN. If a a given spike pattern satisfy the above criterion then it will be participating in the learning process. Thereby, the selective learning process helps in preventing over-training and improves the generalization process.

Further, the selection of learning rate influence the convergence characteristics of batch learning algorithm significantly. Hence, in this paper, we employ a novel adaptive learning rate mechanism which is given by

$$\eta = 0.001 \times \max\left(\min\left(1 - \frac{t_{OCN}^1 - t_{CCN}^1}{T_d}, 2\right), 0.5\right) \tag{14}$$

The learning rate increases or decreases based on the correct classification and the margin of separation between the first spikes of CCN and OCN. The learning decreases by half when the margin of separation is equal to T_d and the sample is correctly classified. Similarly, the learning will be doubled when the sample is mis-classified and the margin of separation is greater than $-T_d$.

To summarise, the self-regulation learning algorithm maximize the margin of separation and novel adaptive learning rate helps in faster convergence. Further, the lateral inhibitory connection in the distributed coding layer projects the receptive field neuron into discriminating features. The pseudo-code for the learning algorithm is given in 1.

Algorithm 1 Pseudocode for Self-regulated Learning Algorithm

```
for each spike pattern in training do
    Simulate spike pattern on DC-SNN for simulation interval(T̄) to generate output spike pattern.
    CCN ← Target class neuron.
    Compute OCN, j*, k*, s* and o* using (5), (8) and (11).
    if t^1_CCN = ∞ (CCN does not fire) then
        t^1_CCN ← T̄
        η ← η_0
        Perform distributed coding layer to output layer weight update using (6).
    else if t^1_OCN − t^1_CCN < T_d then
        Compute adaptive learning rate using equation (14).
        Perform distributed coding layer to output layer weight update using (6) and (7).
        Perform receptive field neuron layer to distributed coding layer weight update using (9) and (12).
        Perform distributed coding layer inhibitory connection weight update using (10) and (13).
    end if
end for
```

III. PERFORMANCE EVALUATION

In this section, the classification performance of DC-SNN is evaluated on multiple binary and multiclass benchmark problems from the UCI machine learning repository [19]. In addition, a comparison between the performance of DC-SNN and SpikeProp [12], SWAT [4], Self-Regulating Evolving Spiking Neural (SRESN) classifier [13], Two stage Margin Maximization SNN (TMM-SNN) and SEFRON is presented. Further, to study the suitability of DC-SNN for real world problems, its performance is also evaluated on the data set available from the BCI competition IV.

The metrics used in performance comparison include training/testing accuracy, number of iterations required for convergence and number of network parameters estimated during training. Classification accuracies (η) during training and testing are equal to the percentage of samples correctly classified during the respective phases, given by

$$\eta = \frac{\text{Number of samples correctly classified}}{\text{Total number of samples}} * 100 \tag{15}$$

The number of network parameters for SWAT, SRESN, TMM-SNN and DC-SNN is equal to $(N_i \cdot N_h + N_h \cdot N_o)$ where N_i, N_h and N_o represent the number of neurons in the input, hidden and output layer of the network, respectively. In case of SpikeProp, every pair of neurons is connected by sixteen synapses with different delays. Therefore, the total number of parameters in SpikeProp is equal to $16 \cdot (N_i \cdot N_h + N_h \cdot N_o)$. SEFRON employs time-varying weights which are represented as a sum of multiple Gaussian functions. The center of each Gaussian function is adapted by the learning algorithm during

TABLE II
COMPARISON BETWEEN CLASSIFICATION ACCURACIES OF DC-SNN , SPIKEPROP, SWAT, SRESN, TMM-SNN AND SEFRON

Data set	Learning Algorithm	Architecture	Training Accuracy (%)	Testing Accuracy (%)	# Epochs	# Network Parameters
Breast Cancer	SpikeProp	55-15-2	97.3 (0.6)	97.2 (0.6)	1000	13680
	SWAT	54-702-2	96.5 (0.5)	95.8 (1.0)	500	1404
	SRESN	54-(8-12)	97.7 (0.6)	97.2 (0.7)	306	(432-648)
	TMM-SNN	54-(2-8)-2	97.4 (0.3)	97.2 (0.5)	70	(112-448)
	SEFRON	55-1	98.3 (0.8)	96.4 (0.7)	100	19250
	DC-SNN	**54-8-2**	**97.4 (0.7)**	**97.8 (0.5)**	**1000**	**448**
Liver	SpikeProp	37-15-2	71.5 (5.2)	65.1 (4.7)	3000	9360
	SWAT	36-468-2	74.8 (2.1)	60.9 (3.2)	500	936
	SRESN	36-(6-9)	60.4 (1.7)	59.7 (1.7)	715	(216-324)
	TMM-SNN	**36-(5-8)-2**	**74.2 (3.5)**	**70.4 (2.0)**	**442**	**(190-304)**
	SEFRON	37-1	91.5 (5.4)	67.7 (1.3)	100	6290
	DC-SNN	36-8-2	68.8 (4.4)	70.0 (2.0)	1000	304
PIMA	SpikeProp	55-20-2	78.6 (2.5)	76.2 (1.8)	3000	16640
	SWAT	54-702-2	77.0 (2.1)	72.1 (1.8)	500	1404
	SRESN	54-(9-14)	70.5 (2.4)	69.9 (2.1)	254	(486-756)
	TMM-SNN	**54-(5-14)-2**	**79.7 (2.3)**	**78.1 (1.7)**	**160**	**(280-784)**
	SEFRON	49-1	84.1 (1.5)	74.0 (1.2)	100	18816
	DC-SNN	49-12-2	78.6 (1.9)	77.8 (1.2)	1000	612
Ionosphere	SpikeProp	205-25-2	89.0 (7.9)	86.5 (7.2)	3000	82800
	SWAT	204-2652-2	86.5 (6.7)	90.0 (2.3)	500	5304
	SRESN	204-(16-23)	91.9 (1.8)	88.6 (1.6)	1018	(3264-4692)
	TMM-SNN	204-(23-34)-2	98.7 (0.4)	92.4 (1.8)	246	(4738-7004)
	SEFRON	199-1	97.0 (2.5)	88.9 (1.7)	100	34825
	DC-SNN	**199-20-2**	**97.1 (1.2)**	**92.7 (1.5)**	**1000**	**4020**
Iris	SpikeProp	25-10-3	97.2 (1.9)	96.7 (1.6)	1000	4480
	SWAT	24-312-3	96.7 (1.4)	92.4 (1.7)	500	936
	SRESN	24-(6-10)	96.9 (1.0)	97.3 (1.3)	102	(144-240)
	TMM-SNN	24-(4-7)-3	97.5 (0.8)	97.2 (1.0)	246	(108-189)
	SEFRON	-	-	-	-	-
	DC-SNN	**24-11-3**	**96.1 (2.9)**	**97.7 (1.4)**	**1000**	**297**
Wine	SpikeProp	79-10-2	99.2 (1.2)	96.8 (1.6)	1000	12960
	SWAT	78-1014-3	98.6 (1.1)	92.3 (2.4)	500	2028
	SRESN	78-(5-10)	96.9 (1.6)	91.0 (1.2)	128	(390-780)
	TMM-SNN	**78-3-3**	**100 (0)**	**97.5 (0.8)**	**80**	**243**
	SEFRON	-	-	-	-	-
	DC-SNN	78-10-3	98.2 (2.0)	96.3 (1.0)	1000	810

training. The total number of Gaussian functions is equal to $(N_i \cdot N)$ where N is the number of samples used for training.

The experiments pertaining to evaluation of DC-SNN have been conducted in Python 3.7 on a machine running Windows operating system with 16 GB memory. The machine had a CPU with 6 cores and a speed of 3.2 GHz. The real valued features in the data sets are converted to spike patterns using the well-known population coding scheme [13]. As in [7], each feature is encoded using six receptive field neurons which generate a spike in the interval $[0, 300]$ milliseconds. The overlap constant for population coding is set to 0.7. The network is simulated for a duration of 900 milliseconds for each sample. Time constant for the LIF neuron is chosen as 300 milliseconds. For all simulations T_d is set to 50 milliseconds. A time constant of 50 millisecond and a magnitude of 1 is used for STDP in all simulations presented in this section.

A. Performance Evaluation

In this section, the classification accuracy of DC-SNN is evaluated and compared with the performance of SpikeProp, SWAT, SRESN, TMM-SNN and SEFRON on four binary and two multiclass problems from the UCI machine learning repository [19]. Table I provides details pertaining to different attributes of the datasets used for comparison.

The results of performance evaluation for SpikeProp, SWAT, SRESN and TMM-SNN have been reproduced from [7] and for SEFRON, results have been reproduced from [6]. The training/testing accuracy of all the algorithms have been estimated over ten random trials. The average and standard deviation of accuracy across these ten trials is reported. It may be noted that SEFRON has been specifically developed for binary classification problems, therefore, it has not been evaluated on the multi-class classification problems.

Table II provides details about the architecture, training/testing accuracy, number of epochs needed for convergence and the number of network parameters for each learning algorithm. The architecture has been shown using the format $N_i - N_h - N_o$. Note that SRESN and TMM-SNN evolve the number of hidden neurons during training which can results in varying number of neurons in models obtained in the ten trials. Therefore, for SRESN and TMM-SNN, number of hidden neurons (N_h) have been shown as a range across all trials.

Results presented in Table II show that, for simple problems like breast cancer, the testing accuracy of DC-SNN is 0.6-

2% better than other algorithms. In terms of convergence, DC-SNN converged after 1000 epochs of training whereas TMM-SNN is the fastest to convergence which required 70 epochs for training. Among different learning algorithms used for comparison, DC-SNN and TMM-SNN employed networks with smallest number of network parameters.

For difficult problems like Liver and PIMA, TMM-SNN has the highest testing accuracy. The testing accuracy of DC-SNN is 0.4% and 0.3% lower than that of TMM-SNN for Liver and PIMA problems, respectively. SEFRON needed smallest number of epochs for convergence but its testing accuracy is lower than both TMM-SNN and DC-SNN . Both TMM-SNN and DC-SNN employed compact network architectures in comparison to the other learning algorithms.

For a problem with large number of features like Ionosphere, DC-SNN has the highest testing accuracy which is 0.3-6.2% than other learning algorithms. SEFRON converged fastest among all learning algorithms used for comparison but its testing accuracy is 3.8% lower than DC-SNN . Further, DC-SNN employed a network architecture with smallest number of network parameters.

For multi-class problems like Iris and Wine, DC-SNN and TMM-SNN have similar testing accuracy which are better than other learning approaches used in the comparison. TMM-SNN converged faster than DC-SNN . Additionally, TMM-SNN employed a compact network architecture in comparison to other learning algorithms. These results show that DC-SNN and TMM-SNN perform better than other learning approaches while using smaller network architecture.

B. Performance of DC-SNN on the BCI Competition IV Data Set

In this section, the classification accuracy of DC-SNN is evaluated on a real-world data set obtained from the BCI competition IV [20]. BCI technologies enable communications between brains and computers. This communication is usually achieved by recording and interpreting the electroencephalogram (EEG) signals recorded from the brain. Such technologies can be very effective in improving the lifestyle of persons with disabilities. However, interpreting EEG signals recorded from the brain is a difficult problem because of their inherently noise nature. The aim of BCI competition IV is to advance research directed at decoding signals recorded from motor areas in the brain.

BCI competition IV consists of multiple data set. In this study, we used the data set for motor-imagery task which is commonly referred as 2a in the competition. The data set used in this study consisted of EEG recordings obtained from 9 subjects performing a motor-imagery task. The recordings are conducted using 22 electrodes with a sampling frequency of 250 Hz. The subjects are asked to imagine moving of one of the four body parts, namely left hand, right hand, both feet and tongue which correspond to four different classes. For each subject, recordings are conducted in two sessions each of which consisted of 288 trials. Each session consisted of equal number of trials from the four classes resulting in 144 trials for each class. For the purpose of this study, we considered the left hand vs right hand classification problem.

TABLE III
PERFORMANCE COMPARISON OF DC-SNN WITH CSP-LDA AND SRIT2NFIS ON BCI COMPETITION IV DATASET 2A

Subjects	CSP-LDA	SRIT2NFIS	DC-SNN
A1	88.89	93.06	92.36
A2	51.39	68.75	66.67
A3	96.53	97.22	96.53
A4	70.14	75.00	77.78
A5	54.86	65.97	63.89
A6	71.53	72.22	73.61
A7	81.25	86.11	82.64
A8	93.75	97.22	97.92
A9	93.75	93.75	93.75
SM	78.01	83.26	82.79
SD	17.01	12.76	12.27

To evaluate the classification performance of DC-SNN , a separate classifier was built for each subject. Out of 144 trials in each class, 72 trials are used for training and remaining trials are used for testing. Each trial is processed using Robust Common Spatial Pattern (RoCSP) for feature extraction resulting in a total of 6 features. Each feature is encoded using six receptive fields and other parameters for population coding are set as described previously. The classification performance of DC-SNN is also compared with the classification performance of Linear Discriminant Classifier (LDA) on features extracted using Common Spatial Patterns (CSP) which is succinctly referred as CSP-LDA.

Table III presents the results of performance comparison between DC-SNN and CSP-LDA for each of the 9 subjects. Additionally, performance evaluation results of Self-Regulated Interval Type-2 Neuro-Fuzzy Inference System (SRIT2NFIS) [21] have also been provided for comparison with an approach that is specifically designed to handle the problem of covariate drift that is prevalent in EEG signals. It may be noted that features for both SRIT2NFIS and SR-STDP have been extracted using RoCSP. The table also provides mean (SM) and standard deviation (SD) in the performance of both learning algorithms across all subjects. It can be clearly observed from the table that DC-SNN performs better than CSP-LDA for all subjects. Overall, the mean performance of DC-SNN across subjects is 4.8% higher than CSP-LDA. With regards to SRIT2NFIS, performance of SR-STDP is similar to the performance of SRIT2NFIS. Furthermore, DC-SNN exhibits lower standard deviation in comparison to both CSP-LDA and SRIT2NFIS in classification performance across different subjects.

IV. CONCLUSIONS

In this paper, a new architecture for multi-layer spiking neural networks referred to as Distributed coding SNN (DC-SNN) in which hidden layer neurons have inhibitory interconnections, has been presented. The proposed self-regulated learning algorithm selects the appropriate samples, neurons in different layer and adapts the learning rate such that the network converges faster. Also, self-regulation helps in preventing over-training. The selected neurons from same class

uses STDP and other neuron uses anti-STDP for synaptic weight update. The performance results indicate that DC-SNN is one of the better multi-layer end-to-end training models without error gradient based back propagation approach for multi-class problems. The performance results also indicate that, the approach followed in this paper is capable of handling the real-world classification problems. The future direction will be extending the proposed learning algorithm for deep spiking neural networks.

REFERENCES

[1] W. Maass, "Noisy Spiking Neurons with Temporal Coding have more Computational Power than Sigmoidal Neurons," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 211–217.

[2] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs," *Science*, vol. 275, no. 5297, pp. 213–215, Jan. 1997.

[3] G.-Q. Bi and M.-m. Poo, "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, Dec. 1998.

[4] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A Spiking Neural Network Training Algorithm for Classification Problems," *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.

[5] F. Ponulak and A. Kasiński, "Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting," *Neural Computation*, vol. 22, no. 2, pp. 467–510, Feb. 2010.

[6] A. Jeyasothy, S. Sundaram, and N. Sundararajan, "SEFRON: A new spiking neuron model with time-varying synaptic efficacy function for pattern classification," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 4, pp. 1231–1240, 2018.

[7] S. Dora, S. Sundaram, and N. Sundararajan, "An interclass margin maximization learning algorithm for evolving spiking neural network," *IEEE transactions on cybernetics*, vol. 49, no. 3, pp. 989–999, 2018.

[8] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, Mar. 2018.

[9] C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep Spiking Convolutional Neural Network Trained With Unsupervised Spike-Timing-Dependent Plasticity," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 3, pp. 384–394, Sep. 2019.

[10] J. C. Thiele, O. Bichler, and A. Dupret, "Event-Based, Timescale Invariant Unsupervised Online Deep Learning With STDP," *Frontiers in Computational Neuroscience*, vol. 12, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2018.00046/full

[11] G. Srinivasan, S. Roy, V. Raghunathan, and K. Roy, "Spike timing dependent plasticity based enhanced self-learning for efficient pattern recognition in spiking neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017, pp. 1847–1854.

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

[13] S. Dora, K. Subramanian, S. Sundaram, and S. Narasimhan, "Development of a Self-Regulating Evolving Spiking Neural Network for classification problem," *Neurocomputing*, vol. 171, pp. 1216–1229, 2016.

[14] S. Dora, S. Sundaram, and N. Sundararajan, "A two stage learning algorithm for a Growing-Pruning Spiking Neural Network for pattern classification problems," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, p. 7.

[15] G. Babu, S. Suresh, and B. Mahanand, "A novel pbl-mcrbfn-rfe approach for identification of critical brain regions responsible for parkinson's disease," *Expert Systems with Applications*, vol. 41, no. 2, pp. 478–488, 2014.

[16] K. Subramanian, S. Ramasamy, and S. Suresh, "A metacognitive complex-valued interval type-2 fuzzy inference system," *Neurocomputing*, vol. 25, no. 9, pp. 1659–1672, 2014.

[17] S. Suresh, K. Dong, and H. Kim, "A sequential learning algorithm for self-adaptive resource allocation network classifier," *Neurocomputing*, vol. 73, no. 16-18, pp. 3012–3019, 2014.

[18] K. Subramanian and S. Suresh, "A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system," *Applied soft computing*, vol. 12, no. 11, pp. 3603–3614, 2012.

[19] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[20] M. Naeem, C. Brunner, R. Leeb, B. Graimann, and G. Pfurtscheller, "Seperability of four-class motor imagery data using independent components analysis," *Journal of Neural Engineering*, vol. 3, no. 3, pp. 208–216, 2006.

[21] A. K. Das, S. Sundaram, and N. Sundararajan, "A Self-Regulated Interval Type-2 Neuro-Fuzzy Inference System for Handling Nonstationarities in EEG Signals for BCI," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 6, pp. 1565–1577, Dec. 2016.