Cyber Fraud Detection using Evolving Spiking Neural Network

Amit Soni Arya[1], Vadlamani Ravi[2], Vadali Tejasviram[3], Neelava Sengupta[4], N.Kasabov[5]

Abstract— **With the rapid growth of the internet, most of the businesses are now moving online. Since the internet is ubiquitous and can be accessed from anywhere websites are susceptible to attacks. One of such attack is phishing website attack. In which an attacker creates a duplicate copy of the website and tries to pose it as a legitimate to steal user's information. So it is the utmost need to detect such phishing websites. Machine learning techniques have been successfully applied to detect the phishing websites. The neural network is one of the efficient ways for detecting these phishing attacks.**

In our work, we have applied the spiking neural network approach to detect these phishing websites. The spiking neural network is biologically inspired by neuroscience literature, evolving spiking neural classifier for the pattern classification problem. We have compared it with various other machine learning techniques and we show that the evolving spiking neural network performs better than the existing machine learning techniques.

I. INTRODUCTION

Artificial Neural Network(ANN) is inspired by the biological nervous system which processes information such as the brain. A spiking neural network comes under third generation neural networks[29], [11], [22], contains the computational units in terms of discrete events with time. It is also called spikes[27], [19]. Each neuron has certain threshold value. The action potential is generated if the potential of a neuron crosses the particular threshold value of the neuron, then the neuron fired. It is also called the "action potential"[8]. Sudden increase in the voltage of the membrane of the neuron is due to the incoming signals generated by the other connected input neurons. The spike is produced by the neuron to transmit the information. The information is represented as trains of spikes[25], [28].

This work is based on neuron model which inspired by the neuroscience literature. It is an evolving spiking neuron model. It contains two layers fixed input layer and a changing output layer. The first need to convert the real-valued data into spikes. The real-valued data also called external stimuli or static data and spikes called internal stimuli[5], [19], [4]. The generated spikes in terms of varying amplitude and times. Each input layer neuron has a specified number of receptive fields also called responders[3], [25]. The output of the each receptive field defines in term of time and amplitude function. The output layer neurons are uses of integrate-and-fire spiking neuron model[12], [17].

Initially, the system starts with zero neurons at the output layer. The output layer neurons are evolved during training of the network. Whenever the new training sample arrives at the network, quickly adapt the new knowledge[14], [23], [15]. The training samples come one by one to the network. The system added the neuron at output layer based on the information stored in the system. The network either adds the neuron at output layer or update the synaptic weights. If the incoming training sample is containing the similar information already hold by the system, then it just updates the synaptic weights, if the information is different i.e. network doesn't have then new neuron added to store the knowledge[6], [30], [31], [9].

II. Literature Review

A. Integrate and Fire (IF) neuron model

This model is the more simplified mathematical model of Hodgkin-Huxley model[12], [8]. In this model, formulation in term of capacitance C and potential u(t), as follows:

$$C\frac{du}{dt} = -\frac{1}{R}(u(t) - u_{rest}) + I(t) \tag{1}$$

$$u(t^{(f)}) = \theta \quad when \quad u'(t^f) > 0 \tag{2}$$

Where R is the resistance, u_{rest} is the resting potential of the membrane, $t^{(f)}$ firing time of neuron, θ is neuron firing threshold, $u(t^{(f)})$ describes the membrane potential of firing neuron, $u'(t^{(f)})$ is its derivative; I(t) is input current[1], [8].

B. Evolving Spiking neuron Network

The evolving spiking neural network (eSNN) the classifier don't have any output neuron at the output layer. The neuron is evolving during the training phase shown in fig.1. The classifier tries to create the output neuron depending on the class label[23]. For every input sample, there is a spike train. In some cases, none of them are fired. So in this case, we need to evolve an extra neuron for that class label sample during the training phase. eSNN grow their functionality and formation in an on-line manner. A new neuron is connected and weight allocated dynamically for every new input pattern[14], [13], [7].

[1]School of Computer and Information Sciences, University of Hyderbad, India, amitsoniuoh@uohyd.ac.in

[2]Center of Excellence in Analytics,Castle Hills, Road No.1, Masab Tank, IDRBT, Hyderabad - 500 057, India, vravi@idrbt.ac.in

[3]Center of Excellence in Analytics, Castle Hills, Road No.1, Masab Tank, IDRBT, Hyderabad - 500 057, India, vtejasviram@gmail.com

[4]KEDRI, Auckland University of Technology, AUT Tower, Level 7, Rutland and Wakefield Street, Auckland 1010, New Zealand, neelava.sengupta@aut.ac.nz

[5]KEDRI, Auckland University of Technology, AUT Tower, Level 7 Corner Rutland and Wakefield Street, Auckland 1010, New Zealand nkasabov@aut.ac.nz

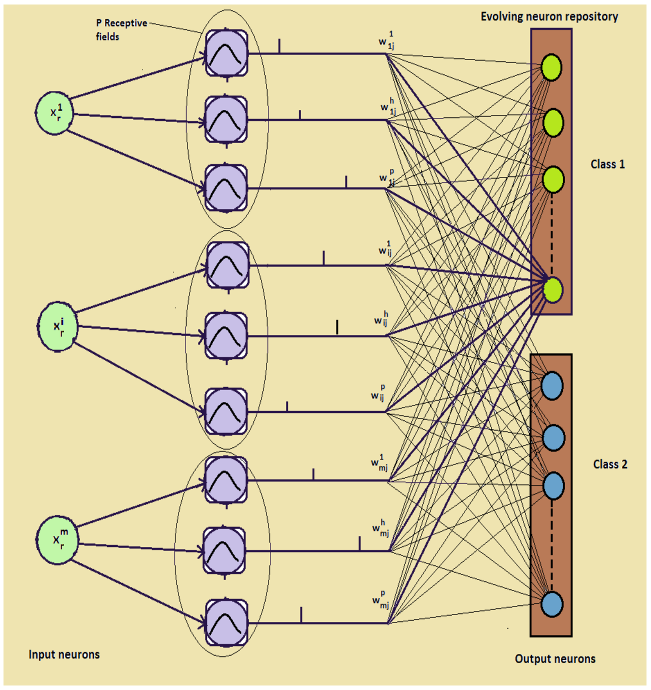

Fig. 1. Evolving spiking neuron architecture

C. Learning in SNN

For the connectionist network, the learning rules, are the changes in the weights of connections in the network. There are many learning standards and algorithms available. Similar to tradition neural networks, in SNN mainly there are three different learning paradigms. Those are mainly referred as supervised, unsupervised and reinforcement learning. Reinforcement learning is probably least common among three in SNN. The Hebbian learning is the form of unsupervised learning. It is one of the most biologically practical learning. This learning is called spike-time dependent plasticity (STDP) [20][14]. Supervised learning imposes certain of the relationship between input and output. This feature is necessary for the practical application of SNN. It is very much inspired by the image processing done by the human eye. It was seen that only a few spikes of the neuron can contribute to the entire treatment of the stimulus. Earlier spike carries most information about the stimulus. The practical example called rank order population encoding. Here some of the most traditional learning methods are discussed.

Three different type of learning can be distinguished in SNN:

a) Supervised learning: In this approach, both the input data and corresponding class label are given for classification. Here the classifier need to learn (train) with given train sample with a class label. For the new input sample to the classifier should predict the class label. Some of the popular examples of supervised learning algorithm are the rank-order (RO) learning and spike prop[10](error backpropagation)[2].

b) Unsupervised learning: In this approach, the input data samples are unlabelled. To interpret the class label training is based on only unlabelled input pattern. STDP is one of the most common unsupervised learning approaches in SNN.

c) Reinforcement learning: In the reinforcement learning based on the "rewards" or "punishments" depends on the success of the learning process[24], [16].

D. Rank order learning

The rank order learning work on the basis of the order of the spikes across all the synapses connected to the particular neuron. It creates the priority in the input spikes depending on the order of spike arrival to the neuron which makes extra information to the network concerning the order of the spike[26], [18].

III. Proposed Methodology

The proposed method is based on the evolving spiking neural network model for classification. It builds on the Thorpe's model, in which early spikes have given more importance, which is highly influenced by the visual pattern recognition system. This method has fast supervised one pass learning. The eSNN have two layers an input layer and an output layer. Initially, there is no output neuron at the output layer. The neurons are added to the output layer depends on the input samples during the training phase.

To deals real-valued data sets, each data sample needs to map with the sequence of spikes using a precise neural encoding technique. Rank order population encoding is used here. It uses the Gaussian receptive fields to encode the real-valued data. It is the most popular mechanism for converting real-valued data (external stimuli) into spike patterns (internal stimuli). Each input goes through a fixed number of Gaussian receptive fields, and it generates a peak at the certain point of time. Following steps are performed during whole classification process:

Rank Order Population Encoding is based on rank order encoding. Every input encoded individually using a set of P responders (i.e.receptive fields). For i^{th} input neuron with P receptive fields (P>2) whose input feature varies from I^i_{min} to I^i_{max}. The center and width of the h^{th} receptive field is given by:

$$\mu_i^h = I^i_{min} + \frac{(2h-3)}{2}\frac{(I^i_{max} - I^i_{min})}{(P-2)} \tag{3}$$

Where P is a number of receptive fields.

$$\sigma_i^h = \frac{1}{\gamma}\frac{(I^i_{max} - I^i_{min})}{(P-2)} \tag{4}$$

Where γ directly controls the width of the receptive field means how much overlap between two receptive fields.

The τ_i^h is the firing time of the neuron calculated using:

$$\tau_i^h = \lfloor T(1-\phi_i^h) \rfloor \tag{5}$$

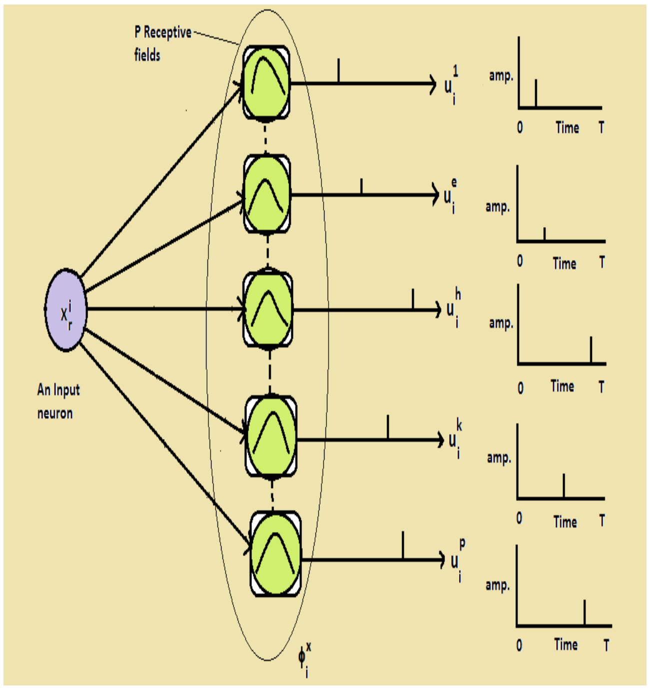

Fig. 2. An input neuron model

Where T is the simulation interval and ϕ_i^h is the Gaussian receptive field output defined as follows:

$$\phi_i^h = exp\left(-\frac{(x_r^i - \mu_i^h)^2}{2(\sigma_i^h)^2}\right) \tag{6}$$

The output of the h^{th} responder of the i^{th} input neuron is given by:

$$u_i^h(t, x_r^i) = f_i^h(x_r^i)\delta_i^h(t - \tau_i^h) \tag{7}$$

Where $f_i^h()$ is spike amplitude function and $\delta_i^h()$ is dirac delta function or firing time function.
The amplitude of the spikes $f_i^h()$ is given by:

$$f_i^h(x_r^i) = \frac{(\lambda)^{r_i^h}}{1 + |x_r^i - \mu_i^h|} \tag{8}$$

Where r_i^h is the rank of the h^{th} responder of i^{th} neuron and λ is the slope of amplitude function. The rank of the spikes is determines using the ranking function as follows:

$$F_R(x, y) = \begin{cases} 0, & \phi_i^x \geq \phi_i^y \\ 1, & otherwise \end{cases}$$

Where x and y are the indices of the any two receptive field of the i^{th} neuron.
The rank of spike generate by the h^{th} receptive field of the i^{th} neuron is given by:

$$r_i^h = 1 + \sum_{y=1, y\neq h}^{P} F_R(h, y) \tag{9}$$

The above whole processes are of the spike generation. Now, after the spike generation needs to create the output neuron depending on the input sample, establish the synaptic connection between the input and output neuron and initialize the synaptic weight. It is a sequential learning architecture starts with zero output neuron. The algorithm is either chosen to add the new neuron at output layer or update the synaptic weight for training sample. In the presented network, there are three measures are formed:
An addition of output neuron: To create the new neurons and organize the links between the input and output neurons. The new output neuron store the hidden knowledge contain by the new data sample.
Conflict resolution approach: If any misclassification happens, to handle this situation the nearest neuron of the same class goes into long-term potentiation (LTP) and closest output neuron of the different class synaptic weight undergoes in long-term depression (LTD).
Synaptic weight update approach: If the new sample contains the similar to the knowledge already contains the network, then synaptic weight undergoes in long-term potentiation.
The first input sample, first output neuron created and synaptic weight is initialized as follows:

$$w_i = f \tag{10}$$

Where $f = [f_1^1, ...f_1^h, ..., f_m^P]$ and $w_k = [w_{11}^1, ..., w_{11}^h, ...w_{11}^P, ..., w_{m1}^P]$ The neuron is fire at certain threshold value, so the threshold $(\theta_1)$of the neuron given by:

$$\theta_1 = \alpha w_1^T w_1 \tag{11}$$

Where the α is the threshold factor which control the how easy the output neuron to generate the spike for the similar sample to network.
Addition of output neuron output neuron addition strategy to evolve the neuron if the current sample satisfies the following condition:

$$\hat{c} = \{\phi\} \; OR \; (c \notin c_{overall}) \; OR (c \neq \hat{c} \; AND \, \|f - w_{\text{nrs}}\| > \beta_a)$$

Where nrs is nearest output neuron of same class, β_a is distance threshold constant. f is set of current sample spike amplitude response and w_{nrs} is the existing weight of synapse of the same class in the network. ϕ represents for the current sample none of the output neurons fired. $c_{overall}$ is the class label associated with the current output layer neurons and c is the actual class label input training sample. The nearest output neuron is evaluated using the Euclidean distance between the current sample amplitude response and existing synaptic weight of all the output neurons of the same class.
If the new sample satisfies the above condition then new output (k+1) neuron is added its synaptic weight and threshold is given as:

$$w_{k+1} = f \tag{12}$$

$$\theta_{k+1} = \alpha w_{k+1}^T w_{k+1} \tag{13}$$

Conflict resolution approach: In this strategy, When the current training sample associated class is not same as the class associated with the fired output neuron. It means the different class label neuron fire first (nrl) and there exists another nearest output neuron of the same class (nrs). So the weight vector of different classes associated with output neurons is conflicting in nature. To detect this conflict problem following condition should be satisfied:

$$c \neq \hat{c} \quad AND \, \|f - w_{\text{nrs}}\| < \beta_a$$

To resolve this conflict, the nearest neuron of the same class goes long term potentiation, and output neuron of the different class goes into long term depression. The synaptic weight update is as follows:

$$w_{nrs} = (1 - \eta_{nrs})w_{nrs} + \eta_{nrs} f \tag{14}$$

$$\theta_{nrs} = (1 - \eta_{nrs})\theta_{nrs} + \eta_{nrs}\alpha f^T f \tag{15}$$

Where η_{nrs} is self-adaptive learning factor. It is also called self-adaptive potentiation factor of output neuron, early given higher priority later stages for the potentiation factor decreases automatically for similar training sample. The factor η_{nrs} for output neuron given as:

$$\eta_{nrs} = \frac{\eta_{nrs}}{1 + \eta_{nrs}} \tag{16}$$

The output neuron of different class goes to long-term depression so the effect of this the output neuron will not fire for the similar samples and the weight update as follows:

$$w_{nrl} = (1 + k)w_{nrl} - kf \tag{17}$$

Where k is the depression factor, which controls the synaptic weight depression factor. It is close to zero because the higher value of k will result in a massive shift in the synaptic weight. Resulting information loss will happen which is stored in the network.

Synaptic weight update approach: If the actual class label (c) is same as the class label associated with the fired output neuron then the synaptic weight of connection and threshold of the output neuron are updated as follow:

$$w_{nrs} = (1 - \eta_{nrs})w_{nrs} + \eta_{nrs} f \tag{18}$$

$$\theta_{nrs} = (1 - \eta_{nrs})\theta_{nrs} + \eta_{nrs}\alpha f^T f \tag{19}$$

where nrs is the output neuron of same class fired first.

The voltage of the output neuron at any time calculated as follows:

$$V_j(t_1) = \sum_{t=0}^{t_1} \sum_{h=1,i=1}^{P} u_i^h(t, x_r^i) w_{ij}^h \tag{20}$$

The pseudo code of the proposed algorithm is given as follows:

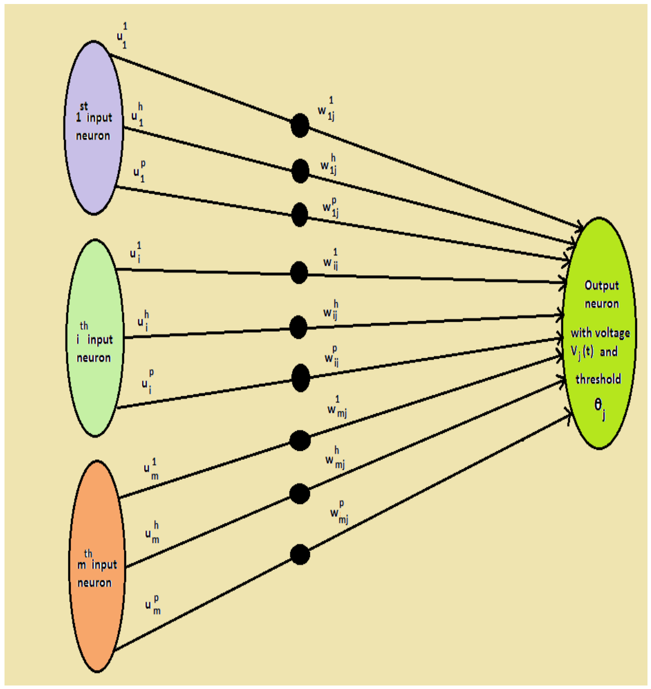

Fig. 3. An output neuron model

```
Sequential learning classification algorithm using eSNN
Require: Input training samples with class label
   $[(x_1, c_1), (x_2, c_2), ..., (x_r, c_r), ..., (x_n, c_n)]$
$\star$ Network has m input neurons indexed by i
$\star$ Every input neuron has P receptive fields indexed by h
for i $\leftarrow$ 1 to n do
$f_i^h \leftarrow$ amplitude of the $h^{th}$ responder of $i^{th}$ neuron $\forall i, h$
$\delta_i^h \leftarrow$ firing time of output neuron        neuron $\forall i, h$
$\hat{c} \leftarrow$ predicted class label
if($\hat{c} = \{\phi\}$   $OR$   $c \notin c_{overall}$   $OR$
          $(c \neq \hat{c} \quad AND \, \|f - w_{\text{nrs}}\| > \beta_a)$)
    $\star$ output neuron addition strategy
  $w_{k+1} = f \qquad \forall i, h$
  $\theta_{nrs} \leftarrow \alpha w^T w$
elsif $(c \neq \hat{c} \quad AND \, \|f - w_{\text{nrs}}\| < \beta_a)$ then
    $\star$   conflict resolution strategy
  $w_{nrs} = (1 - \eta_{nrs})w_{nrs} + \eta_{nrs} f$
  $\theta_{nrs} = (1 - \eta_{nrs})\theta_{nrs} + \eta_{nrs}\alpha f^T f$
  $w_{nrl} = (1 + k)w_{nrs} - kf$
  $\eta_{nrs} = \frac{\eta_{nrs}}{1+\eta_{nrs}}$
elseif $(c = \hat{c})$ then
    $\star$ synaptic weight update strategy
  $w_{nrs} = (1 - \eta_{nrs})w_{nrs} + \eta_{nrs} f$
  $\theta_{nrs} = (1 - \eta_{nrs})\theta_{nrs} + \eta_{nrs}\alpha f^T f$
  $\eta_{nrs} = \frac{\eta_{nrs}}{1+\eta_{nrs}}$
  end if
end if
end for
```

A. *Parameter selection of eSNN classifier*

There are different types parameters used for the classification task. The parameters are varies depending on the dataset. A first parameter is the number of receptive fields (P). During the experiment, we observe that the number of receptive field increases the firing time decreases. The number of receptive fields depending on the dataset. The number of the receptive field should be in the range of 5 to 11. It also controls the amplitude of the input neuron. For the experimented dataset, the number of the receptive field taken is 8. The second parameter is γ overlap factor which controls the overlap between receptive fields which regulates the width of the receptive field. It influences equally to the width of the each receptive field hence impact only the firing time function. In the experiment, the γ value is taken 3, which means 30% overlap between two subsequent receptive fields. It controls the range of the of receptive field and useful for temporal coding. The third factor is λ represent the slope of the amplitude function $f()$. There are other parameters α, β and k are learning parameters. The parameter α is threshold fraction which controls the firing time of the output neuron. In the experiment, we perform 10-fold cross validation method to perform to predict the class label of each fold. The value of α should be in the range [0.5-0.9]. γ is the self-adaptive potential factor. Its value is initialized to 0.5 and decay to 0. It gives higher importance to the similar near future and less importance when it many time appears to network. Another parameter β is a constant, Which controls the addition of the neuron at the output layer. The range of the parameter β depends on the number of features if the number of features is more in the dataset the value of β should be small. The range of β is [0.5-0.8]. The depression would factor k control if the neuron fired wrongly during training the corresponding weights goes into long term depression. The value of k should be tiny. Otherwise, the information which is stored in the network will be lost. The range of k is varies between [0.01-0.35].

B. *Proposed testing method of eSNN classifier*

After the training is over, the knowledge stored in the network. For testing, first, need to convert the real-valued input data into the spikes. Each output neuron associated with certain threshold value calculated during the training. If the incoming potential (signal) summation crosses the specific threshold value of the particular neuron, then neuron fired, and the corresponding class label is predicted. It is often case many output neurons fired for the given input. To resolve this issue, we apply here the k-nearest neighbor to determine the predict the class label. To get the optimal accuracy we check the different value of k to predicted the class label, the value of k are 3, 5, 7, 9, 11, 13, 15,..., etc. The k-nearest neighbor is performed using the Euclidean distance between the amplitude of spikes of the current input testing sample and the synaptic weight corresponding to the fired neuron which is stored by the network at the time of training i.e. $\|\mathbf{f}-\mathbf{w}\|$ with weights corresponding to each fired output neuron.

IV. Dataset description

To test the network, we took the web phishing dataset for phishing website detection. For the experiment, we analysed 200 phishing websites. In which there are 50% phishing website's URLs and rest are legitimate website's URLs. The URLs are collected from PhishTank (www.phishtank.com). The dataset is built by extracting the features based on t-statistics values from web pages source code and URLs. In the phishing website dataset, we have 17 features.

V. Results and Analysis

TABLE I
Comparison of performance of eSNN with other iterative methods for average 10-Fold data on phishing website data

Classifiers	Senstivity	Specificity	Accuracy
GP	99	100	99.5
eSNN	93	92	92.5
LR	91	88	89.5
MLP	89	80	84.5
CART	94	90	92
GP+CART	94	87	90.5

TABLE II
Comparison of performance of one-pass methods eSNN with PNN for average 10-Fold data on phishing website data

Classifiers	Senstivity	Specificity	Accuracy
eSNN	93	92	92.5
PNN	89	90	89.5

TABLE III
t-test based model comparison

Classifiers	t-statistic value (Accuracy)
eSNN vs. GP	4.33197
eSNN vs. LR	1.12942
eSNN vs. PNN	1.1767
eSNN vs. MLP	3.04188
eSNN vs. CART	0.21822
eSNN vs. GP+CART	0.86155

The eSNN classifier has been very much suited to detect the fraudulent activity. The experiments performed on the web phishing data sets. We obtain 92.5 % accuracy in web phishing data sets. In the table I and II eSNN compared with the iterative method and one pass learning method respectively. The eSNN and PNN both are one pass learning. In the Table-2 comparison of one pass learning algorithms, the eSNN perform better than the PNN. In the phishing website dataset, we compare eSNN with the different models [21] based on t-statistic value. The t-test values compare with eSNN and other models GP, LR, PNN, MLP, CART and GP+CART are 4.33197, 1.12942, 1.1767, 3.04188, 0.21822 and 0.86155 respectively in table III. The t-test value is less than 2.83 (t-table value concerning 18 degrees of freedom, i.e., 10+10-2=18) in all the classifier except GP and MLP. For all the cases it performs statistically same but in two cases, i.e., GP and MLP, need to consider the accuracy of these two classifiers, GP gives the higher accuracy than eSNN, so GP is better in this case. Compare with MLP the eSNN perform better. So, eSNN is statistically performing better than the MLP.

VI. Conclusions

The proposed methodology uses the evolving spiking neural network which is very much adaptive if any changes happen in the input data then it easily learn. The network is very much flexible to adapt new changes in the environment. In the presented method has two layers input layer and the output layer. The output layer is evolving in behaviour. In input layer first, we convert the static data (real value data) to the into spikes. The output layer based on the self-adaptive learning and update synaptic weights using both long term potentiation and long term depression. The weight update mechanism uses computationally inexpensive rules which based on only elementary algebraic operations.

For the performance of the proposed eSNN architecture are very much depends on the parameter tuning. In the proposed method there are around seven parameters need to tune to get the better results. These parameters are very much influence to the network performance. The parameters selection and tuning are the major challenges of this network.

References

[1] L. F. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5):303–304, 1999.

[2] S. M. Bohte, J. N. Kok, and H. La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1):17–37, 2002.

[3] S. M. Bohte, H. L. Poutré, and J. N. Kok. Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer rbf networks. *Neural Networks, IEEE Transactions on*, 13(2):426–435, 2002.

[4] J. M. Brader, W. Senn, and S. Fusi. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural computation*, 19(11):2881–2912, 2007.

[5] L. Buesing, J. Bill, B. Nessler, and W. Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7(11):e1002211, 2011.

[6] K. Dhoble, N. Nuntalid, G. Indiveri, and N. Kasabov. Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–7. IEEE, 2012.

[7] S. Dora, S. Suresh, and N. Sundararajan. A sequential learning algorithm for a spiking neural classifier. *Applied Soft Computing*, 36:255–268, 2015.

[8] W. Gerstner and W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[9] S. Ghosh-Dastidar and H. Adeli. Improved spiking neural networks for eeg classification and epilepsy and seizure detection. *Integrated Computer-Aided Engineering*, 14(3):187–212, 2007.

[10] S. Ghosh-Dastidar and H. Adeli. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, 22(10):1419–1431, 2009.

[11] S. Ghosh-Dastidar and H. Adeli. Spiking neural networks. *International Journal of Neural Systems*, 19(04):295–308, 2009. PMID: 19731402.

[12] E. M. Izhikevich et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[13] N. Kasabov. Evolving spiking neural networks and neurogenetic systems for spatio-and spectro-temporal data modelling and pattern recognition. In *Advances in Computational Intelligence*, pages 234–260. Springer, 2012.

[14] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri. Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, 41:188–201, 2013.

[15] N. Kasabov, V. Feigin, Z.-G. Hou, Y. Chen, L. Liang, R. Krishnamurthi, M. Othman, and P. Parmar. Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. *Neurocomputing*, 134:269 – 279, 2014. Special issue on the 2011 Sino-foreign-interchange Workshop on Intelligence Science and Intelligent Data Engineering (IScIDE 2011)Learning Algorithms and Applications-Selected papers from the 19th International Conference on Neural Information Processing (ICONIP2012).

[16] N. Kubota. Computational intelligence for structured learning of a partner robot based on imitation. *Information Sciences*, 171(4):403 – 429, 2005. Intelligent Embedded Agents.

[17] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[18] W. Maass. On the computational power of winner-take-all. *Neural computation*, 12(11):2519–2535, 2000.

[19] W. Maass and C. M. Bishop. *Pulsed neural networks*. MIT press, 2001.

[20] T. Masquelier, R. Guyonneau, and S. J. Thorpe. Competitive stdp-based spike pattern learning. *Neural computation*, 21(5):1259–1276, 2009.

[21] M. Pandey and V. Ravi. Text and data mining to detect phishing websites and spam emails. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 559–573. Springer, 2013.

[22] H. Paugam-Moisy and S. Bohte. Computing with spiking neuron networks. In *Handbook of natural computing*, pages 335–376. Springer, 2012.

[23] S. Schliebs and N. Kasabov. Evolving spiking neural network-a survey. *Evolving Systems*, 4(2):87–98, 2013.

[24] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[25] S. Thorpe, A. Delorme, and R. Van Rullen. Spike-based strategies for rapid processing. *Neural networks*, 14(6):715–725, 2001.

[26] S. Thorpe and J. Gautrais. Rank order coding. In *Computational Neuroscience*, pages 113–118. Springer, 1998.

[27] S. J. Thorpe. Spike arrival times: A highly efficient coding scheme for neural networks. *Parallel processing in neural systems*, pages 91–94, 1990.

[28] R. VanRullen, R. Guyonneau, and S. J. Thorpe. Spike times make sense. *Trends in neurosciences*, 28(1):1–4, 2005.

[29] J. Vreeken et al. Spiking neural networks, an introduction. *Institute for Information and Computing Sciences, Utrecht University Technical Report UU-CS-2003-008*, 2002.

[30] S. G. Wysoski, L. Benuskova, and N. Kasabov. On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition. In *Artificial Neural Networks–ICANN 2006*, pages 61–70. Springer, 2006.

[31] J. Xin and M. J. Embrechts. Supervised learning with spiking neural networks. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 3, pages 1772–1777. IEEE, 2001.