IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 7, 2020, accepted March 24, 2020, date of publication April 2, 2020, date of current version April 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2985257

A Multi-Spiking Neural Network Learning Model for Data Classification

BAAGYERE EDWARD YELLAKUOR[1,2], AGEBURE APAMBILA MOSES[2], (Graduate Student Member, IEEE), QIN ZHEN[1], (Member, IEEE), OYETUNJI ELKANAH OLAOSEBIKAN[3], AND ZHIGUANG QIN[1] (Member, IEEE)

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]Department of Computer Science, University for Development Studies, Tamale TL 1350, Ghana
[3]Department of Mechanical Engineering, Lagos State University, Lagos 102101, Nigeria

Corresponding author: Zhiguang Qin (qinzg@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672135, and in part by the Sichuan Science Technology Support Plan Program under Grant 2015GZ0095 and Grant 2016JZ0020.

ABSTRACT Classical Artificial Neural Networks (ANNs) though well exploited in solving classification problems, do not model perfectly the information encoding process in the human brain because ANNs encode information using rate-based coding. However, biological neurons in the brain are known to encode information using temporal coding. In order to mimic the biological method of encoding information, various Spiking Neural Network (SNN) models have been developed. However, some of these models are limited in the number of spikes and do not leverage well on some classification problems. In order to address some of the inherent challenges associated with SNN, a multi-layer learning model for a multi-spiking network is proposed in this paper. The model exploits the temporal coding of spikes and the least-squares method to derive a weight update scheme. It also employs a spike locality concept in order to determine how the synaptic weights are to be adjusted at a particular spike time so as to minimize the learning interference, and thereby, increasing the number of spikes for learning. The performance of the model is evaluated on benchmarked classification datasets. A correlation-based metric is combined with a threshold concept to measure the classification accuracy of the model. The experimental results showed that the proposed model achieved better classification accuracy than some state-of-the-art multi-layer SNN learning models.

INDEX TERMS Multi-spiking neural network, supervised learning, temporal coding.

I. INTRODUCTION

Artificial Neural Network (ANN) is one of the main learning algorithms in machine learning, particularly in supervised learning with its variants such as Spiking Neural Networks (SNN) [1], Deep Neural Network (DNN) [2] with its various sub-divisions, Growing and Pruning Learning algorithm for Deep Neural Networks (GP-DLNN) [3] *et cetera* have been used to obtain state-of-the-art results in various fields of application. The architecture and learning philosophy and mathematical formulation used in this class of algorithm are modeled after biological learning processes that occur in the mammalian brain and has evolved over the years towards biological plausibility following discoveries on information presentation and learning mechanisms in the brain [4]. It is established via experimentation that neurons in the cortical area of the brain encode information using temporal coding (precise timing of spikes), which is said to allow efficient information processing and learning [4]–[8].

ANN learning techniques formulated using temporal coding of information are referred to as Spiking Neural Networks (SNN) and classified as the third generation of ANN [1]. Unlike rate-based learning methods that largely use sigmoid and Radial Based functions, neural activities in this generation of ANN are modelled using more biologically plausible neural models such as the Hodgkin-Huxley (HH) model [9], Integrate-and-Fire (IF) models [4], [10], Izhikerich's model [4], [11], and Spike Response Model (SRM) [4], [10]. Supervised learning in the second generation of ANN using rate-based encoding is well established, however, data presentation in SNN using the temporal coding scheme makes it impossible to directly apply supervised rate-based learning methods to train SNN. As a result and also, taking into consideration the prospects

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Asif.

SNN presents to the machine learning domain, there have been series of studies geared towards formulating supervised learning algorithms specific to SNN using varying synaptic weight optimisation techniques such as Spike Timing Dependent Plasticity (STDP) [12]–[14], Error Backpropagation [15], [16], evolutionary optimisation techniques [17], [18], among others, which have been successfully evaluated on some benchmark classification problems. However, a major challenge with these learning techniques is the uncertainty of convergence when the number of spikes emitted by neurons in a network particularly, in the output neurons are more than one (1). It has been demonstrated that multi-layered networks with multi-spiking neurons are required to adequately learn and classify non-linearly separable data [19]. The challenges associated with using the above SNN learning models to efficiently train multi-layer multi-spiking networks are attributed to the difficulty in defining appropriate cost functions for multiple output spikes in the case of Error Backpropagation and evolutionary based models, and the increase in learning interference on synaptic weights due to changes imposed by synapses with multiple input spikes that contribute to different output spikes [12], [19]. In an attempt to minimize the challenges associated with existing supervised SNN learning models and improve the performance of SNN reported in literature particularly, on data classification tasks, a supervised learning model for multi-layer networks with multi-spiking neurons is presented in this paper. The aim of supervised learning in SNN is to train the network to emit output spikes at some defined spike times in response to input spikes arriving at the synapses of the output neuron before a respective output spike time. To achieve this, techniques are formulated mostly, using the difference between the output and input spike times to adjust the synaptic weights with respect to input spike times to enable the output neuron to emit a spike at a given output spike time [4]. The proposed model is derived based on a concept that exploits the relationship between input and output spikes times and synaptic weights that exist in a neural model [20]. In a supervised SNN setup, the input and desired output spike times are mostly known, but the exact values of synaptic weights required by the network to emit spikes at the desired output spike times are normally not known, hence, are often initialized using random processes and trained to obtain the required values. Knowing the input and desired output spike times together with the initial synaptic weights, their relationship defined by a synaptic model as illustrated in [20], is exploited to establish a system of equations from which the change in synaptic weights required by an output neuron to emit a spike at a given time in response to a set of input spikes preceding it is approximated using the least-squares method. Our motivation is in deriving a direct approach for approximating change in synaptic weights in the course of learning without the need to define and minimize cost functions using output spike times which increases the computational complexity and degrades networks performance as in the case of error back propagation methods when the number of desired spikes increases [15], [16]. Training a multi-layer SNN using the proposed model require a simultaneous update of hidden and output synaptic weights. To achieve this a biofeedback signal [14], is sent back to the hidden neurons at every output spike time to guide the weight update processes in the hidden layers. To minimize the learning interference that is introduced by the continuous update of same synaptic weights at different spike times, a spike locality concept is introduced in the proposed learning model, in that, only the weights of synapses that have spikes directly contributing to an output spike at a given time are updated. In all our proposed multi-spiking learning model for multi-layer SNN produced better classification performance than most existing SNN learning models. The contributions of this paper are as follows:

- A novel weight update scheme for training a multi-layer SNN is derived and applied in solving data classification problem.
- A multi-spiking learning model for training multi-layer SNN that employs a spike time locality concept thereby minimizing learning interference is presented.
- The performance of our proposed model on data classification is assessed on seven (7) benchmarked datasets and the model showed a significant and better classification accuracy than some state-of-the-art SNN learning models.

The remaining sections of the paper are arranged as follows: A review of related works is presented in section II, followed by the research methodology in Section III, results and discussion in Section IV, and conclusion in Section V.

The various notations used in the paper are shown in Table1.

TABLE 1. Notations used in the paper.

SYMBOL	DESCRIPTION
I	Number of neurons in input layer
H	Number of neurons in hidden layer
J	Number of neurons in output layer
i	Index of input neurons
h	Index of hidden neurons
j	Index of output neurons
t_i	Spike time of input neuron i
t_h	Spike time of hidden neuron h
t_j	Spike time of output neuron j
t_j^f	The f^{th} spike of output neuron j
t_d	Desired output spike time
t_d^n	Desired output spike train
z_i, z_h, z_d	z-domain transforms of t_i, t_h, and t_d
w_{hi}, w_{jh}	Synaptic weights connecting i to h and h to j
$\Delta w_{hi}, \Delta w_{jh}$	Weight change associated with w_{hi} and w_{jh}
$\Delta z_i, \Delta z_h$	Denotes $\frac{z_d}{z_i}$ and $\frac{z_d}{z_h}$
η	Learning rate
C	Correlation-based metric
C_{th}	A threshold value based on C
C_c	Pairwise interclass correlation
C_d	Classification determinant based on C
N_p	Number of possible class-label pairs
A_t^e	Training Accuracy at epoch e
PSP	Postsynaptic Potential

II. RELATED WORK

A review of some related supervised learning models for SNN is presented in this section.

A. SINGLE LAYER LEARNING MODELS

The Remote Supervised Method (ReSuMe) [21], is one of the first biologically plausible learning rules derived for single-layer networks in which neurons in the network can emit multiple spikes. The learning rule is derived using STDP (Hebbian) and anti-STDP (anti-Hebbian) processes [22].

This rule was derived to overcome some key functional limitation of the first supervised learning rule, the SpikeProp [15]. These include the minimization of the error between the target and actual spikes without the need for gradient calculations, which eliminated the difficulty in defining appropriate cost functions and the computational requirements of gradient descent-based methods. Unlike the SpikeProp, the ReSuMe is neural model-independent and neurons in the network could fire multiple spikes, thus, making it suitable for spike sequence learning. However, with the ReSuMe, convergence is not guaranteed when the number of spikes in the output spike train are more than one (1). Following the uncertainty regarding convergence of the ReSuMe with respect to multiple spikes in an output spike train, [13], [23], modified it to include delay learning, which they called Delay Learning-ReSuMe (DL-ReSuMe) and Extended Delay Learning-ReSuMe (EDL-ReSuMe) respectively. Their methods led to a significant improvement in both the accuracy and time requirements of ReSuMe. DL and EDL-ReSuMe, like the tradition ReSuMe, are single-layer learning rules and also suffered a significant loss in performance when the learning period, T is extended beyond 500ms with increased spike firing rates. This implies that there is still the need for more robust learning rules for tasks with extended learning periods and firing rates. Several other algorithms have also been developed to simulate, basically spiking neural networks with the same architecture used in the ReSuMe. An example of which is Spike Pattern Association Neuron [24]. Though these rules are proposed to solve similar problems as the ReSuMe, they are either deficient in biological plausibility or had a sub-optimal overall performance as compared to the ReSuMe. Though the ReSuMe and its variants have been successfully applied to a variety of tasks, their efficient application to classification tasks is hindered by the number of network layers they are defined for, particularly on non-linearly separable datasets. It has been asserted that single-layer networks are unable to efficiently handle complex classification tasks [16], [19].

B. MULTI-LAYER SINGLE SPIKING LEARNING MODELS

The first successful supervised learning rule for spiking neural network called the SpikeProp was introduced less than two (2) decades ago by [15]. It was derived for multi-layer networks with single spiking neurons mainly for data classification tasks using Error Back Propagation similar to what is used in second-generation ANN. Following the successful introduction of the SpikeProp algorithm, several variants of it such as; BP with momentum [25], QuickProp [26], Resilient Propagation [26] and other methods were proposed as modifications of the original SpikeProp algorithm with the core aim of improving the convergence rate and classification accuracy. More recently, [20] proposed a multi-layer single spike learning model in which they mapped the neuron model parameters into the $z-domain$ with respect to the relationship between input spike times and the time of the first output spike. They defined and minimized a cost function based on this relationship. Another set of multi-layer single spiking neural network learning models are those that employed evolutionary methods. These studies come in two folds: Those that used evolutionary algorithms for feature and parameter optimization in evolving Spiking Neural Network (eSNN) [24], [27]–[29] and those that used evolutionary methods to derive Spiking Neural Network learning rules [17], [30]. The learning principles of eSNN is derived from the classical Evolving Connectionist Systems used with sigmoidal neural networks. In eSNN, spiking neural model (the earliest works used Thorpes model [31]) is used instead of the sigmoid model. Evolutionary algorithms were later introduced into the eSNN learning paradigm for feature selection and network parameter optimisation [24], [28], [32], [33], in that, in each evolution a sub-feature space is selected and used to train an eSNN and the feature space, network parameters and performance are adapted by the evolutionary algorithm. This process is repeated in each evolution and the appropriate search operators applied according to the principles of the chosen evolutionary algorithm. Contrary to the above approach of learning using evolutionary models, the methods introduced by [17], [30], used evolutionary methods to optimize network synaptic weights and delays by minimizing the error between desired and actual network output spikes. Reference [17], made use of Differential Evolution, which is a novel minimization technique that can solve non-differentiable, nonlinear, and multi-modal objective functions. They tested their approach on three (3) benchmarked classification problems; The XOR problem, Diabetes and IRIS datasets. The method proposed by [30], on the other hand optimized both synaptic weights and delays by minimizing an error function using an evolutionary method. They asserted that their choice of an evolutionary method is due to their ability to process real numbers without the need to convert them into binary form. They successfully tested their proposed scheme using the benchmarked XOR problem and IRIS dataset and reported impressive classification performance. Though these single-spike multi-layer learning models presented in previous studies produced competitive results as compared to sigmoidal networks in classification tasks, those that are based on Error Back Propagation and evolutionary methods are computationally expensive and also, are considered non-biologically plausible. The fact that neurons in the networks could fire only single spikes is a limitation due to the limited amount of information single spikes can convey [21].

C. MULTI-LAYER MULTI-SPIKING LEARNING MODELS

Due to the limitations of single spike networks, there has been extensive research into extending some of these learning rules into multi-spiking rules as well as the proposition of new multi-spiking learning rules mainly for data classification. This is necessary because multi-spiking neurons are capable of learning any non-linearly separable data [19], which is a characteristic desired in all learning algorithms.

As an extension into a multi-layer multi-spiking learning rule, [34] and [35] respectively, modified the SpikeProp rule to allow neurons in the input and hidden layers to fire multiple spikes. However, neurons in the output layer could only fire single spikes due to the difficulty associated with defining appropriate error functions for multiple spikes at the output neuron.

The authors in [16] proposed the first learning rule that allowed neurons in all layers of a multi-layer network to fire multiple spikes. The learning rule was derived using the error backpropagation method. This rule, like other error backpropagation based methods, is difficult to train particularly when the number of spikes in the desired spike train increases.

The authors in [19] proposed the first biologically plausible multi-layer network learning rule in which all neurons could fire multiple spikes. They combined the error function definition used in gradient descent to the learning process used in ReSuMe to form the new learning rule. Though this learning rule is more biologically plausible than the rule presented in [16]. It also recorded a drop in performance when the number of spikes in the desired spike train increases.

More recently, [14], presented another biologically plausible multi-layer multi-spiking learning rule. They adopted and modified the ReSuMe learning method to enable all neurons in all layers of the network to fire multiple spikes. Unlike the rule represented in [19], they considered spikes fired by neurons in the hidden layer when training hidden layer weights. Similar to the multi-spiking rules proposed earlier, they assessed the classification performance of the rule on some benchmark datasets and the results recorded were comparable to some sigmoidal methods.

III. METHODOLOGY

This section presents a description of the proposed learning scheme and the tools and datasets used for simulation.

A. PROPOSED LEARNING SCHEME

The multi-layer learning model presented in this paper is built-on the weight update scheme for single-layer networks derived using the least-squares approach presented in [36]. While the single-layer learning model was ideal for spike sequence learning, the proposed multi-layer learning model is suitable for data classification.

For completeness, an overview of the weight update scheme is presented below. The main aim of the proposed weight update scheme is to establish a system of equations in terms of an output spike time, t_d; given a set of input spike times $t_{i=\{1,2,3,\ldots,I\}}$, where I is the number of input neurons with spikes within the time locality of t_d; and the corresponding synaptic weights, w_{ji} connecting to a given output neuron, j, ($j = \{1, 2, \ldots J\}$ where J is the number of output neurons in the network). For j to fire a spike at desired time, t_d, Δw_{ji} in (1), which is the expected weight change required to induce the postsynaptic potential of j to reach the threshold value, ϑ, at t_d, have to be determined.

$$z_d = \frac{\sum_{i=1:I}(w_{ji} + \Delta w_{ji})z_i}{\sum_{i=1:I}(w_{ji} + \Delta w_{ji}) - 1} \tag{1}$$

where $z_d = exp(t_d)$ and $z_i = exp(t_i)$; thus $t_d = \ln(z_d)$ and $t_i = \ln(z_i)$

Differentiating (1) with respect to z_i yields (2). The differential of z_d with respect to any $z_{i=1\ldots I}$ are obtained in the same form as (2).

$$\frac{dz_d}{dz_i} = \frac{(w_{ji} + \Delta w_{ji})}{\sum_{i=1:I}(w_{ji} + \Delta w_{ji}) - 1} \tag{2}$$

In order to transform (2) into a form from which the system of equations can be obtained with respect to each input spike time z_i, (2) is integrated to obtain (3).

$$z_d = \frac{(w_{ji} + \Delta w_{ji})z_i}{\sum_{i\epsilon I}(w_{ji} + \Delta w_{ji}) - 1} + K \tag{3}$$

where K is the integral constant and is assumed to be zero ($K = 0$) for the purpose of simulation. Zero (0) is one of the possible values in the real number space ($\mathbb{R}$) K can assume.

For simplicity, let's assume the network has only one output neuron ($j = 1$) in which case, w_i and Δw_i are used in place of w_{ji} and Δw_{ji} respectively, in subsequence sections of the paper. Thus (3) can be expressed as (4)

$$\frac{z_d}{z_i} = \frac{(w_i + \Delta w_i)}{\sum_{i\epsilon I}(w_i + \Delta w_i) - 1} \tag{4}$$

Denoting the left hand side of (4) as Δz_i (i.e. $\frac{z_d}{z_i} = \Delta z_i$) and expression (4) in terms of Δw_i yields (5).

$$\Delta z_i \Delta w_1 + \Delta z_i \Delta w_2 + \cdots + \Delta z_i \Delta w_I - \Delta w_i = w_i - \Delta z_i \left(\sum_{i\epsilon I} w_i - 1 \right) \tag{5}$$

Obtaining in the form of (5) for all input spikes, $t_{i=\{1,2,3,\ldots,I\}}$, arriving at the synapse of the output neuron before t_d, yields the system of equations presented in (6), which can be expressed in a matrix form as (7) and solved using the least-squares method as shown in (8).

$$\begin{aligned}
&(\Delta z_1 - 1)\,\Delta w_1 + \Delta z_1 \Delta w_2 + \cdots + \Delta z_1 \Delta w_I \\
&= w_1 - \Delta z_1 \left(\sum_{i=1:I} w_i - 1 \right) \\
&\Delta z_2 \Delta w_1 + (\Delta z_2 - 1)\,\Delta w_2 + \cdots + \Delta z_2 \Delta w_I \\
&= w_2 - \Delta z_2 \left(\sum_{i=1:I} w_i - 1 \right) \\
&\vdots \qquad\qquad \ddots \qquad\qquad \vdots
\end{aligned}$$

$$\Delta z_I \Delta w_1 + \Delta z_I \Delta w_2 + \cdots + (\Delta z_I - 1)\,\Delta w_I = w_I - \Delta z_I \left(\sum_{i=1:I} w_i - 1 \right) \tag{6}$$

$$\begin{bmatrix} (\Delta z_1 - 1) & \Delta z_1 & \dots & \Delta z_1 \\ \Delta z_2 & (\Delta z_2 - 1) & \dots & \Delta z_2 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta z_I & \Delta z_I & \dots & (\Delta z_I - 1) \end{bmatrix} \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_I \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_I \end{bmatrix} \tag{7}$$

where $c_i = w_i - \Delta z_i \left(\sum_{i=1:I} w_i - 1 \right)$.

$$\Delta w = \Delta Z^{+} C \tag{8}$$

where

$$\Delta Z^{+} = \begin{bmatrix} (\Delta z_1 - 1) & \Delta z_1 & \dots & \Delta z_1 \\ \Delta z_2 & (\Delta z_2 - 1) & \dots & \Delta z_2 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta z_I & \Delta z_I & \dots & (\Delta z_I - 1) \end{bmatrix}^{+},$$

is the Moore-Penrose pseudoinverse of the matrix

$$\Delta Z,\ \Delta w = \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_I \end{bmatrix} \text{ and } C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_I \end{bmatrix}.$$

Thus, weights of input neurons that have spikes contributing to an output spike time will be adjusted according to (9).

$$w_i = w_i \pm \eta \Delta w_i \tag{9}$$

where η is a learning rate that controls the magnitude of adjustment made to the weights at a given spike time in the course of learning. That is, the weights are increased at a desired output spike time, t_d or decreased when the output neuron fires a spike at an undesired spike time. The reduction of weights at an undesired spike times decreases the PSP of the neuron which leads a cancellation of the undesired spike.

Equation (9) can be expressed in a general form as (10), taken into consideration other values of j.

$$w_{ji} = w_{ji} \pm \eta \Delta w_{ji} \tag{10}$$

To adopt this weight modification scheme to train a multi-layer network, two key training activities are performed simultaneously at every event time; training of the output neuron(s) and hidden neurons. The training process of multi-layer networks using the proposed weight update scheme is presented in the proceeding sections.

1) WEIGHT UPDATE IN OUTPUT NEURONS

Output neurons in a multi-layer network receive their input spikes from neurons in a preceding hidden layer. The timing of spikes fired by these hidden neurons at the beginning of training is often dynamic and becomes relatively stable as their synaptic weights converge near their optimal values. Since the output neurons receive input from hidden layer neurons, (10) can be redefined as (11) to cater for spikes from hidden neurons.

$$w_{jh} = w_{jh} \pm \eta \Delta w_{jh} \tag{11}$$

where w_{jh} are synaptic weights of hidden neurons $h = \{1, 2, 3, \dots, H\}$ to output neuron j and Δw_{jh} is a vector of the computed weight change required to enable output neuron j fire a spike in response to the hidden spikes or cancel an undesired spike at event time t_j.

In response to input spikes, an output neuron may fire one (1) or several spikes at time(s) t_j, within a time course, T. These output spikes may occur at two distinct event times; at desired time(s), t_d, and undesired time(s), t_u, thus, $t_j = \{t_d \cup t_u\}$. However, it is desired that for a given set of input spikes, $t_j \approx t_d$. To achieve this, at a desired output spike time, t_d where there is no output spike, the quantum of weight change required by all hidden synapses that have spikes within the locality of t_d is approximated using (8) and added to the respective weights using (11). The parameters in (8) are redefined as follows:

$$\Delta w = \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_H \end{bmatrix},$$

$$\Delta Z = \begin{bmatrix} (\Delta z_1 - 1) & \Delta z_1 & \dots & \Delta z_1 \\ \Delta z_2 & (\Delta z_2 - 1) & \dots & \Delta z_2 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta z_H & \Delta z_H & \dots & (\Delta z_H - 1) \end{bmatrix}, \text{ and}$$

$$C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_H \end{bmatrix}.$$

where $\Delta w_{h,\ h=\{1,2,\dots,H\}}$ are the expected weight change associated with hidden neurons that have spikes within the locality of t_d, $\Delta z_{h,\ h=\{1,2,\dots,H\}}$ denotes $\frac{z_d}{z_h}$. z_d and z_h are $z - domain$ transforms of output spike time t_d and hidden spike time, t_h, respectively, and is computed as: $\frac{z_d}{z_h} = exp(t_d - t_h)$, this implies that, $\Delta z_h = exp(t_d - t_h)$.

The locality of an output spike time mentioned above is defined as the inter-spike time between the current output spike time, t_j^f (desired or undesired) and a preceding spike time, t_j^{f-1} (desired or undesired). To update synaptic weights at the current spike time, only hidden neurons that have spikes within this local time frame ($t_j^{f-1} \rightarrow t_j^f$), will have their weights updated. At desired spike times where there are spikes, t_j^{f-1} is set to t_j^f. This ensures that the weights are not modified once there is already a spike at t_d.

2) WEIGHT UPDATE IN HIDDEN NEURONS

Hidden neurons are trained for two (2) main purposes; first to fire multiple spikes within the time locality of a desired output spike, and secondly to cancel hidden spikes that contribute

to the occurrence of undesired spikes in output neurons. To achieve this, (10) is modified into (12) in which the target spikes are hidden spikes, t_h in the respective hidden layer and spikes from neurons in the preceding (in this work the first) layer serve as input spikes.

$$w_{hi} = w_{hi} \pm \eta \Delta w_{hi} \tag{12}$$

where w_{hi} is the synaptic weight of input neuron, i connecting to hidden neuron, h, and Δw_{hi} is computed using (13) similar to (8).

$$\Delta w_{hi} = \Delta Z_{hi}^{+} C_{hi} \tag{13}$$

Also, similar to (6), Δz_i is computed as $\frac{z_h}{z_i} = exp(t_h - t_i)$ and every c_i in the column vector C is computed as $c_i = w_{hi} - \Delta z_{hi} \left(\sum_{i=1:I} w_{hi} - 1\right)$.

In the course of training, when an output spike time, t_j^f, is encountered, a biofeedback signal [14], is sent to the hidden neurons with the time locality of t_j^f. If $t_j^f = t_d$ and there is no spike at t_d, then, for a given $h \in \{1, 2, 3, \ldots, H\}$, if there is any $t_{hi} \in \{t_j^{f-1}, t_d\}$ and h is excitatory, the weights w_{hi} for $t_{hi} \in \{t_j^{f-1}, t_d\}$ are increased using (12). This increment in synaptic weights will enhance the Postsynaptic Potential (PSP) of h, causing it to emit spikes within the respective spike time locality, which in effect will induce an output spike at t_d. However, if h is inhibitory and has spikes within $\{t_j^{f-1}, t_d\}$, its spikes are canceled by reducing the weights of input neurons that contribute to the spikes. The cancellation of hidden spikes from inhibitory neurons is meant to minimize the retarding effect they place on the PSP of output neurons [14].

On the other hand, if an undesired spike occurs at an output spike time, t_j^f, all excitatory hidden neurons that have spikes within the undesired spike locality, have the weights of the respective input neurons reduced using (12). Whilst inhibitory hidden neurons that receive input spikes within the locality of the given output spike time are trained to fire multiple spikes within the time-space. The cancellation of excitatory hidden spikes and increase in inhibitory hidden spikes cumulatively ensure a faster reduction in the total PSP of the output neuron which leads to the cancellation of undesired spikes. Also, all hidden neurons are trained to emit at most one(1) spike within a $5ms$ interval.

B. EXPERIMENTAL SETUP

1) DATASETS

The performance of the proposed model is assessed using seven (7) benchmarked classification datasets, which include: The Wisconsin Diagnostic Breast Cancer (WDBC) dataset [37], BUPA liver disorders (BUPA) dataset, Johns Hopkins University Ionosphere dataset; all available at the UCI machine learning repository [38], and the Pima Indian Diabetes (Pima) Dataset [39]. Which are all binary class data sets. The rest are multi-class datasets, which include, the Fisher IRIS (3 class), Vehicle (4 class), and the Page-Blocks (5 class) datasets, also available at the UCI repository [38]. These datasets are considered because they present different dynamics to learning and have been used to test some existing SNN learning models.

2) NETWORK SETUP AND LEARNING PARAMETERS

The performance of the proposed model is tested using a two (2) layer network comprising of $M \times A + b$ input neurons, 120 hidden neurons, and c_l^n output neurons. M is the population of neurons that each attribute in a dataset is encoded with, A is the number of attributes in a dataset, and b is the number of bias input neurons. In this paper, two bias neurons with spikes at $t_d^f - dt$ are used. The significance of adding bias neurons is explained in [15]. c_l^n is the number of class labels in a dataset.

The Gaussian Receptive Fields (GRF) based population temporal coding scheme proposed by [15] and used by [14], [34], [40], is used in this paper. Each attribute, a, with a range, $\{a_{min}^n, \ldots, a_{max}^n\}$ is encoded with $M = 10$ identical GRFs and the adjustment factor, γ is set to 1.5 similar to what is used in previous studies [14]–[16]. The corresponding input values are multiplied by 100 to produce spike times between $0ms$ and $100ms$, and neurons with spike times of $90ms$ or above are coded not to fire.

Output neuron(s) are coded to fire at defined spike times within the interval of $5ms - 100ms$. The number of spikes generated between this interval depends on the number of desired output spikes per class and the number of class labels in a dataset. Equal sets of the generated output spike times are randomly assigned to each output neuron (class label). The minimum inter-spike interval in output spike times is set to $5ms$, mainly to minimize learning interference.

Performance Metrics and Training Procedure: The correlation-based metric, C, proposed by [12], and discussed in an earlier study [36], is used to measure the correlation between a neuron's actual output spikes and desired output spike train. In a training epoch the values of C at each output neuron, for an input instance, n, after using it to train the network is calculated and the class to which it is classified under is determined as follows:

A threshold value, C_{th} is set and the average pairwise interclass correlation, C_c between the desired output spike trains of the various classes is calculated using (14)

$$C_c = \frac{\sum_{l=l:L} \sum C_{(l,l-r)}}{N_p} \tag{14}$$

where L is the number of class labels, $r \in \{1, 2, \ldots, l-1\}$, and N_p is the number of possible pairs.

The highest C amongst all output neurons for an instance, n is denoted as C_n^h. The second highest C is compared to C_c and the greatest amongst them is added to C_{th} to form a determinant, C_d. For instance, n is classified as belonging to class label, c_l that produced C_n^h if and only if $C_n^h \geq C_d$, else, n is rejected and considered not to be well learned by the network. After each training epoch, e, the training accuracy, A_t^e, is computed and training is continued until $A_t^e = 100\%$ or $e = 100$ is attained.

In the testing phase, the network's weight update function is switched-off and each instance is presented to the network. The process used to classify training instances outlined above is followed to determine the predicted class of test instances.

For each dataset, 40 trails of training and testing are performed, each time with a random selection of training and testing instances, synaptic weight initialization, and output spike trains. The average training and testing accuracies over the 40 trails are computed and used as the final performance of the network.

50% of the dataset is randomly selected as the training set and the remaining 50% used as the testing set in conformity with most previous studies for comparison purposes, except in the page-blocks dataset, where 50% of classes with less than 150 instances were randomly selected as the training set and 70 instances were also randomly selected from classes with more than 150 instances with the remaining instances belonging to the testing set. This selection criteria in the Page-Blocks dataset is due to its imbalance nature.

IV. RESULTS AND DISCUSSION

This section presents the findings obtained from the experiment using both graphical and tabular methods. It further explains the meaning and implications of the findings.

A. EFFECTS OF NUMBER OF DESIRED SPIKES AND CLASS LABELS ON PROPOSED METHOD

To assess the influence the number of desired output spikes and class labels have on the classification performance of the proposed model, the mean training and testing accuracies on seven benchmarked datasets trained with desired output spike train size, t_d^n of 1, 2, 3, 4, and 5 per class label are presented in this section. The number of output neurons used to train each network is equal to the number of class labels in a respective dataset.

The mean classification accuracy of the proposed model on the 7 datasets are shown in Fig. 1 and Fig. 2. On the WDBC dataset, as shown in Fig. 1(a), a mean training and testing accuracy of 97.43% and 95.10%, respectively, are recorded when t_d^n=1. When t_d^n is increased from 1 to 2, 3, 4, and 5, the mean training accuracy increased to 100% for all four spike train sizes with a corresponding increase in the testing accuracy from 95.10% to 98.55%, 99.59%, 99.91%, and 99.98% respectively.

The mean training and testing accuracies for the Pima and BUPA datasets shown in Fig. 1 (b) and (c) respectively, follow a similar trend as that of the WDBC dataset. In which, a 100% mean training accuracy is obtained when t_d^n is set between 2 and 5 for both datasets, and 99.39% and 99.04% respectively when set to 1. Similarly, the testing accuracies increased from

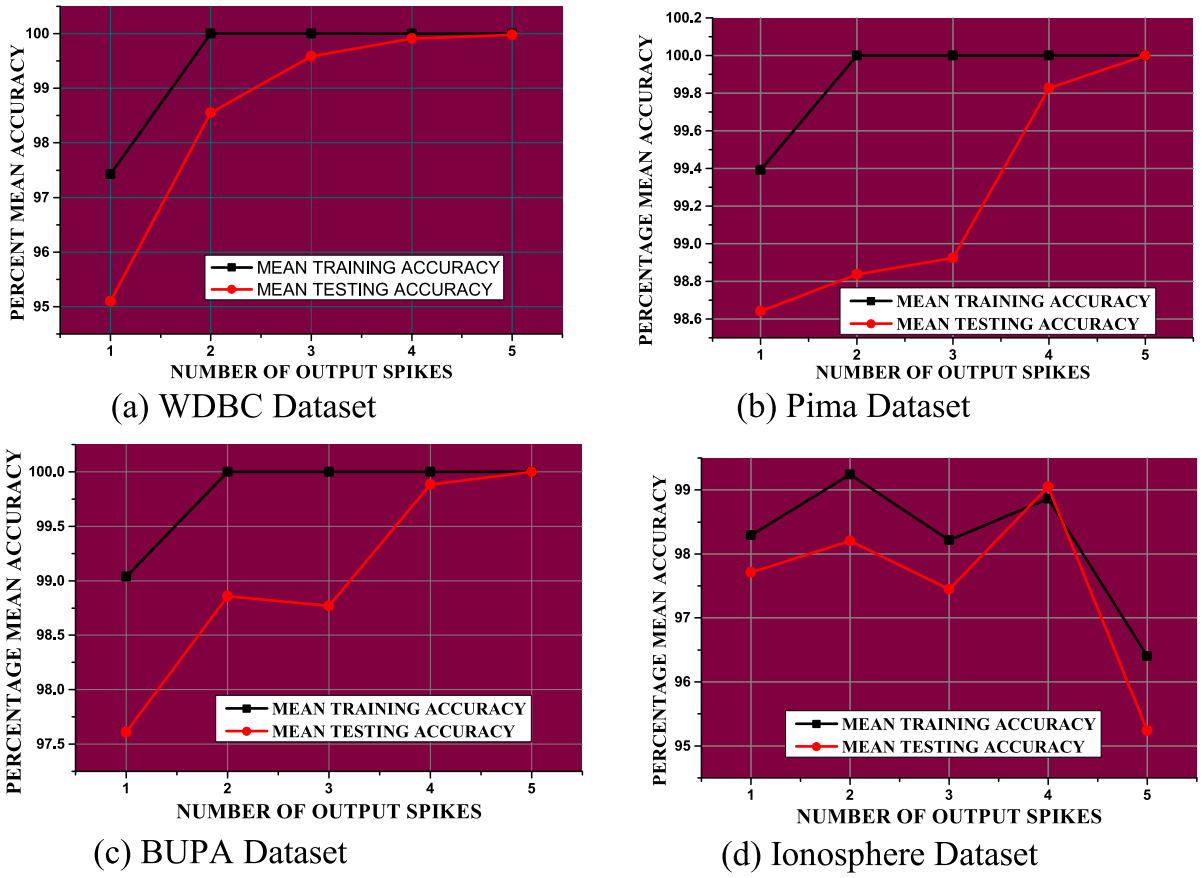

FIGURE 1. The Training and Testing Mean Accuracies for the IRIS, Vehicle and Page-Blocks Datasets (All are 2 Class Datasets).

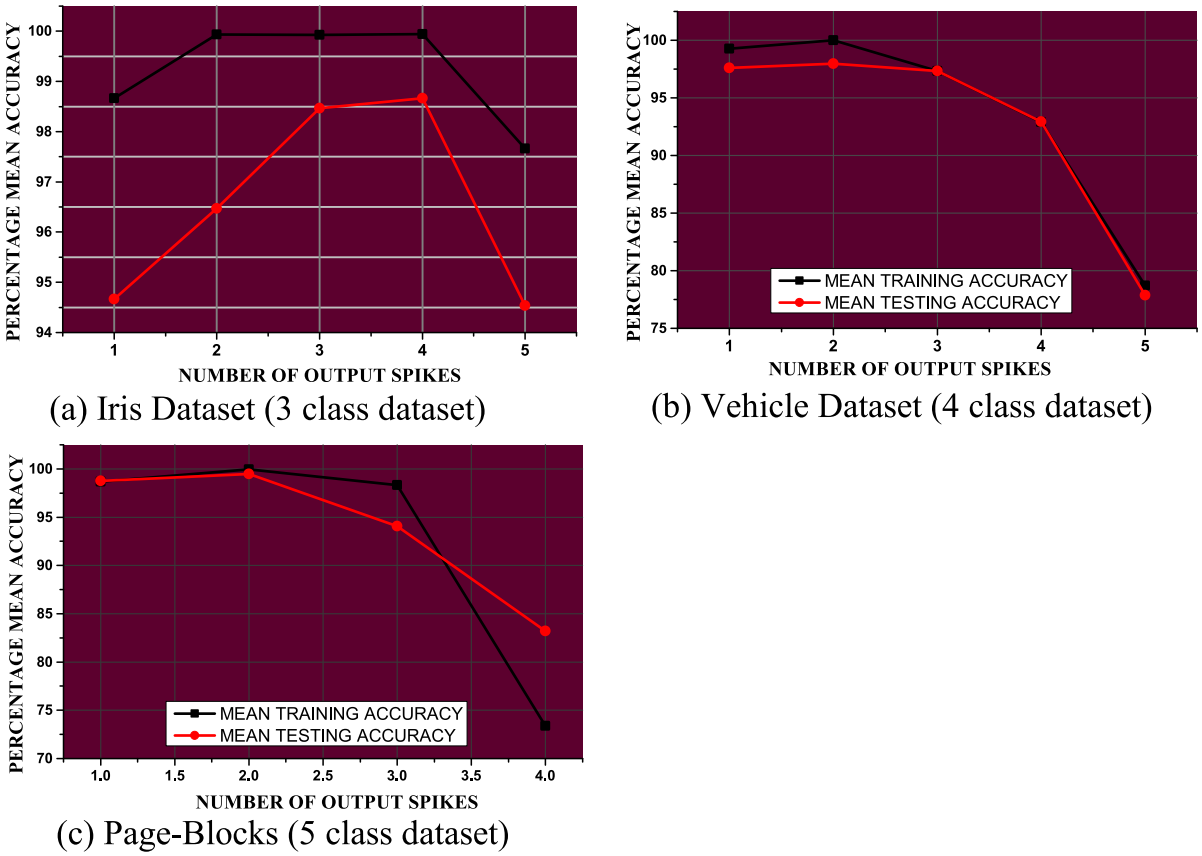

FIGURE 2. The Training and Testing Mean Accuracies for the WDBC, Pima, BUPA and Ionosphere Datasets.

98.64% to 100% for the Pima dataset and 97.61% to 100% for the BUPA dataset when t_d^n is increased from 1 to 5.

However, the dynamics of the classification performance on the Ionosphere dataset shown in Fig. 1(d) is relatively different as both the mean training and testing accuracies fluctuate as t_d^n increases from 1 to 5. The highest mean training and testing accuracy of 98.86% and 99.05% respectively, are obtained when t_d^n of 4 is used for training. Unlike the other binary datasets, the least training and testing mean accuracies of 96.40% and 95.24% are obtained at $t_d^n = 5$.

The mean classification accuracy of the proposed model on the multi-class datasets is shown in Fig. 2. The highest mean testing accuracy on the IRIS dataset is obtained when the number of spikes in t_d^n is set to 4. It, however, dropped sharply to the lowest, 94.53% when t_d^n is increased to 5 contrary to results obtained for the binary datasets.

On the Vehicle and Page-Blocks datasets, the highest mean testing accuracies of 97.98% and 99.50%, respectively, are obtained when t_d^n is set to 2 as shown in Fig. 2 (b) and (c), respectively. The accuracy, however, dropped steadily to the lowest, 77.88% and 83.20%, when t_d^n is set to 5 and 4 respectively, for the Vehicle and Page-Blocks datasets. This drop-in mean accuracy is attributable to the increase in learning interference resulting from the combined increase in the size of t_d^n. With the binary datasets, at $t_d^n = 5$, the cumulative size of the output spike train is 10 which increased to 15 in the case of the IRIS dataset and 20 in that of the vehicle dataset resulting from the increase in the number of class labels.

The results presented above points out that the optimal number of output spikes per class label required to achieve maximum classification accuracy depends largely on the number of class labels in a dataset. It is also evident that datasets with a larger number of class labels require a fewer number of spikes for optimal learning. Thus, there is no predefined number of desired spike size for all datasets using the proposed model, though, for a learning period of 100ms, 3 or more spikes are recommended for binary datasets and 2 to 4 for multi-class datasets.

A trend worth noting is the mean training and testing accuracies at $t_d^n = 1$ for all datasets. It is observed that both training and testing accuracies at $t_d^n = 1$ are relatively lower as compared to higher values of t_d^n. Though, in some cases, it is worse when the optimal value of t_d^n is exceeded. These lower accuracies at $t_d^n = 1$ confirms an assertion made in previous studies that maximum classification can not be achieved with single spiking neurons particularly, on non-linearly separable data.

To illustrate how well the proposed model learned to reproduce desired output spikes, the values of the Correlation metric, C for all instances in each class of a dataset highlighting

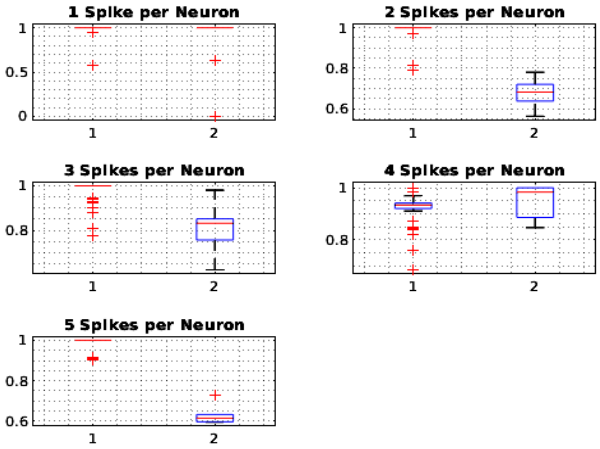

FIGURE 3. WDBC Dataset (2 classes).

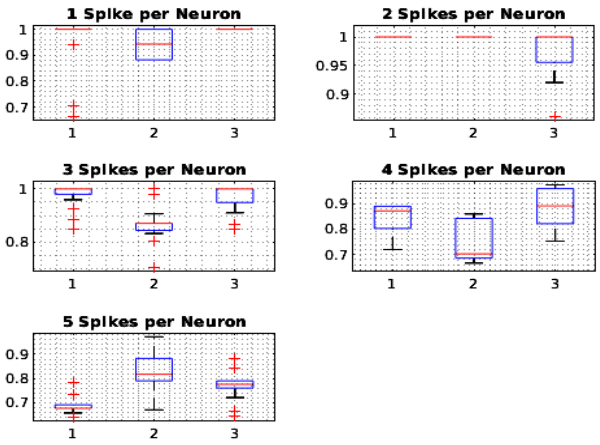

FIGURE 4. IRIS Dataset (3 classes).

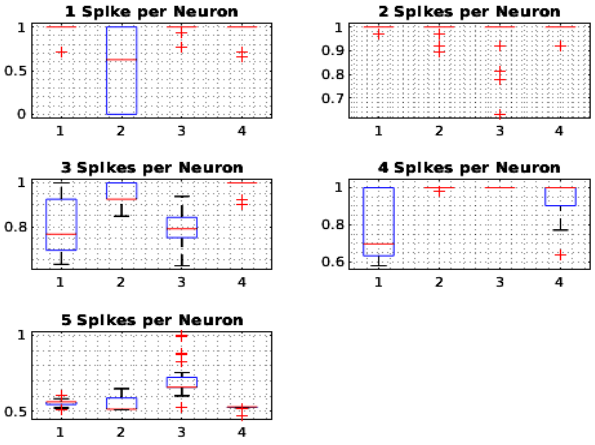

FIGURE 5. Vehicle Dataset (4 classes).

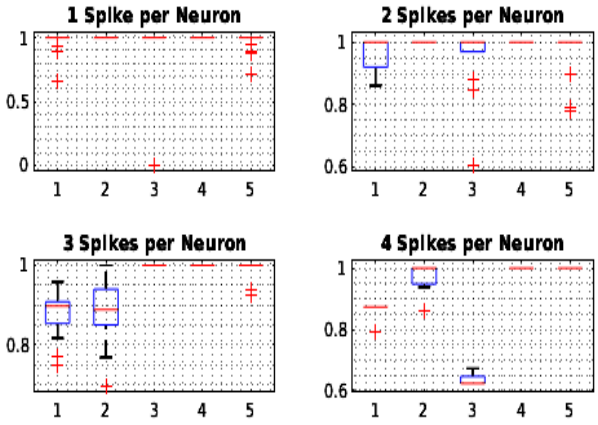

FIGURE 6. Page-Blocks (5 classes).

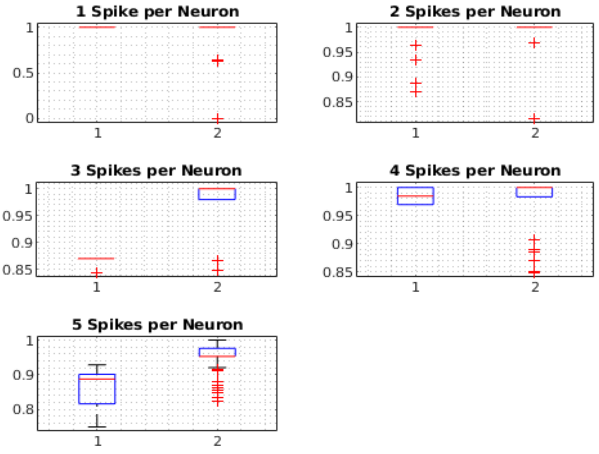

FIGURE 7. BUPA Dataset (2 classes).

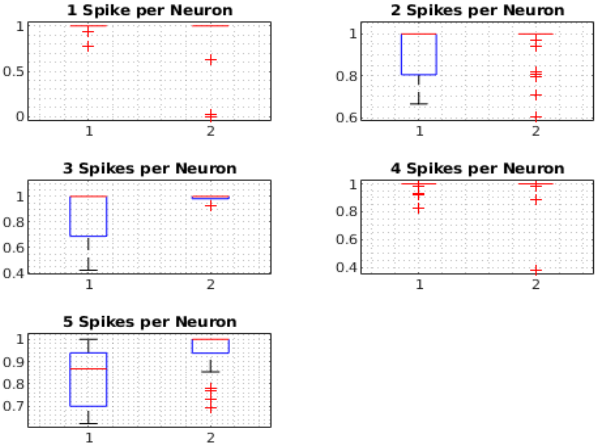

FIGURE 8. Ionosphere Dataset (2 classes).

the median, as well as minimum and maximum values, are shown in Fig. 3 to 9. These values are the training C measures recorded at the point where the training of each network converged. It can be observed from the box plots that for almost all the datasets, a median of 1 is recorded at lower values of t_d^n and drops for some class labels as t_d^n increases. This indicates that the network can learn to emit spikes at the exact desired output spike times for more than 50% of

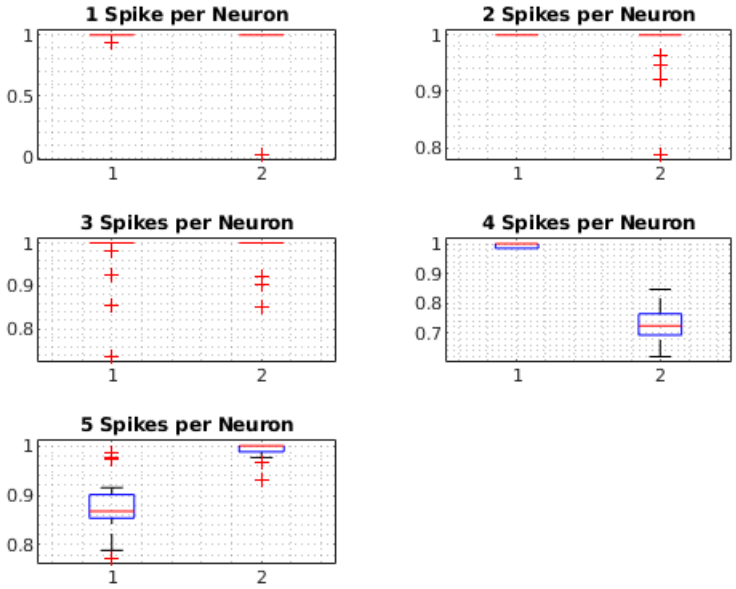

FIGURE 9. Pima Dataset(2 classes).

instances in each class label in the respective datasets. In all the test cases a minimum median of 0.5139 is recorded in the Vehicle dataset when $t_d^n = 5$.

Drawing inference from the median C values and mean testing accuracies at various t_d^n, it is evident that, though the network can learn to emit spikes at the exact desired spike times for lower values of t_d^n, the trained networks are not complex enough to adequately classify instances in the various classes of the testing set as compared to networks trained with relatively higher values of t_d^n.

B. COMPARISON WITH EXISTING LEARNING METHODS

The classification accuracy of the proposed learning model is compared to some existing SNN learning models for data classification. The existing models that we compared our model to include; SpikeProp [15], Multi-Spike(GMES) [16], STDPM [40], MultiSp [14], MuSpiNN [34], SRESN [18]. Similar to the proposed model, all the existing models encode the continuous values in the datasets to spike times using population coding. To ensure a fair comparison, the testing accuracy of the proposed model when $t_d^n = 3$ is used for all comparisons since three (3) output spikes yielded the best results in [14]. Though, the best performance of the proposed model on some datasets are obtained at different values of t_d^n.

The classification performance of the proposed model vis-à-vis some existing SNN learning models on the IRIS dataset is shown in Table 2. The testing accuracy of the proposed model is 98.47% which is comparatively higher than all the other models. Following closely to the proposed model is the performance of SRESN, a single-layer evolving spiking neural network learning model with an accuracy of 97.03%. Of particular interest is the performance of MultiSp, which used a similar network architecture with 3 output spikes per neuron as compared to the proposed model. The accuracy of MultiSp is 95.70%, which is about 2.77% lower than the proposed model. STDPM, an STDP based learning model, however, obtained the lowest accuracy of 83.00% representing about 15.47% lower than the proposed model.

TABLE 2. Classification accuracy on IRIS dataset.

Method	Testing Accuracy (%)
Proposed Method	98.47
SRESN	97.03
SpikeProp	96.10
MuSpiNN	95.83
MultiSp	95.70
Multi-Spike (GMES)	94.72
STDPM	83.00

TABLE 3. Classification accuracy on WDBC dataset.

Method	Testing Accuracy (%)
Proposed Method	99.59
SRESN	97.20
SpikeProp	97.00
Multi-Spike (GMES)	95.32
MultiSp	94.40

The classification accuracy of the learning models on the WDBC dataset is shown in Table 3. The results in this table follow a similar pattern as the IRIS dataset with the proposed model obtaining the highest testing accuracy of 99.59%, followed by SRESN with 97.20% and the SpikeProp with 97.00%. The MultiSp, a relatively more biologically plausible learning model recorded the least accuracy of 94.40%. The classification accuracy of the proposed model, MultiSp, and SRESN on the Ionosphere, BUPA Liver Disorders, and Pima Diabetes datasets are shown in Table 4.

TABLE 4. Classification acuracy on other dataset.

Dataset	Testing Accuracy (%)		
	Proposed Method	MultiSp	SRESN
Ionosphere	97.45	90.50	88.60
BUPA	98.77	61.80	59.70
Pima	98.93	70.60	69.90

The least accuracy recorded by the proposed model is on the Ionosphere dataset, 97.45%, which is better than the 90.50 and 88.60 recorded for MultiSp and SRESN, respectively. A more intriguing performance of the proposed model is on the BUPA Liver Disorders and Pima Diabetes datasets where it recorded performance increase of 36.97% and 39.07% against MultiSp and 28.33% and 29.03% against SRESN respectively.

V. CONCLUSION

A multi-layer learning model based on the least-squares synaptic weight update scheme is proposed and assessed in this paper. The performance of the proposed model on classification tasks is determined using seven (7) benchmarked datasets with different numbers of class labels. The effects of the number of spikes in the output spike train on the proposed model are also presented. The classification accuracy of the model is compared to six exiting spiking neural network learning models and the results showed a significant increase in performance in favor of our proposed model. It is also established that each dataset requires a different number of output spikes to achieve optimal training and testing classification accuracy, and the higher the number of class labels in a dataset the lower the number of output spikes required.

As part of future work, the proposed model will be assessed on different learning periods and datasets with more complexity and compared to rate-based ANN.

REFERENCES

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, Dec. 1997.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, May 2015.
[3] R. Zemouri, N. Omri, F. Fnaiech, N. Zerhouni, and N. Fnaiech, "A new growing pruning deep learning neural network algorithm (GP-DLNN)," *Neural Comput. Appl.*, vol. 31, pp. 1–17, May 2019.
[4] W. Gerstner and M. W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[5] 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.
[6] R. V. Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex," *Neural Comput.*, vol. 13, no. 6, pp. 1255–1283, Jun. 2001.
[7] R. VanRullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends Neurosci.*, vol. 28, no. 1, pp. 1–4, Jan. 2005.
[8] D. A. Butts, C. Weng, J. Jin, C.-I. Yeh, N. A. Lesica, J.-M. Alonso, and G. B. Stanley, "Temporal precision in the neural code and the timescales of natural vision," *Nature*, vol. 449, no. 7158, pp. 92–95, Sep. 2007.
[9] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophys. J.*, vol. 1, no. 6, pp. 445–466, Jul. 1961.
[10] F. Ponulak and A. Kasiński, "Introduction to spiking neural networks: Information processing, learning and applications," *Acta Neurobiol. Experim.*, vol. 71, no. 4, pp. 409–433, 2011.
[11] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.
[12] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, Feb. 2010.
[13] A. Taherkhani, A. Belatreche, Y. Li, and L. Maguire, "Edl: An extended delay learning based remote supervised method for spiking neurons," in *Neural Information Processing* (Lecture Notes in Computer Science), vol. 9490. 2015, pp. 190–197.
[14] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5394–5407, Nov. 2018.
[15] S. M. Bohte and J. N. Kok, "Multilayer RBF Networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 426–435, Mar. 2002.
[16] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Netw.*, vol. 43, pp. 99–113, Jul. 2013.
[17] N. G. Pavlidis, O. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis, "Spiking neural network training using evolutionary algorithms," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Aug. 2005, pp. 2190–2194.
[18] S. Dora, K. Subramanian, S. Suresh, and N. Sundararajan, "Development of a self-regulating evolving spiking neural network for classification problem," *Neurocomputing*, vol. 171, pp. 1216–1229, Jan. 2016.
[19] I. Sporea and A. Grüning, "Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 25, no. 2, pp. 473–509, Feb. 2013.
[20] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2018.
[21] F. Ponulak, "Supervised learning in spiking neural networks with ReSuMe method," doctoral dissertation, Poznán Univ. Technol., Poznań, Poland, 2006. [Online]. Available: http://d1.cie.put.poznan.pl/~fp
[22] D. O. Hebb, "Organization of behavior," *J. Clin. Psychol.*, vol. 6, no. 3, pp. 307–335, 1949.
[23] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "DL-ReSuMe: A delay learning-based remote supervised method for spiking neurons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3137–3149, Dec. 2015.
[24] S. Schliebs and N. Kasabov, "Evolving spiking neural network—A survey," *Evolving Syst.*, vol. 4, no. 2, pp. 87–98, Jun. 2013.
[25] J. Xin and M. J. Embrechts, "Supervised learning with spiking neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 3, no. 3, Jul. 2001, pp. 1772–1777.
[26] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2006, pp. 3970–3977.
[27] G. W. Simei, B. Lubica, and K. Nikola, "Adaptive learning procedure for a network of spiking neurons and visual pattern recognition," in *Advanced Concepts for Intelligent Vision Systems* (Lecture Notes in Computer Science), vol. 4179. Berlin, Germany: Springer, 2006, pp. 1133–1142.
[28] J. L. Lobo, I. Laña, J. Del Ser, M. N. Bilbao, and N. Kasabov, "Evolving spiking neural networks for online learning over drifting data streams," *Neural Netw.*, vol. 108, pp. 1–19, Dec. 2018.
[29] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "A review of adaptive online learning for artificial neural networks," *Artif. Intell. Rev.*, vol. 49, no. 2, pp. 281–299, Feb. 2018.
[30] A. Belatreche, L. P. Maguire, M. Mcginnity, and Q. X. Wu, "Evolutionary design of spiking neural networks," *New Math. Natural Comput.*, vol. 2, no. 3, pp. 237–253, Nov. 2006.
[31] S. Thorpe and J. Gautrais, "Rank order coding," in *Proc. 6th Annu. Conf. Comput. Neurosci., Trends Res.* New York, NY, USA: Plenum Press, 1998, pp. 113–118.
[32] H. N. A. Hamed, N. Kasabov, and S. M. Shamsuddin, "Probabilistic evolving spiking neural network optimization using dynamic quantum-inspired particle swarm optimization," *Austral. J. Intell. Inf. Process. Syst.*, vol. 11, no. 1, pp. 23–28, 2010.
[33] M. Silva, M. M. B. R. Vellasco, and E. Cataldo, "Evolving spiking neural networks for recognition of aged voices," *J. Voice*, vol. 31, no. 1, pp. 24–33, Jan. 2017.
[34] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural Netw.*, vol. 22, no. 10, pp. 1419–1431, Dec. 2009.

[35] O. Booij and H. tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Inf. Process. Lett.*, vol. 95, no. 6, pp. 552–558, Sep. 2005.
[36] A. M. Apambila, O. O. Elkanah, and B. Y. Edward, "A multi-spiking learning method for spike sequence learning," in *Proc. IEEE-AFRICON*, Nov. 2019.
[37] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology.," *Proc. Nat. Acad. Sci. USA*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990.
[38] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[39] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI*, Mar. 2015. [Online]. Available: http://networkrepository.com
[40] A. Sboev, D. Vlasov, R. Rybka, and A. Serenko, "Solving a classification task by spiking neurons with STDP and temporal coding," *Procedia Comput. Sci.*, vol. 123, pp. 494–500, Jan. 2018.

BAAGYERE EDWARD YELLAKUOR received the B.Sc. degree (Hons.) in computer science from the University for Development Studies (UDS), Tamale, Ghana, in 2006, the M.Phil. degree in computer engineering from the Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, in 2011, and the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China, in 2016.

He is currently a Senior Lecturer with the Faculty of Mathematical Science, UDS, and also teaches with the Department of Computer Science. He is also a Postdoctoral Researcher with the School of Information and Software Engineering, University of Electronic Science and Technology of China. He has published more than 50 articles in peer-review journals. His current research interests include machine learning, mobile sensor networks, cryptography, and social networks. He is a member of the International Association of Engineers.

AGEBURE APAMBILA MOSES (Graduate Student Member, IEEE) received the B.Sc. degree in computer science from the University for Development Studies, Tamale, Ghana, in 2008, and the M.Phil. degree in computer engineering from the University of Ghana, Accra, Ghana, in 2014. He is currently pursuing the Ph.D. degree in computational mathematics with the University for Development Studies. After his B.Sc. degree, he worked as a Senior Research Assistant with the Department of Computer Science, University for Development Studies, where he is currently a Lecturer. He has coauthored academic articles published in reputable journals. His research interests include, data mining, machine learning, software engineering, and mobile computing systems.

QIN ZHEN (Member, IEEE) received the B.Sc. degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), in 2005, the M.Sc. degree in electronic engineering from the Queen Mary University of London, in 2007, and the M.Sc. and Ph.D. degrees in communication and information system from UESTC, in 2008 and 2012, respectively. He is currently an Associate Professor with the School of Information and Software Engineering. His current research interests include network measurement, wireless sensor networks, and mobile social networks.

OYETUNJI ELKANAH OLAOSEBIKAN received the B.Sc. degree (Hons.) in electrical engineering from the University of Ilorin, Ilorin, Nigeria, and the M.Sc. and Ph.D. degrees in industrial engineering from the University of Ibadan, Ibadan, Nigeria. He is currently a Professor of industrial and production engineering with the Department of Mechanical Engineering, Faculty of Engineering, Lagos State University, Epe Campus, Lagos, Nigeria. He has attended numerous national and international conferences. His research interest includes production scheduling/operations management (optimization), and algorithm design amongst others. He has published extensively in prestigious local and international journals. He is also a Registered Engineer (COREN) and a member of the following professional bodies, the Nigeria Society of Engineers (NSE), the Nigeria Institute of Industrial Engineering (NIIE), and the South African Institute of Industrial Engineering (SAIIE).

ZHIGUANG QIN (Member, IEEE) is currently a Professor and the Retired Dean of the School of Information and Software Engineering, University of Electronic Science and Technology of China, where he is also the Director of the Key Laboratory of New Computer Application Technology and the UESTC-IBM Technology Centre. His research interests include computer networking, information security, cryptography, information management, intelligent traffic, electronic commerce, distribution, and middleware.