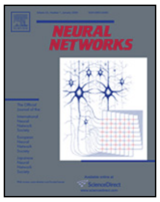
Neural Networks 22 (2009) 1419–1431

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection

Samanwoy Ghosh-Dastidar[a], Hojjat Adeli[a,b,c,d,e,*]

[a] Department of Biomedical Engineering, The Ohio State University, Columbus, OH 43210, United States
[b] Department of Biomedical Informatics, The Ohio State University, Columbus, OH 43210, United States
[c] Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH 43210, United States
[d] Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, United States
[e] Department of Neuroscience, The Ohio State University, Columbus, OH 43210, United States

ARTICLE INFO

Article history:
Received 22 June 2008
Received in revised form 23 March 2009
Accepted 15 April 2009

Keywords:
Spiking neural networks
Epilepsy
EEG classification
Supervised learning

ABSTRACT

A new *Multi-Spiking Neural Network* (*MuSpiNN*) model is presented in which information from one neuron is transmitted to the next in the form of multiple spikes via multiple synapses. A new supervised learning algorithm, dubbed *Multi-SpikeProp*, is developed for training MuSpiNN. The model and learning algorithm employ the heuristic rules and optimum parameter values presented by the authors in a recent paper that improved the efficiency of the original single-spiking Spiking Neural Network (SNN) model by two orders of magnitude. The classification accuracies of MuSpiNN and Multi-SpikeProp are evaluated using three increasingly more complicated problems: the XOR problem, the Fisher iris classification problem, and the epilepsy and seizure detection (EEG classification) problem. It is observed that MuSpiNN learns the XOR problem in twice the number of epochs compared with the single-spiking SNN model but requires only one-fourth the number of synapses. For the iris and EEG classification problems, a modular architecture is employed to reduce each 3-class classification problem to three 2-class classification problems and improve the classification accuracy. For the complicated EEG classification problem a classification accuracy in the range of 90.7%–94.8% was achieved, which is significantly higher than the 82% classification accuracy obtained using the single-spiking SNN with SpikeProp.

1. Introduction

Artificial Neural Networks (ANNs) are simplified mathematical approximations of biological neural networks in terms of structure as well as function. In general, there are two aspects of ANN functioning: (1) the mechanism of information flow starting from the presynaptic neuron to postsynaptic neuron across the network, and (2) the mechanism of learning that dictates the adjustment of measures of synaptic strength to minimize a selected cost or error function (a measure of the difference between the ANN output and the desired output). Research in these areas has resulted in a wide variety of powerful ANNs based on novel formulations of the input space, neuron, type and number of synaptic connections, direction of information flow in the ANN, cost or error function, learning mechanism, output space, and various combinations of these.

Ever since the conception of the McCulloch–Pitt neuron in the early 1940s and the perceptron in the late 1950s (Adeli & Hung, 1995), ANNs have been evolving towards more powerful models. Advancements in the understanding of biological networks and their modes of information processing has led to the development of networks such as feedforward neural networks (Rumelhart, Hinton, & Williams, 1986), counter-propagation neural networks (Adeli & Park, 1995; Dharia & Adeli, 2003; Grossberg, 1982; Hecht-Nielsen, 1988; Sirca & Adeli, 2001), radial basis function neural networks (Karim & Adeli, 2002, 2003; Liu, Wang, & Qiang, 2007; Mayorga & Carrera, 2007; Pedrycz, Rai, & Zurada, 2008), recurrent networks (Hopfield, 1982; Panakkat & Adeli, 2007; Schaefer & Zimmermann, 2007; Zhang, Xu, & Li, 2007), self-organizing maps (Carpenter & Grossberg, 1987; Kohonen, 1982), modular neural networks, fuzzy neural networks (Adeli & Jiang, 2003, 2006; Adeli & Karim, 2000; Jiang & Adeli, 2008; Rigatos, 2008; Sabourin, Madani, & Bruneau, 2007), and spiking neural networks (Grossberg & Versace, 2008; Iglesias & Villa, 2008; Maass, 1996, 1997a; Sejnowski, 1986). Ideally, an ANN can be trained to recognize any given set of inputs by adjusting the synaptic weights. A properly trained network, in principle, should be able to apply this learning and respond appropriately to completely new inputs (Adeli & Hung, 1994). Due to their success, ANNs have been used as powerful computational tools to solve complex pattern

* Corresponding address: 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH 43210, United States.
E-mail address: adeli.1@osu.edu (H. Adeli).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.04.003

recognition, function estimation, and classification problems not readily amenable to other analytical tools.

The feedforward neural network is the simplest (and therefore, the most common) ANN architecture in terms of information flow direction. In fact, many of the other above-mentioned neural network architectures are variations of the feedforward neural network. However, in spite of the widespread popularity of the feedforward network, until a few years ago the processing mechanism of the artificial neuron had undergone only one major conceptual change, which is the use of a mathematically-defined activation or transfer function. The early neurons developed in the 1940s and 1950s (sometimes referred to as *first generation* neurons in the literature) acted as simple integrate-and-fire units which fired when the *internal state* (defined as the weighted sum of inputs to each neuron) reached a threshold. The *second generation* neurons developed in the 1960s, 1970s, and 1980s defined the internal state in a similar manner but used a mathematically-defined activation function, often a smooth sigmoid or radial basis function, instead of a fixed threshold value, for output determination (Maass, 1996).

Biologically, a presynaptic neuron communicates with a postsynaptic neuron via trains of spikes which have a fixed morphology and amplitude (Bose & Liang, 1996). Based on studies of the cortical pyramidal neurons, neuroscientists have concluded the transmitted information is usually encoded in the frequency of spiking (*rate encoding*) and/or in the timing of the spikes (*pulse encoding*) (Kasinski & Ponulak, 2006; Maass, 1996, 1997a; Sejnowski, 1986). The output or response of the second generation neurons is based loosely on rate encoding, an approximate representation of the average rate of firing. The output from the presynaptic neuron approximates the average firing rate of the presynaptic neuron. In the postsynaptic neuron, an activation function is used to transform the input into a proportionate output which approximates the average firing rate of the postsynaptic neuron. Historically, the primary reason for this was the constraint imposed by Rumelhart's widely-used backpropagation (BP) training algorithm (Adeli & Hung, 1994; Ghosh-Dastidar & Adeli, 2006; Hooshdar & Adeli, 2004; Hung & Adeli, 1993, 1994; Rumelhart et al., 1986) which required a continuous and differentiable activation function. As a result, even now a mathematical abstraction is used to avoid modeling the actual spike train.

Rate encoding can be considered to be a special (and less powerful) case of pulse encoding because in pulse encoding the spike timings are known, and the average firing rate can be easily computed based on that information. However, in rate encoding the ability to encode complex spike trains is reduced significantly because the temporal information about individual spikes is completely lost. To overcome this shortcoming, in the past decade, third-generation pulse encoding-based neurons, dubbed *spiking* neurons, have been developed that incorporate the temporal aspect of neuronal firing, an additional degree of biological plausibility (Maass, 1997a).

Spiking neural networks (SNNs) are networks of spiking neurons that have been shown theoretically to have the ability to approximate any continuous function (Maass, 1996, 1997b, 1997c). The inputs to a spiking neuron are discrete spike times. Every input spike results in a postsynaptic potential (PSP) on the neuron. The neuron temporally integrates the PSPs resulting from every input spike into a dynamic time-varying internal state. The spiking neuron has an "all or none" response and fires output action potentials or spikes at the time instants its internal state exceeds the neuron threshold. The output spike train of the ith presynaptic neuron inputted to the jth postsynaptic neuron, the resultant PSPs, and the output spike train of the jth postsynaptic neuron are shown in Fig. 1. In this figure, the superscript (g) indicates the sequence of the particular spike, and the time of the gth output spike of

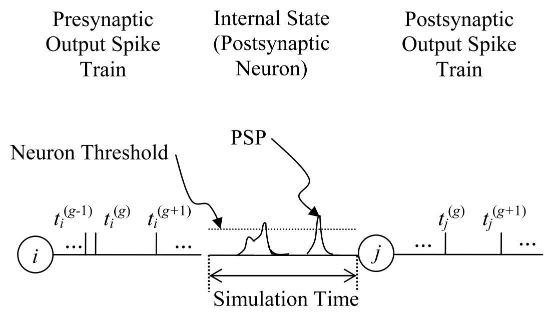


Fig. 1. The output spike train of the ith presynaptic neuron inputted to the jth postsynaptic neuron, the resultant PSPs, and the output spike train of the jth postsynaptic neuron.

the presynaptic neuron i is denoted by $t_i^{(g)}$. For example, the time of the first output spike of the presynaptic neuron i is denoted by $t_i^{(g=1)}$ or $t_i^{(1)}$. The same notation is used for the postsynaptic neuron j except that the subscript is changed to j. Addition of the temporal dimension for information encoding has the potential to result in compact representations of large neural networks, another advantage for SNNs. Despite the promise of improved computational power based on pulse encoding, SNN research is still in its early stages (Ghosh-Dastidar & Adeli, 2007).

Initially, due to the discrete response of spiking neurons, applications of SNNs had been limited to unsupervised learning and self-organizing maps. The lack of a continuous and differentiable activation function relating the internal state of the neuron to the output spike times made spiking neurons incompatible with the error backpropagation required for supervised learning. This problem was overcome when Bohte, Kok, and La Poutré (2002) presented a backpropagation (BP) training algorithm for a feedforward SNN, dubbed *SpikeProp*, similar in concept to the BP algorithm developed for traditional neural networks (Rumelhart et al., 1986). SpikeProp makes error backpropagation possible by assuming that the value of the internal state of the neuron increases linearly in the infinitesimal time around the instant of neuronal firing. In other words, around the instant of neuronal firing the internal state is approximated by a finely discretized piecewise linear function. This assumption is especially significant in light of the highly uneven error surface for SNN training which can wreak havoc with the gradient descent-based training algorithms.

Another problem with SNN training is that the computational effort is usually at least two orders of magnitude more than that for the traditional ANNs for two reasons. First, multiple weights have to be computed for multiple synapses connecting a presynaptic neuron to a postsynaptic neuron. Second, the internal state of each neuron has to be computed for a continuous duration of time, called the *simulation time*, to obtain the output spiking times (Fig. 1). The time resolution, called the *time step*, employed for this computation along with the *simulation time* results in a total of $K \times$ simulation time/time step computations per connection between a presynaptic and a postsynaptic neuron (compared to 1 computation for traditional ANNs). In addition, the number of convergence epochs is another key factor that affects the actual computation time (real time) required to train the network. In order to improve the network training performance, SNNs have been used with various training algorithms such as backpropagation with momentum, QuickProp, resilient propagation (RProp), and Levenberg–Marquardt BP (Ghosh-Dastidar & Adeli, 2007; McKennoch, Liu, & Bushnell, 2006; Silva & Ruano, 2005; Xin & Embrechts, 2001).

An important characteristic of a biological presynaptic neuron is its ability to affect a postsynaptic neuron differentially over time by inducing PSPs of varying magnitudes at various time instants. In a biological system, this ability is incorporated using a combination of two strategies: (1) multiple spikes at different times (*spike train*)

Table 1
Computational efficiency of the SpikeProp learning algorithm for the single-spiking SNN reported in the literature for the XOR problem using a simulation time of 50 ms. [Computational Efficiency = Time Step/(Simulation Time × No. of Epochs for Convergence)].

	Time step	Limiting convergence MSE	No. of Epochs for convergence	Computational efficiency
Bohte et al. (2002)	0.01	0.25	250	0.8×10^{-6}
Moore (2002)	0.01	0.5	121	1.7×10^{-6}
McKennoch et al. (2006)	0.025	0.5	127	3.9×10^{-6}
Ghosh-Dastidar and Adeli (2007)	0.025	0.5	39	1.0×10^{-3}

from the presynaptic neuron to the postsynaptic neuron and/or (2) multiple synapses between two neurons with different synaptic weights and delays. The SNN model used by Bohte et al. (2002) and other researchers extending their work (Ghosh-Dastidar & Adeli, 2007; McKennoch et al., 2006; Moore, 2002; Schrauwen & van Campenhout, 2004; Silva & Ruano, 2005; Xin & Embrechts, 2001) is a simplified model in which only the latter strategy, i.e., multiple synapses, is employed and each presynaptic neuron is restricted to the emission of a single output spike. This model will be referred to henceforth as the single-spiking SNN model. In the literature, to the best of the authors' knowledge, Booij and Nguyen (2005) present the only model that combines both aforementioned strategies.

This research focuses primarily on developing a new BP-learning algorithm for a feedforward SNN that is based on more plausible neuronal dynamics (i.e. pulse-encoding spike-train communication) compared to traditional rate-encoding based ANNs and single-spiking SNNs. Recently, the authors presented an improved and efficient single-spiking SNN model and applied it for epilepsy and epileptic seizure detection, a complicated pattern recognition problem (Ghosh-Dastidar & Adeli, 2007). An extensive parametric analysis was performed to identify heuristic rules and optimum parameter values that increased the computational efficiency of SpikeProp, QuickProp, and RProp by a factor of 588, 82, and 75, respectively, and yielded a high classification accuracy of 92.5% for the EEG classification problem. In this paper, a new SNN model is presented in which the presynaptic neuron transmits information to the postsynaptic neuron in the form of multiple spikes via multiple synapses. The new SNN model is named *Multi-Spiking Neural Network* (*MuSpiNN*). Further, a new supervised learning or training algorithm, dubbed *Multi-SpikeProp*, is developed for training MuSpiNN.

The SNN model and learning algorithm presented in this paper differ from those of Booij and Nguyen (2005) in three key aspects: (1) the dynamics of the spiking neuron, (2) the training algorithm, and (3) the heuristic rules and optimum parameter values discovered by the authors that improve the computational efficiency of the underlying SpikeProp algorithm by two orders of magnitude as summarized in Table 1 (Ghosh-Dastidar & Adeli, 2007). The performance of MuSpiNN and Multi-SpikeProp is evaluated using three different problems: the XOR problem, the Fisher iris classification problem, and the EEG classification problem for epilepsy and seizure detection. For the iris and EEG classification problems, a modular architecture is employed to reduce each 3-class classification problem to three 2-class classification problems with the goal of improving the classification accuracy.

2. Multi-spiking neural network (MuSpiNN) and neuron model

2.1. MuSpiNN architecture

The architecture of MuSpiNN is shown in Fig. 2(a). In contrast to traditional feedforward ANNs where two neurons are connected by one synapse only, the connection between two SNN neurons is modeled by multiple synapses as shown in Fig. 2(b). In this aspect, the MuSpiNN model is similar to the single-spiking SNN described by Natschläger and Ruf (1998) and Bohte et al. (2002). The fundamental difference is in the ability of a MuSpiNN neuron to assimilate multiple input spikes from presynaptic neurons and emit multiple output spikes in response. In other words, information transmitted from one neuron to the next is encoded in the form of a spike train instead of a single spike. The magnified connection in Fig. 2(b) displays the temporal sequence of spikes (short vertical lines) from the presynaptic neuron, the synaptic weights (proportionate to the size of the star shaped units in the center), and the resulting PSPs (proportionate to the size of the PSP).

The network is assumed to be fully connected; i.e. a neuron in any layer l is connected to all neurons in the preceding layer $l+1$ (layers are numbered backward starting with the output layer, numbered as layer 1). Consequently, a neuron j ($\in \{1, 2, \dots, N_l\}$) in layer l is postsynaptic to N_{l+1} presynaptic neurons, where N_l is the number of neurons in layer l. Each presynaptic neuron i ($\in \{1, 2, \dots, N_{l+1}\}$) is connected to the postsynaptic neuron j via K synapses. The number K is constant for any two neurons. The weight of the kth synapse ($k \in \{1, 2, \dots, K\}$) between neurons i and j is denoted by w_{ij}^k. Assuming that presynaptic neuron i fires a total of G_i spikes and the gth spike ($g \in [1, G_i]$) is fired at time $t_i^{(g)}$ (Fig. 1), the kth synapse transmits the gth spike to the postsynaptic neuron at time $t_i^{(g)} + d^k$, where d^k is the delay associated with the kth synapse. The modeling of synapses is identical for all neurons, and the kth synapse between any two neurons has the same delay, d^k. Following the same notation, the output of the postsynaptic neuron j is a sequence of G_j spikes, in which the gth spike ($g \in [1, G_j]$) is fired at time $t_j^{(g)}$ (Fig. 1).

2.2. Multi-spiking neuron and the spike response model

As long as the internal state of a biological postsynaptic neuron does not exceed the neuron threshold (Fig. 3), the internal state is defined as the sum of the PSPs induced by all input spikes from all presynaptic neurons and synapses. Mathematically, the internal state of the postsynaptic neuron j in layer l at time t is modeled as (Gerstner & Kistler, 2002)

$$x_j(t) = \sum_{i=1}^{N_{l+1}} \sum_{k=1}^{K} \sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t - t_i^{(g)} - d^k) \tag{1}$$

where ε represents the spike response function, i.e., the PSP or the unweighted internal response of the postsynaptic neuron to a single spike. The three summations represent the weighted sum over all (G_i) input spikes from all (N_{l+1}) presynaptic neurons to the jth neuron in layer l via all (K) synapses. An internal state equal to zero is called the resting potential (Fig. 3). In this research, the so-called α-function is used as the spike response function (Bohte et al., 2002; Gerstner & Kistler, 2002):

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}} & \text{when } t > 0 \\ 0 & \text{when } t \le 0 \end{cases} \tag{2}$$

where τ is the time decay constant that determines the spread shape of the function, as explained in Ghosh-Dastidar and Adeli (2007). The α-function has a maximum value of 1 at $t = \tau$. Booij

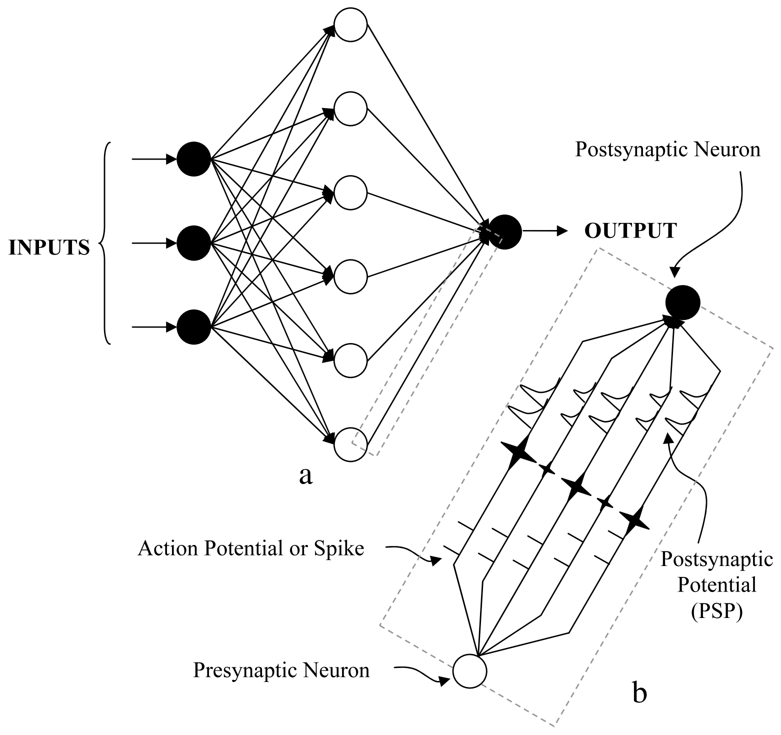


Fig. 2. (a) Spiking neural network architecture; (b) multiple synapses transmitting multiple spikes from a presynaptic neuron to a postsynaptic neuron.

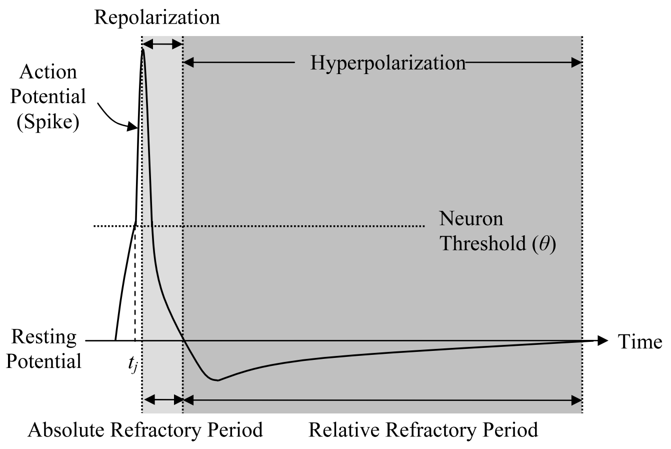


Fig. 3. The internal state of a postsynaptic neuron in response to a presynaptic spike (not shown in the figure) showing the action potential, and repolarization and hyperpolarization phases.

and Nguyen (2005) use a different spike response function defined as the difference of two exponential functions. The model and the learning algorithm presented in this paper can also incorporate that function but no significant changes in the result are expected.

When the internal state exceeds the neuron threshold, θ, the neuron fires an output spike at the time instant t_j, and the internal state immediately starts dropping to the *resting potential* of the neuron (Fig. 3). This phase is known as *repolarization* (Fig. 3). The duration of the repolarization phase is known as the *absolute refractory period* in which the neuron cannot fire regardless of the number or frequency of the input spikes (Fig. 3). Subsequently, the internal state is kept at a value lower than the resting potential by various biological processes. As a result, it becomes difficult for the neuron to reach the threshold and fire again for a period of time, known as the *relative refractory period*. This phase is known as *hyperpolarization* (Fig. 3) (Bose & Liang, 1996; Kandel, Schwartz, & Jessell, 2000). Due to repolarization and hyperpolarization, the internal state of a postsynaptic neuron depends not only on the timing of the input spikes from all presynaptic neurons but also on the timing of its own output spikes. For a more detailed discussion of the modeling of the internal state of a postsynaptic neuron, refer to the recent article by the authors Ghosh-Dastidar and Adeli (2007).

Since a single-spiking neuron is restricted to firing only one output spike, the single-spiking SNN model of Bohte et al. (2002) is unaffected by the timing of its own output spikes. Eqs. (1) and (2) are sufficient to represent the dynamics of a single-spiking neuron but not for the multi-spiking neuron in the new MuSpiNN model. To model the relative refractory period for the multi-spiking neuron, a *refractoriness* term is added to the right-hand side of Eq. (1). This refractoriness term ensures that the membrane potential becomes negative after the firing of a spike which makes it difficult for the neuron to emit subsequent spikes for a period of time, as explained earlier. Therefore, in this research the internal state of the postsynaptic neuron j in layer l at time t is expressed as (Gerstner & Kistler, 2002)

$$x_j(t) = \sum_{i=1}^{N_{l+1}} \sum_{k=1}^{K} \sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t - t_i^{(g)} - d^k) + \rho(t - t_j^{(f)}) \tag{3}$$

where ρ represents the refractoriness function and $t_j^{(f)}$ is the timing of the most recent, the fth, output spike from neuron j prior to time t. For $t < t_j^{(1)}$, the time of the first output spike, the refractoriness term is zero and Eq. (3) is reduced to Eq. (1). Eq. (3) is different from the corresponding equation presented in the model of Booij and Nguyen (2005) where the refractoriness term is summed over all output spikes from neuron j prior to time t (instead of only the most recent one). That model assumes that, at any time t, the internal state of the neuron is affected by the refractoriness due to all spikes prior to time t. This assumption is biologically unrealistic because, for a biological neuron, the ensuing refractoriness after every spike has to be overcome before

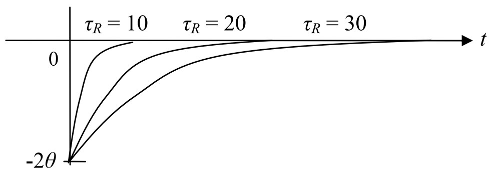


Fig. 4. The refractoriness function for the three different values $\tau_R = 10$, 20, and 30.

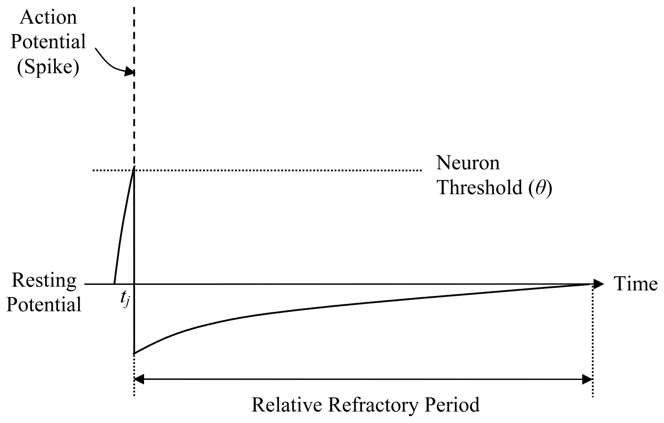


Fig. 5. The mathematical model of the internal state of a postsynaptic neuron in response to a presynaptic spike (not shown in the figure) showing the action potential and the relative refractory period.

the neuron can spike again. Therefore, when the neuron does spike the next time, it is implicit that the preceding refractoriness has been overcome and is of no further consequence. From this progression of events it is clear that, at any time, no refractoriness is retained from any previous spike except the most recent one, which is reflected in Eq. (3).

In this research, ρ is expressed as

$$\rho(t) = \begin{cases} -2\theta e^{-\frac{t}{\tau_R}} & \text{when } t > 0 \\ 0 & \text{when } t \le 0 \end{cases} \tag{4}$$

where θ is the neuron threshold and τ_R is the time decay constant that determines the spread shape of the refractoriness function. Fig. 4 shows the refractoriness function for three different values of τ_R. The function has a negative value in the range $t = 0$ to ∞ with a minimum value of -2θ at $t = 0$. Its value increases with time and approaches zero at $t = \infty$. Substituting Eq. (4) in Eq. (3) ensures that, at the instant of spike firing, $t_j^{(f)}$, the internal state of the neuron decreases instantaneously from the threshold θ to $\theta - 2\theta = -\theta$. Subsequently, the internal state remains lower than the resting potential and increases exponentially to the resting potential which accurately models the relative refractory period and the hyperpolarization phase shown in Fig. 3. The corresponding equation in Booij and Nguyen (2005) uses a coefficient of $-\theta$ for the exponential term in Eq. (4) which reduces the internal state to the resting potential (but not lower) at the instant of spike firing. At no subsequent point in time does the internal state decrease to a value less than the resting potential which does not accurately model the relative refractory period and the hyperpolarization phase (Fig. 3). Fig. 5 shows the mathematical model of the overall neuron dynamics represented by Eq. (3). This model is an approximate representation of the dynamics of the biological neuron shown in Fig. 3. The absolute refractory period observed in the biological model (Fig. 3) is neglected in the mathematical model.

3. Multi-SpikeProp: Backpropagation training algorithm for MuSpiNN

3.1. MuSpiNN error function

Supervised learning requires the availability of a set of actual or *desired* output spike trains which is the desired output from MuSpiNN given a set of input spike trains (one spike train per input neuron). A measure of the difference between the computed and desired outputs is used to compute the network error. MuSpiNN is trained by backpropagating the error (as explained shortly) and adjusting the synaptic weights such that the network error is minimized. Desired outputs for SNNs, unlike those for traditional neural networks, have to be in terms of discrete spike times. The transformation of real-valued outputs to discrete spike times is known as *output encoding* and it can be achieved in a number of different ways (Ghosh-Dastidar & Adeli, 2007). The appropriateness of any particular output encoding depends on the specific problem application.

In this research, neurons in the output layer of MuSpiNN are restricted to emitting a single output spike so that the error function is computed with no additional difficulty. The availability of multiple output spikes at various discrete times requires a more complicated error function. The complexity is compounded by the fact that the number of spikes in the computed output spike trains is variable and highly likely to be different from the number in the desired output spike trains. Such an error function needs to be explored in the context of the output encoding for selected problems, which is beyond the scope of the current research. As such, considering only one output spike, the network error is computed as follows:

$$E = \frac{1}{2}\sum_{j=1}^{N_1}(t_j - t_j^d)^2 \tag{5}$$

where t_j and t_j^d are the computed and desired spike times, respectively, for the jth neuron in the output layer ($l = 1$).

3.2. Error backpropagation for adjusting synaptic weights

The generalized delta update rule is employed to backpropagate the error and adjust the synaptic weights. The weight adjustment for the kth synapse between the ith presynaptic and the jth postsynaptic neuron is computed as

$$\Delta w_{ij}^k = -\eta \nabla E_{ij}^k \tag{6}$$

where η is the learning rate and ∇E_{ij}^k is the gradient (with respect to the weights) of the error function for the kth synapse between the ith presynaptic neuron and the jth postsynaptic neuron. The computation of the gradient is different for the output layer and the hidden layers and is described separately for the two in the following sections. For the sake of clarity, a neuron in the output layer ($l = 1$) will be designated with the subscript j and a neuron in the hidden layer immediately presynaptic to the output layer ($l = 2$) with the subscript i. A neuron in the input or hidden layer presynaptic to the hidden layer $l = 2$ is designated by the subscript h. The subscripts of all other variables are adjusted based on this nomenclature.

3.3. Gradient computation for synapses between a neuron in the last hidden layer and a neuron in the output layer

Using the chain rule, the error gradient at the postsynaptic neuron output spike time instant, $t = t_j$, is represented as the product of three partial derivative terms:

$$\nabla E_{ij}^k = \frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_j}\frac{\partial t_j}{\partial x_j(t_j)}\frac{\partial x_j(t_j)}{\partial w_{ij}^k}. \tag{7}$$

The first, third, and second partial derivative terms are derived in that order, which is the order of their complexity. The first partial derivative term on the right-hand side is computed as

$$\frac{\partial E}{\partial t_j} = \frac{\partial \left[\frac{1}{2}\sum_{j=1}^{N_1}(t_j - t_j^d)^2\right]}{\partial t_j} = (t_j - t_j^d). \tag{8}$$

The third partial derivative term on the right-hand side of Eq. (7) is computed as

$$\frac{\partial x_j(t_j)}{\partial w_{ij}^k} = \frac{\partial \left[\sum_{i=1}^{N_2}\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k) + \rho(t_j - t_j^{(f)})\right]}{\partial w_{ij}^k}. \tag{9}$$

Since the neurons in the output layer are permitted to fire only one output spike, the refractoriness term in the numerator of Eq. (9) becomes zero. Because the weight of any one synapse is independent of the weights of the other synapses, the summations with respect to *i* and *k* vanish, and Eq. (9) is reduced to

$$\frac{\partial x_j(t_j)}{\partial w_{ij}^k} = \frac{\partial \left[\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k)\right]}{\partial w_{ij}^k}. \tag{10}$$

Since the α-function, ε, is independent of the synaptic weights, Eq. (10) is further simplified to

$$\frac{\partial x_j(t_j)}{\partial w_{ij}^k} = \sum_{g=1}^{G_i} \varepsilon(t_j - t_i^{(g)} - d^k). \tag{11}$$

Eq. (11) is a more generalized form of the one used by Bohte et al. (2002) which is limited to single input spikes from neurons presynaptic to the output layer neurons.

The second partial derivative term in Eq. (7), $\partial t_j/\partial x_j(t_j)$, cannot be computed directly because t_j cannot be expressed as a continuous and differentiable function of $x_j(t_j)$, as explained earlier. Bohte et al. (2002) overcome this problem by assuming that $x_j(t_j)$ is a linear function of t_j *around the output spike time instant*, $t = t_j$, and approximate the term $\partial t_j/\partial x_j(t_j)$ as $-1/(\partial x_j(t_j)/\partial t_j)$. In this article, the authors arrive at the same solution but offer a different or, perhaps, a clearer explanation of this approximation.

Consider an example graph of the internal state $x_j(t)$ versus time t shown in Fig. 6(a). The internal state reaches threshold θ and fires an output spike at time t_j. Before reaching the threshold θ, the internal state $x_j(t)$ is independent of the threshold. A decrease in θ leads to a decrease in the output spike time t_j (the neuron fires earlier) and vice versa. This is evident from Fig. 6(a) where output spike time t_j occurs at the intersection of the dashed line representing θ and the solid line representing the graph of $x_j(t)$. As a result, if θ is considered as a variable parameter, any point $(t, x_j(t))$ can be represented by the point (t_j, θ). Fig. 6(b) shows the variation of t_j with θ. It is observed that if θ is reduced to θ_1, t_j is proportionately reduced to t_{j1}. However, when θ_1 is further reduced to θ_2, t_{j1} is disproportionately reduced to t_{j2}. The threshold θ at which the spike time jumps is designated θ_{jump} in this paper.

Following the literature in SNN research, it is assumed that the value of the threshold θ is not close to θ_{jump}. Based on extensive modeling and simulations it has been found that limiting learning rates and weight changes during learning constrains the error function locally within the error surface and avoids drastic jumps in spike times. This can be ensured by choosing a low learning rate and developing heuristic rules to be discussed later. As a result, the assumption of linearity holds around the spike time t_j where the

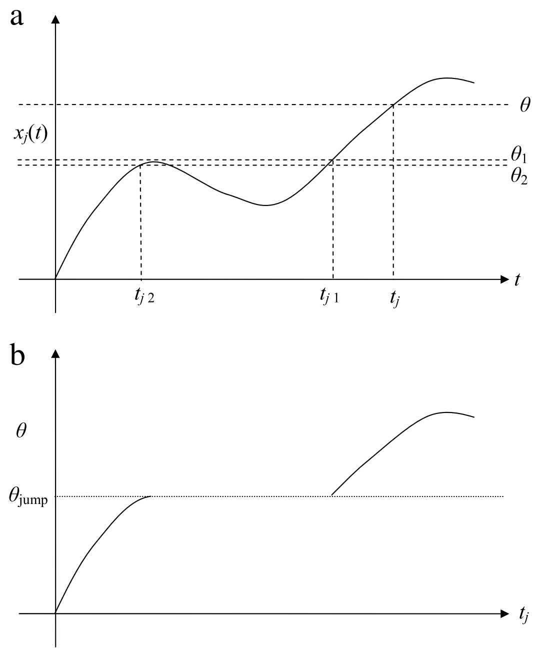


Fig. 6. (a) Internal state, $x_j(t)$, versus time, t; (b) neuron threshold, θ, versus output spike time, t_j.

slopes of the two graphs shown in Fig. 6(a) and (b) are identical, i.e.,

$$\left.\frac{\partial x_j(t)}{\partial t}\right|_{t=t_j} = \frac{\partial \theta}{\partial t_j}. \tag{12}$$

The left-hand side of Eq. (12) is approximated instantaneously as $\partial x_j(t_j)/\partial t_j$ for small changes in $x_j(t)$. As a result of the assumption of linearity around time t_j, the slope of the graph of θ versus t_j is computed as

$$\frac{\partial t_j}{\partial \theta} = \frac{\Delta t_j}{\Delta \theta} = \frac{1}{\Delta\theta/\Delta t_j} = \frac{1}{\partial\theta/\partial t_j} \tag{13}$$

where Δt_j represents an infinitesimal change in t_j and $\Delta\theta$ represents an infinitesimal change in θ. Moreover, to increase the output spike time t_j, the internal state of the neuron $x_j(t_j)$ has to be decreased or, equivalently, the threshold θ has to be increased. This opposite relationship between the internal state and threshold, with respect to the output spike time, is modeled as

$$\frac{\partial t_j}{\partial x_j(t_j)} = -\frac{\partial t_j}{\partial \theta}. \tag{14}$$

From Eqs. (12), (13), and (14), the second partial derivative term in Eq. (7) becomes

$$\frac{\partial t_j}{\partial x_j(t_j)} = \frac{-1}{\partial x_j(t_j)/\partial t_j}. \tag{15}$$

The denominator of Eq. (15) is computed as

$$\frac{\partial x_j(t_j)}{\partial t_j} = \frac{\partial \left[\sum_{i=1}^{N_2}\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k) + \rho(t_j - t_j^{(f)})\right]}{\partial t_j}. \tag{16}$$

Similar to the derivation of Eq. (10), since the neurons in the output layer are permitted to fire only one output spike, the refractoriness term in the numerator of Eq. (16) becomes zero. Because the output spike times transmitted through one synapse are independent of the output spike times through the other

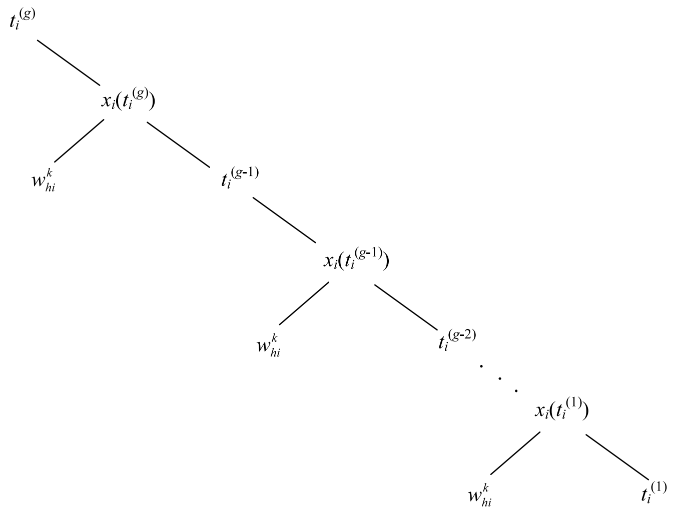


Fig. 7. Recursive dependencies of the times of output spikes from the postsynaptic neuron i on the weights of synapses connecting the presynaptic neuron h to the postsynaptic neuron i.

synapses, the summations can be placed outside the derivative, and Eq. (16) is rewritten as

$$\frac{\partial x_j(t_j)}{\partial t_j} = \sum_{i=1}^{N_2}\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \frac{\partial \varepsilon(t_j - t_i^{(g)} - d^k)}{\partial t_j}$$
$$= \sum_{i=1}^{N_2}\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k)\left(\frac{1}{(t_j - t_i^{(g)} - d^k)} - \frac{1}{\tau}\right). \quad (17)$$

Substituting Eq. (17) in Eq. (15), we obtain

$$\frac{\partial t_j}{\partial x_j(t_j)} = \frac{-1}{\sum_{i=1}^{N_2}\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k)\left(\frac{1}{(t_j-t_i^{(g)}-d^k)} - \frac{1}{\tau}\right)}. \quad (18)$$

In summary, the error gradient for adjusting the synaptic weights, ∇E_{ij}^k, is computed using Eqs. (7), (8), (11), and (18).

Fig. 6(a) and (b) can be used to explain the reason for requiring low learning rates and various heuristic rules to limit the changes of the synaptic weights (Ghosh-Dastidar & Adeli, 2007). In the example shown, assume that the weights need to be increased in order to increase the internal state of the neuron (or, equivalently, decrease the threshold) and obtain an earlier spike time. As long as the weight changes are small, the graph of θ versus t_j remains continuous and the assumption of linearity is not unreasonable. However, if the changes in the weights are too large and lead to the region around θ_{jump} in Fig. 6(b), a small change in θ leads to a disproportionate change in the output spike time t_j which jumps to a much earlier time. In this region, the assumption of linearity does not hold, and SNN training fails.

3.4. *Gradient computation for synapses between a neuron in the input or hidden layer and a neuron in the hidden layer*

For a postsynaptic neuron i in a hidden layer $l = 2$, the error is backpropagated from all neurons in the output layer, and the gradient is computed using the chain rule as

$$\nabla E_{hi}^k = \sum_{j=1}^{N_1} \frac{\partial E}{\partial w_{hi}^k} = \sum_{j=1}^{N_1} \frac{\partial E}{\partial t_i^{(g)}}\frac{\partial t_i^{(g)}}{\partial w_{hi}^k} \quad (19)$$

where the subscript h denotes the hth neuron in layer $l+1$ that is presynaptic to the postsynaptic neuron i in layer l. The first partial derivative term in Eq. (19) models the dependence of the network error in Eq. (5) on the output spike times $t_i^{(g)}$ from the neuron i in the hidden layer $l = 2$, and is expanded using the chain rule as

$$\frac{\partial E}{\partial t_i^{(g)}} = \frac{\partial E}{\partial t_j}\frac{\partial t_j}{\partial x_j(t_j)}\frac{\partial x_j(t_j)}{\partial t_i^{(g)}}. \quad (20)$$

The first two partial derivative terms on the right-hand side are computed according to Eqs. (8) and (18), respectively. The last partial derivative term is computed as

$$\frac{\partial x_j(t_j)}{\partial t_i^{(g)}} = \frac{\partial\left[\sum_{i=1}^{N_{l+1}}\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k) + \rho(t_j - t_j^{(f)})\right]}{\partial t_i^{(g)}}. \quad (21)$$

The output spike times, $t_i^{(g)}$, from neuron i in the hidden layer $l = 2$ are the input spike times for neuron j in the output layer $l = 1$ and independent of the output spike times for neuron j. Therefore, the refractoriness term in the derivative vanishes. The summation with respect to i also vanishes because the output spikes of any neuron in the hidden layer do not depend on the output spikes of any other neuron in the same layer. Since the synaptic weights are independent of the output spike times, the factor w_{ij}^k is placed outside the derivative, and Eq. (21) is reduced to

$$\frac{\partial x_j(t_j)}{\partial t_i^{(g)}} = \sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \frac{\partial \varepsilon(t_j - t_i^{(g)} - d^k)}{\partial t_i^{(g)}}$$
$$= -\sum_{k=1}^{K}\sum_{g=1}^{G_i} w_{ij}^k \varepsilon(t_j - t_i^{(g)} - d^k)\left(\frac{1}{(t_j - t_i^{(g)} - d^k)} - \frac{1}{\tau}\right). \quad (22)$$

The negative sign in Eq. (22) appears because the derivative of the α-function, $\varepsilon(t_j - t_i^{(g)} - d^k)$, is calculated with respect to $t_i^{(g)}$ (unlike Eq. (17) where it is calculated with respect to t_j).

The computation of the second partial derivative term in Eq. (19) is more complicated than in a single-spiking model where the output spike time from any neuron depends on the internal state of the neuron i which, in turn, depends only on the inputs and weights of the synapses to the neuron. In the MuSpiNN model, the internal state of the neuron i, in addition to the aforementioned dependencies, depends on the time of its own most recent output spike. Fig. 7 shows the recursive dependencies of the times of output spikes from the postsynaptic neuron i on the weights of synapses connecting the presynaptic neuron h to the postsynaptic neuron i where the variable represented by any node in the tree is dependent on the variables represented by the nodes in the level immediately below. As a result of these recursive dependencies, the error is backpropagated from output spike to output spike starting from the last output spike. The second partial derivative term in Eq. (19) is computed recursively using the following set of equations:

$$\frac{\partial t_i^{(g)}}{\partial w_{hi}^k} = \frac{\partial t_i^{(g)}}{\partial x_i(t_i^{(g)})}\left[\frac{\partial x_i(t_i^{(g)})}{\partial w_{hi}^k} + \frac{\partial x_i(t_i^{(g)})}{\partial t_i^{(g-1)}}\cdot\frac{\partial t_i^{(g-1)}}{\partial w_{hi}^k}\right] \quad (23)$$

$$\frac{\partial t_i^{(g-1)}}{\partial w_{hi}^k} = \frac{\partial t_i^{(g-1)}}{\partial x_i(t_i^{(g-1)})}\left[\frac{\partial x_i(t_i^{(g-1)})}{\partial w_{hi}^k} + \frac{\partial x_i(t_i^{(g-1)})}{\partial t_i^{(g-2)}}\cdot\frac{\partial t_i^{(g-2)}}{\partial w_{hi}^k}\right] \quad (24)$$

$$\vdots$$

$$\frac{\partial t_i^{(1)}}{\partial w_{hi}^k} = \frac{\partial t_i^{(1)}}{\partial x_i(t_i^{(1)})}\left[\frac{\partial x_i(t_i^{(1)})}{\partial w_{hi}^k}\right]. \quad (25)$$

Similar to the derivation of Eq. (18) from Eq. (15), the first partial derivative term outside the brackets in Eq. (23), $\partial t_i^{(g)}/\partial x_i(t_i^{(g)})$, is computed as shown in Box I.

$$\begin{aligned}
\frac{\partial t_i^{(g)}}{\partial x_i(t_i^{(g)})} &= \frac{-1}{\partial x_i(t_i^{(g)})/\partial t_i^{(g)}} \\
&= \frac{-1}{\partial\left[\sum_{h=1}^{N_I}\sum_{k=1}^{K}\sum_{f=1}^{G_h} w_{hi}^k \varepsilon(t_i^{(g)} - t_h^{(f)} - d^k) + \rho(t_i^{(g)} - t_i^{(g-1)})\right] \Big/ \partial t_i^{(g)}} \\
&= \frac{-1}{\sum_{h=1}^{N_I}\sum_{k=1}^{K}\sum_{f=1}^{G_h} w_{hi}^k \frac{\partial\varepsilon(t_i^{(g)} - t_h^{(f)} - d^k)}{\partial t_i^{(g)}} + \frac{\partial\rho(t_i^{(g)} - t_i^{(g-1)})}{\partial t_i^{(g)}}} \\
&= \frac{-1}{\sum_{h=1}^{N_I}\sum_{k=1}^{K}\sum_{f=1}^{G_h} w_{hi}^k \varepsilon(t_i^{(g)} - t_h^{(f)} - d^k)\left(\frac{1}{(t_i^{(g)} - t_h^{(f)} - d^k)} - \frac{1}{\tau}\right) + \frac{2\theta}{\tau_R}\rho(t_i^{(g)} - t_i^{(g-1)})}
\end{aligned}$$

Box I.

Similar to Eq. (11), the first term within the brackets in Eq. (23) is computed as

$$\frac{\partial x_i(t_i^{(g)})}{\partial w_{hi}^k} = \sum_{f=1}^{G_h} \varepsilon(t_i^{(g)} - t_h^{(f)} - d^k). \tag{26}$$

The first partial derivative term of the second term within the brackets in Eq. (23) is computed as

$$\frac{\partial x_i(t_i^{(g)})}{\partial t_i^{(g-1)}} = \frac{\partial\left[\sum_{h=1}^{N_I}\sum_{k=1}^{K}\sum_{f=1}^{G_h} w_{hi}^k \varepsilon(t_i^{(g)} - t_h^{(f)} - d^k) + \rho(t_i^{(g)} - t_i^{(g-1)})\right]}{\partial t_i^{(g-1)}}. \tag{27}$$

The first term in the numerator does not depend on any previous output spike times; thus Eq. (27) is reduced to

$$\frac{\partial x_i(t_i^{(g)})}{\partial t_i^{(g-1)}} = \frac{\partial\rho(t_i^{(g)} - t_i^{(g-1)})}{\partial t_i^{(g-1)}} = \frac{2\theta}{\tau_R}\rho(t_i^{(g)} - t_i^{(g-1)}). \tag{28}$$

The error gradient for adjusting the synaptic weights is computed using Eqs. (8), (18)–(20), (22)–(25), Box I, (26) and (28).

4. Applications

4.1. Optimum parameter selection and weight initialization

The performance of the MuSpiNN model and Multi-SpikeProp learning algorithm are evaluated using three increasingly difficult pattern recognition problems: XOR, Fisher iris plant classification (Fisher, 1936; Newman, Hettich, Blake, & Merz, 1998), and EEG epilepsy and seizure detection (Adeli, Ghosh-Dastidar, & Dadmehr, 2007; Andrzejak et al., 2001; Ghosh-Dastidar & Adeli, 2007; Ghosh-Dastidar, Adeli, & Dadmehr, 2007, 2008) based on the classification accuracy. The classification accuracy is computed only for the iris and EEG datasets because they are large enough to be divided into training and testing datasets. The transformation of real-valued inputs and outputs to discrete spike times (output encoding) is different for the three problems and therefore is addressed for each problem separately.

The performance of SNNs is affected by three types of parameters that define (a) the spiking neuron (simulation time, time step, neuron threshold, time decay constant τ for the α-function, and time decay constant τ_R for the refractoriness function), (b) network architecture (number of hidden layers, input and output encoding parameters, number of neurons in the input, hidden, and output layers, and number of synapses connecting two neurons), and (c) the training algorithm (learning rate and the convergence criteria). In a recent paper, the authors performed an extensive parametric analysis to select optimum values of aforementioned parameters for a single-spiking SNN model and SpikeProp (Ghosh-Dastidar & Adeli, 2007) with the goal of maximizing accuracy and efficiency. Maximum computation efficiency and classification accuracies were achieved when a learning rate in the range $\eta = 0.001$–0.01 and a time step of 1 ms were employed (the time unit is *virtual* and is used for modeling purposes only). The simulation time employed for the XOR problem was 25 ms and for the iris and EEG problem it was 35 ms. The selection of these parameters has been described in detail in the earlier paper (Ghosh-Dastidar & Adeli, 2007). In MuSpiNN, the underlying model is similar to that of the single-spiking SNN and the difference lies primarily in the learning algorithm's capability of handling multiple spikes. Therefore, it is expected that the same or similar values of parameters will be optimum. The new MuSpiNN model is developed around the same optimum numbers obtained for learning rate, simulation time, time step, number of hidden layers, input and output encoding parameters, and number of neurons in the input, hidden, and output layers. The optimum values of the remaining parameters are obtained differently for the multi-spiking model. Parameter values not discussed in this subsection are different for the three classification problems and, therefore, are discussed separately for each problem in the following subsections.

Weights for MuSpiNN training are initialized in a manner similar to that for a single-spiking SNN. To increase the consistency of convergence of the network training, all neurons are required to fire within the simulation time, at least, in the first epoch of network training (Bohte et al., 2002; McKennoch et al., 2006; Moore, 2002). The initialization method used in this research is as follows (Ghosh-Dastidar & Adeli, 2007; Moore, 2002). The neuron threshold for all spiking neurons in the network is selected as $\theta = 1$ and the weight initialization process and heuristic rules (described shortly) are developed around this number (any value can be chosen for threshold as long as the other parameters and heuristic rules are developed around that value). All spike time inputs are set equal to zero and all synaptic weights are set equal to one, and the corresponding output spike time is obtained by running the SNN model. Next, the weights of all synapses are selected randomly as real numbers in the range 1–10 and normalized by dividing them by the product of the average of all weights and the output spike time for inputs equal to zero computed earlier. The purpose of this normalization is to limit the range of the initial synaptic weights such that all neurons fire within the simulation time, at least, in the first epoch of network training.

4.2. Heuristic rules for multi-SpikeProp

The improved single-spiking SNN model presented in Ghosh-Dastidar and Adeli (2007) employed two heuristic rules that increased the computational efficiency and classification accuracy of the model. The same heuristic rules along with a third new rule are employed for MuSpiNN and Multi-SpikeProp as follows:

(1) In order to prevent catastrophic changes in the synaptic weights, a lower limit of 0.1 is imposed on the denominator in Eqs. (18) and (28) as suggested by Booij and Nguyen (2005) for their equations which are different from the ones used in this research.

(2) If at any time during the training of the network a neuron stops firing, then its contribution to the network error becomes null. During backpropagation of the error, the resulting weight change is very small, which may not be sufficient to restart the firing of the neuron even after several epochs. This issue, referred to as the *silent neuron problem*, leads to a reduction of the effective network size to a size possibly insufficient to model the classification problem which ultimately affects convergence (McKennoch et al., 2006). In this research, the neuron is set to fire at the maximum internal state value if the threshold is not exceeded during the simulation time. Based on this heuristic, every neuron fires during the simulation time.

(3) Our mathematical model of neuron dynamics, represented by Eq. (3), incorporates the relative refractory period but not the absolute refractory period (Figs. 3 and 5). As a result, in some situations during the training of the network, the large weighted inputs to a neuron offset the effect of the refractoriness function, resulting in an effective refractory period equal to 0 ms. In that case, the neuron starts firing continuously, which causes all of its postsynaptic neurons to behave in a similar manner, thus overwhelming the network and adversely affecting convergence. This problem, dubbed the *noisy neuron problem* by the authors of this paper, was not encountered for the single-spiking SNN model which is restricted to only one spike. The problem is analogous to propagation of seizure in an epileptic brain where abnormal neuronal discharges spread via a similar mechanism, i.e., feedforward excitation, to various parts of the brain. In order to overcome this problem, a new heuristic rule is added to model the absolute refractory period by requiring that any neuron be unable to fire again within 2 ms following an output spike.

4.3. XOR problem

The exclusive OR (XOR) problem has been used as a common benchmark for the initial testing of different ANN models. The data consist of two binary input features and one binary output. The dataset consists of four training samples. If the two inputs are identical ({0, 0} or {1, 1}), the output is 0. If the inputs are different ({0, 1} or {1, 0}), the output is 1. The problem is non-trivial as the two classes are not linearly separable. At the same time, the small size (4×2) of the dataset allows fast training of the classification models. Following Bohte et al. (2002), an input value of 1 is encoded as an early spike time (0 ms) whereas a value of 0 is encoded as a late spike time (6 ms). The output values 1 and 0 are encoded as output spike times 10 ms and 16 ms, respectively. Bohte et al. (2002) select an *encoding interval* (difference between the two spike times) of 6 ms and a time decay constant τ for the α-function in Eq. (2) of 7 ms (a slightly larger value than the encoding interval) by trial and error. The same values are used for the XOR problem in this research for the sake of comparison. The number of input and output neurons is selected as three (including the *bias* neuron) and one, respectively. The reason for the use of the bias neuron is explained in Ghosh-Dastidar and Adeli (2007). Only one hidden layer comprising five neurons is used to model the XOR problem. This architecture is the same as that reported in the literature for ease of comparison with those models.

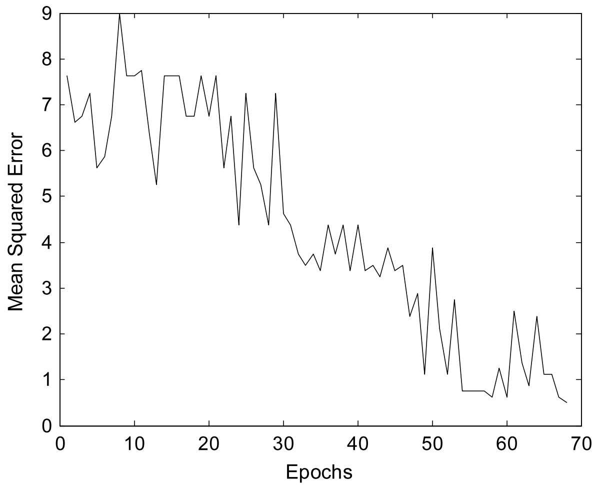


Fig. 8. A sample Multi-SpikeProp convergence curve for the XOR problem.

The model is trained to a convergence mean square error (MSE) value of 0.5. The upper limit for the number of epochs is set to 500. The training is repeated for ten different sets of initialization weights. The learning rate η was selected as 0.005; the model did not converge for higher rates. This value of the learning rate resulted in the fastest learning. The time decay constant for the refractoriness function, τ_R, was selected as 80 ms by trial and error. Other values of τ_R in the range 0 to 100 ms were investigated. Although the convergence was adversely affected for values lower than 20 ms, no consistent pattern was not observed for values greater than that. A sample Multi-SpikeProp convergence curve for MuSpiNN is shown in Fig. 8.

Multi-SpikeProp converges in an average of 78 epochs, which is approximately double the 38 epochs required by SpikeProp (Ghosh-Dastidar & Adeli, 2007). However, Multi-SpikeProp requires only $K = 4$ synapses connecting a presynaptic neuron to a postsynaptic neuron, compared with $K = 16$ synapses required by SpikeProp. A number of synapses lower than 4 appears to be insufficient to model the problem and leads to less consistent convergence. A larger number of synapses, say 6, leads to faster convergence (57 epochs) but the computational effort and time are increased.

Our parametric studies indicate that the delays associated with the synapses should be spread out in the whole range between the earliest input spike (0 ms) and the latest output spike (16 ms). MuSpiNN converges in 78 epochs when delays of 1, 5, 9, and 13 ms are employed for the four synapses but fails to converge when delays of 1, 2, 3, and 4 ms or 1, 3, 5, and 7 ms are employed. In this research, the single-spiking SNN model and SpikeProp were also investigated with the same sets of delays and four synapses for comparison with MuSpiNN. It was observed that SpikeProp also failed to converge when delays of 1, 2, 3, and 4 ms or 1, 3, 5, and 7 ms were employed. The consistency of convergence improved when delays of 1, 5, 9, and 13 ms were employed for the four synapses but the number of convergence epochs was much greater than (>200 epochs) that required by MuSpiNN.

4.4. Fisher iris classification problem

The Fisher iris species classification problem consists of four flower features (petal width, petal length, sepal width, sepal length) and three classes (three species of the iris plant: *Setosa*, *Versicolor*, and *Virginica*) (Fisher, 1936; Newman et al., 1998). The latter two classes are not linearly separable. The dataset consists of 150 samples. Unlike the XOR problem, the inputs are continuous and real valued, and therefore cannot be encoded as easily. A *population encoding* scheme where real-valued inputs are compressed to a small encoding range of discrete spike times appears to represent the input to SNNs more accurately and is therefore used in this research (Bohte et al., 2002; Ghosh-Dastidar & Adeli, 2007). Briefly, each feature is encoded separately by $M > 2$ identically shaped overlapping Gaussian functions centered at M locations. The spread of the Gaussian function is $(1/\gamma)(I_{\max} - I_{\min})/(M-2)$, where $I_{\max}$ and $I_{\min}$ are the maximum and minimum values for the encoded feature, respectively, and γ is an adjustment factor. The center of the ith Gaussian function is located at $I_{\min} + [(2i-3)/2][(I_{\max}-I_{\min})/(M-2)]$, where $i \in \{1, 2, \ldots, M\}$. A single input feature value elicits M different responses in the range 0–1. This range of values is converted linearly to the *encoding range* of 0–10 ms (each response value is multiplied by 10) and rounded to the nearest time step selected for MuSpiNN. The input spike times are obtained by subtracting the result from 10 to encode a higher value with an earlier spike time and vice versa (similar to the XOR problem).

The selection of input encoding parameters and the number of neurons in each layer for MuSpiNN and Multi-SpikeProp parameters is based on previous research using SpikeProp on the iris problem (Bohte et al., 2002; Ghosh-Dastidar & Adeli, 2007). For input encoding using population encoding, a value of $\gamma = 1.5$ yielded the best classification accuracies. The number of input neurons (equal to the number of population encoding Gaussian functions, M) required per input feature is selected as four, resulting in a total of $4M + 1 = 4 \times 4 + 1 = 17$ neurons (including one bias neuron) in the input layer. The number of neurons in the hidden layer is selected as eight. Based on the performance of the single-spiking SNN for the iris problem and MuSpiNN for the XOR problem the time decay constant τ is selected as 11 ms (slightly larger than the encoding interval, i.e., 10 ms), the refractoriness function time decay constant τ_R as 80 ms, and the learning rate η as 0.01. Four synapses are used to connect any presynaptic neuron to a postsynaptic neuron with delays of 1, 5, 9, and 13 ms.

A modular structure comprised of three MuSpiNN modules is employed for solving the 3-class classification problem. Fig. 9 shows the MuSpiNN architectures for (a) the original 3-class classification problem and (b) three 2-class classification problems. In Fig. 9(b), each MuSpiNN module is dedicated to one class and assigned the task of classifying the data as either belonging to that class or not belonging to that class. If the data belong to that class, the MuSpiNN module responds with an output spike at 15 ms, and otherwise at 20 ms. Therefore, the 3-class classification problem in Fig. 9(a) is reduced to three 2-class classification problems in Fig. 9(b). In this research, three identical MuSpiNNs using parameter values discussed earlier are used for the three modules. However, since the three modules are independent of each other, it is not necessary to do so because the suitability of the network architecture is, often, problem-specific. Therefore, each module may be designed individually to maximize the computational efficiency and classification accuracy for the corresponding 2-class classification problem.

One-fifth of the available data (30 training instances or samples) are used for training the network. The MuSpiNN dedicated to identifying a specific class is trained with 10 samples belonging to that class and 20 samples belonging to the other two classes (10 each). A sample Multi-SpikeProp convergence curve for each MuSpiNN module is shown in Fig. 10. It is observed that the three networks converge to an MSE of 0.2 in an average of 11, 60, and 28 epochs. The corresponding classification accuracies of the test data for the three classes are 100%, 94.5%, and 93.1%, which is higher than the 92.7% classification accuracy obtained using the single-spiking SNN with SpikeProp for the 3-class classification problem using similar network parameters and architectures (Ghosh-Dastidar & Adeli, 2007). When half the available data (75 samples) are used to train the network, similar classification accuracies are obtained but the network needs to converge to a much lower MSE of 0.075.

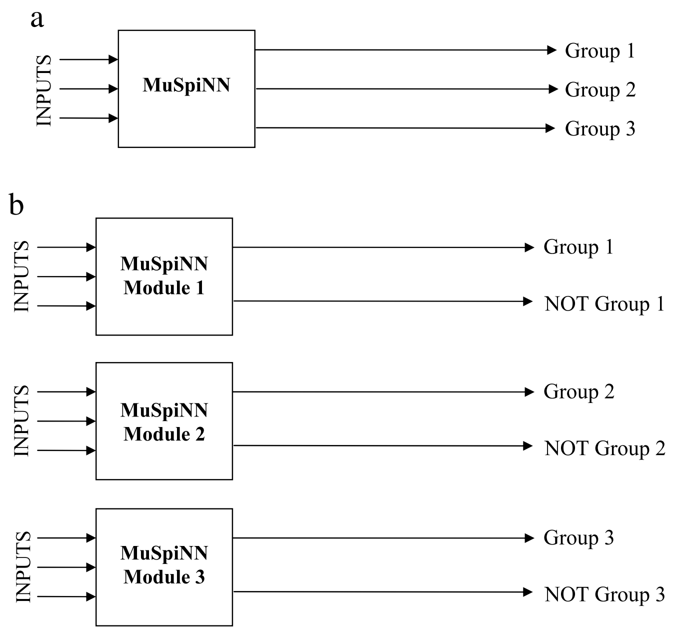


Fig. 9. MuSpiNN architecture for (a) the original 3-class classification problem and (b) three 2-class classification problems.

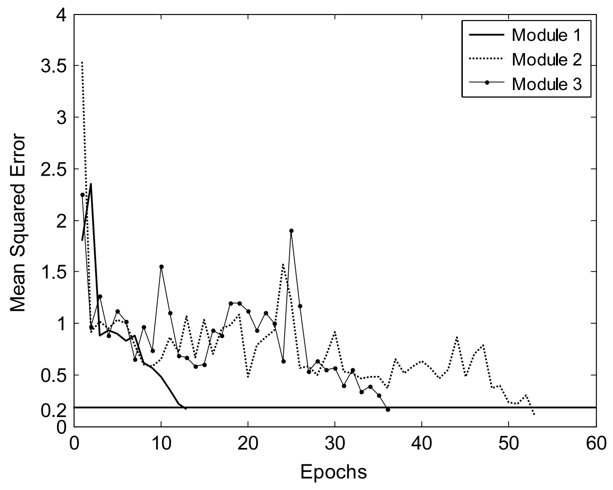


Fig. 10. Sample Multi-SpikeProp convergence curves for MuSpiNN for the iris classification problem.

4.5. EEG classification problem

The EEG data used in this research are collected from three subject groups: (a) healthy subjects, (b) epileptic subjects during a seizure-free interval (interictal EEG), and (c) epileptic subjects during a seizure (ictal EEG) made available online by

Dr. Ralph Andrzejak of the Epilepsy Center at the University of Bonn, Germany (http://www.meb.uni-bonn.de/epileptologie/science/physik/eegdata.html). The type of epilepsy was diagnosed as temporal lobe epilepsy with the epileptogenic focus being the hippocampal formation. Each group contains $K = 100$ single channel scalp EEG segments of 23.6 s duration each sampled at 173.61 Hz (Andrzejak et al., 2001). As such, each data segment contains $N = 4097$ data points collected at intervals of 1/173.61th of a second. Each EEG segment is considered as a separate EEG signal, resulting in a total of 300 EEG signals or EEGs.

Recently, the authors developed a wavelet-chaos methodology for analysis of EEGs and EEG sub-bands to identify potential parameters to be used in seizure and epilepsy detection (Adeli et al., 2007). An important discovery of that research is that a mixed-band feature space comprising the following nine parameters is effective for characterizing the underlying dynamics of the EEG accurately: the standard deviation (STD) computed from the band-limited EEG and *alpha*, *beta*, and *gamma* sub-bands; the correlation dimension computed from the *alpha*, *beta*, and *gamma* sub-bands; and the largest Lyapunov exponent computed from the band-limited EEG and *alpha* sub-band (Adeli et al., 2007; Ghosh-Dastidar et al., 2007, 2008). In this research these nine features are employed in order to accurately classify the EEGs into the three aforementioned classes, none of which are linearly separable. The dataset consists of 300 samples. Similar to the iris problem, the 3-class classification task is divided into three 2-class classification tasks, each of which is solved by a separate dedicated MuSpiNN trained with Multi-SpikeProp. All parameter values are selected to be the same as those for the iris classification problem. The only difference is the number of input features, i.e., nine instead of four. The same population encoding scheme described for the iris classification problem is used with $M = 4$ input neurons for each of the nine input features plus a bias neuron, resulting in a total of $9M + 1 = 9 \times 9 + 1 = 37$ input neurons.

One-tenth of the available dataset (30 training instances or data points) is used for training the network. The MuSpiNN dedicated to identifying a specific class is trained with 10 data points belonging to that class and 20 data points belonging to the other two classes (10 each). A sample Multi-SpikeProp convergence curve for each MuSpiNN module is shown in Fig. 11. It is observed that the three networks converge to an MSE of 0.2 in an average of 53, 36, and 78 epochs. The corresponding classification accuracies of the test data for the three classes are 90.7%, 91.5%, and 94.8%, which is significantly higher than the 82% classification accuracy obtained with SpikeProp for the 3-class classification problem using similar network parameters and architectures (Ghosh-Dastidar & Adeli, 2007). When one-fifth of the available dataset (60 training instances) is used to train the network to a lower MSE of 0.075, similar classification accuracies are obtained for the first two classes but the classification accuracy of the last class increases to 97.1%, a significant improvement, although at additional computational cost.

5. Discussion and concluding remarks

A new multi-spiking neural network (MuSpiNN) and a new training algorithm, Multi-SpikeProp, for training the network have been presented in this paper. The traditional BP-based supervised learning proposed by Rumelhart required a continuous and differentiable activation function for error backpropagation. The lack of such a function for SNNs has led to a strongly-held belief that BP-based learning in SNNs is impossible. In 2002, Bohte et al. (2002) developed a novel learning rule that allowed such an SNN to learn based on adaptation in the timing of *single* spikes. Our novel network and learning algorithm extends that development to a more general case: BP-based learning in neural networks that can communicate via spike trains (i.e. multiple spikes instead of single spikes).

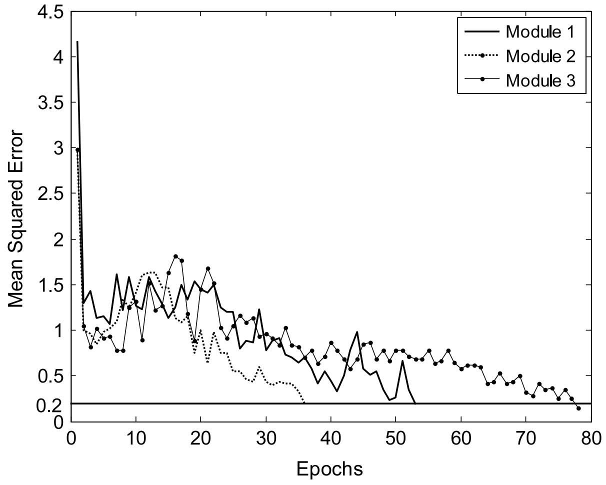


Fig. 11. Sample Multi-SpikeProp convergence curves for MuSpiNN for the EEG classification problem.

The application of MuSpiNN and Multi-SpikeProp demonstrates the training of a neural network that employs neurons based on *an increased degree of* biological plausibility compared to (1) traditional ANN neurons which simulate spike train communication with unrealistic implementations of rate encoding and (2) simplistic single-spiking SNN neurons (Bohte et al., 2002). Although the MuSpiNN model can reconstruct temporal sequences of spikes as the network output, for the sake of simplicity we have restricted the output of only the output layer neuron to a single spike. An output spike train would require the selection of appropriate spike trains for representing the various classes and an appropriate error function. Research into the *appropriateness* of output spike trains and the error function will be another subject of future research towards harnessing the computational power of pulse encoding.

Many detailed mathematical models have been developed to quantitatively characterize neuronal behavior based on detailed modeling of the neuronal membrane potential and ion channel conductances (Ermentrout, 1996; Hille, 1992; Hodgkin & Huxley, 1952; Hoppensteadt & Izhikevich, 1997; Kopell & Ermentrout, 1986; Rinzel & Ermentrout, 1989). Networks of such neuronal models have proved to be very valuable in studying the behavior of biological neural networks, neuronal learning mechanisms such as long-term potentiation and depotentiation, and neurotransmitter-based signaling (Izhikevich, 2007). However, the level of detail, although ideal for reproducing electrophysiological responses accurately, increases the complexity of the models, making them difficult to analyze (Abbott & Kepler, 1990; Kepler, Abbott, & Marder, 1992). This complexity also imposes a significant computational burden for neural network based classification or pattern recognition tasks of this magnitude that employ BP as the learning mechanism.

Another obstacle to the use of these detailed models for such classification and pattern recognition tasks is imposed by the dynamics of the BP algorithm which usually requires a single activation function (representing changes in membrane potential) for backpropagating the error term through the neuron. The detailed models are usually based on multiple differential equations that capture the behavior of different ion channels and currents that affect the membrane potential. It remains to be seen if error backpropagation is even mathematically possible in the face of such complexity. Alternatively, biologically plausible learning mechanisms such as Hebbian learning and Spike Time Dependent Plasticity (STDP) that have been used on such detailed models for demonstrating dynamics of small neuronal networks may need to be adapted for classification and complex pattern recognition tasks.

Spike response models are phenomenological models that are simpler than the detailed models and offer a compromise between computational burden and electrophysiological detail (Ermentrout & Kopell, 1986; Gerstner, 1995; Gerstner & Kistler, 2002; Izhikevich, 2001; Kistler, Gerstner, & van Hemmen, 1997; Rinzel & Ermentrout, 1989). As a result, spike response models are preferred for systemic studies of memory, neural coding, and network dynamics. Following Bohte et al. (2002), in this paper, such a spike response model (originally presented by Gerstner (1995)) is selected by the MuSpiNN neuron for demonstrating that BP-based learning is possible in such a network. Other spike response models (not investigated in the current work) may also be adapted for the Multi-SpikeProp algorithm provided that their activation function can be adapted for error backpropagation.

The biological plausibility of the backpropagation learning algorithm itself has been debated since its conception in the 1980s (Carpenter & Grossberg, 1987; Grossberg, 1988; Mazzoni, Andersen, & Jordan, 1991; Stork, 1989). For the sake of discussion, consider a purely feedforward network where the flow of information is unidirectional. In such a network, implementations of BP (similar to Rumelhart et al. (1986)) by themselves are biologically unrealistic in the *local* sense that they do not directly model strengthening or weakening of a particular synapse based on the activity of the pre- and post-synaptic neurons for that synapse. However, the feedforward network and the BP learning algorithm together can arguably be an abstract representation of a biologically plausible system where the presynaptic neurons in the network get excitatory or inhibitory feedback based on the appropriateness of the final output from the network (i.e. the size of the error function). This abstraction models learning in a *global* sense because this feedback is an assumption that is external to the actual neural network. SpikeProp and Multi-SpikeProp have their origins in Rumelhart's BP concept, and therefore are only biologically plausible in the systemic sense. However, the Multi-SpikeProp learning algorithm results in the adjustment of individual spike times and could, in the future, be integrated with mechanisms such as STDP in order to increase the biological plausibility of the model.

In this paper, the performance of the network and training algorithm was investigated using three different classification problems. It is found that MuSpiNN learns the XOR problem in twice the number of epochs compared with the single-spiking SNN model but requires only one-fourth the number of synapses. MuSpiNN and Multi-SpikeProp were also applied in a modular architecture to solve the 3-class iris and EEG epilepsy and seizure detection problems, resulting in an increase in the classification accuracy compared with the single-spiking SNN and SpikeProp, especially in the case of the EEG problem. SNNs demonstrate great potential for solving complicated time-dependent pattern recognition problems defined by time series because of their inherent dynamic representation.

References

Abbott, L. F., & Kepler, T. B. (1990). Model neurons: From Hodgkin–Huxley to Hopfield. In L. Garrido (Ed.), *Statistical mechanics of neural networks*. Berlin: Springer.

Adeli, H., & Hung, S. L. (1994). An adaptive conjugate gradient learning algorithm for effective training of multilayer neural networks. *Applied Mathematics and Computation*, *62*(1), 81–102.

Adeli, H., & Hung, S. L. (1995). *Machine learning — Neural networks, genetic algorithms, and fuzzy sets*. NY: John Wiley and Sons.

Adeli, H., & Jiang, X. (2003). Neuro-fuzzy logic model for freeway work zone capacity estimation. *Journal of Transportation Engineering*, *129*(5), 484–493.

Adeli, H., & Jiang, X. (2006). Dynamic fuzzy wavelet neural network model for structural system identification. *Journal of Structural Engineering*, *132*(1), 102–111.

Adeli, H., & Karim, A. (2000). Fuzzy-wavelet RBFNN model for freeway incident detection. *Journal of Transportation Engineering*, *126*(6), 464–471.

Adeli, H., Ghosh-Dastidar, S., & Dadmehr, N. (2007). A wavelet-chaos methodology for analysis of EEGs and EEG sub-bands to detect seizure and epilepsy. *IEEE Transactions on Biomedical Engineering*, *54*(2), 205–211.

Adeli, H., & Park, H. S. (1995). Counter propagation neural network in structural engineering. *Journal of Structural Engineering*, *121*(8), 1205–1212.

Andrzejak, R. G., Lehnertz, K., Rieke, C., Mormann, F., David, P., & Elger, C. E. (2001). Indications of non-linear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, *64*(6), 0619071–8.

Bohte, S. M., Kok, J. N., & La Poutré, J. A. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1–4), 17–37.

Booij, O., & Nguyen, H. T. (2005). A gradient descent rule for multiple spiking neurons emitting multiple spikes. *Information Processing Letters*, *95*(6), 552–558.

Bose, N. K., & Liang, P. (1996). *Neural network fundamentals with graphs, algorithms and applications*. NY: McGraw Hill.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, *37*, 54–115.

Dharia, A., & Adeli, H. (2003). Neural network model for rapid forecasting of freeway link travel time. *Engineering Applications of Artificial Intelligence*, *16*(7–8), 607–613.

Ermentrout, G. B. (1996). Type I membranes, phase resetting curves, and synchrony. *Neural Computation*, *8*, 979–1001.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*, 179–188.

Ermentrout, G. B., & Kopell, N. (1986). Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM Journal on Applied Mathematics*, *46*, 233–253.

Gerstner, W. (1995). Time structure of the activity in neural network models. *Physical Review E*, *51*, 738–758.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models. Single neurons, populations, plasticity*. NY: Cambridge University Press.

Ghosh-Dastidar, S., & Adeli, H. (2006). Neural network-wavelet microsimulation model for delay and queue length estimation at freeway work zones. *Journal of Transportation Engineering*, *132*(4), 331–341.

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

Ghosh-Dastidar, S., Adeli, H., & Dadmehr, N. (2007). Mixed-band wavelet-chaos-neural network methodology for epilepsy and epileptic seizure detection. *IEEE Transactions on Biomedical Engineering*, *54*(9), 1545–1551.

Ghosh-Dastidar, S., Adeli, H., & Dadmehr, N. (2008). Principal component analysis-enhanced cosine radial basis function neural network for robust epilepsy and seizure detection. *IEEE Transactions on Biomedical Engineering*, *55*(2), 512–518.

Grossberg, S. (1982). *Studies of mind and brain*. Boston: Reidel Press.

Grossberg, S. (1988). Competitive learning: From interactive activation to adaptive resonance. In D. Waltz, & J. A. Feldman (Eds.), *Connectionist models and their implications: Readings from cognitive science* (pp. 243–283). Norwood, NJ: Ablex.

Grossberg, S., & Versace, M. (2008). Spikes, synchrony, and attentive learning by laminar thalamocortical circuits. *Brain Research*, *1218C*, 278–312.

Hecht-Nielsen, R. (1988). Application of counterpropagation networks. *Neural Networks*, *1*(2), 131–139.

Hille, B. (1992). *Ionic channels of excitable membranes* (2nd ed.). Sunderland, MA: Sinauer Associates.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *Journal of Physiology*, *117*, 500–544.

Hooshdar, S., & Adeli, H. (2004). Toward intelligent variable message signs in freeway work zones: A neural network approach. *Journal of Transportation Engineering*, *130*(1), 83–93.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational properties. *Proceedings of the National Academy of Sciences*, *79*, 2554–2558.

Hoppensteadt, F. C., & Izhikevich, E. M. (1997). *Weakly connected neural networks*. New York: Springer-Verlag.

Hung, S. L, & Adeli, H. (1993). Parallel backpropagation learning algorithms on Cray Y-MP8/864 supercomputer. *Neurocomputing*, *5*(6), 287–302.

Hung, S. L., & Adeli, H. (1994). Object-oriented back propagation and its application to structural design. *Neurocomputing*, *6*(1), 45–55.

Iglesias, J., & Villa, A. E. P. (2008). Emergence of preferred firing sequences in large spiking neural networks during simulated neuronal development. *International Journal of Neural Systems*, *18*(4), 267–277.

Izhikevich, E. M. (2001). Resonate-and-fire neurons. *Neural Networks*, *14*, 883–894.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, *17*, 2443–2452.

Jiang, X., & Adeli, H. (2008). Dynamic fuzzy wavelet neuroemulator for nonlinear control of irregular highrise building structures. *International Journal for Numerical Methods in Engineering*, *74*(7), 1045–1066.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (2000). *Principles of neural science* (4th ed.). NY: McGraw- Hill.

Karim, A., & Adeli, H. (2002). Comparison of the fuzzy – wavelet RBFNN freeway incident detection model with the California algorithm. *Journal of Transportation Engineering*, *128*(1), 21–30.

Karim, A., & Adeli, H. (2003). Radial basis function neural network for work zone capacity and queue estimation. *Journal of Transportation Engineering*, *129*(5), 494–503.

Kasinski, A., & Ponulak, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, *16*(1), 101–113.

Kepler, T. B., Abbott, L. F., & Marder, E. (1992). Reduction of conductance-based neuron models. *Biological Cybernetics*, *66*, 381–387.

Kistler, W. M., Gerstner, W., & van Hemmen, J. L. (1997). Reduction of Hodgkin–Huxley equations to a single-variable threshold model. *Neural Computation*, *9*, 1015–1045.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69.

Kopell, N., & Ermentrout, G. B. (1986). Subcellular oscillations and bursting. *Mathematical Biosciences*, *78*, 265–291.

Liu, H., Wang, X., & Qiang, W. (2007). A fast method for implicit surface reconstruction based on radial basis functions network from 3D scattered points. *International Journal of Neural Systems*, *17*(6), 459–465.

Maass, W. (1996). Lower bounds for the computational power of spiking neural networks. *Neural Computation*, *8*(1), 1–40.

Maass, W. (1997a). Networks of spiking neurons: The third generation of spiking neural network models. *Neural Networks*, *10*(9), 1659–1671.

Maass, W. (1997b). Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems*: *Vol. 9* (pp. 211–217). Cambridge, MA: MIT Press.

Maass, W. (1997c). Fast sigmoidal networks via spiking neurons. *Neural Computation*, *9*(2), 279–304.

Mayorga, R. V., & Carrera, J. (2007). A radial basis function network approach for the computational of inverse continuous time variant functions. *International Journal of Neural Systems*, *17*(3), 149–160.

Mazzoni, P., Andersen, R. A., & Jordan, M. I. (1991). More biologically plausible learning rule than backpropagation applied to a network model of cortical area 7a. *Cerebral Cortex*, *1*, 293–307.

McKennoch, S., Liu, D., & Bushnell, L. G. (2006). Fast modifications of the SpikeProp algorithm. In *Proceedings of the international joint conference on neural networks* (pp. 3970–3977).

Moore, S. C. (2002). *Backpropagation in spiking neural networks. M.S. thesis*. University of Bath.

Natschläger, T., & Ruf, B. (1998). Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems*, *9*(3), 319–332.

Newman, D. J., Hettich, S., Blake, C. L., & Merz, C. J. (1998). *UCI Repository of machine learning databases*. Irvine, CA: Department of Information and Computer Science, University of California. Available: http://www.ics.uci.edu/mlearn/MLRepository.html.

Panakkat, A., & Adeli, H. (2007). Neural network models for earthquake magnitude prediction using multiple seismicity indicators. *International Journal of Neural Systems*, *17*(1), 13–33.

Pedrycz, W., Rai, R., & Zurada, J. (2008). Experience-consistent modeling for radial basis function neural networks. *International Journal of Neural Systems*, *18*(4), 279–292.

Rigatos, G. G. (2008). Adaptive fuzzy control with output feedback for H-infinity tracking of SISI nonlinear systems. *International Journal of Neural Systems*, *18*(4), 305–320.

Rinzel, J., & Ermentrout, G. B. (1989). Analysis of neuronal excitability and oscillations. In C. Koch, & I. Segev (Eds.), *Methods in neuronal modeling* (pp. 135–169). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing*: *Vol. 1* (pp. 318–362). Cambridge, MA: MIT Press.

Sabourin, C., Madani, K., & Bruneau, O. (2007). Autonomous biped Gait pattern based on fuzzy-CMAC neural networks. *Integrated Computer-Aided Engineering*, *14*(2), 173–186.

Schaefer, A. M., & Zimmermann, H. G. (2007). Recurrent neural networks are universal approximators. *International Journal of Neural Systems*, *17*(4), 253–263.

Schrauwen, B., & van Campenhout, J. (2004). Extending SpikeProp. In *Proceedings of the international joint conference on neural networks* (pp. 471–476).

Sejnowski, T. J. (1986). Open questions about computation in the cerebral cortex. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing*: *Vol. 2* (pp. 372–389). Cambridge, MA: MIT Press.

Silva, S. M., & Ruano, A. E. (2005). Application of Levenberg–Marquardt method to the training of spiking neural networks. In *Proceedings of the international conference on neural networks and brain: Vol. 3* (pp. 1354–1358).

Sirca, G., & Adeli, H. (2001). Neural network model for uplift load capacity of metal roof panels. *Journal of Structural Engineering*, *127*(11), 1276–1285.

Stork, D. G. (1989). Is backpropagation biologically plausible? In *Proceedings of the international joint conference on neural networks* (pp. II 241–246). New York: IEEE Press.

Xin, J., & Embrechts, M. J. (2001). Supervised learning with spiking neural networks. In *Proceedings of the international joint conference on neural networks: Vol. 3* (pp. 1772–1777).

Zhang, B., Xu, S., & Li, Y. (2007). Delay-dependent robust exponential stability for uncertain recurrent neural networks with time-varying delays. *International Journal of Neural Systems*, *17*(3), 207–218.