Copyright Notice:

SWAT: A Spiking Neural Network Training Algorithm for Classification Problems

John J. Wade, Liam J. McDaid, Jose A. Santos, and Heather M. Sayers

***Abstract*— This paper presents a synaptic weight association training (SWAT) algorithm for spiking neural networks (SNNs). SWAT merges the Bienenstock–Cooper–Munro (BCM) learning rule with spike timing dependent plasticity (STDP). The STDP/BCM rule yields a unimodal weight distribution where the height of the plasticity window associated with STDP is modulated causing stability after a period of training. The SNN uses a single training neuron in the training phase where data associated with all classes is passed to this neuron. The rule then maps weights to the classifying output neurons to reflect similarities in the data across the classes. The SNN also includes both excitatory and inhibitory facilitating synapses which create a frequency routing capability allowing the information presented to the network to be routed to different hidden layer neurons. A variable neuron threshold level simulates the refractory period. SWAT is initially benchmarked against the nonlinearly separable Iris and Wisconsin Breast Cancer datasets. Results presented show that the proposed training algorithm exhibits a convergence accuracy of 95.5% and 96.2% for the Iris and Wisconsin training sets, respectively, and 95.3% and 96.7% for the testing sets, noise experiments show that SWAT has a good generalization capability. SWAT is also benchmarked using an isolated digit automatic speech recognition (ASR) system where a subset of the TI46 speech corpus is used. Results show that with SWAT as the classifier, the ASR system provides an accuracy of 98.875% for training and 95.25% for testing.**

***Index Terms*— Automatic speech recognition, Bienenstock–Cooper–Munro, dynamic synapses, spike timing dependent plasticity, spiking neural networks.**

I. Introduction

IT IS WIDELY accepted that the brain's computational ability is distributed across a connectionist system of neurons which communicate with each other using a complex web of synaptic connections. Much research has focused on emulating the functionality of the brain by building networks of neurons which can be trained to assign meaning to complex data patterns. However, these networks are limited in their computational ability because the level of understanding of brain functionality is still very much at the embryo stage. Despite this, a range of useful computations are possible with spiking neural networks (SNNs), even with relatively primitive coding and learning techniques. This realization has stimulated significant research on the development and deployment of SNN architectures that can be implemented in either hardware or software and used to inspire new computing paradigms. However, critical to the development of brain inspired computing is the ability of SNNs to learn from experience. This paper presents the learning algorithm, synaptic weight association training (SWAT), which is based on earlier work [1] where spike timing dependent plasticity (STDP) is combined with Bienenstock–Cooper–Munro (BCM) theory to implement a learning rule. The rule overcomes the need to cap synaptic weights because weight stabilization is achieved using the sliding threshold associated with the BCM model. The sliding threshold controls the magnitude of potentiation/depression using the activity history of the postsynaptic neuron. This, in essence, stops the neuron becoming unstable because, as its firing rate approaches its threshold frequency, the plasticity window is adjusted to ensure long term depression (LTD) dominates over long term potentiation (LTP) for subsequent weight updates. A similar approach to this has been reported [2], where the learning rule correlates weight updates with the current synaptic strength. However, this approach virtually eliminated competition between synapses and, to overcome this problem, activity-dependent synaptic scaling was introduced. Furthermore, it has also been shown that the parameters of STDP and BCM can be linked to result in a clearer understanding of how these two forms of plasticity are related [3]. However, this paper does not investigate how BCM can be combined with STDP to remove the need for capping the weights.

The SNN presented in this paper uses the merged STDP/BCM rule to train a network of spiking neurons and its performance is benchmarked using several benchmark problems. The SNN uses a feed-forward loop topology to connect the input to the hidden layer, similar topologies have been reported to exist in the hippocampus to facilitate adaptive filtering. The loop consists of inhibitory and excitatory facilitating synapses where their frequency transition from facilitation to depression is used to filter information, thereby routing it to different neurons in the hidden layer. A single-layer training neuron is used to determine the weights for all output neurons using the above rule. The weights are then mapped to the appropriate output neurons according to the relative occurrence of similar data across the classes.

Section II presents a brief review of the subject domain while Section III describes the SNN topology. This section also presents the STDP/BCM learning rule. Section IV discusses the neuron model used in the SNN, while Section V presents

Manuscript received March 18, 2010; revised July 2, 2010 and August 23, 2010; accepted August 25, 2010. Date of current version November 3, 2010. This work was supported in part by a Vice-Chancellor's Research Scholarship at the University of Ulster, Magee Campus.

The authors are with the Intelligent Systems Research Center, University of Ulster, School of Computing and Intelligent Systems, Derry, Northern Ireland BT48 7JL, U.K. (e-mail: jj.wade@ulster.ac.uk; lj.mcdaid@ulster.ac.uk; ja.santos@ulster.ac.uk; hm.sayers@ulster.ac.uk).

Digital Object Identifier 10.1109/TNN.2010.2074212

experimental results using the Iris and Wisconsin Breast Cancer (WBC) datasets. Section VI discusses the robustness of SWAT, while Section VII benchmarks SWAT using an isolated digit automatic speech recognition (ASR) system. Finally, Section VIII draws conclusions to this paper.

II. Review

To underpin the research presented in this paper, Part A of this section presents a brief review of SNN training algorithms, while Parts B and C review short- and long-term plasticity.

A. Training Algorithms

The first supervised training algorithm for SNNs [4], called SpikeProp, was an adaptation of the gradient-descent-based error-backpropagation method [5]. SpikeProp overcame the problems inherent to SNNs using a gradient-descent approach by allowing each neuron to fire only once. However, since the neurons are allowed to fire only once, the algorithm can only be used in a time-to-first spike coding scheme. Therefore, training the network using patterns which consist of multiple spikes is not feasible. Further to this [6], a learning algorithm similar to the error-backpropagation method was presented, which trained a network of neurons based on their mean output firing rates. Although this method was found to generalize well with several engineering tasks, the network structure and method of input encoding lacked biological realism. In a different approach [7], a probabilistic gradient-descent method was employed that consisted of a biological-like learning window and an injected supervisory teacher signal. Again, this approach utilizes single spikes and therefore cannot learn patterns represented using spike trains.

Evolutionary strategies (ESs) have also been used to train SNNs. One such method [8] used ESs to optimize the weight and delay values of a three-layer fully connected feed-forward network. It outperformed the SpikeProp algorithm when tested on the XOR and Iris benchmark problems. However, since the algorithm was based on an ES, it was extremely time consuming to train. More recently, an online evolving SNN was implemented [9], which adds a new neuron to the output layer for each sample. If the weights of the new neuron are similar to any of the previously added neurons, then the two neurons are merged. In this way, the output is evolved to reflect the spatiotemporal patterns within the data. The main advantage of this algorithm is that training is completed in a single epoch and, whenever new data is presented to the network, the data can be included through another neuron and no retraining is necessary.

From a biological perspective, a training algorithm should update synaptic weights based on the temporal correlation of pre and postsynaptic spikes in keeping with Hebbian theory [10]. A supervised Hebbian learning (SHL) algorithm [11], which included a training signal, was used to ensure that the output neuron would fire at the desired time. It was shown that the algorithm could learn to successfully reproduce the firing patterns of Poisson spike trains. However, even after learning the target pattern, the algorithm still continued to train, and therefore limits must be added to ensure stability [12].

Biological experiments [13], [14] showed that Hebbian correlation takes place in the form of STDP where the exact timing of pre and postsynaptic events are responsible for changes in synaptic efficacy. The remote supervision method [15] is very closely related to the SHL algorithm but manages to counteract the stability problems. In this method, two STDP-like windows are used to adjust the synaptic weights. The first window increases the weight whenever the input is temporally correlated with the desired output (teaching signal). The second decreases the weight based on the correlation of the input against the actual output. In an alternative approach [16], STDP was used with a two-layer network topology interconnected by multiple delay pathways. The weight updates for the network are based on the cross-correlation of data presented to the network and a similar correlation is proposed here, the weights are changed based on the relative occurrence of spikes across all classifications. STDP has also been used to train a SNN to perform a 2-D coordinate transformation, from polar to Cartesian coordinates, to produce a virtual map of haptic inputs [17]. This network was found to be more robust with better noise immunity than its classical counterparts.

An issue when developing a learning algorithm using STDP is that the outputs of the neurons can become increasingly unstable as the network learns [18], [19]. To remove this instability, the weights can be capped, which implies that the maximum value of the weight vector is predetermined and therefore bears no relation to the temporal characteristics of the input data. One such learning algorithm [20] that used the capping of weights between a maximum and minimum value was developed. It was tested on several benchmark problems and produced excellent results. However, this algorithm only used single spike event encoding. Stability issues were also addressed in other work [21] by implementing a 'stop learning' criteria which was based on the weighted sum of the input synapses and the activity of the neuron.

B. Use Dependency

Use dependency depends on the amount of resources available for the generation of postsynaptic potentials (PSPs) [22], where each time a presynaptic spike arrives at the synapse, a fraction of these resources is used. For a regular spike train, the PSPs generated will successively decrease in magnitude until they reach a level known as the stationary amplitude. When the frequency of the input is increased beyond a limit, the magnitude of the stationary amplitude decreases inversely with frequency. Therefore, the limiting frequency restricts the range over which the synapse can effectively transmit information. If all synaptic resources are activated at the same time, a PSP with a magnitude relative to the absolute synaptic strength can be generated, otherwise, the PSP is proportional to the percentage of resources activated. This type of synapse is known as a depressing synapse, the opposite of which is a facilitating synapse [23], [24]. A popular model for a depressing synapse [24] is described by

$$\frac{dx}{dt} = \frac{z}{\tau_{rec}} - U_{SE}\, x(t_{sp} - 0)\delta(t - t_{sp}) \tag{1}$$

$$\frac{dy}{dt} = -\frac{y}{\tau_{in}} + U_{SE}\, x(t_{sp} - 0)\delta(t - t_{sp}) \tag{2}$$

$$\frac{dz}{dt} = \frac{y}{\tau_{in}} - \frac{z}{\tau_{rec}} \tag{3}$$

where x, y, and z are the fractions of resources in the recovered, active, and inactive states of the synapse, respectively. τ_{rec} and τ_{in} are the recovery and inactive state time constants respectively. t_{sp} is the arrival time of an action potential and U_{SE} is the utilization of synaptic efficacy. The current I^i_{syn} that is received by the postsynaptic neuron from synapse i is proportional to the fraction of resources remaining in the active state and is given by

$$I^i_{syn} = A_{SE}\, y_i\,(t) \tag{4}$$

where A_{SE} is the absolute synaptic efficacy and $y_i\,(t)$ is the fraction of resources available to the synapse at time t. Equation (4) suggests that A_{SE} can be interpreted as the weight and a formulation relating this parameter to LTP is used in the proposed learning algorithm. Equations (1)–(3) model the characteristics of a depressing synapse where the level of U_{SE} remains constant. To model a facilitating synapse, U_{SE} is allowed to grow every time there is an input spike at the synapse. U^1_{SE} is therefore a running total of U_{SE} over time. This evolution is given by (5), where U^1_{SE} is the evolving state of U_{SE}, U_{SE} is the step increase of U^1_{SE} for each spike, and τ_{facil} is the relaxation time constant for a facilitating synapse. This model of a facilitating synapse is used throughout SWAT

$$\frac{dU^1_{SE}}{dt} = -\frac{U^1_{SE}}{\tau_{facil}} + U_{SE}\left(1 - U^1_{SE}\right)\delta(t - t_{sp}). \tag{5}$$

C. STDP and Activity Dependency

Traditional STDP-based learning algorithms calculate synaptic weight values associated with LTP using (6) and (7) where $\delta\omega$ is the synaptic weight change, A_+ and A_- are the maximum value of weight potentiation and depression, respectively, τ_+ and τ_- reflect the width of the plasticity window, and Δt is the difference between pre and post firing times [18], [19], [25]. For $\Delta t < 0$ we have

$$\delta\omega = A_+ \exp\left(\frac{\Delta t}{\tau_+}\right) \tag{6}$$

and for $\Delta t \geq 0$

$$\delta\omega = -A_- \exp\left(\frac{-\Delta t}{\tau_-}\right). \tag{7}$$

These equations implement long-term weight potentiation/depression according to the temporal distribution of the pre and postsynaptic spikes. In the next section, we merge LTP/LTD with use dependency, which allows the training algorithm to modify the level of facilitation of the output layer synapses.

In addition to the STDP training rule, biological evidence exists to support the theory that synaptic plasticity depends on the history of activity of the postsynaptic neuron. This led to the development of the BCM model [26] and we use this rule to achieve convergence during training. BCM assumes a synaptic modification threshold θ_m, which leads to either potentiation or depression depending on the postsynaptic activity at any given instant. As the output frequency of the neuron increases beyond θ_m, the synaptic weight is potentiated and, if the frequency is less than θ_m, the weight is depressed. Also, the activation threshold is not fixed but can be changed based on the prior average output activity of the postsynaptic neuron. Therefore, θ_m becomes a sliding value depending on the activity history of the postsynaptic neuron [27], for increasing activity, θ_m adjusts to ensure LTD and vice versa for decreasing activity. This mechanism for negative feedback is used here to modulate the height of the plasticity window associated with STDP, ensuring convergence during training.

The theory of synaptic plasticity underpins the work presented in the next section where we introduce a novel SNN topology which uses facilitating synapses. A STDP/BCM-based learning rule is developed and used to adjust the level of facilitation during the training period.

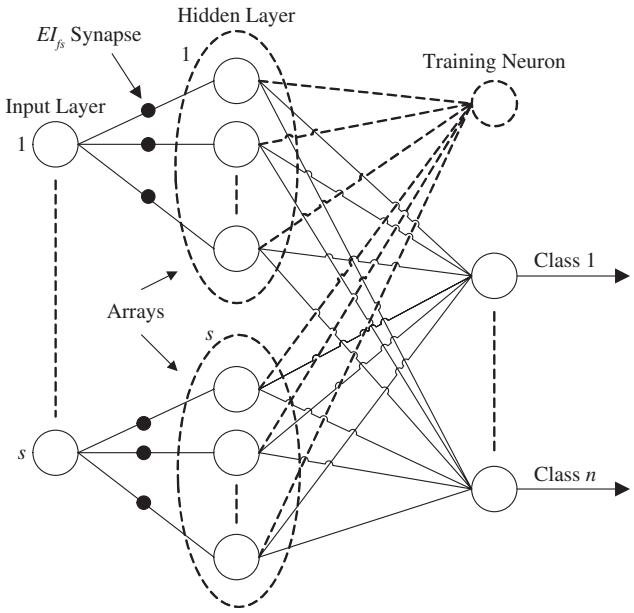

Fig. 1. Network topology consisting of s input and n output neurons with a single training neuron.

III. SNN Topology and Training

Part A of this section proposes a SNN topology that can be trained by a STDP/BCM-based rule to reflect similarities in the training data, across all the classes, in the post trained weight distribution. Part B describes the STDP/BCM learning rule.

A. SNN Topology

The SNN topology shown in Fig. 1 accommodates n data classes where each class can have a different number of samples, m_n, and each sample has s variables. The network therefore has s input and n output neurons and all input data values are mapped to the frequency domain. Hence each data sample will yield an output spike train from each of the s input neurons and we detect these different firing patterns in the hidden layer. A tradeoff between computational effort and

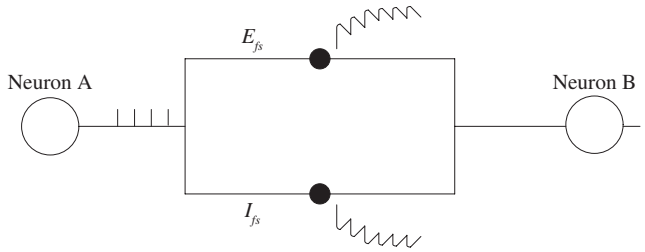

Fig. 2. Feed-forward loop implementation: a spike train at frequency f from neuron A passes to neuron B via the E_{fs} (Excitatory) and I_{fs} (Inhibitory) facilitating synapses.

biological plausibility is made here, where we use linear spike train encoding (constant interspike interval) of the input data. While we accept that biological networks use Poisson-like spike trains, the computational overhead for large networks would be excessive. However, the SNN presented in Fig. 1 can be extended to process information encoded in Poisson spike trains.

The hidden layer comprises identical arrays of neurons whose synapses act as frequency-selective filters, an array is assigned to each of the s input neurons and therefore there are s arrays. If the frequency selectivity of a particular "filter" synapse (its receptive field) associated with a neuron in the array is narrow and centered at f_1 with a pass band $\pm\delta f$, then only that neuron will fire if the spike train frequency from the input neuron is within δf from f_1. However, if the spike train frequency changes by more than δf, then this neuron will cease firing and another neuron in the array is stimulated, and so on. Hence, at least one of the neurons in each array fires continuously for the duration of the stimulus. If the pass band of the entire array, which is the sum of all synaptic filter pass bands in the array, is broad enough, then as the input spike train frequency changes, different neurons in the array are stimulated and therefore one neuron in each of the s arrays will fire for any of the samples. This process segregates the input data according to frequency and is a key processing step in the proposed topology.

To develop the ability to filter presynaptic inputs, the synapse is required to be sensitive to a band of frequencies. To achieve this, we propose a parallel combination of facilitating synapses consisting of an excitatory (E_{fs}) and an inhibitory (I_{fs}) synapse, as shown in Fig. 2. We propose this because the topology is very similar to feed-forward loops commonly found in the hippocampus [28] where excitatory and inhibitory synapses can selectively amplify high-frequency bursts, CA3 cells branch to have an excitatory input to CA1 cells and also activate inhibitory inter-neurons that provide a feed-forward inhibition onto the same CA1 cells. The feed-forward inhibition is locked in with the excitatory input from the CA3 and this locked property results in an excitatory/inhibitory postsynaptic response sequence in the CA1 cells that acts like an adaptive filter. The filtering occurs because both E_{fs} and I_{fs} switch from facilitation to depression where the switch has either a temporal dependency (in some cases) or a frequency dependency [28]–[31]. In the present case we investigate the latter.

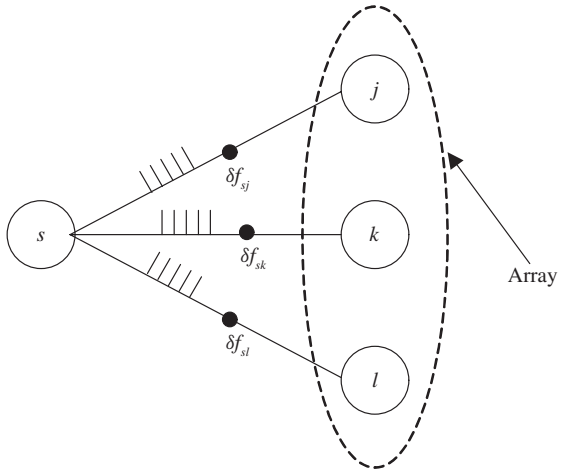

Fig. 3. SNN fragment containing an input neuron s, which is connected to each of the j, k, and l neurons using the feed-forward loop modeled as an EI_{fs} synapse.

Consider the case where the transition from facilitation to depression (switching) occurs at f_1 for I_{fs} and f_2 for E_{fs}, where $f_1 < f_2$. In the situation where $f < f_1$, the postsynaptic responses from E_{fs} cancel with that from I_{fs} and no facilitation occurs. With $f_1 < f < f_2$, I_{fs} is depressing with E_{fs} facilitating, resulting in a facilitation postsynaptic response. For $f > f_2$, both I_{fs} and E_{fs} are depressing and consequently there is no postsynaptic response. Hence the feed-forward loop shown in Fig. 2 will only route information to neuron B if the spike train frequency f of neuron A lies between f_1 and f_2. Essentially, our proposed combination of synapses in Fig. 2 (termed EI_{fs}) implements frequency filtering with a pass band of $\delta f = f_2 - f_1$. In our implementation, both E_{fs} and I_{fs} are modeled using (1)–(5), when f is below f_1 or f_2, (1)–(5) are used, and above f_1 or f_2, (1)–(4) are used. Experimental evidence has been reported [32] that shows synaptic switching in the hippocampus occurs at 1 Hz, while other publications [29]–[31] have reported the switching from facilitation to depression is possible over the frequency range 10–100 Hz. Therefore, we can make the approximation that this switching activity occurs at different frequencies for different synapses and we use this concept to implement frequency filtering as follows.

Consider the network fragment in Fig. 3, where neuron s is one of the input neurons and is presynaptic to an array consisting of the postsynaptic neurons j, k, and l. Each pathway contains a feed-forward loop, modeled as an EI_{fs} synapse that facilitates over a frequency band: pathways $s-j$, $s-k$, and $s-l$ facilitate over δf_{sj}, δf_{sk}, and δf_{sl}, respectively. Therefore, spike trains from neuron s will be routed to one of the j, k, or l neurons depending on its frequency and it is this routing capability that is central to the computational ability of the proposed SNN. With reference to Fig. 1, each of the s input neurons is connected to an array of hidden layer neurons where every neuron in the array is sensitive to a frequency band. Consequently, for each of the samples we have at least one neuron from each array firing and these firing patterns stimulate the training neuron in the output layer of Fig. 1.

B. Learning Rule

A training algorithm, specific to the SNN topology presented in Fig. 1, was previously presented [33], [34], where we combined the sliding threshold of BCM with STDP. The height of the plasticity window of all synapses connected to the training neuron was modulated according to

$$\theta_m(c_t) = \left(\frac{c_t}{c_o}\right)^{\alpha} c_t \tag{8}$$

where c_t is the average firing frequency of the training neuron t and c_o and α are constants, which dictate the dependency of $\theta_m(c_t)$ on c_t. The height of the plasticity window is related to $\theta_m(c_t)$ by

$$A_+(\theta_m) = A_p \frac{1}{1+\theta_m(c_t)} \tag{9}$$

where $A_+(\theta_m)$ replaces A_+ in (6) for all synapses connected to the training neuron and approximates to A_p for $\theta_m(c_t) \approx 0$. A_p is the maximum possible amplitude of the potentiation window. The magnitude of the depression window is found from

$$A_-(\theta_m) = A_p - A_+(\theta_m) \tag{10}$$

where $A_-(\theta_m)$ replaces A_- in (7) for all synapses connected to the training neuron, and A_p is the maximum possible amplitude of the depression window. Note that, if c_t is much greater than c_o, then $A_+(\theta_m)$ will approach zero and the height of the depression window $A_-(\theta_m)$ will approach A_p, thus leading to LTD. Alternatively, if c_t is much less than c_o, then $A_+(\theta_m)$ will tend toward A_p and LTP occurs. This competition between LTP and LTD implements the BCM rule, yielding a mechanism for negative feedback that stabilizes the output firing frequency after a period of training. However a bimodal weight distribution will result from this rule, whereas a unimodal weight distribution is characteristic of real biological systems [2]. This bimodal distribution can be understood if we consider the case where a neuron's output frequency has reached the desired rate, which results in the STDP modulation stopping. One input may continue to further potentiate while the other continues to depress. The resulting updates lead to the same approximate output frequency, and therefore there is no change to the STDP window. This continues until one input is completely depressed while the other accounts for the total output.

In this paper, a modified form of (8) is used. Given that the output firing frequency of the training neuron is related to the aggregate of all the synaptic responses, then the weight values of each synapse contributes to the output activity of the training neuron. Consequently, each synapse can be assigned a $\theta_m(\omega_t)$ value based on its weight according to

$$\theta_m(\omega_t) = \left(\frac{\omega_t}{c_o}\right)^{\alpha} \omega_t \tag{11}$$

where $\theta_m(\omega_t)$ is again used to adjust the STDP window for the synapse and replaces $\theta_m(c_t)$ in (9), ω_t is the absolute synaptic efficacy, and c_o and $\alpha(=2)$ are constants used to vary the dependency of $\theta_m(\omega_t)$ on ω_t. Note that c_o was found experimentally and this will be discussed in Section VI. Equation (11) is preferred over (8) because it not only implements BCM but also results in a unimodal distribution of weights.

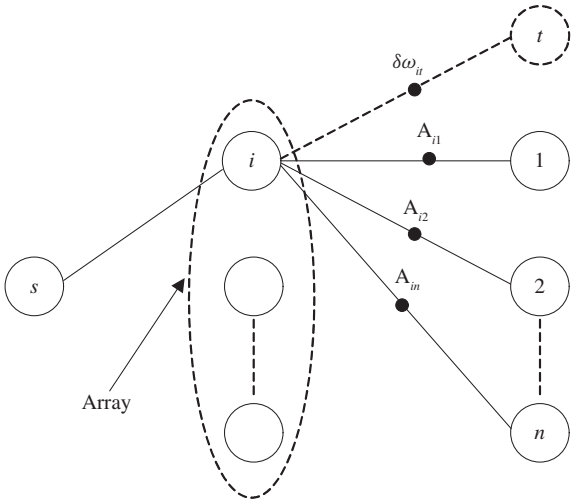

Fig. 4. Two-layer SNN fragment with a training neuron t and n classifying neurons.

An important aspect of any classification network is its ability to select attributes of the input data that are more strongly associated with a particular class. A localized training rule such as STDP lacks the ability to reflect similarities in the data, across all classes, in the weights of a network. Consequently, our approach is to extend the conventional STDP rule to assign absolute synaptic efficacy to classification of neurons based on the relative occurrence of similar data patterns across all the classes. Thus, we provide a global learning mechanism that reflects the similarities in the data and reintroduces synaptic competition which is lost as a result of the unimodal STDP/BCM rule. Such an association was demonstrated to be effective in classification problems in an earlier publication [16], and we therefore adopt this approach here. Essentially, we are proposing that the relative occurrence is found by the training neuron and subsequently mapped to the classifying neuron in Fig. 1, neuron-to-neuron signaling via astrocytes [35]–[39] could be the mechanism by which this mapping process occurs.

We now develop a STDP-based training rule that operates on all synapses associated with the training neuron during the training period only. The resulting weight updates calculated by this rule are continually mapped to the classifying neurons, the proportion of the weight update mapped to each classifying neuron reflects the relative occurrence of similar data patterns across classes.

Consider a fragment of the SNN in Fig. 1, shown in Fig. 4, where n output neurons $(1, 2, ..., n)$ classify data from neuron i, neuron i is any one of the neurons in the hidden layer arrays of Fig. 1. This neuron is stimulated by one of the s input neurons and will repeatedly fire at a frequency f for the duration of the applied stimulus, neuron i will also fire a frequency burst at f for similar inputs from all classes. Assume that there are p samples at frequency f in class 1, q samples at f in 2, and r samples again at f in class n. The

p samples of class 1 will yield a total weight update over 1 epoch, $\delta\omega_{i1}$, given by

$$\delta\omega_{i1} = \sum_{0}^{p} \delta\omega \tag{12}$$

where $\delta\omega$ is given by (6) and (7). For each output spike of neuron t, we only consider the temporal difference between the nearest before and after input spike [3] and $\delta\omega$ is the average weight obtained from the sum of the all potentiation (before) and depression (after) weight changes. Convergence is assured because of the combined STDP/BCM rule where the plasticity window shifts vertically with increasing output activity of neuron t until the level of potentiation and depression associated with each output spike cancels with $\delta\omega \sim 0$ (9), (10), and (11) are used in this calculation. Similarly, for class 2 we can write

$$\delta\omega_{i2} = \sum_{0}^{q} \delta\omega \tag{13}$$

and for class n we can write

$$\delta\omega_{in} = \sum_{0}^{r} \delta\omega. \tag{14}$$

With reference to (14), we note that the potentiating $\delta\omega_{in}$, after a period of training, will reflect the number of samples at frequency f in class n training data. This is also true of (12) and (13), and consequently the post-trained weight distribution across all classifying neurons will be a unimodal representation of similar data patterns in the training samples. If we assume unequal sample sizes for each class, where classes 1, 2, and n have samples sizes of m_1, m_2, and m_n, then the probability P_n that a particular sample belongs to class n is given by

$$P_n = \frac{\dfrac{\delta\omega_{in}}{m_n}}{\dfrac{\delta\omega_{i1}}{m_1} + \dfrac{\delta\omega_{i2}}{m_2} + \cdots\cdots + \dfrac{\delta\omega_{in}}{m_n}}. \tag{15}$$

The total weight increase across all classes ($\delta\omega_{it}$) in one epoch is given by

$$\delta\omega_{it} = \sum_{1}^{n} \delta\omega_{in}. \tag{16}$$

Finally, referring to (4), (15), and (16), we can write a generic learning rule for the synapse associated with the n^{th} output neuron A_{in} as

$$A_{in} = K\omega_{in} = K\left(\omega_{in}\left(pre\right) + P_n\delta\omega_{it}\right) \tag{17}$$

where $\omega_{in}(pre)$ is the existing weight value and $K\ \left(= 10^{-12}\right)$ is a scaling factor, A_{in} is assigned to the synapse of neuron n through the mapping process. Note that for the situation where all classes have equal sample sizes ($m_1 = m_2 = \cdots\cdots = m_n$), the rule in (17) reduces to

$$A_{in} = K\left(\omega\left(pre\right) + \delta\omega_{in}\right). \tag{18}$$

It is important to note that the training neuron is used to ensure that the relative occurrence of samples at frequency f across the classes is accurately reflected in the absolute synaptic efficacy for each epoch. This is ensured because, during any training epoch, the output firing frequency of the training neuron will be the same for all classes, as will all other parameters. This is not the case for the neurons 1 to n since their weights will be different after a period of training, as predicted by (17) and (18). Therefore, our training neuron uses the STDP/BCM rule to associate weight updates with the number of times a particular neuron in any of the arrays fires for each class and subsequently assigns an absolute synaptic efficacy, through the mapping process described by (17) and (18), to each of the synapses of the associated classifying neuron to reflect that association. Hence a high occurrence of particular input data values in one class will be reflected, through the mapping, in the PSP of the associated classifying neuron and consequently it will fire faster. The training neuron also ensures that, after a period of training, the network will reach a global minimum and therefore overtraining is avoided. When all synapses reach this condition, the average output frequency of the training neuron will remain relatively constant with $\delta\omega_{it} \to 0$ because $\delta\omega \to 0$. Hence, stability of the output frequency is used as the stopping condition for training. A further point to note is that, because the proposed algorithm correlates commonality across the input data, using the probability function in (15), then for data with a small sample size (e.g., XOR problem) the accuracy of the classification reduces. Equation (15) reflects the probability that a particular sample belongs to a class and therefore there must be a sufficiently large sample population for this equation to become statistically meaningful.

C. Training Algorithm

We now discuss the training procedure for an n class problem where each class has m_n data patterns.

Each epoch of training starts by calculating the position of the STDP window, using (9)–(11). Subsequently the entire training data m_n for each of the n classes is passed to the SNN, and for each synapse associated with the training neuron, the total weight value $\delta\omega_{it}$ is calculated using (6), (7), (14), and (16), note that each sample is presented for a duration of 2 s. At the end of the epoch, these values are used to find A_{in}, which is then mapped to the synapses associated with the output neuron.

At the end of each epoch, we calculate both the classification accuracy (CA) using the training data and the average firing activity of the training neuron (c_t), where the latter is used to evaluate the stopping criteria for SWAT. Once this condition is met, the weights are fixed and the testing data is applied. The read-out function used in this paper is similar to that presented in [21], where output neurons were grouped into classes and classification is assigned by using a majority vote system. However, since we employ only one neuron per output class, the output neuron with the highest firing frequency represents the class association.

While this approach to training does not suffer from overtraining because BCM acts to stabilize the output frequency of t after a period of training, it is necessary to apply a stopping criteria for the purpose of testing the network. This criteria is

based on the average firing frequency c_t associated with the training neuron. Once c_t has stabilized, the weights will also have stabilized and the training is then stopped. We apply this condition by continually monitoring c_t and, when it remains constant to within ± 0.2 Hz for 100 consecutive epochs, the training is stopped and the weights are then fixed.

IV. Neuron Model

The neuron model used in this paper is the leaky integrate and fire model [40], the output of which is given by

$$\tau_m \frac{dv}{dt} = -v(t) + R_m I_{tot}(t) \tag{19}$$

where τ_m is the membrane time constant, v is the membrane potential, R_m is the membrane resistance, and I_{tot} is the total current generated by all synapses connected to the neuron. This passive membrane model was chosen because it provides a good model for the biological neuron with minimal computational overhead. In this paper, $\tau_m = 60\ ms$ and $R_m = 1000\ M\Omega$ [24]. The firing threshold of the neuron is implemented using [41]

$$Vth_{New} = m \times Vth_{Orig} \exp\left(-\frac{t - t_i}{\tau_{decay}}\right) \tag{20}$$

where Vth_{New} is the new value of the firing threshold after the neuron has fired, m is a multiplication factor, Vth_{Orig} is the original firing threshold, t is the current time, t_i is the time the neuron fired, and τ_{decay} sets the rate at which the threshold decays back to its equilibrium value. In the present case $\tau_{decay} = 20\ ms$, $Vth_{Orig} = 9\ mV$ [42] and m was set to 11.1, this sets the threshold to 100 mV after firing. When the neuron fires at time t_i, the threshold is multiplied m times and then decays back to its original value. This allows the absolute and relative refractory periods to be modeled without the information presented by the PSP being lost [41], [43]. Using dynamic synapses and variable thresholds in this way also approximates the output of a biological neuron under constant-current experiments [44].

V. Benchmarking

In this section, we use SWAT to classify both the Iris and WBC datasets to facilitate benchmarking against existing training algorithms.

A. Iris Dataset

The three-class Iris dataset problem ($n = 3$) was initially used to benchmark SWAT. The three different classes represent the different species of the Iris plant, i.e., *Iris Setosa Canadenisis* (Class A), *Iris Veriscolor* (Class B), and *Iris Virginica* (Class C). The full dataset consists of a total of 150 samples, 50 for each species [45]. Each sample contains four attributes: sepal length, sepal width, petal length, and petal width.

Training and testing was achieved in these experiments by the process described in Section III-C. To encode the data while minimizing the number of neurons in each array in the hidden layer, the squared cosine method [20] was used. Using this method, the complete attribute range is multiplied by a factor of 10, so that the new attributes are now integer values. Each attribute is then passed through four squared cosine functions ($s = 16$) with centers at 1, 20, 40, and 60 Hz. The width of each function is 40 Hz with an output range of 25–66.6 Hz or an inter spike interval (ISI) of 15–40 ms. The 150 samples were split randomly into five groups of 30, with each group containing 10 samples from each species, $m = 10$. Fivefold cross-validation was then carried out where training is performed five times, and each run uses four different groups for training and the remaining group for testing. The mean results for training and testing were then calculated.

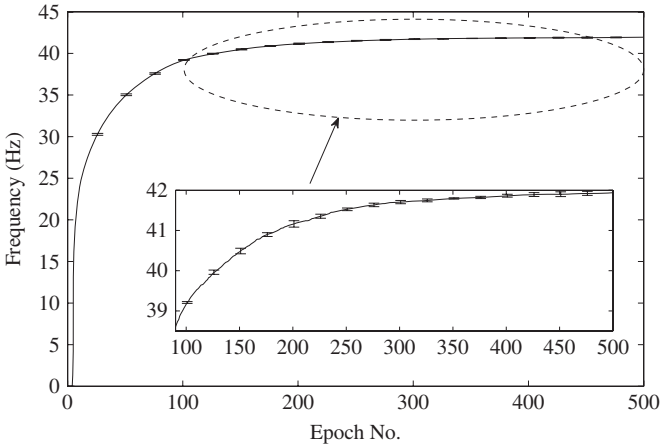

Fig. 5. Average output frequency of the training neuron vs. epoch for the Iris data over the five training sets. The error bars are shown in more detail on the inset, which reflects a standard deviation over epochs 100 to 500.

The SNN topology in Fig. 1 was used in the following experiments, where each input neuron was connected to an array of 13 hidden layer neurons with associated filtering EI_{fs}, each synapse is sensitive to a band-pass equivalent to an ISI of 2 ms and hence the requirement for 13 neurons to account for a total frequency range of 25–66 Hz or an ISI from 40 to 15 ms. The total number of neurons in the hidden layer is therefore 208 (13 array neurons $\times$ 16 input neurons). The "weights" for the hidden layer neurons are fixed so that each neuron in the hidden layer fires at approximately 40 Hz for the corresponding input range, these values were found experimentally. The hidden layer is then fully connected to a single training neuron whose synaptic weight updates are governed by the STDP/BCM rule. In addition, weights associated with the output layer neurons were initialized to a constant value (1 pA). This is necessary to ensure that mapping similarities in the input data, calculated by the STDP/BCM rule, are not offset by an uneven initial weight distribution. This initialization is similar to other work [46] and does not degrade the network's ability to generalize. Note that in this experiment the maximum height of the plasticity window A_p was 0.5, which is equivalent to δA_{SE} of 0.5 pA ($K = 10^{-12}$), and c_o was 4000.

Fig. 5 shows an average of the training neuron's output activity over the five training runs along with the standard deviation of the data every 25 epochs. From this, it can be seen that the firing rate of the training neuron stabilized at approximately epoch 500.

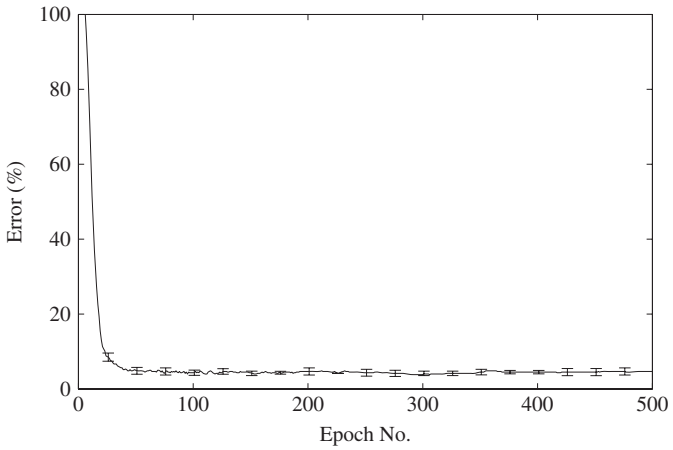

Fig. 6. Convergance plot for Iris training data for all five folds.

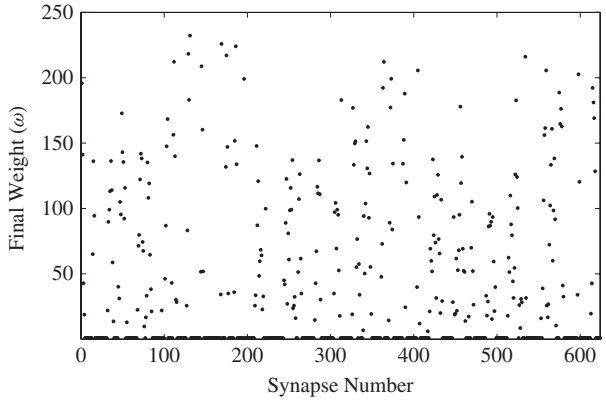

Fig. 7. Post-trained weight distribution for the Iris dataset.

In these experiments, the classification is determined by the output neuron with the highest firing rate, the firing frequency for each output neuron is determined by counting the number of spikes over a fixed duration. Fig. 6 shows the average percentage error and standard deviation against the number of epochs over the five sets of training data, where, at approximately epoch 500, the training reaches and stabilizes at a minimum. For this dataset, the CA for the training data is 95.5% and for testing data it is 95.3%. Also note that, after a minimum error has been reached and the network has stabilized, there is no overtraining and the error remains at a global minimum.

Fig. 7 shows the post-training weight distribution for all synapses associated with the output neurons. It can clearly be seen that a unimodal weight distribution results from the training and that a significant proportion of the weights are untrained. Therefore the post-trained SNN is only partially connected between the hidden and output layers. Also, from the weight distribution it is clear that the mapping process associated with the training neuron causes competition between synapses.

Table I compares the convergence accuracy of SWAT against existing algorithms for the Iris dataset [20]. MatlabBP and MatlabLM are built-in functions used by MATLAB that implement the backpropagation and Levenberg–Marquardt training algorithms, respectively. It should be noted that, while the training accuracy falls slightly short of other approaches, the test data is comparable, which suggests that SWAT can generalize better.

TABLE I
COMPARISON OF TRAINING ALGORITHM: RESULTS FOR IRIS DATASET

Algorithm	Training set	Testing set
SpikeProp	$97.4\% \pm 0.1$	$96.1\% \pm 0.1$
MatlabBP	$98.2\% \pm 0.9$	$95.5\% \pm 2.0$
MatlabLM	$99.0\% \pm 0.1$	$95.7\% \pm 0.1$
Weight Limit Learning	100%	96.6%
SWAT	$95.5\% \pm 0.6$	$95.3\% \pm 3.6$

B. WBC Dataset

The two-class ($n = 2$) WBC dataset was also used to benchmark SWAT. This dataset uses breast cytology gained by fine needle aspirations obtained from the University of Wisconsin Hospital and classifies the results into benign or malignant tumors [47]. The complete dataset contains 699 samples, however, 16 samples have missing data so these have been removed for these experiments. The remaining 683 comprise of 444 and 239 benign and malignant tumors, respectively [48]. These samples are again randomly split into five groups for fivefold cross-validation training and testing. To achieve five equally sized groups, four random samples have been removed from each class and therefore each group contains 88 benign and 47 malignant samples. Because of the unequal class size, the rule described by (17) is applied to train this dataset. Each sample has nine attributes, which measure different features of the cytology with integer values in the range 1–10. Therefore, it is not necessary to use the squared cosine method to encode the data as in the previous experiments. Each value can be mapped directly to a linear spike train in the range of 34–50 Hz (29–20 ms ISI), where a value of 1 is assigned to the lowest frequency and a value of 10 to the highest.

The same network topology and training regime as in the last experiments were used. The input layer comprises nine neurons, one for each attribute. To ensure consistency between experiments, the same number and type of hidden layer neurons were used in each array, hence the hidden layer contains 117 neurons (13 array neurons $\times$ 9 input neurons). Note that, in this experiment, c_o remains the same as in the previous experiment, while the maximum height of the plasticity window A_p is 0.1, which is equivalent to δA_{SE} of 0.1 pA. This reduction in δA_{SE} is due to the greater number of samples, compared to the Iris dataset, used for training, the greater number of training samples per epoch correlates with the relative firing rate of the classifying neurons after training. Results gained from these experiments showed that, again, after a period of training, the firing frequency of the training neuron stabilizes at epoch 500, as shown in Fig. 8, which presents an average of the training neuron activity and standard deviation over the five training sets.

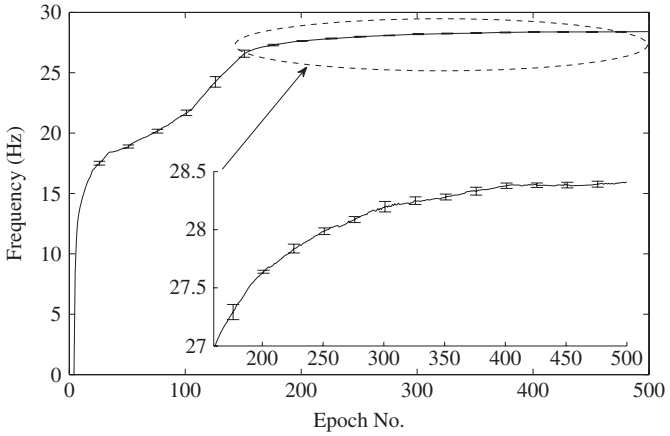

Fig. 8. Average output frequency of the training neuron vs. epoch for the WBC dataset over the five training sets. Error bars on the inset show the standard deviation over epochs 160 to 500.

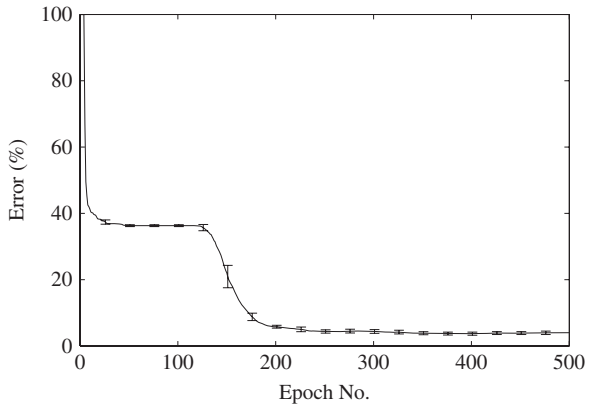

Fig. 9. Convergence accuracy (CA) for WBC training data for all five sets.

Fig. 9 shows the average percentage error and standard deviation against the number of epochs over the five sets of training data where, again, the training reaches and stabilizes at a minimum for each of the training runs. Interestingly, the error reaches a plateau at approximately epoch 50, but is driven past it at around epoch 110. This is caused by attributes of the input data that have a high occurrence across the classes. These attributes initially cause a strong potentiation of some of the weights assigned to the classifying neurons but, because of the negative feedback implemented using (11), these weight values saturate. However, attributes that occur less frequently are still causing other weights to potentiate much more slowly, and so c_t in Fig. 9 begins to move toward a global minimum again, note that the reduction in the rate of change of c_t in Fig. 8 corresponds with the plateau of Fig. 9.

Fig. 10 shows the post-training weight distribution for all synapses associated with the output neurons, and again a unimodal weight distribution results from the training. Note also that a significant portion of the weights is untrained and consequently the post-trained SNN is only partially connected between the hidden and output layers. Also, the weight distribution indicates that the mapping process associated with the training neuron causes competition between synapses.

Table II compares the convergence accuracy of SWAT against existing algorithms for the WBC dataset [20]. For this dataset, the test data accuracy is comparable to that of the other approaches.

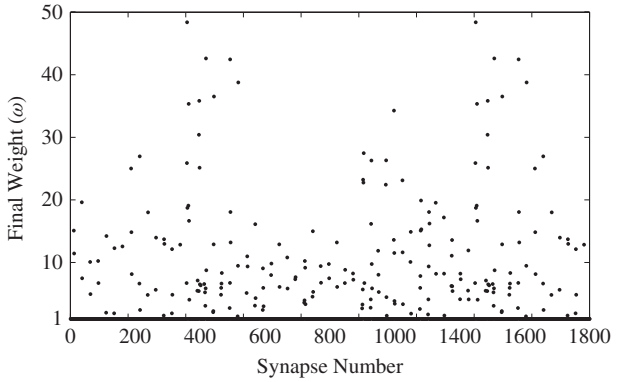

Fig. 10. Post-trained weight distribution for the WBC dataset.

TABLE II
COMPARISON OF TRAINING ALGORITHM: RESULTS FOR WBC DATASET

Algorithm	Training set	Testing set
SpikeProp	$97.6\% \pm 0.2$	$97.0\% \pm 0.6$
MatlabBP	$98.1\% \pm 0.4$	$96.3\% \pm 0.2$
MatlabLM	$97.7\% \pm 0.3$	$96.7\% \pm 0.6$
Weight limit learning	100%	97.9%
SWAT	$96.2\% \pm 0.4$	$96.7\% \pm 2.3$

Although the training for both the Iris and WBC experiments stabilized at 500 epochs, the experiments ran for more than twice that number of epochs and it was observed that the CA remained stable at around its minimum value. This is due to the negative feedback mechanism associated with BCM which stabilized the weights at a level that reflected similarities in the input data, across all classes, in the absolute synaptic efficacy. Therefore, no overtraining is possible with SWAT.

VI. ROBUSTNESS

The robustness of SWAT was explored by analyzing the convergence accuracy across different plasticity windows. For this experiment, three different shaped windows were used. The first window (Standard) was symmetrical about the time axis and was based on (9) and (10). where τ_+ and τ_- were set to 15 ms. The second window (Bi and Poo) was asymmetrical about the time axis and can also be described by (9) and (10), where the parameters τ_+ and τ_- were set to 16.8 and 33.7 ms, respectively [13], [25]. The final window (Gerstner) [49] was an asymmetrically skewed "sinusoidal" shaped window where τ_+ and τ_- were set to 12 and 24 ms, respectively. Both datasets were trained using the three plasticity windows, and the classifying accuracy is shown in Table III. It can clearly be seen that results are very similar for the three windows, however, the Bi and Poo shaped window performed the best during testing and was used for the generation of results in Tables I and II.

Further experiments were also carried out to explore how the variance of c_o and A_p would affect the performance of

TABLE III
COMPARISON OF STDP WINDOWS FOR IRIS AND WBC DATA SETS

Window	Training set		Testing set	
	Iris	WBC	Iris	WBC
Standard	95.5%	96.6%	94.7%	96.4%
Bi and Poo	95.5%	96.2%	95.3%	96.7%
Gerstner	96%	96.2%	94.7%	96.64%

SWAT. It was found that, if c_o is reduced significantly, then CA is also reduced because the maximum weight values tend to stabilize at lower values and the resulting PSP, associated with the output classifying neurons, fails to exceed the threshold level for some test samples. Alternately, an upper limit on c_o is set by the maximum firing rate (set by the refractory period) of the output classifying neurons, a more realistic upper limit on c_o is the convergence time, which for our network was limited to approximately 500 epochs. A similar observation was reported elsewhere [27]. The lower limit on A_p is set by the training time, but care must be exercised when choosing the upper limit to avoid oscillations in c_t and consequently the weight distribution. This effect is very similar to that reported for the backpropagation algorithm when the learning rate is too large [50]. In this paper, the optimum values for A_p and c_o reported earlier were found experimentally.

SWATs immunity to noise was also investigated, whereby varying levels of additive white Gaussian noise were added to the Iris and WBC datasets and the average CA was measured for each dataset, note that the SNN was trained on the "noiseless" data as explained in Section V and the resulting post-trained topology and weights were used to classify the noisy data. Fig. 11 presents the results of these experiments, where it can be seen that the CA remains above 90% up to a signal-to-noise ratio (SNR) of 18 dB for the Iris dataset and up to a SNR of 8 dB for the WBC dataset. The difference in noise robustness between the datasets is a result of the preprocessing encoding scheme used. In the Iris dataset, four squared cosine filters are used on each input, while for the WBC data no preprocessing was used.

VII. ASR

In this section, SWAT is benchmarked in an ASR system. The dataset used for this application is a subset of the TI46 speech corpus which contains isolated spoken words collected by Texas Instruments in 1980. The complete TI46 corpus contains 16 speakers: 8 male and 8 female, and each speaker utters 46 words, where each utterance is repeated 26 times [51]. In these experiments, we use a subset of this corpus consisting of the digits 0–9, and all eight speakers are used with five utterances of each digit. This yields a total dataset of 400 samples (10 digits $\times$ 8 speakers $\times$ 5 utterances).

A. ASR Front End

We developed a signal analysis (front end) stage for the ASR system. This contained elements that allowed us to extract features from the speech signal and convert them to a series of inputs for processing by SWAT.

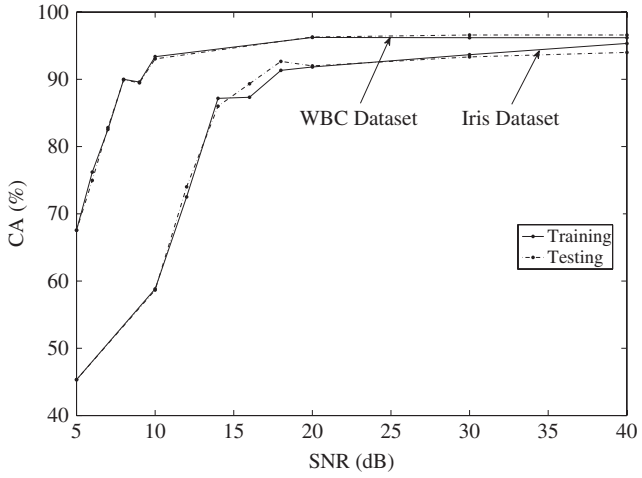

Fig. 11. SWAT noise immunity for the post-training SNN parameters from Section V.

Fig. 12. ASR front end used for the generation of input spike trains from the speech data.

Fig. 12 illustrates the front end which is described as follows.

1) **Endpoint detection**. An important feature of any recognition system is the ability to locate the start and end of a word. This isolates the signal from the silence that is located before and after the utterance and reduces the amount of processing time required, the sample lengths are shortened dramatically. This is achieved by using the endpoint detection algorithm implemented by [52].
2) **Pre-emphasis**. Within speech signals, there is a typical roll off of -6 dB/octave. To eliminate this, the signal is processed by a high-pass first-order FIR filter with a 6 dB/octave gain [53] and is described as

$$y(n) = x(n) - \alpha x(n-1) \tag{21}$$

where $x(n)$ is the original signal, $y(n)$ is the pre-emphasized signal, and α is the filter coefficient. In these experiments, $\alpha = 0.94$.
3) **Windowing**. The pre-emphasized signal is now divided into quasi-stationary frames. The normal technique is to use a window with a fixed width and crossover. However, the speech samples are of various lengths, and this method would result in a wide ranging number of windows, depending on the sample length, since the topology of SWAT has a fixed number of input neurons, this is undesirable as there must be an input for every extracted coefficient from each window. To combat this, we used a variable window width of N samples given by

$$N = \frac{f_s \cdot s_l}{W_f} \tag{22}$$

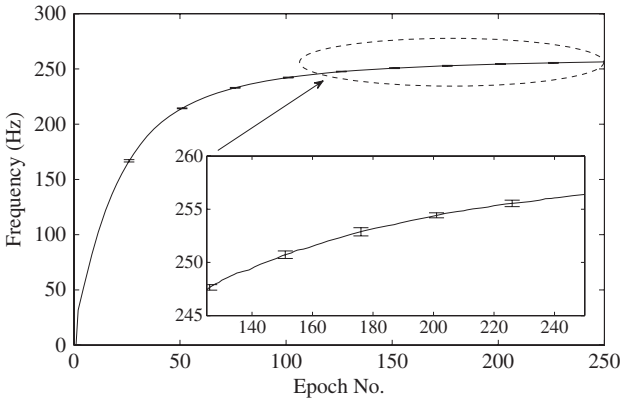

Fig. 13. Average output frequency of the training neuron vs. epoch number for the speech dataset over five training sets. Error bars on the inset show the standard deviation over epochs 125 to 250.

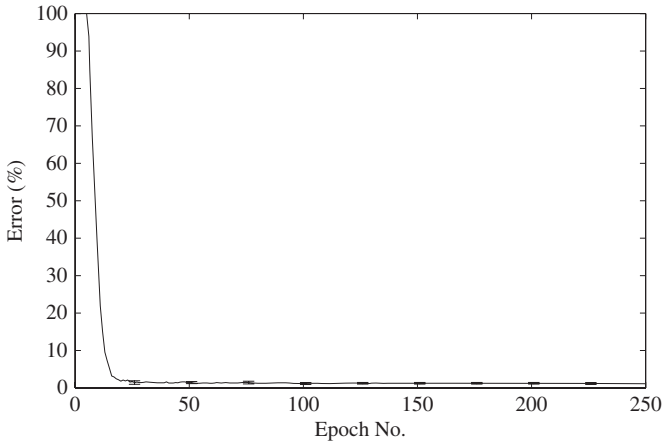

Fig. 14. Convergence accuracy (CA) for TI46 digits training data for all five sets.

where f_s is the sampling frequency, s_l is the signal length in time, and W_f is the number of windows. The frame step S_f, which determines the overlap, is then given by

$$S_f = \frac{N}{K} \tag{23}$$

where K can be adjusted to give varying degrees of overlap. In this paper $K = 2$, this provides a 50% overlap. Variable window sizes have been used by [54] and it was found that they provided better results than fixed window sizes.

4) **LPC feature extraction**. Features are now extracted from each of the frames using LPC coefficients, which allow us to extract good features with a minimal sized network. In this paper, the LPC coefficients were extracted using the Auditory Toolbox [55].
5) **Spike train generation**. Finally, the extracted LPC coefficients are now converted to linear spike trains which can be processed by SWAT. Depending on the number of coefficients extracted, the features are in the range -2.5 to 2.5. To convert these values to spike trains, we offset all values by O_s given by

$$O_s = 1.3 - L_v \tag{24}$$

where L_v is the lowest value. These values are then multiplied by a factor of 10. This results in a series of ISIs, which are then used to generate the spike trains.

TABLE IV
CA OF THE 10 DIGITS

	0	1	2	3	4	5	6	7	8	9	CA (%)
0	37				2					1	92.5
1		40									100
2			40								100
3		1	1	36	1	1					90
4					39				1		97.5
5		2		3	2	33					82.5
6				1		1	38				95
7					1			39			97.5
8									40		100
9					1					39	97.5

B. Results

We then used SWAT as a component of the ASR to classify a subset of the TI46 speech corpus using the digits 0–9 ($n = 10$). As with the previous experiments, we again randomly split the data into five groups for fivefold cross-validation. This results in five datasets of 80 samples, with each dataset containing 8 samples of each digit. For each of the five training and testing runs, there are 320 training samples and 80 testing samples (four groups for training, one group for testing). The optimal number of windows used and features to extract were found experimentally to be 20 and 9, respectively. This results in an input layer of 180 neurons (20×9). The hidden layer consists of 28 EI_{fs} filters for each input, each with an ISI of 2 ms, covering a frequency band of 14.5–77 Hz (ISI 13–69 ms). Therefore, the number of neurons in the hidden layer is 5040 (180×28). The total network size is therefore $180 \times 5040 \times 10$. Other network parameters were set as before: A_p is 0.1, which is equivalent to δA_{SE} of 0.1 pA and c_o was set to 500.

Results gained from this experiment show that, after a period of training, the firing frequency of the training neuron stabilizes, as shown in Fig. 13. Note that the average firing frequency is much higher than in the Iris and WBC experiments. This is because there are more classes and, therefore, each weight value calculated by the STDP/BCM is proportionally segregated across many more synapses, making it harder for the output neurons to fire, the greater the number of classes, the higher the average firing frequency of the training neuron.

Fig. 14 shows the average percentage error and standard deviation against the number of epochs over each of the five

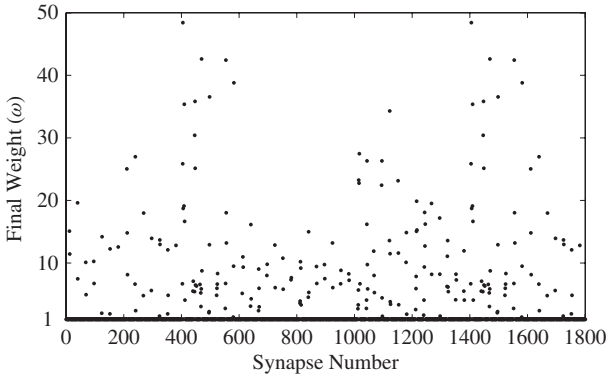

Fig. 15. Post-trained weight distribution of the first 1800 output synapses of the ASR system.

training sets where, again, the training reaches and stabilizes at a minimum for each of the sets. The CA over the five training runs was 98.875%, while the CA for the testing data was found to be 95.25%.

Table IV shows the breakdown of the CA for each of the 40 test samples for each digit during the five testing runs. The rows represent the digit samples, and the columns represent the result of the classification. For example, row 1 shows that the digit 0 was correctly classified 37 times, was mistaken for 4 twice, and for 9 once, resulting in an overall CA for digit 0 of 92.5%.

Fig. 15 highlights the post-training weight distribution for the first 1800 synapses associated with the output neurons. These final weights again show that there is good competition between the synapses.

VIII. Conclusion

This paper presented a training algorithm specific to a novel SNN topology which classifies data represented by spike trains. The algorithm utilized a combined STDP/BCM training rule which accounts for unequal sample sizes in the training data and correlated commonality across the classes in a post-trained unimodal weight distribution. The SNN topology utilizes a frequency routing capability to segregate the input data. This is achieved using a parallel combination of excitatory and inhibitory facilitating synapses, which is similar to feed forward loops found in the hippocampus. A single training neuron is then used to assign weights to the synapses associated with the classifying neurons according to similarities or "relative occurrence of similar data values" in the input data for all classes. Variable firing thresholds were also used to emulate the absolute and relative recovery periods inherent in a biological neuron. SWAT was benchmarked using the Iris and WBC dataset problems, and the results highlight the algorithm's capability to classify these datasets. Overall, the results were similar to other approaches and any comparisons must be interpreted with caution. For example, SpikeProp and WLL use single spike encoding, while other algorithms use neuron models that are less representative of real biological neurons. In contrast, SWAT is more realistic, as it attempts to represent real neural systems by encoding data in spike trains where the associated frequency is used as a processing parameter. Moreover, by reflecting features of the input data, common to all classes, in the post-trained weights, SWAT demonstrates good generalization and noise immunity. Results also show that SWAT can classify complex datasets, whereby a subset of the TI46 speech corpus was used to benchmark SWAT in an ASR system. The CA for this data was 98.875% for training and 95.25% for testing.

References

[1] L. Benuskova and N. Kasabov, "Modeling L-LTP based on changes in concentration of pCREB transcription factor," *Neurocomputing*, vol. 70, nos. 10–12, pp. 2035–2040, Jun. 2007.

[2] M. C. W. van Rossum, G. Q. Bi, and G. G. Turrigiano, "Stable Hebbian learning from spike timing-dependent plasticity," *J. Neurosci.*, vol. 20, no. 23, pp. 8812–8821, Dec. 2000.

[3] E. M. Izhikevich and N. S. Desai, "Relating STDP to BCM," *Neural Comput.*, vol. 15, no. 7, pp. 1511–1523, Jul. 2003.

[4] S. M. Bohte, J. N. Kok, and H. L. Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, Oct. 2002.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, J. L. McClelland and D. E. Rumelhart, Eds. Cambrige, MA: MIT Press, 1986.

[6] P. Rowcliffe and J. Feng, "Training spiking neuronal networks with applications in engineering tasks," *IEEE Trans. Neural Netw.*, vol. 19, no. 9, pp. 1626–1640, Sep. 2008.

[7] J.-P. Pfister, D. Barber, and W. Gerstner, "Optimal Hebbian learning: A probabilistic point of view," in *Proc. ICANN/ICONIP*, Istanbul, Turkey, Jun. 2003, pp. 92–98.

[8] A. Belatreche, L. P. Maguire, T. M. McGinnity, and Q.-X. Wu, "A method for supervised training of spiking neural networks," in *Proc. IEEE Cybern. Intell.-Challenges Adv.*, Reading, U.K., Sep. 2003, pp. 39–44.

[9] S. Soltic, S. G. Wysoski, and N. K. Kasabov, "Evolving spiking neural networks for taste recognition," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Hong Kong, China, Jun. 2008, pp. 2091–2097.

[10] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley, 1949.

[11] B. Ruf and M. Schmitt, "Learning temporally encoded patterns in networks of spiking neurons," *Neural Process. Lett.*, vol. 5, no. 1, pp. 9–18, 1997.

[12] R. Legenstein, C. Naeger, and W. Maass, "What can a neuron learn with spike-timing-dependent plasticity?" *Neural Comput.*, vol. 17, no. 11, pp. 2337–2382, Nov. 2005.

[13] G. Bi and M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, Dec. 1998.

[14] 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.

[15] A. Kasinski and F. Ponulak, "Experimental demonstration of learning properties of a new supervised learning method for the spiking neural networks," in *Proc. Int. Conf. Artificial Neural Netw.*, Warsaw, Poland, Sep. 2005, pp. 145–152.

[16] T. J. Strain, L. J. McDaid, L. P. Maguire, and T. M. McGinnity, "A novel mixed supervised-unsupervised training approach for a spiking neural network classifier," in *Proc. Conf. Intell. Cybern. Syst., SMC UK-RI Chapter*, Londonderry, U.K., Sep. 2004, pp. 202–206.

[17] Q. Wu, T. McGinnity, L. Maguire, A. Belatreche, and B. Glackin, "Adaptive co-ordinate transformation based on a spike timing-dependent plasticity learning paradigm," in *LNCS: Advances in Natural Computation*, vol. 3610, L. Wang, K. Chen, and Y. S. Ong, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 420–428.

[18] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: Taming the beast," *Nat. Neurosci. Suppl.*, vol. 3, pp. 1178–1183, Nov. 2000.

[19] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nat. Neurosci.*, vol. 3, no. 9, pp. 919–926, Sep. 2000.

[20] Q. X. Wu, T. M. McGinnity, L. P. Maguire, B. Glackin, and A. Belatreche, "Learning under weight constraints in networks of temporal encoding spiking neurons," *Neurocomputing*, vol. 69, nos. 16–18, pp. 1912–1922, Oct. 2006.
[21] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, Nov. 2007.
[22] A. M. Thomson and J. Deuchars, "Temporal and spatial properties of local circuits in neocortex," *Trends Neurosci.*, vol. 17, no. 3, pp. 119–126, Mar. 1994.
[23] M. V. Tsodyks and H. Markram, "The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability," *Proc. Natl. Acad. Sci.*, vol. 94, no. 2, pp. 719–723, Jan. 1997.
[24] M. V. Tsodyks, K. Pawelzik, and H. Markram, "Neural networks with dynamic synapses," *Neural Comput.*, vol. 10, no. 4, pp. 821–835, May 1998.
[25] R. C. Froemke and Y. Dan, "Spike-timing-dependent synaptic modification induced by natural spike trains," *Nature*, vol. 416, pp. 433–438, Mar. 2002.
[26] P. Jedlicka, "Synaptic plasticity, metaplasticity and BCM theory," *Bratisl. Lek. Listy*, vol. 103, nos. 4–5, pp. 137–143, 2002.
[27] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," *J. Neurosci.*, vol. 2, no. 1, pp. 32–48, Jan. 1982.
[28] V. A. Klyachko and C. F. Stevens, "Excitatory and feed-forward inhibitory hippocampal synapses work synergistically as an adaptive filter of natural spike trains," *PLoS Biol.*, vol. 4, no. 7, p. e207, Jul. 2006.
[29] A. J. Delaney and C. E. Jahr, "Kainate receptors differentially regulate release at two parallel fiber synapses," *Neuron*, vol. 36, no. 3, pp. 475–482, Oct. 2002.
[30] J. S. Dittman, A. C. Kreitzer, and W. G. Regehr, "Interplay between facilitation, depression, and residual calcium at three presynaptic terminals," *J. Neurosci.*, vol. 20, no. 4, pp. 1374–1385, Feb. 2000.
[31] M. Mori, M. H. Abegg, B. H. Gähwiler, and U. Gerber, "A frequency-dependent switch from inhibition to excitation in a hippocampal unitary circuit," *Nature*, vol. 431, no. 7007, pp. 453–456, Sep. 2004.
[32] C. Saviane, L. P. Savtchenko, G. Raffaelli, L. L. Voronin, and E. Cherubini, "Frequency-dependent shift from paired-pulse facilitation to paired-pulse depression at unitary CA3–CA3 synapses in the rat hippocampus," *J. Physiol.*, vol. 544, no. 2, pp. 469–476, Oct. 2002.
[33] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "A biologically inspired training algorithm for spiking neural networks," in *Proc. Irish Signals Syst. Conf.*, Londonderry, U.K., Sep. 2007, pp. 7–12.
[34] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: An unsupervised SNN training algorithm for classification problems," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Hong Kong, China, Jun. 2008, pp. 2648–2655.
[35] A. Araque, G. Carmignoto, and P. G. Haydon, "Dynamic signaling between astrocytes and neurons," *Annu. Rev. Phys.*, vol. 63, no. 1, pp. 795–813, 2001.
[36] P. Kurosinski and J. Götz, "Glial cells under physiologic and pathologic conditions," *Arch. Neurol.*, vol. 59, no. 10, pp. 1524–1528, Oct. 2002.
[37] G. Perea and A. Araque, "Properties of synaptically evoked astrocyte calcium signal reveal synaptic information processing by astrocytes," *J. Neurosci.*, vol. 25, no. 9, pp. 2192–2203, Mar. 2005.
[38] G. Perea and A. Araque, "Synaptic information processing by astrocytes," *J. Physiol.-Paris*, vol. 99, nos. 2–3, pp. 92–97, Mar.–May 2006.
[39] G. Perea and A. Araque, "Astrocytes potentiate transmitter release at single hippocampal synapses," *Science*, vol. 317, no. 5841, pp. 1083–1086, Aug. 2007.
[40] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[41] N. Kasabov and L. Benuskova, "Computational neurogenetics," *J. Comput. Theor. Nanosci.*, vol. 1, no. 1, pp. 47–61, Mar. 2004.
[42] R. F. Thompson, *The Brain: A Neuroscience Primer*, 2nd ed. San Francisco, CA: Freeman, 1993.
[43] N. Kasabov, "Neuro-, genetic-, and quantum inspired evolving intelligent systems," in *Proc. Int. Symp. Evolving Fuzzy Syst.,* Ambleside, U.K., Sep. 2006, pp. 63–73.
[44] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, 4th ed. New York: McGraw-Hill, 2000.
[45] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[46] Z. Yang and A. F. Murray, "An artificial early visual model adopting spike-timing-dependent plasticity," *Neurocomputing*, vol. 69, nos. 16–18, pp. 1904–1911, Oct. 2006.
[47] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. Natl. Acad. Sci.*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990.
[48] *Breast Cancer Wisconsin Dataset* [Online]. Available: http://archive.ics.uci.edu/ml/datasets/
[49] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, pp. 76–78, Sep. 1996.
[50] I. A. Basheer and M. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application," *J. Microbiol. Methods*, vol. 43, no. 1, pp. 3–31, Dec. 2000.
[51] G. R. Doddington and T. B. Schalk, "Speech recognition: Turning theory to practice," *IEEE Spectrum*, vol. 18, no. 9, pp. 26–32, Sep. 1981.
[52] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterances," *J. Acoust. Soc. Am. Bell Syst. Tech.*, vol. 54, no. 2, pp. 297–315, Feb. 1975.
[53] X. Zhang, Y. Guo, and X. Hou, "A speech recognition method of isolated words based on modified LPC cepstrum," in *Proc. IEEE Int. Conf. Granular Comput.*, Washington D.C., Nov. 2007, p. 481.
[54] R. Fernández-Lorenzana, F. Pérez-Cruz, J. M. García-Cabellos, C. Peláez-Moreno, A. Gallardo-Antolín, and F. Díaz-de-María, "Some experiments on speaker-independent isolated digit recognition using SVM classifiers," in *Proc. ISCA Tutorial Res. Workshop Non-Linear Speech Process.*, Le Croisic, France, May 2003, pp. 1–7.
[55] *Auditory Toolbox Version 2* [Online]. Available: http://cnmat.berkeley.edu/link/3630

John J. Wade was born in Derry, Northern Ireland, U.K., in 1977. He received the B.Eng. (Hons) degree in electronics and computing, the M.Sc. degree in computing and intelligent systems, and the Ph.D. degree in computational neural systems, all from the University of Ulster, Derry, in 2004, 2005, and 2010, respectively.

He is currently employed as a Researcher at the Intelligent Systems Research Center, University of Ulster, where he is a part of the Bio-Inspired and Neuro-Engineering Teams. His current research interests include developing computational models of neural and glial systems to aid understanding of how the brain functions and learns.

Liam J. McDaid received the B.Eng. (Hons) degree in electrical and electronics engineering from the University of Liverpool, Liverpool, U.K., in 1985, and the Ph.D. degree in solid-state devices from the same institution.

He is currently employed as a Senior Lecturer at the School of Computing and Intelligent Systems, University of Ulster, Derry, Northern Ireland, U.K. He has co-authored over 90 publications in his career to present. His current research interests include software/hardware implementations of neural-based computational systems and modeling the mechanisms that underpin self-repair in the human brain, thus providing the blueprint for advanced architectures that exhibit a fault-tolerant capability well beyond existing computational systems. He has received several research grants in this domain.

Dr. McDaid is a Founder of the Nanoelectronics Research Group within the Intelligent Systems Research Center at the University of Ulster. He is currently a Guest Editor for a special issue entitled "Advances in Spiking Neural Networks and their Applications" to appear in the *International Journal of Neural Systems*.

Jose A. Santos was born in Maracay, Venezuela, in 1973. He received the B.E. degree from the Electronics Department, Universidad Simon Bolivar, Caracas, Venezuela, in 1998, and the Ph.D. degree in electronic engineering from the School of Electrical and Mechanical Engineering, University of Ulster, Derry, Northern Ireland, U.K., in 2003.

He has been a Lecturer in computer science at the School of Computing and Intelligent Systems, University of Ulster, since 2002. His current research interests include ambient intelligence and mobile computing, wireless communication systems, neural networks and computational intelligence, biomedical engineering, sensor technology, and robotics.

Dr. Santos is a member of the Venezuelan College of Engineers. He is a member of the Intelligent Systems Research Center, University of Ulster, where he is part of the Ambient Intelligence Research Group.

Heather M. Sayers received the M.Sc. degree in computing and information systems and the Ph.D. degree in computer science (interacting in 3-D environments) from the University of Ulster, Derry, Northern Ireland, U.K., in 1998 and 2004, respectively.

She was employed as a Lecturer in 2000 and is currently a Senior Lecturer at the School of Computing and Intelligent Systems, University of Ulster. She has published a number of research papers in leading journals and in both national and international conference proceedings. Her current research interests include the integration of human–computer interaction and intelligent techniques for usability.

Dr. Sayers is a member of the British Computer Society.