<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>An Online Unsupervised Structural Plasticity Algorithm for Spiking Neural Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Roy, Subhrajit; Basu, Arindam</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2016</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/41059">http://hdl.handle.net/10220/41059</a></td>
</tr>
</tbody>
</table>

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [http://dx.doi.org/10.1109/TNNLS.2016.2582517].
An Online Unsupervised Structural Plasticity Algorithm for Spiking Neural Networks

Subhrajit Roy, Student Member, IEEE and Arindam Basu, Member, IEEE

Abstract—In this article, we propose a novel Winner-Take-All (WTA) architecture employing neurons with nonlinear dendrites and an online unsupervised structural plasticity rule for training it. Further, to aid hardware implementations, our network employs only binary synapses. The proposed learning rule is inspired by Spike-Timing-Dependent Plasticity (STDP) but differs for each dendrite based on its activation level. It trains the WTA network through formation and elimination of connections between inputs and synapses. To demonstrate the performance of the proposed network and learning rule, we employ it to solve two, four and six-class classification of random Poisson spike time inputs. The results indicate that by proper tuning of the inhibitory time constant of the WTA, a trade-off between specificity and sensitivity of the network can be achieved. We use the inhibitory time constant to set the number of subpatterns per pattern we want to detect. We show that while the percentage of successful trials are 92%, 88% and 82% for two, four and six-class classification when no pattern subdivisions are made, it increases to 100% when each pattern is subdivided into 5 or 10 subpatterns. However, the former scenario of no pattern subdivision is more jitter resilient than the later ones.

I. INTRODUCTION AND MOTIVATION

The Winner-Take-All (WTA) is a computational framework in which a group of recurrent neurons cooperate and compete with each other for activation. The computational power of WTA [1]–[3] and its function in cortical processing [1], [4] have been studied in detail. Various models and hardware implementations of WTA have been proposed for both rate [5]–[12] and spike based [13]–[15] neural networks. In recent past, researchers have looked into the application of Spike-Timing-Dependent Plasticity (STDP) learning rule on WTA circuits. The performance of competitive spiking neurons trained with STDP has been studied for different types of input such as discrete spike volleys [16]–[18], periodic inputs [19], [20] and inputs with random intervals [15], [21], [22].

In this paper, for the first time we are proposing a WTA network which uses neurons with nonlinear dendrites (NNLD) and binary synapses as the basic computational units. This architecture, which we refer to as Winner-Take-All employing Neurons with NonLinear Dendrites (WTA-NNLD), uses a novel branch-specific Spike-Timing-Dependent Plasticity based Network Rewiring (STDP-NRW) learning rule for its training. We have earlier presented [23] a branch-specific STDP rule for batch learning of a supervised classifier constructed of NNLDs. The primary differences of our current approach with [23] are:

- We present an unsupervised learning rule for training a WTA network.
- We propose an online learning scheme where connection modifications occur after the presentation of each pattern.

In this article, we consider spike train inputs with patterns occurring at random order which is the same type presented in [15]. The primary differences between our work and the one proposed in [15] are:

- Our WTA network is composed of neurons with nonlinear dendrites instead of traditional neurons with no dendrites.
- Unlike the network proposed in [15] that requires high-resolution weights, the proposed network uses low-resolution non-negative integer weights and trains itself through modifying connections of inputs to dendrites. Hence, change of the ‘morphology’ or structure of the neurons (in terms of connectivity pattern) reflects the learning. This results in easier hardware implementation since a low-resolution non-negative integer weight of W can be implemented by activating a shared binary synapse W times through time multiplexing schemes like address event representation (AER) [24], [25].
- In [15], though the neurons were allowed to learn and respond to subpatterns, there was no actual guideline or control parameter to set the number of subpatterns to be learned. Here we utilize the slow time constant of the inhibitory signal to select the number of subpatterns we want to divide a pattern into.

In the following section, we will present an overview of NNLD, propose the WTA-NNLD architecture and STDP-NRW learning rule and show how the inhibitory slow time constant can be used to select subpatterns within a pattern. Then we shall provide guidelines on selecting the parameters associated with WTA-NNLD and STDP-NRW. In Section IV, we will describe the classification task considered in this article which will be followed by the results. We will also present the robustness of the proposed method to variations of parameters in Section V, a quality that is essential for its implementation in low-power, subthreshold neuromorphic designs that are plagued with mismatch. We will conclude the paper by discussing the implications of our work and future directions in the last section.

II. BACKGROUND AND THEORY

In this section, we shall first present the working principle of a NNLD. This will be followed by a description of the WTA-NNLD architecture and STDP-NRW learning rule. Lastly, we will describe the role of the inhibitory time constant in balancing the specificity and sensitivity of the network.

A. Neuron with nonlinear dendrites (NNLD)

The computational model of NNLD was first proposed by Mel et al. in [26] where they showed that such neurons have
higher storage capacity than their non-dendritic counterparts. They used two such NNLDs to construct a supervised classifier and demonstrated its performance in pattern memorization. Recently, NNLD has also been employed to develop computationally powerful rate [27] and spike based [28], [29] supervised classifiers. The structure of NNLD is isomorphic to a feedforward spiking neural network with a single layer of hidden neurons and one output neuron [30]. The lumped dendritic nonlinearities \( b() \) are equivalent to the hidden neurons interposed between the input and output layers. However, spiking neurons implement nonlinear thresholding, integration, refractory period etc. Hence, it is typically a much larger circuit compared to the square-law nonlinearity of a dendrite, which makes the NNLD an area efficient architecture.

As depicted in Fig. 1, a NNLD consists of \( m \) dendritic branches having lumped nonlinearities, with each branch containing \( k \) excitatory synaptic contact points of weight 1. If we consider a \( d \) dimensional input pattern, then each synapse is driven by any one of these input dimensions where \( d >> k \). We use the Leaky Integrate-and-Fire (LIF) model to generate output spikes. Thus, the neuronal membrane voltage is guided by the following differential equation:

\[
\frac{dV(t)}{dt} + \frac{V(t)}{R} = I_{in}(t) \\
\text{If } V(t) \geq V_{thr}, V(t) \rightarrow 0; \text{ and } f(t) \rightarrow 1 \\
\text{else } f(t) = 0
\]

where \( V(t) \), \( V_{thr} \), \( I_{in}(t) \) and \( f(t) \) are the membrane voltage, threshold voltage, input current and output spikes of the NNLD respectively. Let us denote the input spike train arriving at the \( i^{th} \) input line as \( e^i(t) \) which is given by:

\[
e^i(t) = \sum g \delta(t - t_{ig})
\]

where \( g = 1, 2, \ldots \) is the label of the spike. Then, the input current \( I_{in}(t) \) to the neuron can be calculated as:

\[
I_{in}(t) = \sum_{j=1}^{m} I_{j, out}(t)
\]

\[
I_{j, out}(t) = b(I_{j, in}(t))
\]

\[
I_{j, in}(t) = \sum_{i=1}^{d} w_{ij}(t - t_{ig})
\]

where \( b(.) \) is the nonlinear activation function of the dendritic branch characterized by \( b(z) = z^2 / x_{thr}, I_{j, out}(t) \) is the output current of the \( j^{th} \) dendrite and \( I_{j, in}(t) \) is the input current to the dendritic nonlinearity. Here, \( x_{thr} \) describes the behavior of the dendritic nonlinear function by setting a limit on the minimum number of coactive synapses required to produce a supra-linear response. \( K \) denotes the post-synaptic current kernel given by:

\[
K(t) = I_0(e^{-\frac{t}{\tau_f}} - e^{-\frac{t}{\tau_s}})
\]

where \( \tau_f \) and \( \tau_s \) are the fast and slow time constants governing the rise and fall times respectively and \( I_0 \) is a normalizing constant. In this article, we consider low-resolution non-negative integer weights associated with the input lines. So \( w_{ij} \in \{0, 1, \ldots, k\} \) means the number of times the \( i^{th} \) input line is connected to the \( j^{th} \) dendritic branch. Like our earlier work [29], we allow multiple connections of one input dimension to a single dendrite but restrict the number of connections per dendrite to \( k \) by enforcing \( \sum_{i=1}^{d} w_{ij} = k \) for each \( j \). The output of the NNLD is a spike train and can be denoted as

\[
f(t) = \sum_{a} \delta(t - t_a)
\]

where \( a = 1, 2, \ldots \) is the spike index.

B. Winner-Take-All employing Neurons with NonLinear Dendrites (WTA-NNLD)

We propose a spike based WTA network, depicted in Fig. 2, which is composed of \( N \) such NNLDs. Each NNLD is composed of \( m \) dendrites, where each dendrite chooses (repetition
allowed) $k$ of the $d$ available input lines and connects to $k$ synapses having weight 1. The membrane voltage, threshold voltage and input current of the $n^{th}$ NNLD are denoted by $V^n(t), V_{thr}$ and $I^n_{in}(t)$ respectively and the dynamics of the membrane voltage is governed by Equation 1. For the $n^{th}$ NNLD, while the pre-synaptic spike-train arriving at the $i^{th}$ input line is denoted by $e^i(t)$ as before, the emitted output spike train is given by $f^n(t) = \sum_a \delta(t - t^n_a)$. Note that for any applied input pattern, $t^n_a$ is measured from its beginning.

We have modeled the effect of lateral inhibition by providing each NNLD with a global inhibitory current signal $I_{inh}(t)$ supplied by a single inhibitory neuron $N_{inh}$ through synapses. The signal $I_{inh}(t)$ is provided by the inhibitory neuron to all the NNLD whenever any one of them fires an output spike. $I_{inh}(t)$ is modeled as $I_{inh}(t) = K_{inh}(t - t^n_{last})$, where the last post-synaptic spike is produced by the $n^{th}$ NNLD at $t^n_{last}$. The inhibitory post-synaptic kernel, $K_{inh}$, is given by:

$$K_{inh}(t) = I_{0,inh}(e^{-\frac{t}{\tau_{s,inh}}} - e^{-\frac{t}{\tau_{f,inh}}})$$

where $\tau_{f,inh}$ and $\tau_{s,inh}$ are the fast and slow time constants dictating the rise and fall times of the inhibitory current respectively and $I_{0,inh}$ sets its amplitude.

C. Spike-Timing-Dependent Plasticity based Network Rewiring learning rule (STDP-NRW)

Since we consider binary synapses with weight 0 or 1, we do not have the provision to keep real-valued weights associated with them. Hence, to guide the unsupervised learning, we define a correlation coefficient based fitness value $c^n_{pj}(t)$ for the $p^{th}$ synaptic contact point on the $j^{th}$ dendrite of the $n^{th}$ NNLD of the WTA network, as a substitute for its weight. In the proposed algorithm, structural plasticity or connection modifications happen on longer timescales (at the end of patterns) which is guided by the fitness function $c^n_{pj}(t)$ updated by a STDP inspired rule in shorter timescales (at each pre- and post-synaptic spike). The operation of the network and learning process comprises the following steps whenever a pattern is presented:

- The value of $c^n_{pj}(t)$ is depressed at pre-synaptic and potentiated at post-synaptic spikes according to the following rule:
  1. Depression: If the pre-synaptic spike occurs at the $p^{th}$ synapse on the $j^{th}$ dendritic branch of the $n^{th}$ NNLD at time $t^{pre}$, then the value of $c^n_{pj}(t)$ at $t = t^{pre}$ is updated by a quantity $\Delta c^n_{pj}(t = t^{pre})$ given by:

$$\Delta c^n_{pj}(t) = -b^i_j(t) f^n(t) \bigg|_{t = t^{pre}}$$

where $f^n(t) = K(t) * f^n(t)$ is the post-synaptic trace of the $n^{th}$ NNLD and $b^i_j()$ denotes derivative of the nonlinear function $b()$.

2. Potentiation: If the $n^{th}$ NNLD of the WTA-NNLD network fires a post-synaptic spike at time $t^{post}$ then $c^n_{pj}(t)$ at $t = t^{post}$ ∀ $p = 1, 2, ..., k; j = 1, 2, ..., m$ i.e. for each synapse connected to the $n^{th}$ NNLD is updated by $\Delta c^n_{pj}(t = t^{post})$ given by:

$$\Delta c^n_{pj}(t) = b^i_j(t) \bar{e}^i(t) \bigg|_{t = t^{post}}$$

where $\bar{e}^i(t) = K(t) * e^i(t)$ is the pre-synaptic trace of the corresponding input line connected to it.

A pictorial explanation of this update rule of $c^n_{pj}(t)$ is shown in Fig. 3. Note that for a square-law nonlinearity, $b'(z) \propto z$ and hence can be easily computed in hardware without requiring any extra circuitry to calculate the derivative.

- During the presentation of the pattern whenever a spike is produced by any of the $N$ excitatory NNLDs, the inhibitory neuron $N_{inh}$ sends an inhibitory signal to all the NNLDs of the WTA.

- After the network has integrated the input spikes over the current pattern duration $T_p$, the synaptic connections of the NNLDs which have produced at least one spike are modified.

- If we consider that $Q$ out of $N$ NNLDs have produced post-synaptic spike/spikes for the current pattern, then the connectivity of the $q^{th}$ NNLD $q = 1, 2, ..., Q$ is updated by tagging the synapse $(s_{min}^q)$ having the lowest value of correlation coefficient at $t = T_p$ out of the $m \times k$ synapses connected to it for possible replacement.

- To aid the unsupervised learning process, randomly chosen sets $R^q$ containing $r_R$ of the $d$ input dimensions are forced to make silent synapses of weight 1 on the dendritic branch of $s_{min}^q \forall q = 1, 2, ..., Q$. We term these synapses as “silent” since they do not contribute to the computation of $V^n(t)$ - so they do not alter the classification when the same pattern set is re-applied. The value of $c^n_{pj}(t = T_p)$ is calculated for synapses in $R^q$ and the synapse having maximum $c^n_{pj}(t = T_p)$ in $R^q$ denoted by $r_{max}^q \forall q = 1, 2, ..., Q$ is identified. Next, the input line connected to $s_{min}^q$ is swapped with the input line connected to $r_{max}^q$. Hence, instead of the traditional method of training by changing of high-resolution synaptic weights, our learning rule modifies the connections between the inputs and dendrites based on the fitness values.

- All the $c^n_{pj}(t)$ values are reset to zero and the above mentioned steps are repeated whenever a pattern is presented. Here, we define an epoch for C-class classification as a set of patterns consisting of one pattern from each of the $C$ classes- in random order. We define another term $l_{mean}$ as the average of the latencies of the post-synaptic spikes in the network over time period of the last epoch which is given by:

$$l_{mean} = \frac{< \sum_n \sum_a t^n_a >_1}{N}$$

where $< \cdot >_1$ denotes averaging over one epoch. We note the value of $l_{mean}$ for every epoch and the learning is considered to converge when the value of a ‘Convergence Measure’ (CM) based on $l_{mean}$ reaches saturation. We define our ‘Convergence Measure’ in Section III.
D. Specificity and Sensitivity: Role of Inhibitory Time Constant

When a pattern is presented to the WTA-NNLD and any one of the \( N \) NNLDs produce an output spike, a global inhibition current \( I_{inh}(t) \) is injected into all the \( N \) NNLDs. The slow time constant \( \tau_{s,inh} \) of this signal controls the output firing activity of the WTA-NNLD. Typically, a large value of \( \tau_{s,inh} \) (w.r.t to \( \tau_p \)) is set, and only one NNLD produces an output spike i.e. patterns of the same class are encoded by a single NNLD. The post-synaptic spike latency for a pattern \( P \) is defined as the time difference between the start of the pattern and the first spike produced by any one of the \( N \) neurons of WTA-NNLD. During training of WTA-NNLD for this case, different NNLDs get locked onto different classes of pattern and the latency gradually decreases until the end of the training. Thus, after completion of training, the unique NNLDs which have learned different classes of pattern rely only on the first few spikes (determined by the latency at the end of training) to predict the pattern’s class thereby significantly reducing the prediction time [15]. So, the sensitivity of the network is increased. However, the problems with this approach are:

- The percentage of successful classifications can be less due to the strict requirement of different neurons firing based only on first few spikes of different patterns (shown in Section IV).
- Though the prediction time of a pattern’s class is significantly reduced, this method neglects most part of the pattern after the first few spikes which may lead to a lot of false detections.

We demonstrate the limitation mentioned in the above point by a simple example in Fig. 4. Let us consider that we are performing a \( C \)-class classification task and assume that after the training phase is complete, NNLD \( N_{f1} \) responds to patterns belonging to Class 1. NNLD \( N_{f1} \) has trained itself to provide an output spike depending on the position of the first few spikes (red spikes in dashed box of Fig. 4) of the pattern. It neglects the rest of the pattern while providing a prediction. However, for longer patterns there is a chance that this spike set can occur anywhere inside a random pattern (not belonging to any class or to another class). The same NNLD \( N_{f1} \) responds to such patterns by producing a post-synaptic spike. Hence, we see that though trained WTA-NNLD is very sensitive in this case, it loses specificity. On the other hand, if we set a moderate value of \( \tau_{s,inh} \), then for a single pattern multiple NNLDs are capable of producing output spikes. Hence, patterns of the same class are now encoded by a sequence of successive firing of few NNLDs where each NNLD fires for one subpattern. Let \( n_{sub} \) be the number of subpatterns that is set by a proper choice of \( \tau_{s,inh} \). Thus, the original case of one NNLD firing for each pattern corresponds to \( n_{sub} = 1 \). In this article, for a \( C \)-class classification task, we define a successful trial as one in which (a) during the training phase WTA-NNLD learns different unique representations for patterns of different classes and (b) after completion of training and achieving success in (a), the network produces the same representation, when presented with testing patterns corresponding to classes that it had learned during the training phase. When \( n_{sub} = 1 \) i.e. no pattern subdivisions are made, this unique representation is a different neuron firing for different classes of patterns. When \( n_{sub} > 1 \), the unique representation is a different sequence of successive NNLDs firing for different classes of patterns. When, \( n_{sub} > 1 \), we allow the NNLDs to detect subpatterns within patterns. Since in this approach the WTA-NNLD gives weightage to the entire pattern before predicting its class, the number of false detections can be largely reduced. However, this method has a limitation of being less jitter resilient — one of the many subpatterns can be easily corrupt by noisy jitters in spike (shown in Section IV) and fail to produce a unique identifier during the testing phase. Hence, the choice of \( n_{sub} \) and consequently the inhibitory time constant depends on the amount of temporal jitter in the application.

III. CHOICE OF PARAMETERS

The following is an exhaustive list of the parameters used by WTA-NNLD and STDP-NRW:

1) \( T_p \): Duration of a pattern
2) \( d \): Dimension of the input
We will now provide some guidelines on choosing the key parameters:

a) The total number of synapses per NNLD (s): The number of synapses allocated to each neuronal cell of WTA-NNLD are kept as equal to the dimension (d) of the input patterns. This is done to ensure NNLD uses the same amount of synaptic resources as the simplest neuron–a perceptron. Thus, if the proposed network is comprised of N such neuronal cells then the total number of synaptic resources required are d × N.

b) Number of dendrites per NNLD (m): In [26], a measure of the pattern memorization capacity, $B_N$, of the NNLD (Fig.1) has been provided by counting all possible functions realizable as:

$$B_N = \log_2 \left( \frac{(k+d-1) + m - 1}{m} \right) \text{bits} \quad (12)$$

where $m$, $k$ and $d$ are the number of dendrites, number of synapses per dendrites and dimension of the input respectively for this neuronal cell. When a new classification problem is encountered, we first note the value of $d$, which in turn sets our $s$ since we have considered $s = d$. Since $s = m \times k$, for a fixed $s$ all possible values which $m$ can take are factors of $s$. We calculate $B_N$ for these values of $m$ by Equation 12. The value of $m$ for which $B_N$ attains its maxima is set as $m$ in our experiment. As an example, we show in Fig. 5 the variation of $B_N$ with $m$ when $d = 100$. It is evident from the curve that the capacity is maximum when $m = 25$ and so in our simulations for classifying 100 dimensional patterns we employ neuronal cells having 25 dendrites.

c) Number of synapses per branch (k): After $s$ and $m$ have been set, the value of $k$ can be computed as $k = \frac{s}{m}$.

d) The normalization constant ($I_0$), slow ($\tau_s$) and fast time constant ($\tau_f$) of excitatory PSC kernel: The fast time constant ($\tau_f$) and the slow time constant ($\tau_s$) have been defined in Section II-C. In hardware implementation of a synapse [29] $\tau_f$ usually takes a small positive value and is typically not tuned. The slow time constant, $\tau_s$, is responsible for integration across temporally correlated spikes and the performance of the network is dependent on its value. If $\tau_s$ takes too small a value, then the post synaptic current due to individual spikes dies down rapidly and thus temporal integration of separated inputs does not take place. On the other hand, large values of $\tau_s$ render all spikes effectively simultaneous. So, in both extremes, the extraction of temporal features from the input pattern is hampered. In [29] we have provided a mathematical formula for calculating $\tau_{s, opt}$, the optimal value of $\tau_s$, with respect to the inter-spike interval (ISI) of the input pattern for which optimal performance of the network is obtained. If we are considering $d$ dimensional patterns and the mean firing rate of each dimension is $\mu_f$, then the mean ISI across the entire pattern is given by $\mu_{ISI} = 1/(d \times \mu_f)$. We can then set $\tau_{s, opt}$ according to the formula:

$$\tau_{s, opt} = 52.83 \mu_{ISI} - 3.1 \quad (13)$$

In our simulations, we keep $\tau_f$ as $\tau_f = \tau_0$. Since the weights of all the active synapses are 1, we set $I_0 = 1.4351$ to normalize the amplitude of the PSC to be 1.

e) Threshold of nonlinearity ($x_{thr}$): During the training of WTA-NNLD, the STDP-NRW rule preferably selects those connection topologies where correlated inputs for synaptic connections are connected to the same branch. Thus, the lumped dendritic nonlinearity $b(z) = \frac{z}{x_{thr}}$ should give a supra-linear output only when correlated input dimensions are connected to the dendrite. To ensure this, we keep the value of $x_{thr}$ equal to the average input to the nonlinear function in case of random connections. We create numerous instances of dendrites having k synapses and calculate the average input to the nonlinear function, $b_{in, avg}$, for the given pattern set. Then we set the value of $x_{thr}$ as $x_{thr} = b_{in, avg}$.

f) $V_{thr}$ of NNLD: The NNLD should provide a postsynaptic spike only when correlated inputs have been connected to its dendrites. We consider a NNLD having $m$ dendrites and $k$ synapses and create numerous instances of random connections to these synapses. We measure the average value of the maximum membrane voltage ($V_{max}$) produced when this NNLD integrated spikes over the pattern duration for all these instances and set $V_{thr} = (V_{max})_{avg}$.

g) The normalization constant ($I_{0,inh}$), slow ($\tau_{s,inh}$) and fast ($\tau_{f,inh}$) time constant of $I_{inh}(t)$: The post-synaptic firing activity of the WTA-NNLD network is dependent on $\tau_{s,inh}$ and $I_{0,inh}$. To simulate the hardware scenario we set $\tau_{f,inh}$ to a small value given by $\tau_{f,inh} = \frac{\tau_s}{10}$. To set $I_{0,inh}$ and $\tau_{s,inh}$, we first excite WTA-NNLD with $c_{pat}$ epochs of patterns prior to training and calculate the average excitatory current ($I_{e,avg}$)
to the NNLDs as:

\[ I_{e,\text{av}} = \langle \frac{1}{N} \sum_{n=1}^{N} I_{\text{inh}}(t) \rangle_{\text{epochs}} \]  

(14)

where \( \langle \cdot \rangle_{\text{epochs}} \) denotes averaging over \( \text{epochs} \) epochs. The idea is to generate a \( I_{\text{inh}}(t) \) which, if provided by \( N_{\text{inh}} \) at the beginning of a subpattern, decays exponentially to \( I_{e,\text{av}} \) at the end of the subpattern i.e. after time \( T_{sub} \) has elapsed. This ensures that once a post-synaptic spike is generated by a NNLD in a particular \( T_{sub} \) time window, other NNLDs are unable to fire during that same \( T_{sub} \) time window. Assuming \( \tau_{f,\text{inh}} \ll \tau_{s,\text{inh}} \), we can derive that the required \( I_{\text{inh}}(t) \) is implemented by setting \( \tau_{s,\text{inh}} \) as:

\[ \tau_{s,\text{inh}} = \frac{T_{sub}}{\ln(I_{\text{inh}}/I_{e,\text{av}})} \]  

(15)

Note that \( \tau_{s,\text{inh}} \) has an inverse logarithmic relation to \( I_{0,\text{inh}} \).

h) Convergence Measure (CM): The formula for calculating \( CM \), applicable to both \( n_{sub} = 1 \) and \( n_{sub} > 1 \), for detecting the convergence of learning is given by:

\[ CM = \frac{l_{\text{mean}}}{n_{sub}} - \frac{(n_{sub} - 1) T_{sub}}{2} \]  

(16)

Note that for \( n_{sub} = 1, CM = l_{\text{mean}} \) and so it computes the time-to-first spike for patterns averaged over an epoch. For \( n_{sub} > 1, CM \) calculates the average time-to-first spike from the beginning of each subpattern of \( C \) patterns of an epoch. We consider the learning has converged when the value of \( CM \) saturates.

IV. EXPERIMENTS AND RESULTS

In this section, we will describe the classification task considered in this article. To show how the network performs for multi-class classification, we will consider two, four and six-classes of patterns. We will be showing the performance of WTA-NNLD and STDPRNRW for three values of \( n_{sub} \) given by \( n_{sub} = 1, 5 \) and 10.

A. Problem Description

The benchmark task we have selected to analyze the performance of the proposed method is the Spike Train Classification problem [31]. In the generalized Spike Train Classification problem, \( C \) arrays of \( h \) Poisson spike trains having frequency \( f \) and length \( T_{p} \) are present which are labeled as classes 1 to \( C \). Jittered versions of these templates are created by altering the position of each spike within the templates by a random amount that is randomly drawn from a Gaussian distribution with zero mean and standard deviation \( \sigma_{\text{jitter}} \). The network is trained by these jittered versions of spike trains, and the task is to correctly identify a pattern’s class. In this article, unless otherwise mentioned, we have considered \( h = 100, f = 20, T_{p} = 0.5 \text{ sec} \) and varied \( C \) and \( \sigma_{\text{jitter}} \). Inspired by [15], we also consider the scenario when \( h/2 \) randomly chosen afferents do not contain any spikes, while the remaining \( h/2 \) afferents have Poisson spike trains.

B. Case 1: \( n_{sub} = 1 \)

In this case we have \( T_{sub} = T_{p} \), so one NNLD is capable of firing only once when a pattern is presented. Considering \( \sigma_{\text{jitter}} = 0 \), we have varied the number of NNLDs and noted the percentage of successful trials as depicted in Fig. 6(a). To make the horizontal axis invariant of the number of classes, we have taken \( N \) as the horizontal axis. From the figure, we can conclude that the percentage of successful trials gradually increases with an increase in \( N \) and finally becomes constant after \( N = 11 \). Thus, unless otherwise mentioned, we will keep \( N = 11 \times C \) when \( n_{sub} = 1 \). It can be seen from Fig. 6(a) that the percentage of successful trials cannot go beyond 92%, 88% and 82% for two, four and six-class classification respectively.

Earlier in Section III, we mentioned that the learning converges when \( CM \) saturates. For \( n_{sub} = 1, CM = l_{\text{mean}} \) is the average of the time-to-first spikes for patterns in an epoch. As an example, we consider a particular trial of four-class classification and show in Fig. 7 that during training, the latencies of the four NNLDs, \( N_{f1}, N_{f2}, N_{f3} \) and \( N_{f4} \)
which uniquely recognize the four class of patterns gradually reduce until reaching a saturation point. Moreover, in Fig. 6(b) we show the epochwise evolution of $CM$ averaged over 50 trials for two-class, four-class and six-class classification. It is evident from the figure that the value of $CM$ decreases thereby showing that the algorithm is favoring correlated inputs such that the post-synaptic spikes can occur faster and finally saturates after some epochs have passed. We denote the number of epochs taken by the algorithm for saturation of $CM$ as $ep_{sat}$ and note its value for 50 trials. The average value of $ep_{sat}$ for 50 trials, $ep_{sat,avg}$, is then computed to be $ep_{sat,avg} = 149$, 157 and 165 for two, four and six-class classification respectively. Moreover, this phenomenon clearly indicates that while the WTA-NNLD network is being trained by STDP-NRW learning rule, $C$ unique NNLDs which have locked onto the $C$ different classes of pattern, are trying to recognize the start of repeating patterns for different classes. Fig. 6(b) also suggests that after the training has stopped, these $C$ unique NNLDs, instead of looking at the whole pattern of duration $T_p = 500ms$, can now look only at the starting 44.8ms, 53.1ms and 56.1ms of the patterns for $C = 2, 4$ and 6 respectively to predict its class.

Let us now consider the effect of jitter and we show in Fig. 6(c) the performance of the proposed method when the intensity of jitter is varied. Next, we look into the performance of the proposed method when patterns with 50% empty afferents are considered. Fig. 8(a) depicts the results obtained by our network for this case with varying amounts of jitter.

C. Case 2: $n_{sub} > 1$

Next, we consider $n_{sub} = 5$ i.e. we divide each pattern into 5 subpatterns by setting $\tau_{s,inh}$ and $I_{0,inh}$ as per Equation 15. For $C$ class classification, the maximum number of subpatterns can be $C \times n_{sub}$ so we set $N = C \times n_{sub}$ in this case i.e. we keep $N = 10, 20$ and 30 for two, four and six-class classification respectively. Considering $\sigma_{jitter} = 0$, the evolution of $CM$ with epochs for two, four and six-class classification averaged over 50 trials is shown in Fig. 8(b). Moreover, the value of $ep_{sat,avg}$ (averaged over 50 trials) is found out to be 210, 221 and 230 when $C = 2, 4$ and 6 respectively. Unlike Case 1, here we consider the response to a pattern as a unique firing sequence of few NNLDs. As an example, we consider a particular trial of four-class classification and look into the first and last 3 epochs during its training. It is evident from Fig. 9 that during the first 3 epochs, WTA-NNLD produces arbitrary sequences of spikes. However, it can be seen that after the training of the network is complete, WTA-NNLD produces different firing sequences for different patterns while producing the same sequence when same patterns are encountered. WTA-NNLD trained by this method produces 100% accuracy in recognizing different patterns by producing its unique firing sequence for $C = 2, 4$ and 6. The performance of the network with varying intensity of jitter is shown in Fig. 6(c) (spikes present in all afferents) and Fig. 8(a) (spikes present in only half of the afferents) which depict that the $n_{sub} = 5$ case is less jitter resilient than the $n_{sub} = 1$ case.

We further increase the resolution of pattern subdivision by decreasing $\tau_{s,inh}$. We consider $n_{sub} = 10$ and following the principle of $n_{sub} = 5$, the number of NNLDs employed for $n_{sub} = 10$ are 20, 40 and 60 for two-class, four-class and six-class classification. This approach also provides 100% accuracy in providing a unique sequence of firing whenever a particular pattern is encountered when $\sigma_{jitter} = 0$. However, the performance of the network falls rapidly with the increase of $\sigma_{jitter}$ as shown in Fig. 6(c) and Fig. 8(a). We also show the evolution of $CM$ with the epochs in Fig. 8(b). Furthermore, the number of epochs needed for convergence of $CM$, in this case, is much more than the previous cases as depicted in Fig. 8(c). We conclude that dividing a pattern into too many subpatterns hampers the network performance.

Next we delve a bit further and show the statistics of causes for the failure of the system in producing successful trials. A trial may fail if either condition (a) or (b) (described in Section II-D) is not satisfied. We denote the failure of condition (a) as F1. Note that for a trial, condition (b) can fail if a pattern is misclassified as a pattern of another class (denoted as F2) or as a random pattern (denoted as F3). Table I shows the statistics of failed trials for $n_{sub} = 1, 5$ and 10 when $C = 4$ and $\sigma_{jitter}/\tau_s \approx 0.1$. Note that F1 is high for $n_{sub} = 1$ since a unique NNLD might lock onto multiple classes of patterns. F1 reduces for $n_{sub} = 5$ and increases again for $n_{sub} = 10$ since sometimes the network fails to produce a unique 10 indices long representation for all patterns of the same class.

Moreover, we test our network with random patterns and note the cases where a learned unique representation is produced for a random input pattern i.e. a false positive error occurs. The percentage of false positive errors produced for $n_{sub} = 1, 5$ and 10 when $\sigma_{jitter}/\tau_s \approx 0.1$ are 8%, 0% and 0% respectively. Note that no false positive errors occur for $n_{sub} = 5$ and 10 since it is highly unlikely for a random pattern
to make a sequence of neuron firing same as any of the learned representation.

<table>
<thead>
<tr>
<th>Case</th>
<th>$n_{sub} = 1$</th>
<th>$n_{sub} = 5$</th>
<th>$n_{sub} = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>12%</td>
<td>2%</td>
<td>6%</td>
</tr>
<tr>
<td>F2</td>
<td>2%</td>
<td>2%</td>
<td>2%</td>
</tr>
<tr>
<td>F3</td>
<td>2%</td>
<td>4%</td>
<td>8%</td>
</tr>
</tbody>
</table>

V. VLSI IMPLEMENTATION: EFFECT OF STATISTICAL VARIATION

In this section, we analyze the stability of our algorithm to hardware nonidealities by incorporating the statistical variations of the key subcircuits. The primary subcircuits needed to implement our architecture are synapse, dendritic squaring block, neuron and $e_{pj}^n$ calculator. While the variabilities of the synapse circuit are modeled by mismatch in the amplitude ($I_0$) and time constant ($\tau_s$) of the synaptic kernel function, the variabilities of the squaring block are captured by a multiplicative constant ($cb_{ni}$) [29]. We do not consider the variation of the inhibitory current kernel since it is global and only a single instance of it is present in the architecture. In our earlier work [29], [32], we proposed the circuits for implementing the synapse and squaring block of NNLD and performed Monte Carlo analysis to find their variabilities. We presented that the $\sigma_{n_{sub}=1}$ of $I_0$, $\tau_s$ and $cb_{ni}$ for the worst case scenario are 13%, 10.1% and 18% respectively. The mismatch of the LIF neuron circuit proposed in [32] was captured by variations in the firing threshold $V_{thr}$, the $\frac{\sigma}{\mu}$ of which was computed to be 12.5%. Lastly, the nonidealities of the $e_{pj}^n$ calculator block, described in [33], are modeled as a multiplicative constant ($cc_{ni}$). Monte Carlo analysis of the $e_{pj}^n$ calculator block revealed that its $\frac{\sigma}{\mu}$ for the worst case is 18%.

Figure 10 shows the performance of the proposed method when these nonidealities are included in the model for $n_{sub} = 1$ and $n_{sub} = 5$ keeping $\sigma_{jitter}/\tau_s \approx 0.1$. The bars corresponding to $I_0$, $\tau_s$, $cb_{ni}$, and $cc_{ni}$ denote the performance degradation when statistical variations of $I_0$, $\tau_s$, $cb_{ni}$, and $cc_{ni}$ are included individually. The results of Fig. 10(a) and Fig. 10(b) depict that the performance of the proposed algorithm is most affected by $\tau_s$ and $cc_{ni}$ and least by $cb_{ni}$. Finally, to mimic the proper hardware scenario, we consider the simultaneous implementations of all the nonidealities, which is marked by (...). The (...) bars show that there is an 8% and 6% decrease in performance for $n_{sub} = 1$ and $n_{sub} = 5$ respectively.

VI. CONCLUSION

We have proposed a new neuro-inspired Winner-Take-All architecture (WTA-NNLD) and a STDP inspired dendrite specific structural plasticity based learning rule (STDP-NRW) for its training. Motivated by recent biological evidence and models suggesting nonlinear processing properties of neuronal dendrites, we employ neurons with nonlinear dendrites to construct our WTA architecture. Moreover, we consider binary synapses instead of high-resolution synaptic weights. Hence, our learning rule, instead of weight updates, trains the network by modification of the connections between input and synapses. We have also provided a method by which the number of subpatterns per pattern learned by WTA-NNLD can be controlled. WTA-NNLD encodes patterns of different classes by either activity of distinct NNLDs or by a distinct sequence of NNLD firings. To demonstrate the performance of WTA-NNLD and STDP-NRW, we have considered two, four and six-class classification of 100 dimensional Poisson spike trains. We can conclude from the result that the slow time constant of inhibitory signal ($\tau_{s,inh}$) can be properly set to obtain a tradeoff between specificity and sensitivity of the network. Our immediate future work will include studying the effects of connection changes after the network gets integrated over multiple patterns. This will reduce the number of required computations. On another note, we will look into the classification of spike based MNIST [34], [35] datasets by our method. Our network can be immediately scaled to learn...
the digits of MNIST dataset, the only requirement being additional simulation time and computational memory compared to the tasks considered in this article. Furthermore, to achieve invariance to scaling and rotation during image classification, we will be constructing NNLD based convolutional neural networks [36] trained by structural plasticity. We will also implement the proposed network in hardware and apply it for real time online unsupervised classification of spatio-temporal spike trains.

REFERENCES

Fig. 10: Stability of WTA-NNLD trained by STDP-NRW is plotted with respect to different hardware nonidealities for $n_{sub} = 1$ (a) and $n_{sub} = 5$ (b). The constant red line indicates the percentage of successful trials obtained by our method without any nonidealities when $\sigma_{jitter}/\tau_s \approx 0.1$. The bars represent the percentage of successful trials obtained after inclusion of nonidealities. The rightmost bar marked by (...) represents the performance when all the nonidealities are included simultaneously.


