frontiers in Neuroscience

ORIGINAL RESEARCH
published: 31 March 2016
doi: 10.3389/fnins.2016.00113

Multiclass Classification by Adaptive Network of Dendritic Neurons with Binary Synapses Using Structural Plasticity

Shaista Hussain and Arindam Basu **

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore

The development of power-efficient neuromorphic devices presents the challenge of designing spike pattern classification algorithms which can be implemented on low-precision hardware and can also achieve state-of-the-art performance. In our pursuit of meeting this challenge, we present a pattern classification model which uses a sparse connection matrix and exploits the mechanism of nonlinear dendritic processing to achieve high classification accuracy. A rate-based structural learning rule for multiclass classification is proposed which modifies a connectivity matrix of binary synaptic connections by choosing the best "k" out of "d" inputs to make connections on every dendritic branch ($k << d$). Because learning only modifies connectivity, the model is well suited for implementation in neuromorphic systems using address-event representation (AER). We develop an ensemble method which combines several dendritic classifiers to achieve enhanced generalization over individual classifiers. We have two major findings: (1) Our results demonstrate that an ensemble created with classifiers comprising moderate number of dendrites performs better than both ensembles of perceptrons and of complex dendritic trees. (2) In order to determine the moderate number of dendrites required for a specific classification problem, a two-step solution is proposed. First, an adaptive approach is proposed which scales the relative size of the dendritic trees of neurons for each class. It works by progressively adding dendrites with fixed number of synapses to the network, thereby allocating synaptic resources as per the complexity of the given problem. As a second step, theoretical capacity calculations are used to convert each neuronal dendritic tree to its optimal topology where dendrites of each class are assigned different number of synapses. The performance of the model is evaluated on classification of handwritten digits from the benchmark MNIST dataset and compared with other spike classifiers. We show that our system can achieve classification accuracy within $1-2\%$ of other reported spike-based classifiers while using much less synaptic resources (only 7%) compared to that used by other methods. Further, an ensemble classifier created with adaptively learned sizes can attain accuracy of 96.4% which is at par with the best reported performance of spike-based classifiers. Moreover, the proposed method achieves this by using about 20% of the synapses used by other spike algorithms. We also present results of applying our algorithm to classify the MNIST-DVS dataset collected from a real spike-based image sensor and show results

OPEN ACCESS

Edited by:
Themis Prodromakis, University of Southampton, UK

Reviewed by:
Omid Kavehei, Royal Melbourne Institute of Technology, Australia
Hesham Mostafa, Institute for Neuroinformatics, Switzerland

***Correspondence:**
Arindam Basu
arindam.basu@ntu.edu.sg

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *06 September 2015*
Accepted: *07 March 2016*
Published: *31 March 2016*

Citation:
Hussain S and Basu A (2016) Multiclass Classification by Adaptive Network of Dendritic Neurons with Binary Synapses Using Structural Plasticity. Front. Neurosci. 10:113. doi: 10.3389/fnins.2016.00113

comparable to the best reported ones (88.1% accuracy). For VLSI implementations, we show that the reduced synaptic memory can save upto 4X area compared to conventional crossbar topologies. Finally, we also present a biologically realistic spike-based version for calculating the correlations required by the structural learning rule and demonstrate the correspondence between the rate-based and spike-based methods of learning.

Keywords: active dendrite, structural plasticity, binary synapses, multiclass classification, neuromorphic

1. INTRODUCTION

There has been significant research in the last decade aimed at designing neuromorphic systems which can emulate the architectural and computational principles of the brain. These systems exploit the spike-based operation of human brain, with minimal power consumption during long inactive periods, to implement power-efficient neuromorphic devices. Moreover, this attempt to mimic the neuronal function can enable us to design event-driven, compact hardware systems which can provide efficient, real-time, intelligent solutions for several applications like robotics and brain-machine interfaces. Conversely, the neuromorphic systems can be used to understand the working principles of brain. The development of event-driven sensors like the artificial retina (Lichtsteiner et al., 2008; Posch et al., 2011; Serrano-Gotarredona and Linares-Barranco, 2013) and cochlea (Liu et al., 2013), which produce continuous and asynchronous spikes encoding the sensory information, make it essential to interface these sensors with spike-based classifier systems to enable the classification of real-world complex stimuli. The spike classification algorithms designed to this effect can also attain large computational power of spiking neural networks (Maass and Schmitt, 1999).

The spike-based neuromorphic systems implemented in very-large-scale integration (VLSI) technology consist of hybrid analog-digital circuits, where the neuronal and synaptic computations are usually performed in analog form on the chip (though TrueNorth Merolla et al., 2014 and Spinnaker Painkras et al., 2013 are notable exceptions) while the synaptic connectivity information is stored on or off-chip in a digital memory. An asynchronous communication protocol called the address-event representation (AER; Boahen, 2000; Choi et al., 2005; Serrano-Gotarredona et al., 2009; Vogelstein et al., 2007), is used to transmit neuronal spikes between neuromorphic chips on a shared fast digital bus. The AER-based neuromorphic systems have the added advantage of reconfigurability since the configuration details of a network are stored in a separate memory, thereby giving the user flexibility to reconfigure the network connectivity.

However, the statistical variations in VLSI devices which reduce the accuracy of the synaptic weights are a major cause for concern in attaining performance comparable to software simulations. To mitigate the effect of increasing device mismatch with progressively shrinking transistor sizes, the usual solutions are to increase device sizes or employ a large number of neurons, both of which increase chip area. For example, a spiking network classifier implemented on a neuromorphic hardware system achieved performance comparable with standard machine learning linear classifier and exhibited tolerance against variability by using population coding (Schmuker et al., 2014). However, a limitation of this model is the large number of high resolution weights used to attain the reported performance. Similarly, spike classifiers consisting of Restricted Boltzmann Machine (RBM) constructed with integrate and fire neurons, use a large number of recurrent synaptic connections (Neftci et al., 2014; O'Connor et al., 2013) rendering these algorithms impractical for compact VLSI implementation. A simple and robust solution to the problem of device mismatch can be obtained by using binary synaptic weights. A spike-based STDP learning rule using bistable synapses was implemented in Brader et al. (2007) with the VLSI implementation in Mitra et al. (2009) to classify complex stimuli. A pool of neurons was used to improve the classification accuracy by employing a voting scheme, which again leads to the problem of increased number of synapses. Digital implementations do not suffer from mismatch issues like their analog counterparts; however, the usage of a lot of memory to implement high resolution weights for deep networks increases chip area significantly.

A spike classification model proposed in Hussain et al. (2013, 2015) offers a solution to the problem of large number of synaptic weights by using a structural plasticity based learning rule which involves formation of *sparse* connections with *binary* weights. Moreover, it was shown that a simple correlation-based learning rule provides an alternative to the traditional weight-based learning rules and is more suitable for implementation on neuromorphic chips. The problem of reduced memory capacity of a network with binary synapses as compared with that of continuous-valued synaptic weights (Senn and Fusi, 2005) is alleviated by the use of nonlinear dendritic processing, which emerges due to the presence of voltage gated ion channels (London and Hausser, 2005; Magee, 2000). A limitation of this method is that it uses a preassigned network size and the number of dendrites and synapses required for solving a classification problem of given complexity is not known. Hence, it is desirable to use an approach which learns to allocate the required number of synaptic resources for a specific problem.

Several adaptive approaches have been used to control the network size and structure. A constructive approach involves training with a minimal architecture, for example a single hidden-layered network with one hidden neuron, and then

adding further hidden units and weights to implement the desired mapping (Islam et al., 2009; Kwok and Yeung, 1997; Lahnajarvi et al., 2002). The second approach for automating the design of appropriate neural network is by pruning in which a network larger than necessary is trained and then redundant connections and/or neurons are removed until an acceptable solution is obtained. A group of pruning algorithms eliminate a neuron or a connection which have the least effect on the error function (Karnin, 1990). The other group of pruning algorithms are referred to as regularization methods which add a penalty term proportional to the sum of weights to the objective function. Hence, the unnecessary weights are driven to zero during training and are eliminated in effect (Kwok and Yeung, 1996). Several other pruning methods are reviewed in Islam et al. (2009) and Reed (1993). Another class of algorithms that are developed to control the network size and structure use a hybrid approach of combining the constructive and pruning methods (Fiesler, 1994). The growth and pruning algorithms discussed here have not considered the number of dendrites as an adaptive parameter.

In this paper, we present a multiclass classifier using neurons with nonlinear dendrites (NNLD) and a structural learning rule for finding sparse, binary weight matrices. The unique contributions of this work are: (1) Showing that for ensembles of NNLD with the same number of synapses, having a single dendritic branch (perceptron) or having too many dendrites are sub-optimal; the optimal cases is a moderate size of the dendritic tree. (2) Developing an algorithm that adapts the size of the dendritic tree for each class according to the difficulty of classifying that pattern category. (3) Applying this network to the problem of handwritten digit recognition task from MNIST and MNIST-DVS datasets to show its benefit in achieving high accuracy with very small memory usage for weights. (4) Demonstrating memory size reductions possible in VLSI implementations by using this training method.

The paper is organized as follows. First, the rate-based multiclass spike pattern classification model is presented in Section 2.1.1, followed by the description of an ensemble method which combines the outputs of several dendritic classifiers to obtain improved classification accuracy in Section 2.1.2. In Section 2.1.3, we propose an adaptive structural learning scheme which involves growth of the network by adding dendrites based on the progress of learning process. The performance of different learning schemes on classification of handwritten digit samples is demonstrated in Sections 3.2–3.4, while noise sensitivity analysis is in Section 3.5 and analysis of dendrite weights in Section 3.6. Next, in Section 3.7, we present an approach to optimize the performance of our method by utilizing theoretically determined optimal neuron topology. Finally, the evaluation of our algorithm on event-based MNIST-DVS dataset is presented in Section 3.8 followed by a comparison of the performance of our model with the results obtained using other spike-based classification algorithms in Section 3.9. The relevance of our work in terms of the biological plausibility, comparisons with other related studies, hardware considerations and a discussion of future work is included in Section 4.

2. MATERIALS AND METHODS

2.1. Spike Classification with Nonlinear Dendrites and Structural Learning

We have proposed a margin-based neuron model with nonlinear dendrites (NLD) for spike pattern classification (Hussain et al., 2015). The model comprises nonlinear neurons having lumped dendritic nonlinearity where a nonlinear neuron (NL-neuron) consists of multiple (m) dendritic branches with each branch governed by its nonlinearity $b()$ and k excitatory synapses on each branch driven by one of the d input components (**Figure 1A**). The model uses binary weight for the ith synapse on the jth dendrite, $w_{ij} \in \{0, 1\}$, where $i \in \{1, \cdots d\}$, $j \in \{1, \cdots m\}$. The advantage of such binary weight connections are in hardware implementations since they require much less memory resources, are more resistant to mismatch in analog implementations and also enable easy digital implementations as done in Merolla et al. (2014). Moreover, we enforce sparse connectivity on each dendritic branch ($k << d$) and allow the learning to choose the best "k" connections on each branch. Hence, we can write:

$$\sum_{i=1}^{d} w_{ij} = k \text{ for } j = 1, \cdots, m \tag{1}$$

This model was used to perform supervised binary classification of spike patterns. In our current work, we extend the model to perform multiclass classification of spike patterns belonging to N_C classes. The multiclass classifier consists of $(+)$ and $(-)$ neurons corresponding to all N_C classes. The inputs to the neurons for class μ are received from a pair of excitatory and inhibitory dendritic trees, also referred to as positive and negative dendritic trees (PDT and NDT), respectively, each with m dendrites. These input currents for class μ neurons are given by:

$$I_{in}^{\mu+}(t) = I_{PDT}^{\mu}(t) - I_{NDT}^{\mu}(t) \tag{2}$$

$$I_{in}^{\mu-}(t) = I_{NDT}^{\mu}(t) - I_{PDT}^{\mu}(t) \tag{3}$$

where $\mu \in \{1, \cdots N_C\}$ and $I_{PDT}^{\mu}(t)$, $I_{NDT}^{\mu}(t)$ are the currents generated by the PDT and NDT of class μ, calculated by taking the sum of dendritic output currents $I_{PD,j}^{\mu}(t)$ and $I_{ND,j}^{\mu}(t)$ respectively, which can be expressed as nonlinear functions of the total synaptic current on a dendrite. To make notations simpler for the ease of reading, we drop the superscript μ for the μth class in the rest of the paper except where the outputs of two or more classes are compared. Hence, we denote the PDT/NDT currents for class μ neuron as $I_{DT}(t)$ and the output current of the jth dendrite of PDT/NDT as $I_{D,j}(t)$. Therefore, the equations for $I_{DT}(t)$ can be written as:

$$I_{DT}(t) = \sum_{j} I_{D,j}(t) \tag{4}$$

$$= \sum_{j} b\left(\sum_{i} w_{ij}\left(\sum_{t_{ij}<t} K(t - t_{ij})\right)\right) \tag{5}$$

$$= \sum_{j} b(z_j(t)) \tag{6}$$

Here $K(t)$ denotes the postsynaptic current (PSC) generated by the spike input at a synapse of the PDT or NDT of class μ and is given by Gutig and Sompolinsky (2006):

$$K(t - t_{ij}) = I_0 \left(exp[-(t - t_{ij})/\tau_f] - exp[-(t - t_{ij})/\tau_r] \right) \quad (7)$$

where t_{ij} denotes the times at which spikes arrive at this synapse; I_0 is the normalization constant; and τ_f and τ_r are the fall and rise time constants of the PSC respectively; $z_j(t)$ is the total synaptic activation on the jth dendrite and $b(z_j(t))$ is the dendritic nonlinear function given by:

$$b(z_j(t)) = g(z_j(t) - z_{leak,j}(t))(z_j(t) - z_{leak,j}(t))^2 \quad (8)$$

Here, $g()$ is a Heaviside step function that gives an output 1 at all times where the argument $(z_j(t) - z_{leak,j}(t))$ is positive and 0 otherwise and $z_{leak,j}(t)$ is the average synaptic activation on the jth branch corresponding to the initial random connections. The use of $z_{leak,j}(t)$ term serves to balance the excitation on each branch by subtracting the mean activation level on that branch, which can be regarded as a signal from a pool of inhibitory neurons. Finally, the spike output n_{spk} of a neuron of class μ, receiving the input current $I_{in}(t)$, is generated using the leaky integrate and fire neuron model described next.

The dynamics of the membrane potential $V_m(t)$ of the (+) and (−) neurons of class μ is explained through the equations given below:

$$\tau_V \frac{dV_m}{dt} = (u - V_m) + I_{in}(t) \quad (9)$$

$$\tau_u \frac{du}{dt} = -u \quad (10)$$

$$\text{If } V_m \geq V_{thr}, V_m \rightarrow V_{reset};$$
$$u \rightarrow u_{reset}$$
$$u_{reset} = V_{reset} < 0$$
$$n_{spk} \rightarrow n_{spk} + 1$$

where u denotes a hyperpolarization variable which relaxes back to 0 with a time constant τ_u and is set to u_{reset} after a postsynaptic spike and τ_V and τ_u are the time constants governing the fast and slow dynamics of the membrane voltage and hyperpolarization respectively. Here, the variables $V_m(t)$, u and n_{spk} are used to describe the dynamics of both (+) and (−) neurons of class μ and the input current $I_{in}(t)$ can be computed using Equations (2) and (3).

As shown in **Figure 1B**, the outputs of all the (+) and (−) neurons are connected to a WTA circuit to generate the overall classifier output as:

$$y_\mu = g(o^\mu_{spk} - o^\nu_{spk}),\ o^\nu_{spk} \geq o^\xi_{spk}, \forall \nu, \xi \neq \mu \quad (11)$$

where μ, ν, ξ are integers in the range $[1, N_C]$ and y_μ is the binary-valued μ^{th} component of the N_C-dimensional output y, $o^\mu_{spk} = n^{\mu+}_{spk} - n^{\mu-}_{spk}$; $n^{\mu+}_{spk}$ and $n^{\mu-}_{spk}$ are the spike counts of the (+) and (−) neurons of class μ respectively.

In this paper, we have developed rate-based and spike-time-based learning schemes for our multiclass classification model with NLDs. For these schemes using binary synapses, we present structural learning rules, which involve the modification of connection matrix instead of connection weights. The motivation for using rate-based learning is driven by the fact that the spike-time-based learning requires significant processing time and therefore, to mitigate this problem we have used the strategy of training on mean rate inputs to reduce the training time. With the use of faster rate-based learning, we further propose an ensemble method which combines the outputs of several NLD classifiers in order to achieve performance gain over individual classifiers. We also develop an adaptive learning scheme which learns to allocate the required number of dendrites to a neural network along with learning the connection patterns on these dendrites. The bio-realistic spike-based approximation of the structural learning rule for multiclass classification is an extension of an online spike-based binary classification rule proposed in our earlier work (Hussain et al., 2015). The derivation of this rule and the correspondence between our spike-based and rate-based learning schemes are presented in the Appendix. Next, we describe the rate-based learning rule.

2.1.1. Rate-Based Learning Scheme for Multiclass Classification

The rate-based learning rule for NLD model is valid for rate encoded inputs like Poisson spike trains and place/synchrony encoded single spike patterns. This validity results from the fact that for such spike inputs, the average synaptic activation $z_{syn,ij} = \frac{1}{T}\int_0^T \sum_{t_{ij}} K(t - t_{ij})dt$ is directly proportional to the input arriving at that synapse, x_{ij}, where T is the pattern duration (Hussain et al., 2015). This reduced model was developed to improve the training time and it uses encoded binary vectors obtained by mapping spike train with "high" firing rate or single spike to binary value "1" and spike train with "low" firing rate or absence of spike to binary value "0." A rate-based NLD model for multiclass classification was proposed in Hussain et al. (2014), which consisted of N_C neurons representing N_C classes. Here, we present a modified version of this model consisting of PDT and NDT corresponding to each of the N_C classes. The output of the PDT and NDT of class μ to input x is given by:

$$a_{PDT}(x) = \sum_{j=1}^{m} b\left(\sum_{i=1}^{d} w_{PDT,ij} x_{PDT,ij} \right) \quad (12)$$

$$a_{NDT}(x) = \sum_{j=1}^{m} b\left(\sum_{i=1}^{d} w_{NDT,ij} x_{NDT,ij} \right) \quad (13)$$

The difference of outputs of the positive and negative dendritic trees of each class, $o(x) = (a_{PDT}(x) - a_{NDT}(x))$, are then used as inputs to the WTA circuit which computes the overall output and therefore decision of the classification task, $y(x)$ using a similar logic as in Equation (11).

$$y_\mu(x) = g(o^\mu(x) - o^\nu(x)),\ o^\nu(x) \geq o^\xi(x), \forall \nu, \xi \neq \mu \quad (14)$$

During training, the classifier output is computed using a margin-based output function, $g_{margin}(x)$ instead of $g(x)$ to enforce a

margin around the classification boundary. The $g(x)$ function is only used during the testing phase to calculate the output. Therefore, the classifier output during training calculated using the $g_{margin}(x)$ function is given by:

$$y_\mu(x) = g_{margin}(o^\mu(x) - o^\nu(x)),\ o^\nu(x) \geq o^\xi(x)$$
$$y_\nu(x) = g_{margin}(o^\nu(x) - o^\mu(x)), \forall \nu, \xi \neq \mu \quad (15)$$

The function $g_{margin}()$ is defined as:

$$g_{margin}(\alpha) = 1 \text{ if } \alpha \geq \delta$$
$$= 0 \text{ if } \alpha \leq -\delta$$
$$= \frac{0.5}{\delta}\alpha + 0.5 \text{ otherwise} \quad (16)$$

where the margin is set using different values of the parameter δ for different classes to which input patterns belong. The value of margin δ for each class is determined using the following steps:

(1) Multi-class model is first trained using the $g(x)$ function, where the output of the model is calculated using 14. Connection matrices for all neurons are saved.
(2) Cross validation set patterns are presented and the values $\alpha_{\mu\nu} = (o^\nu(x) - o^\mu(x))$ are recorded for all the cases for which patterns from class μ are misclassified as belonging to class ν.
(3) The set of $\alpha_{\mu\nu}$ values for each class $\mu \in \{1, \cdots N_C\}$ are saved. The margin for class μ, δ^μ is set to the highest value of $\alpha_{\mu\nu}$.
(4) δ^μ value is reduced to 80% of its present value whenever learning algorithm gets stuck in the same local minimum for 5 consecutive times.

The learning process which is based on the mechanism of structural plasticity involves formation and elimination of synaptic connections. The connections are modified by computing a metric based on correlation values c_{ij} for each synapse of the PDT and NDT. Hence, the learning rule for the rate-based learning scheme is given by:

For class μ, for PDT:

$$c_{ij}^{PDT} = < x_{ij}^{PDT} b_j^{PDT} sgn(y_\mu^d - y_\mu) > \quad (17)$$

For class μ, for NDT:

$$c_{ij}^{NDT} = - < x_{ij}^{NDT} b_j^{NDT} sgn(y_\mu^d - y_\mu) > \quad (18)$$

where the desired output y^d is available as the teacher signal during the training phase and is a N_C-dimensional binary vector consisting of $(N_C - 1)$ zeros and a 1 corresponding to the class to which the input pattern belongs; b_j is the output of the j^{th} dendrite of a PDT or NDT; $sgn()$ is the signum function with a value of 1 for $y_\mu^d > y_\mu$, -1 for $y_\mu^d < y_\mu$ and 0 for $y_\mu^d = y_\mu$; and the output y is computed using Equation (15).

The connection changes are done by using the following logic: a synapse with the smallest c_{ij} value corresponds to a poorly-performing synapse and is a candidate for replacement. To replace this synapse, a set of silent synapses are first formed on the chosen branch as possible candidates for the new connection. The silent synapse with the highest c_{ij} is chosen as the replacement.

At the start of the learning process, the input connections for all the PDT and NDT corresponding to N_C classes each with m dendritic branches, k synaptic contacts per branch and $s = m \times k$ total synapses are initialized by randomly selecting afferents from among d input lines with weight $w_{ij} = 1$. Training set consisting of P input patterns (x) belonging to N_C classes is presented. The learning process comprises the following steps in every iteration consisting of the presentation of all P patterns:

(1) The misclassification error rate is calculated by taking the average of the fraction of patterns for which $y_\mu \neq y_\mu^d$ for any $\mu \in \{1, \cdots N_C\}$.
(2) For each dendritic tree of class μ, a random set T consisting of n_T $(< s)$ synapses is selected as candidates for replacement.
(3) The synapse with the lowest c_{ij} in T is targeted for replacement and the dendrite j_T on which it is located is identified. A random replacement set R is created by placing n_R "silent" synapses from d input lines $(n_R < d)$ on the j_T^{th} dendrite. The synapse with the lowest c_{ij} in T is replaced by the best-performing (maximum c_{ij}) synapse in R. The silent synapses do not contribute to the calculation in step (1).
(4) Synaptic connections are modified if the replacement led to either a reduction or no change in error rate. If the error increased with the replacement, a new replacement set R is created. If the error does not decrease after repeating this step n_{ch} $(= 50)$ times, we assume a local minimum is encountered. We then do a replacement with the last choice of R, even if it increases error in an attempt to escape the local minimum. The connection matrix corresponding to the local minimum that gave the least error is saved.
(5) Steps (1)–(4) are repeated until either all the patterns have been memorized or n_{min} $(= 150)$ local minima are encountered. At this point, the learning process is stopped. The connection patterns corresponding to the best minimum become the final learned synaptic connections of the neuron.

After the completion of training, the final learned connections are saved and then used to calculate the error rate on the spike test pattern set for the testing phase. This is done by mapping the learned connections onto an equivalent spiking network by introducing synaptic integration and spike initiation mechanisms. The testing classification output is computed using Equation (11).

2.1.2. Ensemble Learning Scheme for Multiclass Classification

We have also used an ensemble method in this work, where several classifiers when combined together yield better classification accuracy than any of the single classifiers in the ensemble. Our ensemble model consists of individually trained NLD classifiers which are then combined to classify novel test patterns. Since, previous research has demonstrated that combining identical classifiers doesn't produce any gain over individual classifiers (Opitz and Maclin, 1999), hence we created ensemble by combining several classifiers which are individually trained and disagree on their predictions. The complementary information about the novel patterns obtained from different classifiers can be exploited to produce a more

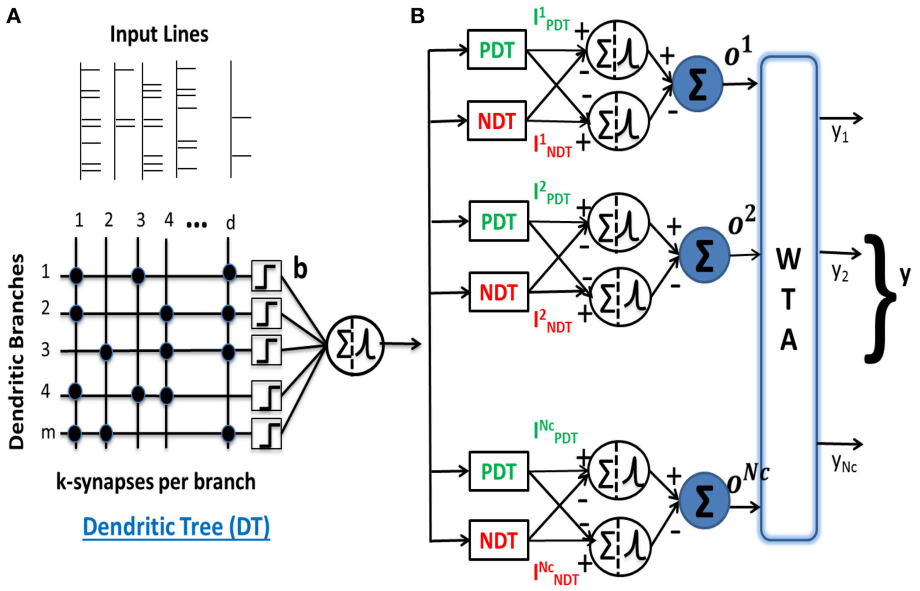


FIGURE 1 | (A) A neuron consisting of nonlinear dendrites characterized by the lumped nonlinear function $b()$ for every dendritic branch. It forms k binary connections from d input afferents on m dendrites. **(B)** Architecture of a multiclass pattern classifier consisting of nonlinear neurons receiving inputs from PDT and NDT and the neurons connected to a WTA to compare the outputs of the N_C classes. The output of the classifier is equal to 1 for the class corresponding to the highest output o^{μ}.

accurate overall output. In order to generate different predictions for different classifiers, individual networks were initialized with different random synaptic connections. **Figure 2** shows the basic framework for the classifier ensemble scheme. It consists of N individually trained multiclass classifiers as members of the ensemble. Each classifier generates the intermediate output, $o_n^{\mu}(x)$, which is the difference in outputs of PDT and NDT for class μ of the n^{th} classifier. The intermediate outputs are combined in a class-specific manner to give $O^{\mu}(x) = \sum_{n=1}^{N} o_n^{\mu}(x)$. Finally, the ensemble output is generated using:

$$\hat{y}_{\mu}(x) = g\left(O^{\mu}(x) - O^{\nu}(x)\right),\ O^{\nu}(x) \geq O^{\xi}(x), \forall \nu, \xi \neq \mu \quad (19)$$

The performance of ensemble model gives us an insight into the number of component classifiers needed in an ensemble. However, it is impractical to use a large number of classifiers in an ensemble due to the long training time required and also because it leads to increased synaptic resources. Therefore, we use the ensemble created with a few classifiers as a trade-off between trainability and accuracy. Moreover, it is not clear what level of complexity is required for individual classifiers and whether an ensemble of perceptrons will perform better than that of complex dendritic trees. To address this problem, we have developed an adaptive learning rule for allocating the number of dendrites according to the difficulty of the problem being solved as well as the difficulty to learn a specific class. This scheme is discussed next.

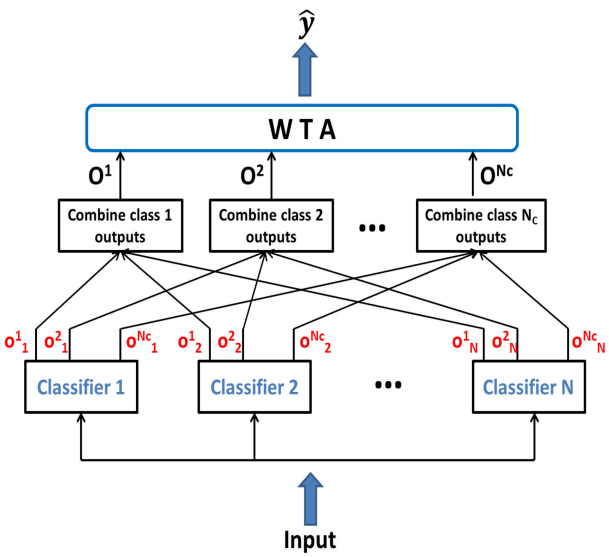


FIGURE 2 | Ensemble classifier combining the intermediate class-specific outputs o_n^{μ} to compute the combined class outputs O^{μ}, which are compared by the WTA to generate the final classifier output $\hat{y}$.

2.1.3. Adaptive Learning Scheme for Multiclass Classification

For the learning schemes described above, a fixed number of dendrites (m) was chosen and assigned to all the dendritic trees in the model. Since, it is difficult to choose an optimal value of m that matches the complexity of a given problem, we have proposed an algorithm in which the number of dendrites corresponding to each class is learned during the training process. The value of m is adapted for each class dendritic trees independently while the number of synapses per branch, k, is kept

constant throughout. Hence, the learning process generates the relative sizes of the connection matrix for each neuron as well as the optimal sparse connections within that matrix. The learning rule used is same as that for the rate-based scheme with fixed m except that all the computations in every iteration are done for the current value of m for each neuron. Additional steps are included in the learning algorithm to adapt the value of m. After making a correlation-based connection change, we check if the learning process for a particular class μ has slowed down using the following steps:

(1) If the error rate for class μ, which is calculated as the average of the number of patterns for which $y_\mu \neq y_\mu^d$, does not decrease after n_{ch} $(= 50)$ iterations, the learning process for class μ has encountered a local minimum. In this case, a new dendrite is added to both PDT and NDT corresponding to class μ, if the error is the highest amongst all the N_C classes.

(2) This is done by drawing k random connections from the d input lines to represent a new dendrite. The connection matrix is updated by appending the new dendrite to the existing matrix as a new row. The value of m for class μ is incremented according to:

$$\begin{aligned} m^{PDT} &\rightarrow m^{PDT} + 1 \\ m^{NDT} &\rightarrow m^{NDT} + 1 \end{aligned} \tag{20}$$

(3) After adding a dendrite, error rate for cross validation set is computed. If this validation error increases in the last 3 dendrite addition steps, the learning process is stopped. The connections are frozen and saved for the testing phase.

The above steps are added after the step (4) of the rate-based learning algorithm in Section 2.1.1 as steps 4.1, 4.2, and 4.3. All the steps from (1)–(4.3) are then repeated until learning stops by one of the stopping conditions discussed in step (5).

This adaptive learning method is guided by the level of difficulty of each neuron's classification task and moreover, it is used to learn the relative size of each neuron and not the optimal neuron topology. While performing this adaptation, we keep the number of synapses per branch, k, to be a constant for all neurons and just vary the number of dendrites per neuron, m. Hence, at the end of the adaptation, both PDT and NDT of the μth class have a total number of synapses s^μ_{adapt} given by $s^\mu_{adapt} = m^\mu_{adapt} \times k$, where m^μ_{adapt} is the number of dendritic branches learned for the μth class neurons after adaptation. This is shown as the step-1 in **Figure 3**, where the topology of a class neuron is denoted by the rectangle with its sides representing m and k values.

However, from theoretical considerations of function counting as shown in Poirazi and Mel (2001), the theoretical capacity of a neuron with nonlinear dendrites with a fixed value of total number of synapses (s_{opt}) is maximum for a relatively large value of m_{opt} and small value of k_{opt}. This theoretical capacity was given by the combinatorial expression derived by counting all possible ways in which d afferents can connect to synapses on dendrites resulting in distinct memory fields (Poirazi and Mel, 2001). Hence, for a neuron with NLDs, the capacity C_{NL} in bits was calculated as:

$$C_{NL} = 2log_2\binom{f+m-1}{m} \tag{21}$$

where $f = \binom{k+d-1}{k}$ is the number of distinct branch functions. As shown in the inset plot for the optimal topology determination step in **Figure 3**, C_{NL} plotted as a function of m has a maximum at m_{opt} for a fixed value of s_{opt}. These derivations were also discussed in our previous work (Hussain et al., 2015). We can now use this theory to change the structure of the μth neuron to optimize capacity while preserving the same number of synapses. Hence, by setting $s^\mu_{opt} = s^\mu_{adapt}$ (fixing the area of rectangle as shown in step-2 in **Figure 3**), the theoretical optimal topology $(m^\mu_{opt}, k^\mu_{opt})$ is determined such that $s^\mu_{opt} = s^\mu_{adapt} = m^\mu_{opt} \times k^\mu_{opt}$. The use of this approach to boost the capacity of our adaptively learned dendritic network is demonstrated in Section 3.7.

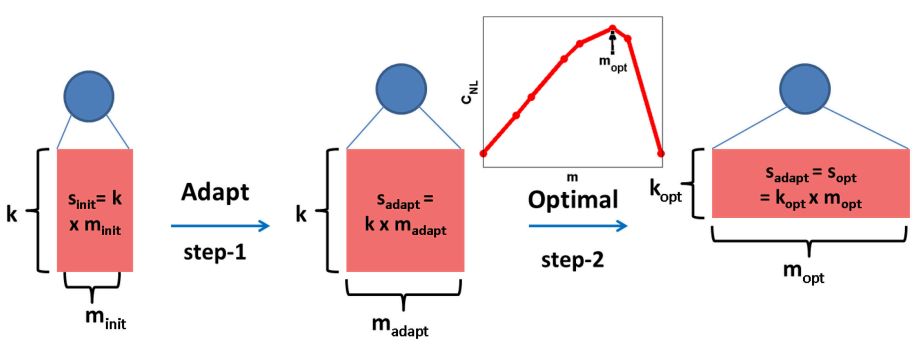


FIGURE 3 | Dendritic classifier is trained adaptively by adding dendrites (step-1) and keeping the number of synapses/dendrite (k) as a constant for all neurons. The total number of synapses for the μth neuron learned in this manner (s^μ_{adapt}) is fixed and the corresponding optimal topology (m^μ_{opt}, k^μ_{opt} inset plot) is theoretically determined (step-2). The length of sides of the rectangle denote m and k values and the area represents the total number of synapses, s.

3. RESULTS

3.1. Input Generation

We have demonstrated the performance of our multiclass classification NLD model on the MNIST dataset consisting of grayscale images of handwritten digits belonging to one of the $N_C = 10$ classes (0 to 9). It is a benchmark machine learning dataset used previously to test the performance of many classification algorithms (LeCun et al., 1998). The input patterns consisting of 28×28 images were converted into $d = 784$ dimensional binary vectors by thresholding. The training set consists of 20,000 patterns randomly selected from the full MNIST dataset, with equal number of samples of each digit. The testing set consists of a total of 10,000 patterns. In case of adaptive learning scheme, 20% of the training patterns comprise the cross validation set which is used to compute the error whenever a dendrite is added to the network and finally to stop adding dendrites when the cross validation error doesn't reduce further. The learning procedure is stopped when either all the patterns are correctly classified or when 150 minima are encountered. For different training runs 150 minima are encountered in different number of learning iterations and for each data point generated, this training process is repeated 3 times.

The binary input vectors are used for rate-based training. For spike-based training, patterns of single spikes are used, which are generated by mapping binary input "$x_i = 1$" to a single spike arriving at $T_{syn} = 100$ ms and "$x_i = 0$" to no spike, where the stimulus duration $T = 200$ ms. Testing for both forms of rate-based and spike-based learning is done on two types of spike inputs: 1) place/synchrony code of single spikes with jittered spikes arriving within a time window $[T_{syn} - \frac{\Delta}{2}, T_{syn} + \frac{\Delta}{2}]$ corresponding to binary input "1" and no spike for "0" input and different amounts of jitter Δ; and 2) rate encoded Poisson spike trains with mean firing rates of f_{high} and f_{low} mapped to binary inputs "1" and "0" respectively. The Poisson spike inputs have only been used to test the noise sensitivity of the model. The parameters used for the rate-based and spike-based models are given in **Tables 1**, **2** respectively, unless stated otherwise.

The significance of using these spike inputs can be understood by considering that single spike representations are commonly used in time-to-first-spike (TTFS) imagers (Chen and Bermak, 2007; Qi et al., 2004). These imagers when presented with binary images such as in the MNIST dataset used in this work, generate a cluster of spikes corresponding to the white pixels and another cluster much later in time corresponding to the black pixels. Such place/synchrony code is also abundantly used in neuroscience (Gerstner and Kistler, 2002). Moreover, rate encoded outputs are commonly available from neuromorphic sensors such as the artificial cochlea in Chan et al. (2007). Such rate encoded Poisson spike trains are also often used to test the performance of neuromorphic classifiers as demonstrated in Marti et al. (2015), O'Connor et al. (2013), and Brader et al. (2007).

3.2. Effect of Training Set Size

In the first experiment, we measured the effect of training set size on the classification performance by training the model on $P = 200 - 20,000$ patterns using the rate-based learning scheme. Each training accuracy is obtained by averaging over 3 trials and the testing accuracy is computed by averaging across 10 presentations of 10,000 test patterns for each of the learned network, where testing is done on jittered single spike inputs with $\Delta = 10$ ms. The training was done on a network consisting of $m = 10$ dendrites in the PDT and NDT of all $N_C = 10$ classes. Hence, the total number of dendrites used is $M = 200$. As shown in **Figure 4**, for the rate-based learning, the training error increases while the testing error on spike versions of the inputs reduces as the training set size increases. It is clear from this result that as more data is added, it becomes difficult to memorize the data. However, the generalization performance improves with more training data. A comparison between the spike-based and rate-based learning schemes is included in the Appendix, where we have shown that we can achieve similar performance for these two forms of learning. However, due to the long simulation times of the spike-based learning, we have used rate-based learning for the remaining analyses in the paper.

TABLE 1 | Parameters for rate-based multiclass model.

d	m	k	n_R	n_T	T_{syn}
784	10	10	25	25	100 ms

TABLE 2 | Parameters for spike-based multiclass model.

τ_V	τ_u	τ_{pre}	τ_{post}	V_{thr}	T	T_{syn}
5 ms	200 ms	10 ms	200 ms	0.1 mV	200 ms	100 ms

3.3. Performance of the Ensemble Method

Next, we studied the effect of combining several NLD multiclass classifiers in an ensemble. These individual classifiers were trained on $P = 5000$ patterns using rate-based learning rule and tested on 10,000 jittered single spike inputs with $\Delta = 10$ ms. The network consisted of $m = 2, 5, 8, 15$, and 20 dendrites in PDT and NDT for each of $N_C = 10$ classes. Hence, the total number of dendrites used in the ensemble is given by $M = 2 \times m \times N_C \times N$. As shown in **Figure 5A**, the error rate for $m = 2$ reduces by about 48% when up to 25 classifiers are added in the ensemble. However, the error rate doesn't change by much or it increases slightly if further classifiers are combined. Moreover, most of the error reduction, 40% out of the total change of 48%, is achieved when first 5 classifiers are added. We have also looked at the effect of size of individual networks in the ensemble. **Figure 5B** illustrates this effect where error rate is plotted as a function of the total number of dendrites in the ensemble, M. As shown, the error rate reduces with the number of classifiers for all values of m. For a fixed value of M, $m = 2$ gives the highest error rate, which reduces with larger values of m. However, as m is increased beyond a certain value ($m = 8$ in this case, the reduction in errors is not significant, and therefore the use of more complex dendritic trees is not leading to significant advantage in terms of performance.

These results indicate that the ensemble model can attain reasonably good performance when the member classifiers are trained using a moderate number of dendrites. In an ideal scenario, we can use a large number of classifiers to achieve significant improvement in performance. However, it is impractical to train several classifiers and hence, we use only a few classifiers which result in most of the performance gain of the ensemble. In case of limited resources (total number of dendrites M), an intermediate level of complexity of the ensemble comprising a moderate number of member classifiers with a moderate number of dendrites in each classifier can be used. Next, we investigate how to determine this intermediate level of complexity for a particular classification problem. For an ensemble created with a few classifiers (3−5), we do not know the number of dendrites required by each classifier and also further, if different levels of network complexity underlie representation of different classes to which data belongs. Therefore, we use the adaptive learning rule to automatically learn the number of dendrites suitable for each class of a given classification problem.

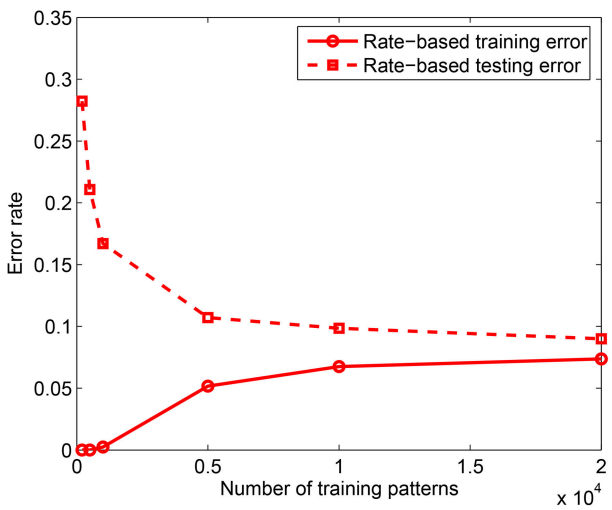


FIGURE 4 | Training and testing error rates for rate-based learning as a function of the training set size. Test performance is measured on 10, 000 jittered single spike input patterns with $\Delta = 10$ ms. Each data point is obtained by averaging over 3 training simulation trials and testing on 10 randomly drawn test sets for each learned network; $m = 10$ and $k = 10$.

3.4. Performance of the Adaptive Learning Scheme

We have used the adaptive learning scheme in which $P = 20,000$ patterns were used for training by initializing the network with $m = 5$ dendrites in each dendritic tree of a class, and then adaptively increasing m in a class-specific manner. The learning process was stopped when 150 minima were encountered and therefore, each adaptively learned classifier took different number of iterations to complete 150 minima. If dendrites are added to only 5 worst-performing classes whenever their learning slows down, referred to as scheme-1, the accuracy obtained on 10, 000 test single spike inputs is 92.1%. The number of dendrites learned by each class in one simulation run is shown by the bar plot at the top of **Figure 6A**. It can be seen that the digits "2," "3," "5," "8," and "9" use most of the dendrites and hence are most difficult to learn while "0," "1," and "6" are the easier ones requiring only a small number of dendrites. The confusion matrix at the bottom of **Figure 6A** shows the classification accuracy when an actual digit (column-wise) is predicted by the model as represented row-wise. We can see that the neurons for easier digits "0," "1," and "6" can attain good accuracy by utilizing small number of dendrites whereas the difficult digits like "8" and "9" exhibit lower accuracy, where "9" is mostly misclassified as "4" and "7" having similar features.

A drawback of scheme-1 is that we are not allocating the resources appropriately in this case since adding dendrites to only "difficult to learn" classes might not be helpful beyond a point and adding dendrites to the easier classes also can allow the learning to converge faster and attain better performance. This idea was supported by our results obtained by adding dendrites to all the

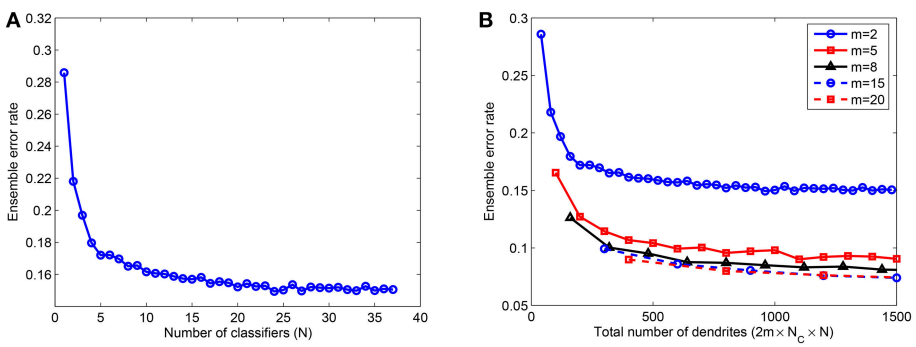
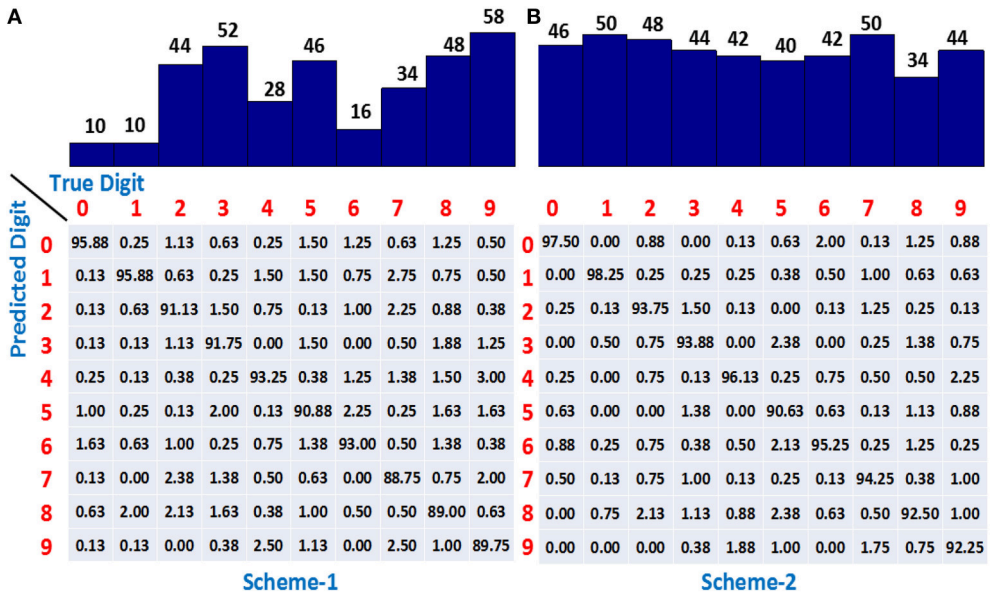


FIGURE 5 | (A) Ensemble error rate as a function of the number of classifiers combined, N, for 10, 000 test samples; $m = 2$ and $k = 10$. **(B)** Ensemble error rate as a function of the total number of dendrites used, M, showing that a moderate number of dendrites per dendritic tree ($m = 8$) gives reasonably good performance and there is no significant improvement when m is further increased. Training done on $P = 5000$ patterns randomly drawn from the MNIST samples. Each data point is obtained by randomly selecting a subset (N) of the total number of learned classifiers for each case of m (37 for $m = 2$). This process is repeated 10 times and the average of the test errors of the combined classifiers is computed.



Predicted Digit \ True Digit	0	1	2	3	4	5	6	7	8	9
0	95.88	0.25	1.13	0.63	0.25	1.50	1.25	0.63	1.25	0.50
1	0.13	95.88	0.63	0.25	1.50	1.50	0.75	2.75	0.75	0.50
2	0.13	0.63	91.13	1.50	0.75	0.13	1.00	2.25	0.88	0.38
3	0.13	0.13	1.13	91.75	0.00	1.50	0.00	0.50	1.88	1.25
4	0.25	0.13	0.38	0.25	93.25	0.38	1.25	1.38	1.50	3.00
5	1.00	0.25	0.13	2.00	0.13	90.88	2.25	0.25	1.63	1.63
6	1.63	0.63	1.00	0.25	0.75	1.38	93.00	0.50	1.38	0.38
7	0.13	0.00	2.38	1.38	0.50	0.63	0.00	88.75	0.75	2.00
8	0.63	2.00	2.13	1.63	0.38	1.00	0.50	0.50	89.00	0.63
9	0.13	0.13	0.00	0.38	2.50	1.13	0.00	2.50	1.00	89.75

Predicted Digit \ True Digit	0	1	2	3	4	5	6	7	8	9
0	97.50	0.00	0.88	0.00	0.13	0.63	2.00	0.13	1.25	0.88
1	0.00	98.25	0.25	0.25	0.25	0.38	0.50	1.00	0.63	0.63
2	0.25	0.13	93.75	1.50	0.13	0.00	0.13	1.25	0.25	0.13
3	0.00	0.50	0.75	93.88	0.00	2.38	0.00	0.25	1.38	0.75
4	0.25	0.00	0.75	0.13	96.13	0.25	0.75	0.50	0.50	2.25
5	0.63	0.00	0.00	1.38	0.00	90.63	0.63	0.13	1.13	0.88
6	0.88	0.25	0.75	0.38	0.50	2.13	95.25	0.25	1.25	0.25
7	0.50	0.13	0.75	1.00	0.13	0.25	0.13	94.25	0.38	1.00
8	0.00	0.75	2.13	1.13	0.88	2.38	0.63	0.50	92.50	1.00
9	0.00	0.00	0.00	0.38	1.88	1.00	0.00	1.75	0.75	92.25

FIGURE 6 | (A) The number of dendrites learned using adaptive scheme-1 (top) and the resulting confusion matrix (bottom) showing higher accuracy for easier classes (0, 1, 6) and lower accuracy for difficult digits (8, 9). **(B)** Number of dendrites learned using adaptive scheme-2 (top) demonstrating all the class neurons adding dendrites when required. Confusion matrix (bottom) showing improved accuracy for all the classes. Training set consists of 20, 000 binary digit samples and testing done on 10, 000 jittered single spikes ($\Delta = 10$ ms). Model was initialized with $m = 5$ dendrites for all the classes.

classes, referred to as scheme-2, which yielded a classification accuracy of 94.2%. The number of dendrites learned by all classes is shown at the top of **Figure 6B**, indicating that the dendrites are added to not only the difficult classes but the easier digits like "0," "1," and "6" also utilize more resources. The confusion matrix shows that the classification accuracy of almost all the digits increase as compared to scheme-1, thereby contributing to the overall improved performance.

3.5. Test of Noise Sensitivity

We have also tested the performance of an ensemble created with classifiers learned using the adaptive scheme-2, in the presence of noisy spike inputs. **Figure 7** shows the classification accuracy for 10, 000 test patterns consisting of single spike inputs with different levels of jitter Δ and Poisson spike inputs with $f_{high} = 250$ Hz and $f_{low} = 1$ Hz. For reference, we also present the testing accuracy obtained when non-spiking binary inputs are used (solid blue), which increases from 94.6% to 96.1% by adding just 3 classifiers to the ensemble and further increases to about 96.4% by adding 2 more classifiers. The accuracy for single spikes with no noise ($\Delta = 0$) is about $0.2 - 0.5\%$ less than that of binary vectors, which can be attributed to the effects of integrate and fire process of the neurons in the model. As noisy single spikes are presented, accuracy reduces further. However, the change is only about 0.5% for $\Delta = 20$ ms as compared to $\Delta = 0$. Further, the performance of the ensemble on spiking inputs with and without noise become very similar when just 3 dendritic classifiers are combined together. Similarly, for Poisson spike inputs the ensemble yields similar classification accuracy as in the absence of noise. Hence, the ensemble consisting of a few classifiers where each classifier is individually trained using the adaptive learning scheme offers the following advantages: (1) performance gain; (2) feasible to train few classifiers; (3) automatically learned network complexity; (4) noise sensitivity

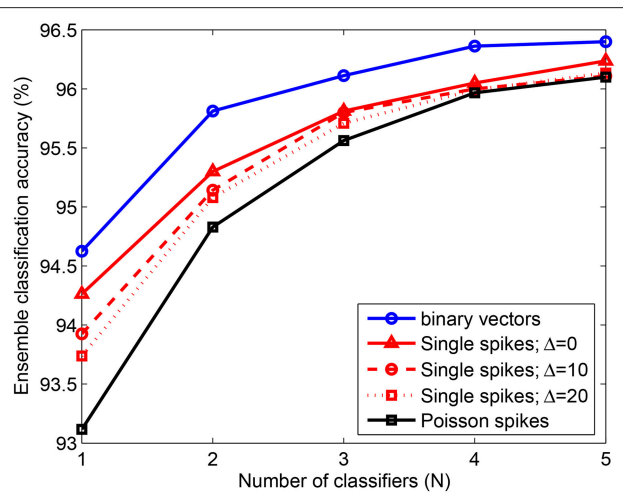


FIGURE 7 | Error rate of ensemble consisting of NLD classifiers trained on $P = 20,000$ MNIST samples using the adaptive scheme-2. The ensemble accuracy on 10, 000 test patterns is computed for binary vectors and spike inputs with different levels of noise. The accuracy obtained for single spike inputs with time window size $\Delta = 20$ ms is 96.1% by combining 5 classifiers. For Poisson spike inputs, $f_{high} = 250$ Hz and $f_{low} = 1$ Hz. Each data point is obtained by randomly selecting a subset (N) of the total number of learned classifiers (5). This process is repeated 5 times and the average of the test errors of the combined classifiers is computed.

and (5) limited resources due to the use of a few classifiers with adaptively learned network size.

3.6. Analysis of Dendrite Weights: Feature Maps

We also analyzed the weights of dendrites to understand the features of inputs learned by the nonlinear dendrites. Since our learning rule encourages correlated inputs to be grouped together on the same dendrite, we expected the dendrite weight maps to capture the input correlations. For this purpose, we mapped the $d = 784$-dimensional binary weight vector $w_{\cdot j} = [w_{1j} w_{2j} \ldots w_{dj}]$ for the jth dendrite to a 28×28 matrix consisting of binary elements (since there can be multiple connections from the same afferent on a dendrite, the weight vector can have integer values > 1). For this analysis, 20, 000 patterns were trained using the adaptive scheme-2 by initializing the network with $m = 5$ dendrites for each class.

The weight matrices are visualized as shown in **Figure 8**. The first row depicts the mean weights of all the dendrites corresponding to a digit. It can be seen that the dendrites learn the features of digit images presented to the network. The dendrites belonging to a particular digit class together represent the complete digits. The feature maps look pixelated due to sparse integer weights learned by each dendrite. The remaining rows show the weights of 4 individual dendrites, m_1 to m_4, randomly selected out of the total learned dendrites for each digit. These maps demonstrate that each dendrite learns some features of the input digits. However, some of the dendrites like m_4 of digit "3" and m_3 of digit "7" can learn to represent complete digits, which suggests that some of the dendrites for these classes are redundant and hence, can be removed. A hybrid approach which involves adding new dendrites and pruning the superfluous ones can therefore be used to design the network.

3.7. Boosting the Performance using Optimal Neuron Topology

Next, we determine the theoretically optimal configuration of the network which can be used to boost the performance of our structural learning scheme, as discussed in Section 2.1.3. For this analysis, we used one instance of the final number of dendrites learned for each class using the adaptive scheme-2 (**Figure 6B**). We then computed the theoretical capacity for each neuron of all classes by fixing the total number of synapses learned by the μ^{th} neuron, s^{μ}_{adapt}, and varying the m and k values such that $s^{\mu}_{adapt} = s^{\mu}_{opt} = m \times k$ (step-2 in **Figure 3**). **Figure 9A** shows the theoretical capacity of class "5" neuron plotted as a function of m, where m^5_{adapt} is the number of dendrites learned using the proposed adaptive method while m^5_{opt} is the number of dendrites corresponding to the maximum capacity for a neuron with the same $s^5_{opt} = s^5_{adapt}$ as that of a neuron trained adaptively. The number of dendrites, m^7_{adapt} learned by our method and the corresponding optimal value, m^7_{opt} for class "7" neuron are shown in **Figure 9B**.

We then trained a network with neurons consisting of m^{μ}_{opt} dendrites on 20, 000 MNIST samples using our structural learning rule. **Figure 9C** shows the comparison between the performance of our adaptively learned network (blue) and the corresponding theoretically optimal network (red). These classifiers were combined in ensembles and tested on 10, 000 non-spiking binary inputs and jittered single spike patterns. The results show that the optimal topology can achieve about $0.5 - 0.9\%$ higher accuracy than the adaptively learned configuration. Moreover, the use of optimal m and k values also helps to generate the highest accuracy of our model, 96.6%, obtained on the binary test inputs. Further advantage of using a network with optimal topology can be understood by comparing these results with the accuracies depicted in **Figure 7**. We can see that the ensemble of 3 classifiers with optimal network topology can attain higher accuracy than the ensemble of 5 adaptively learned classifiers. Hence, theoretical capacity predictions of the optimal network configuration can be used to boost the classification performance of our structural learning rule while using same synaptic resources as the adaptive case.

3.8. Classification of Event-based MNIST-DVS Dataset

We also evaluate the performance of our adaptive learning algorithm on the actual event-based MNIST dataset consisting of dynamic vision sensor (DVS) recordings of different handwritten digits (Serrano-Gotarredona and Linares-Barranco, 2014). This dataset was generated by using 10, 000 of 28×28 pixel digit images from the original MNIST dataset which were enlarged to three different scales using smoothing interpolation algorithms. These upscaled digit images were displayed on an LCD monitor with slow motion and a 128×128 pixel AER DVS (Serrano-Gotarredona and Linares-Barranco, 2013) was used to record these moving digits for a total time duration of 2 s. We used 10, 000 recordings of moving digits upscaled to scale-4 for our analysis, which were also used to evaluate the performance of the event-driven categorization system proposed in Zhao et al. (2015). The training was performed on randomly selected 90% of the total 10, 000 recordings while the remaining 10% recordings were used for testing. For training, the event streams for 128×128 DVS recordings were converted to 128×128 pixel images by calculating the total number of events occurring at each pixel location for two different time durations, 100 ms and full length of 2 s. The 128×128 images were then cropped to digit patches by selecting the location of 28×28 squares with maximum events occurring.

The rate-based adaptive scheme was used for training followed by testing on event streams occurring at the 28×28 patch extracted from the original 128×128 DVS recordings. The training procedure was repeated 3 times and the average testing accuracy over 3 trials was determined as shown in **Table 3**, which also compares with the performance of the event driven system in Zhao et al. (2015). The results show that our adaptive dendritic algorithm can achieve comparable testing accuracy (88.1% for 2 s) with the other reported performance on MNIST-DVS dataset, with higher accuracy attained when longer recordings are used. Moreover, our method yields lower training accuracy than Zhao

et al. (2015) while the testing accuracy is similar, suggesting that our method is more robust to overfitting.

3.9. Comparison with Other Spike-based Classification Models

Finally, we compare the performance of our model with other spike classification models consisting of a network of spiking neurons. The spike classifiers considered here use the two main approaches of rate-based and spike-based learning. The models consisting of spiking Restricted Boltzmann Machines (RBM) map the weights learned using offline rate-based scheme onto spiking neural network in O'Connor et al. (2013) and use a spike-based event-driven learning rule based on STDP in Neftci et al. (2014). In Brader et al. (2007), a stochastic spike-driven synaptic plasticity rule was used to train a network of binary synapses and classification accuracy determined by voting over a pool of neurons. **Table 4** compares the training, test sizes, number of neurons or dendrites, synapses used and the accuracy attained by

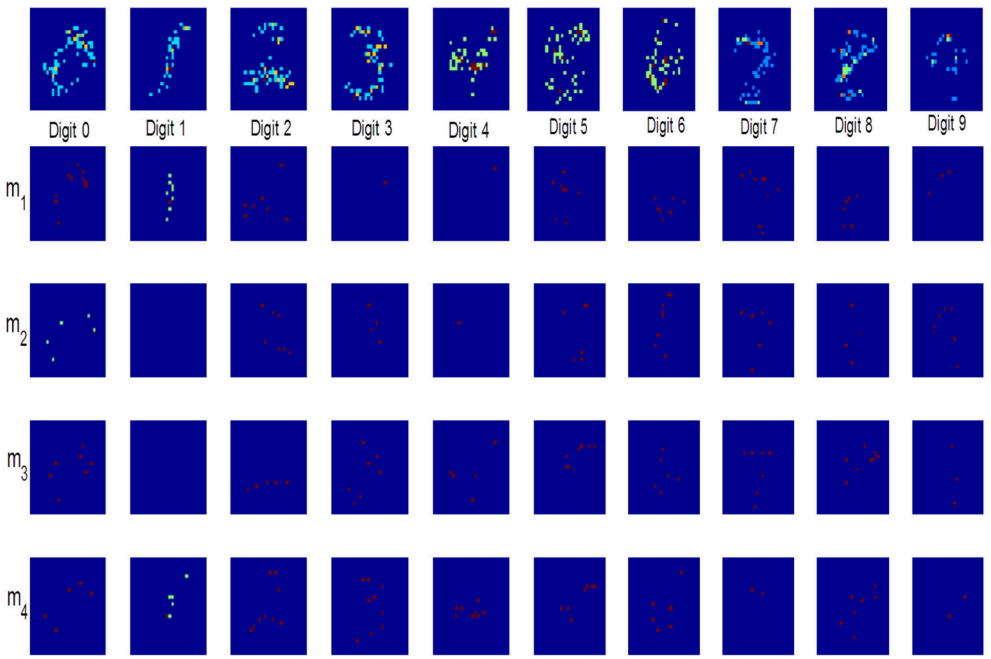


FIGURE 8 | Weight maps of all dendrites of neurons belonging to a particular digit class (top row). Maps in the remaining rows show individual dendrite weights with completely or partially learned features of input digits.

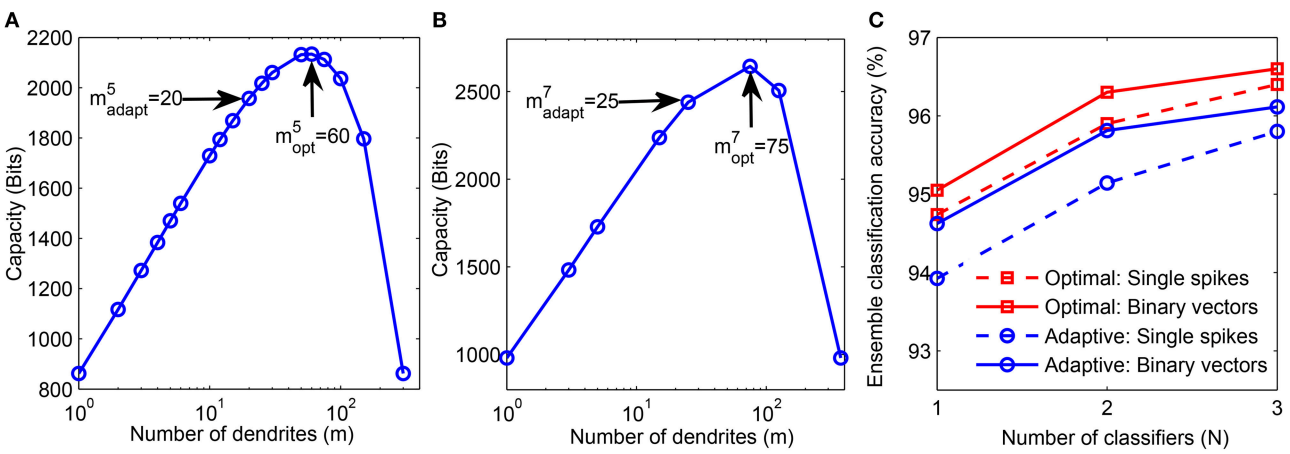


FIGURE 9 | (A) Theoretical capacity of neuron corresponding to digit "5" as a function of number of dendrites (m). The number of dendrites learned adaptively is m^5_{adapt} and the optimal number of dendrites corresponding to maximum capacity is m^5_{opt}. **(B)** Adaptively learned and optimal m values for digit "7." **(C)** Performance comparison of adaptively learned network (blue) with theoretically determined optimal network (red). Training was done on 20, 000 binary digit samples and test performance was measured on 10, 000 binary inputs (solid) and jittered single spike input patterns with $\Delta = 10$ ms (dashed).

TABLE 3 | Performance on MNIST-DVS dataset.

Time length of recording used	Training accuracy (%)	Testing accuracy (%)
100 ms (Zhao et al., 2015)	98.9	76.9
2 s (Zhao et al., 2015)	99.1	88.2
100 ms (this work)	93.7	80.2
2 s (this work)	97.3	88.1

these classification algorithms on MNIST dataset. The number of neurons mentioned is the total number of neurons in the network excluding those in the input layer. For our model, each dendrite can be considered as a processing subunit and is therefore also shown here. For the other networks, all synaptic currents sum up linearly implying the use of only one dendritic branch per neuron. The number of training examples used by Brader et al. (2007) and Neftci et al. (2014) and our method is 20, 000 while the training set in O'Connor et al. (2013) consisted of 120, 000 samples generated by introducing small random translations, rotations and scalings in the original MNIST training examples. The number of test patterns used by all the models is 10, 000.

The results suggest that the NLD model can achieve accuracy comparable with other algorithms by utilizing significantly less, which is about $2-7\%$ of the total number of synapses used by the other methods. Moreover, the use of low resolution integer weights in our work in contrast with the high resolution weights used by spiking RBM models (Neftci et al., 2014; O'Connor et al., 2013) renders our structural learning rule for NLDs more feasible for implementing spike classification in hardware. Further, the ensemble of NLD classifiers yielded 96.1% accuracy by using rate-based learning and testing on Poisson spike inputs. Moreover, the use of optimal network topology, which was determined using the theoretical capacity calculations, enabled us to attain even higher accuracy by reducing the synaptic resources by about 1.7 times. These results are at par with the best performance achieved by Brader et al. (2007). It is important to note that both models in Brader et al. (2007) and our work use binary synapses; however, we obtain similar performance as in their work by training an ensemble of only a few classifiers resulting in still less number of synaptic resources, about 20% of that used in Brader et al. (2007). We also expect that our learning rule will be more amenable for hardware implementation since the dendritic polynomial nonlinearity is much simpler than implementation of a full neuron as in the population of output neurons in Brader et al. (2007). Therefore, the use of much less number of synaptic resources with binary weights and a more simpler learning rule render our NLD model more hardware-friendly.

4. DISCUSSION

Here, we discuss the neurobiological relevance of our work and its potential for future hardware implementation. We also compare our method with other studies based on these neurobiological mechanisms. Finally, we discuss the future directions of our work.

4.1. Role of Dendritic Nonlinearity in Neuronal Processing

Several experimental evidences support the nonlinear processing in dendrites including active backpropagation of axonal spikes into the dendritic tree and dendritic spikes (Hausser et al., 2000; Schiller et al., 2000). However, there are not many evidences regarding the role of these nonlinear mechanisms in synaptic integration in pyramidal neurons. Experimental and compartmental modeling studies of pyramidal neurons have indicated that nearby synaptic inputs on the same dendrite sum sigmoidally while inputs on different dendrites sum up linearly (Poirazi et al., 2003b; Polsky et al., 2004). These findings support the notion of a two-layer model of neurons, thereby having implications for the synaptic plasticity and the computational capacity of cortical tissue. Mel and his group presented several computational studies to elucidate the role of dendrites in neuronal processing. (Mel, 1991; Poirazi et al., 2003b). In more recent studies, an abstract two-layer model using sigmoidal dendritic nonlinearity was shown to predict the firing rate of a detailed compartmental model of a pyramidal neuron (Poirazi et al., 2003a) and much larger storage capacities were computed for dendritic neurons with degree 10 polynomial nonlinearity in Poirazi and Mel (2001). In contrast to these studies, we use a more hardware-friendly quadratic nonlinearity which is easier to implement than a sigmoid or a high order polynomial. We also modify the learning rule to adapt the structure of the dendritic tree of different neurons in the network according to difficulty of the classification task.

4.2. Structural Plasticity as a Learning Mechanism

The phenomenon of structural plasticity involving formation and elimination of synapses thereby leading to alterations to the cortical wiring diagram (Butz et al., 2009; Chklovskii et al., 2004) provides for alternative form of long term information storage in addition to the traditional synaptic weight plasticity. The information storage capacity associated with structural plasticity lies in the ability to change wiring diagram in a sparsely connected network, which provides a large number of functionally distinct circuits available to encode information (Chklovskii et al., 2004) and hence has important implications for the computational properties of the network. The computational modeling study by Poirazi and Mel (2001) demonstrated the use of structural plasticity to modify binary synaptic connections on dendritic branches. Similar to our model, a poorly performing active synapse is eliminated and replaced by the best performing synapse in a set of silent synapses. However, our learning rule is simple and easier to implement in hardware systems as compared with the learning rule used by Poirazi and Mel (2001).

4.3. Binary Synapses: Computational Challenges

There is accumulating experimental evidence that biological synapses exist in only a small number of states which can be restricted to even two states (O'Connor et al., 2005; Petersen et al., 1998). The use of synapses with only one or two bits of long-term

TABLE 4 | Comparison with spike classifiers on MNIST data.

Model	#Train	#Test	#Neurons	#Dendrites	#Synapses	Accuracy %
O'Connor et al., 2013	120, 000	10, 000	1010	1010	647, 000	94.09
Neftci et al., 2014	20, 000	10, 000	540	540	412, 000	91.9
Brader et al., 2007	20, 000	10, 000	150	150	117, 600	96.5
Adaptive NLD	20, 000	10, 000	20	440	6720	94.2
Ensemble NLD	20, 000	10, 000	100	2312	35, 285	96.1
Optimal NLD	20, 000	10, 000	100	3960	20, 163	96.4

information has severe implications for the storage capacity of networks working as classifiers or associative memories with capacity for binary synapses reducing by more than half as compared to the capacity using continuous-valued synapses (Senn and Fusi, 2005). Some studies have presented learning algorithms as biological solutions to deal with the reduced storage capacity of networks with binary synapses. A stochastic spike-driven synaptic plasticity rule was used to train a network of binary synapses, where a pool of output neurons was used to calculate the classification accuracy by a voting mechanism (Brader et al., 2007). This results in a large number of synapses being used. In comparison to this study, our model employs a sparsely connected network of binary synapses which learns by using a correlation-based structural plasticity rule. The use of dendritic nonlinearity yields higher computational power thereby alleviating the problem of reduced capacity of binary weights. Also, the adaptive learning of number of dendrites according to problem complexity reduces the number of synapses compared to a brute force approach. Hence, our model can achieve higher accuracy by utilizing a small number of binary synapses.

4.4. Binary Synapses and Structural Plasticity: Considerations for Hardware Implementation

Over the past decade, several low-power neuromorphic systems have been built to perform classification of spike patterns. A common feature in several of these systems is the usage of binary synapses (Arthur and Boahen, 2007; Indiveri et al., 2006; Mitra et al., 2009). One reason for this is the ease with which two states can be stored in current CMOS technology using a latch. This also makes the system more robust to parametric variations due to mismatch in device - it is unlikely that high resolution weights can be obtained from a massive array of analog synapses due to a combination of systematic and random mismatch (Linares-Barranco et al., 2003). Even a recently introduced multi-core asynchronous digital chip (Merolla et al., 2014) uses a limited number of weight values per axon per core. Our algorithm is consistent with this philosophy of low-resolution weights since we limit the number of synaptic connections per dendrite and each connection is a binary value. Effectively each input afferent (or axon) connects with a small integer weight to a dendrite.

Another advantage of our architecture is that the learning happens by modifying connectivity patterns of the network. In most current event-based neuromorphic systems, this connection matrix is stored in a separate memory (Liu, 2014) either on or off chip. This implies that since our hardware architecture enforces sparsity, we require less memory and memory reads to store and access connection information respectively. Before expanding on this point, it is important to note that we are not considering advantages of hardware implementations of on-chip learning to find optimal connections (though we have presented some initial results on the same in Roy et al., 2014b). We are only comparing the advantages of using our proposed architecture to implement the final network and using structural plasticity to reduce the memory requirement of this implementation. In this context, it should be noted that normal weight learning methods do not necessarily produce sparse weights and simple quantization of small weights to zero values increase errors. This was shown to be true for an ensemble of perceptrons trained by the p-delta algorithm in Roy et al. (2014a). More recently, there have been efforts to improve rounding algorithms to reduce weight resolution for efficient implementation of deep networks (Muller and Indiveri, 2015). Even with these methods, a two layer fully connected neural network with 500 hidden nodes needs at least 4 bits per synaptic weight to achieve comparable performance ($\sim$ 96%) as our network on the MNIST dataset. This results in approximately 397, 000 4-bit weights as opposed to $\sim$ 21, 000 1-bit weights in our case.

To generalize this result, let us consider a two layer network for the conventional case with d inputs, H hidden layer neurons and C output neurons for "C" classes. The comparable network in our proposed case has H dendrites and C output neurons. "2m" out of the H dendrites connect to each of the C output neurons ($H = 2m \times C$) using unit weights and hence can be implicitly implemented by accumulators. Considering each weight of the conventional network having resolution of "b" bits, the total number of bits required by the conventional network (NOB_{conv}) is given by:

$$NOB_{conv} = b \times H \times d + b \times H \times C \approx b \times H \times d \text{ for } C << d \quad (22)$$

For the proposed case, the connection matrix is of size $d \times H$ though only $k \times H$ entries are non-zero where $k << H$. To implement this sparse connectivity efficiently in an address event framework, we propose to use a two tier addressing scheme as shown in **Figure 10**. Here, the incoming address will be used to index into a pointer array of "d" entries with $\lceil log_2(H \times k) \rceil$ bits per

entry. An incoming spike address, say "i," is used to index into this array and read the two consecutive values a_i and a_{i+1}. As shown in the figure, suppose $a_i = p$ and $a_{i+1} = q$. $n_i = a_{i+1} - a_i$ is the number of synapses connected to this input. If $n_i > 0$, then $a_i = p$ is used as a pointer to the pth location in a dendrite address array. This second array has $H \times k$ entries with $\lceil log_2(H) \rceil$ bits per entry that hold the address of the dendritic branch where the synapse is located. n_i consecutive values (d_p to d_{q-1}) are read as destination addresses to route the spikes. Now, the total memory required by the look up table in the proposed method (NOB_{prop}) can be estimated as:

$$NOB_{prop} = d \times \lceil log_2(H \times k) \rceil + H \times k \times \lceil log_2(H) \rceil \quad (23)$$

The memory requirements for proposed and conventional methods are compared in **Figure 11** by setting $b = 4$, $H = 10^4$ and varying d over a wide range for $k = 16$, 32, and 64. It can be seen that the proposed method requires much less memory than the conventional case for large values of d when the sparsity is higher while the overhead of having a pointer array is more for small values of d. The crossover typically happens for $d < 200$ for values of k as large as 64. Since for most practical cases d is much larger, we expect our method to be widely applicable.

To underline the importance of this memory reduction, we consider a digital implementation of this system following principles similar to the ones in Merolla et al. (2014) and Seo et al. (2011). In particular, we assume that the dendritic nonlinearity/hidden neuron can be a shared physical circuit that can be time multiplexed across all required instances and we assume synaptic weight resolution is 4 bits following Seo et al. (2011). Here, for simplicity we ignore the overhead needed if the network is spread across multiple cores. Using the numbers quoted in Merolla et al. (2014), the area requirement of a neuron circuit is 2900 μm^2. Compared to that, the area required to implement a conventional crossbar of 4 bit weights for the MNIST case of $d = 784$ and say $H = 1000$ is 470, 400 μm^2 where we estimate 0.15 μm^2 area per bit from Merolla et al. (2014). This is clearly the dominant factor in chip area. Compared to this, our scheme with even $k = 64$, $H = 1000$ and $d = 784$ requires approximately 98, 000 μm^2 area, a reduction by $> 4X$.

4.5. Future Work

The classification performance attained by our model on the benchmark MNIST data is not state-of-the-art. The best MNIST classification result achieved so far is 99.06% accuracy using maxout networks (Goodfellow et al., 2013). Hence, we need to bridge this gap by enhancing our model. Our present model consists of lumped dendritic nonlinearity such that each dendrite is considered to be a single compartment where all the synaptic inputs are lumped together. The storage capacity of this network can be increased by introducing multiple compartments on each dendrite. The dendritic compartments represent time delays in signal propagation along a dendrite and therefore, the information about the location of synaptic inputs on a dendrite is important. We will utilize this additional source of spatial information to enhance our dendritic structural learning rule which will involve finding the optimal location on the optimal dendritic branch for a synaptic connection.

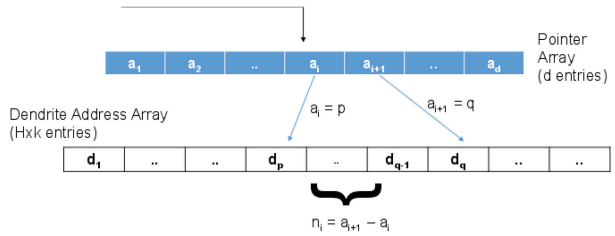


FIGURE 10 | To implement the sparse connectivity matrix, a two level addressing scheme is proposed where the first memory (indexed by the incoming address of the event) holds pointers to valid connection addresses stored in the second memory.

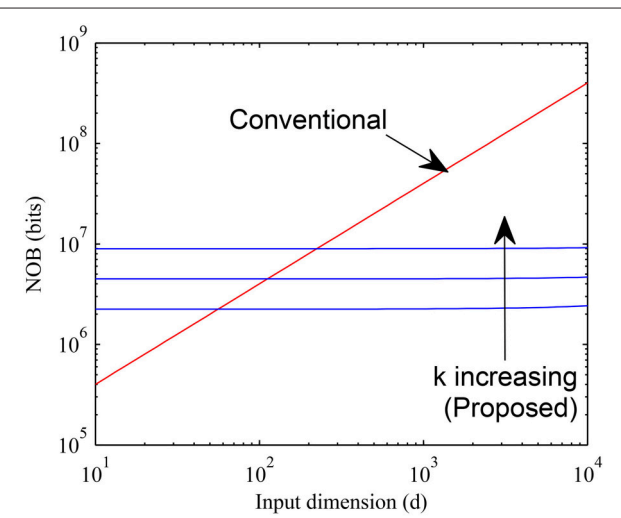


FIGURE 11 | Memory requirement of proposed sparse connectivity scheme is compared with that of a normal fully connected weight memory with 4 bits per weight. The proposed scheme is far more memory efficient for high dimensional inputs ($d > 200$).

The storage capacity of the network can further be increased by including distributed nonlinearity along a dendrite such that the nonlinear output of each compartment serves as input to the next compartment on the dendrite. This scheme is also more bio-realistic from the perspective of real neurons consisting of extended dendritic trees with complex branching patterns. We will also enhance our adaptive learning rule to prune the redundant or least "salient" synapses. This pruning method combined with the progressive addition of dendrites will yield an optimally sized network that will fit the data. The network will learn both the number of required dendrites as well as the number of synapses on each dendrite. This approach to obtain the smallest network can also improve the generalization performance.

In this work, we have not used temporal information to classify input patterns and have focussed on rate and place

encoding of the binary images from the MNIST dataset. In a recent work (Roy et al., 2015), we have used structural plasticity to learn binary classification spatiotemporal patterns as used in Gutig and Sompolinsky (2006). Hence, a natural extension of our present work is to combine the use of spike timing information with structural learning to enable classification of multiclass temporal codes.

AUTHOR CONTRIBUTIONS

SH and AB developed the idea for the paper. SH did all the modeling simulations. SH analyzed the results and both authors discussed the results and contributed to the manuscript preparation. AB supervised this work. This work was supported by MOE, Singapore through grant ARC 8/13.

REFERENCES

Albers, C., Westkott, M., and Pawelzik, K. (2013). "Perfect associative learning with spike-timing-dependent plasticity," in *Advances in Neural Information Processing Systems 26*, eds C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (South Lake Tahoe, CA: Curran Associates, Inc.), 1709–1717.

Arthur, J., and Boahen, K. (2007). Synchrony in silicon: the gamma rhythm. *IEEE Trans. Neural Netw.* 18, 1815–1825. doi: 10.1109/TNN.2007.900238

Boahen, K. (2000). Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans. Circ. Syst. II* 47, 416–434. doi: 10.1109/82.842110

Brader, J., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Butz, M., Worgotter, F., and Ooyen, A. V. (2009). Activity-dependent structural plasticity. *Brain Res. Rev.* 60, 287–305. doi: 10.1016/j.brainresrev.2008.12.023

Chan, V., Liu, S.-C., and van Schaik, A. (2007). AER EAR: a matched silicon cochlea pair with address event representation interface. *IEEE Trans. Circ. Syst. I* 54, 48–59. doi: 10.1109/TCSI.2006.887979

Chen, S., and Bermak, A. (2007). Arbitrated time-to-first spike CMOS image sensor array with on-chip histogram equalization. *IEEE Tran. VLSI* 15, 346–357. doi: 10.1109/TVLSI.2007.893624

Chklovskii, D. B., Mel, B. W., and Svoboda, K. (2004). Cortical rewiring and information storage. *Nature* 431, 782–788. doi: 10.1038/nature03012

Choi, T., Merolla, P., Arthur, J., Boahen, K., and Shi, B. (2005). Neuromorphic implementation of orientation hypercolumns. *IEEE Trans. Circ. Syst. I* 52, 1049–1060. doi: 10.1109/TCSI.2005.849136

Fiesler, E. (1994). "Comparative bibliography of ontogenic neural networks," in *Proceedings of the International Conference on Artificial Neural Networks ICANN 1994*, Vol. 1, (Sorrento), 793–796. doi: 10.1007/978-1-4471-2097-1_188

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models Single Neurons, Populations, Plasticity*. Cambridge, UK: Cambridge University Press. doi: 10.1017/CBO9780511815706

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning* (Atlanta), 1319–1327.

Gutig, R., and Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nat. Neurosci.* 9, 420–428. doi: 10.1038/nn1643

Hausser, M., Spruston, N., and Stuart, G. J. (2000). Diversity and dynamics of dendritic signaling. *Science* 290, 739–744. doi: 10.1126/science.290.5492.739

Helias, M., Rotter, S., Gewaltig, M., and Diesmann, M. (2008). Structural plasticity controlled by calcium based correlation detection. *Front. Comput. Neurosci.* 2:7. doi: 10.3389/neuro.10.007.2008

Hussain, S., Gopalakrishnan, R., Basu, A., and Liu, S.-C. (2013). "Morphological learning: increased memory capacity of neuromorphic systems with binary synapses exploiting AER based reconfiguration," in *Proceedings of the International Joint Conference on Neural Networks* (Dallas, TX). doi: 10.1109/ijcnn.2013.6706928

Hussain, S., Liu, S.-C., and Basu, A. (2014). "Improved margin multi-class classification using dendritic neurons with morphological learning," in *Proceedings of the International Symposium on Circuits and Systems* (Melbourne). doi: 10.1109/iscas.2014.6865715

Hussain, S., Liu, S.-C., and Basu, A. (2015). Hardware-amenable structural learning for spike-based pattern classification using a simple model of active dendrites. *Neural Comput.* 27, 845–897. doi: 10.1162/NECO_a_00713

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Islam, M., Sattar, A., Amin, F., Yao, F. X., and Murase., K. (2009). A new adaptive merging and growing algorithm for designing artificial neural networks. *IEEE Syst. Man Cybern. Soc.* 39, 705–722. doi: 10.1109/TSMCB.2008.2008724

Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE Trans. Neural Netw.* 1, 239–242. doi: 10.1109/72.80236

Kwok, T.-Y., and Yeung, D.-Y. (1996). "Bayesian regularization in constructive neural networks," in *Proceedings of the 1996 International Conference on Artificial Neural Networks ICANN 1996* (Bochum), 557–562. doi: 10.1007/3-540-61510-5_95

Kwok, T.-Y., and Yeung, D.-Y. (1997). Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Trans. Neural Netw.* 8, 630–645. doi: 10.1109/72.572102

Linares-Barranco, B., Serrano-Gotarredona, T., and Serrano-Gotarredona, R. (2003). Compact low-power calibration mini-dacs for neural arrays with programmable weights. *IEEE Trans. Neural Netw.* 14, 1207–1216. doi: 10.1109/TNN.2003.816370

Lahnajarvi, J. J., Lehtokangas, M., and Saarinen, J. (2002). Evaluation of constructive neural networks with cascaded architectures. *Neurocomputing* 48, 573–607. doi: 10.1016/S0925-2312(01)00630-0

LeCun, Y. L., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128x128 120 dB 15us latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circ.* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Liu, S.-C. (2014). *Event-Based Neuromorphic Systems*. Hoboken, NJ: John Wiley & Sons.

Liu, S.-C., Schaik, A. V., Minch, B. A., and Delbruck, T. (2013). Asynchronous binaural spatial audition sensor with 2 x 64 x 4 channel output. *IEEE Trans. Biomed. Circ. Syst.* 8, 453–464. doi: 10.1109/TBCAS.2013.2281834

London, M., and Hausser, M. (2005). Dendritic computation. *Ann. Rev. Neurosci.* 28, 503–532. doi: 10.1146/annurev.neuro.28.061604.135703

Maass, W., and Schmitt, M. (1999). On the complexity of learning for spiking neurons with temporal coding. *Inf. Comput.* 153, 26–46. doi: 10.1006/inco.1999.2806

Magee, J. C. (2000). Dendritic integration of excitatory synaptic input. *Nat. Rev. Neurosci.* 1, 181–190. doi: 10.1038/35044552

Marti, D., Rigotti, M., Seok, M., and Fusi, S. (2015). Energy-efficient neuromorphic classifiers. *arXiv5:1507.0023*

Mel, B. W. (1991). "The clusteron: toward a simple abstraction for a complex neuron," in *Proceedings of the Neural Information Processing Systems* (San Mateo, CA: Morgan Kaufman), 35–42.

Merolla, P. A., Arthur, J. V., Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Eng.* 3, 32–42. doi: 10.1109/tbcas.2008.2005781

Muller, L., and Indiveri, G. (2015). Rounding methods for neural networks with low resolution synaptic weights. *arXiv:1504.05767*

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Front. Neurosci.* 7:272. doi: 10.3389/fnins.2013.00272

O'Connor, D. H., Wittenberg, G. M., and Wang, S. S.-H. (2005). Graded bidirectional synaptic plasticity is composed of switch-like unitary events. *Proc. Natl. Acad. Sci. U.S.A.* 102, 9679–9684. doi: 10.1073/pnas.0502332102

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178

Opitz, D., and Maclin, R. (1999). Popular ensemble methods: an empirical study. *J. Artif. Intell. Res.* 11, 169–198.

Painkras, E., Plana, L., Garside, J., Temple, S., Galluppi, F., Patterson, C., et al. (2013). Spinnaker: a 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid-State Circ.* 48, 1943–1953. doi: 10.1109/JSSC.2013.2259038

Petersen, C. C., Malenka, R. C., Nicoll, R. A., and Hopfield, J. J. (1998). All-or-none potentiation at CA3-CA1 synapses. *Proc. Natl. Acad. Sci. U.S.A.* 95, 4732–4737. doi: 10.1073/pnas.95.8.4732

Poirazi, P., Brannon, T., and Mel, B. (2003a). Pyramidal neuron as two-layer neural network. *Neuron* 37, 989–999. doi: 10.1016/S0896-6273(03)00149-1

Poirazi, P., Brannon, T., and Mel, B. W. (2003b). Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell. *Neuron* 37, 977–987. doi: 10.1016/S0896-6273(03)00148-X

Poirazi, P., and Mel, B. (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron* 29, 779–796. doi: 10.1016/S0896-6273(01)00252-5

Polsky, A., Mel, B., and Schiller, J. (2004). Computational subunits in thin dendrites of pyramidal cells. *Nat. Neurosci.* 7, 621–627. doi: 10.1038/nn1253

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid-State Circ.* 46, 259–275. doi: 10.1109/JSSC.2010.2085952

Qi, X., Guo, X., and Harris, J. (2004). "A time-to-first spike CMOS imager," in *Proceedings of the International Symposium on Circuits and Systems* (Vancouver, BC), 824–827.

Reed, R. (1993). Pruning aalgorithm-a survey. *IEEE Trans. Neural Netw.* 4, 740–747. doi: 10.1109/72.248452

Roy, S., Banerjee, A., and Basu, A. (2014a). Liquid state machine with dendritically enhanced readout for low-power, neuromorphic vlsi implementations. *IEEE Trans. Biomed. Circ. Syst.* 8, 681–695. doi: 10.1109/TBCAS.2014.2362969

Roy, S., Kar, S. K., and Basu, A. (2014b). "Architectural exploration for on-chip, online learning in spiking neural networks," in *IEEE International Symposium on Integrated Circuits* (Singapore), 128–131. doi: 10.1109/isicir.2014.7029541

Roy, S., San, P. P., Hussain, S., Wei, L. W., and Basu, A. (2015). Learning spike time codes through morphological learning with binary synapses. *IEEE Trans. Neural Netw. Learn. Syst.* doi: 10.1109/tnnls.2015.2447011. [Epub ahead of print].

Schiller, J., Major, G., Koester, H. J., and Schiller, Y. (2000). NMDA spikes in basal dendrites of cortical pyramidal neurons. *Nature* 404, 285–289. doi: 10.1038/35005094

Schmuker, M., Pfeil, T., and Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proc. Natl. Acad. Sci. U.S.A.* 111, 2081–2086. doi: 10.1073/pnas.1303053111

Senn, W., and Fusi, S. (2005). Convergence of stochastic learning in pperceptron with binary synapses. *Phys. Rev. E* 71(6 Pt 1):061907. doi: 10.1103/PhysRevE.71.061907

Seo, J., Brezzo, B., Liu, Y., Parker, B. D., Esser, S. K., Montoye, R. K., et al. (2011). "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proceedings of IEEE, Custom Integrated Circuits Conference (CICC)* (San Jose, CA), 1–4. doi: 10.1109/cicc.2011.6055293

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linarez-Barranco, A., Paz-Vicente, R., Goméz-Rodríguez, F., et al. (2009). CAVIAR: a 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking. *IEEE Trans. Neural Netw.* 20, 1417–1438. doi: 10.1109/TNN.2009.2023653

Serrano-Gotarredona, T., and Linares-Barranco, B. (2013). A 128x128 1.5 % contrast sensitivity 0.9 % FPN 3ms latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE J. Solid-State Circ.* 48, 827–838. doi: 10.1109/JSSC.2012.2230553

Serrano-Gotarredona, T., and Linares-Barranco, B. (2014). The mnist-dvs database. [online].

Vogelstein, R., Mallik, U., Culurciello, E., Cauwenberghs, G., and Etienne-Cummings, R. (2007). A multichip neuromorphic system for spike-based visual information processing. *Neural Comput.* 19, 1–20. doi: 10.1162/neco.2007.19.9.2281

Zhao, B., Ding, R., Chen, S., Linares-Barranco, B., and Tang, H. (2015). Feedforward categorization on aer motion events using cortex-like features in a spiking neural network. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1963–1978. doi: 10.1109/tnnls.2014.2362542

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2016 Hussain and Basu. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

A. APPENDIX

We have proposed a biologically realistic branch-specific spike-time dependent structural plasticity (BSTDSP) rule for binary classification in Hussain et al. (2015), which was inspired from a recent study in which reverse spike-timing dependent plasticity (RSTDP) in concert with hyperpolarization of the postsynaptic neuron was used to modify the synaptic weights (Albers et al., 2013). Our learning rule involving modification of connections instead of synaptic weights, is in contrast with Albers et al. (2013) and is used to find the bio-realistic correlation values c_{ij}. However, the bio-realistic basis of adapting the number of dendrites is not clear.

The correlation values needed for structural plasticity rule were obtained from an online spike-based learning rule. The biological relevance of these correlations can be found in calcium concentration in spines which is a correlation sensitive, spike-time dependent signal and has been implicated as a guide for structural plasticity (Helias et al., 2008). Here, we present an extension of BSTDSP to multiclass classification and also demonstrate the correspondence between rate-based and spike-based learning approaches.

A. Spike-based Structural Learning for Multiclass Classification

The multiclass BSTDSP is derived using the example of synchronous single-spike inputs such that each afferent either fails to fire or fires a spike at a fixed time T_{syn} and these spike inputs are assigned to N_C classes. The single-spike input is denoted by the presynaptic spike train $s_{ij}(t)$ at the ith synapse of the jth dendrite, and the postsynaptic spike train is denoted by $r(t)$ given by:

$$r(t) = \sum_{t_{post}} \delta(t - t_{post}) \tag{A1}$$

where t_{post} is the postsynaptic spike time. These pre- and postsynaptic spikes also drive exponentially decaying memory traces–the presynaptic trace $\bar{s}$ and the postsynaptic trace $\bar{r}$ given by:

$$\bar{s} = exp(-t/\tau_{pre}) \tag{A2}$$

$$\bar{r} = exp(-t/\tau_{post}) \tag{A3}$$

If a pattern belonging to class μ is presented, a teacher signal forcing a postsynaptic spike (at time $t = 1$ ms) is present for the $(+)$ neuron and absent for $(-)$ neuron of class μ. For the remaining $(N_C - 1)$ classes, opposite conditions for the teacher signal exist, i.e., teacher signal is absent for $(+)$ neuron and present for $(-)$ neuron.

The spike-based structural plasticity learning rule is similar to the rate-based learning method and also involves computing correlation values c_{ij} for each synapse belonging to the PDT and NDT corresponding to class μ. This learning scheme is analogous to the RSTDP rule such that if a postsynaptic spike arrives before a presynaptic spike, then the correlation value for that synapse is potentiated. The condition for depression does not use postsynaptic spike–instead, the relevant event is when the membrane voltage $V_m(t)$ crosses a subthreshold voltage V_{st} from below, where $0 < V_{st} < V_{thr}$. These subthreshold crossing events occurring at times t_{st} are denoted by $r_{st}(t) = \sum_{t_{st}} \delta(t - t_{st})$. The correlation value is decreased when presynaptic spike time t_{ij}^s occurs before t_{st}. The reason for using $r_{st}(t)$ instead of $r(t)$ is to enforce a margin as will be explained later in this section. Finally, the change in correlation value $\Delta c_{ij}(t)$ for every pattern presentation is computed using these plasticity conditions in a branch-specific manner. Hence, the learning rule for the PDT and NDT belonging to the μth class can be written as:

For PDT of class μ:

$$\Delta c_{ij}^{PDT}(t) = I_{D,j}^{PDT}(t)\bar{r}^{PDT}(t)s_{ij}^{PDT}(t) - \gamma I_{D,j}^{PDT}(t)\bar{s}_{ij}^{PDT}(t)r_{st}^{PDT}(t) \tag{A4}$$

For NDT of class μ:

$$\Delta c_{ij}^{NDT}(t) = I_{D,j}^{NDT}(t)\bar{r}^{NDT}(t)s_{ij}^{NDT}(t) - \gamma I_{D,j}^{NDT}(t)\bar{s}_{ij}^{NDT}(t)r_{st}^{NDT}(t) \tag{A5}$$

where γ is a constant used to balance the potentiation and depression and ensure that $c_{ij} = 0$ when a pattern is learned. The correlation values averaged over all the patterns can be written as: $c_{ij}^{PDT} =< \Delta c_{ij}^{PDT}(t) >$ and $c_{ij}^{NDT} =< \Delta c_{ij}^{NDT}(t) >$, where $< . >$ indicates average calculated over an epoch. It was shown in Hussain et al. (2015) that by assuming the presynaptic time constant τ_{pre} to be much greater than the membrane time constant τ_V, the value of γ can be computed as:

$$\gamma = \frac{exp(-T_{syn}/\tau_{post})}{exp(-T_{rise}/\tau_{pre})} \tag{A6}$$

where the presynaptic spike arrives at time T_{syn} and $V_m(t)$ crosses V_{st} at time T_{rise}. The connection changes are done on the basis of c_{ij} values using the same process as discussed for rate-based learning. The use of a subthreshold voltage V_{st} leads to a margin δ_{spike} for classifying spike patterns. A desired margin δ_{spike} for a particular class μ can also be preset, which can then be used to calculate the class specific values of V_{st} and V_{reset} as explained in Hussain et al. (2015). The class margins suited for a particular problem can be assigned by using margins set for rate-based learning. The value of the desired margin, δ_{spike} is determined by using the spike equivalent value of δ. This is done by computing the difference in the membrane voltage of the neuron (analogous to $a_{PDT} - a_{NDT}$) when a single synapse is activated. This value ΔV_m multiplied by δ gives δ_{spike}, since in the binary input case the synaptic strength is normalized to 1.

The spike-based and the rate-based structural learning rules were compared in Hussain et al. (2015) and a relationship between the correlation terms $[\Delta c_{ij}]_{spike}$ and $[\Delta c_{ij}]_{rate}$ for the two forms of learning was derived as given below.

$$[\Delta c_{ij}]_{spike} = exp(-T_{syn}/\tau_{post})[\Delta c_{ij}]_{rate} \tag{A7}$$

Therefore, any differences between the performance of the two learning methods can be understood on the basis of this

relationship. This suggests that as the value of τ_{post} is increased, the correlation values for the rate-based learning, $[\Delta c_{ij}]_{rate}$ well approximate the correlation values for the spike-based learning, $[\Delta c_{ij}]_{spike}$. Since the two forms of learning tend to coincide with increasing τ_{post}, therefore, we have used the faster rate-based approach instead of the computationally prohibitive spike-based learning scheme for the enhancements proposed in the model.

B. Comparison of Spike-based Learning vs Rate-based Learning Schemes

We have compared the performance of the spike-based and rate-based learning schemes. Since spike-time-based learning takes very long and the training time increases with the size of training set, therefore, we have trained the spike model on smaller number of training patterns, $200 - 1000$. For both forms of learning schemes, the networks consisted of $m = 10$ dendrites in the PDT and NDT of all $N_C = 10$ classes. Hence, the total number of dendrites used by both models is $M = 200$. **Figure A1A** shows the comparison between the performance of spike-based and rate-based learning schemes. Both forms of learning exhibit the trend of decreasing test error with the training set size. However, the spike-based learning method results in about 1.5 times higher errors than that for rate-based learning. This difference in performance of the two methods can be attributed to three reasons: (1) Firstly, the relationship between these two methods (Equation A7) suggests that as the value of τ_{post} is increased, the correlation values for the two learning methods become more similar. Therefore, we need to repeat our spike learning simulations with higher values of τ_{post} to achieve greater agreement between the two forms of learning; (2) Secondly, our estimation of γ in Equation (A6) is based on the assumption that $\tau_{pre} \gg \tau_V$. Hence, smaller τ_{pre} will result in a non-ideal value of γ which will give different results for the two learning schemes; and (3) Finally, the discrepancy in results can arise if the margin setting is not exact. As discussed above, the margin for spike-based learning was set using $\delta_{spike} = \Delta V_m \times \delta$, where a wrong estimate of the constant ΔV_m will mean that δ_{spike} and δ are not truly analogous to each other, leading to different results.

Based on this discussion, we attempted to reduce the difference between the performance of rate-based and spike-time-based learning approaches. Therefore, we increased the value of τ_{post} from 200 to 500, 1000 and 2000 ms and τ_{pre} from 10 to 50 ms. As shown in **Figure A1B**, the test error for $\tau_{post} = 200$ ms corresponds to our prior parameter settings for $P = 200$ patterns (**Figure A1A**, blue curve). As τ_{post} is increased, the testing performance of spike-based learning rule improves and gradually becomes closer to the testing performance of rate-based method, shown with blue line. We expect that we can achieve further agreement between the two forms of learning by getting the margin values δ_{spike} and δ to be analogous to each other or by determining the desired δ_{spike} directly in the spike domain. Hence, we have shown that the error rate for spike-based learning becomes similar to that of rate-based learning if τ_{post} is long enough.

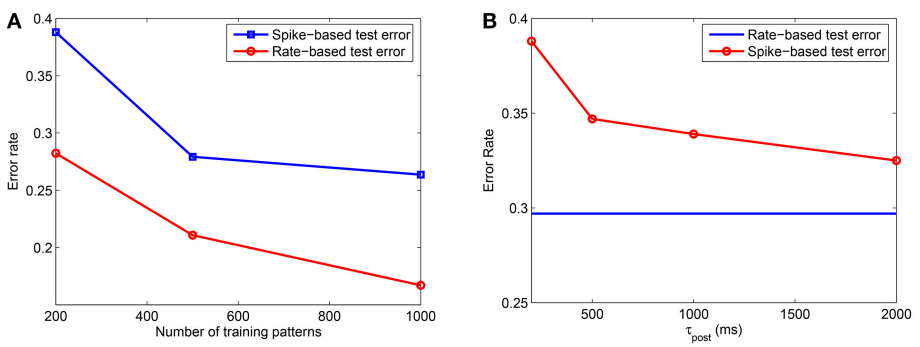


FIGURE A1 | (A) Comparison of performance of rate-based and spike-time-based learning schemes. The test error rates for the spike-based learning rule are about 1.5 times higher than the rate-based test errors; **(B)** Comparison of test error rate for $P = 200$ patterns learned with rate-based (blue line) and spike-based (red plot) methods as a function of τ_{post}. Each data point is obtained by averaging over 3 training simulation trials and testing on 10,000 single spike inputs with jitter $\Delta = 10$ ms; $m = 10$ and $k = 10$.