Improved Margin Multi-Class Classification using Dendritic Neurons with Morphological Learning

Shaista Hussain*, Shih-Chii Liu† and Arindam Basu*

*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
†Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland.
Email: arindam.basu@ntu.edu.sg

***Abstract*— We present an architecture of a spike based multi-class classifier using neurons with non-linear dendrites and sparse synaptic connectivity where each synapse takes a binary value. The learning in this model happens not through weight updates but through structural changes, i.e. a change of connectivity between inputs and dendrites. Hence, it is well suited for implementation in neuromorphic systems using address event representation (AER). We present a new learning rule that allows better generalization of the system to noisy testing data making it feasible to transfer learnt weights in software to a hardware device interfacing with noisy spiking sensors. The new rule improves testing accuracy by $7-10\%$ compared to earlier versions. We also present preliminary results for multi-class classification on handwritten digits from the MNIST database and show that our system can attain comparable performance ($\approx 3\%$ more error) with other reported spike based classifiers while using at least 50% less synaptic resources.**

I. Introduction and Motivation

A lot of effort in the last decade has been directed to understanding the principles of operation in the brain and applying those principles to create smart, energy-efficient artificial systems. In particular, the spike based operation of the human brain can help in creating systems with large computational power as well as low-power operation since the system is event driven and does not consume any power during long inactive periods [1]. Hence, spike based retina [2] and cochlea [3] chips have been developed for processing visual and auditory information respectively. Now, there is a need to develop low-power event based systems that can process the spike stream emanating from these sensors. This paper presents one such supervised multi-class classifier using dendritic neurons that can operate on high dimensional binary patterns of rate encoded spike trains or synchronous single spikes.

We have earlier presented a binary classifier employing such neurons with dendrites where the learning occurs through an AER friendly connection modification rule [4]. In this paper, we present a learning rule that improves the generalization performance on noisy spike train inputs by using the concept of margin maximization similar to support vector machine (SVM). We also extend the earlier setup to multi-class classification and demonstrate the performance on recognizing handwritten digits from the MNIST data set.

Financial support from MOE through grant ARC 8/13 is acknowledged.

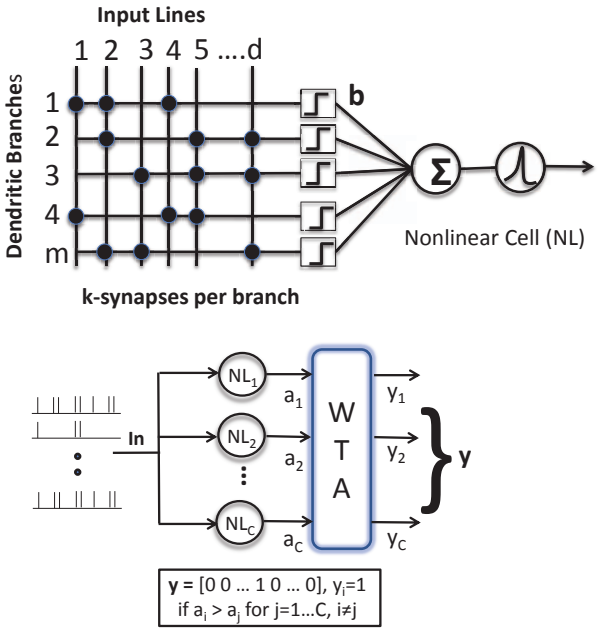


Fig. 1: (a) A nonlinear cell consisting of nonlinear dendrites characterized by the lumped nonlinear function $b()$ for every dendritic branch. (b) Architecture of a multiclass pattern classifier consisting of NL-cells and a WTA to compare the outputs of the C cells trained to respond to patterns belonging to C classes. The output of the classifier is equal to 1 for the class corresponding to the cell having highest output $a(\boldsymbol{x})$.

II. Model of Nonlinear Dendrites for Classification

A. Conversion of Spiking inputs to 'Static' inputs

Spike train inputs are functions of time; but for the special case of binary input patterns considered here, we can convert both the rate encoded spike trains or synchronous single spikes to an equivalent 'static' vector $\boldsymbol{x} \in R^d$ where each component $x_i \in 0, 1$. For each of the d afferents in rate encoded case, if the mean firing rate of the spike train is higher than a threshold, the corresponding x_i is set to 1; else it is set to 0. A similar technique can be applied for the case of binary patterns of synchrony. The results we present in this paper are for rate encoded Poisson spike trains only–however, we have tested our method for binary patterns of synchronous spikes as well and achieved similar results. We shall use $\boldsymbol{x}$ for training the

network while the original binary spike train vector is used for testing. This allows us to save a lot of time in training; however, care must be taken to ensure proper generalization to the spike based testing case.

B. Architecture

We have earlier described a model of nonlinear dendrites in [4] where it was used to perform supervised classification of binary patterns of rate encoded spike train inputs. The model comprises neuronal cells having lumped dendritic nonlinearity which can exhibit much larger computational capacity than their counterparts with linear subunits as shown in [4], [5]. The nonlinear cell (NL-cell) consists of a simplified neuron with m dendritic branches and k excitatory synapses on each branch (Fig. 1(a)) with each branch governed by its nonlinearity $b()$. Each synapse is driven by one of the d components of the input vector $\boldsymbol{x}$. The response of the j^{th} branch, r_j is the nonlinear integration of the activations of k synapses on that branch,

$$r_j(\boldsymbol{x}) = b(z_j) = b(\sum_{i=1}^{k} w_{ij}x_{ij}) \tag{1}$$

where w_{ij} is the weight of the i^{th} synapse formed on the j^{th} branch and x_{ij} is the input arriving at that synapse and z_j denotes the total synaptic activation on j-th branch. $b()$ denotes the nonlinear activation function of the dendritic branch. We have used square dendritic nonlinearity in our simulations given by $b(z) = z^2$ since it can be easily implemented in hardware. The activation level of the NL-cell $a(\boldsymbol{x})$ to input pattern $\boldsymbol{x}$ is given by the sum of response of all the branches as shown in the equation below:

$$a(\boldsymbol{x}) = \sum_{j=1}^{m} b(\sum_{i=1}^{k} w_{ij}x_{ij}) \tag{2}$$

In this work, we have extended the application of nonlinear dendrite model to multi-class classification problems. As shown in Fig. 1(b), the classifier consists of C cells, where C is the number of classes to which input patterns belong. Each cell is trained to preferentially respond to patterns belonging to that class. A winner take all (WTA) [6] circuit is used to select the winning cell and produce the output of the classifier. The output of the model $\boldsymbol{y}(\boldsymbol{x})$ is a C-dimensional binary vector in which the *i-th* component corresponding to the class i with the largest activation level is 1 while rest of the elements are 0. Therefore, the overall response of the network is given as:

$$y_i(\boldsymbol{x}) = g(a_i(\boldsymbol{x}) - a_j(\boldsymbol{x})),\ a_j(\boldsymbol{x}) \geq a_k(\boldsymbol{x}),\ \forall j,k \neq i \tag{3}$$

where g is a Heaviside step, i, j, k are integers in the range $[1, C]$ and y_i is the *i-th* component of the output $\boldsymbol{y}$ respectively.

C. Learning Algorithm

For training the network, we assume a teacher signal, $\boldsymbol{t}$ is available to indicate the desired output during the training phase, where $\boldsymbol{t}$ is a C-dimensional binary vector consisting of $(C-1)$ zeros and a 1 corresponding to the class to which the input pattern belongs. The learning process can be regarded as

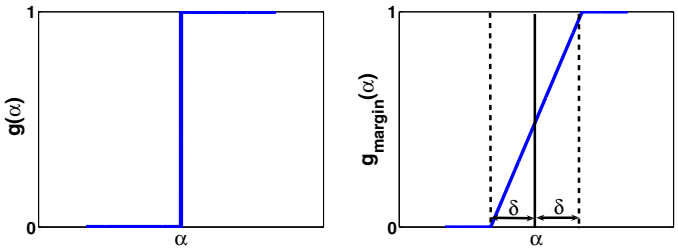


Fig. 2: (a) Heaviside step function $g()$. (b) g_{margin} function to enforce a margin δ around the classification boundary.

a combinatorial search problem of selectively forming synaptic connections on each dendritic subunit. It is based on a modified form of Hebbian learning in which a fitness function c_{ij} for each synapse is calculated as the correlation between synaptic input and the output of the dendrite on which the synapse is located modulated by classification error. The computation of c_{ij} is as follows:

$$\begin{aligned}
&\text{For class 1 cell, } c_{ij} =< x_{ij}b_j sgn(t_1 - y_1) > \\
&\text{For class 2 cell, } c_{ij} =< x_{ij}b_j sgn(t_2 - y_2) > \\
&\qquad\vdots \\
&\text{For class } C \text{ cell, } c_{ij} =< x_{ij}b_j sgn(t_C - y_C) >
\end{aligned} \tag{4}$$

where $< \ . \ >$ indicate averaging over the entire training set and $sgn()$ is the signum function with a value of 1 for $t_i > y_i$, -1 for $t_i < y_i$ and 0 for $t_i = y_i$. Since, we use binary weights such that $w_{ij} = 1$ if a connection exists and 0 otherwise, the learning process involves the formation and elimination of synapses instead of a weight modification. The fitness function serves as a guide in this process–synapses with most negative c_{ij} are candidates for replacement. Since we operate with fixed synaptic resources, this synapse is replaced by a synapse having high c_{ij}. Hence, we modify the grouping of connections on a dendrite (or equivalently 'morphology') guided by a biologically-realistic Hebbian learning rule. Also, the calculation of c_{ij} can be implemented easily in hardware since it needs only local information while the global error signal is reduced to a binary one.

The algorithm for modifying connections based on the fitness function is same as the one described earlier in [4] and we do not describe it again. However, it was found in [4] that the testing error on rate encoded spike train inputs was $\approx 5 - 10\%$ higher than training errors for static inputs implying poor generalization. Next, we present a modification of the learning rule to rectify this issue which is one of the main contributions of this work.

D. Improving the Generalization Performance of the Model

We propose a modification of the dendritic learning rule used in [4] which involves the use of $g_{margin}()$ function (Fig. 2(b)) to train the model while the Heaviside step function $g()$ is used during testing only. This function enforces a margin δ around the classification boundary and improves generalization performance similar to margin maximization in SVM [7]. We

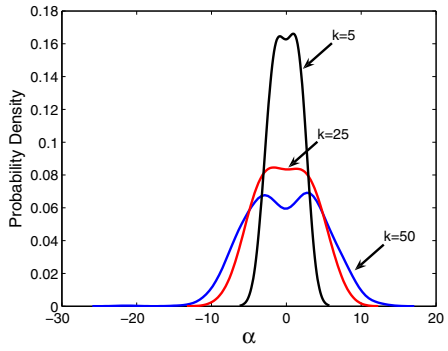


Fig. 3: Determination of δ for $g_{margin}()$ function for a 2-class classification problem. The distribution of α for misclassified spiking inputs. $P = 500$ and $m = 20$.

can define $g_{margin}() : \Re \rightarrow \Re$ as:

$$\begin{aligned} g_{margin}(\alpha) &= 1 \text{ if } \alpha \geq \delta \\ &= 0 \text{ if } \alpha \leq -\delta \\ &= \frac{0.5}{\delta}\alpha + 0.5 \text{ otherwise} \end{aligned} \tag{5}$$

where α denotes $a_i - a_j$ as shown in equation 3. The aim of using this function is to improve the noise tolerance and generalization performance of the model.

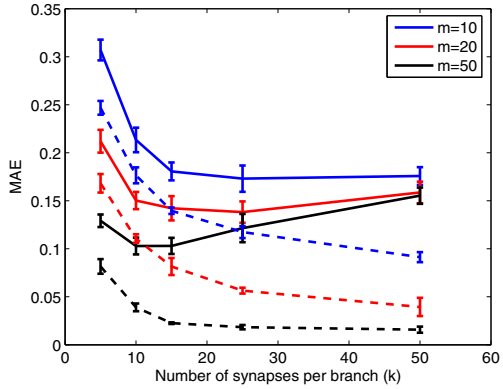
We found that the choice of δ depends on the cell geometry– hence, an adaptive method was used to set the value of δ. An initial step of training was performed *using the* $g()$ *function* and after the learning process was completed, the network was tested on the spike train versions. The values of α were recorded for all cases in which the network misclassified the spiking inputs. Since the worst case of misclassification corresponds to the highest value of α for a wrongly classified pattern, denoted by α_{max}, we try to alleviate this problem by introducing a margin of α_{max} around the classification boundary. This will lead to a learning process in which most of the misclassified patterns lying close to the boundary will get pulled towards the correct classification region. However, at some stage during learning and also for smaller k, this choice of δ might make learning very difficult. Therefore, we assign an initial $\delta_0 = \alpha_{max}$ and if the learning algorithm gets stuck in the same local minimum for 5 consecutive times, the δ value is reduced by 80%.

As an example, we show the results of applying this algorithm in a 2-class classification problem of $P = 500$ binary input patterns. This experiment was repeated for NL-cells with different number of branches ($m = 10, 20$ and 50) and number of synapses per branch ranging from $k = 5$ to 50. The distribution of α as a function of cell geometry is shown in Fig. 3. We can see that the distribution of α varies with k, increasing with both k and m (not shown in the Figure).

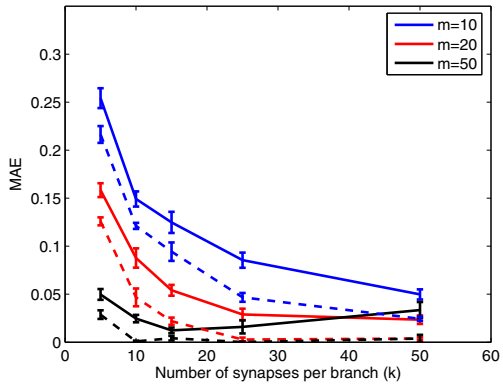
III. Results

A. Improved Generalization on Random Patterns

First, we present the results to show the improved generalization performance of the network utilizing the $g_{margin}()$ function for training for a 2-class classification problem similar



(a)



(b)

Fig. 4: Improvement in generalization performance on spiking inputs (solid) when $P = 1000$ patterns are trained using (b) $g_{margin}()$ function instead of (a) Heaviside step function $g()$. The training errors are shown in dashed lines.

to [4]. $P = 1000$ random patterns were generated from a 40-dimensional Gaussian distribution and then mapped to 10 binary receptive fields. These 400-dimensional patterns were randomly assigned to class (+) and (-). The model was trained on these 'static' patterns while for testing, the static binary patterns were converted into spiking patterns by mapping the input values 1 and 0 into Poisson spike trains with mean firing rates $f_{high} = 250$ Hz and $f_{low} = 1$ Hz respectively.

Fig. 4(a) shows the training and testing errors in dashed and solid lines respectively when the model is trained using the $g()$ function for different values of m and k. It can be seen that the testing errors are far worse than the ones in training implying poor generalization. The differences are particularly higher (by about $9 - 13\%$) at larger k. Fig. 4(b) shows the testing errors for training done using $g_{margin}()$ function. There is significant improvement in both the training and testing errors in this case. The generalization performance on spike patterns improves drastically and at larger k the testing errors are higher by only $2 - 3\%$ compared to training errors. Therefore, the training process involving the use of the proposed $g_{margin}()$ function improves the generalization performance on spiking inputs by about $7 - 10\%$. Hence, we have utilized this modified training method in our simulations for classifying multi-class input

patterns as discussed next.

B. Performance Evaluation on Multiclass Classification Problem

We have obtained results for multi-class classification by training the model of nonlinear dendrites on the MNIST dataset consisting of grayscale images of handwritten digits belonging to one of the 10 classes (0 to 9). It is a benchmark machine learning dataset used previously to test the performance of many classification algorithms [8]. Here, we have compared the performance of our algorithm on MNIST with that of two other classifiers consisting of a network of spiking neurons [1], [9].

The input patterns consisting of 28x28 gray-scale images were converted into 784-dimensional binary vectors by thresholding. The digits belonging to the $C = 10$ classes were used to train on 10 NL-cells consisting of $m = 200$ dendrites and $k = 25$ synapses on each dendrite (hence, total synaptic resource per NL-cell $s = 5000$), where each cell learned to respond preferentially to a class. The training set consisted of $10,000$ patterns randomly selected from the full MNIST dataset, with 1000 patterns belonging to each class. The testing set consisted of a total of 5000 patterns. After the training process, the generalization performance was evaluated on the testing set and classification accuracies were calculated.

TABLE I: PERFORMANCE MEASURES FOR CLASSIFICATION OF MNIST DATASET

Criteria	[9]	[1]	This work
#Train samples	$20,000$	$120,000$	$10,000$
#Test samples	$10,000$	$10,000$	5000
#Neurons	150	1000	10
#Dendrites	150	1000	2000
#Synapses	$117,600$	$647,000$	$50,100$
Test Acc.	96.5	94.09	90.26

Table I compares the performance measures of the different spiking classification models on the MNIST dataset along with the resources used for classification. We trained our model on a smaller training set of $10,000$ patterns compared to the other references due to limitations of computing resources. We have also compared the size of the network used in each case for classification of MNIST digits. The total number of neurons in the network excluding those in the input layer are mentioned in Table I. 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 proposed network achieves $4 - 6\%$ less accuracy than the other models. However, this reduced classification accuracy was achieved due to smaller number of training samples and limited network size. Also, due to the large computational time, we could not run the learning algorithm for a long time–we are currently procuring better computational resources to address these issues. Nevertheless, it can still be seen that our proposed method utilizes far less number of neurons and synapses than [1] and we can expect to achieve similar performance by increasing number of training examples. Compared to [9], we expect to achieve similar accuracy versus synaptic resource tradeoff since the number of nonlinear functions expressible are similar in both cases (we can consider each neuron in the population of output neurons in [9] to be acting similar to a dendrite). However, for the case of on-chip learning, we expect our learning rule to have an easier hardware implementation since the dendritic polynomial nonlinearity is much simpler than implementation of a full neuron. Also, we do not depend on the presence of an ideal amount of noise for good performance.

IV. Conclusion and Discussion

We have used a model consisting of nonlinear dendrites [4] to perform multi-class classification of high dimensional binary vectors of rate encoded spike trains. We have also presented a modification of the earlier learning rule to improve the generalization performance of the model. The new rule reduces the difference between training and testing errors by $7 - 10\%$ compared to the earlier one. Finally, we presented the results of classification of handwritten digits from the MNIST database using our modified algorithm and compared the performance with other spiking classification algorithms. Our proposed method achieved about 90% accuracy on this task when trained on $10,000$ samples while using $< 50\%$ synapses compared to other approaches. This performance can be improved further by using more training examples and is currently being investigated.

References

[1] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-Time Classification and Sensor Fusion with a Spiking Deep Belief Network," *Frontiers in Neuroscience*, vol. 7, pp. 1–13, Oct. 2013.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–76, Feb 2008.

[3] V. Chan, Shih-Chii Liu, and A. van Schaik, "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 1, pp. 48–59, 2007.

[4] S. Hussain, R. Gopalakrishnan, A. Basu, and S.-C. Liu, "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*, Aug. 2013.

[5] P. Poirazi and B. Mel, "Impact of Active Dendrites and Structural Plasticity on the Memory Capacity of Neural Tissue," *Neuron*, vol. 29, pp. 779–96, Mar 2001.

[6] J. Lazzaro, S. Ryckebusch, M. Mahowald, and C. Mead, "Winner-Take-All Networks of O(N) Complexity," in *Proceedings of the Neural Information Processing Systems*. 1988, pp. 703–11, The MIT Press.

[7] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, MacMillan Publishing Company, 1994.

[8] Y. L. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[9] J.M. Brader, W. Senn, and S. Fusi, "Learning Real-World Stimuli in a Neural Network with Spike-Driven Synaptic Dynamics," *Neural Computation*, vol. 19, pp. 2881–2912, 2007.