# Hardware efficient, Neuromorphic Dendritically Enhanced Readout for Liquid State Machines

Subhrajit Roy, Arindam Basu and Shaista Hussain

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798
Email: subhrajit.roy@ntu.edu.sg, arindam.basu@ntu.edu.sg

***Abstract*— In this article, we describe a new neuro-inspired, hardware-friendly readout stage for the liquid state machine (LSM) that is suitable for on-sensor computing in resource constrained applications. Compared to the state of the art parallel perceptron readout (PPR), our readout architecture and learning algorithm can attain better performance with significantly less synaptic resources making it attractive for VLSI implementation. Inspired by the nonlinear properties of dendrites in biological neurons, our readout stage incorporates neurons having multiple dendrites with a lumped nonlinearity (two compartment model). The number of synaptic connections on each branch is significantly lower than the total number of connections from the liquid neurons and the learning algorithm tries to find the best '*combination*' of input connections on each branch to reduce the error. Hence, the learning involves network rewiring (NRW) of the readout network similar to structural plasticity observed in its biological counterparts. We show that even while using binary synapses, our method can achieve $2.4-3.3$ times less error compared to PPR using same number of high resolution synapses. Conversely, PPR requires $40-60$ times more synapses to attain error levels comparable to our method.**

## I. Introduction

There is a pressing need to develop low-power, machine learners to enable on-sensor processing in the various medical sensors that have been developed in the last decade to monitor a wide range of physiological signals. For example, in brain-machine interfaces, a low-power machine learner that can decode movement intentions [1] using extracellularly recorded spikes from the motor cortex directly on-chip can drastically reduce the power needed to wirelessly stream this massive data. Such spike based learners can also process the artificial spike trains from spike based retinas [2] or cochleas [3].

One hardware friendly candidate architecture is provided by the Liquid State Machine (LSM) [4], which requires training of very few weights while the others can be random. The structure of LSM, shown in Fig. 1, has three stages: an input layer which projects the input pattern into the the second stage or the liquid, which is a recurrent neural network (RNN) with randomly weighted, mostly local interconnections performing the task of mapping the input to internal states. These states are then read by the third stage or the readout circuit to provide the overall system output. The readout is trained in a job-specific way which requires updating the weights of the synapses connecting the liquid and the readout. LSM provides a big advantage over other RNNs since only the weights of the synapses connecting the RNN to the readout

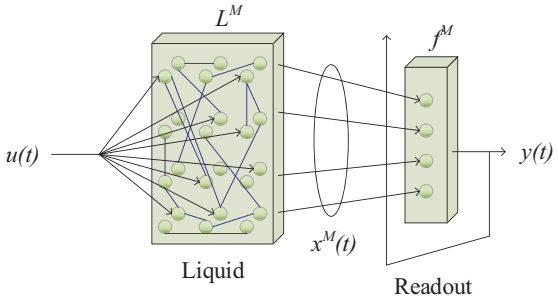



Fig. 1: The structure of LSM comprising three stages.

need to be modified in an LSM, while the interconnections within the RNN pool are fixed. This is clearly useful for VLSI implementations since tunability or high resolution are not needed for most weights. However, the number of tunable weights needed in the readout stage for good performance may become a bottleneck. The state of the art readout stage of LSM is usually composed of a single layer of parallel perceptrons (we denote LSM with a parallel perceptron readout as LSM-PPR) that are trained by the *p*-delta algorithm [5]. For this readout stage of LSM-PPR, the number of tunable weights will be very high and equal to ( $L \times n$ ), where $L$ and $n$ are the number of liquid and readout neurons respectively, thereby making it infeasible for low-power smart sensors. To decrease the number of tunable synapses, neuromorphic systems often use an asynchronous multiplexing technique called address event representation (AER) where the connection matrix is stored in a configurable digital memory. Using AER, it is possible to have a large number of synapses for readout; however, the huge power dissipated in accessing memory for every spike makes this solution infeasible for low-power applications.

In this paper, we propose a novel architecture and training procedure for the readout stage of LSM that is inspired by the nonlinear processing properties of dendrites and structural plasticity (forming or breaking of synapses, re-routing of axonal branches etc.) in biological neurons. This method, which we refer to as LSM with dendritically enhanced readout (LSM-DER), has the following benefits:

- It provides better results as compared to *p*-delta which is the current state of the art algorithm for the training of LSM readout [6].
- It uses much less synaptic resources than parallel per-

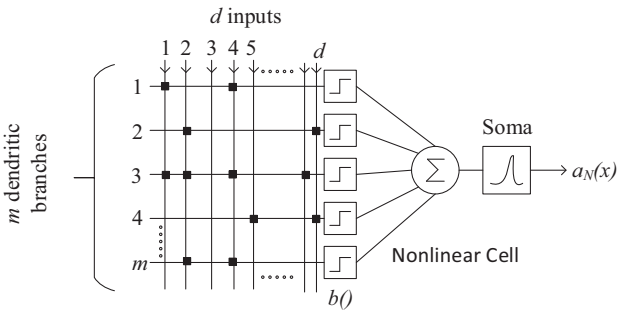



Fig. 2: A neuronal cell with $m$ active dendritic branches.

ceptrons while achieving comparable performance thus making it feasible for implementation in low-power smart sensors.

- The synapses connecting the liquid and readout can even have binary values without compromising performance. This is also very useful in hardware implementations since it removes the need for high resolution weights.

We have earlier presented a dendritic neuron with network rewiring (NRW) rule for classifying high dimensional binary rate encoded patterns [7]. The primary differences in our current work compared to the earlier one are the modification needed to handle arbitrary spike train inputs (not only rate codes), its application to LSM and the modification of the NRW rule for application to approximation problems.

## II. Background and Theory

We will first describe the dendritic neuron model with non-spiking inputs $\mathbf{x}\epsilon\mathbb{R}^d$, then describe the NRW learning rule when these neurons are employed as readout for LSM and finally show how to use this neuron with spiking inputs.

### A. Model of Nonlinear Dendrites

Mel and Poirazi [8] showed that neurons with lumped dendritic nonlinearity possess higher storage capacity than their linear analogue. Such a nonlinear (NL) cell is depicted in Fig. 2 which has $m$ identical branches connected to it with each branch having $k$ excitatory synapses. If $\mathbf{x}\epsilon\mathbb{R}^d$ is an input vector to this system $(d >> k)$, then each synapse is excited by any one of the $d$ dimensions of the input vector. The output response of $j^{th}$ dendritic branch is calculated as a nonlinear weighted sum of the $k$ synaptic inputs connected to the branch and is given by $b(\sum_{i=1}^{k} w_{ij}x_{ij})$, where $b()$ is the dendritic nonlinearity modeled as a nonlinear activation function, $w_{ij}$ is the weight of the $i^{th}$ synapse on the $j^{th}$ branch and $x_{ij}$ the input arriving at that particular synaptic connection. For ease of implementation, we consider binary synapses in this model, i.e. $w_{ij}\epsilon\{0,1\}$. Combining all the dendritic responses the overall output $a_N(\mathbf{x})$ of the neuronal cell is given by:

$$a_N(\mathbf{x}) = \sum_{i=1}^{m} b(\sum_{i=1}^{k} w_{ij}x_{ij}) \tag{1}$$

For both the classification and approximation tasks, two NL-cells are employed and the final output $y$ was calculated as:

$$y = g[a_N^+(\mathbf{x}) - a_N^-(\mathbf{x})]. \tag{2}$$

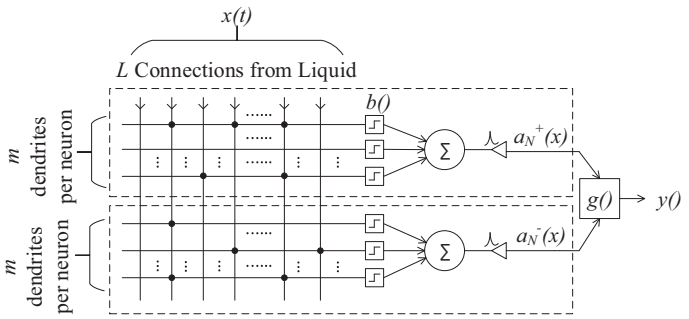



Fig. 3: In LSM-DER, the readout stage following the liquid is composed of two NL-cells. The final readout output y() is obtained by taking the difference of the output of the two cells and passing it through the function g().

where $a_N^+$ and $a_N^-$ are the outputs of the individual neurons and $g : \mathbb{R} \rightarrow \mathbb{R}$ denotes the task specific network function. For the 2-class classification task, $g$ is the signum function. The final output $y$ then takes a value of either 1 or 0 implying that the pattern belongs to $'+'$ or $'-'$ categories respectively. For the approximation task, $g$ has to produce a continuous range of values and hence is chosen to be $g(z) = \frac{1}{1+exp(-(z/2))}$.

### B. Liquid State Machine with dendritically enhanced output

In the LSM-DER architecture shown in Fig. 3, the liquid network described in [4] is followed by the two neuron combination mentioned in the last section acting as readout. For each of the $m$ branches, $k$ synaptic contacts with weight 1 are initially formed by randomly selecting afferents from the $d = L$ input lines, where $L$ is the number of liquid neurons. The NRW algorithm tries to find a new wiring diagram between the liquid neurons and readout that can minimize error. The learning process is comprised of the following steps in every iteration:

1) For each pattern of the entire training set, compute the output $y$ and compare with the target $t$ to get the Mean Squared Error (MSE). A batch learning scheme is applied by averaging MSE over the $P$ patterns in the training set.
2) A random set $T$ of $n_T$ synapses were selected for probable replacement. The performance index $c_{ij}$ corresponding to $i^{th}$ synapse of the $j^{th}$ branch was calculated for each synapse in the set $n_T$ as $c_{ij} =< x_{ij}b'_j(t-y) >$ for the positive cell and $c_{ij} = - < x_{ij}b'_j(t-y) >$ for the negative cell. Here, $<>$ indicate averaging over the training set.
3) The synapse with the lowest value of performance index $(c_{ij})$ in $T$ was labeled for replacement with the synapse with the highest value of performance index from another stochastically chosen replacement synapse set $R$ having $n_R$ of the $d$ input lines. The set $R$ was created by placing $n_R$ 'silent' synapses from $d$ input lines on the branch with the lowest $c_{ij}$ synapse. They do not contribute to the calculation in step (1).
4) The synaptic connections were updated if the replacement led to a decrease in MSE. If there is no such

reduction of MSE, a new replacement set $R$ is chosen. If $t$ such choices of $R$ do not reduce MSE, it is assumed that the algorithm is stuck in a local minima and connection changes are made in an attempt to escape the local minima even if it increases the MSE .

5) The above mentioned steps were repeated for a $max_{iter}$ number of iterations after which the algorithm is terminated and the connection corresponding to the best minima is saved as the final connection.

This performance index is directly related to the gradient of the error function and modifying the connections in the prescribed manner leads to gradient descent learning [7], [8].

The two main differences in our method as compared to [7], [8] are in the hardware friendly forms of correlation $c_{ij}$ and the particular functional form of dendritic nonlinearity $b()$ that we used. The dendritic nonlinearity $b()$ used in our simulations was $b(x) = x^2/x_{thr}$ whereas in earlier work [8] nonlinearities like $b(x) = x^{10}$ were used. The reason for this choice is that for a quadratic nonlinearity, the value of its derivative, needed to compute $c_{ij}$, is a scaled version of the input and leads to easy VLSI implementation. For the classification task in [7], the functional form of $c_{ij}$ used was $c_{ij} =< x_{ij} b_j f(t-y) >$ where $f$ is the signum function. However, we found that for the approximation task shown here, it was necessary to have the exact value of error $(t-y)$ in the calculation of $c_{ij}$. Also, any expansive nonlinearity (like a square law) will result in unrealistically large output $b(x)$ for large values of $x$. Hence, we included a saturation level, $x_{sat}$ at the output such that for $b(x) > x_{sat}$, $b(x) = x_{sat}$. This saturation reduces power dissipation in analog current mode implementations of $b()$ and is also bio-realistic.

### C. Spike trains as input to Dendritic Neurons

The output of the liquid neurons are spike trains, while the earlier theoretical description of the NL cells had inputs $\mathbf{x} \in \mathbb{R}^d$. Thus, to use the NL cells to act as readout, we need a method to transform spike trains to act as inputs to the NL cells. Suppose, the arbitrary spike train $s(t)$ is given by $s(t) = \sum_{t_f} \delta(t - t_f)$ where $t_f$ indicate the spike firing times. One way to convert this to an analog waveform is to convolve it with a low-pass filtering kernel. We choose a fast rising, slow decaying kernel function, $h(t) = I_0(e^{-\frac{t}{\tau_s}} - e^{-\frac{t}{\tau_f}})$, that mimics post-synaptic current (PSC) waveform and is popularly used in computational neuroscience. Here, $\tau_f$ and $\tau_s$ denote the fast and slow time constants dictating the rise and fall times respectively and $I_0$ is a normalizing factor. Hence, the final filtered analog waveform, $s_a(t)$ corresponding to the spike train $s(t)$ is given by $s_a(t) = \sum_{t_f} h(t - t_f)$. Finally, to train the NL-cells in the readout, we need a set of discrete numbers which we obtain by sampling $s_a(t)$ at a desired temporal resolution $T_s$. The sampled waveform, $s'_a(t)$ is given by $s'_a(t) = s_a(t) \sum_i \delta(t - iT_s)$. Therefore, for the $L$ spike trains produced by the liquid neurons, if the temporal duration of the spike trains are $T$, then it will result in a total of $\lfloor T \backslash T_s \rfloor$ samples $\mathbf{x_i} \epsilon \mathbb{R}^{\mathbf{L}}$. For our simulations, we have chosen the temporal resolution $T_s = 25$ ms.

## III. Experiments and Results

### A. Problem Description

In this sub-section, we describe the two tasks used to demonstrate the performance of our algorithm. Both of these are standard problems that are shown in the original publication on LSM [4] and are included as examples in the LSM toolbox [9]. Also, we chose one task to be a classification and the other to be an approximation since these are representative of the class problems solved by LSM. We present brief descriptions here, while the readers can refer to [4], [9] for more details.

*1) Task I: Classification of spike trains:* The first benchmark task we have considered is the Spike Train Classification problem [9]. Two poisson spike trains having mean frequency of 20 Hz and length 0.5 sec are labeled as templates 1 and 2. These spike trains are used as input to the LSM, and the readout is trained to identify each class. Next, a jittered version of each template is generated by altering each spikes within the template by a random amount and the task is to correctly identify the class from which it has been drawn.

*2) Task II: Retrieval of sum of rates:* The next task is an approximation problem. Four poisson spike trains, the firing rates of which are modulated by a randomly chosen function $r(t) = A + Bsin(2\pi ft + \alpha)$ lying in the range $(0, 1)$, are injected into the liquid. At any point of time $t$, the job of the network is then to give as output the normalized sum of input rates averaged over the last 30 ms.

### B. Choice of Parameters

The values of $m$, $k$, $L$, $\tau_s$, $\tau_f$, $n_T$, $n_R$, $max_{iter}$ and $x_{sat}$ were kept to be 7, 10, 140, 7.5 ms, 30 ms, 15, 25, 1000 and 75 respectively for both the tasks. Hence, the total number of synapses in the readout stage is $2 \times m \times k = 140$ which is same as the number of synapses needed by a single perceptron.

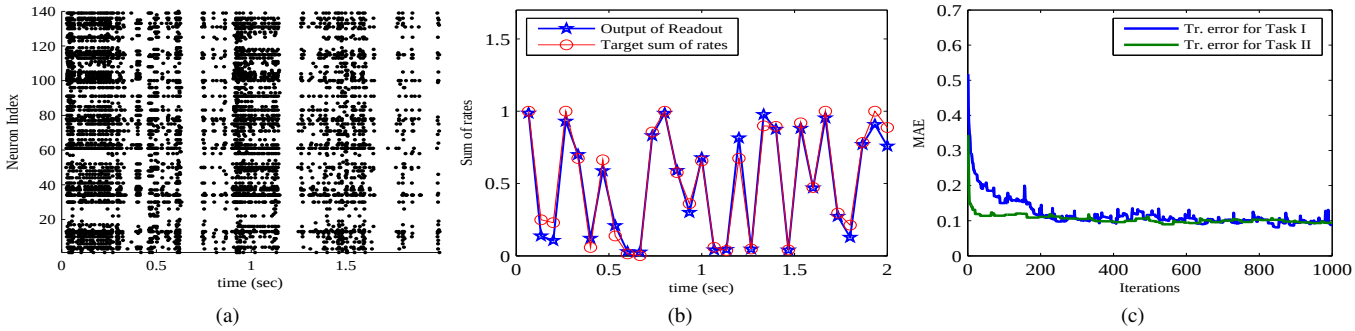
The value of $x_{thr}$ was taken as 1.8 and 7 for Task I and Task II respectively. For selecting $x_{thr}$, we need to note that the operating principle of the NRW learning rule is that the nonlinear function $b()$ should give a supra linear output when *more than one* synaptic inputs are co-activated. For the nonlinear function used here, $b(x) > x$ for $x > x_{thr}$. Hence, the choice of $x_{thr}$ is given by:

$$\overline{I_{syn}} < x_{thr} < 2\overline{I_{syn}} \tag{3}$$

where $\overline{I_{syn}}$ denotes the average post-synaptic current from an active synapse. Performing this calculation for a large number of input patterns for both tasks, we obtained the values of $x_{thr}$ mentioned.

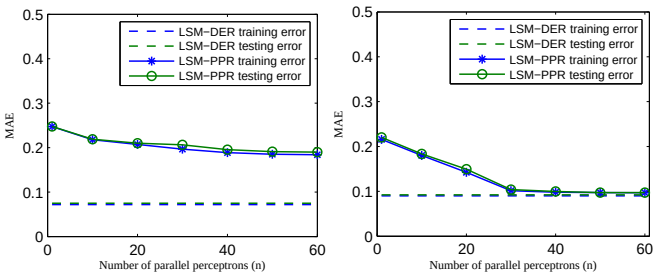
### C. Results: Performance of LSM-DER and NRW algorithm

The proposed readout is separately trained for Task I and Task II. Similar to the method followed in [4], [9], we calculate mean absolute error (MAE) by averaging the error in approximation or classification across all the patterns. Figure 4(a) shows a sample output of the liquid for task II while Fig. 4(b) plots both the target function and the output of the




(a) (b) (c)

Fig. 4: (a) The output of the liquid for an instance of Task II (b) The output produced by LSM-DER after training is superimposed on the target function for a randomly chosen time window and show a very close match. (c) The training error obtained by LSM-DER in each iteration are plotted for Task I and Task II when trained on a set of 200 patterns averaged over 10 trials.




(a) (b)

Fig. 5: Performance Comparison of LSM-DER and LSM-PPR with varying $n$ (MAE averaged over 10 trials) for (a) task I and (b) task II.

readout stage (during a randomly selected time window) of LSM-DER for a test pattern in Task II. The figure shows that the readout is able to approximate the desired sum of rates very closely. Moreover, Fig. 4(c) demonstrates the convergence of the NRW algorithm by plotting the error against iterations for both the tasks. The decrease in the error with iterations proves the successful operation of the NRW algorithm in finding new suitable connections.

### *D. Results: Comparison between LSM-DER and LSM-PPR*

Next, we compare the performance of our proposed method with the state of the art–LSM-PPR. The results for LSM-PPR are obtained by using the standard LSM toolbox provided in [9]. An important parameter in LSM-PPR is the number of readout neurons denoted by $n$. The number of synapses in the readout layer is $L \times n$–hence, larger values of $n$ require more synaptic resources. Figure 5 plots the variation in error for both tasks when $n$ is increased from 1 to 60. The training and testing error of LSM-DER is plotted as a constant line and is always less than that obtained by LSM-PPR. Note that the synaptic resource consumed by LSM-DER is same as that for $n = 1$ or a single perceptron. Hence, we can conclude that, when $n = 1$, LSM-DER can attain 3.3 and 2.4 times less error than LSM-PPR with high resolution weights for Task I and II respectively. Moreover, LSM-PPR requires $40-60$ times more synapses to attain error levels comparable to LSM-DER.

## IV. Conclusion

In this article we have proposed a novel architecture (LSM-DER) and an efficient learning rule (NRW) for the readout stage of Liquid State Machine. Inspired by the nonlinear properties of dendrites in biological neurons, the readout neurons of LSM-DER employs multiple dendrites with lumped nonlinearities. The results depict that LSM-DER along with NRW is able to achieve better performance than the state-of-the-art LSM-PPR with much less number of synapses thus being suitable for VLSI implementations. Moreover, unlike LSM-PPR which uses high precision analog syanptic weights, LSM-DER uses binary synapses thus being very advantageous for hardware implementations.

## References


[1] S. Acharya, F. Tenore, V. Aggarwal, R. Etienne-Cummings, M.H. Schieber, and N.V. Thakor, "Decoding Individuated Finger Movements Using Volume-Constrained Neuronal Ensembles in the M1 Hand Area," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 1, pp. 15 –23, feb. 2008.

[2] E. Culurciello, R. Etienne-Cummings, and K.A. Boahen, "A biomorphic digital image sensor," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 281 – 294, feb 2003.

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

[4] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.

[5] P. Auer, H. Burgsteiner, and W. Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons," *Neural Networks*, vol. 21, no. 5, pp. 786 – 795, 2008.

[6] E. Hourdakis and P. Trahanias, "Improving the Classification Performance of Liquid State Machines Based on the Separation Property," in *Engineering Applications of Neural Networks*, vol. 363, pp. 52–62. Springer Berlin Heidelberg, 2011.

[7] 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 *IEEE Intl. Joint Conference on Neural Networks*, 2013.

[8] P. Poirazi and B. W. Mel, "Impact of active dendrites and structural plasticity on the memory capacity of neural tissue," *Neuron*, vol. 29, no. 3, pp. 779–796, Mar. 2001.

[9] T. Natschläger, H. Markram, and W. Maass, *Computer models and analysis tools for neural microcircuits*, chapter 9, Kluver Academic Publishers (Boston), 2002.