Critical Branching Neural Computation

Christopher T. Kello and Marshall R. Mayberry

***Abstract*—Liquid state machines have been engineered so that their dynamics hover near the "edge of chaos" [1], [2], where memory and representational capacity of the liquid were shown to be optimized. Previous work found the critical line between ordered and chaotic dynamics for threshold gates by using an analytic method similar to finding Lyapunov exponents [3]. In the present study, a self-tuning algorithm is developed for use with leaky integrate-and-fire (LIF) neurons that adjusts postsynaptic weights to a critical branching point between subcritical and supercritical spiking dynamics. The tuning algorithm stabilizes spiking activity in the sense that spikes propagate through the network without multiplying to the point of wildfire activity, and without dying out so quickly that information cannot be transmitted and processed. The critical branching point is also found to maximize memory and representational capacity of the network when used as liquid state machine.**

I. Introduction

Nervous systems tend to be characterized by recurrent loops across a wide range of spatial and temporal scales [4]. In particular, if one traces the branching of synaptic connections projecting out from a given starting neuron, numerous branches can be found to recurrently connect back to the starting neuron. These recurrent loops may consist of a wide range of intervening numbers of neurons, and intervening neurons may range from spatially proximal to distal with respect to the starting neuron.

Spiking dynamics are thresholded and thus inherently nonlinear. When spiking dynamics are instantiated in recurrent loops of various scales, the resulting collective activity is often associated with chaotic dynamics [5], and considered to be complex in this regard. Model systems have been proven to be chaotic [6], and real nervous systems have been observed to exhibit signatures of chaotic dynamics, e.g., in terms of Lyapunov exponents [7], and entropic measures of collective neural activity [8].

Evidence for near-chaotic neural dynamics has led researchers to consider whether this property of complexity might be important for neural information transmission and processing [9], rather than just a by-product of nonlinearities and recurrent loops. One possibility is that near-chaotic dynamics are essential to producing metastable, responsive spiking activity [10]. Neural networks at all scales (from microcircuits to subcortical and cortical structures to whole brains) must be sensitive to external inputs, where external inputs are spikes originating from outside the network in question. If incoming spikes are unable to create or perturb spiking activity in the network, then information would not be transmitted or processed. However, networks can also be overly sensitive to perturbations, in which case incoming spikes may spread like wildfire through the system, again diminishing the ability of incoming spikes to perturb ongoing activity.

A. Critical Branching

A balance can be struck between unresponsive and overly responsive spiking by relating their dynamics to a critical branching process [11]. Critical branching processes describe sequences of discrete choice events, where each "ancestor" event may cause some number of subsequent "descendant" events, which in turn lead to further event branching. The expected number of descendant events is described by σ: the ratio of the number of descendant events over the number of ancestor events. Spikes are the events in our case, and external inputs can be described as ancestor spikes that cause descendant spikes in a given neural system. If $\sigma < 1$, then the number of spikes will diminish over time, and information will not be transmitted throughout the system in terms of propagating spiking activity. If $\sigma > 1$, then the number of spikes will grow over time and eventually come to saturate the network. Critical branching occurs at $\sigma = 1$, the point at which spikes are conserved over time, and can thus propagate throughout the system without dying out or running rampant.

Electrophysiological recordings of neural activity – both *in vitro* [12], [13] and *in vivo* [15] – have provided evidence for critical-branching dynamics. The evidence comes in the form of so-called "neural avalanches" that describe distributions in bursts of neural activity. In slice preparations of rat somatosensory cortex, local field potentials were recorded to measure spontaneous neural activity. This activity was found to occur in bursts that spread over the tissue, and burst sizes were measured in terms of the amount of activated tissue for each time interval, summed over consecutive time intervals for detected activity. An analogous method was used to measure bursts from human electroencephalogram (EEG) recordings during the resting state. In both cases, burst sizes were found to be power-law distributed with an exponent of -3/2 (see Fig 1).

Probabilistic models of critical branching have been shown to simulate power-law distributions of recorded neural activity [Fig 1; [11]], but due to their probabilistic nature, they do not have memory or representational capacity. Specifically, previous models did not include terms for simulating membrane potentials, action potentials, and weights on synaptic connections. Thus, they cannot serve as mechanistic models of neural computation. To our knowledge, one critical-branching model has been reported that uses linear integrate-and-fire neurons and all-to-all connectivity among neurons [16], but the models memory and representational capacities were not tested. The authors also describe an algorithm for tuning synaptic connection weights to the

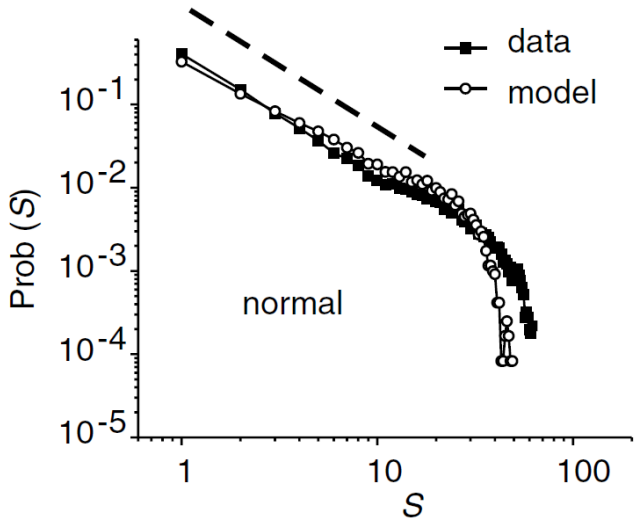

Fig. 1. Power law distributions of burst sizes (S) expressed as a probability function, Prob(S), plotted in log-log coordinates. The dashed line shows an ideal power-law distribution with a -3/2 exponent. Open circles are data from rat somatosensory slice preparations, and filled squares are from a probabilistic critical branching model. Taken from [11].

critical branching point, but they do not report tests of the algorithms performance.

B. Criticality and Liquid State Machines

Critical-branching processes may be useful for regulating neural spiking dynamics, but critical branching is only one kind of model of criticality, and other such models have been shown to support computation. Criticality refers to phenomena observed near phase transitions in systems described by statistical mechanics. In particular, transitions between ordered (regular) and disordered (chaotic) phases have been associated with computational capacity [17], [18].

Most relevant to the current work, "edge of chaos" computing has been simulated in terms of liquid state machines [1], [2]. Liquid state machines use recurrent loops among nonlinear processing units to create dynamical activity that can be perturbed with external inputs. Loops are created using largely arbitrary patterns of synaptic connectivity (e.g., random or stochastic lattice connectivity), the rationale being that irregular patterns of connectivity should capture computationally relevant characteristics of real neural networks [6], although other patterns of connectivity have been explored in the literature that are more neurobiologically motivated [19]. Liquid state machines use a separate "readout" function, such as a linear classifier, to extract information about previous external inputs from current, instantaneous activities. Analyses and simulation results have shown that dynamics in liquid state machines play the role of kernel functions in support vector machines [20], thereby making nonlinear input classifications linearly separable.

In addition to random connectivity, liquid state machines are also typically designed with weights on synaptic connections chosen *a priori* in order to produce computationally useful, general-purpose dynamics. For the edge-of-chaos liquid state machine, Bertschinger and Natschläger sampled weights at random from Gaussian distributions with zero mean and parameterized variances. They used discrete linear threshold gates as their processing units, as well as biases towards positive or negative outputs served as a second parameterization on discrete network dynamics. Lastly, each neuron was randomly connected to K other neurons in the network, where K served as a third parameterization on network dynamics.

These three parameters were set by hand to the critical line between ordered and chaotic dynamics using a sampling approach for discrete dynamics that is analogous to estimating Lyapunov exponents for continuous dynamics [3]. Networks were tested when poised near the critical line, and to either side of it. Results showed that computational performance was enhanced near the critical line, where performance was related to the ability of network dynamics to encode information about external inputs over time. In particular, network dynamics were perturbed by random bit sequences of external inputs, and a linear classifier was trained on arrays of neuronal outputs at each time t. Classifiers were trained on the 3-bit 3-bit parity function for 3-bit sequences going back in time $t-\tau$. Thus to obtain accurate classifications, the network state at time t needed to code external inputs going back $t-\tau$ steps, and the state needed to linearly separate sequence information presented in a nonlinearly separable form.

Results showed that, when tuned near the critical line, liquid state machines produced dynamics with the longest and most accurate encodings (as defined above) of external inputs. Critical line tuning also resulted in performance that scaled up most sharply with increased numbers of model neurons.

II. Self-Tuned Critical-Branching Model

Here we present a model that simulates neural computation near criticality, but in a network of spiking neurons instead of threshold gates. The model includes a self-tuning algorithm that is local to each neuron's postsynaptic array, and local in time with respect to each presynaptic firing event and its immediate postsynaptic consequences. By comparison, Natschläger et al. [2] proposed a presynaptic tuning algorithm for threshold gate neurons, using moving average computations to estimate threshold crossing probabilities. The model is described next, followed by analyses showing the efficacy of tuning in terms of performance as a liquid state machine.

A. Leaky Integrate-and-Fire Neurons

For each time step t, the membrane potential of every model neuron is determined from the weighted sum of spikes from presynaptic neurons, as well as from external inputs, according to the equation

$$\mathbf{v}^{t+1} = \delta\left[(\mathbf{v}^t + \varepsilon^t) \circ (\mathbf{1} - \mathbf{s}^t) + (\mathbf{W}^t - \zeta\,\mathbf{I})\,\mathbf{s}^t\right], \qquad (1)$$

where $\circ$ denotes the Hadamard product (i.e., element-wise multiplication). The notation $\mathbf{1}$ denotes the column vector of ones. Each term represents:

- δ**:** the leak time constant (set to 0.9);
- $\mathbf{v}^t$**:** the membrane potentials of neurons at time t;
- ε^t**:** external input/perturbation to $\mathbf{v}^t$;
- $\mathbf{W}^t$**:** the weight matrix between neurons;
- ζ **:** an analogue of the refractory period (set to 1.0);
- **I:** the identity matrix;
- $\mathbf{s}^t$**:** the Boolean vector denoting spikes in $\mathbf{v}^t$, i.e.,
$$\mathbf{s}^t \doteq [\mathbf{v}^t \geq 1],$$
using Iverson notation for a Boolean condition.

B. Critical Branching Tuning Algorithm

The tuning algorithm adjusts the postsynaptic weights of a given model neuron so that, when it spikes, one and only one spike is expected to follow over the postsynaptic array of neurons. The algorithm weights each descendent spike relative to its number of ancestor spikes n over its presynaptic array on the preceding time step, i.e,. $1/n$. The algorithm decreases weights projecting out from a given postsynaptic neuron by a factor β when the sum of weighted spikes over its postsynaptic array is greater than one. The algorithm increases weights by β when the sum is less than one (no change is made when equal to one). β was set to 0.01 for all simulations. More formally, the critical-branching tuning algorithm can be described by the following equations, presented for readability

$$\begin{aligned}
\mathbf{c}^t &= \mathbf{S}^t\mathbf{s}^t \\
\mathbf{y}^t &= (\mathbf{c}^t + \overline{\mathbf{c}}^t)^{-1} \\
\mathbf{z}^{t+1} &= (\mathbf{S}^t)^T(\mathbf{s}^{t+1} \circ \mathbf{y}^t) \\
\mathbf{n}^{t+1} &= \text{sgn}(1 - \mathbf{z}^{t+1}) \\
\mathbf{N}^{t+1} &= \mathbf{1}(\mathbf{s}^t \circ \mathbf{n}^{t+1})^T \\
\mathbf{\Delta}^{t+1} &= \beta(\mathbf{S}^t \circ \mathbf{N}^{t+1}) \\
\mathbf{W}^{t+1} &= \mathbf{W}^t + \mathbf{\Delta}^{t+1}
\end{aligned} \tag{2}$$

where, letting $\mathbf{p}^t = \mathbf{S}^t\mathbf{1}$ denote the number of postsynaptic neurons for each presynaptic neuron in $\mathbf{v}^t$, the terms in Equation 2 represent:

- $\mathbf{S}^t$**:** the Boolean matrix defined by $[\mathbf{W}^t \neq 0]$;
- $\mathbf{c}^t$**:** the count of postsynaptic spikes w.r.t. $\mathbf{s}^t$;
- $\mathbf{y}^t$**:** the fraction of postsynaptic spikes to $\mathbf{p}^t$;
- $\mathbf{z}^{t+1}$**:** the sum of presynaptic terms w.r.t. $\mathbf{s}^{t+1}$;
- $\mathbf{s}^{t+1}$**:** the Boolean vector denoting spikes in $\mathbf{v}^{t+1}$;
- $\mathbf{n}^{t+1}$**:** the sign of weight updates w.r.t. $\mathbf{z}^{t+1}$;
- $\mathbf{N}^{t+1}$**:** the signed outer product w.r.t. $\mathbf{s}^t$ and $\mathbf{n}^{t+1}$;
- $\mathbf{\Delta}^{t+1}$**:** the update matrix based on $\mathbf{S}^t$ and $\mathbf{N}^{t+1}$.

It is important to note that the critical-branching tuning equations do not involve any matrix inversions, and indeed, the update weight matrix is only a function of the Boolean matrix $\mathbf{S}^t$ and its transpose. Although current research is focussed on simplifying the equations further, the current formulation suggests that Equations 1 and 2 should be amenable to analysis for convergence. Also, it is interesting that the state vector $\mathbf{s}^t$ has a matrix analogue $\mathbf{S}^t$, the primary function of which is the computation of presynaptic and postsynaptic spikes. Finally, the notation $\overline{\mathbf{c}}^t$ simply denotes the Boolean complement of $\mathbf{c}^t$ to avoid division by zero.

C. Network Connectivity

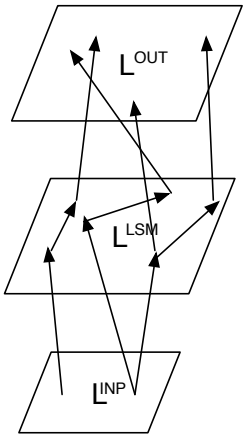

Fig. 2. Schematic of three-layer network architecture used for liquid state machines (not all connections are shown).

All of the liquid state machine results were obtained using the layered connectivity shown in Figure 2. External inputs were presented as "forced" spike patterns on the input layer, which consisted of N neurons. Each input layer neuron was randomly connected to m LSM neurons, of which there were $2N$ in number. Each LSM neuron was randomly connected to $2m$ LSM and output neurons at random, and there were $2N$ output neurons. The numbers of neurons and post-synaptic connections were manipulated in some simulations, but unless noted otherwise, $m = 6$ and $N = 250$.

Classifiers were trained on spike patterns over the LSM layer (see below). Classifiers could also be trained on the output layer, and preliminary results suggest that similar performance is obtained for the output layer. However, in the current simulations, the output layer served a different purpose. Output neurons had no post-synaptic connections, which meant that spikes occurring in the output layer "exited" the network. That is, the critical branching algorithm did not count these spikes as ancestors with descendant spikes. Spiking models that are critical branching need a way for spikes to exit the network. Otherwise, as external inputs create spikes, those spike will fill the network to the point of saturation. The reason is that, in a closed recurrent network, perfect critical branching will cause each spike to propagate forever.

III. Critical-Branching Results

Convergence of the tuning algorithm was tested using the default layered network as described above. Weights were initialized in either a subcritical or supercritical regime by uniformly sampling weights from either a narrow range around zero, $[-0.125 : 0.125]$, or around two (Fig 3). Input layer neurons were divided randomly into two halves, and on each time step, neurons in one of the two halves were forced to spike. For the purpose of testing convergence, this external input procedure was only a means of driving the network with spikes and engaging the tuning algorithm. The

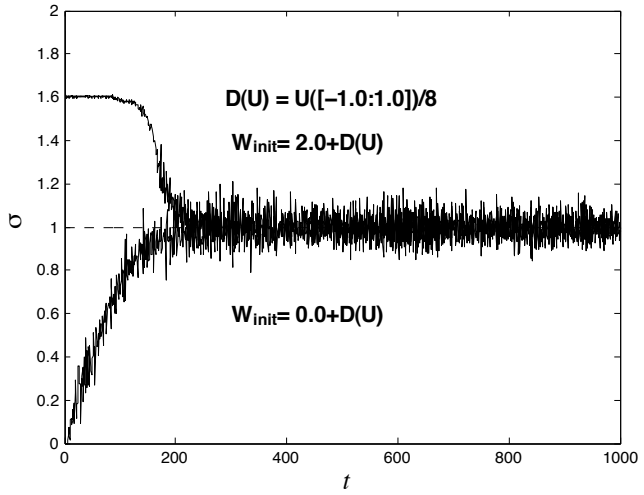

Fig. 3. Instantaneous estimates of the critical branching exponent plotted for each time step of simulation (dashed lines are placed at the critical branching points). Weights were initialized around a mean of either 0 (subcritical) or 2 (supercritical). Initial transients are not shown for sake of clarity.

two halves of the input layer are used again later to test memory and representational capacity of tuned networks as liquid state machines. Finally, critical branching (σ) was estimated instantaneously by dividing the number of spikes at time t by those at $t-1$. Fig 3 shows that the network quickly converges to critical branching and remains there.

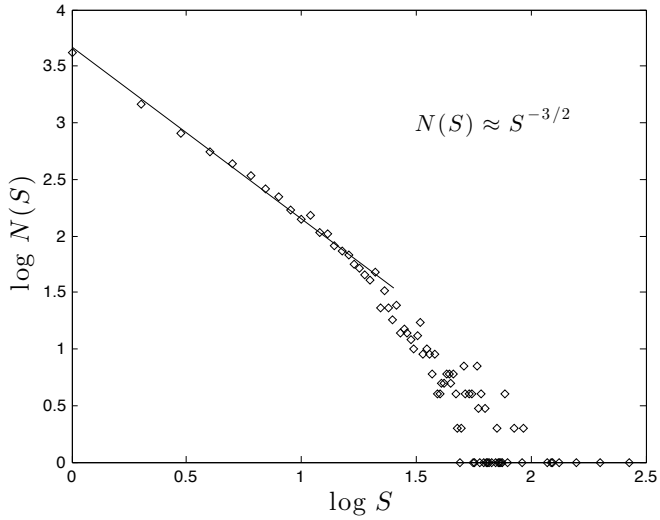

Fig. 4. Histogram of neural avalanche sizes, plotted in log-log coordinates. A regression line (dashed) was fit to the first 20 points of the distribution, and its resulting slope was $\approx -3/2$.

As noted earlier, critical branching is associated with neural avalanches in cortical slice preparations as well as simple probabilistic spiking models. Neural avalanches are observed when network activity is mostly intrinsic, and only occassionally perturbed by small amounts of external input. The layered network prohibits power law distributions in avalanches because spikes exit the network too quickly. Therefore, avalanches were investigated using a single recurrent layer of 1000 neurons, with $m = 6$, and no output layer. Instead of an output layer, spike saturation was avoided by forcing each spike to exit the network with 0.1 probability on each time step. In other words, any given neuron served as an "output" neuron with 0.1 probability when it spiked. The network was tuned to critical branching while driving network activity by forcing each neuron to spike with 0.01 probabilty on each time step. In other words, any given neuron served as an "input" neuron with 0.01 probability on each time step.

Once tuned, the network was driven with "avalanche pings" instead of the procedure just described for initial tuning. To start an avalanche from a silent (i.e. non-spiking) network, a single neuron was forced to spike. Propagated spikes were then counted over time until network activity died out, at which point another neuron was pinged at random, and the process was repeated for 50,000 time steps. Avalanche sizes corresponded to numbers of spikes between pings, and the histogram of avalanche sizes is plotted in Fig 5 in log-log coordinates. The figure shows that simulated neural avalanches followed the predicted power law distribution with a $-3/2$ exponent for most avalanches, i.e., those of small to medium size. The power law fell off for large avalanches due at least in part to limited model size.

IV. Liquid State Machine Results

To assess memory and representational capacity of tuned spiking dynamics, the layered network was tested on three different classification tasks. The first was to compute the XOR function for external inputs $t-\tau$ steps back in time, using only the pattern of spikes produced at time τ. XOR assesses the ability of spiking dynamics to separate external inputs, and maintain separation and representation over time. The second task was to compute the 3-bit parity function over external input sequences τ time steps long, which is nonlinearly separable like XOR, but assesses combinations over successively longer input sequences. 3-bit parity also allows us to compare our results with those of [1], [2]. The XOR and 3-bit parity tasks include an element of noise, in that intervening bits between τ and the current time step must be ignored. The third task was n-bit parity classification going $t-\tau$ steps back in time, which requires memory of all bits going $t-\tau$ steps back in time.

For the XOR and 3-bit parity functions, bit sequences were presented as external inputs by choosing half the neurons in the input layer at random to represent 0, and the other half to represent 1. Bit 0 or 1 was presented to the network at time $t-\tau$ by setting $\varepsilon_i(t-\tau) = 1$ for all i representing either 0 or 1, respectively ($\varepsilon_i(t-\tau) = 0$, otherwise).

Weights were initialized in the range $[-1.0, 1.0]$. Networks were tuned for 5000 time steps on each of which a bit representation was presented on the input layer as external input. Weights were then frozen and external inputs continued to be presented for another 50000 time steps. Networks produced spike patterns over neurons in their output layers on each time step, and patterns for four of every five time steps were

used to train classifiers. Patterns on the remaining time steps were used to test trained classifiers.

A separate, non-spiking perceptron classifier was trained for each classification task and each sampled value of $\tau \in [1, 15]$. Gradient descent was performed on the perceptron weights w_i on the task-specific objective function, $B(t-\tau) = \Theta\left(\sum_i s_i(t) w_i\right))$, where Θ is a binary threshold function, $s_i(t)$ are the spikes produced by neuron i, and $\sum$ ranges over all neurons i. For each classifier, the gradient descent algorithm was run for a total of 1.5 million training epochs, at which point error always reached asymptote for the training spike patterns.

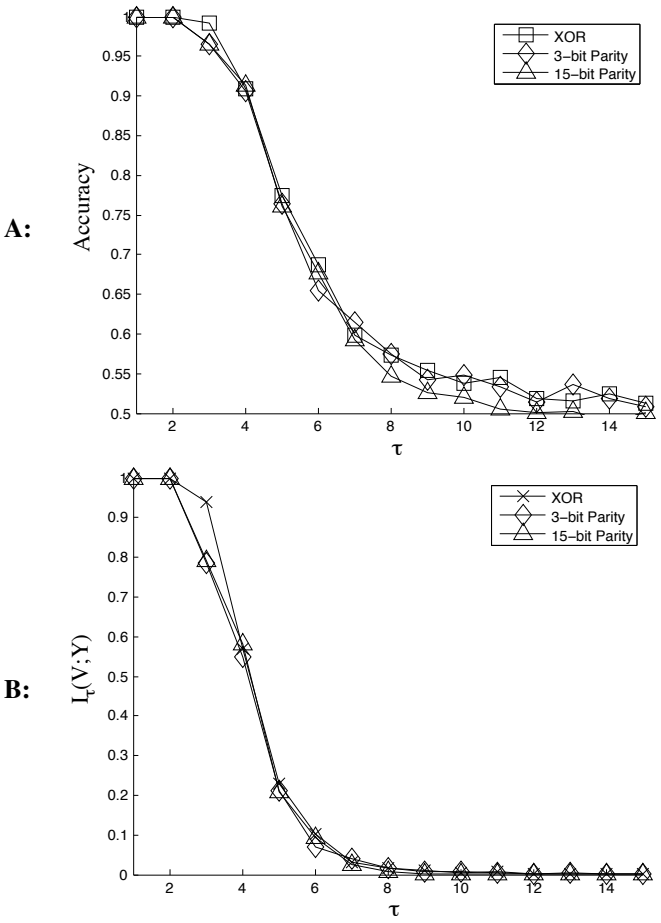

Fig. 5. Liquid state fading memory capacity (τ) for three different functions: XOR, 3-bit parity, and n-bit parity classification. Panel **A** shows performance in terms of accuracy. Panel **B** shows the performance in terms of mutual information, $I(V;Y)$.

Liquid state machine performance on the three classification tasks was evaluated in terms of accuracy (percent correct), as well as mutual information and a corresponding measure of memory capacity. Memory capacity is defined as the sum, $MC = \sum_\tau I_\tau(V;Y)$, of the mutual information, $I_\tau(V;Y)$ between classifier output, $v_\tau(\cdot)$ trained on a delay of τ time steps, and the target function, $y_\tau(\cdot)$. Results in [1] showed this definition of memory capacity was less susceptible to nonlinearities in the output than performance measures premised on temporal correlations such as the determination coefficient originally used to define memory capacity in [21], whose analysis nonetheless inspired the performance measure used in [1]. Fig 5 shows classifier performance in terms of accuracy as well as mutual information, $I_\tau(V;Y)$, as a function of τ. In addition to its ability to capture nonlinear correlations in temporal classification tasks, mutual information has the additional advantage in that it is naturally *linear*: that is, $I_\tau(V;Y)$ lends itself to simple summation over any range of delays because it is based on logarithms. In particular, mutual information relates two random variables, V and Y, with respect to the joint and marginal distributions of their possible outcomes. Mutual information is defined as

$$I(V;Y) = \sum_{v'} \sum_{y'} p(v', y') \lg \frac{p(v', y')}{p(v')\, p(y')} \tag{3}$$

where $p(v', y')$ represents the joint probability that V has outcome v' and Y, outcome y', while $p(v')$ and $p(y')$ are the marginal probabilities of the random variables. More precisely,

$$p(v', y') = P(V = v' \wedge Y = y'), \tag{4}$$

is the joint probability for each possible pair of outcomes, and

$$p(v') = P(V = v') \quad \text{and } p(y') = P(Y = y') \tag{5}$$

are the marginal probabilities of the outcomes alone. If V and Y are statistically independent, then $p(v', y') = p(v')\, p(y')$, in which case the rational argument of the logarithm in Equation 3 is 1, and $I(V;Y)$ is accordingly 0.

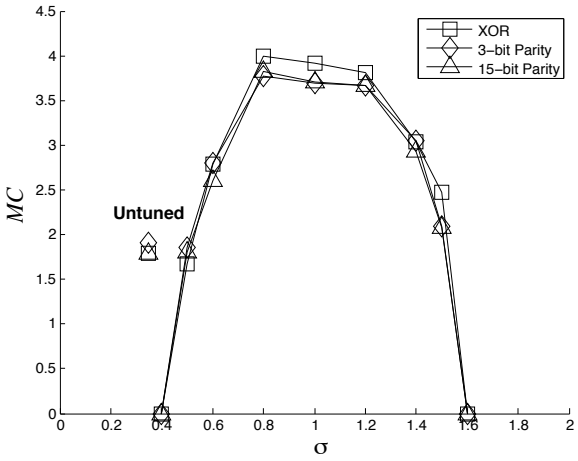

Fig. 6. Liquid state performance for the three different classification tasks (XOR, 3-bit parity, and n-bit parity) plotted as a function of estimated σ after tuning with biased towards subcritical (left of $\sigma = 1$) or supercritical (right of $\sigma = 1$). Memory capacity is plotted from $\tau \in [1, 15]$, and m = 6 for this simulation.

Fig 6 shows liquid state performance as a function of tuning to subcritical, critical, and supercritical dynamics. Tuning was manipulated by making weight changes away

from critical with some probability. Subcritical versus supercritical dynamics were tuned by biasing weight changes in one direction or the other with uniform probability in the range $[0.0 : 0.6]$. This bias was used to achieve a range of σ estimates. Fig 6 shows that performance was maximized near the critical branching point $\sigma \approx 1$.

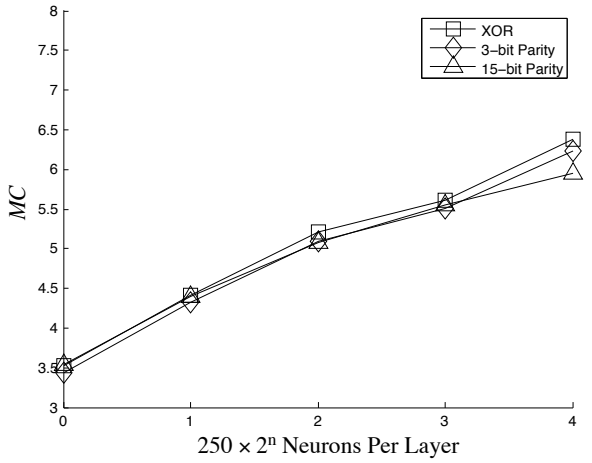

Fig. 7. Liquid state performance in terms of MC for the three different classification tasks (XOR, 3-bit parity, and n-bit parity) plotted as a function of number of neurons per LSM layer (250, 500, 1000, 2000, and 4000).

Fig 7 shows the effect of total number of model neurons on liquid state performance at the critical branching point. For all three classification functions, performance scaled roughly linearly with the number of neurons, as also reported in [1] in their edge-of-chaos liquid state machine. Again, this is to be expected given the logarithmic behavior of mutual information that is inversely proportional to the exponential increase in neurons per layer shown in Fig 7.

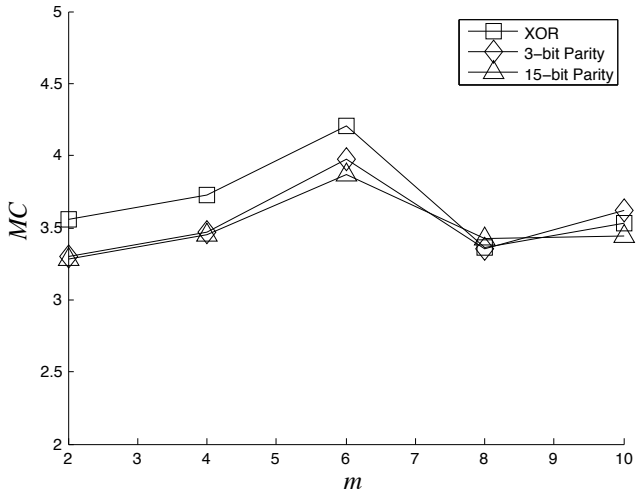

Fig. 8. Liquid state performance for the three different classification tasks (XOR, 3-bit parity, and n-bit parity) plotted in terms of MC as a function of the number of postsynaptic connections (m) on the input layer ($2m$ on the LSM and output layers).

Finally, Fig 8 shows the effect of number of postsynaptic connections on performance after tuning to the critical-branching point. Overall, performance was not greatly affected in the range of m from 2 to 10 postsynaptic connections per input neuron (i.e., 4 to 20 per LSM and output neuron), but performance was slightly maximal at $m = 6$.

V. Conclusion

A network of leaky integrate-and-fire neurons self-tuned to a critical-branching point using an algorithm that is local to each model neuron and its postsynaptic array. The model simulated neural avalanches that are predicted by critical branching, and have been demonstrated in cortical activity measured *in vivo* and *in vitro* [11]–[13], [15], [22]. The model also showed maximal performance as a liquid state machine when tuned to critical branching. Thus the model relates stability of spiking dynamics with memory and representational capacity. Results raised a number of questions to be addressed in future work:

- The algorithm counted numbers of postsynaptic spikes, yet it is not clear how such a count would be implemented in neurophysiology (c.f. presynaptic scaling; [23]).
- The tuning algorithm does not need to differentially adjust each array of synapses onto the postsynaptic neuron (although it does in the current implementation); alternatively, preliminary work indicates that synaptic weights can be uniformly increased or decreased to achieve critical branching. Further work is needed to determine whether such a postsynaptic renormalization algorithm can work in tandem with either Hebbian or synaptic timing dependent plasticity algorithms [24], [25]. One can also imagine learning algorithms that might work with the current tuning algorithm to replace the engineered read-out function used in liquid state machines.
- Neurons tend to be either excitatory or inhibitory, and their connectivity patterns differ from each other, and vary as a function of cortical and subcortical areas. Further work is needed to investigate critical branching and corresponding tuning algorithms in the context of different network architectures.
- Criticality is also associated with long-range correlations (i.e., $1/f$ noise) in fluctuations in neural activity [26]–[29]. Preliminary work indicates that critical branching may also generate $1/f$ noise fluctuations in spiking activity, further suggesting a link between criticality and computational capacity of neural circuits.

Acknowledgment

This work was funded by the DARPA SyNAPSE program. The authors thank Dharmendra Modha and the rest of the IBM SyNAPSE team for their helpful comments and support.

References

[1] N. Bertschinger and T. Natschläger, "Real-time computation at the edge of chaos in recurrent neural networks," *Neural Computation*, vol. 16, no. 7, pp. 1413–1436, 2004.

[2] T. Natschläger, N. Bertschinger, and R. A. Legenstein, "At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 145–152.

[3] B. Derrida and Y. Pomeau, "Random networks of automata: A simple annealed approximation," *Europhysics Letters (EPL)*, vol. 1, no. 2, pp. 45–49, 1986.

[4] G. Buzsáki, *Rhythms of the Brain*. Oxford, UK: Oxford University Press, 2006.

[5] W. J. Freeman, "Neurodynamics: an exploration in mesoscopic brain dynamics," in *Perspectives in Neural Computing*, J. G. Taylor, Ed. London; New York: Springer, 2000, pp. 1–24.

[6] C. van Vreeswijk and H. Sompolinsky, "Chaos in neuronal networks with balanced excitatory and inhibitory activity," *Science*, vol. 274, pp. 1724–1726, 1996.

[7] A. Babloyantz, J. Salazar, and C. Nicolis, "Evidence of chaotic dynamics of brain activity during the sleep cycle," *Physics Letters A*, vol. 111, no. 3, pp. 152–156, 1985.

[8] O. A. Rosso, S. Blanco, J. Yordanova, V. Kolev, A. Figliola, M. Schürmann, and E. Basar, "Wavelet entropy: a new tool for analysis of short duration brain electrical signals," *Journal of Neuroscience Methods*, vol. 105, no. 1, pp. 65–75, 2001.

[9] M. Adachi and K. Aihara, "Associative dynamics in a chaotic neural network," *Neural Networks*, vol. 10, no. 1, pp. 83–98, 1997.

[10] G. Tononi, G. M. Edelman, and O. Sporns, "Complexity and coherency: integrating information in the brain," *Trends in Cognitive Science*, vol. 2, no. 11, 1998.

[11] C. Haldeman and J. M. Beggs, "Critical branching captures activity in living neural networks and maximizes the number of metastable states," *Physical Review Letters*, vol. 94, no. 5, p. 058101, 2005.

[12] J. M. Beggs and D. Plenz, "Neuronal avalanches in neocortical circuits," *The Journal of Neuroscience*, vol. 23, no. 35, pp. 11 167–11 177, 2003.

[13] ——, "Neuronal avalanches are diverse and precise activity patterns that are stable for many hours in cortical slice cultures," *The Journal of Neuroscience*, vol. 24, no. 22, pp. 5216–5229, 2004.

[14] T. Montez, S.-S. Poil, B. F. Jones, I. Manshanden, J. P. A. Verbunt, B. W. van Dijk, A. B. Brussaard, A. van Ooyen, C. J. Stam, P. Scheltens, and K. Linkenkaer-Hansen, "Altered temporal correlations in parietal alpha and prefrontal theta oscillations in early-stage alzheimer disease," *Proceedings of the National Academy of Sciences, USA*, vol. 106, no. 5, pp. 1614–1619, 2009.

[15] D. Hsu and J. M. Beggs, "Neuronal avalanches and criticality: A dynamical model for homeostasis," *Neurocomputing*, vol. 69, pp. 1134–1136, 2006.

[16] A. Levina, U. Ernst, and J. M. Hermann, "Criticality of avalanche dynamics in adaptive recurrent networks," *Neurocomputing*, vol. 70, no. 10-12, pp. 1877–1881, 2007.

[17] C. G. Langton, "Computation at the edge of chaos: Phase-transitions and emergent computation," *Physica D*, vol. 42, pp. 12–37, 1990.

[18] R. A. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural microcircuit models," *Neural Networks*, vol. 20, no. 3, pp. 323–334, 2007.

[19] T. Natschläger and W. Maass, "Dynamics of information and emergent computation in generic neural microcircuit models," *Neural Networks*, vol. 18, pp. 1301–1308, 2005.

[20] 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, 2002.

[21] H. Jaeger, "Short term memory in echo state networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 152, 2002.

[22] J. M. Beggs, "The criticality hypothesis: How local cortical networks might optimize information processing," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 366, pp. 329–344, 2008.

[23] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," *Nature Reviews Neuroscience*, vol. 5, no. 2, pp. 97–107, 2004.

[24] J. Burrone and V. N. Murthy, "Synaptic gain control and homeostasis," *Current Opinion in Neurobiology*, vol. 13, no. 5, pp. 560–567, 2003.

[25] G. G. Turrigiano and S. B. Nelson, "Hebb and homeostasis in neuronal plasticity," *Current Opinion in Neurobiology*, vol. 10, no. 3, pp. 358–364, 2000.

[26] K. Linkenkaer-Hansen, V. V. Nikouline, J. M. Palva, K. Kaila, and R. J. Ilmoniemi, "Stimulus-induced change in long-range temporal correlations and scaling behaviour of sensorimotor oscillations," *The Journal of Neuroscience*, vol. 19, no. 1, pp. 203–218, 2004.

[27] K. Linkenkaer-Hansen, V. V. Nikouline, J. M. Palva, and R. J. Ilmoniemi, "Long-range temporal correlations and scaling behavior in human brain oscillations," *The Journal of Neuroscience*, vol. 21, no. 4, pp. 1370–1377, 2001.

[28] K. Linkenkaer-Hansen, D. J. A. Smit, A. Barkil, T. E. M. van Beijsterveldt, A. B. Brussaard, D. I. Boomsma, A. van Ooyen, and E. J. C. de Geus, "Genetic contributions to long-range temporal correlations in ongoing oscillations," *The Journal of Neuroscience*, vol. 27, no. 50, pp. 13 882–13 889, 2007.

[29] C. Bèdard, H. Kröger, and A. Destexhe, "Does the 1/f frequency scaling of brain signals reflect self-organized critical states?" *Physical Review Letters*, vol. 97, no. 11, p. 118102, 2006.