Spatiotemporal Pattern Recognition via Liquid State Machines

Eric Goodman, *Sandia National Laboratories*, and Dan Ventura, *Brigham Young University*

***Abstract*— The applicability of complex networks of spiking neurons as a general purpose machine learning technique remains open. Building on previous work using macroscopic exploration of the parameter space of an (artificial) neural microcircuit, we investigate the possibility of using a liquid state machine to solve two real-world problems: stockpile surveillance signal alignment and spoken phoneme recognition.**

I. Introduction

From a theoretical perspective, spiking neurons have been shown to be more computationally efficient than perceptrons or sigmoid units [1] [2] [3]. Also, some initial work in attempting to realize this computational power for real-world learning tasks has been done [4] [5] [6] [7]. However, the question of exactly how to extract these beneficial properties remains open. Here, we explore the application of liquid state machines (LSMs) [8] [9] [10] [11] [12] to spatiotemporal pattern recognition.

Given a signal $x : \mathcal{T} \rightarrow \mathcal{R}^n$, a function of time [1], a *spatiotemporal pattern*, α, is a tuple $(\alpha_s, \alpha_e, \{x(t)\}|_{\alpha_s}^{\alpha_e})$ where

- $\alpha_s \in \mathcal{T}$ is start time of the pattern
- $\alpha_e \in \mathcal{T}$ is the end time of the pattern
- and $\{x(t)\}|_{\alpha_s}^{\alpha_e}$ is the signal x between times α_s and α_e.

Signal x is said to *contain* pattern α.

It is also convenient to talk about a *class of patterns*, which is defined as a set of spatiotemporal patterns. A *recognizer*, $g : \mathcal{X} \rightarrow \{\mathcal{T} \times \mathcal{T} \times \mathcal{C}\}$, is a function that takes as input signals $x \in \mathcal{X}$ and returns sets of tuples of the form (t_s, t_e, c), where $t_s \in \mathcal{T}$ is the start time of a pattern, $t_e \in \mathcal{T}$ is its end time, and $c \in \mathcal{C}$ is the class of the pattern. A recognizer g is said to *recognize* α belonging to class c and contained in signal x if $(\alpha_s, \alpha_e, c) \in g(x)$. Sometimes, we will be interested a specific member of the tuple(s) in $g(x)$, which we will denote $g_i(x), 1 \leq i \leq 3$. We are interested in constructing LSM-based recognizers.

II. Liquid State Machines

LSMs are composed of two basic parts, a liquid and a readout function. To understand the basic idea behind LSMs, imagine a pool of water into which various objects are dropped [11]. As the objects enter the liquid, they perturb its surface, producing complex patterns, encoding both temporal and spatial information about the object. The readout function transforms this information into a useful form, e.g into a classification.

The "liquid" we use in this paper attempts to model the complex behavior of the brain with a recurrently-connected spiking neural network [13], or neural microcircuit, defined as

- a finite set V of *spiking neurons*,
- a set $E \subseteq V \times V$ of *synapses*,
- a *weight* $w_{u,v} \in \mathcal{R}$, *delay* $d_{u,v} \geq 0$, and a *response function* $\gamma_{u,v} : \mathcal{R}^+ \rightarrow \mathcal{R}$ for each synapse $\langle u, v \rangle \in E$,
- and a *threshold function* $\Theta_v : \mathcal{R}^+ \rightarrow \mathcal{R}^+$ for each neuron $v \in V$.

For the model we use, synapses are asymmetric, meaning that if a synapse ξ connects neuron α with neuron β, ξ does not connect β to α; β can receive a spike from α via ξ, but ξ does not enable spikes to reach α from β. An *excitatory synapse* is one that has $w_{u,v} \geq 0$. An *inhibitory synapse* is one that has $w_{u,v} < 0$. An *excitatory neuron* has only excitatory outgoing synapses. An *inhibitory neuron* has only inhibitory outgoing synapses. All neurons we use will either be excitatory or inhibitory. Also, conforming to biologically plausible values, the spiking networks used here are composed of 80% excitatory neurons and 20% inhibitory neurons.

As stated previously, unlike many artificial neuron models in use today, (e.g. perceptrons and sigmoidal units), the neurons in a neural microcircuit actually model the spiking behavior of real biological neurons. A spiking neuron can be thought of as an electrical circuit with a resistor and a capacitor (see Figure 1). Current enters the circuit through

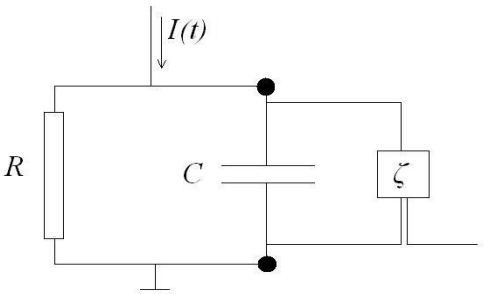


Fig. 1. Leaky Integrate-and-fire Neuron - The neuron receives input current in the form of a time varying signal $I(t)$ (spikes from incoming synapses). The resistor R constantly leaks current present in the neuron. C is a capacitor and if the voltage across the capacitance ever exceeds the threshold ζ, the neuron fires and a spike is emitted. The general diagram idea is taken from [14].

Eric Goodman is with the Information Solutions and Services of Sandia National Laboratories, Albuquerque, NM 87123, USA (email: elgoodm@sandia.gov).

Dan Ventura is with the Department of Computer Science, Brigham Young University, Provo, UT 84602, USA (email: ventura@cs.byu.edu).

[1] $\mathcal{T}$ is defined as the set of real numbers, $\mathcal{R}$, with some associated time unit, t_u. For example, the number $1 \in \mathcal{T}$ with a t_u of seconds would indicate the time 1 second and $x(1)$ would be the signal x evaluated at time = 1 second.

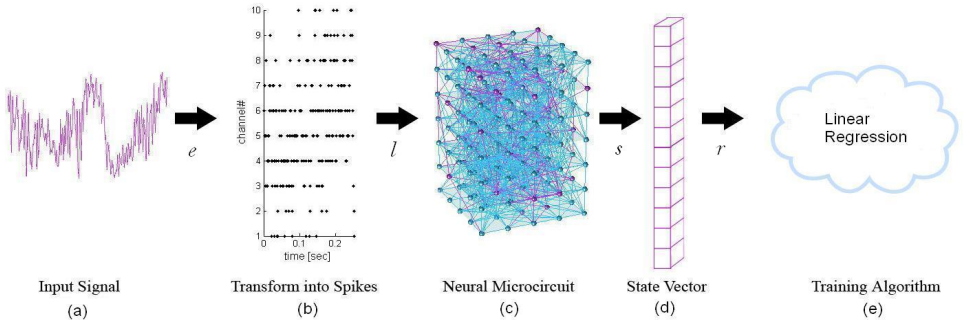


Fig. 2. Training a Liquid State Machine - An input signal (a) is transformed into spike trains via an encoding process (b). The spikes then stimulate the liquid [neural microcircuit] (c). At regular intervals, the state of the liquid is transformed into a multi-dimensional state vector (d). From the sequence of state vectors, a training algorithm [readout function] can be employed to classify the input data.

$I(t)$, which then slowly leaks away because of the resistor R. However, if ever the current in the circuit exceeds the threshold ζ, a spike is released. This type of neuron model is known as a *leaky integrate-and-fire* neuron and is what we use in this paper. For more information on spiking neurons and networks, see [14].

Network dynamics are affected by a large number of parameters, including the weight and delay values, the connection topology, the time constant associated with the response function, etc. Values for these parameters are often determined by drawing randomly from some governing distribution, and earlier work investigated the affect of these population statistics on network performance in pattern recognition tasks [15], [16]. The modeling software we use to simulate the spiking neural network comes from [12], where the default network parameters are based on empirical results gathered from recordings of the somatosensory cortex in rats [17] [18], and, unless otherwise stated, we use these default parameters.

The LSM performs pattern recognition as follows. The signal x is first encoded as spike trains with some function $e : \mathcal{T} \times \mathcal{R}^n \rightarrow \mathcal{T} \times \mathcal{R}^m$ so that it will interact with neurons in the circuit. This encoded signal is then transformed into another signal with a function $l : \mathcal{T} \times \mathcal{R}^m \rightarrow \mathcal{T} \times \mathcal{R}^p$ that encapsulates the dynamics of the liquid. Also, to enable the use of a wide variety of training algorithms which can not directly use spikes, samples of the state of the liquid are taken and form a sequence of vectors, called *state vectors*, which can then be used to train a readout function. This sampling process will be denoted by $s : \mathcal{T} \times \mathcal{R}^p \rightarrow (\mathcal{R}^p_k)$, a function that transforms a signal into sequences of state vectors. Finally, the readout function $r : (\mathcal{R}^p_k) \rightarrow (\{0, 1, ..., N\}_k)$, can be trained using these state vectors to represent the inputs. Therefore, an LSM-based recognizer is a functional composition, $g(x) = a(r(s(l(e(x)))))$, with $a()$ being some post-processing to determine timing (e.g. Algorithm 1 in Section IV). Figure 2 displays graphically how an LSM works.

III. Advantages of LSMs

One advantage of using a spiking neural network is that it projects the input into a high-dimensional space, allowing the learned readout function to be simple. Of course this advantage of projecting inputs into higher-dimensional spaces is common to many learning methods, such as the kernel of a support vector machine.

Another advantage of using an LSM is the ability to have a memory-less readout function. Any snapshot of the state of the network will contain information about both current and past inputs; the waves of spikes produced by input in the past will continue to propagate for some time, intermingling with the waves from the current input. This process will be referred to as *integration of inputs over time*. When a network properly integrates inputs over time, a readout function can be memory-less, relying on the network to remember and represent past and current inputs simultaneously.

Integration of inputs over time allows patterns with extent in time to be identified. For example, to recognize the entire word (a nonsense word from the film "Mary Poppins"), "supercalifragilisticexpialidocious", one must still remember that "super" had been said by the time "docious" is enunciated; proper integration of inputs, in this case the syllables of the word, is vital to the recognition of the entire word.

Figure 3(a) gives an example when integration over time does not occur. Input spikes create clusters of activity within the network, all of which die out by the time the last spike of the stimulus occurs. Thus, it would be practically impossible to recognize the entire sequence of spikes from snapshots of the circuit; the neural microcircuit is unable to "remember" previous inputs because the network parameters are not set correctly.

In other words, imagine that each spike represents some segment of the Mary Poppins nonsense word, e.g. the first spike somehow meaning "super" and the last spike representing "docious." Since spike activity in the liquid dies out after each input, the neural microcircuit is unable to remember that "super" was ever said, and it would be impossible for a readout function to learn and recognize the word in its

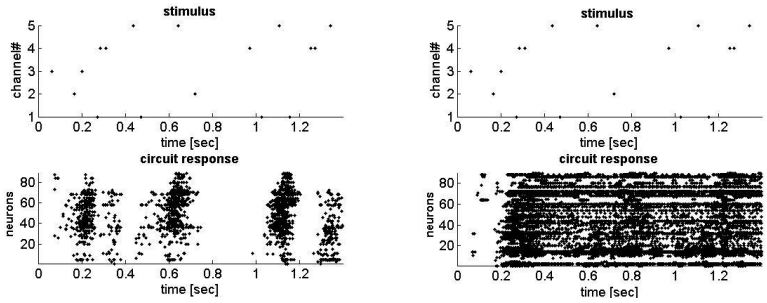


Fig. 3. A stimulus encoded by five neurons is presented to two different circuits of size 90 neurons. The black dots represent when a particular neuron has fired. The circuits are identical except for differing delay times and time constants. The first circuit experiences temporal stratification. The second circuit behaves quite differently – the resultant activity from each of the input spikes blends together.

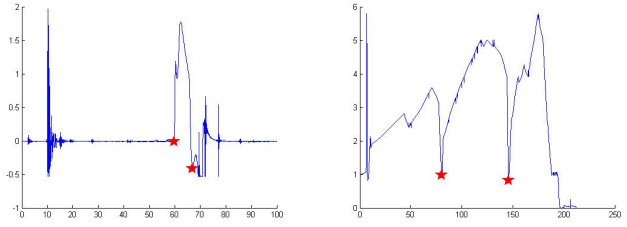
entirety.

A more desirable example is that of Figure 3(b). The same input spike train is fed to a neural microcircuit, however in this case the neural microcircuit has appropriately set network parameters that allow the input spikes to create a series of reactions within the recurrent network which interact over time. Thus any snapshot of the circuit could potentially contain information about inputs that occurred some time in the past.

This paper explores how the benefits of LSMs, integration of inputs over time and projection into higher dimensional spaces, can be used to solve practical problems.

IV. Stockpile Surveillance Data Alignment

Stockpile surveillance data consists of one dimensional signals collected from non-nuclear tests of the nuclear stockpile. Our task is to identify the initial boundary of the "interesting" part of the signal. Formally, given a class of signals X_i, each signal $x \in X_i$, contains a spatiotemporal pattern $\alpha = (\alpha_s, \alpha_e, \{x(t)|_{\alpha_s}^{\alpha_e}\})$, and our task is to identify the pattern's start time, α_s; that is, we would like to construct a recognizer g such that $g_1(x) = \alpha_s$. The data set contains six classes of signals, X_1, X_2, ..., X_6, with X_1 through X_4 containing about 30 example signals each, and X_5 and X_6 containing 45 example signals (see Figure 4). We show that LSMs can solve this problem robustly with very few training examples.



(a) X_2 (b) X_6

Fig. 4. Example stockpile surveillance signals. Each signal is prototypical of the signals in its respective class. Stars indicate the boundaries of the target pattern.

Our encoding function, e, is a simple spatial encoding in which a number (n_{in}) of input neurons represent the signal over time. Each of the input neurons is assigned to cover a unique portion of the range, ρ_j, of the signal:

$$\rho_j = \begin{cases} [\frac{(j-1)(\Omega-\omega)}{n_{in}} + \omega, \frac{j(\Omega-\omega)}{n_{in}} + \omega) & 1 \leq j < n_{in} \\ [\frac{(j-1)(\Omega-\omega)}{n_{in}} + \omega, \Omega] & j = n_{in} \end{cases} \quad (1)$$

where $\Omega = \max_{x \in X_i, t \in domain(x)} x(t)$ and $\omega = \min_{x \in X_i, t \in domain(x)} x(t)$. Then for each time t having signal value $x(t)$, the input neuron j that has $x(t) \in \rho_j$ will fire at time t.

For each class X_i, we train a readout function r_i using a simple linear least-squares regression model on $X_{train} \subset X_i$. The set $X_i - X_{train} = X_{val}$, is referred to as the validation set. Each $x \in X_{train}$ is translated into spikes via Equation 1) and fed into the neural microcircuit from which a sequence of state vectors, $(o_k = s(l(e(x)))_k)$, are obtained. Each state vector is assigned an output value, $r_i(o_k) \in \{0, 1\}$, where a 0 signifies the state vector does not represent the target pattern and a 1 signifies otherwise.

Algorithm 1 is used to post-process the series of output values to compute $g_1(x)$. Intuitively, the algorithm slides a window across the sequence of output values looking the longest sequence of (mostly) 1s. Values for the algorithm's window size, $\delta = 8$, and threshold, $\vartheta = 0.5$, were determined empirically. Also, since as a general rule the LSM tends to be late in its prediction, as a final step we offset our prediction for $g_1(x)$ by the median error (on the training set): $g_1(x) = g_1(x) - median(\epsilon)$, with ϵ calculated as

$$\epsilon = \frac{|g_1(x) - \alpha_{s_x}|}{\alpha_{e_x} - \alpha_{s_x}} \quad (2)$$

Finally, we use a square column network topology of $3 \times 3 \times 15$ neurons. The state vectors are composed of one element for each neuron in the circuit, and they are sampled every 0.01 seconds. The number of input neurons, n_{in}, is set to 10. Unusual network parameter settings include eliminating the recurrent connections, and using a mean synapse delay time of 0.1 seconds with an associated standard deviation of 0.01 seconds.

Algorithm 1 Finding the Pattern Start Point

GIVENS:
A sequence of state vectors, $(o_1, \ldots, o_n)$
Window size δ
Threshold ϑ
$\xi : \mathcal{Z} \rightarrow \mathcal{T}$ gives the time that the k^{th} term, o_k, occurred, and $\xi(NULL) = \infty$
ALGORITHM:
$index_{max} = NULL$
$current\_max = -\infty$
for $i = 0$ to n **do**
$index_s = \underset{i \leq j \leq n-\delta}{\arg\text{ first}} \left(\sum_{k=j}^{j+\delta} \frac{r(o_k)}{\delta} > \vartheta \right)$
$index_e = \underset{t_s < j \leq n-\delta}{\arg\text{ first}} \left(\sum_{k=j}^{j+\delta} \frac{r(o_k)}{\delta} < \vartheta \right)$
if $\sum_{k=index_s}^{index_e} r(o_k) > current\_max$ **then**
$current\_max = \sum_{k=index_s}^{index_e} r(o_k)$
$index_{max} = index_s$
end if
end for
return $\xi(index_{max})$

Since the raw surveillance data possesses a wide variety of signals, we normalize them so that they are all treated equally by an LSM. The length of each $x \in X_i$ is equalized to 3.5 seconds, and each signal is resampled at a rate of 100 samples/second.

Also, for X_3, the target pattern is so short relative to the length of the entire signal that the duration of the target pattern is too short for the LSM to process properly. The solution is for the user do some rough cropping of non-interesting portions of the signal (in this case, 70% from the beginning and 20% from the end) and allow the LSM to do the fine-scale work. The other set, X_4, is also problematic because the end of the signal is very similar to the target pattern. In fact, the last portion may in fact be another instance of the target pattern; however, we have made the assumption that the user is only interested in one target pattern per signal. The confusion can be solved this time by cropping 20% from the end of the signal.

A. Results

Figure 5 displays the mean error (Equation 2) on the validation set for an LSM for the stockpile surveillance dataset for varying sizes of training sets. Typically, at least four training examples are needed to achieve fairly good accuracy. After that, the gain in accuracy for each additional training example is nominal.

Table I compares the mean error obtained (on the validation set) using LSMs with the results obtained via a commonly used analytical method, cross correlation:

$$g_1(x) = \arg\max_{\varsigma} \left\{ \frac{\sum_t [(x(t) - \mu_x)(v(t-\varsigma) - \mu_v)]}{\sqrt{\sum_t (x(t) - \mu_x)^2} \sqrt{\sum_t (v(t-\varsigma) - \mu_v)^2}} \right\}$$

where ς is the delay, μ_x is the average of signal x, v is a prototype for the appropriate signal class, and μ_v is the average of signal v. Intuitively, one can imagine taking v and sliding it across u, where at each shift, the sum of the product of the signals is computed. The value of ς that gives the largest value is when the signals are most highly correlated.

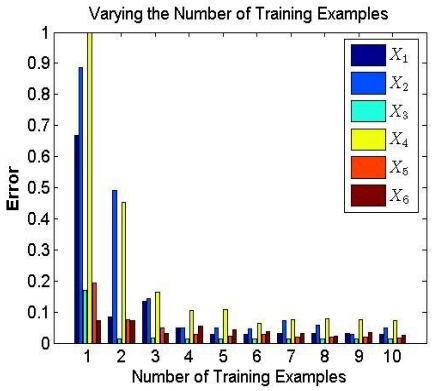


Fig. 5. Amount of training data vs. error for an LSM on the stockpile surveillance dataset.

TABLE I
MEAN VALIDATION SET ERROR OF LSMS AND CROSS CORRELATION FOR THE STOCKPILE SURVEILLANCE DATASET. RESULTS ARE OVER 10 RANDOM TRIALS USING 5 TRAINING INSTANCES EACH TRIAL.

Class	LSM	Standard Deviation	Cross Correlation	Standard Deviation
X_1	0.028	0.022	0.0042	0.0039
X_2	0.048	0.075	0.0047	0.0017
X_3	0.014	0.012	0.0141	0.0095
X_4	0.109	0.281	0.0056	0.0040
X_5	0.023	0.074	0.3153	0.1302
X_6	0.041	0.072	0.1661	0.0038
All	0.044	0.034	0.0850	0.1295

LSMs are able to robustly handle a variety of different signals, with an average error of 4.4%. While cross correlation performs better than LSMs on three of the six cases, its results are substandard on two sets. Therefore, at least for these data, LSMs appear to be the more robust method as reflected in the standard deviation in the accuracies across classes (0.034 for LSMs vs. 0.1295 for cross correlation). Further testing needs to be done to ensure that LSMs may be applied broadly to all types of signals and that our pre- and post-processing steps are not over-fitting the six signal classes represented in the dataset.

V. SPOKEN PHONEME RECOGNITION

We use the the TIMIT speech corpus [19], which consists of 6300 sentences (10 unique sentences, 630 unique speakers). However, 1260 of the sentences (according to the corpus documentation, the SA sentences) are "shibboleth" sentences used to distinguish between dialects and are not used in here, leaving a total of 4040 sentences (SX and SI sentences). Following the convention of previous papers [20] [21] [22], we reduce the 61 phonetic labels to a subset of 39, folding several phonemes classes together. Given a speech signal x, our task is to label each frame of that signal

with the appropriate phonetic class. Thus, the goal is to have $g_3(x_{frame}) = c$, where c is the correct phonemic label.

As is common in speech recognition tasks, we use the standard 13 Mel frequency cepstral coefficients (mfccs) as input features. To convert the speech signals into spikes, every 10ms we calculate mfccs for a frame size of 16ms. We also calculate first and second derivatives of the mfccs, for a total of 39 input features, each of which has a single spiking neuron representing it with a rate-based encoding:

$$Rate_i(t) = \frac{mfcc_i(t)}{(\Omega_i - \omega_i)} \cdot MaxRate$$

where Ω_i is the largest i^{th} mfcc (Δmfcc, $\Delta\Delta$mfcc), ω_i is the smallest i^{th} mfcc (Δmfcc, $\Delta\Delta$mfcc), and where the maximum rate is set to 200 Hz.

For this application the network had a topology of 6x6x25, and parameter modifications include scaling the input connection probability by 0.1, the recurrent connection probability by 0.5, and the recurrent connection weights by 0.12. The time constant was set to 0.003 and the mean delay was drawn from a uniform distribution between 0.001 and 0.01.

Since linear least-squares regression requires the inversion of a matrix whose size is proportional to the number of examples, application of this training algorithm is infeasible for such a large corpus of data. To ameliorate this problem, a combination of m models trained on m distinct subsets of the corpus, is used instead, with classification determined by majority vote (we tried values of m between 3 and 20 with best results at $m = 10$). Also, for comparison we include the results for single-layer (one per phoneme, winner-take-all) and multi-layer perceptrons (topology: 900x1800x39).

Table II summarizes the results on a validation set of 25 sentences for each of the different approaches. Reported accuracy is the frame-by-frame accuracy, i.e. the number of frames correctly classified divided by the total number of frames. For comparison, frame-by-frame accuracies reported on TIMIT in the literature for three other techniques are also shown.

Comparing the results from the best LSM (51.25%) to the best accuracy reported in the literature (74.20%) indicates that more work needs to be done. As all of the readout functions behave poorly, the deficiency may lie not with the approximative ability of the readout functions but instead with the separation ability of the liquid.

A. Addressing Separation

Given a set of state vectors, $O = \{o_1, o_2, ..., o_n\}$, N output classes, and target values $T = \{r_t(o_1), r_t(o_2), ..., r_t(o_n)\}$, where r_t gives the correct output class for each o_i, we divide O into N distinct subsets, $O_1, O_2, ...O_N$, where $\forall i, j, o_j \in O_i \iff r_t(o_j) = i$. For each of these N subsets, we calculate the center of mass:

$$C_m(O_i) = \frac{\sum_{o_j \in O_i} o_j}{|O_i|}$$

Thus $C_m(O_i)$ is a vector that gives the location of the center of mass for output class i. We propose a separation measure, $Sep(\Psi, O)$, for a given circuit Ψ and set of state vectors O:

$$Sep(\Psi, O) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{||C_m(O_i) - C_m(O_j)||_2}{N^2} \qquad (3)$$

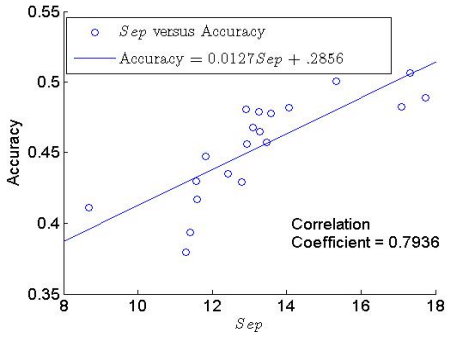
Intuitively, Sep can be defined as taking the mean distance from each O_i to each O_j, resulting in N means, and then taking the mean of those N means.

Using Equation 3, we calculate the separation for two different sets of sentences, the validation set and another randomly selected set of 25 sentences, for 20 different randomly generated LSMs with multi-layer perceptron models as the readout function. Figure 6 shows that the separation values are positively correlated with accuracy; over the validation set, the 20 data points have a correlation coefficient of 0.7936, and for the other set, the correlation is 0.7065.

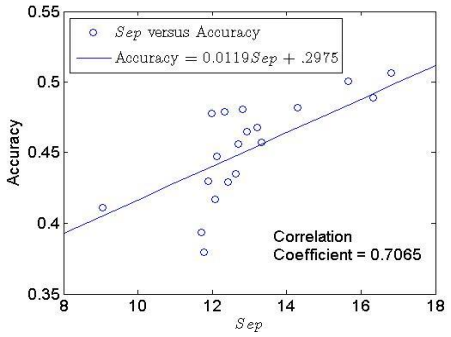
TABLE II

FRAME-BY-FRAME ACCURACY ON THE TIMIT CORPUS FOR LSMS WITH THREE DIFFERENT READOUT FUNCTIONS AND FOR THREE OTHER TECHNIQUES REPORTED IN THE LITERATURE.

Method	Accuracy(%)
LSM with Single-layer Perceptron readout function	39.06
LSM with m Model Regression readout function	47.84
Hidden Markov Model with ICA [21]	50.89
LSM with Multi-layer Perceptron readout function	51.25
Hidden Markov Model with mfccs [20]	52.70
Time-delayed recurrent neural network [22]	74.20



(a) Validation Set



(b) Another Randomly Chosen Set

Fig. 6. The ability of Sep to predict accuracy

VI. Comments

We have shown that LSMs are a robust technique for identifying the boundaries of spatiotemporal patterns in stockpile surveillance data. We have also shown that LSMs have potential for solving difficult temporal pattern classification problems such as occur in a continuous speech phoneme recognition task. We have also proposed a measure of the liquid's ability to separate inputs which positively correlates with accuracy.

Several avenues for future work exist, including the investigation of alternative input encoding schemes and the development of a readout function that directly incorporates the timing of spike activity (thus alleviating the need for state vectors and the sampling function s). However, perhaps the most pressing direction for future research involves the development of a robost training algorithm for the liquid parameters. Here we have focused on manipulating macroscopic network parameters (e.g. the mean synaptic delay time and the recurrent connection probability) rather than directly manipulating the parameter settings of individual neurons and synapses. This high-level approach is useful for understanding general principles but inevitably sacrifices some of the representational power of the spiking network. A first natural approach to manipulating individual network parameters might be an evolutionary exploration of the parameter space, perhaps using the Sep metric as a fitness function. However, the ideal way to set individual neuronal and network properties would be with a self-organizing spiking network, perhaps driven through a reinforcement scheme that rewards separation of inputs for a particular problem.

Acknowledgments

We thank Sandia National Laboratories for partially funding this work and for providing the stockpile surveillance data.

References

[1] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9. Denver, CO: MIT Press, 1997, pp. 211–217.

[2] ——, "Fast sigmoidal networks via spiking neurons," *Neural Computation*, vol. 9, pp. 279–304, 1997.

[3] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *Journal of Computer and System Sciences*, vol. 69, no. 4, pp. 593–616, 2004.

[4] S. Bohte, H. L. Poutré, and J. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and muli-layer rbf networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 426–435, 2001.

[5] T. Natschläger and B. Ruf, "Spatial and temporal pattern analysis via spiking neurons," *Network: Computation in Neural Systems*, vol. 9, no. 3, pp. 319–332, 1998.

[6] S. Bohte, J. Kok, and H. L. Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, pp. 17–37, 2002.

[7] A. Belatreche, L. Maquire, M. McGinnity, and Q. Wu, "An evolutionary strategy for supervised training of biologically plausible neural networks," in *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*, September 2003, pp. 1524–1527.

[8] N. Bertschinger and T. Natschläger, "Real-time computation at the edge of chaos in recurrent neural networks," *Neural Computation*, vol. 16, pp. 1413–1436, 2004.

[9] S. Häusler, H. Markram, and W. Maass, "Perspectives of the high dimensional dynamics of neural microcircuits from the point of view of low dimensional readouts," *Complexity (Special Issue on Complex Adaptive Systems)*, vol. 8, no. 4, pp. 39–50, 2003.

[10] 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.

[11] T. Natschläger, W. Maass, and H. Markram, "The "liquid computer": a novel strategy for real-time computing on time series," *Special Issue on Foundations of Information Processing of TELEMATIK*, vol. 8, no. 1, pp. 39–43, 2002.

[12] T. Natschläger, "Neural micro circuits," http://www.lsm.tugraz.at/index.html, 2005.

[13] W. Maass, "On the complexity of networks of spiking neurons," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 7. Denver, CO: MIT Press, November 1995, pp. 183–190.

[14] W. Gerstner and W. Kister, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[15] E. Goodman and D. Ventura, "Time invariance and liquid state machines," in *Proceedings of 8th Joint Conference on Information Sciences*, Salt Lake City, UT, July 2005.

[16] ——, "Effectively using recurrently-connected spiking neural networks," in *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Canada, August 2005.

[17] Y. W. A. Gupta and H. Markram, "Organizing principles for a diversity of gabaergic interneurons and synapses in the neocortex," *Science*, vol. 287, pp. 273–278, 2000.

[18] H. Markram, Y. Wang, and M. Tsodyks, "Differential signaling via the same axon of neocortical pyramidal neurons," *Proceedings of the National Academy of Sciences*, vol. 95, pp. 5323–5328, 1998.

[19] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1, 1993, texas Instruments, Inc.

[20] S. Young, "The general use of tying in phoneme-based hmm speech recognisers," in *Proceedings of the International Conference on Acoustics, Speech, Signal Processing*, San Francisco, CA, March 1992, pp. 1569–1572.

[21] O.-W. Kwon and T.-W. Lee, "Phoneme recognition using ica-based feature extraction and transformation," *Signal Process.*, vol. 84, no. 6, pp. 1005–1019, 2004.

[22] R. Chen and L. Jamieson, "Experiments on the implementation of recurrent neural networks for speech phone recognition," in *Proceedings of the Thirtieth Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 1996, pp. 779–782.