Effectively Using Recurrently-Connected Spiking Neural Networks

Eric Goodman
ya_krutoi@hotmail.com

Dan A. Ventura
ventura@cs.byu.edu

Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 200

Effectively Using Recurrently-Connected Spiking Neural Networks

Eric Goodman
Computer Science Department
Brigham Young University
Provo, UT 84602
E-mail: eric_goodman@byu.edu

Dan Ventura
Computer Science Department
Brigham Young University
Provo, UT 84602
E-mail: ventura@cs.byu.edu

Abstract— **Recurrently-connected spiking neural networks are difficult to use and understand because of the complex non-linear dynamics of the system. Through empirical studies of spiking networks, we deduce several principles which are critical to success. Network parameters such as synaptic time delays and time constants and the connection probabilities can be adjusted to have a significant impact on accuracy. We show how to adjust these parameters to fit the type of problem.**

I. Introduction

A number of artificial neural models have been developed in an attempt to emulate the characteristics of the brain that allow learning. Much research has focused on two general models, perceptrons and sigmoidal units. While these two models have been shown to be useful and applicable to a wide range of problems, recent research has revealed several reasons, theoretical and biological, that compel an investigation into a more complex representation, one that actually models the spiking behavior of biological neurons.

Unlike perceptrons and sigmoidal units, biological neurons communicate and convey information via electrical pulses, commonly called spikes. For instance, the speed at which a muscle contracts is proportional to the rate at which neurons within the muscle produce spikes, or fire [8]. This type of encoding is known as rate-based, signifying that the firing rate of the neuron communicates information.

The assumption of a rate-based encoding allows perceptrons and sigmoidal units to abstract away the spiking behavior of biological neurons. However, with the extra degrees of freedom available to spiking neurons via a host of additional parameters, they have much more flexibility and representational power than their rate-based cousins. Additionally, spiking neurons have an inherent advantage when learning time-varying functions since they are by definition a temporal phenomenon. However, despite these apparent advantages, the complex non-linear dynamics of recurrently-connected spiking networks defies attempts at analytical study and comprehension; as a result, no general method exists that efficiently uses the full capability of spiking neurons.

Through empirical studies of spiking networks, we provide results that prove insightful, allowing more efficient use of the representational power of spiking neurons. In particular, we examine four network parameters: the mean synaptic delay, $\overline{d}$, the standard deviation of the synaptic delay, d_σ, the mean synaptic time constant, τ, and the connection probability between input neurons and network neurons, c_{prob}.

This paper explores spiking networks within a paradigm known as liquid state machines (LSMs) [5] [7] [6], described in more detail in section II and III. An artificial problem, described in section IV, is developed in order to elucidate important principles necessary for success using a spiking neural network. Section V presents the results. Section VI then wraps up with some conclusions and ideas for future work.

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 [7], where each object belongs to certain output class in the set $\{S_0, S_1, ..., S_N\}$. As the objects plunge into the liquid, they perturb the surface of the liquid, resulting in complex patterns. These patterns provide a history and describe both temporally and spatially how the objects entered the liquid. Stated another way, we have a signal $x : \mathcal{T} \rightarrow \mathcal{R}^n$, a function of time, which is transformed into another signal with a function $l : \mathcal{T} \times \mathcal{R}^n \rightarrow \mathcal{T} \times \mathcal{R}^m$ that encapsulates the dynamics of the liquid. Then a readout function $r : \mathcal{T} \times \mathcal{R}^m \rightarrow \mathcal{T} \times \{0, 1, ..., N\}$, can then be trained from the transformed signal $l(x)$ to classify the inputs. Overall, the process can be described succinctly as $r(l(x))$.

Now, instead of a pool of water, consider for a moment the human brain as a liquid. Inputs enter the brain through a variety of sources - through eyes and ears and any of the other senses. These inputs are encoded via spike trains, or in other words, series of electrical impulses which form the basis of communication between neurons. These input spikes in turn cause a cascade of spikes within the brain, producing complex interactions, analogous to the ripples and interference patterns produced in the pool of water.

The liquid we use in this paper attempts to model the complex behavior of the brain with a recurrently-connected spiking neural network, often called a neural microcircuit. Formally, a *spiking neural network* [4] consists of

- a finite set V of *spiking neurons*,
- a set $E \subseteq V \times V$ of *synapses*,

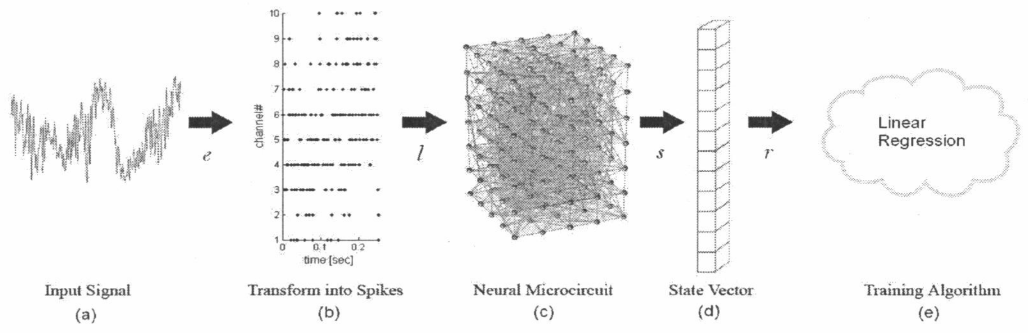

Fig. 1. **Training a Liquid State Machine** - First an input signal (a) is transformed into spikes trains via some encoding process (b) (black dots represent times when a neuron spiked). The spikes then stimulate the liquid (c), which in this case is a neural microcircuit. At regular intervals, the state of the liquid is transformed into a multi-dimensional state vector (d). From the sequence of state vectors (d), a training algorithm can be employed to classify the input data, in this case linear regression.

- a *weight* $w_{u,v} \geq 0$, *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, each synapse is directional, 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. For more information on spiking networks, see [1]. The modeling software we use to simulate the spiking neural network comes from [6], where the default network parameters are based on empirical results gathered from recordings of the somatosensory cortex in rats [2] [3].

As stated before, a readout function r is trained on the output of the liquid, $l(x)$. However, since the liquid we use is a neural microcircuit, often x must first be encoded as spike trains with some function $e : \mathcal{T} \times \mathcal{R}^n \rightarrow \mathcal{T} \times \mathcal{R}^n$ in order to interact with neurons of the circuit. 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 $s : \mathcal{T} \times \mathcal{R}^m \rightarrow \{(\mathcal{R}^m)_k\}$, a function that transforms a signal into sequences of state vectors. All together, the application of an LSM to a signal x can be described by the expression $r(s(l(e(x))))$, where $r : \{(\mathcal{R}^m)_k\} \rightarrow \{(\{0, 1, ..., N\})_k\}$, i.e. a function from sequences of state vectors to sequences of output classes. Figure 1 displays graphically how an LSM works.

In the simplified problem we study in this paper, each signal x belongs to a single output class, i.e. $\forall k, r(s(l(e(x))))_k = i$, for some $i \in \{0, 1, ..., N\}$. Thus, we simplify the readout function to be a function that takes sequences of state vectors, combines them in some fashion, and outputs class membership, i.e. $r : \{(\mathcal{R}^m)_k\} \rightarrow \{0, 1, ..., N\}$.

This paper explores several network parameters and their effect on performance. All of the parameters that we examine are related to the synapses between neurons. As stated before, each synapse has an associated delay time, $d_{u,v}$. This is the time it takes for a spike to propagate along the synapse from one neuron to the other. For all the networks we use in this paper, all of the $d_{u,v}$ are drawn from a gaussian distribution. We vary both the mean, $\overline{d}$, and the standard deviation, d_σ, of this distribution to understand their effects.

We examine both $\overline{d}$ and d_σ for each of the four following different types of synapses: (1) from an excitatory neuron to an excitatory neuron, EE, (2) from excitatory to inhibitory, EI, (3) from inhibitory to excitatory, IE, and (4) from inhibitory to inhibitory, II. The default $\overline{d}$ for each type of synapse is either 8×10^{-4} or 1.5×10^{-3} seconds. However, unless explicitly stated that the default values are being used, $\overline{d}$ will refer to a mean over all types of synapses, i.e. $\overline{d} = \overline{EE} = \overline{EI} = \overline{IE} = \overline{II}$. The default value for d_σ is $0.1\overline{d}$.

The synaptic time constant, τ, is another parameter we investigate. The time constant is related to the response function, $\gamma_{u,v}$, and governs how long a spike's influence remains. For instance, consider the simple response function:

$$\gamma(t) = I_0 * e^{\frac{-(t-t_0)}{\tau}}, t \geq t_0 \tag{1}$$

where $I_0 \in \mathcal{R}^+$ is the value the synapse attains after a spike and t_0 is the time of the most recent spike. The larger τ is, the longer the influence of a spike will last. Unlike the synaptic delay, τ is the same for each synapse of the same type. The default value for τ is either 3×10^{-3} or 6×10^{-3}. As with the mean synaptic delay, τ will be the same across all synapse types unless the default values are specified.

Finally, the fourth parameter we adjust is the connection probability, c_{prob}, between the input neurons and the neural microcircuit. The default value for c_{prob} is 0.2 for inhibitory neurons in the circuit and 0.3 for excitatory neurons. We scale these probabilities by a factor $k \in \{1, 2, 3, 4, 5\}$. When $k = 5$, the probabilities are above 1.0 and every input neuron forms a synapse with every network neuron.

Except for the above listed parameters, most of the default settings of the modeling software are used. However, all

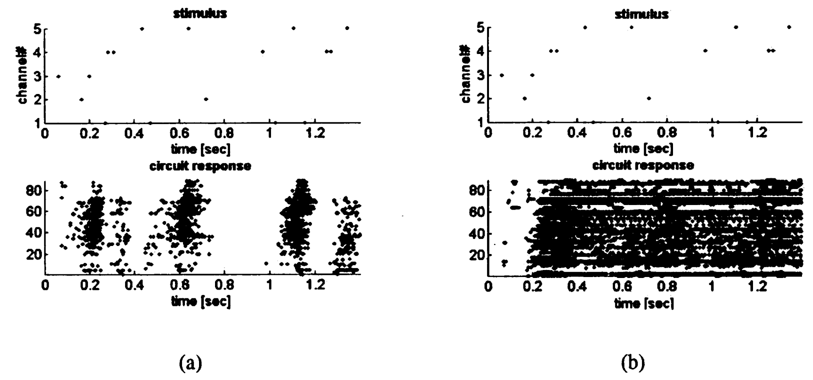

Fig. 2. A stimulus encoded by five neurons is presented to two different circuits of size 90 neurons. The black dots represent when a particular neuron has fired. The circuits are identical except for differing delay times and time constants. The first circuit experiences temporal stratification. The second circuit behaves quite differently; the resultant activity from each of the input spikes blends together.

experiments have the following parameter settings in common. Linear least-squares regression, a fairly simple algorithm limited in its representational power, is chosen for the training of the readout function so that results can be attributed more to the neural microcircuit and its ability to simplify the task of classification rather than to the power of the learning algorithm. Also, all of the experiments use a network size of 90 neurons, with the topology of the network being a square column of dimension $3 \times 3 \times 10$ neurons. This size possesses sufficient representational capability to solve most of the problems we discuss, while at the same time being small enough to exhibit some weaknesses. Recurrent connections within the circuit have an average distance of 2 (i.e. 2 neurons apart within the $3 \times 3 \times 10$ regular grid) and are governed by the default values for c_{prob} for each type of synapse. The ratio of excitatory to inhibitory neurons is 4:1. The state vectors are composed of 90 elements, one element for each neuron in the network. Each element is a rough approximation of the firing rate of the neuron at the time of the sample.

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.

Figure 2(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 before the last spike of the stimulus. 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.

A more desirable example is that of Figure 2(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 input spike activity interaction 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 to best make use of the benefits of LSMs: projection of inputs into higher dimensional spaces and integration of inputs over time.

IV. Problem Description

Datasets are constructed in the following manner. N templates are created, each representing a different output class. Each template is composed of λ input channels. The input channels are poisson-distributed spike trains, strictly monotonically increasing sequences $t_0, t_1, ..., t_n$, with a mean value of 100 spikes per second. For all experiments, the lengths of the spike trains of each input channel are set to be the same value, ζ, which implies that each $t_n \leq \zeta$. From these N templates, gaussian jitter is added to the templates to form new instances of the form $t_0 + \epsilon_0, t_1 + \epsilon_1, ..., t_n + \epsilon_n$, where each ϵ_i is a number drawn from a gaussian distribution with zero mean and standard deviation of 10^{-3} seconds. These instances are then divided up into a training set and a validation set. Also, for some experiments, random noise is inserted at the beginning of the spike trains, i.e. $t_0 + \epsilon_0, t_1 + \epsilon_1, ..., t_n + \epsilon_n$ becomes $r_0, r_1, ..., r_m, t_0 + \epsilon_0 + \delta, t_1 + \epsilon_1 + \delta, ..., t_n + \epsilon_n + \delta$, where $r_0, r_1, ..., r_m$ is a randomly generated sequence and δ is the length of that sequence. The portion $t_0 + \epsilon_0 + \delta, t_1 + \epsilon_1 + \delta, ..., t_n + \epsilon_n + \delta$ shall be identified as the *target pattern*.

The goal then is to train a readout function, r, so that given an instance η, $r(s(l(\eta))) = c(\eta)$, where $c(\eta)$ returns the output class of η. The function r combines N linear regressive models, $m_1, m_2, ..., m_N$. Each m_i is trained on the set of state vectors from the training set with target values $\{0, 1\}$, a 1 for state vectors belonging to class i, and a 0 otherwise. Then,

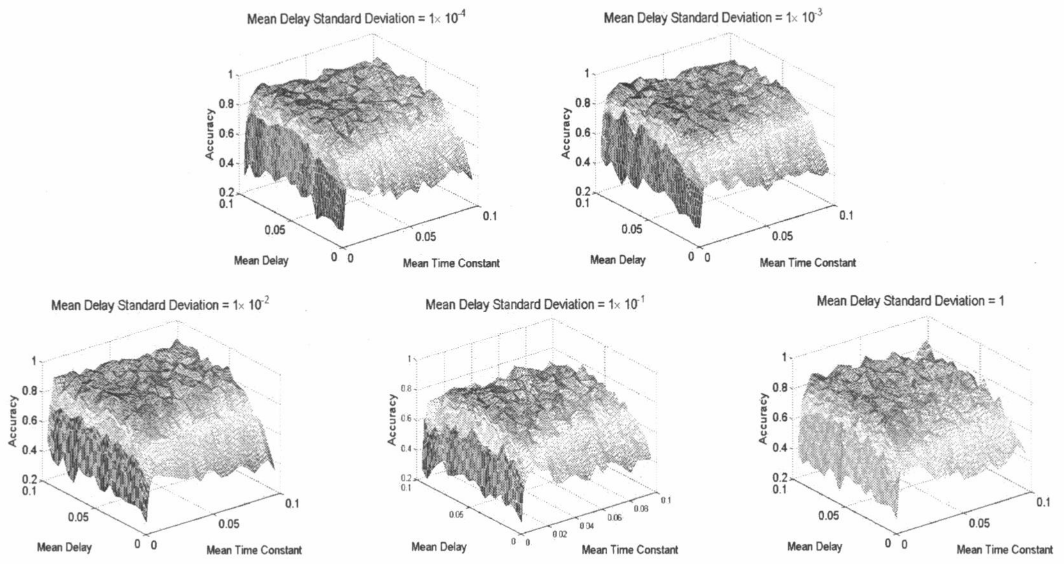

Fig. 3. Results from varying $\overline{d}$, τ, and d_σ. Each graph has d_σ held constant while $\overline{d}$ and τ are varied. Color indicates accuracy.

given a sequence of state vectors for a particular instance, $\{s(l(\eta))_k\}$, r outputs

$$\arg\max_i \frac{\sum_{j=1}^{k} m_i(s(l(\eta))_j)}{k} \tag{2}$$

V. Results

This section presents the results of a number of experiments using an LSM to solve the above task. The first experiment varies three network parameters, $\overline{d}$, τ, and d_σ, to see how they influence accuracy. The second experiment adds noise to the beginning of the target patterns to examine its effect. The third experiment again explores how noise affects performance, but also assess how well the LSM can handle large numbers of output classes. Finally, the last three experiments try to solve the noise problem by adding more information into the circuit.

For all experiments except 3, the state vectors are sampled every 0.01 seconds, starting at time

$$0.5l + \delta + \min\{\overline{EE}, \overline{EI}, \overline{IE}, \overline{II}\} \tag{3}$$

and ending before time

$$1.33l + \delta + \max\{\overline{EE}, \overline{EI}, \overline{IE}, \overline{II}\} \tag{4}$$

The rationale behind these equations is to provide data on the state of the network when the target pattern is actually occurring. The constants 0.5 and 1.33 are an indication that the circuit may take a while to respond to the target pattern's input spikes, and that the network may continue to represent the pattern past the time when input spikes are being received.

Also, for all experiments except 5, the number of input channels is set at $\lambda = 1$. For each run of the algorithm, sets of instances and neural microcircuits are uniquely generated within parameter constraints. Finally, each data point represents the mean over ten trials.

A. Experiment 1: Parameter Exploration

This first experiment reveals the effect of varying $\overline{d}$, d_σ, and τ on the target task. We set $N = 20$ and both $\overline{d}$ and τ are sampled at values in the range $[0.001, 0.100]$ while d_σ is sampled at the five discrete values $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$. The length of each template is set to $\zeta = 0.1$ seconds and the length of the noise inserted at the beginning is set to $\delta = 0$. The training set size is set to 400 and the validation set to 200. Figure 3 displays the validation results of the experiments.

From Figure 3, several trends are distinctly noticeable. For one, varying d_σ does not have a significant impact as long as d_σ is small enough. Associated with each value d_σ is a set of points in $\mathcal{R}^3$ consisting of a value for $\overline{d}$, a value for τ, and the classification accuracy achieved. Linearly interpolating from these points, sampling at each combination $(\overline{d}, \tau), \forall \overline{d}, \tau \in \{0.001, 0.002, ..., 0.100\}$, produces a 100×100 matrix of interpolated accuracy values. The Frobenius norm is calculated on the difference between each of the five resultant matrices (one for each value of the time delay standard deviation), which shows quantitatively how similar they all are to one another. Table I presents these results below. As indicated from the table, the values 10^{-4}, 10^{-3}, and 10^{-2} give accuracies that are very close to each other.

TABLE I

Norm of the Differences Between Results

d_σ	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1
10^{-4}	0	2.54	3.02	8.50	6.31
10^{-3}	2.54	0	3.00	8.59	6.3
10^{-2}	3.02	3.00	0	8.13	5.88
10^{-1}	8.50	8.59	8.13	0	4.82
1	6.31	6.30	5.88	4.82	0

Also interesting is that biologically non-realistic values for $\overline{d}$ performed the best. Generally speaking, the range $\overline{d} \in [0.4, 1.0]$ and $\tau \in [0.005, 0.04]$ provided the best results, especially when $d_\sigma = 10^{-4}, 10^{-3}, 10^{-2}$. Since the simulation of the circuit is limited to a short time according to (4), such large values for $\overline{d}$ ensure that the state of the network at time t is almost wholly dependent on inputs around time $t-\overline{d}$. Thus, in this case, the performance of the LSM depends heavily upon the circuit's ability to project inputs into a larger dimensional space rather than integrating inputs over time. Smaller values for $\overline{d}$ allow spikes propagating along recurrent connections to interfere with the current inputs.

B. Experiment 2: Effect of Noise

If the reason that larger values for $\overline{d}$ result in better performance is due to lack of interference from recurrent inputs, then adding enough noise to the beginning of the signal should result in markedly degraded performance of the LSM. If the noise at the beginning of the signal has length δ, such that $\delta \geq \overline{d}$ and $\delta \geq \zeta$, then circuit will first be stimulated by the noise that later causes interference along recurrent connections during the entire processing of the target pattern. Having $\delta \geq \overline{d}$ insures that interference occurs at the beginning of target pattern processing while $\delta \geq \zeta$ insures that the interference continues throughout the entire processing of the target pattern. To test this hypothesis, we repeated the same experiment as above, limiting it to $d_\sigma = 1 \times 10^{-4}$, but setting $\delta = 0.1$, the same length as the target pattern. Figure 5 displays the results of this experiment.

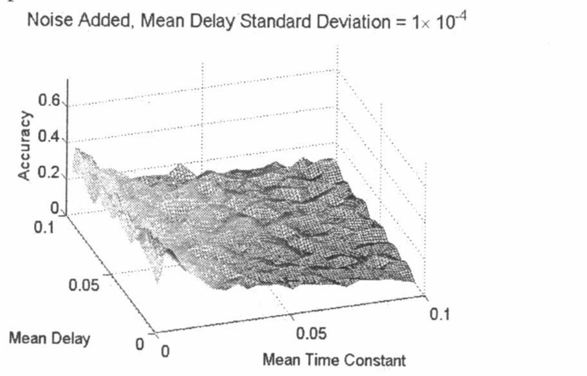

Fig. 4. Results from varying $\overline{d}$ and τ, keeping $d_\sigma = 10^{-4}$, and adding noise, $\delta = 0.1$.

As expected, the performance of the LSM is severely degraded. However, the primary factor affecting performance is not $\overline{d}$ but instead τ. Smaller values for τ provide the best accuracy in this situation, which is most likely due to τ's large impact on how long a neuron's spike affects the network. If τ is large enough, a critical point is reached when activity within the network continues indefinitely with just a few input spikes to start it. Thus, large values of τ have the potential to allow disruptive interference from the past.

C. Experiment 3: Large N and the Effect of Noise

The following experiment tests the robustness of the LSM to large numbers of output classes but also tests how noise affects performance. The parameters of the network are set to values that perform well in Experiment 1, namely $\overline{d} = 0.5$, $\tau = 0.010$, and $d_\sigma = 10^{-4}$. The only variables that are modified are N, ranging from 20 to 200, and δ, set to either 0 or 0.1. Also, two sets of trials are conducted using only one sample point from each instance, the end of the simulation according to (4). The other two sets of trials use the sampling procedure as defined in the introduction of this section. Finally, the training set size is set to $20N$ and the validation set size to $10N$. Figure 5 displays the results.

Again, noise has a significant detrimental effect on performance. However, without noise, the LSM shows surprising robustness in its ability to handle large numbers of output classes, especially when only one sample point is used. When multiple sample points are used, performance for both noisy and non-noisy trials is worse than when only one sample point is used. This is probably due to the simpleness of the learning algorithm and its inability to reconcile multiple state vectors as representing the same output class.

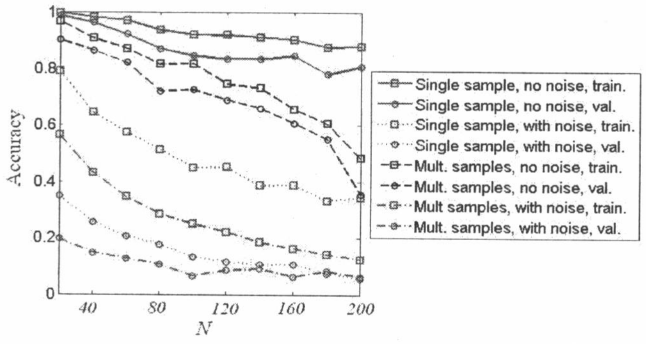

Fig. 5. **Varying the Number of Output Classes:** N is varied from 20 to 200 in increments of 20.

D. Experiment 4: Increasing Information by Increasing Connection Probability

The more information the network has about target patterns, the more likely the training algorithm will be able to learn the output classes. The first method we examine to increase the information available to the network is by simply increasing the connection probability, c_{prob}. With only one input channel and with c_{prob} by default ranging between 0.2 and 0.3, the number of network neurons that receive synapses from the input neuron is likely less than one third. With such a low number of network neurons receiving input spikes, perhaps the computational power of the network is being wasted.

Two sets of $\overline{d}$ and τ are tried: the default values and $\overline{d} = 0.003$ and $\tau = 0.004$. These values are chosen since they behave poorly in the previous experiments with noise; here we desire to elucidate principles that can be used to improve performance in the event that input data might contain noise. Other parameter settings include the following: $d_\sigma = 10^{-4}$, $\delta = 0.1$, $\zeta = 0.1$, the training set size is 1000, and the validation set is of size 200. The results are shown in Figure 6. Increasing c_{prob} does increase accuracy, but it is still not at the level when no noise is present.

E. Experiment 5: Increasing Information with More Input Channels

This experiment tests whether more input channels, and thus more information, increase accuracy. Each additional input

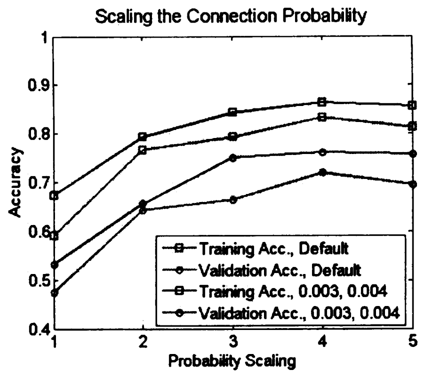

Fig. 6. The probability of a connection between an input neuron and a network neuron is scaled from its default value. At a scale factor of 5, the probability of connecting a input neuron and network neuron is one.

channel is created in the same fashion as when $\lambda = 1$. Since each channel is created randomly, the correlation between the channels should be fairly low, thus providing much additional information. This experiment is identical to the previous experiment except for the fact that c_{prob} is now kept constant at its default value and that the number of input channels now varies. The results are below in Figure 7. Increasing the number of input channels has a positive effect on accuracy.

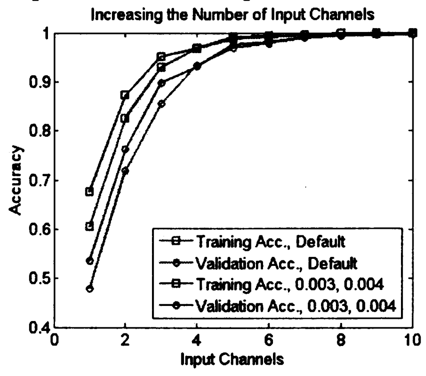

Fig. 7. The number of input channels is varied from 1 to 10.

F. Experiment 6: Increasing Information with Longer Pattern Lengths

Finally, this experiment tests how increasing the pattern length might help the LSM. This experiment is exactly like the previous, except now ζ is no longer fixed but δ remains so. Also, the number of input channels is fixed at 1 and c_{prob} is scaled by 4, as Experiment 4 indicates will probably be best for $\lambda = 1$. Figure 8 displays the results. As expected, the longer pattern lengths are easier to recognize; however, the accuracy does peak. This is probably due to the fact that as ζ increases, so do the number of sample points. Thus the readout function must reconcile an increasingly larger set of states that represent the same output class.

VI. Conclusions and Future Work

Taken together, the experiments suggest two different approaches to two different problems, differentiated by the presence of noise. If all the instances have no noise, then the large $\overline{d}$ values appear to work very well. Using just a single sample point of the network, we are able to achieve good accuracy for over a hundred output classes. However, if the instance contains noise, then small τ values work the best.

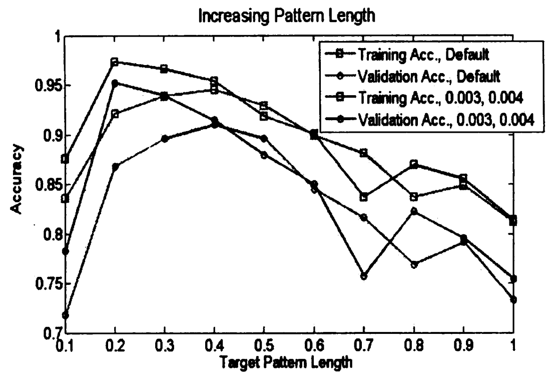

Fig. 8. The target pattern length is varied from 0.1 to 1.0 seconds.

However, if there does exist noise, the task is much more difficult. Even with small τ values, the accuracy achieved does not compare to the no-noise situation. Much more information is required by the network to solve the task. The best way to do this is through multiple channels that contain somewhat uncorrelated information about the target pattern. Also, longer pattern lengths can help. Finally, a small benefit is gained by increasing the input to network connection probability.

A scenario that this paper does not address is the case when an instance contains multiple target patterns. Thus, instead of random noise interfering with classification, target patterns from different output classes cause the interference. As future work, it would be interesting to see if the results concerning noise also apply to this case.

This paper has barely touched the surface of understanding the complex dynamics behind recurrently-connected spiking neural networks. With so many parameters, it is difficult to say with certainty that under all conditions a certain principle holds true. More work should be done to validate the conclusions of this paper across other problems, both artificial and real.

References

[1] W. Gerstner and W.M. Kister, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.
[2] A. Gupta, Y. Wang, and H. Markram, "Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex," *Science* vol. 287, pp. 273–278, 2000.
[3] H. Markram and 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.
[4] W. Maass, "On the complexity of networks of spiking neurons," *Advances in Neural Information Processing*, vol. 7, pp. 183–190, Cambridge: MIT Press, 1995.
[5] W. Maass and T. Natschläger and H. Markram, "Real-time Computing Without Stable States: a New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14(11), pp. 2531–2560, 2002.
[6] Neural Microcircuits, http://www.lsm.tugraz.at/index.html.
[7] T. Natschläger and 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(1), pp. 39–43, 2002.
[8] S. Thorpe and A. Delorme and R. Van Rullen, "Spike-based Strategies for Rapid Processing" *Neural Networks* vol. 14, pp. 715–725, 2001.