Improving Liquid State Machines Through Iterative Refinement of the Reservoir

David Norton, Dan Ventura

Computer Science Department, Brigham Young University, Provo, Utah, United States

Abstract

Liquid State Machines (LSMs) exploit the power of recurrent spiking neural networks (SNNs) without training the SNN. Instead, LSMs randomly generate this network and then use it as a filter for a generic machine learner. Previous research has shown that LSMs can yield competitive results; however, the process can require numerous time consuming epochs before finding a viable filter. We have developed a method for iteratively refining these randomly generated networks so that the LSM will yield a more effective filter in fewer epochs than the traditional method. We define a new metric for evaluating the quality of a filter before calculating the accuracy of the LSM. The LSM then uses this metric to drive a novel algorithm founded on principals integral to both Hebbian and reinforcement learning. We compare this new method with traditional LSMs across two artificial pattern recognition problems and two simplified problems derived from the TIMIT dataset. Depending on the problem, our method demonstrates improvements in accuracy of from 15 to almost 600%.

Keywords: Spiking Neural Network, Liquid State Machine, Recurrent Network

1. Introduction

By transmitting a temporal pattern rather than a single quantity along synapses [1], spiking neural networks (SNNs) can process more information at each node than traditional artificial neural networks [2, 3, 4, 5]. Furthermore, each node in an SNN has its own memory adding to the computational potential of SNNs. Augmenting SNNs with recurrence allows the model to maintain a trace of past events that can aid in the analysis of later inputs.

Despite these strengths, training recurrent SNNs is a problem without a satisfactory solution.

Liquid state machines (LSMs) are a model that attempts to take advantage of recurrent SNNs without training the network. Instead, a randomly generated reservoir, or liquid, of spiking neurons is used as a filter for a simple learner, such as a perceptron. As a filter, the liquid has the potential to transform complex temporal patterns into more basic spatial patterns that are more amenable to machine learning. Unfortunately, randomly generated liquids often do not act as a useful filter, which requires researchers to generate many random liquids until a useful filter is found. Since there is no form of regression in this random creation process, there is no way of iterating upon the best liquid found so far. This results in an unsatisfactory termination criteria. Furthermore, in order to determine whether each filter is useful or not, a separate learner must be trained on the liquid and then evaluated.

We present an algorithm for training the randomly created liquid of an LSM so that a useful filter can be found after fewer liquid generations than using a strictly random search. We show that this algorithm not only decreases the number of liquids that must be created to find a usable filter, it also evaluates each liquid without training a learner and provides a satisfactory termination criteria for the search process.

The algorithm does not attempt to directly improve the accuracy of the system like most machine learners; rather, it attempts to improve the ability of the liquid to separate different classes of input into distinct patterns of behavior—a more general task. We define a metric to measure this ability to separate and aptly call it *separation*. Because separation drives the proposed algorithm, we call it Separation Driven Synaptic Modification (SDSM). In our results, SDSM yields LSMs with higher accuracy than traditional LSMs on all of the problems that we explore.

In order to motivate the subject of LSMs, we begin this paper with Section 2 by describing in detail the characteristics of recurrent SNNs and LSMs. In Section 3 we define separation and provide motivation for its use in driving a learning algorithm. In Section 4, we formally define SDSM. The next two sections show the implementation and results of several experiments comparing SDSM to traditional LSMs. For these experiments we choose two artificial problems and two simplified problems derived from the TIMIT dataset (commonly used in speech recognition) to evaluate SDSM. In addition to our results, Section 5 contains the description and motivation for the artificial problems we use. In Section 6 we describe the problems derived

from the TIMIT dataset, how we convert speech data into spike trains for the LSM, and the results for those problems. In Section 7 we show some results demonstrating transfer learning occurring with SDSM. Finally, the conclusions and implications of our results are discussed in Section 8.

For duplication purposes, we note that all of the LSMs used in this paper are created using CSIM [6, 7].

2. Background

To appreciate the benefits of liquid state machines we first describe the strengths and weaknesses of recurrent spiking neural networks. We then explain, in detail, liquid state machines themselves.

2.1. Recurrent Spiking Neural Networks

Spiking neural networks (SNNs) emulate biological neurons by transmitting signals in a sequence of spikes with constant amplitude. Information has the potential to be encoded in the varying frequencies of spikes produced by the neurons rather than in the single rate value commonly used in non-spiking networks [1]. SNNs can, in theory, convey temporal information more accurately by maintaining non-uniform spiking frequencies. This allows SNNs to have greater computational power for problems in which timing is important [2, 3], including such problems as speech recognition, biometrics, and robot control. Even in problems where timing isn't an explicit factor, SNNs achieve competitive results with fewer neurons than traditional artificial neural networks [4, 5].

Recurrent neural networks allow cycles to occur within the network. This property theoretically improves neural networks by allowing neural activity at one point in time to affect neural activity at a later time. In other words, context is incorporated indirectly into the network, endowing the network with the capability to solve timing-critical problems.

Despite the obvious advantages of using recurrent SNNs over traditional neural networks, they suffer from a distinct lack of satisfactory training algorithms. Concerning the recurrent property alone, only a handful of established algorithms exist, all of which have very high computational costs, limiting them to very small networks [8]. All of these methods also require very specific and sensitive parameter settings for each application in which they are used. Training non-recurrent SNNs is also a developing area of research. Most SNN training algorithms currently proposed only allow for

a single output spike from each neuron [4, 5, 9]. This is unrealistic and computationally limiting. Combining recurrence with SNNs compounds the difficulty of training—though biologically plausible models, such as those of cortical microcircuits, are showing promise [10].

2.2. Liquid State Machines

The Liquid State Machine, or LSM, is one approach for harnessing the power of recurrent SNNs without actually training them [6, 11]. LSMs are a type of reservoir computing comparable to echo state networks (ESN) [12] and Backpropagation Decorrelation (BPDC) [13], neither of which employs spiking neurons. LSMs are composed of two parts: a *reservoir* featuring a highly recurrent SNN, and a *readout function* characterized by a simple learning function. Temporal input is fed into the reservoir which acts as a filter. Then the state of the reservoir, or *state vector*, is used as input for the readout function. See Figure 1. In essence, the readout function trains on the output of the reservoir. No training occurs within the reservoir itself. This process has been analogized with dropping objects into a container of liquid and subsequently reading the ripples created to classify the objects—hence the name liquid state machine.

The aforementioned state vector is specifically a vector containing the binary state of each neuron in the reservoir at a given time—either the neuron is firing or it isn't. Since action potentials occur instantly, the state of each neuron is determined by analyzing a thin slice of time rather than an instance. In other words, the state vector indicates which neurons fired during a given time slice. For all of the experiments in this paper, the time slice for the state vector was taken at the end of the temporal input sequence giving most of the input sequence a chance to percolate throughout the reservoir.

Because no training occurs in the reservoir, the quality of the LSM is dependent upon the ability of the liquid to effectively separate classes of input. Here the term "effectively separate" is defined as the ability of a liquid to yield a state vector with which a readout function can attain acceptable accuracy. Typically, the liquid is created randomly according to some carefully selected parameters specific to the problem at hand. This parameter selection has been the topic of much research [14, 13] although the research has not yet led to a consistent and general method of generating liquids for all problems [15]. Even when adequate parameters for a given problem have been implemented, the creation of a useful liquid is not guaranteed. Typically hundreds or even thousands of liquids will be created to find a suitable

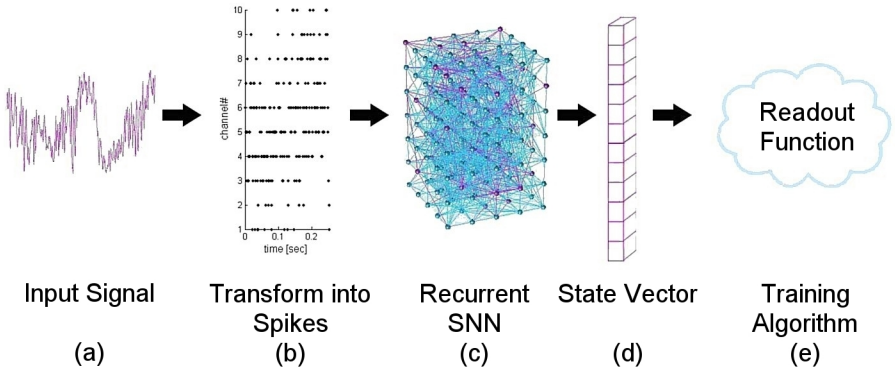


Figure 1: Outline of a liquid state machine: We transform the input signal (a) into a spike train (b) via some function. We then introduce the spike train into the recurrent SNN, or "liquid" (c). Next, the LSM takes snapshots of the state of the "liquid"; these are called state vectors (d). Finally, the LSM uses these state vectors as input to train a learning algorithm, the readout function (e).

filter. Fortunately, once such a liquid is discovered, the results of the LSM are comparable with the state-of-the-art [16, 13, 17]. Such results, coupled with the lack of a sufficient algorithm to train these promising networks, fuel the exploration of LSMs.

3. Separation

Separation is a metric used to determine the effectiveness of a liquid. It measures how well the liquid separates different classes of input into different reservoir states, or state vectors, and is related to supervised clustering in that separation essentially measures how well state vectors are clustered into their respective classes. State vectors are described in detail in Section 2. A metric for separation was first devised by Goodman [16] and is inspired by the description of the properties of a liquid presented by Maass [18]. Goodman's definition of separation essentially finds the mean difference between the state vectors obtained from each of the classes in a problem. In order to more accurately perform the desired measure of separation, we have revised Goodman's definition to take into consideration the variance between state vectors of the same class.

Separation is initialized by dividing a set of state vectors, $O(t)$, into n

subsets, $O_l(t)$, one for each class, where n is the total number of classes. Individual state vectors are represented by $\mathbf{o}$, and the more state vectors available for the metric, the more accurate the calculation of separation. In our notation, rather than strictly representing time, t represents the current iteration of the soon to be described SDSM algotihm.

Separation is divided into two parts, the inter-class distance, $c_d(t)$, and the intra-class variance, $c_v(t)$. $c_d(t)$ is defined by Equation 1 and is the mean distance between the center of mass for every pair of classes. The center of mass for each class, $\mu(O_l(t))$, is calculated with Equation 2. For clarity, we use $|\cdot|$ notation for set cardinality, and $\|\cdot\|_k$ notation for the L_k-norm.

$$c_d(t) = \sum_{l=1}^{n} \sum_{m=1}^{n} \frac{\|\mu(O_l(t)) - \mu(O_m(t))\|_2}{n^2} \tag{1}$$

$$\mu(O_l(t)) = \frac{\sum_{\mathbf{o_m} \in O_l(t)} \mathbf{o_m}}{|O_l(t)|} \tag{2}$$

$c_v(t)$ is defined by Equation 3 and is the mean variance of every cluster, or class, of state vectors. $\rho(O_l(t))$ is the average amount of variance for each state vector within class l from the center of mass for that class. We calculate the mean variance as shown in Equation 4.

$$c_v(t) = \frac{1}{n} \sum_{l=1}^{n} \rho(O_l(t)) \tag{3}$$

$$\rho(O_l(t)) = \frac{\sum_{\mathbf{o_m} \in O_l(t)} \|\mu(O_l(t)) - \mathbf{o_m}\|_2}{|O_l(t)|} \tag{4}$$

Separation can now be defined by Equation 5. $C_v(t)$ is incremented by one to ensure that separation never approaches ∞.

$$\text{Sep}_\Psi(O(t)) = \frac{c_d(t)}{c_v(t) + 1} \tag{5}$$

Here Ψ is the liquid that produced the set of state vectors, $O(t)$. Separation essentially measures the mean distance between classes of state vectors divided by the average variance found within those classes.

In Figure 2 we show that separation does correlate with the effectiveness of a liquid. Here, effectiveness is measured as the accuracy of the LSM at

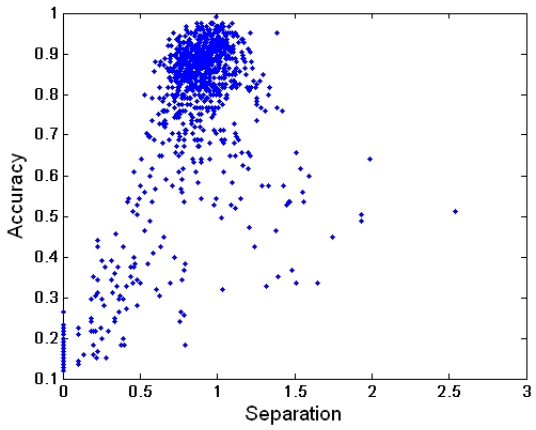


Figure 2: Correlation between accuracy and separation in 1000 different liquids run on an artificial problem. The correlation coefficient is 0.6876.

classifying inputs in an artificial problem. One thousand liquids were generated with varying parameter settings to create a large variety of separation values. The artificial problem consisted of five input classes expressed as template spiking patterns for four input neurons. The system needed to identify which class a variation of the template spiking pattern belonged to. These variations were created by jittering each spike in a template by a normal distribution. The accuracies displayed in Figure 2 were obtained by training the readout with 2000 training instances and then testing with 500 test instances. Separation was calculated with only three examples from each class. Since we were not applying a synapse modifying algorithm to the liquids, only one iteration, t, was observed. The correlation coefficient between accuracy and separation is a convincing 0.6876.

Another method for identifying the computational power of liquids, called the *linear separation property*, has been proposed by Maass et al. [19]. This metric, while geared towards more biologically plausible models than ours, is shown to generalize to a variety of neural microcircuits. Development of an algorithm for preparing effective reservoirs based on the linear separation property may prove useful in future research.

4. Separation Driven Synaptic Modification

Separation Driven Synaptic Modification, or SDSM, is an approach used to modify the synapses of the liquid by using the separation metric defined in Section 3. The underlying principals of SDSM are in part inspired by Hebbian learning and bare some resemblance to prior work [20]. However, SDSM is more akin to a linear regression model since it is guided by the separation metric. Following is an equation that represents SDSM:

$$w_{ij}(t + \Delta t) = \text{sgn}(w_{ij}(t))(|w_{ij}(t)| + e(t)\lambda f(t)) \tag{6}$$

Here $w_{ij}(t)$ is the weight of the synapse from neuron j to neuron i at iteration t, λ is the learning rate, $\text{sgn}(w_{ij}(t))$ is the sign of $w_{ij}(t)$ (i.e. whether the synapse is excitatory or inhibitory), $e(t)$ is a function of the effect of separation on the weight at iteration t, and $f(t)$ is a function of the firing behavior of all neurons in the liquid at iteration t.

First we will look at the function $E(t)$. To explain this function and its derivation it is first important to understand what we mean by relative synaptic strength, r_s, defined by Equation 7.

$$r_s = \frac{|w_{ij}(t)| - \mu_w}{m_w} \tag{7}$$

Here μ_w estimates the expected value of the magnitude of synaptic weights in the initial liquid. m_w is an estimate of the maximum of the random values drawn from the same distribution used to generate synaptic weights in the initial liquid. (These approximations were obtained via simulation with 10,000 samples). m_w normalizes the synaptic strength while μ_w is used to differentiate weak synapses and strong synapses. A negative r_s is considered weak while a positive r_s is considered strong.

Too little distance between centers of mass, $c_d(t)$ (Equation 1), or too much variance within classes, $c_v(t)$ (Equation 3), can decrease separation and thus the overall effectiveness of the liquid. Generally speaking, if there is too little distance between centers of mass, it is because strong synapses are driving the liquid to behave a particular way regardless of input class. To rectify this, we want to strengthen weak synapses and weaken strong synapses. This will drive the liquid towards a more chaotic structure that will yield results more dependent on the input. On the other hand, if there is too much variance within classes, it is because the liquid is too chaotic to drive inputs of the same class to behave similarly. To relieve this problem, it

is necessary to strengthen strong synapses and weaken weak synapses even more. This will polarize the liquid, requiring greater differences in input to cause a change in the liquid's behavior (in other words, the liquid will be less chaotic).

The motivation behind the function $e(t)$ (see Equation 6) is to, at the level of an individual synapse, find a balance between differentiating classes of input and reducing variance within classes. The solution to the problem of differentiating classes of input, $d_i(t)$, is implemented with Equation 8.

$$d_i(t) = \alpha_i(t) \left(1 - \frac{c_d(t)}{\text{Sep}^*_\Psi}\right) \tag{8}$$

$$\alpha_i(t) = \frac{\sum_{k=1}^{n} \mu_i(O_k(t))}{n} \tag{9}$$

Here $\alpha_i(t)$ is the activity of a specific neuron i (the post-synaptic neuron of synapse w_{ij}) at iteration t and is defined by Equation 9. $\alpha_i(t)$ contains $\mu(O_k(t))$ which is the mean of the state vectors in class k. Specifically, $\mu_i(O_k(t))$ is the value of the i^{th} element of the mean state vector. This is also the fraction of state vectors belonging to class k in which neuron i fires. As in Section 3, n is the number of classes for the given problem. Sep^*_Ψ is the optimal separation value for the given problem and liquid. We evaluate Sep^*_Ψ by performing a one-time approximation of the maximum separation for an artificial set of n state vectors that have the same cardinality as state vectors from the given liquid. The artificial set of state vectors is constructed such that for all vectors the average Hamming distance to all other vectors is (approximately) maximized. Sep^*_Ψ must be approximated in this manner because we are not aware of a method to explicitly calculate optimal separation. Since this approximation far exceeds the inter-class distances that we have encountered in actual liquids, it is suitable as a reference point to calculate what is essentially an error function for inter-class distance.

In Equation 8, the normalized value of $c_d(t)$ is subtracted from one so that $d_i(t)$ will provide greater correction for smaller values of $c_d(t)$. Essentially, Equation 8, multiplies the activity of a particular neuron by the amount of correction necessary for too little distance between class centers of mass. We assume that neuron activity is to blame for this problem. This may or may not be the case; however, consistently assuming correlation between $c_d(t)$ and neuron activity should eventually impose this correlation on the liquid and ultimately yield the desired results.

The solution to the problem of reducing variance within classes, is implemented with Equation 10.

$$v_i(t) = \frac{\sum_{k=1}^{n} \mu_i(O_k(t))\rho(O_k(t))}{n} \tag{10}$$

$v_i(t)$ is calculated similarly to $\alpha_i(t)$ except that each instance of $\mu_i(O_k(t))$ is multiplied by the mean variance for class k, because mean variance is determined class by class. The end result is that Equation 10 provides greater correction for larger values of $c_v(t)$, which is desirable since we are trying to reduce intra-class variance. Like the equation for $d_i(t)$ (Equation 8), the equation for $v_i(t)$ assumes a correlation between the neuron's activity and $c_v(t)$.

Equations 8, 9, and 10 all depend upon a sample of state vectors produced by the liquid at iteration t. To maintain the efficiency of SDSM and avoid conventionally training the liquid, only three randomly selected state vectors per class are used.

The function $e(t)$ is derived from the Equations 7-10 as follows:

$$e(t) = r_s \left(v_i(t) - d_i(t)\right) \tag{11}$$

Here $d_i(t)$ is subtracted from $v_i(t)$ because, as mentioned previously, we want the distance correction, $d_i(t)$, to strengthen weak synapses and weaken strong synapses while we want the variance correction, $v_i(t)$ to strengthen strong synapses and weaken weak synapses. In other words, we want $d_i(t)$ to increase the chaotic nature of the liquid and $v_i(t)$ to decrease the chaotic nature of the liquid. Ultimately the goal of Equation 11 is to find a balance between a liquid that is too chaotic and one that is too stable [21]. This goal is in accordance with other studies that have identified the importance of building systems that operate near the edge of chaos, a.k.a. edge of stability [22, 23].

We now turn our attention to $F(t)$, the function of the firing behavior of all neurons in the liquid at iteration t. This function is expressed in three parts as follows:

$$f(t) = \begin{cases} \frac{1}{\phi(t)}, & \text{if } w_{ij}(t)e(t) \geq 0 \\ \phi(t), & \text{if } w_{ij}(t)e(t) < 0 \end{cases} \tag{12}$$

$$\phi(t) = 2^{kA(t)-b} \tag{13}$$

$$a(t) = \frac{\sum_{\mathbf{o} \in O(t)} \frac{\sum_{\eta \in \mathbf{o}} \eta}{|\mathbf{o}|}}{|O|} \tag{14}$$

Here $a(t)$ is the activity of the entire liquid at iteration t and is calculated by finding the average fraction of neurons, η, that fire in each state vector in $O(t)$. $\phi(t)$ is a transformation of $A(t)$ that reduces it to a function that will allow $f(t)$ to be multiplied by $e(t)$ in Equation 6. $\phi(t)$ contains two variables, k and b, that represent, respectively, the scale and offset of the transformation. For our experiments, $k = 6$ and $b = 3$ were found, through preliminary tuning, to yield the highest separation values. $f(t)$ uses the state of the synapse and the results of $e(t)$ to determine how the global activity of the liquid at iteration t will effect the change in weight. The effect of $f(t)$ is to promote the overall strengthening of excitatory synapses while promoting the overall weakening of inhibitory synapses if less than half of the neurons in the liquid fire. If more than half of the neurons fire, the effect of $f(t)$ is reversed. The goal of $f(t)$ is to direct the liquid to a "useful" amount of activity. This assumes that half of the neurons firing for all state vectors is the desired fraction of activity to achieve maximum separation. This is not an unreasonable assumption as such a condition would allow for statistically the most variation between states, and thus allow for the highest degree of separation possible between numerous states.

5. Applying SDSM

We develop two artificial problems to test the effect of Separation Driven Synaptic Modification (SDSM) on the accuracy of an LSM given limited liquid creations. This section explores the results of these experiments and shows the strength of this algorithm in selecting ideal liquids for these problems.

We call the simpler of the two problems the *frequency recognition* problem. This problem has four input neurons and five classes. Each input neuron fires at a *slow* or *fast* frequency. The five classes are defined by specific combinations of fast and slow input neurons as shown Table 1, where 1 represents a fast input neuron and 0 a slow one. These particular patterns are chosen to challenge the liquid with a variety of combinations as well as the task of ignoring one channel (input neuron 4).

Individual samples from each class are generated by following the class template and then jittering the frequencies. Since each class is distinctly

	Input 1	Input 2	Input 3	Input 4
Class 1	1	0	0	0
Class 2	0	1	0	0
Class 3	1	1	0	0
Class 4	0	0	1	0
Class 5	1	0	1	0

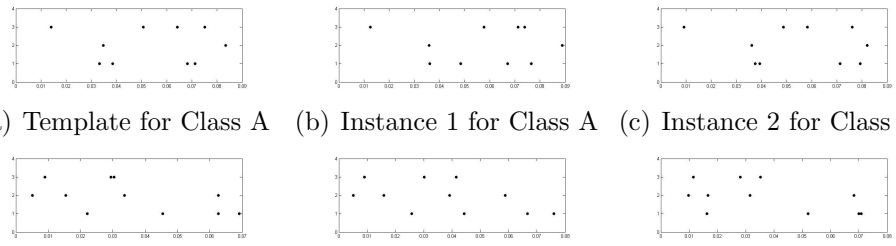
Table 1: Frequency patterns for each class in the frequency recognition problem. Each input represents one of four input neurons. A 1 indicates a fast spiking frequency while a 0 represents a slower spiking frequency.

defined by a particular pattern of behavior with only two options for each neuron, this is a fairly simple problem.

The second problem, which is more complex, we call the *pattern recognition* problem. This problem has eight input neurons and a variable number of classes. Each class is based on a template spike pattern randomly created for each input neuron. We generate this random pattern by plotting individual spikes with a random distance between one another. This distance is drawn from the absolute value of a normal distribution with a mean of $10ms$ and a standard deviation of $20ms$. Once we create the template pattern for each input neuron in a class, individual instances from the class are created by jittering each spike in the template. The spikes are jittered by an amount drawn from a zero-mean normal distribution with a standard deviation of $5ms$ making the problem particularly difficult. We derive all of these values through empirical analysis to create a solvable but difficult problem. A simplified example with only two classes and three input neurons is shown in Figure 3.

5.1. Parameter Settings

The models we used for the neurons and synapses for all of our experiments were taken from CSIM [6, 7]. Specifically, we used the *StaticSpikingSynapse* model for synapses, and the *SpikingInputNeuron* and *LifNeuron* models for the input and reservoir neurons respectively. We determined the best setting for each of the many parameters in the liquid by performing extensive preliminary tuning. The parameters that apply to traditional liquids were chosen based on the best tuning results prior to any application of SDSM. The different parameters we looked at in these preliminary ef-



(a) Template for Class A (b) Instance 1 for Class A (c) Instance 2 for Class A

(d) Template for Class B (e) Instance 1 for Class B (f) Instance 2 for Class B

Figure 3: Simplified example templates for two classes, A and B, are shown in (a) and (d) respectively. Each class has three input neurons designated by the y-axis. The x-axis is time spanning 100ms. (b) and (c) show examples of instances from class A created from jittered versions of the template. (e) and (f) show examples of instances from class B.

forts were the number of neurons, connection probability, the mean synaptic weight and delay, the standard deviation of the synaptic weight and delay, the number of samples per class used to determine separation at each instance, the number of iterations to run, the learning rate, the synaptic time constant, and the amount of noise present in each neuron. While excitatory and inhibitory neurons are not explicitly created, positive synaptic weights are excitatory and negative synaptic weights are inhibitory. Table 2 shows the parameters we use for all of the results presented in this section as well as Section 6. As the results of this section and the next section show, these parameters generalize very well as long as the temporal scale of the input is on the order of one second.

Some of the parameters presented in Table 2 require further explanation to more easily replicate the results. The connection probability is the probability that any given neuron (including input neurons) is connected to any other liquid neuron (liquid neurons cannot connect back to input neurons). The value for the connection probability indicated in Table 2 means that each neuron is connected to roughly one third of the other neurons. The mean and standard deviation for synaptic weight define the distribution from which the initial weight values are drawn for every synapse. The same applies to the mean and standard deviation for synaptic delay—although the synaptic delay values for each synapse remain constant during SDSM. The "samples per class" parameter refers to the number of training samples used from each class when calculating separation. This in turn is what drives the SDSM algorithm. The more samples used, the more accurate the separation

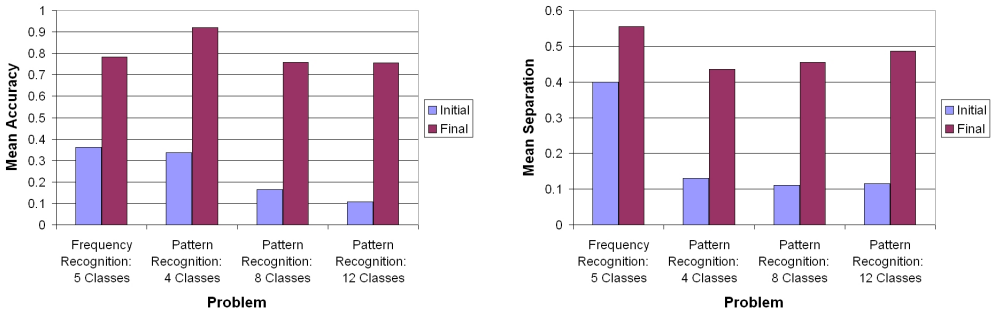
Neurons	64
Connection Probability	0.3
Synaptic Weight Mean	$2 \cdot 10^{-8}$
Synaptic Weight SD	$4 \cdot 10^{-8}$
Samples per Class	3
Training Iterations	500
λ	$5 \cdot 10^{-10}$
Inoise	$5 \cdot 10^{-8}$
τ	0.003
Synaptic Delay Mean	0.01
Synaptic Delay SD	0.1
State Vector Length	0.05

Table 2: Parameters used by SDSM for Artificial and Phonetic problems.

calculation will be, but at the cost of speed. The number of iterations is simply how long to run the SDSM algorithm. In our experiments, by 500 iterations, most liquids had reached a plateau in separation improvement. λ is the learning rate first shown in Section 4. τ is the synaptic time constant which refers to the rate at which the membrane potential of each synapse decays. Inoise is the standard deviation of the noise added to each neuron and is necessary for an efficient liquid [24]. Finally, the "state vector length" parameter is the length, in seconds, of the time slice at the end of each temporal input sequence from which each state vector is collected.

5.2. Results

Using the established parameters, we create LSMs with SDSM for both the pattern and the frequency recognition problems (explained above). For the pattern recognition problem we explore 4-, 8-, and 12-class problems. We only explore the specifically defined 5-class scenario for the frequency recognition problem. For each problem we run fifty experiments, each with a unique randomly generated initial liquid. State vectors obtained from both the initial liquid and the liquid after five hundred iterations of SDSM are used as input to multiple perceptrons. We train each perceptron to classify members of one particular class, so there are n binary classifiers. We then compare the output of each perceptron, assigning the class of the perceptron with the greatest confidence to the state vector in question. This readout



(a) Mean Accuracy

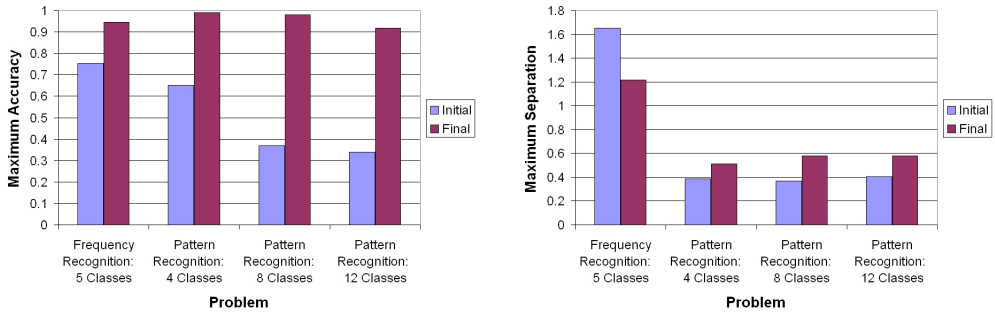
(b) Mean Separation

Figure 4: A comparison of traditional liquids (initial) and those shaped by SDSM (final) across four problems. Results are the mean accuracy (a) and separation (b) of fifty LSMs.

function is used because it is very simple, thus allowing us to more carefully scrutinize the quality of the liquid. For all of our experiments, the test size is one hundred samples per class.

Figure 4(a) shows the mean accuracy (over all fifty experiments) of the LSM for each problem. Additionally Figure 5(a) shows the best accuracy obtained out of the fifty experiments for each problem. In practice, the liquid with the maximum accuracy is the one that we would select for future use. However, since we are interested in the potential of SDSM to decrease the number of liquids that must be created, we are also interested in the mean accuracy. Note that the initial liquid is the typical LSM scenario: a strictly randomly generated liquid. So, both of these figures are a comparison of standard LSMs to LSMs using SDSM. We did not perform a statistical analysis of these results since the improvement of SDSM over the traditional method is clearly substantial in all cases.

In addition to the accuracy of the LSMs, Figures 4 and 5 show the separation of the liquids. It should be noted that the liquid with the maximum separation is not necessarily the same liquid that accounts for the maximum accuracy. Overall these results support the correlation between separation and accuracy, though there is an anomaly in Figure 5(b) where a higher separation occurs in an initial random liquid than in any SDSM shaped liquid. Such anomalies are bound to occur occasionally due to the random nature of LSMs. The mean results in Figure 4(b) confirm the abnormal nature of this event.



(a) Maximum Accuracy

(b) Maximum Separation

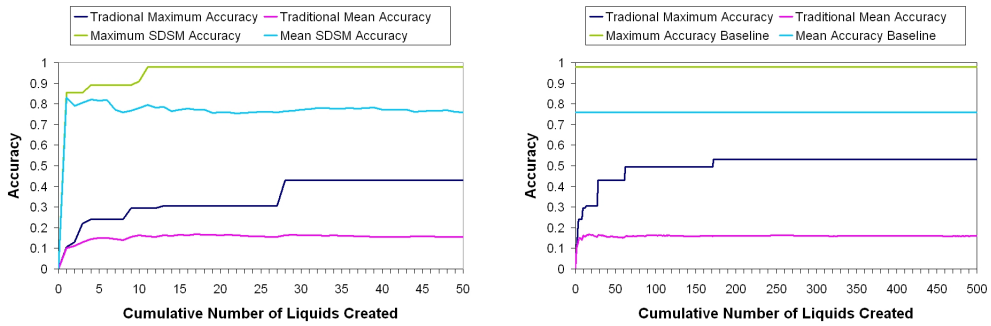
Figure 5: A comparison of traditional liquids (initial) and those shaped by SDSM (final) across four problems. Results are the best accuracy (a) and separation (b) obtained out of fifty LSMs.

5.3. Discussion

These results show a substantial increase in the performance of liquids using SDSM, particularly in the problems with more classes. Note that SDSM's mean accuracy is higher than the traditional method's best accuracy—for every problem (Figures 4(a) and 5(a)).

Figure 6 compares the trends in liquid exploration for both traditional and SDSM generated liquids. The results in this figure are from the pattern recognition problem with eight classes. Figure 6(a) shows the accuracy of the best liquid obtained so far, given an increasing number of total liquids created. (The mean accuracy is also shown). Figure 6(a) demonstrates that far fewer liquids are required to obtained a satisfactory liquid using SDSM when compared with the traditional method. Figure 6(b) extends the graph further for traditional liquid creation, and compares the results to the best results obtained from SDSM after only fifty liquid creations. We see from this figure that even after 500 liquid creations, a liquid has not been created by the traditional means that can compete with SDSM after only eleven liquid creations. This figure illustrates that not only can SDSM find a suitable liquid in fewer iterations than traditional methods, but it can also potentially create a liquid that obtains higher accuracy than what is possible with conventional LSMs in a reasonable amount of time.

Figure 7 shows how separation changes with each iteration over the history of a typical SDSM trial. Figure 8 shows the mean value of separation



(a) Accuracy Over 50 Liquid Creations (b) Accuracy Over 500 Liquid Creations

Figure 6: A comparison of accuracy trends in traditional liquids and those generated with SDSM. These results show how the accuracy of the best liquid obtained so far improves as more liquids are created. The results also show that the mean accuracy stabilizes as more liquids are created. (a) shows the trend up to fifty liquid creations. (b) shows the trend up to 500 liquid creations for traditional liquids only (the accuracy for SDSM generated liquids is shown as a baseline only).

over fifty trials of SDSM. These results show that separation is clearly improving over time for the pattern recognition problems. For the frequency recognition problem, separation quickly increases and then slowly begins to decrease. This is somewhat distressing and may indicate an instability in the algorithm when confronted with certain problems; however, even after 500 iterations of SDSM, the average separation is still higher than that of the initial liquid. Also, the results in Figure 4(a) indicate that, even on the frequency recognition problem, the accuracy of liquids trained with SDSM greatly outperform that of traditional liquids. While we currently do not know the source of this anamoly, we are satisfied with the overall increase in liquid performance and will leave a deeper investigation of the abnormality for future research.

Overall, these results indicate that the SDSM algorithm is behaving as expected (i.e. improving separation). Since SDSM results in a significant improvement in accuracy, the notion that separation correlates with accuracy is also strengthened by these results. It is important to keep in mind that separation in both of these figures was calculated using only three samples per class for each iteration and is thus a very rough estimate. The initial and final separation values indicated in Figures 4(b) and 5(b) are much more accurate approximations of separation—here, we use one hundred samples

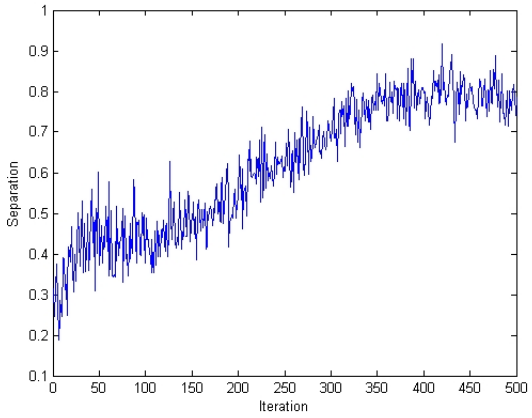
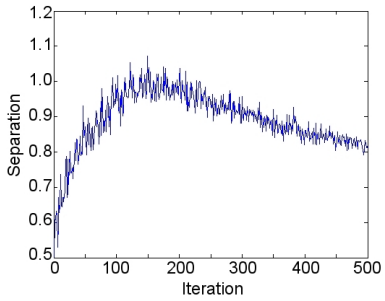


Figure 7: The separation of a liquid at each iteration during training of SDSM. This is a single representative example out of two hundred different liquids created in this set of experiments. This particular liquid was created using the pattern recognition problem with eight classes.

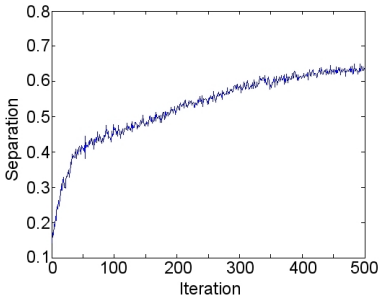
per class.

Though the correlation between accuracy and separation is not perfect, it is satisfactory as a metric for both evaluating the quality of a liquid and revealing beneficial changes that can be made to a liquid. As the algorithm in Section 4 reveals, the relationship between separation and synaptic modification within the SDSM algorithm is complicated and can't necessarily be predicted. Some flaws of the separation metric become apparent when studying Figure 5(b). While the accuracy for the frequency recognition problem and the pattern recognition problems are similar, the best liquid separation is distinctly higher in liquids created with the frequency recognition problem. This demonstrates that different problems will yield different separation values due to their unique properties. In this particular case the disparity is probably at least partly a result of the differing number of input neurons present in each problem. The pattern recognition problems all have twice as many input neurons as the frequency recognition problem. This characteristic of the separation metric does not necessarily call for a change in the metric. It does mean however, that separation values between different problems cannot be adequately compared to one another.

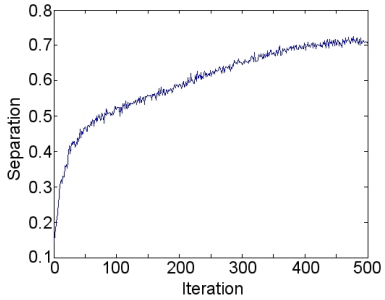
Finally, for our experiments we fixed the number of training iterations



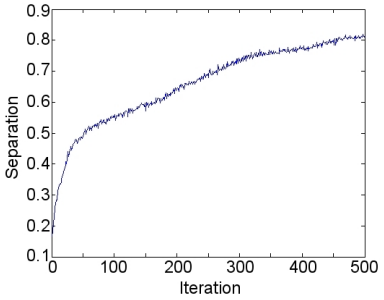
(a) Frequency Recognition: 5 classes



(b) Pattern Recognition: 4 classes



(c) Pattern Recognition: 8 classes



(d) Pattern Recognition: 12 classes

Figure 8: The mean separation history using SDSM for the four problems explored in our experiment. Each history is an average of fifty trials.

to 500 in order to bring attention to the limitations of traditional liquid creation methods. However, since SDSM typically yields a increase in separation, surpassing a specific separation threshold could be used as a desirable termination criteria in practical applications. Such a threshold would not be as reliable in the traditional approach as there is not an increase in separation over time.

6. TIMIT Classification with SDSM

Liquids created with SDSM show significant improvement over traditional LSMs in tightly controlled artificial problems. Now we will explore the use of SDSM in classifying phonemes found within the TIMIT dataset [25].

6.1. TIMIT

TIMIT consists of 6300 spoken sentences, sampled at 16 kHz, read by 630 people employing various English dialects. Since we are interested in identifying context independent phonemes, we break each sentence down into its individual phonemes using phonetic indices included with the sound files. This results in 177389 training instances. We then convert each of these phoneme WAV files into its 13 Mel frequency cepstral coefficients (mfccs) [26] sampled at 200 Hz. Finally, we convert the mfccs into thirteen spike trains (one for each mfcc) with a firing rate calculated using Equation 15 (taken from Goodman [16]).

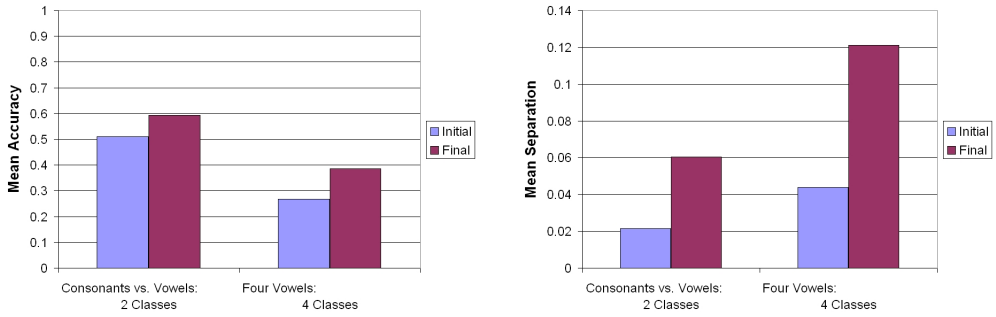
$$\text{Rate}_i(t) = \frac{\text{mfcc}_i(t) - \omega_i}{(\Omega_i - \omega_i)} \cdot \text{MaxRate} \tag{15}$$

Here $\text{mfcc}_i(t)$ is the i^{th} mfcc value at time t which corresponds with the firing rate for input neuron i; ω_i is the minimum value of the i^{th} mfcc, and Ω_i is its maximum value. $MaxRate$ is the maximum rate of firing possible for a given input neuron. $MaxRate$ is a constant that, as per Goodman, we defined as 200 spikes per second. Additionally, preliminary work shows that in order for the liquid to achieve any appreciable level of separation, the time span of the input needs to be on the order of one second due to the operating timescale of the liquid. Since single phonemes have a length on the order of $50ms$, we temporally stretch each of the spike trains using Equation 16.

$$t_{new} = \frac{1}{1 + e^{-k \cdot t_{old}}} \tag{16}$$

Here t_{old} is the original length of a spike train while t_{new} is the new stretched length. The variable k is a sigmoid gain that we set to five, based on preliminary tuning.

The TIMIT dataset contains roughly fifty-two phonemes. Out of context, correctly identifying all of these phonemes is a daunting task. Using six natural classes of phonemes, we reduce this problem to two simpler problems. The first is a general problem that involves identifying phonemes as either consonants or vowels. For this problem "stops", "affricates", and "fricatives" are considered consonants. "Nasals" and "semivowels" are removed to avoid ambiguous sounds. The training data consists of 1000 instances of each class and the test data contains 100 instances of each class. The second problem is more specific and involves identifying one of four distinct "vowel" phonemes.



(a) Mean Accuracy (b) Mean Separation

Figure 9: A comparison of SDSM and traditional LSMs across two problems derived from the TIMIT dataset. Results are the mean accuracy (a) and separation (b) of fifty LSMs.

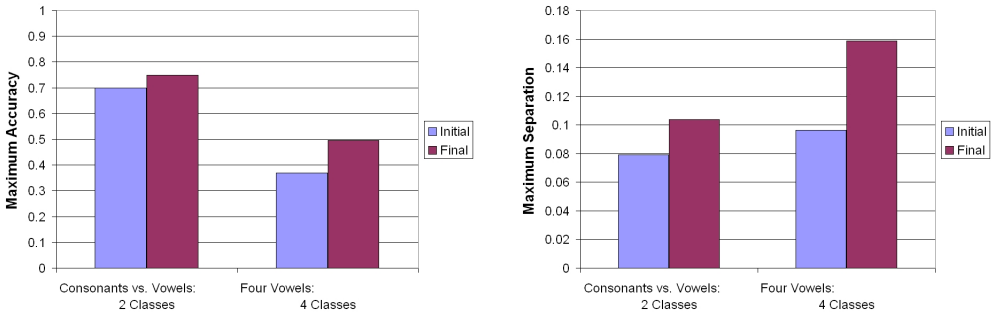
The phonemes used in this problem are ē as in b*ee*t; ĕ as in b*e*t; ŭ as in b*u*t; and the *er* sound in butt*er*. For this problem, the training data consists of 150 instances of each class while the test data contains 50 instances.

6.2. Results

The two problems outlined above are run on LSMs using SDSM, and on traditional LSMs. The mean results can be found in Figure 9 and the best results are shown in Figure 10. These results are obtained by running the problems on fifty liquids either generated with SDSM or created randomly, with both the accuracy of the LSMs as well as the separation of the liquids displayed (keep in mind that the liquids showing the best separation do not necessarily correspond to the LSMs with the highest accuracy). The parameter settings we use in these algorithms are the same as those outlined in Section 5 with the exception of the number of training iterations. In experiments with TIMIT, we run SDSM for only two hundred iterations, since liquids tend to reach a plateau in separation improvement by this point. Again, we did not perform a statistical analysis of these results since the improvement of SDSM over the traditional method is clear in all cases.

6.3. Discussion

Although the results of SDSM on TIMIT data are not as distinguished as the results obtained with artificial template matching (Section 5), SDSM



(a) Max Accuracy

(b) Max Separation

Figure 10: A comparison of SDSM and traditional LSMs across two problems derived from the TIMIT dataset. Results are the best accuracy (a) and separation (b) obtained out of fifty LSMs.

still shows a significant improvement over traditional liquids. Based on Figure 9(a), on average traditional liquids do no better than guessing with either of the phoneme recognition problems while LSMs using SDSM improve over this baseline.

While neither of these phoneme recognition problems are performed at an immediately applicable level, even when only looking at the best liquids created with SDSM (Figure 10(a)), it should be reiterated that these results are obtained by classifying individual phonemes completely out of context. Most of these sounds last one twentieth of a second and span a diverse range of accents. Speech recognition tasks usually involve classifying phonemes as they are heard in a stream of data, thus providing context for each sound (i.e. preceding phonemes and pauses). Even in cases where words are being classified (such as classifying spoken digit), the fact that each word consists of multiple phonemes aids in the classification process. We chose to perform only out-of-context experiments in order to keep these problems parallel to the artificial ones in Section 5 and to focus on the separation properties of the liquid. The poor accuracy obtained in classifying phonemes may also lie with our process of converting mfccs into spike trains. However, different methods of encoding input into spike trains is a whole domain of research in and of itself, so we will leave that to future work.

Even on these difficult real-data problems, the improvement of using SDSM over traditional methods is clear. Also, the fact that the results

from both of these real-data problems were obtained using essentially the same parameters as those obtained from both of the artificial-data problems in Section 5, emphasizes another strength of the SDSM algorithm—a robustness of algorithm parameters. Extensive parameter exploration of the liquids for these problems did not show a marked improvement over the settings already obtained in Section 5. Parameter exploration on a problem by problem basis is a ubiquitous and time consuming component of machine learning. While these results do not exclude the possibility of the necessity for parameter exploration on other problems, they show that there is a level of robustness here not common in machine learning.

7. Transfer Learning with SDSM

Transfer learning refers to the idea of transferring acquired knowledge from one domain to another similar, but distinctly different, domain. In machine learning it often refers to the approach of using a model already trained for success on one problem to realize instantaneous performance gains on another related problem [27, 28]. One of the strengths of LSMs is the ability of the liquid to be transfered to different problems while maintaining effectiveness as a filter—effectively transfer learning. This is an important property since useful liquids can be difficult to find. Because SDSM essentially uses training data to create the liquid, there is a concern that the liquid's ability to transfer may be compromised. In order to test the transfer-learning capability of the SDSM generated liquids, we run each of the problems addressed in Figures 4 and 5 on every liquid used to generate those figures. In other words, we are using the same liquid generated for one problem as the reservoir for a new readout on a different problem. The results are shown in Figure 11. When discussing the problem used to create a specific liquid, we refer to the problem as the *source problem*.

Input neurons are considered part of the liquid, thus their synapses are modified as part of SDSM. When a liquid is created from a source problem, it has i input neurons, where i is the number of spike trains that encode the source problem's input. Because different problems have differing numbers of spike trains, when transfering problems across liquids, discrepancies between the number of spike trains and number of input neurons must be resolved.

When running a new problem on a liquid that was trained with a different source problem, we use the following approach. If the new problem's input is encoded in fewer spike trains than the source problem, then we arbitrarily

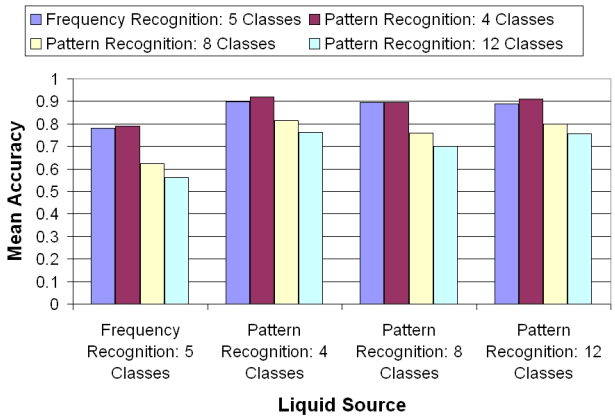


Figure 11: We used SDSM to generate fifty unique liquids for each of four different source problems. The corresponding liquid types are indicated on the x-axis. Each of the four problems was then applied to each of these liquids as indicated by the colored bars. The mean accuracy of the readouts trained on each liquid-problem combination is reported.

map the spike trains to a subset of the liquid's input neurons. The excess input neurons receive a null signal as input. If the new problem's input is encoded in more spike trains than the source problem, then we arbitrarily map multiple spike trains from the new problem's input to individual input neurons in the liquid. We combine the spiking patterns by simply including all spike instances from multiple inputs into a single spike train. For each experiment, we use the same arbitrary mapping for every input to maintain consistency.

The results shown in Figure 11 are obtained using two types of problems. All of the pattern recognition problems use eight spike trains for each instance while the frequency recognition problem uses only four spike trains. When the frequency recognition problem is run on a source liquid that was generated by applying SDSM to a pattern recognition problem, four of the input neurons have no signal. When a pattern recognition problem is run on a source liquid that was generated by applying SDSM to the frequency recognition problem, each input neuron combines the signals of two spike trains.

Figure 12 shows results using initial random liquids as the source liquid. Since all of the pattern recognition problems have the same number of spike trains, and since the initial liquids are untrained, the three data points show-

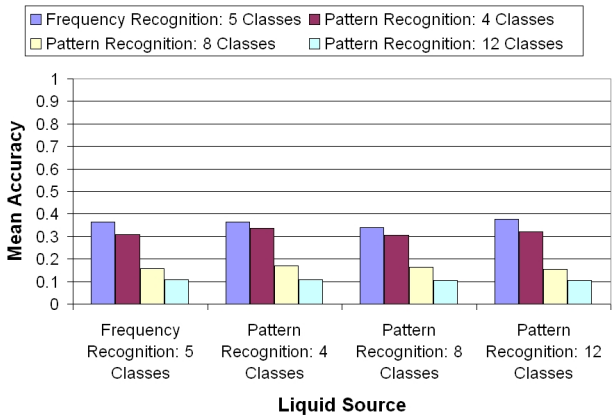


Figure 12: We randomly generate fifty unique liquids for each of four different source problems. The corresponding liquid types are indicated on the x-axis. Each of the four problems was then applied to each of these liquids as indicated by the colored bars. The mean accuracy of the readouts trained on each liquid-problem combination is reported.

ing the results for these initial liquids are actually redundant. However, they are included to allow a side by side comparison with Figure 11.

These results demonstrate the ability of liquids to transfer knowledge to different problems. Figure 12 emphasizes this by the distinct lack of variation in behavior between liquids generated for frequency recognition and pattern recognition problems. One might expect a difference in behavior between these two different liquid states since liquids created for frequency recognition only have four input neurons while liquids created for pattern recognition problems have eight. The fact that the results show no significant difference indicates that the liquid is indeed acting as a general temporal filter.

SDSM uses training data to create new liquids from those randomly generated for Figure 12. Since the new liquids that are created depend upon the problem used to train them (the liquid source), one might expect that the efficacy of the learning transfer would be compromised. Interestingly, the results shown in Figure 11 clearly demonstrate that this is not the case. In fact, liquids not created with frequency recognition as the source problem performed better on that problem than liquids actually created with frequency recognition as the source problem. However, liquids created with the various pattern recognition source problems did perform better on those problems than liquids generated with frequency recognition as the source problem. In both

cases, SDSM still performed significantly better than traditional LSMs (compare Figures 11 and 12). The fact that liquids created with pattern recognition performed better on both problems indicates that the source problem used to create the liquid can make a difference. Looking at Figure 11 we see that all of the liquids created with pattern recognition source problems found liquids that performed better on all of the problems than liquids created with frequency recognition as the source problem. Pattern recognition is arguably the more complicated of the two problems because of the number of inputs and more distinct separation of classes in the frequency recognition problem; and, by applying SDSM with the more complicated source problem, the liquid may be primed for overall better performance. It should be noted that transferring learning across different numbers of classes and input neurons alone is a difficult problem that SDSM apparently overcomes. Future work should investigate a greater variety of problems to determine the extent of generalization that is possible using SDSM.

8. Conclusions

For a variety of temporal-pattern-recognition problems, we have shown that by training the liquid, our new algorithm, SDSM, reduces the number of liquids that must be created in order to find one that acts as a useful filter for the LSM. Additionally, separation provides a metric for directly evaluating the quality of the liquid, and can thus act as a satisfactory termination criteria in the search for a useful filter.

Also, despite training the liquid for a specific problem, we have demonstrated the ability of liquids generated with SDSM to transfer between distinct problems effectively. This requires individual liquids to elicit variable numbers of distinct responses to account for differing numbers of classes between problems. Furthermore, transfer learning occurs despite a naive translation from one input representation to another in cases with differing numbers of input neurons.

SDSM may allow researchers to more effectively create liquids for most LSM research—both improving results and cutting down on experimental run-time. More generally, since the liquid in an LSM is a recurrent SNN, and since SDSM trains this liquid, our algorithm may present a useful option for researchers of recurrent SNNs to explore.

[1] W. Gerstner, W. Kistler (Eds.), Spiking Neuron Models, Cambridge University Press, New York, 2002.

[2] S. M. Bohte, Spiking neural networks, Ph.D. thesis, Centre for Mathematics and Computer Science (2003).

[3] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural Networks 10 (9) (1997) 1659–1671.

[4] S. M. Bohte, J. N. Kok, H. L. Poutré, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (2001) 17–37.

[5] O. Booij, H. T. Nguyen, A gradient descent rule for spiking neurons emitting multiple spikes, Information Processing Letters 95 (2005) 552–558.

[6] T. Natschläger, Neural micro circuits, http:/www.lsm.tugraz.at/index.html (2005).

[7] T. Natschläger, W. Maass, H. Markram, Computer models and analysis tools for neural microcircuits, in: Neuroscience Databases. A Practical Guide, Kluwer Academic Publishers, 2003, Ch. 9, pp. 123–138.

[8] H. Jaeger, A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach, Tech. rep., International University Bremen (2005).

[9] W. Mass, Networks of spiking neurons: the third generation of neural network models, Transactions of the Society for Computer Simulation International 14 (1997) 1659–1671.

[10] W. Maass, P. Joshi, E. D. Sontag, Computational aspects of feedback in neural circuits, PLoS Computational Biology 3 (1:e165) (2007) 1–20.

[11] T. Natschläger, W. Maass, H. Markram, The "liquid" computer: A novel strategy for real-time computing on time series, Special Issue on Foundations of Information Processing of TELEMATIK 8 (1) (2002) 39–43.

[12] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science (2004) 78–80.

[13] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Networks 20 (2007) 391–403.

[14] E. Goodman, D. Ventura, Effectively using recurrently connected spiking neural networks, Proceedings of the International Joint Conference on Neural Networks 3 (2005) 1542–1547.

[15] D. Verstraeten, B. Schrauwen, D. Stroobandt, Adapting reservoirs to get Gaussian distributions, European Symposium on Artificial Neural Networks (2007) 495–500.

[16] E. Goodman, D. Ventura, Spatiotemporal pattern recognition via liquid state machines, Proceedings of the International Joint Conference on Neural Networks (2006) 3848–3853.

[17] D. Verstraeten, B. Schrauwen, D. Stroobandt, J. V. Campenhout, Isolated word recognition with liquid state machine: a case study, Information Processing Letters 95 (2005) 521–528.

[18] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computations based on perturbations, Neural Computation 14 (11) (2002) 2531–2560.

[19] W. Maass, N. Bertschinger, R. Legenstein, Methods for estimating the computational power and generalization capability of neural microcircuits, Advances in Neural Information Processing Systems 17 (2005) 865–872.

[20] S. Bornholdt, T. Röhl, Self-organized critical neural networks, Physical Review E 67.

[21] N. Brodu, Quantifying the effect of learning on recurrent spiking neurons, Proceedings of the International Joint Conference on Neural Networks (2007) 512–517.

[22] N. Bertschinger, T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks, Neural Computation 16 (7) (2004) 1413–1436.

[23] T. Natschläger, N. Bertschinger, R. Legenstein, At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks, Advances in Neural Information Processing Systems.

[24] K. Jim, C. L. Giles, B. G. Horne, An analysis of noise in recurrent neural networks: Convergence and generalization, IEEE Transactions on Neural Networks 7 (1996) 1424–1438.

[25] J. S. Garofolo et al., Timit acoustic-phonetic continuous speech corpus, http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?ca talogId=LDC93S1, linguistic Data Consortium, Philadelphia (1993).

[26] ETSI ES 202 212STQ: DSR, Extended advanced front-end feature extraction algorithm; compression algorithms; back-end speech reconstruction algorithm, Tech. rep., European Telecommunications Standards Institute (ETSI) (2003).

[27] S. Thrun, L. Pratt (Eds.), Learning to Learn, Kluwer Academic Publishers, 1998.

[28] K. Yu, V. Tresp, Learning to learn and collaborative filtering, Neural Information Processing Systems workshop "Inductive Transfer: 10 Years Later".