Preparing More Effective Liquid State Machines Using Hebbian Learning

David Norton and Dan Ventura

Abstract—**In Liquid State Machines, separation is a critical attribute of the liquid—which is traditionally not trained. The effects of using Hebbian learning in the liquid to improve separation are investigated in this paper. When presented with random input, Hebbian learning does not dramatically change separation. However, Hebbian learning does improve separation when presented with real-world speech data.**

I. Introduction

SPIKING recurrent neural networks (neural microcircuits) have powerful representational power due to the complex time dimension of the network. Spiking neurons closely emulate biological neurons by transmitting signals in the form of spike trains where each spike has constant amplitude. Information is encoded in the varying frequencies of spikes produced by the neurons rather than the single rate value commonly used in non-spiking networks [1]. Spiking neural networks can convey temporal information more accurately by maintaining non-uniform spiking frequencies. In addition, the recurrent property of neural microcircuits allows for input acquired previously in time to change how later input affects the network. Unfortunately, the additional computational abilities of the network result in more variables which complicate the training of the network. The recurrent properties of neural microcircuits further obscure the networks so that only very simple neural microcircuits have been successfully trained.

In an attempt to exploit the power of very complex neural microcircuits, the liquid state machine (LSM) [2][3], was invented. The liquid state machine consists of two major components: the liquid and the reading function. An input signal (Fig. 1a) is introduced into the liquid in the form of spike trains (Fig. 1b). This instigates a pattern of spikes throughout the neurons in the liquid (Fig. 1c) which is allowed to persist for a set duration. In this paper the duration is always the length of the input spike train. The spiking neurons are recorded as a state vector (Fig. 1d) at set intervals determined by a sampling rate. Each of these state vectors can be thought of as a snap shot of the current state of the liquid. Once the state vector(s) have been collected, they are passed as input to the reading function (Fig. 1e) which performs the task of classifying the input. [4]

David Norton and Dan Ventura are with the Computer Science Department, Brigham Young University, Provo, UT 84602 USA (email: ghotikun@hotmail.com, ventura@cs.byu.edu).

Intuitively, the liquid state machine can be compared to electroencephalography in which a device measures brain activity and creates EEGs (electroencephalograms) that contain information about the state of the brain. This information can provide insight into how the brain interprets any sensory stimulus it may have been experiencing at the time the EEG was recorded. In this example, the input into the liquid is the sensory stimulus that the brain is exposed to. The liquid is the brain itself and the EEGs represent the state vectors. The reading function is the human or machine that interprets the EEGs.

The LSM is, in effect, an attempt to avoid the necessity of training a neural microcircuit. Rather, a random neural microcircuit is created in order to change a complex temporal input space into the spatial domain that traditional learning algorithms perform well on. This paper investigates the possibility of actually training the liquid through the use of Hebbian learning in an attempt to create structure in the liquid that will allow for a more effective mapping of the input from the temporal to the spatial domain. Because of the unsupervised nature of Hebbian learning, the effects of both random and non-random input on the Hebbian-trainable liquid (Hebbian liquid) are explored in this paper. Section II.A contains a description of how Hebbian Learning is used to mold the neural microcircuit in this paper. Section II.B defines a method of measuring the effectiveness of neural microcircuits used as liquids for LSMs. Section III reports on two experiments: the effects of random and non-random input on liquid structure. Finally a discussion of the results is presented in section IV.

II. Methods

A. Hebbian Learning and STDP Synapses

Hebbian learning is often implemented in neural microcircuits with STDP synapses (spike-time-dependant plasticity synapses). As with other spiking synapses there is a weight and time delay associated with the synapse. In addition, the STDP synapse has several other parameters that are related to how its weight changes as its pre- and post-synaptic neurons fire [3]. The synapse's weight changes in

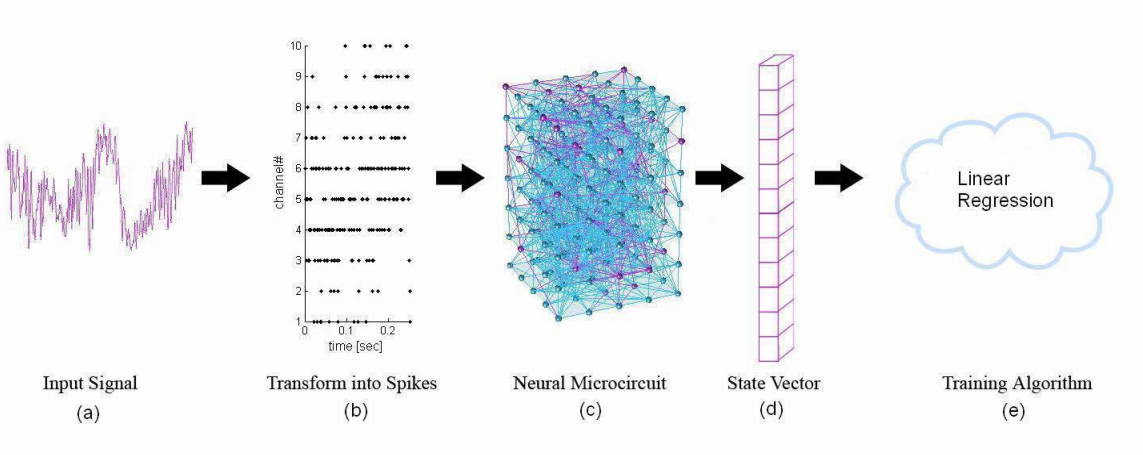

Fig. 1 Diagram of a liquid state machine. (a, b) The input signal is transformed into a series of spikes via some function. (c) The spike train is introduced into the neural microcircuit, or "liquid". (d) Snapshots of the state of the "liquid" are recorded in state vectors. (e) The state vectors are used as input to train a learning algorithm, the reading function.

proportion to the temporal correlation between the pre- and post-synaptic neurons. If the pre-synaptic neuron fires first, then the weight is increased; if the post-synaptic neuron fires first then the weight is decreased. In this way, synapses that participate in a neuron's breach of threshold (resulting in a spike) are strengthened while those that don't are weakened [1]. We will refer to an LSM that uses STDP synapses as a Hebbian Liquid State Machine (HLSM).

B. *Separation*

The effectiveness of an LSM is a function of two qualities: the *approximation* and the *separation* of the LSM. Approximation refers to the reading function's ability to classify the state vectors acquired from the liquid. Separation refers to "the amount of separation between trajectories of internal states of the system that are caused by two different input streams" [5]; or in other words, the ability of the liquid to produce discernibly different patterns when given different classes of inputs. Since any machine learning algorithm can be used for the reading function, this paper focuses on improving the separation attribute of the LSM.

In order to measure the separation of a liquid we use the following definition presented first by Goodman [6]:

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

where Ψ is a neural microcircuit (or liquid), O is a set of state vectors, and N is the total number of output classes represented by O. O is divided into N subsets each of which contains all elements of O belonging to a common output class. The center of mass, C_m for each subset is calculated as follows:

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

The Euclidean distance between every pair wise combination of center of mass vectors is determined. Each center of mass is the estimate of the representative state vector for each output class. The ideal liquid would separate the state vectors for each output class into distinct clusters within the vector space. Goodman's research indicates that there is a positive correlation between this separation measure and the accuracy of the LSM [6].

There are certain negative behaviors common in LSMs that can significantly decrease their separation. These behaviors are termed *pathological synchrony* and *over-stratification*. Pathological synchrony occurs when most of the neurons in the liquid get caught in infinite positive feedback loops with respect to their firing. These infinite loops continuously influence the state of the liquid overriding the flow of information from the input. In extreme cases the entire liquid can begin firing continuously in synchrony (see Fig. 2). Such liquids have low separation because of the loss of pattern associated with such crowded spiking.

The opposite extreme is over-stratification—when groups of neurons do not propagate a series of spikes induced by an input spike long enough. In these cases, input spikes do not influence each other within the liquid, thus resulting in a loss of temporal coherency that can be represented by the liquid (see Fig. 3). Since both of these features are detrimental to the effectiveness of LSMs, this study emphasizes eliminating them.

III. Results

Two experiments were performed in order to observe how Hebbian learning affects the structure and separation of liquids. Random input was introduced into the liquid for the first experiment while speech data was used as input for the second experiment. Also, Hebbian learning was compared

to random weight updates in the first experiment.

A. *Effects of Random Input on HLSMs*

For this experiment, the effect of a single channel of random input on an HLSM's separation was tested in order to better understand the dynamics of the HLSM architecture. Two initial liquid states were explored, a pathological synchrony state and an over-stratification state. These initial states were selected in order to observe how Hebbian Learning could potentially recover from them. For each initial state, actual Hebbian learning versus random weight updates were compared. Each of these four experiments employed 100 iterations of training of the liquid. This training was either Hebbian learning or random weight update. For each iteration of training the separation of the liquid was determined with a set of state vectors, *O*, of size 100. Each state vector in *O* was created by introducing a randomly generated train of 25 spikes over a 1.0 second time interval, *d*, as input into the liquid. The state vector was measured at time *d* with ε = 1.0 ms. Since the input was random, each state vector belonged to a unique output class. Thus, for this experiment $|N| = |O|$ (see section II.B above).

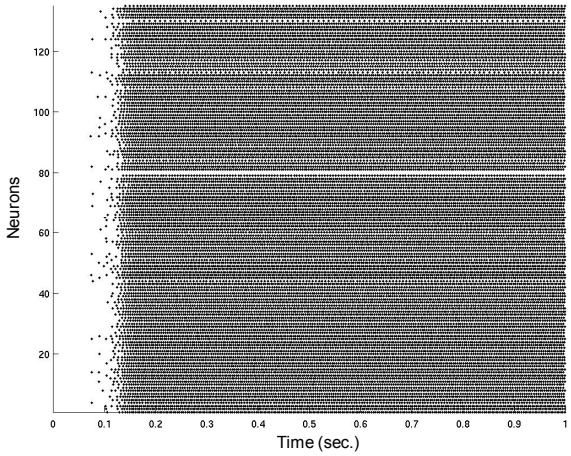

Fig. 2 Pathological Synchrony. This occurs when most of the neurons in the liquid get caught in infinite positive feedback loops with respect to their firing. Dots indicate the presence of a spike.

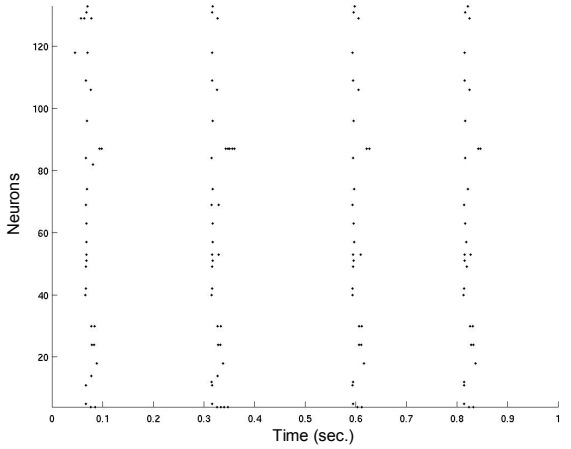

Fig.3 Over-Stratification. The x-axis is time from zero to one seconds. The y-axis contains neuron identification numbers. Dots indicate spikes

Each liquid was prepared with 135 neurons to be comparable to previous research [7]. The input was encoded as a spike train from a single input neuron. Future research will expand the number of input neurons used. The remainder of the settings were chosen based on a series of preliminary experiments and reflect the best results obtained in those trials. The connection probability from the input neuron to the other inter-neurons was 0.1 while the probability of connection between the inter-neurons was 0.05. The mean delay for the synapses of the liquid was 10 ms with a standard deviation of 1 ms. For the liquids initiated in a pathological synchrony state, the mean weight value for synapses was set at 1×10^{-7} while the mean weight value in liquids initiated in an over-stratification state was 8×10^{-8}. The standard deviation of weight values for both initial states was 1×10^{-8}.

For the Hebbian learning, all STDP Synapse settings were selected based on preliminary experiments. The maximum weight value allowed for all synapses in both Hebbian learning and random weight update experiments was 1×10^{-5}. Each training of the liquid involved introducing a randomly generated train of 25 spikes over a 1.0 second time interval. This is identical to when separation data is collected except that the weights are now allowed to change in accordance with the STDP synapse rules outlined earlier.

For random weight updates, each synapse's weight was updated by a value drawn from a normal distribution with a mean of -2.8742×10^{-8} and a standard deviation of 1.5726×10^{-7}. This mean and standard deviation were obtained by calculating and averaging the mean and standard deviation of weight changes in ten preliminary runs through unique liquids using Hebbian Learning. Thus, the values, though random, represent reasonable changes in weight for the liquids in this study.

The results of the above experiment are seen in Figure 4. For each of the four experiments, the average separation of ten unique liquids is displayed. The Hebbian learning trials don't show a significant change in separation while the random weight update trials drop steadily in separation after only ten iterations.

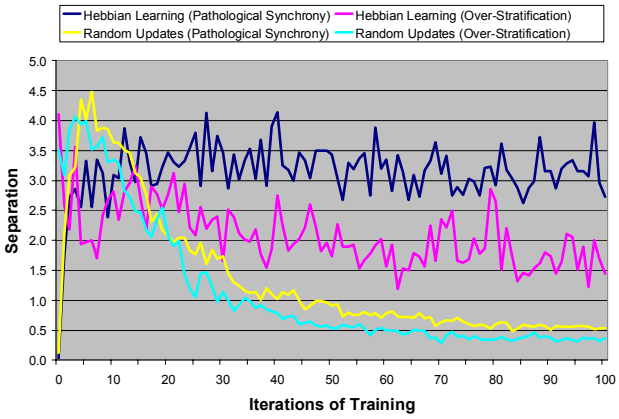

Fig. 4 Separation values for four experiments given completely random input. Separation values are the mean reported by ten trials each with a different initial liquid. The Hebbian learning trials don't show a significant change in separation while the random weight update trials show a steady drop in separation after only ten iterations.

Figure 5 demonstrates how the physical structure of the liquid changes with training. The images show how the synapses (lines) connect to each of the neurons (dots). The brighter the synapse, the stronger the magnitude of the weight. We don't differentiate between positive and negative weights. Black synapses have effectively zero weight. When stimulated with random input, Hebbian learning eliminates many of the synapses while strengthening those that remain. Random weight updates, on the other hand, results in overall significantly strengthened synapses after random stimulation.

Figures 6 and 7 demonstrate how the spiking patterns of each experiment change with training. For each graph, the *x*-axis represents time in seconds and the *y*-axis the neuron ID number (there are 135 total). Hebbian learning relieves the state of pathological synchrony as seen in the reduction of firing (Fig. 6). It also overcomes over-stratification by generating denser firing patterns. Random weight updates results in over-stratification regardless of the initial state (Fig. 7). This seems unusual since the synapses are much stronger according to the results in Figure 5. This occurs because most of the synapses become strongly inhibitory due to the mean negative weight update.

B. *Effects of Non-Random input on HLSMs*

For this experiment, the effects of non-random input on an HLSM's separation were tested in order to predict how the HLSM may behave as a complete machine learning architecture. The input for this experiment was a selection of 3519 training files from the TIDIGIT dataset [8]. These files consist of different people speaking single digits: one through nine, and zero and 'oh' (both for 0). To convert the files into spike trains, all silence was removed from the sound files, and they were converted into thirteen Mel frequency cepstral coefficients (mfcc) as is common in speech applications [9]. The frame size for the Fourier transform was 256 bytes with a frame step of 128 bytes. Thirteen input neurons were used, one for each of the thirteen mfcc's. The firing rate of each of these neurons was determined with the following equation taken from [6]:

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

where Ω represents the largest frequency for a given mfcc, ω represents the smallest frequency, and t is the time interval of a given firing rate, determined by the frame step.

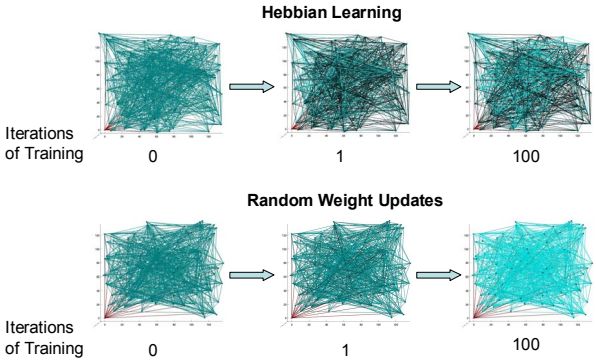

Fig. 5 The physical characteristics of the liquid can be seen before and after training. Bright colors indicate strong weights (either negative or positive), dark colors indicate weak weights. Top: Hebbian learning eliminates many of the synapses while strengthening those that remain. Bottom: Random weight updates results in overall significantly strengthened synapses after random stimulation.

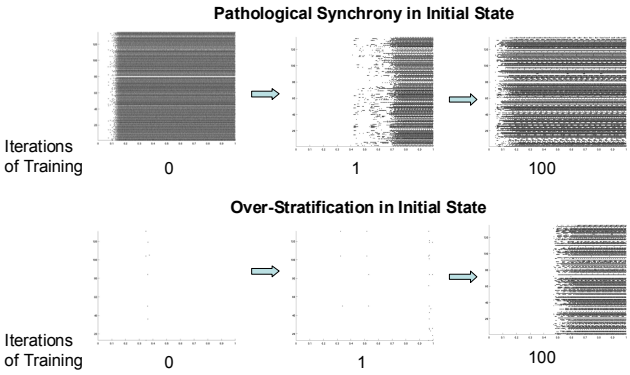

Fig. 6 Spiking patterns in liquids trained with Hebbian learning. Top: the pathological synchrony state of the liquid is somewhat relieved by Hebbian learning—there are fewer neurons firing continuously and less dense patterns of firing. Bottom: Over-stratification is clearly relieved by iteration 100 through the Hebbian process.

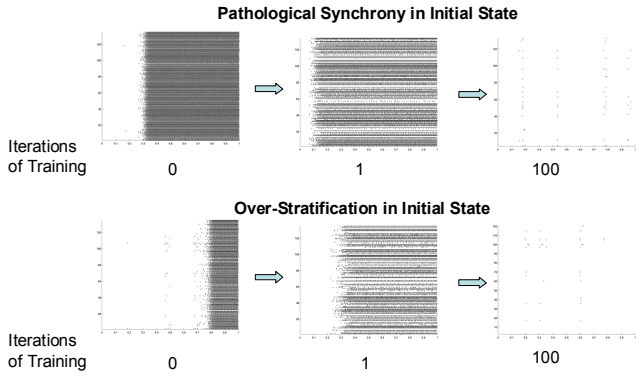

Fig. 7 Spiking patterns in liquids trained with random weight updates. Random updates to synapse weights results in over-stratification over time regardless of the initial state of the liquid.

The separation of the liquid was calculated before and after 1000 iterations of training on the TIDIGIT training dataset. The training dataset contained 3519 files, 1000 of which were randomly selected to train the liquid. To calculate separation, a set of state vectors, O, of size 100 was used as in the previous experiment. In this case each state vector of O was created by introducing one of 3519 randomly selected test files from the TIDIGIT testing dataset as input into the liquid. This test data was different from the training data but was prepared for the HLSM in the same fashion. Each file had a unique time interval, d. The state vector was measured at time d for each file with $\varepsilon = 1.0$ ms. In order to allow for an exhaustive permutation correlation test, the 100 test samples chosen before and after the training were identical.

This experiment was run ten times on different liquids with the results indicated in Figure 8. The average improvement in separation for all ten trials was 0.064 and is statistically significant with a p-value of <0.001. This p-value was calculated by finding the average difference in separation for every permutation of differences for the 10 trials. A single permutation consisted of swapping the order of the difference calculation (pre-training - post-training rather than post-training - pre-training) for a single trial. The number of permutations with averages greater than 0.064 was tabulated and divided by the total number of permutations, 1024, to yield the given p-value. Unsupervised Hebbian learning can improve the separation of the liquid indicating a strong likelihood that the organization of the liquid has become a more effective component for learning.

IV. Discussion

The experiment introducing random inputs into the liquid showed that given ideal initial liquids, Hebbian learning cannot improve separation with random input. However, it showed that given poor initial conditions it can improve performance. It also showed that even under the fabricated random input scenario, Hebbian learning does more than simply randomly update weights. The experiment introducing speech data into the liquid showed that Hebbian learning can improve separation in the liquid given non-random input.

A. Random Input Experiments

In the experiments initiated in a pathological synchronous state, both random update and Hebbian learning improved the separation of the liquids dramatically after only a single iteration of training. In fact, in the initial pathological state, the separation was zero in all trials. This dramatic improvement can best be explained by the assumption that arbitrary pruning of synapses reduces the number of infinite loops in the liquid. This also concurs with previous findings that investigated the reduction of neuron inter-connections to reduce synchronous firing [6].

Other than the initial improvement in pathological states, Hebbian learning doesn't improve the separation of the liquid over successive iterations of training. Also, the amount of separation at each level of training fluctuates greatly. The overall lack of improvement is likely due to the fact that the input for the training is entirely random—the input is effectively noise. While it is clear that the effectiveness of the liquid is not lessened by this noise, there is no useful structure in the data.

In the experiments using random weight update training, after the initial increase in separation, we see a steady decline in separation, until it levels off close to zero. The change in spiking patterns indicate that the patterns become over-stratified (see Fig. 7) explaining the poor results. The primary benefit of these random weight update experiments is that through comparison, we can see that Hebbian Learning performs a role beyond random weight changing, even when confronted with nothing but noise.

B. Non-random Experiments

Fig. 9 and 10 show how the physical structure of the liquid changes with non-random input from a speech recognition task. Notably, very little does change in comparison to the experiments with random input. The noticeable change is that a few connections are greatly strengthened. These figures were representative of all ten trials. More interesting was the improvement in separation noted after Hebbian learning took place, demonstrating that unsupervised learning can improve separation in complex neural microcircuits. The new question raised is whether or not the improvement comes at a lower cost than simply creating an effective liquid to begin with. The effectiveness of the original liquid is a product of all of the

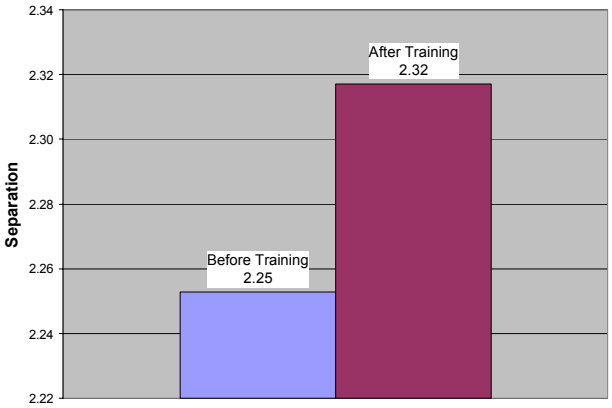

Fig. 8 Average separation in liquid before and after training on the TIDIGIT dataset.

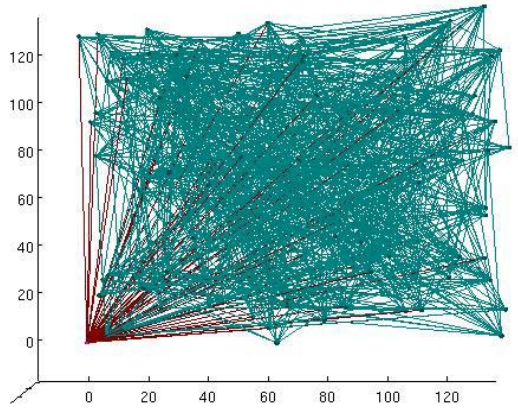

Fig. 9 The network connections of a liquid prior to training with Hebbian learning. The brighter the color of the connection, the stronger the weight of the synapse.

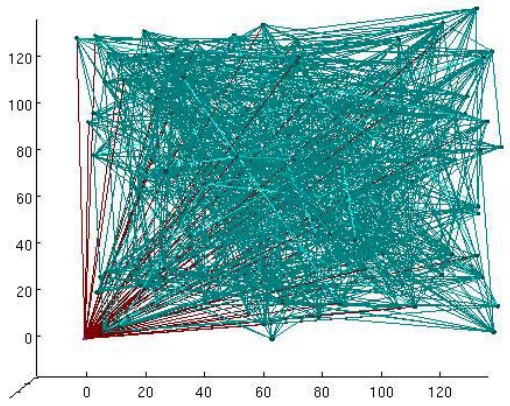

Fig. 10 The network connections of a liquid after 1000 iterations of training on the TIDIGIT dataset using Hebbian learning. Note that there are several bright connections that were not present prior to training corresponding to synapses strengthened by Hebbian learning.

parameters used to create it, including mean weight, probability of inter-neuron connections, mean delay values, etc. Separation values for the pre-training liquids ranged from 2.09 to 2.70, a much greater difference than the average difference between pre- and post-training separation. Also, the effectiveness of the Hebbian learning is sensitive to initial parameter settings (with different settings for the parameters used in the Hebbian learning, the post-training liquids actually resulted in lower separation). It is unclear whether discovering the ideal settings for Hebbian learning is less difficult then the effort required discovering the ideal settings for the initial liquid. It is also uncertain whether Hebbian learning or any other post-parameter-setting adjustments provide a gain in separation that is not available in the parameter setting stage. Future work will investigate the possibility of separation gain exclusive to learning in the liquid.

Other possible avenues for further research include experimenting with different (liquid) training algorithms (e.g some variant of back-propagation through time [10] or reinforcement learning) and exploring theoretical bounds for separation. Preliminary studies indicate that the separation values presented in this paper are low compared to the ideal separation, given the values of $|O|$ and $|N|$ used here.

REFERENCES

[1] W. Gerstner, W.M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.

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

[3] T. Natschläger, Neural Micro Circuits, http://www.lsm.turgraz.at/index.html, 2005.

[4] E. L. Goodman, D. Ventura, "Effectively Using Recurrently Connected Spiking Neural Networks", in *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Canada, August 2005.

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

[6] E. L. Goodman, D. Ventura, "Spatiotemporal Pattern Recognition Via Liquid State Machines", in *Proceedings of the International Joint Conference on Neural Networks*, these proceedings, 2006.

[7] E. L. Goodman, D. Ventura, "Time Invariance and Liquid State Machines", in *Proceedings of the 8th Joint Conference on Information Sciences*, Salt Lake City, UT, July 2005.

[8] R.G. Leonard and G. Doddington, *TIDIGITS speech corpus*, http://morph.lds.upenn.edu/Catalog/LDC93S10.html, Texas Instruments, Inc., 1993.

[9] ETSI ES 202 212—STQ: DSR, "Extended advanced front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction algorithm", *European Telecommunications Standards Institute (ETSI)*, 2003.

[10] Paul J. Werbos, "Backpropagation Through Time: What it Does and How to Do It", in *Proceedings of the IEEE*, vol. 78, no. 10, October 1990.