Stability and Topology in Reservoir Computing

Larry Manevitz, Hananel Hazan

Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel
[manevitz,hhazan01]@cs.haifa.ac.il

Keywords: Reservoir Computing, Small world topology, robustness, Machine Learning.

Abstract. Recently Jaeger and others have put forth the paradigm of "reservoir computing" as a way of computing with highly recurrent neural networks. This reservoir is a collection of neurons randomly connected with each other of fixed weights. Amongst other things, it has been shown to be effective in temporal pattern recognition; and has been held as a model appropriate to explain how certain aspects of the brain work. (Particularly in its guise as "liquid state machine", due to Maass et al.) In this work we show that although it is known that this model does have generalizability properties and thus is robust to errors in input, it is NOT resistant to errors in the model itself. Thus small malfunctions or distortions make previous training ineffective. Thus this model as currently presented cannot be thought of as appropriate as a biological model; and it also suggests limitations on the applicability in the pattern recognition sphere. However, we show that, with the enforcement of topological constraints on the reservoir, in particular that of *small world* topology, the model is indeed fault tolerant. Thus this implies that "natural" computational systems must have specific topologies and the uniform random connectivity is not appropriate.

1 INTRODUCTION

Recently Jaeger (Jaeger, "The ëcho state" approach to analysing and training recurrent neural networks, 2001), Maass (Maass, Natschläger, & Markram, 2002) and others have put forth the paradigm of "reservoir computing" as a way of computing with highly recurrent neural networks. This reservoir is a collection of neurons randomly connected with each other of fixed weights. Amongst other things, it has been shown to be effective in temporal pattern recognition; and has been held as a model appropriate to explain how certain aspects of the brain work. (Particularly in its guise as "liquid state machine", due to Maass et al.)

This is particularly impressive as processing in artificial neurons typically is a-temporal. This is because the underlying basic neuronal model, that of McCullough-Pitts (McCullough & Pitts, 1943) is atemporal by nature. As a result, most applications of artificial neural networks are related in one way or another to static pattern recognition. On the other hand, it has long been recognized in the brain science community that the McCullough-Pitts paradigm is inadequate. Various models of differing complexity have been promulgated to explain the temporal capabilities (amongst other things) of natural neurons and neuronal networks.

However, during the last decade, computational scientists have begun to pay attention to this issue both from the neurocomputational and biological perspectives e.g. (Maass W. , 1999; Maass, Natschläger, & Markram, 2002; Maass, Natschlger, & Markram, 2004; Fernando & Sojakka, 2005; Jaeger, The "echo state" approach to analysing and training recurrent neural networks, 2001), and investigations as to the computational capabilities of various models are being investigated.

Two such models, the “echo state machine” (Jaeger, The "echo state" approach to analysing and training recurrent neural networks, 2001) and the “Liquid State Machine” (see *Fig. **1.***) (Maass, Natschlger, & Markram, 2004; Maass, Natschläger, & Markram, 2002), have had substantial successes recently. These two models are identical on the abstract level and have recently been renamed “reservoir computing” (Lukosevicius & Jaeger, 2009). In these models there is a somewhat different paradigm of computation. Information is stored, not in "attractors" as is usually assumed in recurrent neural networks, but in the reverberating activity pattern in a sufficiently recurrent and inter-connected network. This information can then be retrieved by any sufficiently strong classifying detector. The idea is that the history of, e.g. timings of rocks thrown into a pond of water, is completely contained in the wave structure.) Moreover, the "persistence of the trace" (or as Maass put it, the "fading memory") allows one to recognize at a temporal distance the signal that was sent to the liquid; and sequence and timing affects of inputs.

This is an exciting idea; and, e.g. Jaeger, Maass and his colleagues have published a series of papers on it. Amongst other things, they have recently shown that once a detector has been sufficiently trained at any time frame, it is resilient to noise in the input data; and so it can be used successfully for generalization. (Maass, Natschläger, & Markram, 2002; Fernando & Sojakka, 2005). In particular, experiments have been performed for speech recognition.

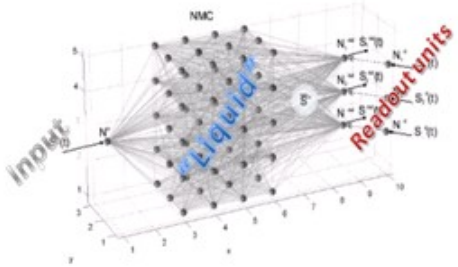

Fig. 1. Liquid State Machine (*figure* taken from (Maass, Natschläger, & Markram, 2002))

However, there is a claim that this abstraction is faithful to the potential capabilities of the natural neurons and thus is explanatory to some extent from the viewpoint of computational brain science. It is this issue we address in this paper. Note that one of the underlying assumptions is that the detector works without memory; that is the detector should be able to classify based on instantaneous static information; i.e. by sampling the liquid at a specific time. That this is theoretically

possible is the result of looking at the dynamical system of the liquid and noting that it is sufficient to cause the divergence of the two classes in the space of activation.

Note that the detector systems (e.g. a back propagation neural network, a perceptron or an SVM) are not required to have any biological plausibility; either in their design or in their training mechanism, since the model does not try to account for the way the information is *used* in nature.

Despite this, since natural neurons exist in a biological and hence noisy environment, for these models to be successful in this domain, they must be robust to various kinds of noise. As mentioned above (Lukosevicius & Jaeger, 2009) (Maass, Natschläger, & Markram, 2002) addressed one dimension of this problem by showing that the systems are in fact robust to noise in the input. Thus small random shifts in a temporal input pattern will not affect these models to recognize the pattern. From a machine learning perspective, this means that the model is capable of generalization.

However, there is another component to robustness; that of the components of the system itself.

In this paper we report on experiments performed with various kinds of "damage" to these models and unfortunately have shown that, e.g. the LSM with any of the above detectors is *not* resistant, in the sense that small damages to the LSM neurons reduce the trained classifiers dramatically, even to essentially random values.

Seeking to correct this problem, we experimented with different architectures of the liquid. The essential need is that there should be sufficient recurrent connections so that on the one hand, the network maintains the information in a signal, while on the other hand it separates different signals. The models typically used are random connections; or those random with a bias towards "nearby" connections. Our experiments with these topologies show that the network is very sensitive to damage because the recurrent nature of the system causes substantial feedback.

Taking this as a clue, we tried networks with "hub" or "small world" (Albert-László & Réka, 1999; Albert-László & Réka, 1999; Bianconi G, 2001; Albert R, 2000) architecture. This architecture has been claimed (Danielle & Bullmore, 2006; Chklovskii, 2009) to be "biologically feasible".

The intuition was that the hub topology, on the one hand, integrates information from many locations and so is resilient to damage in some of them; and on the other hand, since such hubs follow a power rule distribution, they are rare enough that damage usually does not affect them directly. This intuition was in fact borne out by our experiments.

2 Non Robustness of the Models

2.1 The experiments

To test this resistance to noise, we downloaded the code of Maass et al from his laboratory site[1] and then implemented two kinds of damage to the liquid. We also

[1] A neural *C*ircuit *SIM*ulator: http://www.lsm.tugraz.at/csim/

reimplemented the LSM code so that we could handle variants. These models use a kind of basic neuron that is of the "leaky integrate and fire" (LIF) variety and in Maass' work, the neurons are connected randomly. In addition, some biologically inspired parameters are added: 20% inhibitory and a connectivity constraint giving a preference to geometrically nearby neurons over more remote ones. (For precise details on these parameters, see: neural *C*ircuit *SIM*ulator[1]) External stimuli to the network were always sent to 15% of the neurons, always chosen to be excitatory neurons. Initially, we experimented with two parameters: (i) the percentage of neurons damaged (ii) the kinds of damages.

The kinds were either transforming a neuron into a "dead" neuron; i.e. one that never fires or transforming a neuron into a "generator" neuron ,i.e. one that fire as often as its refractory period allows it, regardless of its input.

We did experiments with different kinds of detectors: Adaline (Widrow & Hoff, 1960), Back-Propagation, SVM and Tempotron (Gütig & Sompolinsky, 2006).

Classification of new data could then be done at any of the signal points; We ran experiments as follows: we randomly chose twenty temporal inputs; i.e. random sequences of 0s and 1s of length 45, corresponding to spike inputs over a period of time; and trained an LSM composed of 240 integrate and fire neurons like in (Maass, Natschläger, & Markram, 2002) to recognize ten of these inputs and reject the other ten, each choice of architecture was run 666 times varying the precise connections randomly.

We tested the robustness of the recognition ability of the network with the following parameters:

- The neurons in the network were either leaky integrate and fire neurons (Maass W. , 1999) or Izhikevich (Izhikevich, 2003) style neurons.
- The average connectivity of the networks was maintained at about 20% chosen randomly in all cases although with different distributions.
- The damages were either "generators," i.e. the neurons issued a spike whenever their refractory period allowed it; or they were "dead" neurons that could not spike.
- The degree of damage was systematically checked at 1%, 2%…15% in randomly chosen neurons.

The results that shown in tables throughout the paper are in percents, over the (666) repeated tests. 100% indicates that all the 20 vectors of one test, over 666 repetitions of the test were fully recognize correctly. 50% indicates that only half the vectors over 666 times were recognized (corresponding to chance baseline).

2.2 Results

First, there was not much difference between the detectors; so eventually we restricted ourselves to the Back-Propagation detector which had data points of 30 randomly sampled time points of the entire liquid. (To be fair, none of units of the liquid input were accessed by the detectors allowed to be input neurons of the liquid.)

It turned out that while the detector was able to learn the randomly chosen test classes successfully with sufficient average connectivity almost any kind of damage

caused the detector to have a very substantial decay in its detecting ability (See *Table* 1.).

Damage	Non	5%	10%
Dead Neurons	90.45%	80.35%	60.3 %
Noisy Neurons	92.01%	59.08%	53.8%

Table 1. Maass's implementation: distribution preferring local connections.

Reimplementing the LSM to allow for different connectivities; showed the same basic responses. In fact, when the network is connected randomly without bias for geometric closeness, the network is even more sensitive (Compare *Table* 1. and *Table* 2.). After our later experiments, we returned to this point (see concluding remarks, below).

Damage	Non	1%	5%	10%
Dead Neurons	100%	53%	53%	50%
Noisy Neurons	100%	55%	53%	52%

Table 2. 10% random connections[2]

In *Fig*. ***2.*** we illustrate the difference in reaction of the network by a raster (ISI) display. Note that with 10% damage, it is quite evident to the eye that the network diverges dramatically from the noise free situation. In *Table* 3. one can see this as well with 5% noise for purely random connectivity. Actually, with low degrees of damage the detectors under even the Maass connectivity show dramatic decay in recognition although not to the extremes of random connectivity. (See *Table* 2.) These results were robust and repeatable under many trials and variants.

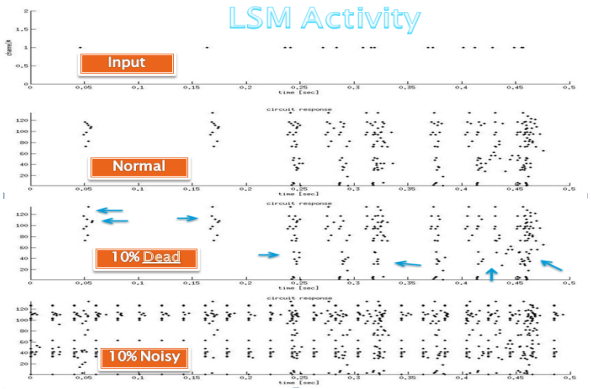

Fig. 2. Maass LSM (a) normal operation (b) with 10% dead damage (c) with 10% noise

Accordingly, we conclude that the LSM, either as purely defined with random connectivity, or, as implemented in (Maass, Natschlger, & Markram, 2004) cannot serve as a biologically relevant model.

[2]For all the Tables that shown in this paper, 50% is the baseline of random classification

3 Topological Modifications and Restoration of Robustness

3.1 Different Kinds of Basic Neurons

In attempts to restore the robustness to damage, we experimented with the possibility that a different kind of basic neuron might result in a more resilient network. Accordingly, we implemented the LSM with various variants of "leaky integrate and fire neurons" e.g. with history dependent refractory period (Manevitz & Marom, 2002) and by using the model of neurons due to Izhikevich (Izhikevich, 2003). The results under these variants were qualitatively the same as the standard neuron. (The Izhikevich model produces a much more dense activity in the network and thus the detector was harder to train but in the end the network was trainable and the results under damage were very similar.).

3.2 Allowing Detectors to Have Memory

In trying to consider how to make the model more robust to damage, we focused first on the fact that the detector has no memory. Perhaps, if we allow the detector to follow the development of the network for some time amount, both in training and running, it would be more robust. To check this, we took the most extreme other case; we assumed that the detector system in fact takes as input a full time course of its input neurons for about 30 iterations. This means that instead of a NN with input of 204; we had one with 30 times 204 time course inputs. It seemed reasonable that (i) with so much information, it should be relatively easy to train the detector (ii) one could hope that damage in the liquid would be local enough that over the time period, the detector could correct for it. In order to test this, we re-implemented the LSM to allow for the time entry.

Our detector was trained and tested as follows. There were 204 output units. At a “signal point” each of them was sampled for the next 30 iterations and all of these values were used as a single data point to the detector. Thus the detector had 204 times 30 inputs. We chose separate detector points typically at intervals of 80. We then used back propagation on these data points. This means that eventually the detector could recognize the signal at any of the “signal points” after training there was no particular importance to the choice of separation of the signal points except that there was no overlap between the data points. While we did not control for any connection between the intervals of data points (i.e. 80, and we also checked other time intervals) and possible natural oscillations in the network, we do not believe there was any. As anticipated, there was no significant trouble in training the network to even 100% of recognition of the training data.

The "detectors" were three level neural networks, trained by back-propagation. We also did some experiments with the Tempotron (Gütig & Sompolinsky, 2006); and with a simple Adaline detector (Widrow & Hoff, 1960). Training for classification could be performed in the damage-less environment successfully with any of these detectors.

We exhaustively ran tests on these possibilities. Some sample results with 5% and 10% damage for the neural network detectors are presented in the *Fig.* ***4.*** Through *Fig.* ***9.*** below. (Since the results for the other detectors were similar, we did not run as many tests on them)

Damage	Non	1%	5%	10%
Dead Neurons	100%	61%	58%	56%
Noisy Neurons	100%	60%	58%	57%

Table 3. 5% random connectivity

In all of these tests, following Maass, we assumed that approximately 20% of the neurons of the liquid were of the inhibitory type. The architecture of the neural network detector was 204 input neurons (which were never taken from the neurons in the LSM which were also used as inputs to the LSM.) 100 hidden level neurons and one neuron for the output. Results running the Maass et al. architecture are presented in Table 1. and can be compared with a random connected network of 20% average connectivity. See Table 4.

Damage	Non	1%	5%	10%
Dead Neurons	100%	79%	49%	49%
Noisy Neurons	100%	97%	71%	63%

Table 4. 20% random connectivity

The bottom line was that even with low amounts damage and under most kinds of connectivity, the networks would fail; i.e. the trained but damaged network loss of function was very substantial and in many cases could not perform substantially differently from a random selection.

3.3 Changing the Architecture

Our next approach, and ultimately the successful one, was to experiment with different architectures. We looked at many variants of the following ideas:

1. Random Connectivity as a baseline. (Note: this is not a "straw dog". This is actually the basic definition of the LSM.)
2. Varying the amount of connectivity. Lowering the average degree of connectivity shows decreased sensitivity in all architectures. Unfortunately, lowering the connectivity also decreases the strength the network has in representability and, importantly, in the persistence of the signal. (That is, a low degree of connectivity causes the activity to die down quickly because of the lack of feedback. Thus the network is bounded in time and cannot recognize an "older" input signal.) Thus we see, as is to be expected from the analysis in (Jaeger, The "echo state" approach to analysing and training recurrent neural networks, 2001; Maass, Natschlger, & Markram, 2004)that a higher connectivity gives a larger set of "filters" that separate signals, but on the other hand makes it more sensitive to changes. In any case, even with low connectivities, the random topology was not robust; nor was the Maass topology. (While not at random levels of identification, as we have seen,

e.g. in *Table* 1. it suffered very substantial decays with even small amounts of damages. In addition, our experiments with connectivities below 15% - 20%, show that the networks do not maintain the trace for very long. (Not shown here.)

3. "Hub" topologies (see *Table* 5.). Here we designed by hand topologies with essentially one hub. In this case, the robustness was substantially increased but the persistence was weak; and under the algorithm chosen, there were substantial disconnected components in the liquid.
4. Small world topologies (see *Table 6.*). In this system the connectivity follows a power rule law. We constructed these networks in different ways. In all cases, the number of connections was chosen based on the average connectivity desired.
5. Assign a link from a uniformly randomly chosen neuron to a second neuron chosen randomly according to a power law. In this case the input connectivity follows a power law; while the output connectivity follows a Gaussian distribution.
6. Reversing the above. In this case the input connectivity is Gaussian while the output connectivity is power law.
7. We also replaced Gaussian with uniform in the above.
8. We also tried choosing a symmetric network with Power law connectivity (ipso facto for both input and output). (Note that in this case, the *same neurons* served as "hubs" both for input and output.)
9. Finally, we designed an algorithm to allow distinct input and output connectivity but both obeying the same power law. (See algorithm1 and algorithm 2 below).

Algorithm 1: Generate a random number between min and max value with Power law distribution

```
Input: min,max, size
How_many_numbers
counterArry = array
Magnify = 5
for  i = 1  to How_many_numbers
 index = random(array.start ,
array.end)
 end_array = array.end
 candidate = array[index]
 AddCells(array , Magnify);
 for t = 0 to Magnify
  array[end_array+t]=candidate
 end for
 shuffle(array)
 output_Array[i] = candidate
 counterArry[candidate]++
end for
shuffle(counterArry)
Output output Array,counterArry
```

Algorithm 2: Create the connectivity matrix for the liquid network using the algorithm 1

```
Input weight_Matrix
use algorithm 1 to creart
(arraylist, counterArry)
counter = 0
for i=1 to
counterArry.lenght
 for t=1 to counterArry[i]
  weight_Matrix[i,
arraylist[counter]]=true
  counter++
 end for
end for
```

One problem with the various algorithms for designing power law connectivity is that under a "fair" sampling, the network might not be connected. This means that such a network actually has a lower, effective connectivity. Since we already knew that lower connectivity results in less sensitivity to noise, we decided to eliminate this problem by randomly connecting the disconnected components (either from an input

or output perspective) to another neuron chosen randomly but proportionally to the connectivity. (This does not guarantee connectivity of the graph, but makes it unlikely, so that the effective connectivity is not substantially affected.)

Damage	Non	1%	5%	10%
Dead Neurons	100%	93%	66.73%	64.09%
Noisy Neurons	100%	98%	84.33%	70.77%

Table 5. *One hub network*

Damage	Non	1%	5%	10%
Dead Neurons	100%	87%	71%	68%
Noisy Neurons	100%	88%	79%	71%

Table 6. Power-law distribution with small worlds

3.4 Results

All architectures with a power law distribution, whether on the input, output or both sides resulted in substantial improvements in the resistance to noise, except for case (8) where the input and output hubs were the same. This was even worse than the random baseline choice at the same connectivity. The best result was obtained in case (9) when both input and output connectivity were power law; but distinctly chosen. *Fig.* ***3*** shows the connectivity distribution in this case. *Table 6.* shows the results in this case for some sample damages in the 20% average connectivity situation.

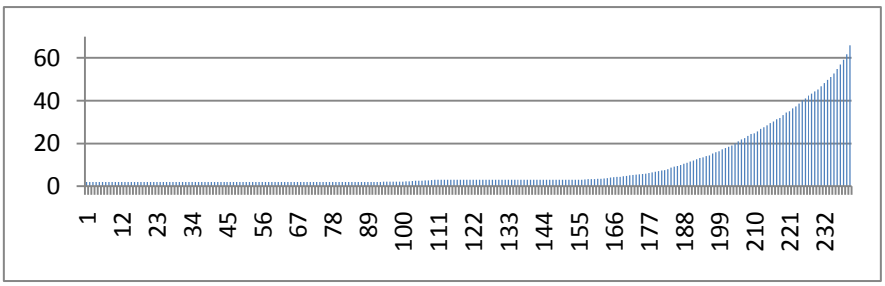

Fig. 3: Connection distribution according to the power-law.

The results presented are the average of many experiments. However, since this work is about robustness, we thought it important to consider the distribution of such results over many experiments. Thus, instead of giving less revealing statistics, we display the complete histograms for the different kinds of networks under different amounts of damages. Note that for all figures and tables 50% is the random baseline.

In *Fig.* ***4.*** through *Fig.* ***9.***, we display the histograms of hundreds of networks at different levels of success under each of the architectures. The horizontal axis is the accuracy of classification and the vertical axis is the histogram count.

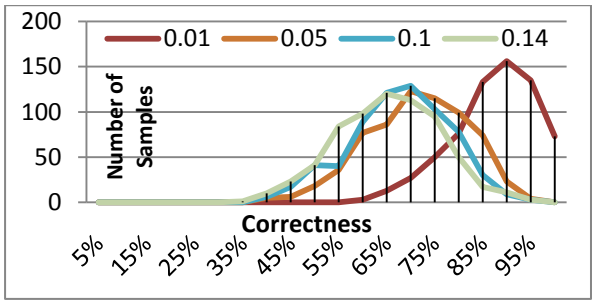

Fig. 4. Histograms of correctness results in liquid networks with different amounts of "dead" neuron damage for liquid networks with an average connectivity of 20% with a power law distribution of connectivity.

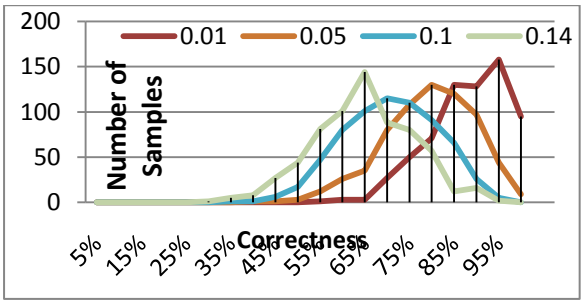

Fig. 5.: Histograms of correctness results in liquid networks with different amounts of "noise generator" neuron damage for liquid networks with an average connectivity of 20%, with connectivity of a power law distribution.

Table 4. , *Fig.* ***6.*** and *Fig.* ***7.*** show the distribution of damage for a random connectivity network with average connectivity of 20%.

Table 5., *Fig.* ***8.*** and *Fig.* ***9.*** show the distribution of damage for one hub network with average connectivity of 20%.

Table 6., *Fig.* ***4.*** and *Fig.* ***5.*** show the robustness results for the power law small word distribution.

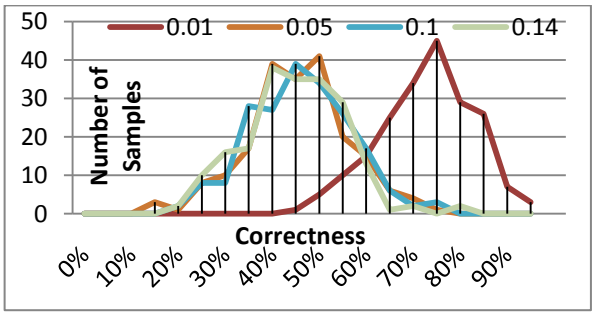

Fig. 6.: Histograms of correctness results in liquid networks with different amounts of "dead" neuron damage for liquid networks with an average connectivity of 20% with a connectivity of random connections distribution.

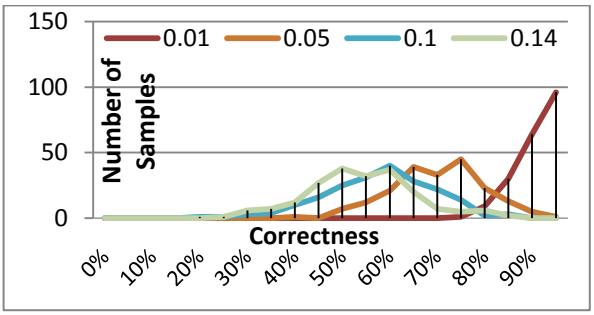

Fig. 7.: Histograms of correctness results in liquid networks with different amounts of "noise generator" neuron damage for liquid networks with an average connectivity of 20% with a random connections[3] distribution of connectivity

[3] For all the *Table*s that shown in this paper, 50% is the baseline of random classification

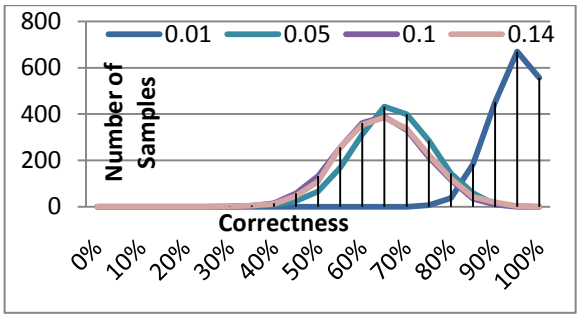

Fig. 8. Histograms of correctness results in liquid networks with different amounts of "noise generator" neuron damage for liquid networks with an average connectivity of 20%, with distribution of one hub.

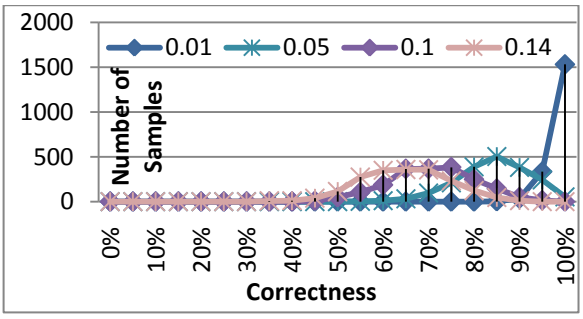

Fig. 9.: Histograms of correctness results in liquid networks with different amounts of "dead" neuron damage for liquid networks with an average connectivity of 20%, with distribution of one hub.

4 Discussion

We have shown experimentally that the basic LSM is not robust to "damages" in its underlying neurons and thus without elaboration cannot be seen to be a good fit for a model for biological computation. We mention (data not shown here) that this result holds even if training is continued while the network is suffering damage. However, choosing different distributions of the connectivity can result in more robust maintenance of the pertinent information over time.

In the papers (Danielle & Bullmore, 2006; Chklovskii, 2009), a distribution was chosen for biological reasons to allow preference for close neurons. This distribution is superior to the totally random one, but is still not sufficiently robust. Choosing a power law distribution and being careful to making the assignments differently for in and out connectivity proved to be the best. This is thought of as a potentially biological arrangement (Danielle & Bullmore, 2006; Albert-László & Réka, 1999); so LSM style networks with this additional topological constraint can, as of this date, are considered sufficiently biological. Other distributions may also work.

5 Bibliography

Albert R, B. A.-L. (2000). Topology of evolving networks: local events and universality. *85:5234–7*.

Albert-László, B., & Réka, A. (1999). Emergence of Scaling in Random Networks. *SCIENCE , 286*, 509 - 512.

Bianconi G, B. A.-L. (2001). Competition and multiscaling in evolving networks. *Europhys Lett* , 54:436.

Chklovskii, L. R. (2009, Jul). Structural Properties of the Caenorhabditis elegans Neuronal Network.

Danielle, B. S., & Bullmore, E. (2006). Small-world brain networks. *Neuroscientist , 12(6)*, 512-523.

Fernando, C., & Sojakka, S. (2005). Pattern Recognition in a Bucket. In *Advances in Artificial Life* (Vol. 2801/2003, pp. 588-597). Springer Berlin / Heidelberg.

Gütig, R., & Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decision. *Nature Neuroscience , 9(3)*, 420-428.

Izhikevich, E. M. (2003). Simple Model of Spiking Neurons. *IEEE TRANSACTIONS ON NEURAL NETWORKS , 14* (6), 1569-1572.

Jaeger, H. (2001). *"The ëcho state" approach to analysing and training recurrent neural networks.* German National Research Institute for Computer Science.

Jaeger, H. (2001). *The "echo state" approach to analysing and training recurrent neural networks.* German National Research Institute for Computer Science.

Lukosevicius, M., & Jaeger, H. (2009). Reservoir Computing Approaches to Recurrent Neural Network Training. *Computer Scinece Review* , 127-149.

Maass, Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation , 14* (11), 2531-2560.

Maass, W. (1999). *Paradigms for computing with Spiking Neurons.* Springer.

Maass, W., Natschlger, T., & Markram, H. (2004). Computational models for generic cortical microcircuits. In J. Feng, *Computational Neuroscience: A Comprehensive Approach* (pp. 575-605). Boca Raton: Chapman & Hall/CRC.

Manevitz, L. M., & Marom, S. (2002). Modeling the process of rate selection in neuronal activity. *Journal of Theoretical Biology , 216(3)*, 337–343.

McCullough, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics , 5*, 115−127.

Widrow, B., & Hoff, M. (1960). Adaptive Switching Circuits. *IRE WESCON Convention Record , 4*, 96-104.