Problem-Oriented Analysis of Self-Organizing Neural Networks

Vins Sharma
Electrical and Computer Engineering
Carnegie Mellon University
vmsharma@andrew.cmu.edu

Anand Raju
Electrical and Computer Engineering
Carnegie Mellon University
amraju@andrew.cmu.edu

***Abstract*—Liquid state machines (LSMs) are machine learning models that create more biologically plausible neural networks by self-organizing neurons into a 'liquid'. In this project we aim to analyze a variety of 'liquids' created through the MNIST dataset in order to determine correlations and patterns that exist in these graphs. We intend to utilize this information to draw conclusions about the liquid's training model of Spike-Timing-Dependent Plasticity (STDP), as well as understand more about liquid state machines themselves. We accomplish this by: (1) Implementing a framework to generate LSMs on particular datasets and configurations, and (2) Analyzing generated LSMs for overlapping patterns and similarities.**

***Index Terms*—liquid state machines, temporal neural networks, random graph models, reservoir computing**

I. Introduction

A. Motivation

The portion of the brain primarily responsible for human thought and cognitive function is thought to be the neocortex, which covers the outside shell. The unfolded neocortex is the size of a dinner napkin, and is made up of tiny perpendicular 'micro-fibers' called cortical columns [1].

Cortical columns are thought to be the microcircuit that implements all cognitive thought. Though it is well documented that different regions of the brain are responsible for different functions, it has been shown [2] that the makeup of the cortical column is nearly identical across the entire neocortex, excluding the inputs. This indicates that a circuit exists that can be plugged into different problems and solve all of them without significant modification or significant power consumption.

One approach to designing such a 'perfect' circuit is through induction. In section II, we'll discuss this approach in more detail, and why we are not choosing it. However, another approach is from a top down - We hope to generate networks from a variety of configurations, look at these networks, and attempt to understand what similarities exist between them or how different parameters and problem-specific features affect their design.

B. General Approach

We start by generating a framework to quickly and easily 'plug in' different configurations and generate networks. These networks can organize themselves based on features of the data, and we can analyze how they organize themselves over time simulations.

With this framework we analyze a variety of unique input configurations to understand how the network constructs itself. Neurons within the liquid may reconfigure as they so choose, based closely on their learning algorithm. We hope to derive some insight into how the network rearranges itself based on the input data. These similarities and conclusions may serve as the first steps in constructing a cortical column-like neural circuit.

C. Potential Impacts

While we do not think we can architect such a circuit as exists in the human brain, we hope to gain some significant insight into how such a circuit might be formed based on neuronal dynamics and connectivity.

Several studies have previously looked at brain connectivity from a graph theoretical viewpoint [3] and have made only passing comments about the evolution of brain circuitry to handle particular problems. It has been posited that the brain forms itself to decrease axonal connection cost from a structural standpoint. However, we choose to look closely at the functional side of neural circuit development throughout this project. We hope to demonstrate (in particular) patterns generated from functional connectivity and analyze how they may be related to the problem being solved.

II. Previous Work

In this section we examine inspirations for our approach, as well as motivations for our choices. We pick out five papers in particular that we draw from in our implementation.

The first paper, [5], demonstrates the alternate route of pursuing such a cortical column circuit. The second paper [4] describes the fundamental building blocks and neuronal models for either of these approaches. The third paper [6] delves into the encoding of data fitting for these models. The fourth [7] and fifth [9] papers combine to describe the reservoir computing model, as well as some particulars we test throughout our implementation.

A. Smith (2023)

As discussed in I, the goal of this project is to get closer to a cortical column-like neural circuit. Professor Smith's rendition of his macrocolumn in [5] is an example of the ground-up approach to solving this problem. He proposes a circuit specialized towards one task - Having a mouse navigate a maze. Future work in this project involves applying a slightly modified circuit to another problem.

One major issue with this avenue of thought is the exponentially increasing amount of effort. In future work, when introducing a new problem, Professor Smith now has to modify the circuit to solve the new problem while still preserving the old functionality. As such, we aim to avoid this approach, and instead take a top-down approach to generalize further.

B. Nair, Shen, Smith (2021)

This paper [4] describes the temporal neuron and a strategy for training it in an unsupervised way. The temporal neuron simply encodes data inputs on spike timing lines (in a process known as rate coding). This allows it to consume significantly less power (as the spike is only on instantaneously), and also allows for a non-statistical simple unsupervised approach.

For our approach, we implement these temporal neurons, but use them without the paper's described 'columnar' structure (for the most part). As such, our usage of temporal neurons is novelly applied in a more arbitrary network structure.

C. Purdy (2016)

Purdy's paper [6] discusses the best-practices for encoding data in temporal systems, such as we will throughout this project. He focuses on the generation of proper sparse distributed representation (SDR) matrices as good-for-encoding inputs. We apply this knowledge to adapt datasets to temporal input.

D. Maass (2011)

Maass defines the liquid state machine (LSM) [7] model for reservoir computing. The LSM has a reservoir with a series of spiking neurons that form some arbitrary excitation pattern; A separate set of (trained) neurons then read out this pattern and correlate it to an output. The internals of the reservoir can be structured in any way, but are conceptually similar to self-organizing maps [8].

Kohonen posits, through his work, that arbitrary higher dimensional features of a problem are encoded into the graph's organization for a self-organizing map. As such, we analyze the organization of the reservoirs in order to draw possible conclusions about the features of the problem.

E. Hazan, Manevitz (2012)

Hazan and Manevitz [9] propose in their paper that the LSM previously defined by Maass is not a good model due to its network's robustness, and that a more robust model would better represent the brain. They draw a new conclusion towards small-world assumptions in graph topology being key to solving this problem. We aim to test this hypothesis throughout our work, and pay close attention to the topologies constructed out of small-world model configurations for our reservoirs.

III. Approach

A. Dataset

Throughout our project we apply the MNIST dataset [13]. This dataset requires the tools of image recognition and classification, and is very popular in generic machine learning projects.

As we are utilizing a controlled dataset for all models, any similarities or differences in constructed graph topologies are fundamentally responses to the individual parameter changes, the network structure, or the act of 'problem-solving' itself.

B. Assumptions

While we vary a variety of parameters, we assume that the underlying temporal neurons function as a strong analogue for biological processes. This assumption allows us to generalize our results to neural basis of cognition rather than design of artificial neural networks.

C. Analysis

The focus of analysis is the network generated within the liquid reservoir. This liquid has nodes of neurons and edges of weighted synapses.

We test the robustness claim of [9] by analyzing the connectivity of these networks over their growth. We also analyze diameter in order to understand the 'distance' between neurons in the network (and the amount of time it takes for a spike to propagate).

D. Research Questions

Through these network features, and more, we aim to understand the following:

1) Why did the network configure itself the way it did?
2) How do networks of different configurations compare to each other?

IV. Network Configuration

Within this section we introduce our system. We define a temporal liquid state machine (TLSM) as an input vector, encoded in time-based spikes, followed by a reservoir of temporal neurons, and finally a singular output column (with winner-takes-all lateral inhibition). The input vector uses encoding strategies discussed in [6], and the output column is identical to the one described in [4].

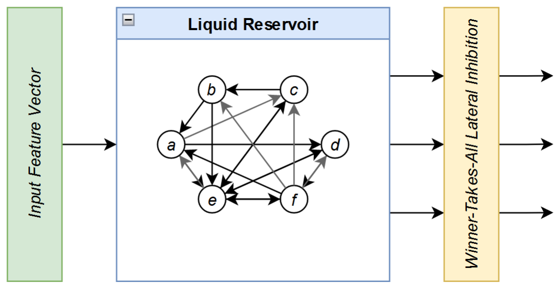

Fig. IV.1. Diagram of a temporal liquid state machine (TLSM).

A. Input Encoding

The input vector is a series of time-encoded one-hot vector spikes. For MNIST data, the input 28×28 image is flattened into a 784×1 vector. Each element of the vector is subsequently expanded into a $t_{res} \times 1$ one-hot time-spike encoding vector, where the time represents a pixel's brightness. If the pixel is simply black, the time is set to zero; Any low values will also round to zero. Higher values may round to higher indices.

More and different input processing techniques may be necessary in order to improve the accuracy of the network, which is not a focus of this project. For instance, in order to ensure that bright pixels do not 'dominate' the input, the image may be duplicated and inverted to include a pos-neg encoding scheme, resulting in a 1568×1 input vector. Furthermore, completely zeroed input values may be removed from the input vector, encoding the input time as $t = \infty$. As the focus of the project is the network arrangement and not the network's accuracy at classifying digits, we leave these techniques to future consideration.

The temporal resolution, t_{res}, is an important hyperparameter of this encoding scheme, as with higher resolutions the network may take longer to train and infer. However, as resolution increases, so too does accuracy. We set this value to $t_{res} = 10$ for our implementation.

B. Temporal Neuron

The temporal neuron is a simple model of a neuron that fires in response to a series of time-spiking encoded inputs. It is mainly discussed in [4], but we will briefly summarize its functionality here. Chiefly, the temporal neuron consumes significantly less power than point-integrator alternatives, and is generally trained in an unsupervised fashion. The temporal neuron is also considered a more biologically plausible neuron model, as it more closely relates to the Hodgkin-Huxley model [10] of a neuron.

The temporal neuron takes in a series of inputs on a number of lines, generally encoded in a one-hot manner. Each line has an internal weight represented with it, encoding the dendritic segment's channel strength from the pre-synaptic neuron. The input lines are multiplied and summed, and the result increases the neuron's body potential. When the body potential reaches a threshold θ, the neuron will spike, sending a high signal on its axonal output line, and the body potential will reset.

The method in which neurons accumulate body potential is a potential hyperparameter of temporal network design. We implement the Step No-Leak (SNL) neuron in this work, which simply instantly increases the body potential based on the incoming potentials from the dendritic segments (inputs). A more involved strategy is the Ramp No-Leak (RNL) neuron model, where body potential will increase over time corresponding to the input and weight. The Leaky Integrate-and-Fire (LIF) neuron model is an even more biologically plausible model, where the body potential will (in addition to the ramping nature of the RNL neuron) 'leak' out over time, decreasing the excitation rate.

C. STDP Training

We implement Spike-Timing-Dependent Plasticity (STDP) training rules for the temporal neurons within our work. STDP rules are based closely on Hebbian theory [11], though the actual definitions of the rules vary. For our implementation, we utilize unsupervised STDP rules as well as supervised STDP rules as proposed in [4] for some input spike time t_i and output time t_j:

$$\Delta w_{ij} = \begin{cases} B(\mu_c) & \text{if } t_i \leq t_j, \qquad t_j \neq \infty \\ -B(\mu_b) & \text{if } t_i > t_j, \\ B(\mu_s) & \text{if} \qquad t_i \neq \infty, \quad t_j = \infty \end{cases}$$

The parameters of μ_c, μ_b, and μ_s are the STDP capture, backoff, and search parameters. They represent finite probabilities, and the notation of $B(\mu_*)$ refers to the choosing of a Bernoulli random variable. This implementation of STDP is unsupervised; [4] also proposes a supervised approach to training called R-STDP, which will be used for an output layer. Generally, μ_* is set to inverse powers of two. For our implementation, we utilize $\mu_c = 2^{-3}$, $\mu_b = 2^{-7}$, and $\mu_s = 2^{-10}$. Generally, high capture values μ_c correspond to a network's willingness to learn new information; Lower backoff values μ_b teach the network to forget poorly formed connections less, and lower search values μ_s are added in order to force the network from a state of dormancy to excitation.

Alongside these, the parameters of W_{max} and W_{min} are the maximal and minimal allowed weight values. After STDP training, all weights are clamped to this particular range. These parameters are usually $W_{min} = 0$ and $W_{max} = 2^3$. The value of W_{max} corresponds to the maximal weight resolution, and from [4] 2^3 is generally 'biologically enough.' From our testing, we find insignificant benefits increasing past 2^3, but we leave our neurons at 2^6 in order to slow down excitation and observe effects more granularly.

Finally, the threshold value θ is generally defined as a function of the number of input neurons n_i and the maximal weight W_{max}:

$$\theta = \max(1, W_{max} \times n_i \times \theta_f)$$

The value of θ_f is a hyperparameter of the network, and is bounded within $0 < \theta_f < 1$. We set the value of $\theta_f = 0.1$, though other higher values could slow down excitation further.

D. Reservoir

The reservoir is the focus of network analysis, containing a series of temporal neurons arranged in a randomly connected graph, or a 'liquid'. This graph of neurons will change over time with training and with STDP rules, adding and removing connections arbitrarily in response to the inputs. We start with the parameter of n neurons, which we set to $n = 150$ for our implementation, based on the number of neurons loosely suggested to exist in a cortical column [1].

Furthermore, a 'seed' configuration for the network may be specified, indicating its initial connectivity. This seed configuration W_0 is generated through a number of random graph algorithms, and is a hyperparameter of the network. We test on seed networks of fully connected (FC) nodes, Erdos-Renyi (ER) graphs, Barabasi-Albert (BA) graphs, and Watts-Strogatz (WS) graphs. Each of these networks contains their own particular parameters:

1) FC: No parameters.
2) ER: The probability of edge creation p.
3) BA: The number of edges to attach from a new node m.
4) WS: The probability of rewiring each edge p, and the number of edges within the base lattice k.

The liquid reservoir is simulated on a time quanta for each particular and unique training sample. On each iteration, the reservoir is fed a new spiking column of the input vector (if it exists), and any excitations are then fed back to the reservoir's neighboring connections. When the input vector is exhausted, the reservoir continues until t_{max}, the maximal amount of time for the quanta. The reservoir is then trained with STDP rules, before being readied for the next quanta. The maximal amount of time in a quanta is a hyperparameter of the network, and allows the network to 'settle' into a more favorable position after a series of excitations. We set this value $t_{max} = 20$.

E. Discriminant Column

The output discriminating column, responsible for classifying the state of the liquid, is implemented as a TNN minicolumn as described in [4]. The minicolumn is a series of n neurons (for n output classifications), and is furthermore fed into a winner-takes-all lateral inhibition (WTA-LI) layer. The WTA-LI layer implements the biological function of astroglia, inhibiting outputs of neurons as necessary. STDP rules for this output column take into account the output time t_j as the time when the entire column spikes.

With the WTA-LI column, the first neuron to spike inhibits all other neurons, and will dominate the output. Other forms of inhibition and astrocyte modelling have been explored in related work (see [12]), but will not be explored within our project.

The temporal minicolumn utilizes R-STDP rules for learning, which simply modulate the prior cases of STDP (capture, backoff, and search) based on whether the output matches some target value:

1) If the output is correct, then STDP search is disabled; The other STDP rules are performed as normal.
2) If the output is incorrect, then STDP backoff is disabled. The capture weights are decremented instead of incremented, and search is still enabled.
3) If the column never spikes, only search operates.

All other parameters for neurons (θ, W_{max}, W_{min}, μ_*) are configured to be identical to the liquid's neurons for our implementation.

V. Results

The following are a series of our experimentations and results for particular networks.

A. Varying Training Steps

We first examine the effect of number of training samples on the network. We can look at the fully-connected (FC) network as a baseline and analyze the network's configuration, which can be seen in V.1.

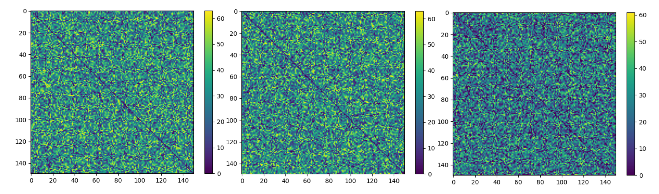

Fig. V.1. Diagrams of the fully-connected network's adjacency matrices. The left diagram is at $n = 10$ samples, the middle at $n = 100$ samples, and the right at $n = 1000$ samples.

Over larger training steps, the network's adjacency matrix becomes more significantly sparse. We can pull out the difference matrices to further elaborate this point in V.2.

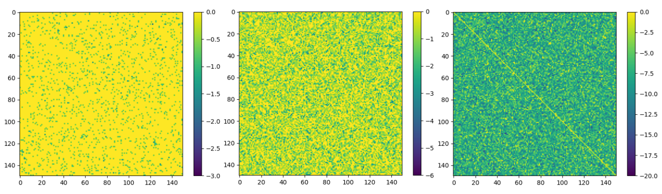

Fig. V.2. Diagrams of the fully-connected network's adjacency matrix differences from their initial seeds. The left diagram is at $n = 10$ samples, the middle at $n = 100$ samples, and the right at $n = 1000$ samples.

The matrices definitely get more sparse over time, despite extremely low μ_b backoff values. Networks are notably capable of building new connections, and the connection removal parameter μ_b being very low indicates that the network is still removing connections it deems 'unnecessary.' This tendency to forget irrelevant information rapidly seems to be an underlying feature of STDP training. This pattern repeats itself across a variety of networks, and can be seen through the average degrees of all networks, in V.3.

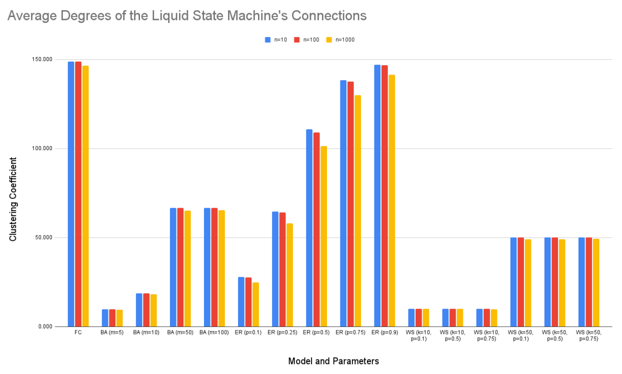

Fig. V.3. Average degrees of all networks. The blue bars are $n = 10$ samples, the red $n = 100$, and the yellow $n = 1000$.

The important trend here is the decreasing of average degree experienced by all models. We see this difference exaggerate itself in Erdos-Renyi networks, which we can further examine in V.4. This finding is more pronounced in the differences graph in V.5.

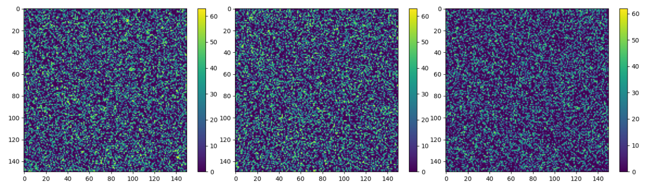

Fig. V.4. Diagrams of the Erdos-Renyi network's adjacency matrices (with connection probability $p = 0.5$). The left diagram is at $n = 10$ samples, the middle at $n = 100$ samples, and the right at $n = 1000$ samples.

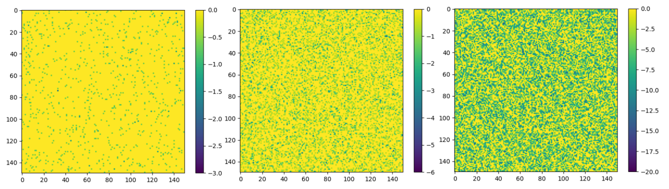

Fig. V.5. Diagrams of the Erdos-Renyi network's adjacency matrix differences from initial seed configuratons (with connection probability $p = 0.5$). The left diagram is at $n = 10$ samples, the middle at $n = 100$ samples, and the right at $n = 1000$ samples.

In particular, Erdos-Renyi suffers from a very high rate of connection removal with increasing training samples. This is likely due to the extremely high degree connectivity, which is a feature of this network.

B. The Watts-Strogatz Model

As shown in V.3, the Watts-Strogatz model degenerated to extremely low average degrees over time. In order to see if the model kept its small-world characteristics, we examined clustering coefficients in V.6.

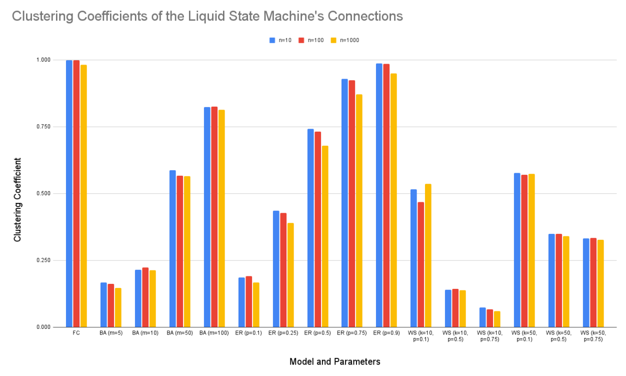

Fig. V.6. Clustering coefficients of all networks. The blue bars are $n = 10$ samples, the red $n = 100$, and the yellow $n = 1000$.

We noticed that the Watts-Strogatz models quickly degenerated to extremely low clustering coefficients, which is in direct contrast to the small-world principle of having significantly large clustering coefficients when compared to Erdos-Renyi models. We also looked at the shortest path length, which is a direct indicator of network diameter, in V.7.

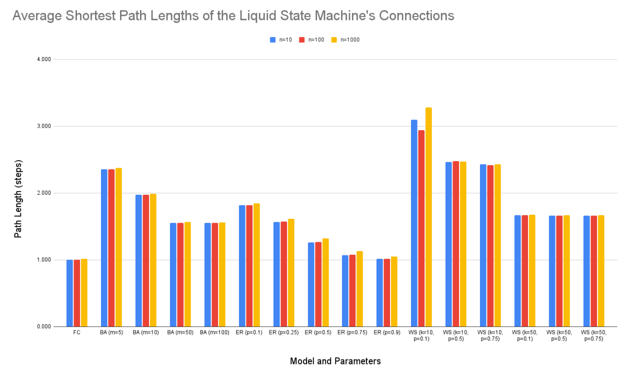

Fig. V.7. Average shortest path lengths of all networks. The blue bars are $n = 10$ samples, the red $n = 100$, and the yellow $n = 1000$.

Once again we see data in direct contrast to expectation. The small-world property is defined with small network diameters and large clustering coefficients. However, we see that STDP training rules degenerate a network into a very sparse graph with low clustering coefficients and high network diameters. One interpretation for this is that sparser models are more 'fitting' for data; This backs up the idea of inhibition being a key to the network, as discussed in [12]. We can see the graph generated in V.8.

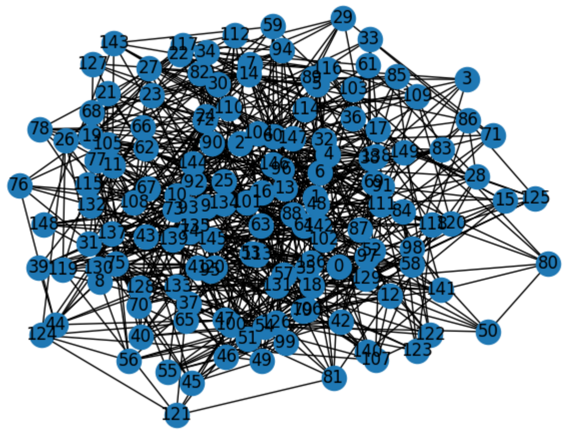

Fig. V.8. Diagram of the generated Watts-Strogatz graph with parameters $p = 0.75$, $k = 10$, after training for $n = 1000$ samples.

We can further dive into the network's structure by looking at the adjacency matrices. The adjacency matrix comparison is in V.9, and the difference matrix comparison is in V.10.

A significant band of negations forms, indicating that the network is removing connections in order to reduce clustering coefficient and subsequently increase path length.

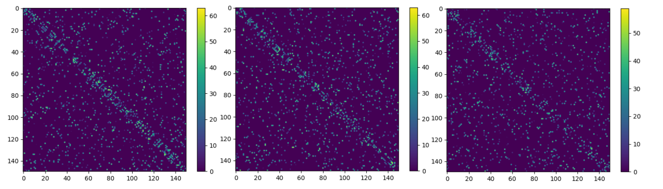

Fig. V.9. Diagrams of the Watts-Strogatz network's adjacency matrices (with $p = 0.75$, $k = 10$). The left diagram is at $n = 10$ samples, the middle at $n = 100$ samples, and the right at $n = 1000$ samples.

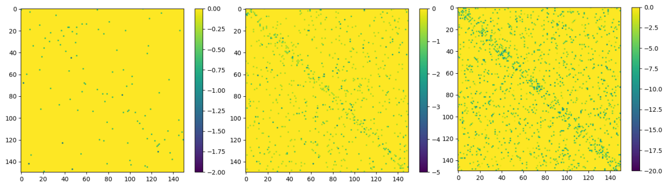

Fig. V.10. Diagrams of the Watts-Strogatz network's adjacency matrix differences from the initial seed configuration (with $p = 0.75$, $k = 10$). The left diagram is at $n = 10$ samples, the middle at $n = 100$ samples, and the right at $n = 1000$ samples.

C. Information Walks

In order to further explore this idea of inhibition being key to the network, we examined how new information would spread throughout the network through application of an SI spreading model. Nodes n_i and n_j that have a corresponding directed weight w_{ij} between them would transmit information with a probability of $\frac{w_{ij}}{\theta}$. As such, information is 'transmitted' through the network based on the probability that a particular neuron's excitation causes another neuron to fire. One such information walk is presented in V.11.

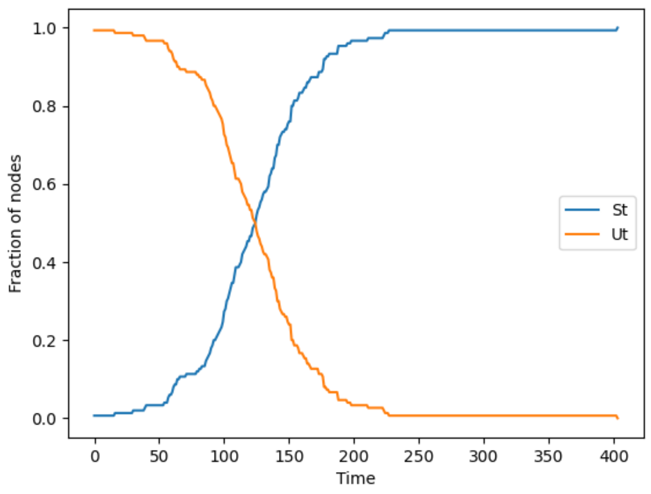

Fig. V.11. 'Information walk' SI spreading process throughout the Watts-Strogatz network trained with $n = 10$ samples, with parameters $p = 0.75$ and $k = 10$.

We compiled these results results for all networks, which can be seen in the graph in V.12.

As shown in the graph, information walk times experience an extreme spike in only the Watts-Strogatz models. A real-

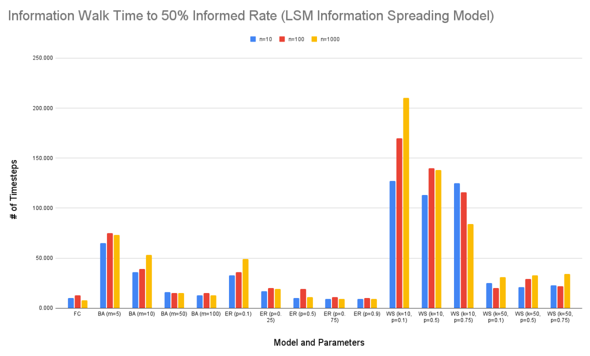

Fig. V.12. Graph of times to reach 50% of the nodes in the network for all networks. The blue bars are $n = 10$ training samples, the red $n = 100$, and the yellow $n = 1000$.

world interpretation of this comes from the motivation for astroglia within the neocortex. A lower walk time indicates that nodes activate more rapidly from new information, which may actually be poor for a network. Inhibitory cells like astrocytes exist such that the human brain does not constantly experience seizures or overwrite information. The neural connections modelled may experience better and more stable firing patterns when more resilient to incoming input changes.

Another interesting note from this data is how particularly chosen smaller parameters of Watts-Strogatz models seem to increase information walk time. Some Barabasi-Albert models come close with extremely low parameter values as well. Increasing inhibition may end up leading to increased performance with the network - As the network changes less in response to new information, the output discriminant column may have a more stable and accurate comparison of inputs to outputs.

D. Limitations

The main limitation of this project is scope - In the interest of time, the sizing of the networks as well as a variety of parameters could not be explored. Different theoretical models for learning, such as Kheradpisheh et al.'s STDP model [14] could also be explored, and it's unknown if the network degeneration we experienced is a feature of STDP or a feature of the NCAL implementation of Hebbian theory. This and more is examined in VI-A.

VI. Conclusions

Our project presents a framework for generating and analyzing self-organizing neural networks in order to derive insight into the neural basis for cognition through the orientation of the problem. Through analyzing the networks, we uncovered interesting insights to the strength of inhibition in network design, as well as the importance of the network's ability to be resilient in the face of changing input data. Our work acts as a base to inspire particular design choices for liquid state machines and temporal neural network design in general.

A. Future Work

In future renditions of this work, we would like to explore a vast variety of different network parameters and configurations, starting with the model of the neuron. More complicated neuronal designs (discussed in IV-B) may interact with STDP rules differently.

Other STDP models and models for Hebbian theory implementations may also be explored. In particular, our project demonstrated to us the value and importance of inhibition in biologically-plausible networks - As such, we are interested to see if astrocyte modulation of synaptic weights or other forms of inhibition would lead to more stable and accurate networks.

Finally, exploring the effects of differing networks on different datasets would help draw similarities and differences between learning in the general sense. Our structure is currently extensible to a variety of machine learning problems, so picking relatively orthogonal problems could offer interesting insight to how these problems may be solved in the brain.

B. Contribution

Throughout the semester, Vins focused on the implementation of the temporal liquid state machine, while Anand focused on the construction and implementation of the input vector and initial seed configuration networks. Both teammates contributed to the overall design and analysis of the generated networks. Anand focused on the information walk analysis, while Vins focused on analyzing the network configurations and adjacency matrices.

References

[1] V. B. Mountcastle. "The Columnar Organization of the Neocortex." *Brain: A Journal of Neurology* (1997).

[2] J. Hawkins, R. Dawkins. "A Thousand Brains: A New Theory of Intelligence." *Basic Books* (2021).

[3] E. Bullmore, O. Sporns. "Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems." *Nature Reviews Neuroscience* (2009).

[4] H. Nair, J. P. Shen, J. E. Smith. "A Microarchitecture Implementation Framework for Online Learning with Temporal Neural Networks." *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2021).

[5] J. E. Smith. "A Macrocolumn Implemented with Spiking Neurons." *arXiv preprint arXiv:2207.05081* (2023).

[6] S. Purdy. "Encoding Data for HTM Systems." *arXiv preprint arXiv:1602.05925* (2016).

[7] W. Maass. "Liquid State Machines: Motivation, Theory, and Applications." *Computability in Context: Computation and Logic in the Real World* (2011).

[8] T. Kohonen. "Self-Organized Formation of Topologically Correct Feature Maps." *Biological Cybernetics* (1982).

[9] H. Hazan, L. M. Manevitz. "Topological Constraints and Robustness in Liquid State Machines." *Expert Systems with Applications* (2012).

[10] A. L. Hodgkin, A. F. Huxley. "Currents Carried by Sodium and Potassium Ions Through the Membrane of the Giant Axon of Loligo." *The Journal of Physiology* (1952).

[11] M. M. Taylor. "The Problem of Stimulus Structure in the Behavioural Theory of Perception." *The South African Journal of Psychology* (1973).

[12] V. Ivanov, K. Michmizos. "Increasing Liquid State Machine Performance with Edge-of-Chaos Dynamics Organized by Astrocyte-modulated Plasticity." *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[13] AstroDave, W. Cukierski. "Digit Recognizer." *Kaggle* (2012). www.kaggle.com/competitions/digit-recognizer

[14] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier. "STDP-Based Spiking Deep Convolutional Neural Networks for Object Recognition." *arXiv preprint arXiv:1611.01421* (2016).