TemporaLSM: A Simulation Framework for Temporal Neural Liquid State Machines

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 a type of reservoir computer that aim to construct more biologically plausible neural network architectures than artificial counterparts. Rather than purposely constructing neurons in particular arrangements, an LSM self-organizes its neurons into a 'liquid', which is read out to perform a task. Many arguments against LSMs stem from this difficult-to-control random procedure, and due to their complicated setup, current LSM designs are not well explored.**

Through this work, we present TemporaLSM, a simulation framework for liquid state machines using good-for-hardware temporal neuron designs. Furthermore, we analyze a series of liquids generated by TemporaLSM to determine how different hyperparameters affect the construction of the liquid, and attempt to identify similarities between liquids on differing tasks.

***Index Terms*—liquid state machines, temporal neural networks**

I. Introduction

II. Network Construction and Operation

This work mainly follows the temporal neuron as defined in [1]. As such, this section presents an overview of the network construction process and the individual (hyper)parameters of a constructed network.

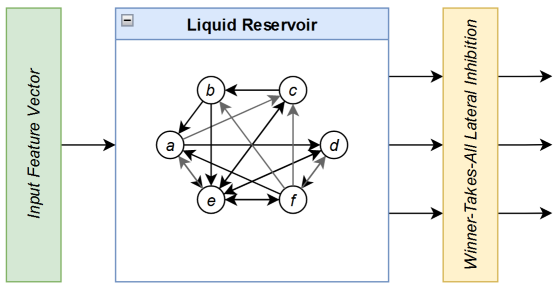

Fig. II.1. Diagram of a temporal liquid state machine (TLSM).

The overall temporal liquid state machine, as depicted in figure II.1, consists of an input feature vector, a reservoir, and an output column. This entire structure (composed of three parts) implements an unsupervised online learning algorithm, and is heavily parameterized.

A. Input Feature Vector

The input feature vector is a one-dimensional time-encoded vector of the input values discretized and normalized in some arbitrary capacity. As temporal neurons are intended to be good-for-hardware models, they are simply integral in type (as per [1]). As such, the input vector is also integral in type.

The inputs are encoded as time spikes, where a 'higher' value corresponds to a quicker spike (with a lower 'entry time'). These spikes notably do not have any magnitude associated with them, and are simply an impulse function on the line.

The choice of input encoding is arbitrary, and is focused on more in section III for each problem. However, some notable features of 'good' encoding schemes for biologically inspired neural networks are discussed in [2]. In particular, the overlapping of contextual information is a key feature of good encoding schemes - 'Proximal' patterns can be encoded in a way that allows for the network to learn similarities between input patterns.

B. Network Configuration

The reservoir contains a series of neurons, the number of which is a hyperparameter to the network. Each neuron within the reservoir is connected to other neurons in some particular way. This connectivity is, itself, another hyperparameter of the network. As such, the reservoir (upon initialization) takes in a 'seed matrix' W_0. For a TLSM of size n, this matrix is of size $n \times n$. The seed matrix forms an adjacency matrix for the reservoir, indicating its connectivity.

There are two particular features regarding the network indicated by the seed matrix. For one, the seed matrix may be asymmetric. This makes the output network a directed graph. Secondly, the seed matrix may be sparse and is not necessarily fully connected. The construction of the seed matrix may be done using one of many random graph algorithms, such as the Erdos-Renyi model or the Barabasi-Albert model. We leave this choice as a hyperparameter as well, opting to implement several different connectivity models to test their performance individually for each problem.

C. Neuron Activation Functions

Lorem ipsum

D. Network Training

Lorem ipsum

III. Problem-Specific Analysis

A. MNIST Handwritten Digits

IV. Conclusions

A. Notes on Network Organizations

B. Future Work

References

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

[2] S. Purdy. “Encoding Data for HTM Systems.” *arXiv preprint arXiv:1602.05925* (2016).

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

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