Modifying Synaptic Connections On The Spiking Neural Network Architecture (SpiNNaker) In Real-Time

Nishant Shukla Matthew Frazier

***Abstract*—Artificial Neural Networks is a promising approach to study human brain computation. Recent computer architecture design of a low-power 72-core processor by the University of Manchester (SpiNNaker) has made it easier to study highly parallel networks. We designed an algorithm on the SpiNNaker chip to enable synaptic removal, addition, and randomization on a neural network topology during run-time. Additionally, we explored scalability issues and unintended pitfalls with this approach.**

***Keywords*—Neural Networks, Synaptogenesis, SpiNNaker**

I. Introduction

THE human brain is fast and low in energy. [TODO: Matt] blah blah blah. One property of the brain is it's plasticity (cite Lashley, or some Psyc studies). Our approach to neural networks is greatly influenced by this biological idea, and we want to enable networks to self-modify. With energy effeciency in mind, we enable this functionality on the Spiking Neural Network Architecture (SpiNNaker) provided by the University of Manchester.

A. SpiNNaker

[TODO: Matt] The SpiNNaker is cool. Here's why. (Energy effecient) It's so epic. And we have one. And we added a cool functionality to it.

B. Synaptogensis

[TODO: Matt] Define Synaptogenesis and explain why it's cool. Here's why. Explain it's previous success from Levy's work.

[TODO: Matt] We're combining the two. And it's gonna be super userful.

II. Design

NEURAL networks consist of biologically inspired relationships between individual nodes (neurons). Learning in such a network occurs by intelligently adjusting weights on the links between the nodes. We define the relationship between nodes into the following three categories:

(a) **Static** - where the network topology is fixed, and can only change by manually adjusting the relatiionship between nodes. Most neural networks fall into this category because regardless of the number of nodes n, only one fixed relationship forms between them:

$$\text{NumberOfTopologies}(n) = 1$$

(b) **Semi-Dynamic** - in which the network is not static, and there is some freedom for nodes to rewire with other nodes in real time. The complexity grows exponentially. Each node has non-zero probablity to be rewired with a constant c number of other nodes. Given n such nodes, the number of possible topologies can be up to

$$\text{NumberOfTopologies}(n) = c^n$$

(c) **Dynamic** - where each node has non-zero probability to be wired with any other neuron in the entire network. Given n nodes, the number of possible topologies becomes

$$\text{NumberOfTopologies}(n) = n^{n-1}$$

A biological neural network such as the human brain is not static. Connections between neurons are regularily formed and removed over time. Moreover, such a neural network is not fully dynamic either, since locality is a physical constraint. For example, a neuron on the far end of the left hemisphere of a brain may never directly connect with a neuron on the right hemisphere.

We designed a programming framework on SpiNNaker's API to enable a semi-dynamic network topology. In our implementation, each neuron had the flexibility to rewire with none, all, or some of the fixed number of other neurons.

III. Implementation

BY taking advantage of the multiple independent cores on the SpiNNaker, we were able to emulate effecient semi-dynamic synaptogenesis. We implemented a perceptron with fifteen nodes consisting of nine input layers and six output layers. As diagramed in Figure 1 below, each of the nine neurons broadcasted its excitation to an arbitrarily chosen six output neurons.

[Figure 1]

We used the weight modification rule proposed by Levy to categorize an input set of letters.

[Delta W rule]

We implementd a single-layer perception as a proof of concept to demonstrate synaptogenesis on the SpiNNaker board. In our network, every input node broadcasts a data packet to six of the output nodes. To simulate the creation and rewiring of synapses, not all messages are registered by the output neurons.

Each packet received by an output node is looked up in the output node's local list of incoming connections. If the address from the packet is not listed in the incoming connections list, it will be ignored, and no further computation will be done.

Synaptic connectivity is broken when weights fall below some negative threshold. More specifically, when an average running weight dropped below $-\delta$ for some $\delta > 0$, the link is considered disconnected.

New connections are established between nodes when an average rate of activity drops below 25%. In this case, the next data packet received from a muted neuron becomes unmuted, in hopes to raise the average rate of activity.

IV. Pitfalls

SOME disadvantages are present when considering our apprach for synaptogenesis on the SpiNNaker. [TODO: Matt] Blah blah.

V. Further Study

THIS paper reveals multiple questions that still need to be examined further.

VI. Conclusion

The conclusion goes here.

Appendix A
Shannon's Entropy

Appendix one text goes here.

Appendix B
Calculaton of Mutual Information

Appendix one text goes here.

Appendix C
Calculation of Statistical Dependence

Appendix two text goes here.

Acknowledgment

The authors would like to thank Professor Worthy Martin, Associate Professor of Computer Science at the University of Virginia.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.