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

Matthew Frazier Nishant Shukla Worthy Martin*

***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 subfield of AI known as neural network computation has recently received a large and growing amount of attention. Put simply; neural networks (NN) are computational models inspired by biological brains that are capable of machine learning. A NN usually consists of interconnected "neurons" (in quotation marks for they are artificially simulated to varying degrees of biological accuracy – but that topic is outside the scope of this paper) which compute from their inputs and produce various outputs, depending on the model. This paper focuses on types of spiking neural network (SNN) models, in which the neuronal communication (and therefore, the computation of the system as a whole) is achieved via message spikes, or action potentials, from one neuron that is synapsed onto another. In this introduction, we first describe how a SNN is a good computational representation of the brain. We then describe various different spike-dependent spike-time learning methods used on a SNN, followed by an exploration of various types of SNN. We discuss why the neuromimetic hardware we use is a good platform for experimentation with SNN, and then describe our approach to the problem of simulating synaptogenesis on such a device and propose a solution. Our design section explores this solution in detail, and our implementation section describes our results.

A. The Brain and other Spiking Neural Networks

The brain is a spiking neural network. That is to say; biological neurons pass information around the brain via action potentials which travel from a neurons' soma (cell body) down the axon and generate synaptic events which are received by the dendrites of any neurons that the one in question has synapsed to. [Include figure illustrating this?] The process has been simplified greatly as neurobiology is not the focus of this paper, nor is the biological process yet completely understood. What's important are the overarching properties exhibited by brains. Namely; that brains are fast, power-efficient, and capable of obtaining, representing, and integrating complex multi-dimensional information sets into useful knowledge (and from noisy sensory inputs, at that). Moreover, the property of being able to fairly reliably solve highly complex problems is something we want computers to have. [Include TSP example stating that human could do in a day what a procedural program would take hundreds of thousands of man-years to do?] Progress is however being made towards the goal of making machines more like minds; indeed, the bus in a computers' hardware may operate at tens of MHz, while an axon may carry only tens to hundreds of action potentials per second, about five orders of magnitude slower than a machine (Furber 2007). The power efficiency of the brain is still far ahead of even the best neuromimetic hardware but the gap is closing quickly, considering that biological evolution had several hundred million years of a head start. One very important property of the brain that computers have a hard time with, which is also believed to be quintessential in mankind's ability to adapt to and solve new and ever-more difficult problems, is neural plasticity. It would be erroneous to say that no degree of plasticity has been implemented in any type of SNN, but given that we do not yet fully understand the mechanisms governing neural plasticity in our own heads, it suffices to say that there is yet much work to be done in this field of computational neuroscience in which artificial neural networks dynamically alter their own topology.

B. Overview of Spike-time coding and learning methods

Before discussing different types of artificial spiking neural networks, it is necessary to describe various spike-time coding and learning methods that are relevant to understanding how the different types of spiking neural networks function. Firstly, the distinction between rate- and rank-order coding must be addressed. Rate order coding is, as it sounds, a way to describe neural responses to stimulus in terms of firing rate, paying attention to the timing between spikes of a single input. Rank order, on the other hand, focuses on the relative timing of spikes from all inputs. As has been argued in depth elsewhere [cite], rate order has a number of drawbacks, though the primary one is its inability to transmit and process information in a short period of time; with n neurons being able to transmit over the course of 10 ms only $log_2(nC1)$ bits of information. Rank order, under the same constraints, is capable of achieving $log_2(n!)$ bits of information. As detailed in [cite: Thorpe 1998],

rate order can be considered an $analog - to - frequency$ converter, while rank order would be an $analog - to - delay$ converter, as the time an integrate-and-fire neuron takes to reach threshold is dependent upon input strength. In short, Rank Order (RO) coding is assumed to be a better model as it can process more information in a shorter amount of time with fewer redundant neurons. Moving on to learning methods, spike time dependent plasticity (STDP) implements plasticity through the use of long-term potentiation (LTP) and depression (LTD). In other words, the connection weight between two neurons increases if the pre-synaptic neuron spikes before the post-synaptic neuron, and the weight decreases in the reverse case. Spike driven synaptic plasticity (SDSP) is a semi-supervised variant of STDP in which a threshold $Vmth$ is given to the post-synaptic neurons' membrane potential. If the membrane potential is above $Vmth$ when an input spike arrives, potentiation occurs. Otherwise, the synapse experiences depression. [edit: use formulas from Kasabov?] These two cases are typically either shortly before or shortly after a post-synaptic spike is emitted, respectively. We will later describe a new learning method referred to as entropic synaptogenetic plasticity (ESP), governed by biologically inspired formulas taken from (Levy).

C. Types of Spiking Neural Networks

[TODO: Matt] There's SNN (default, outdated) - static eSNN (evolves -alters synaptic weights- between a learning and recall phase) deSNN (evolves dynamically - there's no differentiation between learning and recall (learns continuously))

D. SpiNNaker

[TODO: Matt] The SpiNNaker is cool. Here's why. It's: -fast -power efficient -massively parallelized -capable of running (arbitrarily?) complex SNN sims And we have one. And we added a cool functionality to it (maybe).

E. Synaptogensis

[TODO: Nishant] Define Synaptogenesis and explain why it's cool. (And why this model's different from deSNN – entropy) Here's why. Explain it's previous success from Levy's work.

[TODO: Matt] We're combining the two and extending the deSNN model into an edeSNN model, which will shortly be enslaving the entire human race.

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 relationship 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 probability 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 does not follow the static network topology. Connections between neurons are regularly 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 efficient semi-dynamic synaptogenesis. We implemented a perceptron with fifteen nodes consisting of nine input layers and six output layers. As diagrammed 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 implemented 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 approach 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
Calculation 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.