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 in hopes of achieving similar learning by artificial agents. Recent architecture design of a low-power supercomputer by the University of Manchester (SpiNNaker) has made it easier to design highly parallel brain-inspired algorithms. We used the SpiNNaker chip to implement a neural network capable of rewiring its connection in real-time while trying to minimize information loss and maximize the decrease in statistical dependence. 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. Furthermore, we discuss future work that remains to be conducted, including interesting implications such as computability beyond Von Neuman architecture and Turing machines.

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 all other connected neurons. Figure 1 diagrams the relevant biological prerequisites.

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 illustrating that a 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 has received little attention by Computer Scientists, 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 involving the dynamic creation or removal of synapses. It would be erroneous to say that no degree of plasticity has been implemented in any type of SNN, but to the best of our knowledge there is yet no model which allows networks to add or remove synapses at runtime. 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. [cite: Kasabov 2013]

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. During an initial learning phase, in which a new output neuron i is created for each N-dimensional training input pattern, connection weights $w_j, i(j = 1, 2, ..., N)$ are calculated based on the rank order learning rule:

$$w_j, i(t) = \alpha \cdot mod^{order(j,i)} \quad (1)$$

where w_j, i is the connection weight between post-synaptic neuron i and pre-synaptic neuron j, α is a learning parameter, mod is a modulation factor (determining the importance of the order of the first spikes), with $order(j, i)$ representing the order of spikes arriving at neuron i from all synapses to the neuron i (has value 0 for the first spike to neuron i, increases incrementally for each subsequent spike from other pre-synaptic neurons). [cite: Kasabov 2013] Some models using rank order coding will also merge output neurons with similar weight vectors based on the Euclidean distance between them, thus reducing the number of redundant neurons that must be simulated. In short, rank order (RO) coding is assumed to be a better basis for model construction as it allows networks to 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), and influenced by Information Theory from Shannon.

C. Types of Spiking Neural Networks

(a) **A spiking neural network (SNN)** is the most basic form, in which synaptic weights are static and neurons communicate via message spikes. While they have some drawbacks, an SNN is still a fairly powerful biologically-inspired model.

(b) **An evolving spiking neural network (eSNN)** evolves its functionality and structure based on incoming information using the RO rule – formula (1), but only during a learning phase. Once a neurons' weight has been set, however, this model does not allow for any further tuning of synaptic weights to reflect on other incoming spikes at the same synapse. That is to say; these synapses can capture some long-term memory during the learning phase, but have little ability to capture any short term memory, which is detrimental to the computational capabilities of the network as a whole.

(c) **A dynamically evolving spiking neural network (deSNN)** represents an improvement over eSNN in that it implements not only RO learning during an initial learning phase, but also a dynamic synaptic plasticity mechanism so that it will continue to learn and improve performance during a recall phase; the deSNN proposed in [cite: Kasabov 2013] uses the SDSP mechanism. This fixes the primary shortcoming of the eSNN model; allowing for short term memory to be stored easily, and to be potentiated as long as it remains useful.

D. SpiNNaker

The Spiking Neural Network Architecture (SpiNNaker) board is a neuromimetic hardware system designed by the University of Manchester with the eventual intent of simulating the human brain. It is the brainchild of Steve Furber, and has a number of properties that make it useful for neural computation:

- fast
- power efficient
- massively parallelized
- globally asynchronous
- locally synchronous
- capable of running complex SNN simulations [superfluous?]
- scalable

[cite: all of the above] [TODO: explain why these properties make it suitable for this experiment – also the overlap of these properties with those of the brain]

E. Synaptic Plasticity

A difficult problem in designing neural networks is configuring the initial parameters to optimize the network. For example, defining how the neurons are connected together greatly affects the efficiency and performance in learning [cite]. Perceptrons are often designed as bipartite graphs, where every neuron in the pre-synaptic layer is connected to every neuron in the post-synaptic layer. This brute-force approach to constructing a network topology may achieve satisfactory results if the weights converge to the optimal values within an acceptable time. However, a bipartite graph scales exponentially to the number of nodes in each layer. In our solution, we let the neural network modify itself, providing a "hands-off" approach to designing a network.

Other than updating synaptic weights to achieve learning, we introduce two additional forms of synaptic plasticity. In our study, we allow new synapses to be formed (synaptogenesis), or existing synapses to be entirely disconnected. The three types of plasticity undergo the following principles (Levy 2004):

(a) **Synaptogenesis** - Unlike most networks, we allow ours to form new connections between neurons while learning occurs. The creation of a new synapse depends on whether the post-synaptic neuron is at an optimal activation. If not, the network considers forming a new synapse. Additionally, synaptogenesis only occurs between neighboring neurons, ensuring a constraint on locality. Synapses are formed by taking into account both receptivity and avidity (Levy 2004).

Avidity is the measure of ability for each neuron to participate in a new synaptic connection, defined as

$$A_i(t) = \frac{a}{(a + \sum_j \sum_k W_{i_k j}(t))}$$

$$a = \begin{cases} 1.0 * 10^{33} & \text{for unlimited avidity} \\ 1.0 & \text{for moderate avidity} \\ 1.0 * 10^{-3} & \text{for limiting avidity} \end{cases}$$

The receptivity of new synaptic connections is inversely proportional to the running average of the neuron's activation. We use the following equation for receptivity,

$$R_j(t) = \frac{r_1}{r_1 + \bar{y}_j(t)^{r_2}}$$

$$\bar{y}(t) = 0.99\bar{y}_j(t-1) + 0.01Y_j(t)$$

(r_1 and r_2 are experimental constants)

Finally, the probability of forming a new synapse between two neurons i and j depends on both avidity and receptivity.

$$\text{Prob(of new synapse } ij) \propto A_i(t) * R_j(t)$$

(b) **Weight Modification** - Similar to most unsupervised weight modification rules, we use a deterministic process depending on the pre- and post-synaptic activities. This type of weight adjustment is based off the Hebbian rule, where links between nodes that fire together strengthens. Specifically, weights are adjusted using the following formula,

$$\triangle w(t+1) \propto f(\text{post}_j(t)) * g(\text{pre}_j(t), w_{ij}(t))$$

(c) **Synapse Removal** - Lastly, we allow our network to undergo synaptic removal, a stochastic process where the probability of removal becomes non-zero if excitatory synapses fall bellow a specified threshold. Given some constant $\delta > 0$, probability of synaptic removal is defined as

$$\text{Prob(removal } ij) = \begin{cases} 0 & \text{if } w_{ij}(t) + \triangle w_{ij}(t+1) > \delta \\ > 0 & \text{otherwise} \end{cases}$$

Synaptic connections are slowly modified through additions or removals until optimal output firing levels are obtained.

Levy asserts that "information theory has gained popularity in recent years as a tool for understanding brain recordings."

[TODO: Matt] We're combining the two and extending the deSNN model into an edeSNN model, which will shortly achieve sentience and enslave 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 in the brain are regularly formed and removed over time [cite]. 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 might never directly connect with a neuron on the right hemisphere.

We designed a programming framework on SpiNNaker's interface to enable a semi-dynamic network topology. Each neuron has the flexibility to rewire with none, all, or some of the fixed number of other neurons, following our synaptic plasticity rules. To test performance, the network analyzed the MNIST database of handwritten digits. We compared information loss (see appendix A) and statistical dependency

(see appendix B) from a network following our synaptic plasticity model, to that without synaptogensis and synapse removal.

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.

The network was composed of 784 input neurons and 17 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 and Mutual Information

Appendix A text goes here.

Appendix B
Calculation of Statistical Dependence

Appendix B 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.