AFRL-RY-WP-TR-2020-0280

IMPLEMENTATION OF ASSOCIATIVE MEMORY WITH ONLINE LEARNING INTO A SPIKING NEURAL NETWORK ON NEUROMORPHIC HARDWARE

Michael James Hampo
University of Dayton

SEPTEMBER 2020
Final Report

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
September 2020	Thesis	2 September 2020 –2 September 2020

4. TITLE AND SUBTITLE
IMPLEMENTATION OF ASSOCIATIVE MEMORY WITH ONLINE LEARNING INTO A SPIKING NEURAL NETWORK ON NEUROMORPHIC HARDWARE

5a. CONTRACT NUMBER
FA8650-18-C-1607

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
69199F

6. AUTHOR(S)
Michael James Hampo

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
Y1R7

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Dayton
300 College Park
Dayton, OH 45469

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYDR

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2020-0280

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO case number 88ABW-2020-2754, Clearance Date 2 September 2020. © 2020 Michael James Hampo. Presented in partial fulfillment of the requirements for the Degree of Master of Science in Computer Engineering at The School of Engineering of the University of Dayton. This work was funded in whole or in part by Department of the Air Force. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government. Report contains color.

14. ABSTRACT
Implementing cognitive algorithms on robots is one potential direction to realize autonomous artificial agents. There is an effort to push robotics and artificial intelligence into many aspects of daily life. An important step in this process is leveraging concepts known to work from human cognitive features on computer systems to improve the performance of robotic systems. Spiking Neural Networks (SNNs) allow these computational models to be instantiated in a low size, weight, and power (SWaP) form factor due to the biological efficiencies they approximate. This paper shows an associative memory in the form of an SNN, an application of the associative memory, and some performance benchmarking. The model is created using a neural network simulator and run on a low SWaP CPU and Intel's neuromorphic processor, Loihi, an artificial intelligence accelerator highly optimized for spiking neural algorithms. In addition, the model is employed on a mobile robotic platform that explores the real world and uses online learning to make associations. When the model was run on Loihi the overall power usage decreased as well as the run time of the simulation as compared to the low SWaP CPU proving beneficial to implement the neuromorphic hardware.

15. SUBJECT TERMS
associative memory, autonomy, low SWaP, machine learning, neuromorphic, spiking neural network

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ashley DeMange
Unclassified	Unclassified	Unclassified	SAR	57	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

IMPLEMENTATION OF ASSOCIATIVE MEMORY WITH ONLINE LEARNING INTO A SPIKING NEURAL NETWORK ON NEUROMORPHIC HARDWARE

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Computer Engineering

By

Michael James Hampo

Dayton, Ohio

August, 2020

IMPLEMENTATION OF ASSOCIATIVE MEMORY WITH ONLINE LEARNING

INTO A SPIKING NEURAL NETWORK ON NEUROMORPHIC HARDWARE

Name: Hampo, Michael James

APPROVED BY:

Tarek M. Taha, Ph.D.
Advisory Committee Chairman
Associate Professor, Electrical and
Computer Engineering

Eric J. Balster, Ph.D.
Committee Member
Associate Professor and Chair,
Electrical and Computer Engineering

Trevor Bhil, Ph.D.
Committee Member
Research Engineer,
Air Force Research Laboratory,
Sensors Directorate

Robert J. Wilkens, Ph.D., P.E.
Associate Dean for Research and Innovation
Professor
School of Engineering

Eddy M. Rojas, Ph.D., M.A., P.E.
Dean, School of Engineering

ABSTRACT

IMPLEMENTATION OF ASSOCIATIVE MEMORY WITH ONLINE LEARNING INTO A SPIKING NEURAL NETWORK ON NEUROMORPHIC HARDWARE

Name: Hampo, Michael James
University of Dayton

Advisor: Dr. Tarek M. Taha

Implementing cognitive algorithms on robots is one potential direction to realize autonomous artificial agents. There is an effort to push robotics and artificial intelligence into many aspects of daily life. An important step in this process is leveraging concepts known to work from human cognitive features on computer systems to improve the performance of robotic systems. Spiking Neural Networks (SNNs) allow these computational models to be instantiated in a low size, weight, and power (SWaP) form factor due to the biological efficiencies they approximate. This paper shows an associative memory in the form of an SNN, an application of the associative memory, and some performance benchmarking. The model is created using a neural network simulator and run on a low SWaP CPU and Intel's neuromorphic processor, Loihi, an artificial intelligence accelerator highly optimized for spiking neural algorithms. In addition, the model is employed on a mobile robotic platform that explores the real world and uses online learning to make associations. When the model was run on Loihi the overall power usage decreased as well as the run time of the simulation as compared to the low SWaP CPU proving beneficial to implement the neuromorphic hardware.

To my family and soon to be wife

ACKNOWLEDGMENTS

AFRL: For Funding for my masters program, the opportunity to work on such a great project as well as all the support with it.

- Kerry Hill
- Todd Jenkins
- Trevor Bihl
- Stefan Westberg
- Ashley DeMange

UDRI: For all the support and wisdom from my colleagues given on the project and guiding me as I learned many new things.

- David Fan
- Stephen Reynolds

Committee Members: For offering their time and knowledge to support me and offer advice where I have needed it.

- Tarek Taha
- Eric Balster
- Trevor Bihl

Special thanks to my advisor, Dr. Tarek Taha: For supporting me through this whole project with wisdom and experience, and for always pushing me to solve problems on my own and do things I didn't think were possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Autonomous systems have the potential to serve as human surrogates in dangerous environments or serve as a force multiplier where multiple autonomous agents report to a much fewer number of human operators. However, some issues exist in creating viable autonomous systems. Primarily these include the current inability of machines to handle uncertainty, whereas humans and animals can make conclusions and decisions in new environments from limited data [4].

By incorporating basic cognitive concepts that humans and animals use every day to make decisions, autonomous systems can exhibit more animalistic type behavior that can handle uncertainty [5]. One such concept is associative memory. Associative memory allows an agent to draw a conclusion about something it has observed and alter its plan of action accordingly without the assistance of a human operator or complex logical control software. Developing this ability is potentially a key enabler to future autonomous systems and part of developing true artificial intelligence (AI). AI algorithms typically require large amounts of processing power and thus need significant computing resources. To enable AI systems to run on low size, weight, and power (SWaP) platforms, one must consider energy efficiency developments in novel algorithms and hardware combinations to reduce operating constraints [6].

This work is part of a larger effort to build a biologically inspired cognitive agent that started in [7]. In this work we present a fully contained system equipped with an associative memory model capable of performing online learning, interacting with its environment, and running on low SWaP neuromorphic hardware. This model takes advantage of both supervised and unsupervised learning rules to train weights in the network to be able to

associate two input signals together, i.e. when one signal is presented, the model will recall the corresponding signal.

The model was first run on an Up board, a low SWaP option for traditional computing hardware [8], to not only confirm the model performs as expected but also to give a benchmark on dynamic energy consumption. This model proved to be capable of learning associations and recalling the associations given multiple input signals. This system was also run on Intel's neuromorphic research chip, Loihi [9], to take advantage of the specialized low SWaP hardware. This model is the first known model of its type to be implemented on Intel's neuromorphic research chip Loihi [9] and interact with live sensor input data. The results between the two hardware architectures show that there is a loss in accuracy on Loihi but improved energy efficiency.

The goal for designing a model as such was to try and advance past traditional neural networks and deep learning techniques to take full advantage of the neuromorphic brain-inspired hardware such as Loihi. With this hardware, if algorithms can be converted into spiking neural networks, not only should there be increased efficiency in execution time as well as energy consumption, but the networks can also use online learning capabilities built into Loihi [9]. This online learning is a key feature as it allows the network to train and improve while it is currently running and making other decisions.

Portions of this thesis were presented at the International Conference on Neuromorphic Systems (ICONS) 2020 [10]. That work found that the current model could not be ported over to Loihi to use the online learning features, using the chosen SNN simulator. The model was still tested and compared to traditional hardware with pre-trained weights, but unfortunately, due to the limitations of the hardware, no easy workaround was found to

perform online learning with the current model. The new work presented in this thesis not only extends the model in [10] but also proposes a new model to solve the problems that arose previously. In the interest of the goal of the project, the new model was designed to be simpler and not rely on the use of outside SNN simulators. The new model designed is a single layer neural network aiming to provide the same output as the other model given identical input, while being small enough to directly program on Loihi. The new model created was put through the same tests as the previous model and completed them with results as good, if not better, than the previous model while running on the CPU, and is currently being implemented on Loihi.

CHAPTER II

BACKGROUND

2.1 Spiking Neural Networks

Spiking neural networks (SNNs) are a family of artificial neural networks (ANNs) that imitate the spiking behavior of biological neurons. SNNs differ from traditional ANNs in that the former spikes only when it needs to send information, and the later updates continually. This type of network follows that of a biological neural network more closely than traditional ANNs. Since SNNs do not have to communicate continuously, they typically consume less power that ANN algorithms [11] [12].

Multiple approaches exist for developing SNNs for traditional AI, i.e. analytics or machine learning, applications. One approach is to encode inputs into spikes then connect the input neurons to output neurons and adjust the weights between them to get the desired output. Yusoff et al. show this in [13]. They show this when trying to associate color and words together, setting up neurons representing each state and then stimulating both until the connection is formed via Spike-timing-dependent plasticity (STDP), or a biological representation of adjusting the connection strength between neurons. Additional approaches exist, but STDP has enabled deploying SNNs on traditional data problems. Right now, the focus is on trying to implement current ANNs into SNN form as they are more tried and true than developing a newly designed SNN. Lisitsa et al. [14] discuss the prospects for creating SNNs and show how the importance of ANNs drives innovation for SNNs. In addition, papers also try to theorize how SNNs can be useful, looking into their benefits and testing how they would/could compare to traditional methods.

2.2 The Neural Engineering Framework

SNNs can be encoded through various forms. One available software library for instantiating SNNs is the Nengo python library. Nengo models SNNs based on the Neural Engineering Framework (NEF) described more fully in [11], and based on these three principles:

- Neural representations are defined by the combination of nonlinear encoding (exemplified by neuron tuning curves) and weighted linear decoding.
- Transformations of neural representations are functions of variables that are represented by neural populations. Transformations are determined using an alternately weighted linear decoding (i.e., the transformational decoding as opposed to the representational decoding).
- Neural dynamics are characterized by considering neural representations as control-theoretic state variables. Thus, the dynamics of neurobiological systems can be analyzed using control theory.

This framework is leveraged to implement the associative memory model and the results described in this paper.

2.3 Semantic Pointer Architecture

One important component used in the model described in this paper is the Semantic Pointer Architecture (SPA) described in [15] and used in conjunction with the NEF. SPA is a type of cognitive architecture that uses a high dimensional vector to represent different concepts and uses various mathematical operations to relate the concepts together. The

high dimensional vector representations are referred to as semantic pointers and thus getting them to interact in various ways creates the architecture. This architecture is important because it will be the foundation for how concepts are learned and recalled in the model presented in this paper. NEF and SPA were used in this research to develop biologically inspired cognitive constructs.

2.4 Associative Memory

Associative memory is an important aspect when developing cognitive models and can be observed in different biological cognition's [16][17]. One relatable example is how humans can associate a kitchen with cooking, an alarm with waking up in the morning, or even a smell with a certain person. This is further seen in other creatures such as when a dog anticipates going on a walk when its owner gets a leash, or when the food cabinet is opened it may know it is dinner time. For this paper, associative memory is understood as relating two "concepts" together. Herein, these concepts are signals, whereby stimulating one signal it brings about the other in the memory. Implementing this concept in AI could allow for easy learning and the ability to connect and simplify a large network of signals down to a comprehensive representation.

Prior research has found the importance of associated memory in higher cognition, for instance, Sandamirskaya et al. [18] presented an in-depth discussion on a stance toward higher cognition and emphasized the importance of associative memory and what role it can play in developing higher cognition. Horzyk et al. [19] introduced a new associative tuning algorithm for SNNs. They discuss how to train the networks and how to construct hetero associative memories. The work from both papers is important as the development

of systems, like the one presented in this paper, is used to shape the direction this field is heading, pushing for higher cognition.

He et al. [20], constructed an SNN based associative memory using Hebb's rule for STDP and reinforcement learning to optimize the weights of the synapses between neurons. Beyond this work, Bekolay et al. [21], and Voelker [22], describe how the Prescribed Error Sensitivity (PES) rule, a biologically plausible form of supervised learning elaborated on in chapter III, works and discusses the benefits and power of using these rules in an SNN since they closely relate to functions seen in biology.

In each paper discussed above, the focus seems to be on very low-level implementation, and less on higher representations. STDP is a common factor among all the papers using it as the main agent to update weights between connections. As discussed in this short review, current research focuses on neuron to neuron association, how to connect and adjust weights between neurons, and less about the representation abstracted from the neurons.

2.5 Neuromorphic Hardware

Neuromorphic hardware appears to be growing in more research areas, such as Intel's Neuromorphic Research Community (INRC) [2], for its unique architecture that tries to mimic components of a brain. Neuromorphic hardware is designed in such a way that there is a mesh of connected cores that imitate neurons and send spikes out to one another rather than traditional hardware that follows clock cycles to perform calculations serially. This is extremely important as it can support running SNNs along with decreases in computational energy and time.

One current instantiation of the state of the art is the prototypic Intel Loihi neuromorphic processor. Fig. 2.1 gives a unique look at the architecture of the chip. Loihi is currently a research chip featuring 128 neuromorphic cores with 3 embedded x86 processors and off-chip communication interfaces [9]. Loihi has cores to simulate neurons or groups of neurons and channels to connect them [9]. The architecture is designed asynchronously such that spiking activity can flow freely; the inspiration for such asynchronicity being biological brain and neuron activity. Loihi also supports Nengo, the software mentioned above, making it a very appealing device for use whereby the focus can be on conceptual design.

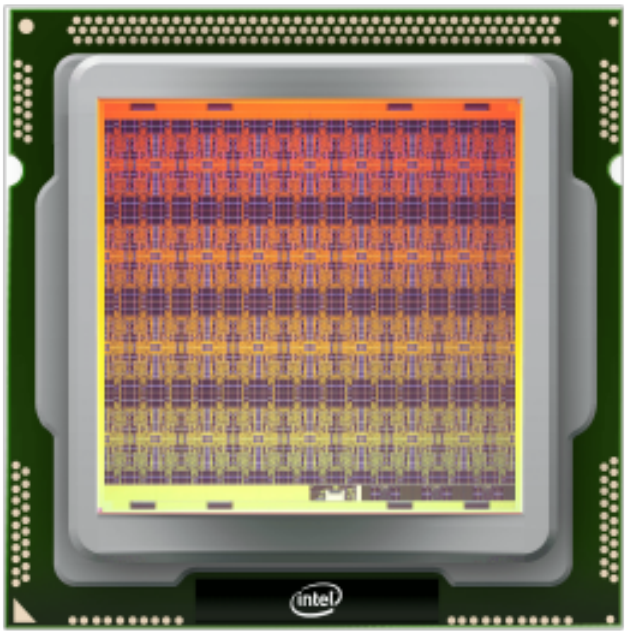

Figure 2.1: Intel Loihi Chip Architecture, picture taken from [1]

One prominent feature of Loihi, as well as many pieces of neuromorphic hardware, is that the chips can be easily connected on board to create large networks. In the case of Loihi, there are different prototype boards containing multiple Loihi chips. The primary version of Loihi used in this work is the Intel Loihi Kapoho Bay which is a board containing

2 Loihi chips with a USB port as well as options to connect either another board or spiking DVS cameras [2], shown in Fig. 2.2. In addition the Nahuku board, a prototype board containing 32 Loihi chips was also used as the board provided energy measuring capabilities that the Kapoho Bay lacked, Fig. 2.3.

Figure 2.2: Intel Loihi Kapoho Bay USB, picture taken from [2]

Neuromorphic hardware offers lots of benefits such as lower power consumption and latency due to its unique architecture [23]. Most of the work done in this area is geared toward robotics, including the development of autonomous robotic agents. . Glatz et al. [24] is a perfect example of how the use of neuromorphic hardware is aiding in this field. Their work is on trying to create an adaptive motor control given mixed input signals. This again takes advantage of neuromorphic hardware's ability to support learning and change weights of connections over time. Glatz et al. b6 also investigated the power savings when switching to the neuromorphic hardware. These features of learning and increased power saving of neuromorphic hardware, help to make the transition toward having autonomous

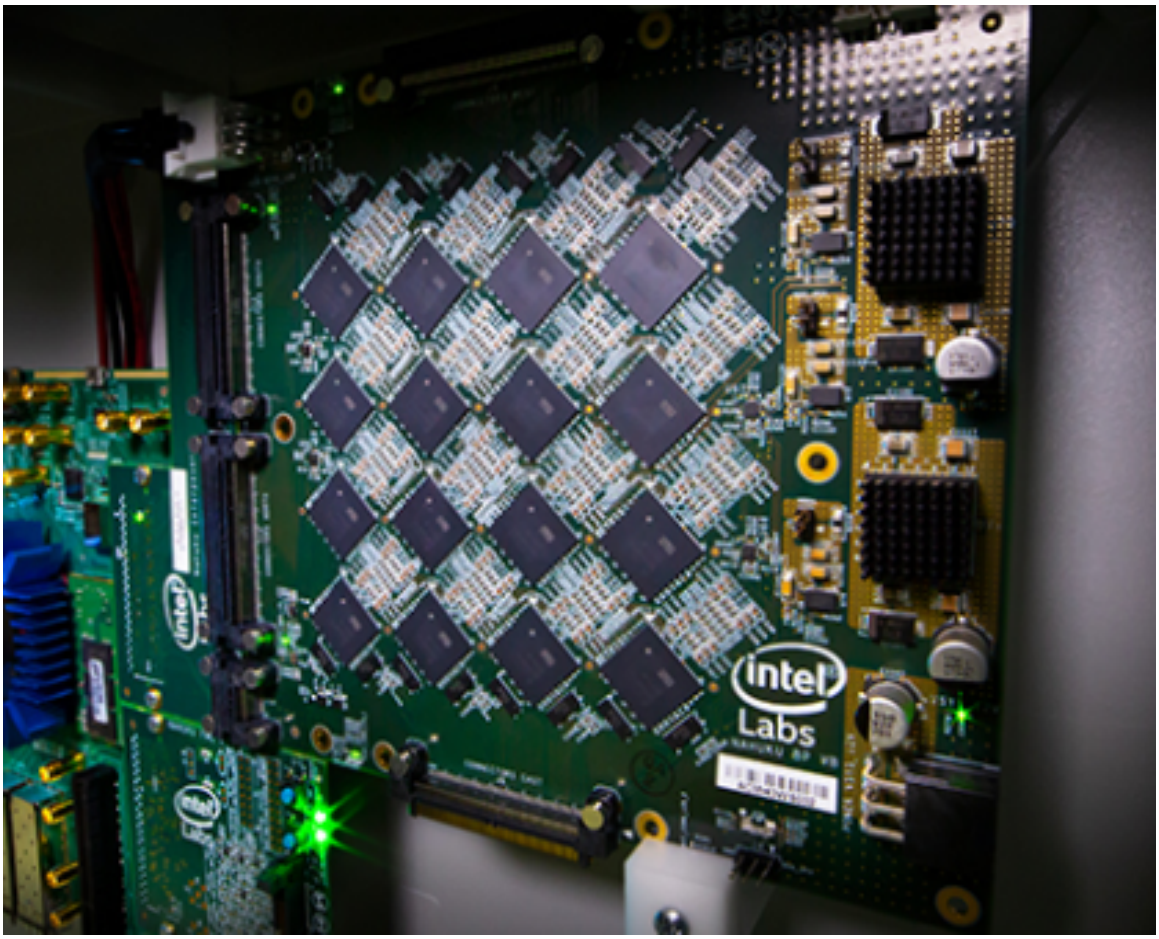

Figure 2.3: Intel Loihi Nahuku Board, picture from [3]

agents take over more daily tasks much more appealing. In many cases adding these agents to take over daily tasks can reduce cost, hazards, mistakes, and time.

CHAPTER III

ASSOCIATIVE MEMORY MODEL AND SETUP

3.1 Associative Memory Model and Online Learning

To provide a starting point for an associative memory model, this thesis will begin with the "Learning New Associations" example in Nengo [25]. The model from "Learning New Associations" is shown in Fig. 3.1, giving the general overview of how the model performs the association given two inputs or a Key-Value pair. In this form, the Key-Value pair could be any set of signals as long as they can be converted into a semantic pointer by the concept used to describe the signal.

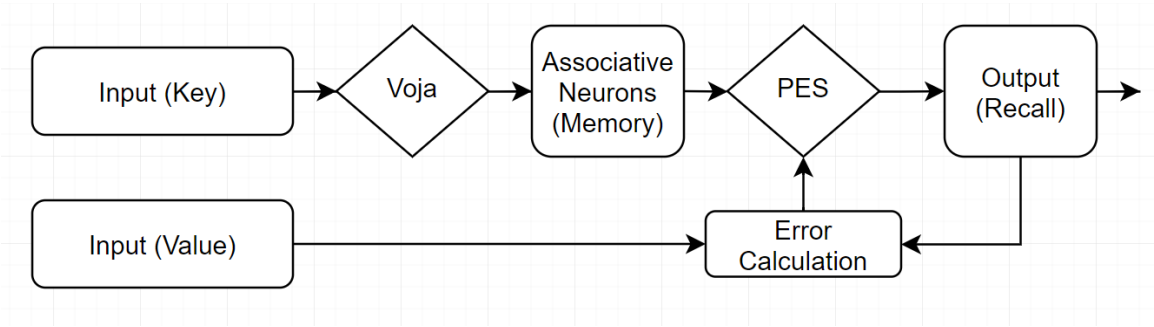

Figure 3.1: Associative Memory Model.

Essentially the model shown in Fig. 3.1 is given two inputs, which could be any two signals that are desired to be associated together and can be represented using multidimensional arrays. The model then uses a form of unsupervised learning and supervised learning to adjust the weighted connections between the input and output.

The form of unsupervised learning is based on Oja's rule, but in order to work with the spiking neurons and vector representations as described in the NEF, it is modified slightly

to work using vectors and called the Vector Oja rule, or Voja. Voelker et al. [26] presented

$$\triangle e = \eta(a(t)x^T - a(t)e) = \eta a(t)(x^T - e) \tag{3.1}$$

as how to calculate the adjustment of weights for the Voja connection, given that η is the learning rate, x is the input vector, $a(t)$ is the activity of the neurons and e is the encoder matrix of the neurons. Continuing from Voelker et al, "Thus the effect of this rule is to make a subset of the middle layer neurons fire only when x or a noisy version of it, is presented" [26]. In other words, equation (3.1) limits the number of representations of a single input by filtering the noise.

The second part of this model shows the supervised learning rule that affects the weights between the associative neuron group and the recall neuron group. This is done using PES described in Voelker [22] as "a biologically plausible supervised learning rule that is frequently used with the NEF. PES modifies the connection weights between populations of neurons to minimize an external error signal." Essentially this rule allows the model to compare the output from the memory to the second input signal that it is trying to learn, and then adjust the weights in the connection to obtain that result. This is represented as

$$\triangle d = kEa \tag{3.2}$$

where the change in weights given learning rate k, activity a, and error signal E. With these two learning rules in place, the model is now able to learn associations given two distinct input signals.

An important aspect of this associative memory model to note is that it has incorporated online learning, such that the model can learn while it is running. Both the Voja and PES learning rules, equations (3.1) and (3.2), act on the weighted connections in real-time when running. This allows the model to change and learn as new input comes into the system.

This is a valuable aspect of the model as not every application of neural networks has data to train on or can be pre-trained before use. This allows the model to not only be used in real-time but also to adapt over time.

3.2 Hardware Implementation and Demonstration Platform

To provide for a real-world demonstration of these abilities, the Turtlebot-2 platform was used. The Turtlebot is a small robot that can be controlled using the open-source Robot Operating System (ROS), shown in Fig 3.2. Detailed description of the sensors and compute components is given in section 3.4. ROS is an open-source software created to help easily control robots by providing libraries that can communicate between different parts of a robot. In this project, it is used to control the sensors and communicate information to the cognitive model through ROS topics. The topics are set up such that all the processes running on the Turtlebot can publish or subscribe to them to share information. The purpose of running this system on the Turtlebot to demonstrate feasibility on a real world robot which introduces complexity not found in modeling and simulation.

3.3 System Constraints

The purpose of implementing this associative memory model in an SNN instead of an ANN is the power and time advantages that come about when running this network on neuromorphic hardware. For this project, the model was run on the Loihi chip, and since the purpose of this experiment is to see if there is an advantage to using SNNs, the model will be compared to itself running on an UP board. The UP board, manufactured by UP-Board, is a compact x86 embedded board designed to give high performance with low power consumption [8]. It was chosen since it is small in size and weight, low power, but

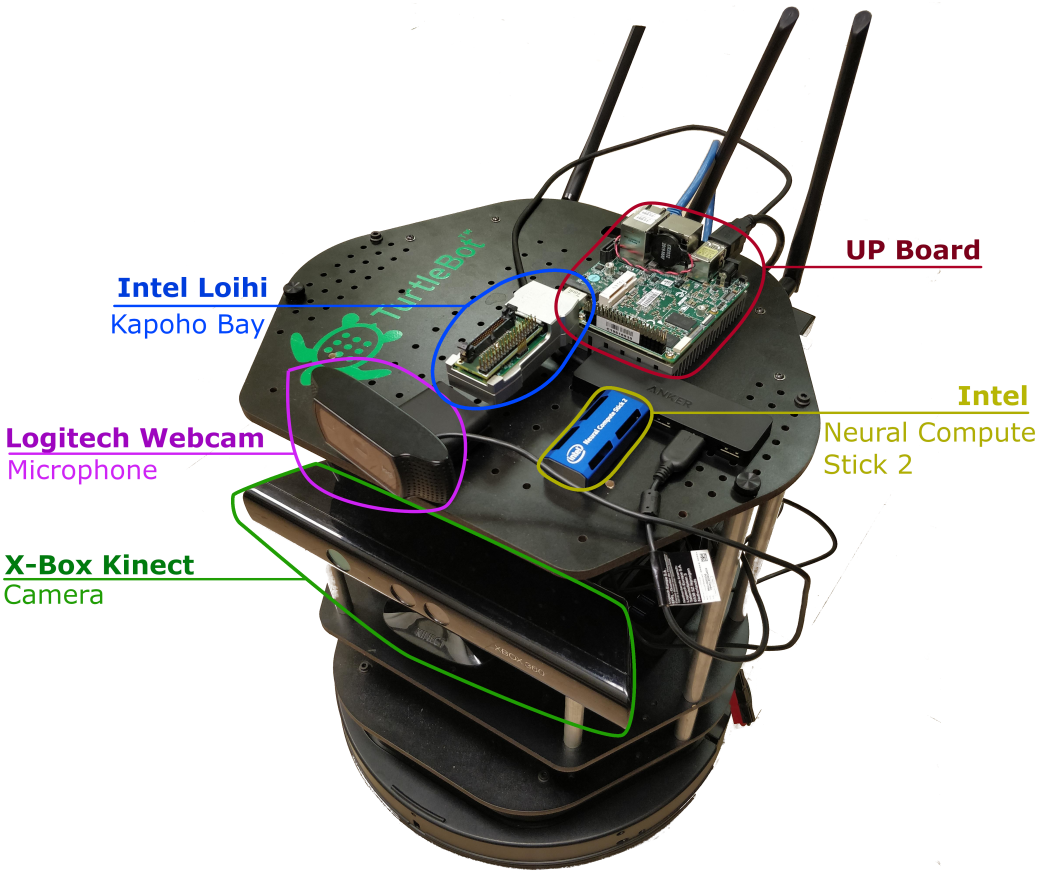

Figure 3.2: Hardware Set up on Turtlebot

has full x86 capable system running. This allowed a platform for easy development and was able to give a reasonable comparison, in terms of size and power consumption, to the Loihi hardware.

3.4 Sensors and Set up

The associative memory model as shown in Fig. 3.1 is set up to interact with the Turtlebot through ROS by reading input signals and writing output signals. The two input signals used in this project are visual and audio, coming from an Xbox Kinect camera and a Logitech microphone. Each component shown in Fig. 3.2 was connected to an UP board and set up on a Turtlebot. Images from the camera are fed into the Intel Neural Compute Stick, which was running a convolutional neural network (CNN) to recognize everyday objects. The CNN on the Neural Compute Stick was loaded with pre-trained weights from an open-source setup [27], that when given an image simply outputs the name of the most prevalent object in the image as a string. In this project, the network was only allowed to output an object from a predefined list, and if it did not recognize anything would output the string "none". The string output of the CNN was sent to the associative model, via a ROS topic, and the model was able to convert the string into a semantic pointer. The microphone input operated on a similar premise, where the audio signal was sent to a process running on the Up Board that determines the frequency of the sound. If the frequency of the sound was within the range of predefined options, then it would output that frequency as a string to the associative model via a ROS node. Just as with the image signal, the audio signal was converted to a semantic pointer and input into the model.

3.5 Associative Memory Model

By extending the model in Fig. 3.1, we developed the multi-input model seen in Fig. 3.3. This model can take input from a source, in this case, the sensors on the Turtlebot via ROS, and associate two different input signals together, such that when one is seen the model can recall the other. The model has been modified to not only accept input from ROS but also use the SPA to represent different predefined concepts. Starting on the far left of Fig. 3.3 there is a "robot" node that takes input from ROS topics and feeds it into the model. The model passes the input to the "detection0" and "detection1" nodes, where they are converted to semantic pointers. The semantic pointers are then fed into the "key" and "value" nodes where they are interpreted as a key-value pair. The model then uses the two learning rules, Voja and PES, to adjust the weights of the connections between the "key" and "memory", and "memory" and "recall" nodes. The learning connections are shown as green dashed lines. This allows the model to arrive at the same value the key was paired with during the learning phase. Once a signal is recalled the model can have a predefined reaction that is sent back to the source.

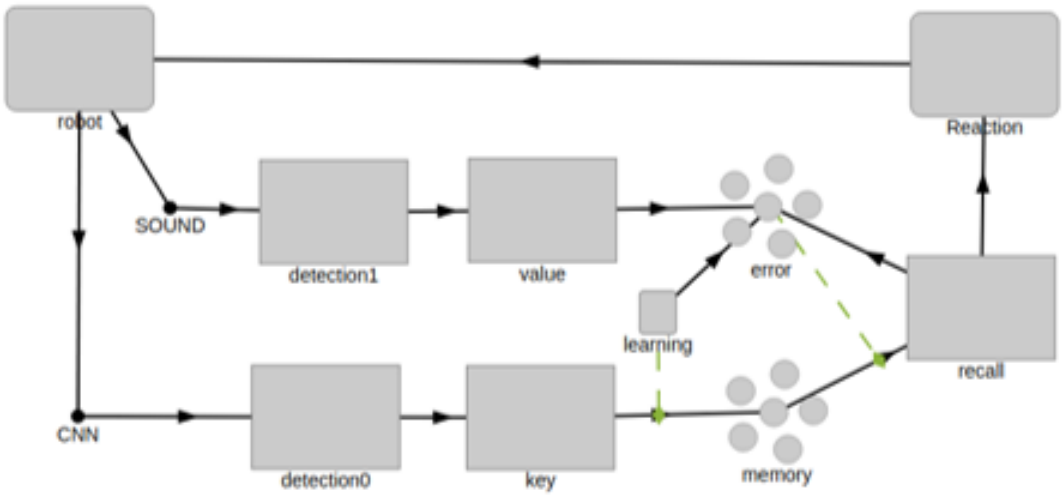

Figure 3.3: Input to the Model

CHAPTER IV

RESULTS

4.1 Initial Results

The first test of the model was to reproduce similar results presented in [26] showing how the model decreased in accuracy as the number of associations increased. Sensor data was collected from the camera and microphone and saved to use for testing. Table 4.1 shows the key-value pairs that are input into the model using the sensors on the Turtlebot. An image of a vehicle was shown to the camera, via a picture, and a tone between 500 and 2000 Hz was played at the same time. The two signals are classified, and their labels are then sent into Nengo where they are converted to semantic pointers and processed. The model was run on the Up board 100 times given the same input, with seeded values, and a varying number of associations from 1 to 4. The model had one second to train on each association and was then required to recall that same association for 1 second. When the model had to learn more than one association, all the training was done first before the model was tested on recalling.

Table 4.1: Key-Value Association Inputs

Association #	Key (Image)	Value (Sound)
1	CAR	SOUND_500
2	AEROPLANE	SOUND_1000
3	BICYCLE	SOUND_1500
4	BOAT	SOUND_2000

The root means square error (RMSE) was calculated using

$$\sqrt{\frac{\sum_{t=1}^{T}\sum_{n=1}^{N}(V-R)^2}{T}} \tag{4.1}$$

where V and R are the value and recall vectors, N is the number of dimensions in the vectors, and T is the number of time steps in the run. The accuracy obtained for each run was then averaged together over the 100 trials. Fig. 4.1 shows the %RMSE, RMSE values calculated divided by the max possible RMSE, for these trials for each number of associations as well as the standard deviation error bars. The test was then run again with ideal input, which is the input that was calculated precisely and fed into the model at the proper time steps.

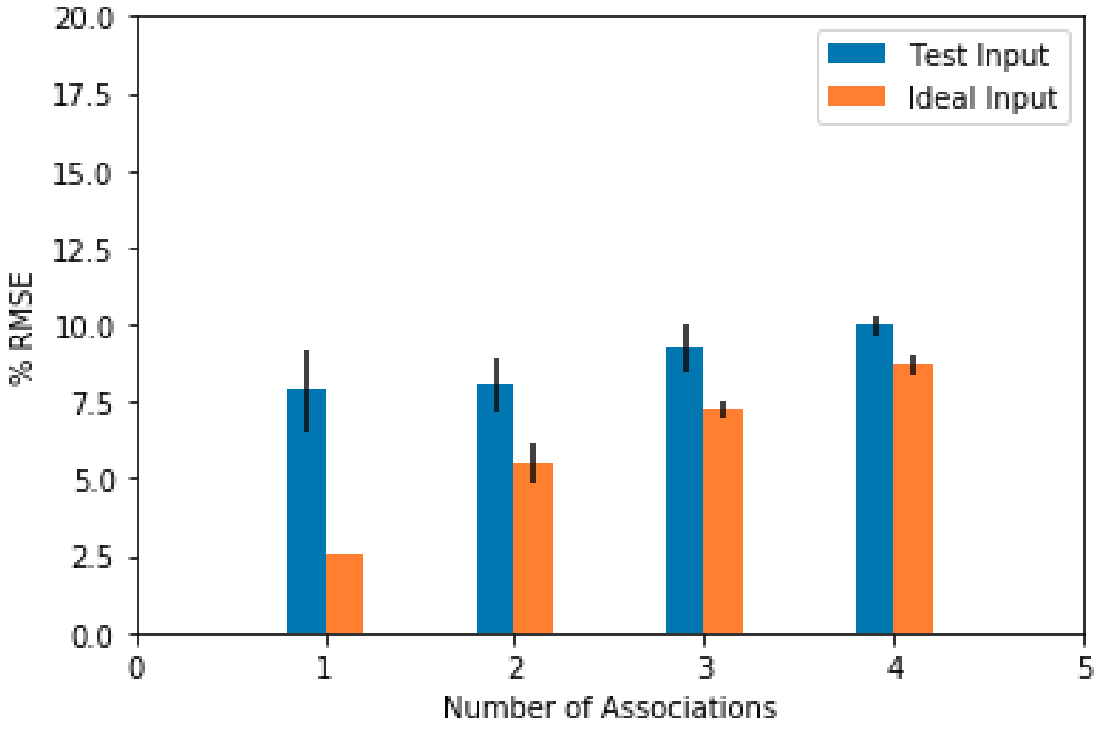

Figure 4.1: %RMSE on Up Board with Learning with Standard Deviation

The test was run twice, once with test input gathered from the sensors, and once with the ideal input, for a few reasons. The first being to see how well the model could potentially perform given near perfect input. The second being to see how the model reacts to live

input and to showcase the possibility of running this model in a real environment. The third, to get a comparison between real test input and ideal input, showing how well the model can handle imperfect and or noisy data. Fig. 4.2 shows an example of the error during the recall of this first test with the top figure being from the ideal input and the bottom figure being from the test input.

The input is the only difference between the two plots. The number of associations recalled was four, and the seed value was the same. Since the test input is not as clear-cut and precise as the ideal, it leads to more error on average as well as error when switching between recalls. This can be seen in the bottom graph in Fig. 4.2, where there are large spikes during the switching of input signals. This is because there is uncertainty in the test data as it switches, the sensors can have moments where it is unsure what the image or sound is being observed. With the ideal input that data is perfectly recognized at each time step, preventing any moments of lost data and uncertainty.

As seen in Fig. 4.1, the RMSE is less than 10% given 4 associations to remember. However, it's more important to see how that error affects what the model chooses to recall. To check this the recall vector is compared to a list of semantic pointers created to determine which semantic pointer the recall is closest to. Fig. 4.3 shows the similarity among all the semantic pointers, with the top figure using the output of the "recall" node and the bottom using the output of the "value" node to show the correct choice. Since there is error in the model, the recall vectors are not exactly like the original vectors. However, the recall vectors are still far more similar than any of the other choices, and thus in each case, the model chooses the correct option.

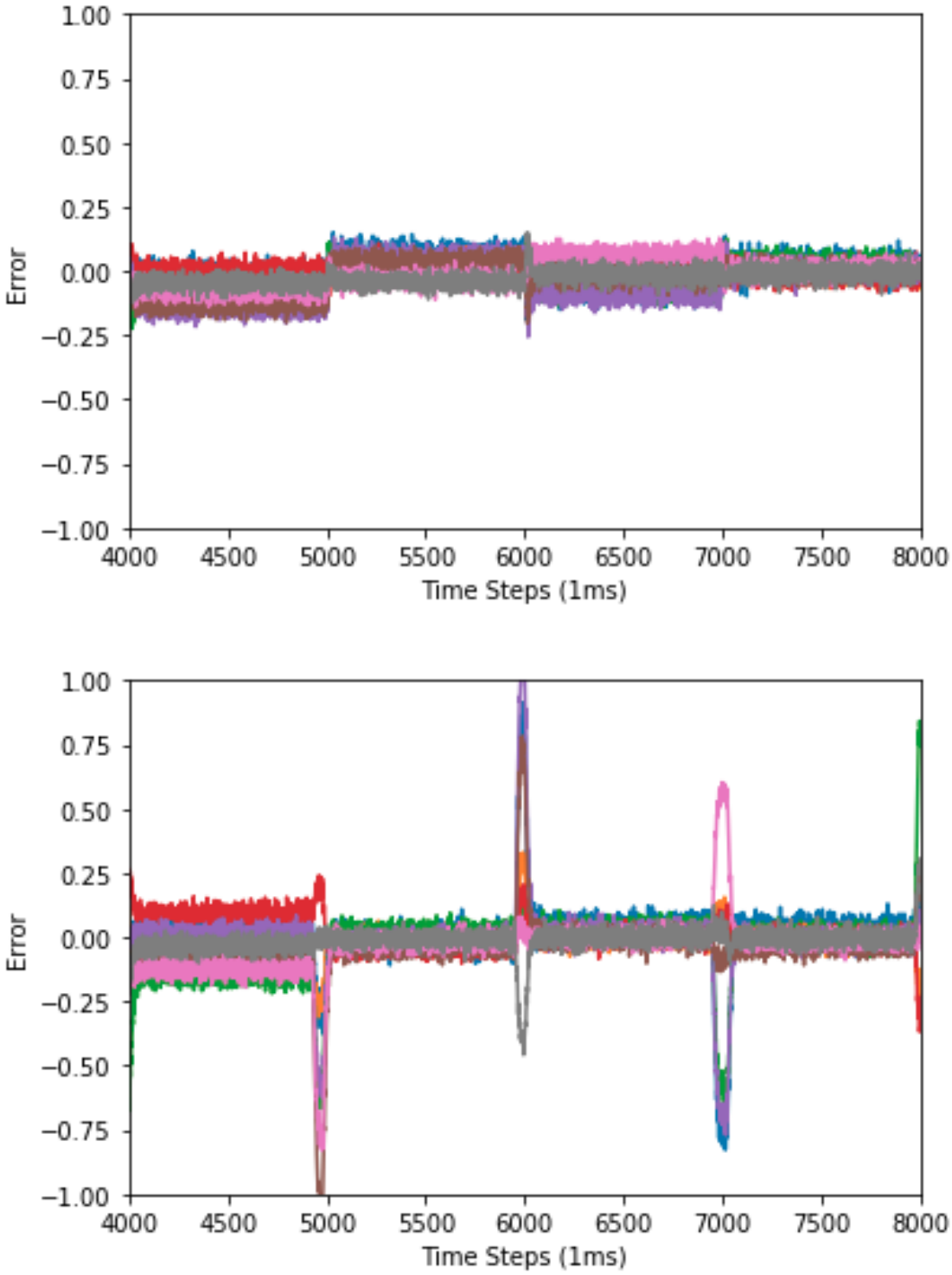

Figure 4.2: Error Comparison between Test and Ideal Input

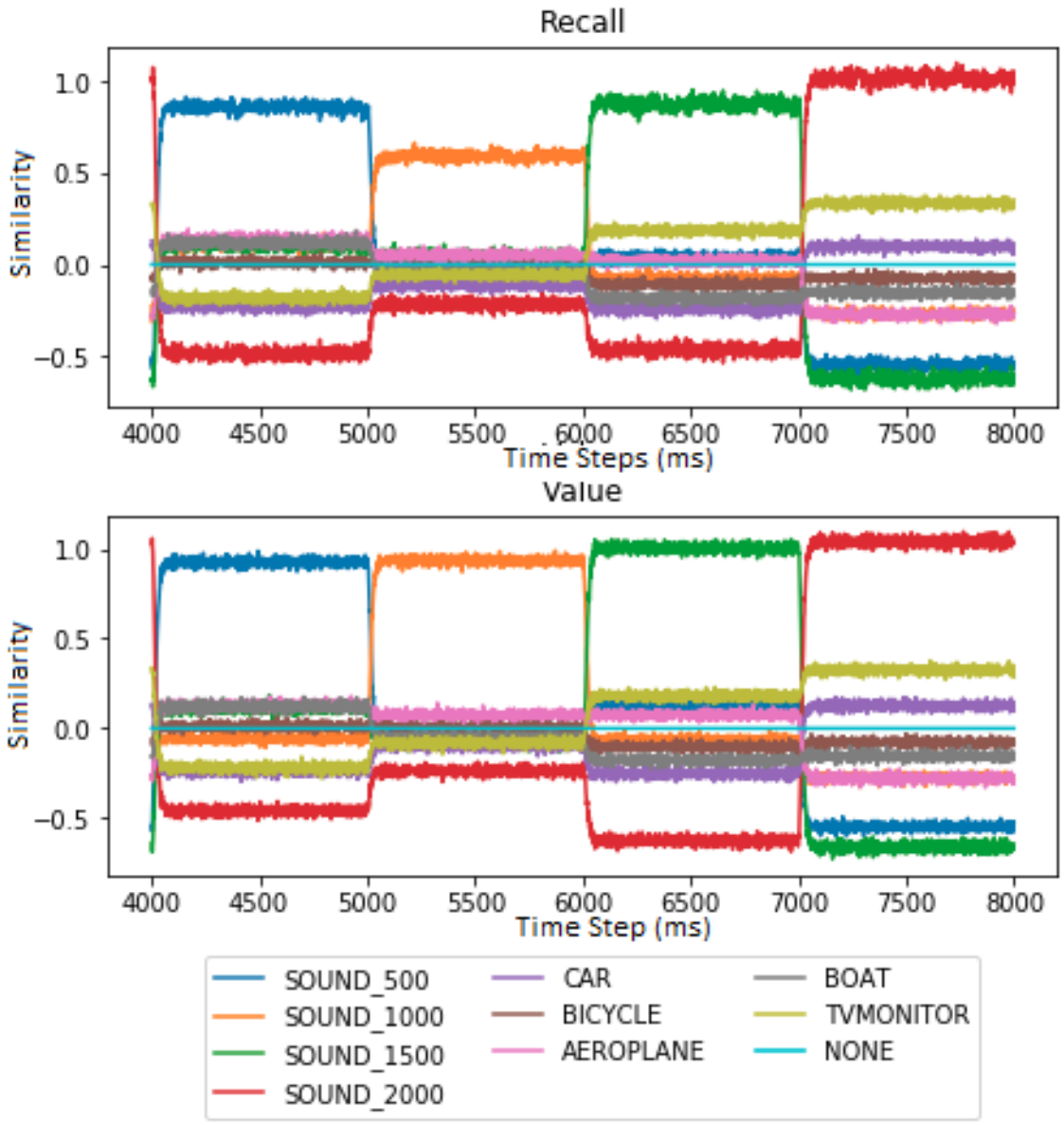

Figure 4.3: Recall Similarity on Up Board with Learning

Figure 4.3 is showing the similarity between the recall signal and the correct associated signal. A value of ne would mean the signals are identical and a value of negative one would mean it is identical to the inverse. The x-axis is time (ms) plotting at each time-step. The bottom graph showing the expected results, as it is expected to be at a value of one for each of the associations showing it can produce a signal identical to the correct associated signal.

While running the first test, the weights of the connections that Voja and PES modify were saved. The same test was run again except now the weights that were saved from the first test were loaded into the model. This was done to see how well the model can perform if it is pre-trained instead of trained online. The %RMSE was calculated for accuracy once again, averaged over 100 runs for each number of associations, and shown in Fig. 4.4. The accuracy shown in the figure is nearly identical, as expected, to the accuracy when the model trained just before it had to recall. This shows that there is no difference in how the model reacts if the training is done while the model is running or beforehand.

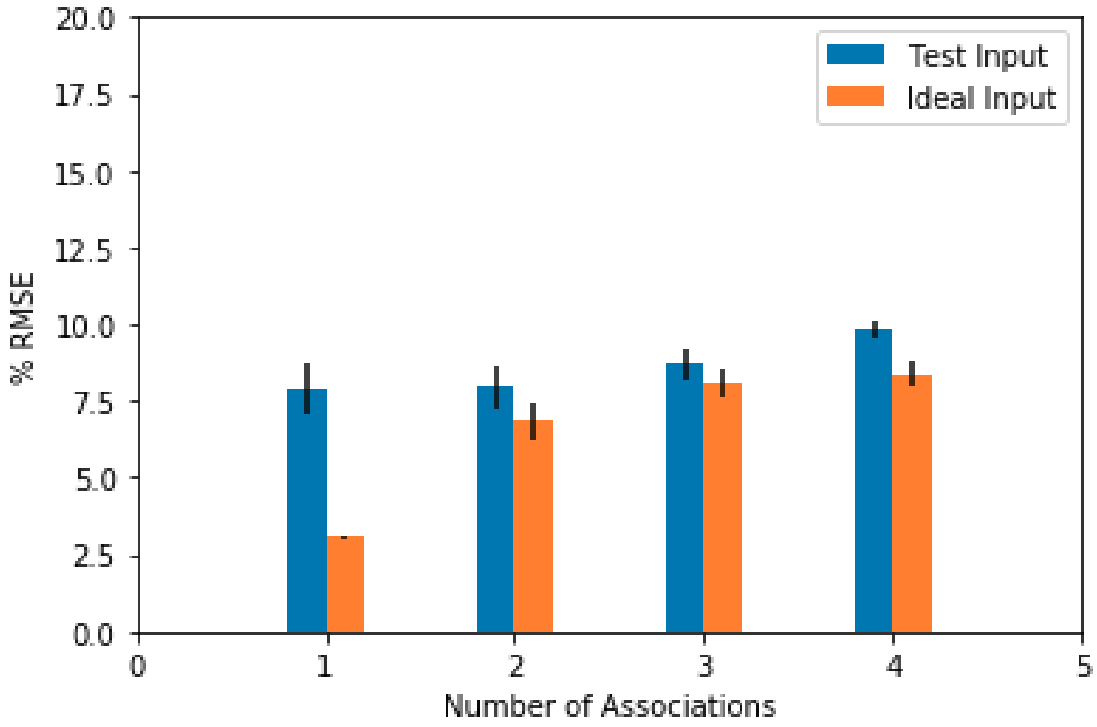

Figure 4.4: %RMSE on Up Board with Pre-Trained Model with Standard Deviation

The final test on the associative memory model was to see how well it performed on Intel's Loihi. The development of the Nengo Loihi back end is still in the early stages, so

learning rules in this model do not work as intended on Loihi. With that restriction, the final test is the same as the previous one in which a pre-trained version of the model will be uploaded to Loihi and run. The same 100 trials with the number of associations varying up to 4 were run and the %RMSE was calculated just as before. Fig 4.5 shows the results.

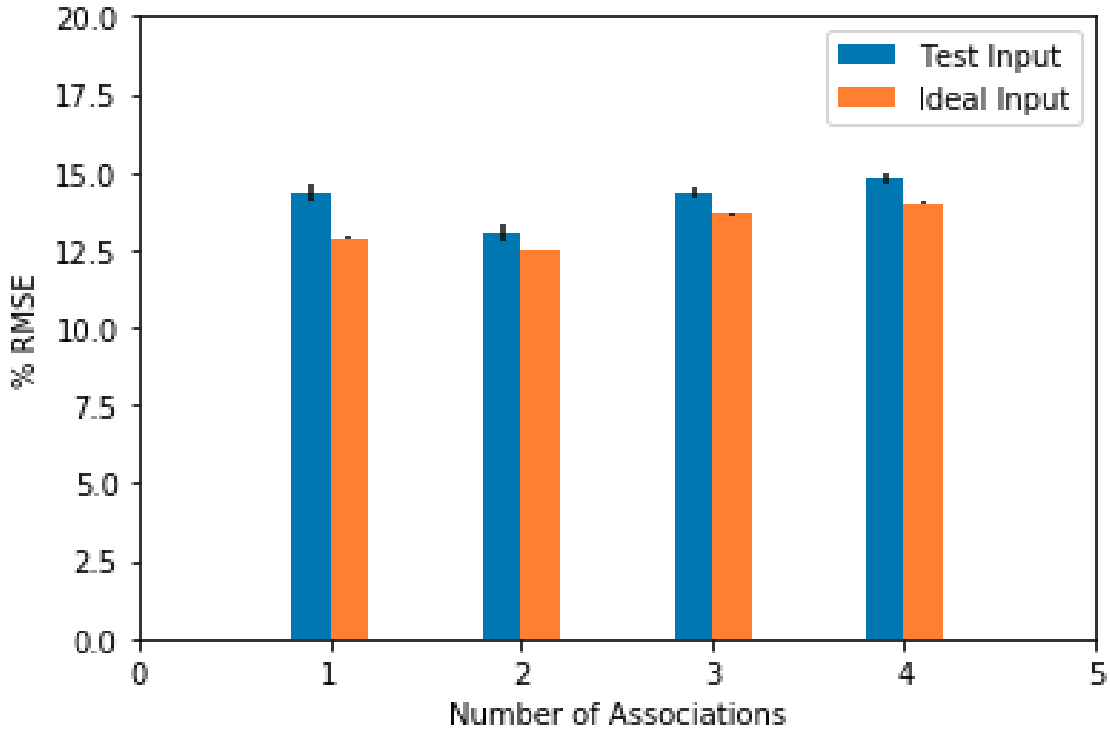

Figure 4.5: %RMSE on Loihi with Pre-Trained Model with Standard Deviation

If the process of implementing this model on the new hardware was perfect, the results in Fig. 4.5 should look identical to that in Fig. 4.4. However, the %RMSE when running on Loihi increased anywhere from 4.5% to 7.5% for both types of input across all four groups of associations. This is due to Loihi working in nine-bit precision for synaptic weights between neurons [9] whereas the model was trained beforehand in sixty-four-bit floating-point precision, thus there is some loss due to the conversion when the weights are loaded

onto Loihi. When looking at the error for one particular trial, Fig. 4.6, there is significantly more noise at both the lower frequency and higher frequency ranges compared to the recall error on the Up board shown in Fig. 4.2 above. This can be attributed to the fact that Loihi only supports up to a fixed nine-bit precision for synaptic weights between neurons [9]. When the model was trained beforehand, the weights were saved as a sixty-four-bit floating-point precision value, so there is some loss due to the conversion when the weights are loaded onto Loihi.

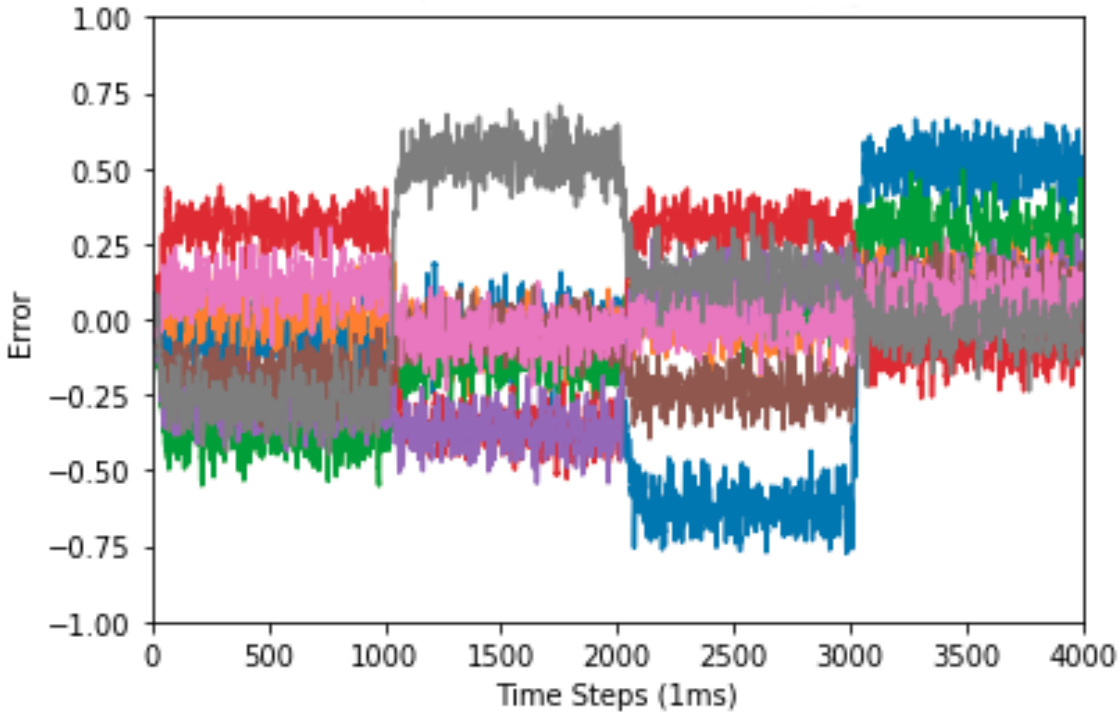

Figure 4.6: Recall Error on One Trial on Loihi

Before taking the time to retrain all the weights, the recall vector was checked against the list of known semantic pointers as it was done in the first test. This is done to check if

the increase in error is too much such that the model can no longer determine the correct output. Fig. 4.7 shows recall results on the top and the ideal values on the bottom.

Even though the similarity appears up to 70% lower, it is still clear that the model can recall the correct choice in each of the four associations it made. However, as shown in the results presented in this paper as well as in [26], the error will increase as the number of associations increase, and with the already low results in Fig. 4.7, it would be expected that the model will not work with a larger number of associations.

4.2 Energy Efficiency

As discussed earlier, a big advantage for SNNs is that when implemented on neuromorphic hardware, they have the potential to be far more energy-efficient than when running on traditional hardware. To evaluate this potential, execution time, power, and energy were measured for the second and third tests. To be more specific the execution time, power, and energy were recorded and averaged over each of the 100 trials with four associations of the pre-trained model running on the Up board and the Loihi hardware. The results are shown in Table 4.2. The results for the Up board were obtained by measuring the dynamic power draw of the board while the tests were running. The results for the Loihi hardware were obtained using the built-in energy probes running on the 8 chip Nahuku board on Intel's research cloud server. The Nahuku board is a board that holds more Loihi chips that Intel has available for INRC members to use. It's main purpose, that the Kapoho Bay lacks, is to provides energy probes for each chip on the board allowing the user to gather accurate energy readings. The model ran on a single chip on the Nahuku board, utilizing 81 of the 128 neuromorphic cores.

Table 4.2: Up Board and Loihi Dynamic Energy Efficiency

Test	Execution Time (seconds)	Power (watts)	Energy (joules)
Loihi Pre-Trained Model	15.865 ±0.921	1.109 ±0.048	17.608 ±1.457
Up Board Pre-Trained Model	20.323 ±1.255	3.035 ±0.757	61.647 ±0.951

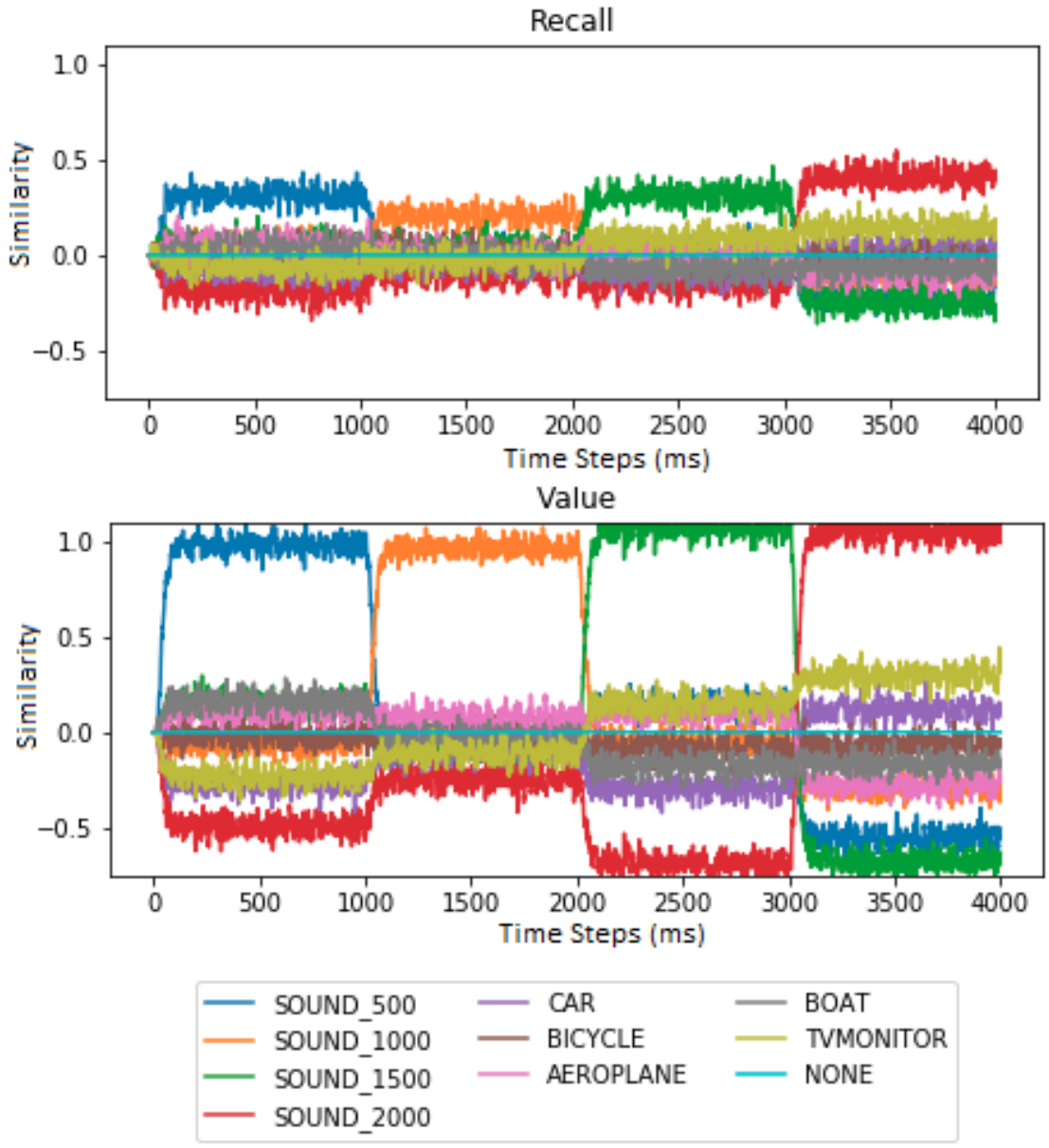

Figure 4.7: Recall Similarity on Loihi

CHAPTER V

INCREASING ASSOCIATIVE CAPABILITY AND ONLINE LEARNING ON LOIHI

5.1 Testing the Capacity of the Associative Memory Model

In Chapter IV, the model was only tested on up to four associative combinations at a time; with four associative combinations, we were able to show the degrading of the model with an increasing number of associations. However, since the results in Chapter IV showed results with four associations, it is of interest to determine a benchmark on the number of associations at which the model would begin to fail. The number of associations was increased from four up until ten sequentially, because having more semantic pointers than dimensions will cause the semantic pointers to not be orthogonal to each other, we expected that eight dimensions would start to cause drastic issues for the model since non-orthogonal vectors have necessarily increased similarity.

Twelve new semantic pointers were defined, six for the key inputs and six for the value inputs, and the new input generation was defined. The model went about the same testing as in Chapter IV, with one second to train each association, one second to recall, and all the training done beforehand. The tests were run multiple times with a varying number of associations from one to ten as well as multiple trials for each configuration. First, the configuration with online learning running on the Up Board was run with the results shown in Fig 5.1.

Objectively Fig. 5.1 displays a visually nicer depreciation curve than that in Fig. 4.1. The model reaches its failing point around 10 associations as the curve that it follows starts to reach an asymptote at around the 12% RMSE. This matches the previous tests as it is the same % RMSE value at which the Loihi started to fail in Fig. 4.5. The weights

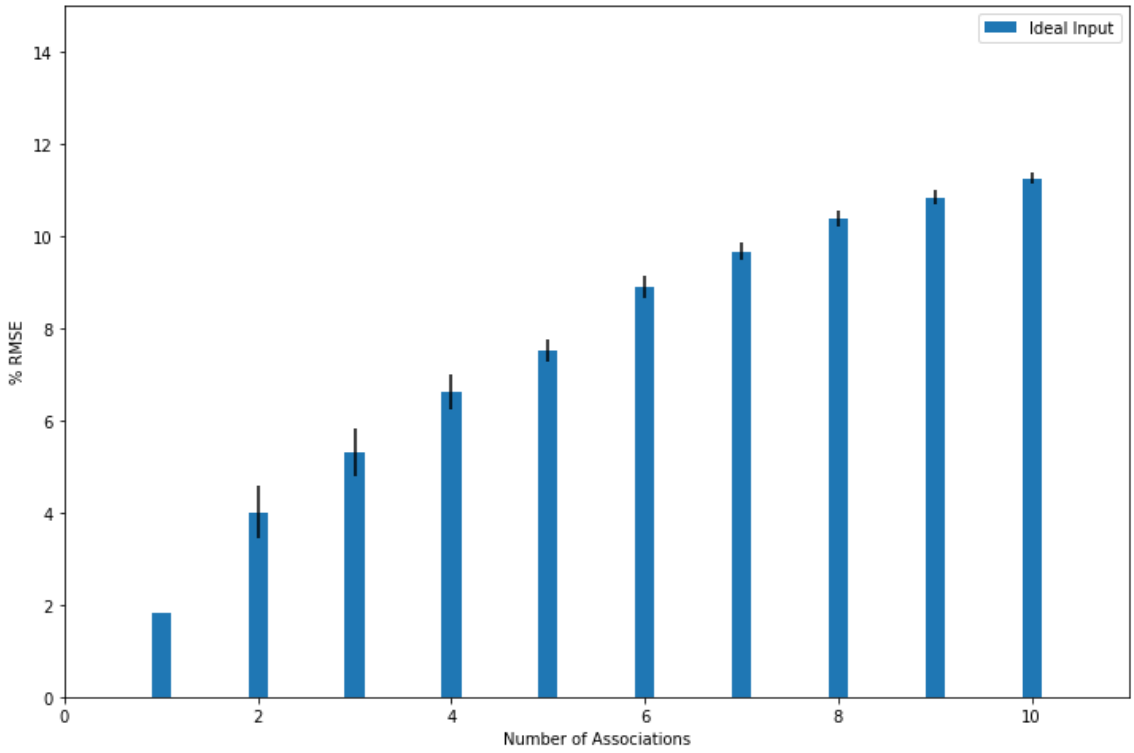

Figure 5.1: %RMSE on Up Board with Learning with Standard Deviation Up to 10 Associations

from this test were saved to run the next configuration of the Up Board with pre-trained weights, just as done before. Fig 5.2 shows the results from that test are near identical to the previous configuration in Fig. 5.1, confirming the results from Chapter IV where there were no differences in the model's performance when using online learning or pre-trained weights.

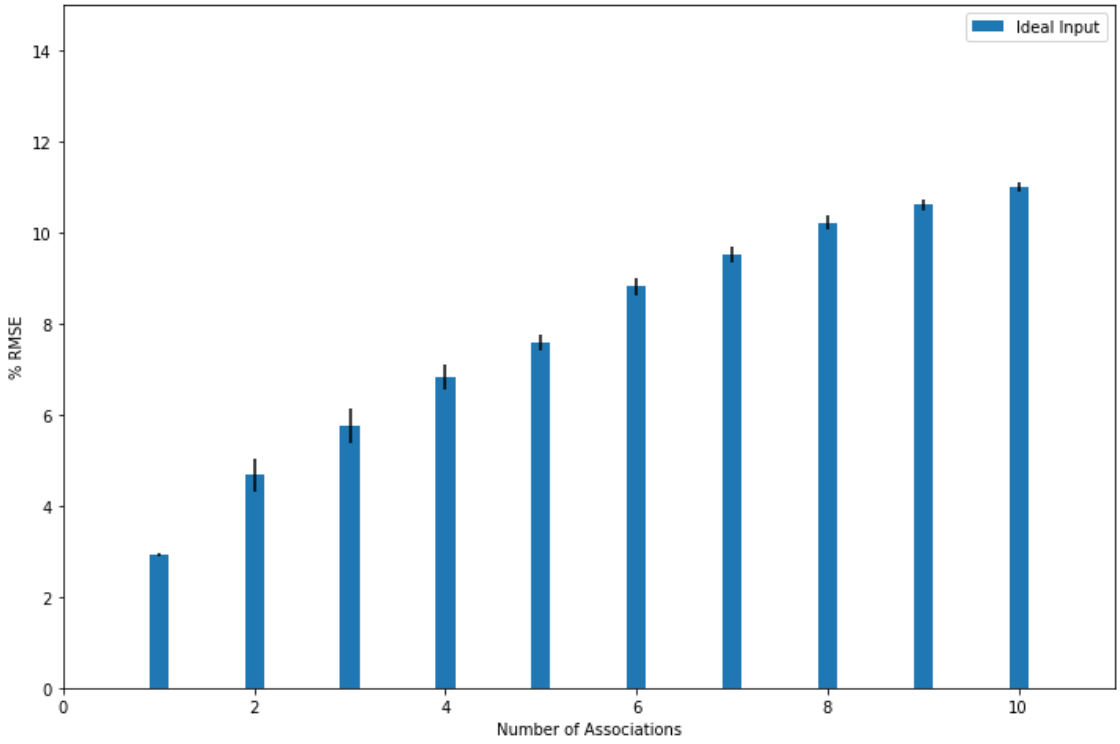

Figure 5.2: %RMSE on Up Board with Pre-Trained Weights and Standard Deviation Up to 10 Associations

Finally, the third configuration that was run on the Loihi was the model using the pre-trained weights from the first configuration. The same procedure was followed as in Chapter IV, using the same input from the previous two configurations. Fig. 5.3 shows the % RMSE of the Recall while running on Loihi and the model reaches its peak right

around 5 associations as afterward the graphs fluctuate up and down within the standard deviation. It can also be seen that it peaks at that similar 12-13% as seen before in the previous configurations confirming that once the model reaches that limit of error it can no longer distinguish the semantic pointers from one another due to the high similarity among them in this configuration.

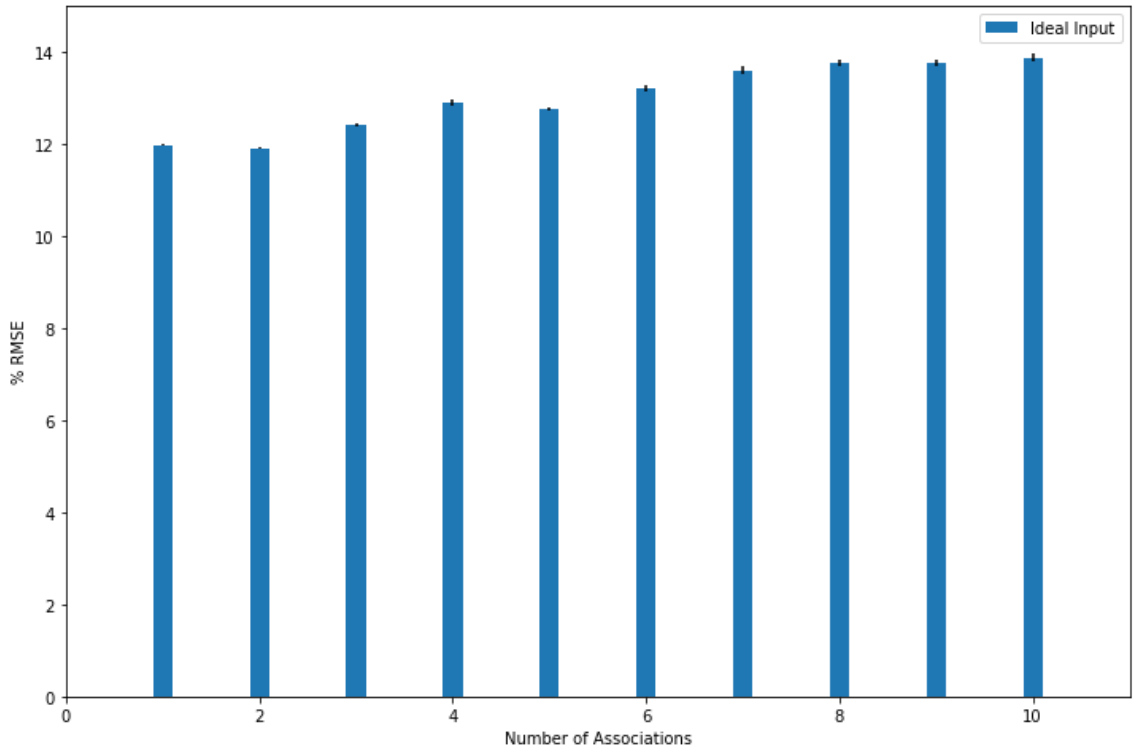

Figure 5.3: %RMSE on Loihi with Pre-Trained Weights and Standard Deviation Up to 10 Associations

One solution to this problem would be to expand the number of dimensions set to define the semantic pointers as it will reduce the similarity amongst each pointer. However, this solution cannot be implemented to run on Loihi since it it would require expanding the number of dimensions by expanding the number of neurons in the model and necessarily

pushing the model across multiple Loihi chips. Currently, the Nengo does not have support for splitting this model across multiple chips; however, another solution will be proposed in the next section.

5.2 Analyzing Problems with the Current Model

In the results preceding, the model was only tested on Loihi with the weights being pre-trained and then uploaded to the board instead of training the weights online. This was the case because the model created uses learning rules from the NEF and designed in Nengo that do not port over to Loihi. Currently, Loihi only provides a few different variables in the connections between neuron compartments, such that the learning is restricted to several two-factor learning rules [9]. The model described earlier uses a three-factor learning rule, meaning that it needs an additional input, error calculation, to perform the learning [28]. Now for that calculation to be performed on Loihi, the data must come off the chip to the processor on the device and then back again to the chip. This causes an enormous amount of overhead to the point where it is not even comparable to running the model completely on a CPU. In addition, the learning rules also limit Nengo's ability to compile the model across multiple chips limiting the size of the model, as described in the section previously.

5.3 Creating a Simpler Network

Objectively the goal is to have an associative memory model running independently on Loihi such that it can obtain the energy efficiency as well as the increased speed that was seen previously without the need to train beforehand. The only way this can be accomplished is if the model can leverage the learning capabilities that Intel provides on Loihi. To do this, a few popular neural networks were looked at to see if they can not only replicate an

associative memory but also be converted into an SNN. The most common neural network that is used as a type of associative memory is the Hopfield Network [29] [30]. However, the major problem is that the Hopfield Network does not support online learning, as the weights are derived from a matrix multiplication of a training set of data and cannot be easily updated while running on Loihi.

Since the Hopfield Network is not easily implementable, a single layer perceptron neural network was chosen as its weights can be continually updated while running to support online learning. It is an extremely simple network given the low dimensional test input, and can be implemented onto Loihi without the need of additional software libraries. The network, Fig. 5.4, is a single layer perceptron network with an 8-dimensional input and 4-dimensional output. The 8-dimensional input comes from the Key input and the 4-dimensional output is a selection of the 4 known output possibilities. The model will take note of which Value input was received at the same time as another Key input and train the network to output to that corresponding node when given the same input. For example, if the first Key-Value pair seen is Car and Sound_500, the model will assign the first output node to represent Sound_500 and with the 8-dimensional input from Car will train the network to output a 1 to the first node and 0 to all the rest. When the model receives a 1 from the output nodes it simply must go back and select the corresponding Value vector, in this case, Sound_500, and feed that to the recall node.

The network was created in python and using the same input as in Chapter IV was trained and tested. The network proved to work and was then implemented in Nengo as a node to effectively replace the previous model in the same setup. This helped to test the new network with identical inputs and produce outputs that are comparable to the previous model. Figures 5.5 and 5.6 which are analogous to 4.2 and 4.3, respectively, show the recall

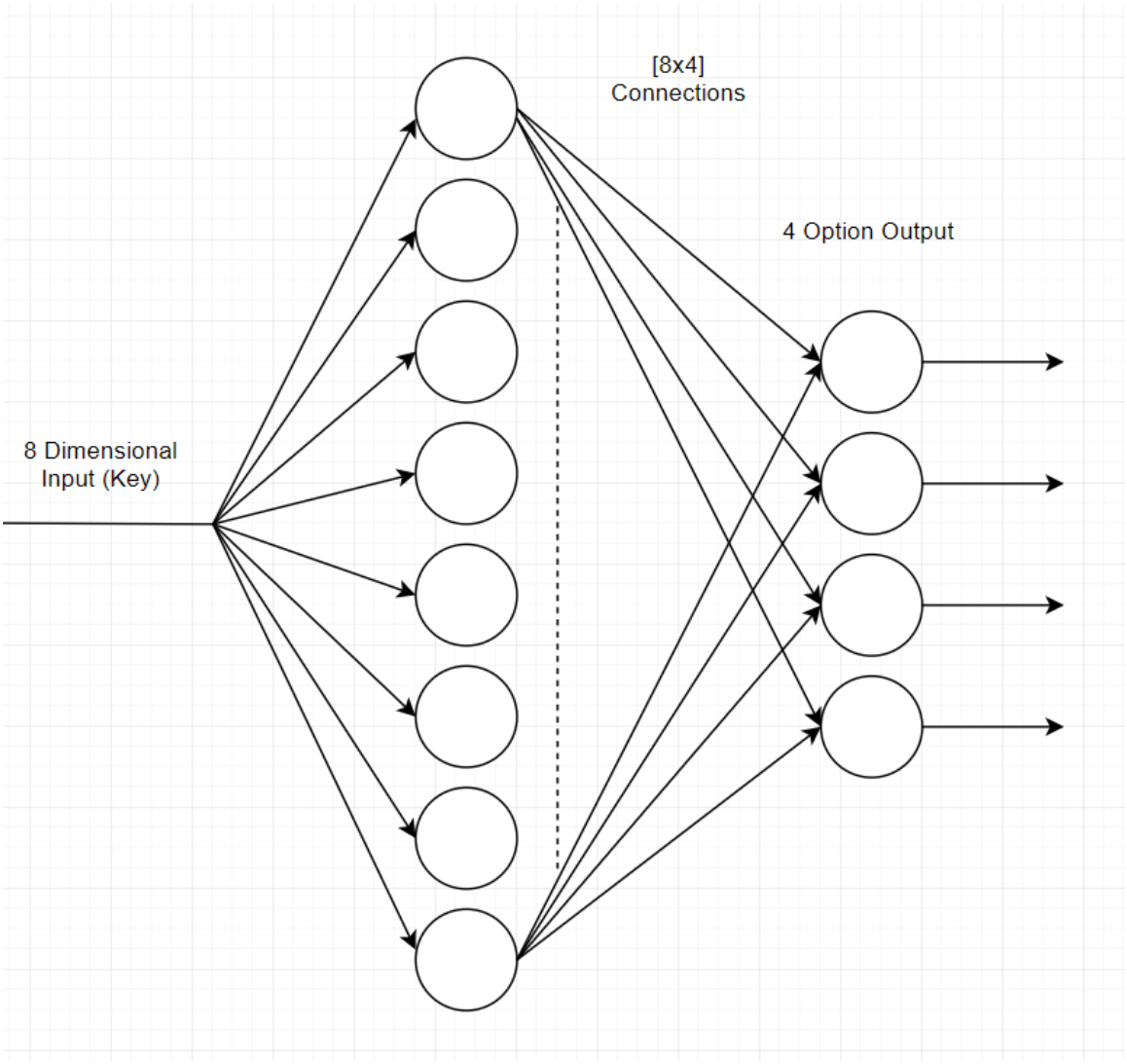

Figure 5.4: Single Layer Perceptron Neural Network

error and similarity graphs respectively while testing the single layer perceptron network designed as an associative memory. The transition spikes can be seen in Fig. 5.5 just as in Fig. 4.2 showing the network has a few milliseconds of lag when transitioning in between associations.

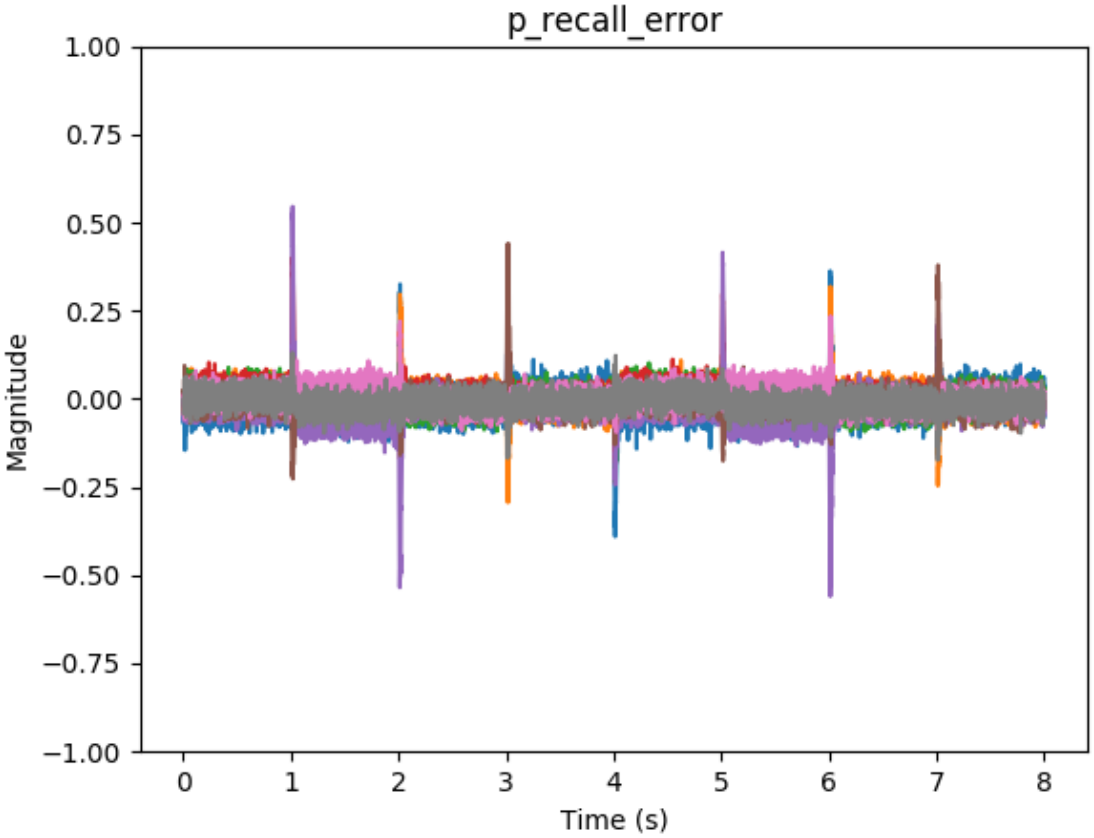

Figure 5.5: Recall Error Running Single Layer Perceptron Network

5.4 Converting to a Spiking Network

As the previous section showed the results matched if not exceeded the results from the original Nengo model in Chapter IV. The next step is to implement the network to run

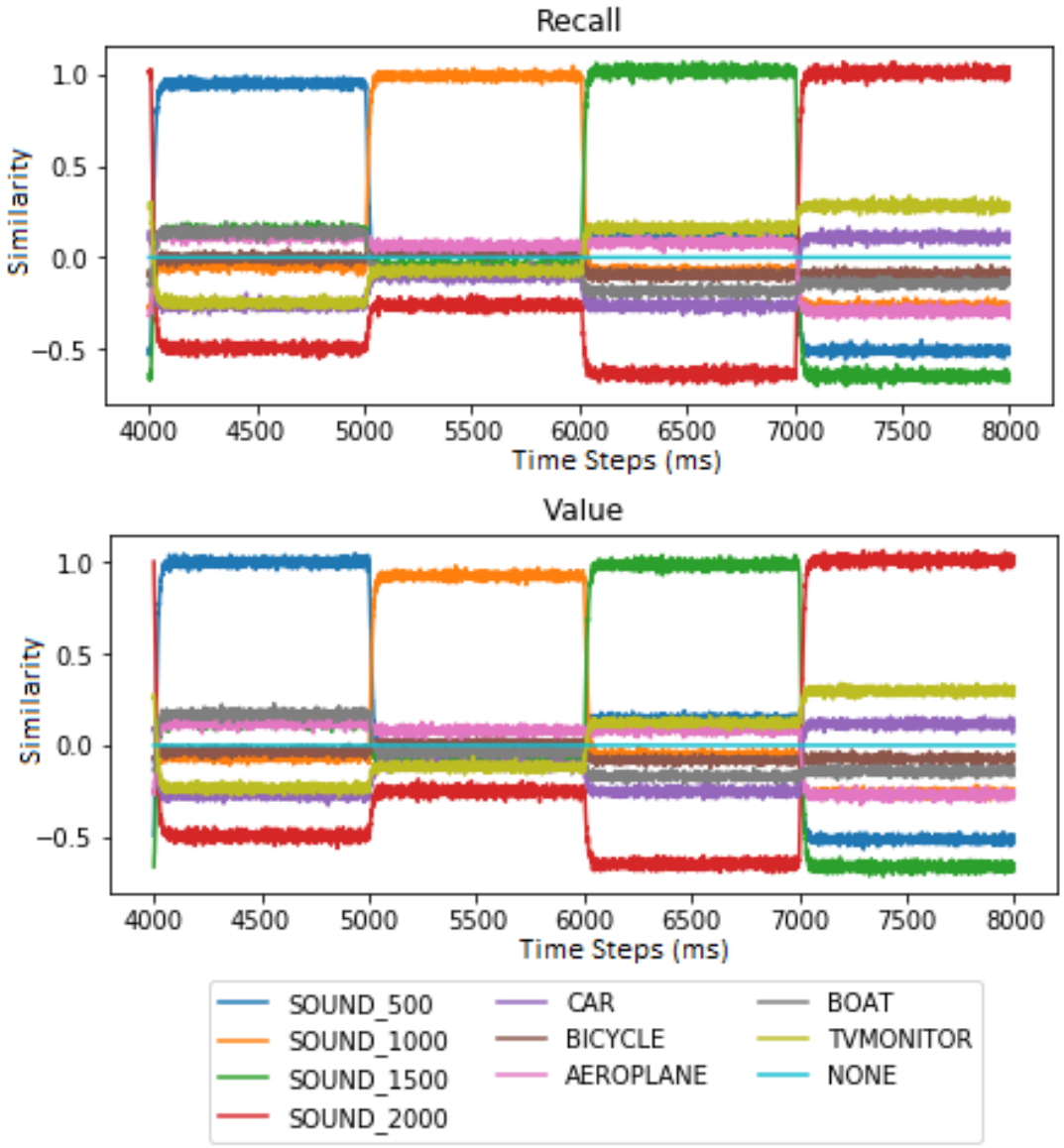

Figure 5.6: Recall Similarity Running Single Layer Perceptron Network

completely on the Loihi with the use of NxSDK, Intel's software package for programming Loihi. Since the model is now simplified down to a few nodes and connections, it can be directly converted into an SNN without the need of Nengo, as each node in 5.4 will represent one neuron. The only difference is that the current network uses backpropagation to adjust the weights of the connections and the that learning rule is not natively supported by Loihi; however, forms of STDP are. This changes the model slightly, but the premise is the same. Fig 5.7 shows the setup of the network implemented on Loihi being very similar to Fig 5.4 with just the addition of 4 excitation and inhibitory.

These will be used to send postsynaptic spikes to the group of output neurons to adjust the weight of the connections between the input and output neurons. The excitation and inhibitory neurons will send spikes based on the Value input it receives just as before. For example, the first Key-Value pair seen is Car and Sound_500, the model will assign the first output node to represent Sound_500 and train the network by sending excitation spikes to the first output neuron and inhibitory to the rest. This will help the network learn to fire only the first output neuron when it receives the input signal for CAR. Currently, the model is being implemented and tested in NxSDK needing more time to determine proper parameters for spike times and neuron compartments to find a stable configuration.

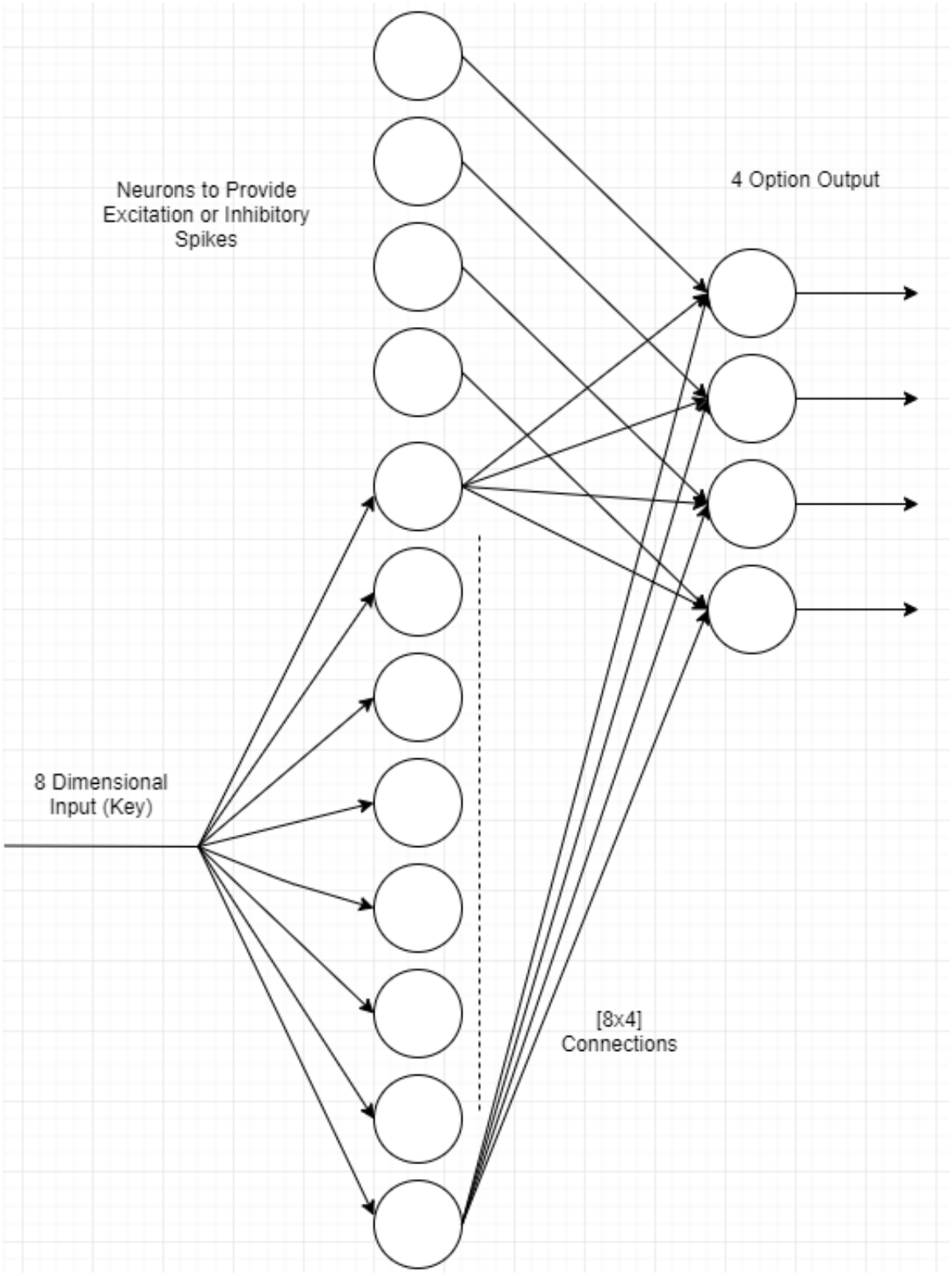

Figure 5.7: Single Layer Perceptron Neural Network Using STDP

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this thesis, we have shown a model for associative memory in the form of an SNN, implemented it on neuromorphic hardware, employed it for real world reasoning on a mobile robot, and showed the potential for energy savings when running on neuromorphic hardware. The model was run on an Up board where it had to train on a varying number of associations, then recall them, with the accuracy being recorded. This showed that the model performed just as well when training during running or beforehand. Finally, the model with the saved weights was uploaded to Intel's Loihi to see how well it ran on neuromorphic hardware. The tests proved the accuracy on Loihi is lower than on the Up board. The model still proved to work properly but will surely degrade faster with a higher number of associations. The power and energy to run the tests on both the Up board and the Loihi showed that the Loihi was able to outperform the Up board by 28% on execution time and 174% on dynamic power consumption.

The model was then expanded in its current form to test its accuracy up to 10 association pairs. While running the configurations on the UP Board, the model showed to fail around 10 association pairs by reaching its terminal %RMSE error. This is simply because the semantic pointers become too similar to one another given the low dimensionality they are. When running on Loihi the model proved to fail not much after 4 associations as by then it had already reached the terminal %RMSE error value. To try and prevent this the number of dimensions for the semantic pointers must be increased; however, Nengo currently restricts this as it would require multiple Loihi chips. The solution would be to

design a smaller network to replace the model, such that it could be programmed directly on Loihi, which would not only allow for larger dimensional input but also the use of Loihi's learning capabilities. A new network was designed as a single layer perceptron and when used in place of the previous model in Nengo, provided similar results. Steps are now being taken to implement the network on Loihi with the use of STDP to train the network.

6.2 Lessons Learned

Out of this project came many lessons learned not only with the associative memory model but with the neuromorphic hardware as well. To start, there proved to be no difference in results when the model was trained before testing or during testing with online learning. This shows that if a situation does not allow the model to be trained before running, it likely will not suffer from any loss in accuracy. However, one solution to improve the accuracy would be to increase the dimensionality of the semantic pointers. This would allow there to be a greater separation among the semantic pointers thus allowing the model to distinguish them more easily. The only issue is that it would require a greater number of neurons to hold the values in each dimension, and it was found that the model could not span across multiple Loihi chips to access more neurons. The new model designed aims to get around that problem by removing the dependency holding the model on one chip.

6.3 Future Work

The first continuation will be to finish implementing the spiking form of the new network. Once that is completed the model can be compared to the previous one for accuracy, energy efficiency, and execution time. With online learning on this model, it will provide much more insight into the capabilities of the network as well as neuromorphic hardware. From

there the new network will be built into a larger network to help drive cognitive decision making and complete the closed loop of an autonomous agent. The network would help the Turtlebot run on its own, learning new associations from different sensor inputs and making control movements on its own. With the addition of Loihi, the Turtlebot should be able to keep up with real-time interactions as well as remove the need for either wireless communication or short sessions due to the increased power saving from Loihi.

BIBLIOGRAPHY

[1] "Neuromorphic computing - next generation of ai." [Online]. Available: https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html

[2] "Intel announces neuromorphic research progress," Dec 2018, https://newsroom.intel.com/news/intel-announces-neuromorphic-computing-research-collaborators/gs.ao254z.

[3] "Neuromorphic computing," Jul 2020. [Online]. Available: https://newsroom.intel.com/press-kits/neuromorphic-computing/gs.c6nz5s

[4] T. Bihl, C. Cox, and T. Jenkins, "Finding common ground by unifying autonomy indices to understand needed capabilities." SPIE Defense and Commercial Sensing, 04 2018.

[5] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, vol. 40, p. e253, 2017.

[6] T. Bihl and M. Talbert, "Analytics for autonomous c4isr within e-government: a research agenda." Hawaii International Conference on System Sciences, 01 2020.

[7] T. Bihl, T. Jenkins, C. Cox, A. Demange, K. Hill, and E. Zelnio, "From lab to internship and back again: Learning autonomous systems through creating a research and development ecosystem," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, p. 9635–9643, 2019.

[8] "Up squared specifications," https://up-board.org/upsquared/specifications/.

[9] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. PP, pp. 1–1, 01 2018.

[10] M. Hampo, D. Fan, T. Jenkins, A. DeMange, S. Westberg, T. Bihl, and T. Taha, "Associative memory in spiking neural network form implemented on neuromorphic hardware," International Conference on Neuromorphic Systems 2020.

[11] C. Eliasmith and C. H. Anderson, *Neural engineering: computation, representation, and dynamics in neurobiological systems.* MIT Press, 2003.

[12] W. Maass, "Fast sigmoidal networks via spiking neurons," *Neural Computation*, vol. 9, no. 2, p. 279–304, 1997.

[13] N. Yusoff and F. K. Ahmad, "Stimulus-stimulus association via reinforcement learning in spiking neural network," *2013 13th International Conference on Intellient Systems Design and Applications*, pp. 131–136, 2013.

[14] D. Lisitsa and A. Zhilenkov, "Prospects for the development and application of spiking neural networks." IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 01 2017, pp. 926–929.

[15] C. Eliasmith, *How to build a brain: a neural architecture for biological cognition.* Oxford University Press, 2015.

[16] J.-H. Wang, *Associative memory cells: basic units of memory trace.* Springer, 2019.

[17] A. G. Hanlon, "Content-addressable and associative memory systems a survey," *IEEE Transactions on Electronic Computers*, vol. EC-15, no. 4, pp. 509–521, 1966.

[18] Y. Sandamirskaya, S. K. Zibner, S. Schneegans, and G. Schöner, "Using dynamic field theory to extend the embodiment stance toward higher cognition," *New Ideas in Psychology*, vol. 31, no. 3, p. 322–339, 2013.

[19] A. Horzyk and J. A. Starzyk, "Associative fine-tuning of biologically inspired active neuro-associative knowledge graphs," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 2068–2075.

[20] H. He, Y. Shang, X. Yang, Y. Di, J. Lin, Y. Zhu, W. Zheng, J. Zhao, M. Ji, L. Dong, and et al., "Constructing an associative memory system using spiking neural network," *Frontiers in Neuroscience*, vol. 13, 2019.

[21] T. Bekolay, C. Kolbeck, and C. Eliasmith, "Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks." 35th Annual Conference of the Cognitive Science Society, 01 2013, pp. 169–174.

[22] A. R. Voelker, "A solution to the dynamics of the prescribed error sensitivity learning rule," Centre for Theoretical Neuroscience, Tech. Rep., 10 2015. [Online]. Available: http://www.researchgate.net/publication/282366687

[23] A. R. Young, M. E. Dean, J. S. Plank, and G. S. Rose, "A review of spiking neuromorphic hardware communication systems," *IEEE Access*, vol. 7, pp. 135 606–135 620, 2019.

[24] S. Glatz, J. Martel, R. Kreiser, N. Qiao, and Y. Sandamirskaya, "Adaptive motor control and learning in a spiking neural network realised on a mixed-signal neuromorphic processor," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9631–9637, 2019.

[25] "Learning new associations," http://www.nengo.ai/nengo/examples/learning/learn-associations.html.

[26] A. R. Voelker, E. Crawford, and C. Eliasmith, "Learning large-scale heteroassociative memories in spiking neurons," in *Unconventional Computation and Natural Computation.* Springer International Publishing, 07 2014.

[27] "Intel® neural compute stick 2 and open source openvino™ toolkit." [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-neural-compute-stick-2-and-open-source-openvino-toolkit.html

[28] Kuśmierz, T. Isomura, and T. Toyoizumi, "Learning with three factors: modulating hebbian plasticity with errors," *Current Opinion in Neurobiology*, vol. 46, p. 170–177, 2017.

[29] P. K. Mungai and R. Huang, "A study on merging mechanisms of simple hopfield network models for building associative memory," *2017 IEEE 16th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pp. 199–206, 2017.

[30] R. A. Sinni and R. Daniel, "Biophysical analysis for implementing genetic associative memory using hopfield networks," *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4, 2019.