Michigan Technological University

Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2023

Neuromorphic Computing Applications in Robotics

Noah Zins
Michigan Technological University, nwzins@mtu.edu

Copyright 2023 Noah Zins

Recommended Citation

Zins, Noah, "Neuromorphic Computing Applications in Robotics", Open Access Master's Thesis, Michigan Technological University, 2023.
https://doi.org/10.37099/mtu.dc.etdr/1608

Follow this and additional works at: https://digitalcommons.mtu.edu/etdr

 Part of the Artificial Intelligence and Robotics Commons, Computational Engineering Commons, Electrical and Computer Engineering Commons, and the Robotics Commons

NEUROMORPHIC COMPUTING APPLICATIONS IN ROBOTICS

By

Noah Zins

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical and Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2023

© 2023 Noah W. Zins

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Electrical and Computer Engineering.

Department of Electrical and Computer Engineering

Thesis Advisor:	*Hongyu An*
Committee Member:	*Xiaoyong Yuan*
Committee Member:	*Lan Zhang*
Department Chair:	*Jin W. Choi*

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to thank my family and my partner Emily Grant for their unconditional support throughout my graduate studies, I could not have done it without them.

I would also like to thank Dr. Hongyu An for his guidance as an advisor, unending patience, and believing in my ability to become a successful researcher and graduate student. The lessons I learned under your mentorship will surely be invaluable in my future endeavors. Additionally, I would like to thank Dr. Lan Zhang and Dr. Xiaoyong Yuan for their guidance on the committee, and their excellent lectures that jumpstarted my research and expanded my interests in the field. I would also like to thank Dr. Yan Zhang for her instrumental efforts in helping me accomplish publications.

Next, I would like to thank Dr. Shane Oberloier for lending invaluable robotics expertise, helping grow my skills as a teacher, and being an amazing mentor throughout my time at grad school. I would like to thank Dr. Tony Pinar for providing me opportunities to pursue my interests, and the fruitful discussions that furthered my understanding of concepts critical to our work. I would also like to thank Dr. Christopher Middlebrook for giving me the opportunity to pursue graduate studies and persuading me to try research.

Finally, I would like to thank Dr. Yijing Watkins and Gavin Parpart at PNNRC for pioneering sparse coding research and inspiring our use of these ideas which became key to our success. I would also like to thank Tim Shae and the Intel Neuromorphic Research Community for pushing the boundaries of the technologically feasible, and the

astounding presentations that motivated me to further the progress of neuromorphic computing, as well as providing occasional tech support.

List of Abbreviations

NC	Neuromorphic Computing
SNN	Spiking Neural Network
LIF	Leaky Integrate and Fire
FPGA	Field Programmable Gate Array
DL	Deep Learning
DNN	Deep Neural Network
AI	Artificial Intelligence
LCA	Locally Competitive Algorithm
CS	Conditional Stimulus
US	Unconditional Stimulus
UGV	Unmanned Ground Vehicle
SWaP	Size Weight and Power

Abstract

Deep learning achieves remarkable success through training using massively labeled datasets. However, the high demands on the datasets impede the feasibility of deep learning in edge computing scenarios and suffer from the data scarcity issue. Rather than relying on labeled data, animals learn by interacting with their surroundings and memorizing the relationships between events and objects. This learning paradigm is referred to as associative learning. The successful implementation of associative learning imitates self-learning schemes analogous to animals which resolve the challenges of deep learning. Current state-of-the-art implementations of associative memory are limited to simulations with small-scale and offline paradigms. Thus, this work implements associative memory with an Unmanned Ground Vehicle (UGV) and neuromorphic hardware, specifically Intel's Loihi, for an online learning scenario. This system emulates the classic associative learning in rats using the UGV in place of the rats. In specific, it successfully reproduces the fear conditioning with no pretraining procedure or labeled datasets. The UGV is rendered capable of autonomously learning the cause-and-effect relationship of the light stimulus and vibration stimulus and exhibiting a movement response to demonstrate the memorization. Hebbian learning dynamics are used to update the synaptic weights during the associative learning process. The Intel Loihi chip is integrated with this online learning system for processing visual signals with a specialized neural assembly. While processing, the Loihi's average power usages for computing logic and memory are 30 mW and 29 mW, respectively.

1 Introduction

The natural world has inspired innovations in science and engineering across a broad range of disciplines. In recent decades the interest in studying biological mechanisms in order to imitate them, or biomimicry, has surged in popularity and produced new fields as well as several technological advancements [1]. One of these emerging fields is neuromorphic computing, which focuses on applying principles of neuroscience and the study of nervous systems to solve current problems in computing and artificial intelligence (AI). By creating novel brain-inspired architectures and specialized processing algorithms, neuromorphic computing introduces a new computing paradigm aiming to take advantage of the extraordinary capabilities and efficiency of biological nervous systems.

One such capability is the ability to remember relationships between previously unrelated objects or ideas through associative learning, a form of online learning that is pervasive throughout the animal kingdom [2-4]. Associative learning gives animals the ability to relate and memorize events in temporal proximity, contrasting the mainstream data-driven learning methods prevalent in modern AI and machine learning (ML). The nervous system realizes associative memory through its ability to modify the strength of synaptic connections between neurons in response to their firing activity, known as synaptic plasticity. The memorization mechanism is due to the strengthening of the synaptic connections occurring from the neurons firing in response to concurrent events. The more a synaptic connection is strengthened, the larger the quantity of the

neurotransmitter delivered to the neurons on the receiving end of the synapse becomes. If the strength is high enough, neurons that did not previously fire in response to a given stimulus will instead fire. In the case of these receiving neurons being “response neurons” triggering a certain response from the system, this strengthening can cause the response to occur to a stimulus that did not previously cause the response neurons to fire. This “signal pathway modification” is the mechanism used to memorize the relationship between concurrent events, instead of standard backpropagation.

1.1 Motivation and Contribution

Associative memory can be used in AI systems to provide an alternative realization of active self-learning through environmental interaction and exposure. The prevalent issue of reliance on large datasets is avoided because the process of signal pathway modification requires only a few repetitions of the training process. This is important because large datasets lead to long training phases and high computational demands, which consequently lead to very large energy demands. Furthermore, large labeled datasets are infamously difficult to produce and verify the quality of. Data-driven deep learning, like what has been demonstrated with Deep Neural Networks (DNNs), suffers from these staple data issues. While many significant achievements have been made with deep learning, these issues restrict its application to systems incorporating unwieldy supercomputers. Thus, many AI endeavors must look beyond deep learning for scenarios necessitating constraints such as size, weight, and power (SWaP), little to no data or pretraining, and autonomy independent from a remote computer [5, 6].

Previous work has yielded successful implementations of associative learning in neuromorphic systems; however, this work has not moved beyond simulations with relatively simple tasks and small neural networks [4, 7-14]. Many of these works also require labeled datasets for pretraining [10-14]. These issues leave the goal to mimic animals' exploration and learning capabilities far from realized, but we introduce some new real-world associative learning experiments to get one step closer. A mobile robot with a neuromorphic system is an ideal platform to replicate a classic associative learning experiment called fear conditioning. Traditionally fear conditioning experiments were performed with a rat which learns to associate a neutral stimulus, like a buzzer, with a undesirable stimulus, like an electric shock. The undesirable stimulus will immediately evoke a "fear response", such as running away, while the neutral stimulus will not evoke the response until after presenting the two stimuli together for several repetitions. Once the association is learned, the rats demonstrate enduring behavioral changes indicating their associative memory.

Several brain regions have been proven to be involved in the learning process, including frontotemporal amygdala, hippocampus, etc. The process of fear conditioning cannot be reproduced by other state-of-the-art associative memory models [4, 7-14] due to their limited neural network sizes. The simple neural network models cannot process informative signals, such as visual signals. These informative signals are processed with large-scale neural assemblies rather than simply a few neurons in the brain [15-20]. To resolve these limitations, the design uses a large-scale biological plausible neural assembly to process the visual signals. Specifically, in this system and experimental

designs, the mobile robot with sensors serves as the substitute for the rats in fear conditioning experiments. The neuromorphic chip (Intel Loihi) provides a computational platform for the associative memory learning operation. In the experiment, the brightness of a light emulates the visual stimulus, and the vibration signals from the accelerometer mimic the shock signals to the rats. Thus, the vibration signals are the undesirable stimulus and light is the neutral stimulus. The movement of the mobile robot emulates the fear response. The perception of the light and the vibration are separately processed within two different neural assemblies. Two neural assemblies connect to the response neuron, which stimulates the movement of the robot, with two signal pathways. One signal pathway with a weak synaptic connection serves as the conditional signal pathway, while another one with a stronger synaptic connection is the unconditional signal pathway.

The mobile robot provides this neuromorphic system with an ideal interface to interact with the environment, enabling for the first time, to our best knowledge, the implementation of associative memory as real-time online learning with no pretrained procedure. The contributions of this work are summarized as follows:

1) In contrast to other state-of-the-art works [10-14], associative learning was implemented in Intel's Loihi chip for online learning in a mobile robotics application.

2) This work reproduces the classic fear conditioning of rats with solid biological rationales from the cellular level (Hebbian learning) to the behavior level (neural assemblies).

3) The goal in training the neural network is signal pathway modification, the novel learning paradigm of associative learning.

4) This work does not require labeled datasets.

1.2 Background

By reverse engineering the underlying biological mechanisms enabling associative learning in animals, a novel self-learning paradigm is created that implements the associative learning capability on a neuromorphic system. This section will introduce state-of-the-art neuromorphic systems and neuromorphic computing hardware, followed by an analysis of the mechanisms from the cellular to the behavioral level that enable associative learning.

1.2.1 Neuromorphic Computing

Neuromorphic system emulates nervous systems, such as human brains, aiming at implementing Artificial Intelligence [21-26]. Human brains have the capability of executing sophisticated missions in unbelievably ultra-low energy. The average power of human brains is as small as ~20 watts [2]. In addition, unlike the training process required for Artificial Neural Networks (ANNs) using big data, the nervous systems can adjust their responses by constantly interacting with their surroundings. This learning process is referred to as associative learning [2]. These incredible capabilities of nervous systems are attributed to their parallelization, high degree of connectivity, adjustable

network topology, the colocation of data memory computation, and spike-based information representation.

Human brains consist of billions of neurons and trillions of synapses forming a high-degree and three-dimensional neural network. Through this extraordinarily complex network, an individual neuron can communicate with more than ten thousand other neurons simultaneously. Within this complex neural network, neurons are mainly signal processing units and the synapse between neurons is connecting organs. As computing units, the neurons integrate the received spiking signals in their cell body and send another sequence of spiking signals to other neurons through synapses. The signal strength received by other neurons depends on the connection strength of the synapses. The connection strength among neurons can be adjusted. This feature is named as synaptic plasticity [2, 27, 28]. In specific, the connection strength among neurons becomes strong if the presynaptic neuron and postsynaptic neuron are firing together. This synaptic connection strength change inspired a learning paradigm known as Hebbian learning [29-32].

In addition, the computational units (neurons) and the memory units (synapses) are located in close proximity. This structure eliminates one of the biggest inefficiencies in von Neumann architecture that separates computing units and memory at different locations. The physical separation leads to data needing to be constantly transferred back and forth between memory and central processing units (CPUs). Furthermore, neuromorphic systems use sparse and event-based computation, meaning that only a small percentage of the available computing resources are active for a given task, and

they're only activated and consuming power as needed in response to present events. Neuromorphic computing attempts to exploit these useful properties by modeling the architecture, neuron and synaptic cells, and the way of learning observed in the brain, enabling a new era of computers and AI [33].

Neuromorphic systems utilize specialized neuromorphic chips with artificial neurons. These chips are generally used to operate spiking neural networks (SNNs), which encode the information with a sequence of spikes just like nervous systems. In an SNN, neurons communicate with each other with discrete "spike" signals. There are various types of neuromorphic chips, such as Intel's Loihi [34, 35]. Unlike traditional GPUs and CPUs built upon von Neumann architecture operating on digital information, Loihi chips are specifically designed for neuromorphic computing and asynchronous SNNs. To date, two generations of Loihi chips have been released. The first generation of Loihi chip was revealed in 2017 [34, 35]. Loihi-1 chips consist of 130,000 electronic neurons and 130 million synapses at 128 neuromorphic cores. The advanced 14 nm process of Intel renders the area of the Loihi-1 chip as small as 60 mm^2. Loihi-1 chips implement the digital leaky-and-fire neurons located on 128 cores. At each core, the communication among neurons is organized in a mesh configuration. The synapses in Loihi-1 chips are fully configurable and further support weight-sharing and compression features. The plasticity of synapses can be manipulated with various biologically plausible learning rules, such as Hebbian rules, STDP, and reward-modulated rules [34, 35]. The firing behavior of neurons in Loihi chips is implemented when received spikes accumulate to a

threshold value in a certain time, the neurons will fire off their own spikes to their connected neurons.

Loihi-1 chips are offered with several neuromorphic platforms providing distinct interfaces for integrating the Loihi-1 chip with other computer systems or Field-Programmable Gate Array (FPGA) devices. Kapoho Bay, shown in Figure 1.1, includes 1-2 Loihi chips with a USB interface.

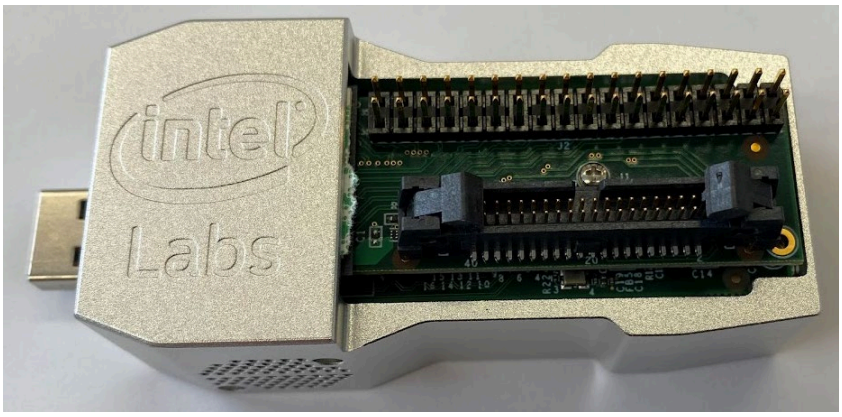

Figure 1.1: Kapoho Bay with two onboard Loihi chips used in this work.

Nahuku is a 32-chip Loihi board with a standard FPGA Mezzanine Card (FMC) connector. The FMC connector allows the Nahuku system to communicate with the Arria FPGA development board. Pohoiki Spring is a large-scale Loihi chip with 100 million neurons equipped as a server for remote access. The second generation of the Loihi chips, namely Loihi-2, was introduced in late 2021 [36]. Loihi-2 is fabricated in Intel 4 process, previously referred to as 7 nm technology. Powered by this advanced technology, the area of the Loihi-2 reduces to 31 mm^2 from 60 mm^2 of the first generation Loihi chips. Unlike the rigid neuron models in the last generation of Loihi chips, Loihi-2 realizes fully

programmable neuron models. In Loihi-2, the specific behavior of the neurons can be programmed with microcode instructions. The microcode instructions support basic bitwise and math operations that can be used to specify custom neuron models. Loihi-2 chip is dedicatedly designed for neuromorphic computing and edge devices with parallel computations achieving high computational and energy efficiency. The comparison between two generations of Loihi chips is summarized in Table 1.1

Table 1.1: Introduction to Loihi and Loihi 2 Chips.

Feature	**Loihi 1**	**Loihi 2**
Technology	Intel 14 nm	Intel 4 (7 nm)
Die Area	60 mm^2	31 mm^2
Max # Neurons/Chip	128,000	1 million
Max # Synapses/Chip	128 million	120 million
Neuron Model	Generalized digital LIF	Fully programmable

Loihi 1 uses a Leaky Integrate and Fire (LIF) model for implementing neurons. The LIF model is popular because it provides the information processing functionality of neural dynamics while remaining simple enough to evaluate with limited computation. LIF neurons are characterized using the following equations [37]:

$$C_m \frac{dV_m}{dt} = G_L(E_L - V_m) + A * I_{app}, \quad if\ V_m > V_{th}\ then\ V_m = V_{reset} \tag{1}$$

$$\tau_{RC} = C_m/G_L \quad (2)$$

where C_m is the membrane capacitance, G_L is the leak conductance, E_L is the leak potential, V_m is the membrane potential, A is the input signal gain, I_{app} is the input current, and τ_{RC} is the membrane RC time constant. A variation of the LIF neuron model is the simpler Integrate and Fire model. As the name suggests, Integrate and Fire neurons are the same as LIF neurons but without the decaying membrane potential; in other words, τ_{RC} in (2) is set to infinity.

1.2.2 Associative Learning

Animals have the capability of memorizing different events if they occur at the same time or with a small-time lag. The capability is referred to as associative memory [2]. Associative memory learning is first studied by Ivan Pavlov in the 1890s when he was studying salivation reflex actions in dogs [2]. During Pavlov's experiments, the dogs originally had a salivation reflex to the presence of food, instead of the sound of whistles. However, if these two signals were presented together several times, the dogs salivated even if they only listened to the sound of a whistle with no food provided. This means the dogs can memorize the sound of whistles as a sign of food [2, 7, 38] through a learning/memorizing process. Through a series of experiments, Pavlov concluded that dogs have the capability of associating two originally irrelevant signals together through a training process, which is referred to as associative memory learning later. In general, two types of stimuli exist in associative memory learning: unconditional stimuli (US) and conditional stimuli (CS). The unconditional stimuli evoke the response with no training

required. On the contrary, conditional stimuli demand an associative learning process to acquire corresponding reactions. For instance, in Pavlov's experiments, the presence of food is the unconditional stimulus, and the sound of whistles is a conditional stimulus (CS). After dogs, further studies demonstrate that associative memory learning is a self-learning paradigm of a large variety of animals such as rats, bats, sea slugs [2].

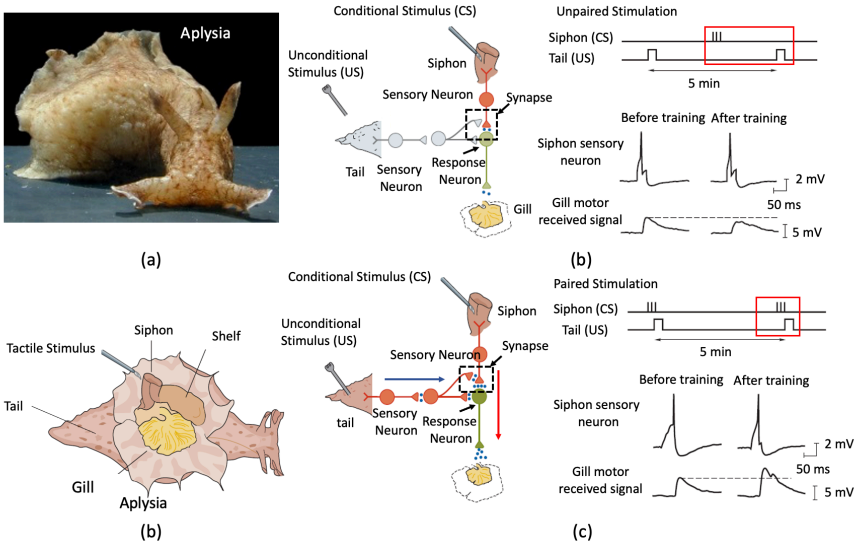

Figure 1.2: Illustration of associative memory learning of Aplysia.

The studies in neuroscience exhibit that signal pathway modification and synaptic plasticity are highly related to associative memory learning [2, 23]. In a nervous system, the shapes of the spiking signals are almost identical (spikes) whatever the signals come from the sensation of light or hearing. Thus, neuroscientists hypothesize that the brains distinguish these signals by the signal pathways they are traveling to rather than their shapes. This hypothesis is much more straightforward in invertebrates that have simple

nervous systems. Figure 1.2 illustrates part of the nervous system of Aplysia that has two signal pathways from siphon to gill and from tail to the gill, separately.

With these two signal pathways, Aplysia can accomplish a simple version of associative memory learning by memorizing the touch on the tail and stimulus from the siphon. When the tail of an Aplysia is touched, its gill shrinks, demonstrating an unconditional signal pathway. On the contrary, the gill does not shrink if the siphon is cut, exhibiting a conditional signal pathway. By applying a touch to the tail and stimulus on the siphon at the same time several times, the gill motor neuron becomes more responsive to the touch on the siphon alone. At the cellular level, the concurrent stimulus on the siphon and tail leads to a spiking signal overlapping when the stimulus is applied at the same time, shown in Figure 1.2. As a result, the synaptic connection among neurons, from the siphon to the gill, becomes stronger than the original state. This means the signal pathway from the siphon to the gill becomes unimpeded from blocked. These experiments on Aplysia demonstrate two critical factors for associative memory learning: (1) signal pathway modification; and (2) synaptic plasticity.

For more complicated animals, such as rats, the sensation signals are processed not in individual neurons but in a group of neurons. These groups of neurons are referred to as neural assemblies [30, 39-41]. For example, fear conditioning experiments in rats involve two types of stimuli: electric shock on the food and a sound as neutral stimuli. These two types of signals are processed at different neural regions: auditory thalamus and somatosensory thalamus. The experimental goal is to let the rats associate the neutral sound with undesired electric shock by applying these two stimuli at the same time. Thus,

it is one type of associative memory learning scheme. The studies have strong experimental evidence showing that signal pathway modification potentially occurs in lateral nucleus because the output signals from the auditory thalamus and somatosensory thalamus converge at the lateral nucleus [2]. This hypothesizes that associative memory learning in higher animals is accomplished via the association of two, or several, neural assemblies together, rather than individual neurons.

1.2.3 Sparse Coding and the Locally Competitive Algorithm

Traditional convolutional neural networks have shown unmatched performance in image classification and related tasks, but are easily attackable, suffer from training and large data issues, and usually require power hungry GPUs for computation [42]. In the pursuit to understand and mimic biological brains, researchers have discovered spatial and temporal sparsity help enable efficient signal processing in neurobiological systems [43]. Trying to model this effect has enabled networks to perform feature extraction similar to the V1 region of the primary visual cortex [44]. Sparse coding has been used to compress various types of sensory information, model the transmission of visual information from the eye to the brain, and even predict future frames in a video stream [45, 46].

The nature of "sparsifying" a signal into its essential components makes sparse coding inherently resistant to noise and adversarial attack [47]. The goal of the sparse coding problem is to represent an input (often an image) with a minimal subset of features, or "atoms", chosen from an overcomplete "dictionary" basis set. The sparsity arises from trying to use the least number of atoms to represent the input, creating a tradeoff between

sparsity and accuracy. This problem can formally be defined by the following LASSO equivalent regression problem:

$$E(a) = \frac{1}{2}\|x - \Phi \cdot a\|_2^2 + \lambda \cdot \|a\|_1 \quad (3)$$

$$a^* = \underset{a}{argmin}\, E(a) \quad (4)$$

Where $E(a)$ is the cost, a is the "sparse code" vector consisting of the feature coefficients a_i, x is the input signal, and Φ is the dictionary matrix, the columns of which are the features Φ_i. The cost is determined by how close input matches the reconstruction $\Phi \cdot a$, and the size of the sparse code vector $\|a\|_1$, where λ is an additional sparsity penalty parameter to control the tradeoff between sparsity and accuracy. Eq. (4) shows a^* is the optimal sparse code for minimizing the cost in Eq. (3).

Models for solving sparse coding like the Locally Competitive Algorithm (LCA) offer biologically plausible methods for efficient and robust information processing that can be realized in neuromorphic hardware [48]. LCA is inspired by lateral inhibition observed in neuroscience and uses it to emulate the sparse coding that occurs in the V1 region. The algorithm uses only local competition between neighboring neuron elements, and it can be implemented with SNNs unlike other solutions to the sparse coding problem. The basic idea is that each dictionary feature is represented by a neuron, with its firing rate or activation indicating its feature's contribution to the input. The activation is achieved by the weights of the connections from the input to these "feature neurons". The higher a feature neuron's activity level is, the more it inhibits, or reduces, its neighboring neurons'

activity levels. In [49] its shown that a discrete spiking implementation solves Eq. (3,4) given a non-negative constraint to the input arising from the binary nature of spikes only being able to represent positive values; without this constraint the convergence of LCA is not guaranteed [50, 51]. The process can be implemented with the following Spiking LCA (S-LCA) dynamical systems algorithm:

$$\dot{u} = \frac{1}{\tau}(\Phi^T x - u - (\Phi^T \Phi - I)),\ a = T_\lambda(u) \tag{5}$$

$$T_\lambda(u) = 0 \ \text{ if } \ u \leq \lambda\ , \ \text{else } \ T_\lambda(u) = u - \lambda \tag{6}$$

Where τ is the discrete timestep, a_i is the average firing rate of neuron i, u_i is the average soma current for the neuron, and T_λ is the thresholding function that determines if neuron i is going to fire. This is achieved by adjusting the bias, V_{th}, of the neurons to -λ. It has been shown that the S-LCA system dynamics converge to the set of average firing rates a_i corresponding to the optimum solution a^*.

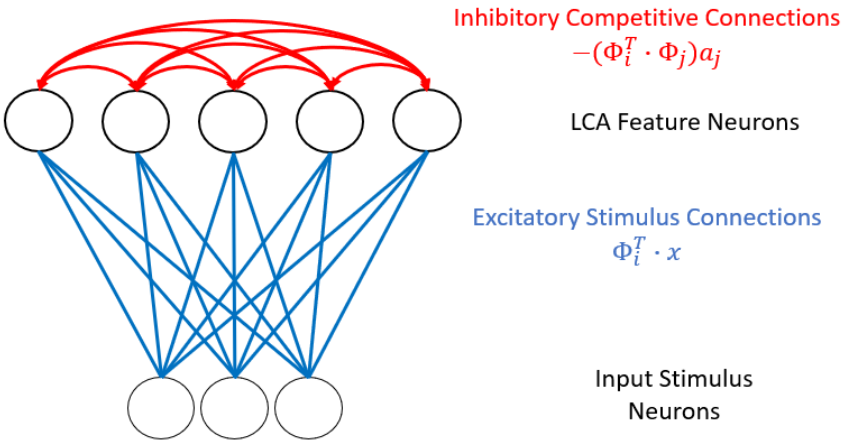

Figure 1.3: The LCA network model.

There are really two parts to the sparse coding problem when trying to find the optimal sparse representation of a given input: finding the sparse code vector a^*, and finding the optimal dictionary atoms Φ_i that yield the best a^* solution. Several unsupervised methods can be used to learn the dictionary, including both online and offline learning methods [52]. Models using learned dictionaries offer more sparsity and accuracy as there are more variables giving more points to optimize. One way to think of this effect is lowering yet complicating the energy landscape. The basic idea is to alternate between updating the coefficients a_i and the dictionary atoms Φ_i for a given input or batch of inputs. Several variations of this method for both spiking and non-spiking LCA implementations have been demonstrated and analyzed, some of which address the binary spiking constraints and some of which take advantage of emerging neuromorphic hardware capabilities like signed and nonbinary spikes [50, 52, 53]. It has even been shown that simulating periods of sleep during training can improve learning [54]. Sparse coding models show a learned dictionary can generalize well to different types of data, demonstrating transfer learning [55]. The dictionary learning process alone has implemented associative learning by utilizing multimodal sensory data as the input [56].

1.2.4 Mobile Robotics

Mobile robotics provide an ideal application for neuromorphic computing due to the conditions and constraints in their common use cases. One important benefit of neuromorphic computing is the impressive minimal size, weight, and power (SWaP) requirements for performing complex cognition tasks necessary in many robotics scenarios. Mobile robots need to maintain high energy efficiency as they have limited

power supplies. Many applications limit or do not provide communication with a remote host, requiring the robot to perform independent autonomous execution of tasks using c only the onboard computational resources. These applications are more generally referred to as edge computing, which is the computing paradigm that seeks to process data near the source instead of communicating with more powerful remote computational resources. Another ideal aspect of neuromorphic computing in robotics applications arises in scenarios that offer limited preexisting data. The scenarios may inhibit creation of data due to cost or impracticality of collection, such as Lunar and Martian terrain data for space exploration applications [6]. Several mobile robotics applications are a poor fit for traditional DL approaches as they have large power requirements and rely on datasets for training.

Clearpath's Jackal UGV fitted with a ZED 2 stereo camera and VLP-16 LIDAR sensor, shown in Figure 1.4 is selected as the mobile robotics platform.

Figure 1.4: Clearpath Jackal UGV used for mobile robotics experiment.

The Jackal UGV also contains an important sensor, the Inertial Measurement Unit (IMU). IMU's generally consist of an accelerometer and gyroscope that measure forces used to calculate the device's acceleration and orientation. Clearpath provides several software packages used to interface with the UGV via the Robot Operating System (ROS) framework. ROS is designed to operate with several independent processes, called nodes, that perform individual specific tasks. The nodes can use a variety of communication paradigms to communicate with each other, including a publish-subscribe framework and client-service framework. Publish-subscribe communication consists of dedicated

channels, called topics, that nodes can access as a publisher or subscriber. The publishing nodes transmit data messages corresponding to the appropriate topic, and all of the nodes subscribed to that topic receive the data whenever it is transmitted. In contrast, the client-service framework consists of a service node serving a specific function that is only performed when a corresponding client node sends a request to do so. The popular Gazebo simulation environment integrates with ROS and is selected for simulating experiments because Clearpath provides supporting packages for working with Gazebo and the Jackal.

2 Associative Memory Experiment

This section explores the methodology and design process for using mobile robotics to reproduce fear conditioning in rats. The electric shock used as the unconditional stimulus in the classical fear conditioning experiment is replaced with vibration from a constructed vibration platform. Similarly, the buzzer tone used as the conditional stimulus is replaced by light from a mounted light source. The neuromorphic system utilizes Leaky Integrate and Fire (LIF) style neurons because they are capable of modeling the signal pathways and higher-level information processing in neural assemblies, while simple enough to remain computationally inexpensive. The neural network is implemented in Nengo, a neuromorphic simulator developed by Applied Brain Research [57]. Intel's Loihi chip is used as a backend for Nengo in several experiments. The ROS framework is used to operate the mobile robot, a Clearpath Robotics Jackal UGV, and for transferring data in and out of the Nengo program [58].

2.1 Simulation and Preliminary Testing

The main goal of the experiment is to achieve signal pathway modification through Hebbian learning, so initially a simple network is created to demonstrate simple associative learning with Nengo. Two programmatically controlled network inputs, called nodes, are created to represent the conditional and unconditional stimuli. Shown in Figure 2.1, each input node is connected to a single LIF neuron that activates, or fires, to represent recognition of the US or CS. These neurons are referred to as the US neuron

and CS neuron, respectively. A third LIF neuron, the response neuron, is created to signal the response of the network to the stimuli.

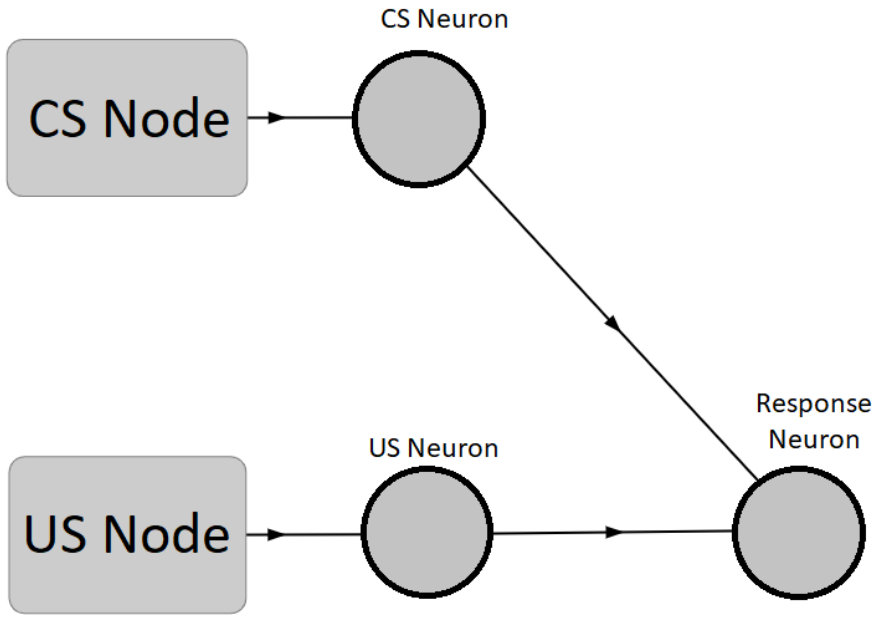

Figure 2.1: Neural network for demonstrating Hebbian learning.

The values used for the LIF parameters from Eq. (1,2) are given for the three LIF neurons in Table 2.1.

Table 2.1: LIF neuron parameters for simple associative learning.

Neuron Types	τ_{RC}	A	V_{reset} (V)	V_{th} (V)
US neuron	0.02	1.0	0.01	1.0
CS neuron	0.02	1.0	0.01	1.0

Neuron Types	τ_{RC}	A	V_{reset} (V)	V_{th} (V)
Response neuron	0.02	1.0	0.01	1.0

For all LIF neurons in Nengo, the firing threshold is fixed at 1 V and input gain is modified instead. The initial value for τ_{RC} in Nengo's standard implementation of spiking LIF neurons is 0.02, which was not modified as the neurons exhibited desired behavior. V_{reset} was set at 0.01 V so the neurons fire when they receive the default programmable input stimulus without modifying the default gain of 1.0. The US neuron's output spikes are relayed to the to the response neuron via an unmodifiable synaptic connection modeled by the lowpass filter with the impulse response:

$$h(t) = \frac{1}{\tau} e^{\frac{-t}{\tau}} \tag{7}$$

Where τ is the time constant in seconds. This filter is chosen for all synapses because it is the default model in Nengo and works well for the intended purpose. The default τ of 0.005 seconds, which corresponds to a cutoff frequency about 32 Hz, is initially used for all synapses. Synaptic connections have "weights" that are scalar values multiplied with the output signal to act as a gain for the synapse output current. Inhibitory connections that reduce the postsynaptic neuron's potential can be implemented using a negative value for the synapse weight. The CS neuron is connected to the response neuron by a modifiable synapse that uses the Hebbian learning rule in (8) to modulate its weight.

$$\Delta w = \eta r_i r_j \tag{8}$$

Where w is the synaptic weight, η is the learning rate constant, r_i is the presynaptic neuron's filtered activity, and r_j is the postsynaptic neuron's filtered activity. This equation states that when the postsynaptic and presynaptic neuron fire simultaneously, the synapse strength increases proportional to the firing rates of the neurons. The synaptic weight of the learning synapse is initially set to 0.0001 so the US input signal triggers the response neuron, but the CS input signal does not. The learning rate, η, was empirically chosen as 2×10^{-5} so that the learning could be completed in relatively few training cycles without being unrealistically short (i.e., one cycle or less). Each training cycle consists of presenting the US and CS each for two seconds, with one second of overlap, followed by 1 second of no stimulus, illustrated by Figure 2.2.

First the initial response to both stimuli is demonstrated, followed by training cycles to increase the synaptic weight. After enough repetitions, the synapse strength has increased enough that presentation of CS causes the response neuron to fire due to the CS neuron's output alone, demonstrating successful associative learning.

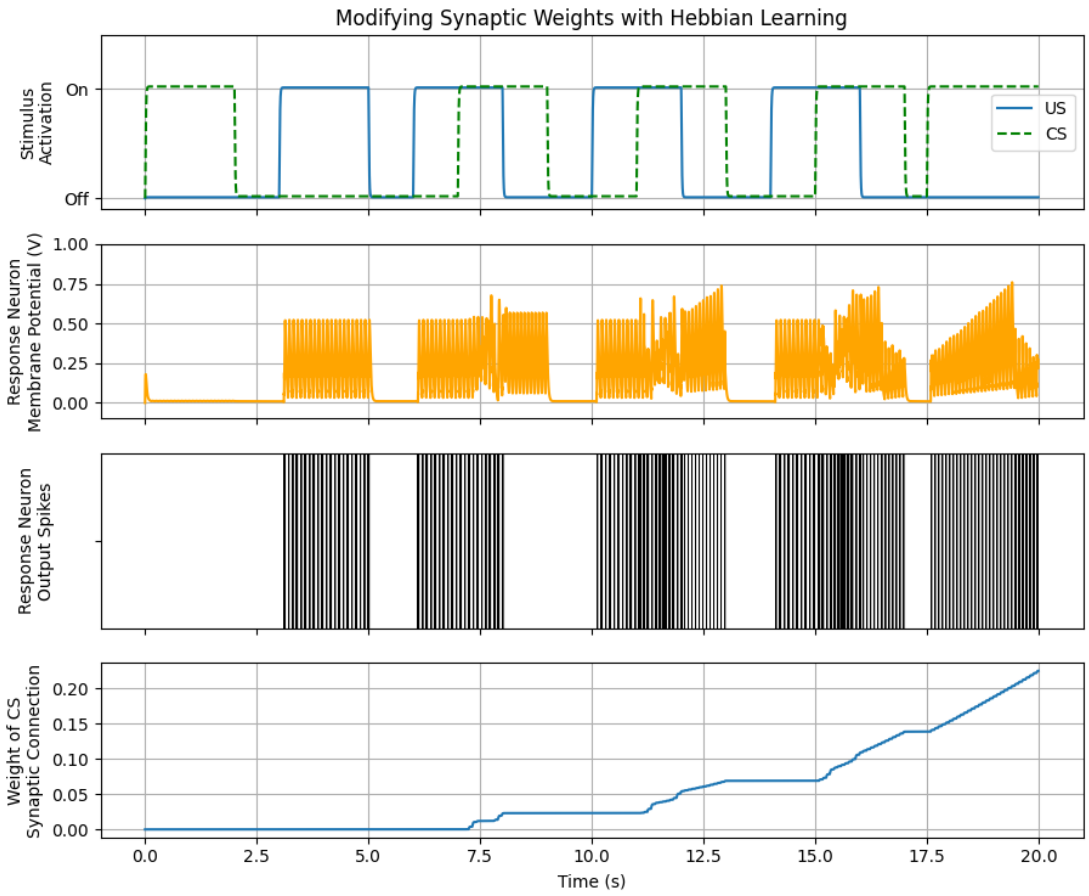

Figure 2.2: Hebbian learning causing synaptic weight modification.

With a validated proof of concept for implementing associative learning in Nengo, a simulation of the mobile robotics experiment is conducted. Clearpath has developed support for the Jackal UGV in the Gazebo simulation environment, so it is chosen for simulating the experiment [59]. Unfortunately, implementing a vibration platform to provide the US proved quite problematic as Gazebo tends to become unstable when objects start rapidly colliding with each other. Consequently, it was deemed sufficient to record real vibration data for use in the simulation. The vibration table generates 15 Hz vibration with an amplitude of 1.2 mm, the lowest available settings offered by the

vibration table which were chosen to minimize impact on the UGV. These vibrations are measured by the Jackal UGV's onboard IMU. Clearpath's ROS software running on the UGV receives the stream of data from the IMU and publishes the robot's acceleration to an IMU topic where other ROS nodes can access it. This vibration data can be recorded and played back at any time, and it will appear as though the data publishing is happening in that moment. Creating an instance of Nengo running inside of a ROS node subscribed to the IMU topic provides a ROS a communication channel to Nengo. The raw acceleration data during vibration, shown in Figure 2.3, is more easily integrated with the network after undergoing some minimal preprocessing.

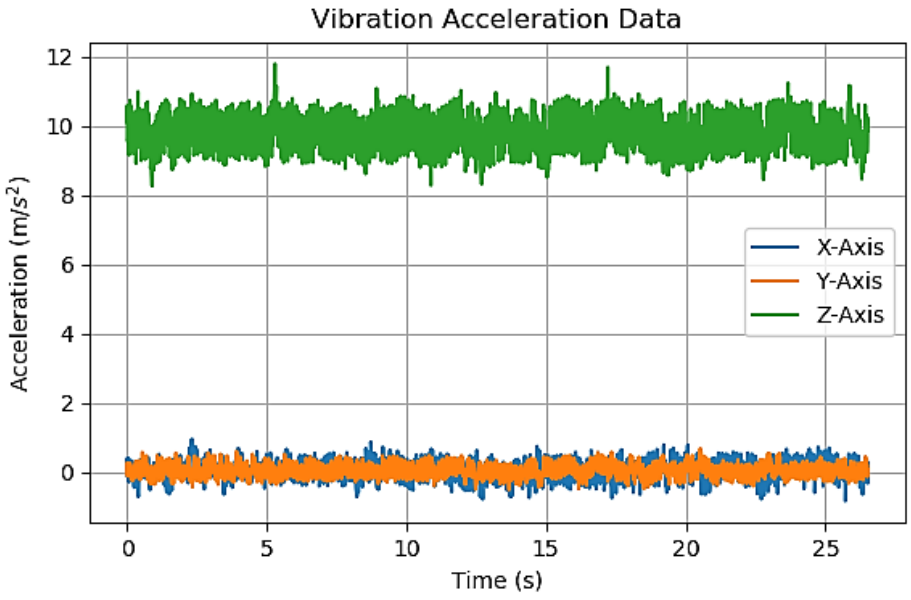

Figure 2.3: Acceleration data from vibration table at 15 Hz.

As shown above, the z-axis has an average magnitude due to the Earth's gravity of 9.81 m/s^2, which is removed to bring the resting acceleration for all three axes to zero.

Initially, the resultant acceleration shown in (9) was used to evaluate the vibration state of the UGV.

$$a_{res} = \sqrt{{a_x}^2 + {a_y}^2 + (a_z - 9.8)^2} \tag{9}$$

However, observing Figure 2.3, one can see the z-axis acceleration deviates from its resting value much more than the other two axes. This signifies the z-axis acceleration measurement alone should be sufficient for indicating vibration, so the preprocessing equation is simplified to the following equation:

$$a_r = |a_z - 9.8| \tag{10}$$

This new equation is ideal as preprocessing should be minimized to take full advantage of the neuromorphic system's potential for efficiency. A comparison of a_{res} and a_r during vibration is shown in Figure 2.4. The rectified z-axis acceleration shows trends and magnitudes similar to the resultant acceleration, so it is selected for vibration preprocessing. The US input node is converted to receive the IMU data, evaluate (10), and use a_r as its output. The node's output can be thought of as a spike generator with a firing rate proportional to its value, though the actual implementation details stray from this concept.

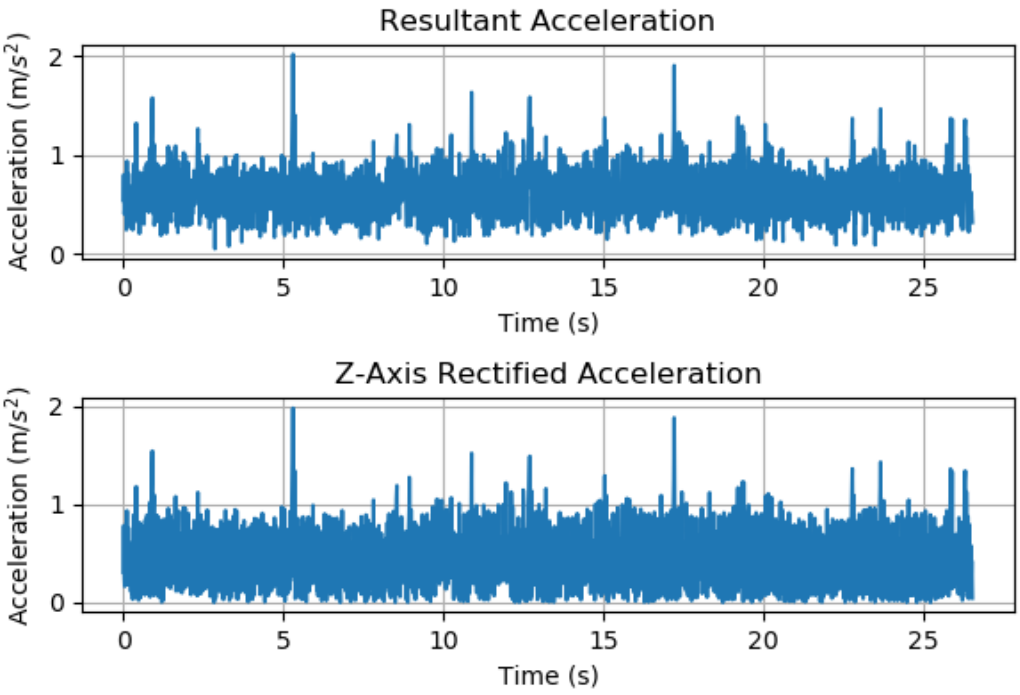

Figure 2.4: Comparison of resultant acceleration, a_{res}, and rectified z-axis acceleration, a_r.

To keep preprocessing minimal, the raw values from the node are not scaled or normalized to a standard range (e.g., -1 to 1). Instead, the input synapse and LIF parameters of the US neuron, now the vibration detection neuron is adjusted to achieve the desired response to the vibration signals. The values summarized in Table 2.2 are selected for the simulation experiment.

Table 2.2: LIF neuron parameters for associative learning in mobile robotics simulation.

Neuron Types	τ_{RC}	A	V_{reset} (V)	V_{th} (V)
(US) Vibration neuron	0.02	1.3	0.6	1.0
(CS) Brightness neuron	0.02	0.9	0.15	1.0

Neuron Types	τ_{RC}	A	V_{reset} (V)	V_{th} (V)
(Response) Movement neuron	0.02	1.0	0.01	1.0

The default τ_{RC} of 0.02 seconds is kept because it is sufficient for the desired functionality. The other two parameters are calculated and optimized based on the experimental setups so that they produce the desired responses for their respective uses.

Figure 2.5: Vibration neuron response to acceleration.

For the vibration detection neuron, gain (A) and bias (V_{reset}) were empirically derived so the vibration neuron only exhibits the desired behavior of continuously sending output spikes if the vibration platform is enabled, shown in Figure 2.5. The filter time constant

of the synapse between the input and vibration neuron is increased to 0.2 seconds to prevent the vibration neuron from firing when the UGV makes small sudden movements like stopping. Next, the simulated stereo camera is used to measure the brightness of a light, the CS replacing the buzzer tone for rats, in Gazebo. Unfortunately, Gazebo does not support the ZED 2 stereo camera installed on the Jackal UGV; however, the Bumblebee2 is supported and has a similar function, so it is used as the stereo camera in simulation. Only one camera is needed for the experiment, so just the right one is used. A Gazebo environment similar to the real-world experiment setup is constructed with a light panel and placeholder vibration platform, shown in Figure 2.6.

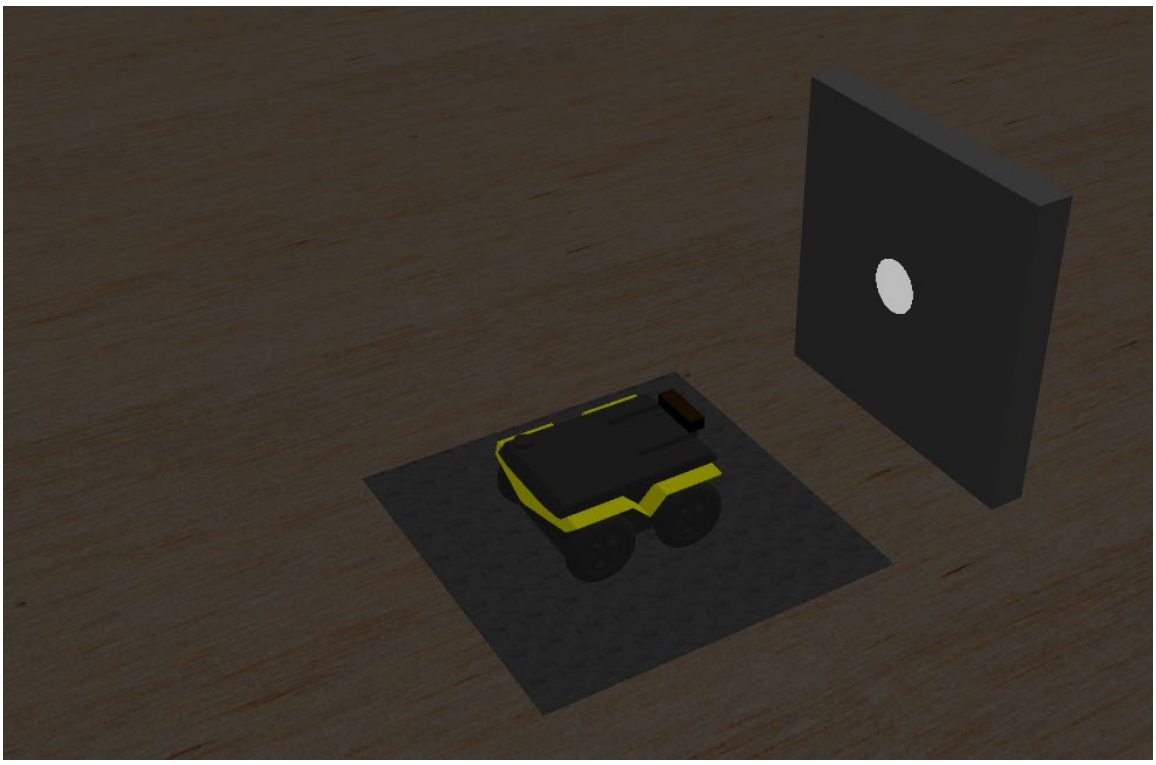

Figure 2.6: Gazebo environment setup for associative learning experiment.

The light panel has a circular light that is centered in the right camera's frame when the UGV is on the vibration platform, shown in Figure 2.7. The image data is not as easily interfaceable with Nengo as the acceleration data, because it requires a custom "Nengo process" to be created among other additional support processes. Nengo processes can be used for several things, like making a node output a simple function, or describing a dynamical system to a group of neurons. A Nengo process, the "camera process," is created to receive an image upon request from a dedicated ROS service, downsample the image to a given resolution, and rescale the pixel values from a discrete range of 0 to 255 to a continuous range of -1 to 1 for use as stimulus outputs.

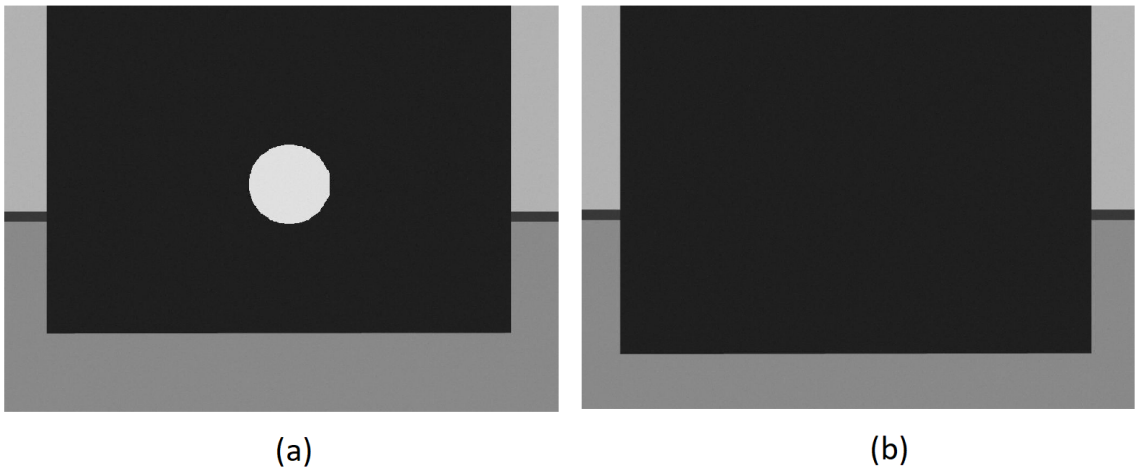

Figure 2.7: Images from simulated camera light panel in Gazebo (on and off).

The ROS service subscribes to the image data topic and stores the most recent frame for delivery to any requesting client, acting as a one sample buffer for the image stream. This keeps the camera process from being bombarded with image data it is not ready for. Preliminary testing showed the framerate of the image data coming from the image topic

was unreasonably slow, so the camera process was modified to use a compressed image stream. The compressed image stream requires significant additional software to interface with the stream and decode the images compared to the uncompressed stream, so the corresponding modifications were made to the Nengo process and the ROS service. With the camera process implemented and receiving a smooth video stream of images, the CS node can be converted to use the camera process and output the rescaled pixel intensity data to the network. Initially only the CS neuron, now the brightness detection neuron, is used to detect the light. Shown in Figure 2.8, the image is rescaled to 5x3 pixels so one center pixel contains the approximate average value of the entire light, and the camera process' extraneous computations are minimized.

Figure 2.8: Resized images for brightness detection in simulation (on and off).

The middle pixel's intensity is used as the output of the CS node that connects to the brightness neuron, the rest of the CS node outputs are left disconnected. The LIF parameters for the brightness neuron in Table 2.2 are derived empirically to achieve the desired functionality, analogous to the vibration neuron. With these parameters, the

brightness neuron fires continuously when the light panel is activated, and ceases to fire as soon as the light is turned off, as shown in Figure 2.9.

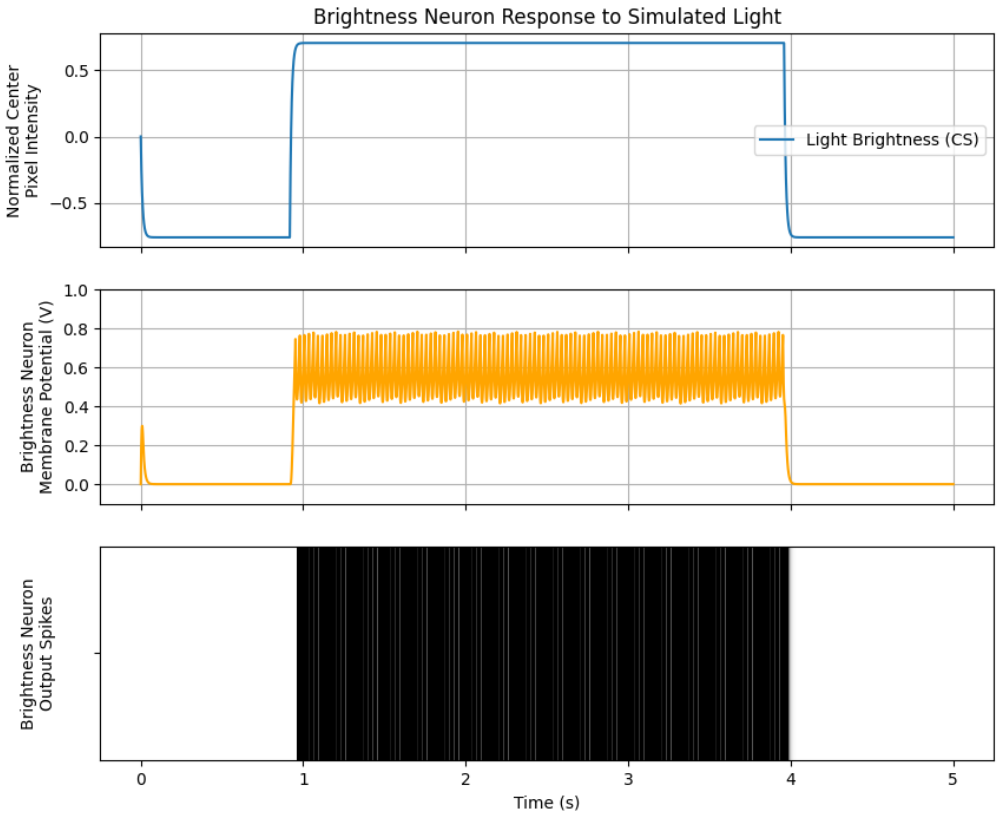

Figure 2.9: Brightness neuron activating with light stimulus in simulation.

Finally, the response neuron is converted to a movement neuron by creating a "movement output node" that sends movement commands to the UGV, simulating the fear response of the rat. First, a slight bias is added to V_{reset} for the movement neuron in Table 2.2 because it is found to slightly improve stability in the Nengo's execution (e.g., avoiding division by zero). Like the vibration node, the movement node communicates with the

UGV through a movement ROS topic. Conversely, the Nengo node now publishes the data, and the Jackal software subscribes to the movement topic. Clearpath's software on the Jackal UGV receives the movement commands and controls the motor drivers while utilizing wheel encoders and other sensor data to calculate odometry. Upon receiving a spike from the movement neuron, the movement node starts sending command the UGV to move backwards with a velocity of 0.3 m/s. The node keeps sending commands until about one second after the last spike is received, moving the UGV to the neutral position shown in Figure 2.10.

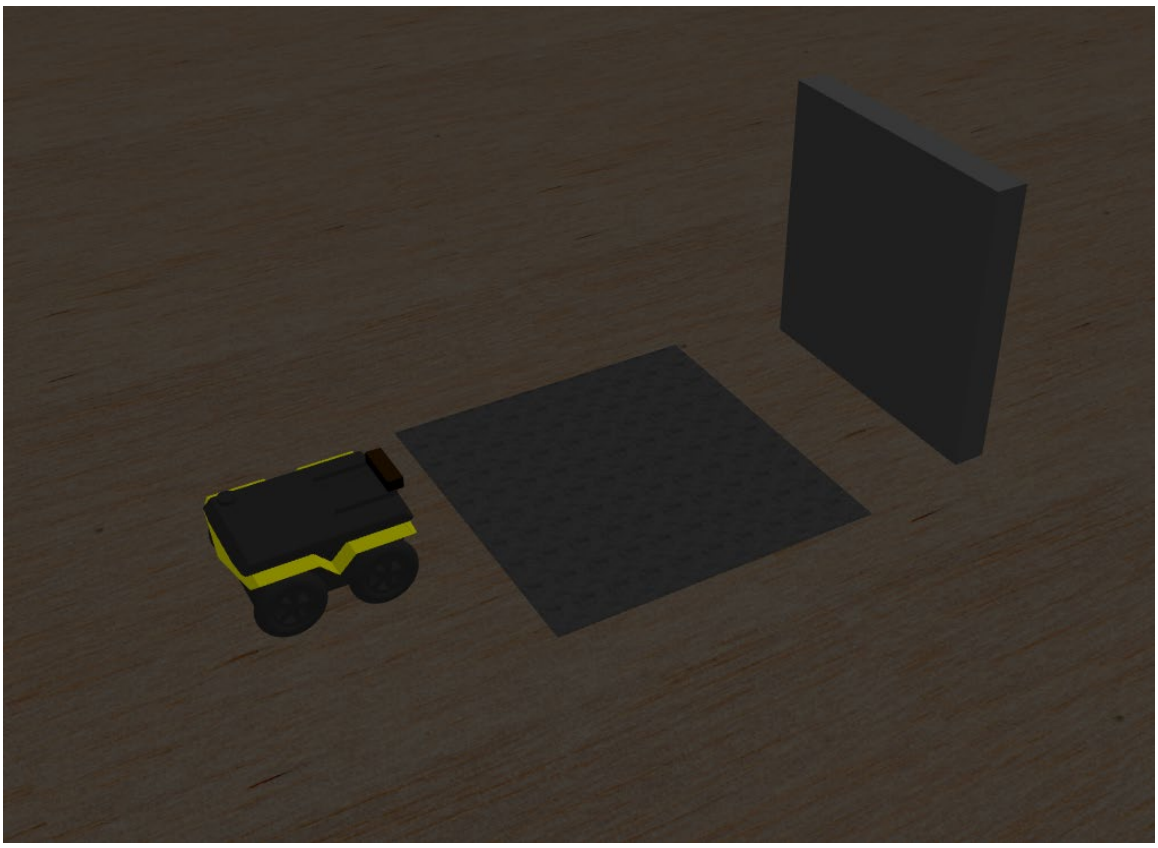

Figure 2.10: The Jackal UGV moved to neutral position after movement response.

The associative learning network and three new functioning subsystems are ready to be tested for the simulation experiment, shown in Figure 2.11.

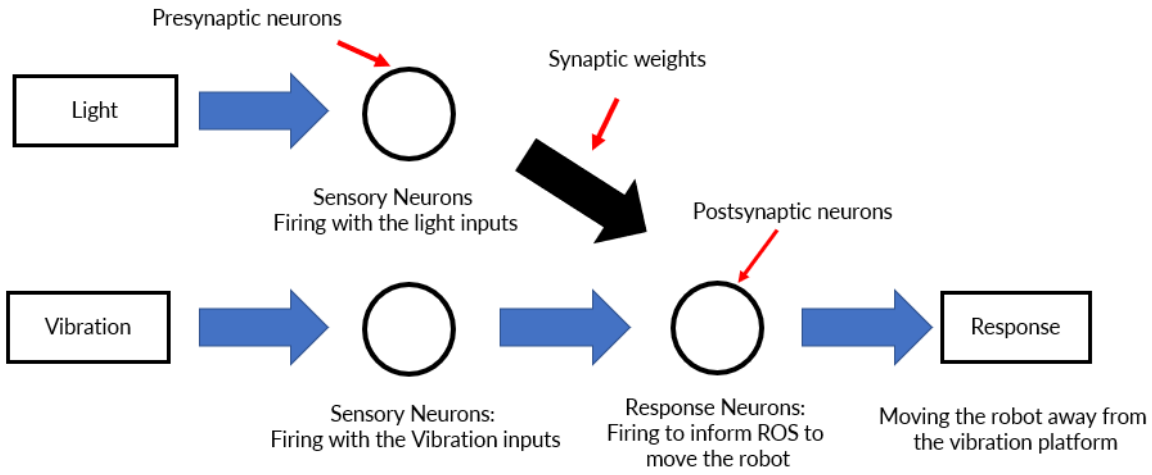

Figure 2.11: System for associative learning with mobile robotics experiment.

Initial simulation issues revealed the need for multithreaded programming to force Nengo to execute simulation time steps as close to real-time as possible. Nengo's original operation is to execute the time steps as fast as it can, while ROS and the Gazebo simulator's default behavior is to try to execute in real-time. This leads to undesirable behavior because Nengo is receiving time stretched "slow-motion" data. This real-time modification enables Nengo networks to be executed with a cumulative difference of less than one timestep from perfect real-time throughout the execution, which means it is perfectly optimized and any further improvements would yield no benefit. The modification proved invaluable when migrating from simulation to a real-world experiment and increasing the network size. The NengoGUI version of Nengo does include real-time simulation execution; however, the techniques used in this work yield

significantly more accurate and consistent execution timing. This is presumably due to the custom real-time Nengo attempting to keep the average time step execution in sync with real-time, instead of each individual time step. Furthermore, NengoGUI is much more resource intensive and computationally limited than standard Nengo, which makes it poorly suited for larger, more complicated networks.

The real-time Nengo modifications enable the experiment to be successfully simulated in the Gazebo environment. First, the light is activated to demonstrate the lack of response to the CS, then the vibration table is "activated" by replaying vibration data to show the US triggering the movement response. Next, training cycles are performed in which the two stimuli are presented simultaneously in overlapping periods. Finally, the light is again activated to trigger the movement response without the vibration stimulus, demonstrating the successful associative learning process. The learning is accomplished through the Hebbian learning modification of the synaptic weight. The same learning rate η is used as before: 2×10^{-5}.

2.2 Experimental Validation

Next, the associative learning experiment is migrated to the real-world through a series of modifications to the simulation experiment. The vibration input node and vibration neuron have already been optimized for the vibration signals coming from the real-world vibration platform, so no additional adjustments are necessary. Similarly, the existing movement response neuron and output node are sufficient. The new experimental setup, shown in Figure 2.12, differs mainly in the brightness perception process.

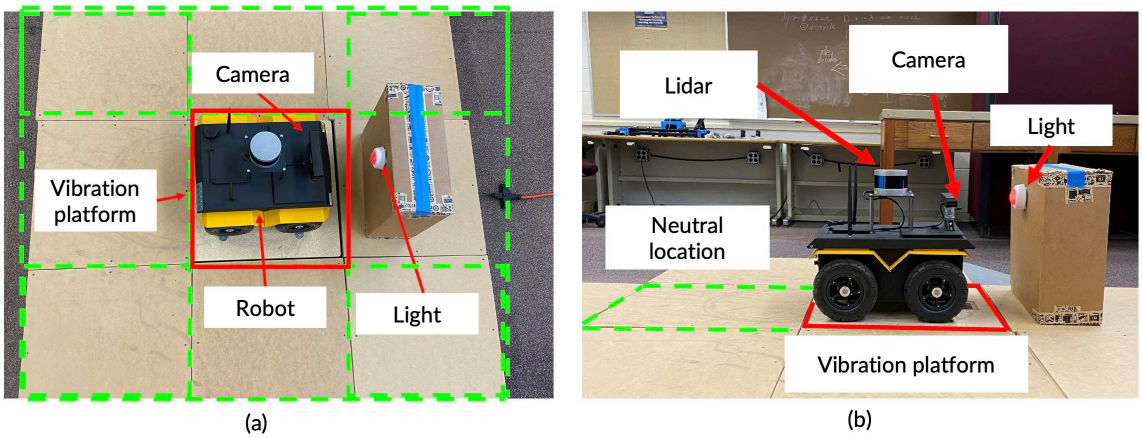

Figure 2.12: Experimental setup for real-world associative learning experiment.

The Jackal UGV sits atop an eight inch tall testing platform consisting of nine 23 inch by 23 inch wooden panels constructed for the experiment. The center panel is a vibration platform, outlined in red in Figure 2.12, with the vibration table beneath it. The vibration platform provides the 15 Hz vibration signals read by the IMU. The new background, light panel, and environmental lighting provide completely different images, shown in Figure 2.13, for the new camera, the Jackal UGV's ZED 2.

Figure 2.13: Images from the ZED 2 camera showing the light panel on and off.

The new camera requires a revised Nengo camera process and ROS service to input image data into the network, but the 5x3 image center pixel output setup remains the same. The 5x3 images of the light panel are shown in Figure 2.14.

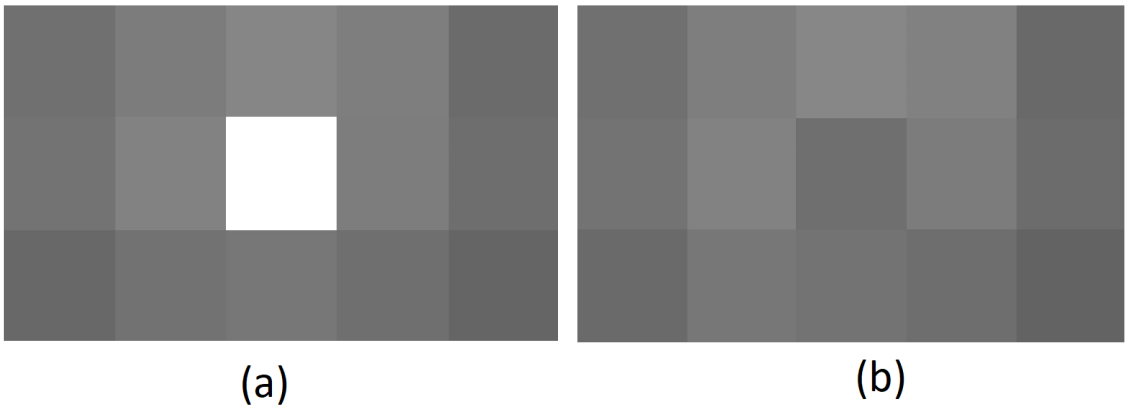

Figure 2.14: Example images of the light panel from real-world experiment.

Because of the new environment, the LIF parameters must again be empirically derived, yielding the values listed in Table 2.3.

Table 2.3: LIF neuron parameters for associative learning in real-world mobile robotics experiment.

Neuron Types	τ_{RC}	A	V_{reset} (V)	V_{th} (V)
(US) Vibration neuron	0.02	1.3	0.6	1.0
(CS) Brightness neuron	0.02	0.3	-1.0	1.0
(Response) Movement neuron	0.02	1.0	0.01	1.0

The updated brightness neuron parameters reproduce the desired brightness detection function of the neuron as shown in Figure 2.15.

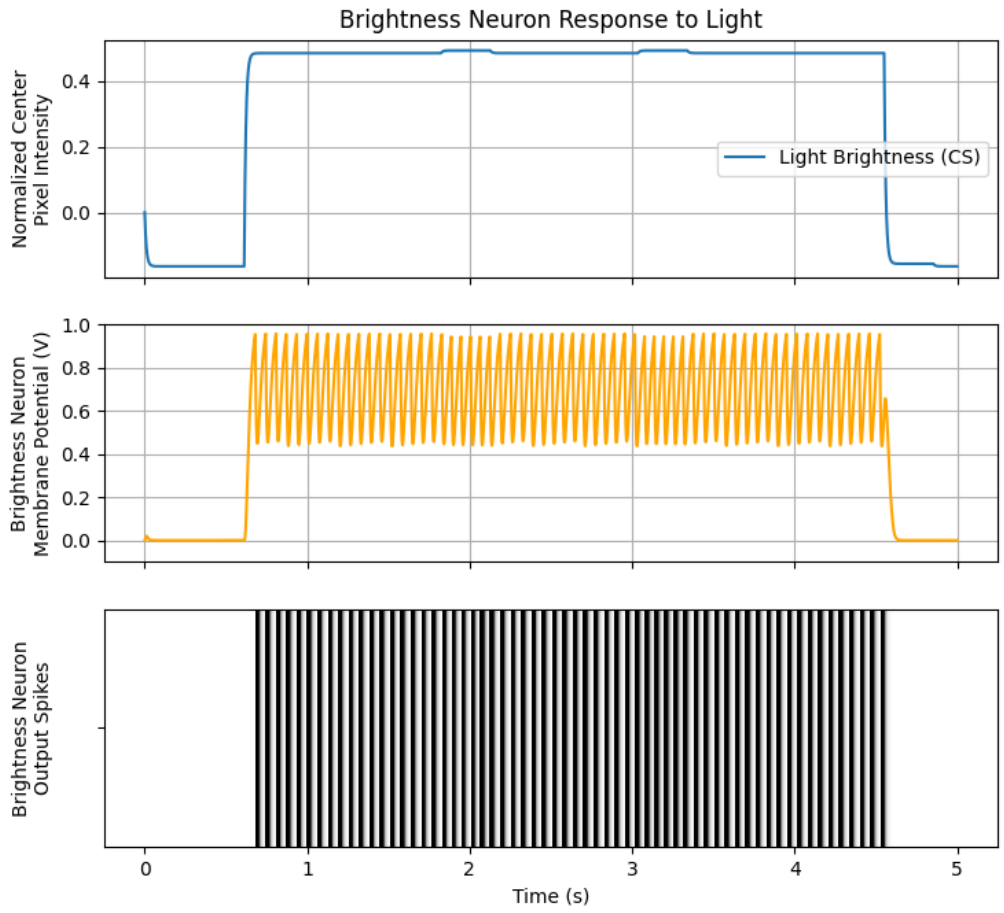

Figure 2.15: Brightness neuron firing in response to the activated light panel.

Now that all subsystems are functional in the real-world experimental setup, The same process from the simulation is used to train the network, shown in Figure 2.16.

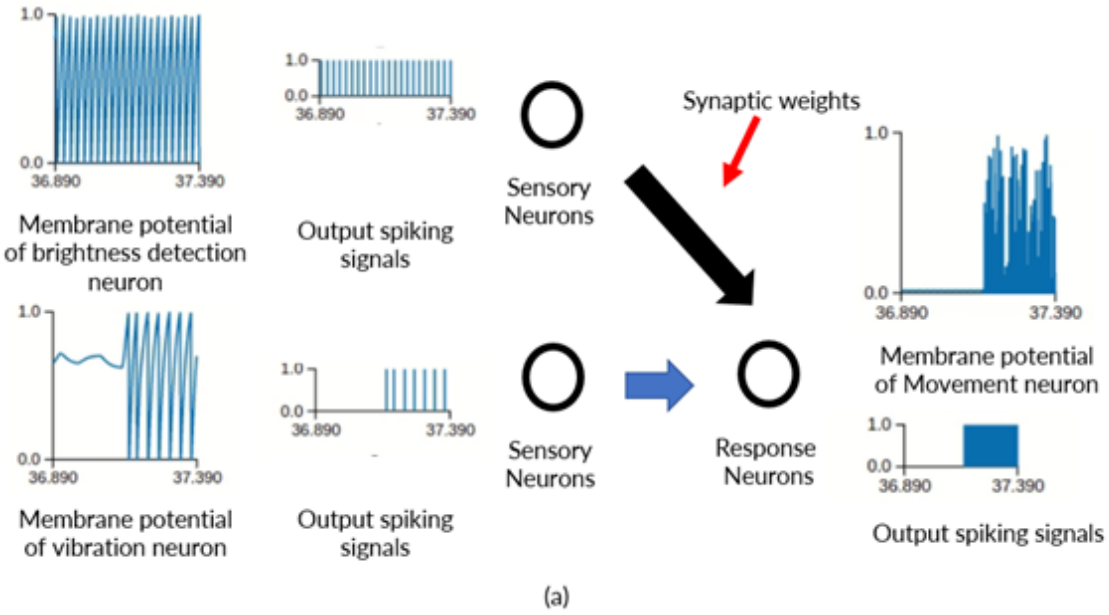

Figure 2.16: Real-time experiment data of associative memory with UGV: (a) The membrane potentials and spiking outputs of the brightness detection neuron, vibration detection neuron, and movement neuron. (b) The UGV is moving away from the vibration platform

Again, the same value of 2×10^{-5} is used as the learning rate η for the Hebbian learning process. The strength of the synaptic connection in the conditional pathway increases as the stimuli are presented simultaneously. The preliminary results show successful associative learning in a mobile robotics rendition of the fear conditioning experiment. The experiment comparison is summarized in Figure 2.17.

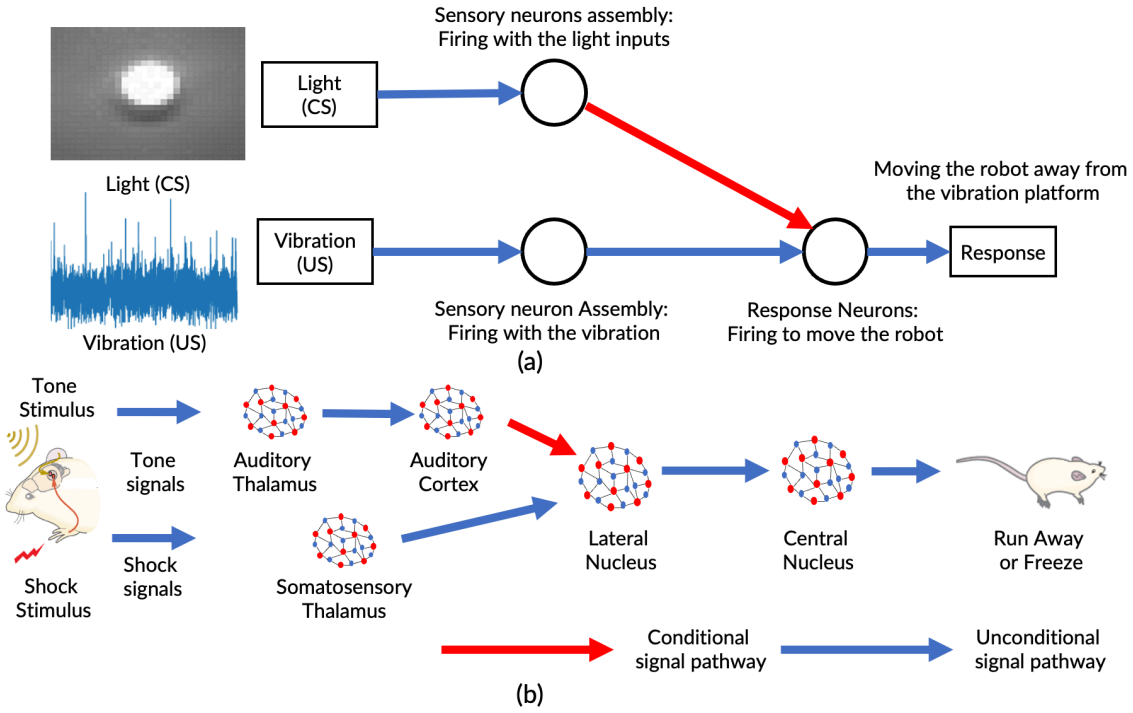

Figure 2.17: Neuromorphic system for reproducing fear conditioning in rats.

The system uses biological rationales to implement associative learning; however, these neural assemblies are still over simplified and do not easily extrapolate to more complicated information processing tasks.

2.3 Adding the Locally Competitive Algorithm

To improve the capabilities of the associative learning network, a sparse coding inspired LCA network is created for the brightness detection neural assembly. The same camera process is used to interface with the image stream, but the images are now resized to 24x48 pixels, shown in Figure 2.18, to provide a more realistic image processing scenario.

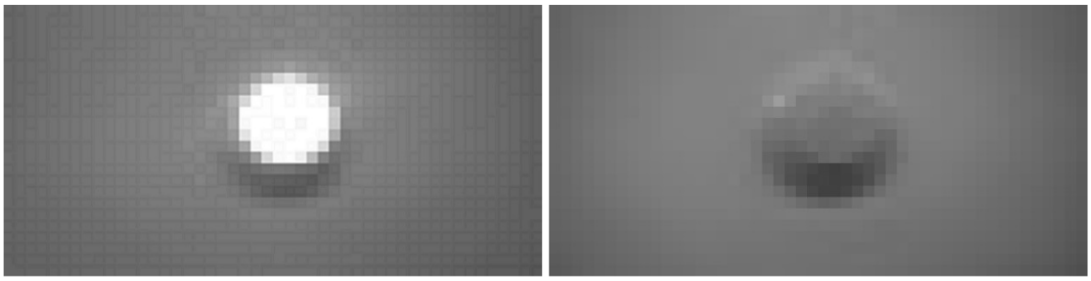

Figure 2.18: Higher resolution images of the light panel on (a) and off (b).

Recall that LCA attempts to represent an input in terms of features from a dictionary, Φ. Each feature, Φ_i, is the size of the LCA input, in this case a 3x3 image patch. The images in Figure 2.18 are divided into nine regions that are further subdivided into 3x3 image patches, shown in Figure 2.19.

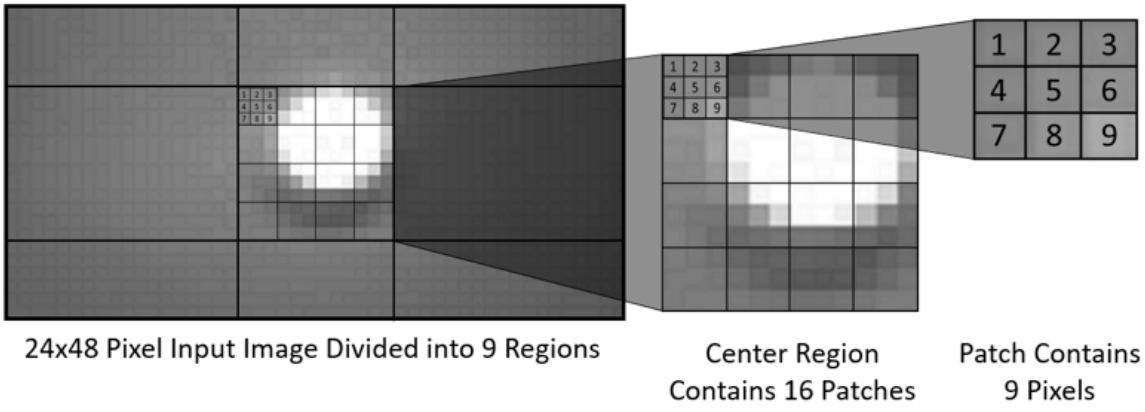

Figure 2.19: Image division layout showing regions and patch structure.

Only the center region containing the light is used as an input to the LCA network to reduce extraneous neural activity computations. The input patches typically overlap in convolutional LCA, which introduces connections between the patch feature neurons of overlapping patches. A convolutional stride of three is chosen in order to avoid overlapping patches and isolate each individual LCA network as it is theorized the desired functionality will still be achieved. This simplifies implementation of the LCA networks and reduces the computational resources required to execute the model. Each of the 16 patches in the center region are inputs to individual LCA optimization solving networks. The pixel intensity is converted to spiking input via a spike generator layer. This "rate-codes" the values with Integrate and Fire neurons spiking at a proportional firing rate. A depiction of one of these networks is shown in Figure 2.20.

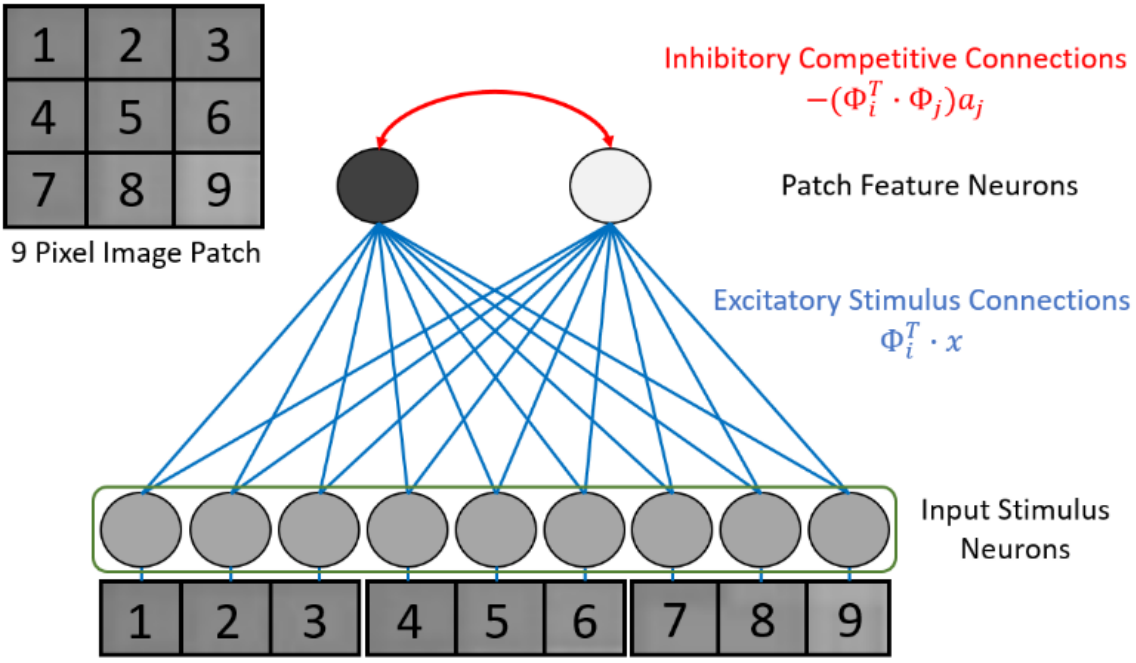

Figure 2.20: Single layer LCA network for brightness detection in one image patch.

This network configuration is referred to as single layer because there is one layer of "feature neurons" solving the LCA optimization function in conjunction with the synapses that connect the feature neurons to the inputs and to each other. For convenience, the LCA equations are restated below.

$$\dot{u} = \frac{1}{\tau}(\Phi^T x - u - (\Phi^T \Phi - I)),\ a = T_\lambda(u) \qquad (5)$$

$$T_\lambda(u) = 0 \ \text{ if } \ u \leq \lambda\ , \ \text{else } \ T_\lambda(u) = u - \lambda \qquad (6)$$

The input x is transmitted from the output of the camera process to the input of LCA layer, which solves for the "sparse code" a^*. The layer solves a^* as firing rates of each feature neuron converge to the coefficients a_i. Each feature neuron receives the input associated with one dictionary atom, or feature, Φ_i because the weights of the synapses

connecting it to the input are the elements of the vector Φ_i. A dark feature is manually created for the dictionary by choosing elements for the vector that are all negative, because the pixel intensities range from -1 to 1. An analogous light feature is created by choosing positive vector elements. It is important that the feature vectors Φ_i all have unit norm, so the magnitude of the feature vectors does not affect the sparsity penalty in (3). In addition, the dictionary Φ should be overcomplete, meaning a should have a larger dimension than x; however, the only two features used in the experiment are light and dark. Hence, the neural assembly for brightness detection can only be referred to as "inspired by" sparse coding. The undercomplete dictionary does not truly solve the sparse coding problem and may not even converge to a solution; however, the underlying competition mechanics are believed to be sufficient for the desired functionality. The dictionary could be made overcomplete by increasing the number of features, and therefore feature neurons, so it is greater than the size of the input. It is also possible to generate learned dictionary features through training with spiking neurons through a modified implementation of LCA [53]. The additional feature neurons could contain a combination of positive and negative as variations of light and dark features with minimal changes to the theory of operation. On the contrary, making them gradients or other basic image elements could potentially reduce unwanted activity of the light feature neuron(s) by providing features that better represent the less definite patches of the image. Furthermore, expanding the input to accept RGB pixel values instead of scalars would allow for more selective dictionary features, theoretically reducing the unwanted activity in feature neurons presented with an input dissimilar to their corresponding

feature. The brightness perception network has a third layer after the LCA layer that is used to integrate the output spikes of the corresponding patch feature neurons in each region, shown in Figure 2.21.

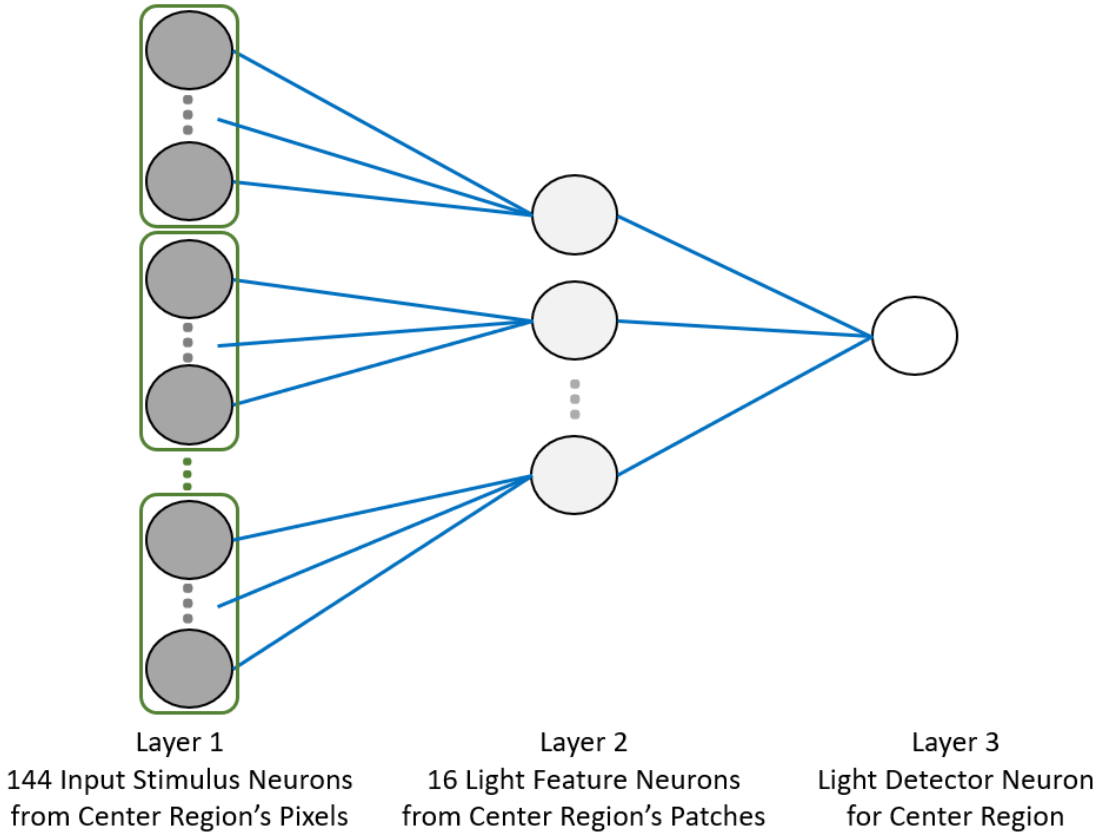

Figure 2.21: Partition of brightness perception network used for the images' center region. Note that only the neurons associated with the light feature are depicted.

Proper implementation of LCA requires the LCA layer (feature neurons) to use the Integrate and Fire neuron model and replace the synapse model with a buffer (one time step delay). Coincidentally, the existing brightness neuron from Table 2.3 performs the

desired function of the Layer 3 neuron in Figure 2.21 without any adjustments. Table 2.4 summarizes the parameters of the neurons in the network.

Table 2.4: LCA neuron parameters for light detection.

Neuron Types	τ_{RC}	A	V_{reset} (V)	V_{th} (V)
(Layer 1) Input layer neuron	∞	25	0.5	1.0
(Layer 2) LCA neuron	∞	1.0	-λ = 0.85	1.0
(Layer 3) Light detector neuron	0.02	0.3	-1.0	1.0

The input spike generator layer's parameters are determined so that the maximum input value corresponds to the maximum firing rate of once per timestep. Layer 2's V_{reset} parameter, which implements the sparsity penalty, was empirically adjusted for the desired function. The synapse model is modified by removing the filter and simply multiplying the weight by the spikes to implement spiking LCA. The weights of the synapses connected to the feature neurons are set to the values of the corresponding dictionary atom. The feature neurons have inhibitory connections between them with weights given by the matrix $-(\Phi_i^T \cdot \Phi_j)a_j$. The synaptic weights are summarized in Table 2.5.

Table 2.5: Synaptic weights for LCA neurons.

Presynaptic Connection	**Postsynaptic Connection**	**Weight**
Input spike generator	Light feature neuron	0.111
Input spike generator	Dark feature neuron	-0.111
Dark feature neuron	Light feature neuron	-1.0

Presynaptic Connection	Postsynaptic Connection	Weight
Light feature neuron	Dark feature neuron	-1.0

Intuitively, the weights of the synapses connecting the inputs to the light and dark feature neurons produce excitatory signals when the corresponding light or dark input is applied, and inhibitory signals when the opposing input is applied. It is worth noting that Loihi 1 neurons and almost all biological neurons cannot be both excitatory and inhibitory. Providing the image with the light on, (a) from Figure 2.18, as the input stimulus to the network significantly increases the center region light feature neurons' activity and switching the input image to (b), where the light is off, immediately reduces the activities of the neurons. The images are presented alternately for two second periods producing the activity shown in Figure 2.22.

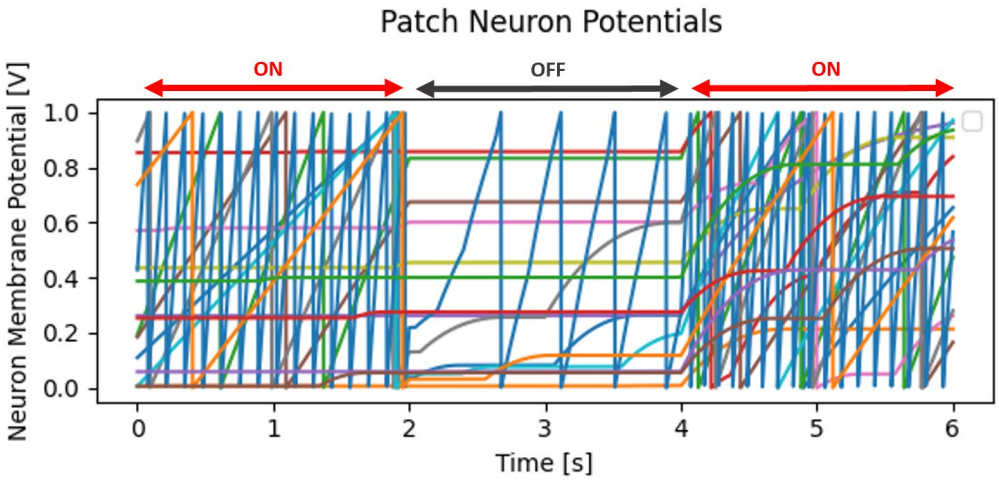

Figure 2.22: Membrane potentials of light feature neurons corresponding to the center region.

The output spikes from the light feature neurons drive the activity of the third layer brightness detection neuron, which is shown in Figure 2.23. The neuron output reliably spikes when the light is on and not when it is off.

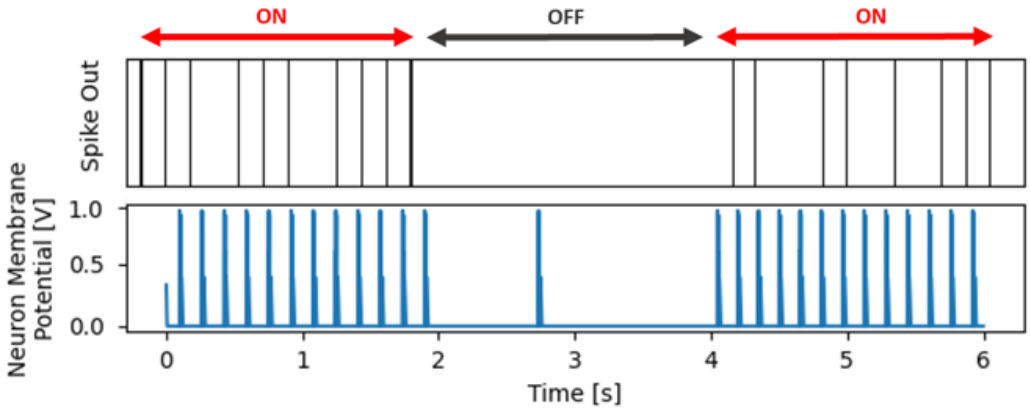

Figure 2.23: Membrane potential and spike output of layer 3 brightness detection neuron corresponding to the center region.

The functioning brightness detection LCA network is easily integrated with associative learning network for the experiment as they were both implemented in Nengo. The existing network is modified by simply replacing the brightness neuron with the LCA network to as the output from the camera process. No further modifications are necessary as the layer 3 neuron connecting to the learning synapse, or conditional pathway, is identical to the previous brightness neuron. The associative learning experiment can now be repeated with the new network configuration, the results are shown in Figure 2.24.

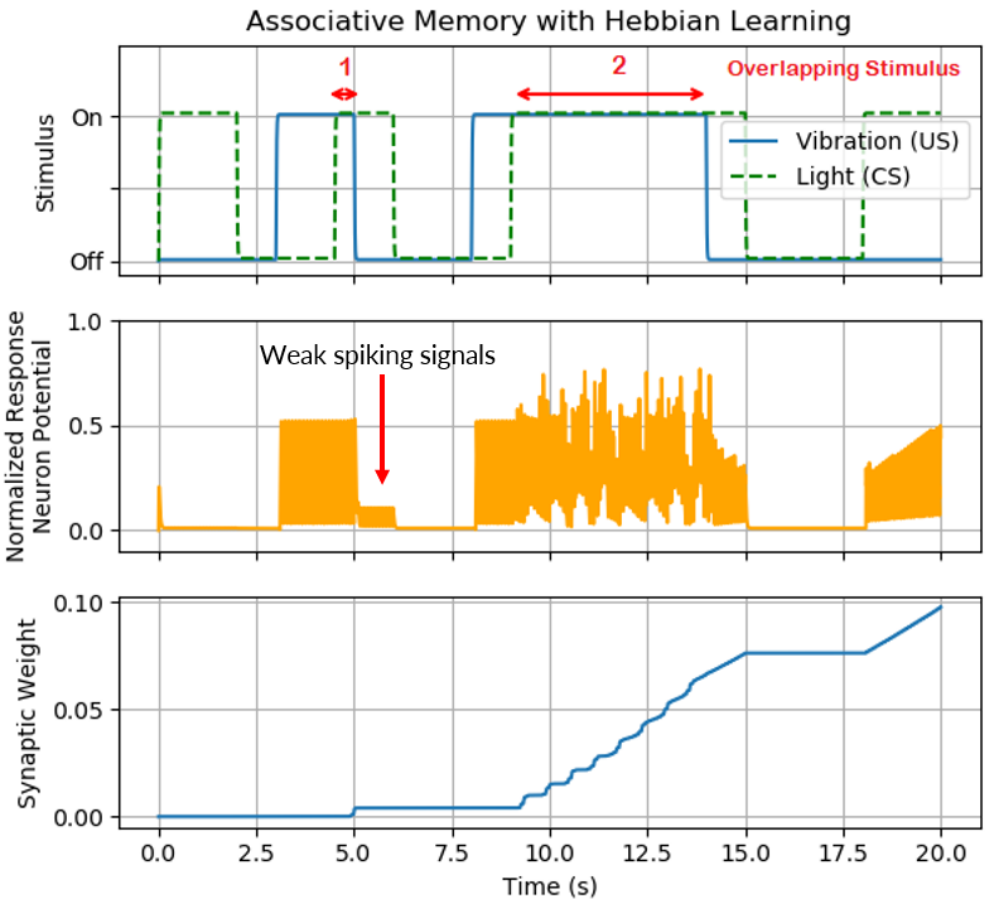

Figure 2.24: Weight change from Hebbian learning in associative learning experiment with LCA.

Integrating the LCA neural assembly into the existing associative learning model yields a network much larger than those found in other state-of-the-art works, compared in **Table 2.6**.

Table 2.6: Comparison of experimental setup and network model size with other state-of-the-art works.

Publication	Neuron Count	Synapse Count	Dataset	Experiment Type	Biology Scenario
[10]	6	3	N/A	Simulation	N/A
[11]	3	1	N/A	Simulation	N/A
[14]	5	6	N/A	Simulation	N/A
[12]	3	1	N/A	Simulation	N/A
[60]	3	1	N/A	Simulation	N/A
[61]	3	2	N/A	Simulation	N/A
[62]	3	2	N/A	Simulation	Cellular Association in Aplysia
[7]	20	100	Pretrained with dataset	Simulation	N/A
This work	1419	1420	No dataset required	Real-world Experiment	Fear conditioning in rats

In addition to the larger network scale, the associative learning in this work reproduces fear conditioning in rats in a real-world experiment without any pretraining, setting it apart from the other works.

Next the applicability of the experiment is expanded with Intel's Loihi neuromorphic hardware. Implementing the neural network on specialized neuromorphic hardware brings the real-world effort of the experiment one step further and takes advantage of the performance and energy efficiency of NC. ABR, the developers of Nengo, created an

extension to the program called Nengo Loihi that utilizes Intel's proprietary NxSDK platform to execute network models created in Nengo using Loihi as the computational backend. Conveniently, the network neurons and modules are compatible with Nengo Loihi without any modifications; however, only the STDP learning rule is implemented, so the extension must be modified to support Hebbian learning. Because of this, only the LCA portion of the network, which is the vast majority, runs on the Loihi. Intel's NxSDK software provides an implementation of LCA optimized for the Loihi, so it is used to create the same LCA network used previously. Loihi 1 is not capable of producing negative output spikes, which are required for the standard version of spiking LCA. Loihi 2 does not have this limitation, so it could be used in the future. A solution to this issue for Loihi 1 was attempted by the NxSDK developers, however the effectiveness of this solution is disputed in [50]. The capabilities of Loihi necessitate a couple other changes to the network. Loihi's synaptic weights are represented with 4-bit of resolution instead of the 24-bit resolution of the standard floating-point representation used by Nengo. The other major change is the reduction of the simulation time step from 1 ms to 20 ms when executing on Loihi. The power consumption of the Loihi is monitored during execution of the LCA network to measure its energy efficiency, shown in Figure 2.25. The power consumption is monitored for the compute logic, represented by VDD, the SRAM memory units, represented VDDM, and the IO interface, represented by VDDIO.

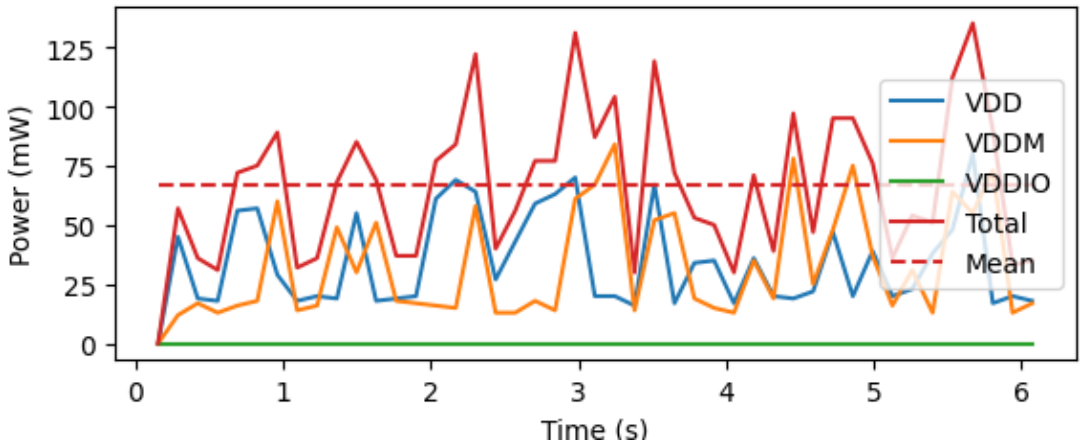

Figure 2.25: Power consumption of LCA network running on Loihi.

The power measurements are made throughout the experiment, and the values are averaged and reported every 8 time steps. Figure 2.25 shows the VDDIO power consumption is negligible compared to the other two measurements. VDD and VDDM have approximately equal average power consumptions of 30 mW and 29 mW, respectively.

3 Conclusion and Future Work

This work implements a classic self-learning paradigm in rats: associative learning (fear conditioning) using a mobile robot and a neuromorphic system (Loihi chip) in an online learning scenario. In specific, a mobile robot is the substitute for the rats in fear conditioning experiments. Two signal pathways are assigned for conditional and unconditional stimulus. In the experiments, vibration signals emulate the unconditional stimulus, while brightness of lights is assigned as the conditional stimulus. Originally, the mobile robot only moves when it detects vibration signals. After providing these two signals concurrently several times, the robot performs a movement when light signals are present alone. The detection of lights and vibrations are implemented with Leaky Integrate and Fire Neurons. In addition, the movement of the robot is controlled by specially designed response neurons. The signal pathway modification during associative memory learning is implemented with Hebbian learning. Compared to other state-of-the-art works, this successfully reproduces the fear conditioning of rats in a real-world scenario with no labeled data or pretraining process.

My own contributions in neuromorphic computing applications for robotics are:

- Reviewing neuromorphic computing field and applications: [63],
- Reproducing fear conditioning of rats with UGVs: [64],
- Implementing associative learning in mobile robots with neuromorphic computing [65].

There are several opportunities for furthering the presented work. The LCA implementation could be improved by utilizing overlapping LCA patches, an overcomplete dictionary, a feature dictionary learned from experimental data, and using the Loihi 2 hardware. Improving the capability of the LCA network to differentiate colors in RGB images instead of brightness in grayscale images would be applicable to more scenarios. A common application for sparse coding and LCA is natural language processing, which can be viewed as processing oscillating one dimensional signals. In a similar manner, LCA could be used to provide a more robust and biologically plausible neural assembly for vibration detection analogous to the one for light detection. The vibration detection could be made more neuromorphic with the simpler modification of using the delta of z-axis acceleration as an input to reduce preprocessing and "eventify" the input so that neural activity corresponds with changes in the environment. Hebbian learning could also be added to Nengo Loihi in order to run the full model on the Loihi. Alternatively, Intel's Lava development platform could be used in place of Nengo for a more tightly integrated system. This would be ideal; however, ROS support in Lava is not yet available at the time of this work. Most robotics works involving neuromorphic systems utilize a neuromorphic Dynamic Vision Sensor (DVS) camera. Ideally, the system would incorporate a DVS camera in place of the ZED 2. These cameras generate event based spiking representations of the input in place of integer pixel values, making them extremely efficient and able to interface with the neuromorphic computing hardware at a much lower level.

4 References

[1] E. Lurie-Luke, "Product and technology innovation: What can biomimicry inspire?," *Biotechnology advances,* vol. 32, no. 8, pp. 1494-1505, 2014, doi: 10.1016/j.biotechadv.2014.10.002.

[2] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. Hudspeth, *Principles of neural science*. McGraw-hill New York, 2000.

[3] T. Kohonen, *Self-organization and associative memory*. Springer Science & Business Media, 2012.

[4] J. Sun, G. Han, Z. Zeng, and Y. Wang, "Memristor-based neural network circuit of full-function pavlov associative memory with time delay and variable learning rate," *IEEE transactions on cybernetics,* 2019.

[5] I. Goodfellow, B. Yoshua, and C. Aaron, "Deep Learning," *Deep Learning,* p. 785, 2016, doi: 10.1016/B978-0-12-391420-0.09987-X.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805,* 2018.

[7] H. An, Q. An, and Y. Yi, "Realizing Behavior Level Associative Memory Learning Through Three-Dimensional Memristor-Based Neuromorphic Circuits," *IEEE Transactions on Emerging Topics in Computational Intelligence,* 2019.

[8] S. G. Hu *et al.*, "Associative memory realized by a reconfigurable memristive Hopfield neural network," *Nature Communications,* vol. 6, pp. 1-5, 2015, doi: 10.1038/ncomms8522.

[9] K. Moon *et al.*, "Hardware implementation of associative memory characteristics with analogue-type resistive-switching device," *Nanotechnology,* vol. 25, 2014, doi: 10.1088/0957-4484/25/49/495204.

[10] J. Yang, L. Wang, Y. Wang, and T. Guo, "A novel memristive Hopfield neural network with application in associative memory," *Neurocomputing,* vol. 227, pp. 142-148, 2017, doi: 10.1016/j.neucom.2016.07.065.

[11] X. Liu, Z. Zeng, and S. Wen, "Implementation of memristive neural network with full-function pavlov associative memory," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 63, no. 9, pp. 1454-1463, 2016.

[12] K. Moon *et al.*, "Hardware implementation of associative memory characteristics with analogue-type resistive-switching device," *Nanotechnology,* vol. 25, no. 49, p. 495204, 2014.

[13] S. B. Eryilmaz *et al.*, "Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array," *Frontiers in neuroscience,* vol. 8, 2014.

[14] X. Hu, S. Duan, G. Chen, and L. Chen, "Modeling affections with memristor-based associative memory neural networks," *Neurocomputing,* vol. 223, pp. 129-137, 2017, doi: 10.1016/j.neucom.2016.10.028.

[15] D. S. Roy *et al.*, "Brain-wide mapping reveals that engrams for a single memory are distributed across multiple brain regions," *Nat Commun,* vol. 13, no. 1, pp. 1-16, 2022.

[16] S. A. Josselyn and S. Tonegawa, "Memory engrams: Recalling the past and imagining the future," *Science,* vol. 367, no. 6473, p. eaaw4325, 2020.

[17] H. Nomura, C. Teshirogi, D. Nakayama, M. Minami, and Y. Ikegaya, "Prior observation of fear learning enhances subsequent self-experienced fear learning with an overlapping neuronal ensemble in the dorsal hippocampus," *Molecular brain,* vol. 12, no. 1, pp. 1-8, 2019.

[18] L. A. DeNardo *et al.*, "Temporal evolution of cortical ensembles promoting remote memory retrieval," *Nature neuroscience,* vol. 22, no. 3, pp. 460-469, 2019.

[19] O. Khalaf, S. Resch, L. Dixsaut, V. Gorden, L. Glauser, and J. Gräff, "Reactivation of recall-induced neurons contributes to remote fear memory attenuation," *Science,* vol. 360, no. 6394, pp. 1239-1242, 2018.

[20] T. Kitamura *et al.*, "Engrams and circuits crucial for systems consolidation of a memory," *Science,* vol. 356, no. 6333, pp. 73-78, 2017.

[21] C. Mead, "Neuromorphic electronic systems," *P Ieee,* vol. 78, no. 10, pp. 1629-1636, 1990.

[22] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature,* vol. 575, no. 7784, pp. 607-617, 2019.

[23] N. Zins, Y. Zhang, C. Yu, and H. An, "Neuromorphic Computing: A Path to Artificial Intelligence Through Emulating Human Brains," in *Frontiers of Quality Electronic Design (QED)*: Springer, 2023, pp. 259-296.

[24] H. An, "Powering Next-Generation Artificial Intelligence by Designing Three-dimensional High-Performance Neuromorphic Computing System with Memristors," Virginia Tech, 2020.

[25] K. Bai and Y. Yi, "Opening the “Black Box” of Silicon Chip Design in Neuromorphic Computing," in *Bio-Inspired Technology*: IntechOpen, 2019.

[26] K. Bai and Y. Yi, "DFR: An Energy-efficient Analog Delay Feedback Reservoir Computing System for Brain-inspired Computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC),* vol. 14, no. 4, p. 45, 2018.

[27] E. Baird, M. V. Srinivasan, S. Zhang, and A. Cowling, "Visual control of flight speed in honeybees," *Journal of experimental biology,* vol. 208, no. 20, pp. 3895-3905, 2005.

[28] R. Kern, N. Boeddeker, L. Dittmar, and M. Egelhaaf, "Blowfly flight characteristics are shaped by environmental features and controlled by optic flow information," *Journal of Experimental Biology,* vol. 215, no. 14, pp. 2501-2514, 2012.

[29] R. Kempter, W. Gerstner, and J. L. Van Hemmen, "Hebbian learning and spiking neurons," *Physical Review E,* vol. 59, pp. 4498-4514, 1999, doi: 10.1103/PhysRevE.59.4498.

[30] N. Levy, D. Horn, I. Meilijson, and E. Ruppin, "Distributed Synchrony of Spiking Neurons in a Hebbian Cell Assembly," *Advances in Neural Information Processing Systems 12,* vol. 14, pp. 129-135, 2000, doi: no DOI, URL correct.

[31] M. C. Van Rossum, G. Q. Bi, and G. G. Turrigiano, "Stable Hebbian learning from spike timing-dependent plasticity," *Journal of neuroscience,* vol. 20, no. 23, pp. 8812-8821, 2000.

[32] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a Hebbian learning rule," *Annu. Rev. Neurosci.,* vol. 31, pp. 25-46, 2008.

[33] K. Bai, Q. An, and Y. Yi, "Deep-DFR: A Memristive Deep Delayed Feedback Reservoir Computing System with Hybrid Neural Network Topology," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019: ACM, p. 54.

[34] M. Davies *et al.*, "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook," *P Ieee,* 2021.

[35] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro,* vol. 38, no. 1, pp. 82-99, 2018.

[36] G. Orchard *et al.*, "Efficient neuromorphic signal processing with loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, 2021: IEEE, pp. 254-259.

[37] P. Miller, *An introductory course in computational neuroscience*. MIT Press, 2018.

[38] P. I. Pavlov, "Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex," *Annals of neurosciences,* vol. 17, no. 3, p. 136, Jul 2010, doi: 10.5214/ans.0972-7531.1017309.

[39] A. Lansner, "Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations," *Trends in neurosciences,* vol. 32, no. 3, pp. 178-186, 2009.

[40] G. S. Snider, "Self-organized computation with unreliable, memristive nanodevices," *Nanotechnology,* vol. 18, 2007, doi: 10.1088/0957-4484/18/36/365202.

[41] W. J. Greig, "Integrated circuit packaging, assembly and interconnections," *Integrated Circuit Packaging, Assembly and Interconnections,* pp. 1-296, 2007, doi: 10.1007/0-387-33913-2.

[42] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial Examples: Attacks and Defenses for Deep Learning," *IEEE transaction on neural networks and learning systems,* vol. 30, no. 9, pp. 2805-2824, 2019, doi: 10.1109/TNNLS.2018.2886017.

[43] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature (London),* vol. 381, no. 6583, pp. 607-609, 1996, doi: 10.1038/381607a0.

[44] Y. Karklin and M. S. Lewicki, "Emergence of complex cell properties by learning to generalize in natural scenes," *Nature,* vol. 457, no. 7225, pp. 83-86, 2009, doi: 10.1038/nature07481.

[45] E. Kim, E. Lawson, K. Sullivan, and G. Kenyon, "Spatiotemporal Sequence Memory for Prediction using Deep Sparse Coding," in *ACM International Conference Proceeding Series*, 2019: ACM, in NICE '19, pp. 1-7, doi: 10.1145/3320288.3320295.

[46] Y. Watkins, A. Thresher, D. Mascarenas, and G. Kenyon, "Sparse Coding Enables the Reconstruction of High-Fidelity Images and Video from Retinal Spike Trains," in *ACM International Conference Proceeding Series*, 2018 2018: ACM, in ICONS '18, pp. 1-5, doi: 10.1145/3229884.3229892.

[47] E. Kim, J. Yarnall, P. Shah, and G. Kenyon, "A Neuromorphic Sparse Coding Defense to Adversarial Images," in *ACM International Conference Proceeding Series*, 2019 2019: ACM, in ICONS '19, pp. 1-8, doi: 10.1145/3354265.3354277.

[48] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Comput,* vol. 20, no. 10, pp. 2526-63, Oct 2008, doi: 10.1162/neco.2008.03-07-486.

[49] P. T. P. Tang, T.-H. Lin, and M. Davies, "Sparse coding by spiking neural networks: Convergence theory and computational results," *arXiv preprint arXiv:1705.05475,* 2017.

[50] K. Henke, M. Teti, G. Kenyon, B. Migliori, and G. Kunde, "Apples-to-spikes: The first detailed comparison of LASSO solutions generated by a spiking neuromorphic processor," in *Proceedings of the ICONS*, 2022, pp. 1-8.

[51] P. T. P. Tang, "Convergence of LCA Flows to (C)LASSO Solutions," 2016, doi: 10.48550/arxiv.1603.01644.

[52] Y. Watkins, A. Thresher, P. Schultz, A. Wild, A. Sornborger, and G. Kenyon, "Unsupervised Dictionary Learning via a Spiking Locally Competitive Algorithm," in *ACM International Conference Proceeding Series*, 2019 2019: ACM, in ICONS '19, pp. 1-5, doi: 10.1145/3354265.3354276.

[53] G. G. Parpart *et al.*, "Dictionary Learning with Accumulator Neurons," in *ICONS*, United States, 2022 2022.

[54] Y. Watkins, E. Kim, A. Sornborger, and G. T. Kenyon, "Using Sinusoidally-Modulated Noise as a Surrogate for Slow-Wave Sleep to Accomplish Stable Unsupervised Dictionary Learning in a Spike-Based Sparse Coding Model," 2020 2020: IEEE, pp. 1482-1487, doi: 10.1109/CVPRW50498.2020.00188.

[55] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: transfer learning from unlabeled data," in *ICML*, 2007 2007: ACM, in ICML '07, pp. 759-766, doi: 10.1145/1273496.1273592.

[56] E. Kim, D. Hannan, and G. Kenyon, "Deep Sparse Coding for Invariant Multimodal Halle Berry Neurons," 2017, doi: 10.48550/arxiv.1711.07998.

[57] T. Bekolay *et al.*, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in neuroinformatics,* vol. 7, p. 48, 2014.

[58] V. DiLuoffo, W. R. Michalson, and B. Sunar, "Robot Operating System 2: The need for a holistic security approach to robotic architectures," *International Journal of Advanced Robotic Systems,* vol. 15, no. 3, p. 1729881418770011, 2018.

[59] "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, 2004, vol. 3: IEEE, pp. 2149-2154.

[60] M. Ziegler *et al.*, "An Electronic Version of Pavlov's Dog," *Advanced Functional Materials,* vol. 22, no. 13, pp. 2744-2749, 2012, doi: 10.1002/adfm.201200244.

[61] Y. V. Pershin and M. Di Ventra, "Experimental demonstration of associative memory with memristive neural networks," *Neural Networks,* vol. 23, no. 7, pp. 881-886, 2010.

[62] H. An, Z. Zhou, and Y. Yi, "Memristor-based 3D neuromorphic computing system and its application to associative memory learning," *2017 IEEE 17th International Conference on Nanotechnology, NANO 2017,* pp. 555-560, 2017, doi: 10.1109/NANO.2017.8117459.

[63] N. Zins, Y. Zhang, C. Yu, and H. An, "Neuromorphic computing: A path to artificial intelligence through emulating human brains," in *Frontiers of Quality Electronic Design (QED) AI, IoT and Hardware Security*: Springer, 2023, pp. 259-296.

[64] N. Zins and H. An, "Reproducing Fear Conditioning of Rats with Unmanned Ground Vehicles and Neuromorphic Systems," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023: IEEE, pp. 1-7.

[65] N. Zins, Y. Zhang, and H. An, "Implementation of Associative Memory Learning in Mobile Robots Using Neuromorphic Computing," 2023.