Michigan Technological University

Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2024

The Integration of Neuromorphic Computing in Autonomous Robotic Systems

Md Abu Bakr Siddique
Michigan Technological University, msiddiq5@mtu.edu

Copyright 2024 Md Abu Bakr Siddique

Recommended Citation

Siddique, Md Abu Bakr, "The Integration of Neuromorphic Computing in Autonomous Robotic Systems", Open Access Master's Thesis, Michigan Technological University, 2024.
https://doi.org/10.37099/mtu.dc.etdr/1743

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, Hardware Systems Commons, and the Robotics Commons

THE INTEGRATION OF NEUROMORPHIC COMPUTING IN AUTONOMOUS ROBOTIC SYSTEMS

By

Md Abu Bakr Siddique

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical and Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2024

© 2024 Md Abu Bakr Siddique

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: *Tan Chen*

Committee Member: *Yan Zhang*

Department Chair: *Jin W. Choi*

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I am deeply thankful to my academic advisor, Dr. Hongyu An, for his unwavering support, encouragement, and guidance throughout my master's program. His faith in me and the liberty he allowed me to explore innovative concepts throughout my research period at MTU have been incredibly meaningful. As an early member of his team, I had the unique opportunity to observe and contribute to the growth of the Brain-Inspired AI Laboratory from its inception to full maturity. The insights and experiences gained during this time are sure to play a significant role in shaping my future career.

I also want to express my appreciation to my thesis committee members, Dr. Tan Chen and Dr. Yan Zhang, for their continuous support and guidance during the drafting of my thesis and in the lead-up to my final defense. Their valuable feedback and recommendations have significantly enhanced my thesis in numerous aspects. Their constructive criticism and insights, especially in the initial phases of my research, were crucial and greatly appreciated.

I extend my gratitude to the graduate and undergraduate students at MTU with whom I collaborated and served as a mentor. Their contributions were crucial in developing and reinforcing significant ideas for many projects. Their vibrant energy and positivity infused me with motivation every single day.

I owe a debt of gratitude to Dr. Dylan Muir and his team at SynSense for their assistance when I faced challenges with the Xylo chip in audio classification tasks. Dr. Muir generously took the time to resolve all my queries. Equally, the Agilex support team deserves my thanks for their readiness to answer every question I had regarding the Agilex LIMO robot. My appreciation also extends to Dr. Traci Yu from Michigan Tech's biomedical department for her insightful advice on Parkinson’s disease. I am eager to have the chance to work with and collaborate with these remarkable individuals and distinguished researchers again in the future.

I feel privileged to have met and worked with some exceptional faculty and distinguished academics at Michigan Tech. My gratitude goes to Dr. Hongyu An, who introduced me to the practical aspects of robotics. His dedication and discipline have left a lasting impression on me. My appreciation also extends to Dr. Shane Oberloier, Prof. Christopher Cischke, Prof. Trever Hassell, and Dr. Glen Archer for their mentorship, advice, and encouragement, which were instrumental in developing my teaching skills—the support I genuinely value. I'm also grateful to Mr. Mark Sloat for helping me set up the camera and T-maze in our lab.

My gratitude extends to Prof. Jin Choi and the Department of Electrical and Computer Engineering at Michigan Tech for their support in the form of a Graduate Teaching Assistantship throughout my time as a master's student. I am equally thankful to the graduate school and Dr. Hongyu An for granting me a research assistantship during the summer of 2023.

Finally, I extend my heartfelt gratitude and appreciation to my supportive family and friends. My parents, siblings, and close friends have been a constant source of strength, guiding me through the most difficult times. They have been my cheerleaders in pursuit of success, my comfort during failures, and have offered their unwavering support in every possible way. I owe my academic career to them.

List of Abbreviations

DNNs	Deep Neural Networks
UGV	Unmanned Ground Vehicle
SNN	Spiking Neural Network
AFE	Analog Front End
AI	Artificial Intelligence
ML	Machine Learning
SWaP	Size, Weight, and Power
ANNs	Artificial Neural Networks
CMOS	Complementary Metal Oxide Semiconductor
ASICs	Application-Specific Integrated Circuits
LIF	Leaky Integrate and Fire
US	Unconditional Stimulus
CS	Conditional Stimulus
RS	Response Stimulus
DL	Deep Learning
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
IMU	Inertial Measurement Unit
SSC	Spiking Speech Command
Xylo HDK	Xylo Hardware Development Kit
TPUs	Tensor Processing Units
HSV	Hue Saturation Value

Abstract

Deep Neural Networks (DNNs) have come a long way in many cognitive tasks by training on large, labeled datasets. However, this method has problems in places with limited data and energy, like when planetary robots are used or when edge computing is used [1]. In contrast to this data-heavy approach, animals demonstrate an innate ability to learn by communicating with their environment and forming associative memories among events and entities, a process known as associative learning [2-4]. For instance, rats in a T-maze learn to associate different stimuli with outcomes through exploration without needing labeled data [5]. This learning paradigm is crucial to overcoming the challenges of deep learning in environments where data and energy are limited. Taking inspiration from this natural learning process, recent advancements [6, 7] have been made in implementing associative learning in artificial systems. This work introduces a pioneering approach by integrating associative learning utilizing an Unmanned Ground Vehicle (UGV) in conjunction with neuromorphic hardware, specifically the XyloA2TestBoard from SynSense, to facilitate online learning scenarios. The system simulates standard associative learning, like the spatial and memory learning observed in rats in a T-maze environment, without any pretraining or labeled datasets. The UGV, akin to the rats in a T-maze, autonomously learns the cause-and-effect relationships between different stimuli, such as visual cues and vibration or audio and visual cues, and demonstrates learned responses through movement. The neuromorphic robot in this system, equipped with SynSense's neuromorphic chip, processes audio signals with a specialized Spiking Neural Network (SNN) and neural assembly, employing the Hebbian learning rule to adjust synaptic weights throughout the learning period. The XyloA2TestBoard uses little power (17.96 μW on average for logic Analog Front End (AFE) and 213.94 μW for IO circuitry), which shows that neuromorphic chips could work well in places with limited energy, offering a promising direction for advancing associative learning in artificial systems.

1 Introduction

The burgeoning field of neuromorphic computing, drawing inspiration from the natural world and the principles of neuroscience, is reshaping the landscape of science and engineering. This novel computing paradigm seeks to emulate the exceptional efficiency and capabilities of biological nervous systems, particularly focusing on associative learning—a pervasive form of online learning observed in the animal kingdom. Associative learning enables animals to form and memorize connections between previously unrelated events based on temporal proximity, contrasting with the data-driven methodologies predominant in modern Artificial Intelligence (AI) and Machine Learning (ML) [8]. Synaptic plasticity in the nervous system makes this learning simpler. This is when the strength of synaptic connections between neurons changes based on how they are active. This makes it easier to remember how events are connected by making those connections stronger [9, 10].

Deep Neural Networks (DNNs) have shown remarkable proficiency in various cognitive tasks by relying on extensive datasets for training [11-14]. The larger datasets and neural networks lead to higher accuracy [14, 15], thereby necessitating a demand for excessive pursuit of the large scale of datasets and neural networks [14-18]. This process involves adjusting weights through algorithms to minimize discrepancies between the output and the labeled truth in datasets. However, the escalating scale of DNNs and their reliance on large datasets present challenges like high power consumption, data scarcity, and reduced flexibility in autonomous operations. These limitations are particularly problematic in applications with constraints on Size, Weight, and Power (SWaP) [14, 15], such as planetary rovers that require high adjustability and minimal human intervention in energy- and communication-limited environments [15, 19, 20].

To address these challenges, neuromorphic systems, mimicking the associative learning of animals, offer a promising alternative. By emulating brain functions, these systems can facilitate more energy-efficient AI, enhancing the autonomous operating capabilities of intelligent robots [21, 22]. This approach not only provides a solution to the limitations of traditional DNNs but also leverages the self-learning mechanism of associative learning, allowing robots to adapt to their environment by interacting with and memorizing concurrent events [23-27]. Putting neuromorphic computing and associative learning together is a big step forward in making smart systems that can work efficiently and on their own in a wide range of difficult situations.

1.1 Motivation and Contribution

Associative learning in AI systems offers an innovative approach to self-learning through environmental interaction, significantly deviating from traditional data-driven deep learning methods like DNNs. This approach circumvents the drawbacks of relying on extensive datasets, which are not only challenging to create and verify but also result in prolonged training phases and substantial computational demands. Instead, associative learning leverages signal pathway modification, requiring fewer repetitions and reducing energy demands [28].

Associative learning has been shown to work in neuromorphic systems recently, but these have mostly been limited to experiments involving basic tasks and compact neural networks [26, 29-38]. Many existing models also depend on pretraining with labeled datasets [33-37]. However, our work advances beyond these limitations by implementing real-world associative learning experiments, particularly focusing on the classical paradigm of fear conditioning in T-maze navigation in two different experimental setups.

The classic T-maze experiment in rats for fear conditioning is a variation of the standard T-maze setup, specifically designed to study fear responses and conditioned learning. In this context, the T-maze is used to create a scenario where rats learn to associate a particular environment or cue with an aversive stimulus, typically a mild electric shock.

The T-maze consists of a starting arm and two choice arms. One of the arms is designated as the 'safe' arm, and the other as the 'shock' arm, where a mild electric shock is delivered through the floor. During the training phase, the rat is placed in the maze and allowed to explore. When it enters the shock arm, it receives a mild electric shock, creating an unpleasant experience. This is repeated several times across different sessions, conditioning the rat to associate the shock arm with the negative stimulus. In the testing phase, the rat is again placed in the maze, but this time, no shock is administered. The researchers observed that the rat avoids the arm previously associated with the shock, which demonstrates associative learning.

Our first experiment takes inspiration from the classic T-maze associative learning observed in rodents [39]. Here, a mobile robot equipped with the Xylo neuromorphic chip [40] is used to replicate the spatial navigation and decision-making processes observed in animals. The robot navigates a T-maze, learning to associate different audio and visual signals with certain outcomes. Audio command data and red color serve as conditional and unconditional stimuli, respectively. The robot's task is to navigate the maze and learn from these stimuli, adjusting its behavior based on the learned associations. This experiment not only demonstrates the robot's ability to process complex sensory information but also showcases the efficiency of the neuromorphic system in real-world scenarios.

Numerous areas in the brain, such as the frontotemporal amygdala and hippocampus, play a critical role in learning. Modern advanced associative learning models can't really copy the old T-maze tests that are used in studies because their neural networks are too small [31-37]. Basic neural network models struggle to handle complex signals like audio and visual inputs. Numerous neural networks, rather than a small number of neurons in the brain, process these types of signals [41-46]. To overcome these challenges, a system using a large-scale, biologically realistic neural network is employed to process both audio and visual inputs. In this setup, a mobile robot equipped with sensors acts as a replacement for rats used in standard T-maze tests. The Xylo neuromorphic chip is utilized as the computational core for processing audio signals in associative learning tasks. During these experiments, commands for left or right directions simulate the auditory stimuli, while the color red replicates the shock signals given to rats. Consequently, the color red represents an adverse stimulus, whereas the audio commands

are neutral. The robot's avoidance of the red-colored arm in favor of the non-red arm is an imitation of a fear response. Two distinct neural assemblies oversee the robot's ability to recognize both auditory commands and visual cues. These assemblies are linked to a response neuron that triggers the robot's movement through two separate signal pathways. The pathway with a weaker synaptic connection acts as the conditional signal route, and the other with a robust synaptic connection functions as the unconditional signal route.

The second experiment revolves around the implementation of associative learning in a mobile robotics application. This setup aims to replicate the classic fear conditioning experiment traditionally performed with rats [47] in a T-maze setting. In this scenario, a mobile robot equipped with various sensors acts as the subject. The robot is trained to associate two stimuli: a visual stimulus (red color) and a vibration stimulus (shock), akin to the neutral and undesirable stimuli in fear conditioning. The process of learning and the subsequent behavioral changes in the robot's movements reflect associative learning, like the responses observed in rats. This experiment stands out for its real-time online learning capabilities without pretraining, a novel approach in the field of AI and robotics. The large-scale neural assembly in the neuromorphic system allows for efficient processing of complex signals like visual inputs, an improvement over a simpler neural network model.

These experiments represent significant strides in associative learning. Fear conditioning works like a complex learning model at both the cellular (Hebbian learning) and behavioral (neural assemblies) levels. Meanwhile, the T-maze experiment emulates rodent associative learning, applying these principles in real-time scenarios. These approaches contribute to the field by offering robust, real-world applications of associative learning, free from the constraints of large-scale datasets and pretraining. These studies represent a shift from traditional data-driven approaches, reducing the dependency on large, labeled datasets and extensive training. The integration of neuromorphic systems in mobile robots, simulating complex learning processes found in animals, opens new avenues for research and practical applications in areas where size, weight, power constraints, and autonomy are critical. The combination of these approaches demonstrates the versatility and potential of neuromorphic systems for replicating complex biological learning processes, marking a significant advancement in the fields of artificial intelligence and robotics.

The contributions of this research are significant in several aspects, including:

1. This work successfully implemented associative learning for online learning in mobile robotics applications using neuromorphic chips, bypassing the need for pretraining and large datasets.
2. Classic animal learning models were simulated, such as fear conditioning in T-maze navigation. This provides strong biological reasoning from the cellular (Hebbian learning) to behavioral (neural assemblies) levels.
3. This work, focused on signal pathway adjustment, introduces a unique learning methodology for associative learning.

4. These advancements open new possibilities for AI applications in scenarios demanding adaptability, energy efficiency, and real-time learning, marking a step forward in the endeavor to mimic complex animal learning and exploration capabilities in AI systems.

1.2 Background

By dissecting and understanding the biological processes that allow animals to learn through association, a new approach to self-learning is developed. This approach applies this concept of associative learning to brain-inspired computing systems. This section will present the latest advancements in neuromorphic (brain-inspired) computing systems and the hardware used for such computing. It will then delve into the details of how associative learning functions, ranging from the level of individual cells to the overall behavior exhibited.

1.2.1 Neuromorphic Computing

Neuromorphic computing, designed to emulate the complex functions of human brains, is revolutionizing the field of Artificial Intelligence. The human brain, with its capability to perform intricate tasks using an astonishingly low amount of energy, approximately 20 watts, serves as the primary inspiration for these systems [48-50]. This energy efficiency far surpasses that of modern computers, which require significantly more power for basic tasks. For instance, a typical computer uses around 250 watts just to recognize 1,000 item types, whereas the human brain can perform various complex cognitive functions, including stimulus classification and future predictions, within its energy budget [24].

The remarkable efficiency and computational prowess of the human brain are attributed to its billions of neurons and trillions of synapses, creating a complex, three-dimensional neural structure with a high level of interconnectedness. Each neuron, capable of communicating with over ten thousand others, serves as a signal processing unit, while synapses act as connectors and are pivotal in learning and memory. This complex structure allows for parallel processing, high connectivity, adjustable network topology, and spike-based information representation.

Carver Mead's concept of neuromorphic computing, which aims to take advantage of these traits [51-54], was born in the 1980s. It aims to create neuromorphic hardware [55-60] that mimics the brain's architecture and learning methods [61-63], thus overcoming the limitations of traditional von Neumann computer architecture, which suffers from the inefficiency of separating computing and memory units [64]. On the other hand, the brain's sparse, event-based computation and colocation of computational (neurons) and memory (synapses) units make it a model for more powerful and efficient computer systems [24, 65].

These systems utilize Spiking Neural Networks (SNNs) [66-68], a form of Artificial Neural Networks (ANNs) that more closely emulate biological neural networks. SNNs operate on dynamic binary-spiking inputs, making them fundamentally different from traditional ANNs with fixed, continuous-valued activities. This spiking mechanism,

reflective of how neurons in the human brain communicate, is essential for the energy efficiency and real-time processing capabilities of neuromorphic systems.

Researchers are looking into both traditional Complementary Metal Oxide Semiconductor (CMOS) technology and new devices such as memristors and synaptic transistors [69-71] to make neuromorphic computing possible. These technologies aim to replicate the brain's efficient processing and learning capabilities, potentially leading to a new era in computing that is faster, more energy-efficient, and capable of handling complex cognitive tasks much like the human brain.

By marrying the principles of neuroscience with advanced computing technology, neuromorphic systems stand at the forefront of AI development, promising to push the boundaries of machine intelligence and computational efficiency [72]. Their evolution reflects a significant shift from traditional computing paradigms, positioning them as a critical component in the future landscape of technology and AI [70].

Neuromorphic systems have taken a significant leap forward with the development of advanced neuromorphic chips like the Xylo-Audio (SYNS61201) [73], a standout member of the Xylo family of Application-Specific Integrated Circuits (ASICs) [40]. These chips are specifically designed for processing digital Spiking Neural Networks (SNNs). What sets the Xylo-Audio apart is its innovative analog audio front-end, which efficiently converts single-channel audio inputs into asynchronous events for SNN processing. This real-time operation is remarkably energy-efficient, consuming less than 10 mW, making it ideal for a variety of applications such as environmental analysis, keyword detection, and monitoring of various biological and industrial activities. Figure 1.1 shows the Xylo-Audio board used in this work.

The Xylo-Audio development kit brings a new dimension to convolutional audio processing. It's not just for keyword detection; its versatility extends to identifying a broad range of audio attributes. Alongside the chip, the kit includes a high-quality MEMS microphone, a power measurement device, an external audio input port, and a USB-C port. It's compatible with the open-source Python library Rock-pool and the SynSense device toolchain Samna, offering a comprehensive solution for real-time audio detection.

Rockpool is an open-source Python package that focuses on the development of signal processing applications using SNNs. It provides facilities to build, simulate, train, test, and deploy networks, either in simulation environments or on event-driven neuromorphic hardware. Rockpool supports various simulation backends, including Brian2, NEST, Torch, JAX, Numba, and NumPy. This versatility makes it a powerful tool for machine learning applications based on SNNs, though it is not designed for detailed simulation of biological networks [74, 75].

Samna, on the other hand, serves as the developer interface to the SynSense toolchain and runtime environment, facilitating interaction with all SynSense devices. It is designed for efficiency and user-friendliness, featuring a Python API with the core running in C++. Samna boasts an event-based stream filter system that enables real-time, multi-branch

processing of the event-based stream from the device. Its just-in-time compiler integration allows for adding user-defined filter functions at runtime, enhancing flexibility to accommodate various scenarios [76, 77].

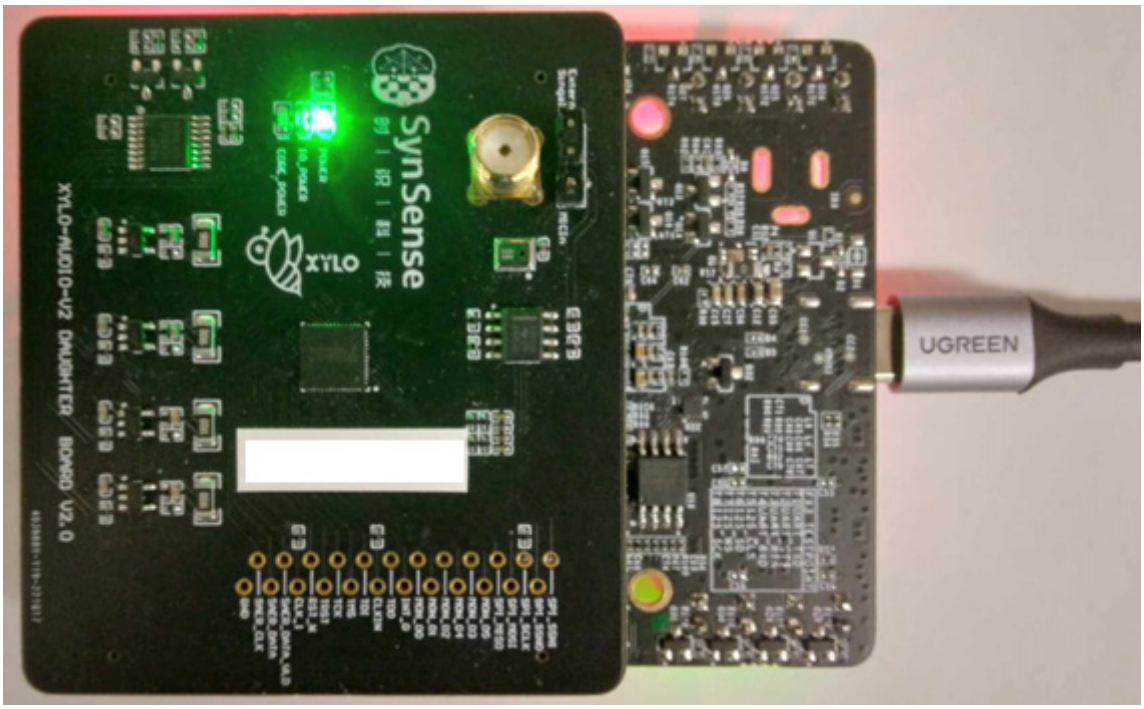

Figure 1:1: Xylo Audio neuromorphic chip.

The Xylo-Audio chip also boasts impressive technical features. Its Analog Front End (AFE) can handle both single-ended and differential modes, with event-based analog feature extraction for energy efficiency. The programmable AFE gain adapts to different sound intensities. The chip supports up to 1000 reservoir neurons and 8 classifications, with a configurable interrupt for signaling classification completion. Additionally, it has an internal operating frequency of up to 100 MHz and a low memory footprint, emphasizing its efficiency.

The electrical characteristics of the device include a core operating voltage of 1.1 V, an IOVDD voltage of 2.5 V, and a digital communication range of 0-2.5 V. It processes input signals between 100 Hz and 20 kHz and starts operating within 0.5 seconds, with a typical power requirement staying under 350 µW.

In summary, the Xylo SNN core is a cutting-edge component capable of simulating a Leaky Integrate and Fire (LIF) spiking neuron population. It supports up to 1000 hidden and 8 output neurons and accommodates a wide range of inputs and outputs. Its high customizability allows users to adjust synaptic and membrane time constants and set individual spiking thresholds for each neuron. The AFE module improves this ability by using a band-pass filter bank to split a single-channel analog signal into 16 separate channels. This makes neuromorphic computing possible and makes it work well.

Table 1.1 shows the features of Xylo Audio neuromorphic chip.

Table 1.1: Features of Xylo neuromorphic chip.

Feature	**Xylo Neuromorphic Chip [78]**
Technology	Advanced analog and digital SNN processing
Die Area	Not specified
Max # Neurons/Chip	Up to 1000 reservoir neurons
Max # Synapses/Chip	Not specified
Neuron Model	Digital LIF spiking neuron models
Power Consumption	Below 10 mW
Input Signal Bandwidth	100 Hz to 20 kHz
Operating Voltage (VDD)	1.1 V
IOVDD Voltage	2.5 V
Analog Front End	Capable of single-ended and differential modes
Output	Up to 8 binary event output channels
Memory Footprint	Approx. 150 KB
Operating Frequency	Up to 100 MHz
Startup Time	Within 0.5 seconds
Average Power	Approx. 350 µW

The Leaky Integrate and Fire (LIF) model [79-81] is widely used due to its effective balance between capturing essential neural processing dynamics and maintaining computational simplicity. Equation 1.1 describes the LIF neuron.

$$C_m \frac{dV_m}{dt} = \frac{C_m}{\tau_{RC}}(E_L - V_m) + A \times I_{app} \quad 1.1$$

$$if\ V_m > V_{th}\ then\ V_m = V_{reset}$$

$$\tau_{RC} = \frac{C_m}{G_L}$$

In the Leaky Integrate and Fire (LIF) neuron model, various parameters are used to characterize the neuron's behavior. These include the membrane capacitance (C_m), leak conductance (G_L), leak potential (E_L), membrane potential (V_m), input signal amplification (A), and the applied input current (I_{app}). Additionally, the membrane RC time constant (τ_{RC}) is a crucial factor in this model. A simpler variant of this model is the Integrate and Fire neuron model, which is essentially a LIF neuron but without the

decaying aspect of the membrane potential. This means that in this variation, the membrane RC time constant (τ_{RC}) is considered to be infinitely large.

1.2.2 Associative Learning

Associative learning [82], a fundamental process observed in various animals, hinges on the ability to connect separate events that occur close together in time. This concept, first explored by Ivan Pavlov in the 1890s [83], revolutionized our understanding of animal behavior and learning processes.

Pavlov's work, primarily with dogs, laid the foundation for this field [84]. He observed that dogs, which naturally salivate at the sight of food, began to salivate upon hearing a whistle if the sound was repeatedly paired with the presentation of food. This phenomenon demonstrated that the dogs had learned to associate the whistle, a neutral stimulus, with food, a natural stimulus. In the context of associative learning, the food represents an Unconditional Stimulus (US) that naturally elicits a response, while the whistle exemplifies a Conditional Stimulus (CS) that acquires the ability to trigger a response through association.

The principle of associative learning extends beyond Pavlov's dogs to a wide range of animals, each demonstrating unique examples of this learning process. For instance, sea slugs (Aplysia) exhibit associative learning through modifications in their neural pathways [85]. In Aplysia, a touch to the tail (an unconditional stimulus) naturally causes the gill to retract. However, if a simultaneous stimulus is applied to the siphon (a conditional stimulus), over time, the gill will retract with just the siphon stimulus, showing learned association.

In more complex animals, like rats, fear conditioning experiments have shown associative learning involving neural assemblies [86]. Rats can learn to associate a neutral sound with an electric shock. Initially, the sound (a neutral stimulus) does not provoke a fear response. However, when paired with an electric shock (an aversive stimulus), the rats learn to associate the sound with the shock and exhibit a fear response to the sound alone.

Mechanisms Behind Associative Learning

1. Signal Pathway Adjustment and Synaptic Plasticity [10, 87]: In neuroscience, associative learning is understood in terms of signal pathway adjustment and synaptic plasticity. Essentially, the brain learns to connect different stimuli through changes in how neurons communicate and form networks.
2. Neural Assemblies [88, 89]: In more complex brains, such as those of mammals, associative learning involves the integration and interaction of neural assemblies—groups of neurons that process and respond to different types of stimuli.

Advances in neuroscience have shed light on the brain processes underlying associative learning. This learning involves changes in the neural circuitry, known as synaptic

plasticity [90], where the strength of synaptic connections between neurons is altered. For example, in the case of Pavlov's dogs, the repeated pairing of the bell and food led to changes in the brain's neural pathways, making the sound of the bell alone sufficient to trigger salivation.

In more complex animals, associative learning involves the integration of information across different neural assemblies. For instance, fear conditioning in rats involves changes in the amygdala, a brain region critical for emotion processing [91], where sensory information about the conditioned and unconditioned stimuli converges and gets integrated.

Real-life examples of associative learning

1. Maze Learning in Rodents [92]: Rodents, like mice and rats, can learn to navigate mazes to find food. They associate certain paths or cues within the maze with the reward (food), demonstrating their ability to form complex spatial associations.
2. Birdsong Learning in Birds [93]: Many bird species learn their songs by associating environmental sounds with their own vocalizations. Young birds often memorize the songs of their parents or surrounding adults and, through a process of trial and error, learn to replicate these sounds. This associative learning is critical for communication and mating in the bird world.
3. Pet Training [94]: Domestic pets, like cats and dogs, learn various commands and behaviors through associative learning. For example, a dog may learn to sit when it hears the word "sit" if this command is consistently paired with a reward like a treat or affection.
4. Human Learning [95]: In humans, associative learning is a part of everyday life. For example, a person might associate the smell of a specific perfume or cologne with a particular individual. Over time, just smelling that scent can evoke memories or emotions related to that person. Similarly, hearing a song that was frequently played during a significant period in one's life can bring back vivid memories of that time.
5. Human Language Acquisition [96]: Humans, particularly in early childhood, exhibit associative learning through language development. Children learn to associate sounds (words) with objects, actions, or concepts, a process that forms the basis of language comprehension and usage.

In conclusion, associative learning is a complex and wide-reaching concept that plays a crucial role in the behavior and adaptation of various organisms. Its principles are evident in a myriad of real-life examples and have significant implications across different fields, from psychology and neuroscience to education and robotics. This learning process highlights the adaptability and sophistication of the brain in forming associations and adapting to the environment.

1.2.3 Mobile Robotics

Mobile robotics serves as an excellent platform for the implementation of neuromorphic computing, aligning well with the operational constraints and conditions typically

encountered. The standout advantage of neuromorphic computing in this field is its minimal demand on Size, Weight, and Power (SWaP), which is essential for the complex cognitive functions that robots need to perform. Given that mobile robots operate with limited energy reserves, the low power consumption of neuromorphic computing is particularly beneficial. Furthermore, in many use cases, these robots must operate autonomously without the support of remote servers, relying solely on their own computational resources, a practice commonly known as edge computing. This approach involves processing data at or near the source rather than depending on distant, more robust computing facilities. Neuromorphic computing is also advantageous in situations where there is a lack of preexisting data, which can be due to the high costs or impracticality of data collection, a common issue in areas like lunar or Martian terrain analysis for space exploration [15]. Traditional Deep Learning (DL) methods use a lot of power and rely on large datasets for training. Neuromorphic computing, on the other hand, is a better and more efficient way to train mobile robots, especially in places with little data and limited power. For our associative learning experiment, we have selected the Agilex LIMO UGV as a mobile robotics platform, as shown in Figure 1.2.

The AgileX Limo robot offers a significant advancement in the field of mobile robotics, particularly in the context of neuromorphic computing. Its compact and versatile design makes it an ideal platform for applications where Size, Weight, and Power (SWaP) are critical considerations. The AgileX Limo, with its multi-modal mobility, allows for different steering modes such as omni-directional, tracked, Ackermann, and four-wheel differential steering, enhancing its adaptability to various environments [97].

One of the key features of the AgileX Limo is its compatibility with the Robot Operating System (ROS), making it a suitable platform for both educational and research purposes in robotics. ROS enables the operation of various independent processes or nodes that carry out specific tasks using communication paradigms like publish-subscribe and client-service frameworks. In the publish-subscribe model, nodes can either publish or subscribe to data channels (topics), facilitating efficient data exchange among different components of the robot. The client-service model, on the other hand, involves a service node that performs specific functions upon requests from client nodes [98].

The AgileX Limo is equipped with an NVIDIA Jetson Nano, EAI XL2 LiDAR, ORBBEC® Dabai stereo depth camera, and a suite of other sensors, which together support advanced robotic applications. These applications include precise autonomous positioning, Simultaneous Localization and Mapping (SLAM), path planning and navigation, obstacle avoidance, and traffic light recognition. Because edge computing environments often don't have reliable communication with remote hosts, these features are very important for mobile robotics when they need to operate on their own.

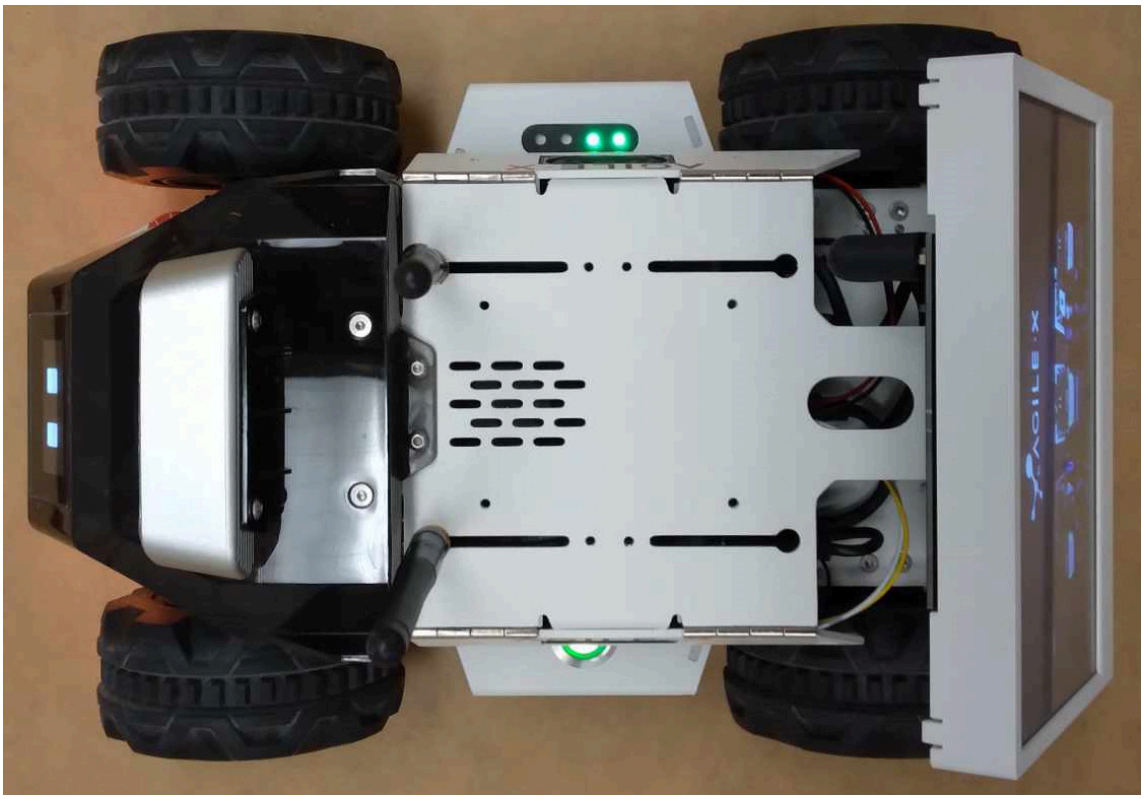

Figure 1:2: Agilex LIMO robot used for our neuromorphic experiment.

Furthermore, in situations with limited pre-existing data, such as in space exploration on Lunar or Martian terrains, the AgileX Limo's sensor suite and computational capabilities make it an excellent choice. Its ability to perform complex tasks autonomously using on-board resources without relying on large datasets for training sets it apart from traditional deep learning approaches in robotics.

Key features of the AgileX LIMO include:

1. **Robot Operating System (ROS) Compatibility**: The LIMO is designed for ROS development and learning, offering a versatile platform for robot education, research, and development. This compatibility is crucial for facilitating autonomous task execution without relying on remote computational resources.
2. **Multi-Modal Steering Modes**: The robot integrates four steering modes: omni-directional, tracked, Ackermann, and four-wheel differential. This flexibility allows the LIMO to adapt to various terrains and environments, essential for edge computing applications in challenging conditions, such as space exploration on Lunar or Martian terrains.
3. **Advanced Sensor Array**: Equipped with an array of sensors, including the EAI X2L LiDAR, ORBBEC® DaBai Stereo Depth Camera, and an Inertial Measurement Unit (IMU) comprising an accelerometer and gyroscope, the LIMO is adept at navigating

complex environments. These sensors enable functionalities like precise autonomous positioning, SLAM, path planning, obstacle avoidance, and traffic light recognition.

4. **Onboard Computing Power:** Powered by an NVIDIA Jetson Nano, the LIMO offers enough computational power for sophisticated robotic applications, supporting tasks like mapping, navigation, and computer vision.
5. **Energy Efficiency**: With a 12V Li-ion 5600mAh battery and efficient power management, the LIMO maintains the high energy efficiency necessary for mobile robots with limited power supplies.
6. **Gazebo Integration for Simulation**: The integration with the Gazebo simulation environment is an essential feature, allowing for the testing and development of robotic applications in simulated conditions.
7. **Open-Source Software and Customizability**: The LIMO's open-source nature and the availability of various ROS packages make it highly customizable and adaptable to different research and educational needs.

Given these features, the Agilex Limo Robot is an ideal platform for edge computing in mobile robotics. Its ability to perform autonomous tasks using onboard computational resources makes it a strong candidate for applications where data collection is challenging or impractical, such as space exploration or unstructured environments.

For experimentation and simulation, the Gazebo simulation environment can be integrated with ROS, offering a comprehensive platform for developing and testing robotic applications. This integration is particularly beneficial for educational and research purposes [99], allowing for a deeper understanding of robotic systems and their operation in simulated real-world scenarios.

In summary, the AgileX Limo robot, with its advanced features and ROS compatibility, stands out as a versatile and efficient platform for mobile robotics applications, particularly in neuromorphic computing and edge computing scenarios.

2 Designing Methods for Associative Learning

Our associative learning [100] system comprises two signal pathways dedicated to processing both audio and visual signals, as illustrated in Figure 2.1. This architecture enables the neuromorphic robot to simultaneously receive information from auditory and visual stimuli.

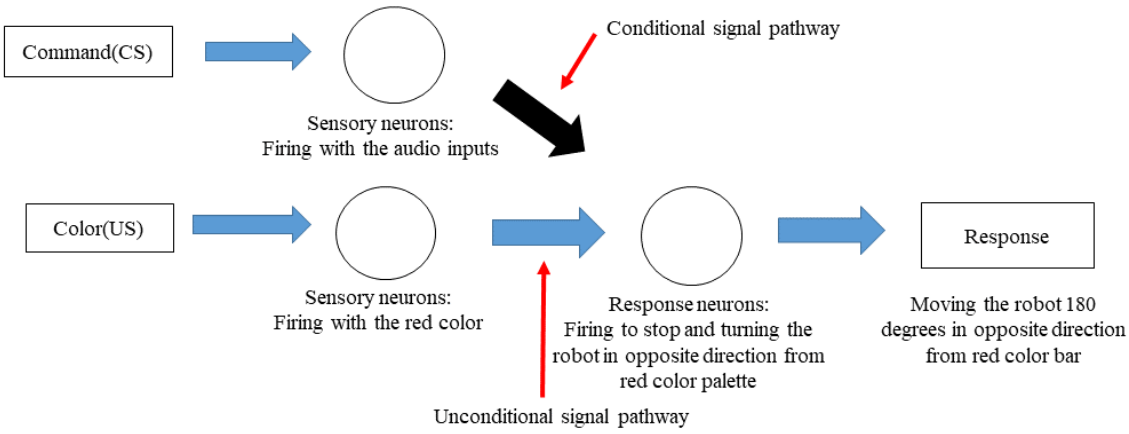

Figure 2:1: Overall associative learning implementation.

The auditory pathway is designed to capture and interpret audio cues, allowing the robot to discern and learn from sound-related stimuli within its environment. Meanwhile, the visual pathway focuses on processing visual information, enabling the robot to recognize and understand visual stimuli crucial for its associative learning. The neuromorphic robot's overall cognitive abilities are improved by connecting these two signal pathways. This helps researchers learn more about how audio and visual inputs affect each other.

2.1 Foundations of Associative Learning: Network Initialization and Learning Dynamics

The neural simulation begins with the creation of an associative learning network. This network serves as a container, encapsulating all the components we'll be using—the neurons, their interconnections, and various other features.

Two primary stimuli are introduced into this system: the Conditional Stimulus (CS) and the Unconditional Stimulus (US). These are like the triggers that set the simulation in motion. To visualize this, imagine trying to teach a dog to salivate at the sound of a bell. The bell's ring would be the CS, while the sight of food (which naturally makes the dog salivate) would be the US. Over time, the aim is for the dog to start associating the bell with the food, so it begins to salivate just at the sound of the bell [101, 102].

Within this memory network, we establish groups of neurons, referred to as 'ensembles'. Each ensemble represents a specific stimulus or a response. For our purposes, we're

mainly focusing on three ensembles: one for the CS (like the bell's sound), one for the US (like the sight of food), and another that might be seen as the resultant response (the dog's salivation).

These neurons have been modeled with specific properties in mind, such as the range of their firing rates and the values at which they start firing. Additionally, we've ensured that they are mostly positively oriented, meaning they are more likely to fire in response to positive stimuli.

The properties of these ensembles are crucial for determining how they respond to stimuli and how they interact with each other. We've tabulated the primary properties of the three ensemble neurons we have used in Table 2.1.

Table 2.1: Primary properties of the ensemble neurons.

Property	**CS Ensemble**	**US Ensemble**	**RS Ensemble**
Number of Neurons	1	1	1
Dimensions	1	1	1
Neuron Type	LIF	LIF	LIF
Time constant	0.02s	0.02s	0.02s
Refractory period	0.002s	0.002s	0.002s
Encoders	Evenly split between +1 and -1	Same as CS	Same as CS
Max Firing Rate	Uniformly distributed between 200 and 400 Hz	Same as CS	Same as CS
Intercepts	Between -1 and 0.9	Same as CS	Same as CS
Radius	1.0	1.0	1.0

For the system to work, these neurons and stimuli must be interconnected. The conditional and unconditional stimuli are connected directly to their respective neuron ensembles. These connections determine how information flows within the network.

A crucial part of associative learning is the "learning rule"—the mechanism that determines how our neurons adapt and change over time based on the stimuli they receive. In this case, we're employing a variant of the Hebbian learning rule known as Oja's learning rule [103]. Simply put, if two neurons fire together often, the connection between them is strengthened; if they don't, the connection weakens. It's the neural equivalent of the saying, "What fires together, wires together [104]."

Beyond just the stimuli and response, the system is also designed to recognize and learn the association between the conditional and unconditional stimuli directly. This is a pivotal aspect of associative learning, as the network must understand and internalize the relationship between these two stimuli to produce the desired response.

Once the network is set in motion, it's essential to monitor its behavior. This is done using 'probes', which can be visualized as diagnostic tools or sensors. They track various metrics within the network, such as neuron firing rates, learned connection weights, and the overall output. This data is crucial for understanding how well the network is learning and functioning.

After running the simulation, the system examines the output. If the unconditional stimulus is present and the network's output doesn't cross a certain threshold, it takes this as a sign that the association isn't strong enough. As a response, the network reinforces learning by repeatedly presenting the same stimuli. Conversely, if the output crosses a specific threshold, it triggers an emergency response, indicating a strong association has been made.

In essence, this entire setup is a simplified model of how associative learning might work in the brain. It tries to replicate the process where, over time and through repeated exposures, the brain learns to associate one stimulus with another and produce a specific response based on that association.

2.1.1 Mechanisms of Associative Learning: Formation and Strengthening of Neural Connections

The core of associative learning is the associations between the CS (audio commands) and the US (red color spikes), focusing on the weights of their connections and how they evolve over time.

At the heart of associative learning lie the connections, or "weights," between neuron ensembles. These weights determine how strongly one neuron influences another. In our model, the connection weights represent the strength of the association between the CS and the US. Initially, the connections have predefined weights. However, as the network is exposed to stimuli, these weights change according to the specific learning rules applied, aiming to capture and strengthen the association between the CS and the US.

Table 2.2 summarizes the initial connection weights and their evolution over time.

Table 2.2: Initial connection weights among ensemble neurons and their evolution over time.

Connection	Initial Weight	Adjustment Method	Expected Outcome
US input to US neuron	1	Static (doesn't change)	Remains strong throughout, representing the direct effect of US input.
CS input to CS neuron	1	Static (doesn't change)	Remains strong throughout, representing the direct effect of the CS input.
US neuron to RS neuron	1	Static (doesn't change)	Remains strong throughout, representing the direct effect of the US neuron.
CS neuron to RS neuron	0.001	Adjusted by Hebbian learning	This weight increases if CS neurons and RS neurons fire together frequently, strengthening the association.
CS neuron to US neuron	0.01	Adjusted by Hebbian learning	Enhances the learned association between the CS neuron and the US neuron over time.

In the beginning, the US naturally has a more direct and potent connection to the Response Stimulus (RS) with a weight of 1, representing the innate, unlearned response. In contrast, the CS has a minimal initial weight compared to the RS and the US neurons, signifying its weak influence.

However, when the CS and US are presented together, the learning mechanism (Hebbian learning) springs into action. The idea behind this learning is quite intuitive: if the CS and the US fire together frequently, the connection between them gets stronger. This is in line with the old adage in neuroscience: "Cells that fire together, wire together [105]." Over time, with repeated paired presentations of the CS and US, the weight between the CS and RS (and also between the CS and US neurons) increases. This signifies that the association between the two stimuli is getting stronger.

As the system undergoes several learning iterations, the association between CS and US becomes more robust. If the learning is successful, the CS alone can evoke a strong response, even in the absence of the US. The network has effectively learned to associate

the CS with the US, much like how a dog learns to associate the sound of a bell with forthcoming food.

In practical terms, when the CS presents a certain value, the output from the network (which was initially weak) becomes more pronounced, reflecting the strengthened association. If this output crosses a specific threshold, it might even trigger further actions or reactions, as encoded in the system.

This dynamic change in weights and the resultant strengthening of associations is the crux of associative learning [106], modeling how we, and many other organisms, learn from associations in our environment.

2.1.2 Enhancing Associative Learning through Hebbian and Oja's Rules

Hebbian learning [107, 108] is a type of associative learning rule based on the principle that if two neurons activate together, the strength of the connections between them should increase. In simpler terms: "Neurons that fire together, wire together."

The basic idea behind Hebbian learning is that if neuron A frequently activates just before neuron B, then the synapse from A to B should be strengthened. Mathematically, it is possible to calculate the change in synaptic weight (Δw) using Equation 2.1.

$$\Delta w = \eta \times (pre \times post) \quad 2.1$$

Where:

- η is the learning rate.
- *pre* is the firing rate of the presynaptic neuron.
- *post* is the firing rate of the postsynaptic neuron.

Oja's rule [103] is an extension of the basic Hebbian learning rule that includes a normalization term. This ensures that the synaptic weights do not grow indefinitely. The Oja's weight update rule is as follows, as in Equation 2.2.

$$\Delta w = \eta \times (pre \times post - \beta \times w \times post^2) \quad 2.2$$

Where:

- η is the learning rate.
- *pre* is the firing rate of the presynaptic neuron.
- *post* is the firing rate of the postsynaptic neuron.
- β is a constant that determines the strength of the normalization.

- w is the current synaptic weight.

The term $\beta \times w \times post^2$ serves to ensure that weights don't grow without bound. It acts as a subtractive normalization, scaling the weight changes depending on the size of the weight and the postsynaptic activity.

For our work, Hebbian learning is applied to the connections between certain neuron ensembles. Specifically, it's used for two crucial connections:

1. CS Neuron to RS Neuron
2. CS Neuron to US Neuron

For these connections, the Oja learning rule, which is a variation of the Hebbian rule, is utilized. The Oja rule strengthens connections when pre- and post-neurons fire together and ensures that weights do not grow indefinitely. Table 2.3 summarizes the Hebbian learning mechanism for our work.

Table 2.3: Hebbian learning rules employed in our work.

Connection	Initial Weight	Learning Rule	Learning Rate	Beta	Expected Change
CS to RS neurons	0.001	Oja (Hebbian variant)	6e-2	0.1	Weight increases as the CS neuron and the RS neuron activate concurrently.
CS to US neurons	0.01	Oja (Hebbian variant)	6e-2	0.1	Weight strengthens with simultaneous activations of CS neurons and US neurons.

When the CS is presented in conjunction with the US, both the CS neurons and the US/RS neurons are activated. The Oja learning rule adjusts the weights of the connections based on the degree to which these neuron groups are activated together. Over repeated pairings:

1. The weight between the CS neurons and RS neurons grows, leading to a stronger response in the RS neurons even when only the CS is presented.
2. Similarly, the bond between the CS and US neurons gets enhanced, reinforcing the learned association between the CS and US.

To sum up, the network improves its weights using Hebbian learning and the Oja rule to show the connection between the CS and US. Over time, as learning progresses, the CS alone can evoke responses reminiscent of those initially produced only in the presence of both the CS and US.

2.2 Acquiring the Conditional Stimulus for Enhancing Associative Learning

A conditional stimulus, or CS for short, is a cue or signal that is initially neutral but becomes linked with a certain outcome or reaction after being paired with an US repeatedly. Over time, the presence of this CS alone can trigger a learned response. For instance, if the presentation of food comes right after a bell rings, a dog may start salivating just from hearing the bell [109].

In simpler terms, a CS is like a hint or sign that points towards an expected event or result based on past experiences. Through repeated pairing, our brains come to associate this "hint" with a particular outcome, even if the outcome doesn't always follow.

2.2.1 Acquiring Audio Command Data as the Conditional Stimulus

2.2.1.1 Designing SNN for Audio Command Prediction

We designed a SNN using the **Rockpool** library to predict “left” and “right” audio command data. An input-to-hidden stage and a hidden-to-output stage make up the two main stages of the SNN's overall architecture, which is sequential. The audio perception neural network architecture is shown in Figure 2.2 (a). The auditory data exhibited will be imported into 16 channels to align with the capabilities of the Xylo chip using a microphone, as shown in Figure 2.2 (b). In our work, we deliberately restricted our analysis to a subset of commands, specifically “left” and “right”. Transforming the events into frames reorganized the auditory data. The Xylo neuromorphic chip used for processing audio perception is illustrated in Figure 2.2 (c).

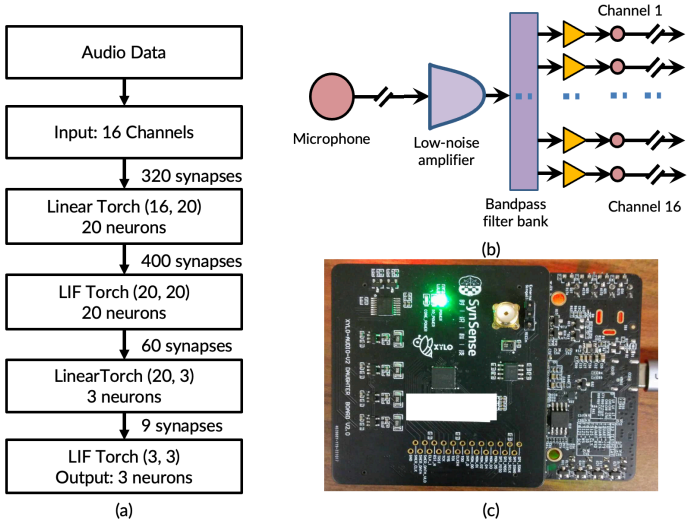

Figure 2:2: (a) Audio perception neural network architecture. (b) The preprocessing module of the Xylo neuromorphic chip. (c) XyloA2TestBoard for audio command detector model deployment.

The network is designed to accept input with a dimensionality of 16 channels. This is further processed through a hidden layer composed of 20 neurons. The output from this hidden layer is then passed through the final stage, resulting in an output with a dimensionality of 3 units. The network simulations utilize a timestep (**dt**) of 10 milliseconds.

The network layering and information flow are as follows:

Input Layer: Data enters the network via an input layer defined by its 16-channel structure.

Linear Transformation to Hidden Layer: Following the input, the data is subjected to a linear transformation designed to map the 16-dimensional input to a 20-dimensional hidden layer.

Hidden Layer with LIF Dynamics: This transformed data then encounters the hidden layer, which has 20 neurons. These neurons, governed by LIF dynamics, process the data, adding a layer of non-linearity and temporal dynamics to the transformation.

Linear Transformation to Output Layer: Post-processing in the hidden layer, the data is again linearly transformed, this time targeting the final output layer with a dimensionality of three.

Output Layer with LIF Dynamics: Analogous to the hidden layer, the output layer's three neurons utilize LIF dynamics to produce the final output, adding another layer of non-linear temporal processing.

In both the hidden and output layers, neurons are based on the Leaky Integrate-and-Fire (LIF) dynamics, which is a well-known way to model how biological neurons fire spikes. There are several defining parameters that determine the characteristics and behavior of these LIF neurons:

Membrane Time Constant: Set to 80 milliseconds, this parameter gauges the rate of decay of the neuron's membrane potential towards its baseline state when devoid of external inputs.

Synaptic Time Constant: Configured at 50 milliseconds, it quantifies the duration over which an incoming synaptic transmission affects the receiving neuron's membrane potential.

Bias: A consistent bias of 0.2 is imparted to every neuron, influencing its propensity to fire or spike upon receiving synaptic input.

Threshold: The neurons are calibrated with a spiking threshold set at a membrane potential value of 1. Once this threshold is surpassed due to synaptic inputs, the neuron responds by generating and dispatching a spike.

2.2.1.2 SSC Dataset for Audio Command Detection

We used the Spiking Speech Command (SSC) dataset [110] to train our spiking neural network. This dataset, found in the tonic library, gives us audio commands in a unique format known as "spikes.". The SSC dataset is made up of tiny bits of data recorded very fast—one every microsecond. Every audio command in this dataset is split into 700 channels. In total, there are 35 different audio commands. For our work, we decided to only focus on two commands: "left" and "right." These are located at positions 22 and 34 in the dataset. By doing this, we wanted to teach our model to tell the difference between these two commands.

We made a plot to show when and where the spikes for the "left" command happen. On this plot, the spikes look like small vertical magenta lines. Time is shown from left to right, and the 700 channels go from bottom to top. Figure 2.3 shows spike events for a sample left audio command data.

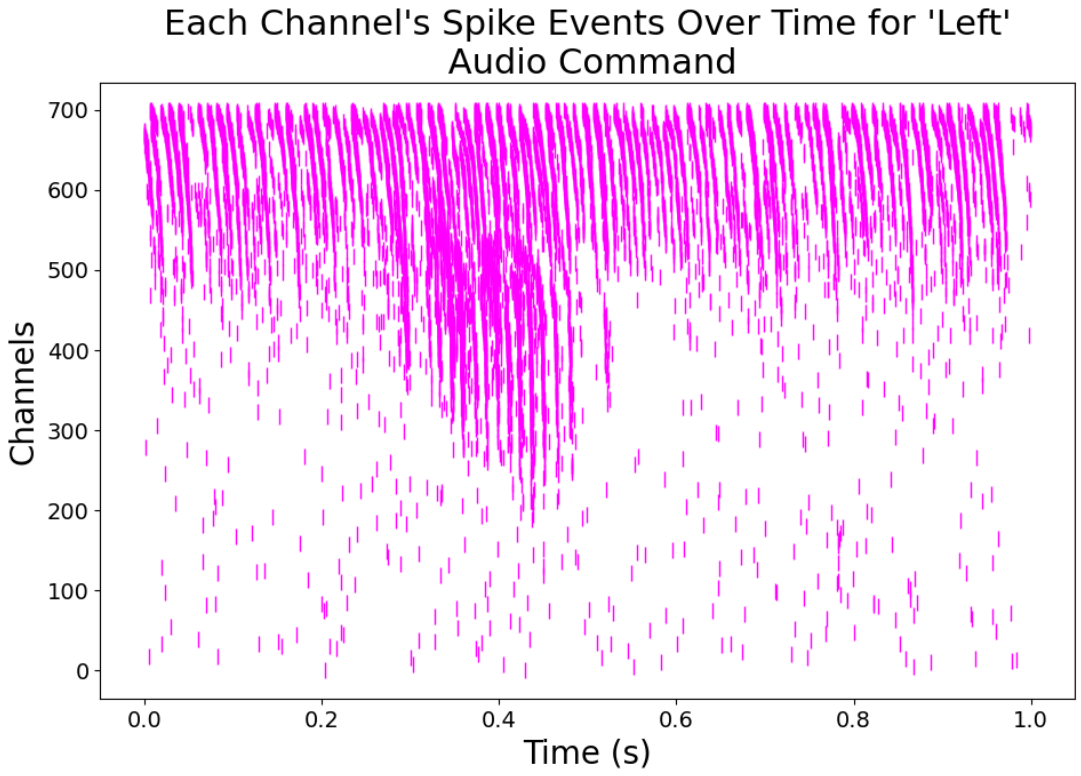

Figure 2:3: Spike events for left audio command data.

We made another plot to show when and where the spikes for the "right" command happen. On this plot, the spikes look like small vertical magenta lines. Time is shown from left to right, and the 700 channels go from bottom to top. Figure 2.4 shows spike events for a sample of right audio command data.

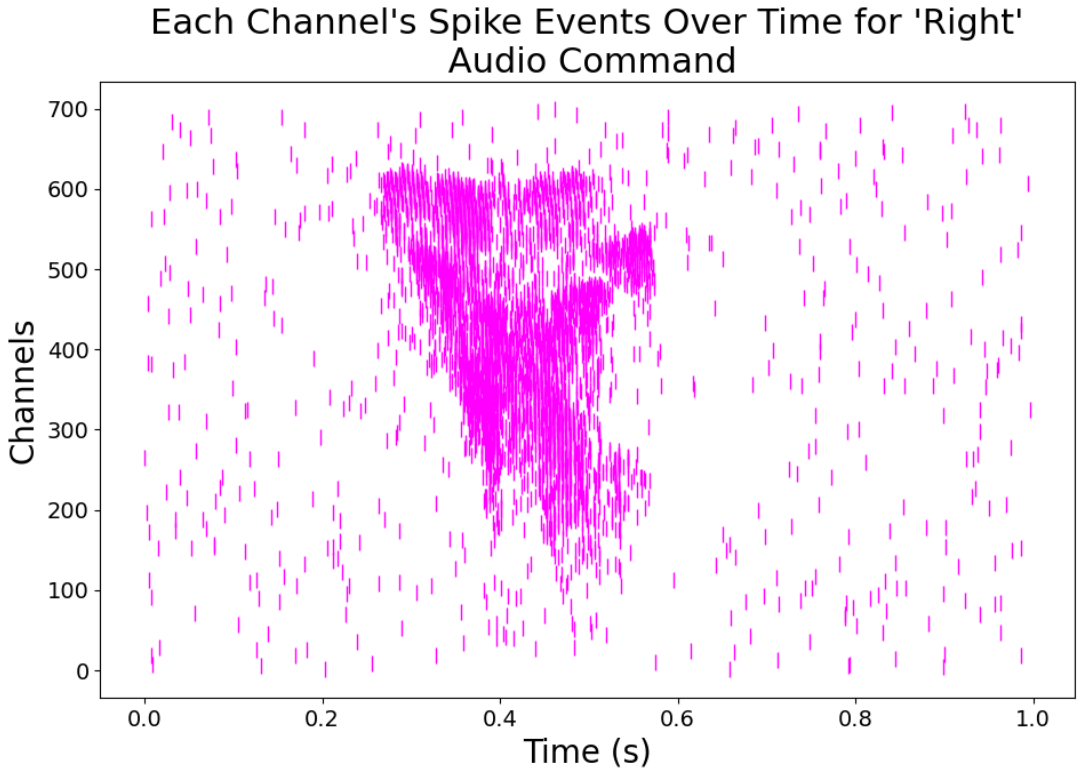

Figure 2:4: Spike events for the right audio command data.

2.2.1.3 Data Preparation and Transformation for Deployment on the Xylo Neuromorphic Chip

To deploy our trained network on the Xylo neuromorphic chip, we need to adjust or "transform" the data so that it fits perfectly. The transformation process tailored the dataset to be in line with the chip's requirements. The Xylo chip has specific constraints. It can process data with a maximum of 16 channels. To accommodate this, we undertook a series of transformations to mold the dataset to meet these prerequisites.

Downsampling: Given the chip's constraints, our first task was to reduce the complexity of the data. This involved:

Temporal Downsampling: The original dataset had a microsecond-level temporal granularity, with each timestep being 1*e-6 seconds. For the transformation, we aimed to alter the timestep to 10*e-3 seconds. The ratio for this temporal down-sampling was determined by dividing the desired timestep (10*e-3) by the original timestep (1*e-6).

Spatial Downsampling: We were working with a dataset that had 700 channels. But for the Xylo chip, we needed to compress this to 16 channels. By dividing the desired channels (16) by the original channels (700), the spatial down-sampling ratio was thus determined.

Rasterizing the Events: Once down-sampling was achieved, we reformatted the dataset by converting the events into "frames." Think of this as turning a film reel into individual photographs, giving us static snapshots of the data across time.

Data Conversion to Tensors: For seamless integration with our model, the data was transformed into a tensor format. Picture a tensor as a multi-layered grid, storing data across several dimensions.

Duration Adjustment: Our final transformation ensured the data's time length didn't exceed 250 seconds, trimming any excess.

After these transformations, we displayed the "left" audio command's spikes on a graph. Each spike appears as a magenta line. This visualization helps us easily see the frequency and timing of the spikes after we transformed the data. Figure 2.5 shows the spike events for the down-sampled left audio command data.

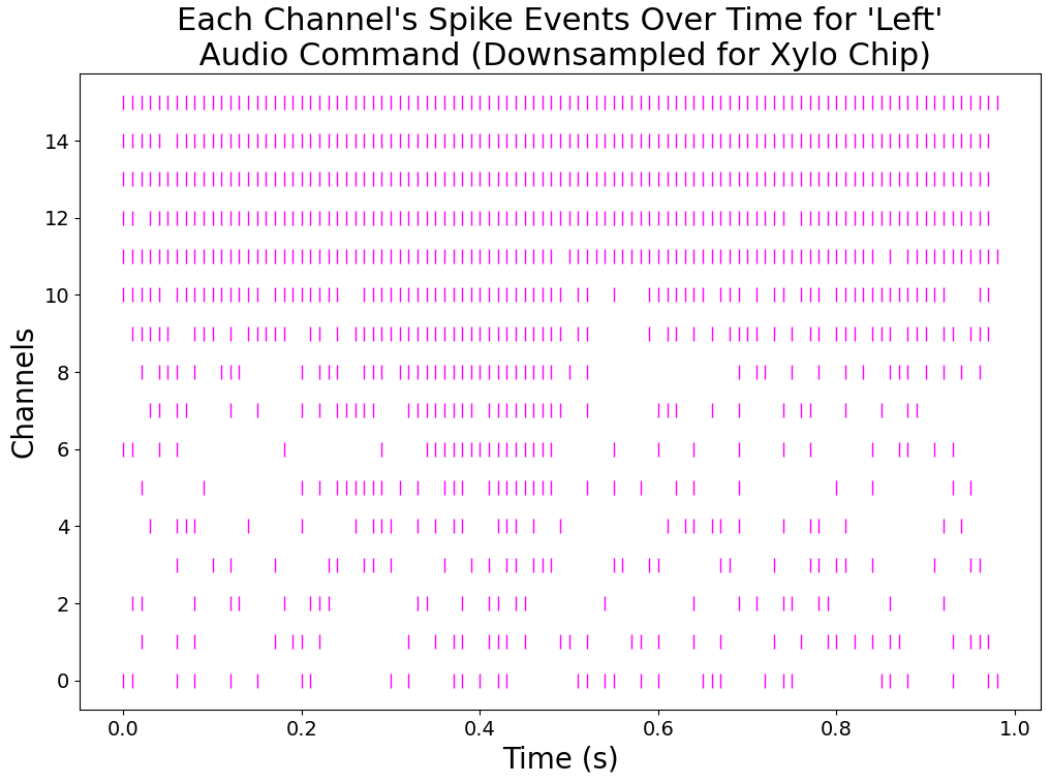

Figure 2:5: Spike events for a down-sampled left audio command data.

We also made a plot for the "right" audio command. This visual representation helps us compare the transformed data of both commands before we used it on the Xylo chip.

Figure 2.6 shows the spike events for the down-sampled right audio command data.

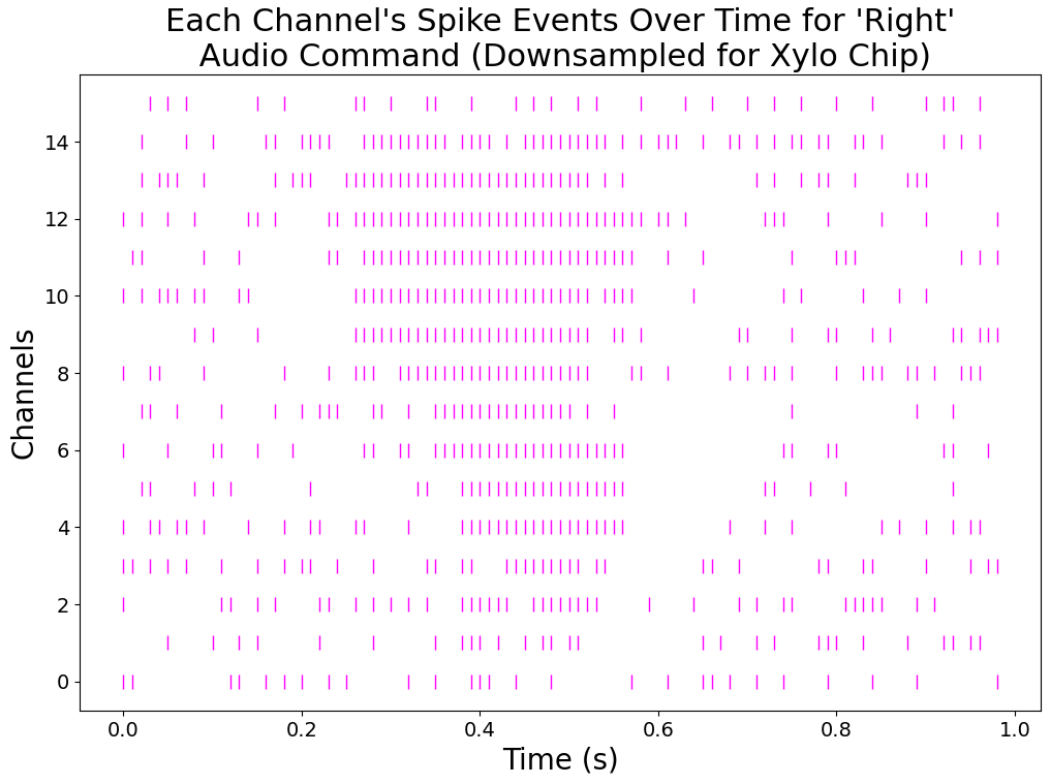

Figure 2:6: Spike events for a down-sampled right audio command data.

2.2.1.4 Training the Spiking Neural Network Model

We trained our Spiking Neural Network for 17,500 epochs. The Adam optimizer with a learning rate of 1*e-5 guided the learning process, and we used the cross-entropy loss function to measure errors.

At the beginning, the model could only correctly identify "left" and "right" audio commands 27% of the time, and the error was measured at 1.87e+04. As the training progressed, the model improved significantly. By the end, it achieved an impressive 92% accuracy in distinguishing between the two commands, and the error was reduced drastically to 6.39e-01.

For clarity, we plotted the model's performance throughout its training. The plot shows how its accuracy increased and error decreased over the 17,500 epochs.

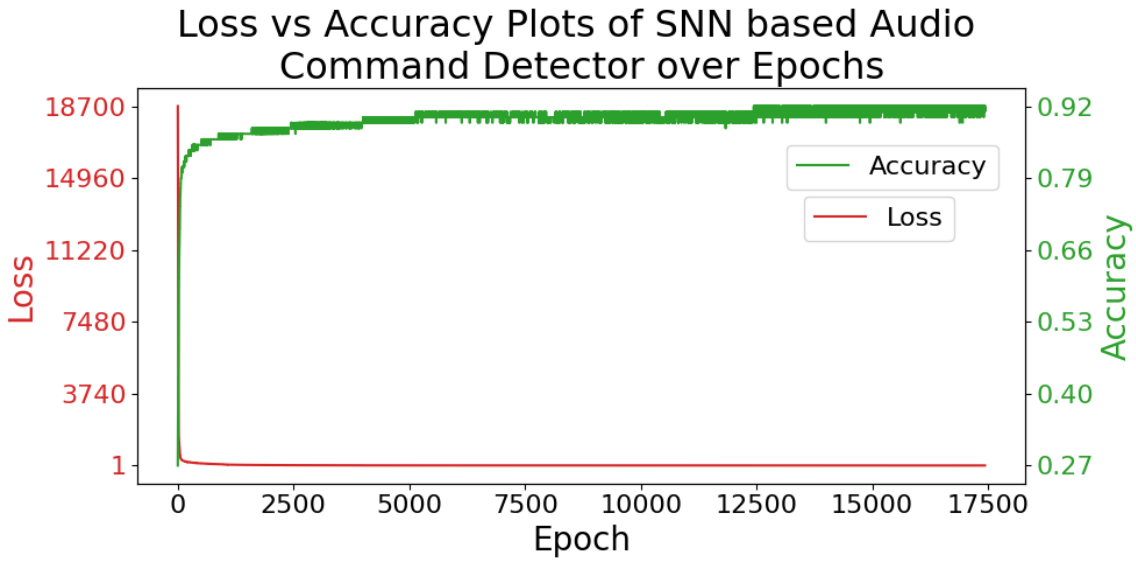

Figure 2:7: Loss vs. accuracy plot of the SNN-based audio command detector over epochs.

2.2.1.5 Model Deployment and Visualization on XYLO Platforms

After training, we tested our model on the Xylo Hardware Development Kit (Xylo HDK) and the Xylo Simulator using sample "left," "right" audio commands, and "no input" data.

We ran the model on the Xylo HDK with the "left" and "right" commands and "no input" data. The resulting visualization shows the model's reactions as spikes, with a clear mark indicating when the "left" and "right" commands were played. Similarly, the model was tested on the Xylo Simulator. The visual output was again shown, with spikes representing the model's response.

Both tests let us compare the model's performance on actual hardware versus a simulated environment. The visualizations help us understand how our model responds to the "left" and "right" commands and “no input" data in different settings. Figure 2.8, Figure 2.9, and Figure 2.10 show the Xylo HDK and simulator outputs for "left," “right” command data, and “no input," respectively.

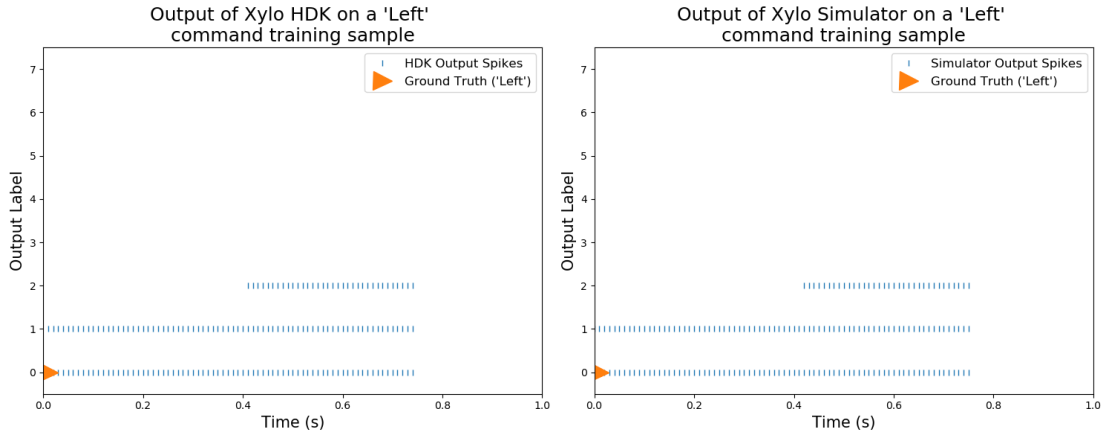

Figure 2:8: Output of a Xylo HDK and Simulator for “left” command data.

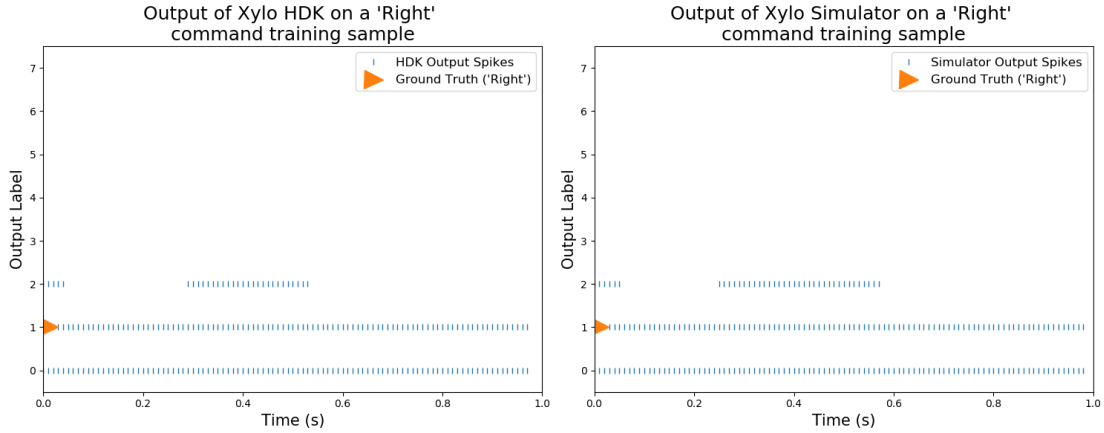

Figure 2:9: Output of a Xylo HDK and Simulator for “right” command data.

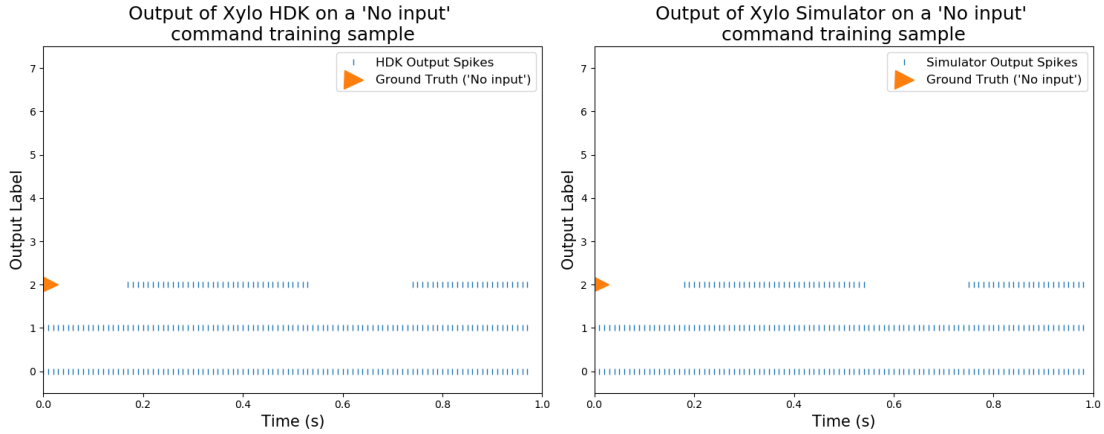

Figure 2:10: Output of a Xylo HDK and Simulator for “no input”.

2.2.1.6 Visualization of Output Membrane Potentials and Synaptic Current on Xylo Platforms

After testing the trained model, we wanted to gain deeper insights into its behavior. Specifically, we aimed to understand how the model reacts internally to different audio commands: "left," "right," and the absence of any command ("no input").

Parallel Visualization: Using side-by-side plots, we showcased the model's reactions on two platforms:

Xylo HDK: The left plot shows the neuron's membrane potential and synaptic current variations on the Xylo HDK when exposed to the different audio commands.

Xylo Simulator: The right plot does the same, but for the Xylo Simulator environment.

The side-by-side plotting offers a clear comparison between how the model operates in a real hardware setting and a simulated one. This visualization aids in predicting the robot's decisions, especially when the model runs live on the Xylo HDK.

Figure 2.11 and Figure 2.12 show the output membrane potentials and output synaptic currents, respectively, for Xylo HDK and Simulator.

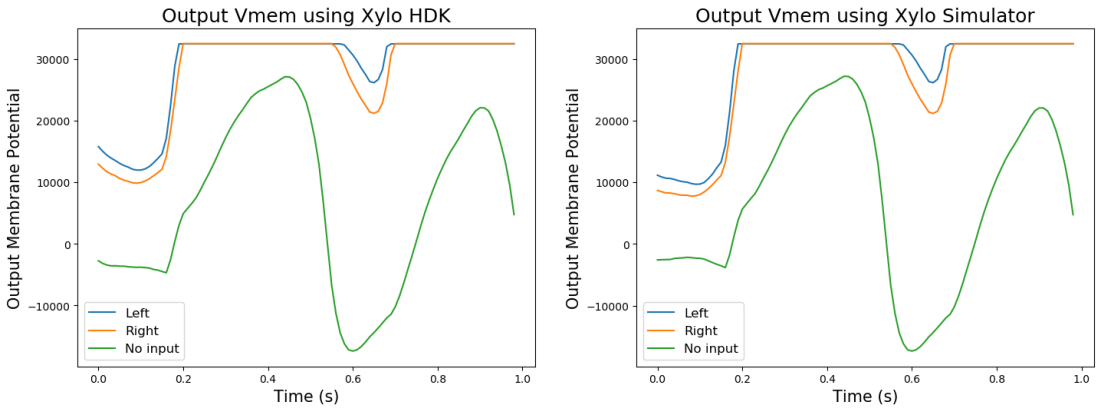

Figure 2:11: Output membrane potentials for Xylo HDK and Simulator.

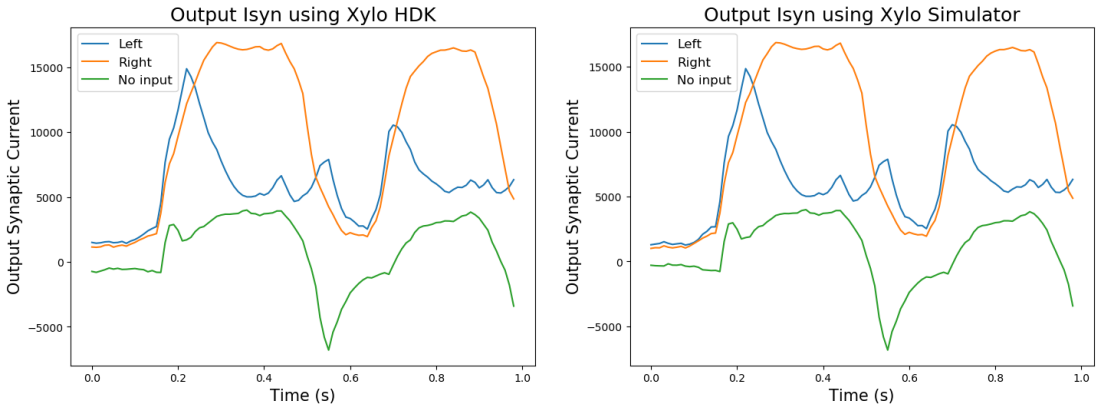

Figure 2:12: Output synaptic currents for Xylo HDK and simulator.

Furthermore, we also plotted the 20 hidden neurons' spikes for both Xylo HDK and Simulator in Figure 2.13. On both platforms, neuronal behavior looks similar.

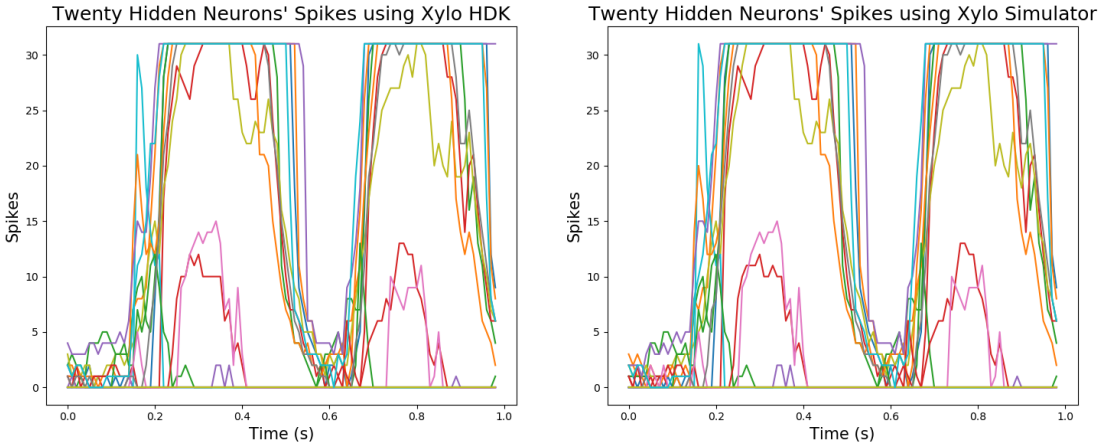

Figure 2:13: Hidden neurons' spikes for Xylo HDK and Simulator.

2.2.1.7 Energy Usage in the Xylo HDK While Running Spiking Neural Networks

In the deployment phase of our trained spiking neural network on the Xylo HDK, power consumption becomes a critical metric to gauge the efficiency of the network. The Xylo HDK provides real-time power measurements, allowing for detailed insights into the energy demands of the model when processing the "left" and "right" audio commands. As tabulated in Table 2.4, the most significant consumption comes from the logic component, drawing 1844.26 µW. The IO components, both direct and associated with the Analog Front End (AFE), have consumption values of 213.94 µW and 230.98 µW, respectively. Notably, the Logic AFE remains highly efficient with a mere 17.96 µW. These metrics offer a comprehensive view of the power demands, underscoring the deployment viability and operational efficiency of the neural network on the Xylo HDK platform.

Table 2.4: Power consumption in Xylo HDK during the deployment of the spiking neural network.

Component	Power Consumption (µW)
IO	213.94
Logic AFE	17.96
IO AFE	230.98
Logic	1844.26

Notably, when compared to traditional deep learning architectures and platforms, the Xylo HDK demonstrates significant power savings. According to Jouppi et al. (2017) [111], Tensor Processing Units (TPUs) used for deep learning tasks in data centers usually use a lot of power—watts or more—which is a lot more than the Xylo HDK's microwatt-level use. Esser et al. (2016) [112] have showcased the energy efficiency of neuromorphic computing with convolutional networks, emphasizing the promise of neuromorphic chips in reducing power consumption. Horowitz (2014) [113] highlighted the growing energy challenges in computing, especially as it relates to intensive tasks like deep learning, accentuating the need for more energy-efficient solutions like neuromorphic platforms. These measurements and comparisons show that spiking neural networks can be deployed and work efficiently on neuromorphic platforms. This makes it possible for AI to be used in more long-lasting and effective ways.

2.2.1.8 Real-time Robot Navigation via Audio Commands with XyloMonitor

To enable real-time robot navigation based on audio commands, the XyloMonitor deployment tool was utilized. This tool processes live audio inputs, specifically commands for "left" and "right" directions, and predicts the intended direction for the robot.

The XyloMonitor is meticulously set up to capture and process audio commands. A brief waiting period is necessitated for the AFE auto-calibration, ensuring optimal and consistent audio data acquisition.

To account for variations in command delivery, such as differing volumes or distances from the microphone, the sensitivity level of the audio capture system is heightened. This ensures that the commands are recognized clearly and accurately.

Once activated, the system processes incoming audio over a designated duration (T = 60s). Depending on the neuron's membrane potential, indicative of the received audio, it identifies the intended direction. A "left" membrane potential prompts the robot to turn left, and similarly, a "right" potential signals a right turn.

Post-processing, the definitive direction, either "left" or "right," is predicted and conveyed to the robot, guiding its subsequent movement.

2.2.1.9 Real-time Navigation Decision Making for the LIMO Robot

Upon receiving a "left" or "right" command prediction from the XyloMonitor, an auxiliary neural network is employed to drive the LIMO robot's leftward or rightward movements. This network, consisting of ten Leaky Integrate-and-Fire (LIF) neurons, is designed such that it predictably steers the robot left or right.

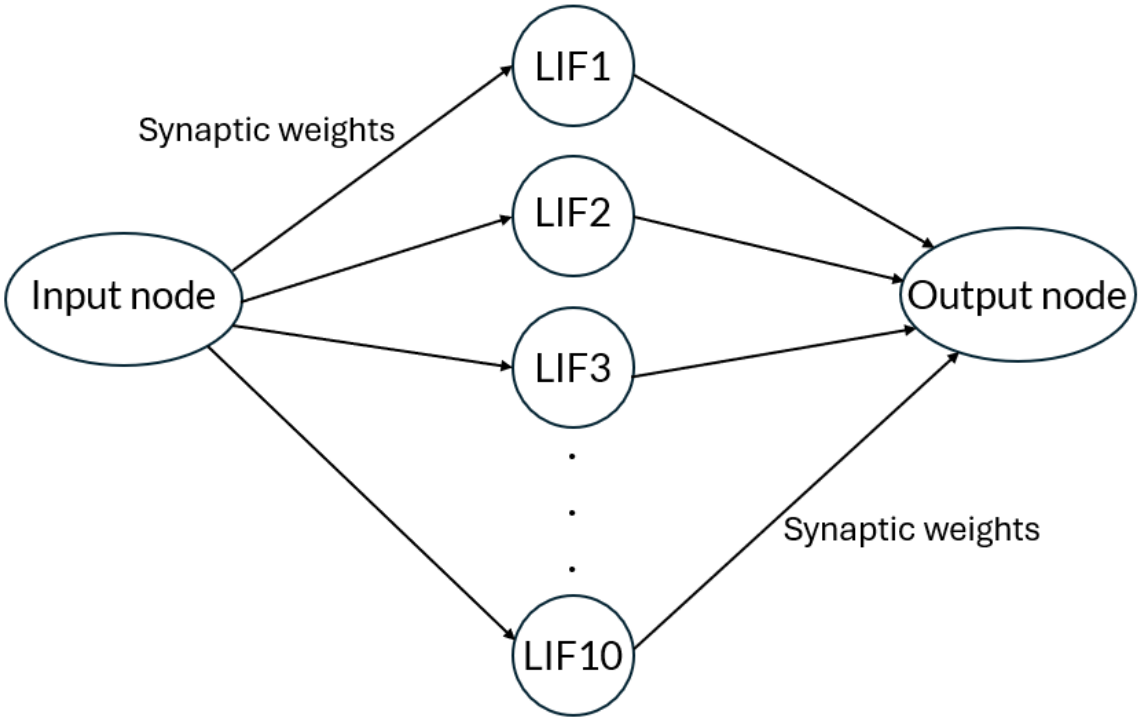

Figure 2:14: Auxiliary neural network with ten hidden LIF neurons directing the LIMO robot's movement upon receiving XyloMonitor commands.

The LIF neuron is characterized by two parameters: the membrane time constant and the refractory period. The refractory period has been minimized to accelerate neuron firing after an output spike. For consistency, the network uses a seed value of 42 to ensure reproducibility in neuron properties.

To ensure the LIMO robot consistently turns left or right, predefined synaptic weights are established between the input and neurons and between neurons and the output.

The network receives a constant input, which, when passed through the neurons with the given synaptic weights, produces a specific firing pattern. This pattern translates into a mean firing probability, which determines the robot's movement direction.

If the firing probability exceeds a threshold of 0.5, it prompts the LIMO robot to turn left; otherwise, turn right.

To ensure the LIMO robot consistently turns left with a mean firing probability of 0.9, the synaptic weights and neuron properties are fixed, as shown in Table 2.5, Table 2.6, and Table 2.7. Figure 2:15 shows the firing probability of LIF neurons over time for the left turn.

Table 2.5: Synaptic weights from input to neurons for the left turn.

Neuron Index	Weight Value
1	45.0
2	-27.0
3	72.0
4	-9.0
5	63.0
6	-54.0
7	18.0
8	81.0
9	-63.0
10	9.0

Table 2.6: Synaptic weights from neurons to output for the left turn.

Neuron Index	Weight Value
1	9.0
2	-18.0
3	27.0
4	-36.0
5	45.0
6	54.0
7	-63.0
8	72.0
9	-81.0
10	9.0

Table 2.7: LIF neuron properties for left turn.

Parameter	Value
Membrane time constant	0.02
Refractory period	0.001
Seed	42

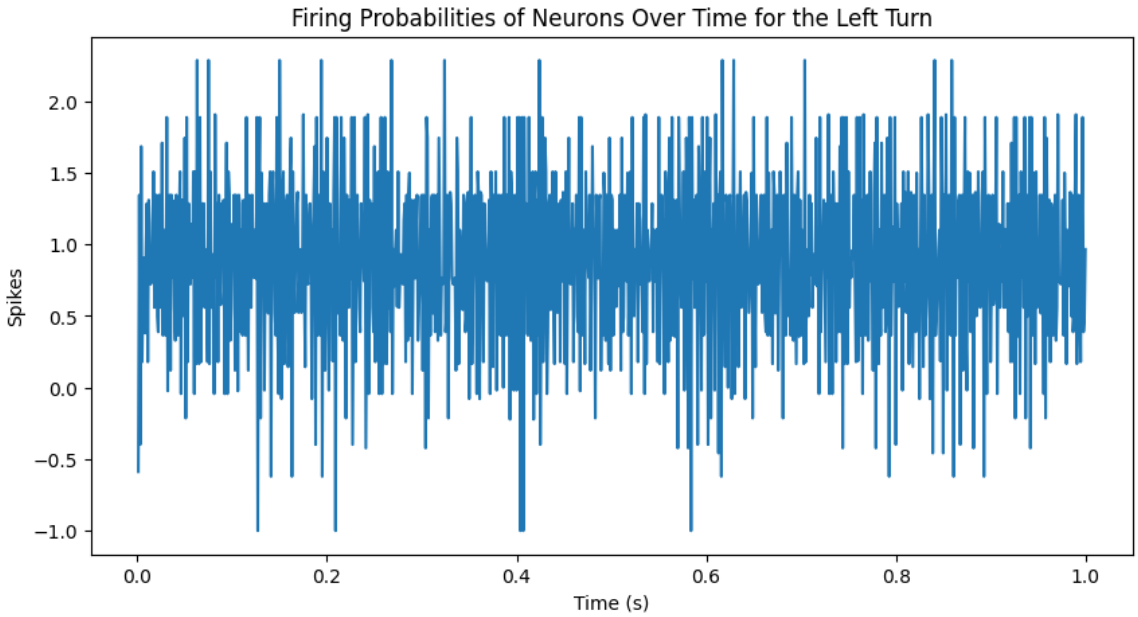

Figure 2:15: Firing probabilities of neurons over time for the left turn move of the LIMO robot in the T-maze.

To ensure the LIMO robot consistently turns right with a mean firing probability of 0.2, the synaptic weights and neuron properties are fixed, as shown in Table 2.8, Table 2.9, and Table 2.10. Figure 2:16 shows the firing probability of LIF neurons over time for the right turn.

Table 2.8: Synaptic weights from input to neurons for the right turn.

Neuron Index	Weight Value
1	2.9
2	-0.5
3	3.7
4	-0.3
5	3.0
6	-0.6
7	0.8
8	3.8
9	-0.7
10	2.8

Table 2.9: Synaptic weights from neurons to output for the right turn.

Neuron Index	Weight Value
1	0.9
2	-0.4
3	0.8
4	-0.6
5	0.7
6	0.8
7	-0.9
8	0.7
9	-1.1
10	0.6

Table 2.10: LIF neuron properties for the right turn.

Parameter	Value
Membrane time constant	0.02
Refractory period	0.002
Seed	42

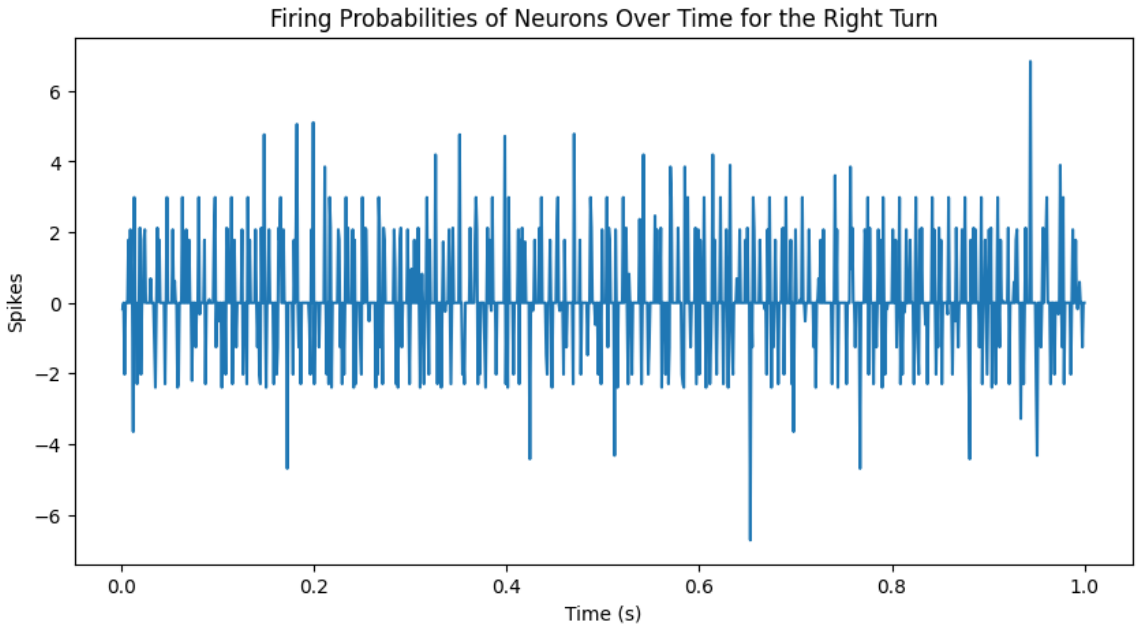

Figure 2:16: Firing probabilities of neurons over time for the right turn move of the LIMO robot in the T-maze.

2.3 Acquiring the Unconditional Stimulus for Enhancing Associative Learning

In associative learning, the US refers to a stimulus that naturally and automatically triggers a particular response without prior learning. Essentially, it's something that our bodies or minds react to instinctively.

For example, in the classic experiment by Ivan Pavlov, when a dog hears the sound of food being prepared, it might salivate. In this case, the food (or the presentation of the food) serves as the unconditional stimulus because it elicits an automatic, or "unconditioned," response (salivation) without the dog having to learn anything about it [109].

2.3.1 Acquiring Red Color as the Unconditional Stimulus

We used spikes of red as an unconditional stimulus for our work. We placed a red color curtain in front of the right arm of the T-maze to detect red color spikes from the live captured images using the LIMO front-end camera.

2.3.1.1 Real-Time Image Acquisition Using the LIMO Camera on the ROS Platform

Our methodology capitalizes on the LIMO camera's real-time imaging capabilities integrated with the Robot Operating System (ROS). Figure 2:17 shows the captured image using the LIMO robot ORBBEC® DaBai Stereo Depth Camera.

Figure 2:17: Captured image using the LIMO robot front camera.

We connect to the LIMO camera by tapping into its continuous raw image stream within ROS, ensuring an uninterrupted flow of visual data in a standard robotic format. To refine the received images for color detection, they are transitioned to a format compatible with the OpenCV image processing library. This conversion facilitates intricate color analysis, enhancing our system's ability to detect and interpret visual nuances.

2.3.1.2 Defining the Central Region of the Captured Image

To isolate and analyze the central region of the captured image, we adopt the following mathematical approach:

Let:

H represents the total height of the image.

W represents the total width of the image.

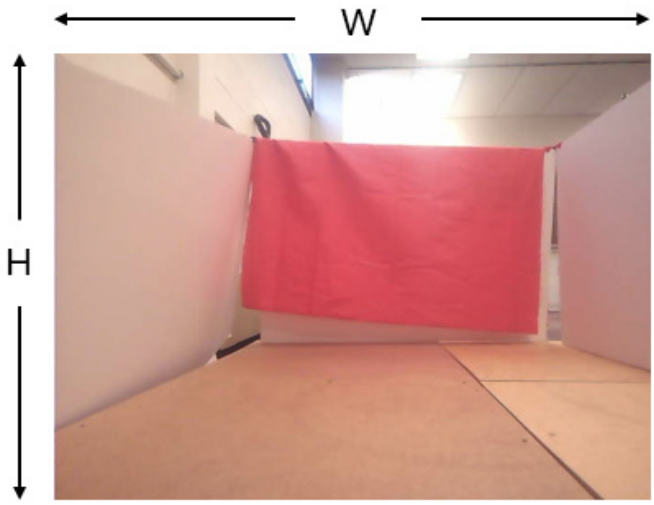

Figure 2:18: Central region calculation of the captured image.

For the height:

Starting y-coordinate: $y_1 = \frac{H}{4}$

Ending y-coordinate: $y_2 = \frac{3H}{4}$

Height of the central region: $\Delta y = y_2 - y_1 = \frac{H}{2}$

For the width:

Starting x-coordinate: $x_1 = \frac{W}{4}$

Ending x-coordinate: $x_2 = \frac{3W}{4}$

Width of the central region: $\Delta x = x_2 - x_1 = \frac{W}{2}$

From these calculations, it's evident that the targeted central region encompasses half of the image's height and half of its width. Graphically, this equates to a centered rectangle that spans 50% of the image's height and width. Consequently, this central region occupies 25% of the image's total area in the middle.

2.3.1.3 Color Detection Using HSV Color Space

To facilitate the precise detection of colors in images, we employ the Hue Saturation Value (HSV) color space. This choice stems from HSV's robustness against lighting variations, as it distinctly segregates color information (hue) from luminance (value).

- **Hue (H)** represents the type of color.
- **Saturation (S)** denotes the intensity or purity of the color.
- **Value (V)** indicates brightness.

After the image's conversion to HSV, we generate distinct masks for the red color. The defined range for the red color in the HSV space is detailed in Table 2.11.

Table 2.11: Ranges of red color in the HSV space.

Color	**Hue (H)**	**Saturation (S)**	**Value (V)**
Red1	0-10	100-255	100-255
Red2	160-180	100-255	100-255

The range for red is divided into two intervals (Red1 and Red2) to capture its entirety in the HSV circle.

2.3.1.4 Transforming Pixel Counts into Neural Spikes for Advanced Color Detection

The process starts with counting the number of red pixels centered in the image. This involves computing the sum of pixels that are specifically red within the delineated central region.

After the extraction of pixel counts for red hues, this data serves as an input to a Nengo neural framework. Nengo operates based on ensembles, collections of neurons designed to collaboratively represent data. In our setup, an ensemble of 100 neurons is employed. This ensemble is tasked with representing the counts of detected red pixels.

One of Nengo's distinguishing features is its reliance on spikes for neural communication. One way to think of a spike is as a binary signal that a neuron emits. As the input (in our case, the red pixel count) changes, so do the spiking patterns of the associated ensemble. This pattern provides a nuanced understanding of the color's intensity and presence in the given data.

With the input integrated, a single timestep in the Nengo simulation is executed. This step ensures that the input data circulates through the network, leading the ensemble's neurons to produce a distinctive pattern of spikes.

The subsequent stage involves scrutinizing this spiking activity. Using specialized probes within the ensembles, the spikes are recorded and totaled for the ensemble. This numerical representation, such as a threshold of 80 spikes, then guides actionable decisions, indicating dominant color presences and prompting specific responses.

In summary, at the beginning, we detect the number of red pixels in the center of an image, use those counts as inputs to a spiking neural network, and then observe the neuron spikes as outputs.

Here's the step-by-step procedure:

1. Receive an image and process it to detect red pixels.
2. Set the detected pixel counts as inputs to the respective Nengo nodes.
3. The nodes send this input to the neural ensemble.
4. The neural ensemble produces spikes in reaction to the input.
5. Probes collect the spike data from the ensemble.
6. If the number of spikes crosses a threshold, a certain condition (**stop-and-reverse**) is met.

The concept here is to use a biologically inspired mechanism (spiking neural network) to detect significant concentrations of red colors in the center regions of images.

2.3.2 Vibration Data Acquisition as the Unconditional Stimulus

In our associative learning experiment, we used vibration data as the US. We recorded real-time vibration data using the vibration plate. We have a vibration plate under our T-maze from Vibration Therapeutic, as shown in Figure 2:19. We used this vibration plate to generate 35-Hz vibration data with approximately 2.8-mm amplitudes.

Figure 2:19: Vibration plate underneath our T-maze, which is used to record vibration data.

The LIMO robot is equipped with an IMU that captures these vibrations. Agilex's ROS software, running on the LIMO robot, processes the IMU data, streaming the robot's acceleration to an IMU topic accessible by other ROS nodes. This allows for the vibration data to be recorded and replayed, simulating real-time data publication. By adding Nengo to a ROS node that is subscribed to the IMU topic, a way for ROS and Nengo to talk to each other is created. This makes it easier to add the raw acceleration data to the network with little preprocessing.

The preprocessing step involves removing the z-axis's average acceleration due to Earth's gravity (9.81 m/s^2) to normalize the acceleration values of all three axes to zero. Initially, the acceleration norm |a| was utilized according to Equation 2.3 to evaluate the LIMO robot's vibration state.

$$|a| = \sqrt{{a_x}^2 + {a_y}^2 + (a_z - 9.81)^2} \qquad 2.3$$

However, given that the z-axis acceleration significantly deviates from its resting value compared to the other axes, it was determined that measuring the z-axis acceleration alone is sufficient for detecting vibrations. Thus, the preprocessing equation was simplified, as shown in Equation 2.4, to enhance the system's efficiency. This simplified process, focusing on z-axis acceleration, aligns with the overall acceleration trends, making it the chosen method for vibration preprocessing.

$$|a_{zr}| = |a_z - 9.81| \qquad 2.4$$

The IMU data is processed through the US input node, which evaluates the simplified equation 2.4 and outputs according to $|a_{zr}|$. This output is conceptualized as a spike generator, with a firing rate proportional to its value.

We plotted linear x, y, and z acceleration data and their resultant acceleration norm data in several scenarios to understand the behavior of the vibration data. Figure 2:20 shows the linear x, y, and z acceleration data and their resultant acceleration norm data under a 0 Hz vibration condition when the robot was moving. From the plot, it seems that due to the robot movement, it creates some fake acceleration spikes. Figure 2:21 shows the linear x, y, and z acceleration data and their resultant acceleration norm data under a 0 Hz vibration condition when the robot was not moving. From the plot, it is evident that there were no vibration spikes as the robot was still and no external vibration was given. Figure 2:22 shows the linear x, y, and z acceleration data and their resultant acceleration norm data under a 35 Hz vibration condition while the robot was moving. From the plot, it is evident that when the robot was on the vibration plate for approximately 0.8 to 2.4 seconds, we got a lot of vibration spikes. Figure 2:23 shows the linear x, y, and z acceleration data and their resultant acceleration norm data under a 35 Hz vibration condition while the robot was still (just sitting on the vibration plate). From the plot, it is evident that, as the robot was on the vibration plate all the time, there were a lot of vibration spikes throughout the whole-time duration.

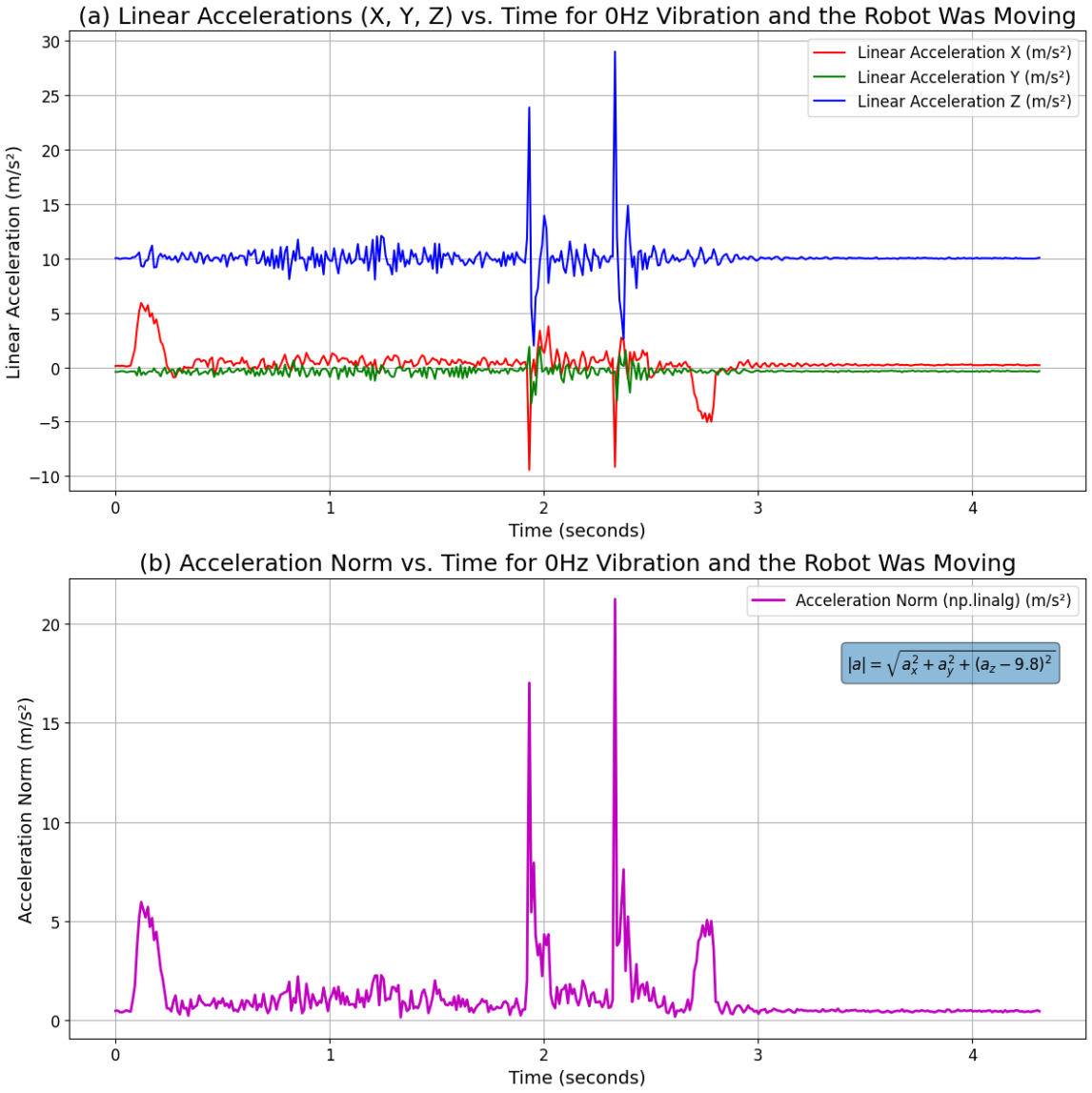

Figure 2:20: Linear acceleration (X, Y, Z) and their resultant acceleration norm with 0 Hz vibration, and the robot was moving.

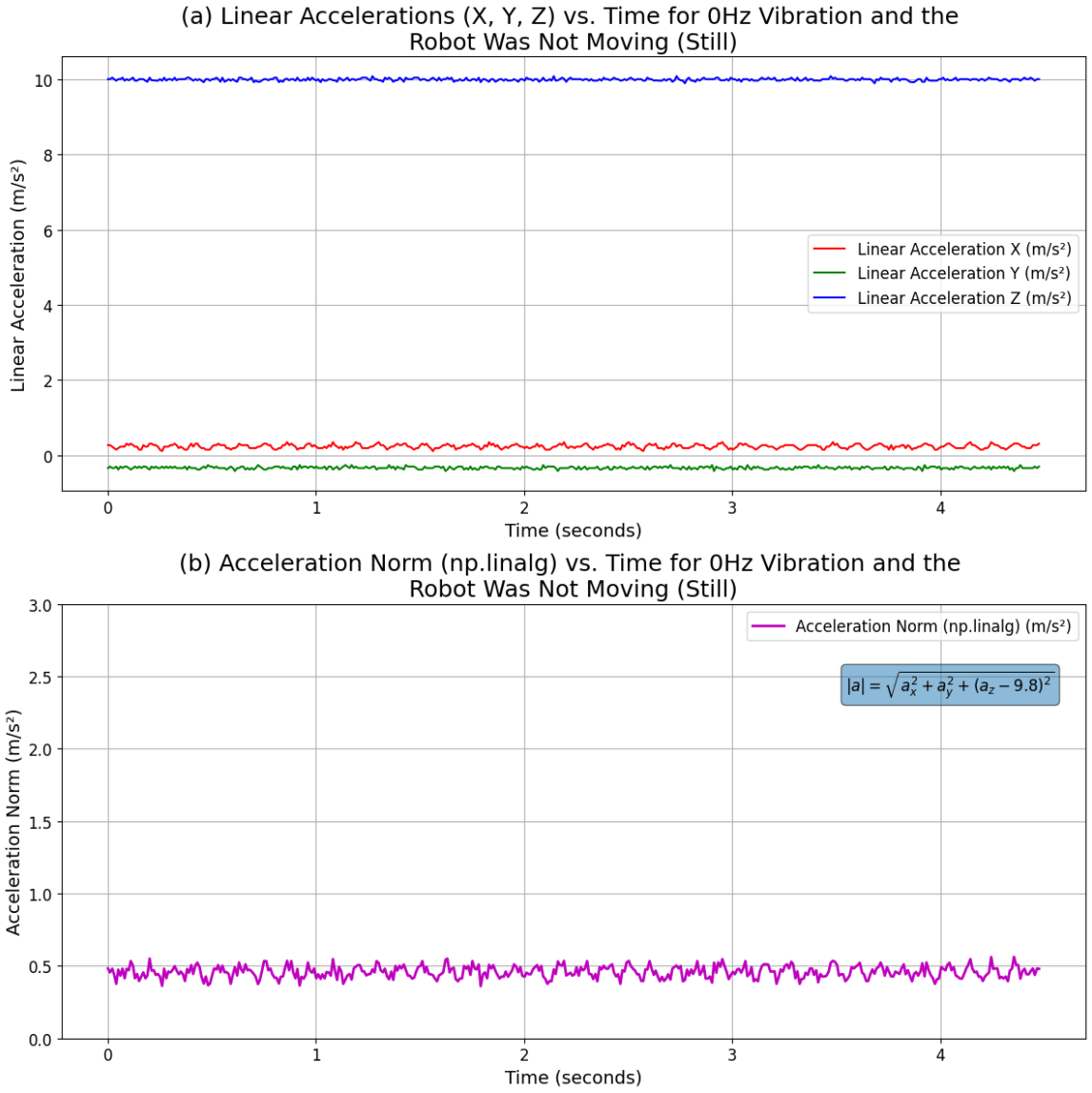

Figure 2:21: Linear acceleration (X, Y, Z) and their resultant acceleration norm with 0 Hz vibration, and the robot was still.

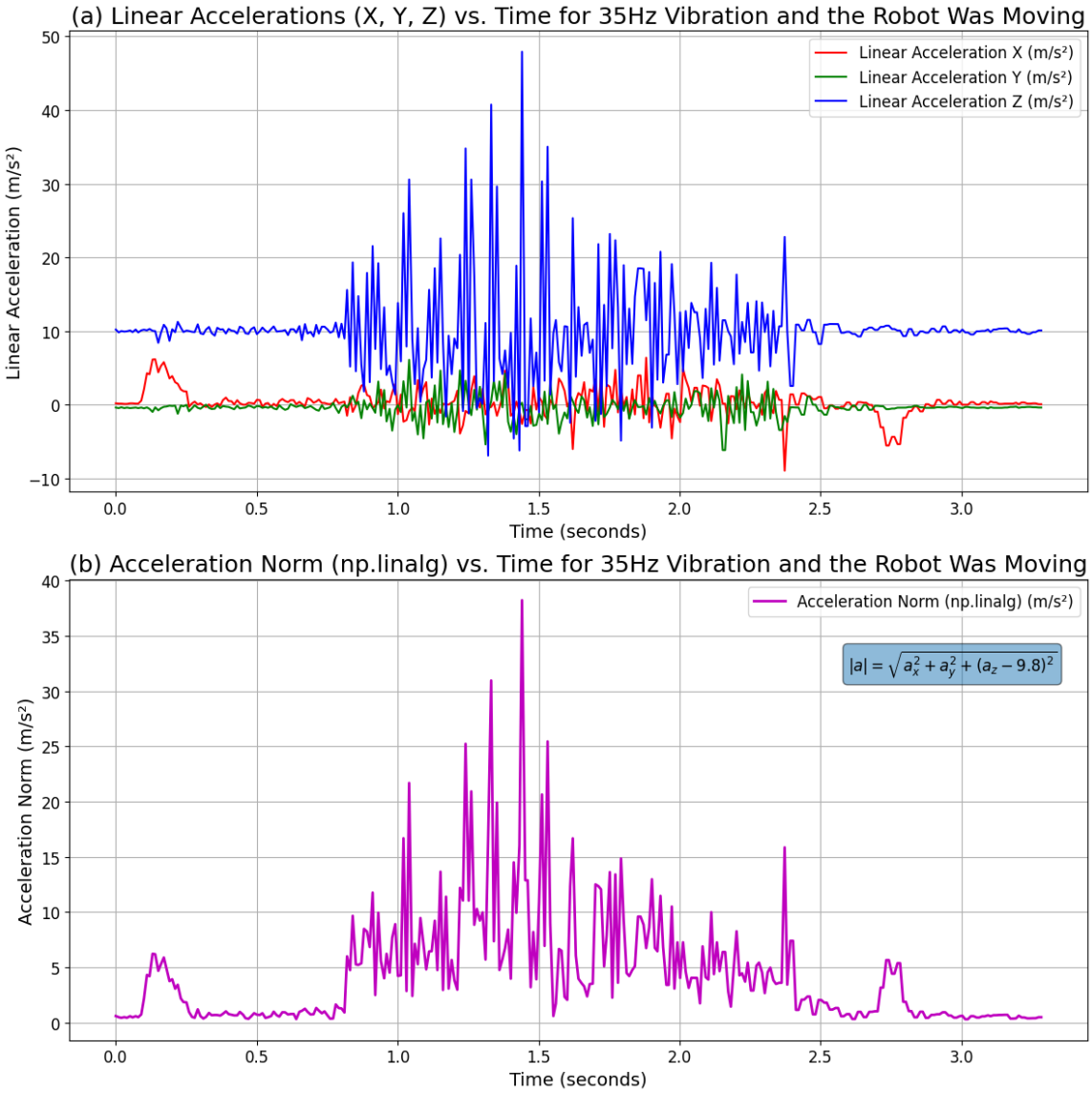

Figure 2:22: Linear acceleration (X, Y, Z) and their resultant acceleration norm with 35 Hz vibration, and the robot was moving.

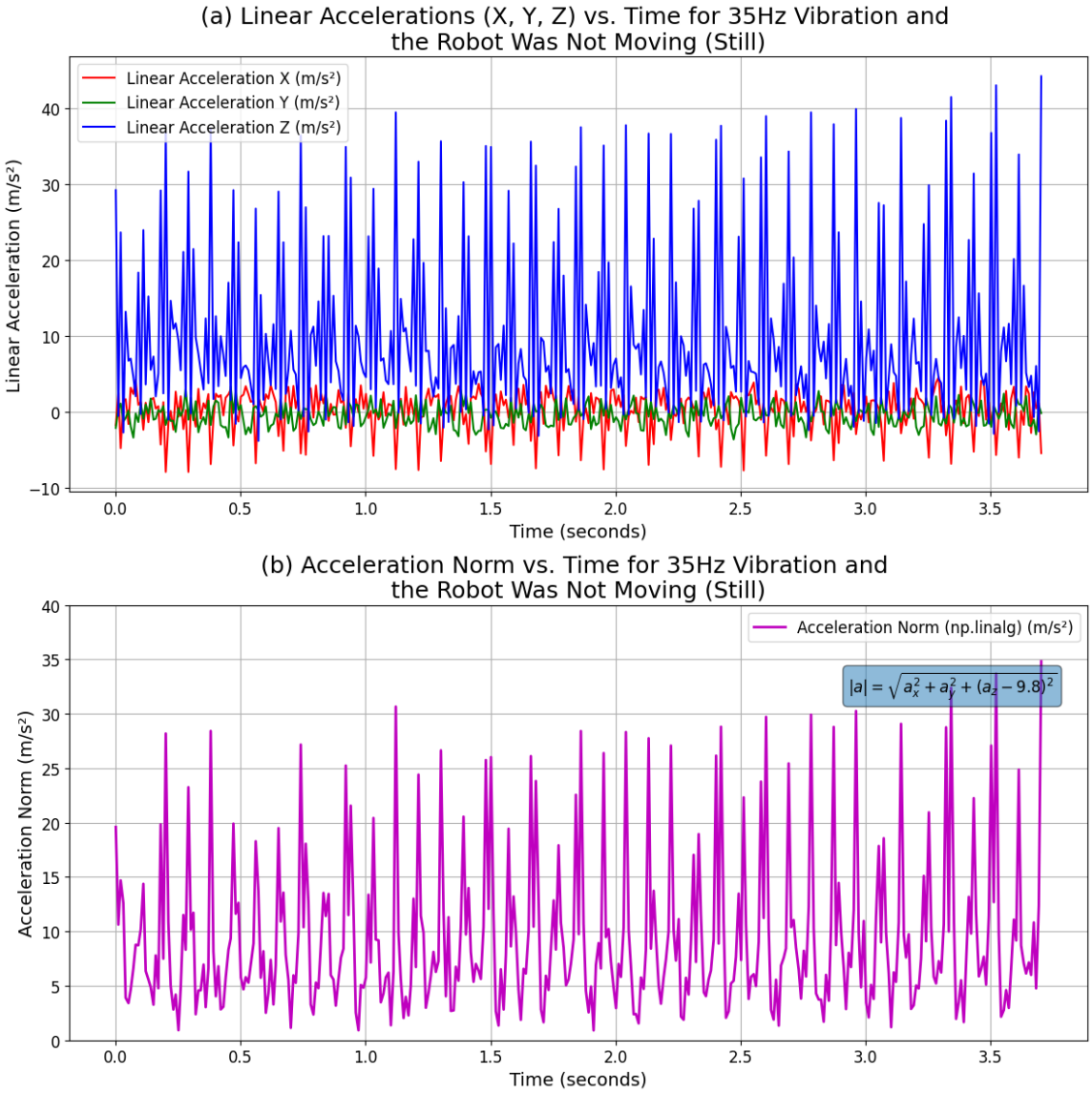

Figure 2:23: Linear acceleration (X, Y, Z) and their resultant acceleration norm with 35 Hz vibration, and the robot was still on the vibration plate.

To mitigate the impact on the LIMO robot while fostering an authentic simulation of extraterrestrial terrains akin to those found on Mars or on adversarial celestial bodies, our study employed vibration data generated from vibration cubes rather than resorting to a vibration plate. Positioned beneath the T-maze, the vibration plate can produce vibrations; however, to accurately emulate the uneven surfaces the robot would encounter, we utilized vibration cubes. These cubes not only generate moderate vibrations but also replicate the challenging, rugged terrain akin to that of hostile planets like Mars or the Moon, providing a realistic test environment for the robot's mobility. Figure 2:24 illustrates the vibration cubes employed in our experiment to produce the relevant vibration data.

Figure 2:24: Vibration cubes used to generate vibration.

Figure 2:25 shows the linear x, y, and z acceleration data and their resultant acceleration norm data when the robot was moving, and the vibration cubes are placed on the robot's path. From the plot, it is evident that when the robot was on the vibration cube, approximately from 0.5s to 1.5s, we got a good number of vibration spikes. These vibration spikes are enough for the vibration neuron to detect vibration.

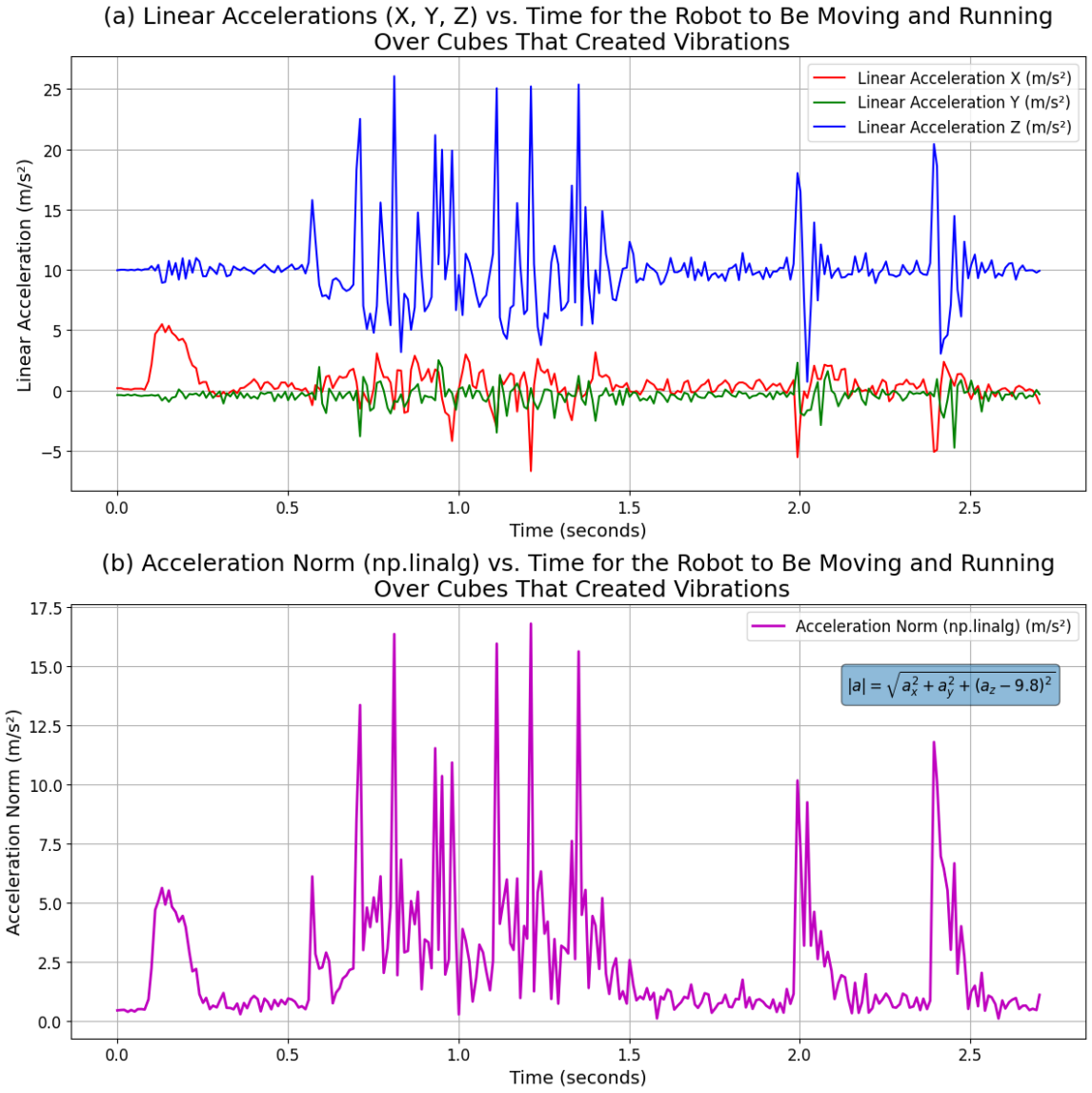

Figure 2:25: Linear acceleration (X, Y, Z) and their resultant acceleration norm when the robot was moving, and vibration cubes are placed on the robot's path.

Figure 2:26 shows the linear x, y, and z acceleration data and the z-axis rectified acceleration data when the robot was moving, and the vibration cubes are placed on the robot's path. From the plot, it is evident that the z-axis rectified acceleration has a similar pattern and magnitude as the resultant acceleration norm data. So, we can use z-axis rectified acceleration instead of resultant acceleration norm data for vibration detection to enhance the system's computing efficiency and for simplicity.

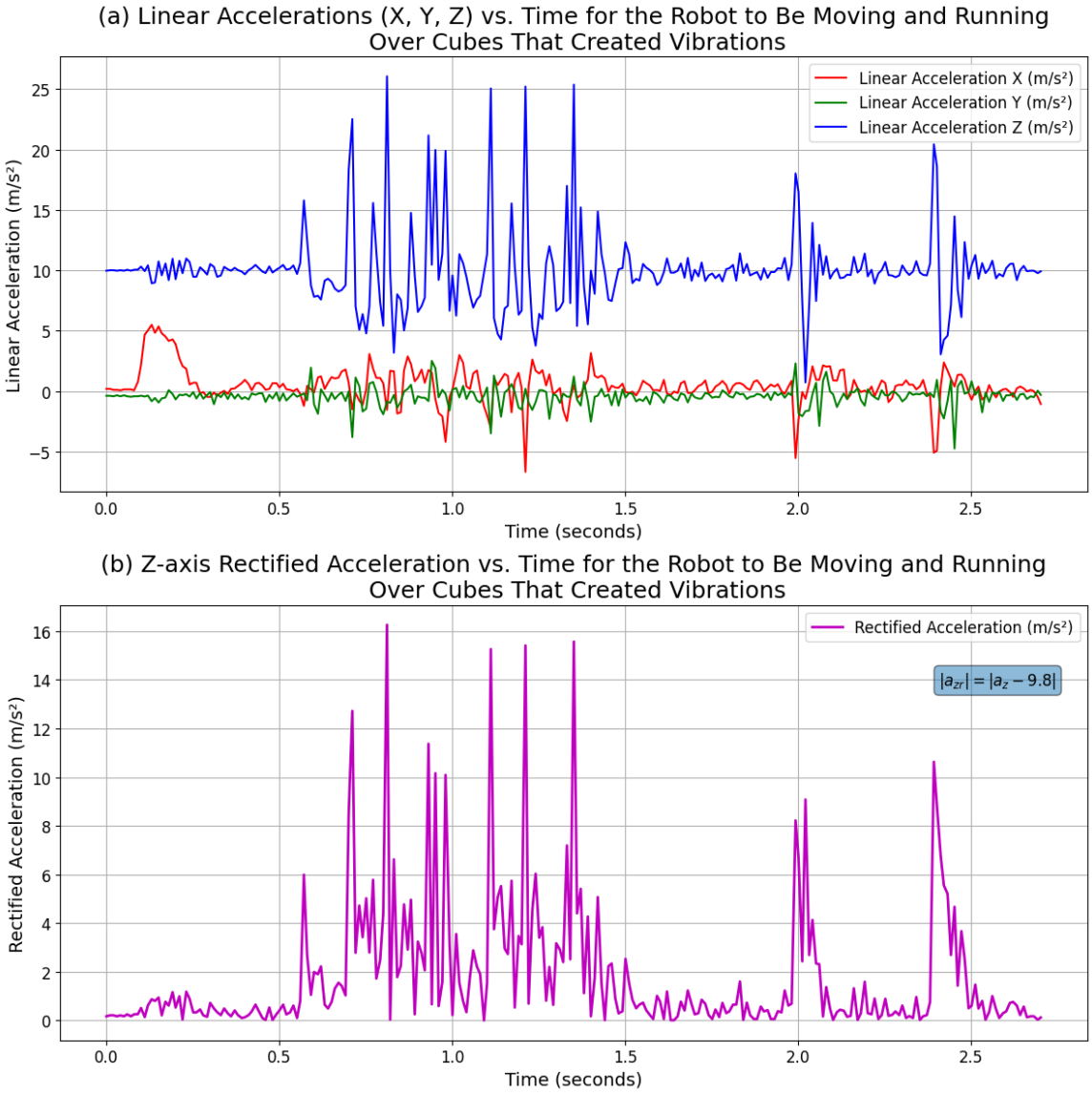

Figure 2:26: Linear acceleration (X, Y, Z) and z-axis rectified acceleration when the robot was moving, and vibration cubes were placed on the robot's path.

We have designed an ensemble LIF neuron to capture vibration data. The raw acceleration data is not converted to a standard range, such as 0 to +1, to cut down on preprocessing. Instead, the vibration detection neuron's input synapse and LIF parameters are fine-tuned so that it responds correctly to vibration signals.

In setting 1 of the vibration neuron, it can generate more output spikes, but it also generates a good number of spikes during times when it gets false vibration signals, as evident in Figure 2:27. From Table 2.12, we can observe that the bias value of the vibration neuron is 0.7. The LIF neuron in the network fires more when the bias is positive. A bias value of positive 0.7 makes the vibration neuron more responsive,

making sure that there is a certain level of output-spiking activity. A positive bias effectively means adding a constant positive current to the neuron. This makes the neuron more likely to reach its threshold voltage and fire spikes, even in the absence of external inputs. The higher the positive bias, the higher the baseline firing rate of the neuron. Neurons with a positive bias can respond more quickly to incoming positive inputs since they are closer to their threshold at rest.

Table 2.12: Properties of LIF neurons.

Neuron Parameters	τ_{RC}	τ_{ref}	$V_{reset}\ (V)$	$V_{th}(V)$	**Gain (A)**	**Bias**
Vibration Neuron	0.02	0.002	0	1	1.3	0.7
Color neuron	0.03	0.02	0	1	0.9	0.15
Movement neuron	0.04	0.002	0	1	1	0.01

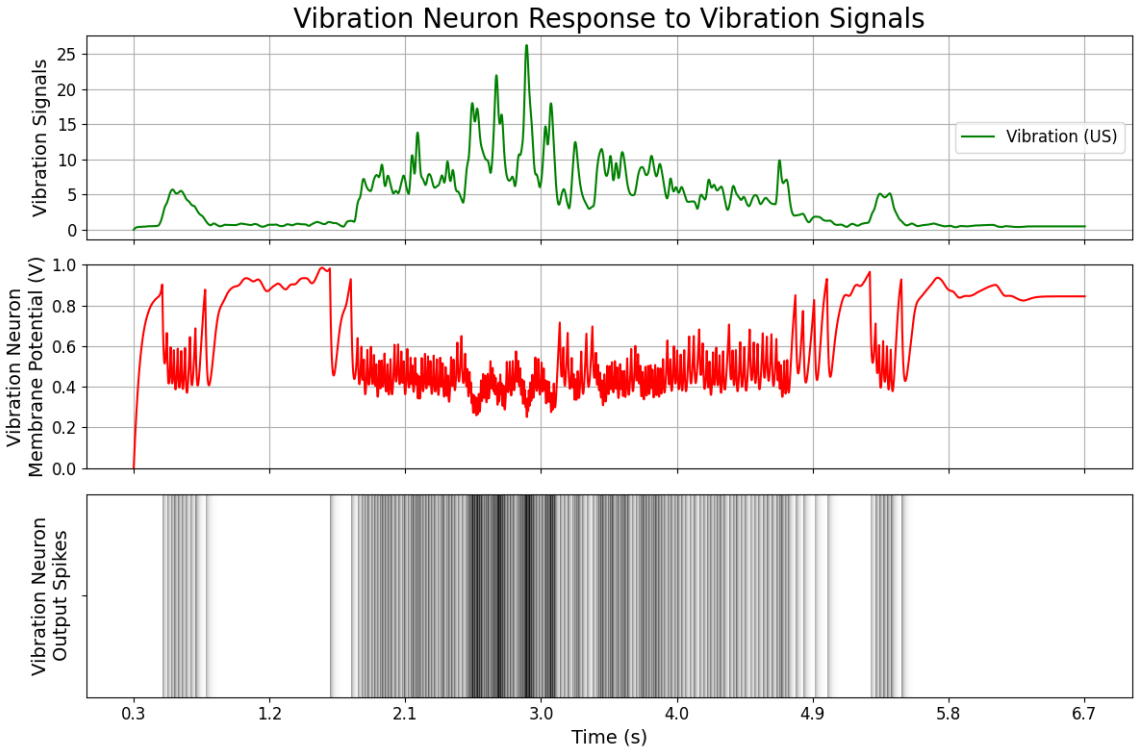

Figure 2:27: Vibration detection neuron response to acceleration.

In setting 2 of the vibration neuron, it can generate a good number of output spikes, but it also generates some spikes during times when it gets false vibration signals, as evident in Figure 2:28. From Table 2.13, we can observe that the bias value of the vibration neuron is 0. Zero bias means no constant current is added to or subtracted from the neuron. The neuron's firing activity will then solely depend on the incoming inputs. Without external inputs, neurons with zero bias will not fire. These neurons have a balanced approach to incoming signals, neither predisposed to firing nor artificially suppressed. Their activity directly reflects the dynamics and magnitude of the input signals they receive. Zero

biases are useful for creating networks that closely reflect the properties of the input signals without intrinsic biases towards higher or lower activity levels.

Table 2.13: Properties of LIF neurons.

Neuron Parameters	τ_{RC}	τ_{ref}	$V_{reset}\ (V)$	$V_{th}(V)$	**Gain (A)**	**Bias**
Vibration Neuron	0.02	0.002	0	1	1.3	0
Color neuron	0.03	0.02	0	1	0.9	0.15
Movement neuron	0.04	0.002	0	1	1	0.01

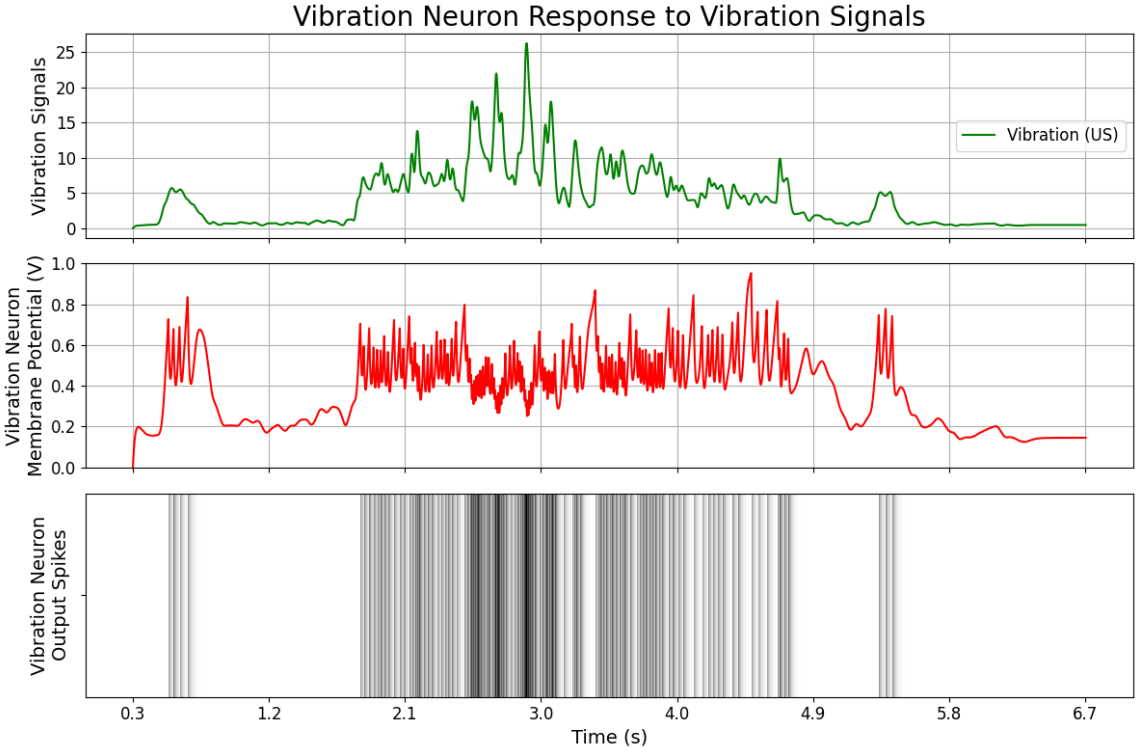

Figure 2:28: Vibration detection neuron response to acceleration.

3 Associative Learning Experiment

In this section, we delve into the techniques, strategies, and experimentation for simulating associative learning in rats using mobile robotics within a T-maze framework. Our research involved conducting associative learning experiments using two distinct approaches. In the initial approach, we substituted the electric shock, traditionally employed as an US in standard T-maze experiments, with the use of red colors. Additionally, we replaced the buzzer sound, which typically acts as a CS, with audio command signals. For our second approach, we employed vibrations from cubes as the US and red color as the CS. To navigate the robot through the T-maze's left and right paths, we processed left and right audio commands using the Xylo neuromorphic chip. We adopted a neural network model comprising the assembly of three Leaky Integrate-and-Fire (LIF) neurons to facilitate associative learning, dedicating one neuron each for processing the conditional and unconditional stimuli and another for integrating the responses from these two neurons. LIF neurons were chosen for their efficiency in simulating the transmission and processing of signals by brain cells, offering a balance between complexity and computational cost. The LIF neural network was executed using Nengo, a neuromorphic simulation tool designed by Applied Brain Research [114]. The operation of the Agilex LIMO robot and the exchange of data with the Nengo software were managed using the ROS framework [115].

3.1 Associative Learning with Visual and Auditory Stimuli in a T-Maze Using a Neuromorphic Robot

In our first approach, the perception of the red color serves as an US, while the audio command functions as a CS. When the robot detects the red color ahead of it, it stops and reverses direction to simulate a rat's fear response.

To enable the robot to replicate a fear-conditioning response, we have created multiple specialized neuron models that are capable of converting audio commands and color data into spike signals. Furthermore, the specific motion neurons are designed for controlling the robot's movement.

These neurons are adapted from the traditional Leaky Integrate and Fire (LIF) neurons, described by Equation 3.1.

$$C_m \frac{dV_m}{dt} = \frac{C_m}{\tau_{RC}}(E_L - V_m) + A \times I_{app} \qquad 3.1$$

where A denotes the input signal gain, τ_{RC} is the membrane RC time constant, C_m defines the membrane capacitance, and V_m is the membrane potential, E_L leak potential.

Table 3.1 documents all the parameter values of LIF neurons. The values are computed and refined to ensure they yield the intended results for their experiments. The gain and

bias of the color-detecting neuron were determined empirically so that it fires when stimuli are red but not when stimuli are other colors.

Table 3.1: LIF neuron parameters.

Neuron Parameters	τ_{RC}	τ_{ref}	V_{reset} (V)	V_{th} (V)	Gain (A)
Color neuron	0.02	0.002	0.5	1.0	1.5
Command neuron	0.03	0.02	-0.5	0.9	1
Movement neuron	0.04	0.002	-0.1	0.8	1

Only when the audio features have a high enough aggregate output—the command neuron, a LIF neuron with experimentally calculated gain and bias—fire. The movement neuron fires whenever it receives continuous input spikes from command or color neurons. The firing activations of these neurons are illustrated in Figure 3.1, Figure 3.2, and Figure 3.3, respectively.

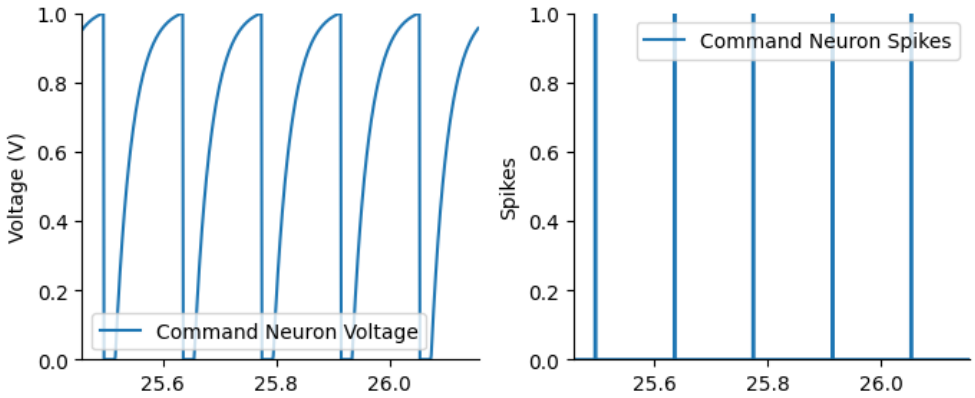

Figure 3:1: Command neuron firing activity.

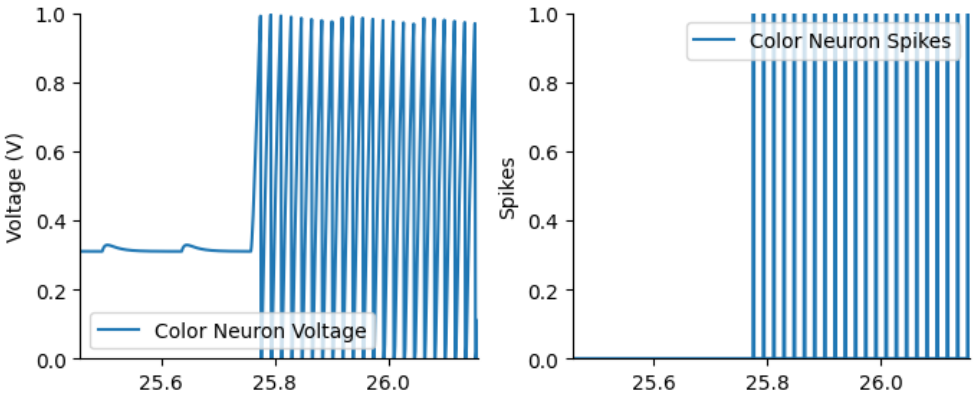

Figure 3:2: Color neuron firing activity.

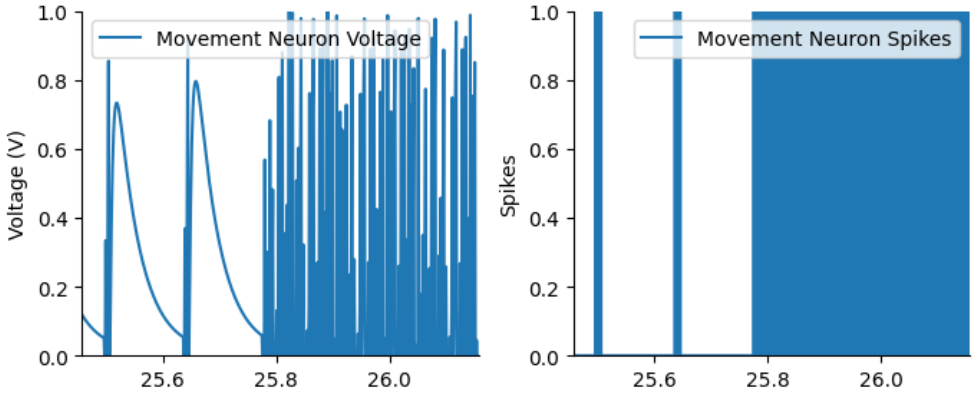

Figure 3:3: Movement neuron firing activity.

Figure 3.4 illustrates the practical implementation methodology of associative learning that we have employed for both simulation and experiment.

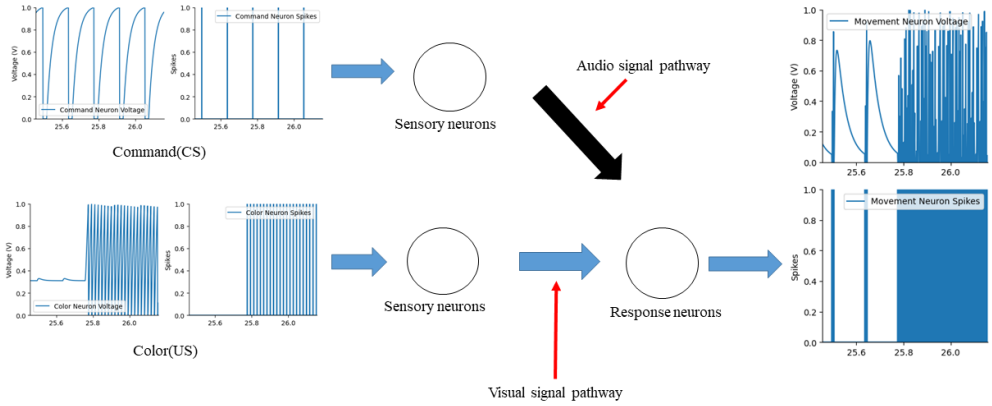

Figure 3:4: Practical implementation methodology of associative learning for simulation and experiment.

Figure 3.5 depicts the experimental setup using our neuromorphic robot. In facilitating real-time robot navigation via audio commands, the XyloMonitor deployment tool was employed. This tool processes live audio inputs specifically directed in "left" and "right" directions and predicts the intended direction for the robot. The XyloMonitor is configured to capture and process audio commands, with a brief waiting period mandated for the Analog Front-End (AFE) auto-calibration, ensuring optimal and consistent audio data acquisition.

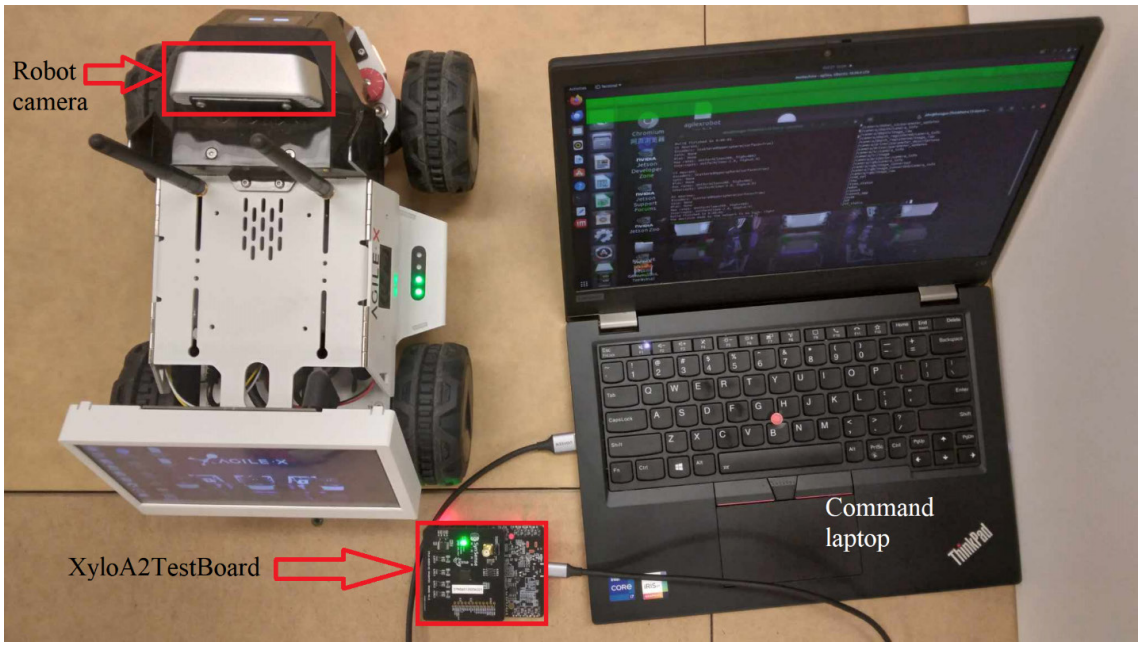

Figure 3:5: Experimental setup of a mobile robot and a Xylo neuromorphic chip.

To accommodate variations in command delivery, such as differences in volume or distances from the microphone, the sensitivity level of the audio capturing system is elevated using XyloMonitor. This enhancement ensures that the commands are recognized clearly and accurately.

The Xylo neuromorphic chip processes incoming audio over a duration of 250 milliseconds. Based on the neuron's membrane potential, indicative of the received audio, it identifies the intended direction. A "left" membrane potential prompts the robot to turn left, and similarly, a "right" potential signals a right turn. Post-processing, the definitive direction, either "left" or "right," is predicted and conveyed to the robot, guiding its subsequent movement.

In response to commands to turn the robot to the left or right, Figure 3.6 shows the membrane potential of audio neurons and the synaptic currents that go with them on the neuromorphic Xylo chip.

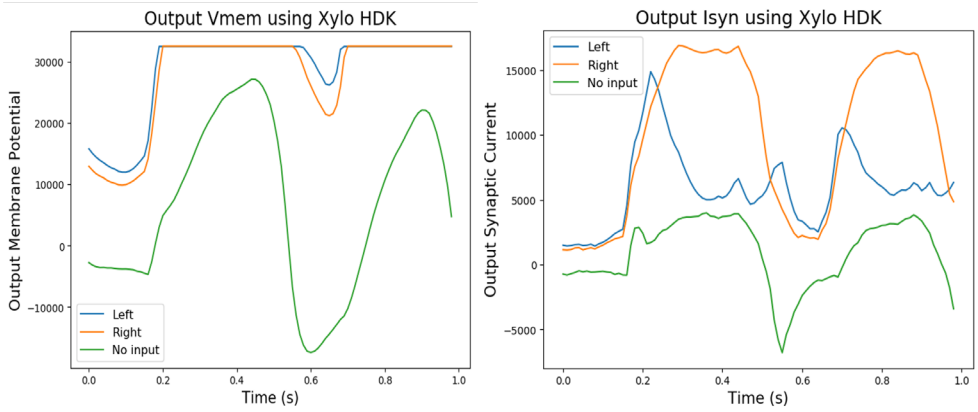

Figure 3:6: Output membrane potentials and synaptic currents of the Xylo chip.

Upon receiving a "Left" or "Right" command prediction from the Xylo neuromorphic chip, an additional neural network is engineered to dictate the neuromorphic robot movement towards left and right. This network comprises ten leaky integrate-and-fire (LIF) neurons designed to predictably steer the robot left or right.

The membrane time constant and the refractory period are the two parameters that define the chosen neuron model, Leaky Integrate-and-Fire (LIF). The refractory period has been minimized to accelerate neuron firing after an output spike.

Table 3.2: Synaptic weights of the neurons for the left turn.

Neuron Index	**Synaptic Weights from Input to Neurons**	**Synaptic Weights from Neurons to Output**
1	45.0	9.0
2	-27.0	-18.0
3	72.0	27.0
4	-9.0	-36.0
5	63.0	45.0
6	-54.0	54.0
7	18.0	-63.0
8	81.0	72.0
9	-63.0	-81.0
10	9.0	9.0

The network receives a constant input, which, when passed through the neurons with the given synaptic weights, produces a specific firing pattern. This pattern translates into a mean firing probability, determining the robot's movement direction as part of the decision-making process. If the firing probability exceeds a threshold of 0.5, it prompts the neuromorphic robot to turn left; otherwise, it turns right.

To ensure the neuromorphic robot consistently turns left with a mean firing probability of 0.9, specific synaptic weights and neuron properties are fixed. Figure 2:15 illustrates the firing probability of LIF neurons over time for the left turn.

Table 3.3: Neuron properties for the left turn.

Parameter	Value
Membrane time constant	0.02
Refractory period	0.001
Seed	42

To guarantee the neuromorphic robot consistently turns right with a mean firing probability of 0.2, specific synaptic weights and neuron properties are established. Figure 2:16 illustrates the firing probability of Leaky Integrate-and-Fire (LIF) neurons over time for the right turn.

Table 3.4: Synaptic weights of right-turning neurons.

Neuron Index	Synaptic Weights from Input to Neurons	Synaptic Weights from Neurons to Output
1	2.9	0.9
2	-0.5	-0.4
3	3.7	0.8
4	-0.3	-0.6
5	3.0	0.7
6	-0.6	0.8
7	0.8	-0.9
8	3.8	0.7
9	-0.7	-1.1
10	2.8	0.6

Table 3.5: Right-turning neuron properties.

Parameter	Value
Membrane time constant	0.02
Refractory period	0.002
Seed	42

Hebbian learning is used to adjust the synaptic weights [116]. Oja's rule defines the change in presynaptic weights given the output response of a neuron to its inputs [117]. Mathematically, it is possible to calculate the change in synaptic weight w by:

$$\Delta w = \eta\,(xy - \beta w y^2) \qquad 3.2$$

where: η is the learning rate. x is the firing rate of the presynaptic neuron. y is the firing rate of the postsynaptic neuron. β is a constant that determines the strength of the normalization. w is the current synaptic weight.

The term $\beta w y^2$ serves as a mechanism to prevent weights from growing without being bound. It functions as a subtractive normalization, adjusting the scale of weight changes based on both the size of the weight and the postsynaptic activity. This approach helps maintain stability and control in the learning process, ensuring that synaptic weights do not undergo uncontrolled growth.

When both pre- and postsynaptic neurons are simultaneously active, Equation 3.2 [118] will change the synaptic weights between them. The rate at which synaptic weight changes depends on the learning rate as well as the firing rates of the pre- and postsynaptic neurons, respectively. The learning rate in our experiment is 1*e-4. Figure 3.7 demonstrates the results of our associative learning.

Initially, the movement neurons and the audio command neuron (CS) have small synaptic weights. Consequently, the movement neuron will not respond to the auditory commands of the "left" and "right" turns, leading to no turning in the T-maze. Through associative learning, the synaptic weights between the movement neurons and the audio command neurons are updated over time. As a result, when the color-detection neuron detects the red color (US) that the camera captured, the movement neuron fires. Both the command detection neuron and the movement neuron become active when color and audio stimuli are presented, leading to an increase in synaptic weights in the conditional signal pathway. Figure 3.7 illustrates that the simultaneous application of red color and audio command stimuli results in an increment of synaptic weights. In the initial timeframe, there is insufficient overlap, which cannot lead to a considerable increase in synaptic weight. Consequently, the application of audio commands alone cannot stimulate the movement neuron to become active, as shown in Figure 3.7. The second overlapping phase, which has a longer overlapped timeframe, successfully leads to a more substantial increase in synaptic weights. Subsequently, even in the absence of a red color input, the movement neuron will fire in response to an auditory stimulus (a "left" or "right" command). This illustrates the achievement of associative learning.

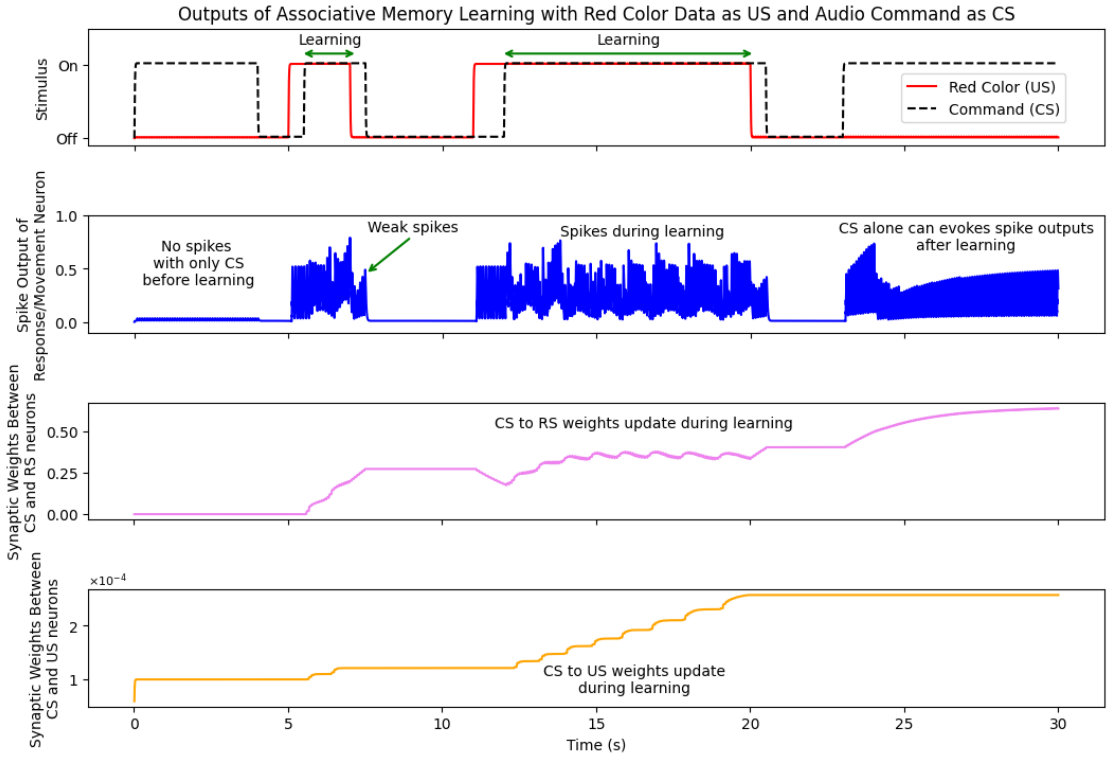

Figure 3:7: Results of associative learning.

To evaluate and validate our associative learning system, we established a T-maze to replicate rat associative learning experiments. The simulation and experimental scenarios of associative learning in the T-maze are illustrated in Figure 3.8.

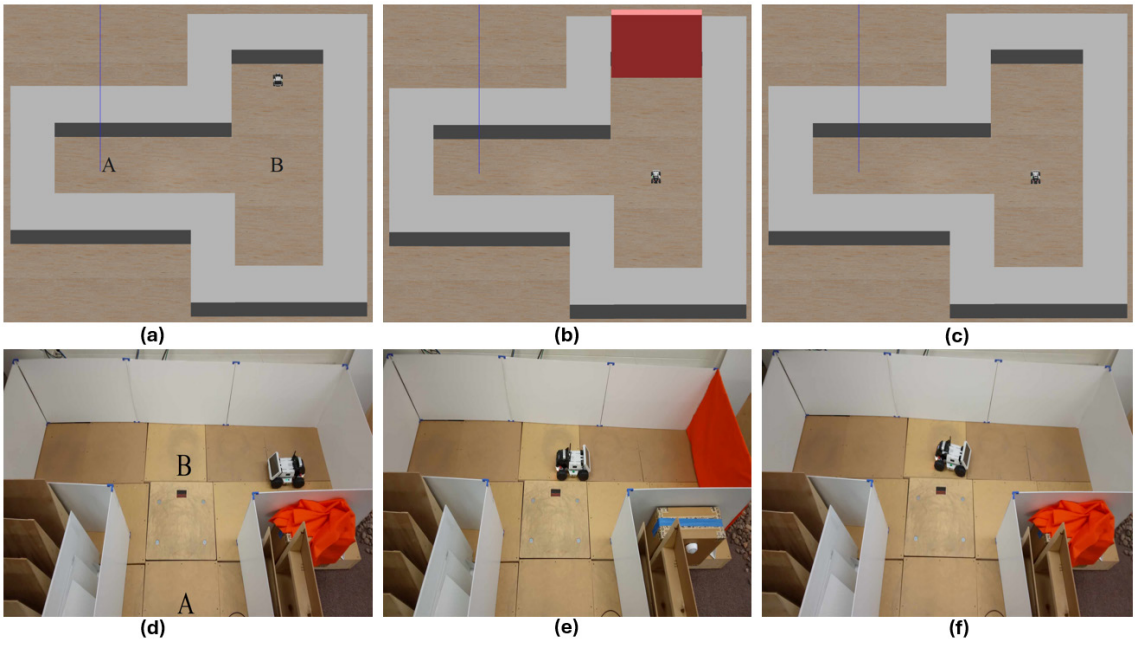

Figure 3:8: Replicating Associative Learning in a T-Maze: Simulation and Experimental Exploration with a Neuromorphic Robot. (a) Gazebo simulation depicting the T-maze for robot navigation. (b) Detection of red color. (US) input at position B. (c) The robot will learn to stop and turn to the arm with no red color presented. (d) The actual experimental setup of T-maze. (e) Detection of red color (US) input at position B in the actual T-maze. (f) The robot will learn to stop and turn to the arm with no red color presented.

In our setups, the red color serves as an US, and the audio commands act as the CS. For both simulation and experimentation, we have implemented the following associative learning steps:

1. When the right command is given, the robot moves in the direction of the right arm of the T-maze (assessing the performance of the conditional signal pathway).

2. Upon resetting the robot's position, present the right arm with the red color only, and the robot will reverse and move in the direction of the left arm (assessing the performance of the unconditional signal pathway).

3. Upon resetting the robot's position, give the right command (CS) and the red color (US) together, and the robot will stop and reverse in the direction of the left arm upon seeing the red color in the right arm direction of the T-maze. (Learning: CS and US together)

4. Repeat step 3 to increase the synaptic weights further.

5. Reset the robot's position. Give only the right command; the robot stops and reverses in the opposite direction when it sees the right arm because of associative learning, even though the red color is not present this time.

Initially, the robot starts moving forward from the starting chamber, marked as position A. The robot proceeds from position A to the turning point, annotated as position B. When the robot reaches position B, based on the "left" or "right" command provided, it turns 90 degrees either left or right and moves forward into the left or right arms of the T-maze, as shown in Figure 3.8 (a).

The red color is assigned as an unpleasing stimulus. The response to this unpleasure is that the robot decides to turn in the opposite direction, towards the arm with no red color presented.

During associative learning, initially, the robot conducts a similar movement from location A to B, where the turning commands and red color are presented simultaneously. Consequently, the color detection neuron fires, and the robot chooses to move towards the arm with no red color presented, as shown in Figure 3.8 (b). During this time, synaptic weights increase from the unconditional stimulus (red color) and the motion neurons as both audio spikes and color spikes are present together.

After 2-3 associative learning trials in the T-maze, the neuromorphic robot memorizes and relates the red color and audio commands together by enhancing the corresponding synaptic strengths. As a result, even with no red color presented, the robot will still move the “safe” arm it memorized. The safe here refers to the arm that was not placed on the red color board. As illustrated in Figure 3.8 (c), when the robot moves to position B, it stops, reverses 180 degrees to the right, and moves forward into the arm, which is memorized as "safe.". The experiments in the actual T-maze replicate the same associative learning, which is shown in Figure 3.8 (d-f).

3.2 Exploring Vibration and Color Cues for Associative Learning in a T-Maze with a Neuromorphic Robot

In our 2nd approach, the vibration data from vibration cubes serves as an US, while the perception of red color serves as a CS. When the robot detects the vibration, it stops and reverses direction to simulate a rat's fear response.

The vibration detection method, designed for real-time vibration data monitoring and analysis, leverages the IMU's ability to capture precise linear acceleration data along three axes. The core steps of the procedure include:

1. Acceleration Norm Calculation

The initial step involves calculating the norm of the acceleration vector derived from the IMU sensor data. This vector norm serves as a quantitative measure of the total acceleration magnitude the sensor experiences at any given moment. The calculation is performed using the Euclidean norm formula for a three-dimensional vector, which is expressed as:

$$Acceleration\ Norm = \sqrt{a_x^2 + a_y^2 + (a_z - 9.81)^2} \quad 3.3$$

Here, a_x, a_y, and a_z represent the linear acceleration components measured along the x, y, and z axes, respectively. 9.81 ms^{-2} is due to gravity, which is subtracted from a_z.

2. Single Vibration Event Detection

Upon determining the acceleration norm, the method involves comparing this value against a predefined threshold to identify significant vibration events. The logic for single vibration event detection is defined as:

$$Vibration\ Event = \begin{cases} True & if\ Acceleration\ Norm > Threshold \\ Fasle & otherwise \end{cases} \quad 3.4$$

This equation categorizes an event as significant if the acceleration norm exceeds the set threshold, indicating a potential state of vibration. For our work, we used a threshold value of 15 ms^{-2}.

3. Temporal Filtering and Event Timestamping

Following the identification of significant vibration events, timestamps are recorded. These timestamps are then filtered to focus on recent events, typically within the last 1 second, to ensure the analysis remains relevant to the system's current state. The filtering criterion is represented as:

$$Filtered\ Timestamps = \{t |\ t \in Vibration\ Event\ Timestamps, Current\ Time - t \leq \Delta T\} \quad (3.5)$$

Here, ΔT signifies the chosen time window for retaining relevant vibration events, with t denoting the timestamps of these events. For our work, ΔT is set to 1 second.

4. Vibration Detection

The final step quantifies whether the observed frequency of significant vibration events within the designated time window suggests abnormal activity. The condition for confirming the detection of a vibration anomaly is:

$$Vibration\ Detected = Count(Filtered\ Timestamps) \geq N \quad (3.6)$$

N is the threshold for the number of vibration events required to identify the observation as indicative of an unusual vibration pattern or anomaly. For our work, we set N to 10, which means if the acceleration norm spikes exceed 15 ms^{-2} 10 times in a quick 1-second period, then the vibration will be detected.

To enable the robot to replicate a fear-conditioning response, we have created multiple specialized neuron models that are capable of converting vibration and color data into spike signals. Furthermore, the specific motion neurons are designed for controlling the robot's movement. The movement neuron fires whenever it receives continuous input spikes from vibration or color neurons. These neurons are all modified from the standard LIF neurons, which are defined using Equation 3.1.

Table 3.6 documents all the parameter values of LIF neurons. The values are computed and refined to ensure they yield the intended results for their experiments. The gain and bias of the color-detecting neuron were determined empirically so that it fires when stimuli are red but not when stimuli are other colors. Similarly, the gain and bias of the vibration neuron were determined empirically so that it fires only when there is vibration.

Table 3.6: Properties of LIF neurons.

Neuron Parameters	τ_{RC}	τ_{ref}	$V_{reset}\ (V)$	$V_{th}(V)$	**Gain (A)**	**Bias**
Vibration Neuron	0.02	0.002	0	1	1.3	-0.7
Color neuron	0.03	0.02	0	1	0.9	0.15
Movement neuron	0.04	0.002	0	1	1	0.01

The vibration neuron can generate a good number of output spikes, but there are no spikes during times when it gets false vibration signals, as evident in Figure 3.9. From Table 3.6, we can observe that the bias value of the vibration neuron is -0.7. A negative bias applies a constant negative current, making it harder for the neuron to reach its firing threshold. This can reduce the baseline firing rate or even prevent the neuron from firing altogether in the absence of sufficiently strong positive inputs. Neurons with a negative bias are less likely to fire spontaneously and require stronger positive inputs to become active. This can be used to model inhibitory effects or to fine-tune the network's sensitivity to inputs. The negative bias of -0.7 can help in dampening noise, reducing

unnecessary firing, and increasing the contrast in the network's response to different inputs.

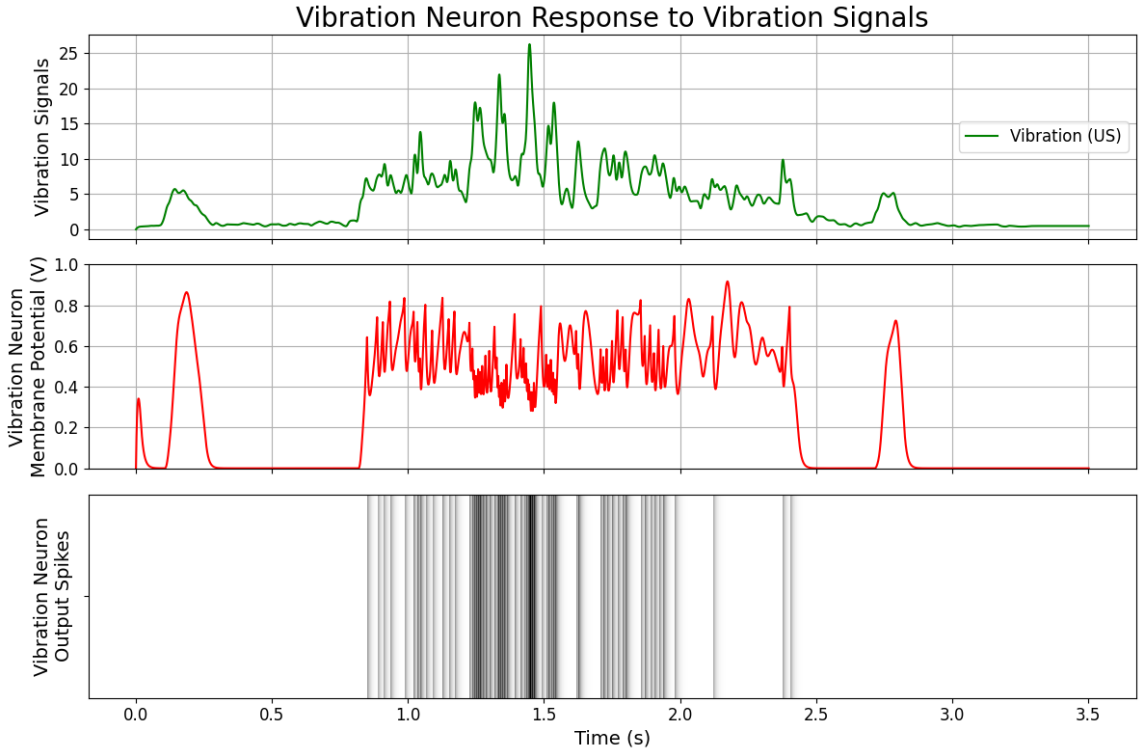

Figure 3:9: Vibration detection neuron response to acceleration.

Furthermore, we have designed a LIF neuron for color detection with a gain of 0.9 and a bias of 0.15, as tabulated in Table 3.6. The gain and bias of the color-detecting neuron were determined empirically so that it fires when stimuli are red but not when stimuli are other colors. From Figure 3.10, it is evident that the color detection neuron spikes on when the red color is available.

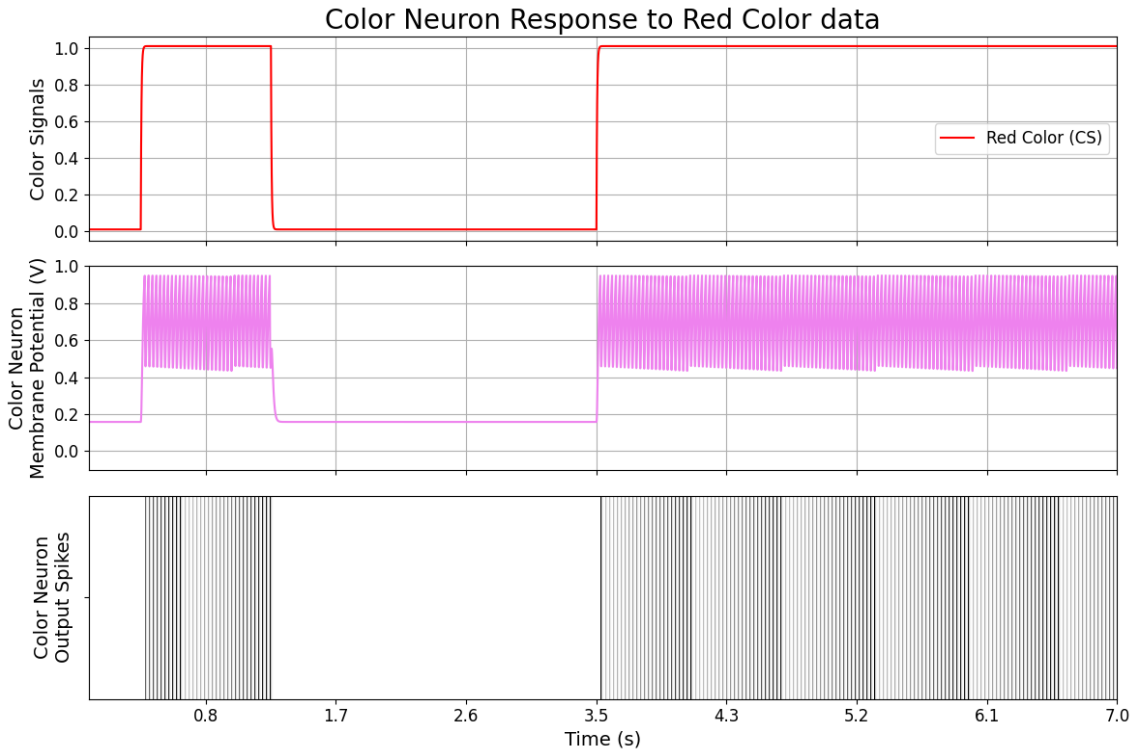

Figure 3:10: Color detection neuron response to red color.

Figure 3.11 illustrates the practical implementation methodology of associative learning that we have employed for the experiment.

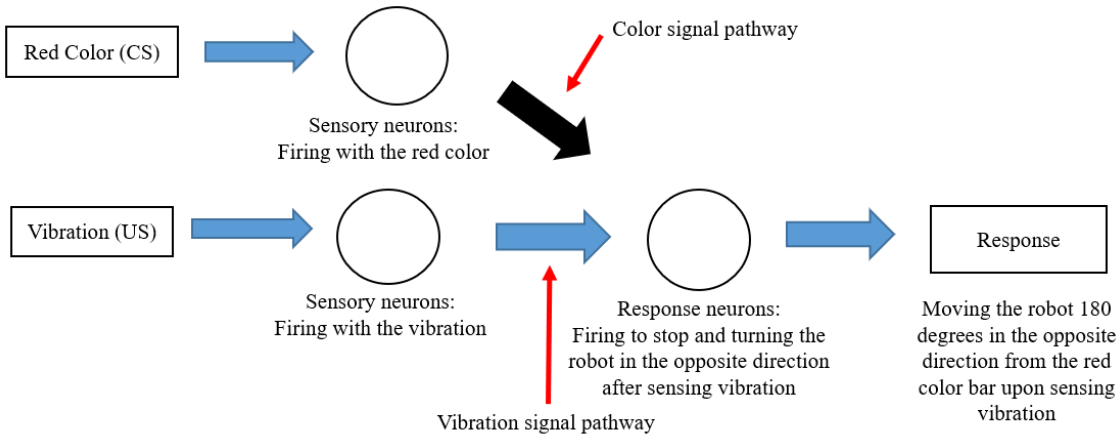

Figure 3:11: Practical implementation methodology of associative learning for the experiment.

Figure 3.5 depicts the experimental setup using our neuromorphic robot. In facilitating real-time robot navigation via audio commands, the XyloMonitor deployment tool was employed. This tool processes live audio inputs specifically directed in "left" and "right"

directions and predicts the intended direction for the robot. The XyloMonitor is configured to capture and process audio commands, with a brief waiting period mandated for the Analog Front End (AFE) auto-calibration, ensuring optimal and consistent audio data acquisition.

To accommodate variations in command delivery, such as differences in volume or distances from the microphone, the sensitivity level of the audio capturing system is elevated using XyloMonitor. This enhancement ensures that the commands are recognized clearly and accurately.

The Xylo neuromorphic chip processes incoming audio over a duration of 250 milliseconds. Based on the neuron's membrane potential, indicative of the received audio, it identifies the intended direction. A "left" membrane potential prompts the robot to turn left, and similarly, a "right" potential signals a right turn. Post-processing, the definitive direction, either "left" or "right," is predicted and conveyed to the robot, guiding its subsequent movement.

In response to commands to turn the robot to the left or right, Figure 3.6 shows the membrane potential of audio neurons and the synaptic currents that go with them on the neuromorphic Xylo chip.

Upon receiving a "Left" or "Right" command prediction from the Xylo neuromorphic chip, an additional neural network is engineered to dictate the neuromorphic robot movement towards left and right. This network comprises ten Leaky Integrate-and-Fire (LIF) neurons designed to predictably steer the robot left or right.

The membrane time constant and the refractory period are the two parameters that define the chosen neuron model, Leaky Integrate-and-Fire (LIF). The refractory period has been minimized to accelerate neuron firing after an output spike.

The network receives a constant input, which, when passed through the neurons with the given synaptic weights, produces a specific firing pattern. This pattern translates into a mean firing probability, determining the robot's movement direction as part of the decision-making process. If the firing probability exceeds a threshold of 0.5, it prompts the neuromorphic robot to turn left; otherwise, it turns right.

To ensure the neuromorphic robot consistently turns left with a mean firing probability of 0.9, specific synaptic weights and neuron properties are fixed. Figure 2:15 illustrates the firing probability of LIF neurons over time for the left turn. Table 3.2 and Table 3.3 document the synaptic weights of neurons and neuron properties, respectively, for the left turn.

To guarantee the neuromorphic robot consistently turns right with a mean firing probability of 0.2, specific synaptic weights and neuron properties are established. Figure 2:16 illustrates the firing probability of Leaky Integrate-and-Fire (LIF) neurons over time for the right turn. Table 3.4 and Table 3.5 document the synaptic weights of neurons and neuron properties, respectively, for the right turn.

Hebbian learning is used to adjust the synaptic weights [116]. Oja's rule defines the change in presynaptic weights given the output response of a neuron to its inputs [117]. Mathematically, it is possible to calculate the change in synaptic weight w using Equation 3.2.

The term $\beta w y^2$ serves as a mechanism to prevent weights from growing without being bound. It functions as a subtractive normalization, adjusting the scale of weight changes based on both the size of the weight and the postsynaptic activity. This approach helps maintain stability and control in the learning process, ensuring that synaptic weights do not undergo uncontrolled growth.

When both pre- and postsynaptic neurons are simultaneously active, Equation 3.2 [118] will change the synaptic weights between them. The rate at which synaptic weight changes depends on the learning rate as well as the firing rates of the pre- and postsynaptic neurons, respectively. The learning rate in our experiment is 6*e-2. Figure 3.12 demonstrates the results of our associative learning.

Initially, the movement neurons and the red color neuron (CS) have small synaptic weights. Consequently, the movement neuron will not respond to the red color, leading to no turning in the T-maze. Through associative learning, the synaptic weights between the movement neurons and the red color neurons are updated over time. As a result, when the vibration neuron detects the vibration (US) that the IMU captured, the movement neuron fires. Both the red color neuron and the movement neuron become active when vibration and red color stimuli are presented, leading to an increase in synaptic weights in the conditional signal pathway. Figure 3.12 illustrates that the simultaneous application of vibration and red color stimuli results in an increment of synaptic weights. In the initial timeframe, there is insufficient overlap, which cannot lead to a considerable increase in synaptic weight. Consequently, the application of red color alone cannot stimulate the movement neuron to become active, as shown in Figure 3.12. The second overlapping phase, which has a longer overlapped timeframe, successfully leads to a more substantial increase in synaptic weights. Subsequently, even in the absence of a vibration input, the movement neuron will fire in response to a red color stimulus. This illustrates the achievement of associative learning.

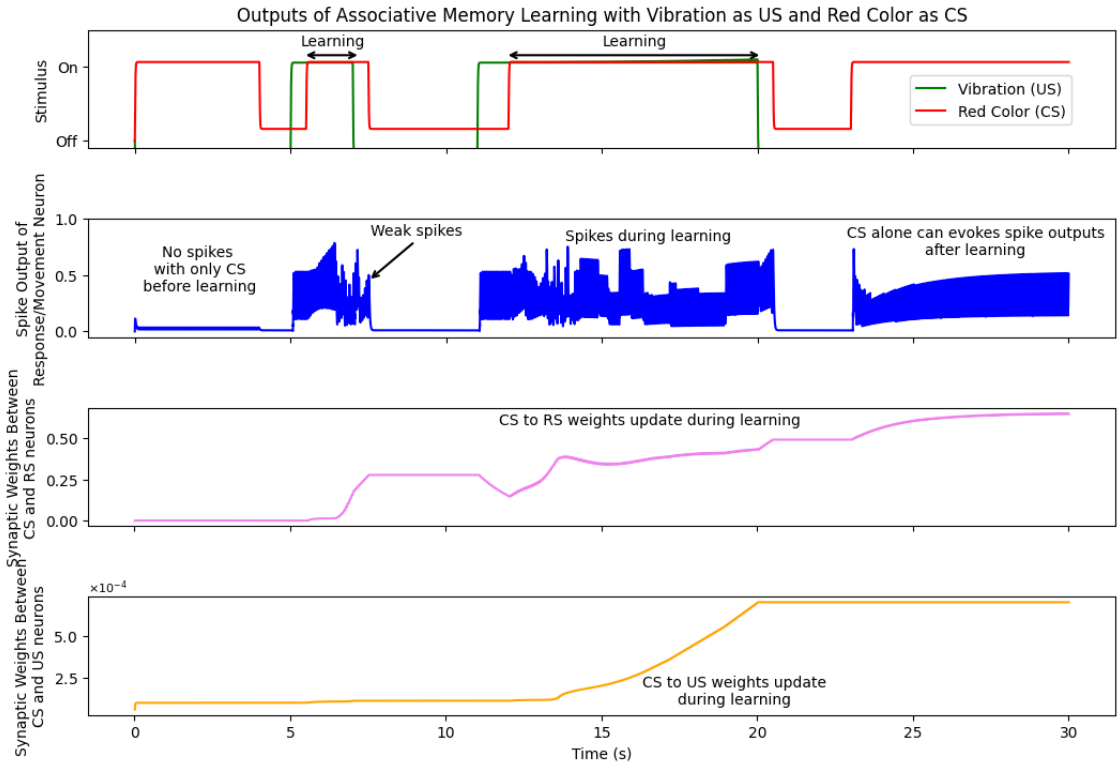

Figure 3:12: Results of associative learning.

To evaluate and validate our associative learning system, we established a T-maze to replicate rat associative learning experiments. The experimental scenarios of associative learning in the T-maze are illustrated in Figure 3.13.

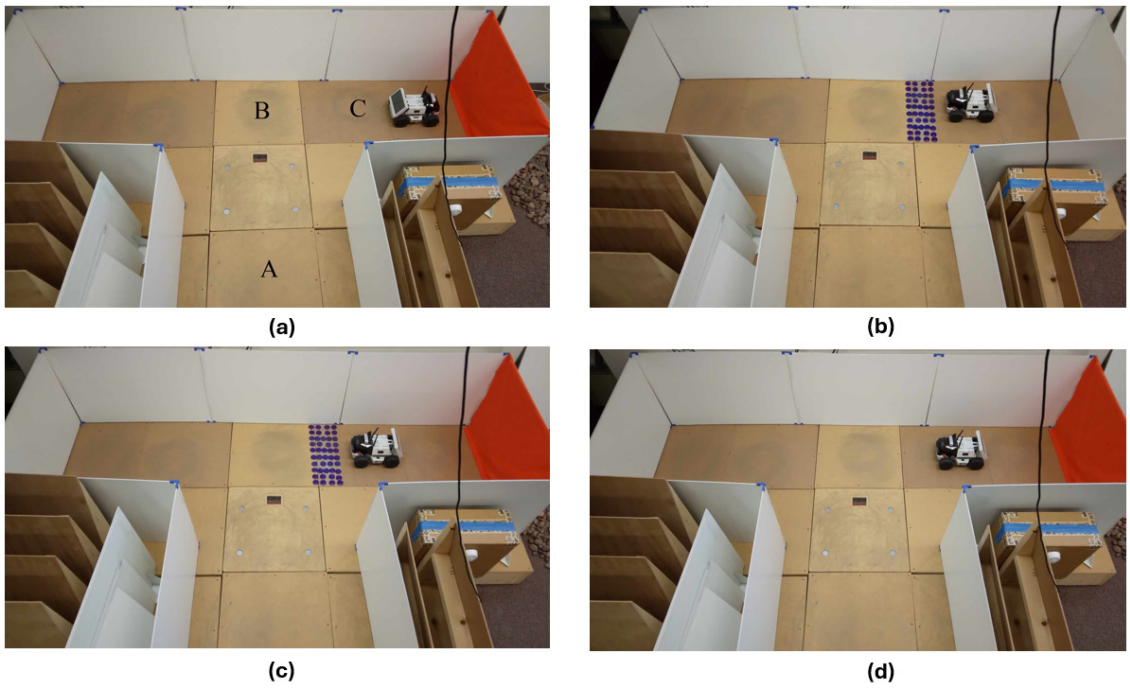

Figure 3:13: Replicating Associative Learning in a T-Maze: Experimental Exploration with a Neuromorphic Robot. (a) The actual experimental setup of T-maze. (b) Detection of vibration (US) input in between positions B and C in the actual T-maze. (c) Present the robot with both vibration and red color stimuli. (d) The robot will learn to stop and turn to the arm with no vibration presented.

In our setups, the vibration serves as an US, and the red color acts as a CS. For experimentation, we have implemented the following associative learning steps:

1. When only the red color (CS) is present, the robot moves towards the right arm based on the right command. (conditional signal pathway)

2. Upon resetting the robot position with only vibration (US), the robot will stop and reverse upon moving over the vibration cubes in the direction of the left arm (arm with no vibration presented) of the T-maze, though the right arm moving command is given. (unconditional signal pathway)

3. Upon resetting the robot position, present the red color (CS) and the vibration (US) together. The robot will stop and reverse upon moving over the vibration cubes in the direction of the left arm (arm with no vibration presented) of the T-maze, though the right arm moving command is given. (Learning: CS and US together)

4. Repeat step 3 to increase the synaptic weights further.

5. Repeat step 3 to increase the synaptic weights further.

6. Reset the robot's position. Present with only red color (CS), the robot stops and reverses towards the left arm (the robot memorizes the left arm as a safe arm as no vibration was presented in the left arm) of the T-maze due to associative learning, though the right command is given and vibration is not present.

Initially, the robot starts moving forward from the starting chamber, marked as position A. The robot proceeds from position A to the turning point, annotated as position B. When the robot reaches position B, based on the "right" command, it turns 90 degrees right and reaches position C, and as we have only presented conditional stimulus (red color), it moves forward into the right arms of the T-maze from position C, as shown in Figure 3.13 (a).

The vibration is assigned as an unpleasing stimulus. The response to this unpleasure is that the robot decides to turn in the opposite direction, towards the arm, with no vibration presented. The robot proceeds from position A to the turning point, annotated as position B. When the robot reaches position B, based on the "right" command, it turns 90 degrees right and reaches position C, and as we have vibration cubes in between position B and C, the robot senses unconditional stimulus (vibration), stops and reverses at position C, and moves towards the left arms of the T-maze from position C, as shown in Figure 3.13 (b).

During associative learning, initially, the robot conducts a similar movement from location A to B to C, where the vibration and red color are presented simultaneously. Consequently, the vibration detection neuron fires, and the robot chooses to move towards the arm with no vibration presented from position C, as shown in Figure 3.13 (c). During this time, synaptic weights increase from the unconditional stimulus (vibration) and the motion neurons as both red color spikes and vibration spikes are present together.

After 2-3 associative learning trials in the T-maze, the neuromorphic robot memorizes and relates the vibration and red color together by enhancing the corresponding synaptic strengths. As a result, even with no vibration presented, the robot will still move to the “safe” arm it memorized from position C. The safe here refers to the arm that was not placed on the vibration cubes. As illustrated in Figure 3.13 (d), when the robot moves to position C, it stops, reverses 180 degrees to the left, and moves forward into the arm, which is memorized as "safe.".

3.3 Comparisons with State-of-the-Art Research Works

Table 3.7 presents a comprehensive comparison of scale and association capabilities between our work and other state-of-the-art research in the field. The comparison is structured around the number of neurons, the specific tasks they were designed to address, the learning methods employed, and the context in which validation occurred—ranging from pure simulation to actual experimental setups. References [33-35, 37, 119, 120] show how to use a small group of neurons in simulated environments without saying what the tasks are or how the neurons are trained. Reference [31] takes this a step further by using 20 neurons that have already been trained. Our experiments, which can be seen in the entries for the first and second experiments, show a big increase to 259 neurons that work on spatial learning and memory, using self-learning and no pretraining methods in both simulated and real-life experiments. This contrast highlights our work's substantial advancement in neuron count and complexity, particularly in applying these systems to concrete tasks like fear conditioning, as demonstrated in the work referenced [121, 122].

Table 3.7: Assessing scale and association capability in comparison with modern benchmark studies.

Research Works	**Number of Neurons**	**Task**	**Learning Methods**	**Validation**
[34]	6	N/A	N/A	Simulation
[35]	3	N/A	N/A	Simulation
[37]	5	N/A	N/A	Simulation
[33]	3	N/A	N/A	Simulation
[119]	3	N/A	N/A	Simulation
[120]	3	N/A	N/A	Simulation
[31]	20	N/A	Pretraining	Simulation
[121, 122]	1419	Fear conditioning	No pretraining	Experiment
1st experiment	259	Spatial learning and memory	Self-learning	Simulation & Experiment
2nd experiment	259	Spatial learning and memory	No pretraining	Experiment

4 Conclusion and Future Work

This research work implements rodent associative learning in a T-maze using a mobile robot and an advanced neuromorphic chip (Xylo) in a real-time learning environment. By integrating the Xylo neuromorphic chip into a mobile robot, the system replicates the classic T-maze experiments observed in rodents. Specifically, the mobile robot takes the place of rats in the T-maze spatial learning and memory experiment. It processes two types of stimuli: one conditional and the other unconditional. In our first experimental setup, the robot associates red color (US) with audio commands (CS) through LIF neurons and Hebbian learning. In our second experimental setup, the robot uses vibration as the unconditional stimulus and color as the conditional one. Initially, the robot was programmed to stop and reverse in the direction of the opposite arm of the T-maze only in response to vibrations. However, after repeated simultaneous exposure to both stimuli, the robot began to stop and reverse in response to the red color alone. Leaky integrate and fire neurons power this detection and response system, while specialized response neurons control the robot's movement. The Xylo neuromorphic chip is responsible for controlling the left and right movements in the left and right directions of the T-maze, respectively. Associative memory formation through signal pathway alteration is facilitated using Hebbian learning. This approach distinguishes itself from others by effectively mimicking rat fear conditioning in a T-maze setting without needing any labeled data or preliminary training sessions. The study demonstrates real-time associative learning, providing insights into potential applications for autonomous robots in environments with energy constraints and adaptive capabilities.

My contributions to the integration of neuromorphic computing in autonomous robotic systems include:

- Conducting an extensive review of neuromorphic computing and its applications [123].
- Successfully replicating rat spatial learning and memory experiments in a T-maze environment using the neuromorphic robot.
- Implementing associative learning techniques for mobile robots through neuromorphic computing.

As a next step in research, we should create different T-maze configurations to look at how adaptable and cognitively deep the neuromorphic system is, add different types of sensors (like tactile and olfactory ones) to improve the associative learning framework, and do longitudinal studies to see how stable and long-lasting neuromorphic memory is over time. Technically, the work could investigate precise energy metrics to test how well the neuromorphic chip works under different operational loads. It could also put these advanced systems to use in real-life situations that stress dynamic adaptability and improve learning algorithms to make synapses more flexible and speed up the learning process. Using detailed behavioral metrics could improve comparisons with biological counterparts, and researchers from different fields should try to use what they learn from neuroscience to improve algorithms and system architectures in neuromorphic robotics.

5 References

[1] H. Yang, Y. Zhu, and J. Liu, "Energy-constrained compression for deep neural networks via weighted sparse projection and layer input masking," *arXiv preprint arXiv:1806.04321,* 2018.

[2] I. Loy, S. Carnero-Sierra, F. Acebes, J. Muñiz-Moreno, C. Muñiz-Diez, and J.-C. Sánchez-González, "Where association ends. A review of associative learning in invertebrates, plants and protista, and a reflection on its limits," *Journal of Experimental Psychology: Animal Learning and Cognition,* vol. 47, no. 3, p. 234, 2021.

[3] J. S. Biane *et al.*, "Neural dynamics underlying associative learning in the dorsal and ventral hippocampus," *Nature Neuroscience,* vol. 26, no. 5, pp. 798-809, 2023.

[4] A. Dickinson, "Associative learning and animal cognition," *Philosophical Transactions of the Royal Society B: Biological Sciences,* vol. 367, no. 1603, pp. 2733-2742, 2012.

[5] R. d'Isa, G. Comi, and L. Leocani, "Apparatus design and behavioural testing protocol for the evaluation of spatial working memory in mice through the spontaneous alternation T-maze," *Scientific Reports,* vol. 11, no. 1, p. 21177, 2021.

[6] F. Zhang, C. Li, Z. Li, L. Dong, and J. Zhao, "Recent progress in three-terminal artificial synapses based on 2D materials: from mechanisms to applications," *Microsystems & Nanoengineering,* vol. 9, no. 1, p. 16, 2023.

[7] L. Wang *et al.*, "Incorporating neuro-inspired adaptability for continual learning in artificial intelligence," *Nature Machine Intelligence,* vol. 5, no. 12, pp. 1356-1368, 2023.

[8] S. J. Gershman, "A unifying probabilistic view of associative learning," *PLoS computational biology,* vol. 11, no. 11, p. e1004567, 2015.

[9] Y. Jeong *et al.*, "Synaptic plasticity-dependent competition rule influences memory formation," *Nature communications,* vol. 12, no. 1, p. 3915, 2021.

[10] W. C. Abraham, O. D. Jones, and D. L. Glanzman, "Is plasticity of synapses the mechanism of long-term memory storage?," *NPJ science of learning,* vol. 4, no. 1, p. 9, 2019.

[11] L. Alzubaidi *et al.*, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of big Data,* vol. 8, pp. 1-74, 2021.

[12] A. Singh *et al.*, "Deep learning-based predictions of older adults' adherence to cognitive training to support training efficacy," *Frontiers in Psychology,* vol. 13, p. 980778, 2022.

[13] J. L. McClelland, "Capturing advanced human cognitive abilities with deep neural networks," *Trends in Cognitive Sciences,* vol. 26, no. 12, pp. 1047-1050, 2022.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[15] 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.
[16] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843-852.
[17] S. Sengupta *et al.*, "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowledge-Based Systems,* vol. 194, p. 105596, 2020.
[18] H. An, M. S. Al-Mamun, M. K. Orlowski, L. Liu, and Y. Yi, "Robust deep reservoir computing through reliable memristor with improved heat dissipation capability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 40, no. 3, pp. 574-583, 2020.
[19] L. Alzubaidi *et al.*, "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications," *Journal of Big Data,* vol. 10, no. 1, p. 46, 2023.
[20] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access,* vol. 8, pp. 225134-225180, 2020.
[21] Y. Sandamirskaya, M. Kaboli, J. Conradt, and T. Celikel, "Neuromorphic computing hardware and neural architectures for robotics," *Science Robotics,* vol. 7, no. 67, p. eabl8419, 2022.
[22] J. Dupeyroux, S. Stroobants, and G. C. De Croon, "A toolbox for neuromorphic perception in robotics," in *2022 8th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*, 2022: IEEE, pp. 1-7.
[23] C. Bartolozzi, G. Indiveri, and E. Donati, "Embodied neuromorphic intelligence," *Nature communications,* vol. 13, no. 1, p. 1024, 2022.
[24] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature,* vol. 575, no. 7784, pp. 607-617, 2019.
[25] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, and S. Mack, *Principles of neural science*. McGraw-hill New York, 2000.
[26] 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,* vol. 50, no. 7, pp. 2935-2945, 2019.
[27] T. Kohonen, *Self-organization and associative memory*. Springer Science & Business Media, 2012.
[28] T. J. Nelson and D. L. Alkon, "Molecular regulation of synaptogenesis during associative learning and memory," *Brain research,* vol. 1621, pp. 239-251, 2015.
[29] S. H. Sung, Y. Jeong, J. W. Oh, H.-J. Shin, J. H. Lee, and K. J. Lee, "Bio-plausible memristive neural components towards hardware implementation of brain-like intelligence," *Materials Today,* 2023.
[30] B. Zivasatienraj and W. A. Doolittle, "Dynamical memristive neural networks and associative self-learning architectures using biomimetic devices," *Frontiers in Neuroscience,* vol. 17, p. 1153183, 2023.

[31] 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,* vol. 5, no. 4, pp. 668-678, 2019.
[32] S. Hu *et al.*, "Associative memory realized by a reconfigurable memristive Hopfield neural network," *Nature communications,* vol. 6, no. 1, p. 7522, 2015.
[33] K. Moon *et al.*, "Hardware implementation of associative memory characteristics with analogue-type resistive-switching device," *Nanotechnology,* vol. 25, no. 49, p. 495204, 2014.
[34] 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.
[35] 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.
[36] S. B. Eryilmaz *et al.*, "Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array," *Frontiers in neuroscience,* vol. 8, p. 205, 2014.
[37] 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.
[38] H. An, Z. Zhou, and Y. Yi, "Memristor-based 3D neuromorphic computing system and its application to associative memory learning," in *2017 IEEE 17th International Conference on Nanotechnology (IEEE-NANO)*, 2017: IEEE, pp. 555-560.
[39] R. M. Deacon and J. N. P. Rawlins, "T-maze alternation in the rodent," *Nature protocols,* vol. 1, no. 1, pp. 7-12, 2006.
[40] H. Bos and D. Muir, "Sub-mW Neuromorphic SNN audio processing applications with Rockpool and Xylo," *Embedded Artificial Intelligence: Devices, Embedded Systems, and Industrial Applications,* p. 69, 2023.
[41] D. S. Roy *et al.*, "Brain-wide mapping reveals that engrams for a single memory are distributed across multiple brain regions," *Nature communications,* vol. 13, no. 1, p. 1799, 2022.
[42] S. A. Josselyn and S. Tonegawa, "Memory engrams: Recalling the past and imagining the future," *Science,* vol. 367, no. 6473, p. eaaw4325, 2020.
[43] 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.
[44] L. A. DeNardo *et al.*, "Temporal evolution of cortical ensembles promoting remote memory retrieval," *Nature neuroscience,* vol. 22, no. 3, pp. 460-469, 2019.
[45] 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.

[46] T. Kitamura *et al.*, "Engrams and circuits crucial for systems consolidation of a memory," *Science,* vol. 356, no. 6333, pp. 73-78, 2017.
[47] S. Maren, G. Aharonov, and M. S. Fanselow, "Neurotoxic lesions of the dorsal hippocampus and Pavlovian fear conditioning in rats," *Behavioural brain research,* vol. 88, no. 2, pp. 261-274, 1997.
[48] S. Furber, "To build a brain," *IEEE spectrum,* vol. 49, no. 8, pp. 44-49, 2012.
[49] J. Q. Yang *et al.*, "Neuromorphic engineering: from biological to spike-based hardware nervous systems," *Advanced Materials,* vol. 32, no. 52, p. 2003610, 2020.
[50] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, "Physics for neuromorphic computing," *Nature Reviews Physics,* vol. 2, no. 9, pp. 499-510, 2020.
[51] C. Mead and M. Ismail, *Analog VLSI implementation of neural systems*. Springer Science & Business Media, 1989.
[52] H. An, K. Bai, and Y. Yi, "The roadmap to realize memristive three-dimensional neuromorphic computing system," *Advances in Memristor Neural Networks-Modeling and Applications,* pp. 25-44, 2018.
[53] 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.
[54] Y. C. Yi, H. An, and A. Engelbrecht, *Neuromorphic Computing*. BoD–Books on Demand, 2023.
[55] H. An, D. S. Ha, and Y. Yi, "Powering next-generation industry 4.0 by a self-learning and low-power neuromorphic system," in *Proceedings of the 7th ACM International Conference on Nanoscale Computing and Communication*, 2020, pp. 1-6.
[56] H. An, M. A. Ehsan, Z. Zhou, F. Shen, and Y. Yi, "Monolithic 3D neuromorphic computing system with hybrid CMOS and memristor-based synapses and neurons," *Integration,* vol. 65, pp. 273-281, 2019.
[57] H. An, J. Li, Y. Li, X. Fu, and Y. Yi, "Three dimensional memristor-based neuromorphic computing system and its application to cloud robotics," *Computers & Electrical Engineering,* vol. 63, pp. 99-113, 2017.
[58] H. An, Z. Zhou, and Y. Yi, "Opportunities and challenges on nanoscale 3D neuromorphic computing system," in *2017 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*, 2017: IEEE, pp. 416-421.
[59] H. An, M. A. Ehsan, Z. Zhou, and Y. Yi, "Electrical modeling and analysis of 3D Neuromorphic IC with Monolithic Inter-tier Vias," in *2016 IEEE 25th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2016: IEEE, pp. 87-90.
[60] M. A. Ehsan, H. An, Z. Zhou, and Y. Yi, "Design challenges and methodologies in 3D integration for neuromorphic computing systems," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016: IEEE, pp. 24-28.

[61] C. Zhao, J. Li, H. An, and Y. Yi, "Energy efficient analog spiking temporal encoder with verification and recovery scheme for neuromorphic computing systems," in *2017 18th International Symposium on Quality Electronic Design (ISQED)*, 2017: IEEE, pp. 138-143.

[62] M. A. Ehsan, H. An, Z. Zhou, and Y. Yi, "Adaptation of enhanced TSV capacitance as membrane property in 3D brain-inspired computing system," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1-6.

[63] H. An, "Powering next-generation artificial intelligence by designing three-dimensional high-performance neuromorphic computing system with memristors," 2020.

[64] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE,* vol. 78, no. 10, pp. 1629-1636, 1990.

[65] J. Stuijt, M. Sifalakis, A. Yousefzadeh, and F. Corradi, "μBrain: An event-driven and fully synthesizable architecture for spiking neural networks," *Frontiers in neuroscience,* vol. 15, p. 664208, 2021.

[66] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems,* vol. 19, no. 04, pp. 295-308, 2009.

[67] F. Nowshin, H. An, and Y. Yi, "Towards Energy-Efficient Spiking Neural Networks: A Robust Hybrid CMOS-Memristive Accelerator," *ACM Journal on Emerging Technologies in Computing Systems,* vol. 20, no. 1, pp. 1-20, 2024.

[68] H. An, M. S. Al-Mamun, M. K. Orlowski, and Y. Yi, "A three-dimensional (3D) memristive spiking neural network (M-SNN) system," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021: IEEE, pp. 337-342.

[69] V. K. Sangwan and M. C. Hersam, "Neuromorphic nanoelectronic materials," *Nature nanotechnology,* vol. 15, no. 7, pp. 517-528, 2020.

[70] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science,* vol. 2, no. 1, pp. 10-19, 2022.

[71] I. Boybat *et al.*, "Neuromorphic computing with multi-memristive synapses," *Nature communications,* vol. 9, no. 1, p. 2514, 2018.

[72] D. Ivanov, A. Chezhegov, M. Kiselev, A. Grunin, and D. Larionov, "Neuromorphic artificial intelligence systems," *Frontiers in Neuroscience,* vol. 16, p. 1513, 2022.

[73] SynSense, "SynSense advances ultra-low-power neuromorphic audio processing with Xylo™Audio 3 tapeout," 2023-07-07 2023. [Online]. Available: https://www.synsense.ai/synsense-advances-ultra-low-power-neuromorphic-audio-processing-with-xyloaudio-3-tapeout/.

[74] SynSense, "Rockpool is an open source Python package for developing signal processing applications with spiking neural networks.." [Online]. Available: https://www.synsense.ai/products/rockpool/.

[75] "rockpool." [Online]. Available: https://pypi.org/project/rockpool/.

[76] SynSense, "Samna is the developer interface to the SynSense toolchain and run-time environment for interacting with all SynSense devices." [Online]. Available: https://www.synsense.ai/products/samna/.

[77] "Welcome to Samna's documentation!." [Online]. Available: https://synsense-sys-int.gitlab.io/samna/.
[78] SynSense, "XYLO™ The world's leading programmable, ultra-low power neuromorphic chip for low-dimensional signal processing.." [Online]. Available: https://www.synsense.ai/products/xylo/.
[79] E. Nordlie, T. Tetzlaff, and G. T. Einevoll, "Rate dynamics of leaky integrate-and-fire neurons with strong synapses," *Frontiers in computational neuroscience,* vol. 4, p. 149, 2010.
[80] D. Tal and E. L. Schwartz, "Computing with the leaky integrate-and-fire neuron: logarithmic computation and multiplication," *Neural computation,* vol. 9, no. 2, pp. 305-318, 1997.
[81] S. Lu and F. Xu, "Linear leaky-integrate-and-fire neuron model based spiking neural networks and its mapping relationship to deep neural networks," *Frontiers in neuroscience,* vol. 16, p. 857513, 2022.
[82] A. G. Barto, R. S. Sutton, and P. S. Brouwer, "Associative search network: A reinforcement learning associative memory," *Biological cybernetics,* vol. 40, pp. 201-211, 1981.
[83] N. Gandhi, G. Ashkenasy, and E. Tannenbaum, "Associative learning in biochemical networks," *Journal of theoretical biology,* vol. 249, no. 1, pp. 58-66, 2007.
[84] D. Todes, *Ivan Pavlov: Exploring the animal machine*. Oxford University Press, 2000.
[85] A. Gelperin, "Associative memory mechanisms in terrestrial slugs and snails," in *Handbook of Behavioral Neuroscience*, vol. 22: Elsevier, 2013, pp. 280-290.
[86] J. Iwata and J. E. LeDoux, "Dissociation of associative and nonassociative concomitants of classical fear conditioning in the freely behaving rat," *Behavioral neuroscience,* vol. 102, no. 1, p. 66, 1988.
[87] M. B. Kennedy, "Synaptic signaling in learning and memory," *Cold Spring Harbor perspectives in biology,* vol. 8, no. 2, p. a016824, 2016.
[88] A. Holtmaat and P. Caroni, "Functional and structural underpinnings of neuronal assembly formation in learning," *Nature neuroscience,* vol. 19, no. 12, pp. 1553-1562, 2016.
[89] B. F. Grewe *et al.*, "Neural ensemble dynamics underlying a long-term associative memory," *Nature,* vol. 543, no. 7647, pp. 670-675, 2017.
[90] L. M. Zeltser, R. J. Seeley, and M. H. Tschöp, "Synaptic plasticity in neuronal circuits regulating energy balance," *Nature neuroscience,* vol. 15, no. 10, pp. 1336-1342, 2012.
[91] J. LeDoux, "The emotional brain, fear, and the amygdala," *Cellular and molecular neurobiology,* vol. 23, pp. 727-738, 2003.
[92] D. K. Ingram, "Complex maze learning in rodents as a model of age-related memory impairment," *Neurobiology of aging,* vol. 9, pp. 475-485, 1988.
[93] J. J. Bolhuis and M. Gahr, "Neural mechanisms of birdsong memory," *Nature reviews neuroscience,* vol. 7, no. 5, pp. 347-357, 2006.

[94] C. Fugazza and Á. Miklósi, "Social learning in dog training: The effectiveness of the Do as I do method compared to shaping/clicker training," *Applied Animal Behaviour Science,* vol. 171, pp. 146-151, 2015.
[95] C. J. Mitchell, J. De Houwer, and P. F. Lovibond, "The propositional nature of human associative learning," *Behavioral and Brain Sciences,* vol. 32, no. 2, pp. 183-198, 2009.
[96] N. C. Ellis, "Usage-based and form-focused language acquisition: The associative learning of constructions, learned attention, and the limited L2 endstate," in *Handbook of cognitive linguistics and second language acquisition*: Routledge, 2008, pp. 382-415.
[97] l. Agilex Robotics (Dongguan) CO, "LIMO Usage and Development Manual." [Online]. Available: https://github.com/agilexrobotics/limo-doc/blob/master/Limo%20user%20manual(EN).md.
[98] "Discovering The Future Of Robotics With AgileX's Limo Robots," November 9, 2023. [Online]. Available: https://www.chironix.com/post/limo-lineup-expansion.
[99] S. Simmie, "BOSTON UNIVERSITY USES AGILEX LIMO FOR RESEARCH," Jul 13, 2023. [Online]. Available: https://indrorobotics.ca/boston-university-uses-agilex-limo-for-research/.
[100] A. Paivio, "Mental imagery in associative learning and memory," *Psychological review,* vol. 76, no. 3, p. 241, 1969.
[101] L. A. Lefton and L. Brannon, "PSYCHOLOGY, 9/e," 2005.
[102] A. Salter, *Conditioned reflex therapy*. Wellness Institute, Inc., 2001.
[103] E. Oja, "Oja learning rule," *Scholarpedia,* vol. 3, no. 3, p. 3612, 2008.
[104] S. Seung, *Connectome: How the brain's wiring makes us who we are*. HMH, 2012.
[105] R. Yuste, "Cells that fire together, wire together," *Journal of NIH Research,* vol. 4, pp. 60-60, 1992.
[106] J. G. Raaijmakers and R. M. Shiffrin, "Search of associative memory," *Psychological review,* vol. 88, no. 2, p. 93, 1981.
[107] R. Kempter, W. Gerstner, and J. L. Van Hemmen, "Hebbian learning and spiking neurons," *Physical Review E,* vol. 59, no. 4, p. 4498, 1999.
[108] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience,* vol. 3, no. 9, pp. 919-926, 2000.
[109] P. I. Pavlov, "Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex," *Annals of neurosciences,* vol. 17, no. 3, p. 136, 2010.
[110] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 33, no. 7, pp. 2744-2757, 2020.
[111] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1-12.

[112] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," *Advances in neural information processing systems,* vol. 28, 2015.
[113] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, 2014: IEEE, pp. 10-14.
[114] T. Bekolay *et al.*, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in neuroinformatics,* vol. 7, p. 48, 2014.
[115] 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.
[116] P. Miller, *An introductory course in computational neuroscience*. MIT Press, 2018.
[117] A. R. Voelker, E. Crawford, and C. Eliasmith, "Learning large-scale heteroassociative memories in spiking neurons," *Unconventional Computation and Natural Computation,* vol. 7, no. 2014, 2014.
[118] P. Dayan and L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.
[119] M. Ziegler *et al.*, "An electronic version of Pavlov's dog," *Advanced Functional Materials,* vol. 22, no. 13, pp. 2744-2749, 2012.
[120] 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.
[121] N. Zins, Y. Zhang, and H. An, "Implementation of Associative Memory Learning in Mobile Robots Using Neuromorphic Computing," 2023.
[122] 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.
[123] M. A. B. Siddique, Y. Zhang, and H. An, "Monitoring time domain characteristics of Parkinson's disease using 3D memristive neuromorphic system," *Frontiers in computational neuroscience,* vol. 17, 2023.