POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Implementation of a Hippocampal-Cortical spiking neural network for memory semantization on Loihi 2 neuromorphic hardware

Advisors
Gianvito Urgese
Walter Gallego Gomez
Vittorio Fra

Candidate
Giovanni Orani

March 2026

Abstract

Neuromorphic computing is emerging as an important frontier of Artificial Intelligence (AI), helping overcome the energy costs and latency limitations of traditional Deep Neural Networks (DNNs) by drawing inspiration from biological neural circuits. In this context, Spiking Neural Networks (SNNs) employ biologically inspired neuron models that span different levels of abstraction, ranging from highly detailed but computationally demanding formulations such as the Hodgkin–Huxley model to simplified and more efficient approximations such as the Leaky Integrate-and-Fire (LIF) model. SNNs encode information through sparse spike events and temporal dynamics, enabling efficient processing of sparse data and naturally supporting temporal coding.
This thesis addresses the challenge of bridging the gap between computational neuroscience and specialized neuromorphic hardware. Specifically, the work focuses on adapting and refactoring a complex SNN model, named *Hippocampal-Cortico Spiking Neural Network (HiCo SNN)* to make it compatible with Intel's neuromorphic hardware Loihi 2.

The HiCo SNN, originally implemented and simulated in Brian2, is based on LIF neurons and is designed to simulate the interaction between hippocampal, perceptual, and neocortical areas encoding semantic and contextual information. Its learning dynamics follow a two-stage process inspired by biological memory consolidation, reproducing both awake and sleep phases. The HiCo SNN was refactored into a modular abstraction compatible with Lava, intel's framework for neuromorphic development. Neuron dynamics were implemented using the adaptive-threshold LIF model, customized to support multiple inputs and to simulate apical contextual signals through an internal projector. Synaptic plasticity was reformulated as a three-factor Spike-Timing-Dependent Plasticity (STDP) rule driven by a neuromodulatory signal, converting the original additive rule into a multiplicative form compatible with hardware constraints. The Sleep phase, initially divided into two sub-phases, was unified into a single biologically consistent process. The final implementation was ported to fixed-point arithmetic, validated on CPU, and deployed for inference on Loihi 2 hardware.

The adapted HiCo SNN was evaluated in three experimental settings: a classical MNIST dataset and two continual learning scenarios, namely class-incremental (Split MNIST) and domain-incremental (Rotational MNIST). In the standard test, the Lava implementation achieved an average accuracy of 82% in floating-point precision and 80.5% in fixed-point precision, while the hardware inference deployment showed similar results to the fixed-point simulation. In the continual learning experiments, the floating-point network obtained average accuracies of 90% on Split MNIST and 78% on Rotational MNIST, whereas the fixed-point version achieved 91% and 78%, respectively. These results are comparable to those obtained with the original Brian2 implementation, with accuracy decreasing at most 4% across the different experiments, indicating limited performance degradation despite hardware-oriented constraints. Overall, this thesis shows the feasibility of executing complex memory consolidation and continual learning mechanisms on neuromorphic hardware, highlighting the potential of such systems for energy-efficient, biologically inspired learning in embedded and robotic applications.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence

CL Continual Learning

DIT Dendritic Integration Theory

DNN Deep Neural Network

HiCo SNN Hippocampal-Cortico Spiking Neural Network

LIF Leaky Integrate-and-Fire

LTD long-term depression

LTP long-term potentiation

SNN Spiking Neural Network

STDP Spike-Timing-Dependent Plasticity

Chapter 1

Introduction

Over the past decade, Artificial Intelligence (AI) has undergone a remarkable transformation driven by the rapid advancement of Artificial Neural Networks (ANNs), which have evolved from early multilayer perceptrons to today's state-of-the-art Deep Neural Networks (DNNs). These models have achieved outstanding performance across a wide range of domains, including computer vision, speech recognition, natural language processing, and autonomous decision-making [1]. Such progress has been enabled by the convergence of several key factors: the availability of massive annotated datasets, the exponential growth of computational power particularly through modern parallel hardware such as GPUs and continuous innovations in network architectures and training algorithms [2].

Despite their impressive performance, conventional deep learning systems exhibit fundamental limitations that hinder their applicability in real world scenarios. Modern deep neural networks (DNNs) typically require large computational resources, high memory bandwidth, and substantial energy consumption during both training and inference. Recent studies have shown that performance improvements are strongly correlated with increases in model size and computational budget [3], while the energy cost associated with training and deploying large-scale models can be significant [4]. This trend has led to extremely high energy demands. For example, training large-scale models such as GPT-3 has been estimated to require on the order of 10^5 kWh, whereas the human brain operates continuously at approximately 12–20 W. This striking contrast highlights a fundamental inefficiency of current AI paradigms when compared to biological intelligence [5]. These requirements make their deployment challenging in real-time environments and in resource-constrained platforms such as mobile devices, autonomous vehicles, drones, embedded systems, and edge-AI infrastructures [6]. Moreover, most deep learning models rely on static training paradigms and struggle to learn continuously from streaming data, often suffering catastrophic forgetting when exposed to sequential tasks [7, 8].

In contrast, biological memory systems exhibit mechanisms that support stable

knowledge accumulation over time [9]. One such mechanism is memory semantization, a process through which initially context-rich episodic memories progressively lose their specific contextual details and are transformed into more abstract semantic representations. This transformation allows the brain to retain generalized knowledge while reducing interference between memories acquired at different times. From a computational perspective, semantization provides a promising principle for continual learning, as it enables the gradual extraction of stable representations from sequential experiences while mitigating catastrophic forgetting [10].

To mitigate these limitations, several approaches have been proposed to reduce network size and computational complexity, including compression techniques such as quantization [11], pruning [12], and knowledge distillation [13]. While these methods can improve efficiency, they do not fundamentally address the architectural mismatch between conventional digital computing systems and the principles of biological neural computation. This observation has motivated the development of a new class of models known as Spiking Neural Networks (SNNs) [14, 2], often referred to as the third generation of neural networks. Unlike traditional ANNs, SNNs communicate through discrete electrical events called spikes and operate in continuous time, enabling sparse and event-driven computation that more closely resembles biological neural processing. By exploiting temporal coding and natural sparsity, SNNs offer significant potential advantages in energy efficiency and latency.

The effectiveness of spiking neural networks (SNNs) is closely tied to neuromorphic computing, a paradigm that seeks to emulate the structure and dynamics of biological neural systems at the levels of both algorithms and hardware. Neuromorphic processors implement massively parallel, asynchronous and event-driven computation, often integrating memory and processing within the same physical substrate. These architectures replicate the key properties of biological neural circuits, offering orders of magnitude improvement in energy efficiency compared to traditional von Neumann systems. They also support real-time processing and biologically plausible learning mechanisms. Examples of such platforms include TrueNorth [15], Loihi [16], SpiNNaker [17], and NeuroGrid [18], which are designed to reproduce essential characteristics of neural computation such as local memory, parallelism, and spike-based communication.

Consequently, neuromorphic computing represents a promising direction for overcoming the bottlenecks of traditional deep learning [5], enabling intelligent systems capable of operating efficiently under strict constraints of power, latency, and computational resources. This approach opens new perspectives for deploying advanced neural models in real-world applications where autonomy, speed, and energy efficiency are critical requirements.

However, deploying complex biologically inspired neural models on neuromorphic hardware remains a challenging task. Many computational neuroscience models are originally developed within flexible simulation environments that allow high numerical precision, unconstrained connectivity, and expressive learning rules. In

contrast, neuromorphic platforms impose strict constraints on numerical representation, memory organization, communication protocols, and plasticity implementation. Bridging this gap between theoretical models and hardware-compatible implementations therefore represents a central challenge for advancing neuromorphic intelligence.

The objective of this thesis is to address this challenge by systematically adapting a hippocampal-cortical spiking neural network model for execution on real neuromorphic hardware. In particular, this work focuses on the HiCo SNN [9], designed to reproduce the process of memory semantization, in which episodic representations gradually evolve into more stable semantic memories through hippocampal–cortical interactions, originally developed within the flexible simulation framework Brian2 [19]. Although such environments provide a highly flexible modeling setting, they do not directly reflect the architectural and computational constraints imposed by neuromorphic processors. For this reason, the original model was carefully redesigned to operate within a hardware-compatible execution environment while preserving its functional behavior and biological plausibility through the neuromorphic framework Lava [20], which was selected to enable deployment on Intel's Loihi 2 neuromorphic processor.

Achieving this objective required modifications across multiple abstraction levels. Neuron dynamics were reformulated for discrete-time execution, synaptic plasticity mechanisms were redesigned to satisfy hardware constraints, architectural components were reorganized to align with neuromorphic communication patterns, and numerical representations were adapted to ensure robustness under reduced precision. Rather than directly porting the original Brian2 model, the development followed a co-design strategy in which algorithmic formulation and hardware requirements were jointly optimized to preserve stability, learning capability, and functional behavior. The main contribution of this thesis is demonstrating that a biologically grounded hippocampal-cortical spiking architecture can be systematically transformed into a neuromorphic-compatible implementation while maintaining its core computational properties. This includes deriving of hardware-compliant multiplicative learning rules, developing both floating-point and fixed-point implementations, performing extensive experimental validation on standard and continual learning benchmarks, and the deployment of the inference phase on Loihi 2 hardware.

Experimental results demonstrate that the adapted implementation preserves the functional behavior of the original model while achieving comparable performance, with only minor accuracy degradation despite the imposed hardware-oriented constraints. These findings provide evidence that complex biologically inspired learning systems can be effectively translated to neuromorphic substrates, supporting the feasibility of deploying structured memory models for energy-efficient continual learning and real-time adaptive intelligence.

The following parts of this thesis are organized as follows. Chapter 2 reviews the

biological and computational principles underlying spiking neural networks, memory consolidation, and neuromorphic computing. Chapter 3 describes the materials, tools, and methods used to redesign and implement the model for neuromorphic execution. Chapter 4 presents the experimental results and evaluates the system across different learning scenarios. Finally, Chapter 5 discusses limitations, implications, and directions for future research.

Chapter 2

Background

The goal of this chapter is to provide the theoretical and methodological foundations necessary to contextualize the proposed work within the broader fields of computational neuroscience, machine learning, and neuromorphic computing. In particular, this chapter first reviews the biological principles that inspire spiking neural networks, including neuronal dynamics, synaptic communication, and plasticity mechanisms. It then introduces the neurocognitive theories of memory organization and consolidation that motivated the design of the reference architecture used in this work.
Building on these foundations, the chapter presents an overview of the hippocampal–cortical model adopted as a starting point, followed by a discussion of continual learning and its associated challenges in artificial systems. Finally, the limitations of conventional artificial intelligence approaches are analyzed, motivating the need for neuromorphic computing paradigms. The chapter concludes with an overview of neuromorphic hardware and software platforms relevant to the experimental implementation described in later chapters.

2.1 Neuroscience Concepts

2.1.1 Biological Neuron

Neurons constitute the fundamental computational units of the nervous system, responsible for processing and transmitting information through electrical signals known as action potentials. A biological neuron is composed of four primary components: dendrites, soma, axon, and synaptic terminals, as illustrated in Figure 2.1. Dendrites act as the input structures of the neuron, receiving chemical signals from other neurons and converting them into electrical signals. These signals are integrated in the soma through a process known as neural integration, which determines whether the neuron reaches the threshold required to generate an action potential. Once initiated, the action potential propagates along the axon, often insulated by a

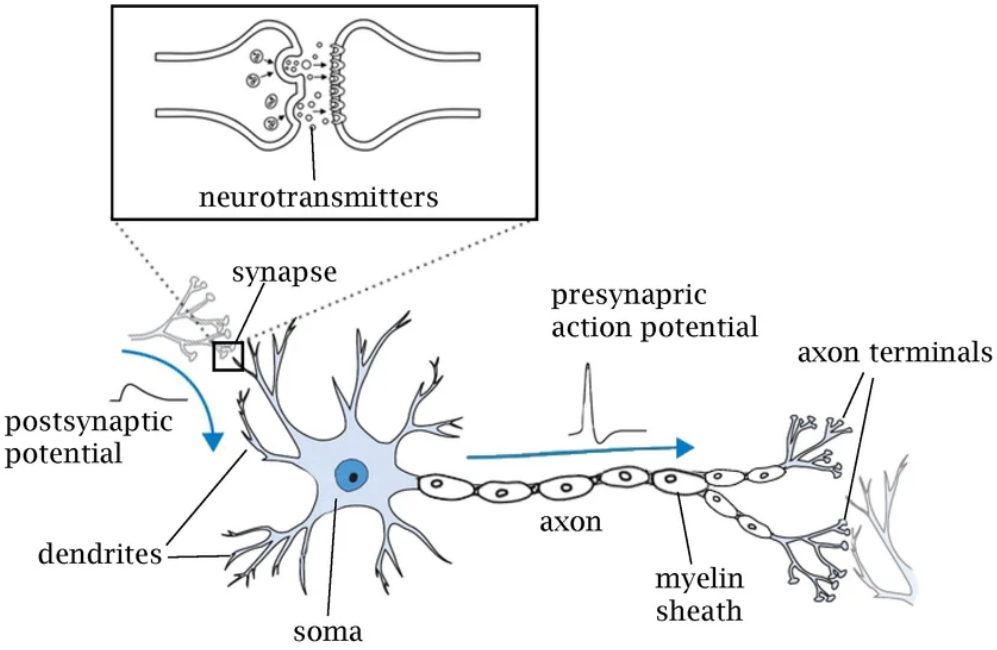

Figure 2.1: Schematic representation of a biological neuron [14].

myelin sheath to prevent signal attenuation, until it reaches the synaptic terminals where communication with downstream neurons occurs. The electrical behavior of neurons is governed by ionic fluxes across the cell membrane, primarily sodium (Na^+), potassium (K^+), and chloride (Cl^-). The resulting voltage difference between the intracellular and extracellular environments defines the membrane potential. At rest, neurons typically maintain a membrane potential of approximately -70 mV, arising from a dynamic equilibrium of ion fluxes through leak channels. When depolarization exceeds a threshold of roughly -55 mV, voltage-gated ion channels open, triggering a rapid influx of Na^+ ions and generating an action potential. This spike lasts about 1 ms, reaches a peak near $+40$ mV, and is followed by repolarization and hyperpolarization driven by K^+ channels, during which a refractory period temporarily inhibits further firing [14, 2].

Neuronal dynamics can be described using mathematical models with varying levels of biological realism. Multicompartment models such as Hodgkin–Huxley [21] or FitzHugh-Nagumo [22] accurately reproduce ionic dynamics but are computationally expensive. At the opposite extreme, abstract rate-based or threshold models prioritize efficiency while discarding temporal structure (Figure 2.2). Intermediate spiking neuron models balance realism and efficiency, capturing essential temporal dynamics and forming the basis of neuromorphic computing approaches.

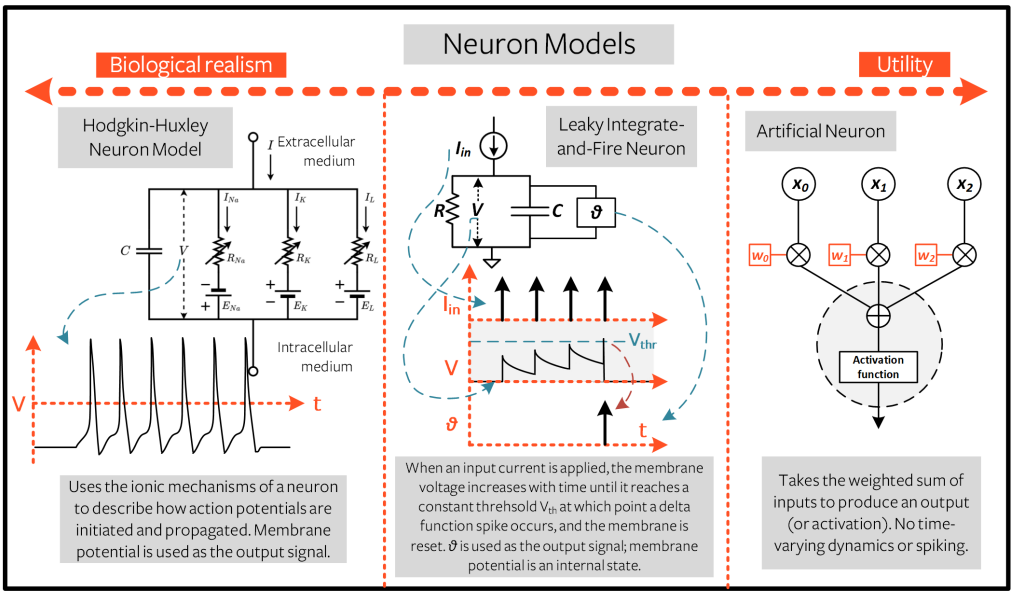

Figure 2.2: Comparison of neuron models with different levels of biological realism. From left to right: the Hodgkin–Huxley model, the Leaky Integrate-and-Fire model, and a standard Artificial neuron. The diagram illustrates the trade-off between biological realism and computational utility [23]

2.1.2 Synapses

Synapses are specialized structures that mediate communication between neurons in a neural network. They are broadly classified into chemical and electrical synapses as depicted in Figure 2.3. Chemical synapses operate through the release of neu-

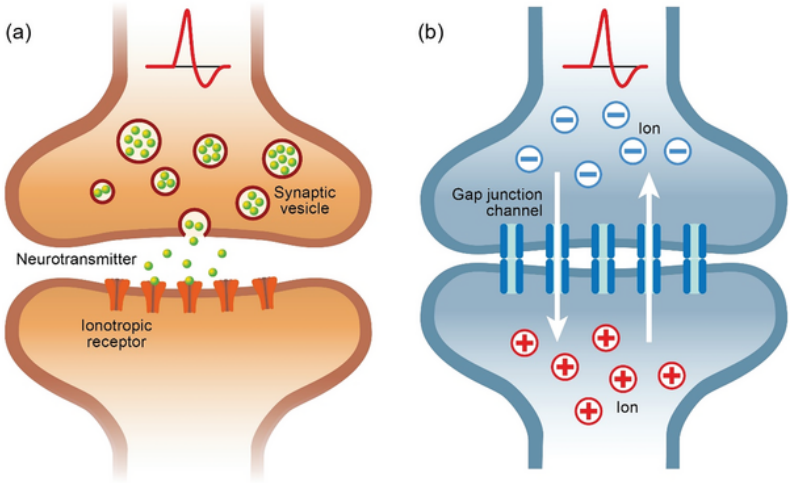

Figure 2.3: Schematic representation of a Chemical synapse (left) and an Electrical synapse (Right) [24]

.

rotransmitters from the presynaptic neuron into the synaptic cleft following the arrival of an action potential. This process is initiated by the influx of calcium ions, which triggers synaptic vesicle fusion with the presynaptic membrane and neurotransmitter exocytosis. Neurotransmitters bind to receptors on the postsynaptic neuron, modifying its membrane potential either directly through ionotropic receptors or indirectly through metabotropic signaling pathways. Although slower than electrical transmission, chemical synapses allow signal amplification and sustained effects [2].

Chemical synapses can be excitatory or inhibitory, depending on their effect on the postsynaptic membrane potential. Excitatory synapses, commonly mediated by glutamate acting on AMPA and NMDA receptors, increase the likelihood of action potential generation. Inhibitory synapses, primarily mediated by γ-aminobutyric acid (GABA) acting on $GABA_A$ and $GABA_B$ receptors, reduce neuronal excitability. Electrical synapses, by contrast, enable direct current flow through gap junctions, allowing rapid but non-amplified signal transmission [14].

Importantly, synaptic efficacy is not static but dynamically modulated by neural activity. Repeated patterns of correlated pre- and postsynaptic firing can induce long-term potentiation (LTP) or long-term depression (LTD), resulting in the strengthening or weakening of synaptic connections. These activity-dependent modifications constitute one of the fundamental cellular mechanisms underlying learning and memory formation [14, 2].

2.1.3 Spike-Timing-Dependent Plasticity (STDP)

Neural circuits exhibit synaptic plasticity, the ability of synapses to change their strength in response to activity. Plasticity is considered a fundamental mechanism underlying learning [14], memory formation, and the adaptation of neural networks to experience. One of the earliest and most influential principles describing this phenomenon is Hebbian learning, which states that the connection between two neurons is strengthened when they are repeatedly activated together [25]. Hebbian learning provides a conceptual framework linking correlated neural activity to long-term changes in synaptic efficacy and serves as the foundation for more temporally precise learning rules.

Building on this principle, Spike-Timing-Dependent Plasticity (STDP) [26] is a temporally asymmetric form of Hebbian learning in which synaptic modifications are driven by the precise relative timing between pre- and postsynaptic spikes. It is widely regarded as one of the fundamental mechanisms underlying learning, memory formation, and the development of neural circuits in biological systems. Unlike rate-based plasticity rules, STDP directly exploits the temporal structure of spiking activity, making it particularly relevant for modeling information processing in spiking neural networks.

The core principle of STDP extends classical Hebbian learning by introducing temporal causality into synaptic updates. When a presynaptic neuron fires shortly before a postsynaptic neuron, the corresponding synapse undergoes long-term potentiation, reflecting a causal contribution of the presynaptic activity to the postsynaptic spike generation. Conversely, when the presynaptic spike follows the postsynaptic one, the synapse is weakened through long-term depression, indicating a lack of causal influence. The magnitude and direction of synaptic change are determined by the relative spike timing and are typically described by an STDP learning window that defines the temporal range over which potentiation and depression occur (Figure 2.4.

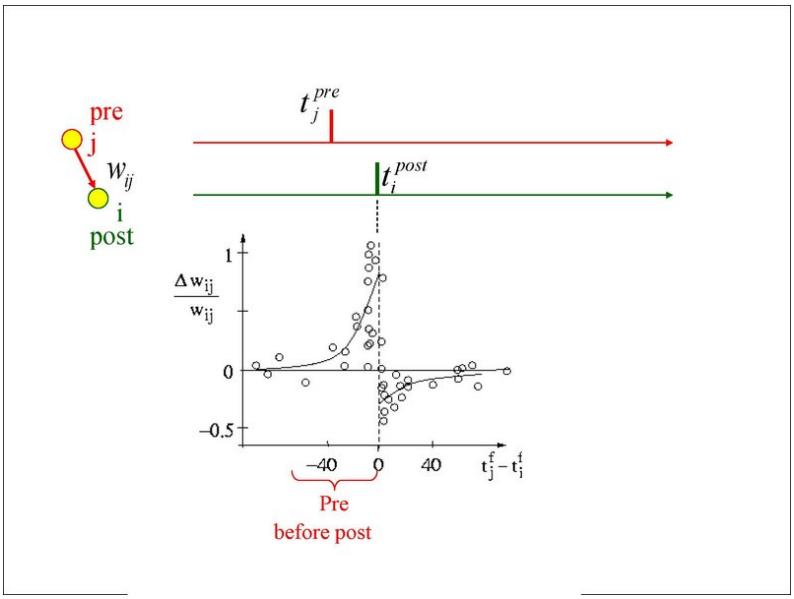

Figure 2.4: Spike-Timing Dependent Plasticity: The STDP function shows the change of synaptic connections as a function of the relative timing of pre- and postsynaptic spikes [26]

At the biological level, STDP is closely associated with intracellular calcium dynamics in the postsynaptic neuron. Brief and high-amplitude calcium transients are generally linked to synaptic potentiation, whereas lower-amplitude but longer-lasting calcium elevations tend to induce synaptic depression. A central role in this process is played by NMDA receptors, which act as coincidence detectors by requiring both presynaptic glutamate release and postsynaptic depolarization to permit calcium influx. This biophysical mechanism provides a direct link between spike timing and synaptic modification.

STDP can be interpreted as a spike-based realization of Hebbian learning and naturally reinforces causal relationships between neural activity patterns. Over time, several biologically grounded extensions of the classical pair-based STDP rule have been proposed. In particular, three-factor learning rules introduce an additional modulatory signal, often associated with neuromodulators such as dopamine,

which gates or scales synaptic plasticity based on behavioral relevance or reward. These mechanisms allow STDP to transition from purely unsupervised learning toward reinforcement-based paradigms. To maintain stability, many models also incorporate homeostatic mechanisms that regulate synaptic strength or neuronal firing rates, preventing unbounded weight growth. Furthermore, experimental evidence suggests that the exact form of STDP is not universal but varies across brain regions, synapse types, and dendritic locations. In cortical pyramidal neurons, for example, synaptic plasticity rules can depend on apical dendritic depolarization and may even invert under specific conditions.

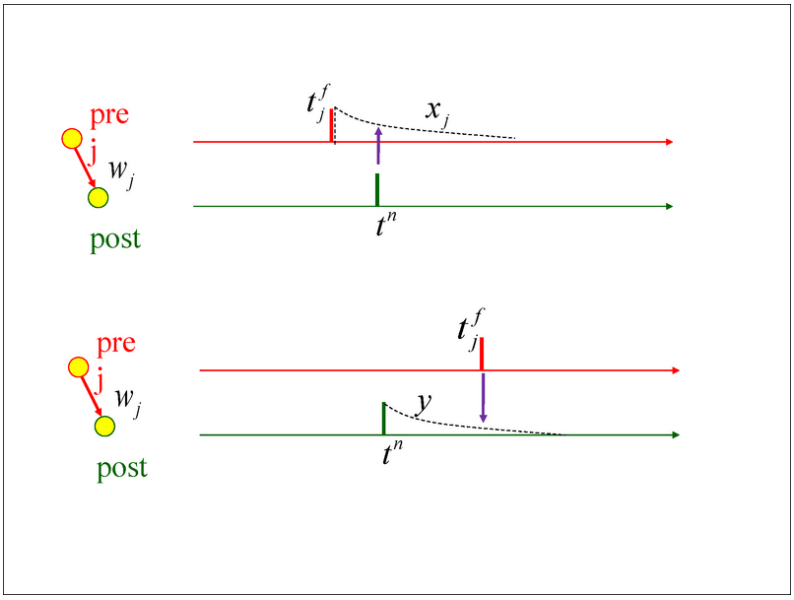

Figure 2.5: Spike-Timing Dependent Plasticity can be implemented by local variables. Top: A presynaptic spike leaves a trace xj(t) which is read out (arrow) at the moment of the postsynaptic spike. The weight change is proportional to that value xj(tn) Bottom: A postsynaptic spike leaves a trace y(t) which is read out (arrow) at the moment of a presynaptic spike [26].

From a computational perspective, STDP is particularly well suited to neuromorphic architectures and Spiking Neural Networks due to its strictly local nature. Synaptic updates rely only on information available at the synapse itself, namely the timing of pre- and postsynaptic spikes, without requiring global error signals or centralized control. STDP is often implemented using pre- and post-synaptic traces-variables that accumulate recent spike activity that allows the synapse to estimate spike timing differences efficiently and apply potentiation or depression accordingly (Figure 2.5). This trace-based approach is especially convenient for hardware implementations on neuromorphic platforms, where learning rules can be executed efficiently in parallel across large numbers of synapses. Despite its strong biological plausibility, achieving feature-learning performance comparable to conventional deep learning methods remains an open challenge, and STDP-based

systems are often augmented with additional mechanisms such as lateral inhibition to promote competition and structured representations.

2.1.4 Memory Consolidation

Several theories have been proposed to explain how memories are consolidated and transformed through interactions between the hippocampus and neocortex (Figure 2.6). The Standard Consolidation Theory [27] posits that newly acquired memories initially depend on the hippocampus to bind distributed cortical representations. Over time, repeated hippocampal reactivation gradually transfers these memories to the neocortex, where they become stabilized and independent of hippocampal retrieval.

In contrast, the Multiple Trace Theory [28] argues that the hippocampus remains permanently involved in the storage and retrieval of detailed episodic memories regardless of their age. Each reactivation creates a new hippocampal trace, while semantic knowledge emerges from overlapping cortical representations. Extending this perspective, the Trace Transformation Theory [29] proposes that memories are not simply transferred but progressively transformed: context-rich episodic memories remain hippocampus-dependent, whereas schematic and semantic representations are gradually established in neocortical circuits. Together, these perspectives suggest a dual memory organization in which episodic detail and semantic abstraction coexist across interacting brain systems.

At the systems level, long-term memory formation relies on coordinated hippocampal–neocortical consolidation processes. Through repeated offline reactivation, mnemonic traces are redistributed and stabilized within cortical networks. Empirical evidence indicates that this process is strongly facilitated during sleep [25], particularly during slow-wave and REM phases, when hippocampal replay of previously learned patterns occurs. Such replay supports synaptic reweighting and promotes the integration of episodic information into more generalized cortical representations. From a computational standpoint, hippocampal–neocortical replay provides a mechanism for transforming experience into structured knowledge, motivating models that incorporate alternating learning phases and offline consolidation dynamics.

At the cellular scale, these processes are supported by the dendritic architecture of cortical pyramidal neurons. These neurons exhibit functionally distinct dendritic compartments that receive different sources of input [31]. Feedforward sensory signals primarily target basal dendrites and somatic regions, whereas feedback or contextual information is conveyed to distal apical dendrites [32].

Interactions between these compartments modulate neuronal output and enable context-sensitive processing. Depending on the behavioral state and neuromodulatory conditions, apical input can amplify sensory-driven activity, directly drive

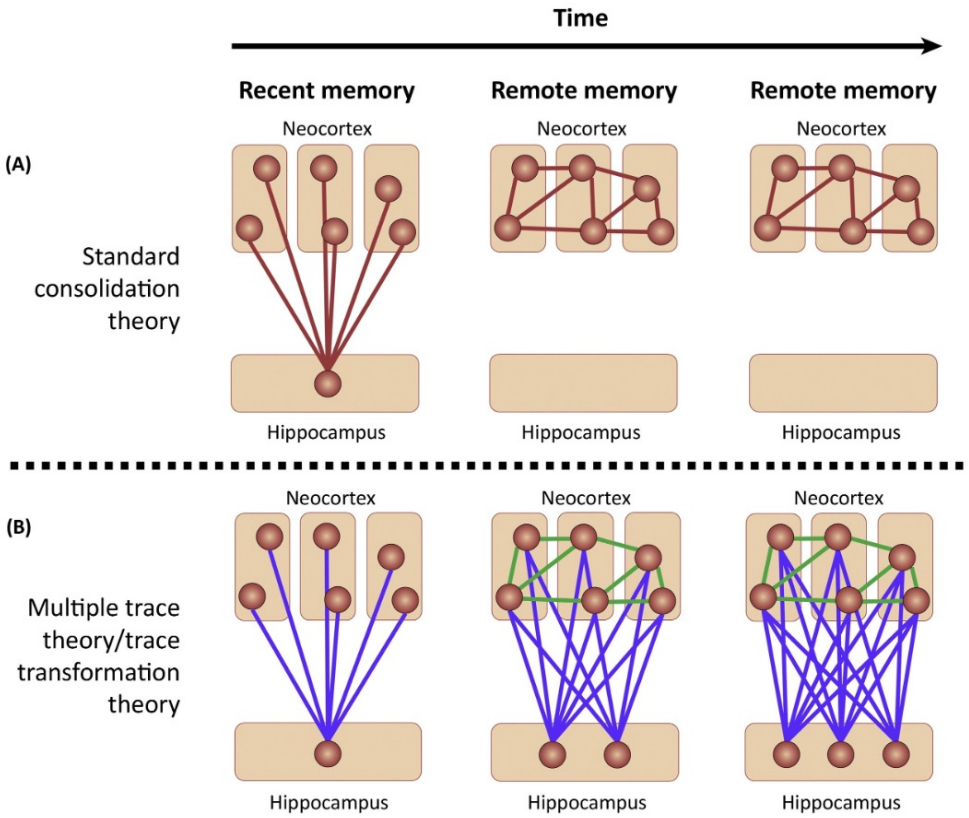

Figure 2.6: Schematic comparison of memory consolidation theories. (A) Standard Consolidation Theory: Recent memories rely on hippocampal-neocortical connections, whereas remote memories become gradually independent of the hippocampus. (B) Multiple Trace Theory / Trace Transformation Theory : Each memory forms multiple traces distributed across the hippocampus and neocortex. Adapted from [30]

somatic spiking, or remain functionally decoupled from the soma. These operating modes are regulated by neuromodulatory systems, including acetylcholine and norepinephrine, which influence dendritic excitability and the coupling between dendritic and somatic compartments.

Taken together, these theoretical, systems, and cellular perspectives outline a coherent multilevel framework for understanding memory organization, linking large-scale brain dynamics with synaptic and dendritic mechanisms. This integrated view provides a principled foundation for computational models that aim to bridge neuroscience and neuromorphic learning systems.

2.1.5 Continual Learning

Continual Learning (CL), also referred to as *incremental learning* or *lifelong learning*, refers to the capability of an artificial system to acquire knowledge from a

continuous stream of data while preserving previously learned information. Unlike traditional machine learning paradigms, which typically assume that training data are independently and identically distributed (i.i.d.), CL systems must operate under non-stationary data distributions [7].

A central challenge in Continual Learning is the well-known *stability–plasticity dilemma*: the system must remain sufficiently plastic to incorporate new information, while being stable enough to prevent the degradation of previously acquired knowledge. In classical neural network architectures, learning a new task often leads to a rapid decline in performance on earlier tasks, a phenomenon known as *catastrophic forgetting*. This occurs because synaptic weights are overwritten during optimization to minimize the loss associated with the new task, effectively erasing prior representations.

Continual Learning Scenarios

In the Continual Learning literature, several incremental learning scenarios are distinguished based on the availability of information about the current task and on how the data distribution changes over time.

- **Task-Incremental Learning (Task-IL):** The model is trained on a sequence of distinct tasks with disjoint output spaces. The Task ID is known both during training and at inference time, allowing the system to select the appropriate output head.
- **Domain-Incremental Learning (Domain-IL):** The task and output space remain unchanged, but the input distribution varies over time, for example due to changes in illumination, viewpoint, or geometric transformations.
- **Class-Incremental Learning (Class-IL):** The model must discriminate among an increasing number of classes over time, without access to the Task ID during inference. This setting is widely regarded as one of the most challenging CL scenarios, as it requires simultaneous class discrimination and task inference.

2.2 Neuromorphic Computing

Neuromorphic computing represents a paradigm shift in the design of computational architectures, grounded in the principles of biological neural systems and inspired directly by the structure and dynamics of the human brain. Rather than treating intelligence as a purely algorithmic problem to be solved on conventional hardware, this approach rethinks computation at the device and architectural level, aiming to emulate neural information processing using brain-inspired representations and mechanisms. As a result, neuromorphic computing has emerged as a

promising framework for implementing machine learning systems that prioritize efficiency, adaptability, and real-time operation [16].

The concept of neuromorphic engineering was originally introduced in the late 1980s by Carver Mead, who proposed the use of analog and mixed-signal integrated circuits to replicate neurophysiological processes by exploiting the intrinsic physics of electronic devices. In its contemporary usage, the term encompasses a broad family of hardware platforms and algorithms inspired by neural computation, spanning different degrees of biological realism. These differences may concern the representation of information, the use of digital or analog substrates, the presence of stochasticity, and the degree of coupling between memory and processing units [14].

A fundamental distinction between neuromorphic systems and conventional von Neumann architectures (Figure 2.7) lies in the way information is represented and processed. While traditional computing platforms rely on deterministic digital logic and binary representations, typically using 32- or 64-bit precision with physically separated memory and processing units, biological brains encode information through sparse patterns of action potentials, or spikes. This spike-based communication occurs in noisy, mixed-signal substrates where computation and memory are intrinsically co-localized, enabling efficient information flow with minimal energy expenditure.

The remarkable efficiency of biological neural circuits constitutes a primary source of inspiration for neuromorphic computing. The human brain is capable of performing complex cognitive tasks at power levels on the order of a few tens of watts, while maintaining millisecond-scale response times. This efficiency is achieved through massively parallel and asynchronous processing, in which neurons operate in an event-driven manner and generate output only when meaningful activity is detected. Spiking Neural Networks (SNNs), which form the computational backbone of most neuromorphic systems, abstract these principles by encoding information in the timing and distribution of spikes, enabling robust and noise-tolerant processing of spatiotemporal signals [2].

Another key advantage of biological computation is the tight integration of memory and processing. In neural tissue, synapses simultaneously store information and participate in computation, thereby eliminating the costly data movement that characterizes von Neumann systems. This architectural feature significantly reduces both latency and energy consumption and is a central design goal of neuromorphic hardware. The overarching objective of neuromorphic computing is to address the fundamental limitations of current AI paradigms. Although Deep Neural Networks have achieved outstanding performance in a variety of domains, their success often relies on large-scale models with prohibitive computational and energy requirements, making them unsuitable for deployment in resource-constrained environments such as mobile devices, autonomous sensors, and edge computing platforms. Neuromorphic systems aim to overcome these constraints by achieving orders-of-magnitude improvements in energy efficiency while retaining competitive

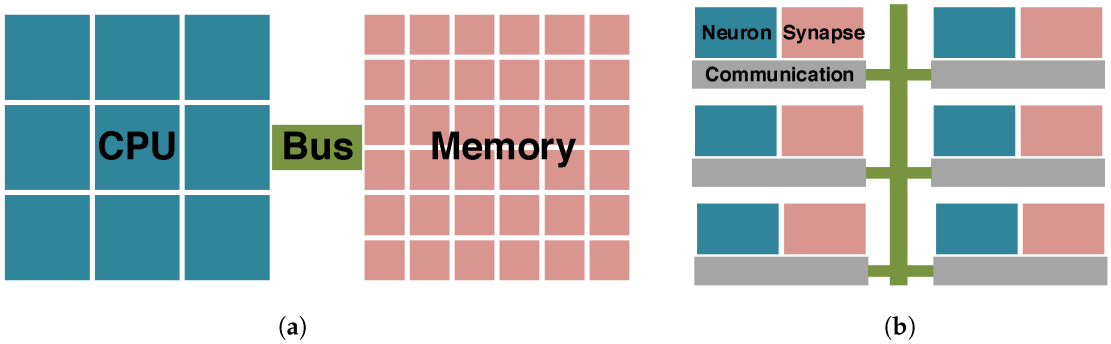

Figure 2.7: Comparison between Von Neumann architecture and neuromorphic architecture [33]

performance.

Beyond efficiency, neuromorphic computing aspires to provide an architectural substrate that is intrinsically suited to intelligent behavior. This includes the ability to learn continuously from experience, adapt to non-stationary environments, and preserve previously acquired knowledge without suffering from catastrophic forgetting. By co-designing hardware and algorithms that more closely reflect the organizational principles of the brain, neuromorphic computing seeks to bridge the gap between artificial and biological intelligence, offering a pathway toward scalable, adaptive, and energy-efficient intelligent systems [7].

2.3 Neuromorphic Hardware: Loihi 2 and the Lava Framework

Neuromorphic computing relies on specialized hardware and software platforms designed to emulate key principles of biological neural systems, including event-driven processing, massive parallelism, and tight coupling between memory and computation. Among existing research platforms, Intel's *Loihi 2* [34, 16] processor and the *Lava* software framework represent a prominent hardware-software ecosystem for the development and deployment of spiking neural networks.

Loihi 2 is a second-generation digital neuromorphic processor that supports configurable neuron and synapse models, asynchronous spike-based communication, highly parallel execution and online learning through a fully programmable learning engine. Its architecture is specifically optimized for sparse, event-driven computation, allowing neural simulations to be performed efficiently while maintaining low power consumption. These characteristics make it particularly suitable for implementing biologically inspired learning systems that would be inefficient on conventional hardware.

Complementing the hardware platform, Lava provides a modular and extensible programming environment for constructing neuromorphic applications. It enables

neural models to be developed, tested, and validated on conventional processors before being deployed on neuromorphic substrates, thereby supporting rapid prototyping and cross-platform experimentation. Its process-based computational abstraction and asynchronous communication model closely mirror the operational principles of neuromorphic architectures.

Together, Loihi 2 and Lava form a co-designed technological stack that bridges neuroscience-inspired algorithms and hardware-efficient execution. In the context of this thesis, they constitute the target platform onto which the proposed hippocampal–cortical network is mapped. A detailed description of their architecture, programming model, and implementation constraints is provided in Chapter 3.

2.4 Hippocampal-Cortico Spiking Neural Network

The Hippocampal-Cortico Spiking Neural Network (HiCo SNN) [9] is a plastic spiking neural network architecture based on Leaky Integrate-and-Fire neurons, designed to model functional interactions between hippocampal and neocortical systems [10]. The primary objective of this model is to reproduce the biological process of memory semantization, through which episodic memories initially encoded in the hippocampus are progressively transformed into stable, abstract representations within cortical structures.

The network dynamics are inspired by the Standard Consolidation Theory and highlight the importance of the *Dendritic Integration Theory (DIT)* [32] as an essential cellular mechanism, leveraging apical amplification processes to modulate neuronal population activity across two distinct operational phases, wake and sleep. During the wake phase, the system focuses on the encoding of novel experiences(Figure 2.8 left), whereas during the sleep phase, hippocampal replay mechanisms drive synaptic consolidation and cortical reorganization (Figure 2.8 right). Through this two-stage operation, HiCo provides a biologically grounded and computationally efficient framework for modeling memory consolidation and knowledge abstraction in spiking neural systems.

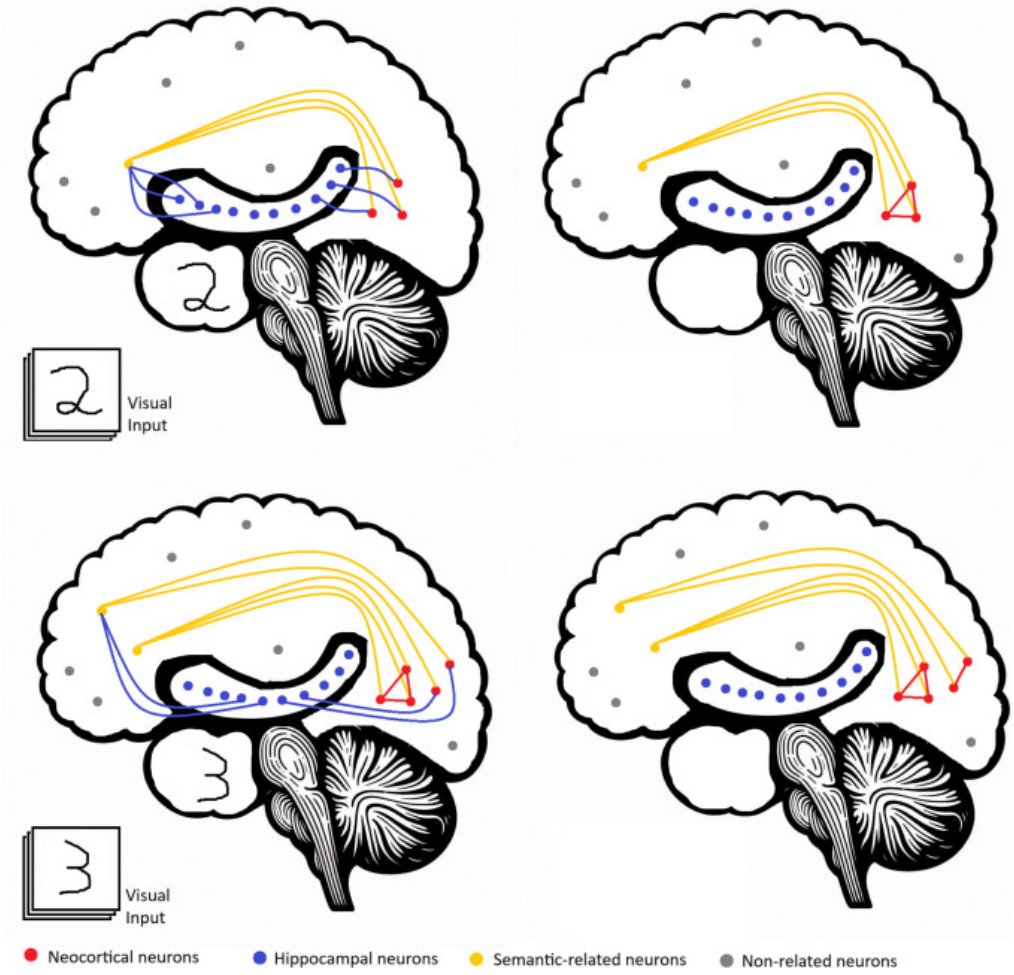

Figure 2.8: On the left are shown neural connections during wakeful learning, while on the right are shown the connections after sleep. Novel stimuli are encoded across multiple brain regions, with the hippocampus binding these elements into episodic memories. During sleep, hippocampal reactivation strengthens connections in other brain regions that share related semantic content [9].

Chapter 3

Materials and methods

3.1 Methodological Overview

The methodological workflow adopted in this work consists of a progressive translation pipeline from a high-level spiking neural network model to a hardware-compatible neuromorphic implementation.

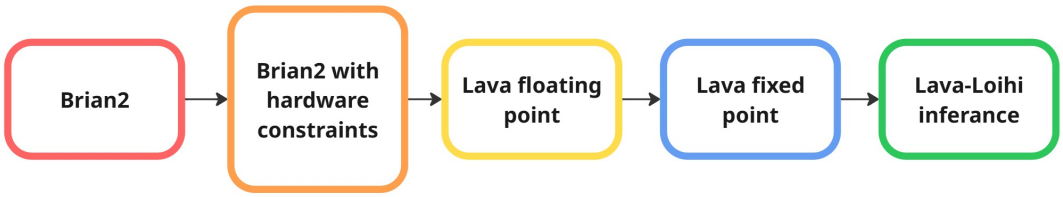

Figure 3.1: Pipeline adopted for the conversion of the original model developed in Brian2 to its Lava fixed-point hardware-oriented version

The process (Figure 3.1) begins with the original Brian2-based formulation of the HiCo architecture, which serves as a reference model. The network is then systematically reformulated to satisfy the computational, numerical, and architectural constraints imposed by the Lava framework and the Loihi 2 neuromorphic processor.

This transition requires modifications at multiple abstraction levels, including neuron dynamics, synaptic plasticity rules, connectivity representations, and numerical precision formats. Each transformation step is validated independently to ensure functional equivalence between the original and adapted models. The final stage consists of the deployment of the inference phase compatible with neuromorphic hardware and verifying that it remains behaviorally coherent.

3.2 Brian2

Brian2 [19] is a scientific simulation framework for spiking neural networks written in Python and available across most computing platforms. It is specifically designed to balance ease of use with high flexibility and extensibility, making it a popular tool for computational neuroscience and biologically inspired modeling.

One of its defining features is its equation-based modeling paradigm, which allows neurons and synapses to be specified directly through systems of differential equations describing their temporal dynamics. A typical example is the leaky integrate-and-fire neuron model, as shown in Equations (3.1) and (3.2)

$$\tau_m \frac{dV}{dt} = -(V - V_{\text{rest}}) + RI(t), \tag{3.1}$$

$$\text{if } V > V_{\text{th}} \text{ then } V \leftarrow V_{\text{reset}}. \tag{3.2}$$

where V is the membrane potential, τ_m is the membrane time constant, V_{rest} is the resting potential, R is the membrane resistance, and $I(t)$ is the input current. This approach enables the implementation of a wide range of models, from simple leaky integrate-and-fire neurons to complex conductance-based systems with adaptive and plastic synapses.

By combining scientific expressiveness with efficient code generation and execution, Brian2 provides a powerful environment for prototyping and validating biologically grounded models, although it does not provide native support for direct deployment on neuromorphic hardware.

3.3 Lava Software Framework

Lava [20] is an open-source software framework for neuromorphic computing developed to facilitate the design, simulation, and deployment of distributed and massively parallel applications on both conventional and neuromorphic hardware. It provides high-level abstractions and tools that enable developers and researchers to construct complex asynchronous systems composed of interacting processes, which can be executed on CPUs, GPUs, and neuromorphic substrates without modifying the core application logic.

At its core, Lava supports the development of neuro-inspired applications by offering a modular, extensible programming model that emphasises composability and hardware agnosticism. Each computational entity in Lava whether representing a neuron, a network, a peripheral interface, or an external algorithmic component is modelled as a stateful process with defined communication channels. This design enables event-based message passing between processes, naturally aligning with the principles of spiking neural computation and asynchronous execution.

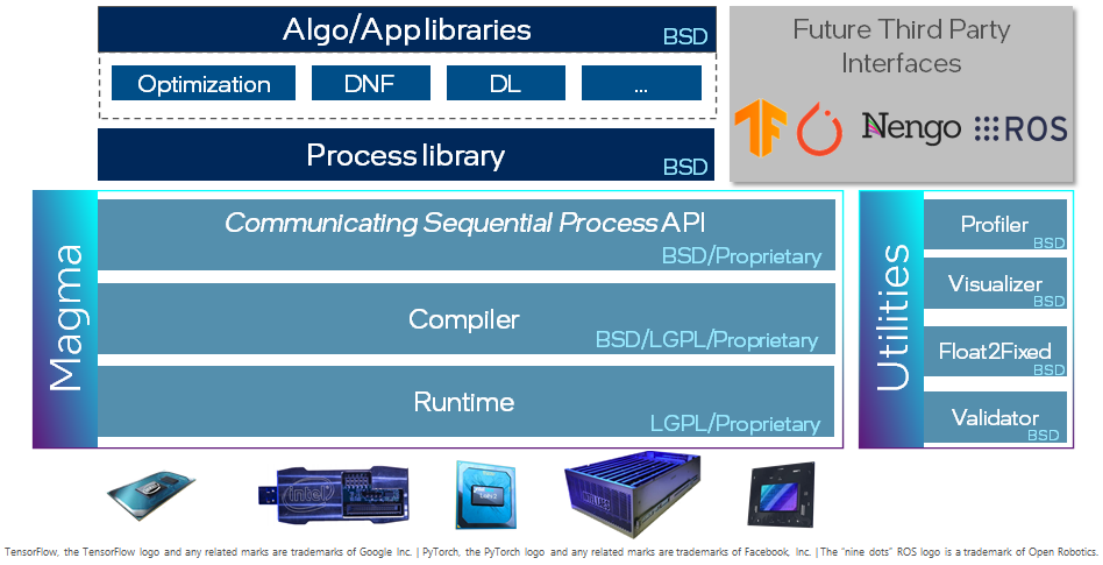

Figure 3.2: high-level overview over the key components of Lava [20].

The framework comprises a collection of libraries and tools that support a broad range of use cases (Figure 3.2), from deep learning and optimization to integration with external ecosystems and robotics platforms. Lava's software stack includes high-level libraries for constructing neural systems, as well as infrastructure for mapping those systems to target execution platforms. Although Lava initially targets Intel's Loihi neuromorphic architectures, its backend abstraction is open for extension to other heterogeneous execution environments, allowing developers to prototype and validate models on conventional processors before deploying them to specialised hardware.

By combining modular abstractions with hardware-aware execution backends, Lava establishes a structured software stack for mapping spiking neural network models onto neuromorphic substrates. This design reduces implementation complexity, supports portability across simulation and hardware targets, and encourages systematic reuse of neuromorphic components in diverse experimental and application scenarios.

3.4 Loihi 2 Neuromorphic Chip

Loihi 2 (Figure 3.3) is a fully digital, many-core neuromorphic processor designed to efficiently execute spiking neural networks using an asynchronous, event-driven computational paradigm [35, 8]. Each chip integrates up to 128 fully asynchronous neuromorphic cores, referred to as neurocores, together with embedded general-purpose processor cores, interconnected through a network-on-chip (NoC) optimized for spike-based communication . All inter-core communication occurs via

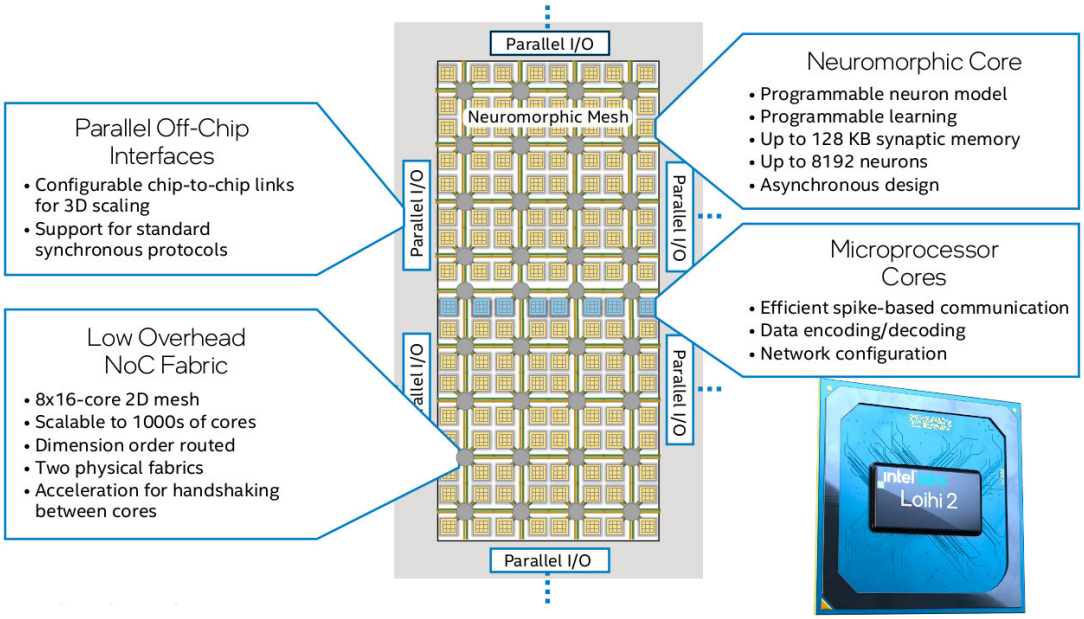

Figure 3.3: Loihi 2 neuromorphic architechture [16]

spike messages, enabling scalable and low-latency information exchange.

The embedded processor cores execute conventional C or Python code and support tasks such as network configuration, monitoring, data encoding and decoding, and system management. Compared to its predecessor [34], Loihi 2 doubles the number of embedded processors per chip, increasing their count from three to six in order to reduce potential bottlenecks associated with conventional processing tasks.

Loihi 2 follows a fully asynchronous design in which neuronal computation and communication are triggered by spike events rather than global clock synchronization. Neuron models are implemented through a programmable microcode pipeline within each neurocore, supporting arithmetic, logical, comparison, and control-flow instructions. This programmability significantly extends the range of neuron dynamics that can be implemented relative to the fixed-function models supported by earlier architectures.

Spike communication is enhanced through graded spike messaging, allowing spikes to carry integer-valued payloads that modulate downstream synaptic weights while preserving sparse, event-driven signaling. Furthermore, Loihi 2 extends on-chip learning capabilities by enabling programmable synaptic update rules with support for generalized third-factor modulatory signals,providing robust support for algorithms ranging from basic Spike-Timing-Dependent Plasticity (STDP) to complex reward-based learning. Loihi 2 also refines memory organization and system scalability. Neurocores allow flexible allocation of local memory between neuron

state and synaptic storage, improving resource utilization.

3.5 Brian2 implementation of the HiCo Network

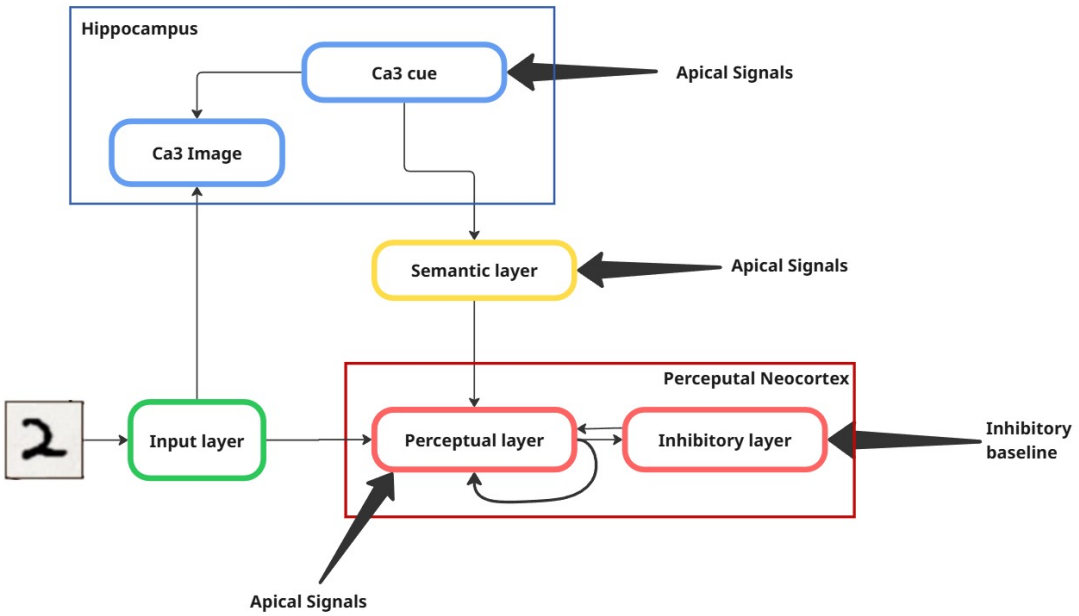

Figure 3.4: Schematic representation of the Hico network,composed of a simplified version of the hippocampus, one excitatory and one inhibitory layer of the perceptual neocortex, an associative network that abstracts other neocortical areas representing neural correlates of semantic and contextual information, and an input layer [9].

As introduced in section 2.4, the **HiCo** architecture simulates the interaction between hippocampal and neocortical structures, focusing on episodic memory encoding, recall, and long-term consolidation. From an implementation perspective, the network is organized into multiple interacting neuronal populations, each representing a specific functional brain area: an input layer, a perceptual neocortex, a hippocampal module (CA3) [36], and a semantic-contextual neocortex.

The ***input layer*** consists of 28×28 spiking neurons, each corresponding to a single pixel of the input image. Pixel intensities are converted into Poisson-distributed spike trains at 63.5Hz during stimulus presentation emulating Gamma waves associated with states of maximum concentration, higher cognition, information processing, memory, and profound insights.

The core part of the architecture, the ***Perceptual Neocortex***, is composed of excitatory and inhibitory neurons and implements competitive dynamics for sparse memory allocation; Excitatory neurons represent perceptual features of the input.

Their number determines the memory capacity of the network. This work adopts an extreme economical regime with 1 excitatory neuron per experience. On the other hand the inhibitory population is set to one quarter of the excitatory neurons, reflecting realistic cortical ratios [37] and enabling Winner-Take-All (WTA) dynamics.

The Hippocampal region follows the same dimensionality of the perceptual neocortex and is composed of other two parts, ***CA3image*** that encodes the sensory representation of the experience. Neurons in this population are connected one-to-one with the input layer and ***CA3cue*** that acts as an index for episodic memories. Each CA3cue neuron is associated with a specific CA3image pattern and can reactivate it during offline replay.

The last piece of the architecture is the ***Semantic-Contextual Neocortex*** that represents category-level or contextual information, implemented as an array of unconnected excitatory spiking neurons where each neuron corresponds to a semantic category and its role is to associate episodic memories with semantic labels or contextual information.

3.5.1 Network States and Functional Phases

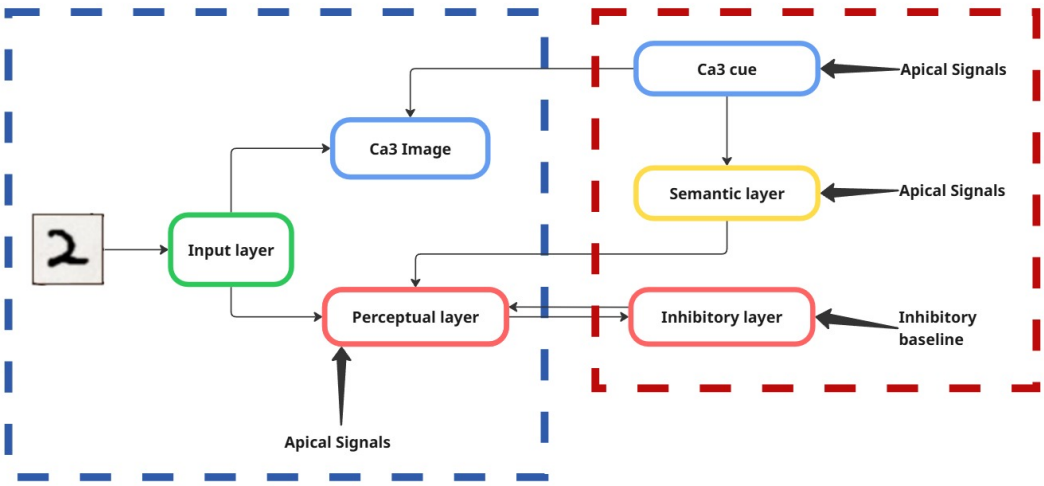

Figure 3.5: **Network architecture during training.** Overview of the model employed in the encoding phase. Highlighted in purple Input presentation during encoding. Sensory input is delivered as Poisson spike trains to both the hippocampal (CA3 image) and perceptual layers. Highilitged in orange: Role of apical signals. Apical projectors excite neuron targets in perceptual, semantic, and Ca3 cue layers, enabling selective learning and supporting winner-take-all dynamics.

The network operates in two distinct functional states, an ***Awake State (Encoding)*** (Figure 3.5) where sensory inputs are presented as Poisson spike trains,

the Hippocampal CA3 circuits rapidly encode episodic memories and competitive dynamics in the perceptual neocortex ensure selective memory allocation. This competitive dynamicas are made possible thanks to apical signals (projectors),a neuron in the perceptual layer and hippocampus (CA3 cue) are excited to learn a particular experience, and in the same way, apical signals excite a neuron in the semantic layer to link that experience to a specific label. Finally, a mechanism similar to apical signals (inhibitory baseline) serves to excite all neurons in the inhibitory layer to facilitate a "winner takes all" dynamic that allows only the preported neuron to learn that particular experience.

In the ***Sleep State (Consolidation)*** (Figure 3.6) reactivations of CA3cue neurons triggers an hippocampal replay, then the reactivated patterns drive cortical reorganization and semantic integration and a homeostatic mechanisms stabilize activity and synaptic strength. At this stage, to allow memory consolidation in the perceptual layer, recursive plastic synapses are activated. Note that during this phase, no external input is present.

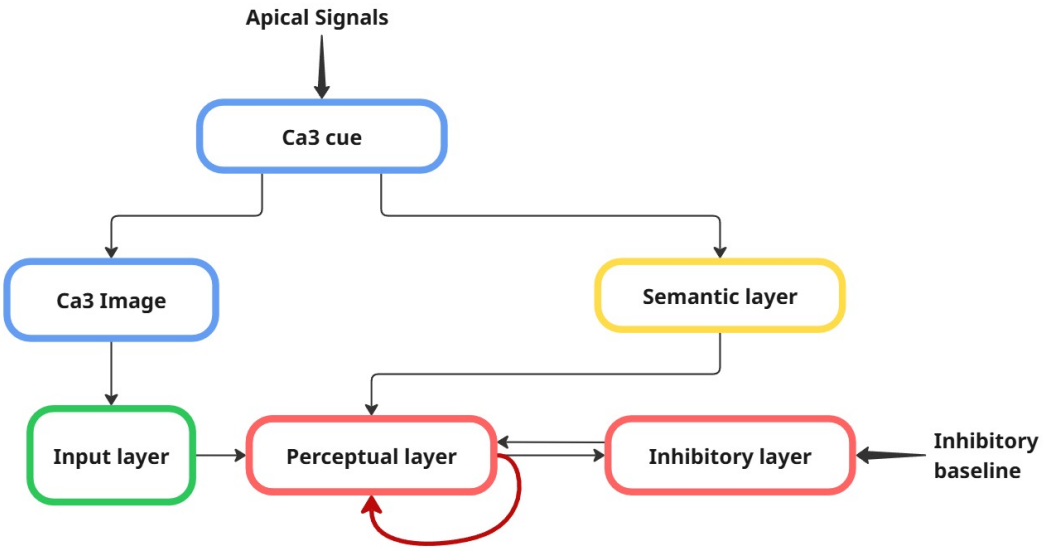

Figure 3.6: **Network architecture during Sleep.** Overview of the model employed in the Consolidation phase.Red arrow: recursive plastic synapses

3.5.2 Neuron Models

All neurons in the network (perceptual, hippocampal, and semantic) are modeled starting from a **Leaky Integrate-and-Fire (LIF)** formulation. The membrane potential v evolves according to 3.3

$$\tau_m \frac{dv}{dt} = -(v - V_0) + R_m I(t), \tag{3.3}$$

where $I(t)$ represents the total synaptic and modulatory input current, V_0 is the resting membrane potential, τ_m is the membrane time constant, and R_m is the membrane resistance. When the membrane potential reaches the threshold θ, the neuron emits a spike and the membrane potential is reset to V_{reset}.

Neurons in the perceptual neocortex constitute the most complex elements of the architecture (Figure 3.7). These neurons receive five distinct inputs: four afferent synaptic currents originating from the modeled brain areas and an additional voltage signal provided by external apical projectors. The total synaptic current is therefore defined as in 3.4

$$I_{\text{tot}} = I_{\text{exc}} + I_{\text{inh}} + I_{\text{stim}} + I_{\text{sem}} \tag{3.4}$$

In addition to the standard LIF dynamics, perceptual neurons implement **Spike Frequency Adaptation (SFA)** through an adaptive firing threshold. The threshold dynamics are described by 3.5

$$\tau_\theta \frac{d\theta}{dt} = -(\theta - \theta_0), \tag{3.5}$$

where θ_0 represents the baseline threshold value and τ_θ the adaptation time constant. Upon spike emission, the threshold θ is increased by a fixed increment b, after which it gradually relaxes back to θ_0 according to Eq. (3.5).

A refractory mechanism is also implemented to prevent spike emission for a fixed number of discrete timesteps following each spike, ensuring biologically plausible inter-spike intervals. Together, spike-frequency adaptation and refractoriness promote sparse activity, prevent runaway excitation, and support competition between neuronal representations.

Neurons in the hippocampal, semantic, and inhibitory populations adopt a simplified formulation. In these layers, the neuronal dynamics follow the basic LIF model augmented only with the refractory mechanism, without the additional adaptive threshold and compartmental input structure used in the perceptual neurons.

3.5.3 Learning Rules

STDP rules are expressed directly within the synaptic equations through the synaptic traces `pre`, `post`, and the neuromodulatory signal `m`, whose dynamics are event-driven. Synaptic updates follow an *additive formulation*, meaning that weight changes are independent of the current synaptic weight value. Stability is enforced through clipping, which constrains weights within a fixed interval after each update.

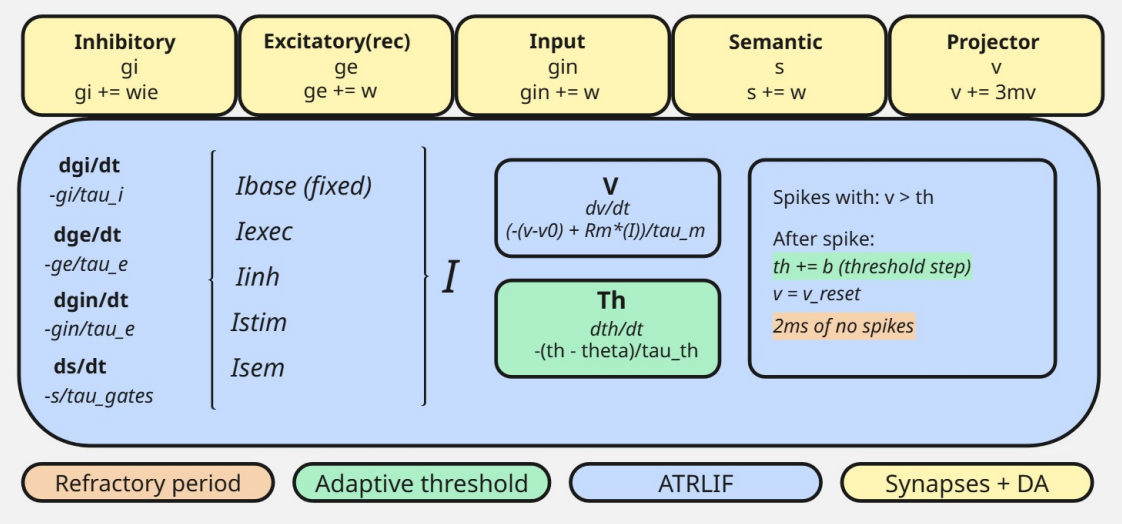

Figure 3.7: **Perceptual neuron model.** Schematic representation of the perceptual layer neuron and its synaptic compartments.

The plasticity mechanisms implemented in the model can be grouped into three main STDP-based learning rules, summarized in Table 3.1.

The coexistence of multiple plasticity mechanisms reflects a key principle of biological learning systems, in which heterogeneous synaptic rules operate concurrently across different brain regions and functional pathways. This diversity enables the interaction between local spike-based learning and global modulatory control, a hallmark of cortical and hippocampal circuits involved in memory consolidation and semantic abstraction [25, 31].

3.5.4 Apical signals (Projectors) and inhibitory baseline

In the original Brian2 implementation of the HiCo network, apical signals (projectors) and the an inhibitory baseline were modeled as external Poisson neuron populations connected to their target layers through static synapses. As mentioned in section 3.5.1 these populations emulate contextual top-down inputs that modulate the activity of cortical neurons through an additional excitatory drive applied to the membrane potential.

Three distinct projector populations are defined in the model, each targeting a specific neural population. and a inhibitory baseline that have a similar behaviour in term of implementation, Table 3.2 summarizes the projectors, with their target neurons, rates, and weights.

The inhibitory baseline targets the entire population and not just a single neuron, maintaining a minimum level of inhibitory activity in the network and contributing to the stabilization of overall dynamics.

Table 3.1: Plasticity mechanisms implemented in the model.

Three-factor STDP with tar (rate of disconnection)	
Layers involved	Input → Perceptual
Equation	$\Delta w = \begin{cases} -a_{2\,\min}\,\text{post}\,\text{plastic} & \text{(on\_pre)} \\ +a_{3\text{plus}}(\text{pre}\,m - \text{tar})\,\text{plastic} & \text{(on\_post)} \end{cases}$
Notes	Neuromodulated plasticity rule used for memory allocation. The term tar promotes competition by weakening neurons with low activation.
Three-factor STDP	
Layers involved	CA3 cue → CA3 image, CA3 cue → Semantic, Semantic → Perceptual
Equation	$\Delta w = \begin{cases} -a_{2\,\min}\,\text{post}\,\text{plastic} & \text{(on\_pre)} \\ +a_{3\text{plus}}(\text{pre}\,m)\,\text{plastic} & \text{(on\_post)} \end{cases}$
Notes	Neuromodulated plasticity rule controlled by the signal m, which depends on the global brain state (e.g., awake or sleep).
Classical STDP	
Layers involved	Perceptual → Perceptual
Equation	$\Delta w = \begin{cases} -a_{2\,\min}\,\text{post}\,\text{plastic} & \text{(on\_pre)} \\ +a_{3\text{plus}}\,\text{pre}\,\text{plastic} & \text{(on\_post)} \end{cases}$
Notes	Standard spike-timing-dependent plasticity relying only on pre- and post-synaptic spike timing correlations.

From a biological perspective, this mechanism is inspired by the dendritic organization of cortical pyramidal neurons, where distal apical dendrites receive feedback and contextual information originating from higher cortical areas. These signals do not simply provide direct excitation but modulate neuronal responsiveness, influencing how feedforward sensory inputs are integrated at the soma.

So in the HiCo architecture, projector activity directly contributes to the membrane potential V of the target neurons, effectively simulating this apical modulation. Functionally, these signals act as a contextual bias that facilitates the co-activation of neurons representing related perceptual, hippocampal, and semantic information. This mechanism enables the network to implement a form of coincidence detection between bottom-up sensory input and top-down contextual signals, promoting selective activation and stabilizing learning dynamics within the distributed hippocampal-cortical memory system.

Table 3.2: Summary of Poisson populations used to drive the network.

Poisson pop.	**Target layer**	**Neurons**	**Rate (Hz)**	**Synaptic weight**
U_{perc}	Perceptual	400	15	3 mV
U_{cue}	CA3 cue	400	15	1 mV
Λ_{sem}	Semantic	200	10	0.5 mV
inh-baseline	Inhibitory	800	7.5	0.4 mV

3.6 HiCo Architecture: From the Brian2 Simulator to the Lava Neuromorphic Framework

To provide a structured overview of the architectural and functional modifications introduced during the transition from Brian2 to the Lava neuromorphic framework, Table 3.3 summarises the main differences discussed in this chapter.

Table 3.3: Summary of the main architectural and algorithmic modifications introduced when porting HiCo from Brian2 to Lava.

Perceptual neuron complexity	
Brian2	Four input currents with independent decay constants plus one voltage-based projector input.
Lava	Three input currents with unified decay constant, reducing hardware complexity while preserving behaviour.
STDP learning rules	
Brian2	$w \leftarrow \text{clip}(w + a_{3+} \cdot \delta,\ 0,\ W_{\max})$
Lava	$w \leftarrow w + a_{3+}(W_{\max} - w)\delta$
Supervised learning	
Brian2	External Poisson populations providing voltage-based apical excitation.
Lava	Internal stochastic membrane excitation embedded in the neuron model, avoiding external spike transmission.
Sleep phase	
Brian2	Two sequential stages: hippocampal reactivation followed by cortical consolidation.
Lava	Single integrated replay process where reactivation and consolidation occur simultaneously.

3.6.1 Neuron Models

The perceptual layer neuron neuron continues to constitute the most computationally complex unit of the HiCo architecture. In the original Brian2 formulation, each perceptual neuron received five distinct inputs originating from functionally inspired brain regions. Four of these were current-based signals, while a fifth acted directly in voltage mode. Specifically, the *input layer* modulated the excitatory conductance g_{in}, inhibitory feedback and recurrent cortical signals targeted the excitatory conductance g_e, and the *semantic layer* influenced a gating variable s. The fifth signal, termed the *projector*, acted directly on the membrane potential V.

In the neuromorphic implementation within Lava, the two excitatory conductances (g_{in} and g_e) are merged into a single effective excitatory input while preserving equivalent dynamics. Furthermore, the projector mechanism is implemented *intrinsically* within the neuron model itself. Instead of relying on an external process, apical modulation is dynamically applied to selected neurons according to discrete simulation steps, improving computational efficiency and ensuring compatibility with the Loihi 2 execution model.

Under this formulation, the perceptual neuron continues to follow the adaptive-threshold leaky integrate-and-fire (LIF) formulation as seen in Eq. (3.3) and (3.5). The Total current instead is now composed only by three componets:

$$I_{\text{tot}} = I_{\text{exc-stim}} + I_{\text{inh}} + I_{\text{sem}}$$

Inhibitory, hippocampal, and semantic neurons remain simplified variants of the perceptual neuron model in which the projector mechanism is implemented internally within the neuron and their membrane dynamics follows the same LIF formulation. In hippocampal (CA3image and CA3cue) and semantic layers, the projector provides a selective excitatory modulation targeting individual neurons as in the perceptual layer, supporting associative activation. In contrast, inhibitory neurons receive a diffuse projector signal that excites the entire population, maintaining a baseline level of activity in the inhibitory layer. All these neurons operate with a fixed firing threshold and a refractory mechanism, without adaptive threshold dynamics.

3.6.2 STDP Learning Rules

In the original HiCo implementation, synaptic plasticity relied on an *additive* weight update rule combined with an explicit *clipping mechanism* that constrained synaptic weights within a predefined interval. The update was defined as in 3.6

$$w \leftarrow \text{clip}(w + a_3^+ \cdot \delta,\, 0,\, W_{\max}), \tag{3.6}$$

where a_3^+ denotes the potentiation learning rate, δ represents the activity-dependent plasticity signal derived from pre- and post-synaptic activity (and, when

present, neuromodulatory factors), and $W_{\max}$ defines the maximum allowed synaptic weight. The clipping operator ensures that weights remain within the interval $[0, W_{\max}]$ after each update.

While this formulation works naturally in the Brian2 simulation environment, it is incompatible with one of the key constraints imposed by the Lava framework. In particular, Lava requires learning rules to be expressed in a *sum-of-products* form and does not support explicit clipping operations in custom plasticity mechanisms.

To address this limitation, the learning rule was reformulated in a multiplicative form that introduces an implicit upper bound on the synaptic weights as in equation 3.7

$$w \leftarrow w + a_3^+ (W_{\max} - w)\delta. \tag{3.7}$$

In this formulation, the factor $(W_{\max} - w)$ progressively reduces the magnitude of the update as the synaptic weight approaches its maximum value. As a result, weight growth naturally slows down near $W_{\max}$, effectively preventing divergence without requiring an explicit clipping operation. This modification preserves the qualitative behavior of the original learning dynamics while satisfying the structural constraints required by the Lava learning framework.

This transformation is non-trivial and substantially alters the underlying weight dynamics. In the absence of clipping, synaptic weights can drift toward undesired regimes, potentially becoming negative or exceeding the effective operational range of the synapse. As a result, the reformulation of the learning rules required not only a syntactic adaptation but also an extensive rebalancing of multiple hyperparameters governing learning stability, convergence, and competition among synapses. These adjustments were essential to preserve the functional behavior of the original model while ensuring compatibility with neuromorphic execution constraints.

3.6.3 The Projector and inhibitory baseline Mechanisms

In the original Brian2 implementation, projector signals and inhibitory baseline were realised as external Poisson neuron populations (neuron groups with a stochastic firing threshold) connected to the target neurons through static synapses. As discussed in subsection 3.6.1, in the Lava implementation this mechanism has been internalised directly within the neuron process models, removing the need for additional network objects.

Each neuron layer maintains an internal simulation-step counter t, which increments at every timestep. After an initial delay of $t_0 = 10$ steps, the projector becomes active and selects exactly one neuron within the layer to receive input. Given the layer size N and a configurable steps_per_target (the number of consecutive steps the projector spends on the same neuron) variable, the target index

is computed as in Equation 3.8

$$i_{\text{target}} = \left\lfloor \frac{t - t_0}{\text{steps\_per\_target}} \right\rfloor \bmod N. \tag{3.8}$$

This mechanism causes the projector to cycle through all N neurons in a round-robin fashion, spending steps_per_target timesteps on each neuron before moving to the next.

At every active timestep, the projector draws a spike count from a Poisson distribution with rate parameter

$$\lambda = 400 \cdot r_{\text{proj}} \cdot \frac{\Delta t}{1000}, \tag{3.9}$$

where r_{proj} is a configurable rate factor and Δt is the integration timestep expressed in milliseconds. The sampled spike count $k \sim \text{Poisson}(\lambda)$ is multiplied by the projector strength s_{proj} and added directly to the membrane voltage of the selected neuron:

$$v_{i_{\text{target}}} \leftarrow v_{i_{\text{target}}} + k \cdot s_{\text{proj}}. \tag{3.10}$$

The same approach has been implemented for the inhibitory baseline, but in this case, the entire population was targeted rather than a single target neuron.

This approach reproduces the effect of the original external Poisson population while keeping the entire projector logic self-contained within the neuron model.

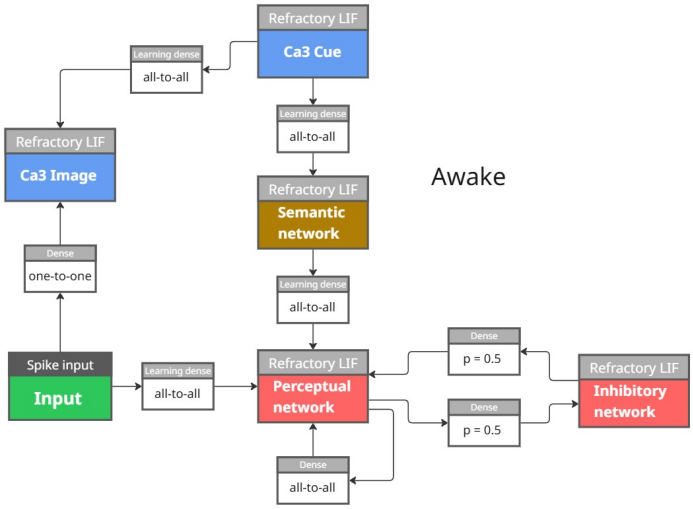

Figure 3.8: Schematic representation of the Awake network in Lava components. During this phase sensory input is delivered as Poisson spike trains to both the hippocampal (CA3 image) and perceptual layers, then, apical projectors excite neuron targets in perceptual, semantic, and Ca3 cue layers, enabling selective learning and supporting winner-take-all dynamics.

3.6.4 Network States and Functional Phases

One of the most significant modifications introduced during the Lava adaptation concerns the execution of the *sleep phase*. In the original implementation, this process was divided into two distinct stages: a first *reactivation phase*, during which neurons in the *CA3cue* layer were reactivated to elicit the memory traces previously learned by the *CA3image* layer; and a subsequent *consolidation phase*, where hippocampal activity was used as input to the *perceptual layer* to reinforce and stabilize the encoded memories. In the revised Lava-based architecture, these two stages were merged into a single integrated process. Reactivation and consolidation now occur simultaneously, as the hippocampal structures are progressively reactivated and directly drive the perceptual network within the same continuous cycle. This modification allows the system to accumulate reactivation patterns over time, producing a smoother and more biologically consistent replay dynamic. This unified formulation reduces execution overhead while preserving the functional role of hippocampal replay in memory consolidation. Furthermore, modeling replay as a continuous process rather than a strictly sequential one aligns more closely with experimental observations of spontaneous hippocampal–cortical reactivation during natural sleep.

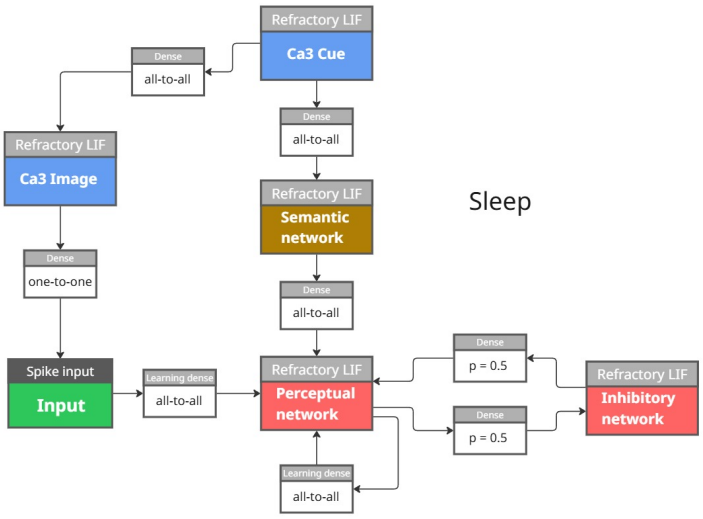

Figure 3.9: Schematic representation of the Sleep network in Lava components. During this phase reactivations of CA3cue neurons triggers an hippocampal replay, then the reactivated patterns drive cortical reorganization and semantic integration and a homeostatic mechanisms stabilize activity and synaptic strength.

From a purely functional point of view, the network has been redesigned for each functional state, for example by replacing LearningDense, i.e., plastic synapses governed by specific learning, with simple Dense (static synapses) where plasticity

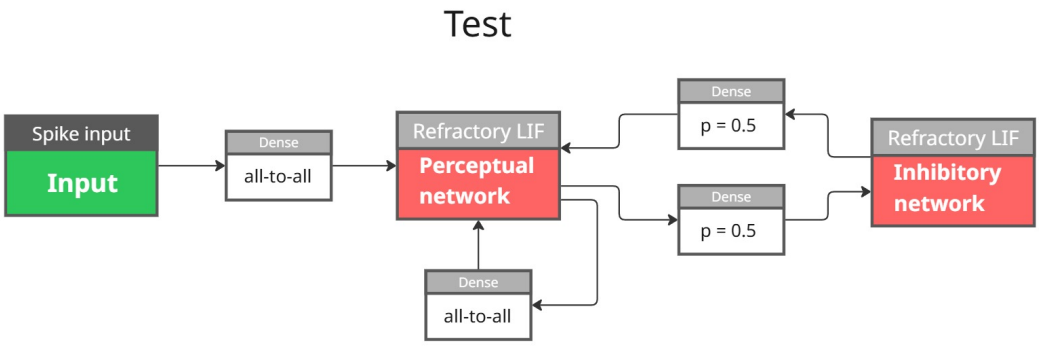

Figure 3.10: Schematic representation of the Test network in Lava components

is not necessary, in order to lighten the computational load, as Dense are much faster. Figures 3.8, 3.9, and 3.10 show a schematic representation of the network in terms of Lava components, optimized for each functional state.

3.6.5 Parameter Rescaling

A significant difference concerns the rescaling of parameters between the two platforms. Table 3.4 summarises the main changes.

These rescalings were necessary to reconcile the continuous-time dynamics of the original model with the discrete-time execution and numerical constraints imposed by the Lava framework and Loihi 2 hardware.

In addition to amplitude rescaling, a structural simplification was introduced in the treatment of temporal constants. While the original Brian2 implementation employed distinct time constants for membrane, inhibitory, and gating dynamics, the Lava implementation adopts a uniform decay constant ($\tau = 2\,\text{ms}$) across these processes, as shown in Table 3.4.

Table 3.4: Parameter comparison between Brian2 and Lava implementations.

Parameter	**Brian2 (Original)**	**Lava**
$a_{3\text{plus}}$	0.0065	0.065
$a_{2\,\text{min}}$	0.00071	0.0071
$\tau_{m,I}$	10 ms	2 ms
τ_{gates}	114 ms	2 ms
τ_i	5 ms	2 ms
$W_{\text{max},h}$	110	20

This choice is motivated by the discrete-time formulation used in neuromorphic

hardware. In continuous time, each dynamical variable follows an equation of the form

$$\tau_x \frac{dx}{dt} = -x + F(t), \tag{3.11}$$

where different processes may evolve on different temporal scales. In contrast, hardware-oriented implementations approximate these dynamics as

$$x[t+1] = \alpha x[t] + F[t], \tag{3.12}$$

with $\alpha = \exp(-\Delta t/\tau_x)$ implemented in fixed-point arithmetic.

Supporting multiple independent decay factors would require additional multipliers, configuration registers, and memory accesses, increasing computational overhead.

Moreover, heterogeneous decay constants in low-precision arithmetic may introduce quantization inconsistencies and cumulative numerical errors over long simulations. A uniform decay improves numerical robustness and facilitates calibration.

Although distinct time constants provide greater biological realism, the consolidation dynamics investigated in this work primarily depend on population-level interactions rather than on fine-grained temporal differentiation. Therefore, enforcing a common decay constant represents a principled trade-off that preserves the qualitative behavior of the network while ensuring compatibility with discrete-time execution and hardware constraints.

3.7 Floating-Point vs Fixed-Point implementations

The Lava framework supports two distinct execution modes corresponding to different stages of development and deployment: a *floating-point* simulation mode and a *fixed-point* execution mode. The floating-point mode emulates neuron and synapse dynamics using standard double-precision arithmetic and is primarily used for algorithm development, debugging, and validation. In contrast, the fixed-point mode constrains all computations to integer arithmetic combined with bit-shift operations that mirror the hardware datapath of the Loihi 2 neuromorphic processor. This mode is required for deployment on neuromorphic hardware and enables accurate prediction of the behaviour of the system when executed on Loihi 2 silicon.

In the floating-point implementation, all state variables and parameters are represented using 64-bit double precision. In the fixed-point implementation, state variables are encoded as 24-bit signed integers, input ports use 16-bit integers, and intermediate computations are carried out in 64-bit integers to avoid overflow before values are shifted back to the 24-bit range. The representable dynamic range is therefore bounded by

$$-2^{23} + 1 \leq x \leq 2^{23} - 1. \tag{3.13}$$

To preserve numerical resolution in the integer representation, voltage-related quantities are scaled by a constant factor referred to as the *voltage unity*

$$U_v = 2^6. \tag{3.14}$$

All membrane potentials, thresholds, and related parameters are multiplied by this factor during initialisation, ensuring proper alignment of the most significant bits in the fixed-point representation. For example, a resting potential of $-70\,\mathrm{mV}$ is represented internally as $-70 \times 2^6 = -4480$ in the integer domain.

Both implementations model synaptic conductances using exponential decay dynamics, but the arithmetic differs significantly. In the floating-point simulation, the conductance update follows the standard exponential formulation

$$g_{\mathrm{new}} = g_{\mathrm{old}} \exp\left(-\frac{dt}{\tau}\right) + I_{\mathrm{input}}. \tag{3.15}$$

In the fixed-point implementation, the exponential decay is approximated using integer arithmetic. A decay constant δ is pre-computed as

$$\delta = \left\lfloor \left(1 - e^{-dt/\tau}\right) \cdot 2^{12} \right\rfloor, \tag{3.16}$$

where 2^{12} defines the scaling factor used for the fixed-point representation of decay coefficients. The update rule then becomes

$$g_{\mathrm{new}} = g_{\mathrm{old}} - \left(\frac{\delta\, g_{\mathrm{old}}}{2^{12}}\right) + I_{\mathrm{input}}. \tag{3.17}$$

Since

$$2^{12} - \delta \approx e^{-dt/\tau} \cdot 2^{12}, \tag{3.18}$$

this formulation is equivalent to

$$g_{\mathrm{new}} \approx \left\lfloor \frac{g_{\mathrm{old}}(2^{12} - \delta)}{2^{12}} \right\rfloor + I_{\mathrm{input}}, \tag{3.19}$$

which approximates the continuous exponential decay. The division by 2^{12} is implemented as a right bit-shift, enabling an efficient hardware realization. The resulting approximation introduces at most a one least-significant-bit quantisation error per update step.

In the floating-point model, the membrane voltage dynamics follow the standard conductance-based formulation

$$\tau_m \frac{dv}{dt} = -(v - v_0) + R_m I_{\mathrm{total}}, \tag{3.20}$$

where v denotes the membrane potential, v_0 the resting potential, τ_m the membrane time constant, and I_{total} the sum of all synaptic current contributions.

Using an explicit Euler discretisation with timestep dt, the update equation becomes

$$v_{t+1} = v_t + \frac{dt}{\tau_m}\left(-(v_t - v_0) + R_m I_{\text{total}}\right). \tag{3.21}$$

This formulation is straightforward to implement in floating-point arithmetic, where multiplications by fractional coefficients such as dt/τ_m can be performed with high numerical precision.

In the fixed-point implementation used for Loihi 2 execution, the same dynamics must instead be reproduced using integer arithmetic only. Directly implementing the Euler update above would require repeated multiplications by fractional coefficients, which would rapidly lead to precision loss in integer arithmetic. For this reason, the update is reformulated using a pre-computed decay coefficient that approximates the exponential solution of the membrane equation.

The decay coefficient is defined as

$$\alpha_v = \left\lfloor (1 - e^{-dt/\tau_m}) \cdot 2^{12} \right\rfloor. \tag{3.22}$$

Using this coefficient, the membrane potential update can be written as

$$\hat{v}_{t+1} = \hat{v}_t + \left\lfloor \frac{\alpha_v}{2^{12}}\left(-(\hat{v}_t - \hat{v}_0) + R_m I_{\text{total}}\right) \right\rfloor, \tag{3.23}$$

where $\hat{v}$ denotes the scaled integer representation of the membrane potential. The division by 2^{12} is implemented as a right bit-shift operation, matching the arithmetic primitives available in the Loihi 2 hardware datapath.

For sufficiently small timesteps, the exponential coefficient satisfies the approximation

$$1 - e^{-dt/\tau_m} \approx \frac{dt}{\tau_m}, \tag{3.24}$$

which implies

$$\frac{\alpha_v}{2^{12}} \approx \frac{dt}{\tau_m}. \tag{3.25}$$

Consequently, the fixed-point update is mathematically equivalent to the Euler discretisation used in the floating-point simulation, up to a small quantisation error introduced by integer truncation. Because typical simulation parameters satisfy $dt \ll \tau_m$, both formulations produce nearly identical membrane dynamics while enabling efficient hardware execution in the fixed-point case.

Plastic synapses follow the same conceptual learning rule in both implementations, but additional scaling is required in the fixed-point case to maintain numerical

resolution. In particular, synaptic trace impulses are amplified by a factor of 16 to prevent rapid quantisation to zero during integer decay operations. The resulting scaling is compensated in the weight update expression, ensuring that the effective learning dynamics remain consistent with the floating-point model.

To compensate for the integer scaling introduced in the learning rule, the maximum synaptic weights are rescaled during synapse initialisation. In particular, cortical synapses use a larger scaling factor than hippocampal synapses so that the effective weight range remains comparable after the trace amplification and the 2^{-7} scaling applied in the weight update equation.

Finally, the adaptive firing threshold of Perc neurons follows an exponential relaxation toward its baseline value in both implementations. In the floating-point model, this behaviour is implemented through an explicit Euler discretisation of the corresponding differential equation. In the fixed-point implementation, the same dynamics are approximated using integer arithmetic with a pre-computed decay constant and right-shift operations. Because the simulation timestep satisfies $dt \ll \tau_{th}$, the exponential and Euler discretisations produce nearly identical dynamics. Additionally, the spike-triggered threshold increment is scaled according to the voltage unity factor so that threshold updates remain consistent with the scaled voltage representation.

3.8 Validation Strategy

To ensure correctness of the hardware-adapted model, validation was performed at multiple levels. First, neuron-level dynamics were compared between Brian2 and Lava implementations to verify numerical consistency. Second, synaptic updates and plasticity traces were monitored to confirm functional equivalence of learning rules. Third, network-level behaviour was evaluated by comparing firing statistics, memory allocation patterns, and task performance metrics. Only after functional equivalence was confirmed in floating-point simulation the model has been converted to fixed-point representation and matching the constraint imposed for Loihi 2 hardware.

All neuron models are implemented as subclasses of `PyLoihiProcessModel` that implements the same phases of execution as the Loihi 1/2 processor within the Lava framework. Each neuron exposes typed input and output ports for synaptic currents and spike events, ensuring compatibility with the Loihi 2 execution model. For example, the perceptual neuron defines ports such as

```
gin_ge_in, gi_in, s_in → s_out
```

and maintains internal state variables including membrane potential V, adaptive threshold θ, synaptic variables, gating variable, and internal counters controlling refractory dynamics and spike emission.

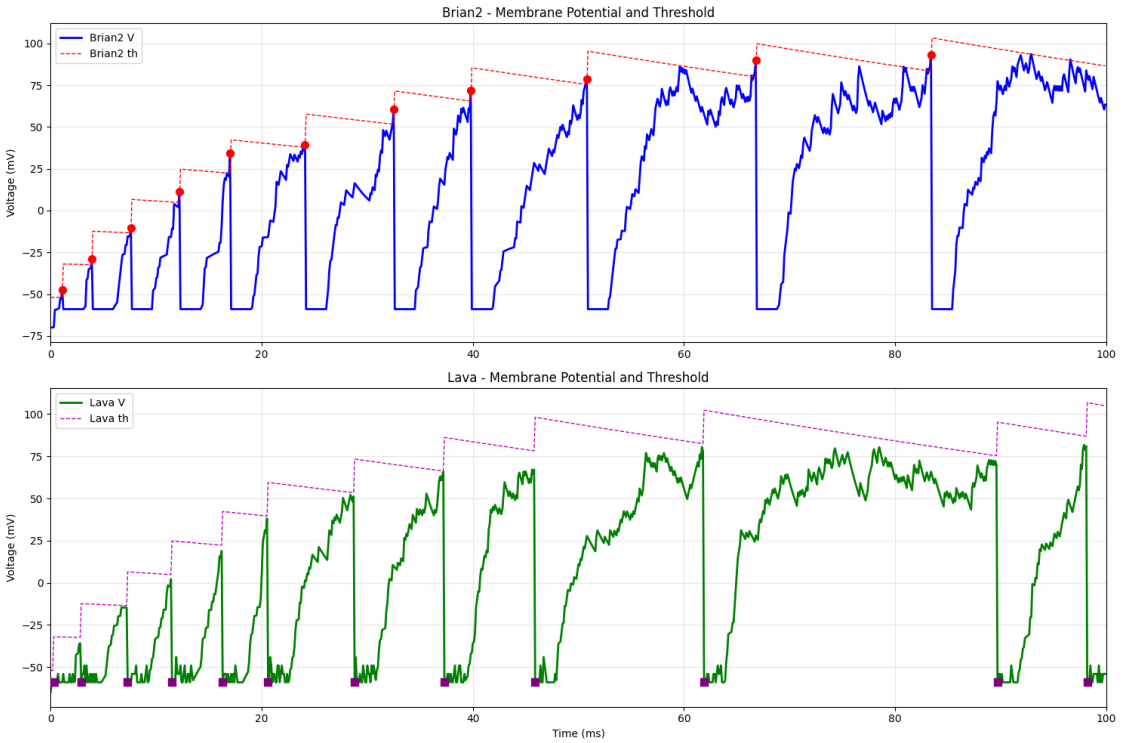

Figure 3.11: Membrane potential and threshold comparison, from Brian2 to Lava

The first level of validation was carried out by testing the individual components separately to verify that, given the same input, they behaved in a similar way to their Brian2 counterparts. For this reason, the version originally modeled in Brian2 was first compared with its counterpart modeled in Lava with floating point arithmetic. As can be seen from the graph in Figure 3.11, apart from a few differences due to the poissonian nature of the projectors, the neuron behaves in the same way in terms of both spikes emitted and threshold adaptation in both models.

As with neurons, synapses were also compared in their Brain2 and Lava versions. In fact, as can be seen from the graphs in Figure 3.12, which represent the evolution of pre, post, and m(neuromodulatory signal) synaptic traces and the actual evolution of weight, the general trend for the same input was almost identical.

Looking at a behavioral comparison of the entire architecture, it can be seen in Figure 3.13 that the weight distribution after the two states of awake and sleep is very similar in both implementations, confirming that the Lava network is able to replicate fairly faithfully what is implemented in Brian2,

The same applies to representations learned from the network and represented by superimposing the weights of the input-perceptual layer synapses for each label. In fact, as can be seen in the Figure 3.14, the representation learned for the class

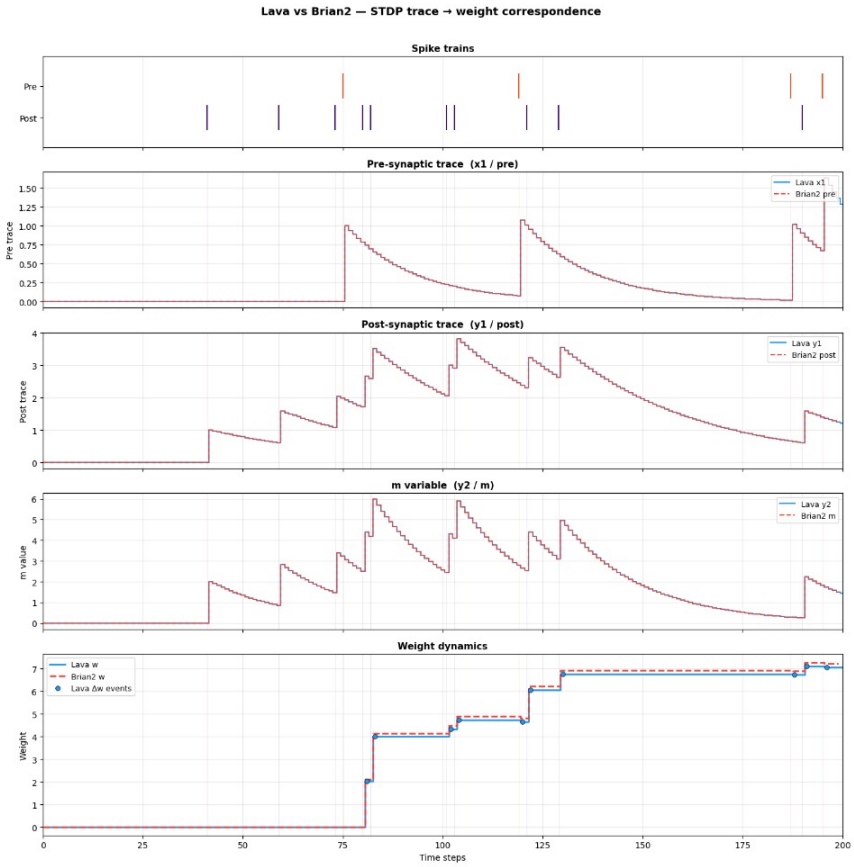

Figure 3.12: Stdp rule for input-perceptual synapses,comparison from Brian2 to Lava

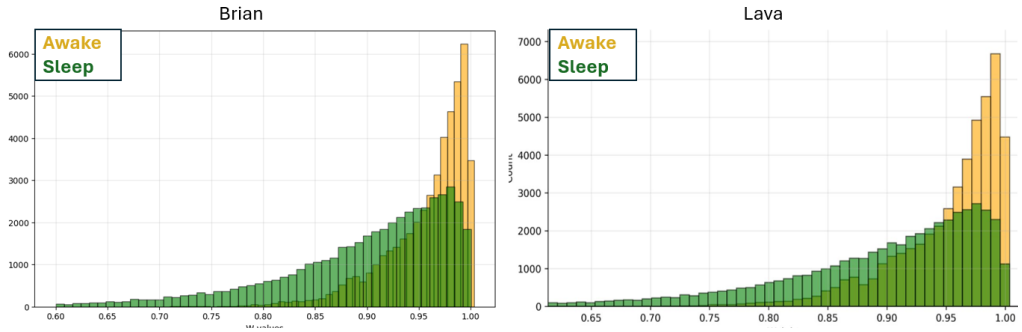

Figure 3.13: Comparison of Input→Perc weight distributions before and after sleep in floating-point simulations. Both Brian2 and Lava implementations

zero, is practically identical in all four the network implementations.

To better analyze the discriminative capability of the network, a pairwise classification accuracy for all class combinations as been computed. Figure 3.15 shows the resulting accuracy matrices for the three considered implementations: Brian

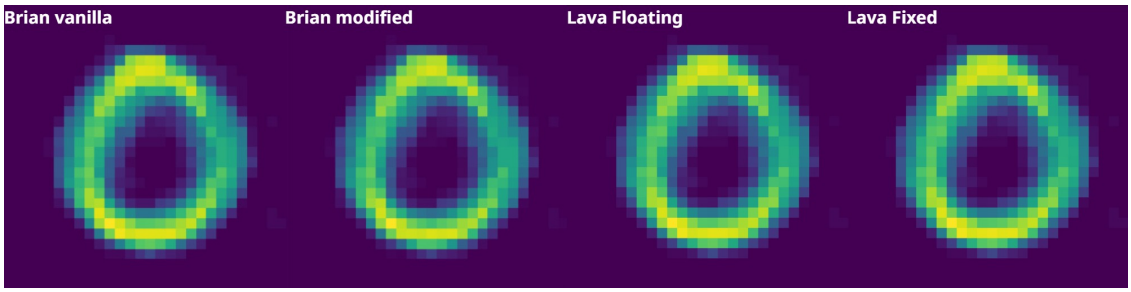

Figure 3.14: Learned representations across the four different network implementatios,from letf to right, Brian2 vanilla, Brian2 modified with multiplicative learning rules, Lava floating-point and Lava fixed point.

(Vanilla), Brian (Mod), and Lava (Fixed).

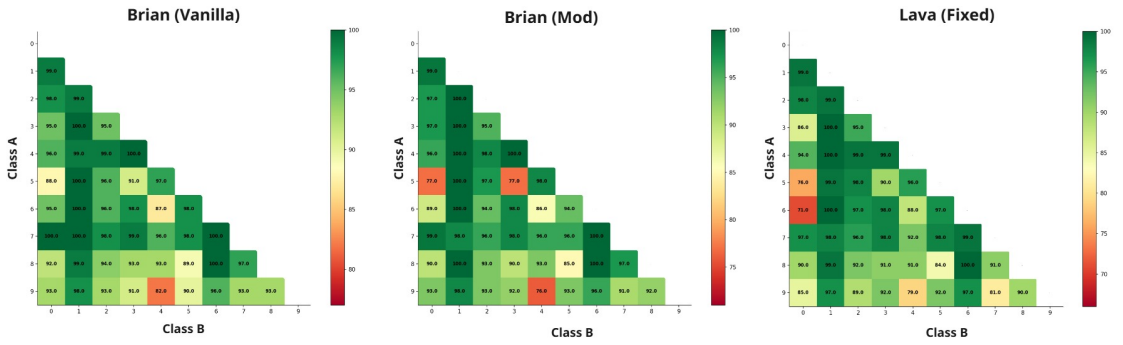

Figure 3.15: Pairwise classification accuracy matrices for the Brian (Vanilla), Brian (Mod), and Lava (Fixed) implementations.

Each matrix element (i, j) represents the classification accuracy obtained when the model is required to distinguish between class i and class j. The diagonal elements are not defined since they correspond to comparisons of identical classes. The color map provides an intuitive visualization of the classification performance, where green values indicate high accuracy and red values indicate more difficult class separations.

This analysis allows us to identify which pairs of classes are more challenging for the model and to evaluate the consistency between the different implementations. Overall, the Lava implementation reproduces the qualitative behavior observed in the Brian simulations, with comparable accuracy values for most class pairs.

Finally, to define an inference protocol on Loihi 2, a minimal network was implemented consisting of a Poisson spiker running on a CPU used to present inputs for inference, a microcode model of the perceptual layer neuron running on Loihi together with the pre- and post-neuron synapses, and a spike collector also implemented on the CPU to derive the predicted label, as shown in Figure 3.16.

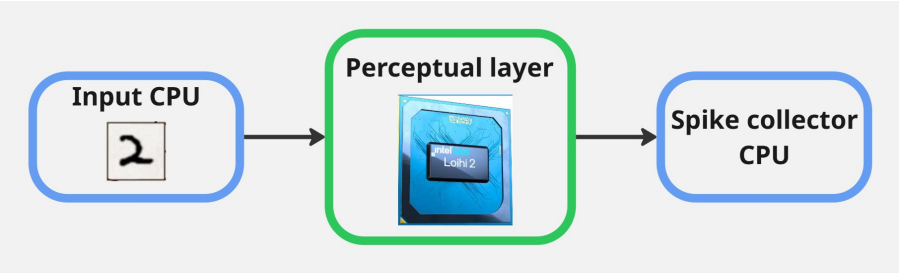

Figure 3.16: Schematic of the implemented inferance net on Loihi 2

Algorithm 1 presents the pseudocode describing the functioning of the perceptual layer neuron implemented in microcode, following the contraints and restricted instruction set of the Loihi 2 architecture.

Algorithm 1 High-level description of the hico neuron microcode implementation

▷ **Conductance update**
Read synaptic input and update excitatory conductance
$g_{in} \leftarrow (g_{in} \cdot decay_{ge}) >> 12 + input_{syn}$
▷ **Synaptic current computation**
$temp \leftarrow v$
$temp \leftarrow (temp \cdot g_{in}) >> 6$
$temp \leftarrow (temp \cdot Rm_{scaled}) >> 12$
$I_{exc} \leftarrow -temp$
▷ **Membrane potential integration**
if $refrac\_cnt == 0$ **then**
    $drift \leftarrow -(v - v_0)$
    $rhs \leftarrow drift + I_{exc}$
    $dv \leftarrow (rhs \cdot decay_v) >> 12$
    $v \leftarrow v + dv$
end if
▷ **Refractory countdown**
if $refrac\_cnt > 0$ **then**
    $refrac\_cnt \leftarrow refrac\_cnt - 1$
end if
▷ **Spike generation**
if $refrac\_cnt == 0$ AND $v \geq \theta_0$ **then**
    $v \leftarrow v_{reset}$
    $refrac\_cnt \leftarrow refractory$
    Emit spike
end if

Chapter 4

Results and Discussion

This chapter presents a systematic evaluation of the neuromorphic hardware adaptation of the HiCo spiking neural network, with the goal of assessing whether the hardware-compatible implementation preserves the functional behaviour of the original model while maintaining stable learning dynamics. Rather than analysing individual components in isolation, the discussion is guided by empirical evidence obtained from synaptic statistics, reconstruction patterns and quantitative performance metrics. Together, these perspectives provide a unified view of how learning, consolidation, and recall emerge from the interaction between plasticity rules and network dynamics.

All experiments were conducted using MNIST [38] stimuli preprocessed through deskewing based on statistical image moments and binarisation with a threshold of 0.5 applied to normalized pixel intensities. Active pixels were converted into Poisson spike trains with maximum firing rate of 63.5 Hz, ensuring a stochastic yet structured input representation.

4.1 Single Task - Classical MNIST

As mentioned in the previous chapter, the first level of validation concerns synaptic statistics. The distributions of Input→Perc weights provide a direct indicator of how learning modifies internal representations. As shown in Figure 3.13, training produces a strong concentration of weights toward high values, reflecting potentiation induced by repeated stimulus exposure. After sleep, however, the distribution becomes broader and shifts toward intermediate values. This redistribution suggests that sleep does not merely preserve previously acquired information but actively reshapes synaptic structure, effectively performing a normalization process that prevents saturation while retaining discriminative information. Importantly, the overlap between curves produced by the reference and hardware-compatible

simulations indicates that the Lava-based implementation reproduces the statistical behaviour of the original model despite operating under different numerical constraints.

A complementary perspective is provided by the fixed-point implementation shown in Figure 4.1. Although discretization limits numerical resolution, the qualitative behaviour remains unchanged: post-sleep weights exhibit the same spreading effect observed in floating-point simulations. This result indicates that consolidation dynamics depend primarily on structural relations among synapses rather than on precise numerical values. The preservation of these statistical signatures confirms that the sleep mechanism is intrinsically robust and compatible with hardware-oriented precision constraints.

Figure 4.1: Comparison of Input→Perc weight distributions before and after sleep in fixed-point simulations.

This interpretation is reinforced by examining reconstructed input patterns derived from synaptic weights. Figure 4.2 shows the representation of digit zero before and after sleep. Before consolidation, the reconstruction appears less uniform and slightly noisy, with locally biased activations that reflect specific training samples. After sleep, the representation becomes smoother and more globally coherent, forming a more regular structure. This qualitative change suggests that replay during sleep integrates information across multiple experiences, effectively performing a statistical averaging process that preserves invariant features of the digit while reducing sample-specific noise.

In Figure 4.3 are shown the connections of the neocortical neurons after the sleep phase, showing a strong correlation among neurons belonging to the same class.

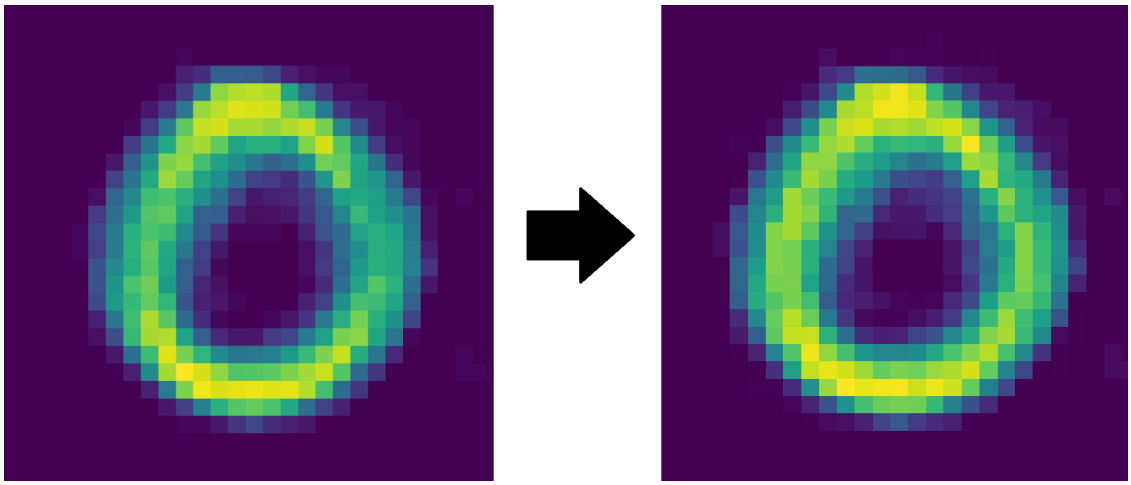

Figure 4.2: Reconstruction of digit zero from Input→Perc weights before and after sleep.

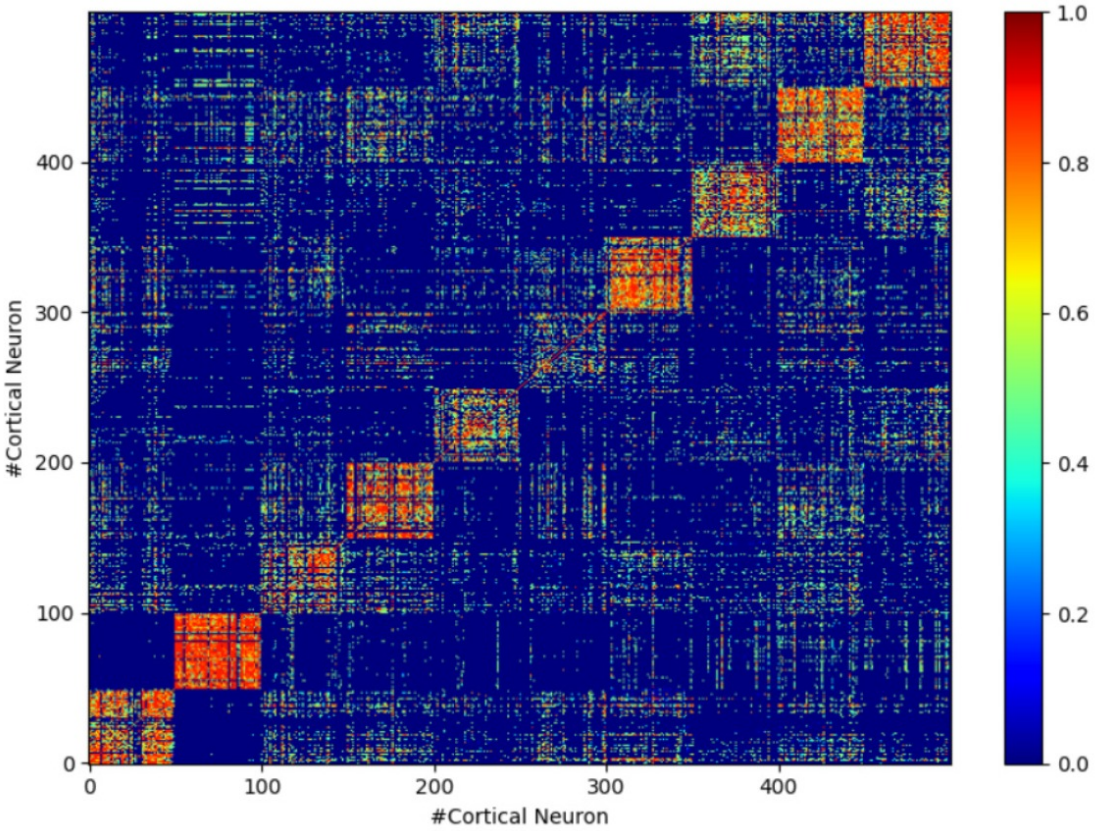

Figure 4.3: Connections of neocortical neurons after the sleep phase.

Quantitative Evaluation. To complement qualitative observations, performance was evaluated using classification accuracy under different decoding strategies. In

the average decoding scheme, the predicted class corresponds to the neuronal population whose neurons exhibit the highest average spike count. In addition, neuron-based decoding strategies were evaluated by selecting the neuron with the highest spike count and assigning the corresponding class label. The same approach was extended to top-k evaluations, where the prediction is considered correct if the target class is associated with one of the k neurons exhibiting the highest spike counts (top-3 and top-5). Table 4.1 reports the accuracy in the average decoding strategy being the overall best, measured after training and after sleep and averaged over 10 test trials for all the implementations. Sleep consistently improves performance, suggesting that replay stabilizes internal representations rather than degrading them.

Table 4.1: Classification accuracy before and after sleep

Implementation	After Training	After Sleep	Gain
Brian-vanilla (float)	75%	86%	+11
Brian (float)	73%	83%	+10
Lava (float)	73%	82%	+9
Lava (fixed)	73%	80.5%	+7.5

Taken together, these results demonstrate that the neuromorphic implementation preserves three fundamental properties of the original architecture: functional correctness, stability across learning phases, and robustness under reduced numerical precision. Stability emerges from the preservation of structured representations after sleep; and robustness is demonstrated by the persistence of these effects even when synaptic precision is reduced.

More broadly, the findings highlight the computational role of sleep within the model. Rather than representing a passive state, sleep constitutes an active consolidation phase that reorganizes synaptic statistics, improves representational smoothness, and stabilizes learned knowledge.

In summary, the experimental evidence confirms that the proposed neuromorphic adaptation successfully reproduces the behaviour of the reference model while remaining compatible with hardware-oriented constraints. The close agreement between floating-point and quantized simulations indicates that the architecture's learning dynamics are intrinsically robust and do not depend on high numerical precision, making it a promising candidate for fully deployment on neuromorphic platforms designed for efficient continual learning.

4.1.1 Inference on Loihi 2

This experiment was conducted using a set of quantized weights and tested in a full fixed-point implementation which includes the inhibitory layer, and on two minimal

Table 4.2: Comparison of the classification accuracy obtained with the three inference implementations. in all the four decoding strategies

Implementation	**Avg**	**Max**	**Top-3**	**Top-5**
Full inference (fixed)	82.60	77.40	80.80	80.80
Min. inference CPU (fixed)	71.00	73.80	78.60	80.60
Min. inference Loihi 2 (fixed)	71.20	74.00	79.40	81.00

inference implementations without an inhibitory layer, one in fixed-point simulation and one run directly on hardware. Table 4.2 shows the results obtained, and, as can be seen, the results for Top-5 encoding where the prediction is based solely on the 5 neurons that emitted the most spikes are similar across all 3 versions. This is mainly due to the fact that without the inhibitory network, many neurons even those not connected to a given label are able to spike, which lowers the accuracy based on the average spike count of the population connected to a given label; however, the neurons that spike the most remain the same.

4.2 Class-Incremental Continual Learning - Split MNIST

To evaluate the continual learning capabilities of the proposed architecture in a class-incremental setting, a Split MNIST protocol was adopted in which digit classes are introduced sequentially across tasks. Rather than training on all categories simultaneously, the network is exposed to pairs of classes at a time, forcing it to incorporate new knowledge while preserving previously acquired representations. This scenario provides a controlled framework for assessing resistance to catastrophic forgetting under progressive learning conditions.

The network is initialized with 100 Perc neurons and 100 CA3cue neurons for the first task, corresponding to the first pair of classes. At each subsequent task, an additional block of 100 neurons is recruited for both populations, while the semantic layer grows proportionally with the number of categories. The full configuration for each task is summarized in Table 4.3. This progressive allocation strategy ensures that representational capacity scales with task complexity, preventing interference caused by resource saturation.

When a new task begins, synaptic matrices are expanded accordingly and previously learned weights are restored into the corresponding sub-block of the enlarged matrices, while newly allocated regions are initialized to zero. This zero-padding procedure preserves all prior knowledge exactly while providing unused synaptic space for subsequent learning. At the same time, projector step offsets are maintained across tasks so that stimulus–neuron correspondences remain temporally

Table 4.3: Network configuration for each task in Split MNIST.

Task	Classes	Perc / CA3cue	Sem	Inh
1	0, 1	100	2	25
2	2, 3	200	4	50
3	4, 5	300	6	75
4	6, 7	400	8	100
5	8, 9	500	10	125

consistent throughout training. Each task follows a structured cycle consisting of learning, consolidation, and expansion. Newly introduced classes are first trained, after which the synaptic state is saved. A sleep phase then consolidates the memories of the current task through replay, followed by a second snapshot capturing post-consolidation weights. Then the network is tested on the so far learned classes and expanded to accommodate the next task. The quantitative evaluation confirms the effectiveness of this strategy for both versions of the network, Floating-Point and Fixed-Point. The Figures 4.4 and 4.5 shows the accuracy per task in all the four decoding strategies tested and the accuracy per class (only for the average strategy) in the various tasks while Tables 4.4 and 4.5 shows the recorded accuracy progression over the various tasks showing both the accuracies of the old and new classes. Overall, the accuracy on previously learned classes remains fairly stable across all tasks. In fact, both experiments recorded an average accuracy across the various tasks of 90% for the floating network and 90.7% for the fixed network. These results indicate that the progressive expansion mechanism allows the architecture to integrate new knowledge without degrading previous memories.

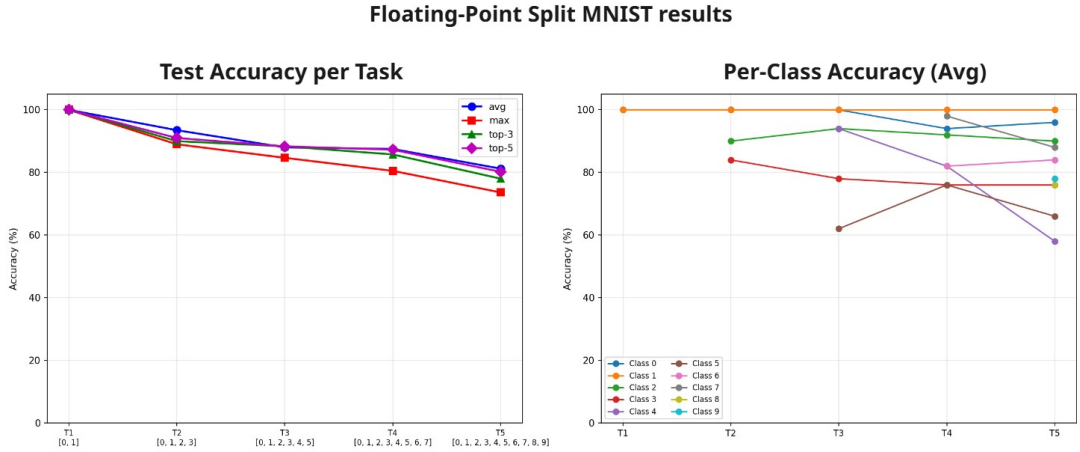

Figure 4.4: Average accuracies evolution over 5 different tasks and class accuracies over tasks - floating point network

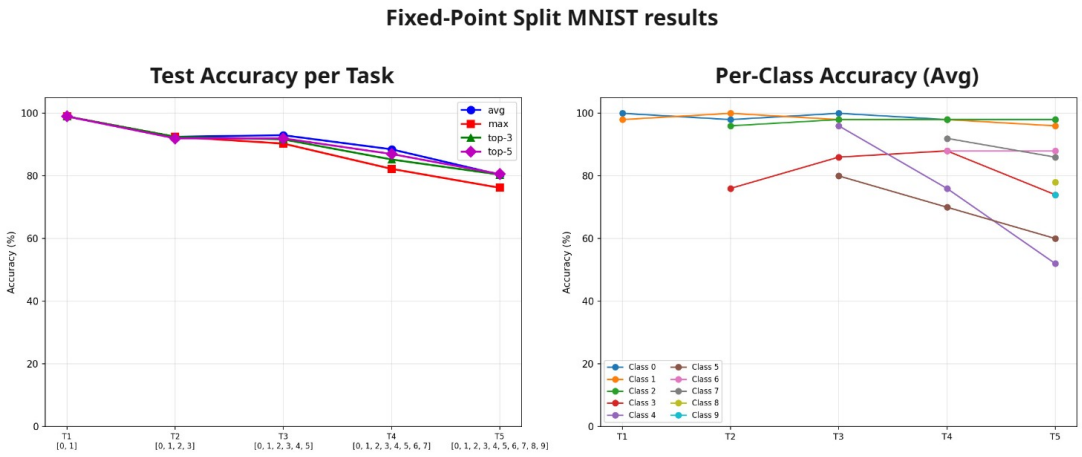

Figure 4.5: Average accuracies evolution over 5 different tasks and class accuracies over tasks - fixed point network

Overall, the Split MNIST experiment demonstrates that the architecture prevents catastrophic forgetting. By assigning distinct neural resources to each task, previously learned representations remain intact while new ones are added, providing a biologically plausible and computationally efficient solution to continuous learning

Table 4.4: Accuracy progression only for the average strategy across Split MNIST tasks for floating point network

Task	Old Classes	Accuracy	New Classes	Accuracy
1	-	-	[0,1]	99
2	[0,1]	99	[2,3]	87
3	[0,1,2,3]	93	[4,5]	84
4	[0,1,2,3,4,5]	87	[6,7]	90
5	[0,1,2,3,4,5,6,7]	82.5	[8,9]	77

Table 4.5: Accuracy progression only for the average strategy across Split MNIST tasks for fixed point network.

Task	Old Classes	Accuracy	New Classes	Accuracy
1	-	-	[0,1]	99
2	[0,1]	99	[2,3]	86
3	[0,1,2,3]	95.5	[4,5]	88
4	[0,1,2,3,4,5]	88	[6,7]	90
5	[0,1,2,3,4,5,6,7]	81.5	[8,9]	76

4.3 Domain-Incremental Continual Learning - Rotational MNIST

Whereas the previous experiment evaluated class-incremental learning, the rotational MNIST scenario investigates domain-incremental adaptation. In this setting, all ten classes are present from the beginning, but the input distribution changes progressively through geometric transformations. The challenge is therefore not to learn new categories, but to recognize the same concepts under varying perceptual conditions.

The network is initialized with 500 Perc and 500 CA3cue neurons for the first rotation and expands by 500 neurons for each subsequent rotation. In contrast to the Split MNIST scenario, the semantic layer remains fixed at ten neurons throughout the experiment because the set of categories does not change. The configuration for each rotation is summarized in Table 4.6.

Table 4.6: Network configuration for each rotation in Rotational MNIST.

Rotation	**Angle**	**Perc / CA3cue**	**Sem**	**Inh**
1	0°	500	10	125
2	15°	1 000	10	250
3	30°	1 500	10	375
4	45°	2 000	10	500
5	90°	2 500	10	625

Images are rotated using bilinear interpolation and subsequently binarised before conversion to spike trains. Within each block of 500 neurons, class representations are organized into fixed subranges of 50 neurons, ensuring that semantic identity is preserved across rotations while perceptual encoding expands linearly. Thus, neurons allocated to earlier rotations remain dedicated to those domains, and newly recruited neurons specialize in representing transformed inputs. For both implementations, the experiment reported an average accuracy of approximately 78%,

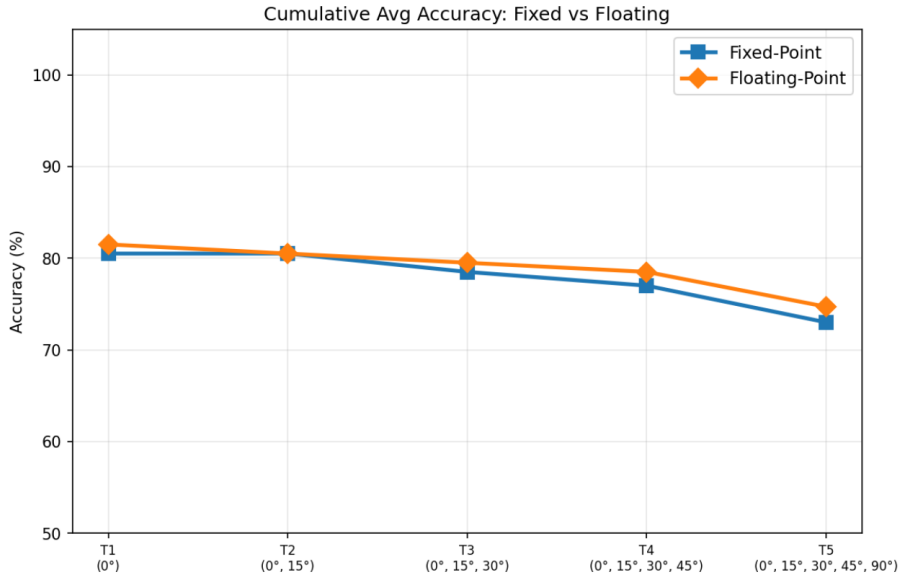

Figure 4.6: Average accuracies over 5 tasks

with slightly higher accuracy in the floating point version. Figure 4.6 shows the accuracy obtained for each task, while Figure 4.7 shows the accuracy obtained per class for task 5, where, especially for images rotated by 90°, when considered individually, the accuracy dropped dramatically, bringing the overall accuracy for task 5 to approximately 74%

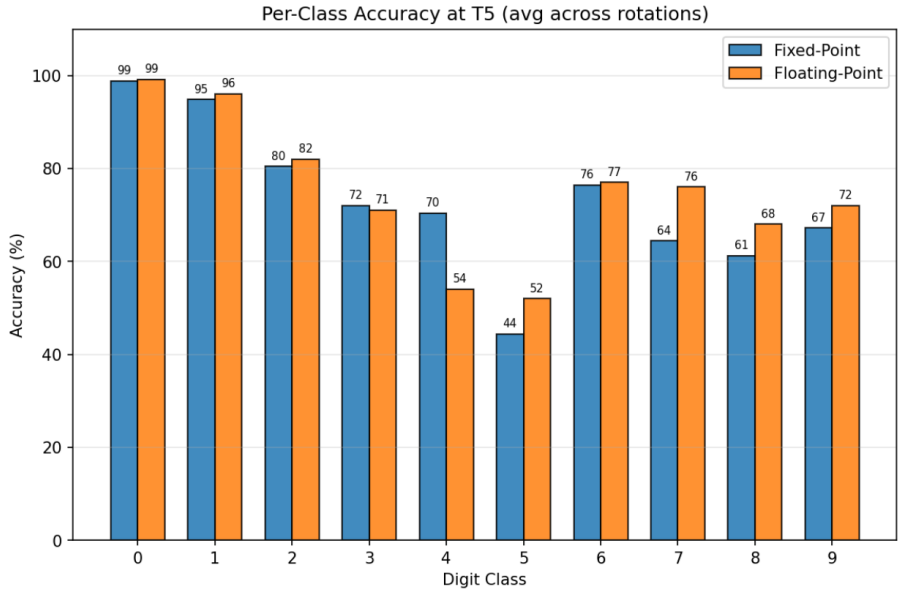

Figure 4.7: Average accuracies per class t5

Chapter 5

Conclusion

This thesis demonstrated that the biologically inspired HiCo spiking neural network can be systematically translated from a high-level simulation environment to a hardware-constrained neuromorphic implementation. The complete development pipeline, starting from Brian2 floating-point differential equations, through Lava process-based models, and finally to a Loihi 2-compatible fixed-point implementation, has been successfully established.

In particular, a key contribution of this work are the four structural modifications made to the original implementation making it hardware-friendly while preserving the dynamics, namely the simplification of the neural model used in the perceptual layer in terms of both input gates and unified decay, the transition from additive synaptic learning rules with clipping to multiplicative rules without clipping, the internalization of projectors and inhibitory baselines within the various neural models, and finally the unification of the sleep process, previously divided into two distinct phases, reactivation and consolidation, into a single step.

Another central step of the work involved the conversion of the neuronal dynamics to fixed-point arithmetic. All neuron populations of the HiCo architecture, including adaptive perceptual neurons, inhibitory neurons, and hippocampal neurons with projector dynamics, were reformulated in a manner compatible with the numerical constraints of Loihi 2. Bit-accurate process models were then developed and validated through CPU-based simulations in order to verify the consistency of the fixed-point implementation with respect to the floating-point reference model.

The correctness of the resulting implementation was validated through a series of analyses aimed at verifying that the behavior of the adapted model remains consistent with the original formulation. In particular, the validation included inspection of synaptic weight distributions, comparison of the neural and synaptic models developed in the two frameworks, visualization through heat maps relating to accuracy for pairs of classes, and analysis of the quality of the learned experiences. These experiments confirmed that the qualitative behavior of the learning process is preserved after the architectural changes.

Although on-chip learning remains an open challenge due to the expressivity gap between HiCo's three-factor plasticity rules and the current constraints of the Loihi 2 learning engine, the adopted strategy of offline training followed by static deployment provides a practical and immediately deployable solution for hardware inference. In this context, the construction of a minimal inference network consisting of an input layer, pre-trained static synapses, and perceptual neurons represents an important first step toward executing HiCo on neuromorphic silicon. This reduced configuration allows the forward computational pathway to be validated while providing a stable foundation for progressively integrating the remaining architectural components.

Beyond establishing this deployment pipeline, the work also provides several technical contributions related to the systematic adaptation of the HiCo model to the Lava framework and to the constraints of neuromorphic hardware. The complete architecture was first reimplemented using Lava-native components operating in floating-point arithmetic, with the goal of preserving as faithfully as possible the biologically inspired mechanisms originally defined in the Brian2 implementation. This stage ensured that the model dynamics could be reproduced within the process-based programming paradigm required by Lava.

Finally, a minimal hardware-deployable inference configuration was constructed, consisting of the input population, static synaptic projections, and the perceptual layer modeled in Loihi 2's proprietary microcode preserving the fixed-point behaviour.

The experimental evaluation confirmed that the adapted network maintains robust classification performance across the three considered evaluation scenarios: standard MNIST classification, class-incremental learning (Split MNIST), and domain-incremental learning (Rotational MNIST). In the standard MNIST setting, the Lava implementation achieved an average accuracy of 82% in floating-point precision and 80.5% in fixed-point precision. The minimal inference network deployed on Loihi 2 hardware achieved a maximum accuracy of 80.5%, maintaining the accuracy of the fixed-point implementation. In the continual learning scenarios, the floating-point implementation reached average accuracies of 90% on Split MNIST and 78% on Rotational MNIST, while the fixed-point implementation achieved comparable results of 91% and 78%, respectively. These results confirm that the conversion to fixed-point arithmetic and the architectural adaptations required for neuromorphic execution do not significantly degrade the learning capabilities of the original model.

5.1 Future Work

The work presented in this thesis opens several directions for future research aimed at both improving the model and advancing its deployment on neuromorphic hardware.

A first direction concerns the systematic optimization of the model hyperparameters. During the transition from floating-point to fixed-point arithmetic, several parameters had to be adjusted to maintain stable network dynamics and satisfactory performance. Automated hyperparameter optimization frameworks, such as *NNI*, could therefore be employed to explore the parameter space more efficiently and identify configurations that better compensate for the numerical constraints introduced by fixed-point implementations.

A second improvement direction, is on the resource utilization of the network. Currently it requires one neuron per sample, limiting its scalability to larger datasets. This could be addressed by using neuron populations of fixed size for each class rather than for each sample.

Another promising direction involves evaluating the model in dynamic and embodied environments. In the original HiCo study, experiments were conducted using a soft pneumatic gripper equipped with force and flex sensors for object classification tasks. Reproducing similar experiments within the neuromorphic implementation would allow the model to be tested in a realistic setting, where spiking neural networks can exploit their event-driven nature and temporal processing capabilities. Such experiments would also provide a meaningful benchmark for assessing the advantages of neuromorphic processing in real-time perception scenarios.

A particularly important research challenge concerns the possibility of enabling on-chip learning. As discussed in this work, the original three-factor plasticity rule used by HiCo cannot currently be implemented directly within the constraints of the Loihi 2 learning engine. Future research could therefore focus on designing approximations of these learning mechanisms that remain compatible with hardware limitations. For instance, simplified plasticity rules based on reduced-factor interactions or sequential updates could approximate the behaviour of multifactor learning while remaining implementable on neuromorphic processors.

Finally, once the full model is deployed on physical Loihi 2 hardware, a systematic evaluation of its computational characteristics will become possible. Profiling metrics such as latency, spike throughput, power consumption, and hardware resource utilization will allow a quantitative comparison between neuromorphic execution and conventional CPU-based simulation. Such analysis will be essential for assessing the practical benefits of neuromorphic implementations in terms of efficiency and scalability.

Overall, these directions highlight how the current work can serve as a foundation for further developments toward fully adaptive neuromorphic systems capable of operating in real-world environments.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[2] Hagar Hendy and Cory Merkel. «Review of spike-based neuromorphic computing for brain-inspired vision: biology, algorithms, and hardware». In: *Journal of Electronic Imaging* 31, 010901 (Jan. 2022), p. 010901. DOI: 10.1117/1.JEI.31.1.010901.

[3] Jared Kaplan et al. «Scaling Laws for Neural Language Models». In: *arXiv preprint arXiv:2001.08361* (2020). URL: https://arxiv.org/abs/2001.08361.

[4] Emma Strubell, Ananya Ganesh, and Andrew McCallum. «Energy and Policy Considerations for Deep Learning in NLP». In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355.

[5] Jason Kamran Eshraghian et al. «Training Spiking Neural Networks Using Lessons From Deep Learning». In: *CoRR* abs/2109.12894 (2021). arXiv: 2109.12894. URL: https://arxiv.org/abs/2109.12894.

[6] Vivienne Sze et al. «Efficient Processing of Deep Neural Networks: A Tutorial and Survey». In: *Proceedings of the IEEE* 105.12 (2017), pp. 2295–2329. DOI: 10.1109/JPROC.2017.2761740.

[7] Mishal Fatima Minhas et al. «Continual Learning With Neuromorphic Computing: Foundations, Methods, and Emerging Applications». In: *IEEE Access* 13 (2025), pp. 124824–124873. ISSN: 2169-3536. DOI: 10.1109/access.2025.3588665. URL: http://dx.doi.org/10.1109/ACCESS.2025.3588665.

[8] Elvin Hajizada et al. *Real-time Continual Learning on Intel Loihi 2*. 2025. arXiv: 2511.01553 [cs.LG]. URL: https://arxiv.org/abs/2511.01553.

[9] Federico D'Alba et al. «Semantization of memories in a hippocampal–cortical spiking neural network». In: *Neurocomputing* 640 (2025), p. 130323. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2025.130323. URL: https://www.sciencedirect.com/science/article/pii/S0925231225009956.

[10] James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. «Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory». In: *Psychological Review* 102.3 (1995), pp. 419–457. DOI: 10.1037/0033-295X.102.3.419.

[11] Dongqing Zhang et al. *LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks*. 2018. arXiv: 1807.10029 [cs.CV]. URL: https://arxiv.org/abs/1807.10029.

[12] Guiying Li et al. «Optimization based Layer-wise Magnitude-based Pruning for DNN Compression». In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 2383–2389. DOI: 10.24963/ijcai.2018/330. URL: https://doi.org/10.24963/ijcai.2018/330.

[13] Xiao Jin et al. *Knowledge Distillation via Route Constrained Optimization*. 2019. arXiv: 1904.09149 [cs.LG]. URL: https://arxiv.org/abs/1904.09149.

[14] Kashu Yamazaki et al. «Spiking Neural Networks and Their Applications: A Review». In: *Brain Sciences* 12.7 (2022). ISSN: 2076-3425. DOI: 10.3390/brainsci12070863. URL: https://www.mdpi.com/2076-3425/12/7/863.

[15] Paul A. Merolla et al. «A million spiking-neuron integrated circuit with a scalable communication network and interface». In: *Science* 345.6197 (Aug. 2014), pp. 668–673. DOI: 10.1126/science.1254642. URL: https://doi.org/10.1126/science.1254642.

[16] Mike Davies. *Taking Neuromorphic Computing to the Next Level with Loihi 2*. Technology Brief. Intel Labs, 2021, pp. 1–7. URL: https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf.

[17] Steve B. Furber et al. «The SpiNNaker Project». In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665. DOI: 10.1109/JPROC.2014.2304638.

[18] Ben V. Benjamin et al. «Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations». In: *Proceedings of the IEEE* 102.5 (2014), pp. 699–716. DOI: 10.1109/JPROC.2014.2313565.

[19] Brian 2 Development Team. *Brian 2 Documentation*. 2026. URL: https://brian2.readthedocs.io/en/stable/ (visited on Feb. 9, 2026).

[20] *Lava Software Framework*. lava-nc.org. 2026. URL: https://lava-nc.org/ (visited on Feb. 9, 2026).

[21] Alan L. Hodgkin and Andrew F. Huxley. «A quantitative description of membrane current and its application to conduction and excitation in nerve». In: *The Journal of Physiology* 117.4 (1952), pp. 500–544. DOI: 10.1113/jphysiol.1952.sp004764.

[22] Richard FitzHugh. «Impulses and physiological states in theoretical models of nerve membrane». In: *Biophysical Journal* 1.6 (1961), pp. 445–466. DOI: 10.1016/S0006-3495(61)86902-6.

[23] snntorch/Spiking-Neural-Networks-Tutorials. *tutorial_2_lif_neuron.ipynb*. https://github.com/snntorch/Spiking-Neural-Networks-Tutorials/blob/main/tutorial_2_lif_neuron.ipynb. GitHub notebook. 2026.

[24] Shaochuan Chen et al. «Electrochemical-Memristor-Based Artificial Neurons and Synapses—Fundamentals, Applications, and Challenges». In: *Advanced Materials* (2023). Figure 4: Basic structure and operation of chemical synapses (a) and electrical synapses (b), retrieved via ResearchGate https://www.researchgate.net/figure b-Basic-structure-and-operation-of-chemical-synapses-a-and-electrical-synapses-b_fig4_370863684. URL: https://www.researchgate.net/figure/a-b-Basic-structure-and-operation-of-chemical-synapses-a-and-electrical-synapses-b_fig4_370863684.

[25] Susanne Diekelmann and Jan Born. «The memory function of sleep». In: *Nature Reviews Neuroscience* 11.2 (2010), pp. 114–126. DOI: 10.1038/nrn2762.

[26] Per Sjöström and Wulfram Gerstner. «Spike-timing dependent plasticity». In: *Scholarpedia* 5 (Jan. 2010), p. 1362. DOI: 10.4249/scholarpedia.1362.

[27] Larry R. Squire et al. «Memory consolidation». In: *Cold Spring Harbor Perspectives in Biology* 7.8 (2015), a021766. DOI: 10.1101/cshperspect.a021766.

[28] Lynn Nadel et al. «Multiple trace theory of human memory: computational, neuroimaging, and neuropsychological results». In: *Hippocampus* 10.4 (2000), pp. 352–368. DOI: 10.1002/1098-1063(2000)10:4<352::AID-HIPO2>3.0.CO;2-D.

[29] Gordon Winocur and Morris Moscovitch. «Memory transformation and systems consolidation». In: *Journal of the International Neuropsychological Society* 17.5 (2011), pp. 766–780. DOI: 10.1017/S1355617711000683. URL: https://pubmed.ncbi.nlm.nih.gov/21729403/.

[30] Daniel N. Barry and Eleanor A. Maguire. «Remote Memory and the Hippocampus: A Constructive Critique». In: *Trends in Cognitive Sciences* 23.2 (2019), pp. 128–142. DOI: 10.1016/j.tics.2018.11.005. URL: https://doi.org/10.1016/j.tics.2018.11.005.

[31] Jaan Aru, Mototaka Suzuki, and Matthew E Larkum. «Cellular mechanisms of conscious processing». In: *Trends in cognitive sciences* 24.10 (2020), pp. 814–825.

[32] Jaan Aru et al. «Apical drive—A cellular mechanism of dreaming?» In: *Neuroscience & Biobehavioral Reviews* 119 (2020), pp. 440–455. ISSN: 0149-7634. DOI: https://doi.org/10.1016/j.neubiorev.2020.09.018. URL: https://www.sciencedirect.com/science/article/pii/S0149763420305753.

[33] Jaeyoung Park. «Neuromorphic Computing Using Emerging Synaptic Devices: A Retrospective Summary and an Outlook». In: *Electronics* 9.9 (2020), p. 1414. DOI: 10.3390/electronics9091414. URL: https://doi.org/10.3390/electronics9091414.

[34] Mike Davies et al. «Loihi: A Neuromorphic Manycore Processor with On-Chip Learning». In: *IEEE Micro* 38.1 (2018), pp. 82–99. DOI: 10.1109/MM.2018.112130359.

[35] Recep Buğra Uludağ et al. *Bio-realistic Neural Network Implementation on Loihi 2 with Izhikevich Neurons*. 2023. arXiv: 2307.11844 [cs.NE]. URL: https://arxiv.org/abs/2307.11844.

[36] Raymond P. Kesner. «Behavioral functions of the CA3 subregion of the hippocampus». In: *Learning & Memory* 14.11 (2007), pp. 771–781. DOI: 10.1101/lm.688207.

[37] Roberta Tatti et al. «Neurophysiology and Regulation of the Balance Between Excitation and Inhibition in Neocortical Circuits». In: *Biological Psychiatry* 81.10 (2017), pp. 821–831. DOI: 10.1016/j.biopsych.2016.09.017.

[38] Y. Lecun et al. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.