People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering
Department of Electronics

Project Report Presented in Partial Fulfilment of

the Requirements of the Degree of

'MASTER'

In: **Electronics**

Option: **Computer Engineering**

Title:

Exploration of Spiking Neurons Leakages and Network Recurrences for Spike-Based Spatio-Temporal Pattern Recognition

Presented by:

- **BOUANANE Mohamed Sadek**

Supervisors:

- **Dr. CHERIFI Dalila**
- **Dr. KHACEF Lyes**

Registration Number: 2021/2022

To my parents
To my whole family
To all those who helped me

Abstract

Brain-inspired computing is being explored to imitate the astonishing capabilities of biological brains to perform robust and efficient computations. To map these capabilities into hardware, a growing number of neuromorphic computers are being built to emulate biological neural networks. These developments created a need to address the lack in understanding of different neuronal behaviours that can enable us to find the right level of abstraction from biology and get the best performance in accurate, efficient and fast inference. Aiming at addressing this problem, we give a detailed overview of the concerned spiking neuron models and surrogate gradient methods, which are used to study the impact of synaptic and membrane leakages in feed-forward, as well as recurrent network topologies on learning visual and auditory information. We also investigate whether or not heterogeneity at the neuronal level plays a functional learning role. We found out that leakages are important when we have both temporal information and a recurrently connected topology. We also found that heterogeneity slightly improves performance on temporal information. The results we obtained will provide more insight on developing efficient neuromorphic hardware.

Keywords

Spiking Neural Networks, Neuromorphic Computing, Spacio-Temporal Patterns, Synaptic Leak, Membrane Leak, Network Recurrences, Neural Heterogeneity.

"It seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers... They would be able to converse with each other to sharpen their wits. At some stage therefore, we should have to expect the machines to take control."

Alan Turing.

Acknowledgements

I would like to express my sincere gratitude to Dr. Cherifi Dalila for the countless help and support she offered me not only for this project but throughout my years at the institute. I am also extremely grateful to my co-suppervisor Dr. Khacef Lyes who gave me the chance to work on neuromorphic computing, enriched my understanding of the field, and ignited the spirit of research and inquiry in me. I want to thank them both for the regular meetings we held throughout an entire year; for their assistance, patient guidance, and useful critiques of this work.

I would also like to thank my parents and my family for their invaluable and unconditional support. I would like to thank them for always being there for me during my ups and downs. They are the people that I want to make the most proud of me.

Contents

List of Figures

List of Tables

List of Abbreviations

AER	**A**ddress **E**vent **R**epresentation
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
ATIS	**A**synchrounous **T**ime-Based **I**mage **S**ensor
BP	**B**ack-**P**ropagation
BPTT	**B**ack-**P**ropagation **T**rought **T**ime
CPU	**C**entral **P**rocessing **U**nit
CUBA-LIF	**Cu**rrent-**Ba**sed **L**eaky **I**ntegrate-and-**F**ire
FSNN	**F**eed-Forward **S**piking **N**eural **N**etwork
GPU	**G**raphical **P**rocessing **U**nit
IF	**I**ntegrate-and-**F**ire
LIF	**L**eaky **I**ntegrate-and-**F**ire
MLP	**M**ulti **L**ayer **P**erceptron
MNIST	**M**ixed **N**ational **I**nstitute of **S**tandard **T**echnology
N-MNIST	**N**euromorphic **MNIST**
PSC	**P**ost **S**ynaptic **C**urrent
RCNN	**R**ecurrently **C**onnected **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
RSNN	**R**ecurrently Connected **S**piking **N**eural **N**etwork
SG	**S**urrogate **G**radient
SHD	**S**piking **H**eidelberg **D**igits
SNN	**S**piking **N**eural **N**etwork

General Introduction

Over the last decade, Artificial Neural Networks (ANNs) have been increasingly attracting interest in both academia and industry as a consequence of the explosion of open data and the sufficient computing performance of today's computers that is necessary for their training and inference. The recent results of deep neural networks on image classification has given neural networks the leading role in machine learning algorithms and Artificial Intelligence (AI) research. However, conventional hardware implementations of applications that require real-time processing on the edge such autonomous vehicles, remain very expensive, especially in energy consumption, due to the non-adaptation of the hardware to the computation model [1]. The human brain, on the other hand, can reliably learn and process complex cognitive tasks with only a few watts of power. For this reason, Spiking Neural Networks (SNNs) have lately been investigated in order to achieve robust and energy-efficient machine intelligence using inspiration from the neuroscience literature. However, there is a lack of understanding of how each of the factors determining the biological neuronal response can be effectively used in learning. Our project tackle this problem by studying the effect of the leakages in three spiking neurons models with variable degrees of biological abstraction on spatio-temporal classification tasks.

This report is organised as follows: Chapter 1 provides some context into the field of neuromorphic computing and the motivations behind it. It summarises all the necessary foundations and concepts to familiarize and prepare the reader for the rest of the report. Chapter 2 briefly evaluates the available literature related to this work, introduces the used spiking neurons models, and presents the problem faced when training SNNs. It also gives an overview of the existing approaches to overcome these problems, and elucidates the methods and tools used in this work. Chapter 3 presents all of the experiments we conducted in order to understand the effect spiking neurons leakages and network recurrences for spike-based spatio-temporal pattern recognition and gives detailed analysis of the results we obtained.

Chapter 1

Overview on Neuromorphic Computing with SNNs

1.1 Introduction

With the continuous development in human civilization, computers have undergone tremendous improvement and become increasingly popular because of their ability to perform tasks much faster than the human brain. Early computers were developed using mechanical and physical components including geared wheels used by the ancient Chinese to navigate directions more than 2000 years ago. The slide rule is another computer which has been around since the 1600s and was widely used in science and engineering professions until the mid-1900s. In fact, rather complicated tasks like navigation optimization were also accomplished in the early 1900s by using mechanical computers like Dumaresq. In the mid-1900s, electrical components like resistors, diodes, amplifiers and vacuum tubes became popular because, unlike mechanical components such as mass-spring systems, reconfiguring them to solve a different problem was much easier than reconfiguring an all-mechanical computing system. The transistor's invention back in 1947 revolutionized the way computers were built and used. Transistors were smaller than prior electrical components like vacuum tubes and could be arranged in dense arrays. Furthermore, the advent of integrated circuit enabled the assembly of dense arrays of electronic components on a single piece of material, commonly silicon, allowing for the packing of greater processing power in a given area and large-scale manufacturability. This sparked a commercially motivated engineering race to continually enhance the performance of computing chips by reducing the size of electronic components and cramming more of them onto a single chip, resulting in what is known as Moore's law [2]. Moore's

law allowed companies to innovate and create increasingly more powerfull hardware such as Central Processing Units (CPUs) and Graphical Processing Units (GPUs) that contributed tremendously to the progress we see today in AI. The increasing computing power of toady's computers made training and inference of deep learning models possible.

1.2 The Need for New Computing Primitives

Although computing capabilities increased at an exponential rate for several decades, computing efficiency started flattening out especially with the current emerging technologies such as the Internet of Things (IoT) and edge computing. There are multiple reasons associated with this change which necessitate finding new computing premitives.

1.2.1 Moore's Law and Dennard Scaling

Moore's Law [2] was never a strict law of nature. It is more of a rule of thumb based on a forecast made by Gordon Moore, co-founder of Intel, in 1965. He noticed that transistors were shrinking at a pretty consistent rate, and he anticipated that the number of transistors that could fit on a silicon chip would roughly double every two years or so. Moore's prediction proved to be very accurate, allowing the industry to forecast the ability of semiconductor circuits to become smaller and more powerful. Dennard scaling which is based on a 1974 work co-authored by IBM researcher Robert H. Dennard [3], is related to Moore's law in that it claims that computing performance per watt grows exponentially at roughly the same rate. Dennard proposed that as transistors get smaller, their power density remains constant, allowing power consumption to remain proportional to area. This enabled CPU manufacturers to increase clock rates from generation to generation while consuming less total circuit power. Dennard's Scaling appeared to be failing between 2005 and 2006 [4]. Because of the inability to operate within the same power envelope, the CPU industry transitioned to multicore designs, posing substantial hurdles for memory technology. While Moore's Law stayed true for more than 50 years, it became increasingly difficult to capitalize on its benefits in the last decade. It has reached its physical limit and has begun to slow significantly [5]. A chip node in the early 1970s was roughly 10000 nm in size; today, those nodes are only a few atoms wide, around 5nm. Once a transistor reaches one atom in

width, it is impossible to make it smaller. Yet, the huge computational demands of AI and the exponential growth of data, requires things to get smaller, faster, and more powerful. As a result, the semiconductor industry has been researching what comes next and exploring remedies to Moore's law's stagnation.

1.2.2 The Von Neumann Bottleneck

The Von Neumann bottleneck is an unavoidable consequence of using a bus to move data between the processor, memory, long-term storage, and peripheral devices. No matter how quickly the bus completes its data transfer, creating a bottleneck that slows down the whole process is always a possibility especially when dealing with massive amounts of data in AI systems. Processor speeds continue to improve with time, while memory and other device advancements focus on the capacity to store more in less space. As a result, with each advancement, the bottleneck becomes more of an issue, leading the processor to spend a significant amount of time idle.

1.2.3 Energy Consumption

For decades, computer architecture specialists have been studying energy usage in order to produce cutting-edge energy-efficient processors. Machine learning researchers, on the other hand, have mostly focused on building high-accuracy models without taking energy usage into account [6]. This is the situation with deep learning, where the goal has been to develop deeper and more accurate models with no computational considerations. In particular, this problem limits the deployments of embedded AI systems with limited battery budget such as neural implants, edge device and autonomous vehicles. More generally, AI already has a significant carbon footprint on the environment [7]. If current industrial trends continue, it will soon become much worse and may potentially become an adversary in the fight against climate change in the coming years. In summary, modern AI models demand enormous amounts of energy, and these energy requirements are increasing at an alarming rate.

1.3 Learning from the Brain

Brains are phenomenal machines. They are the most powerful computing entities while consuming only about 20 watts of power [8]. To put this efficiency and low

power consumption into perspective, AlphaGo, the first computer program to defeat a professional human Go player, requires power of approximately 200 kW to run [9], which means 10,000x more power than the brain, while it only plays the game of Go. Moreover, biological brains tend to be relatively smaller when compared to the size of the body. They account for only about 2% of the entire body weight. Yet, they consume 25% of the energy used by the body, which is surprising if we take into consideration the serious energy demands of the muscles. That is because they are extremely important. They control our thoughts, memory, emotions perception and almost every process that regulates our body. Brains provide us with behavioral flexibility that is unmatched by any of our most sophisticated machinery. They are also constantly adapting to the uncertain, noisy, and rapidly changing world in which they are embedded and embodied.

1.3.1 Brain and biological neurons

The brain is made up of thin sheet called the cortex crammed inside the skull, surrounding other subcortical structures including the basal ganglia and the brainstem, with the cerebellum stuck on the back. the sheet has six distinct layers [8] composed of three main elements: (1) the cell bodies of neurons; (2) the very long thin processes used for communication between neurons, called "axons"; and (3) glial cells. There are about 25 billion neurons in the human cortex. That is about 170,000 neurons in each square millimeter. If we include other subcortical areas, however, there are around 100 billion neurons in the entire human brain. Figure 1.1 demonstrates the major elements of brain anatomy.

Neurons are specialised for communications. In fact, it is generally believed that what provides brains with their impressive computational abilities is the organization of the connections among individual neurons. What makes neurons stand out compared to other cells in our bodies are the many branching processes that project outward from the main cell body. These processes enable short and long distance communication with other neurons. Figure 1.2 represents a simplified anatomy of the biological neuron.

The cellular projections that carry information to the cell body (also called the "soma") are called dendrites. The cellular projection that carries information away from the cell body is called the axon. Dendrites carry signals to the cell body in the form of an ionic current. If sufficient current gathers in the cell body (at the axon

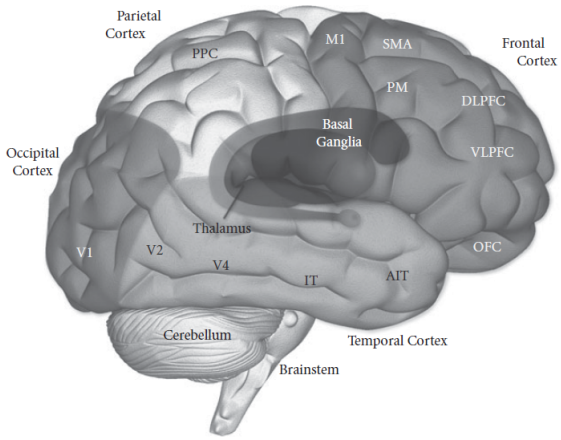


FIGURE 1.1: Elements of brain anatomy [8].

hillock), then a series of cellular events are triggered that result in an action potential, or voltage "spike," that proceeds down the axon. Neural spikes are very brief events lasting for only about a millisecond, which travel in a wave-like fashion down the axon until they reach the end of the axon, called the bouton. When spikes reach the bouton, they cause the release of tiny packets of chemicals called neurotransmitters into the space between the bouton and the dendrite of the subsequent neuron (called the "synapse"). Neurotransmitters then bind to special proteins called "receptors" in the cell membrane of the dendrite. This binding causes small gates, or channels, in the dendrite of the next neuron to open, allowing charged ions to flow into the dendrite. These ions result in the current signal in the receiving dendrite, which is called the Post-Synaptic Current (PSC). The process then continues as it began. Figure 1.3 outlines the main elements underlying neural communication.

Although it can seem simple, this process is extremely complex because the brain a highly heterogeneous system. There are hundreds of kinds of neurons with different kinds of intrinsic dynamical properties such as size, number of inputs and outputs. Also, there are hundreds of different kinds of neurotransmitters and many different kinds of receptors. Different combinations of neurotransmitters and receptors can cause different kinds of currents to flow in the dendrite. As a result, a single spike transmitted down an axon can be received by many different neurons and can have different kinds of effects on each neuron, depending on the mediating neurotransmitters and receptors. Even neurons of the same kind often generate different patterns of spikes to exactly the same current input. This complexity continues to all aspects of this system which results in endless frustration during analysis [8].

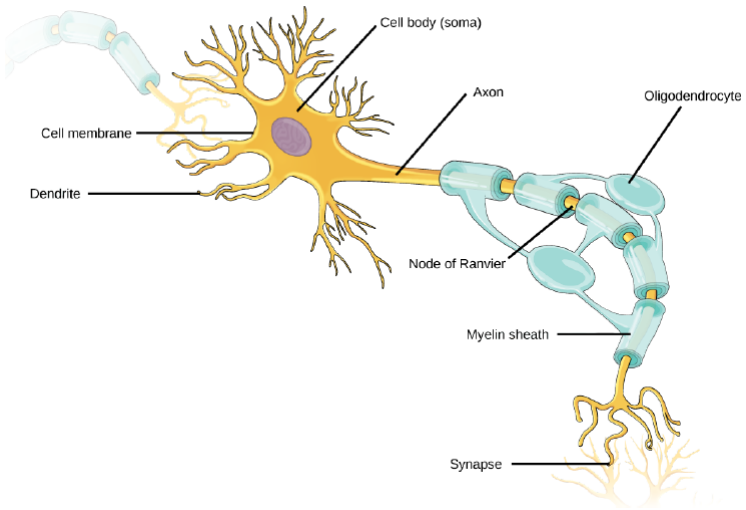


FIGURE 1.2: Anatomy of biological neuron.

1.3.2 Brain-Inspired Computing

For several decades, the brain's supremacy in computing has been well acknowledged. The popular area of artificial intelligence sprang from the premise that computers, like the brain, can incorporate some sorts of learning capability. In fact, it was imagined by Alan Turing back in 1948 in his paper "Intelligent Machinery" [10]. For nearly a century, scientists have modeled the network of neurons within the brain. Several categories of algorithms have been developed with loose ties to models of biological neural networks, which are collectively known as ANNs. Depending on the level of abstraction that describe the brain, there is a wide range of primitives that enable brain-inspired computing, also known as neuromorphic computing. Some of them are illustrated bellow [11, 12]:

- **Parallel and distributed with in-memory neural computing:** The brain performs numerous mathematical calculations concurrently. Parallel processing is a critical component that enables the brain's fast computations. In contrast to modern computers, the brain stores information in the same synapses that do computation. This eliminates the requirement to fetch data before each operation.

- **Analog computing and digital communication:** Today's most common computers are digital in nature, which means they store information in bits of 0's and 1's. However, the majority of real-world information is made up of real

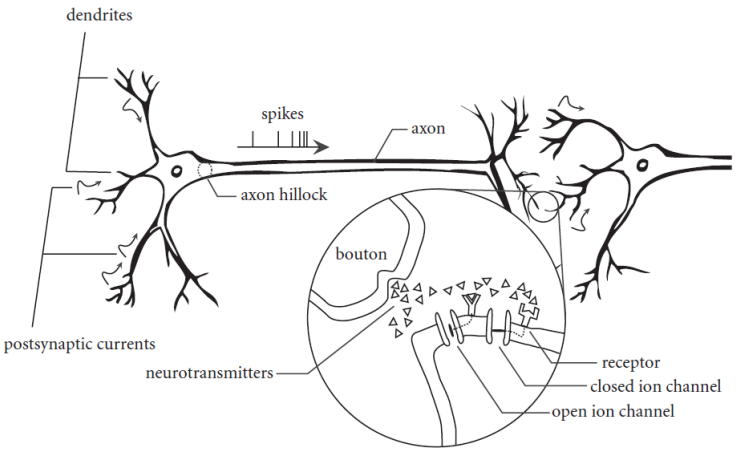


FIGURE 1.3: Elements of neural communication [8].

numbers, which are analog in nature. Every bit of a binary representation of a real number is assigned to a different binary memory unit in a digital computer. While analog signals are very sensitive to noise, digital signals are significantly more robust in long-distance communication. Neurons appear to have worked this out by processing analog information at the soma and communicate digitally down the axon.

- **Sparse and asynchronous computing:** The brain's neural connections are very sparse, which means that not every neuron is connected to every other neuron, and not every neuron in a processing layer is active at any given time step. In fact, neurons spend most of their time at rest. Sparsity is extremely necessary for enabling a large-scale network; its absence will fundamentally limit the system's growth.

- **Online learning with local plasticity:** Plasticity is embodied in the brain through real-time modifications of the biological synaptic weights between neurons during the training process in response to learning new and relevant information. Note that we can distinguish between plasticity and learning such that the physical processes for modifying the weights of a neural network are referred to as plasticity, and learning dictates the direction in which the weights should be altered.

1.3.3 Spiking Neural Networks

SNNs are inspired by information processing in biology. They are ANN models that more closely mimic natural neural networks. SNNs include the element of time into their working model in addition to neuronal and synaptic state. The notion is that neurons in the SNN do not fire at the end of each propagation cycle, but rather when a membrane potential, which is an inherent characteristic of the neuron associated to its membrane electrical charge, reaches a certain value. When a neuron fires, it sends a signal to neighboring neurons, which increase or decrease their potentials in response to the signal [13]. We will give a more detailed overview of how SNNs work in the next chapter. SNNs are promising to drive the next generation of AI into matching humans flexibility and ability to learn from unstructured stimuli with the energy efficiency of the human brain [8].

1.4 Event-Based Sensing

The nervous system transforms physical quantities perceived by primary receptors into trains of events, which are then processed in the brain. Engineers have long sought brain-like ways to sensing and signal processing due to the unparalleled efficiency of information processing [14]. Event-based sensing is an important concept in neuromorphic computing, inspired by the tendency of sensory neurons in the nervous system to preferentially respond to changes in the sensed quantity rather than continuously reporting its ongoing state. A simple depiction of the process in shown in Figure 1.4 where Only dynamic segments of a scene are passed to the output ('1'), while static regions are suppressed ('0').

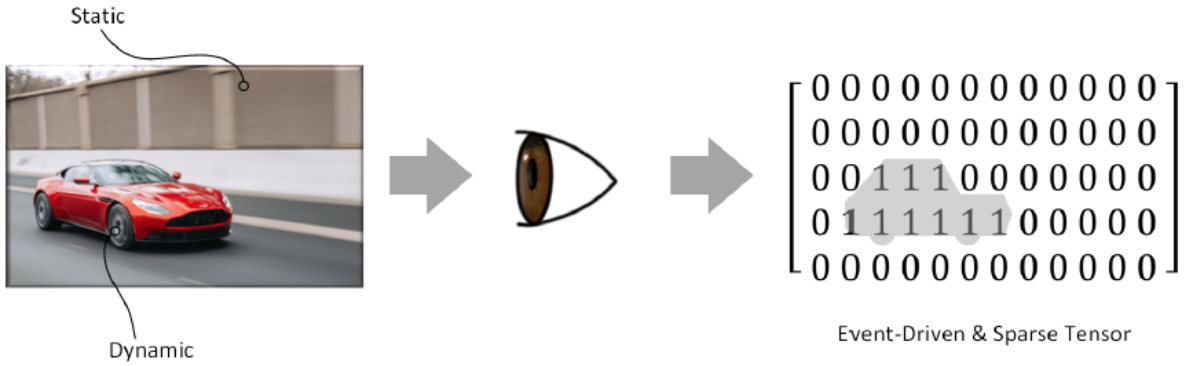


FIGURE 1.4: Event-Driven Processing [11].

When an event is generated by detecting a signal threshold crossing, it is typically emitted as a data structure containing two pieces of information: first, an address, which is the coordinates of the pixel that emitted the event for example, and second, the time of event creation. In real-time systems, time can represent itself, and only the address must be sent. This protocol is known as Address-Event Representation (AER).

1.4.1 Event-Based Vision Systems

Event-based vision clearly outperforms frame-based techniques. First, rather than delivering a snapshot of absolute brightness at all sites in the visual field, event-based vision systems report the exact moments of relative brightness changes. Moreover, because they only gather data when anything happens, event-based sensors can achieve very short latencies that are only bound by the sensor's minimal response time, whereas frame-based systems are compelled to obey the predetermined inter-frame interval. These characteristics make event-based vision ideal for applications such as autonomous robotics, where traditional cameras' relatively high latency, computationally expensive sensing, and processing pipeline are sub-optimal. There are numerous ways for developing silicon retinas. Some examples include [15, 16, 17].

1.4.2 Event-Based Auditory Systems

Using traditional digital signal processing techniques, digital audio recording devices deconstruct signals. In brains auditory region, the mechanical features of the basilar membrane in the cochlea break sound impulses into frequency bands. The band-passed components are converted by hair cells into neural pulses, which are then transmitted to higher auditory areas of the brain. Speech, music, and environmental noise are all temporally structured forms of auditory information. When compared to simple rate codes, the brain is thought to gain a computational advantage by utilizing the timing of action potentials to code information. Significant work has gone into the creation of silicon cochleas for signal processing. To mention some of them, here are few examples: [18, 19, 20].

1.5 Neuromorphic Accelerators

An accelerator is a system of hardware, software, or both that speeds up the computation of solutions to specific types of problems. There are two approaches of brain inspired accelerators: "neuroscience-focused" or "AI-focuses" accelerators. The "neuroscience-focused" approach seeks to mimic fundamental observations of biological neural networks. The emphasis is on biological plausible mechanisms of plasticity as well as biophysically realistic models of spiking neurons. These structures are useful for neuroscientists because they provide a platform for investigating large-scale brain models. In contrast, "AI-focused" approaches start with an end-application in mind and try to build hardware that efficiently achieves a solution. In order to handle real-world issues, these approaches compete directly with traditional hardware such as GPUs and CPUs. Several neuromorphic hardware implementations can be found in the literature. Here, we present some of them:

- **SpiNNaker [21]:** A fully digital system aiming to simulate very large spiking networks in real-time, and in an event-driven processing fashion. It is composed of 864 ARM9 cores, divided into 48 chips containing 18 cores each. The main feature of SpiNNaker is its efficient communication system: all the nodes are interconnected through high-throughput connections designed for small packet routing, which contain AER spikes. SpiNNaker is used to implement massively parallel hardware SNNs in the literature, such as NeuCube where a SNN is implemented on SpiNNaker to capture and classify spatio-temporal information from EEG.

- **Loihi [22]:** An Intel Labs neuromorphic research chip that uses asynchronous SNNs to provide adaptive self-modifying event-driven fine-grained parallel computations used to implement learning and inference at high levels of efficiency. The device is a 128-neuromorphic cores many-core IC manufactured on Intel's 14 nm technology and includes a customizable microcode learning engine for on-chip SNN training.

- **TrueNorth [23]:** This IBM's chip, is a real-time neurosynaptic processor with a non-Von Neumann, low-power, scalable architecture. The TrueNorth chip,

which has 4096 neurosynaptic cores, has 1 million digital neurons and 256 million synapses that are tightly connected by an event-driven routing infrastructure. The design is particularly suited to complicated neural network applications.

- **NeuroGrid [24]:** A hybrid digital–analog system designed for real-time modeling of huge SNNs. Subthreshold circuits are used to model neuronal components. Because of the physics of transistors operating in the subthreshold regime, synaptic functions are directly replicated. The board is made up of 16 NeuroCore chips that are linked together by an asynchronous multicast tree routing digital communication system. Because each core is made up of 256×256 analog neurons, NeuroGrid can mimic up to 1 million neurons and billions of synaptic connections.

1.6 Spatio-Temporal Pattern Detection

Biological brains are complex integrated spatio-temporal systems. Space is represented by "which" neuron fires, while time is represented by "when" the neuron fires. Both of these quantities carry the necessary information to be processed by the cognitive functions. Spatio-temporal data, that is characterized by a strong temporal structure, is extremely common in many fields including engineering (e.g., speech and video), bioinformatics (e.g., gene and protein expression), neuroinformatics (e.g., EEG, fMRI), ecology (e.g., establishment of species), environment (e.g., the global warming process), medicine (e.g., patients risk of disease or recovery overtime), economics (e.g., financial time series, macroeconomics), etc. [25]. However, effective approaches for modeling such data and for spatio-temporal pattern recognition that can enable new discoveries and create more accurate predictions of spatio-temporal events in autonomous systems are lacking [25]. To parallel the brain's energy efficiency and processing capacity, methodologies that operate in both the spatial and temporal domains are important. Inspired by the brain, SNNs posses an intrinsic capability to recognise information encoded in spatio-temporal patterns of spiking signals [25]. Nonetheless, to further improve the efficiency of neuromorphic computing, more research to analyze these patterns is essential for better understanding the working mechanisms of SNNs and neural circuits.

1.7 Summary

In this chapter, we have presented the different limitations of today's computers and machine learning that created a critical need to invent new computing primitives. These limitations include the stagnation of Moore's law and Dennard scaling, the Von Neumann bottleneck, and the huge energy consumption that deep learning algorithms require to learn from data for very complex tasks, along with the environmental risks it could entail. Next, a brief description of biological brains and neuronal cells anatomy in addition to their enormous levels of complexity and heterogeneity that give rise to their sophisticated abilities, but also could get beyond of our comprehension quickly. Moreover, we discussed what can be learned from the brain that can be considered as primitives that enable brain neuromorphic computing and introduced event-based sensing. We also mentioned some existing neuromorphic hardware implementations that can be found in the literature. At the end, we discussed how the brain is a spatio-temporal system and more research is needed to analyse these patterns.

Chapter 2

SNNs Training Methods for Spatio-Temporal patterns

2.1 Introduction

SNNs are nature's highly efficient solution to different problems of signal processing such as fault-tolerance and energy efficiency. The development of neuromorphic hardware that attempts to emulate these biological SNNs created a need to develop tools and methods that can enable them to solve real-world problems. Although SNNs can be trained on real domain specific data just like conventional neural networks, their intrinsic binary and dynamical nature introduces a number of challenges. This chapter briefly evaluates the available literature related to this work, introduces the used spiking neurons models, and presents the problem faced when training SNNs. It also gives an overview of the existing approaches to overcome these problems, and elucidates the methods and tools used in this work.

2.2 Literature Review

Neuromorphic computing literature shows that past studies primarily focused on using spiking neuron models for event-driven applications and low-power embedded hardware implementations of SNNs. They are also widely used in large scale and massively parallel neuromorphic cortex simulators[26]. Some studies compared between FPGA-based spiking models and the conventional non-spiking Multi Layer Perceptron (MLP) models where SNNs have proven to be as reliable in terms of accuracy for simple pattern recognition tasks, but gained 2x in resources and power, although it shows that SNNs are less efficient for static data, so they should be used

in dynamic environments with inherent temporal information [27, 28]. Other studies confronted SNNs on the Loihi neuromorphic hardware and non-SNNs in embedded GPUs. Again, smililar accuracy and a gain of 30x in energy efficiency for multimodal (vision+EMG) hand gesture recognition, but a loss in accuracy and a gain of 250x in energy efficiency for tactile braille letters recognition [29]. Furthermore, some studies focused on exploring the role of the leakages in learning where it has been shown that leaky neuron models enable improved robustness against random noise, without the need for costly retraining procedures or error amplification[30]. However, to the best of our knowledge, the effect of both membrane and synaptic leakages in spiking neuron for spatial-temporal pattern recognition has not been studied comprehensively. Recognising this gap, our work is to study the effect of these leakages from a modeling perspective and assesses their cost in terms of accuracy, as well as spiking activity. Understanding the effect of the leakages will provide the neuromorphic community with insight on power consumption and hardware resources as they implies extra circuitry to implement [27].

2.3 Spiking Neuron Models

A neuron, similar to other cells in the body, is enclosed by a thin membrane. This membrane is an insulating lipid bilayer that separates the neuron's conducting saline solution from the extracellular medium. Analogically, the two conductive solutions separated by an insulator serve as an electrical capacitor. Another function of this membrane is to regulate what enters and exits the cell. The impermeability of the membrane prevents ions (e.g., Na^+) from entering and exiting the neuron body. However, by injecting current into the neuron, certain channels in the membrane are stimulated to open. Electrically, This charge movement can be modelled by a resistor. The French neuroscientist Louis Lapicque was the first to discover this resemblance between neuronal membranes and RC circuits in 1907[31]. He used a short electrical pulse to trigger a frog's nerve fiber and discovered that neuron membranes might be approximated as a capacitor with a leakage. The similarity between neuronal cells and RC circuits is shown in Figure 2.1.

In a neural network, the sub-threshold dynamics of a Lapicque's neuron in layer l with index i can be formally describe with the differential equation [13]:

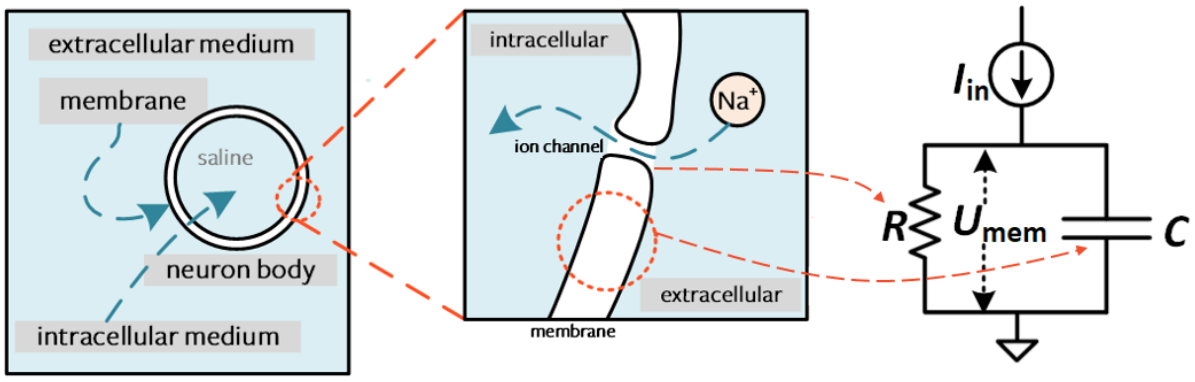


FIGURE 2.1: Neuron cell as an RC circuit [11].

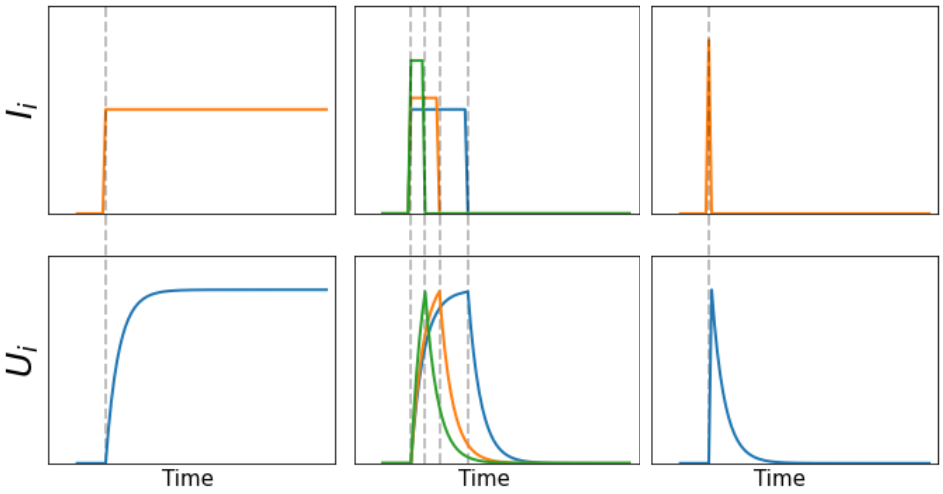
$$\tau_{mem}\frac{\mathrm{d}U_i^{(l)}(t)}{\mathrm{d}t} = -(U_i^{(l)}(t) - U_{rest}) + RI_i^{(l)}(t) \tag{2.1}$$

where $U_i(t)$ is the membrane potential that characterizes the hidden state of the neuron, U_{rest} is the resting potential, τ_{mem} is the membrane time constant, R is the input resistance, and $I_i(t)$ is the input current. Equation 2.1 is called the equation of a passive membrane [13].

If the passive membrane is stimulated by a step input current I_0 at $t = 0$, assuming it never stops, the membrane potential would exponentially approach an asymptotic value $U_i(\infty) = U_{rest} + RI_0$. Once steady state is reached, the charge on the capacitor no longer changes and all input current must then flow through the resistor. For short pulses of of current delivered to the membrane, the amplitude of the voltage response does not depend on the height of the current pulse, but only on the total charge $q = \int I_i(t)\mathrm{d}t$. As the input current pulse increases, the membrane potential rise time also increases. If the width of the current pulse becomes infinitesimally small, in other words, the charge is delivered in an infinitely short period of time, the membrane potential will jump in a zero rise time.

In the absence of input current, the membrane potential decays to its resting value U_{rest} exponentially. This decay is characterised by the membrane time constant τ_{mem} and is referred to as the "membrane leak". The response f the passive membrane to different current stimulus is shown in Figure 2.2.

Figure 2.3 depicts the release of neurotransmitters into the synaptic cleft from a pre-synaptic neuron. The neurotransmitters bind to the receptors at the post-synaptic neuron site. This influx of charge results in a current injection into the post-synaptic



(a) Step input (b) Varying input pulses. (c) Spike input.

FIGURE 2.2: Response of Lapicque's neuron membrane potential for different input currents.

neuron. The type of receptor will determine the synaptic current. AMPA receptors respond extremely quickly, while NMDA show a slow delay of PSC. This current is the input to the post-synaptic neuron and will affect the membrane potential response[13]. A first order approximation to model the behaviour of synaptic currents through time is an exponentially decaying current after each pre-synaptic spike as shown in Figure 2.4. This current decaying behavior is referred to the "synaptic leak".

In SNNs, the pre-senaptic spikes $S_j^{(l)}(t)$ trigger synaptic currents which in turn generate the input current. Conventionally, we will denote a spike train $S_j^{(l)}(t)$ with the sum of Dirac delta functions $S_j^{(l)}(t) = \sum_{s \in C_j^{(l)}} \delta(t-s)$ such that s iterate over the firing times $C_j^{(l)}$ of neuron j from layer l. Assuming synaptic currents sum linearly, their temporal dynamics are given by [32]:

$$\frac{\mathrm{d}I_i^{(l)}}{\mathrm{d}t} = -\frac{I_i^{(l)}(t)}{\tau_{syn}} + \sum_j W_{ij}^{(l)} S_j^{(l-1)}(t) + \sum_j V_{ij}^{(l)} S_j^{(l)}(t) \tag{2.2}$$

where we have now introduced the synaptic decay time constant τ_{syn}, and the synaptic weight matrices: $W_{ij}^{(l)}$ for feed-forward connections, and $V_{ij}^{(l)}$ for explicit recurrent connections within each layer.

To a rough approximation, the dynamics of spiking neurons can be described as an integration process (or a "summation" process) with a tunable leak, combined with an

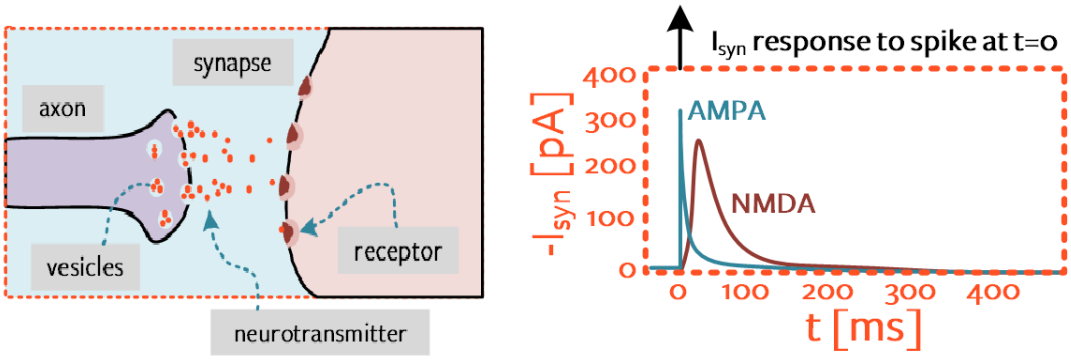


FIGURE 2.3: Schema of synaptic transmission and the dynamics of synaptic current [11].

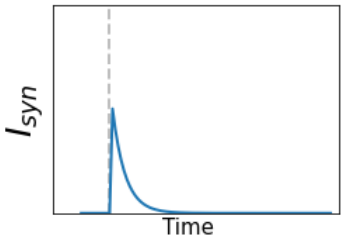


FIGURE 2.4: Synaptic current approximation.

action potential (or "spike") triggering process if a threshold voltage value is reached. In fact, action potentials of a given neuron have always approximately the same shape [13]. Therefore, the shape of the action potential cannot be used to transmit information. It is its presence or absence that contains the information. To that end, action potentials are perceived as "events" that happen at particular moments in time [13] as shown in Figure 2.5.

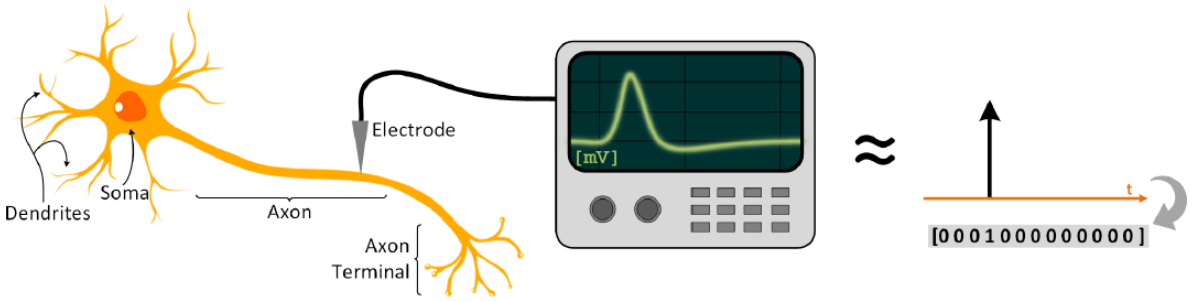


FIGURE 2.5: The neuron's spike approximated with a binary representation [11].

It is customary to approximate the solutions to the above equations in discrete time

assuming a small simulation time step $\Delta t > 0$. With no loss of generality, we assume $U_{rest} = 0$, $R = 1$, and the firing threshold $\vartheta = 1$. The output spike train $S_i^{(l)}[t]$ of neuron i in layer l is expressed as $S_i^{(l)}[t] \equiv \Theta(U_i^{(l)}[t] - \vartheta)$ where Θ is the Heaviside step function such that $S_i^{(l)}[t] \in \{0, 1\}$. t is used to denote the time step to indicate discrete time. The synaptic and membrane dynamics expressed respectively by Equation 2.2 and Equation 2.1 become [32]:

$$I_i^{(l)}[t+1] = \underbrace{\alpha I_i^{(l)}[t]}_{\text{Synaptic Leak}} + \underbrace{\sum_j W_{ij}^{(l)} S_j^{(l-1)}[t]}_{\text{Feed-Forward}} + \underbrace{\sum_j V_{ij}^{(l)} S_j^{(l)}[t]}_{\text{Explicit Recurrences}} \tag{2.3}$$

$$U_i^{(l)}[t+1] = \Big(\underbrace{\beta U_i^{(l)}[t]}_{\text{Membrane Leak}} + \underbrace{I_i^{(l)}[t]}_{\text{Synaptic Input}} \Big) \times \underbrace{(1 - S_i^{(l)}[t+1])}_{\text{Spike Reset}} \tag{2.4}$$

where the decay strengths are given by $\alpha \equiv e^{-\frac{\Delta t}{\tau_{syn}}}$ and $\beta \equiv e^{-\frac{\Delta t}{\tau_{mem}}}$, such that $0 < \alpha < 1$ and $0 < \beta < 1$ for finite and positive τ_{syn} and τ_{mem}. We should note to the distinct terms on the right hand side of both equations, each with its individual effect.

The instantaneous synaptic current I_i and the membrane potential U_i determine the state of neuron i as characterised by Equations 2.3 and 2.4. Figure 2.6[1] demonstrates how the computation necessary to update the cell state is unrolled in time [32]. At the bottom, input spikes $S^{(0)}$ travel propagate upwards to higher layer, while time steps run from left to right. In each time step, the synaptic current I is decayed by α and fed to the membrane potential U. Likewise, U is decayed by β. Applying a threshold nonlinearity to U generates spike trains S which affect the state of the network through resetting U after each spike, or through recurrent connections $V^{(1)}$.

There exists many extensions and variations to the Lapicque's neuron model. In order to find the right level of abstraction from biology to get the best performance in accurate, efficient and fast inference, we will derive and confront three variations with variable degrees of biological plausibility: the Integrate-and-Fire (IF), the Leaky Integrate-and-Fire (LIF), and the Current-Based Leaky Integrate-and-Fire (CUBA-LIF).

[1]The computational graph models a soft reset where "1" is subtracted from the membrane potential if the threshold is crossed. In this work, we used a hard reset such that the membrane potential is set to "0" after crossing the threshold as shown in Equation 2.3.

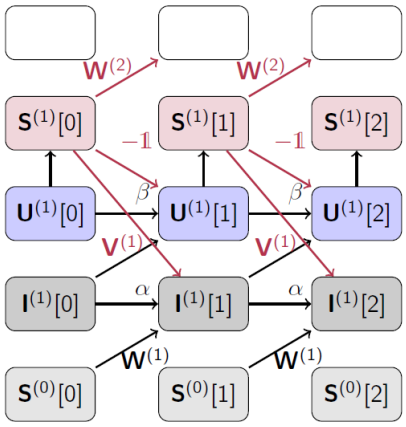


FIGURE 2.6: Computation graph of SNNs in discrete time [32].

2.3.1 Current-Based Leaky Integrate-and-Fire (CUBA-LIF)

The CUBA-LIF neuron is the most biologically plausible model among the three models. It accounts for the temporal dynamics of the input synaptic current caused by the gradual release of neurotransmitters from pre-synaptic neuron to post-synaptic neuron [11]. This neuron model is governed by Equations 2.3 and 2.4 presented before. It has two exponentially decaying terms: αI_i and βU_i. The degree of exponential decay of I_i and U_i is determined by the synaptic time constant τ_{syn} and membrane time constant τ_{mem}, respectively. Figure 2.7 illustrate the dynamics of a CUBA-LIF neuron for some random input stimuli.

2.3.2 Leaky Integrate-and-Fire (LIF)

The LIF neuron model is widely used in computational neuroscience to emulate the dynamics of biological neurons [33]. It integrates the input over time with a leakage such that the internal state represented by the membrane potential goes down exponentially. As shown in Figure 2.8, subsequent input spikes must be maintained for the state not to go to zero. In discrete time, the dynamics the LIF neuron is governed by:

$$I_i^{(l)}[t+1] = \sum_j W_{ij}^{(l)} S_j^{(l-1)}[t] + \sum_j V_{ij}^{(l)} S_j^{(l)}[t] \tag{2.5}$$

$$U_i^{(l)}[t+1] = (\beta U_i^{(l)}[t] + I_i^{(l)}[t]) \times (1 - S_i^{(l)}[t+1]) \tag{2.6}$$

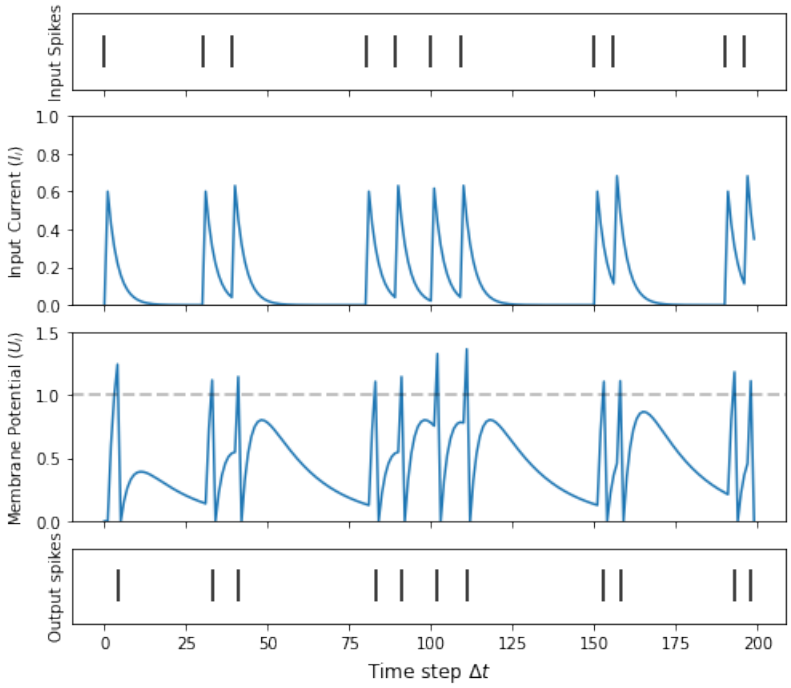


FIGURE 2.7: CUBA-LIF neuron dynamics.

2.3.3 Integrate-and-Fire (IF)

The IF neuron is the simplest and least biologically plausible model. It is the widely used model for implementing learning in spiking neurons[27]. We can think of the IF model as a LIF with no leak. It behaves as a standard integrator that keeps a running sum of its input. Thus, the internal state of the neuron is the mathematical integral of the input [8]. IF neurons do not have any inherent temporal dynamics. In discrete time, IF state can be written as:

$$U_i^{(l)}[t+1] = (U_i^{(l)}[t] + I_i^{(l)}[t]) \times (1 - S_i^{(l)}[t+1]) \tag{2.7}$$

Equations 2.5 and 2.7 do not have the decay (i.e., leak) parameters α and β. α is set to *zero* which causes the synapse to have an infinite leak. The current pulse width is now short, it effectively looks like a weighted spike. β on the other hand is set to *one*, which causes the membrane potential to remain constant between two consecutive spikes. Figure 2.9 illustrates the dynamics of an IF neuron for some random input stimuli.

Because this neuron is leaky, the leak parameter β is present. The strength of β, and hence the exponential decay of U_i, is determined by the membrane time constant

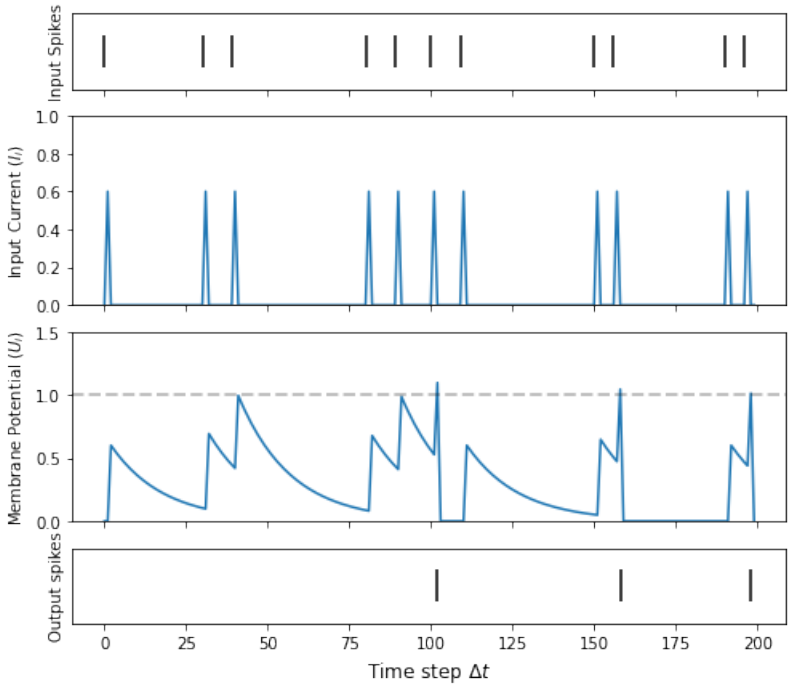


FIGURE 2.8: LIF neuron dynamics.

τ_{mem}. Similar to the IF neuron, the synaptic input current I_i can also be approximated by Equation 2.5 where α is set to zero and results in instantaneous jumps.

2.4 Mapping SNNs to RNNs

There are many shared properties between Recurrent Neural Networks (RNNs) and biological neural networks including learning through synaptic weights changes, temporal dynamics, and an analogous general architecture. Because of these similarities, a lot of research is mapping RNNs to SNNs. Establishing these equivalences allow for the transfer and application of existing RNNs training techniques to SNNs. In the rest of the report, the term RNNs will be used in the general sense to describe networks with temporal dynamics. In other words, networks that evolve with time due to explicit recurrent synaptic connections, or due to internal dynamics without any explicit recurrent connections. On the other hand, the term Recurrently Connected Neural Network (RCNN) will be used to refer to a network with explicit synaptic recurrent connections. Because we are only concerned with training SNNs, we will refer to Feed-forward SNNs as FSNNs and Recurrently connected SNNs as RSNNs.

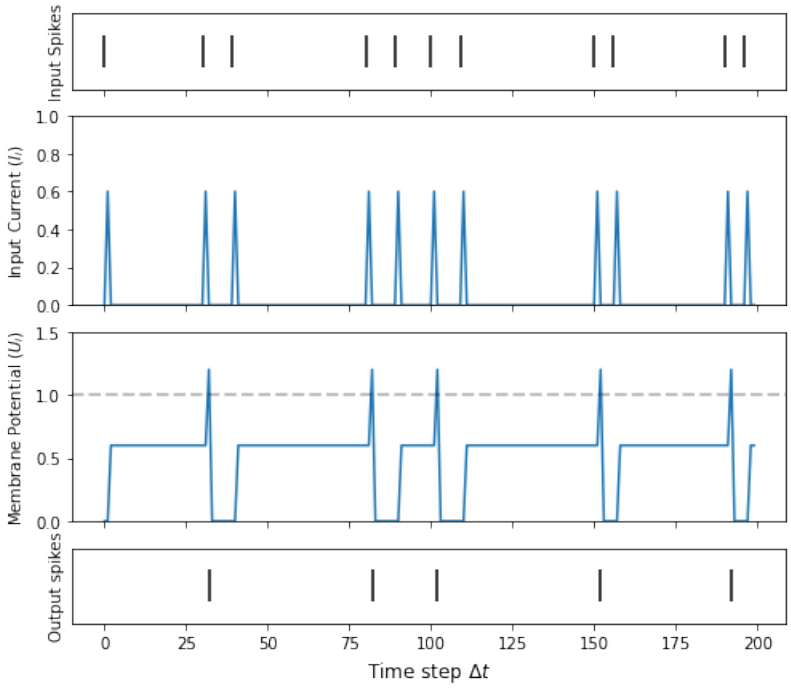


FIGURE 2.9: IF neuron dynamics.

2.4.1 Recurrent Neural Networks

In machine learning, RNNs are a class of stateful networks whose internal state evolves with time. They have been proven to be highly effective at solving problems related to real-time pattern recognition and noisy time series predictions [34]. RNNs can be formally described as networks of interconnected neurons where the network's state at any point in time $a[t]$ is a function of both external input $x[t]$ and the network's state at the previous time point $a[t-1]$. The dynamics of an RNN with L layers are given by:

$$\begin{aligned} y^{(l)}[t] &= \sigma(a^{(l)}[t]) \text{ for } l = 1, ..., L \\ a^{(l)}[t] &= V^{(l)} y^{(l)}[t-1] + W^{(l)} y^{(l-1)}[t-1] \text{ for } l = 1, ..., L \\ y^{(0)}[t] &= x[t] \end{aligned} \tag{2.8}$$

where $a^{(l)}[t]$ is the state vector of the neurons at layer l, σ is the activation function, $V^{(l)}$ the recurrent weights matrix of layer l, $W^{(l)}$ is the feed-forward weights matrix, and $x[t]$ is the external input at the first layer. Figure 2.10 illustrate the computational graph of a RNN.

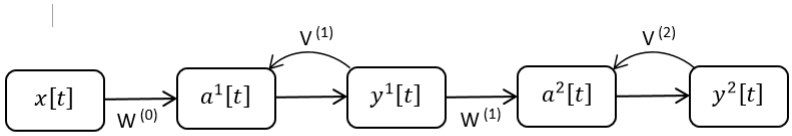


FIGURE 2.10: Example of a computational graph of a RNN.

2.4.2 Supervised Learning in SNNs

There exists powerful machine learning methods to train RNNs for many applications including language translation and automatic speech recognition. Earlier, we distinguished between RNNs with explicit recurrent connections RCNNs and those with implicit recurrences that arise due to its internal dynamics. SNNs thus, constitute a special case of RNNs. The formulation of spiking neurons in a discrete, recursive form allows us to take advantage of the developments in training RNNs.

A training process of neural networks consists of a loss function and mechanism to updates the network's learnable parameters (e.g., synaptic weights) to minimize the loss. A simple and very powerful mechanism to do this is to perform gradient descent on the loss function using the chain rule. This update rule is ubiquitous in deep earning and is known as The Back-Propagation algorithm (BP) [34] and can be written as:

$$W_{ij} \leftarrow W_{ij} - \eta \Delta W_{ij}, \text{ where } \Delta W_{ij} = \frac{\partial \mathcal{L}}{\partial W_{ij}} = \frac{\partial \mathcal{L}}{\partial y_i} \frac{\partial y_i}{\partial a_i} \frac{\partial a_i}{\partial W_{ij}} \tag{2.9}$$

where $a_i = \sum_j W_{ij} x_j$ is the total input to the neuron, y_i is the output of the neuron, and η is the learning rate.
For RNNs, The weight W is applied at every time step, and so the loss is also calculated at every time step. The influence of the weight on present and historical losses must be summed together to define the global gradient:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \sum_t \frac{\partial \mathcal{L}[t]}{\partial W_{ij}} \tag{2.10}$$

Applying BP to RNNs is referred to as Back-Propagation Through Time (BPTT). Next, the applicability of this algorithm to SNNs will be discussed.

2.4.3 The Non-Differentiability of Spikes

The unfolding process of computation in Figure 2.6, depicts the flow of information forward in time (from left to right) to calculate outputs and losses, and backward in time (from right to left) to compute the gradients. Before BPTT can be applied to SNNs, however, a serious challenge regarding the non-differentiability of the spiking non-linearity needs to be overcome.

BP requires the calculation of the derivative of the neural activation function. For a spiking neuron, however, the derivative of $S[t] = \Theta(U[t] - \vartheta)$ is zero everywhere except at $U = \vartheta$, where it tends to infinity as shown in Equation 2.11. This means the gradient will almost always be zero and no learning can take place. This behaviour of the binary spiking non-linearity makes SNNs unsuitable for gradient based optimization and it is known as the "*dead neuron problem*".

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial S} \underbrace{\frac{\partial S}{\partial U}}_{\{0,\infty\}} \frac{\partial U}{\partial I} \frac{\partial I}{\partial W} \tag{2.11}$$

2.4.4 Surrogate Gradient Descent

To overcome the dead neuron problem in training SNNs, many approaches have been proposed in the literature with varying degrees of success. These approaches ranges from restoring the entire biologically inspired models, translating conventionally trained neural networks to SNNs, to defining Surrogate Gradient (SG) as a continuous relaxation of the real gradients. The latter is the approach used in this work as it has been shown to train SNNs to unprecedented levels of performance [32].

The idea behind SG is that rather than changing the non-linearity itself, we only change the gradients. In other words, we keep the Heaviside step function the way it is during the forward pass and change the derivative term $\partial S / \partial U$ with something that does not stop learning during the backward pass. Specifically, we selected the fast sigmoid function for SG to smooth out the gradient of the Heaviside function:

$$\tilde{S} = \sigma(U_i^{(l)}) = \frac{U_i^{(l)}}{1 + \tilde{\beta} \left| U_i^{(l)} \right|} \tag{2.12}$$

where $\tilde{\beta}$ is the steepness parameter that modulates how smooth the surrogate function is. Figure 2.11 summarizes the dead neuron challenge (on the left) and the SG solution (on the right).

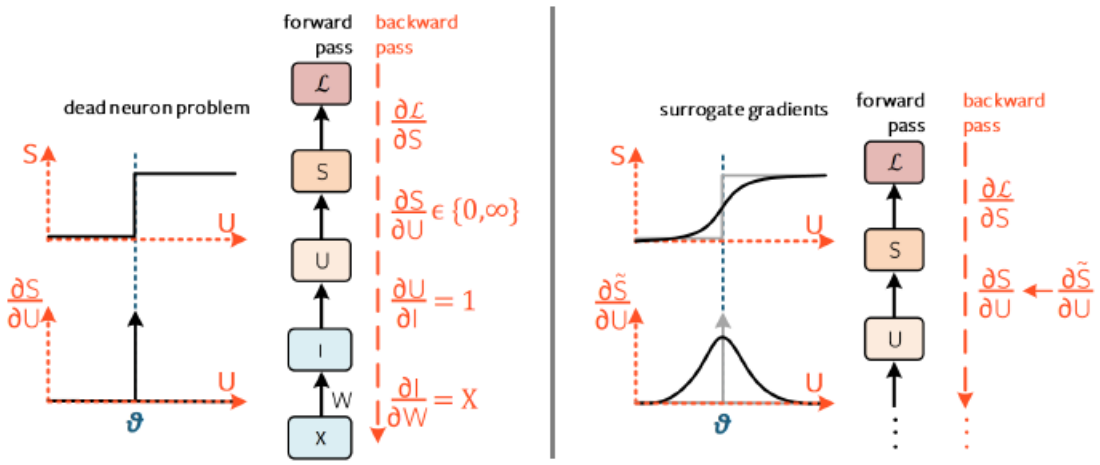


FIGURE 2.11: The dead neuron problem and SG solution [11].

2.4.5 Loss Function

In this work, cross entropy *max-over-time* loss function is chosen as it has been shown to give the best performance [35]. When called, the maximum membrane potential value for each output neuron in the readout layer is sampled and passed through the loss function. This cross entropy loss encourages the maximum membrane potential of the correct class to increase, and suppresses the maximum membrane potential of incorrect classes. On data with batch size of N_{batch} and N_{class} output classes, $\{(x_s, y_s) \mid s = 1, ..., N_{batch}; y_s \in \{1, ..., N_{class}\}\}$ the loss function takes the form:

$$\mathcal{L} = -\frac{1}{N_{batch}} \sum_{s=1}^{N_{batch}} \mathbb{1}(i = y_s) \log \left\{ \frac{\exp(U_i^{(L)}[\tilde{t}_i])}{\sum_{i=0}^{N_{class}} \exp(U_i^{(L)}[\tilde{t}_i])} \right\} \tag{2.13}$$

where $\mathbb{1}$ is the indicator function, and $\tilde{t}$ is the time step with the maximum membrane potential for each readout unit in the readout layer L, such that $\tilde{t}_i = \text{argmax}_t\, U_i^{(L)}[t]$. The cross entropy in Equation 2.13 is minimized using the Adamax optimizer[36].

2.5 Summary

In this chapter, we have laid the foundations of this work. We have also introduced three widely used neuron models that we will be confronting in our experimental part, each model with its level of computational complexity and biological plausibility. We have also explained how SNNs can be understood within the conceptual frame of RNNs due to their internal dynamics. Next, we discussed a key challenge concerning the all-or-nothing behaviour of binary spiking non-linearity that stops the gradients from flowing and prevents learning. The solution we will be using in our experiments is SG, which has been proven to train SNNs to unprecedented levels of performance. Finally, the cross entropy loss function is presented.

Chapter 3

Experiments, Results and Discussion

3.1 Introduction

This chapter is an overview of all the experiments we conducted in order to understand the effect spiking neurons leakages and network recurrences for spike-based spatio-temporal patterns recognition. A detailed analysis of of the results is also presented. First, we will start by introducing the used datasets and software tools. An investigation on the effect of neuron models is next. Then, the impact of adding explicit recurrent connections is studied. Finally, we will examine the role of neural heterogeneity in time constants and discuss all of our findings.

3.2 Datasets

Benchmarks datasets have played a crucial role in the tremendous development of frame-based computer vision. Using common metrics to quantitatively evaluate algorithms on benchmark datasets enables us to fairly and directly compare between different works, encourages competition, and motivates researchers by giving them state-of-the-art targets to reach. In this work, two widely used benchmarks in the neuromorphic community are considered: the Neuromorphic MNIST (N-MNIST) and the Spiking Heidelberg Digits (SHD).

3.2.1 Neuromorphic MNIST

The N-MNIST (short for Mixed National Institute of Standards and Technology) dataset [37] is the neuromorphic version of the ubiquitous frame-based MNIST benchmark. It is captured at the same visual scale as the original one (32x32 pixels) and has the

same 60,000 training and 10,000 testing samples. The N-MNIST dataset is created by moving an Asynchronous Time-based Image Sensor (ATIS) over each MNIST image, where each N-MNIST sample or spike train is 360 ms long. Patterns in the samples are represented as events, each occurring at a specific pixel location or address at a specific time using the AER protocol. Events extracted due to increased pixel intensity are denoted as ON events, and those with decreased pixel intensity as OFF events. If we accumulate those events over time and plot the bins as images, it looks like this:

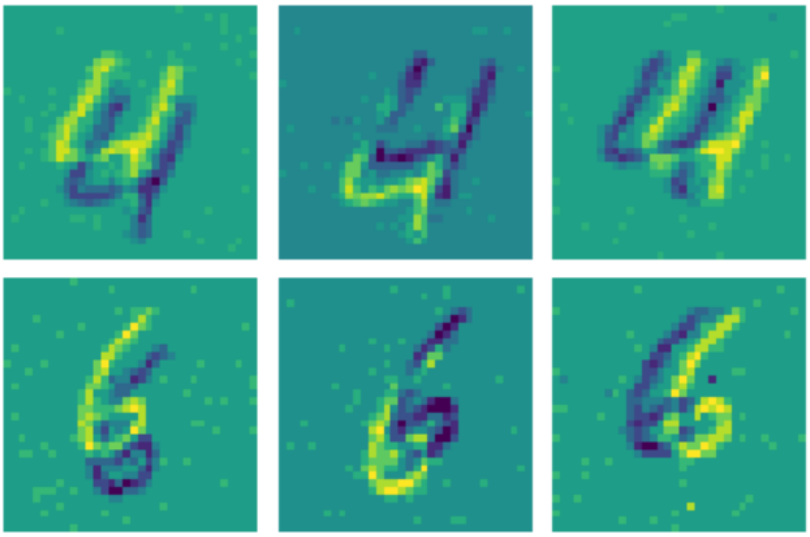


FIGURE 3.1: Samples from N-MNIST dataset [37].

3.2.2 Spiking Heidelberg Digits

The SHD dataset [35] is an audio-based classification dataset. Spikes in 700 input channels were generated using a sophisticated artificial cochlea model. It consists of approximately 10,000 high quality recordings of spoken digits from zero to nine in both English and German languages. Examples of spiking activity from the SHD samples are shown in Figure 3.2:

3.3 Tonic

Tonic[38] is a well tested and stable package that provides publicly available event-based vision and audio datasets and event transformations. This easy-to-use interface

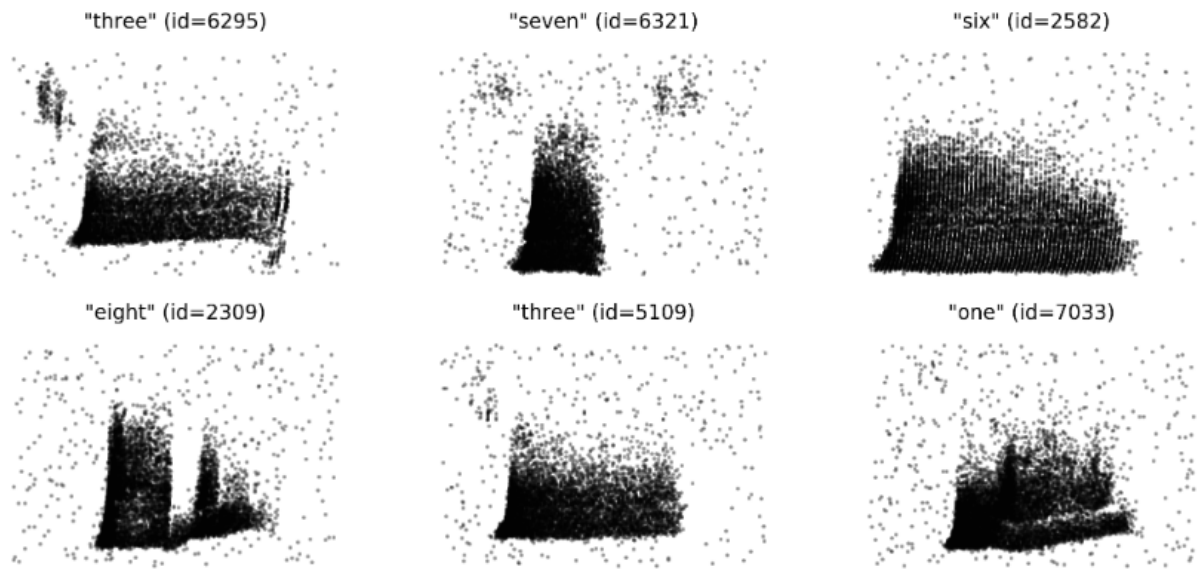


FIGURE 3.2: Samples from SHD dataset [35].

assists researchers with their daily endeavors by facilitating neuromorphic datasets' downloads as well as their conversions to different event representations.

Tonic supports both the event-based frameworks that works directly with events, and frame-based frameworks which might convert events into dense representations in some way. Many such frameworks rely on image datasets and convert pixel values to spikes to introduce a time dimension. Tonic emphasizes that event-based datasets are a good match for such frameworks instead of converting images to spikes artificially. It supports most of the common benchmarking datasets and is fully compatible with PyTorch Vision/Audio to allow for flexibility. In this work, Tonic is used to download and manipulate the N-MNIST dataset.

3.4 SpyTorch

SpyTorch [32] is a tutorial style open source repository that teaches the implementation of SG-based learning in SNNs. It contains a very low level of abstraction Python code. For easy BPTT, SpyTorch makes use of PyTorch's built-in automatic differentiation engine [39] which supports automatic computation of gradient for any computational graph.

3.5 Working with Neuromorphic Datasets

Neuromorphic data are representations of events. However, lists of events cannot be fed into a neural network. Therefore, raw data must be converted into an appropriate format such as a tensor. Tonic and SpyTorch allows us to use a set of transformations and apply them to our data before feeding it to our network. The neuromorphic camera sensor used to generate N-MNIST, as well as the artificial cochlea used to create SHD, have temporal resolutions in microseconds, which when converted into a dense representation ends up very large tensors. We used Tonic transformations and SpyTorch to bin events into larger time windows, which reduces temporal precision but enables us to conduct our experiments given the limited computational resources we have. Both SHD and N-MNIST events are accumulated into 14ms bins. These representations will also serve as time scale for our simulations.

3.6 Experimental Setup

We investigated the role of neurons leakages, network recurrences and neural heterogeneity in task performance by training SNNs to classify visual and auditory stimuli with varying degrees of temporal structure. We adopted a widely used minimal architecture consisting of three layers of spiking neurons: an input layer, a hidden layer with or without recurrent connections as shown in Figure 3.3[1], and a readout layer used to generate predictions. The readout layer consists of neurons that so not spike.

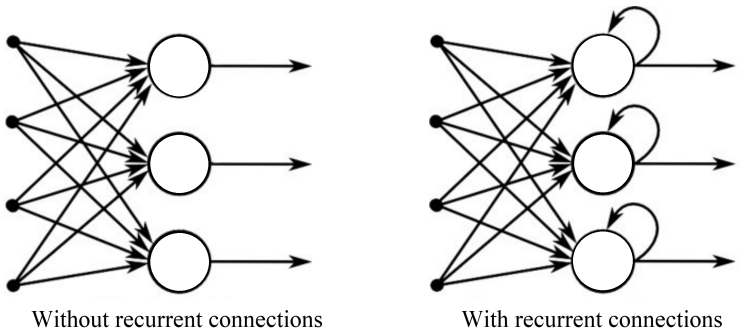


FIGURE 3.3: Hidden layer in FSNN vs. RSNN.

[1]This figure is used as a simplified demonstration. The RSNN used in the experiments is fully connected.

As described in the previous section, we used two datasets. The N-MNIST contains mostly spatial information. It features visual stimuli and has minimal temporal structure, as its samples are generated from static images. By contrast, SHD is auditory and has a rich temporal structure. To allow for fair baselines comparison of performance with previous works, we used the same train/test split suggested by the corresponding dataset authors in all of our cases. The network architectures and common hyper-parameters in our experiments such as the batch size, number of epochs, learning rate η, and steepness parameter $\tilde{\beta}$ were tuned according to state-of-the-art results obtained from the literature [35, 40, 30], as well as our own preliminary experiments in accordance with our limited computational resources. Table 3.1 gives a summary of all parameters used for our experiments. The performance of each configuration is quantified in terms of testing accuracy and sparsity. We note that all reported error measures in this work correspond to the standard deviation of three experiments.

TABLE 3.1: Hyperparameters used in our experiments.

	N-MNIST	**SHD**
Train/Test split	60,000/10,000	8332/2088
Network architecture	2312-200-10	700-200-20
Learning rate (η)	5×10^{-3}	2×10^{-4}
Time step (Δt)	14ms	14ms
Steepness parameter ($\tilde{\beta}$)	100	100
Batch size	256	128
Epochs	50	200

3.7 Impact of the Neuron Model

To assess the effect of the membrane and synaptic leakages, we started by confronting the three concerned neuron models: CUBA-LIF, LIF, and IF in a FSNN using both datasets. Only synaptic weights are learned and leakage parameters are treated as hyper-parameters and chosen to be homogeneous (i.e. the same for all neurons). Leakage parameters α and β are tuned using the synaptic time constant τ_{syn} and the membrane time constant τ_{mem}, respectively, as described by Equations 2.3 and 2.4 in the previous chapter.

3.7.1 Accuracy Analysis in FSNN

We started by the CUBA-LIF neuron where both leakages are of concern. This model can have a wide range of τ_{syn} and τ_{mem}. We performed a grid search across a number of time constants by fixing one and changing the other. Grid search is a simple hyper-parameters tuning technique that helped us evaluate the model for a wide range of combinations. Finding the "sweet spot" for the best classification accuracy requires a grid search across all possible combinations of time constants which is out of the scope of this work. Table 3.2a[1] shows the SHD testing accuracy results for our different combinations of τ_{mem} and τ_{syn}. Figure 3.4 also showcase the effect of τ_{syn} on accuracy for fixed values of τ_{mem} for the same dataset. We can see from our time constants sweeps that the best results seem to push α close to zero. In other words, CUBA-LIF performs better when its dynamics are close to those of the LIF neuron. This trend is also observed for the N-MNIST as shown in Table 3.2b. However, we can see a 10.18% drop between the best accuracy that reached 76.94% and the 66.76% worst accuracy for SHD, while only a 1.59% difference between the 97.41% best and the 95.82% worst accuracy for N-MNIST. This suggests that the leakages seem to have greater impact on data with rich temporal structure, than on data that is intrinsically spatial and low in temporal structure.

TABLE 3.2: CUBA LIF accuracy in FSNN.

(a) SHD

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	74.51% ± 0.68%	73.58% ± 0.45%	71.02% ± 0.45%	67.63% ± 1.73%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	75.79% ± 1.50%	73.79% ± 0.71%	71.19% ± 0.60%	66.76% ± 0.31%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	75.56% ± 0.27%	74.26% ± 1.55%	72.35% ± 0.89%	67.00% ± 1.04%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	76.94% ± 1.13%	75.99% ± 0.51%	72.61% ± 0.86%	67.54% ± 1.03%

(b) N-MNIST

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	97.37% ± 0.01%	96.90% ± 0.10%	96.25% ± 0.25%	95.88% ± 0.01%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	97.41% ± 0.07%	96.95% ± 0.13%	96.28% ± 0.07%	96.08% ± 0.17%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	97.37% ± 0.12%	96.94% ± 0.14%	96.52% ± 0.05%	95.91% ± 0.16%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	97.36% ± 0.10%	96.96% ± 0.20%	96.30% ± 0.07%	95.82% ± 0.11%

[1]Throughout this chapter, color scales used to better visualize data are representative of each table independently.

TABLE 3.3: LIF accuracy in FSNN.

(a) SHD

(ms)	$\tau_{syn} = 0$ **(**$\alpha \approx 0$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	75.06% ± 1.19%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	77.20% ± 0.66%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	76.88% ± 1.12%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	76.50% ± 0.13%

(b) N-MNIST

(ms)	$\tau_{syn} = 0$ **(**$\alpha \approx 0$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	97.63% ± 0.05%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	97.48% ± 0.10%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	97.48% ± 0.11%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	97.64% ± 0.08%

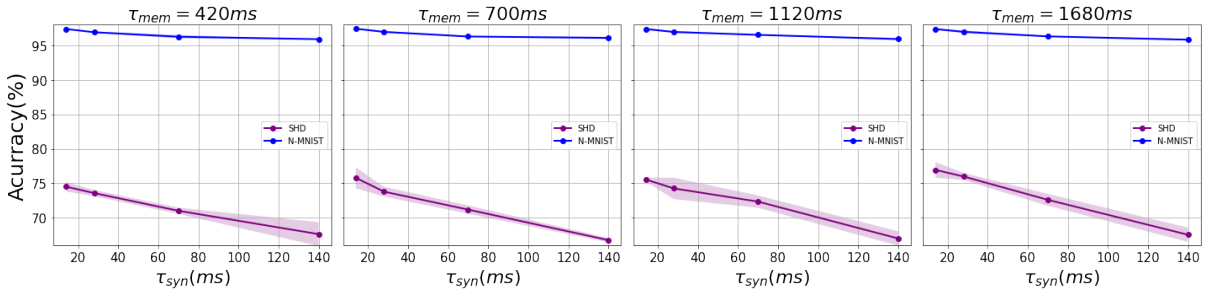


FIGURE 3.4: Effect of τ_{syn} on accuracy for fixed values of τ_{mem} in FSNN.

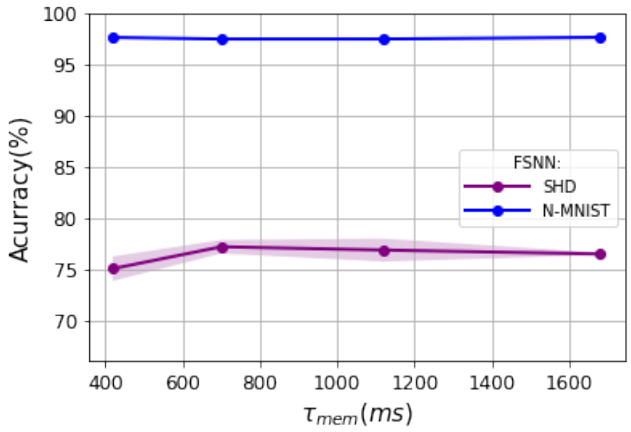


FIGURE 3.5: Effect of LIF's τ_{mem} on accuracy in FSNN.

The LIF neuron has an infinite synaptic leak with $\tau_{syn} = 0$ and a tunable membrane leak. So we varied τ_{mem} as a hyperparameter across a range of values. This range was selected according experiments done by [30, 41] where they suggested that short time constants caused the training loss to diverge. In our case, we selected $420ms$ as our smallest value of τ_{mem}.

The results of our experiments presented in Table 3.3 show that LIF neuron achieved higher accuracy than its CUBA-LIF counterpart for both datasets. Figure 3.5 suggests that the best accuracy also belongs to a sweet spot that requires a fine tuning of τ_{mem}.

TABLE 3.4: Best accuracy comparison between models in FSNN.

	CUBA-LIF	**LIF**	**IF**
SHD	76.94% ± 1.13% [1]	77.20% ± 0.66% [2]	78.36% ± 0.87%
N-MNIST	97.41% ± 0.07% [3]	97.64% ± 0.08% [4]	97.50% ± 0.06%

[1] $\tau_{mem} = 1680ms$, $\tau_{syn} = 14ms$
[2] $\tau_{mem} = 700ms$
[3] $\tau_{mem} = 700ms$, $\tau_{syn} = 14ms$
[4] $\tau_{mem} = 1680ms$

However, due to our selected narrow range, the obtained results are not very revealing on the effect membrane leak on accuracy. Due to time constraints and limitations of computational resources, we were not able to further explore values outside of the selected range. Nevertheless, we expect the performance to drop for smaller values of τ_{mem} as discussed above.

For the IF case, there is an infinite synaptic leak similar to LIF and no membrane leak. so the only possible values for time constants are $\tau_{mem} = \infty$ and $\tau_{syn} = 0$ which corresponds to $\beta = 1$ and $\alpha = 0$ respectively. In spite of its simplicity and lack of temporal dynamics, IF neuron was able to match or even outperform the other models by reaching a test accuracy of 78.36% for SHD and 97.50% for N-MNIST as shown in Table 3.4. This result suggests that introducing inherent temporal dynamics and increasing neuronal complexity does not necessarily lead to an improved classification accuracy even for data with rich temporal structure when using a feed-forward network.

3.7.2 Sparsity Analysis in FSNN

While learning performance is pivotal, it is also crucial to take into considerations the associated computational cost and energy consumption of using each model which are directly linked to the spiking activity of neurons when using a neuromorphic hardware like Intel Loihi [22] that computes asynchronously and exploits the sparsity of event-based sensing. To infer an output class, SNNs feed the input spikes over a number of time steps and perform event based synaptic operations only when spike-inputs arrive. These synaptic operations are considered as a metric for benchmarking neuromorphic hardware [22, 42]. We explored the impact of the leakages on the sparsity of each model by inferring the test set. Spiking activity recordings in the hidden layer shows that classification accuracy is proportional to sparsity. To put it another way,

we see an increase in sparsity as accuracy improves and a decrease as accuracy deteriorates. This is counterintuitive, and there might be a sweet spot that we do not see since we did not try to explicitly increase the sparsity until we see a decrease in accuracy if there are not enough spikes. Table 3.5 shows the number of spikes associated with each combination of time constants for the CUBA-LIF neurons. The decrease is sparsity with higher values of τ_{syn} is associated with the ability of CUBA-LIF neurons to sustain input spikes over a longer duration. However, more spikes do not lead to better performance, at least for our datasets. Table 3.6 presents the results for LIF neurons where we also observe some proportionality between sparsity and accuracy from Table 3.3.

TABLE 3.5: CUBA LIF hidden layer spiking activity in FSNN.

(a) SHD

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	1.73×10^6	1.61×10^6	1.85×10^6	2.84×10^6
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	1.68×10^6	1.79×10^6	1.92×10^6	3.01×10^6
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	1.53×10^6	1.59×10^6	1.68×10^6	3.11×10^6
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	1.48×10^6	1.42×10^6	1.88×10^6	3.57×10^6

(b) N-MNIST

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	4.49×10^6	4.94×10^6	6.84×10^6	6.40×10^6
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	4.39×10^6	5.19×10^6	6.09×10^6	5.94×10^6
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	4.30×10^6	4.63×10^6	5.90×10^6	6.03×10^6
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	4.40×10^6	5.02×10^6	5.33×10^6	5.83×10^6

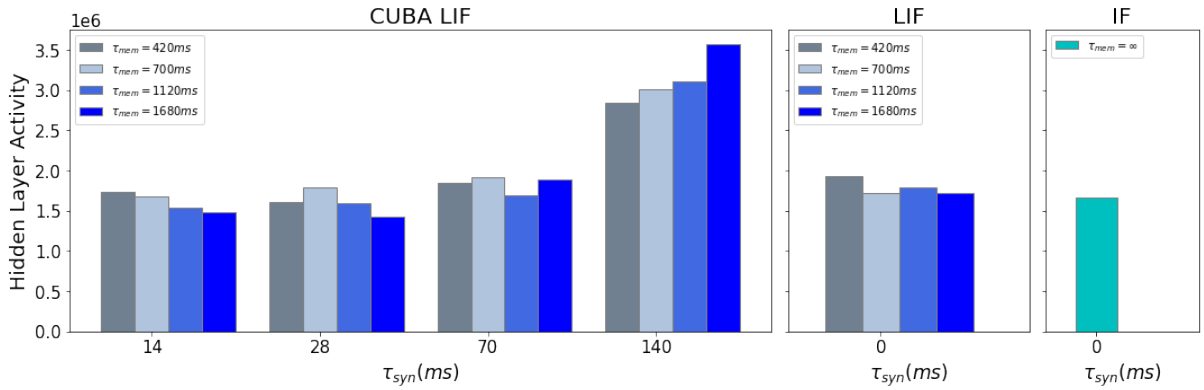
TABLE 3.6: LIF hidden layer spiking activity in FSNN.

(a) SHD

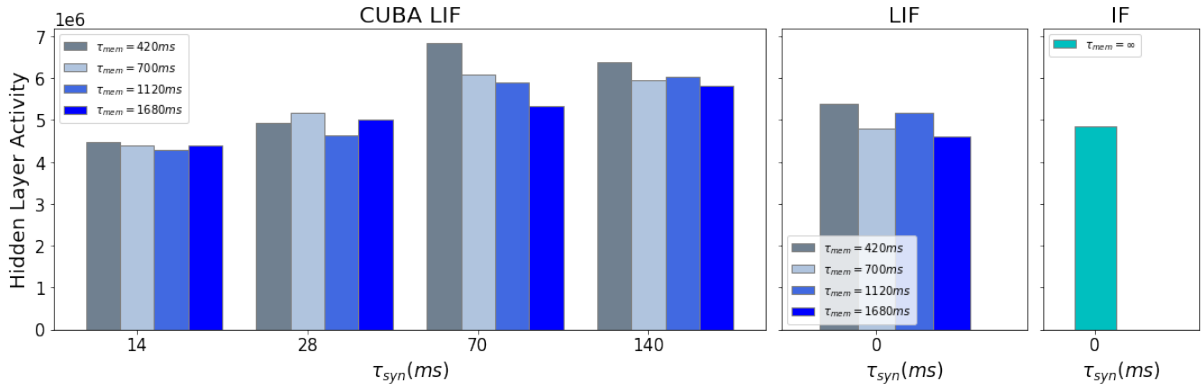
(ms)	$\tau_{syn} = 0$ **(**$\alpha = 0$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	1.93×10^6
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	1.71×10^6
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	1.79×10^6
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	1.71×10^6

(b) N-MNIST

(ms)	$\tau_{syn} = 0$ **(**$\alpha = 0$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	5.40×10^6
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	4.79×10^6
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	5.17×10^6
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	4.60×10^6



(a) SHD

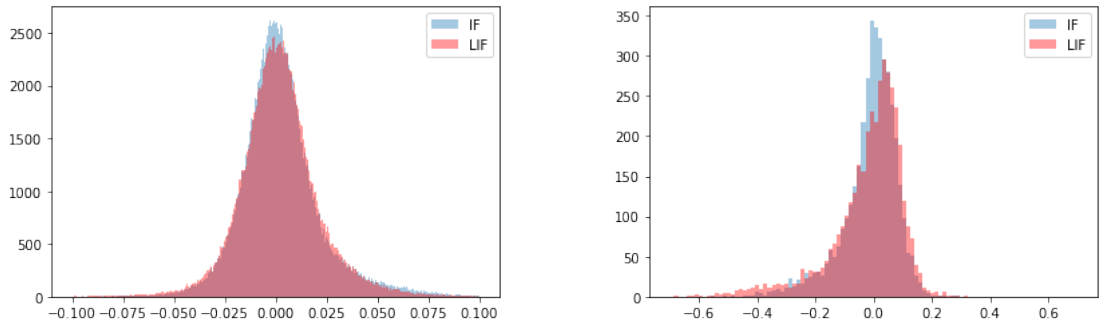


(b) N-MNIST

FIGURE 3.6: Hidden layer spiking activity of the three models in FSNN.

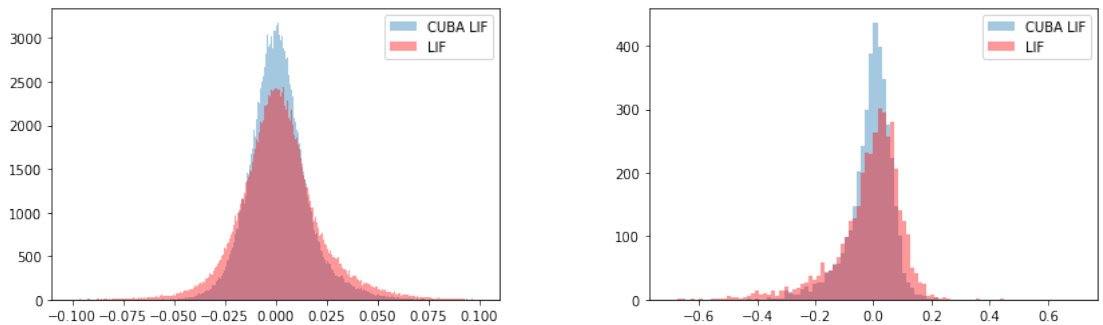
Intuitively, it is widely that if each of three neurons were to receive the same weighted sum of input, the LIF neurons would produce comparatively sparser outputs due to their infinite synaptic leak and the layer-wise decay of spikes caused by its membrane leak that acts as a forgetting mechanism. We plotted the spiking activity of each model as shown in Figure 3.6. For both datasets, We can see that IF neurons produced slightly less output spikes than most experiments of LIF neurons, which is counterintuitive. In an attempt to understand the cause of this non-intuitive result, We plotted the distributions of the trained weights for the LIF experimet that has the lowest sparsity ($\tau_{mem} = 420ms$) to compare it with the weights distribution of the IF as depicted in Figure 3.7. Because it is hard to inspect the distributions visually, we calculated the mean and standard deviation as shown in Table 3.7. We can see from the table that LIF's $W^{(2)}$ standard deviation is larger than that of the IF. This result suggests that BPTT tailors the LIF model to increase the synaptic weights beyond what is needed for the IF model.

The CUBA-LIF model, on the other hand, was able achieve the sparsest activity among the three models for certain combinations of time constants despite its ability to sustain input spikes for longer durations. To that effect, we plotted the weights distributions of CUBA-LIF's highest sparsity experiment ($\tau_{mem} = 1680ms$ and $\tau_{syn} = 14ms$) and compared it with that of the LIF experiment that has the same τ_{mem} value. Again, we calculated the mean and standard deviation for the distributions as shown in Table 3.8. We can see that CUBA-LIF'S $W^{(2)}$ standard deviation is larger than that of the LIF. Once more, this can be attributed to BPTT. Therefore, it seems that the leakages do not necessarily lead to sparser activity.



(a) $W^{(1)}$ (b) $W^{(2)}$

FIGURE 3.7: Trained weights distributions for IF vs. LIF.



(a) $W^{(1)}$ (b) $W^{(2)}$

FIGURE 3.8: Trained weights distributions for CUBA-LIF vs. LIF.

TABLE 3.7: Weights' mean and standard deviation for LIF vs. IF.

(a) Mean

Weights	**LIF**	**IF**
$W^{(1)}$	1.77×10^{-3}	$2.67 \times 10-3$
$W^{(2)}$	-2.07×10^{-2}	-1.23×10^{-2}

(b) Standard deviation

Weights	**LIF**	**IF**
$W^{(1)}$	1.98×10^{-2}	1.99×10^{-2}
$W^{(2)}$	12.58×10^{-2}	9.48×10^{-2}

TABLE 3.8: Weights' mean and standard deviation for CUBA-LIF vs. LIF.

(a) Mean

Weights	**CUBA-LIF**	**LIF**
$W^{(1)}$	1.33×10^{-3}	1.75×10^{-3}
$W^{(2)}$	-1.17×10^{-2}	-1.66×10^{-2}

(b) Standard deviation

Weights	**CUBA-LIF**	**LIF**
$W^{(1)}$	1.44×10^{-2}	1.99×10^{-2}
$W^{(2)}$	0.77×10^{-1}	1.15×10^{-1}

Our findings on training spiking neurons in FSNN indicate that the simple IF neuron could be sufficient for spatio-temporal classification tasks. If used to build neuromorphic hardware for embedded AI applications, IF neuron is very straightforward since all it does is summation. CUBA-LIF and LIF neurons in contrasts are more complex and require multiplication to implement the leakages, which could be very hardware expensive [27].

3.8 Impact of Explicit Recurrences

To study the effect of recurrences on learning spatio-temporal patterns, we added explicit recurrent connections to neurons in the hidden layer and confronted our three neuron models. Similar to the experiments in the FSNN, we performed a grid search across the same combinations of time constants for CUBA-LIF, a sweep for the same τ_{mem} values for LIF, and the same experiments for IF.

3.8.1 Accuracy Analysis in RSNN

As shown in Table 3.9 and Table 3.10, results of the SHD dataset show that recurrent architectures reached a significantly higher performance than their feed forward counterparts across all combinations of time constants for both CUBA-LIF and LIF. However, that is not the case for the N-MNIST. This is not surprising knowing the

inherent ability of RCNN to handle time series and sequential data. Figure 3.9 and Figure 3.10 compare between the performances of RSNNs and FSNNs with varying values of τ_{syn} and fixed values of τ_{mem}. For both datasets though, we can still observe the same trend as before such that CUBA-LIF performed better when α is close to zero, while the LIF reached the highest accuracy among the two. For the IF neuron, however, adding explicit recurrences reduced the accuracy by 0.43% on temporal data achieved similar accuracy on spatial patterns. The highest accuracies reached for each model are shown in Table 3.11.

Given IF's good performance in FSNN and inferior performance in RSNN, it becomes clear that leakages are important when there are both temporal information and a recurrent topology. This result is the most important finding of this work and our unique contribution to the neuromorphic computing literature.

TABLE 3.9: CUBA LIF accuracy in RSNN.

(a) SHD

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	81.96% ± 1.17%	81.71% ± 1.15%	77.05% ± 1.15%	75.15% ± 1.27%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	82.44% ± 0.60%	80.65% ± 0.83%	78.81% ± 1.04%	75.91% ± 1.59%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	82.74% ± 0.17%	80.73% ± 0.39%	78.89% ± 0.99%	75.52% ± 1.73%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	82.25% ± 0.99%	80.68% ± 0.89%	79.83% ± 1.37%	76.06% ± 1.25%

(b) N-MNIST

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	97.26% ± 0.09%	97.18% ± 0.46%	96.27% ± 0.08%	95.57% ± 0.55%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	97.35% ± 0.03%	96.81% ± 0.02%	96.08% ± 0.39%	95.88% ± 0.09%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	97.32% ± 0.02%	96.74% ± 0.03%	96.20% ± 0.04%	95.71% ± 0.45%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	97.22% ± 0.09%	96.80% ± 0.41%	96.21% ± 0.04%	95.95% ± 0.21%

TABLE 3.10: LIF accuracy in RSNN.

(a) SHD

(ms)	$\tau_{syn} = 0$ **(**$\alpha \approx 0$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	82.72% ± 0.33%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	83.06% ± 1.14%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	83.24% ± 0.74%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	83.41% ± 0.37%

(b) N-MNIST

(ms)	$\tau_{syn} = 0$ **(**$\alpha \approx 0$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	97.39% ± 0.08%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	97.44% ± 0.12%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	97.42% ± 0.11%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	97.41% ± 0.05%

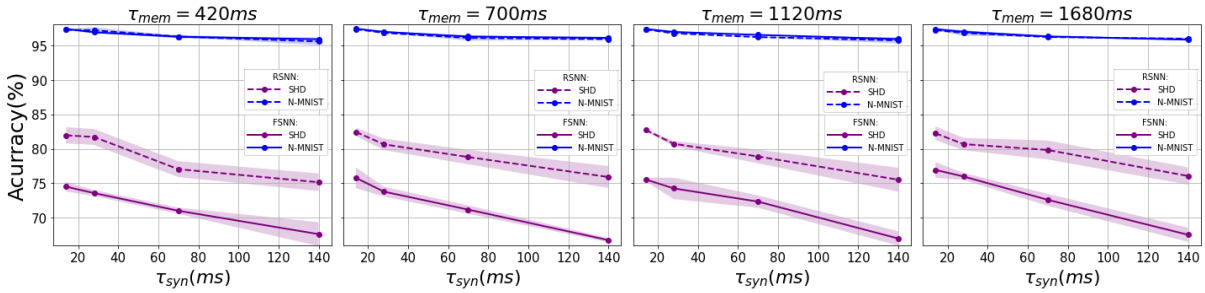


FIGURE 3.9: Effect of τ_{syn} on accuracy for fixed values of τ_{mem} in RSNN vs. FSNN.

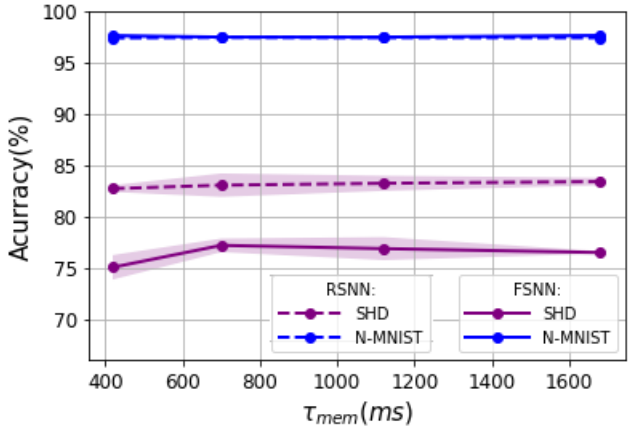


FIGURE 3.10: Effect of LIF's τ_{mem} on accuracy in RSNN vs. FSNN.

TABLE 3.11: Accuracy comparison between models in RSNN.

	CUBA-LIF	**LIF**	**IF**
SHD	$82.74\% \pm 0.17\%$ [1]	$83.41\% \pm 0.37\%$ [2]	$77.93\% \pm 0.33\%$
N-MNIST	$97.35\% \pm 0.03\%$ [3]	$97.44\% \pm 0.12\%$ [4]	$97.54\% \pm 0.06\%$

[1] $\tau_{mem} = 1120ms$, $\tau_{syn} = 14ms$
[2] $\tau_{mem} = 1680ms$
[3] $\tau_{mem} = 700ms$, $\tau_{syn} = 14ms$
[4] $\tau_{mem} = 700ms$

To the best of our knowledge, the highest accuracy we were able to reach on the SHD dataset is state-of-the-art for spiking neurons where only the weights are learned. We associate the results we obtained to a better tuning of time constants. Table 3.12 compares our best results with other works in the literature.

3.8.2 Sparsity Analysis in RSNN

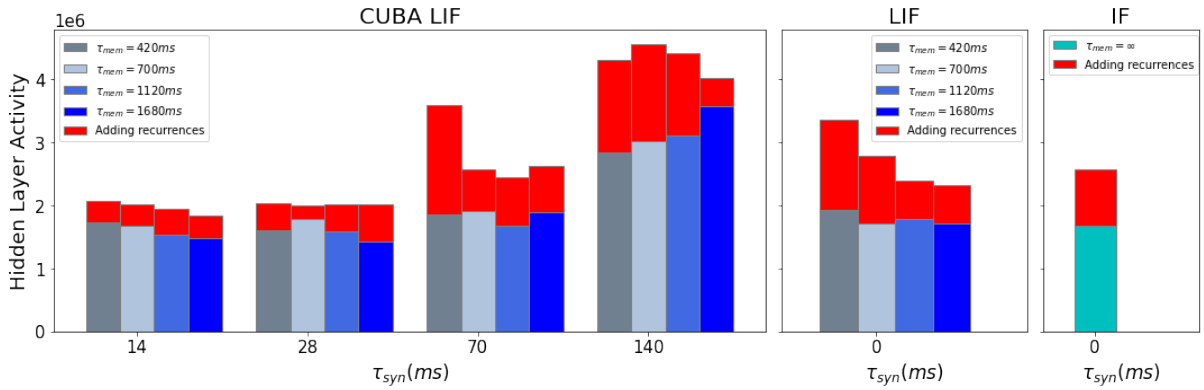
Similar to what we did in FSNN, we recorded the spiking activity of neurons in the hidden layer when the test set is inferred. SHD spikes count recordings plotted in Figure 3.11a show that explicit recurrent connections increases activity in all neurons for every combination of time constants. On average, we have 36.04% increase in spiking activity for CUBA-LIF, 51.92% for LIF, and 53.58% for IF. N-MNIST spikes count, on the other hand, did not increase for every combination time constants. It even decreased for some as shown in Figure 3.11b. On average, we have 4.12% increase for CUBA-LIF, 16.31% for LIF, and 15.75% for IF. It is hard to say whether or not the bigger increase in spiking activity for SHD contributed to its improved classification accuracy given that we saw similar increase for IF neurons but a worsened performance. Given that the CUBA-LIF experiments that resulted in classification performance that is almost as good as that of the LIF also resulted in the slightest increase in spiking activity. The CUBA-LIF model could be more suitable for low power applications especially if tuned better to reach even higher accuracy.

TABLE 3.12: Our best SHD accuracy results compared to [40] and [35].

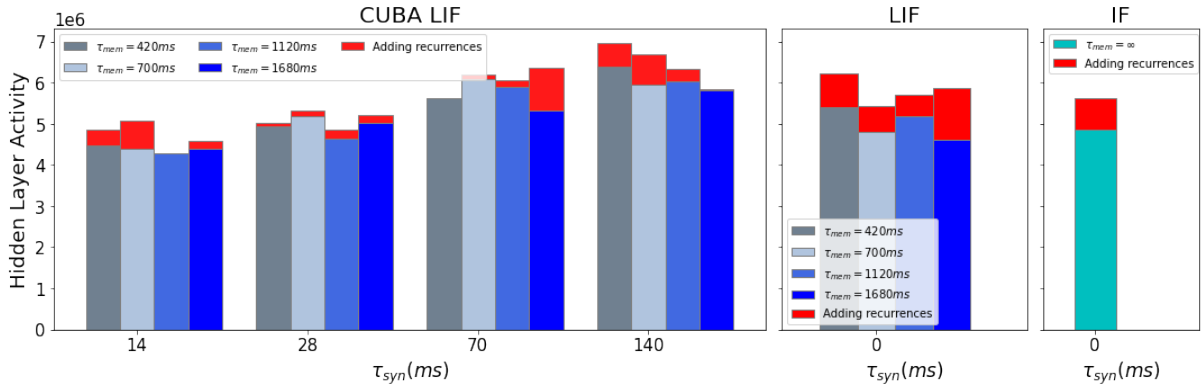
	Neuron model	**Accuracy**
[40]	CUBA-LIF	71.7%
[35]	CUBA-LIF	79.9%
This work	CUBA-LIF	**82.74%**
	LIF	**83.41%**

3.9 Impact of Neural Heterogeneity

Most existing learning methods learn the synaptic weights only while requiring a manual tuning of leakages-related parameters similar to our previously presented experiments. These parameters are chosen to be the same for all neurons, which could limit the diversity and expressiveness of SNNs. In biological brains, neuronal cells have different time constants. To assess whether this heterogeneity plays an important functional role or is just a byproduct of noisy developmental processes, we incorporated learnable time constants in the training process. Hence, τ_{mem} and τ_{syn} will not be treated as hyper-parameters, but learned parameters along with the synaptic weights. We refer to this training process as heterogeneous training.



(a) SHD



(b) N-MNIST

FIGURE 3.11: Impact of adding recurrences to the hidden layer spiking activity of the three models.

For both datasets, we compared two different conditions: values of time constants could be either all initialized to a single value and then trained (homogeneous initialisation), or initialized randomly according to a normal distribution and then also trained (heterogeneous initialisation). Since the IF neuron has fixed values of time costants: $\tau_{mem} = \infty$ and $\tau_{syn} = 0$, it is not concerned with heterogeneous training. On the other hand, LIF neuron has a fixed τ_{syn} equal to zero but a variable τ_{mem} which we were able to train. For CUBA-LIF, both time constants are trained.

3.9.1 Homogeneous Initialisation

To evaluate the performance of incorporating learnable time constants in comparison with the standard training in our previously presented experiments, we initialized τ_{mem} and τ_{syn} to the same values we used in our grid search for both CUBA-LIF and

LIF. We also conducted these experiments in both FSNN and RSNN. As can be shown in Tables 3.14 and 3.15 for FSNN and Tables 3.16 and 3.15 for RSNN, we found that incorporating learnable time constants did not have a profound impact on both datasets. For SHD, accuracy improved on average by 0.69% in FSNN and by 0.1% in RSNN. For N-MNIST, we saw no improvement at all. In fact, accuracy decreased in most cases. This result tells us that heterogeneity in time constants could further improve performance in tasks that are more temporally complex. Having said that, the results also show that performance is still sensitive to initial tuning of time constants since we can observe the same trend in the impact of τ_{syn} and τ_{mem} on accuracy for both CUBA-LIF and LIF neurons As can be seen in Figures 3.12 and 3.13. A comparison between best results obtained with standard training and heterogeneous training are shown in Tables 3.18.

TABLE 3.13: Our best SHD accuracy with heterogeneous training comparing with [40].

	Neuron model	**Accuracy**
[40]	CUBA-LIF	82.70%
This work	CUBA-LIF	**82.84%**
	LIF	**83.47%**

TABLE 3.14: CUBA LIF accuracy in FSNN for different initial values of time constants.

(a) SHD

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	76.07% ± 1.76%	74.80% ± 0.25%	71.52% ± 0.76%	66.23% ± 0.75%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	76.78% ± 0.64%	75.44% ± 1.05%	72.62% ± 2.07%	67.40% ± 1.17%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	76.23% ± 0.82%	75.52% ± 0.42%	71.94% ± 0.66%	67.89% ± 1.93%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	78.69% ± 0.69%	76.78% ± 1.35%	72.79% ± 0.88%	66.79% ± 0.28%

(b) N-MNIST

(ms)	$\tau_{syn} = 14$ **(**$\alpha \approx 0.368$**)**	$\tau_{syn} = 28$ **(**$\alpha \approx 0.606$**)**	$\tau_{syn} = 70$ **(**$\alpha \approx 0.818$**)**	$\tau_{syn} = 140$ **(**$\alpha \approx 0.905$**)**
$\tau_{mem} = 420$ **(**$\beta \approx 0.967$**)**	97.08% ± 0.21%	96.75% ± 0.05%	95.86% ± 0.06%	95.68% ± 0.06%
$\tau_{mem} = 700$ **(**$\beta \approx 0.980$**)**	97.24% ± 0.06%	96.74% ± 0.06%	95.91% ± 0.25%	95.65% ± 0.03%
$\tau_{mem} = 1120$ **(**$\beta \approx 0.987$**)**	97.15% ± 0.11%	96.65% ± 0.12%	96.14% ± 0.07%	95.51% ± 0.19%
$\tau_{mem} = 1680$ **(**$\beta \approx 0.992$**)**	97.08% ± 0.04%	96.86% ± 0.13%	96.20% ± 0.10%	95.50% ± 0.48%

TABLE 3.15: LIF accuracy in FSNN for different initial values of τ_{mem}.

(a) SHD

(ms)	$\tau_{syn} = 0$ ($\alpha \approx 0$)
$\tau_{mem} = 420$ ($\beta \approx 0.967$)	75.67% ± 1.08%
$\tau_{mem} = 700$ ($\beta \approx 0.980$)	78.09% ± 0.25%
$\tau_{mem} = 1120$ ($\beta \approx 0.987$)	78.02% ± 0.89%
$\tau_{mem} = 1680$ ($\beta \approx 0.992$)	79.84% ± 0.31%

(b) N-MNIST

(ms)	$\tau_{syn} = 0$ ($\alpha \approx 0$)
$\tau_{mem} = 420$ ($\beta \approx 0.967$)	97.41% ± 0.08%
$\tau_{mem} = 700$ ($\beta \approx 0.980$)	97.24% ± 0.03%
$\tau_{mem} = 1120$ ($\beta \approx 0.987$)	97.37% ± 0.11%
$\tau_{mem} = 1680$ ($\beta \approx 0.992$)	97.34% ± 0.14%

TABLE 3.16: CUBA LIF accuracy in RSNN for different initial values of time constants.

(a) SHD

(ms)	$\tau_{syn} = 14$ ($\alpha \approx 0.368$)	$\tau_{syn} = 28$ ($\alpha \approx 0.606$)	$\tau_{syn} = 70$ ($\alpha \approx 0.818$)	$\tau_{syn} = 140$ ($\alpha \approx 0.905$)
$\tau_{mem} = 420$ ($\beta \approx 0.967$)	81.48% ± 0.88%	81.48% ± 0.37%	79.00% ± 0.28%	75.58% ± 2.14%
$\tau_{mem} = 700$ ($\beta \approx 0.980$)	82.84% ± 1.17%	81.75% ± 0.86%	76.90% ± 1.40%	75.52% ± 2.25%
$\tau_{mem} = 1120$ ($\beta \approx 0.987$)	82.21% ± 0.60%	81.07% ± 0.81%	78.05% ± 1.86%	77.04% ± 0.75%
$\tau_{mem} = 1680$ ($\beta \approx 0.992$)	82.35% ± 0.44%	80.90% ± 1.05%	79.29% ± 1.37%	76.47% ± 0.68%

(b) N-MNIST

(ms)	$\tau_{syn} = 14$ ($\alpha \approx 0.368$)	$\tau_{syn} = 28$ ($\alpha \approx 0.606$)	$\tau_{syn} = 70$ ($\alpha \approx 0.818$)	$\tau_{syn} = 140$ ($\alpha \approx 0.905$)
$\tau_{mem} = 420$ ($\beta \approx 0.967$)	97.07% ± 0.19%	96.74% ± 0.18%	96.08% ± 0.32%	95.64% ± 0.20%
$\tau_{mem} = 700$ ($\beta \approx 0.980$)	96.92% ± 0.12%	96.79% ± 0.18%	96.17% ± 0.28%	95.59% ± 0.15%
$\tau_{mem} = 1120$ ($\beta \approx 0.987$)	97.14% ± 0.10%	96.55% ± 0.27%	96.00% ± 0.08%	95.76% ± 0.14%
$\tau_{mem} = 1680$ ($\beta \approx 0.992$)	97.02% ± 0.09%	96.49% ± 0.01%	96.00% ± 0.20%	95.41% ± 0.26%

TABLE 3.17: LIF accuracy in FSNN for different initial values of τ_{mem}.

(a) SHD

(ms)	$\tau_{syn} = 0$ ($\alpha = 0$)
$\tau_{mem} = 420$ ($\beta \approx 0.967$)	83.27% ± 0.59%
$\tau_{mem} = 700$ ($\beta \approx 0.980$)	83.47% ± 2.12%
$\tau_{mem} = 1120$ ($\beta \approx 0.987$)	83.10% ± 0.84%
$\tau_{mem} = 1680$ ($\beta \approx 0.992$)	83.10% ± 0.62%

(b) N-MNIST

(ms)	$\tau_{syn} = 0$ ($\alpha = 0$)
$\tau_{mem} = 420$ ($\beta \approx 0.967$)	97.25% ± 0.12%
$\tau_{mem} = 700$ ($\beta \approx 0.980$)	97.24% ± 0.21%
$\tau_{mem} = 1120$ ($\beta \approx 0.987$)	97.38% ± 0.03%
$\tau_{mem} = 1680$ ($\beta \approx 0.992$)	97.23% ± 0.15%

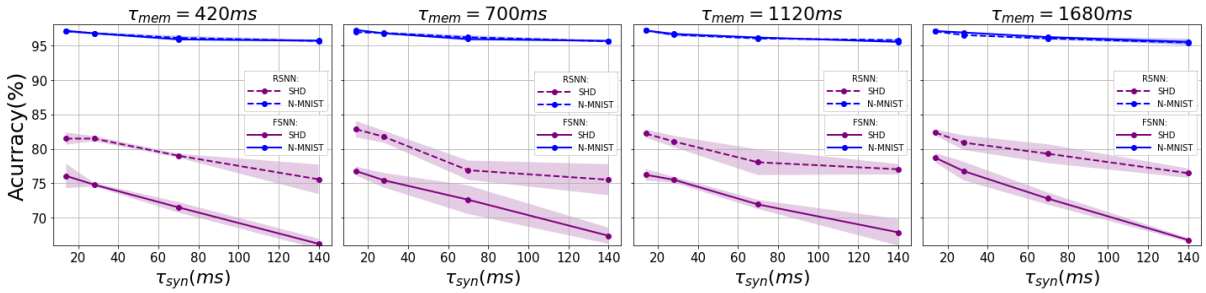


FIGURE 3.12: Effect of initial values of τ_{syn} on accuracy for fixed initial values of τ_{mem} in RSNN vs. FSNN.

TABLE 3.18: Comparison between best accuracies of standard vs. heterogeneous training.

(a) FSNN

	Neuron model	Standard training	Heterogeneous training
SHD	CUBA-LIF	76.94%	**78.69%**
	LIF	77.20%	**79.84%**
N-MNIST	CUBA-LIF	**97.41%**	97.24%
	LIF	**97.64%**	97.41%

(b) RSNN

	Neuron model	Standard training	Heterogeneous training
SHD	CUBA-LIF	82.74%	**82.84%**
	LIF	83.41%	**83.47%**
N-MNIST	CUBA-LIF	**97.35%**	97.14%
	LIF	**97.44%**	97.38%

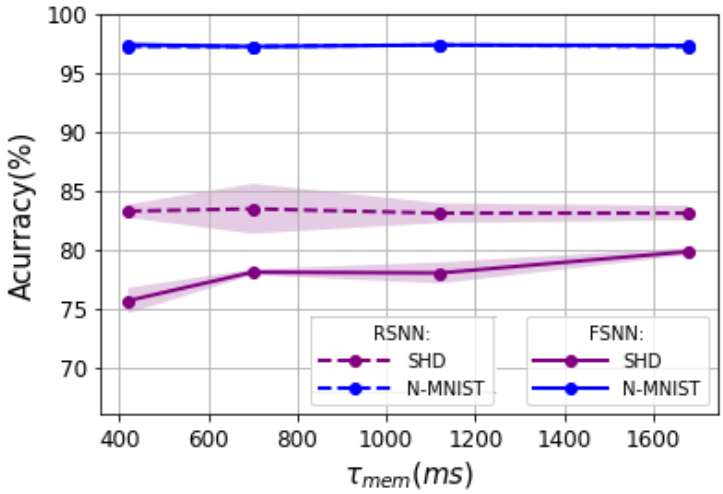


FIGURE 3.13: Effect of LIF's initial values of τ_{mem} on accuracy in RSNN vs. FSNN.

In all of our experiments, the resulting trained time constants distributions approximately match a gamma or log normal distribution as shown in Figure 3.15 for some randomly selected experiments. Nevertheless, it is not clear for us to suggest that the learned distributions may be optimal especially when better accuracies were obtained in standard training for N-MNIST.

Strikingly however, the resultants distributions are biologically plausible. We can see obvious similarities between our trained distributions and experimentally observed distributions of time constants in large number of neurons in different animals and

brain regions which are found on publicly available datasets[43, 44, 45]. Some samples are shown in Figure 3.14.

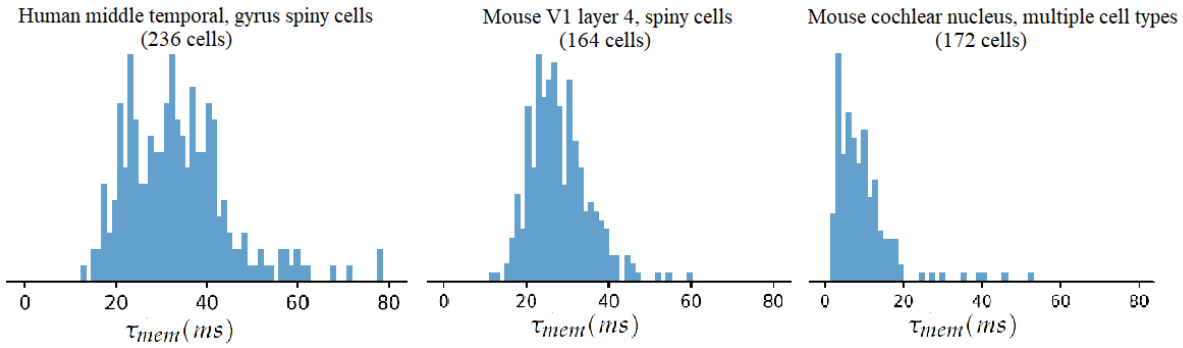
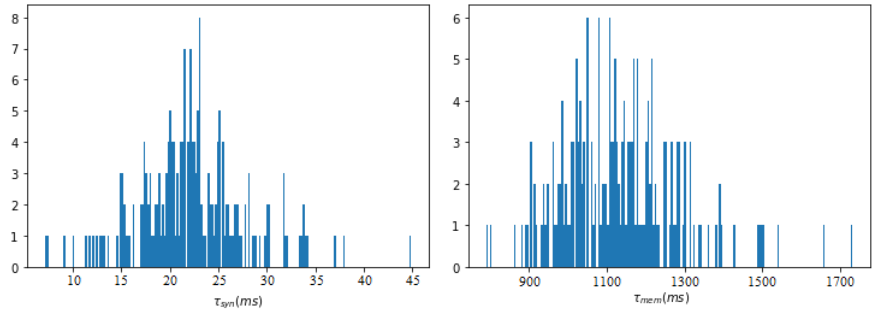
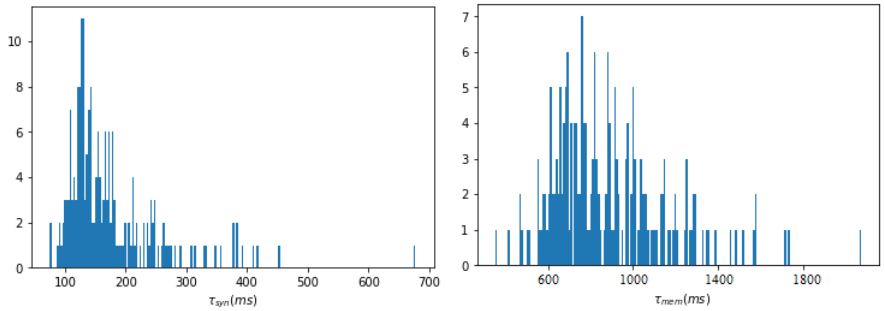


FIGURE 3.14: Experimentally observed distributions of time constants in biological neurons [43, 44, 45]



(a) SHD



(b) N-MNIST

FIGURE 3.15: trained time constants distributions.

Once again, to the best of our knowledge, our best SHD results in RSNN are state-of-the-art when time constant are trained as shown in Table 3.13.

TABLE 3.19: Heterogeneous Initialization results.

	SHD	**N-MNIST**
Mean τ_{syn}	14ms	28ms
Mean τ_{mem}	700ms	700ms
Accuracy in FSNN	76.53% $\pm$ 0.21%	96.80% $\pm$ 0.12%
Accuracy in RSNN	81.73% $\pm$ 0.38%	96.58% $\pm$ 0.14%

3.9.2 Heterogeneous Initialisation

We initialized the time constants of the CUBA-LIF neurons to a normal distribution. The means of the distributions were chosen according to some of the values that gave the best performances in the homogeneous initialization. The standard deviations were set approximately according to the distributions obtained in the homogeneous initialization as well. Results in Table 3.19 show that heterogeneous initialization did not outperform homogeneous initialization.

3.10 Discussion

In the neuroscience literature, it has been reported that leakages in biological neurons exist in many contexts such as sodium ion channels [46, 47] and synaptic transmission in the visual cortex [48]. Many spiking neuron models imitate this leaky behaviour through an exponential decay in the synaptic current and membrane potential. Other models prioritize computational simplicity and practicality than biomimicry. To tackle the lack in understanding of the effect of these leakages from the modeling perspective, we confronted three spiking neuron models with variable degrees of leaky behaviour, namely the CUBA-LIF, LIF, and IF, in classification tasks with a number of degrees of freedom.

We first trained SNNs constituting of our three neuron models in a feed-forward topology to classify auditory information using spoken digits from the SHD datasets and visual patterns using written digits from the N-MNIST dataset. Surprisigly, the IF model, despite the absence of leaky behaviour and the resulting lack of inherent temporal dynamics, outperformed the other models on the SHD and closely matched the LIF model on the N-MNIST. CUBA-LIF on the other hand, had the inferior performance among the three models on both datasets even with its intrinsic temporal dynamics caused by both synaptic and membrane leaks. We also found that CUBA-LIF reached higher accuracies when its dynamics are close to those of the LIF. In this

regard, it is safe to say that in a FSNN, leakages do not necessarily lead to an improved performance even on temporally complex tasks. In terms of sparsity, it was contradictory to our intuition to see sparser activity in IF neurons and CUBA-LIF neurons with smaller values of τ_{syn} than their LIF counterpart. After inspecting the trained weights distributions, it was clear that spike-based BPTT caused this counterintuition by tailoring LIF neurons to have bigger weights and hence more spikes. Therefore, we can say that leakages do not automatically lead to sparser activity. Furthermore, we noticed a correlation between sparsity and classification accuracy such that low sparsity always led to poor performance. It could be the case that if BPTT was not able to minimize the loss, it increased spiking activity in an attempt to perform better. Eventually however, it failed to do so. Overall, IF neurons seem to be sufficient in a feed-forward topology not only in terms of classification accuracy and sparsity, but especially so if we consider neuromorphic hardware design that is based on application-specific needs. IF architecture could be very cheap in terms circuit resources, in contrast to CUBA-LIF and LIF neurons that require multipliers to implement the leak, and thus resulting in expensive hardware. Nevertheless, we should still take into consideration the low-pass filtering effect of leakages that has been shown to eliminate high frequency components from the input and enhancing the noise robustness of SNNs especially in real-world applications [30].

Next, we added explicit recurrent connections to neurons in the hidden layer. Expectedly, we saw a remarkable improvement in performance for the SHD and no improvement at all for the N-MNIST. However, recurrences did not have any impact on the IF neuron on both datasets. It became clear to us that the temporal dynamics introduced by the leakages are only important when there exist both information with rich temporal structure and recurrent topology. In terms of sparsity, we saw bigger increase in spiking activity with the SHD than the N-MNIST. In both datasets, CUBA-LIF model added the smallest number of spikes especially with the time constants combinations that reached the best performances. Given this slight increase, CUBA-LIF could be an adequate choice for low-power applications particularly if time constants are fine tuned to further improve accuracy.

Finally, we introduced heterogeneity to our experiments by incorporating learnable time constants in the training process along with the synaptic weights. This heterogeneous training slightly improved performance particularly on SHD tasks which are intrinsically temporal. This learning paradigm also resulted in a consistant distributions of time constants that are close to a gamma or log-normal. These distributions

match experimental data observed across a large number of neurons in real nervous systems. Since we did not see a remarkable improvement in performance, it is possible that heterogeneous distributions of time constants observed in different brain regions in different animals is simply a product of noisy developmental processes. However, our models are only optimized for a single and relatively simple tasks, while animals in real environments deal with numerous extremely complicated tasks. Thereby, it is not possible to conclude whether or not these distributions are tuned for specific tasks without much more detailed models of particular brain regions in specific animals, along with the real-world challenges they face in their environment.

3.11 Summary

This chapter provided an overview of all of the experiments we ran. A thorough examination of the findings was also provided. First, we introduced the datasets and the tools that were employed. The next step was to look into the impact of neuron models. The impact of adding explicit recurrent connections was then investigated. Finally, we looked at the function of brain heterogeneity and discussed our findings.

Conclusion and Future Work

Throughout this report, we have seen that neuromorphic computing is an interdisciplinary field where many disciplines intersect including computer science, machine learning, embedded electronics, robotics, computational and theoretical neurosciences, etc. The field has emerged as a new generation of AI that can overcome the computational limitations and brittleness of the current generation, which depends on literal, deterministic views of events that lack context and commonsense understanding. Therefore, the next generation will expand AI into areas that correspond to human cognition. In order for neuromorphic computing to overcome the challenges associated with matching amimals flexibility and brains energy efficiency, it has to bridge the gap in understanding of how each of the factors determining the biological neuronal response can be effectively used in learning. For this reason, this work attempted to study the effect of spiking neurons' synaptic and membrane leakages, network explicit recurrences, and neuron level heterogeneity on learning spacial and temporal information.

In our experimental part, we first confronted the CUBA-LIF, LIF, and IF neuron models in a feed-forward network topology to classify written digits from the N-MNIST dataset as spacial patterns and spoken digits from the SHD datasets as temporal patterns. We found that the effects of the leakages were greater on the SHD than on the N-MNIST. However the IF neuron that does not have leaks, ourperformed the other neurons on the SHD by reaching an accuracy of 78.36% and closely matched the LIF neuron on the N-MNIST by reaching a 97.50% accuracy. Looking into the sparsity, BPTT caused the IF model to have sparser activity than the LIF neuron despite the lack of a forgetting mechanism. After adding explicit recurrent connections, accuracy significantly increased for the SHD when using CUBA-LIF and LIF neurons but improvement was seen on the N-MNIST. On both datasets however, the IF neuron underperformed the other neurons. We then tried to imitate neuronal heterogeneity by incorporating learnable time constants. This heterogeneity slightly improved performance on the SHD but not on the N-MNIST. It also resulted in a consistent biologically

plausible distributions of trained time constants. The SHD accuracies we were able to obtained in a RSNN were state-of-the-art by a large margin such that we reached 82.74% with CUBA-LIF and 83.41% with the LIF. The same thing in heterogeneous training where our state-of-the-art results are 82.84% with CUBA-LIF and 83.47% with the LIF.

The methods used to carry out these experiments are very computationally demanding. Indeed, we estimate that we have used more than "two months" of GPU computing time, without taking into account the preliminary experiments that were needed for preparation. This is an example of the inefficiency of today's hardware and its inability to fit the computational model. Overall, this work was a step forward for researchers in industry and academia to better understand the right level of biological abstraction needed to build efficient application-specific neuromorphic hardware, which is critical for the filed to move forward. The most important findings and contributions of our work can be briefly summed up as follows:

- Leakages are important when we have both temporal information and explicit recurrent connections in the network.
- Leakages do not necessarily lead to sparser activity.
- Time constants heterogeneity slightly improves performance when we have temporal information.

One of the limitations of our work is that our results only hold in benchmarking since we only used noise free benchmark datasets. In a real world scenario, such as a keywork spotting system, we have noise or even void. This noise can accumulate and create false positives. Hence, using the IF neuron that does not have any filtering effects could results in a poor classification performance. It would be interesting to further investigate the effects of leakages in noisy real world environments. We also did not explore the hardware implementation costs of the leakages. Although it has been shown in the literature that implementing the leak implies more hardware resources such as DSP blocks for multipliers [27, 49], a quantification of "how much" the extra costs entails is not well understood especially when moving from the LIF that only has the membrane leak, to the CUBA-LIF where both leakages are present. Given that the LIF model achieved a superior performance when compared to the CUBA-LIF, it is also important to investigate where the CUBA-LIF could be better. Due to

its ability to sustain spikes over longer durations, it could be when the temporal relations of the input data occur across long time-scales, when the input spiking pattern is sparse, or when it is important to care about the precise timing of the spikes. It is indeed interesting for further research. Furthermore, exploring how different types of heterogeneity, such as spatial and temporal heterogeneity, offer alternative benefits would be fascinating. We know that different cell types in the brain has distinct stereotyped distributions of time constants [44, 43, 45], and it would be intriguing to look into with multiple neuron types, including more biophysically realistic models such as the Hodgkin–Huxley and Izhikevich spiking neurons [33].

Bibliography

[1] Catherine D Schuman et al. "A survey of neuromorphic computing and neural networks in hardware". In: *arXiv preprint arXiv:1705.06963* (2017).

[2] Gordon E Moore et al. *Cramming more components onto integrated circuits*. 1965.

[3] R.H. Dennard et al. "Design of ion-implanted MOSFET's with very small physical dimensions". In: *IEEE Journal of Solid-State Circuits* 9.5 (1974), pp. 256–268. DOI: 10.1109/JSSC.1974.1050511.

[4] Christian Martin. "Post-dennard scaling and the final years of Moores Law". In: *Technical report* (2014).

[5] Belgacem Haba. *Moore's law is dead? Long Live Moore's law!* URL: https://adeia.com/blog/moores-law-is-dead-long-live-moores-law/?utm_content=205964717&utm_medium=social&utm_source=linkedin&hss_channel=lcp-78783796.

[6] "Estimation of energy consumption in machine learning". In: *Journal of Parallel and Distributed Computing* 134 (2019), pp. 75–88. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2019.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0743731518308773.

[7] Neil C. Thompson et al. *Deep Learning's diminishing returns*. URL: https://spectrum.ieee.org/deep-learning-computational-cost.

[8] Chris Eliasmith. *How to build a brain: A neural architecture for biological cognition*. New York, NY: Oxford University Press, 2013.

[9] Doosan Cho. "Memory Design for Artificial Intelligence". In: *International Journal of Internet, Broadcasting and Communication* 12.1 (2020), pp. 90–94.

[10] Alan Turing. "Intelligent machinery (1948)". In: *B. Jack Copeland* (2004), p. 395.

[11] Jason K Eshraghian et al. "Training spiking neural networks using lessons from deep learning". In: *arXiv preprint arXiv:2109.12894* (2021).

[12] Jack D Kendall and Suhas Kumar. “The building blocks of a brain-inspired computer”. In: *Applied Physics Reviews* 7.1 (2020), p. 011305.

[13] Wulfram Gerstner et al. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107447615.

[14] Mohammad-Hassan Tayarani-Najaran and Michael Schmuker. “Event-Based Sensing and Signal Processing in the Visual, Auditory, and Olfactory Domain: A Review”. In: *Frontiers in Neural Circuits* 15 (2021). ISSN: 1662-5110. DOI: 10.3389/fncir.2021.610446. URL: https://www.frontiersin.org/article/10.3389/fncir.2021.610446.

[15] Tobi Delbruck. “Frame-free dynamic digital vision”. In: *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*. Vol. 1. Citeseer. 2008, pp. 21–26.

[16] Christoph Sulzbachner and Jurgen Kogler. “A load balancing approach for silicon retina based asynchronous temporal data processing”. In: *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE. 2010, pp. 431–435.

[17] Yue Zheng, Yuan Cao, and Chip-Hong Chang. “A new event-driven dynamic vision sensor based physical unclonable function for camera authentication in reactive monitoring system”. In: *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE. 2016, pp. 1–6.

[18] Vincent Chan, Shih-Chii Liu, and Andr van Schaik. “AER EAR: A matched silicon cochlea pair with address event representation interface”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.1 (2007), pp. 48–59.

[19] Shiwei Wang et al. “Design of a silicon cochlea system with biologically faithful response”. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2015, pp. 1–7.

[20] Minhao Yang. “Silicon retina and cochlea with asynchronous delta modulator for spike encoding”. PhD thesis. ETH Zurich, 2015.

[21] Steve B. Furber et al. “The SpiNNaker project”. In: *Proceedings of the IEEE* 102.5 (2014). Cited by: 603; All Open Access, Bronze Open Access, pp. 652–665. DOI: 10.1109/JPROC.2014.2304638. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84900504664&doi=10.1109%2fJPROC.2014.2304638&partnerID=40&md5=4baf57c504029291858aad47c396e48e.

[22] 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.

[23] Filipp Akopyan et al. “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip”. In: *IEEE transactions on computer-aided design of integrated circuits and systems* 34.10 (2015), pp. 1537–1557.

[24] Ben Varkey 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.

[25] Nikola Kasabov et al. “Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition”. In: *Neural Networks* 41 (2013), pp. 188–201.

[26] Runchun M. Wang, Chetan S. Thakur, and André van Schaik. “An FPGA-Based Massively Parallel Neuromorphic”. In: *Frontiers in Neuroscience* 12 (2018). ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00213. URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00213.

[27] Lyes Khacef, Nassim Abderrahmane, and Benoît Miramond. “Confronting machine-learning with neuroscience for neuromorphic architectures design”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489241.

[28] Zidong Du et al. “Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches”. In: *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2015, pp. 494–507. DOI: 10.1145/2830772.2830789.

[29] Simon F Muller-Cleve et al. *Braille Letter Reading: A Benchmark for Spatio-Temporal Pattern Recognition on Neuromorphic Hardware*. 2022. DOI: 10.48550/ARXIV.2205.15864. URL: https://arxiv.org/abs/2205.15864.

[30] Sayeed Shafayet Chowdhury, Chankyu Lee, and Kaushik Roy. *Towards Understanding the Effect of Leak in Spiking Neural Networks*. 2020. DOI: 10.48550/ARXIV.2006.08761. URL: https://arxiv.org/abs/2006.08761.

[31] Nicolas Brunel and Mark CW Van Rossum. “Quantitative investigations of electrical nerve excitation treated as polarization”. In: *Biological Cybernetics* 97.5 (2007), pp. 341–349.

[32] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. *Surrogate Gradient Learning in Spiking Neural Networks*. 2019. DOI: 10.48550/ARXIV.1901.09948. URL: https://arxiv.org/abs/1901.09948.

[33] E.M. Izhikevich. "Which model to use for cortical spiking neurons?" In: *IEEE Transactions on Neural Networks* 15.5 (2004), pp. 1063–1070. DOI: 10.1109/TNN.2004.832719.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[35] Benjamin Cramer et al. "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–14. DOI: 10.1109/tnnls.2020.3044364. URL: https://doi.org/10.1109%2Ftnnls.2020.3044364.

[36] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[37] Garrick Orchard et al. "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades". In: *Frontiers in Neuroscience* 9 (2015). ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00437. URL: https://www.frontiersin.org/article/10.3389/fnins.2015.00437.

[38] Gregor Lenz et al. *Tonic: event-based datasets and transformations.* Version 0.4.0. Documentation available under https://tonic.readthedocs.io. July 2021. DOI: 10.5281/zenodo.5079802. URL: https://doi.org/10.5281/zenodo.5079802.

[39] Adam Paszke et al. "Automatic differentiation in PyTorch". In: (2017).

[40] Nicolas Perez-Nieves et al. "Neural heterogeneity promotes robust learning". In: *Nature communications* 12.1 (2021), pp. 1–9.

[41] Chankyu Lee, Syed Shakib Sarwar, and Kaushik Roy. "Enabling Spike-based Backpropagation in State-of-the-art Deep Neural Network Architectures". In: *CoRR* abs/1903.06379 (2019). arXiv: 1903.06379. URL: http://arxiv.org/abs/1903.06379.

[42] Paul A. Merolla et al. “A million spiking-neuron integrated circuit with a scalable communication network and interface”. In: *Science* 345.6197 (2014), pp. 668–673. DOI: 10.1126/science.1254642. eprint: https://www.science.org/doi/pdf/10.1126/science.1254642. URL: https://www.science.org/doi/abs/10.1126/science.1254642.

[43] Paul Manis, Michael R. Kasten, and Ruili Xie. *Raw voltage and current traces for current-voltage (IV) relationships for cochlear nucleus neurons.* 2019. DOI: 10.6084/M9.FIGSHARE.8854352. URL: https://figshare.com/articles/Raw_voltage_and_current_traces_for_current-voltage_IV_relationships_for_cochlear_nucleus_neurons_/8854352.

[44] Paul B. Manis, Michael R. Kasten, and Ruili Xie. “Classification of Neurons in the Adult Mouse Cochlear Nucleus: Linear Discriminant Analysis”. In: *bioRxiv* (2019). DOI: 10.1101/594713. eprint: https://www.biorxiv.org/content/early/2019/04/02/594713.full.pdf. URL: https://www.biorxiv.org/content/early/2019/04/02/594713.

[45] Michael J Hawrylycz et al. “An anatomically comprehensive atlas of the adult human brain transcriptome”. English. In: *Nature* 489.7416 (2012), pp. 391–399. ISSN: 0028-0836. DOI: 10.1038/nature11405.

[46] Dejian Ren. “Sodium Leak Channels in Neuronal Excitability and Rhythmic Behaviors”. In: *Neuron* 72.6 (2011), pp. 899–911. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2011.12.007.

[47] Terrance P. Snutch and Arnaud Monteil. “The Sodium “Leak” Has Finally Been Plugged”. In: *Neuron* 54.4 (2007), pp. 505–507. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2007.05.005. URL: https://www.sciencedirect.com/science/article/pii/S089662730700339X.

[48] Ömer B. Artun, Harel Z. Shouval, and Leon N Cooper. “The effect of dynamic synapses on spatiotemporal receptive fields in visual cortex”. In: *Proceedings of the National Academy of Sciences* 95.20 (1998), pp. 11999–12003. DOI: 10.1073/pnas.95.20.11999.

[49] Nassim Abderrahmane, Edgar Lemaire, and Benoît Miramond. “Design Space Exploration of Hardware Spiking Neurons for Embedded Artificial Intelligence”. In: *Neural Networks* 121 (2020), pp. 366–386. ISSN: 0893-6080.

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed
BOUGARA - Boumerdes

Institute of Electrical and
Electronic Engineering

Authorization for Final Year Project Defense

Academic year: 2021/2022

The undersigned supervisor: **Dr. Cherifi Dalila**
authorizes the student(s):

Bouanane Mohamed Sadek Option **Computer Engineering**

to defend his / her / their final year Master program project entitled:

Exploration of Spiking Neurons Leakages and Network Recurrences for Spike-Based Spatio-Temporal Pattern Recognition

during the [*] July [ ] September session.

Date: **23/06/2022**

The Supervisor

Dr. CHERIFI Dalila

The Department Head