TÉCNICO LISBOA

Neuromorphic System for the Identification of Three-Dimensional Spatial Maneuvers

Diogo Henrique Monteiro Silva

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Luís Miguel Teixeira d'Ávila Pinto da Silveira

Prof. Luís Jorge Brás Monteiro Guerra e Silva

Examination Committee

Chairperson: Pedro Filipe Zeferino Aidos Tomás

Supervisor: Prof. Luís Jorge Brás Monteiro Guerra e Silva

Members of the Committee: Prof. Nuno Filipe Valentim Roma

May 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Declaração

Declaro que o presente documento´e um trabalho original da minha autoria e que cumpre todos os requisitos do *Código* de Conduta e Boas *práticas* da Universidade de Lisboa.

Acknowledgements

I would like to thank to my supervisors Professor Luís Miguel Teixeira d'Ávila Pinto da Silveira and Professor Luís Jorge Brás Monteiro Guerra e Silva for the continuous support provided throughout the development of this work. This work could not have happened without their guidance, and I feel fortunate for the chance to share this experience with them.

I would also like to specially thank my family and friends who always tried to keep me motivated and give me strength to face any challenge ahead of me. Your support makes me want to be better. Thank you.

Abstract

This work presents an approach for developing a neuromorphic system, that relies on spiking neural networks to identify human workout activities. A tridimensional characterization of maneuvers was acquired, using gyroscope and accelerometer information, collected by a small IoT device (SensorTag). Running in place, torso rotation, jumping jacks and cross toe-touch were the performed maneuvers. A pre-existing firmware was adapted to enable sensor data collection at higher frequency rates using, Bluetooth Low Energy. The samples were subsequently encoded into spike trains using a population of neurons with spatial characteristics and fed into a spiking neural network. The network incorporates a competition layer that implements a one-winner-take-all mechanism to ensure that the competitive neurons distribute themselves to learn different patterns. These neurons implement the leaky-integrate-and-fire neuron model. Also, a learning rule based on synaptic-time-dependent-plasticity is used to train the competitive neurons on the input pattern. The system was able to reach a 100% recognition accuracy for a dataset comprising of activities performed by a singular participant.

Keywords

Spiking Neural Network, Leaky-Integrate-and-Fire, Synaptic-Time-Dependent-Plasticity, Winner-Take-All

Resumo

Este trabalho apresenta uma abordagem para o desenvolvimento de um sistema neuromórfico, baseado em redes neurais de impulsos, capaz de reconhecer atividades físicas. Uma caracterização tridimensional dos movimentos foi obtida através de leituras de um giroscópio e um acelerómetro, coletadas por um dispositivo de IoT (SensorTag). Um firmware foi especificamente desenvolvido para permitir a coleção de dados do sensor com frequência mais elevada, através de Bluetooth Low Energy. As amostras foram posteriormente codificadas em sequências de impulsos usando uma população de neurónios com características espaciais e alimentadas a uma rede neural de impulsos. A rede incorpora uma camada de competição que implementa um mecanismo de "um-vencedor-leva-tudo" para garantir que os neurónios competitivos se distribuem e aprendem diferentes padrões. Estes neurónios adotam o modelo LIF. Além disso, uma regra de aprendizagem baseada na plasticidade sináptica dependente do tempo é usada para ensinar os padrões de entrada aos neurónios competitivos. O sistema foi capaz de alcançar uma precisão máxima para um conjunto de dados composto por quatro atividades realizadas por um único participante.

Palavras Chave

Redes Neuronais Artificiais de Impulsos, Leaky-Integrate-and-Fire, Plasticidade Sináptica Tempo- Dependente, Winner-Take-All

Contents

1 Introduction 11

1.1 Motivation 12

1.2 Contributions 13

1.3 Thesis Outline 13

2 Related work 15

2.1 Artificial Neural Networks 16

2.2 Spiking Neural Networks 19

2.3 Simulating Spiking Neural Networks 22

3 Setup for data acquisition 27

3.1 SensorTag Development Kit 28

3.2 Bluetooth Low Energy 29

3.3 Firmware customization 31

3.4 Data acquisition and processing 31

4 Identification of three-dimensional manoeuvres using a spiking neural network 35

4.1 Objectives 36

4.2 Requirements 36

4.3 System design 36

4.4 Spiking Neural Network 37

4.4.1 Network Architecture 37

4.4.2 Spike Encoding 38

4.4.3 Implementation 40

5 Experimental results 45

5.1 Dataset analysis 46

5.2 Results 50

6 Conclusion 55

6.1 Discussion 56

6.2 Future work 56

List of figures

Figure 1 *Illustration of the neuron anatomy and of a discharge of an action potential at the synapse level* 16
Figure 2 *Physical circuit representation of a LIF neuron* 19
Figure 3 *Pixel values encoded with TTFS and rate encoding* 20
Figure 4 *Value encoding through a population of 5 neurons* 21
Figure 5 *Synaptic weight update according to the STDP learning rule* 22
Figure 6 *Effects of lateral inhibition* 23
Figure 7 *Learning pixel values using rate encoding and a trace based STDP rule* 25
Figure 8 Illustration of the high-level objects used in GATT transactions *30*
Figure 9 *Dataset overview* 32
Figure 10 *Block diagram of the proposed system for manoeuvres identification* 37
Figure 11 *Implemented network architecture* 38
Figure 12 *Example of the proposed encoding spatial-dependent spike encoding scheme* 40
Figure 13 *Illustration of neural response synchronization using delayed synapses* 42
Figure 14 *Response of 9 WTA neurons to a class* 43
Figure 15 *Dataset distribution across per class and sensor axis* 46
Figure 16 *Dataset movements spike patterns* 47
Figure 17 *Spatial distribution of the movements spike patterns over time, referring to the gyroscope* 48
Figure 18 *Spatial distribution of the movements spike patterns over time, referring to the accelerometer* 49
Figure 19 *Dynamics of the implemented network for different input rates* 50
Figure 20 *SNN accuracy for two learning rates and multiple input rates* 51
Figure 21 *WTA neurons final weights* 52
Figure 22 *Spatial representation of the WTA neurons final weights for both sensors* 52
Figure 23 *Recognition neurons response to streams of movement repetitions* 53

List of Tables

Table 1 *Boolean functions computed by M-P neurons* 18
Table 2 *MPU-9250 accelerometer and gyroscope specifications* 28
Table 3 *BLE specifications* 29
Table 4 *SensorTag movement sensor BLE characteristics* 30
Table 5 *SNN parameters* 43
Table 6 *Dataset attributes* 47
Table 7 *Recognition accuracy*. 51

Acronyms

AI	*Artificial Intelligence*
ANN	Artificial Neural Network
STDP	Synaptic-Time-Dependent-Plasticity
SNN	Spiking Neural Network
TTFS	Time-To-First-Spike
WTA	Winner Take All
LTP	Long-Term Potentiation
LTD	Long-Term Depression
M-P	*McCulloch-Pitts*

1

Introduction

1.1 Motivation 12
1.2 Contributions 12
1.3 Thesis Outline 13

This chapter provides with a high-level overview of the contents that will be discussed throughout this thesis. Some insights about the nature of neurons and their inter-connections are given, as well as how these characteristics can transcribe into representative artificial models. The purpose of this work is outlined as well as its main contributions. Finally, the thesis structure is detailed to provide a better understanding of the document content.

1.1 Motivation

The use of artificial intelligence (AI) is becoming commonplace. It has proven to be a prominent solution for problems in important research and industrial areas such as natural language, image, and video processing. Making data-based predictions, building recommendations systems that analyze user behaviors through their historical activity data to provide customized suggestions, or to transcribe recorded speech into text or commands are some of the applications empowered by this technology. These systems are designed to follow logically structured and human-like decision-making patterns. As humans, AIs support their decisions based on what they know. Machine learning is an AI field dedicated to virtually reproducing intelligent systems capable of learning and having memory-like features. This virtual knowledge can then be accessed by AIs to provide any information needed to perform a specific action. Some forms of artificial intelligence take inspiration in biological learning systems found in animals [1]. Neuroscience is a multidisciplinary field dedicated to study the dynamics and characteristics of neurons and neural networks. As the scientific knowledge towards understanding these natural mechanisms grows, better computational representations of these systems emerge. Different neuron models grant distinctive features to the artificial networks they belong to. Their activation function defines their response to any input. The McCulloch-Pitts model [2] represents neurons as simple binary threshold units. More complex neuron models use non-linear and continuous activation functions. The non-linearity enables the artificial neural networks (ANNs) to extract non-linear relations in data and the continuous nature of the activation function allows for differentiation. This last feature made possible the emergence of the most commonly adopted algorithm for supervised learning in ML, known as backpropagation. Another model, which most resembles the biological neuron is the spiking neuron model. It adopts a representation that, as in the biological neuron, uses impulses for inter-neuron communication, Impulses which encode spatiotemporal events. It is still not known how the brain encodes information. Although this mechanism is yet to be understood, there has been a realization on how inter-neurons connections are stimulated.

Motion analysis plays a crucial role in various domains such as sports science, healthcare, robotics and human-computer interaction. Drone and robot control, elderly, and patient surveillance are some specific applications. Such analysis, based on data characterizations of human movement patterns, enables computerized systems to autonomously react to predicted behaviors (e.g. monitor patient falling). Numerous machine learning based solutions have been develop to solve motion analysis problems, demonstrating their effectiveness in various applications. However, most existing solutions present high computational and energy costs. This can pose challenges when deploying such solutions in devices with limited processing power and

energy constraints, such as IoT devices and smartwatches. As a promising alternative, spiking neural networks (SNNs) have the potential to address these challenges effectively.

In this work, an artificial neural network composed of spiking neurons was trained using a biologically plausible learning mechanism, known as Synaptic-Time-Dependent-Plasticity (STDP). The network architecture also incorporates inhibitory connections to evoke competition between the learning neurons. This led to a winner-take-all (WTA) system where neurons distribute themselves to learn distinct patterns. A small dataset of human activities consisting of warm-up physical exercises was compiled, namely: jumping jacks, running in place, torso rotation and cross-toe touch. These movements are performed repetitively and in a rhythmic manner, resulting on a stream of activities with a periodic nature. Jumping jacks consists in simultaneously raising both arms while also spreading both legs, into a star shape, and returning to the initial position (arms close to body and legs together). Running in place mimics regular running with knees lifted and arms pumping. Torso rotation consists in rotating the upper body from side to side, while keeping the still. Finally, cross-toe touch requires standing with legs apart and stretched, while touching each foot with the hand from the opposite side, alternatively. After the network was trained on this dataset and its recognition accuracy was measured against a testing set. The samples were acquired with a gyroscope and an accelerometer and subsequently encoded into spike sequences using an encoding scheme that uses a population of neurons with spatial characteristics. Following the learning process, the competitive neurons behavior is analyzed to set up delayed connections that are established with a recognition layer.

1.2 Contributions

This work proposes a potential methodology for developing a neuromorphic system that can learn spatiotemporal patterns, consisting of human workout activities, in a semi-unsupervised manner. The proposed method shows that a SNN featuring an unsupervised competitive STDP learning rule, alongside supervisedly-tuned delayed synapses, can effectively differentiate among four distinct spatiotemporal patterns. A dataset, consisting of such patterns, was collected using motion sensors built in a microcontroller. A pre-existing firmware was adapted to address and enhance the acquisition of data. This thesis also investigates the use of input neurons with spatial characteristics to encode the compiled dataset.

1.3 Thesis Outline

This thesis is organized as follows:

- Chapter 1 – Introduction: The first chapter intends to establish the context and purpose of the research, outline its main contributions, and introduce the structure of this document.
- Chapter 2 – Related work: This chapter aims to provide an overview of the biological foundations that inspired the development of ANNs. Additionally, this chapter will delve with more detail about SNNs, as well as how they can be simulated and through specialized SNN simulators.

- Chapter 3 – Setup for data acquisition: This chapter focuses on the configuration employed for data acquisition and processing, while also addressing the components and technologies that support it.
- Chapter 4 – Identification of three-dimensional maneuvers using a spiking neural network: This chapter describes the objectives, requirements, and system design of the proposed spiking neural network for identifying three-dimensional maneuvers. It also details the network architecture, spike encoding, and implementation.
- Chapter 5 – Experimental results: This section presents the classification results of the proposed spiking neural network in identifying three-dimensional maneuvers using the dataset described in the previous chapter.
- Chapter 6 – Conclusion: The final chapter summarizes the findings of the research, provides a discussion of the results, and outlines future work.

2

Related work

2.1 Artificial Neural Networks 16
2.2 Spiking Neural Networks 19
2.3 Simulating Spiking Neural Networks 22

This chapter offers an overview of Artificial Neural Networks, including historical discoveries and foundational concepts. Also, it introduces Spiking Neural Networks and describes some tools for simulating them, with emphasis in the Brian2 library.

2.1 Artificial Neural Networks

The ability to accurately model and analyze data has become crucial in today's world with many companies using it to model customer behaviors and make informed data-driven decisions. In healthcare, researchers use it to predict patient outcomes and identify disease patterns. Artificial Neural networks (ANNs) have become a prominent technology for uncovering and modelling such behaviors and patterns. With architectures inspired in biological learning systems found in animals [3], ANNs result from the aggregation of interconnected nodes, termed as neurons, that function as the networks computational units. These systems are able to acquire knowledge (or 'learn') from multiple observations in a process designated as training, and later become able to classify new inputs based on what they have learned in the training phase.

In animals, neurons' aggregations form large and complex webs such as the nervous system [4]. This particular system is known to regulate animal's actions and sensory inputs by carrying electrical impulses all over the body. The neuron, as illustrated in Figure 1, is the elementary unit that defines the nervous system. It is a cell that can switch between an electrically excited or inhibited state by interacting with other neurons. These interactions consist of electrical stimuli that travel through neural networks. These stimuli, or impulses, are transmitted from the axon of a neuron to the dendrites of other neurons, to which it is connected by synapses. Each neuron has a membrane potential that is sensitive to the impulses that arrive at its dendrites. When no impulses reach the dendrites, the neuron rests at a resting potential. As more impulses reach the dendrites, the membrane potential increases. When the membrane potential reaches a threshold value, the neuron will fire, whereby an impulse will leave the cell body through its axon and reach the axon terminals, connected to the dendrites of neighboring neurons. The cell's membrane potential will thus rapidly decrease to a resting value (depolarization). This biological event is known as action potential and is the basis for inter-neuron communication.

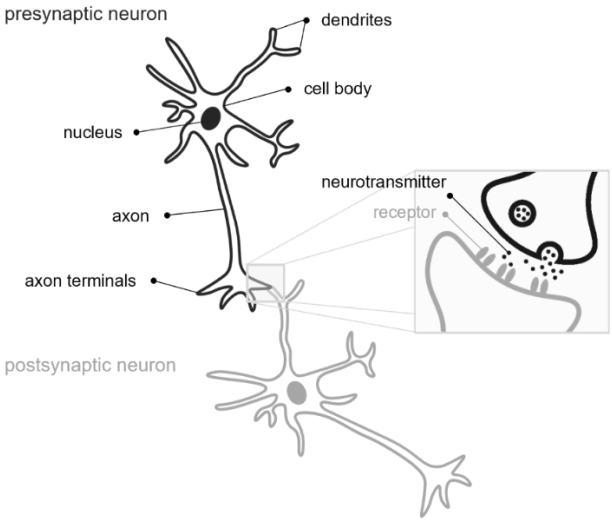

Figure 1 Illustration of the neuron anatomy and of a discharge of an action potential at the synapse level

Donald Hebb [5], a psychologist, considered how stimuli triggered physical changes between neuron's connections. He defended that when the repeated firing of a neuron A caused a neuron B to also repeatedly fire, the connection between these two neurons would be strengthened and they will be more likely to fire together again in the future. On the other hand, if two neurons fire in an uncorrelated manner, their connections will become weaker. This idea can be summarized by the motto "Neurons that fire together, wire together. Neurons that fire apart, wire apart". Here, firing is a term used to describe the case when a neuron membrane potential saturates and causes the release of an action potential, or 'spike', from its cell body to its axon terminals. This hypothesis, that latter became known as Hebbian learning, predicted a biological event that occurs between neurons. This learning mechanism is known as spike-timing-dependent-plasticity (STDP) [6]. In this process, the strength of a connection between a pre- and a post-neuron (i.e., the sending and receiving neuron), will increase if the post-neuron repeatedly fires right after the pre-neuron. The amount of variation in the post-voltage translates into the connection strength (stronger connections lead to a greater variation in the voltage). As already mentioned, for STDP the times at which pre- and post-neurons fire relative to each other is what determines if their connection will be strengthened or weakened. In [7], the authors have identified and characterized a critical time window during which synaptic modifications can occur. If the pre-synaptic neuron fires in the preceding 20ms window before the post-neuron fires, long term potentiation (LTP) occurs and the connection between these neurons in strengthened. If the pre-synaptic neuron fires within a window of 20ms after the post-synaptic neuron, long term depression (LTD) will lead to the weakening of their connection. Outside these windows, connection strength will remain unchanged.

In 1943, Warren McCulloch and Walter Pitts created what would be considered as the first artificial neuron (AN), a computational model that is now known as the M-P model. In this model, all the neuron's inputs are binary. Later, in 1958, Rosenblatt proposed the perceptron [8]. This representation overcame some limitations of the earlier proposed model. More specifically, the inputs of the perceptron take real values and are assigned a weight. This weight enables the perceptron to learn from data. Rosenblatt created an algorithm that updates the weights associated with the inputs, such that the perceptron model fits to a specific dataset. In M-P neurons this is achieved by manipulating the threshold value (see Table 1). Both models are capable of performing binary classification tasks on linearly separable data. During the 60s, significant progress was made in the development of multilayer networks [9]. The most- adopted algorithm, until nowadays, was proposed by Seppo Linnainmaa when he wrote his master thesis [10], in 1970. It is a learning algorithm named backpropagation and is used within feedforward ANNs. These are networks where information propagates solely in one direction, from each layer to the next one. With backpropagation, the weights update during training depends on the error between the network output, and the desired output. After each training step (i.e., each training sample presentation) the error is calculated and backpropagated to update the weights in such a way that the next network output shows a decreased accuracy error.

After an ANN has been properly trained it can be used to predict new input data. There are three techniques that can be used to train an ANN: supervised, unsupervised and reinforcement learning. Supervised and unsupervised learning work in opposite ways. In the former method, the training data is complemented with labels that tell the model what is at his input. This way, later, when it receives the same input, it knows what to output. In unsupervised learning algorithms, the training dataset is just a set of examples with no specific label. The model is responsible for finding hidden patterns within the dataset. Lastly, reinforcement learning is an iterative process where an agent (artificially intelligent agent) attempts to improve in a specific task. If the agent actions lead him towards a pre-specified goal, it will be rewarded. The agent will try to find the best next step until a final reward is given.

In his influential paper [11], Maass classifies ANNs according to the functionality of their underlying neurons. He considers three generations of ANNs. M-P neurons, perceptrons with continuous and differentiable activation functions and spiking neurons are, in this order, the computational units that engender the different generations of ANNs.

The first generation, McCulloch-Pitt neurons, are characterized by taking binary inputs and producing a binary output. This output will equal one if the sum of the inputs exceeds a pre-specified threshold value, and zero otherwise. Equation (1) shows the M-P neuron activation function (Heaviside function):

$$fs = \begin{cases} 1 \; if \; s \geq T \\ 0 \; if \; s < T \end{cases} \tag{1}$$

In Equation (1), s stands for the sum of the input array $x = [x_1, \dots, x_n]$ and T is the specified threshold value. Table 1 shows how the M-P neurons threshold can be manipulated to implement binary operations.

Operation	Condition	Threshold (T)
AND	$If \; x_i = 1 \; for \; \forall i \Rightarrow \sum_i x_i = n$	$\begin{cases} 1 \; if \; s \geq \mathrm{n} \\ 0 \; if \; s < \mathrm{n} \end{cases} \Rightarrow T = n$
OR	$If \; \exists x_i = 1 \; for \; \forall i \Rightarrow \geq 1$	$\begin{cases} 1 \; if \; s \geq 1 \\ 0 \; if \; s < 1 \end{cases} \Rightarrow T = 1$

Table 1 Boolean functions computed by M-P neurons, solely by changing the parameter T in Equation 1 (i is the index of the input and s is the sum of the weighted inputs $wi \times xi$, with $i = 1, \dots, n$)

The second generation of ANN, as classified by Maass, arises from networks, known as multilayer perceptron, that result from having multiple perceptron units, with a continuous and differentiable activation function, grouped by layers. Adopting this neuron model enables the use of the gradient descent backpropagation algorithm. This is the most widely used method for training a multilayer perceptron given a set of labelled patterns (supervised learning). In order to define a multilayer perceptron ANN, it is necessary to define the number of layers, the number of units per layer, the activation function of the units and the initial weights of the net. The training of the net relies on minimizing the empirical risk R,

$$R = \frac{1}{n} \sum_{k=1}^{n} L(y^k, \hat{y}^k) \tag{2}$$

In equation (2), L stands for the loss function. y_k and $\hat{y}_k$ stand for the desired and the real network outputs, respectively, for the input x_k. Commonly the quadratic error is used as the loss function,

$$L(y, \hat{y}) = \|y - \hat{y}\|^2 \quad (3)$$

as shown in equation (3). The only parameters in this type of networks that can be adjusted are the weights. This is usually accomplished using the gradient descent backpropagation algorithm. After a loss function has been defined, the network is iterated with different inputs. In each iteration, the weights between neurons are updated according to equation (4), where w_{ij} denotes the weight of a connection between neuron i and neuron j. The variation in each weight is calculated with equation (5), where η is a constant designated as the learning rate that controls the speed at which the network learns. The other component represents the rate of change of the empirical risk with respect to the weights. A feed forward network propagates an input pattern through its multiple layers, until the last one, from which produces an output (or prediction). An error is computed (according to the chosen loss function) between the network output and the desired output. This error is then used to compute the weights change that will result in a decrease of R. A backward network is used to propagate the error back (from the last layer to the first) and update the weights.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad (4)$$

$$\Delta w_{ij}(t) = -\eta \frac{\partial R}{\partial w_{ij}}\bigg|_{w(t)} \quad (5)$$

2.2 Spiking Neural Networks

According to Maass [11], the third generation of ANNs are the Spiking Neural Networks (SNNs). These are ANNs based on spiking neurons. Some of the most used spiking neuron models include the leaky-integrate-and-fire (LIF) [12] [13] (see Figure 2) and the Hodgin-Huxley [14] models. This generation provides the most accurate representation of the dynamics of a biological neuron. In SNNs neurons share synapses through which they exchange sequences of spikes.

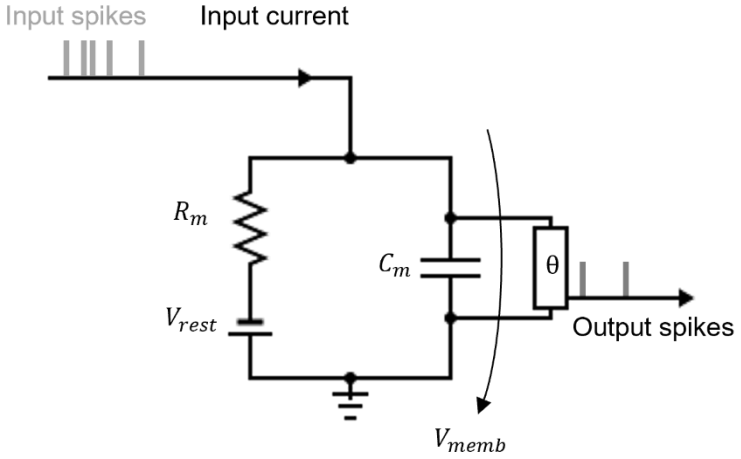

Figure 2 Physical circuit representation of a LIF neuron

Each neuron has an internal membrane potential, V_{memb}, that changes according to a pattern of incoming spikes. Every spike entering the neuron will cause this potential to increase. When it reaches a threshold membrane potential, V_{thresh}, it decreases to a reference potential, V_{reset}, and

the neuron emits a spike. Also, the neuron enters the, so called, refractory period, during which the input is disregarded. SNNs are more energy efficient than the earlier generations (and thus better for low energy platforms) as each cell of the network is only active when a spike is received or emitted, in contrast with multilayer perceptrons that always have their units active. Because this is a spike-based model, the input data must first be encoded into spikes that can be used as input stimulus. Spikes can be generated using different encoding algorithms, meaning that the same input stream of data can take different sequential representations before it is sent to the SNN. Although it is still unknown how the brain encodes information, some encoding schemes have been proposed to explain this mechanism [15]. Rate coding, time-to-first spike (TTFS) and population encoding are the most used methods to encode spatial data into spikes. The first, which is the most widely used, encodes real values using Poisson impulse trains, with firing rates that are proportional to the magnitude of the input value. The hypothesizes that the brain uses such method to encode information appears to be inaccurate and lacks supporting evidence. The human brain exceptional ability to perform a wide range of complex tasks while operating at relatively low frequencies make it super energy efficient [16]. The rate coding scheme mentioned relies on a time window, which needs to be big enough to ensure a reliable representation of a value. Also, it leads to a system that is not energy-efficient, since the number of produced spikes increases with the magnitude of the values. On the other hand, TTFS uses time to represent real values with a single spike. The spike time, within a time window, is inversely proportional to the value itself, leading to earlier impulses for higher values. Figure 3 exemplifies how these encoding methods can be applied to encode grayscale pixel values (ranging from 0 to 255). Population encoding uses a group of neurons, each tuned to a different set of values, to output a series of firing rates or firing times (see Figure 4). Depending on their tuning, each neuron becomes more sensitive to a given range of values.

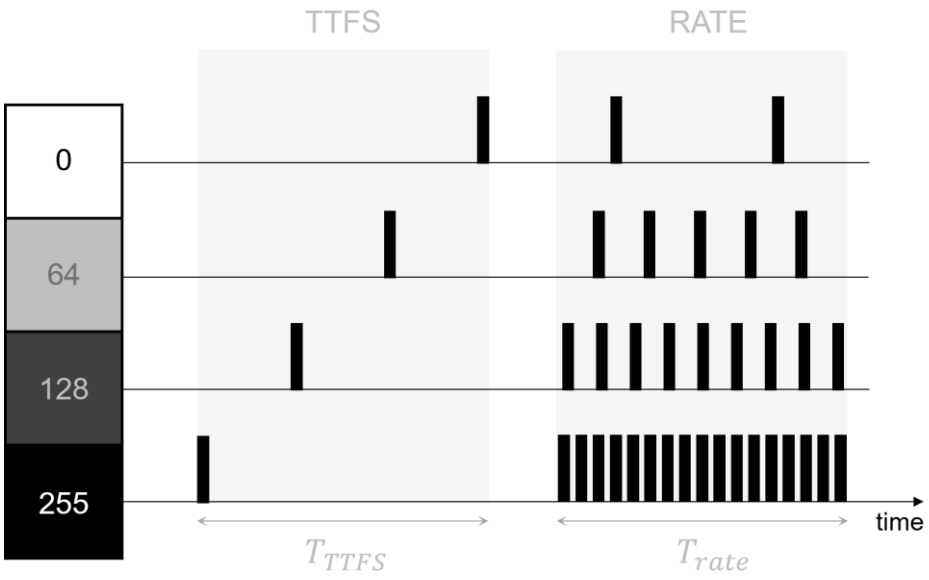

Figure 3 Pixel values encoded with TTFS and rate encoding. For both methods, a time window is chosen during which the real-valued pattern will be presented ($TTTFS$ and $Trate$)

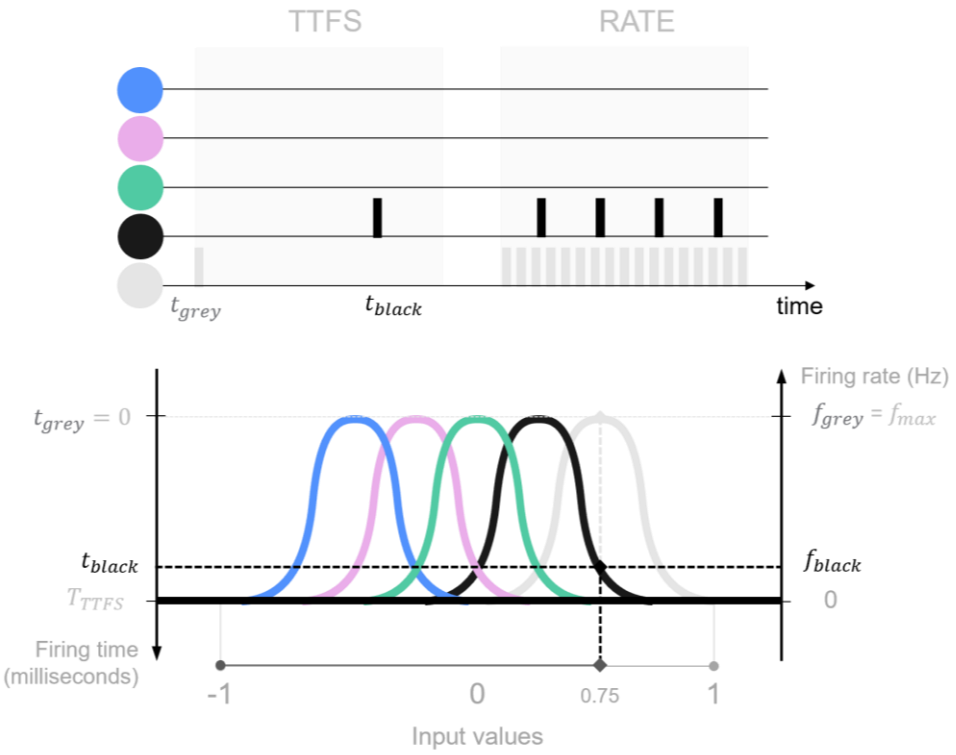

Figure 4 Value encoding through a population of 5 neurons. The grey neuron tuning shows maximum response for the illustrated input value (0.75), while the black neuron shows a weaker response. This results in earlier spiking times (TTFS) and higher spiking frequencies (rate) for the grey neuron.

Given the biological resemblance shown by this generation of neuron models, the algorithms used for training SNNs also seek for biological plausibility. STDP, the biological mechanism behind synaptic plasticity has given rise to a learning algorithm [17]. It delivers an unsupervised approach for updating the weights of a SNN, based on the difference between the firing times of the pre- and post-synaptic neurons. Each time the membrane potential reaches its threshold value and is discharged, the weights of the network are updated according to equations (6) and (7).

$$\Delta w = \begin{cases} a^+ \cdot e^{\frac{\Delta t}{\tau^+}} \quad if\ \Delta t \geq 0 \quad (LTP) & (6) \\ -a^- \cdot e^{-\frac{\Delta t}{\tau^-}} \quad if\ \Delta t < 0 \quad (LTD) & (7) \end{cases}$$

In these equations, a^+ and a^- represent learning rates, and τ^+ and τ^- are time constants. Δw corresponds to the variation of the synaptic weight. This variation depends on the difference between post- and pre-firing times, Δt. As in the natural process, when the post-neuron fires after the pre-neuron, LTP occurs and the connection between these neurons is strengthened. Otherwise, LTD occurs and their connection is weakened. Figure 5 shows how this update depends on the difference between the firing of a post- as pre-neuron.

Besides excitatory synapses, neurons can also share inhibitory connections. Inhibitory connections allow neurons to suppress each other's activity, thus providing a way for incorporating competition in a SNN. In [18], the author implemented an 1WTA using a SNN. In this experiment, the network presented 3 layers: input, excitatory and inhibitory layers. The input layer shares all-to-all connections with the excitatory layer (i.e., each input neuron connects to all excitatory neurons). The excitatory and inhibitory layers have the same number of neurons, and

each excitatory neuron a connection with its corresponding inhibitory neuron (consider neurons are numbered in each layer). Also, each inhibitory neuron is connected to all the excitatory neurons except to the excitatory neuron it shares a connection with. The input layer encodes pixel intensities to spike trains via rate encoding. Once an excitatory neuron fired, it triggered its corresponding inhibitory neuron to send an inhibition signal to all the remaining excitatory neurons. Inhibition led excitatory neurons to compete, forcing them to learn different patterns. [19] shows another experiment where lateral inhibition led to an 1WTA system (see also 2.3). By adding lateral inhibition synapses between excitatory neurons, it was possible to have them identify different parts of the same pattern.

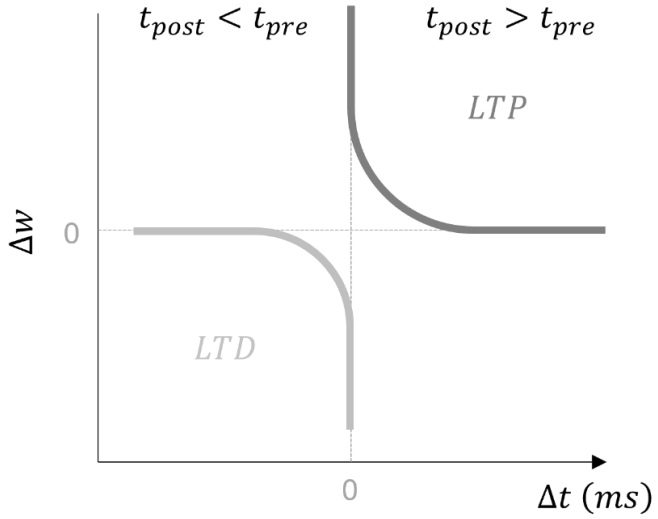

Figure 5 Synaptic weight update according to the STDP learning rule. $tpre$ and $tpost$ refer to the pre- and post-neuron firing time, respectively.

2.3 Simulating Spiking Neural Networks

A neural simulator allows to simulate and test models of biological neural networks. This is not only a key element for consolidating new knowledge obtained from in-vivo observations, but also to realize more powerful algorithms that leverage from these findings. Brian 2 [20], NEURON [21] and NEST [22] are some of the most used spiking neural networks simulators. When deciding on which simulator to use, it is important to consider the requirements of the intended application, as each simulator offers different capabilities and features. NEURON focuses on the simulation of biologically accurate neural models. NEST leverages from parallel computations to efficiently simulate large-scale networks, with up to millions of neurons. Brian2 offers a higher flexibility for model definition when compared to others SNNs. This was the simulation tool used in this work. While other simulators focus on biologically plausible neural representations, Brian2 stands out as a more versatile tool, since it allows to define equation-based neural and synaptic models. This feature expands the spectrum for SNN simulations, as it enables the research of both biologically and non-biologically plausible models. It offers a user-friendly interface written in Python, a high-level programming language suitable for all levels of programming expertise. Also, it takes advantage of code generation to convert the Python code and perform simulations in other languages (e.g., a more efficient language such as C++). The core elements in Brian2 that serve

as a foundation to design SNNs are neurons and synapses. They can be instantiated using *NeuronGroups* and *Synapses* objects, respectively. The first, allows to create a population of neurons by specifying the variables that define the neurons model. These variables are defined through equations. Threshold conditions and reset operations can also be defined. *Synapses* objects implement interactions between pairs of neurons. As for neurons, synapses can be attributed with an equation-based model. They can also define the operations to compute at specific neural events (e.g., pre-/post-neuron fire). Other objects provided by Brian2 include spiking generation methods (time-precise or Poisson-based) and ways to monitor and record simulation variables and neurons populations behaviors. Next, consider a few examples of network implemented with this simulator.

For the first experiment, a SNN with two layers was simulated (see Figure 6). The first layer consists of a population of neurons which fire according to a Poisson process, as depicted in Figure 6a. With Brian2 this can be easily achieved by using a *PoissonGroup* object. All neurons from this population share excitatory connections with two other neurons. These connections are plastic, meaning their value is updated over time. The weights for these connections are initialized randomly. The second layer neuros follow the LIF model described by equation (8). The neurons membrane potential, v, rests at a potential V_{rest} while no spikes are received. Otherwise, pre-spikes increase the excitatory synaptic current, I_e, and, subsequently, v. If v crosses the threshold potential, the neuron emits a spike, resets its potential and enters a refractory period where it disregards any input. I_i is the inhibitory synaptic current and its increment leads to the decrease of v. These currents increase with the arrival of spikes, and decrease exponentially otherwise, as

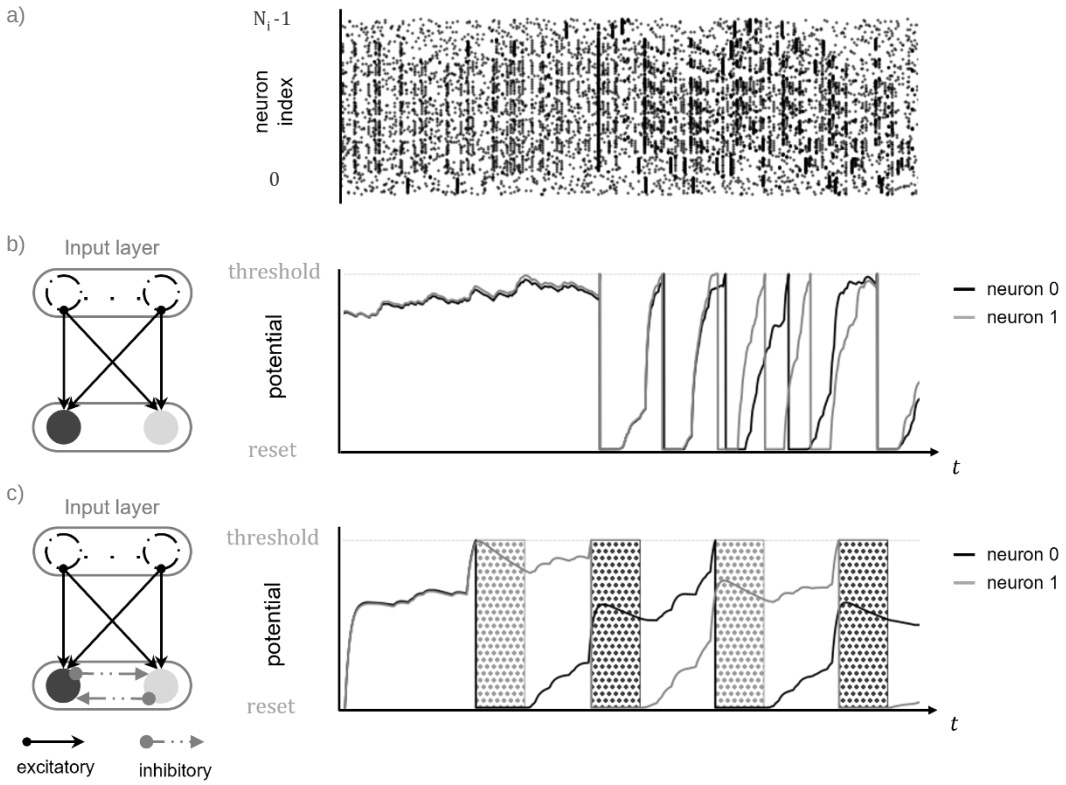

Figure 6 Effects of lateral inhibition. An input layer (a) feeds two LIF neurons. In b) the LIF neurons fire freely, while in (c), only one neuron can fire at a time since now they share inhibitory connections. The dotted areas in c) refer to inhibitory periods

expressed by equation (9). R_e and R_i are, respectively, the excitatory and inhibitory membrane resistances. These values influence the synaptic currents amplitude, and, consequently, the neurons response speed to incoming excitatory and inhibitory spikes. In a first experiment, the LIF neurons don't share inhibitory synapse. They receive the input spike trains and fire freely (see Figure 6b). When lateral inhibitory synapses are established between the LIF neurons, only one neuron fires at a time (see Figure 6c). To implement such mechanism using Brian2, a binary attribute representing the inhibition state of the neuron was added to the model. Whenever a neuron fires, it sends an inhibition signal to the other LIF neuron. This signal sets the post-neurons inhibition attributes to true and increases their inhibitory synaptic current. After the inhibition period (identified with dots) has ended, other signal is sent to reset the post-inhibition states to false.

$$\tau_m \frac{dv}{dt} = (V_{rest} - v) + R_e * I_e - R_i * I_i \quad (8)$$

$$\tau_I \frac{dI}{dt} = -I \quad (9)$$

As explained in chapter 2.2, STDP can be used as an unsupervised learning rule to train SNNs. Although the biological mechanism uses the firing time difference between pre- and post-neurons to perform the weights update, there are more computationally efficient approaches to implement a learning rule which provides a similar outcome. Instead of saving every firing time instants, each synapse is attributed with two additional variables. These are the pre- and post-synaptic traces, a_{pre} and a_{post} (see equations (10) and (11)), respectively. These variables keep track of pre- and post-synaptic synaptic activity, respectively. Whenever a pre- or post-neuron fires, the correspondent trace variable is incremented as in equations (12) and (13), respectively, after which it decays exponentially. Also, when a pre- or post-neuron fire, their weights are updated according to equations (14) or (15), respectively. The condition described in equation (16) ensures that the decrease in weight for LTD is greater than the increase in weight for LTP. Figure 7 shows the results from a simulation which implemented this trace-based learning rule. Four pixel values are converted into spikes via rate encoding (see Figure 7a). Consider that the values are normalized and evenly distributed. The input rates were calculated by multiplying these values by maximum frequency (80 Hz) and adding an offset to zero-valued components (20 Hz). This offset ensures that all input neurons fire. This is essential since no weight updates will be performed for neurons that don't fire. The encoding neurons are connected to a LIF neuron through weighted synapses, equipped with the synaptic trace-based learning rule. The fourth plot from Figure 7b shows the synaptic traces of all the network neurons (identified by colors). It is possible to see that the input neurons presynaptic trace is proportional to their firing frequency. Also, it is possible to observe that the pre-neurons that show higher traces then the post-neuron (i.e., higher frequencies), have higher weight increases (see last two plots of Figure 7b). Otherwise, the weights tend to be depressed (see pink neuron behavior). Overtime, each synaptic weight converges toward values that are proportional to their respective synaptic stimulation frequency.

$$\tau_{pre} \frac{dapre}{dt} = -apre \quad (10)$$

$$\tau_{post}\frac{dapost}{dt} = -apost \tag{11}$$

$$\Delta apre = Apre \tag{12}$$

$$\Delta apost = Apost \tag{13}$$

$$\Delta w = apre \qquad \begin{matrix} if\ w + apre \leq w_{max} \\ else\ w = w_{max} \end{matrix} \tag{14}$$

$$\Delta w = apost \qquad \begin{matrix} if\ w + apost \geq 0 \\ else\ w = 0 \end{matrix} \tag{15}$$

$$\tau_{post} * Apost > -\tau_{pre}Apre \tag{16}$$

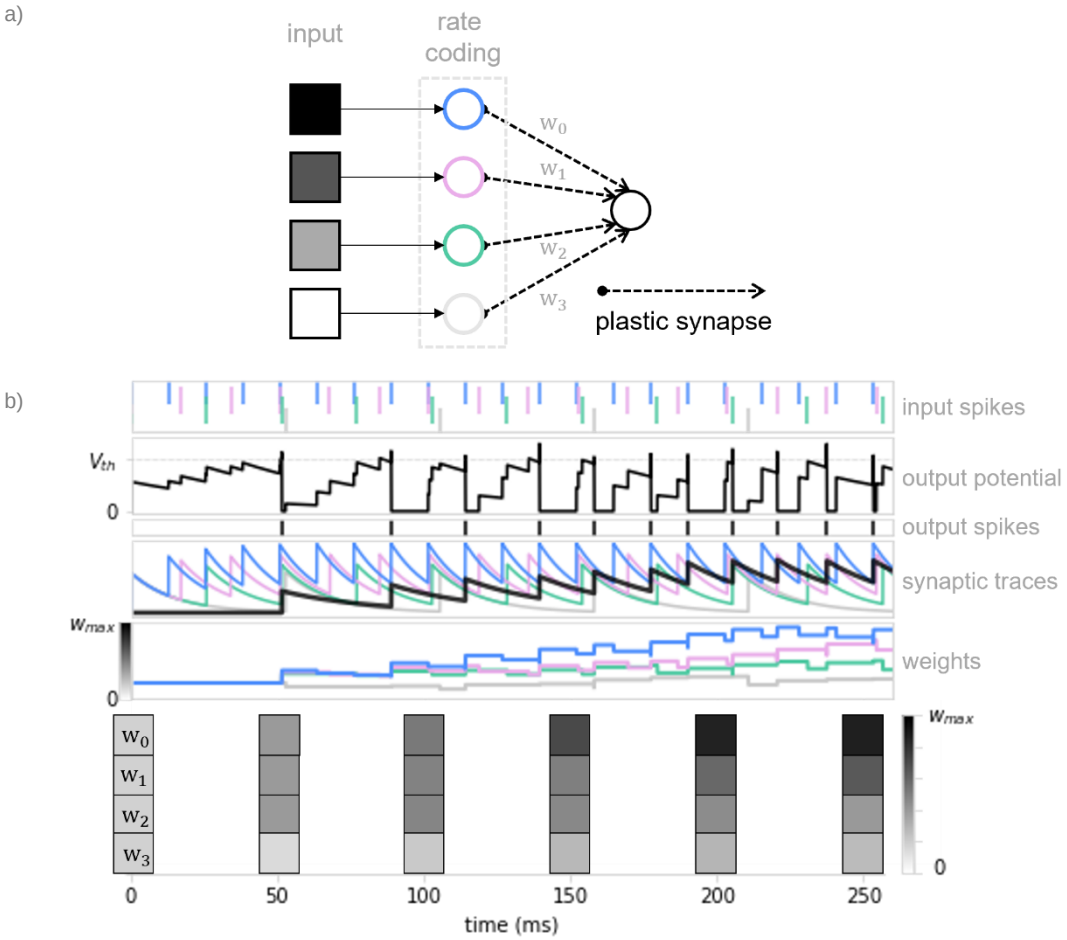

Figure 7 Learning pixel values using rate encoding and a trace based STDP rule. a) shows the architecture of the tested network. It consists of 4 inputs neurons that perform rate encoding on different pixel values and feed the resulting spike train to a single LIF neuron. In b), the grey squares at the bottom represent the evolution of the weights over time.

3

Setup for data acquisition

3.1 SensorTag Development Kit 28
3.4 Bluetooth Low Energy 29
3.5 Firmware customization 31
3.6 Data acquisition and processing 31

This chapter focuses on the setup used for the creation of a datset comprising human workout activities samples, characterized with tridimensional motion sensors data. The microcontroller used to perform the acquisiton of such data is introduced as well as the tools utilized to develop applications with it. An overview about Bluetooth Low Energy is also provided, since it was the used protocol to transmit data. Then, a detailed explanation about the developed pipeline for data acquisition and processing is provided.

3.1 SensorTag Development Kit

The SensorTag is a development kit, by Texas Instruments (TI), specifically designed for prototyping IoT applications. It provides in a single, portable, low-energy system, several sensors and supports a wide range of wireless communications technologies. Based on TI's CC2650 wireless microcontroller-unit (MCU), the SensorTag contains an ARM Cortex-M3 processor, 128KB of programmable flash, a 2.4 GHz RF transceiver that is compatible with Bluetooth Low Energy (BLE) 4.2, Zigbee and 6LoWPAN technologies, and cJTAG and JTAG connections for debugging. This platform has multiple integrated processors that support different system tasks. Besides its main processor, the sensor controller is an ultra-low power MCU that offers enough processing capabilities for handling sensor related system tasks. Lastly, the radio processor, comprised by a Cortex-M0, is an engine responsible for managing low-level radio system tasks. The distribution of tasks contributes to a more energy efficient system by freeing the main CPU from recurrent communication and sensor related tasks that arise from the nature of the platform. The communication between these different modules is done either using the Inter-Integrated Circuit (I^2C) or Serial Peripheral Interface (SPI) communication protocols.

The SensorTag contains ten low-power sensors that include a digital microphone, a light sensor (OPT3001), an inertial measurement unit (IMU) (MPU-9250), and humidity (HDC1000), pressure (BMP280) and temperature (TMP007) sensors as well. For this work only the gyroscope and accelerometer sensors are of interest (see Table 2 for specifications). These are built in the MPU-9250. This is a chip module that contains a gyroscope, an accelerometer, and a magnetometer, all three-dimensional. For each of the sensors the module has a three 16-bit analog-to-digital converter for converting their output to a digital signal. The IMU features I^2C and SPI serial communication interfaces. The communication with the internal registers can be performed either at 400kHz using I^2C or at up to 20MHz using SPI.

Sensor		*Gyroscope*	*Accelerometer*
Ranges		±250, ±500, ±1000, ±2000	±2, ±4, ±8, ±16
Units		degrees/sec	g
Operating current		3.2 mA	450 µA
Sleep mode current		8 µA	8 µA
Output data rate (Hz)	min	4	8000
	max	4	4000

Table 2 MPU-9250 accelerometer and gyroscope specifications

Texas Instruments Real Time Operating System (TI-RTOS) is the operating system used in the CC2650. It is implemented in C and, as the name suggests, it is a real-time operating system developed and supported by TI. RTOSs are characterized for having a deterministic behavior, meaning that computational tasks are precisely scheduled to meet specific timing requirements.

To ease the development, debugging and testing of code for TI's MCUs, TI makes available an integrated development environment (IDE), designated as Code Composer Studio (CCS). The IDE provides a graphical interface that seamlessly integrates code editing, debugging and simulation. In addition to the IDE, CCS also includes a set of software packages and coding examples specifically targeted for each TI board that help new developers to become familiar and take full advantage of the capabilities offered by each board.

The SimpleLink™ CC13x2 and CC26x2 software development kit (SDK) offers a series of software packages that support the development of Sub-1 GHz and 2.4 GHz applications adding support to Bluetooth Low Energy (BLE), threads and multiprotocol solutions on the SensorTag. Another important set of software packages is the BLE software stack archive which provides all the required software to get started on the development of single-mode and network processor BLE. All these resources are available online at TI's website.

3.2 Bluetooth Low Energy

The SensorTag provides BLE communications. As the classic Bluetooth technology, it operates in the 2.400-2.4835 GHz ISM band. This frequency range is divided in forty 2 MHz channels, and each of these channels broadcasts data using a Gaussian frequency shift modulation. BLE communication technology enables short distance communications, with extremely low power consumption. Table 3 provides some specifications about this communication protocol.

Theorical maximum range	Less than 100 meters
Minimum time interval to send data	3 ms
Power consumption	0.01 to 0.50 W
Peak current consumption	Less than 15 mA
Over the air data rates	125 kbit/s, 500 kbit/s – 1 Mbit/s – 2 Mbit/s
Application throughout	0.27 to 1.37 Mbit/s

Table 3 BLE specifications

The Generic Access Profile (GAP) is a protocol used for establishing a connection between two Bluetooth devices. The two most important roles defined by GAP, for each participating device, are the central and peripheral roles, where usually the first corresponds to devices with bigger processing and memory capabilities and the second to low power devices (e.g., the SensorTag). These roles impose the communication rules. GAP defines two types of payloads, which are used by the devices to advertise themselves. A device can advertise itself either through the Advertising Data payload or the Scan Response payload. Both are identical, with 31 bytes of data. The second is an optional payload that can be used, when requested by central devices, to send some extra information. After two BLE devices establish a connection using GAP, the Generic Attribute Protocol (GATT) defines how data should be transferred between them. Data transfer can be done bidirectionally, and connections are exclusive, meaning that a peripheral can only be connected to a central device. In a GATT transaction the peripheral and the central devices play

the roles of GATT Server and GATT Client, respectively. The client is responsible for requesting data to the server and for initiating all the transactions. After being reached by the client, the server will suggest a connection time interval to the client that will try to reach the server every connection interval. An example for the client and server roles is a smartphone acting as a client requesting temperature sensor data to an IoT device. The structure of the messages shared between the two devices is based on structures named Profiles which, in turn, contain Services, Characteristics and Descriptors (depicted on Figure 8). Profiles define how devices work in a specific application. Bluetooth Special Interest Group (SIG) is the organization responsible for the development of Bluetooth standards. This organization pre-defines profiles that can be found in the stock firmware that is loaded into the SensorTag. Custom ones can be designed by application developers to meet specific data arrangement types. Profiles are composed of a set of services, which in turn are composed by one or more characteristics. Each service is identified by an Universal Unique Identifier (UUID). Official BLE services have 16 bits UUIDs while customized services have 128 bits. Characteristics hold the data value to be sent from the server to the client. Their identification is done similarly to services, and they can either be standard SIG defined or customized. Additionally, and optionally, a descriptor can be used to provide extra information

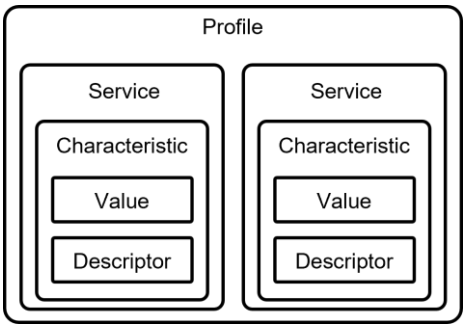

Figure 8 Illustration of the high-level objects used in GATT transactions.

regarding a characteristic (e.g., the units of the transferred data). Table 4 shows some examples of characteristics for the SensorTag movement sensor (MPU-9250).

Type	UUID	Permissions	Size (bytes)	Description
Data	AA81	R	18	Gyroscope, accelerometer, and magnetometer values
Notification	2902	R/W	2	Write 0x0001 to enable notifications, 0x0000 to disable.
Configuration	AA82	R/W	2	Enable/disable axis for each sensor. Enable/disable wake-on-motion. Define accelerometer range
Period	AA83	R/W	1	Resolution 10 ms. Range 100 ms (0x0A) to 2.55 sec (0xFF). Default 1 second (0x64).

Table 4 SensorTag movement sensor BLE characteristics. The 16-bits represented in the table should replace 16-bits in the TIs Base 128-bit UUID, F0000000-0451-4000-B000-000000000000 (e.g., 0xAA01 maps as F000AA01-0451-4000-B000-000)

3.3 Firmware customization

At the start of this work, the required sampling frequency was unknown. This is because it relied on the chosen spike encoding method and network topology. A pre-existing SensorTag firmware was adapted to enable an increase of this frequency and overcome any potential limitations resulting from a low sampling frequency. Essentially, the implemented code optimizes BLE data transmission, by maximizing both the BLE transmission frequency as well as the size of the sent packet. This configuration allows to broadcast packets of 251 bytes at 100 Hz.

SensorTag applications that make use of BLE include a project that implements the BLE stack. This is a driver responsible for handling the flow of data between the SensorTag and other BLE-enabled peripherals. A suitable way for developing BLE applications for the SensorTag is to work with any application provided by TI that already implements the BLE stack and use it as a starting ground for developing the desired solution. The SensorTag firmware is an example of such applications. In this work, an application named ProjectZero [23] was used as basis for the rest of the implementation. This application provides a simpler starting point when compared to the stock firmware, since it only implements few examples of BLE services. TI provides an online tool [24] that generates the necessary code to implement a custom service in the ProjectZero application. The user only needs to specify the service properties (name, UUID and UUID size), which characteristics belong to the service and their properties (name, UUID, UUID length, GATT properties and permissions). This tool was used to generate a custom service with a single characteristic.

Some BLE related parameters were changed to decrease the BLE connection interval and to increase the size of the packet to send. These parameters are listed below followed by their assigned values as well as their definition:

- *DEFAULT_ADVERTISING_INTERVAL*: 40, advertising interval when device is discoverable (units of 625 μseconds, 160 = 100 milliseconds)
- *DEFAULT_DISCOVERABLE_MODE*: GAP_ADTYPE_FLAGS_GENERAL, defines whether the SensorTag advertise mode (general discoverable mode advertises indefinitely, limited discoverable mode advertises for about 30s and then stops)
- *DEFAULT_DESIRED_MIN_CONN_INTERVAL*: 8, minimum connection interval (units of 1.25 milliseconds, 8 = 10 milliseconds)
- *DEFAULT_DESIRED_MAX_CONN_INTERVAL*: 8, minimum connection interval (units of 1.25 milliseconds, 8 = 10 milliseconds)
- *MAX_PDU_SIZE*: 255, maximum size in bytes of the BLE host controller interface (HCI) protocol data unit (PDU). Ranges from 27 to 255. The maximum length of an Attribute Protocol (ATT) packet (Maximum Transmission Unit) is equal to *MAX_PDU_SIZE* - 4

3.4 Data acquisition and processing

In this work, a dataset was compiled using samples acquired with movement sensors. More specifically, the SensorTag accelerometer and gyroscope. The development explained in 3.5, was used for this purpose. Each sensor axis corresponds to 2 bytes. If considering an extra

byte for timestamps, each sampling iteration yields an array with 13 bytes. The packet size allows to send 20 iteration readings, which would theoretically enable to acquire samples at 2 kHz. Experimentally, the highest frequency achieved was 1 kHz, while higher frequencies would cause the BLE connection to become unstable or lost. The purpose of the data acquisition was to create a training and testing datasets.

The gathered dataset is composed by four workout movements (see Figure 9a), which were performed by the author of this work. During the collection of data, each movement was performed

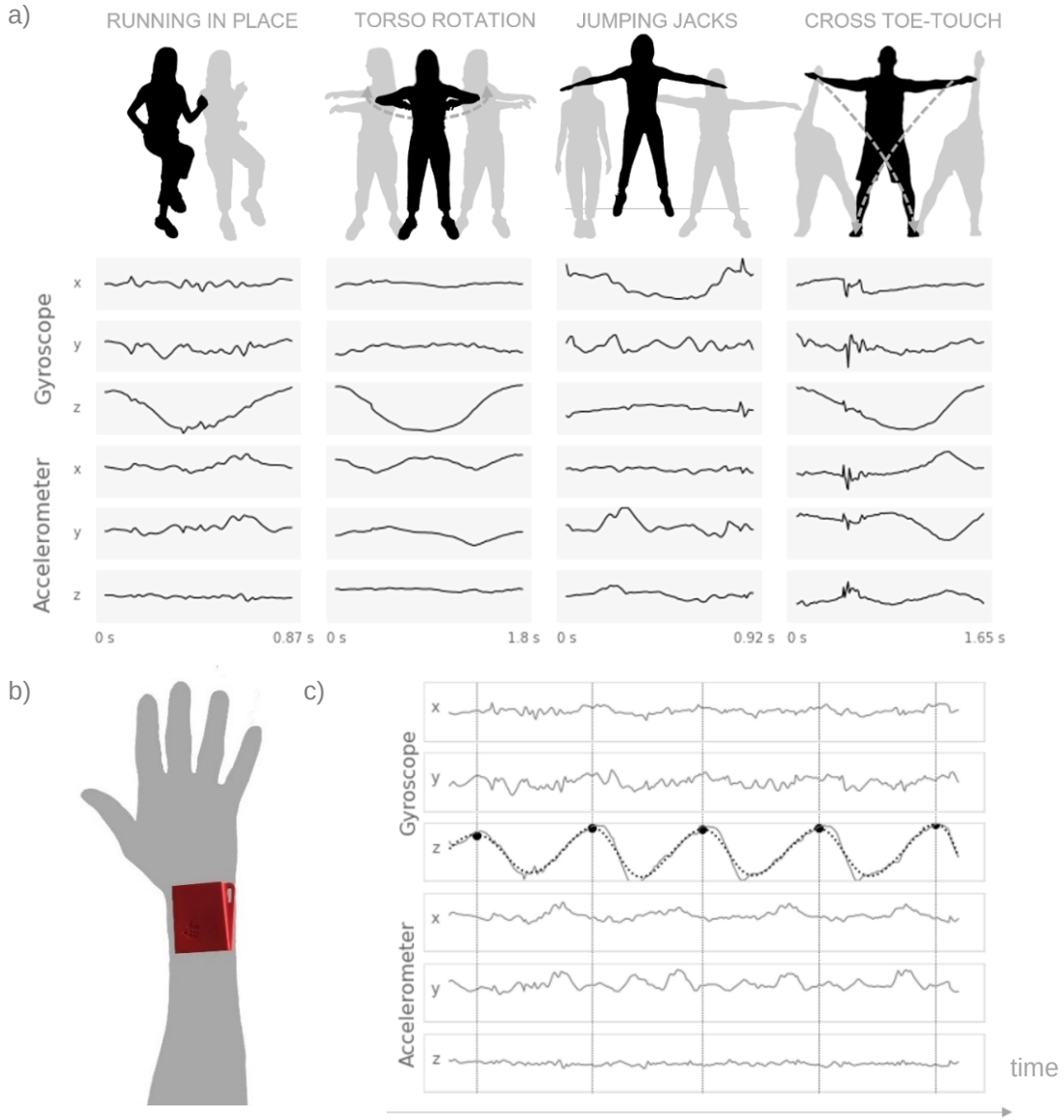

Figure 9 Dataset overview (a) Examples of segmented samples for each movement. The sensor axis range is normalized (b) Sensortag placement during sampling acquisition (c) Sample segmentation. The dotted line refers to filtered version of the gyroscope z-axis.

repeatedly, for a given time duration, with no pauses between consecutive repetitions. The data streams had to be segmented into single movement repetitions. This was required to train the SNN with samples consisting of single repetitions of each movement. A straightforward method was employed to achieve this segmentation. Initially, and for each movement, the sensors axis

exhibiting the most sinusoidal shape was chosen. Subsequently, the values of the selected axis were filtered to further enhance the sinusoidal waveform. As depicted in Figure 9c, this facilitated the identification of the filtered data extremes and enabled the division of the data stream into single movement repetitions.

4

Identification of three-dimensional manoeuvres using a spiking neural network

4.1 Objectives 36

4.2 Requirements 36

4.3 System design 36

4.4 Spiking Neural Network 37

4.4.1 Network Architecture 37

4.4.2 Spike Encoding 38

4.4.3 Implementation 40

This chapter outlines the objectives and requirements of the proposed system following the design of the overall system. The chapter then explores the details of the implemented SNN, providing insights about its architecture, the used spike encoding scheme and further implementation aspects. This explanation is followed by a subsequent analysis of the systems performance for spatiotemporal pattern recognition.

4.1 Objectives

The final purpose of this work is to propose a possible approach for the development of a system for accurately identifying manoeuvres within a three-dimensional space. The system should be capable of recognizing manoeuvres from a pre-defined set. There are several scenarios (i.e., use cases) where motion identification can be useful. For example, several smart watches and fitness trackers, such as Apple Watch [25] and Fitbit [26], are already able to recognize the workout exercise being performed by a user, from a fixed set of possible workout exercises, just by analysing his motion, using their built-in accelerometer and gyroscope. Other applications, were the identification of manoeuvres in a three-dimensional space is beneficial, include drone and robot control, elderly, and patient surveillance, among many others.

Having discussed in the previous chapter how sensor data is acquired and transformed into a suitable format, we will now study how such data can be further processed and fed into a neural network for training and subsequent classification of manoeuvres in a three-dimensional space.

4.2 Requirements

To develop a system that enables the described objectives to be achieved, some key aspects must be considered. The device used for data acquisition must ensure an easy and flexible usability. It should be small and capable of transmitting data wirelessly to allow for unconstrained motion during manoeuvres execution, while attached to a body part. Also, it should be able to achieve suitable sampling frequency, that may enable the necessary data characteristics to be properly captured. The recognition system should provide classification accuracies of at least 95% to ensure reliable and accurate identification of manoeuvres.

4.3 System design

The block diagram of the architecture of the proposed system for solving the above-described problem is depicted in Figure 10. The data acquisition and data processing modules correspond, respectively, to the processes of data collection and segmentation described in chapter 3.6. Together, these modules create multiple samples from data streams acquired with the sensors. Each sample corresponds to a single repetition (instance) of a given movement. The SensorTag is used to perform the data acquisition. It meets the described requirements as it provides a wireless solution for data acquisition with high sampling frequency. The segmented samples are then converted into sequences of spikes, after an encoding method is applied, in the encoding module. Such method will be further described in chapter 4.4.2. Finally, the identification module predicts the different manoeuvres belonging to the dataset when their corresponding sequence of spikes has been provided at its input. To achieve this, a SNN undergoes a training process, to

learn a subset of encoded samples from the dataset. Afterwards, another subset of samples is used to assess the performance of the network at identifying manoeuvres.

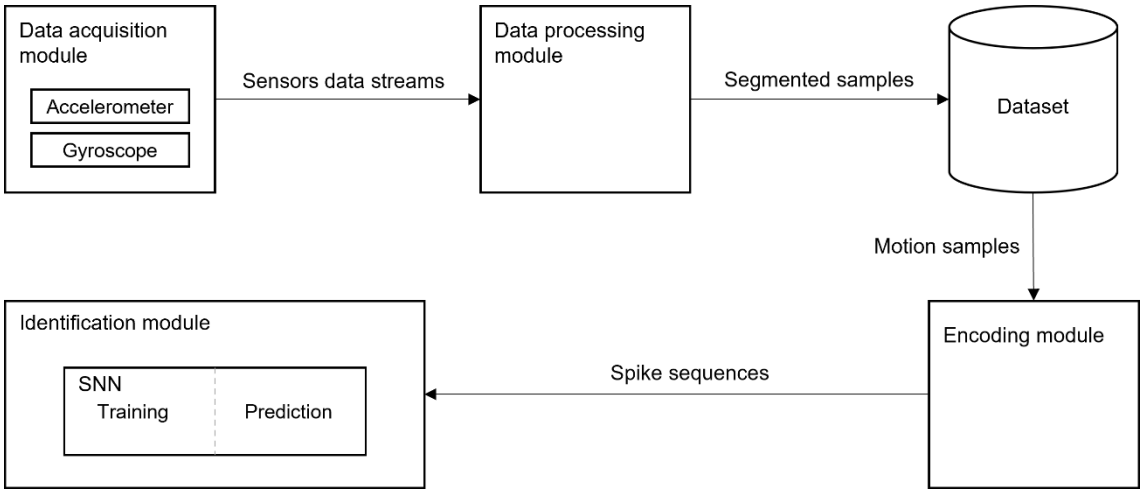

Figure 10 Block diagram of the proposed system for manoeuvres identification

4.4 Spiking Neural Network

In this chapter, a thorough analysis of the experiments carried out is provided. First, the neural network architecture and its mechanisms are outlined. Then the training and recognition procedures are described, to ensure a comprehensive understanding of the chosen methodology.

4.4.1 Network Architecture

The adopted network is composed of three layers (see Figure 11). The first layer encodes the movement samples into spike-sequences. The encoding technique will be further detailed in chapter 4.4.2. Each input neuron is connected to all the second layer neurons. These synaptic connections exhibit synaptic plasticity, with weights that are learned throughout the experiment, using the STDP-like learning rule explained in Chapter 2.3. The second layer is composed LIF neurons, which share lateral inhibitory connections. The connections are established between each neuron and remaining ones. The lateral inhibition scheme enforces a WTA mechanism that drives the neurons to compete and distribute themselves to cover the different patterns presented in the input. The implemented neuron model and lateral inhibition mechanism have been already explained in Chapter 2.3. Besides the neural mechanisms discussed so far, an additional homeostatic process was introduced in the LIF model. Each neuron is also attributed with an adaptive threshold, v_{th}. This variable increases each time a neuron fires, after which it decays exponentially to a reference value, $\mathrm{V}th$, as defined by equation (17). Such mechanism effectively

$$\tau_{v_{th}} \frac{dv_{th}}{dt} = V_{th} - v_{th} \tag{17}$$

limits the neurons firing rate and prevents singular neurons to continuously fire, thus inhibiting other neurons, therefore dominating the WTA layer. The third, and last layer, is the recognition layer. It has as many neurons as the number of classes in the dataset. The activity of each of the

recognition neurons is used to assess the network performance. The recognition layer is only active during the testing phase (i.e., after the learning has been carried out). Each of the output neurons share static and delayed synapses with the competitive neurons (i.e., neurons that share lateral inhibitory connections). The neurons in the third layer are of the leaky-integrate type since they don't have a firing condition. Their behaviour is analysed by extracting the maximum value of their potential over a pattern presentation time window.

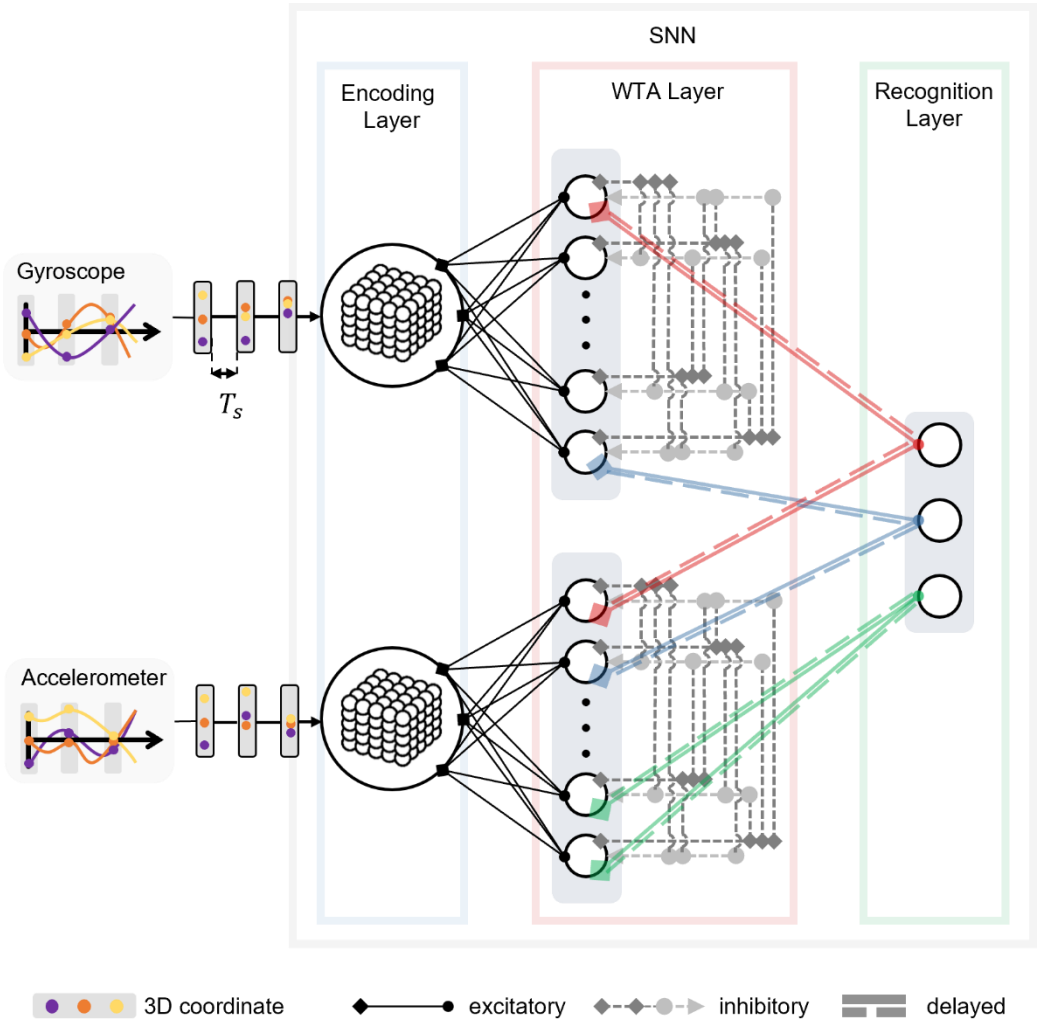

Figure 11 Implemented network architecture

4.4.2 Spike Encoding

SNNs that operate spatiotemporal patterns often apply event-based encoding schemes to the input signals. Such schemes rely on changes in the input signals to produce spikes. Networks that are designed to identify and process these spatiotemporal patterns require more complex neural architectures with additional parameters, thereby increasing the complexity and difficulty of the model. This happens because modelling temporal dependencies requires capturing the intricate relations and patterns that evolve over time, which can pose challenges in terms of network design, training, and optimization. Incorporating these temporal dependencies could potentially yield more accurate and meaningful results in analysing and predicting time series

data, as it would enable the network to capture and exploit temporal correlations. However, the proposed encoding scheme represents the input signal as a sequence of spatial patterns, treating each as an independent snapshot of data and overlooking any temporal correlations between them. Each dimension of the input data is discretized into a reduced set of possible values that are linearly spaced. The total number of input neurons is equivalent to the number of possible combinations resulting from discretizing all the values in all dimensions of the input data. The input patterns are represented using the spatial coordinates of the input neurons, providing a mechanism for encoding and processing spatiotemporal data in a distributed manner. The resolution at which the input patterns are represented increases with the chosen number for possible discrete values per dimension. Given the three-dimensional nature of the dataset samples (for each sensor), their full range is constrained to a cubic space. Taking this aspect into consideration, a population of input neurons is uniformly distributed in a cubical arrangement. These neurons are modelled to exhibit a Poisson process firing behaviour, with a fixed firing rate, dependent on their spatial proximity to the input data. Figure 12 illustrates the input response to a set of three-dimensional coordinates using this encoding scheme. Input neurons dynamics are defined by equation (18). Here, $Poisson$ stands for a rated Poisson process. Each neuron is attributed with a three-dimensional coordinate (represented as c in equation (19)). The readings from a single sensor (represented as pt) are also three-dimensional coordinates. Every simulation timestep, each neuron is said to be $inzone$ if its euclidean distance to the current sensor reading is smaller than a threshold distance, dth (as expressed by equation (19)). Note that $inzone$ is a Boolean variable. If so, the conditioned Poisson activity takes effect. The value of pt is updated according to the sensor sampling period, Ts, and is constant within the interval $[tk, tk + Ts]$ (for a reading at $t = tk$). Note that $fzone >> fmin$ and $fmin$ is used to activate neurons that otherwise wouldn't fire (since the input data is never spatially close to them). This is because if neurons don't fire at all, their weights will never be updated.

$$input\_dyn(\ in_{zone}, f_{zone}, f_{min}) \ = \ in_{zone} * Poisson(f_{zone}) + Poisson(f_{min}) \qquad (18)$$

$$inzone = (||c - pt|| \leq \ dth) \qquad (19)$$

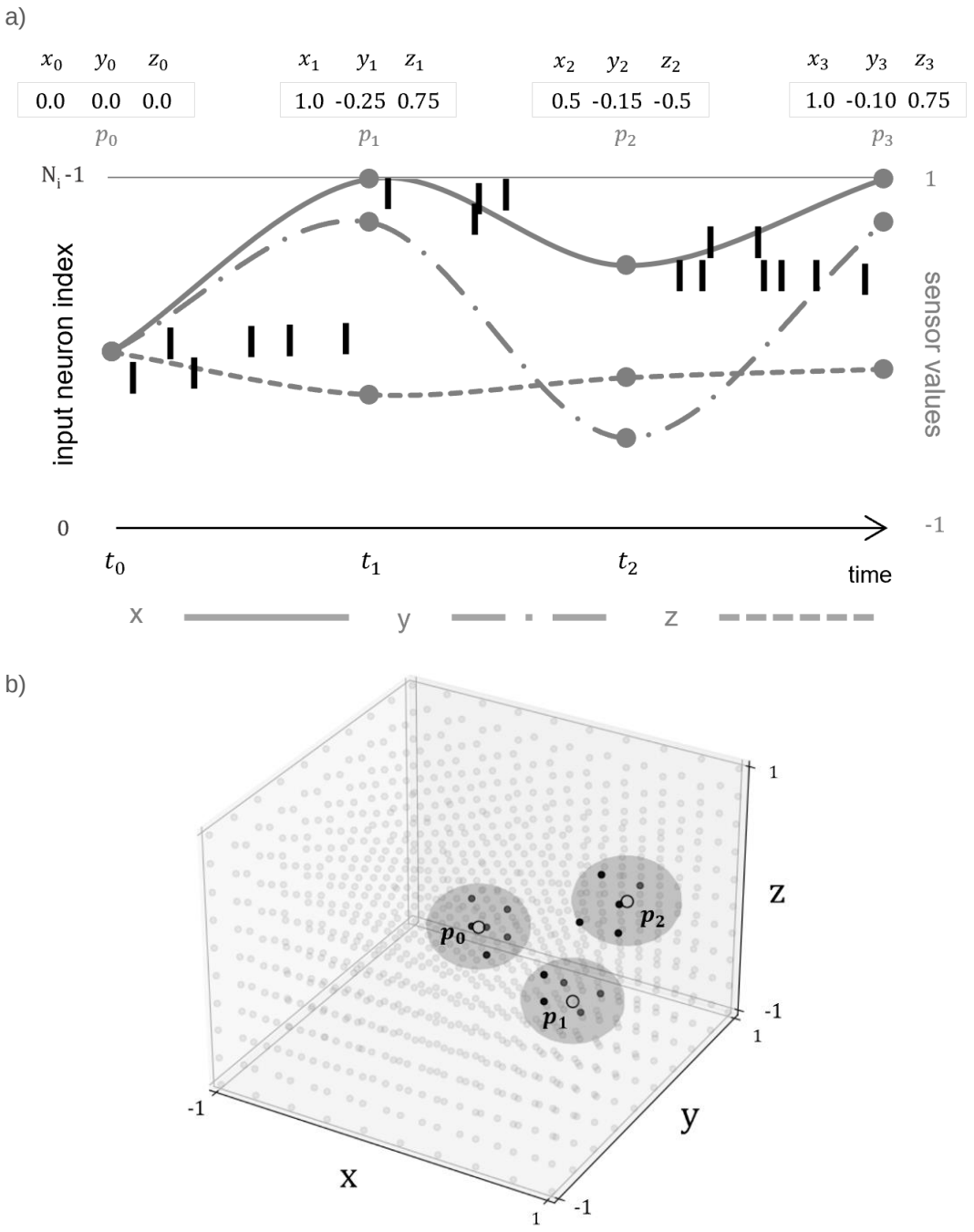

Figure 12 Example of the proposed encoding spatial-dependent spike encoding scheme (a) Three-dimensional values over time and resulting spike pattern (b) The first three 3D coordinates are represented as white points. The shades of grey indicate the areas where input neurons are permitted to fire. Black dots represent firing input neurons.

4.4.3 Implementation

The investigation was conducted using Python and the SNN was simulated with Brian2. The complete workflow was performed using Jupyter Notebooks. A set of Jupyter Notebooks is available at [27] for reference and reproducibility purposes. The custom firmware detailed in chapter 3.5 is also available in this repository. The experiment comprises of three stages: training, supervised parameter estimation (via analysis of firing behaviour) and recognition.

The dataset (see Chapter 3.6) was partitioned into training and testing sets with a ratio of 70/30, respectively. Initially, the network was trained to learn the patterns in the training dataset. The weights between the input and WTA layers were randomly initialized within a small range relative to the maximum weight value. During each simulation iteration, the network is presented with a sample from each class. After each sample presentation, there is a resting stage, without input spikes, where the variables reset to their resting values. The order of class presentation is maintained consistently across iterations to ensure an equitable stimulation between classes and prevent biases. The weights are recorded at fixed time intervals while the training set is iterated multiples times.

A similar network is simulated for testing, the difference being that the weights between the input and the WTA layer are now static and equal to the weights deduced in the learning phase. The network is then simulated with the testing dataset. The purpose of this first trial with the test set is to analyse the neurons behaviour when equipped with the learned weights. After learning, certain WTA neurons exhibit a resembling behaviour when presented with patterns from the same class. More specifically, the neurons firing time, relative to the beginning of the pattern, is proximate. The WTA mechanism promotes the emergence of sparse and distributed synaptic representations across the sub-patterns that constitute the total presented patterns. If a neuron learns a sub-pattern that occurs repeatedly in the full movement pattern, it could lead to multiple activations of that same neuron. The same applies to sub-patterns that occur in different movements. If different movements share the same sub-patterns, a neuron that learns that sub-pattern is expected to respond to multiple classes of movements. For this reason, the neurons behaviour analysis is restricted to only the first emitted spike inside each time window corresponding to a movement pattern presentation. The neurons whose behaviours are more consistent are selected and assigned to a class. To be eligible for a class, neurons must fire at least 90% of the times for that class. Additionally, their spike emission times (relative to the beginning of the pattern presentation) need to be relatively consistent across testing iterations. If the mean absolute deviation is smaller than 150 ms, the neuron is attributed to a class. This value corresponds approximately to 10% to 20% of the durations of patterns. Each class is represented by an output neuron. The selected WTA neurons are then connected to the corresponding output neurons. The firing time differences between WTA neurons assigned to the same class are used to setup delays between the WTA and the recognition neurons. These delays were calculated to synchronize the WTA neurons activity, producing a higher response in the output neurons. WTA neurons will distribute and become responsive to the different spatial patterns that compose the movements spatiotemporal patterns. By synchronizing their responses through delayed synapses, it is possible to assemble a spatiotemporal pattern detector [28]. Figure 13 shows the response of a randomly picked group of WTA neurons, after training, when presented with spike patterns from the testing dataset. There are two conditions for a neuron to be assigned to class. First, it must fire in at least 90% of the class presentations. Also, the sum of the absolute difference between each spike time and the average spike time (referenced to the beginning of the presentation of each pattern) must not exceed a pre-specified threshold. If both conditions are

satisfied, the WTA neuron is connected to the class recognition neuron via a delayed synapse. As depicted in figure 13, for each neuron, this delay is calculated as the difference between its average firing time of neuron and the average firing time of the neuron with the highest average firing time. Figure 14 shows the several responses of competitive neurons are presented to the testing set. It is possible to observe that some of them present a similar behaviour across presentations of patterns belonging to the same class presentation. Finally, a network with the learned weights and asserted delays is once again presented with the testing dataset and the performance of the network is assessed.

The implementation of the proposed SNN was carried out with a focus on the optimization of simulation efficiency. There is no mechanism that prevents two neurons from firing at the exact same time. This was due to the fact of how Brian2 updates neurons variables. Each time step in a Brian2 simulation follows a specific ordered series of update operations. Essentially, these updates include recalculating any time-dependent variables, checking the firing condition, propagate any spikes that have occurred and perform any necessary reset operations. Each of these update operations is applied to all neurons within the same group every timestep and one neuron at a time. To implement a WTA mechanism using Brian2, an evaluation must be performed to each neuron to conclude which one is the “winner”’. The library performs network updates either with Cython or C++ optimized code. If it doesn’t have access to available implementations in these languages, it resorts to Python. This last option introduces a large overhead in simulation speed. A possible workaround would be to write user-defined functions in Cython. The primary motivation for using Brian2 is to eliminate the need for complex and low-level implementations also making this approach less preferred. Instead, in the implemented solution this computation is not performed. To minimize the chance of multiple neurons crossing the threshold in same simulation timestep, certain aspects were considered. Increasing the excitatory synaptic current, rather than the membrane voltage, allows for more differentiation in the response of neurons to stronger inputs. This improvement was suggested[1] by Marcel

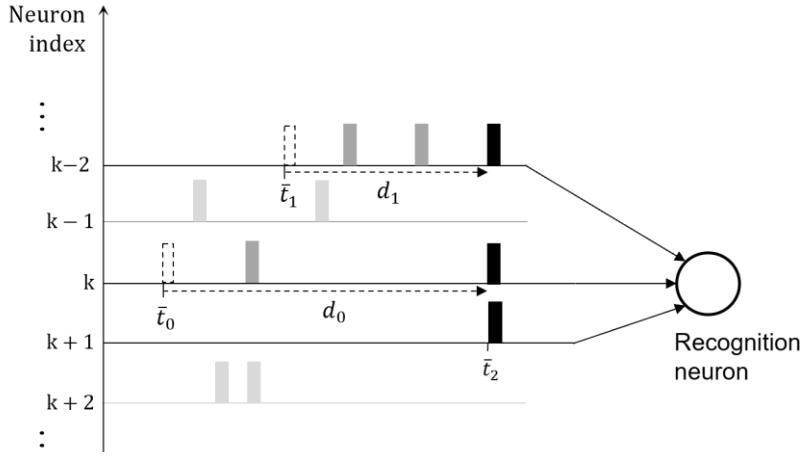

Figure 13 Illustration of neural response synchronization using delayed synapses. A set of neurons is connected to a recognition neuron. A delay is added to synapses connecting the first two firing neurons (k and k-2) such that their spikes occur at the same time as the first spike of the last firing neuron (neuron k+1)

[1]Brian2 Forum Discussion – “Winner Take All mechanism using lateral inhibition”

Stimberg, who dedicates to the development of Brian2. Additionally, the time constants of the neurons membrane voltage, τ_m, and excitatory synaptic current, τ_e, were set to small values to enable faster responses. Table 5 presents the values used in simulations for the various network

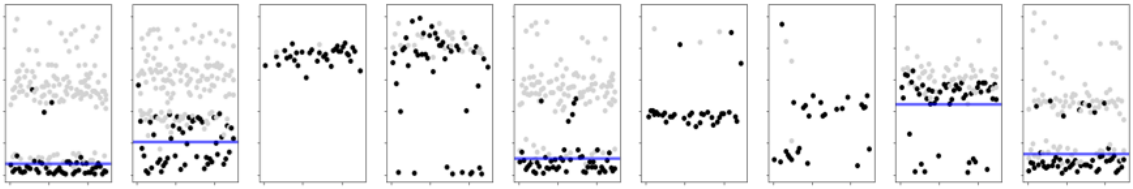

Figure 14 Response of 9 WTA neurons to a class. The samples used are from the testing set. Each column corresponds to a WTA neuron. Each point consists to the firing time (relative to the beginning of each pattern presentation) for each pattern presentation (50 presentations per class). The first spike is represented with a black dot, and the remaining in grey. The average firing time of each neuron for a particular class is represented with a blue horizontal lines. For reference, the latest spikes of each movement occur around 1.0 second, respectively (i.e., the y-axis for each movement).

parameters.

Parameter	Value	Meaning
N_{edge}	20	Number of input neurons dimension
N_i	${N_{edge}}^3 = 8000$	Number of input neurons per sensor
N_o	61	Number of WTA neurons per sensor
V_{rest}	-65 mV	Resting potential
V_{reset}	-65 mV	Reset potential
τ_m	30 ms	Membrane time constant
τ_e	5 ms	Excitatory synaptic current time constant
τ_i	20 ms	Inhibitory synaptic current time constant
R_e	1 Ω	Excitatory membrane resistance
R_e	1 Ω	Inhibitory membrane resistance
τ_r	10 ms	Refractory period
Δ_{th}	3 mV	Adaptive threshold increment
τ_{th}	400 ms	Adaptive threshold time constant
τ_{pre}	20 ms	Presynaptic trace time constant
τ_{post}	20 ms	Postsynaptic trace time constant
w_{max}	1 mA	Maximum weight
w_e	20 mA	Excitatory synaptic weight constant
w_i	1	Inhibitory synaptic weight constant
A_{post}	-1.05 * Apre	Postsynaptic trace increment
T_{inh}	10 ms	Inhibition period
τ_{out}	40 ms	Recognition neurons membrane time constant

Table 5 SNN parameters

5

Experimental results

5.1 Dataset analysis 46
5.2 Results 50

This chapter presents the results and analysis of the conducted experiments to evaluate the effectiveness and performance of the proposed approach. Chapter 5.1. explores the characteristics of the dataset used for training and testing the SNN. The network performance and final state are described in chapter 5.2.

5.1 Dataset analysis

Figure 15 presents the dataset distribution different classes, organized by sensor axis. By inspecting the diagram in a row-wise fashion, it becomes apparent that some classes exhibit similar distributions across the same axis. This also happens for the gyroscope first and last columns movements. Since the learnable weights relate to each sensor individually (i.e., correspond to a particular encoding population and competition layer), overlapping data distributions could pose potential challenges for class discrimination. Table 6 provides with

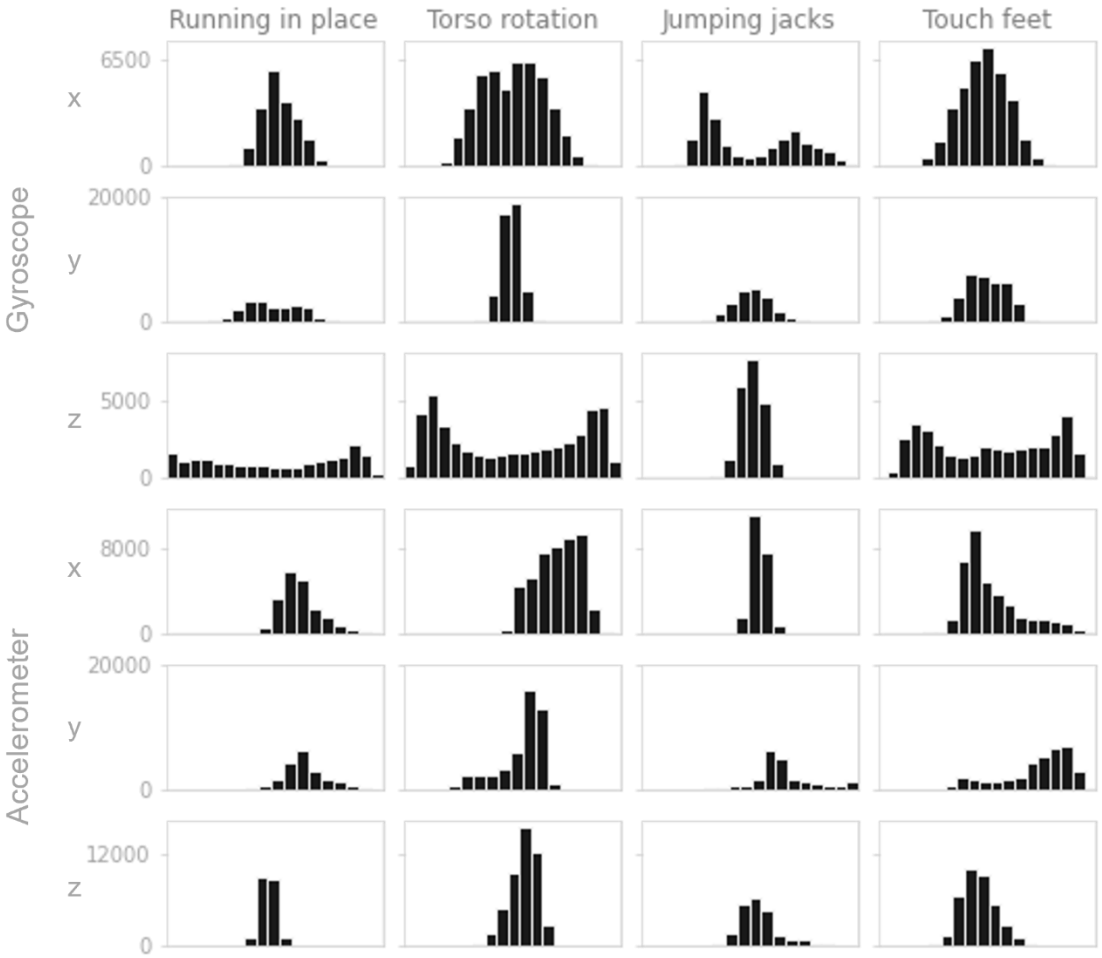

Figure 15 Dataset distribution across per class and sensor axis. The number of bins matches the number of intervals that results from linearly and equally distributing $N_{edge} = 20$ neurons over one dimension (this was the used value for the carried simulations).

additional insights about the temporal aspects, range of values, and variability of the recorded sensor data.

To complement the statistical analysis, visual representations of the resulting input spike patterns are provided. The first, in figure 16, shows the spike times by input neuron for each class. The second, illustrated in figures 17 and 18, leverages the three-dimensional nature of the input layer neurons to illustrate the input firing distribution over time.

Activity	Number samples	$\bar{T}$ (s)	σ_T (ms)	Average minimum	Average maximum	Axis	Sensor
Cross-toe touch	219	1.65	73.90	-0,434	0,753	x	A
				-0,280	0,850	y	
				-0,342	0,311	z	
				-0,392	0,292	x	G
				-0,477	0,365	y	
				-0,771	0,818	z	
Jumping jacks	229	0.92	57.74	-0,100	0,251	x	A
				-0,251	0,997	y	
				-0,214	0,599	z	
				-0,542	0,843	x	G
				-0,389	0,423	y	
				-0,221	0,243	z	
Running in place	233	0.85	44.97	-0,037	0,616	x	A
				-0,027	0,728	y	
				-0,202	0,075	z	
				-0,247	0,349	x	G
				-0,438	0,417	y	
				-0,968	0,861	z	
Torso rotation	259	1.79	157.19	0,042	0,704	x	A
				-0,445	0,306	y	
				-0,048	0,303	z	
				-0,368	0,424	x	G
				-0,177	0,165	y	
				-0,829	0,875	z	
	940	1.3	83.45				
	Total	Average					

Table 6 Dataset attributes. The labels “A” and “G” correspond to the accelerometer and gyroscope, respectively. $\bar{T}$ and σ_T refer to the samples average duration and the standard deviation of the duration, respectively.

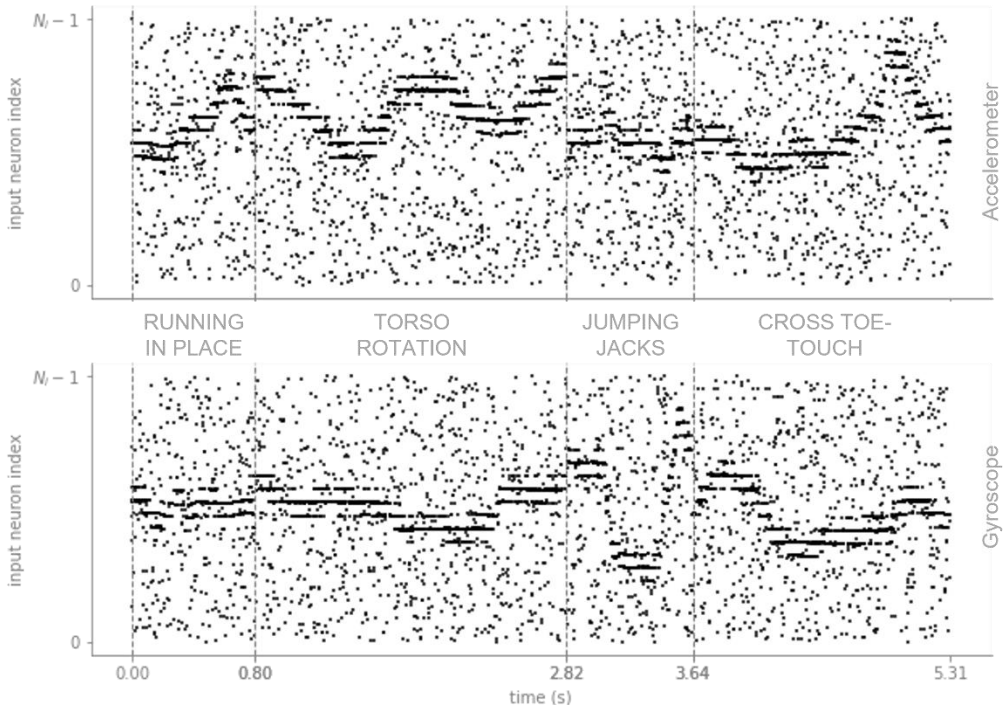

Figure 16 Dataset movements spike patterns

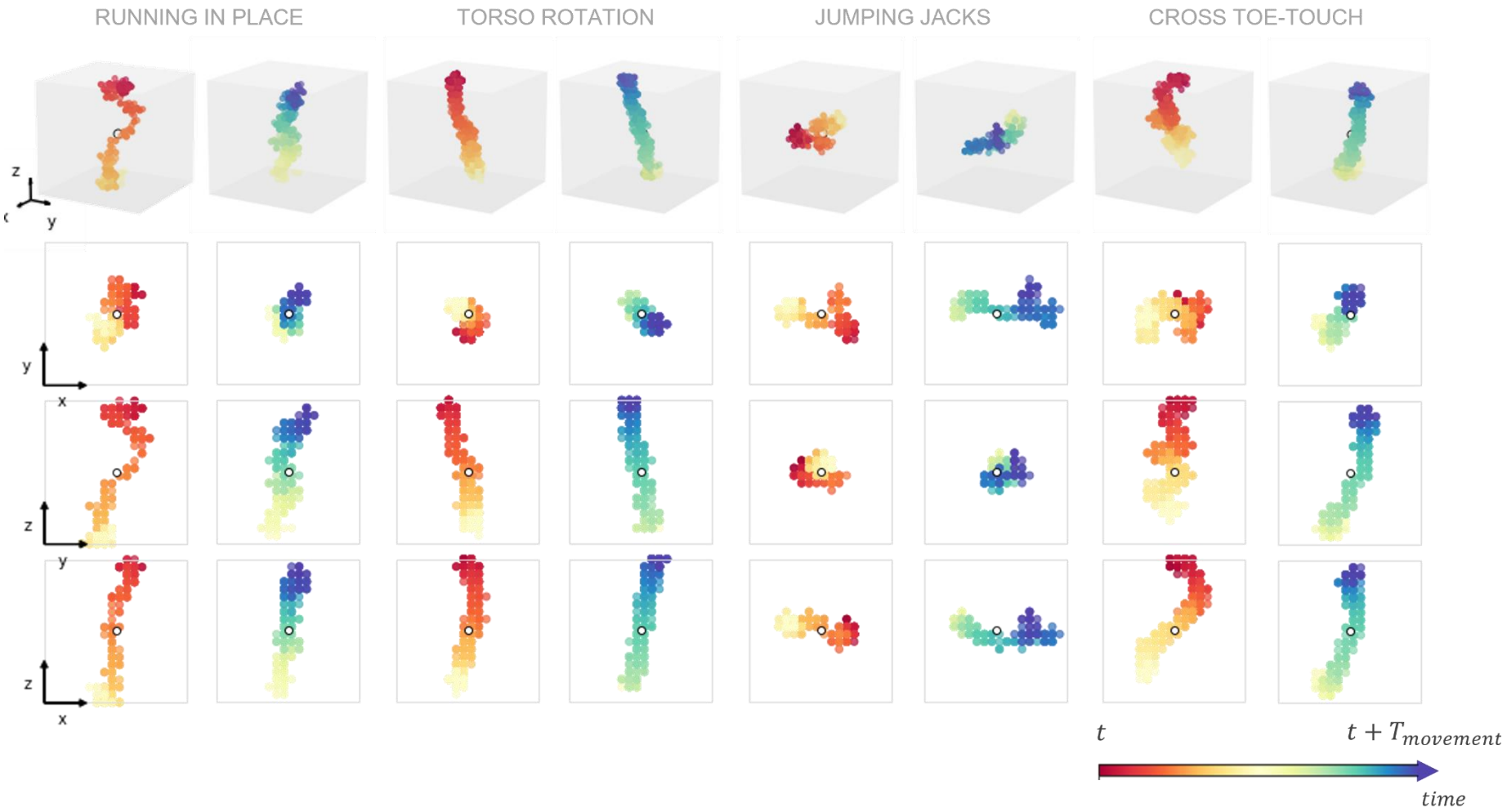

Figure 17 Spatial distribution of the movements spike patterns over time, referring to the gyroscope. $T_{movement}$ represents the duration of each movement (see Table 5 for references)

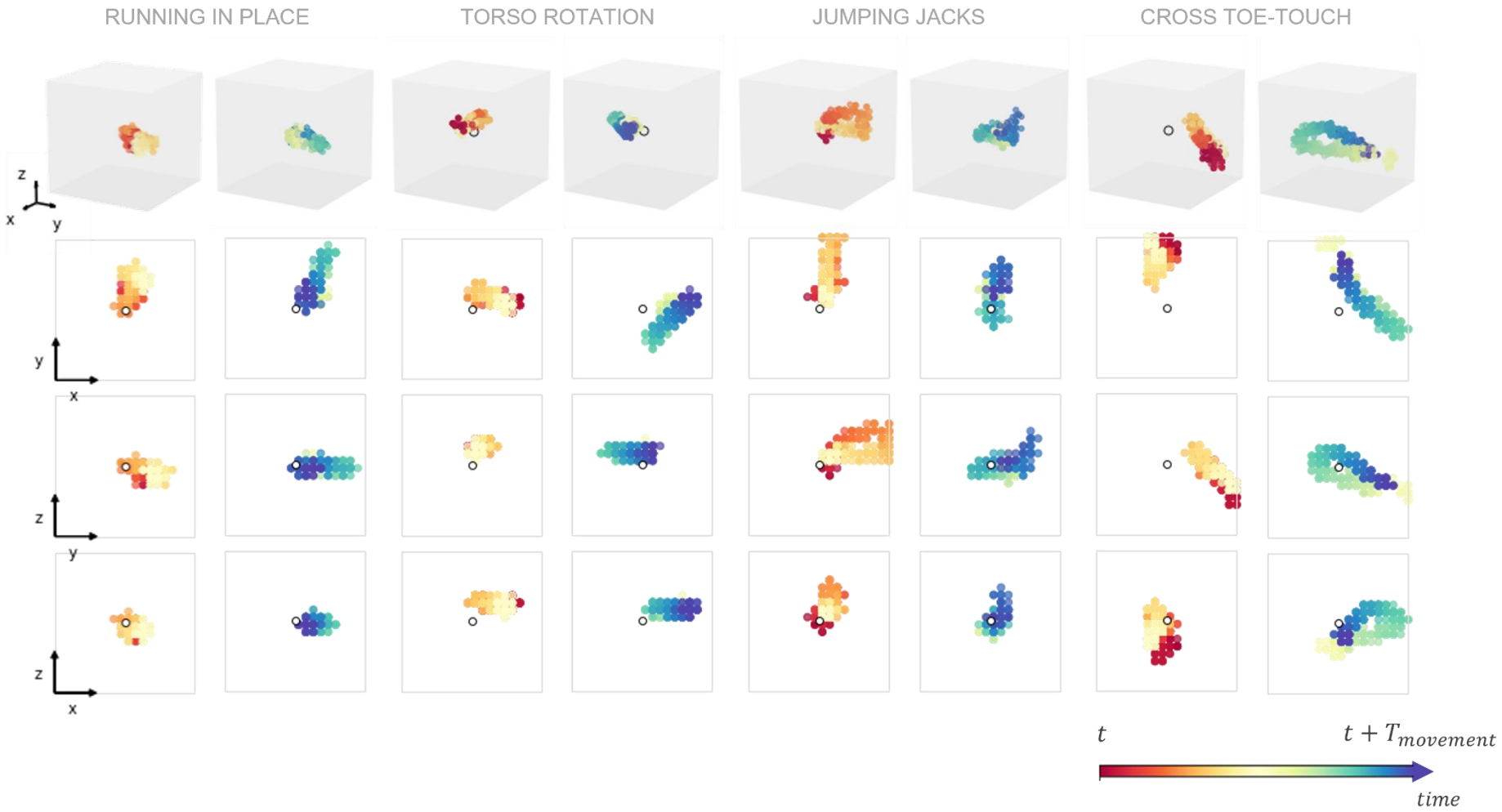

Figure 18 Spatial distribution of the movements spike patterns over time, referring to the accelerometer. $T_{movement}$ represents the duration of each movement (see Table 5 for references)

5.2 Results

The implemented solution demonstrated good levels of performance when tested with various parameters sets. More specifically, the network was tested with different encoding frequencies. Figure 19 illustrates how different the competition layer neurons firing rate is affected by the variation of input encoding rates. More specifically, figure 19a shows the linear relation between these two variables, for a constant threshold potential (-59 mV in this example). The second graphic (see figure 19b) shows how the decrement (note that the axis is reverted) of the WTA neurons threshold potential leads to an increase of their firing rate, for different input rates. This increment appears to follow an exponential pattern, which can be attributed to the inherent properties of the LIF neuron model. Multiple experiments were conducted, with different pairs of values for the input rate and the threshold potential. These values were selected to ensure that the WTA neurons achieve approximate firing rate of 20 Hz across all experiments.

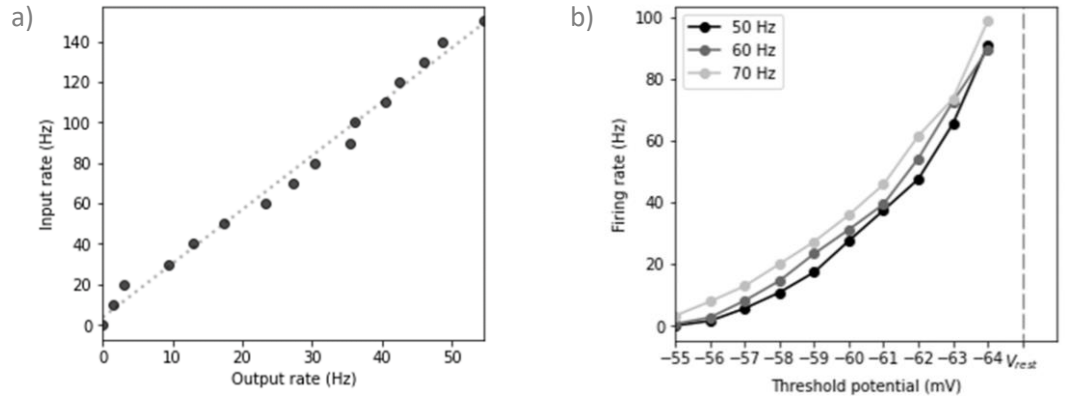

Figure 19 Dynamics of the implemented network for different input rates. (a) shows the variation of the WTA layer neurons firing rate, for different values of f_{zone} (see chapter 4.4.2) and a fixed threshold of -60 mV. (b) WTA neurons firing rate relation with their threshold potential for = [50,60,70] Hz

The accuracy recognition for each class is determined by calculating the percentage of the number of times each output neuron presented the highest membrane potential, when presented with its corresponding class, relative to the number of presentations per class. 150 samples were used for training, and 50 for testing. During training, the weights were recorded with a fixed periodicity. After, the network recognition performance was assessed for each weight recording, allowing to extract its progressive improvement, as shown by Figure 20. This assessment was performed using the testing samples. Figure 20 considers various scenarios, with different input rates, and shows how the network performance improves over time, demonstrating the effectiveness of the training process. There is not a widely adopted criterium that defines when the STDP training is over (i.e., when the weights converged) and this work doesn't aim to establish one. Table 7 further details the maximum accuracy results achieved for each experiment.

Figure 21 represents the learned weights for one the experiments referred above. It is possible to observe that different weights stimulations occurred for each WTA neuron, indicating the successful learning of different input pattern by different WTA neurons. Also, figure 22 demonstrates that the weights that exhibit greater stimulation correspond to a three-dimensional space where the input values reside (see figures 17 and 18).

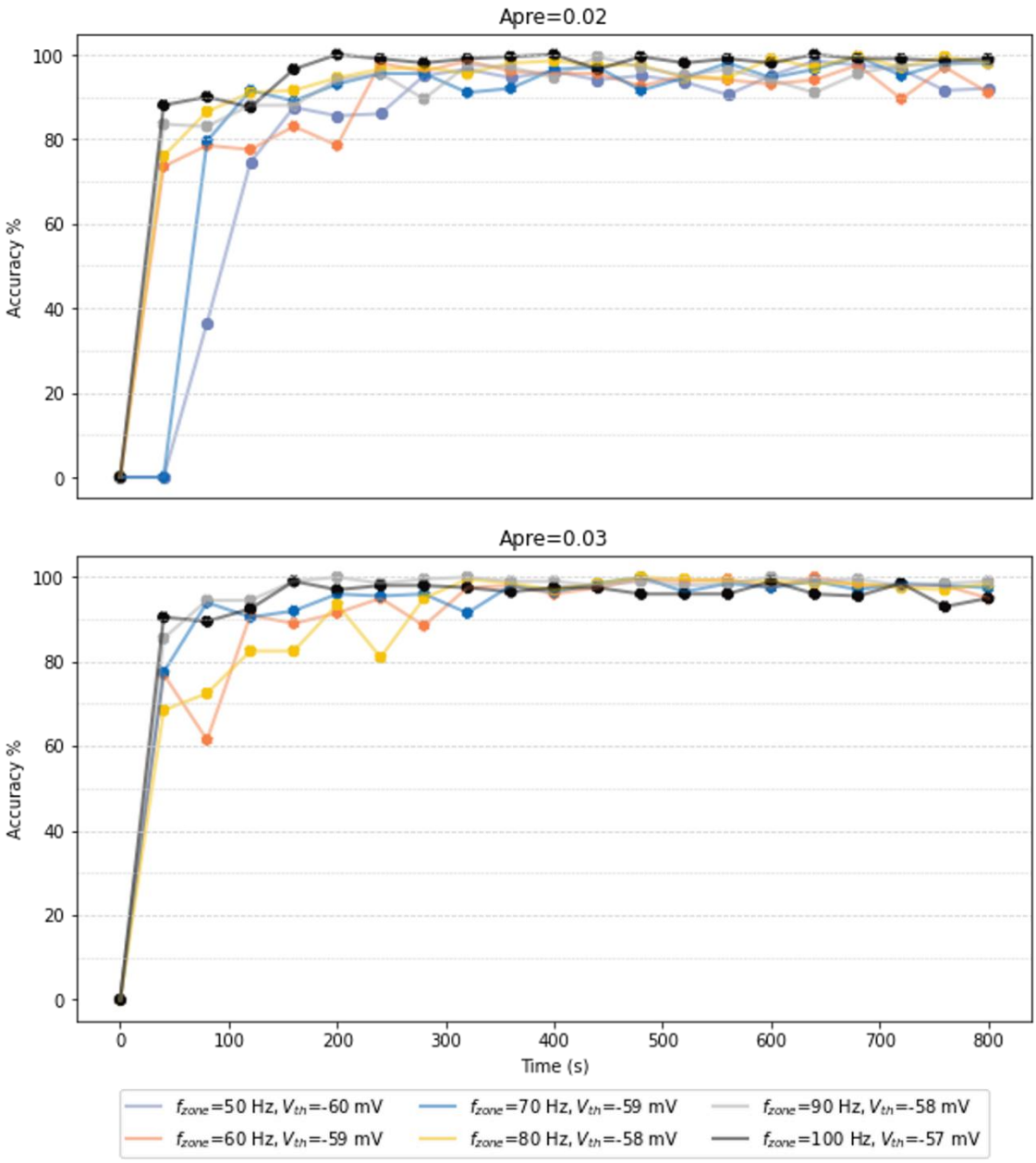

Figure 20 SNN accuracy for two learning rates and multiple input rates

Apre	f_{zone} (Hz)	Accuracy (%)	Time to reach maximum accuracy (s)
0.02	50	98.5	640
	60	98.5	320
	70	99.5	680
	80	99.5	680
	90	99.5	440
	100	100.0	200
0.03	60	100.0	640
	70	100.0	480
	80	100.0	480
	90	100.0	200
	100	99.0	160

Table 7 Recognition accuracy for the experiments considered in figure 20.

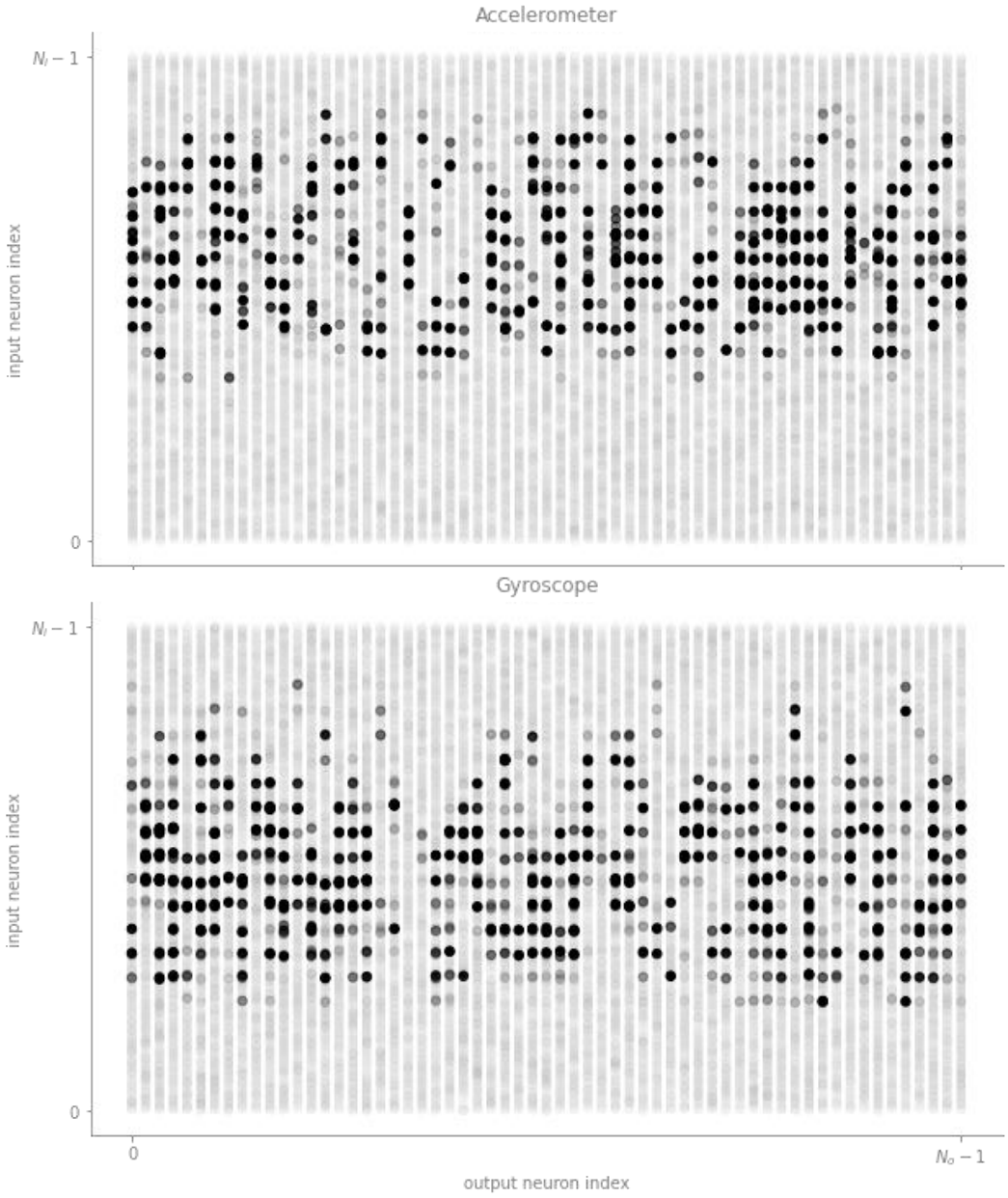

Figure 21 WTA neurons final weights. Blacker dots indicate a higher weight value

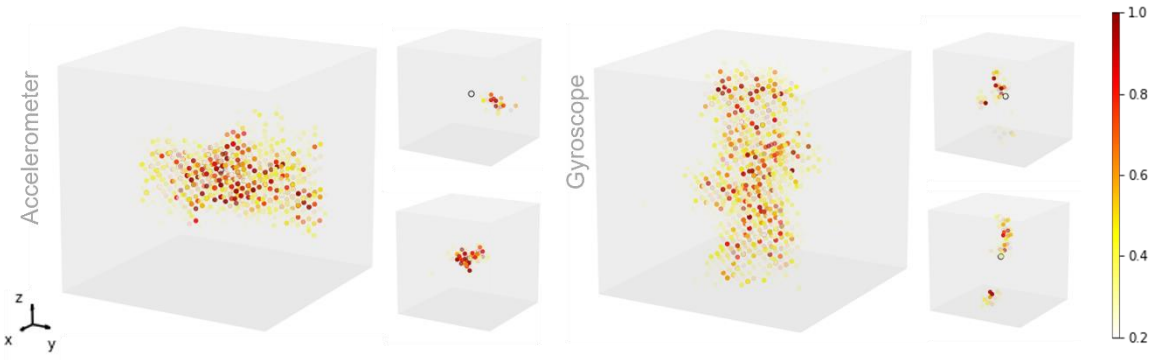

Figure 22 Spatial distiribution of the WTA neurons final weights for both sensors. The bigger cubes show the all the weights, while the smaller cubes refer to individual randomly picked WTA neurons.

A final experiment was conducted to assess the network behaviour when presented consecutive repetitions of movements, instead of segmented samples. Figure 23 illustrates the results of such

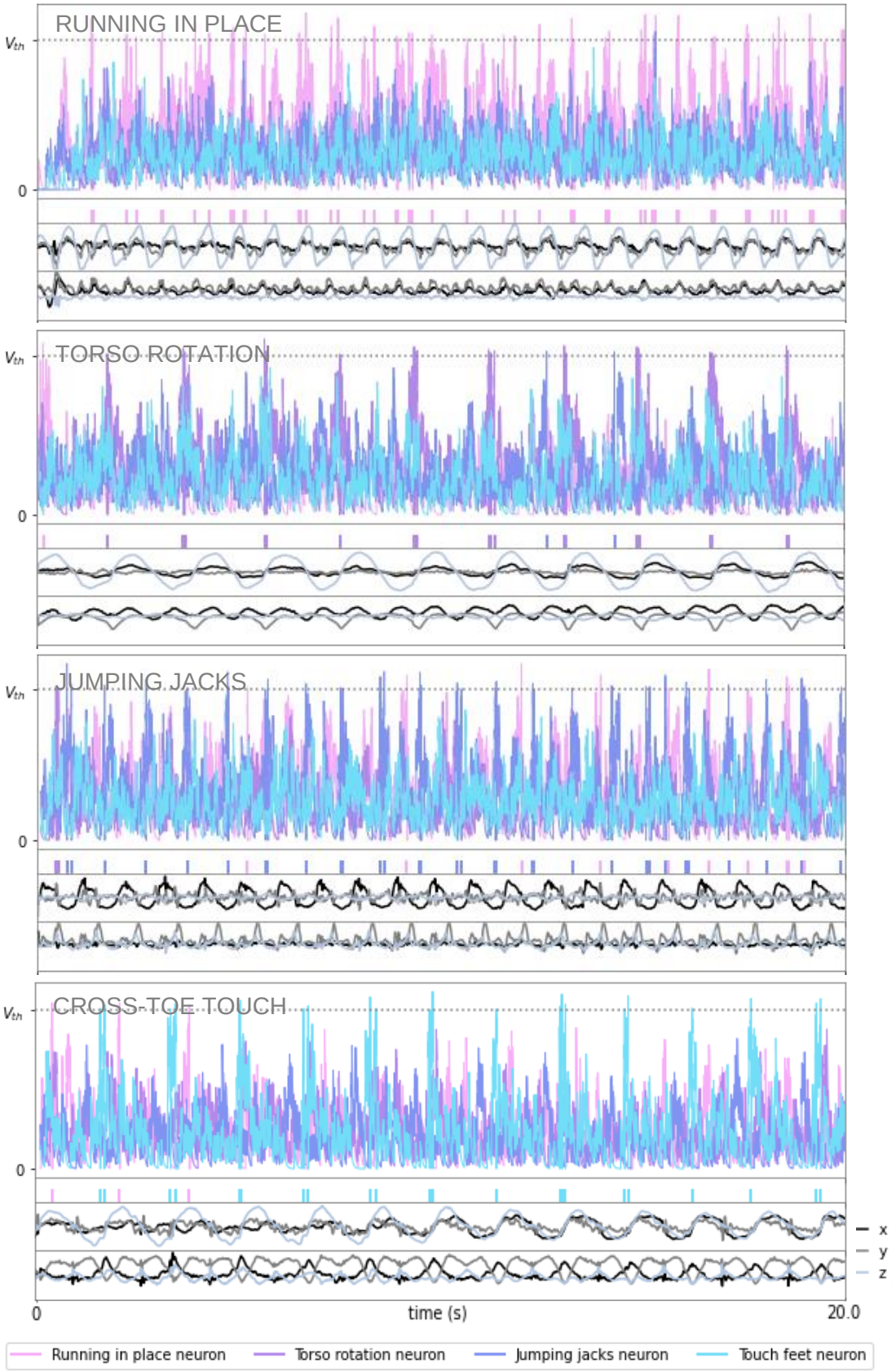

Figure 23 Recognition neurons response to streams of movement repetitions. Each of the four graphics consist of the recognition neurons membrane potential and spikes, and the gyroscope and accelerometer values (if seen vertically) when presented with each class. ($V_{th} = 0.23$)

experiment. For this test, a firing condition was set for the recognition neurons. If the value of their membrane potential, divided by the number of connections received from the WTA layer, is higher than a prespecified threshold, they fire. After, the membrane potential is reset to zero. The recognition neurons show a noticeably higher firing rate when presented with their corresponding class. More specifically, and following the vertical movements order in Figure 23, the number firings of each class neuron versus the number of total firings was 50/51, 20/23, 34/46 and 22/25 (number of firings of corresponding class neuron/number of toral firings). It is possible to notice that when a recognition neuron is presented with a stream of movements corresponding to the neurons attributed class, it exhibits a much higher firing rate then the remaining neurons.

6

Conclusion

6.1 Discussion 56
6.2 Future work 56

To consolidate the conducted work, this final section presents some reflections about the achieved results. the conclusions derived from this work and presents some potential suggestions for future research. In chapters 6.1, the main derived conclusions are outlined. Chapter 6.2 explores potential ideas for future work.

6.1 Discussion

The purpose of this document is to provide a detailed explanation of the conducted research, within the scope of this dissertation. The implementation of the several tasks comprising this project are demonstrated throughout this dissertation. The tasks involved in the execution of the project included setting a data acquisition (using the SensorTag) and processing pipeline for the creation of a dataset, and the development of a SNN capable of discriminating the multiple classes within dataset.

The presented results demonstrate that the competitive neurons effectively learned spatial patterns consisting of distinct sets of three-dimensional coordinates. The proposed encoding method originates multiple synapses between the input and the competition layers. This proved beneficial as it allowed to distribute the learning across a greater number of synapses. By synchronizing the WTA neurons response (i.e., synchronize multiple three-dimensional coordinate detections), using delayed synapses, the recognition layer neurons response became maximized, and differentiated, resulting in an accurate identification of the different dataset classes. Although the network shows good performance, the results would be more meaningful if the same system had been applied to a dataset comprising several participants. Nevertheless, the results indicate the promising potential of the proposed approach.

6.2 Future work

The proposed approach intends to present foundations for future research. It shows reliable to be applied in new scenarios with variations in mechanisms.

It is acknowledged that the SNN performance results reflect a scenario where the dataset samples refer to single individual. To further validate the solution generalization capabilities, a next step could be to test the approach with a dataset comprising several participants. Also, other variations of the proposed encoding scheme can be implemented. In addition to the spatial distance, each input neuron can also be constrained to fire for values changes exceeding a specific threshold. This would enable the encoding method to incorporate time-based events. The spike coding used in this work leads to an almost constant input firing rate. The number of input neurons that are spatially close enough to an inputted tridimensional coordinate, and can thus fire, only changes when these values become closer to the extremes. Consider the spatial distribution of the input neurons, which lie inside a cubic space. If we'd compute the input neurons that are in_{zone} for a tridimensional coordinate corresponding to a vertix of the cube, there would be less available neurons then for a coordinate corresponding to the center of the cube. Event-based spike patterns can cause this rate to significantly vary. Therefore, a few adaptations in the remaining layers of the network would be required for effectively handling such outcomes.

Bibliography

[1] D. a. C. M. Floreano, Bio-inspired artificial intelligence: theories, methods, and technologies, MIT press, 2008.

[2] W. S. M. a. W. Pitts, "A logical calculus of the ideas immanent in nervous activity" *Bulletin of Mathematical Biology,* vol. 5, pp. 115-133, 1943.

[3] S. N. a. M. S. DC Somers, "An emergent model of orientation selectivity in cat visual cortical simple cells" *Journal of Neuroscience,* vol. 15, no. 8, pp. 5448-5465, 1995.

[4] E. R. e. a. e. Kandel, Principles of neural science, New York: McGraw-hill, 2000.

[5] D. O. Hebb, "The organization of behavior: A neuropsychological theory" *The American Journal of Psychology ,* vol. 63, no. 4, pp. 633-642, 1949.

[6] B. a. Poo, "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type" *Journal of Neuroscience,* vol. 18, no. 24, pp. 10464-10472, 1998.

[7] P. M. Bi GQ, "Synaptic modification by correlated activity : Hebb's postulate revisited" *Annu. Rev. Neurosci. ,* vol. 24, pp. 139-166, 2001.

[8] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain" *Psychological Review,* vol. 65, no. 6, pp. 386-408, 1958.

[9] A. G. &. L. V. G. Ivakhnenko, Cybernetic Predicting Devices, vol. 803, Joint Publications Research Service, 1964.

[10] S. Linnainmaa, The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors, Master's thesis, University of Helsinki, 1970.

[11] W. Maass, "Networks of spiking neurons: The third generation of neural network models" *Neural Networks,* vol. 10, no. 9, pp. 1659-1671, 1997.

[12] A. M, "Role of the cortical neuron: integrator or coincidence detector?" *Isr J Med Sci,* no. 18, pp. 83-92, 1982.

[13] E. A. S. W. Konig P, "Integrator or coincidence detector? The role of the cortical neuron revisited" *Trends Neurosci,* no. 19, pp. 130-137, 1996.

[14] A. F. H. A. L. Hodgkin, "A quantitative description of membrane current and its application to conduction and excitation in nerve" *J Physiol,* vol. 4, no. 117, pp. 500-544, 1952.

[15] S. D. A. &. V. R. R. Thorpe, "Spike-based strategies for rapid processing" *Neural Networks,* vol. 14, no. 6-7, pp. 715-725, 2001.

[16] D. &. L. S. B. Attwell, "An energy budget for signaling in the grey matter of the brain" *Journal of Cerebral Blood Flow & Metabolism,* vol. 21, no. 10, pp. 1133-1145, 2011.

[17] T. Masquelier, "STDP Allows Close-to-Optimal Spatiotemporal Spike Pattern Detection by Single Coincidence Detector Neurons" *Neuroscience,* vol. 389, pp. 133-140, 289.

[18] C. M. Diehl PU, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity" *Front Comput Neurosci,* vol. 9, no. 99, 2015.

[19] G. R. T. S. Masquelier T, "Competitive STDP-based spike pattern learning" *Neural Computation,* vol. 21, no. 5, pp. 1259-1276, 2009.

[20] M. S. R. B. D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator" *eLife,* vol. 8, no. e47314, 2019.

[21] N. T. a. H. M. L. (. Carnevale, The NEURON Book, Cambridge, UK: Cambridge University Press, (2006)..

[22] M.-O. G. a. M. Diesmann, "NEST (NEural Simulation Tool)" *Scholarpedia,* vol. 2, no. 4, p. 1430, 2007.

[23] "BLE Project Zero" Texas Instruments, [Online]. Available: https://software-dl.ti.com/lprf/simplelink_academy/modules/projects/ble_projectzero/information.html . [Accessed 2023].

[24] "Bluetooth Low Energy Custom Profile" Texas Instruments, [Online]. Available: https://software-dl.ti.com/lprf/simplelink_academy/modules/ble_01_custom_profile/ble_01_custom_profile.html#example-service-generator. [Accessed 2023].

[25] Apple, "Watch - Apple" 2015. [Online]. Available: https://www.apple.com/watch/. [Accessed 2023].

[26] FitBit, "FitBit Website" [Online]. Available: https://www.fitbit.com/global/us/home. [Accessed 2023].

[27] "Identification of 3D spatiotemporal patterns using SNNs," [Online]. Available: https://github.com/diogohmsilva/HAR. [Accessed 2023].

[28] Deco, T. Masquelier and Gustavo, "Learning and Coding in Neural Networks" 2013.