Eugen-Richard ARDELEAN

Computational Methods for Brain Signal Analysis during Perception and Behaviour

U.T.PRESS
Cluj-Napoca, 2024
ISBN 978-606-737-718-7

Eugen-Richard ARDELEAN

Computational Methods for Brain Signal Analysis during Perception and Behaviour

U.T.PRESS
Cluj - Napoca, 2024
ISBN 978-606-737-718-7

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
http://biblioteca.utcluj.ro/editura

Recenzia:

Prof.dr.ing. Rodica Potolea
Prof.dr.ing. Mihaela Dînșoreanu
Conf.dr.ing. Costin Chiru
Prof.dr.ing. Călin Adrian Popa
C.S.I.dr. Raul Cristian Mureșan
Conf.dr.ing. Camelia Lemnaru

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-718-7

PREFACE

This work is based on the PhD thesis with the same name. I would like to thank my PhD supervisor, **Prof. Dr. Eng. Mihaela Dînșoreanu**, all the members of the guidance and examination committees for spending valuable time in reading and helping me amend this work. I would like to thank **Prof. Dr. Eng. Rodica Potolea** and **Prof. Dr. Eng. Mihaela Dînșoreanu** for trusting me and allowing me to teach the laboratory sessions of their disciplines, thus giving me the opportunity to attain one of my dreams. I would like to thank my colleagues at the Transylvanian Institute of Neuroscience for the various discussions (especially, during smoke breaks), those related to research and even more for those that were not. I would like to thank the students, with whom I had the pleasure to have worked with, those that have chosen me as their supervisor or co-supervisor and those that have shown interest in the domain of neuroscience.

I dedicate this work to the woman that raised me even though she did not birth me, without whom I would never have been able to get to where I am today, to my mother Monica-Adriana Ardelean (Petcheşi).

TABLE OF CONTENTS

1. Introduction 1

Context: An abridged history of neuroscience 1

Motivation: Bridging the gap between neuroscience and computer science 1

Methods for brain activity identification 2

Methods for brain activity characterisation 2

From brain activity to behaviour 2

Structure 3

2. Domain knowledge 5

2.1. Neuroscience domain knowledge 5

2.1.1. The brain 5

2.1.2. Brain complexity 7

2.1.3. Brain oscillations 16

2.1.4 Brain recording techniques 18

2.2. Computer science domain knowledge 24

2.2.1. Architectures of artificial neural networks 24

2.2.2. Machine learning approaches 36

2.2.3. Signal processing 36

2.3. Neuroscience and computer science 40

2.3.1. Spike sorting 40

2.3.2. Burst detection 56

2.3.3. Symbolic Analysis 58

2.3.4. Detection of oscillations 59

3. Methods for brain activity identification 61

3.1. Data analysis in spike sorting 61

3.1.1. Introduction 61

3.1.2. Data 62

3.1.3. Methods 66
3.1.4. Conclusions 108
3.2 Burst detection 109
3.2.1. Introduction 109
3.2.2. Data 110
3.2.3. Methods 111
3.2.4. Conclusions 117
4. Methods for brain activity characterisation 119
4.1. Symbolic analysis 119
4.1.1. Introduction 119
4.1.2. Data 120
4.1.3. Methods 120
4.1.4. Conclusions 128
4.2. Detection of oscillation packets 129
4.2.1. Introduction 129
4.2.2. Data 129
4.2.3. Methods 129
4.2.4. Conclusions 139
5. From brain activity to behaviour 140
5.1. Experimental environment development for behaviour quantification 140
5.1.1. Introduction 140
5.1.2. Methods 141
5.1.3. Conclusions 147
5.2. Behaviour quantification through tracking 149
5.2.1. Introduction 149
5.2.2. Methods 152
5.2.3. Conclusions 156
REFERENCES 157

APPENDICES 170
List of figures 170
List of tables 177
List of abbreviations 179
Glossary 181

1. Introduction

Context: An abridged history of neuroscience

The brain has been the focus of devoted research for more than a hundred years starting with the pioneering work of Santiago Ramon y Cajal and Camillo Golgi during the late 19th century. Their eminent work is the basis of neuroscience today and it allowed the segregation of neuroscience from medicine as a freestanding multidisciplinary science. Nevertheless, the interest in the brain did not start so late.

It is sensible to say that it has been around for a long time as even Egyptian manuscripts hint that brain damage symptoms were a part of the knowledge of the time. Moreover, in the process of mummification, the brain was one of the organs that was excised. Admittedly, at that time, it was the heart that was thought of as the 'seat of intelligence', a paradigm that would last until Hippocrates and Plato. The role and functions of the brain have been much discussed by philosophers to this day.

The mediaeval epoch in the Muslim world was another leap for neuroscience, as the first steps in neurosurgery and neurological diagnosis were made by Abulcasis and others. During the Renaissance, physician and anatomist Andries van Wesel had created the most detailed portrayal of brain anatomy up to that time.

The invention of the microscope and the silver-based staining method of Golgi allowed further advancement in the study of the brain. This staining procedure was used by Ramon y Cajal in his work and through it the neuron doctrine was established - the idea that the neuron is the functional unit of the brain.

Motivation: Bridging the gap between neuroscience and computer science

Lately, there is a tendency towards interdisciplinarity, and we can see it even within the field of computer science. During the 20th century, biologically inspired algorithms have been developed, such as the neural network or genetic algorithms. These kinds of algorithms are widely used today, although their association to biology has been diminished. Domains such as 'Computational Neuroscience' attempt to reestablish these connections and bridge the gap by combining experimental neuroscience, theoretical biology, molecular biology and engineering, and computer science.

In recent times, we have seen computer science algorithms find applications in various domains through the rise of neural networks. In neuroscience, new brain recording tools [1] have been developed with hundreds or even thousands of electrodes that can record brain signals resulting in a never-before-seen amounts of data rendering many analysis methods inadequate. Because of this, neuroscience now truly requires advanced computer science techniques that can deal with all this information as efficiently and robustly as possible. For a long time, neuroscience and computer science existed as separate disciplines, but this need has made both neuroscientists and computer science experts more interested in working together more than ever. This collaboration can allow both disciplines to grow as neuroscience knowledge may inspire the creation of new algorithms and computer science algorithms may provide new discoveries of how the brain works.

It is within this domain that the author attempts to bring a contribution as no endeavour seems more honourable than aiding, however insignificantly, the advancement towards our goal to understand others and ourselves.

Methods for brain activity identification

Neurons communicate through electrical impulses, or spikes, which are recorded using electrodes. Analysing these spikes is important for the understanding how information is processed in the brain at a neuronal level. Spike sorting [2] is a first example of how neuroscience and computer science have come together as it involves the identification and categorisation of spikes through machine learning and signal processing approaches. And it is also a primary research direction for this work. From a computer science perspective, spike sorting [2] can be viewed as a signal processing problem that involves a pipeline of several steps: data filtering, feature extraction, and clustering algorithms. In many cases, spike sorting is still done manually, semi-automatic or with simple approaches such as a combination of Principal Component Analysis [3] and K-means [4]. These methods have been shown to have decent performance but there is a lot of room left for improvement, creating the need for more complex algorithms that are capable of handling the challenges of neural data. Moreover, with the recent hardware advancements [1], there is an increased need for more performant algorithms.

Methods for brain activity characterisation

Shifting from identification to characterisation, an efficient method is required to characterise and interpret the neural activity once it has been identified. Due to the large amounts of data and their complexity, extracting the relevant information with an automated approach and an interpretable format remains a challenge. Symbolic analysis [5] is one approach taken within this work of how to decipher these complex patterns within continuous neural signals through computer science approaches. Through symbolic analysis, neural signals can be encoded into a format more manageable for computation and more interpretable format for us.

Another type of characterisation can be made through the detection and analysis by computer science methods of brain oscillations by moving into the time-frequency domain through signal processing methods. Oscillations have linked to various brain states, and as such through their characterisation, the mechanism through which the brain processes information can be determined.

From brain activity to behaviour

Many experiments are done on anaesthetised subjects that do not allow us to see any link between brain activity and behaviour and they can alter brain activity such that we do not see the normal everyday activity in the brain. As exciting as brain activity analysis is, to understand the brain it is important to link brain activity to behaviour and this can only be done through behavioural experiments. These experiments allow us to understand the brain's responses to stimuli, but they introduce challenges related to experimental design and data analysis. Behavioural experiments must be reproducible and reliable [6], thus their design must be well thought-out to ensure that the measured effects are caused by the control variable and not by other confound factors.

Traditional laboratory settings again do not capture the intricacies of everyday life for any subject; thus, they cannot truly reflect the normal activity of the brain. In contrast,

real-life environments are more dynamic as they involve constant sensory and motor activity. A more naturalistic approach in the experimental environment design [6] can result in more organic behaviour and thus, neural activity which enables researchers to investigate how the brain truly processes information and responds to stimuli.

Computer science can again contribute to this research direction by creating precisely controlled experimental settings, and most importantly by reducing confounding variables. Another avenue for computer science intervention is behaviour tracking. Through computer science methods, meaningful information can be gleaned from the subject's behaviour while also automating the analysis of complex behavioural patterns.

Structure

The first chapter aims to bridge the gap between the neuroscience and computer science domains. Due to recent advancements in hardware, neuroscience requires more performant analysis techniques that computer science can provide.

The second chapter aims to provide the reader with a comprehensive understanding of fundamental concepts in neuroscience (section 2.1), computer science (section 2.2) and computational methods applied in neuroscience (section 2.3). The first section starts with the structural and functional organisation of the brain (section 2.1.1). It continues with a presentation of the structure and functionality of neurons (section 2.1.2) and is concluded with a discussion about intrinsic synchronisation mechanisms of the brain, called brain oscillations (section 2.1.3). This introductory part of the neuroscience domain ends with an overview of different brain recording techniques that provide the researcher with a "window" into the brain for the analysis of data (section 2.1.4). The second section of this chapter presents common machine learning algorithms from computer science, specifically neural networks and signal processing techniques. The last section present computational methods that are applied today within the field of neuroscience.

The next chapters focus on the practical and experimental work done by the author in the domain of neuroscience; a part of this work has been done at the Transylvanian Institute of Neuroscience (TINS). These chapters have been divided into different levels of analysis of experiments.

Chapter 3 presents several newly developed methods for brain activity identification and their analyses. Neurons have two functioning modes (firing patterns), namely tonic firing and bursting, each with its own characteristics. The choice between extracellular recording and intracellular recording involves a trade-off. In intracellular recording, each recorded activity belongs to the neuron into which the electrode was inserted. While in extracellular recordings, where the electrode is introduced between neurons, identifying the activity of a specific neuron becomes challenging due to the overlap with the activity of neighbouring neurons. The assignment of activity to their source neurons is addressed through spike sorting, a pipeline of several techniques. However, spike sorting may not be ideal when dealing with bursting activity, where neurons fire multiple spikes rapidly. In the first section of this chapter (section 3.1), several new approaches are presented that are more efficient and have better performance than several current approaches. These algorithms address the steps of the spike sorting pipeline and tackle the complexities of neuronal data. These algorithms have been shown to have increased performance when compared to classical approaches.

This chapter starts with an introduction to spike sorting (section 3.1.1). Multiple approaches that improve the spike sorting pipeline and their comparative analyses are presented (section 3.1.3): amplitude thresholding approaches based on neural networks (section 3.1.3.1), a new approach in spike sorting for feature extraction based on autoencoders (section 3.1.3.2), a new feature extraction approach for spike sorting based on signal processing (sections 3.1.3.3-4), an approach that combines feature extraction and clustering in a single step (section 3.1.3.5), a new clustering method specifically designed for spike sorting (section 3.1.3.6), an improved version of the clustering algorithm and a new performance evaluation metric for clustering (section 3.1.3.7), and a new approach to distance computation that has applications in clustering and clustering evaluation (section 3.1.3.8). Finally, an overview and the conclusions are presented (section 3.1.4). Section 4.2 presents a new approach for the detection of bursting activity that encompasses the characteristics of neuronal bursts given in the literature. It includes a comparative analysis of existing methods on synthetic data, and it proposes a correlation analysis for real data.

After the identification of brain activity, characterisation is the next necessary step to understanding brain function. Chapter 4 shifts to methods for brain activity characterisation. Section 4.1 presents several approaches to symbolic analysis of EEG data. Symbolic analysis transforms the high amount of data into a more manageable format for information inference and pattern emergence. Section 4.2 presents a newly developed algorithm designed for the detection of brain oscillations from time-frequency representations. Current methods of brain oscillation detection are simple approaches that have limited performance. The introduced method offers a better descriptive power and detection performance albeit with additional execution time.

Chapter 5 makes the transition from brain activity to behaviour as brain activity does not happen in a vacuum and it is inherently related to actions. Section 5.1 presents experimental environment modelling for an experiment that attempts to answer questions about the effects of amblyopia, a visual system disorder, on the brain. Two configurable experimental environments have been developed, the first has the purpose of evaluating the contrast sensitivity of the subject, while the second is the actual experiment to find the impact of amblyopia on visual prediction. Section 5.2 presents a convolutional neural network approach for tracking zebrafish exposed to pharmaceutics. Through tracking and postprocessing that obtains information about location, speed, orientation and movements, the impact of the pharmaceuticals upon the fish can be inferred.

Chapter 6 presents an overview of the whole work in connection to the objectives presented and the results obtained.

2. Domain knowledge

2.1. Neuroscience domain knowledge

Throughout nature, a myriad of complex systems have developed and without a doubt, the brain is one such system. This chapter starts with a brief description of the brain from a structural and functional perspective, followed by an in-depth description of the finer components, neurons and neuronal circuits, and how they interact with each other.

Neuroscience has the goal of understanding the intricacies of the complex interactions within the brain and deciphering how it functions. Neuroscientists are working towards unravelling the mysteries hidden behind the inherent characteristics that define a healthy functioning brain such as cognition, memory, learning, behaviour, and perception, among many others.

2.1.1. The brain

The brain is a bewilderingly intricate system capable of unifying information from both interoception and exteroception while at the same time receiving continuous information and preserving equilibrium in a chaotic environment and allowing actions to be performed and decisions to be made almost instantly. Sight, hearing, touch are the senses through which the brain is able to assess the outside world, whilst the body is a means for interaction. Actions incorporate an unending feedback loop between sensory information received, processing in the brain and behaviour. The brain processes information in a similar fashion, through loops between different areas or subregions until a conclusion is reached in a very short interval.

The brain is only one part of the central nervous system (CNS), which is itself only one part of the nervous system along with the peripheral nervous system. The CNS also includes the spinal cord. The human brain is proportionally similar to that of other primates. The brain weighs about 1.45 kilograms [7], less than 2% of body weight and yet it consumes about 20% of the energy. It is engulfed by the skull, occupying a volume of 1.13 litres in females and 1.26 litres in males (excluding cerebrospinal fluid) [8].

The brain can be anatomically divided into three parts: the brain stem, the cerebellum, and the cerebrum (Figure 2.1). The brain stem can be viewed as the bridge between the cerebrum and the spinal cord. It is heavily involved in the regulation of heart and breathing rate and the sleep cycle. Motor and sensory information of the face, neck and body is bidirectionally sent to and from the brain through the brain stem [9]. The cerebellum, located under the temporal and occipital lobes and in the dorsal part of the brain, is involved in motor control, particularly that of more delicate movements that require coordination and precision [10]. The cerebrum develops prenatally and is the largest and the most developed part of the brain, it incorporates the cerebral cortex and several subcortical regions. The cerebral cortex is present only in mammals, and in larger mammals its surface folds into ridges and furrows, called gyri and sulci, respectively, which greatly enhance its surface area without increasing the volume [11]. It is considered to be the seat for neural integration as it was found to be involved in: awareness, consciousness, thought, attention, memory and language [12]. The cerebral cortex can be further divided into four lobes by the major sulci and gyri, namely: the frontal, temporal, parietal and occipital lobes (as shown in Figure 2.1) [11].

The central sulcus delimits the end of the frontal lobe and immediately ventral to it, the motor cortex is located (see Figure 2.1) [11]. Whilst the lateral sulcus separates the frontal from the temporal lobe. The frontal cortex has been linked to action, while much of the rest has been linked to sensory processing. The prefrontal cortex is the largest in humans [11] and it develops the latest at the age of 25 years on average. It has been deemed responsible for executive functions such as judgement, controlling short-sighted behaviour rendering it the part responsible for long-term goal orientation, decision making and problem solving. Bidirectional communication between sensory regions and the prefrontal cortex is the driving force of action enactment [11]. The assignment of function to sub-regions still eludes us, a reasonable assumption is that the functions emerge based on the communication of distributed networks. Nevertheless, single-case studies and neuroimaging confirm that damage to this area results in impairment of executive functions.

The parietal cortex is responsible for integrating sensory information including, but not limited to, proprioception, the cortical homunculus, and skin-related sensory inputs such as temperature or pain. The somatosensory cortex is a part of the parietal cortex and is located dorsally to the central sulcus (see Figure 2.1) [11]. The somatosensory cortex contains the cortical homunculus which is a distorted mapping of the human body based on the sense of touch. Highly sensitive areas such as the tip of the fingers are extensively represented, in contrast areas such as the back occupy only a small volume. The parietal lobe is also involved in visuospatial processing, object manipulation and the processing of information required for motor control.

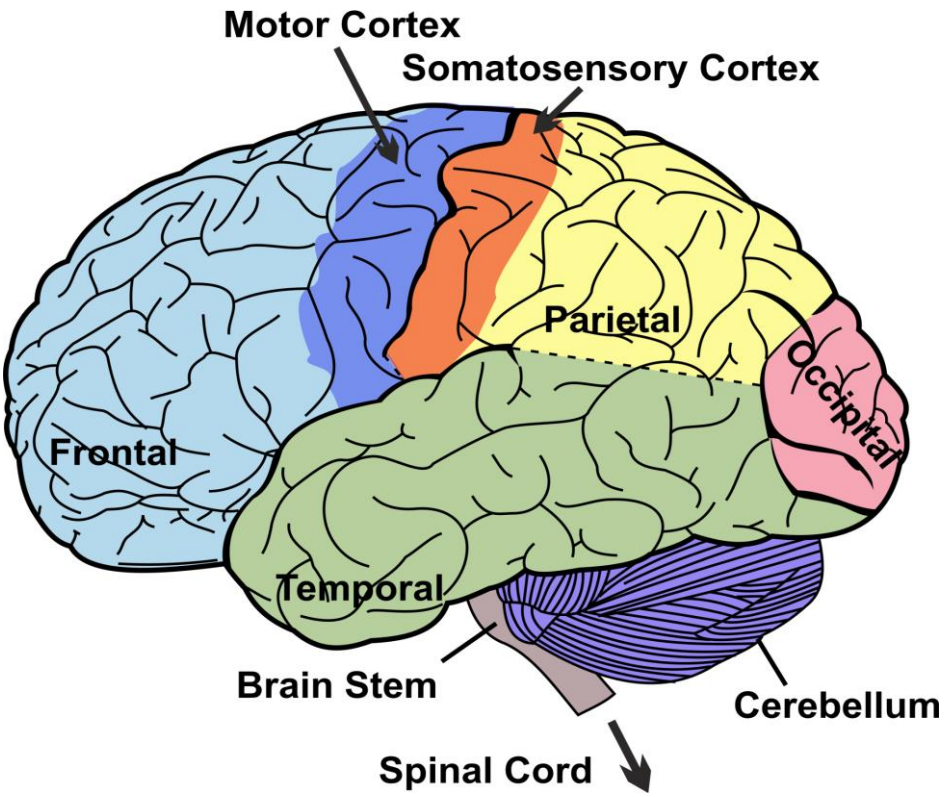

Figure 2.1 - The human brain from a lateral perspective presenting the coarse grouping into lobes. The image was taken from Wikimedia Commons (public domain) and adapted.

The temporal lobe has been found to be involved in memory due to its connections to the hippocampus, language comprehension and processing of sensory inputs. The primary auditory cortex is located in the temporal lobe and as such is responsible for the processing of auditory stimuli. Neuroimaging techniques have shown that the temporal lobe activation in tasks related to language comprehension [13], even signed language, and verbal memory.

The occipital lobe is located at the back of the head and is the main site of visual processing in mammals. The visual cortex is located in this lobe and contains the primary visual cortex, also called V1, which is one of the most studied areas of the brain. V1 processes the visual input of the retina through a very precise retinotopic mapping, although inverted.

2.1.2. Brain complexity

Neurons are the fundamental processing units of the brain and one of the main focuses of Neuroscience. However, in reality, approximately 50% of the cells in the brain are not neurons, they are *glia* [7,14]. It was thought that glia made up 90% of the human brain [7,14], a plausible origin of the myth would be another myth which claims that human beings only use 10% of the brain. This ratio of 10 to 1 of glial cells to neurons seems to be limited to only a low number of brain regions, with other regions, such as the cerebellum, where glial cells are heavily outnumbered by neurons [14].

Take into consideration the typical morphology of cells, they are encased in a membrane that gives them an ovoidal or spherical shape. This is the form of, for example, red or white cells. When looking at a neuron, it is strikingly asymmetric making them decidedly distinctive as cells. Figure 2.2 shows a drawing of neurons made by Santiago Ramon y Cajal, in the early twentieth century, using the Golgi staining technique. Neurons are elongated cells with appendages projecting throughout the encompassing space similar to the branches of a tree, explaining the term “arborized” that is attributed to them [15]. Other neurons in the human body, located most commonly in the spinal cord, have exceptionally long projections exceeding one metre. Evidently, in larger species such as whales, these projections can become even longer.

Humans have the highest ratio of brain size to body size in the animal kingdom [14], although the exact value is dependent on what species is used in the computation. The relative size of brain regions do not determine the number of neurons they contain, as an example the cerebral cortex that weighs 82% of the total brain mass only contains 19% of the neurons [14] while the cerebellum which weighs only 10% of the body mass contains 80% of the neurons. To get a sense of the scale and distribution of neurons in the human brain, there are about 85 billion neurons in the brain, each having between 1000 to 10000 connections to other neurons [16], called synapses. A mention of the scale on which Neuroscience works on, in a single millimetre cube of nervous tissue 100.000 neurons can be found with 0.4 kilometres of dendrites, 4 kilometres worth of axons and 10^9 synapses [17].

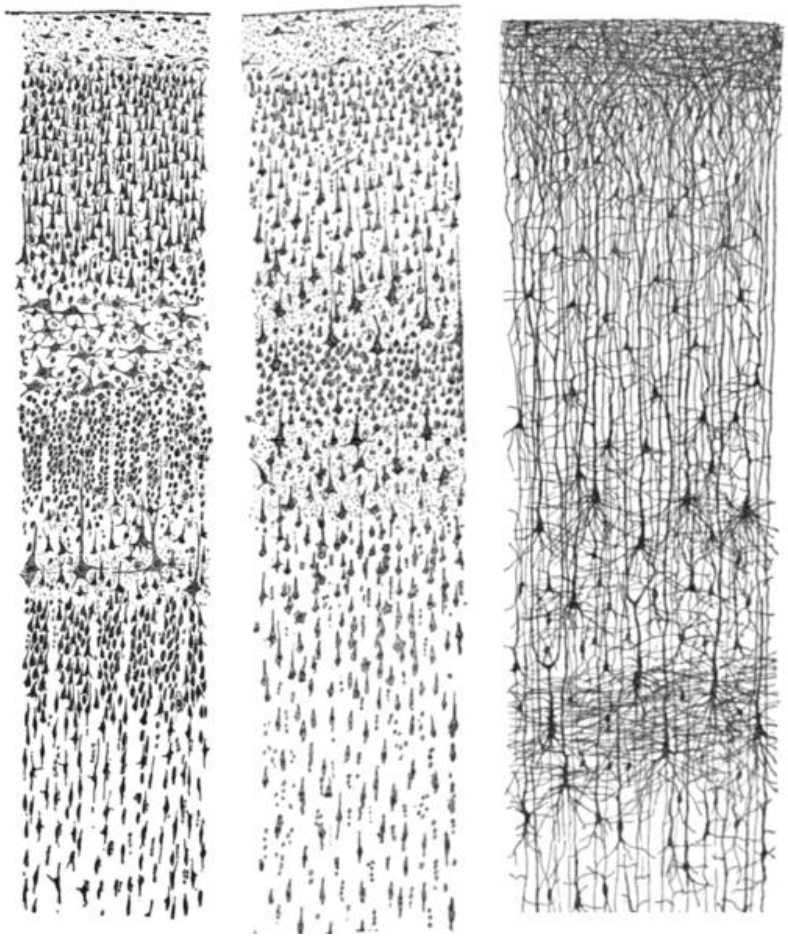

Figure 2.2 - One of Cajal's drawings of neurons using the Golgi staining technique from 1899 [17,18].

Brain complexity at the level of one neuron

A neuron is a complex cell that in general structurally consists of the following main parts: cell body, dendrites, and axon (Figure 2.3). Neurons are electrically excitable cells that communicate through synapses. If we were to make a truly simplified analogy, in the case of neurons, the dendrites would be the ears that are listening to what the previous neurons are saying. Thus, the axon becomes the proverbial mouth in this analogy that is talking to the adjacent neurons.

The output - the axon - is a long wire, but at some point, it starts branching. These ramifications are called axonal endings or terminals. The axonal terminals of previous neurons are connected to the dendrites of the following neuron, in this way when neurons are activated, they send information, and the adjacent neurons are informed through the dendrites that the previous neurons have been activated. Thus, communication is achieved through the connection of the axon terminals of a neuron with the dendrites of another neuron. The point of connection is called a synapse.

Information is transmitted from a neuron to another at the synapse, usually as electrical impulses that are carried down the axon of the first neuron. Upon reaching the synapse, the connection point of an axon terminal of a presynaptic neuron with the dendrite of a postsynaptic neuron, this electrical impulse is converted into a chemical signal, called a neurotransmitter. This chemical signal is then converted by the postsynaptic neuron back into an electrical impulse.

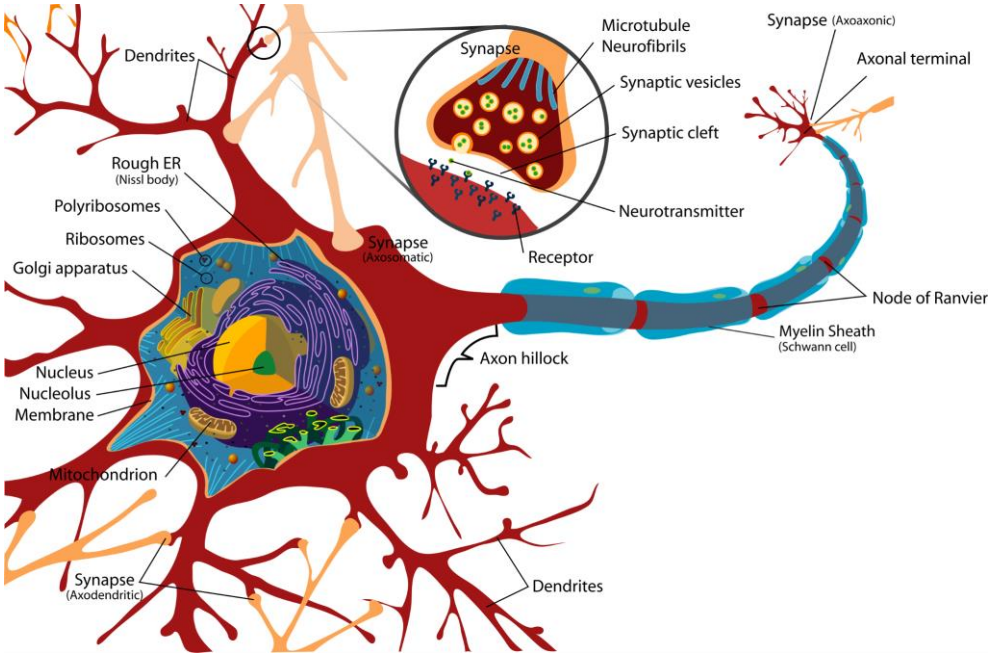

Figure 2.3 - The neuron and its main parts. The cell body, or soma, contains the organelles of the cells, while the axon and dendrites connect the neuron, by forming synapses, to other neurons. The image was taken from Wikimedia Commons (public domain) and adapted.

Thus, the view of a neuron can be simplified as an input-output system, where the dendrites are inputs, and the axon is the output. This simplification was what inspired the creation of the perceptron. However, the brain is a complex system and there are exceptions to this view. For example, in some cases retrograde signalling appears, where the information is sent backward [19] along the axon. Neurons can also communicate through ephaptic interactions, where the electrical fields generated by a neuron can influence other neurons. Moreover, gap junctions allow for direct electrical communication between neurons.

In simple terms, an action potential is required to induce the transmission of information between neurons. In general, the action potentials of a single neuron are similar with respect to their duration, amplitude and shape. These being used as a way of encoding the transmitted information. In the following pages, a more detailed view of the inner workings of the action potential is presented, a first view of the action potential is presented in Figure 2.4.

The action potential

Due to their excitable membrane, neurons generate action potentials at the membrane level. An excitable cell is at rest when it does not generate action potentials. When at rest, a neuron has a negative electrical charge, called resting membrane potential. An action potential is reversal of charge, for a short amount of time the charge of a neuron's membrane becomes positive with respect to the outside. This reversal of charge is produced by the ion channels, these allow for transfer of ions between the inside and the outside of the cell. There are multiple types of ion channels, for Na+, Ca2+, K+ and Cl- and each of these is mostly selective for only their type of ion. As mentioned before, an action potential is a change of charge in the membrane, but this change does not occur linearly, and this is due to the gates of the ion channels that allow the channels to be opened or closed [17].

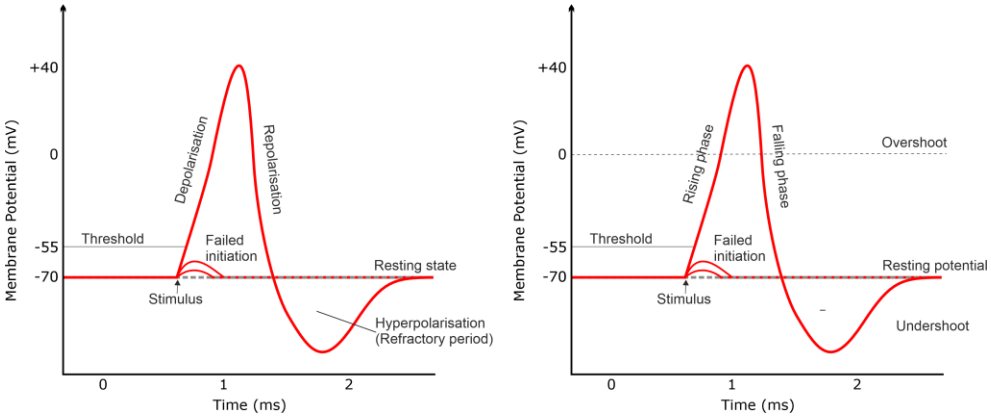

Figure 2.4 - The action potential and its structure. The time course of the membrane potential during the spiking of a neuron and various terminology used that determine its parts. The image was taken and adapted from Wikimedia Commons (public domain). The left and right panels present synonyms that are used within this domain.

The resting potential of neurons, which is an unconditional necessity for functioning, is at -65mV. At rest, the neuron is highly permeable to K+ and thus, it is a very close value to the potassium equilibrium potential of -80mV. But due to some permeability to Na+, this charge is slightly higher. This ratio of permeability can be computed using the Goldman equation and it results in a 40 to 1 permeability in favour of K+ [17]. An action potential can be measured with an electrode implanted in the neuron. The measurement is the difference in potentials between the potential of the cell interior recorded through the electrode and the potential outside the cell (usually connected to the ground). The potential difference of -65mV is displayed when the membrane is at rest. Neurons contain more potassium ions than sodium as the membrane is more permeable to potassium. Any change in the neuron's ion concentration results in a change of membrane potential.

The recording of the potential is made through a comparison between the inside of the cell and the outside. Translating these facts into real world applications, it requires expensive hardware that allows for high precision due to the small size of neurons. Action potentials can be measured due to current loops closing through the outside of the cell as well [20]. Because during a spike the positive ions flow into the cell, on the outside the potential is decreasing. This reversal between intracellular and extracellular recordings of the action potential can be viewed in Figure 2.5. Taking the measurements from outside of the cell is cheaper and easier to do, but action potentials of multiple neurons in the vicinity of the electrode are recorded which can be both an advantage and a disadvantage. Furthermore, cells are not injured resulting in longer and more stable recordings.

An example of an action potential and its defining components is presented in Figure 2.4. An action potential, with all of its parts, is completed in about 2 milliseconds and due to their shape, action potentials are also called spikes [17]. Certain parts of action have been identified, once the depolarization of the membrane starts, the first part called rising phase appears. This only happens once the depolarization surpasses a threshold; thus, action potentials occur only if the depolarization passes a critical level. Depolarization is produced by the Na+ channels being opened as a reaction to the neurotransmitters of other neurons [17]. Once the membrane potential is depolarized sufficiently, a cascade of Na+ ion channels start opening that produce the rising phase.

Through the rising phase, the potential increases until it becomes positive with respect to the outside up to a peak of about 40mV, called an overshoot. This happens due to an increased Na+ permeability of the membrane, which has the equilibrium potential at 62mV. After the peak is reached, the repolarization, called the falling phase, starts. During repolarization, the Na+ channels are closing, the membrane remains only permeable to K+ which exit the cell, restoring the negative polarisation of the cell's interior. Through the falling phase, the potential briefly undershoots below the resting potential, but it is gradually restored to equilibrium. This restoration period is called the refractory period because the sodium channels are closed, and another action potential cannot or is difficult to be produced. Once a sufficiently negative potential is reached the sodium channels open again.

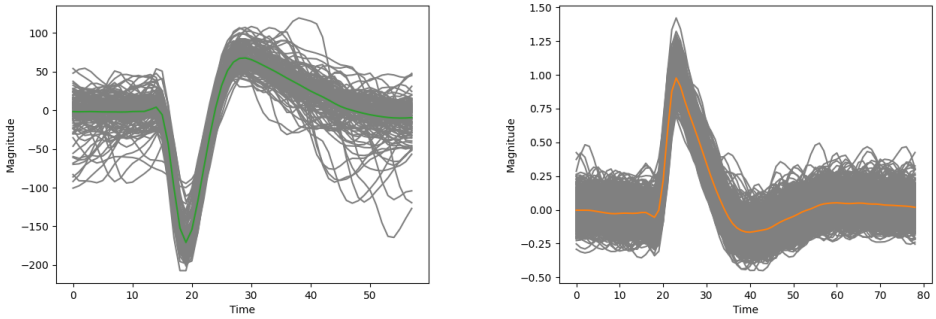

Figure 2.5 - Particularities of action potentials on different types of recordings (left, extracellular from TINS and right, intracellular from [21]).

Electrical excitation is needed in order to communicate. An influx of ions generates electrical excitation that goes through the neuron, from its dendrites to the axon terminals. The excitation of a neuron through a single dendrite allows the flow of ions from the outside to the inside of the neuron, and vice versa. Vast excitation influx is required from the presynaptic neurons, for the membrane potential of the neuron in question to become positive (depolarization). This is the contrasting dichotomy of neuronal signalling, for a neuron to "speak", its inside needs to become positively charged, whilst when the inside is negatively charged it remains silent. For a neuron to generate an action potential, a lot of energy is consumed. The neuron has to use this energy to coordinate pumps and channels in order to generate these action potentials and to maintain the difference of potential between the interior and exterior at rest. It uses the sodium-potassium channels in the membrane to allow more positive ions to leave the cell than to enter it in ratio of 3 to 2 which reduces the membrane potential to reach the resting state. When excitatory signals are received, the pumps are stopped, and channels open allowing positive ions to penetrate the membrane and increase the charge of the neuron's membrane. After this excitation passes, the channels close and the pumps start reestablishing the resting state. All of these actions require energy, and neurons spend half of their energy on the pumps [15]. These facts make it easier to understand how an organ that weighs 2% of body weight manages to consume more than 20% of the energy budget.

Looking back on the drawing of Cajal, presented in Figure 2.2, each neuron has dendritic arborization, each of those wires ending in, what are called, dendritic spines. When a neuron is in its resting state, that is to say that it is negatively charged, if it receives excitation through only a single dendrite it generates relatively little

depolarization in the neuron. Excitation from one dendrite is often not enough to generate an action potential, it only makes the soma of the neuron slightly less negatively charged. The influence of the excitation dissipates the farther we go from the excitation point. Thus, vast amounts of excitation are needed for the change of charge to affect the axon. Repeated stimulation is needed or, most typically, the simultaneous excitation of multiple dendrites, which implies synchronous activation of presynaptic neurons [15].

This picture becomes even more complicated with the introduction of the axon hillock. The axon hillock connects the cell body and the axon, it is a highly specialised component that generates the spike, which then propagates along the axon. More precisely, the excitation arriving at the dendrites travels towards the hillock and when the hillock depolarizes to approximately -40mV the action potential is triggered and transmitted through the axon. After reaching the absolute peak, the depolarization process is followed by repolarization in which voltage is decreased by an efflux of K+ ions. The repolarization is often followed by hyperpolarization, or refractory period, in which the voltage goes below the resting voltage due to a continued efflux of K+ ions or an influx of Cl- ions. The hyperpolarization lasts for about 2 milliseconds, a time during which the neuron is unable to generate another action potential. Finally, Na+ and K+ pumps restore equilibrium by bringing the membrane back to its resting potential of -70 mV.

The transfer of the action potential to the axon involves the same ion channels, thus the travelling of the spike along the axon is more of a regeneration process. Thus, we can treat the spreading of excitation from the dendrites to the cell body as a continuous phenomenon which may or may not produce an action potential at the axon hillock. However, once an action potential is generated, it is usually treated as an all-or-nothing event, an activation that propagates throughout the axon as a discrete phenomenon.

Brain complexity at the level of one neuron

It was established that neurons communicate at the level of synapse and that somehow an action potential once it reaches the axon can generate an action potential in another neuron under some conditions. Synapses are microscopic gaps between the dendritic spines and the axon terminals. As mentioned previously, these signals are converted. Electrical excitation does not get transferred between one neuron to the next, it is converted into a chemical signal and only that is transferred to the following neuron. Notably, a small percentage of synapses do not signal chemically, but rather electrically, the retinal gap junctions are an example [22].

These chemical signals are called neurotransmitters. Spherical objects, called vesicles, can be found in the axon terminal and these contain multiple copies of neurotransmitters. The action potential travels down the axon, setting off the release of neurotransmitters into the synapse [17]. To be specific, the action potential opens calcium channels, and it is the influx of calcium that actually drives the vesicles and allows them to release the neurotransmitters into the synaptic cleft.

Up to this point, neurons have been referred to as ‘previous’ or ‘next’, within this domain, they are denominated in relation to the synapse. As such, the neuron that has an action potential sending neurotransmitters through the synapse is called presynaptic neuron. While the neuron that receives these neurotransmitters is called a postsynaptic neuron. Each neurotransmitter has a unique shape, naturally each copy has the exact same shape. The postsynaptic neuron receives these neurotransmitters through

receptors, each of these having a unique shape in order to receive a certain neurotransmitter. The analogy used in the field is that of a lock and key, only the corresponding neurotransmitter can 'open' its affiliated receptor [15]. Once the receptor is 'unlocked' by the neurotransmitter, it causes the opening of channels in the dendritic spine which starts the excitation by allowing ions to enter the dendrite of the postsynaptic neuron. This process is shown in Figure 2.6. Thus, one neuron can generate an action potential in a connected neuron.

At this point, another complication appears. These neurotransmitters cannot remain linked to the receptors indefinitely, or it would block the generation of new action potentials or hold the cell in permanent excitation. There are two ways in which the brain solves this problem, reuptake, or degradation. Reuptake is done by the axon terminal with specialised pumps that salvage the neurotransmitter and package them back into vesicles for later use. Degradation is done in the synapse by a specialised enzyme and the residual material being expelled from the cerebrospinal fluid into the bloodstream.

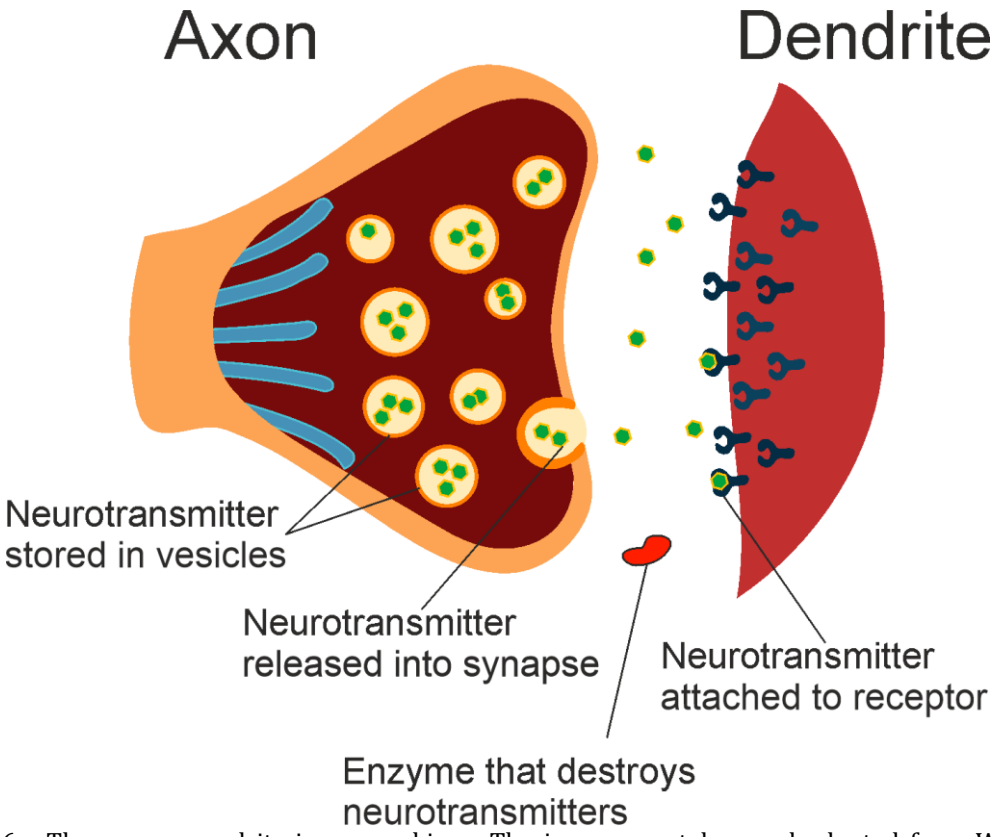

Figure 2.6 - The synapse and its inner workings. The image was taken and adapted from Wikimedia Commons (public domain).

As mentioned previously, the excitability of a neuron can change over time. This can happen in multiple ways in the synapse, by increasing the probability of generating an action potential in the postsynaptic neuron. Firstly, the amount of neurotransmitter released by the presynaptic neuron gets increased. Secondly, the number of receptors in the postsynaptic neuron increases. A more interesting way in which this can be done is to lower the reuptake activity. This will result in lower amounts of neurotransmitters being extracted from the synapse. Thus, increasing the duration in which they influence the synapse and amplifying excitation. Evidently, by lowering the levels of degradation

the same result can be obtained. Even though it is yet little understood, a theory that has been around since the 1950s is that memory is located in the synapse [23]. This has been adapted over time, as different parts have been found to influence the storing of memory. But the idea remains, the synapse and its mechanisms persist as a point of interest for the neuroscientist [24].

It was alluded to above, there are multiple types of neurotransmitters. And to make it even more perplexing or beautiful, depending on the perspective, multiple types of receptors. Some neurotransmitters are excitatory, some inhibitory and the function of some depends on the receptor it binds to. As such, a neuron with 10 dendritic spines may receive excitatory neurotransmitters from 5 presynaptic neurons and 5 inhibitory ones. This makes for an exquisitely complex system where information can be coded in many unique ways.

Brain complexity at the level of multiple neurons

In order to simplify, we have been referring to the synapse as the connection of the dendrite of one neuron to the axon of another. In reality, only axo-dendritic synapses have been discussed. In fact, synapses present a variety of different arrangements, axo-axonic, dendro-dendritic and many others. This section presents concepts of neuron circuitry, while still remaining at a micro level. These wirings are simple when compared to the level of complexity found when looking at more vast regions that regulate behaviour.

A case of neuromodulation through different types of synapses can be seen in Figure 2.7. Neurons 1 and 2 are connected in a classical axo-dendritic synapse, Neuron 1 releases an excitatory neurotransmitter. In comes Neuron3 with an axo-axonic synapse to Neuron1's axon releasing an inhibitory neurotransmitter, called GABA, that attaches to Neuron1's receptors. This type of link will stop the action potentials of Neuron1 to release neurotransmitters that would influence Neuron2, thus keeping Neuron2 from spiking, this is an example of neuromodulation.

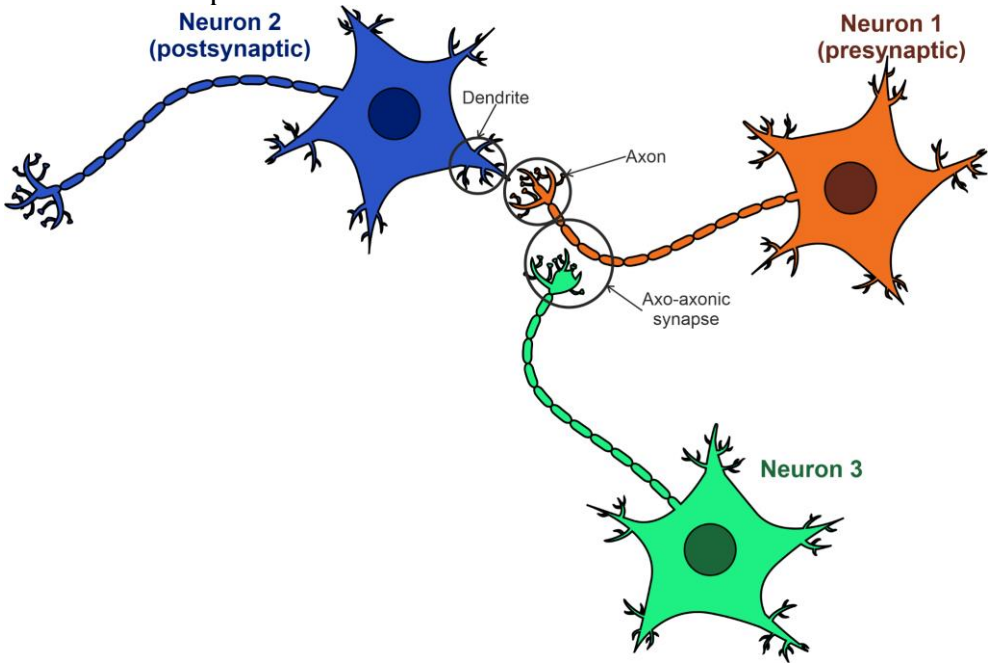

Figure 2.7 - Neuron circuitry. The image was taken from Wikimedia Commons (public domain) and adapted.

These circuits can also create loops as is presented inFigure 2.8. Neuron1 forms an axo-dendritic synapse with Neuron2 and another one with Neuron3, both excitatory. Nevertheless, Neuron3 forms another synapse in a feedback loop with Neuron1 that is inhibitory. In the hypothetical case that Neuron1 has several action potentials, because of the two projections, it will excite both neurons 2 and 3. Once Neuron3 has an action potential it will inhibit the firing of Neuron1, thus can Neuron1 influence itself. This process is called the sharpening of a signal [15]. Additionally, Neuron1 can even control the strength of the feedback loop by the number of axonal projections that it binds to Neuron3.

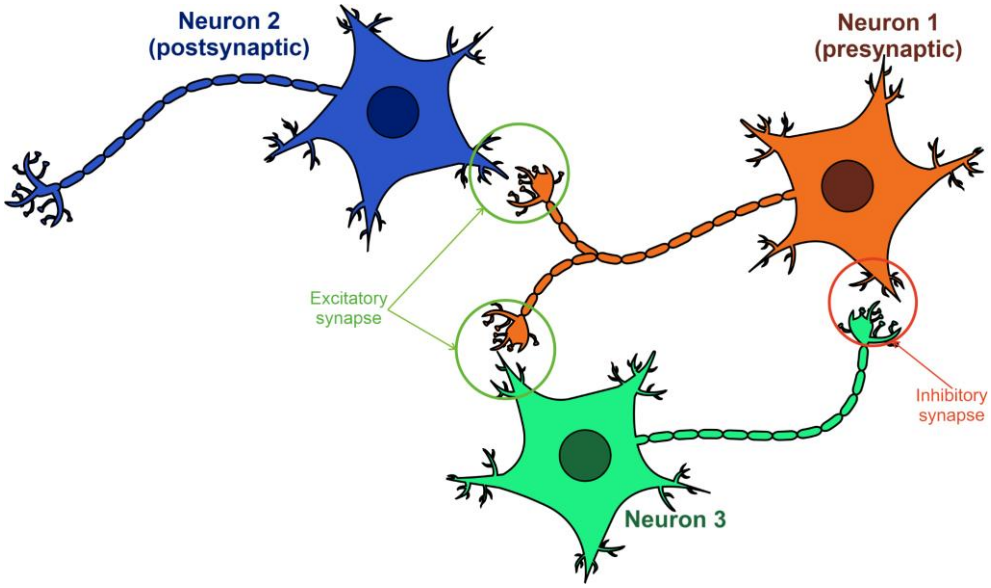

Figure 2.8 - Looping in neuron circuitry. The image was taken from Wikimedia Commons (public domain) and adapted.

Brain complexity at the level of brain regions

When looking at cells in different organs of the body under a microscope, there are rows upon rows of homogeneously distributed cells. Unless an infection occurs that may damage the organ, all cells are arranged correspondingly, and it becomes monotonously similar. This is not the case of the brain, the internal configuration is much more asymmetric, showing extraordinary levels of complexity.

Brain regions are organised by function, thus all neurons in that region have a related role in achieving said activity. Different regions communicate with each other by accompanying intertwined groups of axonal projections, called cables. Thus, resulting in different regions having different functions while also communicating and influencing each other.

A few of these regions, such as the hypothalamus and hippocampus, have familiar names and functions. While others, like the superior olivary nucleus, have less so. Nevertheless, all of these regions number neurons in the millions.

Unquestionably, the information presented here only begins to scratch the surface of the true intricacy of how the brain functions and generates behaviour. Its purpose is to introduce the reader to the complexities that can be found even at the microscopic level, thus allowing to deduce the actual elaborateness of high-level processes in the brain. These are the difficulties with which a neuroscientist has to deal with.

2.1.3. Brain oscillations

The neuronal rhythmical activity has been denominated as brain oscillations. Brain oscillations offer another avenue to further our understanding of the brain as they can be found in many brain areas and have been found as essential to high-level functions such as perception, cognition and behaviour. Through the detection of oscillations during experiments, insights can be gained about how information is processed in the brain. Furthermore, anomalies in brain oscillations have been linked to diseases, increasing the need for precise detection of brain oscillations.

The first type of oscillation, the alpha, was discovered and described by Hans Berger shortly after the invention of the EEG [25] as it was the single oscillation strong enough to be visible to the naked eye. Oscillations were used as a marker for diseases that altered normal brain activity. Later studies have shown that brain oscillations are correlated to brain states and have been involved in clinical and research applications [26]. Curiously, these rhythms are similar across mammalians in spite of the disparity between brain mass and complexity [17].

These oscillations have been demonstrated to be ubiquitous in the brain [27] and have been found at various spatial scales. They can be found in the sub-threshold dynamics of neurons, in the rhythmic discharges of individual neurons, in the coordinated activity of larger neural circuits or even cortical areas large enough to produce rhythmic activity detectable in the EEG. Brain oscillations are intimately connected to cognitive processes. Nevertheless, it has not yet been determined whether the oscillatory activity of the brain is a fundamental functionality of the brain or just a side effect of an underlying phenomenon [28].

Oscillation types

These oscillations have been discretized into bands by using frequency as a criterion, but have been associated with various aspects of information processing and transmission in the central nervous system:

- Infra-slow activity (0.02-0.1Hz): these types of oscillations remain a debate, it is speculated that they could be generated by non-neuronal activity and that they may synchronise and modulate faster oscillations [29].
- Slow activity (0.1-15Hz): these types of oscillations occur predominantly during slow-wave sleep and under specific anaesthetics [29]. These also include the slow oscillations, the delta, theta and alpha band.
    - Slow oscillations (0.1-1Hz): in EEG recordings, it appears that cortical activity vacillates between an excitatory and an inhibitory state during slow oscillations [29].
    - Delta band (1-4Hz): it is suggested that delta oscillations may emerge from two sources, the neocortex and the thalamus [29]. While thalamic delta activities are known to appear during deep sleep [17] through neuron hyperpolarization effects, the neocortical delta oscillations are hypoworked to be produced by bursting neurons [29].
    - Theta band (4-8Hz): appear in the limbic system and most commonly in the hippocampus [13], they can appear in both awake and asleep states

[17]. Although coherent overall, inter-region fluctuations are present. As they occur most frequently in the hippocampus, it has been suggested that they are involved in memory-related mechanisms, such as memory formation and recall [13]. At a synaptic level, the process of long-term potentiation (LTP) by the theta activity is present during a task [13].

    - Alpha band (8-12Hz): these types of oscillations were the first to be discovered. They are prevalent in the thalamus [26], have also been found in the hippocampus and the reticular formation [13], but are largest in the occipital cortex. Alpha rhythms have been linked to quiet waking states [17] and increased attention, more specifically when disregarding unnecessary information [30]. The synchronisation and coordination of excitatory and inhibitory interplay is the source of the alpha oscillatory activity [13]. The alpha oscillations have also been associated with the coordination of gamma activity [13].

- Fast activity (20-100Hz): includes the activity that falls under the umbrella of beta and gamma oscillations. These fast rhythmic activities are the most common oscillation types in the waking state [29]. They appear synchronised across adjoining cortical areas and are present during specific anaesthetics, slow-wave and REM sleep [29].
    - Beta band (15-30Hz): these oscillation types have been found in every area of the cortex and their origination has been associated with neurotransmitter systems [13]. Beta activity can be found in the awake state during various cognitive undertakings, for instance, learning, voluntary motor movements, novelty identification and reward assessment [13]. Studies indicate that they may occur when a noteworthy stimulus is present and that they regulate the coordination of neuronal activity [13].
    - Gamma band (30-80Hz/200Hz): the upper bound of the gamma band is still under debate and it is further classified into low and high gamma [13]. Similar to the beta activity, gamma oscillations are reliant upon the neurotransmitter system. Gamma oscillatory activity has been linked to various cognitive processes, such as consciousness, memory, sensory perception, motor coordination, prediction, attention and language processing [13,17,29]. Increased gamma synchronisation has been also linked to perceptual integration, such as the binding of parts to form a whole [31]. It has also been found that gamma oscillations are produced by the synchronised excitatory and inhibitory activity of local circuits [32].

Each oscillation band has been found to be linked to different cognitive brain functions, yet different brain regions can generate oscillations at different frequencies [22]. A direct relationship between oscillation, brain region and cognitive function is unlikely to be found. A plausible reason is that the amount of oscillatory activity is insufficient to fully comprehend the complex mechanisms of the brain [22]. Nevertheless, they can serve as another method by which our understanding of the brain increases.

Oscillations in the cortex

By and large, high-frequency activity is linked to states of wakeful alertness and the dreaming phase of sleep. Whilst, low-frequency activity is linked to non-dreaming phases of sleep, anaesthetised or coma states [17]. The logical hypowork is that information processing is such a complex state that the activity level is high but uncoordinated, that is to say that small groups of neurons would work together on connected yet different tasks in order to achieve overall cognition [17]. The overall synchronisation is low, such that localised oscillations, mostly in the beta and gamma bands, become more prevalent. In comparison, in states of deep sleep where lower frequency oscillations dominate, many neurons are excited by the same process and they synchronise as no conscious information processing is happening [17].

Oscillatory activity is omnipresent in the brain at various frequency ranges. It has been suggested that they offer a temporal frame upon which information encoding would be possible and even, propitious [13]. Although present and measurable, many concepts remain unclear. It has not yet been established whether an underlying process exists that produces the oscillatory activity as a side effect. The classification of oscillatory behaviour into bands can be slightly different depending on the studied species but they respect a logarithmic distribution. The beta and gamma bands have overlapping functionalities and may be produced by a common mechanism, raising the question if they should be considered as rigorously independent of each other, as a collective or as a continuum [13]. Moreover, the opposite is not implausible either, a possibility is that different mechanisms produce such bands and that a discretization into sub-bands is required [13]. Furthermore, several diseases have been linked to improper oscillatory activity. The question of whether they are the cause or just an effect still remains unanswered.

2.1.4 Brain recording techniques

Different recording techniques provide different scales (temporal and spatial) of observation of nervous activity. Brain recording techniques can be categorised based on the type of signals they record: electrical, magnetic, or hemodynamic. However, they can also be categorised depending on whether they are invasive or not. Invasive methods require insertion of a probe into neural tissue, capturing signals with high precision, while non-invasive methods allow for the study of brain activity without this requirement. Electrical signals can be obtained through the use of microelectrode recordings (MER) or electroencephalography (EEG). MER is an invasive technique that can provide recordings of the activity of a single neuron or a small population of neurons. EEG measures the electrical activity of a large population of neurons through the use of a cap placed on the scalp. Magnetoencephalography (MEG), another non-invasive technique, captures the brain's magnetic fields generated by neural currents. Transcranial Magnetic Stimulation (TMS) is a non-invasive technique that utilises magnetic fields and that can modulate neural activity which has shown potential for therapeutic applications. Functional magnetic resonance imaging (fMRI) can be categorised as hemodynamic as it measures oxygen consumption changes in the blood flow associated with neural activity. A modified fMRI machine can be used together with EEG for simultaneous recordings, alongside behaviour indicators such as with reaction

time, error rate, and others, that can help in the formation of theories about the task and brain activation resulting in a finer function-to-structure mapping [33]. Table 2.1 offers a short description of these brain recording techniques and their various features.

Table 2.1 – A descriptive summary of various brain recording techniques.

Method	Type	Description	Applications	Resolution (Temporal/ Spatial)
MER	Invasive	Captures intricate recordings of the activity of single or small populations of neurons.	Neuroscience research	Very High / Low
EEG	Non-invasive	Measures the electrical activity of a large population of neurons through electrodes placed on the scalp.	Clinical diagnosis, cognitive neuroscience research	High / Low
MEG	Non-invasive	Captures the brain's magnetic fields generated by neural currents.	Cognitive neuroscience, presurgical mapping	Very High / Moderate
TMS	Non-invasive	Utilizes magnetic fields to modulate neural activity.	Research on brain function, therapeutic applications	Moderate / Moderate
fMRI	Non-invasive	Measures oxygen consumption changes in blood flow associated with neural activity.	Cognitive neuroscience, clinical diagnosis	Low/ High

This work focuses on two of the most common brain recording techniques: 'in vivo' microelectrode recordings, and EEG. The microelectrode recording technique allows the recording of even single cells, while EEG allows for the recording of global activity. The global activity of the brain has been correlated with certain behaviours [34] and diseases [35]. Generally, invasive methods are more difficult to perform because of the trauma and the microscopic precision needed and they can only record very small regions, but they offer a cleaner signal and have a better temporal resolution.

'*in vivo*' microelectrode recordings

Microelectrode recordings are an intracranial recording technique requiring direct access to the brain. Such experiments are more difficult to perform due to the required hardware and the level of precision and was most commonly performed on anaesthetised animals because of their invasiveness. It can be performed on humans during operations on the brain, although it happens rarely. The new trend of electrophysiology is to insert chronic implants for a long-term analysis of behaviour.

These types of experiments can be further segregated into two subtypes, namely intracellular and extracellular recordings. Intracellular recordings induce a higher degree of complexity as the recording electrode must be inserted in the cell, which limits the number of neurons that can be recorded and increases the risk of death of the recorded neuron [36]. Although harder to perform, there are advantages, intracellular recordings can provide the inside, sub-threshold view that extracellular recordings lack.

Extracellular recordings imply the insertion of the recording electrode between neurons, thus recording the activity of adjacent neurons allows researchers to process and hopefully understand how neuron populations interact. When measuring potential with an electrode placed in the neuronal field, a reference electrode must be placed in contact with tissue at an area sufficiently far apart so as not to be influenced by the activity within the recording volume. Nevertheless, activity from multiple neighbouring spiking cells, from circuits farther away (orders of mm) and noise are also picked up by the electrodes. Thus, the neuronal activity has to be separated from background noise and individual spiking neurons need to be separated from each other.

Due to the high density of neurons in the brain, neuronal activity can create fields that sum up. Such superpositions of action potentials can appear in the recording in the form of low frequency ripples of up to 300 Hz called Local Field Potentials (LFP) [37] as shown in Figure 2.9. LFP activity can be isolated using a low-pass filter with a cut-off frequency of 300Hz. LFPs comprise the activity of both excitatory and inhibitory neurons and are brought about by a superposition of electric fields produced by a multitude of processes including synaptic activations, oscillations, spikes and afterpotentials that propagate theoretically in a volume of up to several mm^3 [38].

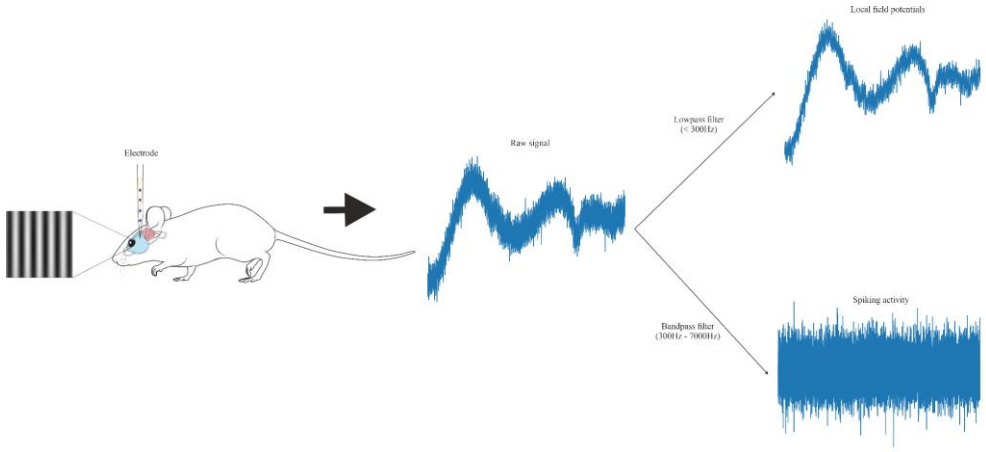

Figure 2.9 - Raw signal filtering resulting in Local Field Potentials and/or Spiking Activity.

The spiking activity, also called multi-unit activity, can be isolated through the use of a band-pass filter with 300-7000Hz cut-off frequencies as shown in Figure 2.9. Activity of individual cells have to be disentangled based on certain characteristics in order to investigate the interactions between individual neurons. This is where Spike Sorting algorithms are used. Figure 2.5 shows the differences between intracellular (Figure 2.5 right) and extracellular (Figure 2.5 left) recordings.

As always, the picture painted here is not as simple in reality. There are many factors that can affect the quality of the recording. Moreover, the distance of the neuron to the recording electrode affects the amplitude of the recorded signal, while the

orientation and the morphology of the dendritic tree can also affect the shape of the action potential [17]. Neurons within 0-50μm can be identified as single units, that is to say their activity is easily separable from the activity of other neurons. While neurons within 50-140μm can only be identified as multi-unit as they can hardly be identified by using their amplitude, that is to say that their activity is likely to be merged with the activity of other neurons in that range. At distances larger than 140μm neuronal activity becomes drowned by the background noise and cannot be separated at all [39].

Despite their name, Local Field Potentials (LFP) [40] are not only local and can be present even when spiking is not present in the recorded area, in which case it indicates that their source may be further away and that LFP amplitude is not always correlated to the local activity. Furthermore, electrodes do not present spatial sensitivity as the electrode records any field potentials that arrive in their neighbourhood. Electrical properties and source geometry are factors that indicate whether a field potential is recorded at a distance. LFPs are residual currents of neuronal population activity [40] that present certain correlations to engaging neurons. The relationship between individual neuronal activity and LFP remains a topic of debate due to its inherent complexity; it has been found that the activity of a subset of neurons is aligned to LFPs while others have no correlation. The characteristics of LFPs are controlled by structural factors at both micro and meso-scale, while temporal patterns are configured as the mix of the temporal dynamics of source neurons.

Multitrodes

The separation of single-unit clusters in extracellular recordings is made on the assumptions that different neurons produce unique amplitudes and waveforms on the recording electrodes. Under this assumption, using these characteristics, partitioning of spikes into groups produced by the same neurons can be achieved.

Multi-channel electrodes are able to capture the action potentials of multiple neurons at different distances on each channel. Therefore, a spike recorded with such a multitrode can have a different amplitude on each recorded channel, depending on the distance and the shape of the neuron producing the spike. In a manner similar to triangulation in 2D space, due to the amplitude differences between channels for a single spike, the identification of source neurons is easier to make [41,42] because ambiguities can be resolved easier using information from multiple channels.

This is the case especially for neurons firing at the same time on single-channel electrodes; the waveform of superimposing spikes is difficult to disentangle. Usually, these overlaps are approached in single-channel recordings through template matching [43]. While for a multi-channel electrode this overlap may happen on only a subset of the channels and the rest may allow the separation of these simultaneous firings [41].

In the case of highly active neurons, this increased spatial resolution of multitrodes may improve performance by reducing the number of overlapping spikes [43,44]. This spatial resolution appears from the correlation between the amplitude of a spike and the distance between the electrode tip and the firing neuron [41].

Knowing these, it might be expected that a further increase of the number of channels should also result in an increased ability to identify the source neurons. One study found that, in recordings that contain a lot of noise, heptodes (seven channels) are better than tetrodes, while in less demanding situations they perform equally. PCA was used in order to reduce the dimensionality and it was determined that the number of

principal components has an impact on the spike sorting performance, with a higher number of principal components increasing the performance. These analyses were made through the use of a signal simulation algorithm [44].

Another study evaluated the impact of noise and number of neurons on the performance of spike sorting with regard to the number of channels available. It has been shown that tetrodes and heptodes outperform single-channel electrodes as the number of neurons or the standard deviation of noise increases. Additionally, heptodes have a slightly higher performance for spike sorting than tetrodes [41,43].

The multi-channel approach, although it solves certain difficulties, brings other challenges: very **large amounts of data** and **high-dimensionality** [2]. This is certainly the case for the new types of electrodes, like Neuropixels, that allow the recording of hundreds to thousands of neurons [1]. These new developments bring the question of how to process all the information as current methods have been underperforming for higher numbers of neurons in a recording. One study shows that out of 20 neurons, current methods are only able to identify 8-10 [45]**.**

Data acquisition

Adult mice of the C57/BL6J strain were used for the recording of in vivo electrophysiological data. The mice were anaesthetised using isoflurane in oxygen (5% for induction, 2 - 2.5% for surgery, 1-3% for maintenance) and then mosunted in a stereotaxic frame (Stoelting Co, Illinois, United States). The heart rate, respiration rate, core body temperature, and pedal reflex were constantly monitored. The body temperature of the animal was maintained at 37°C using a feedback-controlling heating pad with a rectal probe (Harvard Apparatus). The animal's head was shaved and prepared with povidone-iodine and a local anaesthetic (Xylocaine). Then, a midline incision was made, and a circular 2 mm craniotomy was carried out on the left hemisphere. The craniotomy was positioned at stereotaxic coordinates that corresponded to the visual cortex, specifically 0.5-1 mm anterior to lambda and 2-2.5 mm lateral from the midline.

Electrophysiological data was collected using A32-tet probes (NeuroNexus Technologies, Inc) at a sampling rate of 32 kSamples /s (Multi Channel Systems MCS GmbH) while the mice performed a visual perception task involving moving stimuli. The stimuli used were full-field drifting gratings presented monocularly on a Beetronics 12VG3 12-inch monitor with a resolution of 1440x900 at 60fps. The gratings had a spatial frequency of 0.11 cycles/deg and a temporal frequency of 1.75 cycles/s. In some of the datasets used their contrast varied between 25% and 100% and the gratings were presented in eight different directions in steps of 45°. Others had a fixed contrast of 100% while the directions were presented in steps of 45° or 30°.

The electrophysiological signals were recorded at a sampling rate of 32kHz (Multi Channel Systems GmbH), and local field potentials (LFPs) were obtained by applying a band-pass filter (bidirectional, 3rd order Butterworth IIR filter) from 0.1 to 300 Hz, followed by downsampling to 1kHz. Line noise artefacts and their harmonics were eliminated by applying a series of notch filters (bidirectional, 3rd order Butterworth IIR filter) at 50, 100, and 150Hz. In contrast, for multiunit activity, the data has been digitally band-pass filtered (bidirectional, 3rd order Butterworth IIR filter) between 300 - 7000 Hz. Afterwards, a threshold is computed as the standard deviation (SD) of the filtered signal multiplied with a coefficient, which was typically set between 3 and 5 times the SD.

All threshold crossings were then identified as spikes and served as input for the feature extraction algorithm.

All animal experiments were carried out respecting directive 86/609/EEC of the European Communities Council from 24 November 1986, directives 2010/63/EU of the European Parliament and 2010/63/EU of the Council from 22 September 2019. These procedures have been approved by the Local Ethics Committee (approval 3/CE/02.11.2018) and National Sanitary and Veterinary Authority (approval ANSVSA 147/04.12.2018) and performed in accordance with the ethical guidelines of the European Communities Council Directive 2010/63/EU Society for Neuroscience, Romanian laws for the protection of animals and Romanian Law 43/2014 on proper conduct in scientific research. Multiple datasets were collected over 4–6h from each animal to minimise the number of animals and animal use.

Electroencephalography (EEG)

Electroencephalography (EEG) is one of the oldest techniques of brain recording and it is often referred to as a "window on the cortical brain activity" [46]. Moreover, it is non-invasive and commonly used in research and clinical applications as the electrodes are placed on the scalp and it is the best option from an ethical and cost-efficiency perspective. In extracranial recordings, the electrical field generated by neuronal activity must travel through the cerebrospinal fluid, skull, head muscles, and skin in order to be able to reach the recording electrodes. Thus, only the coordinated activity of large neuron populations is recorded, providing a poor spatial resolution due to the smearing effect on the potential. It is not the axonal currents of the action potential that are represented in the EEG signal as the myelination of the axons insulates them, but rather the postsynaptic dendritic currents [47]. Postsynaptic dendritic currents form in the dendrites and soma of a postsynaptic neuron as neurotransmitters from the presynaptic neurons bind to the receptors of the postsynaptic neuron. EEG signals have been correlated to different states of behaviour, such as sleep, focus and perception [26], and used to identify diseases, such as schizophrenia [48] or Alzheimer's disease.

EEG electrode positions have been standardised as their positions have a great effect on the analysis and the interpretations of the data. The recording electrodes must have a reference as the signal is measured as the voltage between the electrode and the reference. Additionally, the contact resistance can be lowered through the use of a special electrically conductive gel.

Data acquisition

Healthy human subjects participated in a visual recognition task, during which EEG data was collected using a high-density EEG cap (Biosemi ActiveTwo) that had 128 electrodes and recorded at a sampling rate of 1024 Hz. Visual stimuli were created using the "Dots" method and were displayed on a 22-inch Samsung SyncMaster 226BW LCD monitor with a resolution of 1480 x 11050 @ 120 fps, and a viewing angle of 8.7x5.6, placed 1.12m away from the participant. After the recording, the EEG data was subjected to band-pass filtering between 0.1-200Hz (Butterworth 3rd order) and power line noise was removed using a band stop filter (49.5-50.5 Hz, 4th order Butterworth). To avoid phase distortions, both filters were applied bidirectionally.

The conducted experiment used a set of 210 stimuli composed of dot lattices resembling the contours of 30 known objects with varying levels of deformation. Each trial consisted of three intervals: fixation, stimulation, and response. During the fixation period, the participants were instructed to maintain fixation for a duration of 1500-2000ms, which served as a baseline period before the stimulus was presented on the monitor. Once the stimulus appeared, the participants were allowed to visually explore the scene to reach a perceptual decision on the stimulus identity. Finally, the response interval required the participants to press one of the three buttons that corresponded to their perceptual decision - seen, uncertain, and nothing.

Human experiments were carried out in compliance with Directive (EU) 2016/680 and Romanian Law 190/2018 and were reviewed and approved by the Local Ethics Committee (1/CE/08.01.2018). All participants involved in the study have provided their written informed consent.

2.2. Computer science domain knowledge

2.2.1. Architectures of artificial neural networks

A neuron can be simplified into an input-output system with multiple inputs (dendrites) and a single output (axon). This simplified model was used in the creation of artificial neural networks as its most basic processing unit and is shown in Figure 2.10. They were inspired by the inner workings of the biological brain and thus mimic its functionality and structure.

A neural network consists of a set of interconnected computational units, namely artificial neurons. Similar to its organic counterpart, the role of a neuron is to receive signals, process them and transmit the output signal to the connected neurons. Generally, artificial neurons in an artificial neural network are grouped into layers. In the simplest architectures, the neurons of one layer receive input from the neurons of the preceding layer and send the outputs to the succeeding layer. More complicated network structures with recurrent connections have also been used and studied.

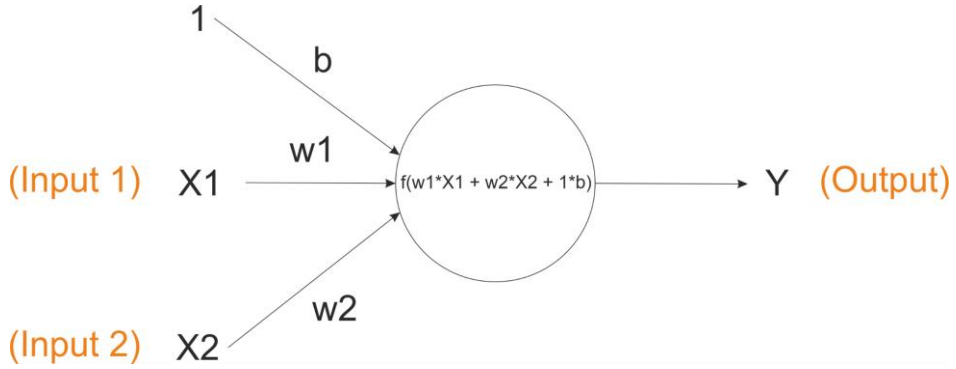

Figure 2.10 - A simple diagram of the perceptron, where *w* indicates the weights, *b* the bias, and *f* the activation function.

Autoencoders

Autoencoders [49,50] are a type of neural network that are capable of learning the intrinsic relationships between data points without the use of ground truth. Autoencoders try to reproduce the input at the output, usually with the same number of features. The autoencoder is part of unsupervised learning as it does not require labelled

data, it learns to represent the data, usually, through a low number of features while ignoring the noise and redundant information. Because it can learn to represent the data in a low-dimensional space while also being able to recreate the input, it is an appropriate method of dimensionality reduction. The general architecture of an autoencoder is presented in Figure 2.11.

A general structure of an autoencoder can be defined as n-p-n, where $0 < p < n$. The autoencoder can be split into three principle interconnected components: encoder, latent code and decoder. The encoder and the decoder can each be viewed as neural networks and can contain multiple hidden layers. The encoder receives the original data and throughout its layers decreases consecutively the number of neurons until it reaches the latent code, thus the output of the encoder is the code. The decoder receives as input the latent code and throughout its layers increases consecutively the number of neurons until it reaches the final output, which usually is of the same size of the input. The goal of the autoencoder is to reproduce the input at the output, while passing the data through a bottleneck. This bottleneck, called the latent code, can be used as a new reduced set of features, as it is able to reproduce the input by using it, thus the autoencoder can be used as a Dimensionality Reduction method.

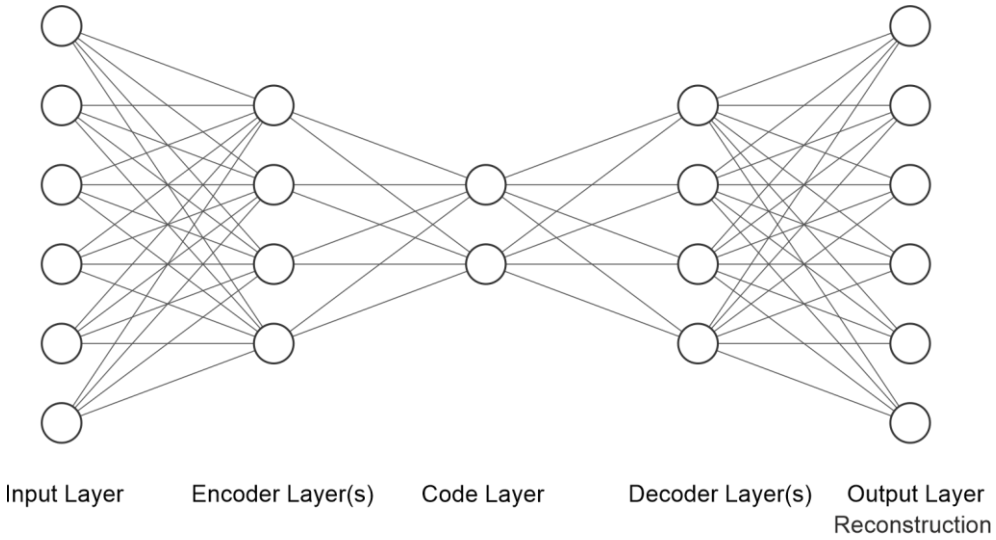

Figure 2.11 - A simple example of an autoencoder architecture.

Autoencoders allow for the design of custom architectures with a chosen number of layers and of neurons per layer. And naturally, the autoencoder is able to learn more complex codes by increasing the complexity of the architecture. Through its quality of being a neural network, the encoder and the decoder can have many variations, with symmetric or asymmetric structures, fully connected layers or even convolutional layers for images.

The goal of an autoencoder is to reconstruct the input at the output as precisely as possible, in order to achieve this, it uses a loss function, generally the mean squared error between input and output. Indiscriminate of the choice of loss function, its role is to punish the autoencoder when the output does not reflect the input, thus by minimising the error, the autoencoder is able to reconstruct the input at the output.

Autoencoders have been shown as a viable dimensionality reduction strategy [51–53] for computer vision using datasets such as MNIST. Recent developments have been

made even on autoencoder-based methods for spike sorting [54,55]. Although their evaluation is limited to synthetic datasets with low numbers of clusters, the evaluation has been made using accuracy as a metric, which is unsuitable for an imbalance problem such as spike sorting.

Variants of autoencoders

There exist many autoencoder variations. Tied [56] autoencoders have the weights of the encoder and the decoder linked from initialisation throughout the learning process, they have been shown to have a faster convergence and to provide better reconstructions.

Autoencoder, just as other neural networks can be pretrained in multiple manners, one option of pretraining is a greedy layer-wise approach [57]. Pretraining can provide a better way of weight initialisation that can increase performance by increasing the likelihood of circumventing local minima. As the name suggests, the layers are trained in a one-by-one fashion and then saved.

The Long Short-Term Memory (LSTM) [58] is a type of neural network that solves the problem of long-term dependencies being lost on Recurrent Neural Networks (RNN). In RNNs, initial inputs are learned, and as new inputs appear, the old ones start to lose importance, this phenomenon is called vanishing. From a distance, an LSTM cell works similarly to that of an RNN cell. The LSTM is composed of three parts, called gates, the input gate, the forget gate and the output gate and an additional part the hidden state. Information as inputs, passes through the LSTM and it automatically modulates whether the information that is retained or forgotten. LSTM cells can be used in any neural network architecture.

Orthogonality [59] is a constraint that can be applied to the weights of a neural network. In the case of autoencoders, it forces the weights of the encoder and decoder to be orthogonal through a kernel regularizer. It can reduce the correlation between the encoded characteristics by applying a penalty on the characteristics based on the covariance matrix.

Contractive [60] autoencoders bring a modification of the loss function. The weights of the autoencoder are constrained through regularisation to remain small. Thus, bestowing robustness to noise by favouring the contraction of the input samples, increasing the robustness or the encoded characteristics but not necessarily that of the reconstruction. The squared Frobenius norm of the Jacobian is used, instead of the usual L2 norm, as the regularizer term. The new loss function is given by the following formula:

$$J_{CAE}(\theta) = \sum_{x \epsilon D_n} \left(L\left(x, g\left(f(x)\right)\right) + \lambda \left|\left| j_f(x) \right|\right|_F{}^2 \right) \quad (1)$$

where λ is a hyper-parameter that represents the strength of the regularisation, L is the reconstruction error, typically chosen as Mean Squared Error (MSE) or crossentropy.

The Generative Adversarial Network

Generative Modelling makes part of the subdomain of Unsupervised Learning in Machine Learning. It focuses on discovering and learning the underlying pattern of the input data in order to be able to generate new samples that could have seemingly come from the original set of data.

Generative Adversarial Networks (GAN) [61] formulate the problem of Generative Modelling as Supervised Learning by separating the solution into two models, a generator and a discriminator. The Generator Model is trained to generate new samples, whilst the Discriminator Model learns to identify (classifies) if the given sample is a real one (from the data) or fake (generated by the Generator). These models are trained adversarially, also known as a zero-sum game, until the Generator is able to deceive the Discriminator consistently but not completely. Reaching this point indicates that the Generator is able to generate samples that are very similar to the original.

GANs can be considered a deep-learning generative model. Moreover, they are an architecture for the training of a generative model and most commonly deep models are used. This introduction of deep models, named Deep Convolutional GAN (DCGAN), into the GAN framework has helped stabilise the generative model. [62] Therefore, most GAN models today are based on the DCGAN [63].

The Generator

The Generator Model receives inputs as vectors of a fixed length containing random values, and as output generates samples in a targeted domain. Usually, the vector contains a random Gaussian distribution. Succeeding the training, the points represented by the fixed-length vector space correspond to the points of the problem domain, thus a compressed representation of the data is formed.

This fixed-length vector space is also called a latent space, another version used is to refer to the vector space as containing latent variables. These latent variables are defined as random variables that cannot be observed directly. [64] The latent space represents a compression of the raw data distribution. For the particular case of GANs, the latent space of points receives meaning from the inner workings of the model, such that each new point of the latent space given to the Generator is used to generate a new and different sample. A simple illustration of the Generator and its inputs and outputs is shown in Figure 2.12.

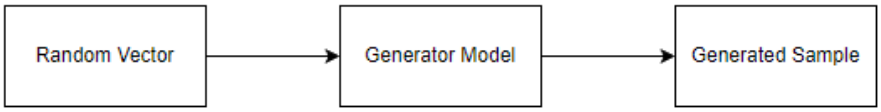

Figure 2.12 - A simple diagram of the Generator model inputs and outputs.

The Discriminator

The Discriminator Model is trained to be able to perform a binary classification, given a new sample it decides whether it is a real sample or a generated one. A real sample comes from the set of data, while a generated sample comes from the Generator Model. The network type of the Discriminator is a well-known normal and simple network architecture, the novelty comes from its use in combination with the Generator. A simple illustration of the Discriminator and its inputs and outputs is shown in Figure 2.13.

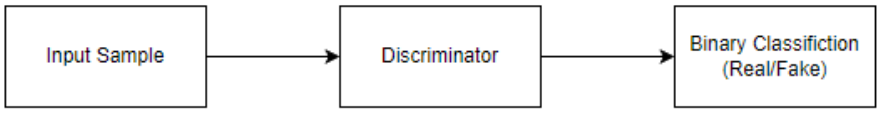

Figure 2.13 - A simple diagram of the Discriminator model inputs and outputs.

Generator and Discriminator Interaction, A Two Player Game

As mentioned above, the GAN architecture allows the framing of the generative modelling problem, which is of the unsupervised learning sort, as a supervised learning algorithm. The generator and discriminator are trained together. The flow starts with the generator's generation of a batch of samples which are given to the discriminator to be classified. This is followed by the update of the discriminator to improve its performance, while the generator is updated on its ability to fool the discriminator. An illustration of the interaction between the Generator and Discriminator in order to create the GAN model is shown in Figure 2.14.

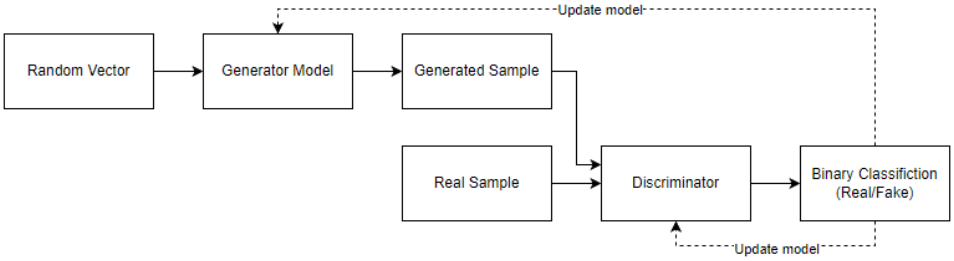

Figure 2.14 - The interaction between the Generator and Discriminator models creating the GAN.

Therefore, the two models are competing against each other and playing a zero-sum game, the term adversarial was taken from game theory to describe the behaviour of this architecture. When the discriminator correctly identifies between real and fake samples, the generator is heavily penalised - large updates are made to the model parameters in order to improve the generation, while the discriminator receives no change. And when the discriminator is fooled, the inverse happens.

The ideal case of the GAN, when convergence is reached, would be a generator that creates perfect replicas of the input, which would result in the discriminator being unable to distinguish between real and fake samples and therefore, its predictions being unsure in every case with a 50% accuracy. [64] This point need not be reached in order to create a successful generator model.

Other Uses

GANs are most typically used when working with images, thus they are using Convolutional Neural Networks (CNN) to implement the generator and discriminator. For the modelling of image data, the latent space - the input of the generator - yields a compressed representation of the data set of images used for training. As expected, the Generator will then generate new images, an output that is easily visualisable by observers allowing for empirical validation of the generation.

A critical development added to the GAN architecture was the Conditional GAN. In this case, the generative model is trained to generate new samples conditioned by some additional input. [65] This additional information conditions both the generator and the

discriminator, for example: a class label. The conditioning can be done by feeding the condition as an additional input layer to both.

Given that the discriminator also receives the conditional information, when classifying the discriminator expects the input to be of the given type. Thus, it forces the generator to create samples of that class otherwise being unable to fool the discriminator. In such manner, a Conditional GAN learns to generate samples from a given domain.

The Conditional GAN models allow for applications such as image-to-image translation [66], these permitting more impressive transformations: season change in images, day-to-night switch, writing style transfer, colorization of images. Within applications such as these, the discriminator is given examples of both real and fake of the intended transform, while being conditioned on the real images of the original.

What GANs provide

Data augmentation is a technique commonly used to boost the model's capacity to learn, while also contributing with a regularisation effect by increasing the model's ability to generalise. The idea behind it is to create new samples from the original dataset on which to train the model on. Thus, it also allows the expansion of small datasets that have too few examples to train a model. In its most simple form, data augmentation is comprised of elementary image processing operations, such as: flips, crops, rotations, addition of noise and others. Therefore, by its definition, generative modelling could be classified as a data augmentation method as it generates new examples, although it provides a more complex and possibly more specific to a given domain approach. Another way of looking at it, data augmentation could be viewed as a more simplified variant of generative modelling [67]. With the emergence of GANs, Variational Autoencoders were quickly replaced in data augmentation.

GAN Failure

The training of the generator and discriminator concomitantly in a zero-sum game results in a trade-off. Improving either model is made at the cost of the other. Thus, a point of balance has to be found in the competition. As the two models each change their parameters separately, the optimization of each keeps changing which generates a dynamic system. Therefore, it is expected to see a higher variance in accuracy and loss than in typical models during the training period. While the most accurate generation of samples is done when stability occurs. It is easy to infer that GANs can have failure to converge. As an example, a discriminator with perfect accuracy in classifying will not produce a suitable generator.

Another difficulty that may appear in the training of GANs is called mode collapse. Mode collapse refers to the case in which the generator maps different inputs to the same output [68]. Thus, resulting in only a small subset of generated samples, or modes [63]. In this case, the mode appertains to the output distribution. As an example, multi-modality refers to the presence of multiple peaks, while mode failure in GANs is the inability to generate diverse samples from a hyper-dimensional input space. The mode collapse in the generator can be identified in two manners. Firstly, an examination of a large number of generated samples will highlight little diversity, as similar or identical samples have been generated. Secondly, an examination of the generator loss will display oscillations that are corresponding to the generator jumping from one mode to another, as the losses are different. The most common occurrence of mode collapse is an insufficiently large input latent space.

Convergence failure happens when the GAN is unable to find the balance point between the generator and discriminator. The most common case is the high accuracy and low loss of the discriminator. In these instances, the generator is creating easily identifiable samples. When applying GANs on images, this can happen due to aggressive loss function, too large or too small kernel sizes. These types of failure are effortlessly detectable by visualising the loss during training.

Wasserstein GAN

Wasserstein GANs (WGAN) have been shown to overcome the most common failures of GANs, in training [69]. WGANs apply a different approach to the GAN formulation, they designate the latent space with a fixed distribution and pass it through a parameterized function, in order to map it to a new distribution. Thus, by varying the parameters the new distribution varies as well. Two advantages emerge from this approach: a higher number of distributions can be learned and the generation of new samples from the latent space becomes easier. In the original paper [69], they propose the use of a different distance metric called Earth Mover or Wasserstein metric. This metric measures the minimum cost required to transform a distribution into another of the same size.

Worth mentioning is the role change of the Discriminator in this approach. In this configuration, it is also called a Critic. Additionally, to distinguish between real and fake samples, it converges to a linear function that is used to improve the training of the generator.

Another improvement added to WGANs is related to the optimization task, as using weight clipping causes issues of exploding and vanishing gradients [70]. The proposed approach is gradient penalty (WGAN-GP). The Discriminator, or Critic, receives a penalty of its gradient norm output with respect to the input, while also adding a light variant for random samples.

A notable improvement has been made through the leveraging of nearest neighbour search algorithms. [71] This allows the training of GANs with a lower amount of data by finding the nearest real sample to each generated sample and adjusting the parameters to narrow the gap. This improvement should also neutralise the mode collapse failures and, allegedly, the instability and vanishing gradient issues. A disadvantage that emerges is the computational cost of nearest neighbour search algorithm addition.

Self-Organizing Map

The Self-Organising Map (SOM), introduced by Teuvo Kohonen in 1981 [72], is an unsupervised learning technique that transforms the input space into a bidimensional (in its most classical form) map of processing units (also called neurons in the literature). It is able to project high-dimensional data into two-dimensional space while preserving the topological structure of the data. SOMs are trained using competitive learning instead of the typical backpropagation with gradient descent. Its most common uses are in data visualisation and analysis [73].

The SOM generates a two-dimensional matrix of neurons, where the width and the height are parameters to be chosen, where each cell contains a weight vector that is randomly initialised. For each sample, during training, the closest node to a given sample, or Best Matching Unit (BMU), is found and the BMUs weights are updated towards the sample according to a learning rate parameter. This is considered the competition part of

the algorithm. The neighbouring nodes are also updated to a smaller degree following a gaussian function, which is the collaboration part. The iterative training process can be stopped after a certain number of iterations or when only minimal changes occur in the mapping. Commonly, during training the learning rate and neighbourhood radius can also be decreased iteratively [74]. The competition part consists of finding the most similar unit, formulated in the first equation, while the collaboration is the sharing of the winner unit *i* with units *j* within a certain distance $d_{i,j}$ [75].

$$i(x) = argmax_j \left|\left|x - w_j\right|\right|_2 \text{ where } j = 1,2,\dots,m \tag{2}$$

$$h_{i,j}(d_{i,j}) = \exp\left(\frac{-{d_{ij}}^2}{2\sigma^2}\right) \tag{3}$$

The training of a SOM network can be expressed in pseudocode:

```
function train(samples):
      learning_rate = 1
      radius = 1
      som_size =10
      som = initialize_som(som_size)
      for k in 1 to len(samples):
            bmu = find_bmu(samples[k])
            update_weights(samples[k], bmu)
            decay(learning_rate)
            decay(radius)
```

Many variations and extensions have been designed [73], and parameters can be changed, such as extending the SOM into a three-dimensional cube and incorporating various neighbourhood and distance functions. The recommended map size depends on the number of training samples [76]:

$$Size = 5 * \sqrt{samples} \tag{4}$$

The initialization method is crucial for SOM training [77], with random initialization being the most common, even though it induces non-determinism and has longer execution times. A more efficient alternative is to initialise the SOM weight vectors with samples from the training data thus reducing the initial distances and updates required, but it does not address the non-determinism. A deterministic alternative is to initialise the weight vectors with linear combinations of principal components (PCs). Nevertheless, it seems that random initialisation has a better performance for non-linear datasets than the PCA initialisation [77].

As other neural networks, SOMs try to minimise the error computed as the distance between samples and nodes. In the case of SOMs, its training performance can be evaluated through quantization or topographic errors. The quantization error (QE) is computed by calculating the average distance of the input data point to the closest nodes in the lattice, given by the following formula:

$$QE = \frac{1}{|D|}\sum_{i=1}^{|D|} ||x_i - w||^2 \quad (5)$$

where D represents the input data, x the current sample and w the weight vector of the best-matching unit.

It is important to highlight that the quantization error (QE) decreases as the SOM lattice becomes larger, therefore it cannot be used to compare maps of different sizes [78].

In contrast, the topographic error (TE) measures the preservation of the data shape in the map. In its simplest form, it calculates the proportion of inputs where best-matching units and second-best-matching units are neighbours in the map, given by the following formula:

$$TE = \frac{1}{|D|}\sum_{x\epsilon D} te(x) \quad (6)$$

where D represents the input data and c_i the i-th best matching unit, and $te(x)$ is equal to 1 if $c_1(x)$ and $c_2(x)$ are neighbours, and 0 otherwise.

The SOM training time has a linear complexity with regard to the number of samples and a quadratic time complexity in relation to the map size [74]. Besides those presented, many other parameters are involved in the training of a SOM.

The main utility of this method is visualisation as the nodes created are viewed as independent clusters, then the number of clusters is equal to the number of nodes in the map. However, by using SOM as a feature extraction algorithm and combining it with a clustering method, such as agglomerative clustering, it can provide satisfactory results as a feature extraction algorithm for clustering [79,80].

U-matrix

The U-matrix, also known as the unified distance matrix or the distance map, is a visualisation technique used to understand the resulting topology and relationships between the artificial neurons of a self-organising map (SOM) post-training. It is represented as a grid, of the same size as the SOM, of values where each cell represents the distance or dissimilarity between neighbouring neurons in the SOM grid. The distance is typically calculated using the Euclidean distance between weight vectors of adjacent neurons. By calculating the U-matrix, a visual representation is obtained of the relationships between neurons allowing us to understand the topology of the SOM by identifying areas of high and low density, allowing underlying structures within the input data.

Pulse Coupled Neural Networks

Pulse Coupled Neural Networks (PCNN) were modelled after the cortical neurons of the cat's visual cortex and the phenomena of oscillating pulses [81,82] and was inspired by gamma band synchronisation, where neurons fire synchronously in a region based on similarity and proximity [81]. PCNN proved to be appropriate for image processing and modifications of the original made have been created to optimise image segmentation and other procedures [83].

The pulses generated by the neurons of the PCNN are resulted from the excitation obtained by the stimulus and the interaction between neighbouring neurons. These determine synchronous firing in the homogenous areas of the image [84]. The neurons used in PCNN are a specific type of leaky integrators, where exponential decay is used as the rate of leaking [81]. While the refractory period of the neurons is modulated through the use of the dynamic threshold by an exponential decrease after firing [81,84] and following pulses require increased amplitudes. Using images as stimuli results in a non-linear transformation, due to the exponential decays and the couplings among neurons [82].

Generally, PCNNs are single-layered 2D networks, where each pixel has a neuron as a correspondent. They can be separated into five main parts: feeding input (F), linking input (L), internal activity (U), the dynamic threshold (Θ), and the output (Y) [85]. The formulas of the different parts of the PCNN model are described by the formulas below [81–85]. The current iteration is defined as *"n"* in these equations and these formulas are applied for each neuron of the network identified by the *i and j* indexes and the neighbours of $neuron_{ij}$ are identified by the *k and l* indexes.

Additionally, the dynamic threshold, feeding and the linking input contain exponential decay factors of the previous states: $e^{-\alpha f}$, $e^{-\alpha l}$ and $e^{-\alpha\theta}$. W_{kl} *and* M_{kl} are the local synaptic weights of the feeding and linking inputs, respectively. Thus, the neighbouring neurons of the PCNN create a visual receptive field, where the strength of the local synapses is given by the weights. V_f, V_l and V_θ are used to modulate the surrounding neurons of the current and V_θ regulates the refractory period by increasing the threshold after a pulse. $\boldsymbol{\beta}$ is a constant used to amplify the linking input. The external stimulus is defined as S, thus each pixel is a stimulus for a certain neuron of the model. The neuron output (Y_{ij}) is set to 1 when the internal activation (U_{ij}) surpasses the dynamic threshold (Θ_{ij}) [81–85]. Figure 2.15 exemplifies the simplified flow of the PCNN model.

$$F_{ij}(n) = e^{-\alpha f}\, F_{ij}(n-1) + V_f \sum_{kl} \left(M_{kl} Y_{kl}(n-1)\right) + S_{ij} \tag{7}$$

$$L_{ij}(n) = e^{-\alpha l}\, L_{ij}(n-1) + V_l \sum_{kl} \left(W_{kl} Y_{kl}(n-1)\right) \tag{8}$$

$$U_{ij}(n) = F_{ij}(n)\left(1 + \beta L_{ij}(n-1)\right) \tag{9}$$

$$\Theta_{ij}(n) = e^{-\alpha\theta}\, \Theta_{ij}(n-1) + V_\theta Y_{ij}(n-1) \tag{10}$$

$$Y_{ij}(n) = 1 \; if \; U_{ij}(n) > \Theta_{ij}(n) \; else \; 0 \tag{11}$$

PCNN models have numerous uses within image processing: image fusion, image segmentation[86], image denoising, image enhancement, feature extraction, image restoration [81]. They have been used in MRI image enhancement [82] and medical image segmentation [85,87]. Moreover, using the results obtained by the PCNN, Artificial Neural Networks (ANN) classifiers can be trained [84,87].

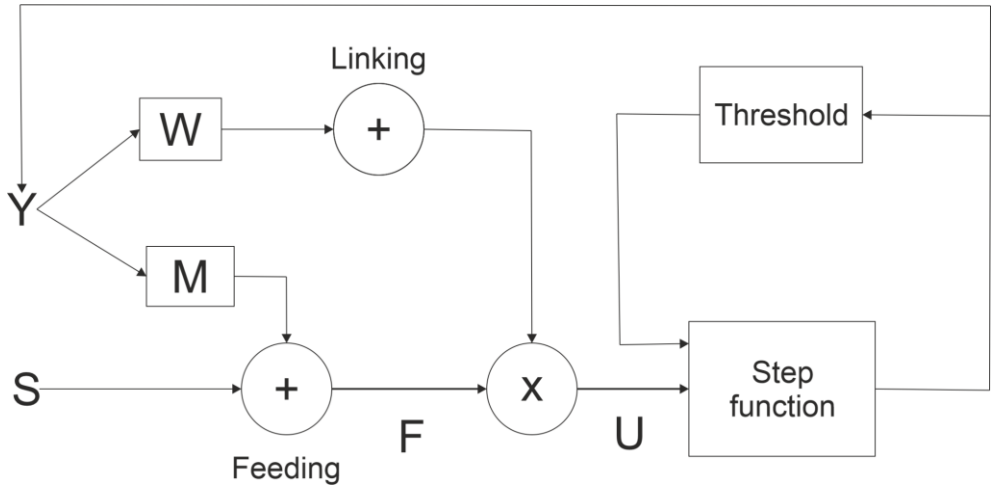

Figure 2.15 - The flow of the PCNN model.

Several modifications of the classic model of PCNN have been developed. The Intersecting Cortical Model (ICM) [85,87] removes the linking unit and therefore is the equivalent of setting $\boldsymbol{\beta} = 0$. The ICM model has been shown to offer the best advantages in image segmentation [88]. Such a model is capable of extracting the fundamental characteristics of an image, such as edges and segments without requiring training on a large dataset of images. Edges or segments are extracted as groups of neurons in a similar state fire together, thus synchronising the firing of neurons according to the texture of the image. Furthermore, the result is a binary image that can be presented to more complex recognition algorithms for analysis.

In the Spiking Cortical Model (SCM) [81,83,87], the membrane is defined only by the linking unit. In the Sigmoidal-Linking Model (SLM) in which the linking input has been changed into the step activation function has been replaced with another function, such as a sigmoid function. In the Feature-Linking Model, the model is further reduced, as are its parameters, it contains only the internal activity module with an additional inhibitory linking term. The Multiple-Linking model contains internal iterations of the classical model and an additional outer calculation of the feeding and internal activity modules, where the internal activity is computed as the geometric mean of the internal linking modules [81].

Performance evaluation metrics

The performance evaluation metrics presented in this section can be calculated from the confusion matrix, normally used to evaluate classification correctness. Fundamentally, given a confusion matrix *C,* each cell $C_{i,j}$ contains the number of samples of class *i* that have been predicted to be in class *j*. In a binary classification there are only 4 cells in the confusion matrix as shown in Table 2.2 [89].

Table 2.2 - Explanation of the Confusion Matrix [89].

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

The confusion matrix presents the counts of predicted versus actual labels in classification. True Negative (TN) is the count of values that have been correctly classified as negative, while True Positive is the count of values correctly classified as positive. The negative and positive denominations are relative to the class of interest. False Positives get assigned the instances in which the actual label was negative, but the classifier marked it as positive. Similarly, False Negative indicates the number of instances that have the actual label as positive but were labelled negative by the classifier.

Accuracy is the most common metric used in classification, however, for imbalanced datasets accuracy can be misleading. To give a simple example, having a dataset of 1000 images showing tumours, of which 990 are benign and only 10 are malignant. The classifier may fail during training and assign to all images the label 'benign'. This would result in an accuracy of 99%. Thus, accuracy, as any other metric, can be useful in certain instances but has to be used mindfully [90–93]. Essentially, accuracy shows the probability with which the classifier predicts correctly. Accuracy can also be expressed mathematically using the confusion matrix:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Precision can be thought of as a measure of exactness or quality of prediction. Precision, also known as positive predictive value, is the measure of the ratio of predicted positives that are in fact correctly predicted. It is often used together with recall in order to evaluate classification performance. Precision depends upon the distribution of classes as imbalanced classes may induce lower precision values, nevertheless it is often used as a measure of quality. Precision can be defined mathematically using the confusion matrix:

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

Precision has its main uses in real-world applications, such as spam filtering. For applications such as these, usually a greater cost assigned is assigned to false positives, thus precision is used to minimise false positives. Similarly, seeking high recall values reduces the number of false negatives.

Recall, also known as sensitivity or true positive rate, measures the ability of the model to correctly classify actual positives. High recall is indicative of a model that is reliable in identifying both positive and negative samples. Recall is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

Recall has been found useful in various real-world applications, such as fraud detection, sentiment analysis and medical diagnosis.

Notably, precision may not decline with recall. Usually, classification employs a prediction threshold. In such cases, a higher threshold might generate more true positives and fewer false positives which increases precision. Thus, by lowering the threshold below the correct value, the number of false positives increases, decreasing precision but leaving recall unchanged. By contrast, recall is not as dependent on the

prediction threshold. Lowering the threshold may increase recall by introducing true positives, which is desirable. But it may also leave recall unchanged as no other true positives are found, while precision wavers.

F1 score is used as an alternative to accuracy, which better reflects the classifier quality. It is often used when optimising recall or precision results in decreased model performance. Mathematically, F1 is the harmonic mean of precision and recall, but it can also be defined using the confusion matrix.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{15}$$

A generalisation of the F1 score, denoted as Fβ score uses a real factor β to create a weighted version of this metric. Through this parameter, recall or precision can be considered more in the computation of the score.

$$F\beta = (1 + \beta^2)\frac{2 \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{16}$$

These metrics are most commonly used in conjunction with each other in order to apprehend all aspects of the model.

2.2.2. Machine learning approaches

SVM is a supervised learning technique that can be applied for classification, outlier detection and even, regression. It is highly adjustable through its kernel and effective for hyper-dimensional data. SVM is considered to be decidedly robust as it is based on statistical learning. In training, SVM attempts to magnify the gap width between the different classes. New samples are assigned to a class by their location in space in relation to said gap.

RandomForests, as the name implies, consist of a number of decision trees that have gone through the learning process independent of each other. Just as its precursor, RandomForest is a supervised learning technique. The data can be split into subsets in order to further dissociate the decision trees, with the additional benefit of reduced training time. Depending on the data, there are multiple ways in which the output of these trees can be combined into a single one. One option is the classical majority vote, where the most common output is chosen, other options are mathematical operations as the average. More complicated options are placed under the banner of ensemble learning.

2.2.3. Signal processing

Spike sorting is using signal processing even in its first step, the filtering. The spike detection is done based on the characteristics of the filtered signal and the spikes themselves are segments of the filtered signal. As such, signal processing techniques can be applied to the domain of spike sorting. One avenue is to increase the amount of information from the spikes by obtaining the frequency information through signal processing techniques such as the Fourier Transform.

Time dependent functions can be represented from the perspective of their frequencies. Mathematically, the frequency domain uses complex numbers for representation. An intuitive representation of the correspondence between time and the frequency domain is presented in Figure 2.16.

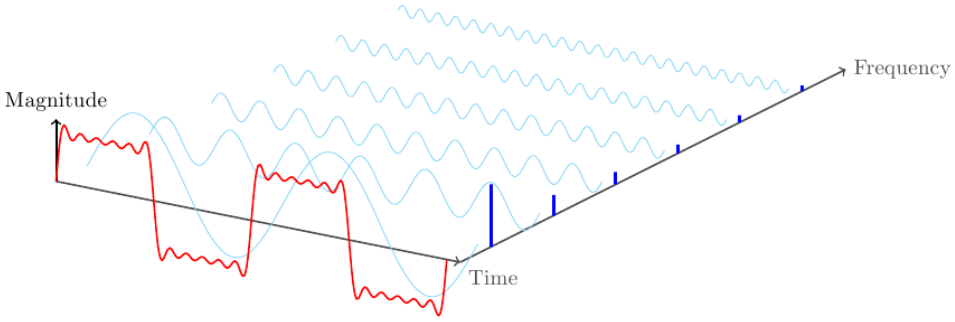

Figure 2.16 - Time vs Frequency Domain. The red trace is the sum of the light blue sinusoidals. The image was taken from Wikimedia Commons (public domain).

Fourier Transform

Jean Baptiste Fourier stated that any function, even non-periodic ones, can be expressed as a sum of sine and cosine functions. A perfect representation requires an infinite sum which can be expressed mathematically as an integral:

$$f(x) = \int_{-\infty}^{\infty} A_\omega \cos(\omega x) + B_\omega \sin(\omega x)\, d\omega \tag{17}$$

The sine and cosine coefficients in the Fourier Integral represent the contribution of each frequency. These coefficients are used in the computation of the Fourier Transform (FT), one of the most common transformations from the time to the frequency domain. FT allows any function to be written as a sum of harmonic functions, sines and cosines:

$$S_x(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-i2\pi f t} dt \tag{18}$$

Worth mentioning is that although the FT is applied most commonly on complex functions resulting in complex values, it can also be applied on real functions, but the result is still a complex-valued representation, albeit with some spectral symmetries between the positive and negative frequencies. Moreover, the Fourier Transform is invertible, thus the original function can be reconstructed, making the time domain and the frequency domain equivalent.

The Discrete Fourier Transform (DFT) is used to apply the FT on digitally sampled signals, which can be represented by a sum of base frequencies multiplied with a factor with a shift from the origin [94].

The Short Time Fourier Transform (STFT) is a FT variant and a modification of the DFT that accounts for time in its representation of the frequency domain.

$$X_l[k] = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} w[n] \cdot x[n + lH] \cdot e^{-i2\pi \frac{kn}{N}}, \text{where l} = 0,1, \ldots \tag{19}$$

The main addition to the original formula is the windowing function. The main parameters of the STFT are: the window, the frame number and the hop size. The window is used with the original signal in a convolution. The signal at a given moment is multiplied with the window values. The frame number, notated “l” in the formula, is the

time index. It is used to iterate over the signal and obtain the number of segments. Thus, the time is segmented and only then is the transition to the frequency domain made, thus introducing temporal information in the representation. The hop size, notated "H" in the formula, represents the jump from one time segment to another. The attentive reader has realised that STFT then outputs a sequence of spectra instead of a single spectrum as the FT does. Each time segment is represented by a spectrum, all of them having the same size.

Returning to the topic of windows, there are many types, such as Blackman, Bohman, Hamming, Hann, and others. Each window offers unique spectra leakage mitigation properties, which are related to the main and side lobes. A thinner main lobe and smaller side lobes reduce the spectral smearing trough leakage. Nonetheless, the window of choice is problem dependent. The STFT uses the concept of windows to apply a sequence of Fourier Transforms to the signal. By contrast to the conventional Fourier Transform, STFT is able to localise in time the spectral information and is useful when the signal's frequency components change over time. With its fixed window size, STFT has a good frequency resolution but its relative temporal precision decreases as frequency increases because the length of an oscillation cycle becomes smaller in comparison to the window size.

Wavelet Transform

One transform that is able to overcome this limitation of the STFT is the Wavelet Transform. Similarly to the STFT, the Wavelet Transform can describe a signal by a series of wavelet coefficients obtained by convolving the signal with the wavelets. The Wavelet Transform can be used, as well, for both the analysis and the synwork of signals. Unlike the Fourier Transforms, base functions which are infinite in time, the wavelets are localised in time. Thus, the Wavelet Transform is best suited to pinpoint oscillation with finite duration.

The wavelet transform contains two main parameters: the mother wavelet and the scale. There are many choices for the mother wavelet, but one of the most widely used is the Morlet, which is obtained by multiplying a complex plane wave with a Gaussian envelope. The scale is used to compress or expand the mother wavelet in time. In doing such the scaled wavelets can be adapted to analyse various frequencies and obtain a full spectrum of a signal. A mother wavelet can be longer or shorter. For instance, for a Morlet, one could choose the number of cycles of the complex wave that fill the Gaussian envelope. This choice of length has impact on the analysis. For a given frequency, short wavelets have good temporal resolution but low resolution in frequency. Conversely, longer wavelets have a greater frequency resolution and lower resolution in time. Take for instance a finite packet of oscillation. If analysed with a much longer wavelet than itself, the oscillatory packet's contribution to the wavelet response is small and the packet is not well revealed by the wavelet representation, a process that is called dilution [95]. Thus, the length of the wavelet requires a compromise between the time and frequency resolutions. Same dilution problems are also found in the FT, when short burst of oscillations is analysed with a too long window. This usually happens when the window is adapted to capture well a couple of cycles of low oscillations.

Superlet Transform

The Heisenberg-Gabor Uncertainty Principle (UCP) explains the limitations of representations that use any sort of finite-length windowing functions such as STFT and

Wavelet. According to UCP signals cannot be localised simultaneously both in time and in frequency precisely [95]. Intuitively, short signals must have broad spectra, and for narrow spectra, the signal must be long. The same principles apply to the window function of the representation which influence the representation itself. For example, STFT is able to provide good temporal resolution with short windows at the cost of frequency resolution, while with long windows it is able to provide a good frequency resolution albeit at the cost of temporal resolution.

The Superlet Transform (SLT) [95] was developed in order to overcome these limitations. At a given frequency (or scale) it combines several wavelets (short and long) in order to pinpoint signals both in time and frequency. Consider a mother wavelet with several cycles and the family of wavelets derived from it. The family comprises compressed and stretched variants of the mother wavelet, each of the variants has its own scale or central frequency and is used by the wavelet transform to generate the spectra at various frequencies (or scales). The family provides a certain amount of time and frequency resolution that is frequency dependent. The frequency resolution decreases as the wavelet's centre frequency rises to match high frequency signal components. By increasing the number of cycles of the mother wavelet the Continuous Wavelet Transform (CWT) displays increased frequency resolution but lower temporal resolution across the whole spectrum.

A Superlet (SL) is defined as a set of wavelets grouped around a central frequency that extend over a range of numbers of cycles, this is represented mathematically in following Equation, where *o* represents the order number (the number of wavelets, bounded within the [2,15] interval, while the number of base cycles within the [1.5, 3] interval) and *c*'s correspond to the number of cycles of each wavelet (a function of the bandwidth of the wavelet). With these parameters the time-frequency resolution can be adjusted. Each SL of the SLT approximates a subset of oscillations of a certain frequency from the signal, thus super-resolution in both time and frequency is achieved.

$$SL_{f,o} = \{\psi_{f,c} |\, c = c_1, c_2, \dots, c_o\} \tag{20}$$

The Superlet Transform of a signal x is the geometric mean of all wavelet responses to x:

$$R\left[SL_{f,c_i}\right] = \sqrt[o]{\prod_{i=1}^{o} R\left[\psi_{f,c_i}\right]} \tag{21}$$

$$R\left[\psi_{f,c_i}\right] = \sqrt{2} \cdot x * \psi_{f,c_i} \tag{22}$$

where $R[\psi_{f,ci}]$ is the i-th wavelet response to the signal.

The two parameters are highly sensitive can produce very different results based on the choice of values. These are the order o and the number of cycles c. The order represents the number of wavelets used in the transformation and can take values between [2,15], whilst the number of cycles is given for the first wavelet and can take values between [1.5, 3], and the following wavelets are calculated by the following formula dependent upon the order:

$$c_i = i * c_1 \quad where\ i = 2,3, \dots, order \tag{23}$$

2.3. Neuroscience and computer science

2.3.1. Spike sorting

The communication between neurons can be recorded through electrodes inserted into the brain as changes in voltage. This transmission of information between neurons is called an action potential, or spikes. Spike sorting [2] is the separation of these spikes into groups based on the neurons that fired them. Spikes are of known fixed duration (of about 2-3ms) and the information about their provenance (the neuron that produced them) is encoded in the distribution, amplitude and their shape. Spike sorting is the strategy that aims to 'sort' these signals into groups based on similarities of shape, as the geometrical conformation of each neuron and the distance to the electrode defines the shapes sensed by the electrode [96]. In other words, spikes fired by the same neuron have similar shapes at the recorded site.

Spikes need to be detected from the raw signal recorded by the electrode. Spike sorting distinguishes the spikes of individual neurons from extracellular recordings where their action potentials are mixed. For small or far away neurons from the electrode, the amplitudes of the spikes on the recording electrode are so low that they cannot be disentangled from noise.

The spike sorting pipeline

Spike sorting is a sequential process that has been refined into four principal steps (Figure 2.17) [2]:

- Filtering
- Spike Detection
- Feature Extraction
- Clustering

The first step of spike sorting is the filtering of the raw signal through the use of a bandpass filter. The bandpass filter allows the separation of signals that are between two specific frequencies from the rest of the spectrum. Through filtering a portion of the noise and low frequencies are removed, while the spectral components of the spikes are kept. The low-pass part of filtering removes the slow components of the raw data, while the high-pass part reduces the noise in shape of the spikes [97,98]. A trade-off in band size has to be made, as a narrow filter can diminish the features of the spike shape, while a broad band introduces noise into the shape.

Spikes are determined by higher amplitudes. The second step of spike sorting is spike detection usually done via amplitude thresholding, whereby voltages above the threshold are considered spikes. Another trade-off needs to be made for the threshold value. A high threshold results in the loss of low-amplitude spikes, while a low threshold can leak noise into the spikes. The standard threshold value [2] is presented in Equation (1).

$$Threshold = \sigma_n, where\ \sigma_n = median\left\{\frac{|x|}{0.6745}\right\} \quad (24)$$

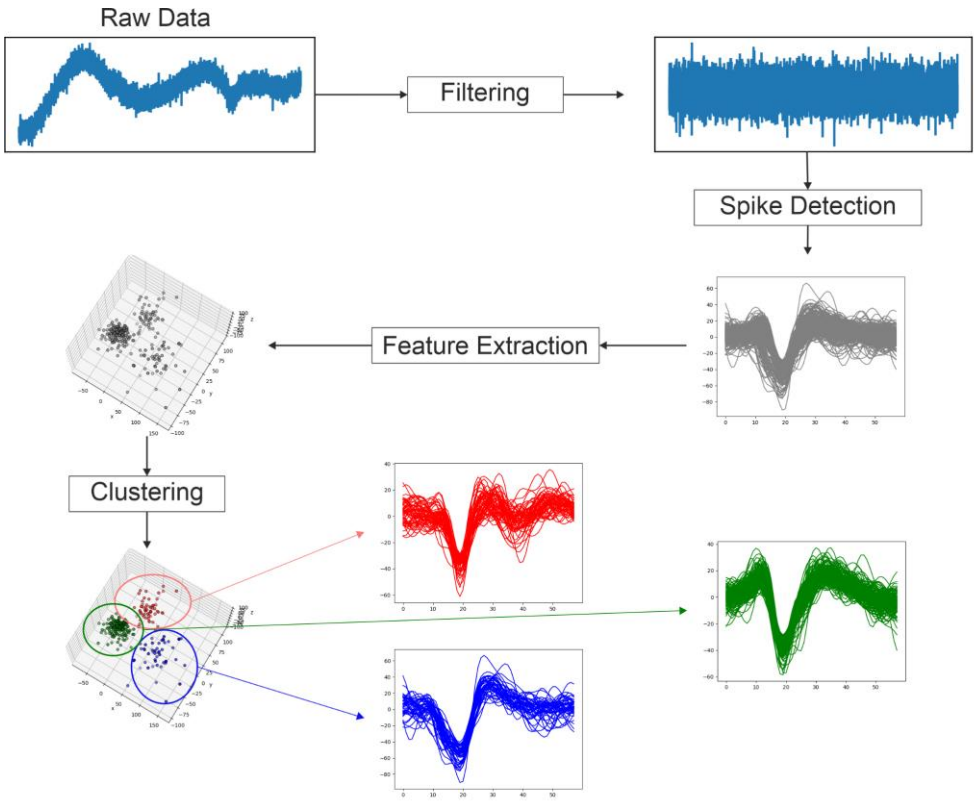

Figure 2.17 – The spike sorting pipeline [2].

The third step of Spike Sorting is the Feature Extraction which consists of the transformation of the initial feature set, composed of the samples of a waveform, that represents the spike into a smaller feature set that retains most of the information of the detected spike. Each spike is defined initially as a waveform through the spike detection step; therefore, it can be viewed as a vector where each feature is a dimension. Thus, a spike is in a waveform length dimensional space. Through Feature Extraction the dimensionality of the spikes is reduced while retaining as much of the information that distinguishes between the spikes of different neurons. Dimensionality reduction reduces the processing time of the clustering algorithm while minimally affecting the performance as the information that separates the spikes is preserved. Usually, the spikes are moved into a 2-/3-dimensional space in order to be visualised by an expert.

The last step of Spike Sorting is the Clustering which receives as input the new feature space returned by the Feature Extraction. Clustering can be done manually by an expert observer, but the use of an automated machine learning algorithm is preferred as manual clustering is a tedious and time-consuming process. Manual intervention is used when automated algorithms are not able to perfectly distinguish clusters, it involves the inspection and categorisation of spikes based on waveform shape, amplitude, duration and other characteristics. Additionally, there are semi-automatic spike sorters, such as WaveClus [99], that require manual toning for top performance. The goal of clustering is to group spikes based on their similarities.

The challenges of spike sorting

Clustering is an unsupervised algorithm, meaning that it does not require the ground truth. Most of the time, the ground truth, the true identity of the neuron that has

fired a certain spike, is not known. Thus, the performance of the result is hard to interpret and is evaluated through special metrics that assess different properties of the spikes within the cluster, such as, the cluster size, the intracluster and intercluster distances and dispersions among others.

Spike sorting is a complex task as many challenges can appear due to the high complexity of neural interactions that appear in the data which translates into a harder task for any computer science algorithm:

- The **distance and orientation of a neuron** with respect to the probe can modify the spike shape [100]. Greater distance to the probe results in lower amplitudes in the recording which **increases the difficulty of spike detection.** Moreover, these differences in amplitude and shape must be accounted for by both the feature extraction and clustering methods.
- **overlap of spikes**: Two or more neurons may fire synchronously within a small window of time producing overlapping spikes in the recording. In this case, the spikes from different neurons are difficult, if not impossible to disentangle. Overlapping spikes result in distorted shapes that increase the difficulty for the feature extraction algorithm to capture relevant characteristics. Furthermore, these overlapping spikes insert ambiguity that may bias the clustering algorithm towards misclassification.
- **electrode drift** [101]: This phenomenon consists in the **change of the spike shape** of a single neuron due to the movement of the electrode throughout the recording, this can lead to increased similarity between spikes of different neurons that results in **overlapping clusters**. Confoundingly, electrode drift may change the shape of the spike, throughout the experiment, so much that it may appear as being produced by different neurons. This spike shape change can lead to spikes from different neurons becoming more similar in shape, blurring the boundaries between clusters resulted from the feature extraction. Moreover, these overlapping clusters hinder the performance of clustering algorithms.
- **different firing rates**: Neurons have different firing rates due to their roles; this leads to less spikes of some neurons being recorded which results in **imbalanced clusters**. From a computer science perspective, algorithms designed for spike sorting must be robust enough to handle such imbalanced datasets.
- **bursting**: Repeated firing of the same neuron in a small amount of time that may **produce different spike shapes**. Bursting introduces ambiguity in spike shapes, as spikes from the same neuron may exhibit variability in waveform shape, such as reduced amplitude. Bursting introduces another variable that is not taken into consideration in the most common spike sorting approaches as they consider spikes as singular events. The temporal dependency of bursting and the modified shape can induce difficulties for the clustering step of the spike sorting pipeline.

All of these complexities (imbalance, overlap, noise, etc.) have specific techniques that are able to deal with them, but when they appear simultaneously, complexity accentuates, and traditional techniques cannot cope [102].

Other challenges are inherent to the problem, or may appear through the use of inappropriate algorithms, thresholds or parameters for the dataset:

- **lack of ground truth**: The **spikes of two or more neurons may be identified as a single cluster** even though they were produced by different neurons as separation was unachievable. The closest equivalent to ground truth are patch clamps, although

they can only be applied to a single neuron and as such can be used as a partial ground truth for extracellular recordings. Without ground truth, the performance of spike sorting has an inherent uncertainty.

- **noise**: In noisy neural data, the shapes of **spikes can be distorted** and as such it is harder for a feature extraction algorithm to correctly approximate a new feature set that offers separability. Noisy data can result in overlapping clusters as noise tends to blur the cluster boundaries, thus a high amount of noise may extend clusters one over the other.
- **high dimensionality** of the feature space: It can hinder the performance of clustering algorithms through the **curse of dimensionality**, but it certainly increases the execution time depending on the time complexity of the clustering algorithm. There can be overcome using feature extraction, as long as the features provide separation. Nevertheless, the more features there are, the harder it is to project them in a lower-dimensional space while also preserving separability.
- **over and underclustering**: Overclustering is the **identification of the spikes of a single neuron as having been produced by multiple neurons**, while underclustering is the identification of the **spikes of multiple neurons as having been produced by a single neuron**. Both are symptoms of improper assignment of labels of the clustering algorithm. Nevertheless, depending on the clustering algorithm, this may be reduced or completely removed through a more performant feature extraction technique. Within the context of spike sorting, overclustering is acceptable because subsequently an expert observer or a post-processing algorithm can be used to merge similar clusters.

The challenges presented above can affect any method in various ways. Thus, it is important to be aware of these challenges, and to employ multiple techniques of feature extraction and clustering in order to achieve adequate spike sorting. Also, it is imperative to highlight the distinct functionalities of the feature extraction and the clustering. The separation of clusters is based on the extracted features. Thus, the quality of the clustering, regardless of the algorithm of choice, is intimately intertwined with the performance of the chosen feature extraction algorithm.

In this subsection, various types of techniques are presented that have been used in the identification of brain activity at a multi-cellular level through spike sorting. A subset of these techniques are well established within the domain of spike sorting, while others are well-known techniques that have not been introduced to spike sorting. All of these techniques have been utilised throughout the analyses made in this work.

Steps of the spike sorting pipeline

In this section, various options for the spike detection, feature extraction and clustering techniques of the spike sorting pipeline are presented. Additionally, several performance evaluation metrics are presented as well as ways to analyse the performance of the spike sorting pipeline.

Spike Detection

Traditional spike detection methods apply a simple amplitude threshold, either set manually or automatically, to identify spikes from the filtered signal. However, this approach has limitations. This type of threshold implies a compromise, high thresholds

may miss spikes with a lower amplitude than the threshold [2], while low thresholds can lead to the false detection of noise as spikes [2]. A more recent approach is based on template-matching, where waveforms extracted through spike sorting are compared to the signal using cross-correlation to detect spike-like activity [103]. However, these types of approaches often require extensive manual curation, rendering them impractical for datasets recorded using a large number of electrodes.

Feature extraction methods

A couple of concepts have to be defined in order to explain what Feature Extraction does. Feature Selection, as its name indicates, is the selection of relevant features from a given feature space. This is done in order to remove uninformative features, whilst retaining most of the information. Feature Extraction is the generation of new features, usually a smaller set, using the original feature space (i.e. the waveform of the spike), that incorporate the majority of the information. Therefore, the main distinction is that feature selection selects certain features, whilst feature extraction uses the original set of features (the waveform and possibly other existing features) to create new features.

Both of these types of algorithms can be framed under the concept of Dimensionality Reduction, which is done in order to avoid, what is called, the Curse of Dimensionality. The Curse of Dimensionality mainly refers to the struggle of algorithms to identify models because of the high number of features and is particularly relevant to distance-based clustering algorithms. Moreover, by reducing dimensionality (to 2 or 3), the data becomes easy to visualise which is important to attain a better understanding of the data and it also reduces the required processing time.

In a real-world scenario, the spike shape is affected by noise, factors and phenomena that can modify spike shape (e.g. electrode drift) that may lead to overlapping clusters. The objective of feature extraction is to generate a representation that remains separable under minor alterations in waveform shape. This invariance to noise provides the basis for separability of the activity of different neurons. There is no universally adopted feature extraction method, as their performance depends on the specific characteristics of the data being analysed [2]. Figure 2.18 shows an example of how feature extraction can separate spikes from a real dataset in the new feature space, while **Error! Reference source not found.** offers a short description for multiple feature e xtraction techniques highlighting their limitations.

Linear approaches

Principal Component Analysis (PCA) [3,104] is the standard Feature Extraction method used. It is the most common dimensionality reduction algorithm that allows significant reduction of the feature set with minimal loss of information [105,106]. PCA has also been used in spike sorting [107]. PCA creates new features that are called Principal Components. These new features, orthogonal amongst them, are linear combinations of the original features. PCA renders the dimensionality reduction of the feature space into a problem of eigenvalues and eigenvectors, while also trying to preserve most of the variance when reducing the number of features. Most commonly, two or three principal components obtained through PCA are kept [105,106] and they usually conserve more than 70% of the variance of the original features. As expected, when data is separable along the directions with the largest variance, PCA can be a useful

tool. However, variance does not always provide the best separability [2,101]. This implies that the separation is captured by the features that have been discarded, which only have a low variance. PCA along with its variants have been used in spike sorting at length [2] and newly developed spike sorting techniques incorporate them [97].

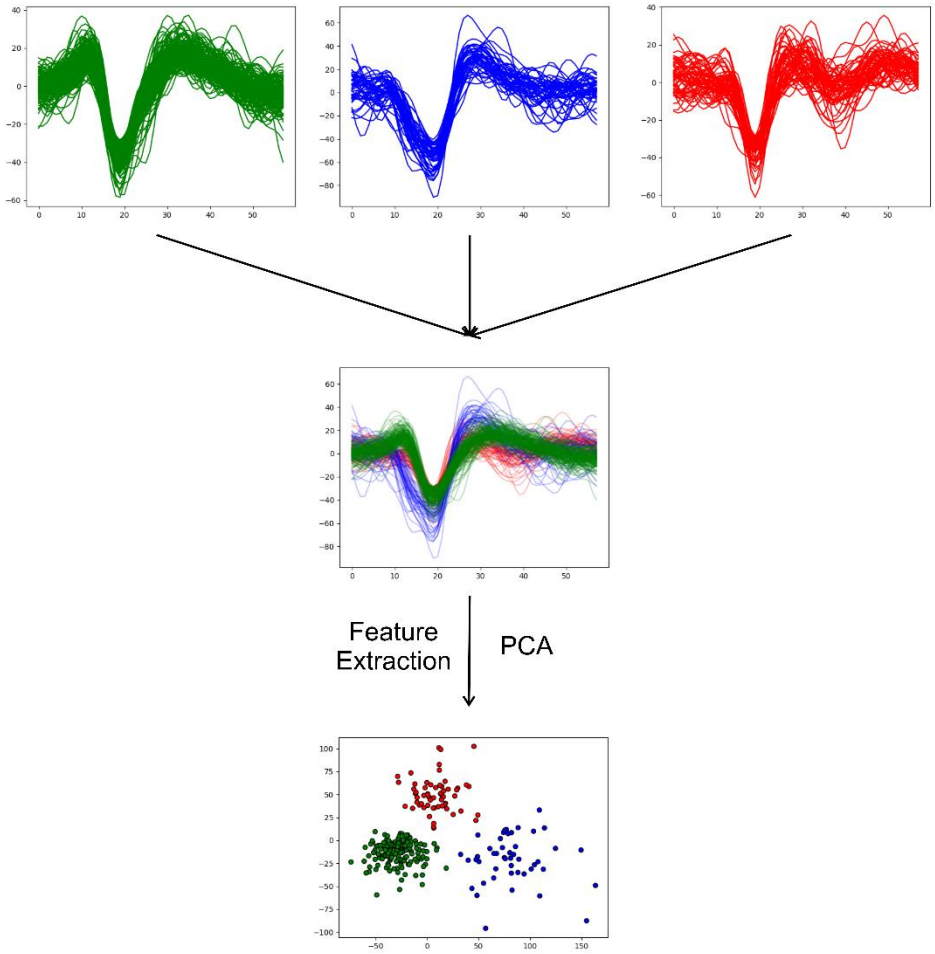

Figure 2.18 – An example of feature extraction process. Spikes produced by three distinct neurons produce separable features in the PCA space.

Independent Component Analysis (ICA) [108] is another common feature extraction technique, which can be categorised as blind source separation method. ICA identifies independent sources, where independent sources are found under the assumption of non-Gaussianity. Within the context of machine learning, it is a linear unsupervised technique that can provide dimensionality reduction by choosing components based on the explained variance criterion or the kurtosis criterion [108]. In contrast to PCA, which aims to maximise variance, ICA's aim is to create independent components. This transpires from its signal processing uses. As a signal can be composed of numerical addition of several sources, the focus of ICA was to identify these independent sources. ICA as a feature extraction technique, has also found uses within spike sorting with encouraging performance [109,110].

Linear Discriminant Analysis (LDA) [111] is a dimensionality reduction technique that finds linear combinations of features that provide separation between different clusters. It is akin to PCA from this perspective as in they both search for linear

combinations. Nevertheless, LDA attempts to model the difference between classes by increasing inter-cluster and decreasing inter-cluster distances. It is a supervised learning technique that assumes a Gaussian distribution of the data. Within spike sorting, LDA is not a suitable candidate on account of multiple factors. Synthetic datasets may allow the evaluation of LDA by providing labels. Nevertheless, spike sorting is unsupervised and ground truth labels are inexistent, while LDA requires the labels of data to function. Additionally, the assumption of Gaussian distribution is the ideal of spike sorting. The irrefutable reality is that this assumption is frequently breached. A subset of issues that impede the generation of Gaussian distributions have been presented above: spike overlap, electrode drift, shape variation. Other effects are multi-unit activity and non-stationary background noise [101].

Non-linear approaches

Isomap [112] is a common low-dimensional embedding method of high-dimensional samples. It is also a non-linear dimensionality reduction method. As for its inner workings, Isomap estimates the inherent geometry of the samples by retaining neighbour distances. Isomap learns the low-dimensional projection of the data through the incorporation of the manifold structure. It uses the geodesic distance, the sum of edge weights along a path over the curved surface of the manifold space, and a neighbourhood graph. It is non-linear which may provide advantages in spike sorting in situations in which clusters are not linearly separable. Furthermore, it is also an unsupervised technique, thus rendering it into a suitable method for comparison.

T-distributed Stochastic Neighbour Embedding (t-SNE) [113], like Isomap, is a non-linear dimensionality reduction method. In contrast to Isomap, t-SNE aims to minimise the Kullback-Leibler (KL) divergence between the low-dimensional projection and the high-dimensional data through the use of pairwise probability similarities. The KL divergence of two distributions is minimised through gradient descent. As such, similar samples become proximal, while dissimilar samples are modelled as distal in the projection. The main function of t-SNE is visualisation. Its time complexity is orders of magnitude higher than that of other feature extraction methods. Through empirical evaluations, it has not been able to provide the separation required for spike sorting.

Clustering Methods

Clustering, from a computer science perspective, becomes intractable as the number of samples and dimensions increase. Clustering through K-Means, which can be considered the most efficient clustering algorithm, is a NP-hard problem [114] and the solution is to use approximation which do not guarantee an optimal solution. Besides the inherent complexity of clustering, neural data brings additional difficulties through their characteristics.

Clustering is the fourth and last step of Spike Sorting that takes the features as input and maps them onto a set of labels that identify the set of spikes fired by one neuron. The Feature Extraction step reduces the number of dimensions while leaving the number of samples unchanged. The clustering step in Spike Sorting belongs to the Unsupervised Machine Learning algorithm that divides a set of points (in the feature space) into groups, called clusters. The main trait of this separation being that objects within a group are more similar, in a certain sense, to the objects of that cluster than to the objects of another

cluster. An illustration is presented in Figure 2.19, while Table 2.3 offers a short description of various clustering methods also highlighting their limitations.

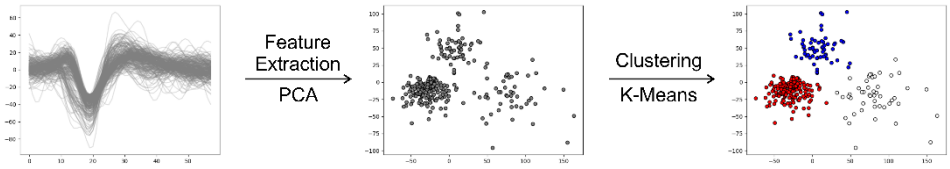

Figure 2.19 - An illustration of clustering on simple data.

There are many algorithms that implement clustering and among these the two most commonly used are K-Means and DBSCAN. A number of clustering algorithms are presented in the following subsections that were used in the comparative analyses. These algorithms have been applied to spike sorting before. In a historical overview of clustering algorithms [115], these clustering algorithms were analysed comparatively in the domain of spike sorting. One of the most recently developed methods, called ISO-SPLIT, achieves the highest performance for the chosen datasets. K-Means, despite being one of the oldest clustering algorithms used, still performs well and ranks third out of the 25 algorithms evaluated. Agglomerative Clustering ranks fifth, Fuzzy C-Means (FCM) ranks seventh, MeanShift ranks twelfth, and DBSCAN ranks last in terms of performance.

Table 2.3 - A descriptive summary of various clustering methods.

Name	Type	Description	Parameters	Challenges
K-Means	Centroid -based	Partitions data into k clusters by iteratively assigning each data point to the nearest centroid and updating the centroids based on the mean of the points in the cluster.	- number of clusters	-identification of cluster number - overlapping clusters - non-convex clusters
Fuzzy C Means	Centroid -based	KMeans extension, where each data point can belong to multiple clusters with varying degrees of membership, which is calculated based on the distances to cluster centroids.	- number of clusters - fuzziness	Same limitations as K-Means.
DBSCAN	Density-based	Groups together densely packed points, thus it separates regions of high density from low density ones.	- maximum distance in neighbourhood - minimum cluster size.	- clusters of different densities - overlapping clusters
HDBSCAN	Density-based	DBSCAN extension that automatically determines the number of clusters and can handle clusters of varying densities. Utilizes a hierarchical approach to cluster assignment.	- minimum cluster size	- overlapping clusters
Agglomerative Clustering	Hierarchical-based	Starts with each data point as a separate cluster and merges the closest clusters	- number of clusters / distance threshold	- high execution time. - noise sensitivity - non-convex clusters - overlapping clusters

		at each step until a stopping criterion is reached.	- linkage method	
MeanShift	Mode-based	Identifies clusters by shifting data points towards the mode of the density function in the feature space.	- bandwidth	- overlapping clusters - non-convex clusters
ISO-Split	Mode-based	Recursively splits the data based on the density peaks, creating clusters by identifying regions of high density.	-	- overlapping clusters

Centroid-based approaches

K-Means [116] is a centroid-based clustering algorithm that divides sample space into K partitions by assigning samples to the cluster with the closest mean. It uses Euclidean distances within a cluster to minimise intracluster variance. K-Means is an iterative process, which repeats two main steps: the first is the assignment of each sample to a single centroid based on distance, whereas the second step is the update of the centroid location to the mean of all the samples assigned to the cluster of the centroid. It has a few disadvantages: it requires the number of centroids as input, it is not deterministic, and has difficulty identifying clusters of arbitrary shapes. In spite of these disadvantages, K-Means is one of the fastest clustering algorithms. It has a time complexity of $O(ndki)$, where n is the number of samples, d is the number of dimensions, k is the number of clusters given as input, and i is the number of iterations. Introduced in 1988 to spike sorting [115,117], it has been intensively used for spike sorting [115] and new clustering and new spike sorting methods are still developed with K-Means as their basis [118,119].

Fuzzy C-Means (FCM) [120] is the progeny of K-Means. In contrast to its precursor, FCM is a soft-clustering algorithm. Hard clustering algorithms, as K-Means, assign one and only one label for each sample of the data, whereas soft-clustering assigns a probability and therefore one sample can be assigned to more than a single cluster. As a descendant of K-Means, FCM has similar inner mechanisms to those of K-Means. It starts with a random initialization of cluster centres. Through iterative updates its cluster centres are shifted to positions that minimise distances. Just as K-Means, it requires the number of clusters as an input parameter which can become a disadvantage when the number of clusters in the data is not known. Even nowadays, FCM variations are being developed to improve its performance for specific tasks in clustering applications, such as image segmentation pipelines.

Density-based approaches

DBSCAN [121], or Density-Based Spatial Clustering of Applications with Noise, is a density-based clustering algorithm. It starts by identifying the cores of clusters as regions with high densities and expands them. Therefore, it defines clusters as high-density regions while low density regions are labelled as noise. It can be inferred that, despite its name, DBSCAN has a high performance only if clusters of similar densities are provided. It has difficulties in separating overlapped clusters and especially embedded ones. Unlike K-Means, DBSCAN does not require the number of clusters as input. In addition, DBSCAN is able to identify clusters with arbitrary shapes and is a mostly

deterministic algorithm. DBSCAN's main disadvantage is its inability to distinguish between different densities of clusters, a significant issue in Spike Sorting due to varying firing rates of neurons. DBSCAN has a time complexity of $O(n^2)$, where n is the number of samples, while the memory complexity of DBSCAN may vary between $O(n)$ and $O(n^2)$ depending on the implementation. This complexity appears from the necessity of computing distances among neighbourhoods. DBSCAN has also found uses within the domain of spike sorting [115].

HDBSCAN [122], as the name suggests, is a modification of the DBSCAN algorithm into a hierarchical approach. Like DBSCAN, it identifies dense groups as clusters and labels sparser groups as noise. Its approach is to associate points as a weighted graph with connections made based on a minimum spanning tree that can be built from algorithms such as Prim's greedy method. Similar to its precursor, it does not require the number of clusters as an input parameter. The most definitory parameter is the minimum cluster size that dictates the lowest number of points that need to be close enough for a cluster to form. The time complexity of the algorithm remains identical, $O(n^2)$, where n represents the number of samples in the dataset. The space complexity of HDSCAN is $O(dn)$, where d represents the number of dimensions (or features) of a sample.

Hierarchical-based approaches

Agglomerative Clustering [123] applies a "bottom-up" approach to hierarchical clustering. Initially, each sample is assigned to an individual cluster. As the algorithm progresses through its iterations these clusters are being merged. Pairs of clusters are merged based on a proximity matrix, which contains distances between points. Agglomerative Clustering uses a linkage function to merge clusters based on distance information. One type of linkage is the ward linkage which analyses cluster variance rather than directly analysing distance. Its main disadvantage is that it requires the number of clusters as input. In addition, its complexity is demanding as well. Considering a dataset with n samples, its time complexity is $O(n^3)$ and its space complexity $O(n^2)$. As spike sorting data can reach high volumes, its taxing complexity can become a challenge. Moreover, the overlap inherent to spike sorting data may also yield unsatisfactory clustering, which could be approached with adequate feature extraction.

Mode-based approaches

MeanShift [124,125] is categorised as a mode-seeking algorithm. It offers a mathematical analysis for the localization of the maxima of a density function. From a clustering perspective, it identifies clusters by the update of centroid candidates to the mean of regional points and in this way, it is similar to K-Means as its geometry is also based on distances between points. Mean Shift is an iterative process that shifts this kernel to a high-density region where it reaches convergence. Considering a dataset with n samples, the time complexity of MeanShift is $O(n^2)$. In contrast to K-Means and FCM, it does not require the number of clusters as an input. Within the context of spike sorting, MeanShift has difficulties with overlapping clusters and can result in underclustering. Nevertheless, this can be circumvented by the feature extraction step and additionally, MeanShift can tackle a high number of clusters and imbalance.

ISO-SPLIT [126] was designed specifically for spike sorting. It operates on the assumption of unimodality of clusters which is determined through iterative isotonic regression. ISO-SPLIT was shown to outperform traditional approaches such as K-Means

or Gaussian Mixture Models in a variety of datasets [115,126]. Its most appealing qualities are that it does not require any parametrization, i.e. it does not require the number of clusters to be known apriori, and that it can deal with non-gaussian clusters as well.

Other approaches

Spectral clustering (SC) [128] converts the data into a graph representation in which nodes stand for data samples and edges for the connections between them. A similarity matrix is created from the data, and based on the eigenvalues and eigenvectors, the graph is divided into K clusters. Due to its reliance on distance measures in the original feature space, SC is able to capture nonlinear relationships, thus clusters of any shape, more effectively than K-Means by using space transformation.

SC is a multi-step process. Using metrics like Gaussian kernel similarity or k-nearest neighbours, a similarity matrix is built based on pairwise similarities between data points which is used to calculate the graph's Laplacian matrix. Next, the Laplacian matrix's eigenvectors that match its smallest eigenvalues are extracted. The data is embedded into a lower-dimensional space using these eigenvectors, which also encode the data's underlying structure. Lastly, K clusters are created by applying common clustering algorithms to the embedded data, like K-Means [116] or normalised cuts [127]. The similarity matrix of all points and the Laplacian matrix's eigenvalue decomposition computations yields an approximate $O(n^3)$ time complexity, where *n* represents the number of samples. The eigenvalue decomposition step may make it computationally expensive, especially when working with large datasets.

Clustering performance metrics

Clustering performance metrics can be categorised into two types: external and internal. External metrics require both the ground truth labels and the clustering labels. They compare, through different statistical methods, the correlation between the labels resulting from the clustering algorithm and those of the ground truth in order to evaluate the correctness of clustering. External metrics are generally not usable in the neuroscience domain as ground truth labels are not available. In general, external metrics (Purity is an exception) have the disadvantage of punishing overclustering which can usually be solved through post-clustering merging.

Internal metrics require the dataset and a set of labels, these metrics analyse the shape of the clusters, their intracluster and intercluster distributions, and other such characteristics in order to assess that the clusters defined by the labels fulfil the assumptions of a standard cluster. Table 2.4 offers a short description for multiple performance metrics, indicating the limitations of each along with other features. From a computer science perspective, internal metrics are restrictive with regards to the characteristics of the clusters obtained, convex cluster that have high distances between clusters (as in no overlap) will receive higher scores, even if a clustering algorithm is capable of perfectly separating overlapping clusters.

Table 2.4 – A descriptive summary of various clustering performance metrics.

Name	Type	Description	Range [worst, best]	Challenges

ARI	External	Pair-by-pair comparison whether the points in the predicted cluster belong in the same true cluster.	[-1, 1]	Require knowledge of ground truth (inexistent for real data)
AMI	External	Mutual information based on entropy is used to calculate the agreement of true and predicted labels.	[0, 1]	
FMI	External	Pairwise comparison between true and clustered data points.	[0, 1]	
Purity	External	Cluster homogeneity as the majority class assignment.	[0, 1]	
VM	External	Harmonic mean of conditional entropies between the true and predicted clusters.	[0, 1]	
DBS	Internal	Ratio of the inter-cluster and intra-cluster sum of squared distances.	(Inf, 0]	Higher values for convex and separated clusters, even if overlapping clusters are identified correctly
CHS	Internal	The average of a function that evaluates inter-cluster distances and the size of the cluster	[0, Inf)	
SS	Internal	Cluster quality is evaluated as the balance between a cluster's tightness and separation	[-1, 1]	

External metrics

As mentioned previously, external metrics require both the ground truth, or true labels, and the clustering labels, or predicted labels. The metrics discussed next are bounded and higher values correspond to better clustering.

Adjusted Rand Index

Adjusted Rand Index (ARI) [127,129–132] is an external metric derived from the Rand Index (RI) metric. RI makes comparisons between pairs of labels, one from the true labels and one from the predicted labels whilst ignoring permutations. Comparisons are made between pairs of labels to determine whether they are part of the same cluster in both true and predicted labels. Two cases can be distinguished: when the two are in the same cluster, called an agreement, and when the two belong to different clusters, called a disagreement.

The RI score is computed as the division of the total number of agreements to the sum of agreements and disagreements. This sum of agreements and disagreements is also the equivalent of the total number of pairs. Therefore, RI can be viewed as the probability of the two sets of labels to agree on a randomly chosen pair. Taking this into consideration, RI is defined as shown in Equation (20).

$$RI = \frac{agreements}{agreements + disagreements} \tag{25}$$

Starting from RI, ARI uses an expected index E to adjust for agreements made by chance in order to ensure a consistent score for random labelling:

$$E\left(\sum_{i,j}\left(\frac{n(i,j)}{2}\right)\right) = \frac{\sum_i\left(\frac{n(i)}{2}\right)\sum_j\left(\frac{n(j)}{2}\right)}{\left(\frac{n}{2}\right)} \tag{26}$$

With the expected index, ARI is computed as:

$$ARI = \frac{RI - ExpectedRI}{MaximumRI - ExpectedRI} \tag{27}$$

ARI scores are bounded within the [-1, 1] interval, with negative values meaning individual assignments and values close to 1 meaning correct clusterings. The advantage of using ARI is that it allows for the identification of random labelling that correspond to values close to 0.

Adjusted Mutual Information

Adjusted Mutual Information (AMI) [133,134] is an external metric derived from the Mutual Information (MI) metric. Moreover, AMI also has the normalisation step of Normalised Mutual Information [135,136]. MI is based upon the concept of entropy (H) defined in Equation (23) which is based upon probability.

$$H(U) = -\sum_{i=1}^{|U|} P(i)\ log\left(P(i)\right) \tag{28}$$

where $P(i)$ is the probability that a sample of U falls into the class U_i, calculated as the division of the number of samples that have this property (defined as $|U_i|$) to the total number of samples (defined as N).

By defining U and V as two label assignments and by deriving the formula of entropy, the formula of MI is obtained in Equation (26):

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\ log\left(\frac{P(i,j)}{P(i)P(j)}\right) \tag{29}$$

where $P(i, j)$ is the probability of a sample being a part of both U_i and V_j. This is the equivalent of the division of the number of samples of the intersection of U_i and V_j to the total number of samples. Thus, the formula of MI can be written also as:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|Ui\ \cap\ Vj|}{N}\ log\left(\frac{N\ |Ui\ \cap\ Vj|}{|Ui\ ||\ Vj|}\right) \tag{30}$$

Using the expected index E, MI can be adjusted for chance to guarantee a consistent score for random labellings defined as AMI:

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{mean\{H(U),H(V)\}\ - E\{MI(U,V)\}} \tag{31}$$

AMI allows for the identification of random labelling through its adjustment for chance by resulting in values close to 0. This metric is bounded within the [-1, 1] interval, with negative values meaning individual assignments and values close to 1 representing correct clustering.

Purity

Purity [137,138] is an external metric and it computes the percentage of samples clustered correctly. This is computed as the division of the sum of the maximum intersections between the true and predicted labels for each cluster by the total number of samples:

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max|C_i \cap L| \tag{32}$$

Where N represents the total number of samples in the dataset, k is the number of clusters in the set of predicted labels, C_i represents the samples of a cluster, i, of the predicted set of labels and L is the set of true labels. Thus, Purity can be viewed as a measure of how many of the samples of the predicted cluster belong to a single true cluster.

Purity is bounded between the [0, 1] interval, where 1 represents a perfect clustering. By definition, purity does not punish overclustering and this is its disadvantage. By assigning each point to a different cluster, a score of 1 is obtained. Therefore, the correctness of the clustering does not consider the number of clusters. Moreover, imbalanced clusters tend to receive high scores as clusters with a lower number of samples receive higher scores in most cases.

Fowlkes-Mallows Index

Fowlkes-Mallows Index (FMI) [139] is an external metric that can be defined as the geometric mean of the pairwise precision and recall, described by the following formulas:

$$FMI = \sqrt{Precision * Recall} = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}} \tag{33}$$

FMI makes no assumption of cluster structure as it only uses the number of samples that belong to a set of labels. It is bounded within the [0, 1] interval. Random labelling receives a value of 0, as the label assignments are independent, while a value of 1 represents significant agreement between the set of true and the set of predicted labels.

V-Measure, Homogeneity and Completeness

V-Measure (VM) [140] is an external metric defined as the harmonic mean of Homogeneity and Completeness. The latter can be weighted through the *beta* parameter: values higher than 1 give more weight to completeness, while values lower than 1 to homogeneity. VM is defined by the following formula:

$$VM = \frac{(1 + beta) * (Homogeneity * Completeness)}{(beta * Homogeneity + Completeness)} \tag{34}$$

Homogeneity and completeness are both defined as conditional entropy of the class distribution, as follows:

$$homogeneity = 1 - \frac{H(U,V)}{H(U)} \tag{35}$$

$$completeness = 1 - \frac{H(V,U)}{H(V)} \quad (36)$$

Where *U* and *V* are label assignments, and the formula of entropy (*H*) is defined above for AMI.

Within this context, classes refer to true labels and clusters to the predicted labels. A cluster is considered homogeneous if and only if all samples of a cluster belong to a single class (as defined by the true labels), while a cluster is considered complete when all samples of a class (as defined by the true labels) have been identified as belonging to the same cluster. Both homogeneity and completeness are bounded within [0, 1] as VM is, with values closer to 1 representing a better clustering. No assumption of cluster structure is made, and random labelling does not yield zero scores for a high number of clusters due to the fact that neither homogeneity or completeness are normalised and consequently, neither is V-Measure.

Internal metrics

Internal metrics require the dataset and a set of labels, which can be the true labels or a clustering assignment. Nevertheless, these metrics do not require a ground truth, they are most commonly used to evaluate clustering with respect to certain characteristics such as: intracluster and intercluster distributions, the shape of clusters and the separation of clusters. Clusters that respect the standard concept, i.e. well separated convex clusters and with high density, receive higher scores. Due to the fact internal metrics require only one set of labels, they can have a dual purpose. They can be used to evaluate a clustering assignment but also the ground truth. When evaluating with a clustering assignment, these metrics evaluate the ability of the clustering algorithm to separate the data based on its structure into cohesive clusters. When evaluating the data with the true labels, these metrics can be used to evaluate the structure of the data and its separability into clusters, this can be especially useful to evaluate how much separability a feature extraction algorithm can induce into the new feature set provided.

Calinski-Harabasz Score

Calinski-Harabasz Score (CHS) [138,141], also known as Variance Ratio Criterion, is an internal metric defined as the ratio between the mean of intercluster dispersion and intracluster dispersion, where dispersion is defined as the sum of squared distances:

$$CHS = \frac{tr(B_k)}{tr(W_k)} \frac{n-k}{k-1} \quad (37)$$

Where *tr(B*) is the trace of the intercluster dispersion matrix, *tr(W)* is the trace of the intracluster dispersion matrix, *n* is the number of samples of a dataset clustered into *k* clusters.

CHS is only lower bound at 0 with higher scores representing better clustering.

Davies-Bouldin Score

Davies-Bouldin Score (DBS) [138,142,143] is an internal metric computed as the mean similarity of a cluster and the cluster most similar to it. In this case, similarity is defined as the ratio of the distances in a cluster and the distances between clusters:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{38}$$

where, R_{ij} represents the similarity between a cluster *i* and its most similar cluster *j*, *s* is the mean distance between any point of the cluster and its centroid, while d_{ij} is the distance between the centroids of cluster *i* and *j*.

Finally, the score is computed as the arithmetic mean of the maximum similarity *R* of pairs of clusters *i* and *j*. As this metric only uses distances between samples to evaluate correctness, its evaluation is exclusively dependent on inherent characteristics of the dataset.

$$DBS = \frac{1}{k}\sum_{i=1}^{k} \max\left(R_{ij}\right), where\ \forall\ j \in\ [1,k]\ and\ i \neq j \tag{39}$$

where, *k* is the number of clusters.

DBS is lower bounded at 0, with 0 corresponding to better separation between clusters. DBS also has higher scores for convex clusters than for other types.

Silhouette Score

Silhouette Score (SS) [138,144] is an internal metric that is computed for each sample of the dataset and it is composed of two other scores *b* and *a*. SS computes the distances between a point and all other samples within a cluster and the distances between a point and all samples of a different cluster. The formula by which the score of one sample is computed is:

$$s = \frac{b - a}{\max(a, b)} \tag{40}$$

where, *b* is the mean distance between a sample and the closest different cluster, and *a* is the mean distance between a sample and the other samples of that cluster. The SS of the dataset is computed as the average of the score of each sample.

SS is bounded within the [-1, 1] interval, where -1 is considered incorrect clustering as samples would be better suited to be assigned to another cluster, 1 is a dense clustering as intracluster distances are low while intercluster distances are high, and values close to 0 represent overlapping clusters.

Spike sorters

Spike sorting algorithms involve multiple steps of the spike sorting pipeline: spike detection, feature extraction, clustering, cluster merging, and potentially even more. Depending on the specific algorithm, they may include additional steps or skip some steps from the standard spike sorting pipeline. Several spike sorting algorithms have been developed over the years, with notable examples being KiloSort [119], SpykingCircus [145], and WaveClus [99].

KiloSort [119] is an automated spike sorting pipeline that is specialised in the analysis of recordings originating from high-density electrodes in real-time. It offers the option for human intervention through a manual user interface for post-processing

curation. Spike detection in KiloSort is performed using template matching instead of the classical amplitude thresholding, and spike prototypes are stored based on the L2 norm difference. KiloSort utilises these spike templates to initialise a modified version of K-Means that employs a custom loss function, which is designed to be invariant to amplitude changes in spikes. One of the main advantages of KiloSort is its computational efficiency, achieved through the use of mathematical models for spike template creation. Notably, KiloSort encompasses more steps in the spike sorting pipeline than clustering such as template matching for spike detection and feature extraction.

SpykingCircus [145] is another automated spike sorting pipeline designed with the purpose of analysing high-density multielectrode arrays. It employs a fast approximate template matching strategy to identify spikes by comparing them to set of predefined templates representative of the waveform shapes of different neurons. To define the templates, a new clustering approach was developed and applied on a subset of all spikes through the use of a smart sub-sampling approach, where spikes are detected as threshold crossings. Additionally, SpykingCircus applies whitening to the data to remove noise and spatio-temporal correlations between channels for overlapping waveforms.

WaveClus [99] is an automated spike sorter developed for extracellular recordings. This sorter applies the classical spike sorting pipeline using a threshold approach for the detection of spikes. For the feature extraction method, it uses a multiresolution decomposition using wavelets, while the clustering is done through the Superparamagnetic Clustering (SPC) algorithm. Additionally, the remaining waveforms that have not been assigned to a cluster are then introduced into a template-matching approach to be assigned to a cluster.

2.3.2. Burst detection

The ISIn [146] operates on the sorted array of timestamps, that represents the occurrences of action potentials at a given time, as its input data. This method uses two parameters regarding the ISI for the detection of bursts. A parameter *N* determines the responsiveness of the algorithm, and it represents the number of spikes between which the ISI is computed. The algorithm computes the time interval between a first spike and the following *N*-th spike, thus for instance, for a value of 2, the algorithm analyses the time interval between a first spike and the immediate next one. The second parameter is a threshold, for a burst to be initiated, the ISI between a given spike and the following *N*-th spike must be below this given threshold. A histogram can be computed, usually in the logarithmic scale with smoothing, to visualise the distribution of ISIs to help establish these parameters. Through iteration of the spiking data, consecutive spikes that have smaller ISIs than the threshold are considered to be a part of the same burst. Thus, the presence of burst activity is indicated by a high frequency of action potentials that are temporally proximal. This approach provided several advantages including a simple implementation, no need for ad-hoc or post-hoc criteria, and precise assignment of burst boundary time points. Unlike existing approaches, detection was not biassed toward longer bursts and could correctly detect the presence of shorter bursts.

The ISI Rank Threshold (IRT) [147] operates on the timestamp data sorted in ascending order. This method involves the computation of the interspike intervals (ISIs) by subtracting consecutive timestamps. Each of the computed ISIs receive a rank relative to the largest ISI in the given spike data, for example the smallest ISI receives the rank of

1. The IRT method uses two thresholds, a given rank threshold and a computed spike count threshold based on a given input. The probability distribution of spike counts is computed using one-second bins in order to calculate the spike count cutoff in such a way that the probability of observing a number of peaks exceeding the threshold is kept below a specified probability limit. A burst is identified if the ISI at a given time point has a lower rank than the rank cutoff, indicating a transition from the inter-burst to a bursting event, and the number of spikes contained in the following one-second timeframe surpasses the spike count cutoff. The burst is considered to have stopped when the spike count in a one-second interval falls below the spike count cutoff divided by two.

The Max Interval (MI) [148] method necessitates the following five parameters that define the characteristics required for burst detection: maximum start interval of an ISI, the maximum end interval of an ISI, the minimum inter-burst interval (IBI), the minimum burst duration, and the minimum number of action potentials of a burst. Chronologically sorted timestamps are the required input for this algorithm. By iterating through these timestamps, a burst is detected when the interval between two consecutive action potentials exceeds the maximum start interval parameter of an ISI. Conversely, a burst is considered terminated when the interval between two action potentials surpasses the maximum end interval parameter of an ISI. IBIs are calculated by subtracting the first timestamp of a current burst from the last timestamp of the preceding burst and are utilised for the merging of bursts. If the combined duration of neighbouring pairs of bursts is smaller than the minimum inter-burst interval, they are merged to encompass both bursts. Subsequently, the algorithm iterates through the resulting bursts, assessing whether they satisfy the conditions of the minimum burst duration and the minimum number of action potentials. Bursts that do not meet these criteria are not considered for further analysis. In conclusion, the Max Interval algorithm is rather simple as any spike train that satisfies the five given thresholds is considered a burst.

The Cumulative Moving Average (CMA) [149] method utilises the cumulative sum of ISIs within a sliding window to detect bursts. The input data consists of a sequence of timestamps that indicate the occurrences of action potentials. The time interval between consecutive action potentials, or ISI, is calculated by subtracting the corresponding consecutive timestamps. The histogram of ISI is computed, and the cumulative moving average is calculated as the sum of the bins of the ISI histogram divided by the bin number. The skewness of this distribution determines the values of the thresholds, $\alpha 1$ and $\alpha 2$, used in the detection of bursts. The maximum value of ISI is considered to be found at the bin which has the closest value of maximum CMA multiplied by $\alpha 1$. Bursts are identified as any spike train of a minimum of three spikes where each ISI is lower than the maximum ISI found. The authors suggest a second cutoff, which is used to extend the bursts to include burst-related spikes and it is computed as the maximum cumulative moving average multiplied by $\alpha 2$.

The Rank Surprise (RS) [150] operates on the concept that bursts exhibit an elevated firing rate of a neuron, which can be detected by comparing the observed ISI with a distribution. Unlike the Poisson Surprise method, RS focuses on the values of the inter-spike intervals rather than the firing rate. The intervals are sorted in ascending order, and each interval is assigned a rank, starting from 1 for the smallest ISI found with ties given the same rank. A fixed threshold for *maxISI* is determined based on the probability distribution of ISIs in the spike train. Bursts are selected such that the

surprise statistic is maximised. The algorithm searches for the first sequence of at least three consecutive spikes with all ISIs smaller than this *maxISI*. Within this sequence, all possible subsequences of ISIs are iterated through to identify the subsequence that yields the highest surprise value. If the surprise value exceeds the predetermined minimum threshold given by a parameter, it is considered a burst.

The Poisson Surprise (PS) [151] method was developed upon the assumption that bursts are characterised by an increase in the firing rate of a neuron and the firing rate of tonic activity follows a Poisson distribution. The algorithm computes the expected firing rate based on a Poisson distribution and the surprise value as the probability of a number of spikes to occur in a given time given a Poisson process. Initially, bursts are identified as sequences of three consecutive spikes with inter-spike intervals (ISIs) shorter than half of the mean ISI observed in the spike train. The algorithm maximises the surprise statistic by iteratively adding to the end and removing from the beginning spikes of each identified burst. This process aims to refine the burst definition by selecting the subset of spikes within the initial burst that exhibits the highest deviation from the expected firing rate. Any bursts found that have a surprise value below the predetermined threshold are discarded, as they do not exhibit a significant deviation from the expected firing rate and are therefore not considered bursts.

2.3.3. Symbolic Analysis

The technique of Colour Sequences was introduced in [152], as a visualization tool for obtaining representations of multiple spike-trains. As its name implies, the visual exploration of the data set is done using sequences of colours that encapsulate the characteristics of input data in a time defined context. It aims to detect meaningful patterns of the activity of neurons during the evolution of the recording.

The working principle of Colour Sequences relies exclusively on a three-dimensional Self-Organising Map for localizing and obtaining the colour label of each sample from the input data set. Colour assignment is performed depending on the position of each model vector in the self-organised map. Each dimension of the map defines the intensities of red, green and blue channels respectively, used in the composition of final RGB values of the colour label. In this way, the space of model vectors envisions an RGB subspace delimited by the size of the self-organising map. All colours composing a colour sequence of a trial reside in this subspace and are found using the same process. In order to obtain the final representation of a colour sequence, the colours of all samples in a trial are placed one after the other where each colour band indicates a unit timestamp of 1ms. An individual colour sequence may not be useful for obtaining meaningful information on the possible brain activity patterns. Nevertheless, groups of colour sequences organized according to a specific criterion, could point out regions with similar patterns. In the case of multi-electrodes spike trains several patterns could be identified by employing such a visualization method.

The large amount of visual information conveyed by colour sequences plots of high dimensional data sets cannot be fully comprehended only by visual colour inspection. Some meaningful patterns can appear by themselves multiple times throughout colour sequences groups, rather than in regions of patterns at certain timestamps. Pattern Specificity Index (PSI) can perform an in-depth colour pattern analysis by ensuring the capture of all the patterns related to a predefined relevant threshold value. The index is computed for a specific pattern, p, in the context of a

stimulus, s, by dividing the number of occurrences of p during trials corresponding to stimulus s, by the number of occurrences of p across all stimuli:

$$PSI_{p,s} = \frac{count(p|stim = s)}{\sum_j count(p|stim = j)} \tag{41}$$

Event Triggered Average (ETA) is a commonly used measure in various domains, employing different names but having an equivalent meaning and representation. This measure has the purpose of observing a certain process around the time interval an event took place. In neuroscience, the Spike-Triggered Average [5] describes how another signal, such as the local- field potential, evolves around individual spikes. In this paper, the Pattern-Triggered Average (PTA) is used to evaluate what happens, on average, with the EEG signal around the moments when a pattern appears. The general idea of computing the event triggered average of a repeating signal is to observe it in a time-defined interval by taking a window of values before and after the timestamp of occurrence. For every signal, these values are added in an arrayof size *2*window_size+1* at the corresponding positions, which are then divided by the number of signal occurrences.

The Peri-stimulus Time Histogram (PSTH) represents an analysis tool used in neuroscience in order to obtain a visual estimation of the rate at which certain patterns of neural activity occur over time. PSTH divides the time axis into small bins of width δ and then counts the frequency of each pattern within the bin. As a result, it establishes a relationship between the occurrences of a certain pattern and time, providing information on the temporal dynamics of a meaningful pattern across stimuli. From a statistical perspective, the classical PSTH is considered a model approximating the Poisson process intensity through bins with width of duration δ. PSTH aims to provide an estimation on the number of occurrences of an event between two timestamps given by the width of the bin.

2.3.4. Detection of oscillations

OEvents is an algorithm for the detection of brain oscillations [153], which operates not on individual frequencies but rather on the whole time-frequency plane where it detects oscillation bursts as local maxima. Bursts span over time and frequency to an extent defined by per-frequency thresholds. Finally, each burst is localized within a rectangular bounding box and is quantified by a set of parameters such as, extent, number of cycles, dominant frequency, and power, etc. In this particular instance, the wavelet TFR was used, and frequencies were equalized in a pre-processing step, which is one of the main strengths of the algorithm and the reason why the algorithm was able to operate up to 200 Hz. This is by far the most successful algorithm to date and has been used to characterize oscillation bursts in intracranial recordings from humans and monkeys. Nevertheless, there is room for improvement. One of the downsides of OEvents is that the fine spatial structures of the oscillations are lost. Rectangular bounding boxes fail to properly characterize the rich repertoire of shapes found in TFRs.

OEvents is designed to identify and analyze oscillation packets in electrophysiological signals by inspecting TFRs. Originally, OEvents was developed on wavelet based TFRs using 7 cycle Morlets. However, the algorithm is readily usable on other TFRs as well. At the core of OEvents are adaptive thresholds. It first computes the

statistics of each frequency of interest across the whole dataset. Next, a local maximum filter is used to detect peaks in the spectrogram, per frequency. Peaks exceeding 4 times the median are considered as moderate power to high-power events. The authors defined a local power peak within a 3x3 window and then seek the time-frequency bounds around it by expanding the peak as long as the power is above a threshold, which is defined as 50% of the peak value or 4x the median, whichever is lower. Next, packets with overlapping areas greater than 50% of the minimum area of each individual event are merged together. Finally, OEvents computes various packet features including frequency span, time span, peak frequency, and other customizable features. The algorithm has been shown to reliably detect the number of cycles and peak frequency of oscillation events with high accuracy for most frequency bands, and at multiple event durations.

3. Methods for brain activity identification

Neurons transmit information through electrical impulses that travel through their axon to adjacent neurons. To further complicate this image, two modes of function have been found when considering multiple instances of activity: the tonic mode and the burst mode. These two modes of function are different from the perspective of the time interval between the instances of activity of the same neuron. Tonic activity is the regular mode of function and most activity seen in the brain is of this type. Instances of neuronal activity in this mode tends to have larger time intervals between them, typically larger than the refractory period. In contrast, when neurons enter burst mode, there is no refractory period as neurons are not able to return to their resting state before the next action potential. Moreover, the shape of these instances of activity tend to get distorted as the burst train progresses. It may seem at this point that they can be easily distinguished, and they can be but only when recording a single neuron. If multiple neurons are recorded at the same time, as is the case in extracellular recordings, it is important to determine whether an interval of high activity represents the synchronized activity of multiple neurons or the burst of a single neuron.

This chapter explores computational methods for the identification of tonic and burst types of activities. The first subchapter proposes various original computer science methods that can improve the identification of tonic activity, while the second subchapter proposes an original method to identify bursting activity and a analytic method to validate the correctness of this identification. From a scientific perspective, the correct identification of tonic activity and bursting activity can improve our understanding of brain functions such as visual processing, while burst activity has been linked to encoding and learning. Moreover, anomalies in bursting have been connected to certain medical conditions such as epilepsy and schizophrenia.

3.1. Data analysis in spike sorting

3.1.1. Introduction

To address the limitations derived from the complexities of spike sorting, this subchapter proposes several original approaches in which the spike sorting pipeline can be improved are presented. Beside the filtering, each step of the pipeline has been addressed and improvements have been proposed in the form of new approaches in spike sorting or even new algorithms. Neural network-based approaches are proposed for the improvement of the standardised threshold method for the detection of spikes. The discrimination of tonic and bursting activity has been tackled, a subject that is rarely addressed. New approaches have been explored and validated for the feature extraction step exhibiting improved performance compared to classical methods. A new clustering algorithm specifically designed for spike sorting is proposed that is capable of handling the challenges of neuronal data that traditional algorithms struggle with. A new clustering performance metric designed for spike sorting data is proposed.

From the challenges presented, a hypowork emerges. The development of novel algorithms for spike sorting, specifically focusing on feature extraction and clustering techniques, can significantly enhance the performance of spike sorting. As the classical approach is based on a simple amplitude threshold for the detection of spikes which may

ignore a part of the neural activity that is below this threshold, spike detection can also be improved.

Feature extraction algorithms were designed to extract relevant temporal and spectral features from neural recordings, capturing the intricate patterns inherent in neuronal spike waveforms. The aim was to research and develop methods that can effectively discriminate between different spike waveforms producing new feature sets that offer maximum separability for the clustering step of spike sorting. Another objective is to explore clustering techniques for grouping these extracted features into distinct clusters representing the spikes of individual neurons. Traditional clustering methods face challenges in handling separating overlapping clusters and high-dimensional data, and with enough dimensions the problem can become intractable. Therefore, the feature extraction is an important step that can not only reduce dimensionality, but it can also create more informative features at the same time.

Newly developed algorithms within the domain of spike sorting must be highly performant and computationally inexpensive; thus, a critical aspect of this work involves the rigorous evaluation and validation of the proposed algorithms. Comprehensive experiments were conducted using both simulated and real neural datasets to assess the performance of the developed methods. Multiple evaluation metrics were used to provide insights into the strengths and limitations of the algorithms compared to existing spike sorting techniques. As the evaluation has been done on real neural datasets, the developed algorithms can be seamlessly integrated into practical spike sorting pipelines, ensuring their usability.

3.1.2. Data

Benchmark synthetic spiking datasets

Synthetic spike data was used to quantify the performance of different clustering algorithms. The data was generated within the Department of Engineering, University of Leicester UK. It contains 95 simulations (datasets), created based on the characteristics of a real “in vivo” dataset recorded from the monkey neocortex. The whole synthetic dataset contains 594 different shapes of spikes. This synthetic dataset contains the ground truth for each spike in each simulation, meaning that each spike has the true label, the correct cluster it should be assigned to [45].

Originally, the data was sampled at a frequency of 96KHz, resulting in waveforms of a length of 316 samples, these were afterwards downsampled to a frequency of 24KHz, which also reduced the length of the waveforms to 79 samples. Therefore, in this dataset, each spike is described by a signal of 79 values, or features, resulting into a 79-dimensional feature space.

The simulations have varying numbers of clusters from 2 to 20, with five simulations for each unique number of clusters, resulting in 95 simulations. Within this dataset, clusters represent a single-unit and are emulating the spikes of neurons within 50μm from the electrode. The amplitudes of single-unit cluster have been set to follow a normal distribution and have been scaled between 0.9 and 2, whereas the firing rate of these neuron models have been set to follow a Poisson distribution with a mean between 0.1 and 2Hz. In order to simplify the problem, no (time) overlapping spikes have been generated. Each spike is separated from another by an interval of at least 0.3ms [45].

Additionally, in order to increase the complexity of clustering problem, a multi-unit cluster was introduced. This multi-unit cluster incorporates 20 different spike shapes to emulate the characteristics of the noise in real data. The neuron models that fire the spikes within the multi-unit cluster are between 50 to 140µm from the electrode. Neuron models that are further away from the electrode, more than 140µm, were considered to be too far away to be identified by the spike detection. The amplitudes of multi-unit clusters have been scaled to 0.5 and have been set to follow a normal distribution. The value of 0.5 was chosen to close to the detection threshold. The firing rate was set to 5Hz and for the 20 neuron models in the multi-unit cluster the firing rate was set according to a Poisson distribution with a 0.25Hz average firing rate [45].

The availability of the ground truth allows different perspectives on the spikes. In theory, all the spikes of a neuron have similar shapes. That is the spikes of a cluster should be similar as can be seen from the left side of Figure 3.1. By contrast, each spike presented in the right side of Figure 3.1 belongs to a different cluster. A lot of variability can be observed between the spikes produced by different neurons. Figure 3.1 shows spikes extracted from one of the simulations presented, showing how spikes from the same neuron model and from different neuron models look in these datasets.

The synthetic datasets [45] were specifically designed to contain no overlapping waveforms. While this simplification does not capture the complexities of real spike sorting data, the study describing the datasets aimed to assess the performance of clustering algorithms. Even with the simplifications in place, including the absence of overlapping waveforms and the presence of a single multi-unit cluster, it was found that no clustering algorithm was able to identify more than 8-10 clusters out of a maximum of 20. This suggests that precise identification and separation of distinct clusters in spike sorting remains a challenging task, even with simplified data characteristics.

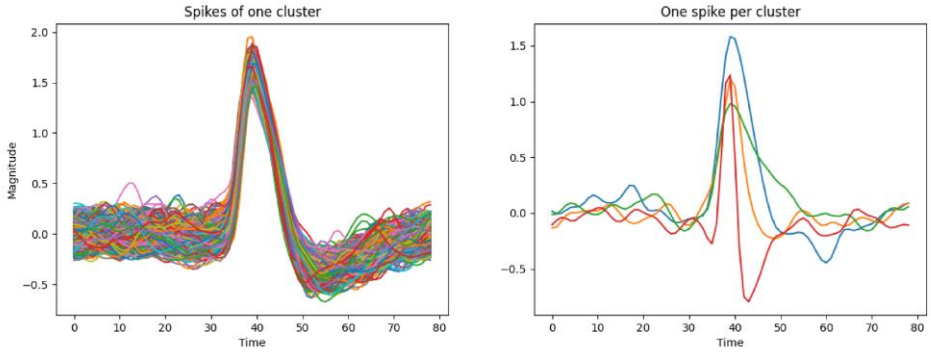

Figure 3.1 - (Left) Synthetic spikes of one neuron model show that even though some variability exists, they are very similar. (Right) Each synthetic spike is from taken randomly from all spikes of a different neuron model showing that there is a high amount of variability between the spikes of different neuron models.

When preprocessing the data, different techniques can be applied depending on the feature extraction method. A few examples of simple preprocessing techniques are: normalisation, scaling and alignment. All of these can have dramatic effects on the performance. Take alignment as an example, it is a simple shifting of all samples in order to line up a common component at a chosen index. Alignment can be reduced to a simple formula:

$$new_{start} = old_{start} - \left(index_{alignment} - index_{peak}\right) \quad (42)$$

Naturally, when working with signals, the alignment implies changing the starting index of the signal indicated by *new_start* and *old_start*. The alignment index $index_{alignment}$ represents the index at which all the signals are shifted to by the point of interest. In this case, the point of interest is given by the global maxima. In the formula, the point of interest is the $index_{peak}$ and it is different for each signal. After this preprocessing step, the indexes of the points of interest are aligned to the chosen index, given by $index_{alignment}$.

Figure 3.2 shows the impact that such a simple technique can have on the feature extraction, the colours shown are the labels of the ground truth. In this case, PCA was the chosen feature extraction technique, without the alignment it splits a cluster into two subclusters. Unfortunately, this cannot be remediated by any clustering technique, as most likely they are identified as two separate clusters and no observer seeing those clusters can consider them as belonging together.

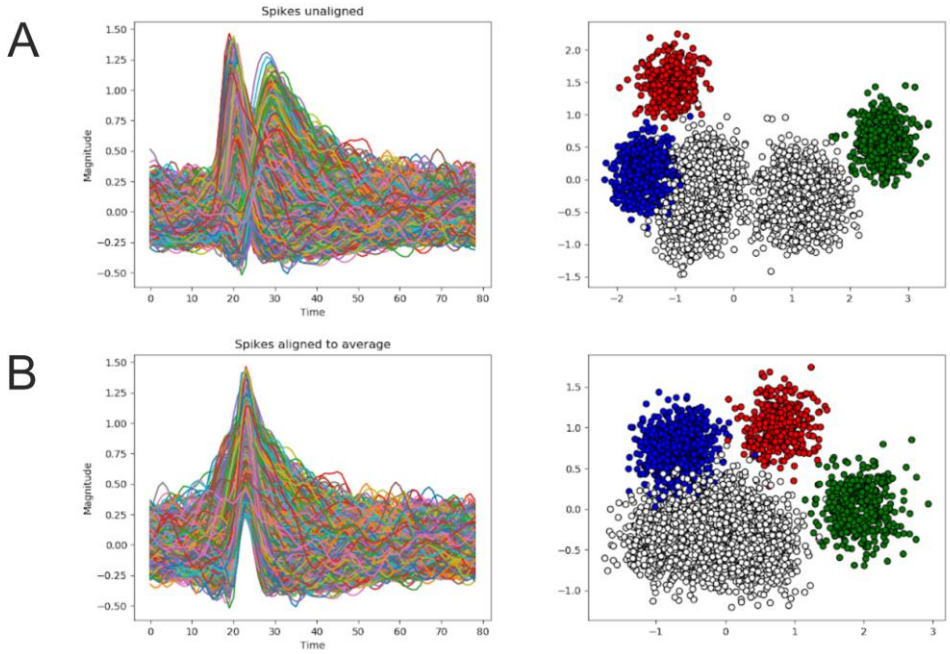

Figure 3.2 - The influence of spike alignment on feature extraction through PCA on synthetic data. Panel A shows the unaligned spikes on the left and the result of feature extraction on the right, the colours represent the ground truth, it is clearly visible that unaligned spikes created a feature space in which the white cluster is fragmented. Panel B shows the aligned spikes on left and the result of PCA on the right, in this case clusters are not fragmented.

The following subset of the 95 datasets are presented within this work:

- Simulation 1 (Sim1), containing 17 clusters in total (15 single-unit clusters and a multi-unit cluster) with 12012 spikes.
- Simulation 3 (Sim3), containing 3 clusters in total (2 single-unit clusters and a multi-unit cluster) with 3494 spikes.
- Simulation 4 (Sim4), containing 5 clusters in total (4 single-unit clusters and a multi-unit cluster) with 5127 spikes.
- Simulation 9 (Sim9), containing 19 clusters in total (18 single-unit clusters and a multi-unit cluster) with 15653 spikes.
- Simulation 10 (Sim10), containing 20 clusters in total (19 single-unit clusters and a multi-unit cluster) with 15149 spikes.

- Simulation 11 (Sim11), containing 20 clusters in total (19 single-unit clusters and a multi-unit cluster) with 14982 spikes.
- Simulation 12 (Sim12), containing 20 clusters in total (19 single-unit clusters and a multi-unit cluster) with 13488 spikes.
- Simulation 15 (Sim15), containing 10 clusters in total (9 single-unit clusters and a multi-unit cluster) with 9098 spikes.
- Simulation 79 (Sim79), containing 21 clusters in total (20 single-unit clusters and a multi-unit cluster) with 14536 spikes.

Real neural datasets

As with any computational endeavour, an important part is the data. The data presented here was recorded by the Transylvanian Institute of Neuroscience with 32-channel probes from the visual cortex of mice (details about the data acquisition procedure can be found in the Appendices). In the analyses that can be found throughout this subchapter several real datasets have been used, as follows:

- *M022* – recorded with a 32-channel probe with a sampling frequency of 32kHz for ~10 minutes resulting in a signal composed of 18,780,800 samples.
- *M045* – recorded with a 32-channel probe with a sampling frequency of 32kHz. The raw data has been filtered in the 300 to 7000Hz frequency band and spikes have been detected using an amplitude threshold. The waveforms of the spikes have been set at 58 samples; thus, each spike has 58 features.
    - *M045-C1* – represents the 1-st channel of the M045 dataset in which 11675 spikes have been detected.
    - *M045-C4* – represents the 4-th channel of the M045 dataset in which 4672 spikes have been detected.
    - *M045-C8* – represents the 8-th channel of the M045 dataset in which 18894 spikes have been detected.
    - *M045-C17* – represents the 17-th channel of the M045 dataset in which 8990 spikes have been detected.

3.1.3. Methods

3.1.3.1. Enhancing spike detection through neural networks

Traditional spike detection methods employ a simple amplitude threshold, either set manually or automatically, to identify potential spikes. However, this approach has limitations. This type of threshold implies a compromise, high thresholds may miss spikes with a lower amplitude than the threshold [2], while low thresholds can lead to the false detection of noise as spikes [2]. A more recent approach is based on template-matching, where waveforms extracted through spike sorting are compared to the signal using cross-correlation to detect spike-like activity [103]. However, these types of approaches often require extensive manual curation, rendering them impractical for datasets recorded using a large number of electrodes.

Source separation in spike sorting

In this section, source signal separation is analysed whether it is applicable within the spike detection step of the spike sorting pipeline. Through its functionality, the GAN architecture can be a candidate for spike detection with an additional advantage of having the capacity to extract spikes from distal neurons that have been drowned in noise.

The aim of source separation is to separate a mixed signal into its estimated sources. The approach of this method is to train a model to approximate as accurately as possible the sources that produced the mixture signal. Typical methods achieve this by minimising the reconstruction error for the mixture, correspondingly the conditional distribution of the mixture related to the estimates is maximised. The method presented accomplishes this through the use of Generative Adversarial Networks. The classic min-max game of the GAN revolves, in this case, around the Generator generating signal samples and the Discriminator attempting to classify them as real or fake [154].

A well-known analogy, used in this domain, is that of the Cocktail Party. The problem of isolating the speech of multiple individuals from a single blended recording. Many techniques have been applied to this problem; the GAN approach proposes the use of GANs in order to estimate the sources of a mixture signal. A noteworthy benefit is that the output distribution is learned implicitly by the GAN and does not need to be specified [154].

The proposed method suggests the creation of a GAN for each of the source estimates. After training, the discriminator can be removed from the problem, while the generator remains as a major component. To better exemplify the inner workings, let us assume that the overall model tries to separate the informative signal and the noise from a recording. Thus, two GANs are created, one for noise and another for the signal. Once the training has finished, two Generators emerge, one able to generate noise samples and another informative signal samples. In order to get the estimated sources another intermediate step needs to occur. The generated samples of the two generators are added together and subtracted from an original mixed signal. The result of the subtraction is then used to find the optimal latent variables of each generator in order to minimise the reconstruction error.

In the original paper [154], the authors also recommend the use of the Fourier Transform in the training of the GANs, transforming the input from the original signal into the concatenation of the real and imaginary outputs.

Approach

The Source Signal Separation approach [154] can be adapted for Spike Detection in Spike Sorting. Starting from a filtered signal and the selection of spike and noise windows of a fixed size, two GANs are created — their architectures are given by Table 3.1 — that receive inputs preprocessed using the Fourier Transform. A diagram showing this process is shown in Figure 3.3 as the GAN component. Thus, each GAN is trained on Fourier transformed inputs, as the concatenation of the real and imaginary outputs of each initial signal. At the end of the training procedure, two generators are ready to be used, one for spikes and one for noise. These generators are then used in the second part of the process, and they are shown in Figure 3.3, as "C1 Spikes" and "C1 Noise". Each generator is given a latent space, or hash, which it uses to generate Fourier transformed spikes or noise. These generated signals are then added in order to create a mixture. This mixture is then subtracted from a spike from the original dataset, naturally the Fourier Transform has been applied to this spike as well. The subtraction is then translated as the error of separation and used to update the latent space, notated as "Hash" in Figure 3.3. The latent space is being updated for a given number of iterations and it generates increasingly better separations as the number of iterations becomes higher, thus the estimated spike and noise are generated. The last step is the classification of the estimated spikes into actual spikes or not. This can be done in multiple ways. The Discriminator of the GAN Spikes can be used. Given the scores outputted by the Discriminator for the estimated spikes, a threshold is chosen as having the best F1 score.

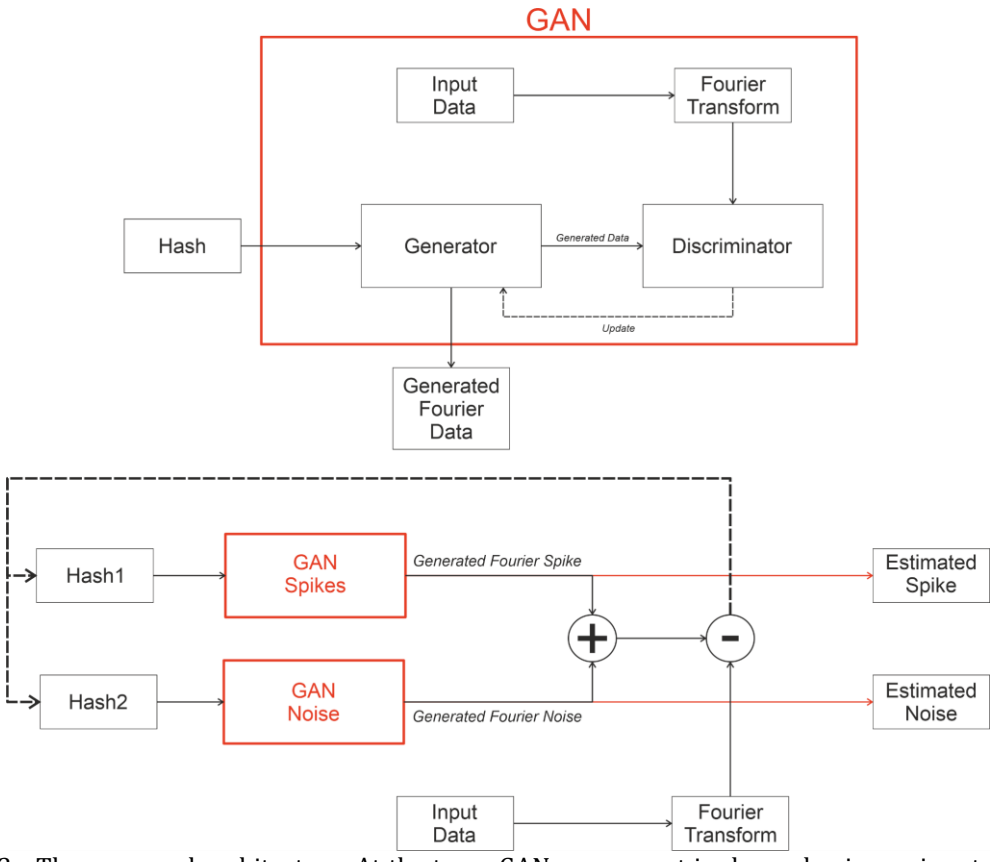

Figure 3.3 - The proposed architecture. At the top a GAN component is shown having as input a random vector, called Hash, and an output, the generated signal. The source signal separation architecture uses two GANs, one that generates each type of source. In this case, the two sources are spikes and noise. This

architecture attempts to generate by iterative updates, the two sources that estimate as well as possible the original signal.

Table 3.1 - Architecture of the Generator and Discriminator in the GAN.

Generator Layer / Units	**Discriminator Layer / Units**
Latent space / 100	Input Sample Size / 79
Hidden Linear / 200	Hidden Linear / 40
Softplus / 200	Tanh / 40
Hidden Linear / 200	Linear Output / 1
Softplus / 200	
Linear Output / 79	

For the training process, many parameters are used. The GANs have been trained initially on 100 epochs with batch sizes of 64 using the Adam optimizer with a learning rate of 0.002. In order to increase the performance of the generator, for each iteration of its training, the discriminator receives 5 training epochs. The initially chosen number of iterations for the update of the latent space was 100. The loss is presented in Figure 3.4 and it can be seen that the model becomes stable as it goes through the training process.

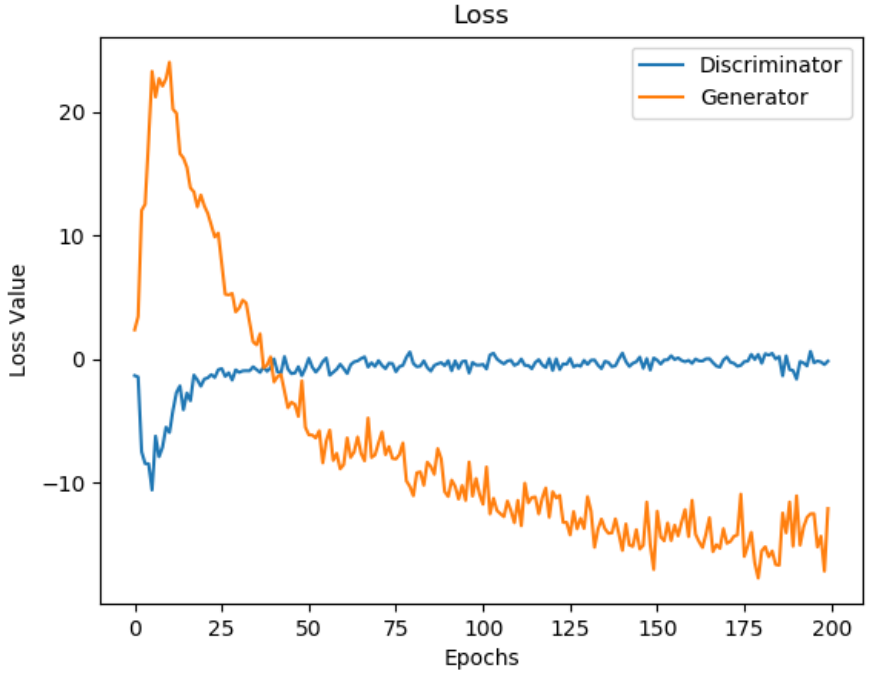

Figure 3.4 - Loss and accuracy of GAN training.

The training of the GAN was made using the Fourier Transform as it increased the reconstruction error of the estimation in the following steps. As such, the generator produces samples that have the characteristics of the Fourier transformed samples, shown in Figure 3.5, while Figure 3.6 shows the Inverse Fourier Transform of these generated samples and they do resemble the attributes of the original samples.

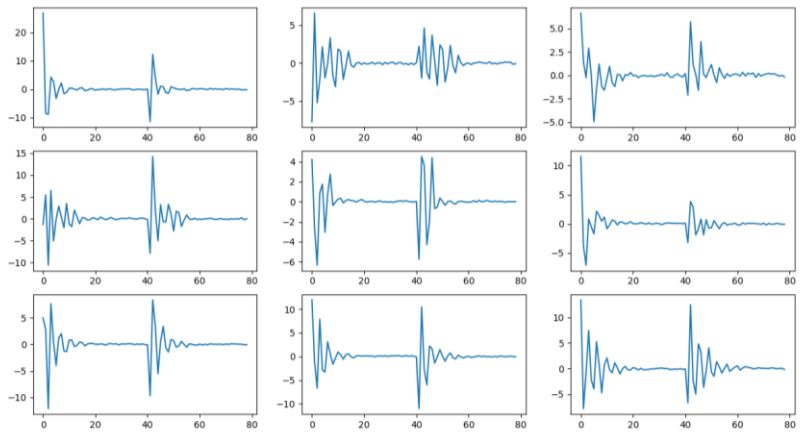

Figure 3.5 - Signals produced by the Generator model trained on the FFT signal of spikes.

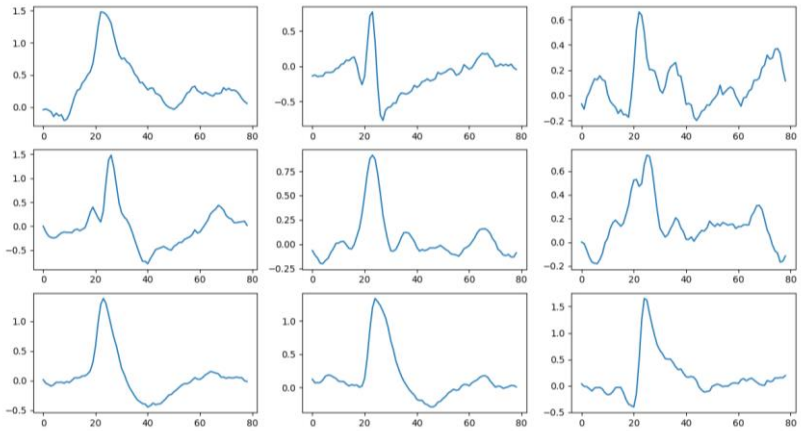

Figure 3.6 - The inverse Fourier Transform of the signals produced by the Generator model shown in Figure 3.5.

Fourier Transform Preprocessing

With regard to the inclusion of the Fourier Transform as a preprocessing step and the input of the GAN becoming the concatenation of the real and imaginary parts of the output, it greatly improves the reconstruction error of the estimated sources as it can be seen from Figures 3.17 and 3.18. The Fast Fourier Transform was applied, which calculates the Discrete Fourier Transform (DFT). When the input of the DFT is only real it still outputs a complex vector, but the real and imaginary parts are mirrored. Thus, the DFT of a signal is described by N/2 complex values or N values if their real and imaginary parts are concatenated. An advantage of the DFT is that it is invertible, thus the GAN can be trained on the transformed signals and view the results by using the inverse transform. Figure 3.7 shows a spike and its DFT. The reconstruction of the GAN, as a sum of component signals, can be viewed in Figure 3.8 and Figure 3.9. By comparing these two figures, the impact of the Fourier preprocessing can also be seen.

The DFT is described by the following formula:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i \frac{2\pi}{N} kn} \tag{43}$$

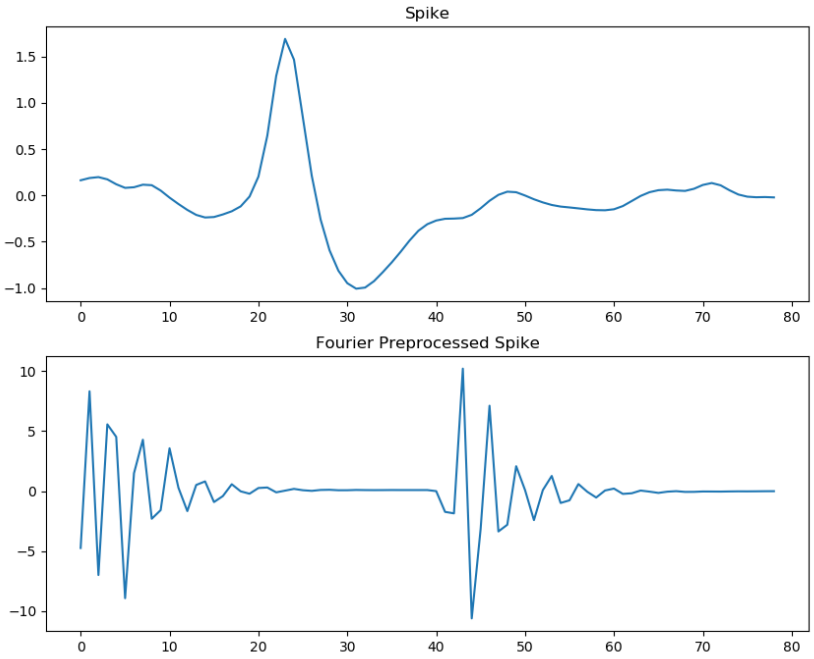

Figure 3.7 - A spike and its Fourier transform.

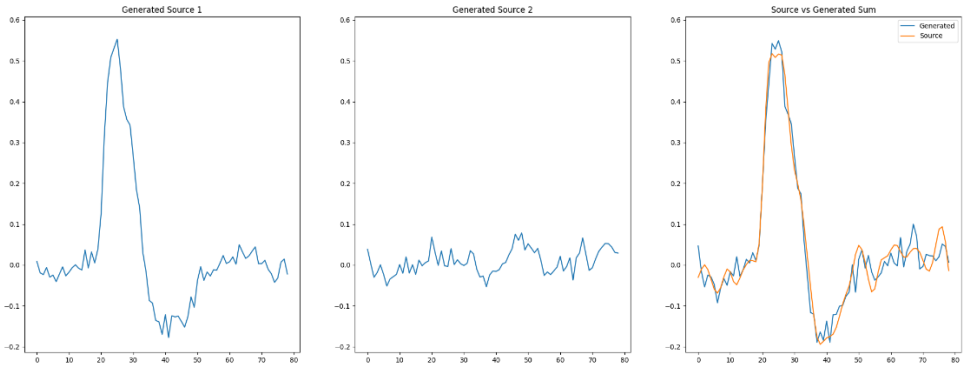

Figure 3.8 - Separation of a signal into spike and noise compared with the original signal with no preprocessing before insertion in the training of the GAN.

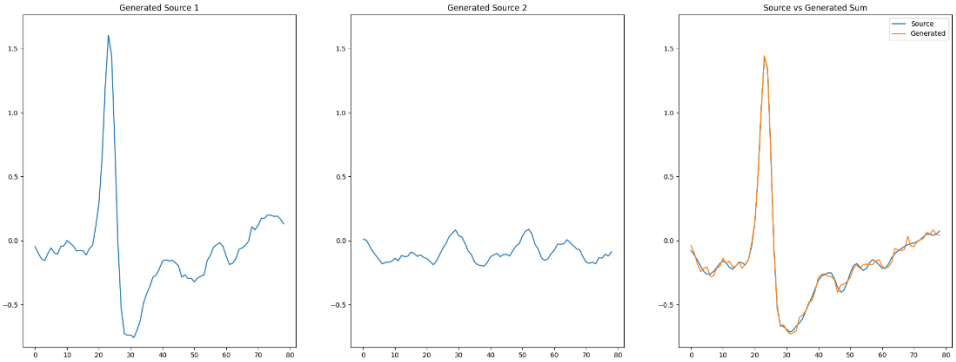

Figure 3.9 - Separation of a signal into spike and noise compared with the original signal when the signal was preprocessed using Fourier Transform before insertion in the training of the GAN.

Drowning Spikes using Signal-to-Noise Ratio

Drowned spikes can be generated through the change of the signal-to-noise ratio (SNR). This allows for the evaluation of the spike detection in various circumstances. The drowned spikes were generated as a linear scaling of a genuine spike with a scaling factor (SF) with an addition of noise:

$$S_{drowned} = SF * S_{spike} + S_{noise} \tag{44}$$

The SNR can be defined as the root mean square of the scaled spike divided by the root mean square of the noise, all to the power of two, for a given spike and noise signal of a fixed size:

$$SNR = \frac{\sqrt{\frac{1}{Size}\sum SF^2 \cdot {S_{spike}}^2}}{\sqrt{\frac{1}{Size}\sum {S_{noise}}^2}} = SF^2 \frac{\sum {S_{spike}}^2}{\sum {S_{noise}}^2} \tag{45}$$

Through some mathematical calculations, using the above formula, the SF can be extracted as:

$$SF = \sqrt{SNR \frac{\sum {S_{spike}}^2}{\sum {S_{noise}}^2}} \tag{46}$$

Using the formula for the generation of drowned spikes on a spike greatly reduces the overall amplitude and the magnitude of each oscillation, this can be viewed in Figure 3.10 where a SNR of 3 was applied.

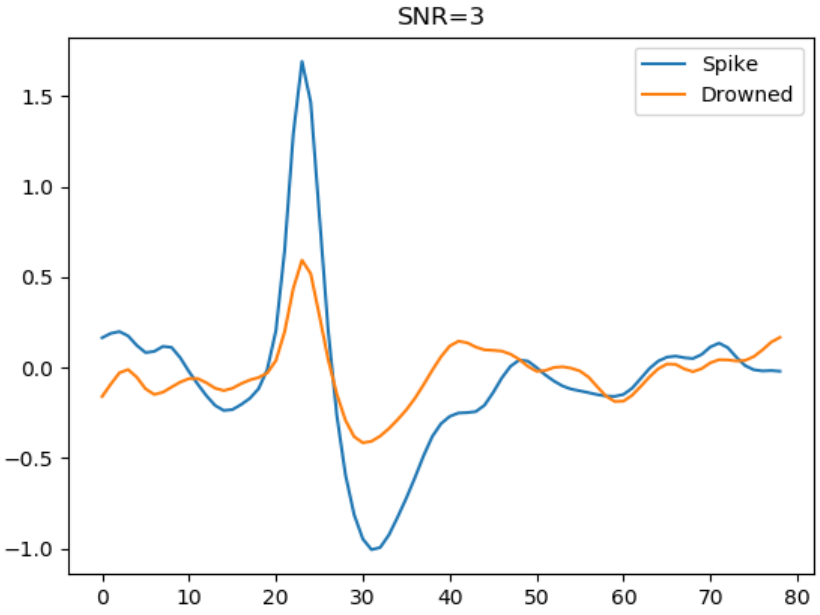

Figure 3.10 - Generation of drowned signals by changing to the signal-to-noise ratio.

Analysis of spike detection using GANs

Provided a specific dataset with a fifty-fifty proportion of spikes and noise that has been generated from synthetic data, this approach can have an overall better performance than the classical amplitude thresholding across all signal-to-noise ratios as

can be seen in Figure 3.11. This figure shows four plots indicating the performance of this approach across four metrics, the x-axis indicates the SNR applied to the dataset. In this case, the classic performance metrics of accuracy, f1 score, precision and recall were chosen rather than analysing the whole spike sorting pipeline with clustering performance metrics as the latter would reflect the performance of feature extraction and clustering as well without the ability to separate how much each of them influences the results.

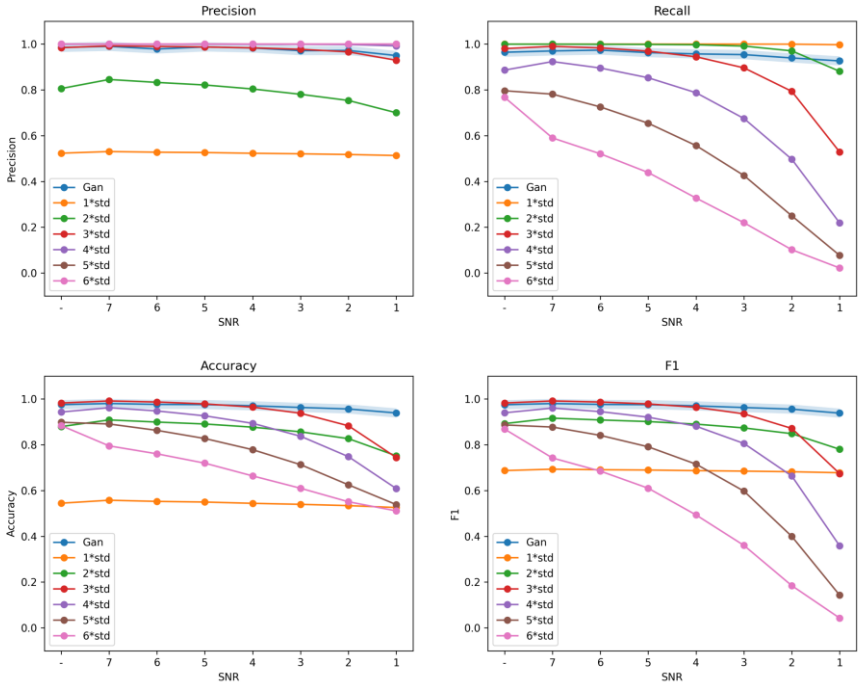

Figure 3.11 - Performance evaluations of spike detection using GANs with a specifically designed dataset of half spikes and half noise on training data. Each plot shows the performance from the perspective of a different metric. Lines in plots indicate the method used to detect spikes, whether it is the proposed approach, or the threshold based on the standard deviation. The x-axis indicates the SNR applied to the dataset.

The approach presented above was a proof-of-concept to assess that GANs would be able to manage the task of Spike Detection. To actually evaluate their performance, the data has to be as similar to other datasets used in Spike Sorting. In this case, no such perfect separation of half spikes and half noise can be made. Thus, the generation of the data has to be changed for suitable testing.

Knowing the window size of spikes, the dataset was separated into such windows that traverse the filtered signal with a given hop size, preferably smaller than the window size. For the circumstances presented here, the window size is 79 and the hop size is 10. The Spike GAN is trained to generate spikes from different perspectives due to the hopping through the signal. Naturally, the SNR is applied only to the windows that contain actual spikes. The results are unsatisfactory, viewable in Figure 3.12. As previously mentioned, even though the accuracy is high, the model does not have a high performance. This happens due to the fact that there are very few spike examples in comparison to noise examples.

Nevertheless, the performance of such a model trained on correctly generated data can be improved, as can be seen by comparing Figure 3.12 and Figure 3.13, by

increasing the number of training iterations. The number of epochs of the GAN training has been increased from 200 to 1000, while the separation update was increased from 100 to 1000.

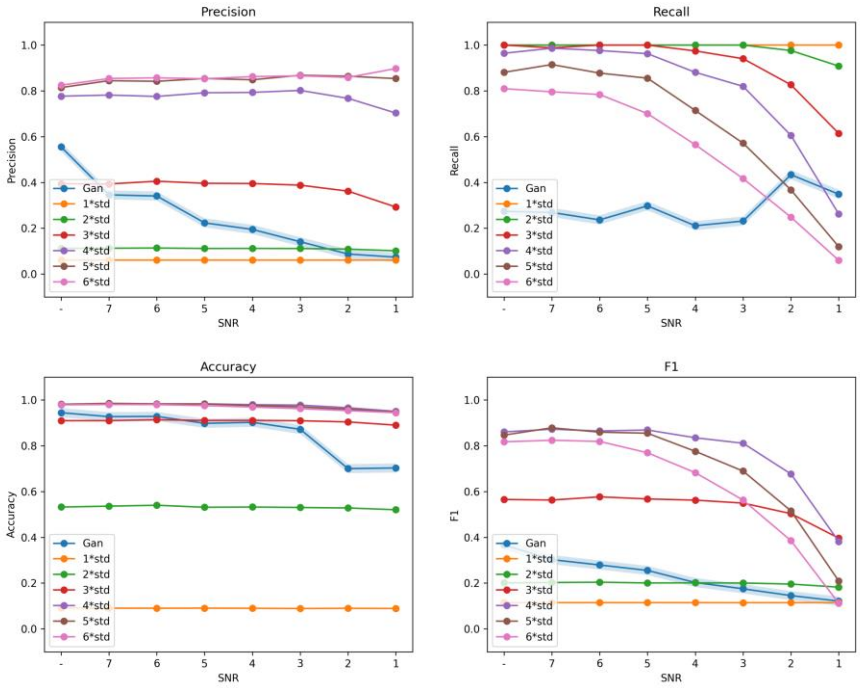

Figure 3.12 - Performance evaluations on the training data of spike detection using GANs on a synthetic dataset with a low percentage of spike windows in comparison with noise examples.

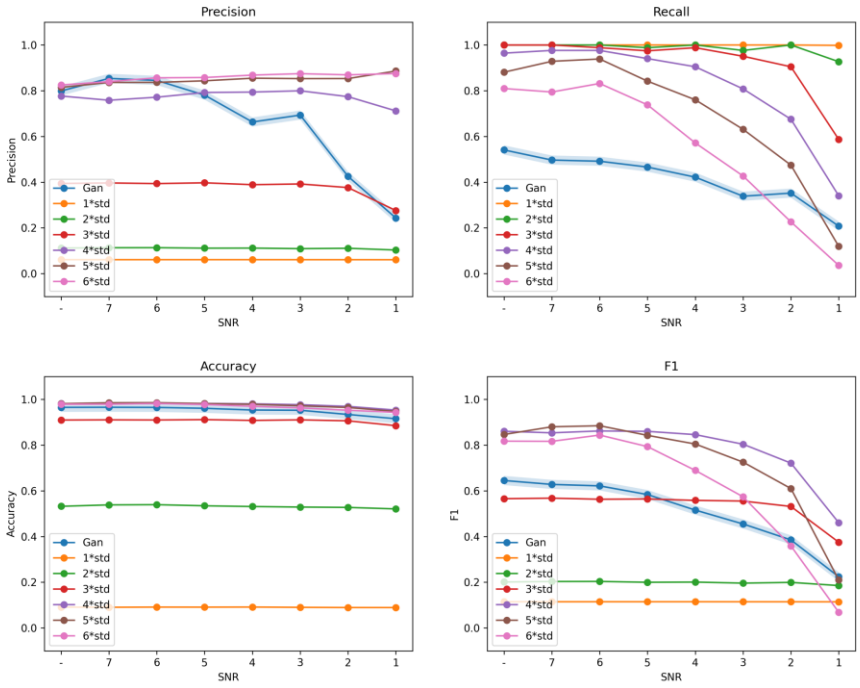

Figure 3.13 - Performance evaluations on the training data of Spike Detection using GANs on a synthetic dataset with a low percentage of spike windows in comparison with noise, with an increased number of training iterations.

Predicting sub-threshold spikes

Proposed method

Our proposed approach is to utilise an Artificial Neural Network (ANN) to enhance the effectiveness of the amplitude threshold-based detection through the identification of sub-threshold spikes based on similarity to those above the threshold. Spikes from neuronal recordings are detected using a hard amplitude threshold determined by a customizable factor of the standard deviation (SD) of the filtered signals. A classifier is trained using the spikes detected by the threshold and random noise segments (of equal size to the spikes) extracted from the recording. The ANN classifier learns to distinguish between the two classes, spikes and noise. To obtain new spikes that have been missed by the threshold, a sliding window is applied to the filtered signal, moving sample by sample, then each extracted window is inputted into the trained classifier to evaluate whether a spike exists within the window. Windows for which the classifier's target output is higher than a given threshold are recognized as spikes.

This approach was applied to the M022 real dataset and as such, it required the preprocessing of data involving several steps of the spike sorting pipeline. First, the channel underwent band-pass filtering using a bidirectional Butterworth IIR filter of order 3 with cutoff frequencies set between 300 Hz and 5000 Hz to obtain signals containing spiking activity. An amplitude threshold was set based on a factor of the standard deviation (SD) of the filtered signal, which is typically between 3 and 5 [2]. Spikes were identified as threshold crossings and were aligned and extracted from the signal.

The neural network architecture of the trained model is presented in Table 3.2 and was achieved through empiric evaluations in the search of a model that is able to learn the shape of spikes without overfitting in order to be able to find the sub-threshold spikes. It contained three hidden layers using the SELU activation function with decreasing sizes of artificial neurons, while the output was composed of a softmax activation function with the size equal to the number of classes. The training of the model was made on a dataset composed of half-spikes and half-noise to avoid imbalance in data, an example is shown in Figure 3.14. The loss function used was binary crossentropy and the optimizer used was RMSprop with a learning rate of 1e-4 and a momentum of 0.8. The loss and accuracy of the training of the model can be viewed in Figure 3.15 indicating that the training of the model becomes saturated at the beginning. Similar results can be obtained with a variety of hidden layer activation functions and optimizers.

Table 3.2 - Architecture of the classifier.

Layer	**Units**	**Activation**
Input	58	-
Hidden Linear	40	SELU
Hidden Linear	20	SELU
Hidden Linear	10	SELU
Output	2	Softmax

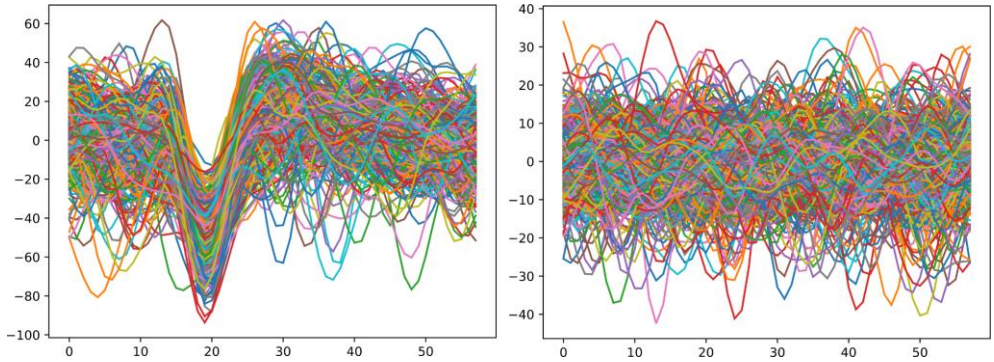

Figure 3.14 - Training data of the classifier: (A) extracellular spikes, (B) noise. The spikes and the noise segments were extracted from the M022 real dataset.

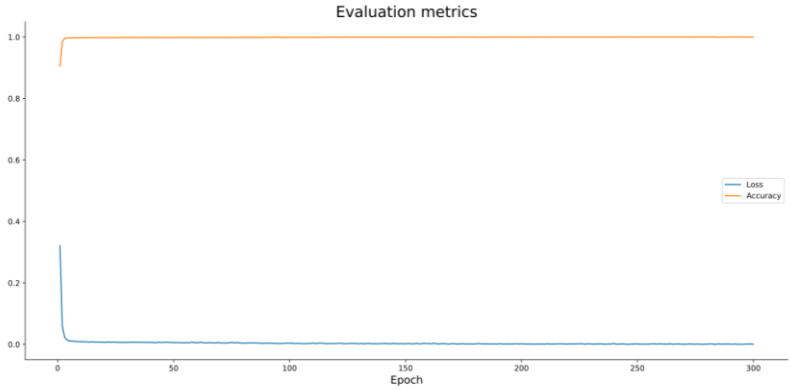

Figure 3.15 - Loss and accuracy of the classifier during training on the M022 real dataset.

Once the classifier has been trained, it can be employed for the detection of spikes by providing it with a window of the signal having the same size as a spike. This window is then moved sample by sample across the entire signal, and the output for the spike class can be recorded at each step.

Performance analysis

In a subsequent step, spikes identified by the classifier are localised by applying a high probability threshold, typically set around 0.99. The spike waveforms can then be extracted from the original signal by selecting windows of the size of a spike where the probability threshold is crossed. This process often yields a large number of potential spikes, including misaligned spikes. To address this issue, spikes that do not have the peak properly aligned with the 20th sample can be discarded. Moreover, the classifier also finds spikes that were identified by the amplitude threshold and that were in the training data, these can also be discarded. A short analysis through a histogram of the predictions over the whole signal, indicates that none of the spikes used in the training data are given a high probability of noise with the highest value being 1e-07. Moreover, all of them are identified even with such a high probability threshold.

The assessment of the validity of the new spikes found by the classifier is difficult. The analysis opted for was a comparison of spike amplitudes and spike shapes between the spikes extracted by the amplitude threshold, those identified by the classifier and all of them together. The comparison between the amplitude threshold approach and the classifier approach reveals differences in the spike amplitude distribution. The

distribution of the amplitude threshold approach appears skewed distribution towards positive values and cut off by the threshold, indicating that some information might be missing. However, incorporating the spikes identified by the classifier shows a more natural distribution with a drop off. Examination of the spike shapes indicates that the shape of spikes detected by the classifier are similar to those of the amplitude threshold; this is also supported by the average waveform. This interpretation can be extracted from Figure 3.16.

Furthermore, an analysis through PCA shows that the new spikes found by the classifier have more overlap with those found by amplitude threshold than random noise from the signal. This can be viewed in Figure 3.17. To acquire a numeric value of the overlap, the convex hulls of the spike and noise clusters can be calculated and the mean distances between new spikes to each cluster can be used to obtain a metric that denotes with which of these two clusters has the most overlap. Using this metric, the distance between the cluster of new spikes and those obtained by the threshold are 6e-5, while the distance between the cluster of new spikes and that of the noise is 0.49, indicating that the new spikes are more similar to those obtained by thresholding than to noise.

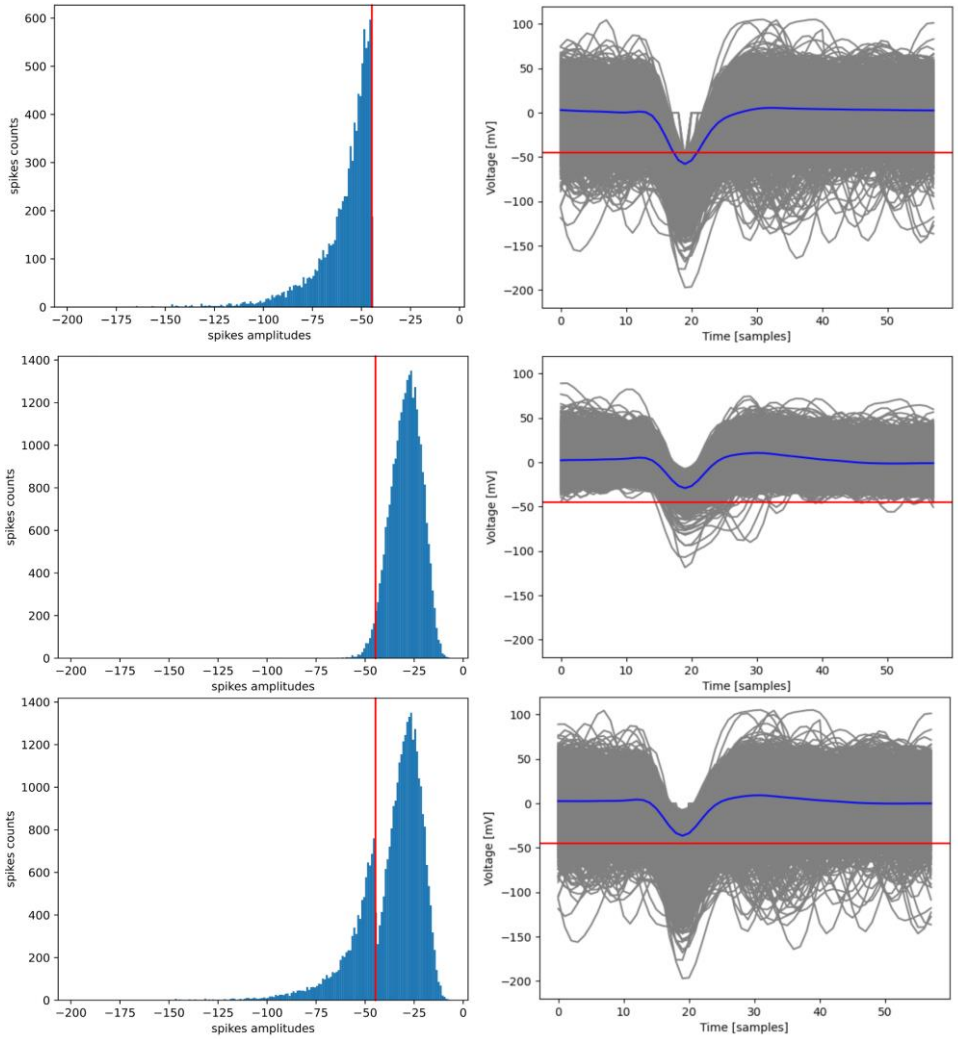

Figure 3.16 - The left side shows histograms of the distributions of spike amplitudes, while the right side shows the spike shapes. The top represents spikes found by the amplitude thresholding approach, the middle shows spikes found by the classifier approach and the bottom shows the sum of the two approaches.

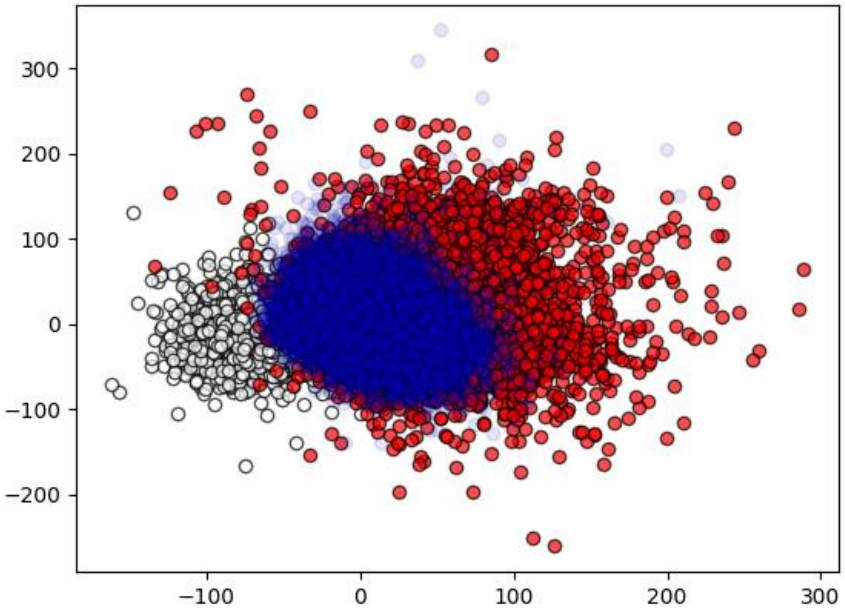

Figure 3.17 - PCA analysis of spikes found by amplitude thresholding (red), new spikes found by classifier (blue), and noise (white).

The proposed ANN classifier requires a new training for every recording, conceivably even for every channel of a recording as there is a high variation across recording sessions and even between the electrodes of a probe in a single recording. Its utility is highest for offline processing in spike sorting.

Findings

Incorporating a classifier alongside traditional threshold-based approaches provides more advantages. While the amplitude threshold method may skew the distribution of spike amplitudes and potentially miss relevant information due to its cutoff nature, the classifier-based approach presents a more natural distribution with discernible drop-offs. Moreover, the similarity in spike shapes and waveforms between the two methods indicates that the classifier's outputs are sub-threshold spikes. This is further supported by the PCA analysis, which demonstrates that the spikes identified by the classifier have more overlap with those detected by amplitude thresholding than with random noise from the signal.

The proposed ANN classifier requires retraining for each recording or even for each channel within a session due to the high variability across recordings and electrodes. Consequently, its primary use lies in offline spike sorting applications.

3.1.3.2. Autoencoders in spike sorting

The autoencoder architectures are neural networks created as multiple fully connected hidden layers, symmetric between the encoder and the decoder that are linked through a bottleneck, and as such they fall into the category of deep autoencoders. In the case of spike sorting, they have been shown to give superior results [54,55] as they are more capable of learning the intricacies of neuronal data.

Proposed method

We propose [155] autoencoders as a feature extraction method for spike sorting. Several variations of autoencoder architectures and preprocessing options have been tested in order to assess the best configuration for neuronal data. Several variants have been previously described. A combination of PCA and AE has been tested, the autoencoder encodes the data into a new lower-dimensionality intermediate space followed by PCA to further reduce the space into a visualizable scope. The Fourier Transform has been added as a preprocessing alternative before entering the training of the AE. By applying the FT, a real and imaginary part are received that can be combined in multiple ways of use: real part, imaginary part, magnitude, phase, power, and various combinations of these. Here, two major options are available: the classical preprocessing of transforming the data using the FT by introducing the real part as the training samples and a windowed variant of FT using the Blackman window as they produced the best results. Additionally, multiple network depths are evaluated in order to assess their performance.

Multiple different performance metrics, both internal and external, have been used to appraise the performance of the various models. These metrics have already been described and presented previously. As the evaluation of the AE as a feature extraction is made in the context of spike sorting, they must be combined with a clustering algorithm in order to correctly evaluate their performance, for this purpose K-Means has been chosen.

The variants presented here have several identical items of configuration, a general representation is shown in Figure 3.18. The encoder and decoder have symmetric mirrored layers containing *ReLU* activation functions, while the code and output layers have the *tanh* activation function. As the *tanh* function is bounded within the [-1,1] range the input data must be scaled to fit within these bounds.

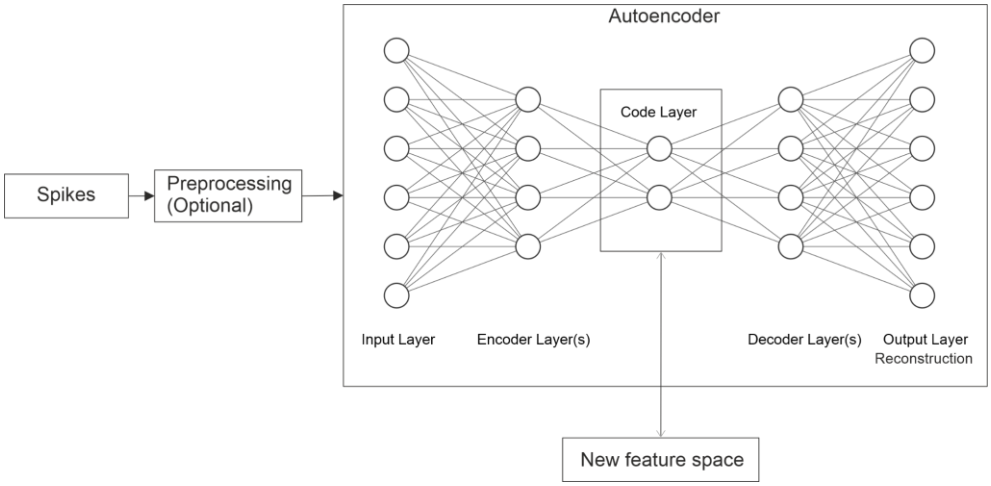

Figure 3.18 - General architecture of an AE.

Additionally, the code layer contains an L1 regularisation of 10e-7 in order to help with preventing overfitting of the training data. The optimiser chosen is the Adam optimizer with the Mean Squared Error (MSE) as the loss function. The MSE produces the best point-by-point estimation of the input at the output. This is important because the reconstruction exhibits the ability of the AE to encode and decode information through the bottleneck.

Performance analysis of autoencoder variants

A simple preprocessing with a high impact on the performance of the feature extraction is the alignment. The assessment of performance of PCA and a general architecture of AE is presented in Table 3.3.

Table 3.3 - Alignment impact on the performance evaluation for PCA and AE on the Sim4 synthetic dataset.

	ARI	**AMI**	**VM**	**DBS**	**CHS**	**SS**
PCA-Unaligned	0.52	0.77	0.77	**0.54**	5383	0.47
PCA-Aligned	0.57	0.8	0.8	0.58	7550	0.48
AE-Unaligned	0.7	0.83	0.83	0.62	6260	0.49
AE-Aligned	**0.98**	**0.96**	**0.96**	0.55	**7807**	**0.61**

The influence of AE training hyperparameters on the performance of a general architecture of AE was analysed. Table 3.4 presents the effect of various numbers of epochs, while Table 3.5 presents the performance of multiple values of learning rates.

Table 3.4 - Performance of different values of the number of epochs for the training process on the Sim4 synthetic dataset.

	ARI	**AMI**	**VM**	**DBS**	**CHS**	**SS**
Epochs=50	**0.98**	**0.97**	**0.97**	**0.27**	**52012**	**0.79**
Epochs=100	0.91	0.88	0.88	0.35	28541	0.66
Epochs=500	0.98	0.96	0.96	0.55	7807	0.61

Table 3.5 - Performance of different values of learning rates for the training process on the Sim4 synthetic dataset [155].

	ARI	**AMI**	**VM**	**DBS**	**CHS**	**SS**
LR=0.1	0	0	0	0	0	0
LR=0.01	0.47	0.69	0.69	0.47	**12450**	0.39
LR=0.001	0.98	0.96	0.96	0.55	7807	0.61
LR=0.0001	**0.98**	**0.97**	**0.97**	**0.43**	12343	**0.62**

The AE variants presented have been compared against widely used feature extraction methods that have been described, namely PCA, ICA and Isomap. The evaluation of the methods has been done on multiple synthetic and real datasets with various numbers of clusters and samples for an exhaustive performance evaluation as different techniques may be more appropriate for different numbers of clusters and samples. Figure 3.19 presents the Sim4 synthetic dataset as it offers a relevant visualisation. The performance evaluation of the presented methods is available in Table 3.6.

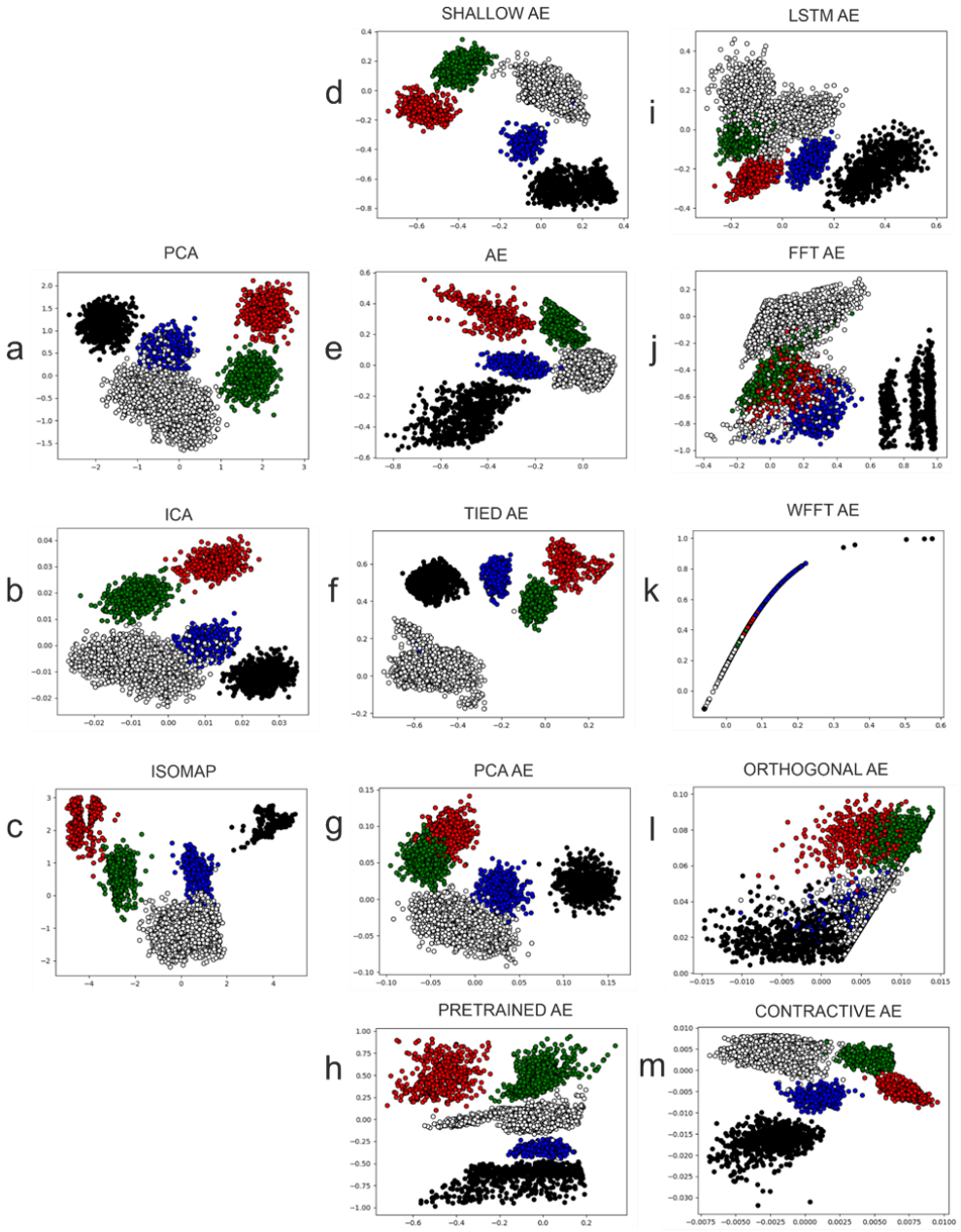

Figure 3.19 - Feature extraction of the Sim4 dataset containing 5 clusters, the colour coding shown represents the ground truth.

Table 3.6 - Evaluation of feature extraction methods on Sim4 containing 5 clusters.

	ARI	**AMI**	**VM**	**DBS**	**CHS**	**SS**
PCA	0.57	0.8	0.8	0.58	7550	0.48
ICA	0.61	0.8	0.8	0.53	7000	0.51
Isomap	0.96	0.95	0.95	0.47	17722	0.61
Shallow AE	0.99	0.98	0.98	0.43	13835	0.65
AE	0.98	0.96	0.96	0.55	7807	0.61
Tied AE	0.59	0.8	0.8	0.44	13698	0.68
PCA AE	0.59	0.79	0.79	0.65	8959	0.51
Pretrained AE	0.91	0.92	0.92	0.27	33,910	0.73

LSTM AE	0.31	0.59	0.59	0.82	3592	0.25
FT AE	0.36	0.51	0.51	1.63	2672	0.18
WFT AE	0.43	0.59	0.59	9.13	2293	0.28
Orthogonal AE	0.32	0.44	0.44	8.71	4866	0.05
Contractive AE	0.97	0.95	0.95	0.46	14852	0.62

A general analysis of the 95 simulated synthetic datasets can be viewed in Figure 3.20 with regard to each of the 6 performance metrics used. Additionally, a ranking of the methods has been made using the Borda rank aggregation [156] and it is presented in Table 3.7.

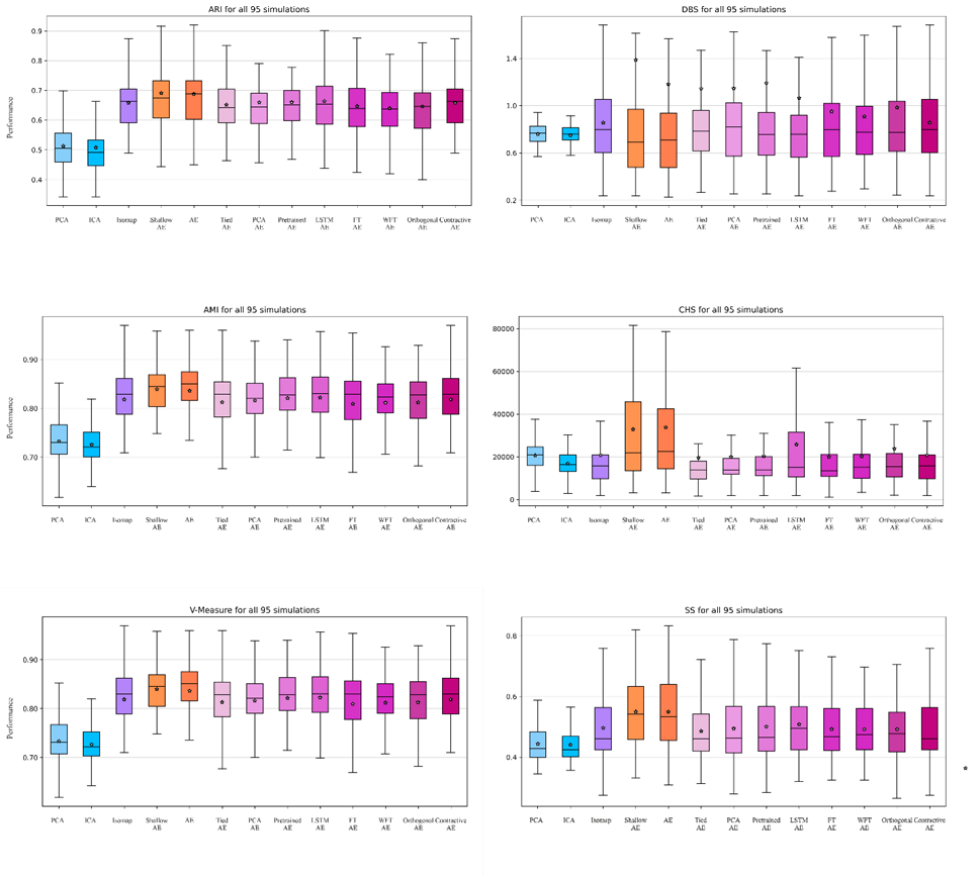

Figure 3.20 - Evaluation of performance for all approaches applied on all 95 synthetic simulations with regard to each performance metric.

Table 3.7 - Approaches ranked using the Borda ranking system with regard to each metric after applying them on all 95 synthetic simulations.

Rank	**ARI**	**AMI**	**VM**	**DBS**	**CHS**	**SS**
1	AE	Shallow AE	Shallow AE	AE	AE	Shallow AE
2	Shallow AE	AE	AE	Shallow AE	Shallow AE	AE
3	Isomap	Isomap	Isomap	ICA	PCA	LSTM AE
4	LSTM AE	LSTM AE	LSTM AE	Pretrained AE	LSTM AE	Pretrained AE
5	PCA AE	Pretrained AE	Pretrained AE	Contractive AE	Orthogonal AE	Isomap

6	Pretrained AE	WFT AE	WFT AE	PCA	Isomap	PCA AE
7	Tied AE	Tied AE	Tied AE	LSTM AE	Pretrained AE	WFT AE
8	Contractive AE	PCA AE	PCA AE	FT AE	ICA	Orthogonal AE
9	FT AE	Contractive AE	Contractive AE	PCA AE	WFT AE	FT AE
10	Orthogonal AE	Orthogonal AE	Orthogonal AE	Orthogonal AE	PCA AE	Tied AE
11	WFT AE	FT AE	FT AE	Tied AE	FT AE	Contractive AE
12	PCA	PCA	PCA	WFT AE	Contractive AE	PCA
13	ICA	ICA	ICA	Isomap	Tied AE	ICA

The following analysis considers the performance of the presented methods on real data, denoted as M045-C17, that was recorded extracellularly from the brain of a mouse. The M045-C17 dataset does not contain a ground truth that can be used, as such the analysis of the approaches are reduced in complexity and only use three of the six performance metrics, namely the internal metrics: CHS, DBS and SS. It is important to mention that this renders the performance biassed with regard to the ability of the clustering algorithm. Table 3.8 presents the analysis of performance for all feature extraction methods, while the results can be visualised in Figure 3.21.

Table 3.8 - Performance analysis of all feature extraction methods on the M045-C17 real dataset.

	DBS	**CHS**	**SS**
PCA	0.63	20637	0.76
ICA	0.72	8200	0.64
Isomap	0.45	**68339**	0.86
Shallow AE	0.65	14000	0.72
AE	0.24	14100	**0.95**
Tied AE	0.45	34700	0.8
PCA AE	**0.19**	52767	0.87
Pretrained AE	0.32	45700	0.92
LSTM AE	1	3844	0.38
FT AE	0.84	12555	0.61
WFT AE	0.79	8700	0.45
Orthogonal AE	0.58	20500	0.58
Contractive AE	0.36	31881	0.8

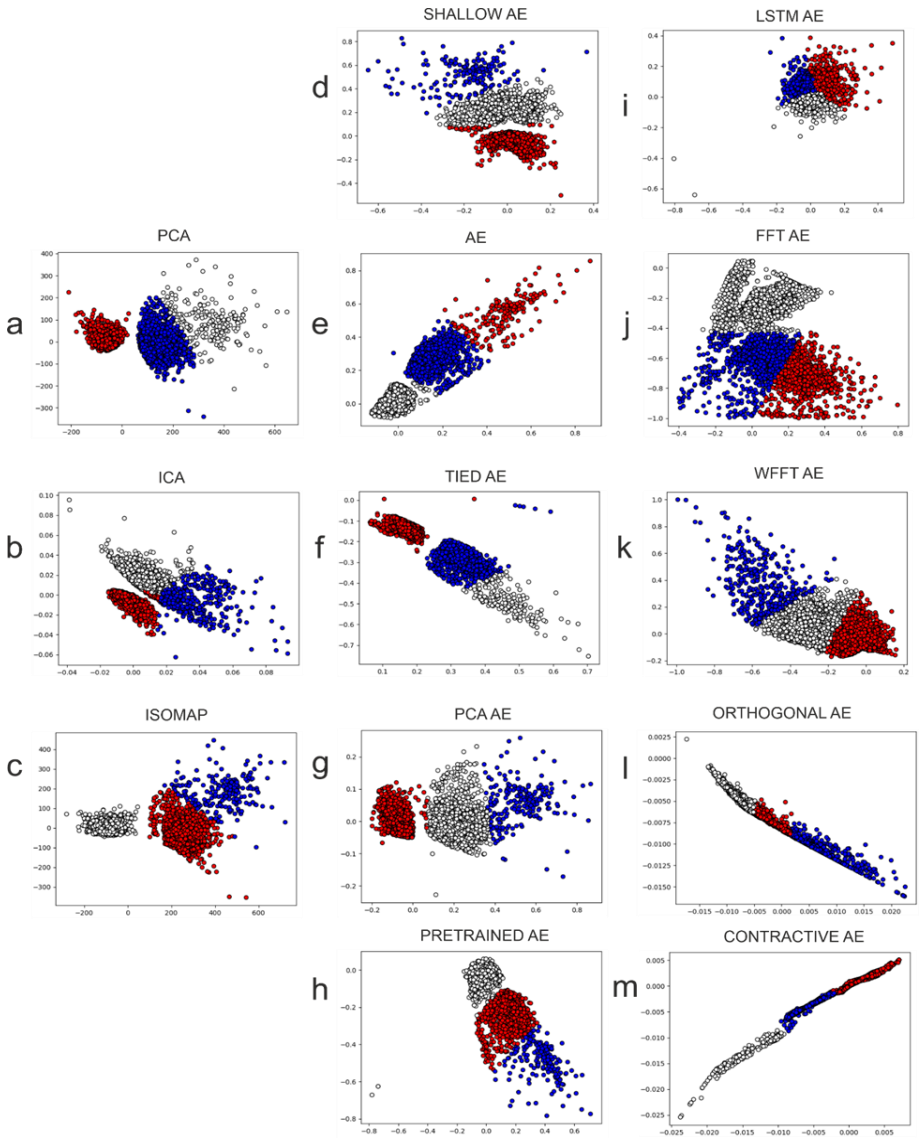

Figure 3.21 - Feature extraction methods applied on the M045-C17 real dataset; colours indicate the labels obtained through K-Means.

Findings

The Autoencoder based variants have demonstrated the ability to generate features that provide enhanced separability for the clustering neuronal spikes, both visually and based on multiple performance evaluation metrics on a large number of datasets. Just as the state-of-the-art methods, these variants do not require prior knowledge of informative features. Among the examined variants, the AE variant consistently outperformed all other methods and autoencoder variants both for synthetic and real datasets when used in conjunction with K-Means. This observation is supported by the evaluations presented of various metrics and the Borda ranking of methods. Other AE variants have shown situational performance, such as the Shallow AE variant, which performed well, but only on synthetic datasets. For real data with more complex spikes, additional layers seem to be necessary to achieve better separation, as demonstrated by the superior performance of the AE variant compared to the Shallow AE variant.

Autoencoders, being unsupervised models, offer a suitable solution for spike sorting as they do not require the ground truth labels of data. They also do not require a segregation between training and testing datasets, as the model is trained on the entire dataset and the latent feature space can be extracted. This inherent characteristic ensures robustness and avoids performance drops due to improper training. However, autoencoder models need to be trained for each new dataset, inducing additional costs in terms of execution time. The additional cost is dependent upon the number of samples of the data and the chosen number of epochs. Despite the additional costs, the improvement in performance justifies the expense.

3.1.3.3. The Superlet Transform in spike sorting. Evaluation of the superlet features

Proposed method

Our proposed method [157] employs the Superlet Transform as a feature engineering algorithm combined with PCA to reduce the dimensionality. The superlet is composed of *o* wavelets, each wavelet of the set has its own number of cycles that is increased multiplicatively. The choice of these parameters can impact the results, as shown in Figure 3.22 and Figure 3.23, as such each parameter must be analysed in order to understand its influence on the results. In this case, the Superlet Transform has been applied to the Sim8 synthetic dataset The transform returns a spectrogram, as shown in Figure 3.24, that is then used as input for PCA in order to reduce the dimensionality to 2 for visualisation. Therefore, the Superlet Transform is used as a feature engineering technique.

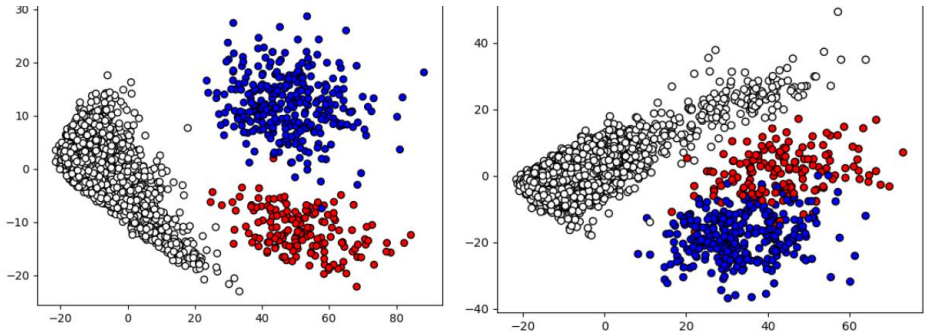

Figure 3.22 - Feature extraction using the Superlet Transform and PCA on the Sim8 synthetic dataset with various values for the *cycles* parameter (c=1.5, left and c=3, right).

In this case, the Superlet Transform is used as a feature engineering technique in the process of Spike Sorting followed by a feature extraction step, the clustering performance must be taken into consideration to determine the most performant parameter values by varying both of these. Through observations made on a number of simulated datasets with varying numbers of clusters, the most performant values for the parameters were found to be the minimum values of each, as shown by Table 3.9. Moreover, these set of parameters seem to be the most performant not only for PCA but for other feature extraction techniques as well. Nevertheless, Superlets produced the best results in combination with PCA.

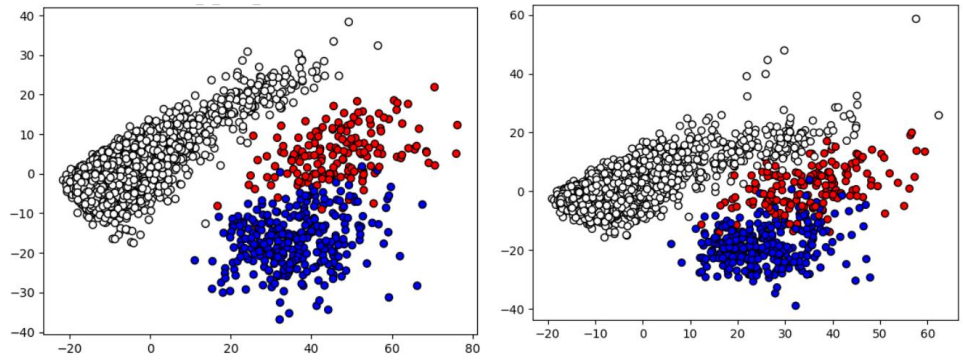

Figure 3.23 - Feature extraction using the Superlet Transform and PCA on the Sim8 synthetic dataset with various values for the *order* parameter (o=5, left and o=15, right).

Performance analysis

As the dataset also provides the ground truth, the impact of different parameter values on the spectrogram can be analysed through the learning capabilities offered through neural network classification. This can be viewed in Figure 3.24, as the spectrogram of a whole cluster was calculated. Here, it is highlighted that the ideal precision in both time and frequency is achieved by the minimum values of the parameters as can be seen from Table 3.9. As the parameters are modified, it produces higher precision in either time or frequency.

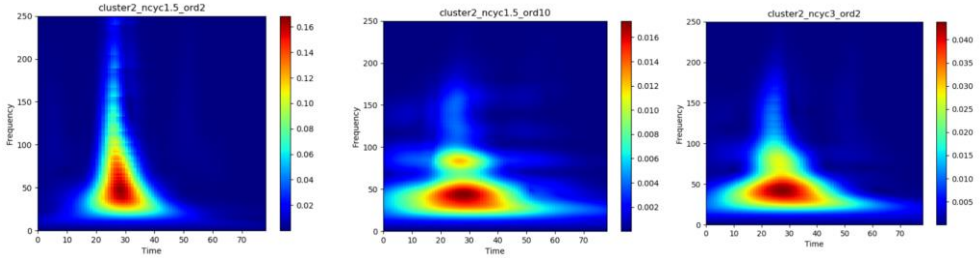

Figure 3.24 - Spectrogram of a cluster from a simulated dataset (Sim8) with varying values for the parameters. The left image shows the best precision with *c*=1.5 and *o*=2, the minimum values. The middle image has a high *order* value resulting in a high frequency precision, while the right image has a high *cycle* value resulting in a high temporal precision.

Table 3.9 - Performance evaluation of Superlets features.

	ord	**c1**	**DBS**	**CHS**	**SS**
Sim 8 (3 clusters)	2	1.5	**0.994**	**7110.6**	**0.731**
	2	3	1.267	3097	0.639
	5	1.5	1.248	3614.6	0.656
	10	1.5	1.507	2346	0.603
	15	1.5	1.661	1885.3	0.574
Sim 15 (10 clusters)	2	1.5	**0.853**	**40316.1**	**0.541**
	2	3	0.947	36443.7	0.471
	5	1.5	0.916	35179.4	0.489
	10	1.5	1.146	23950.9	0.418
	15	1.5	1.521	28150.9	0.379
Sim 79 (21 clusters)	2	1.5	**1.697**	**10183.8**	**0.296**
	2	3	1.828	7481.8	0.255

	5	1.5	1.481	5331.6	0.242
	10	1.5	2.092	4120	0.226
	15	1.5	2.188	4217.5	0.197

The relevance of the features produced by the Superlet Transform can be assessed by classification. As the data contains the ground truth, a neural network can be trained to classify the data by their spectrograms. Thus, if a neural network model is able to learn and predict with a high accuracy the cluster of a spike by its Superlet features then it can be confidently said that the Superlet Transform provides relevant information.

The spectrograms of the data, each signal having its own unique Superlet representation, is split into the training, validation and testing subsets, as is commonly done. Most datasets contain upwards of 3 clusters, as such the problem becomes one of multi-label classification. I have chosen a multi-layer feedforward neural network with 500 neurons for the input, 3 hidden layers of decreasing numbers of neurons by dividing with powers of 2 and an output layer equivalent to the number of unique labels of the dataset. The input represents the number of values in the time-frequency bins of the Superlet representation. The activation of the hidden layer was chosen as ReLU, while the output contains a softmax activation. The chosen loss function was the categorical crossentropy. The labels have been one hot encoded to work with the softmax activation. The evolution of the loss and the performance metrics during training can be viewed in Figure 3.25. The final results in terms of performance of this method on multiple datasets with varying numbers of clusters can be viewed in Table 3.10.

Table 3.10 - Evaluation of the neural network's learning from superlet features.

	Loss	**Accuracy**	**F1**
Sim8 (3 clusters)	0.03	0.988	0.986
Sim33 (5 clusters)	0.06	0.976	0.976
Sim84 (7 clusters)	0.01	0.996	0.993
Sim63 (15 clusters)	0.04	0.983	0.983
Sim79 (21 clusters)	0.06	0.981	0.981

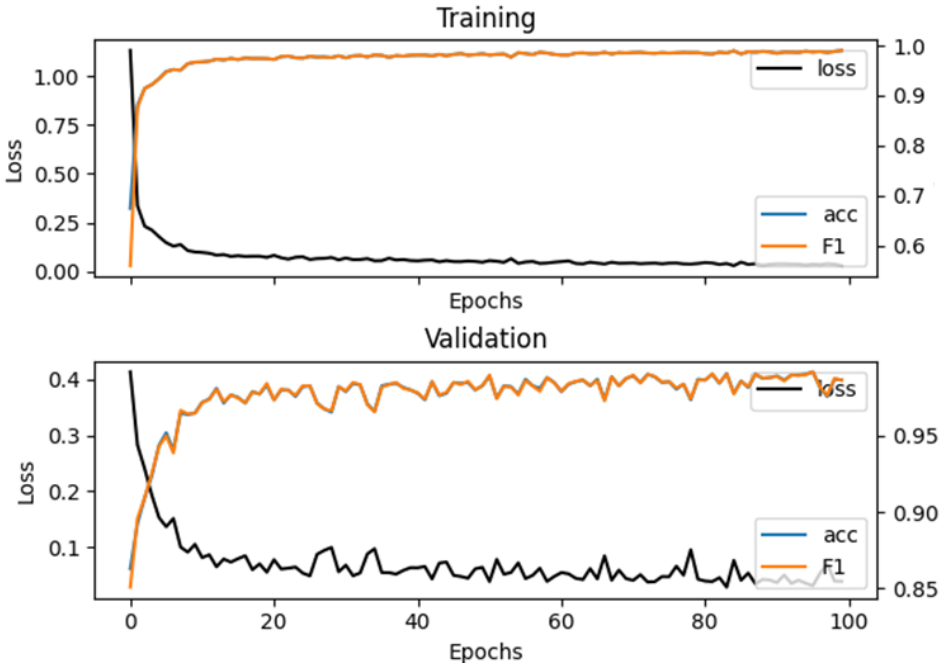

Figure 3.25 - Evolution of loss and performance metrics (accuracy and F1 score, they are overlapped) during training of the Sim79 dataset with 21 clusters.

Findings

The characteristics provided by Superlet Transform have demonstrated superior performance in enhancing the distinguishability of clusters in the spike sorting domain compared to features provided by traditional feature extraction methods.

The performance evaluation indicates that using the minimum values for the Superlet parameters resulted in best performance for the clustering phase and with these parameters it consistently performed well, regardless of the number of clusters.

3.1.3.4. The Superlet Transform in spike sorting. Identifying superlet feature importance through perturbation

In our prior research [157], we have shown that the characteristics provided by Superlet Transform can be used to outperform traditional feature extraction methods by introducing new information that enhance the distinguishability of clusters in spike sorting. Building upon this finding, the objective was to determine the key features within the spectrogram, generated through the Superlet Transform, that facilitate effective differentiation among various spike clusters.

In order to investigate the advantages of the Superlet transform in feature separability, experiments were conducted using real datasets obtained from electrophysiological recordings in the brains of anaesthetised rodents. Additionally, the recorded data underwent manual spike sorting by an expert to establish a reliable "ground truth" for classification and comparative analysis.

Proposed method

Our proposed approach [158] for determining the significant feature areas crucial to model learning is a sequential pipeline illustrated in Figure 3.26. The initial step of the proposed pipeline involves computing the Superlet spectrum for each spike. To reduce dimensionality and computation time, downsampling of the spectra is performed using bicubic interpolation, although this step is not essential for the pipeline. Instead of perturbing individual features, all correlated features are perturbed together to account for compensation from correlated features during training. Thus, the subsequent step in the pipeline entails identifying sets of features that exhibit correlations above a specified threshold. This is accomplished by processing the correlation matrix. Once the correlated feature sets are identified, the comparison process can commence. An instance of the neural network is first trained on the unperturbed features. Next, each individual set of correlated features is perturbed, and a neural network of the same architecture is trained on the modified dataset that contains the perturbation of a single feature set. Through the difference in performance of the neural networks trained on the unperturbed and perturbed dataset, the impact in learning of the feature set is obtained. The following subsections provide a detailed description of these steps.

The classifier (neural network model) takes the spectrogram, generated through the Superlet Transform, as its input. Therefore, it is the characteristics of the spectrogram that is perturbed in the proposed method. Figure 3.27 displays the time-frequency spectrum of three average spike waveforms obtained by applying the Superlet method (order 2, with a cycle number of c1 = 1.5) to the M045-C4 real dataset. The top section of each figure illustrates the average spike from each cluster, while the bottom section presents the representation of the spike in the time-frequency domain.

The spectrogram matrix was resized from its original size of [58, 50] to a smaller size of [14, 12] through bicubic interpolation. This resizing resulted in a total of 168 features remaining in the spectrogram matrix. The bicubic interpolation technique [159] is employed to estimate values between adjacent data points in a two-dimensional representation offering several advantages, particularly in applications such as feature extraction. It simplifies the process of aligning the data in a standardised format, enabling both quantitative and qualitative analyses to be performed on the recorded signals.

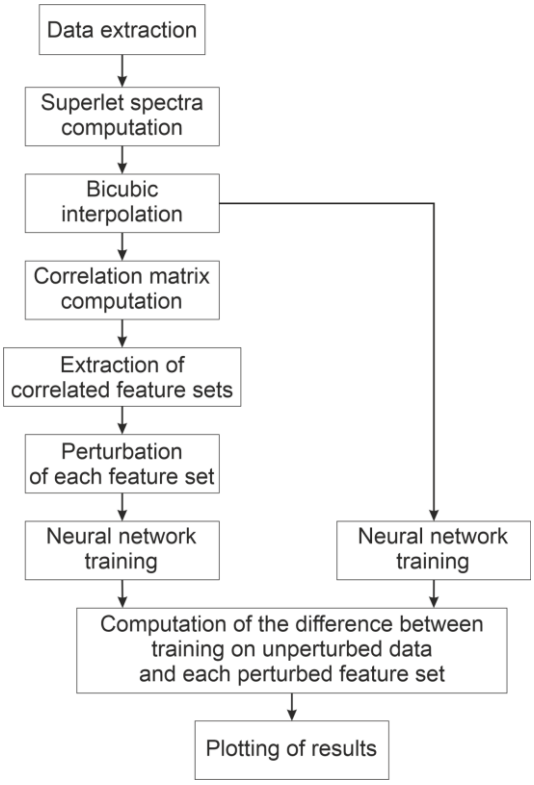

Figure 3.26 - The flow of the correlated feature set perturbation pipeline.

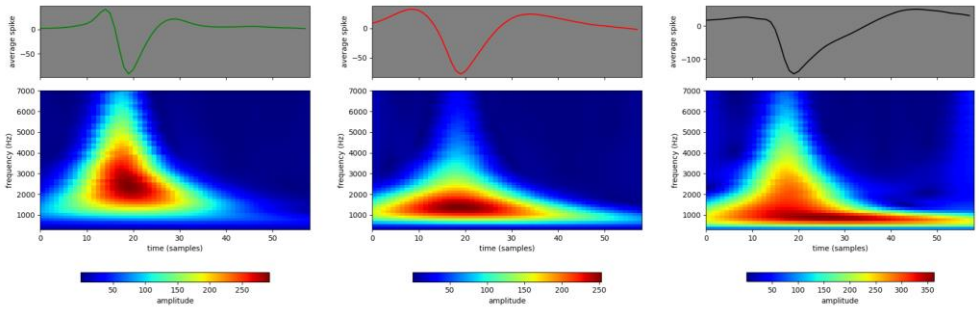

Figure 3.27 - Average spikes and their corresponding spectrograms for 3 different clusters (neurons) of a M045-C4.

The correlation matrix was calculated to visually represent the strength of correlations between the characteristics of the action potentials. This matrix displays the degree of association between all possible pairs of values. Correlation coefficients range from -1 to 1, where -1 indicates a perfect negative correlation, 1 indicates a perfect positive correlation, and 0 indicates no correlation between the coefficients [160]. Figure 3.28 illustrates the correlation matrix using a heat map, where the intensity of the colours (red for high values and blue for low values) represents the magnitude of the correlations between features.

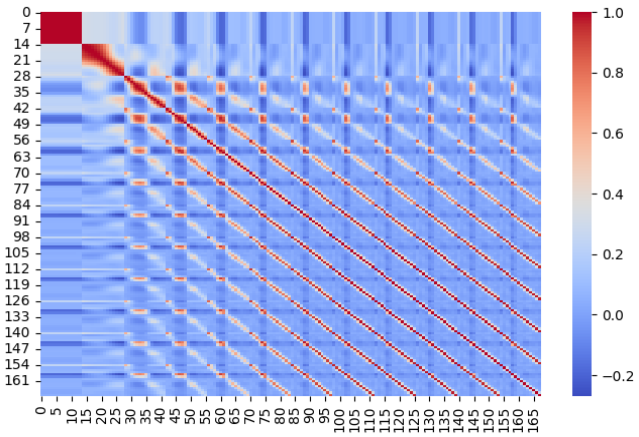

Figure 3.28 - An example of a correlation matrix obtained from M045-C4 real dataset after the downsampling of spectrograms through bicubic interpolation.

In the final step, bicubic interpolation is applied to upscale the perturbation matrix from the reduced shape of [14, 12] back to its original shape of [58, 50]. Figure 3.29 is an example of a perturbation matrix before interpolation and Figure 3.30 shows the matrix from Figure 3.29 after applying interpolation to return it to the original shape of [58, 50].

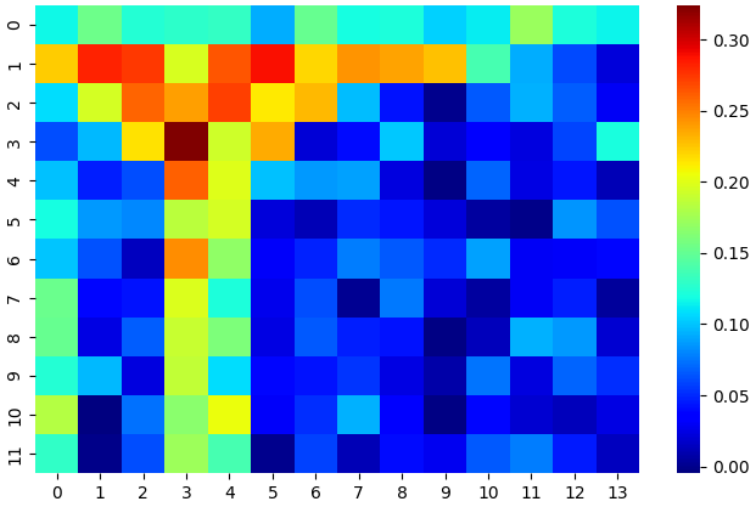

Figure 3.29 - Perturbation matrix showing the effect of perturbation on the learning of a neural network on each feature.

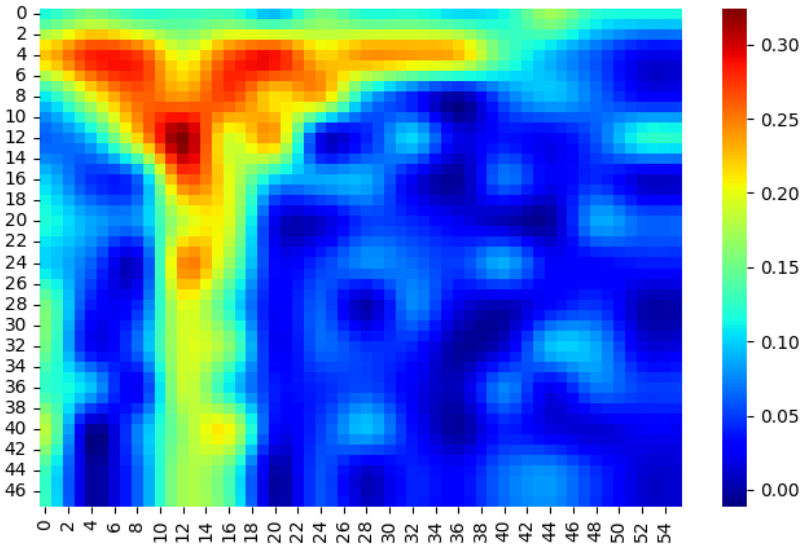

Figure 3.30 - Bicubic interpolation for upsampling to obtain original size.

The feature perturbation method assesses the importance of each feature set in the learning process. Perturbations were applied to each feature set using different thresholds and perturbed them 10, 20, and 50 times. Each feature set contains those features that have a correlation greater than or equal to a certain threshold, and the threshold refers to a value used to decide whether two features are considered correlated or not. After a thorough analysis of the three cases, we noticed significant differences between them. Permuting the features of the spikes 50 times rather than 10 or 20 times, leads to a more stable statistical estimation of the effect of perturbation. Additionally, the more extensive perturbation can help identify features with a greater contribution to correct classification, helping in the elimination of irrelevant features.

In order to determine the most significant spike characteristics for automated learning, they were incorporated into the training of a classifier, in this case a neural network. We used a neural network that features an input layer, a hidden layer with the ReLU activation function, and an output layer with the Softmax activation function. The input layer has the role of receiving the input values of each data set, the number of neurons being determined by the size of a sample of the input data. Regarding the output layer, it produces the final result of the neural network, both for the original set of features and for the disturbed one, and the activation function returns a vector with the length equal to the number of classes in the [0,1] interval and ensures that the sum of probabilities of all classes is 1. The number of classes for a certain data channel is equal to the number of individual neurons observed on the channel and separated during manual spike sorting in order to obtain the ground truth.

Different parameters of the Superlet Transform can impact its performance. Experiments were conducted to select the more performant parameters by testing various combinations and comparing the results. Specifically, the Superlet of order 1 (Wavelet), Superlet of order 2, and Superlet of order 5 were compared on the same real dataset, as depicted in Figure 3.31 on the left panel, middle panel and right panel, respectively. Analysing these visual representations, it was observed that the Wavelet has favourable temporal resolution and the higher order Superlet has favourable resolution

in frequency, while the 2nd-order Superlet exhibits superior precision in multiple areas, including low frequencies, high frequencies, and transient frequencies over time.

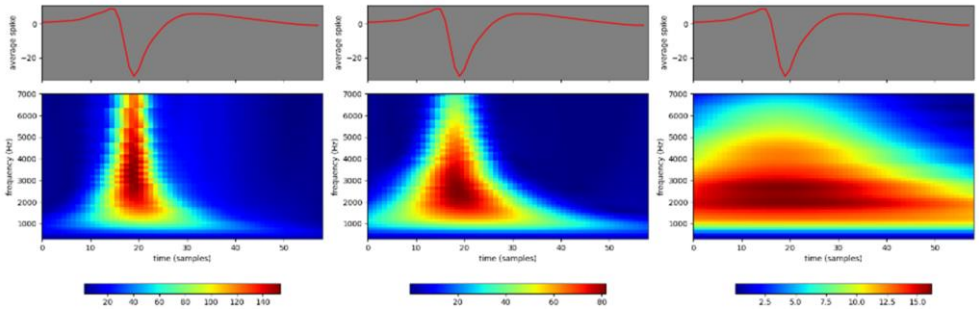

Figure 3.31 - Superlet parameterization impact on a single cluster of spikes from the M045-C4 real dataset, where the heatmap values represent the power values of the spectrogram.

The Superlet Transform has several parameters that may affect performance. Choosing these parameters was done by testing various combinations and comparing results. Thus, we made a comparison between the following parameters: Superlet of order 1 (Wavelet), Superlet of order 2, respectively Superlet of order 5 shown in Figure 3.32 on the same real dataset. Considering these graphical representations, we found that the Wavelet has a good temporal resolution, but the 2nd-order Superlet offers the best precision across multiple areas such as low frequencies, high frequencies, and transient frequencies in time.

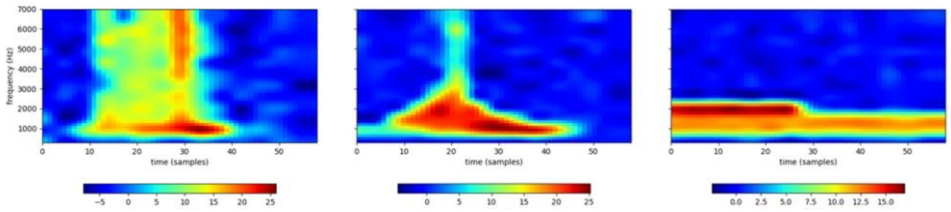

Figure 3.32 - Superlet parameterization impact on the drop in performance, where the values of the heatmap represent the change in performance given by the perturbation of each characteristic at those positions [158].

Parameter dependence analysis

Threshold Dependence

From the comparative analysis we performed in Table 3.11, we can note that when we identify the correlation sets, applying a threshold of 0.3, we obtain a higher performance than in the other cases. This is because the lower the threshold value, the larger the correlated feature sets. Another aspect that emerges from the analysis is that the Accuracy performance metric is not always the best solution to determine how important a feature set is. This analysis has been made on the M045 real dataset. Regarding the values obtained for M045-C8, a big difference can be observed between the results of the 2 metrics, the F1 score having better statistics than accuracy regarding the performance in learning the characteristics of action potentials.

Table 3.11 - Comparative analysis between performance metrics applied to multiple channels with different thresholds on real data.

Data	**ACCURACY**			**F1 SCORE**		
	thr=0.3	thr=0.5	thr=0.8	thr=0.3	thr=0.5	thr=0.8
M045-C1	30%	15%	6%	20%	15%	8%

M045-C8	6%	1.5%	2%	12.5%	6%	8%
M045-C17	40%	8%	6%	25%	7%	5%

Superlet Order Dependance

Based on the results obtained in Table 3.12 for the M045 real dataset, comparing the prediction values for the three orders and applying a threshold of 0.5, a significant difference can be identified. Thus, for the Superlet of order 5, with regard to the values of the two metrics, they are lower than the Superlet of order 1, respectively 2. This demonstrates the fact that, although the Superlet of order 5 offers a better resolution in the frequency domain, the temporal characteristics of the spikes play a crucial role in the process of spike sorting.

Table 3.12 - Comparative analysis between the performance metrics, applied to Superlet of orders 1, 2 and 5 on the M045-C17 real dataset.

THR	ORD	NCYC	ACCURACY	F1 SCORE
0.3	1	1.5	30%	25%
0.5	1	1.5	17.5%	20%
0.3	2	1.5	40%	25%
0.5	2	1.5	8%	7%
0.3	5	3	25%	15%
0.5	5	3	3%	1.5%

Findings

We have explored why the Superlet transform provides useful features for spike sorting. We showed that it is able to isolate important time and frequency components that enable distinguishing spikes of different neurons from one another. It was proven that the Superlet transform of order 2, and the number of cycles 1.5, achieves a better performance regarding spike sorting. This aspect is due to the consideration that the 5th order transform loses its temporal precision, compared to the Wavelet and the 2nd order superlet. The latter seems to offer a sufficient precision both in time and frequency to enable a reliable identification of spikes.

We have found that information about spike characteristics is localized predominantly around the spike peak, extending upwards in frequency, and around a lower frequency component that carries a large fraction of the energy of the spike. Perturbations of this area have the most impact on the learning performance, indicating its relevance to the separability between the activity of different neurons. On the other hand, the amplitude peak of spikes and the temporal width of this peak also seems to be very informative. Indeed, spikes of excitatory neurons are usually wider and larger than spikes that arise in inhibitory neurons [161].

3.1.3.5. A Self-Organizing Map approach for spike sorting

It is normally recommended to normalise all features to the same scale before introducing them to the SOM [74] as to not bias the network towards features that contain higher values or higher variance. In spike sorting, however, action potentials do not have significant differences between them (this is not the case when comparing them with noise). Moreover, information may be encoded in said differences, such as the amplitude, thus normalisation is not recommended in this case. Nevertheless, as a preprocessing step, alignment of the spikes is applied. Figure 3.33 presents the quantization and topographic errors (which have been detailed in the section explaining the self-organising map) of 1000 iterations of training on the Sim4 synthetic dataset. Figure 3.34 presents the results of the SOM training on the Sim4 dataset, alongside with the SOM distance map that represents the average distance between each neuron and its immediate neighbours resulted after training. The distance map can also be visualised in Figure 3.34.

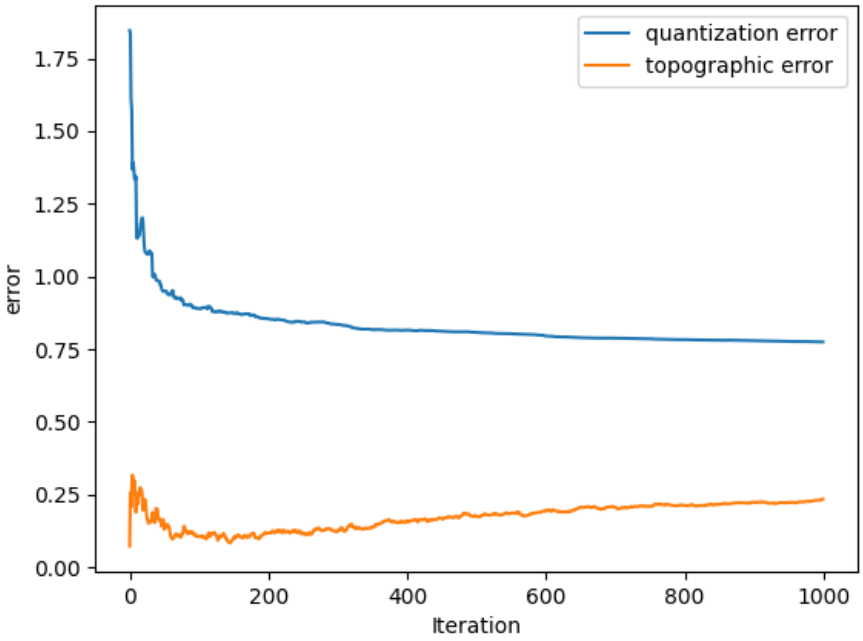

Figure 3.33 - Plot of the quantization and topographic errors during the SOM training across 1000 iterations on the Sim4 synthetic dataset.

Proposed method

Our proposed SOM-based spike sorting method [162] exploits the characteristics of the U-matrix that can be obtained after the SOM training process. Based on the assumption that spikes from the same neuron exhibit greater similarity compared to spikes from different neurons, it is expected that they are located closer to each other in the SOM projection. The U-matrix contains ridges that represent the separation between neurons in the resulting lattice. These ridges serve as markers that define the boundaries of the actual clusters. This can be viewed in Figure 3.35, where the left panel displays the U-matrix resulted from the SOM training on the Sim11 synthetic dataset, and the right panel shows the same U-matrix annotated with ground truth markers represented by different colours for each cluster.

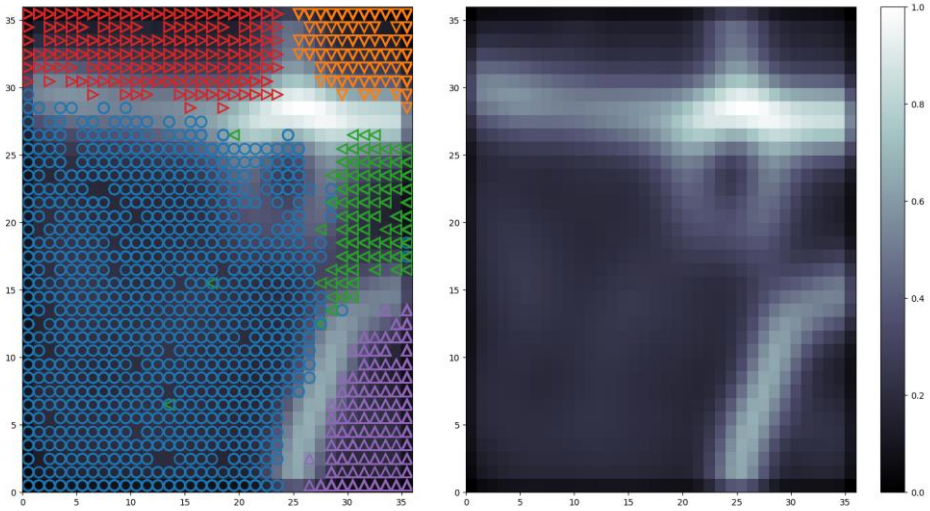

Figure 3.34 - (Left) Result of SOM on the Sim4 synthetic labelled dataset, each colour represents a different cluster in the ground truth. (Right) Plot of distance matrix obtained by the SOM from the training on the Sim4 dataset.

Once the U-matrix is obtained, the clustering problem can be transformed into the task of identifying the ridges, thus identifying the regions separated by them as well. This can be seen as an image segmentation problem, for which various image processing techniques already exist. However, due to the high variability of values within these ridges, most methods may struggle to perform the segmentation. While a convolutional neural network (CNN) approach could be trained for this purpose, it becomes impractical due to the large number of examples required and the need for manual labelling. The extensive dataset and manual labelling process make the CNN approach infeasible in this context.

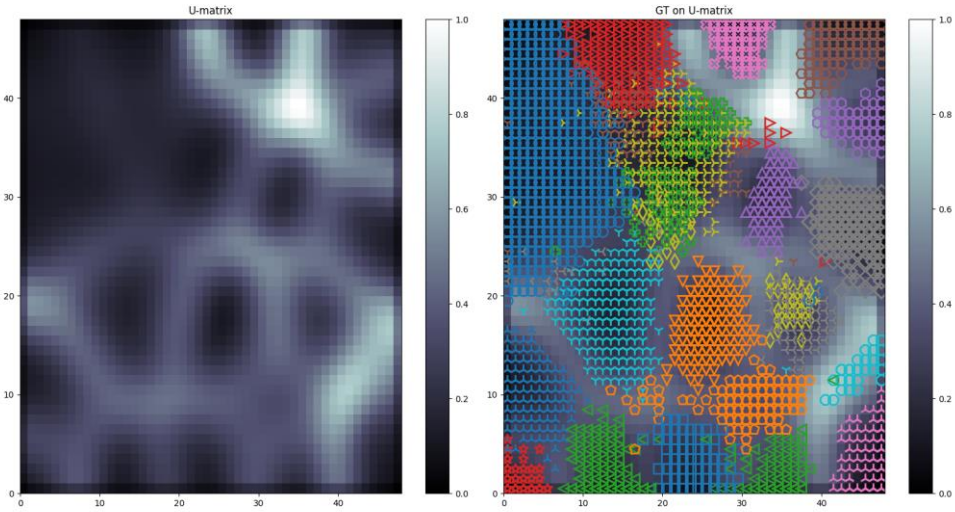

Figure 3.35 - (Left) The resulted U-matrix from SOM training based on the Sim11 synthetic dataset plotted as an image of higher and lower distances denoted by pixel intensity levels, marking the spatial structure of the clusters given by the organisation of the SOM network. (Right) A plot of the resulted U-matrix obtained from the SOM training on the Sim11 synthetic dataset, with the addition of the corresponding ground truth as overlaid annotations over the U-matrix. Each annotation (the individual combinations of colour and shape) represents a distinct cluster.

Our proposed approach utilises simple kernels, specifically horizontal, vertical, and the two diagonal 3x3 kernels, to detect the local maxima. These kernels are applied to generate matrices of the same size, where each cell of the matrix contains a Boolean value indicating the presence or absence of a local maximum at that location. The resulting matrices from each of these kernels are then combined using the logical OR operator. To ensure that the resulting segmentation shape represents a single component, the binary closing operation from image processing is applied to fill any remaining gaps. An automated threshold can be applied to remove lower local maxima that appear as small ridges in the U-matrix which are often associated with sparse clusters. The threshold is determined by calculating the cumulative sum of the histogram of local maximum values and selecting the threshold that surpasses 20% of the total. A specific threshold can also be chosen for each dataset, and in some cases, it offers enhanced performance. Figure 3.36 illustrates an example of the local maxima found on a U-matrix, it can be seen that the local maxima actually follow the ridges of the U-matrix. By comparing with the right panel of Figure 3.36, it can be seen that these ridges actually separate the clusters, thus the local maxima will be used to segment the data into clusters.

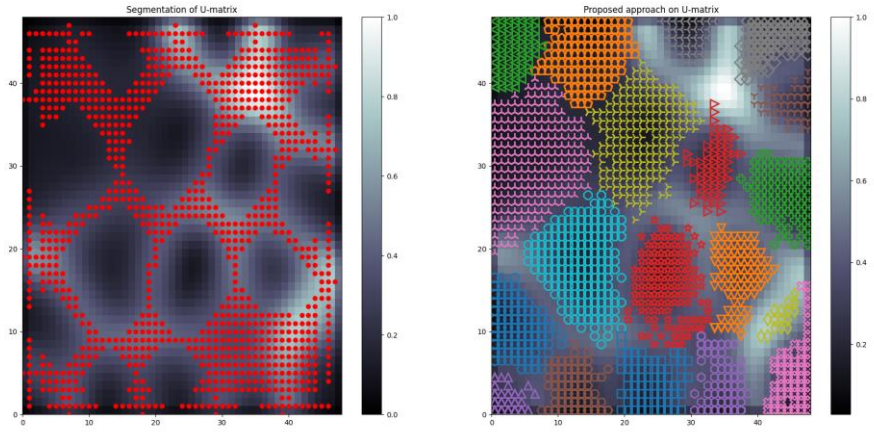

Figure 3.36 - Example of resulted U-matrix local maxima-based segmentation on the same data as in Figure 2. (Left) The computed local maxima with binary closing overlaid as red pixels over the afferent U-matrix. (Right) The label annotations for the resulting clusters determined by the U-matrix local maxima separation method.

By using a simple object detection algorithm, disconnected individual regions can be extracted from the segmented U-matrix , where neighbouring true values form a region, and false values form boundaries. It is worth noting that the identified ridges are typically wider than the actual separation between clusters. To capture as many points as possible, an iterative expansion of each region to its unlabelled neighbours is performed until the regions start connecting, at which point the expansion stops. For easy visual comparison, the obtained labels from Figure 3.36 are projected onto a 2D space using PCA. The ground truth labels of the data are plotted side by side, in Figure 3.37, with the resulting labels from our proposed method (on the same PCA projection).

Our SOM-based spike sorting method was configured as follows:

- **Shape**: The shape of the SOM grid is determined by the formula $2 \cdot \sqrt{5 \cdot \sqrt{N}}$, where N represents the number of samples in the dataset.

- **Features**: The number of features is equal to the number of dimensions of a spike, which is 79 for the synthetic datasets used in this study (Sim9, Sim10, Sim11 and Sim12).
- **Initialization**: PCA initialization is chosen instead of randomised initialization to ensure deterministic results.
- **Sigma**: The sigma parameter is set to 12 for the datasets used in this study as they have a high number of samples. The sharp exponential decay will heavily reduce the sigma parameter throughout training.
- **Learning rate:** Empirical testing suggests that values between $1e^{-1}$ to $1e^{-4}$ yield similar results, and a value of $1e^{-2}$ is selected.
- **Iterations**: The number of iterations is determined by multiplying the number of samples in the dataset by a constant factor. A factor of 5 is set to ensure that each sample updates the weights multiple times.
- **Neighbourhood**: The neighbourhood function used is Gaussian.
- **Distance**: The distance metric employed is Euclidean.
- **Topology**: The SOM grid is configured to have a rectangular topology.

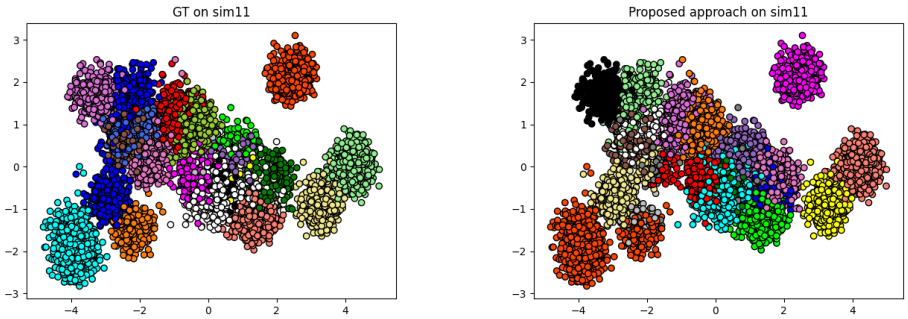

Figure 3.37 - (Left) The ground truth clusters labels mapped to the corresponding data projected in a 2D space by the PCA method. (Right) The resulted clusters' labels mapped to the same corresponding data projected in the PCA-based 2D space.

Performance analysis

The performance analysis of the proposed approach is made in comparison to several combinations of feature extraction methods and clustering algorithms with regards to the metrics presented above on four different synthetic datasets. The presented feature extraction methods have been used to reduce the dimensionality of the data from the original 79-dimensional space to a 2-dimensional space. The two-dimensional space is then given to the clustering algorithm to obtain the labels which are the end result of the spike sorting process. These metrics analyse the amount of overlap between the clustering labels and the ground truth labels. Table 3.13, Table 3.14, Table 3.15 and Table 3.16 display the combination of PCA and Isomap with each clustering algorithm for each of the four datasets. In these tables, the columns indicate the performance metric, while the rows indicate the approach used. The largest values of each column of the tables are bolded to highlight the best results.

The performance analysis indicates that the proposed method can outperform all compared methods across all the datasets and metrics. On the last row of each table, the best threshold, found through a grid search and the performance values obtained with it, is added in italics. The performance can be increased through a manual selection of the threshold. Nevertheless, the automatic threshold has a similar performance, and in some

cases, it even finds the optimal threshold. The choice of feature extraction method appears to have a slight effect on the performance of the clustering algorithms with Isomap increasing the performance by up to 10%, however the performance is significantly lower than the proposed approach. In most cases, the clustering performances of K-Means, Agg and Iso-split are similar, while HDBSCAN has a reasonably lower performance. The visual representation of the result of each method on the Sim11 synthetic dataset can be viewed in Figure 3.38.

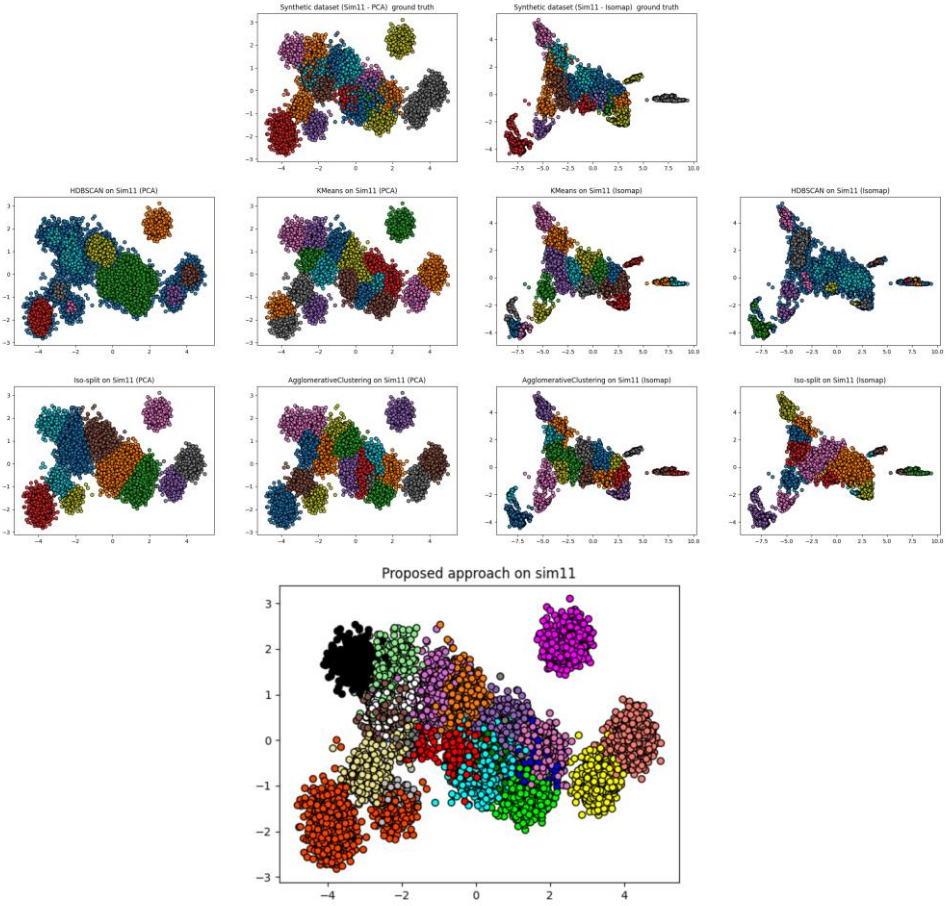

Figure 3.38 - A comparison of all methods on Sim11, on the left side, each clustering method with PCA is presented, while on the right side with Isomap, the result of the proposed approach can be found as the last panel.

Table 3.13 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim9 synthetic dataset.

Sim9	**ARI**	**AMI**	**FMI**	**Purity**
PCA + K-Means	0.509	0.720	0.548	0.756
PCA + Agg	0.500	0.709	0.538	0.591
PCA + HDBSCAN	0.310	0.595	0.43-	0.495
PCA + Iso-split	0.534	0.734	0.615	0.603
Isomap + K-Means	0.532	0.744	0.566	0.734
Isomap + Agg	0.500	0.738	0.537	0.717
Isomap + HDBSCAN	0.461	0.698	0.527	0.615

Isomap + Iso-split	0.555	0.780	0.625	0.695
Proposed method	**0.687**	**0.845**	**0.716**	**0.900**
Proposed method with a threshold of 15	*0.753*	*0.868*	*0.773*	*0.903*

Table 3.14 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim10 synthetic dataset.

Sim10	**ARI**	**AMI**	**FMI**	**Purity**
PCA + K-Means	0.446	0.708	0.485	0.689
PCA + Agg	0.432	0.689	0.471	0.656
PCA + HDBSCAN	0.314	0.609	0.461	0.466
PCA + Iso-split	0.336	0.676	0.505	0.482
Isomap + K-Means	0.558	0.789	0.59	0.756
Isomap + Agg	0.550	0.783	0.582	0.758
Isomap + HDBSCAN	0.387	0.718	0.521	0.557
Isomap + Iso-split	0.476	0.763	0.54	0.598
Proposed method	**0.693**	**0.873**	**0.717**	**0.811**
Proposed method with a threshold of 10	*0.815*	*0.896*	*0.837*	*0.814*

Table 3.15 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim11 synthetic dataset.

Sim11	**ARI**	**AMI**	**FMI**	**Purity**
PCA + K-Means	0.479	0.735	0.518	0.747
PCA + Agg	0.483	0.723	0.520	0.717
PCA + HDBSCAN	0.346	0.667	0.492	0.554
PCA + Iso-split	0.434	0.745	0.520	0.591
Isomap + K-Means	0.576	0.804	0.607	0.811
Isomap + Agg	0.600	0.801	0.630	0.814
Isomap + HDBSCAN	0.472	0.744	0.552	0.699
Isomap + Iso-split	0.496	0.781	0.592	0.685
Proposed method	**0.774**	**0.889**	**0.793**	**0.907**
Proposed method with a threshold of 5	*0.774*	*0.889*	*0.793*	*0.907*

Table 3.16 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim12 synthetic dataset.

Sim12	**ARI**	**AMI**	**FMI**	**Purity**
PCA + K-Means	0.450	0.712	0.497	0.710
PCA + Agg	0.461	0.705	0.507	0.726
PCA + HDBSCAN	0.250	0.600	0.442	0.486
PCA + Iso-split	0.496	0.741	0.616	0.578

Isomap + K-Means	0.550	0.781	0.588	0.772
Isomap + Agg	0.558	0.781	0.596	0.769
Isomap + HDBSCAN	0.574	0.757	0.633	0.686
Isomap + Iso-split	0.609	0.801	0.678	0.707
Proposed method	**0.850**	**0.877**	**0.863**	**0.832**
Proposed method with a threshold of 10	*0.850*	*0.877*	*0.863*	*0.832*

Findings

Our proposed SOM-based spike sorting method [162] has demonstrated its effectiveness when dealing with datasets with a high number of clusters that are imbalanced and overlapping. This method successfully addresses the two most challenging steps of the spike sorting pipelines, namely feature extraction and clustering. In the comparison with other common methods, through the perspective metrics such as ARI, AMI, FMI, and Purity, the proposed method consistently outperformed all combinations of feature extraction and clustering methods. The choice of threshold for local maxima does have an impact on the results, but the option for the automatic threshold selection provides comparable or slightly lower performance. Thus, the proposed method provides an effective solution for spike sorting with higher performance than other conventional methods, particularly for datasets characterised by imbalanced and overlapping clusters while using an outside-the-box procedure that can substitute both the feature extraction and clustering steps of the spike sorting pipeline.

3.1.3.6. Space Breakdown Method

One of our earliest research works in field of neuroscience is the Space Breakdown Method, or SBM [163], a grid-based clustering algorithm that I developed during my Bachelor's Work. SBM was designed with Spike Sorting in mind and as such, it works on the assumption of a unimodal distribution of clusters, where the centre of the cluster is also the densest. Through the use of a grid, SBM can reduce any number of samples to a chosen interval, thus reducing the sample space, while retaining information about the distribution of the data as their densities.

SBM requires two parameters: a partitioning number (PN) and the minimum number of points of a centroid. The minimum number of points of a centroid was introduced as a threshold in order to avoid the labelling of low amounts of clumped points as clusters when in reality they may be noise. This parameter should be adjusted according to the data. The partitioning number is the main parameter of the algorithm, and it represents the number of bins, called chunks. It is easier to think of these chunks in the terms of the bins of a histogram. Similarly to a histogram, SBM divides the data into subsections of a fixed length and retains the number of points in each chunk.

The algorithm consists of five main steps: normalisation, chunkification, centroid search, expansion of the centroids and dechunkification. It starts by normalising the data between zero and the partitioning number [0, PN]. By taking chunks of a fixed size of 1, there are PN chunks created. For the particular case of 2 dimensions, it can be viewed as a matrix of size PNxPN and the chunks as squares, where just as in a histogram, each chunk retains the number of points enclosed within the fixed interval. Therefore, for the

cell 0x0 in the matrix, it counts the number of points that are between the [0, 1) interval in both dimensions. For this case of 2 dimensions, the chunkification is the conversion of the original data into said matrix. And similarly, for the case of 3 dimensions, the chunks could be viewed as cubes.

The centroids of clusters are chunks that contain a higher number of points than all the neighbouring chunks, as such finding the local maxima. Chunks are considered neighbours by adjacency and validated for existence, for instance, the (0,0) cell, would have the following neighbours: (0,1), (1,0) and (1,1). With the additional condition that the centroid needs to have more points than the minimum threshold, that is a parameter. Once all possible cluster centroids have been discovered, the expansion of the clusters starts from their centroids. Perpetually through the expansion, a unimodal distribution is assumed, as a new chunk is required to be smaller than the point of expansion as the expansion becomes more distant with respect to the centroid in order for it to continue.

SBM does not require the number of clusters as input. In addition, it has a linear time complexity with regard to the number of samples. But it has an exponential time complexity with regard to the number of dimensions. Therefore, it requires the feature extraction step to reduce the number of dimensions and improve the processing time. Furthermore, the consideration of unimodal distributions as clusters limits the algorithm's ability to identify specific types of clusters, such as uniform or multimodal. Nevertheless, in comparison with other algorithms such as K-Means or DBSCAN, its performance is higher when dealing with overlapping and imbalanced clusters. The overall complexity of the algorithm is $O(n + PN^d)$, where n is the number of samples, PN is the partitioning number and d is the number of dimensions of the data.

Limitations of the Space Breakdown Method

Let there be a dataset with n samples in a d dimensional space and PN the partitioning number given as a parameter to the SBM algorithm. Deducibly, the chunkification step of SBM creates a d-dimensional array of size PN^d to store the new space. Naturally, such a space becomes impossible to store as the number of dimensions gets higher while also increasing the complexity of the algorithm exponentially. For example, let there be a dataset with 10,000 samples and $d=10$ dimensions, even for a low value for the partitioning number $PN=5$, it requires a space of $PN^d = 5^{10} = 9{,}765{,}625$ to store the auxiliary structure and majority of these chunks contain a value of zero that do not affect the result but increase both the space and time complexity of the algorithm.

3.1.3.7. Improved Space Breakdown Method

Proposed improvements

To address the limitations of our initial work, the SBM clustering algorithm [163], we propose two improvements [164]. The first improvement brought to the algorithm can be encompassed by a transition to a graph structure from the original *d*-dimensional array structure. As the concept of neighbours and that of BFS are available for the graph structure, the algorithm functionality remains unchanged, while the storage space and the complexity are reduced. The pseudocode of the improved version remains similar to that of the original algorithm [163,164]:

```
SBM(dataset, PN, threshold):
   1. X = normalise(dataset, PN)
   2. graph = chunkification(X, PN)
   3. ccs = findCentroids(graph, threshold)
   4. for cc in ccs:
   5.     expand(graph, cc, label, ccs)
   6. labels = dechunkification(graph, X)
   7. return labels
```

By changing the approach of the clustering algorithm from an array structure to a graph structure, the performance remains unaffected but this can limit the maximum numbers of chunks created to the number of samples of the dataset resulting in a significant pruning of the storage space required as shown in Figure 3.39, especially for hyper-dimensional data.

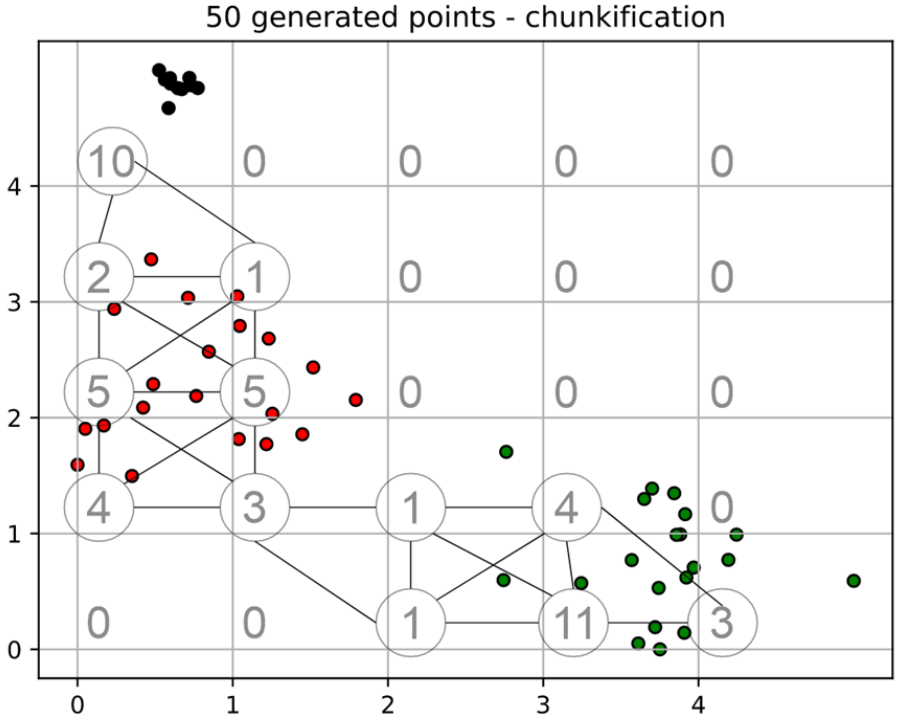

Figure 3.39 - A comparison of the chunkification step of SBM and ISBM, the space presented is already normalised as the first step of the algorithms in the [0,5) interval. Every cell in the grid shown contains a number in the lower-left representing the value of the chunk in both SBM and ISBM chunkification. The

difference appears in the final structure, while SBM saves the whole array, ISBM only retains the values that are encircled, where circles represent the nodes and the lines the edges of the final graph structure.

The second improvement regards the partitioning number. The original version partitions the space equally for all dimensions. The improved version only has the PN separation on the most informative dimension. In this case, the most informative dimension is considered as the one with the highest variance. Consequently, each dimension has a partitioning number scaled proportionally with the amount of variance it has in comparison to the highest variance.

Complexity analysis

The analysis of the algorithm includes the space complexity evaluation between the original version and the improved version shown in Table 3.17. In this evaluation, the Sim4 synthetic dataset was used. For the evaluation, the dimensions were reduced using PCA.

Table 3.17 - Evaluation of the number of chunks/nodes by varying the PN parameter on the Sim4 synthetic dataset.

Number of dimensions	**SBM (array structure)**	**First improvement (graph structure)**	**ISBM (both improvements)**
2	625	343	188
3	15,625	1659	321
4	390,625	4072	532
5	9,765,625	4981	744
6	244,140,625	5111	988

The time complexity analysis includes both the variation of the number of dimensions and the variation of the number of samples. For the dimensionality influence on the complexity the evaluation was made on the Sim4 synthetic dataset. PCA was used to reduce the dimensionality of the dataset from 2 to 6, the time evaluation is shown in Table 3.18. For the influence of the number of samples, a 2-dimensional synthetic dataset was used (see Synthetic Data section). The evaluation of the execution time is shown in Table 3.19. Neither of the improvements affect the linear time complexity in relation to the number of samples.

Table 3.18 - Evaluation of the execution time (in milliseconds) for 100 runs for the number of dimensions on the Sim4 dataset using PCA to reduce dimensionality to the chosen values.

	KMeans	DBSCAN	MeanShift	Agglom.	FCM	HDBSCAN	ISO-SPLIT	SBM	ISBM
2	39	36	1450	475	68	84	83	133	**31**
3	**43**	39	1685	496	80	168	143	161	72
4	**38**	43	1946	515	119	205	145	445	146
5	**43**	50	2416	526	187	253	135	2204	652
6	**43**	56	3118	546	334	306	151	59311	2780

Table 3.19 - Evaluation of the execution time (in milliseconds) for 100 runs by varying the number of samples.

	KMeans	DBSCAN	MeanShift	Agglom.	FCM	HDBSCAN	ISO-SPLIT	SBM	ISBM
4300	48	0.068	1650	254	204	66	65	79	**38**
8600	**54**	174	2154	1183	384	143	78	119	60

12900	**79**	300s	2389	2930	636	221	103	184	**79**
17200	**91**	485	2636	5133	952	345	141	227	102
21500	**107**	686	2728	7970	1224	496	156	263	119
25800	**124**	873	2911	12075	1637	622	198	305	138
30100	**155**	1153	3295	16499	2012	705	236	361	165
34400	**166**	1564	3613	21577	2323	798	278	412	187
38700	**200**	1783	4040	29510	2537	947	312	441	205

Spike Cluster Score

In an exploratory examination, the state-of-the-art clustering performance metrics previously presented have been found unsuitable for the clustering of neuronal data. These metrics are biassed towards punishing overclustering, which is an acceptable case for spike sorting as an expert observer can merge the cluster as a post-processing step. The only exception is the Purity metric [137,138] that through its design does not punish overclustering, as the labelling of each sample as an individual cluster results in a perfect score. A second reason to discount classical clustering performance metrics is that algorithms such as DBSCAN [121] that produce noise labels is penalised.

Therefore, a new metric was proposed [164], designated as Spike Cluster Score (SCS) that does not punish overclustering but penalises underclustering. SCS is an external metric and thus, requires the ground truth. It is calculated as the arithmetic mean of the agreement between a true label *T* and its best matching predicted label *P*. For every true label T_i, let $P(T_i)$ be the corresponding predicted labels of the samples with the true label T_i. Thus, the total number of occurrences of a predicted label *j* within all samples of true label *i* is defined as $count(P(T_i) = P_j)$. The score of a singular label is therefore, computed as the best matching, by count, predicted label divided by the total number of the indicated predicted label. Intuitively, this calculates how many of the points of a predicted label are part of a single true label. The overall performance is given by the following formula:

$$SCS = \frac{1}{C}\sum_{i=1}^{C}\frac{count(P(T_i) = P_j)}{count\left(P = P_j\right)} \tag{37}$$

where *C* is the total number of true clusters, *i* is the current cluster and P_j is the predicted label with the highest count.

This metric is similar to Purity, as it evaluates the case of each sample as its own cluster as a perfect clustering. The disadvantage brought by Purity, that SCS is not affected by, is that it does not punish underclustering. Subsequently, the performance score given by Purity had a low variance across algorithms. Furthermore, SCS is unaffected by noise points as they do not change the performance score, while other metrics require the removal of noise points to obtain a correct evaluation.

Within spike sorting, it is the responsibility of the experimenter to determine the acceptable degree of overclustering for each situation. SCS is not the gold standard, it is recommended to be used with other performance metrics as each evaluates the clustering performance from its own perspective as every method has its own strengths and weaknesses. A comparison of the metric versus the state-of-the-art performance metrics can be viewed by analysing Figure 3.40 and Table 3.20.

Performance analysis

The clustering performance of the algorithm was evaluated using the multiple clustering performance metrics against the original version and multiple clustering algorithms. The performance dataset was evaluated on the Sim4 synthetic dataset which was reduced to 2 dimensions through PCA, the result of each algorithm is presented in Figure 3.40 and the scores of the clustering performance metrics including SCS are presented in Table 3.20.

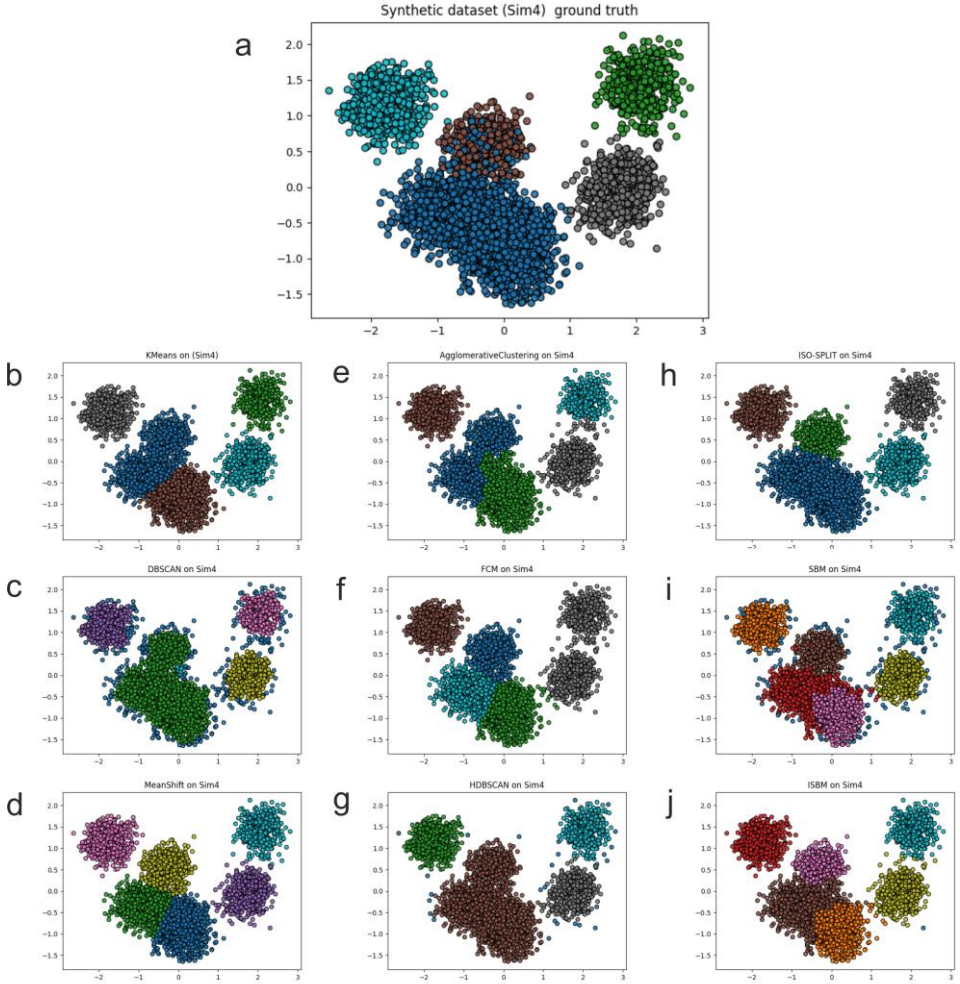

Figure 3.40 - (a). Simulation 4 (Sim4), a synthetic dataset with its ground truth (b). The clustering of K-Means on the dataset (c). The clustering of DBSCAN on the dataset (d). The result of MeanShift on the Sim4 dataset (e). The clustering of Agglomerative Clustering on the Sim4 dataset (g). The clustering of HDBSCAN on the dataset (f). The clustering of FCM on the dataset (h). The clustering of ISO-SPLIT on the dataset (i). The clustering of the original version of SBM on the dataset (j). The clustering of the improved SBM (ISBM) on the dataset.

Table 3.20 - Evaluation of clustering algorithms using various performance metrics on Sim4.

	KMeans	DBSCAN	MeanShift	Agglom.	FCM	HDBSCAN	ISO-SPLIT	SBM	ISBM
ARI	57.3	75.5	58.6	59.1	52.6	79.9	**91.7**	77.5	80.2
AMI	79.6	77.9	80.5	77.7	73.4	87.0	**91.6**	80.2	85.0
Purity	91.3	88.6	**96.3**	90.9	87.1	91.0	**96.9**	90.5	92.0
FMI	71.1	85.2	72.3	72.4	67.6	88.5	**94.6**	85.3	87.1
VM	79.6	77.9	80.5	77.8	73.5	87.0	**91.6**	80.2	85.0

SCS	85.6	80.0	94.0	85.0	85.2	79.8	**94.9**	93.8	**94.9**

The evaluation of the algorithm would be incomplete without taking into consideration its performance on real data. Figure 3.41 presents a single channel of a recording from the brain of a mouse, the dataset is denoted by M045-C1. The data has been filtered and PCA was used to reduce the dimensionality to only 3 dimensions. As real data contains no ground truth, K-Means was used to generate a reference. It generates a natural bias towards K-Means in the performance that can be seen in Table 3.21.

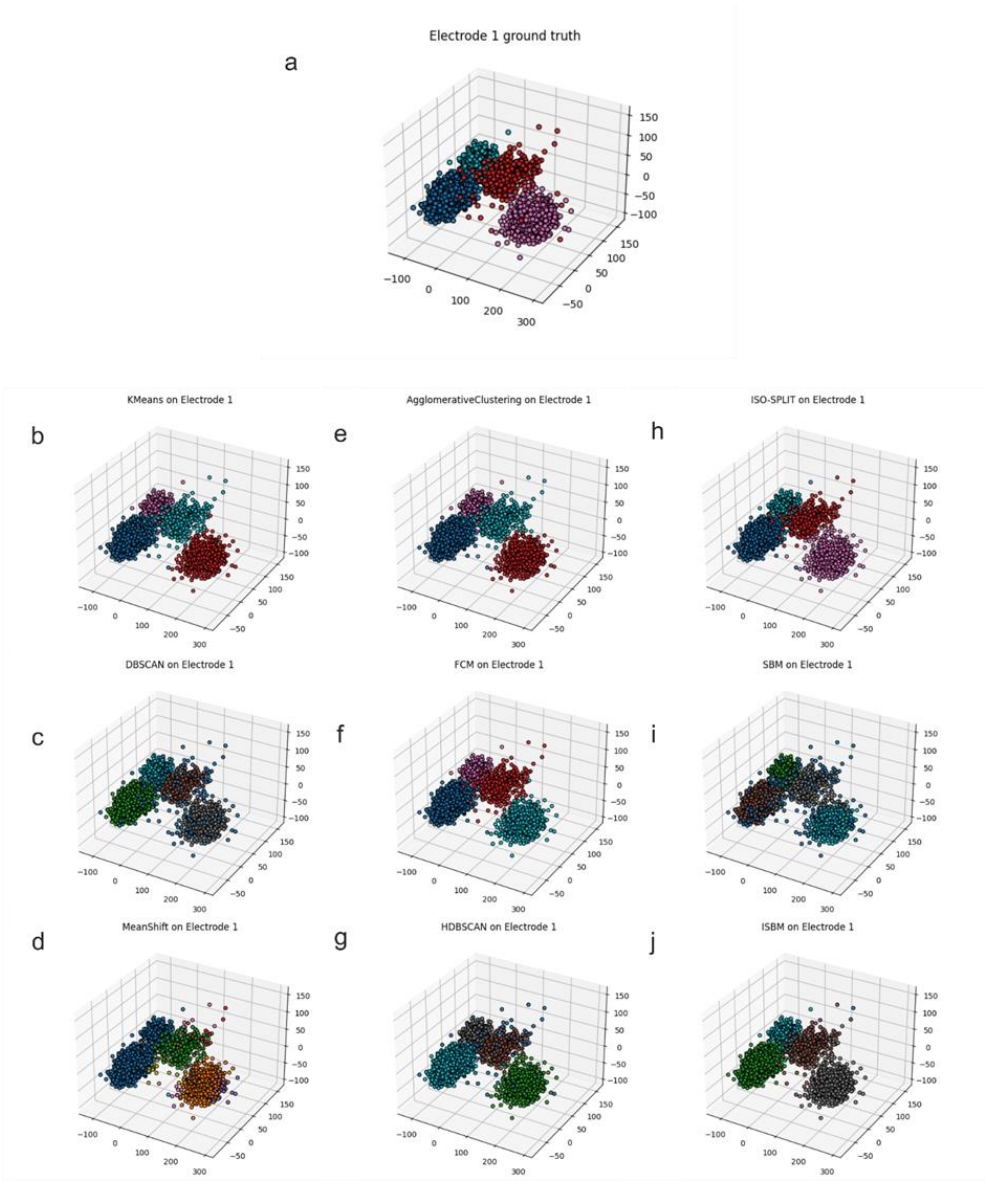

Figure 3.41 – The M045-C1 real dataset that has been cleaned and filtered and recorded from the brain of a mouse (a). Ground truth generated through K-Means (b). The clustering of K-Means on the dataset (c). The clustering of DBSCAN on the dataset (d). The result of MeanShift on the Sim4 dataset (e). The clustering of Agglomerative Clustering on the Sim4 dataset (g). The clustering of HDBSCAN on the dataset (f). The clustering of FCM on the dataset (h). The clustering of ISO-SPLIT on the dataset (i). The clustering of the original version of SBM on the dataset (j). The clustering of the improved SBM (ISBM) on the dataset.

Table 3.21 - Evaluation of clustering algorithms using various performance metrics on M045-C1.

	K-Means	DBSCAN	MeanShift	Agglom.	FCM	HDBSCAN	ISO-SPLIT	SBM	ISBM
ARI	98.8	97.0	97.7	98.4	53.7	98.0	**98.9**	49.3	98.2
AMI	96.3	90.5	91.1	95.8	70.8	92.9	**96.6**	64.0	93.9
Purity	**99.3**	97.8	99.0	99.2	91.5	98.8	**99.3**	85.9	98.7
FMI	**99.4**	98.4	98.7	99.1	71.9	98.9	**99.4**	84.0	99.0
VM	96.4	90.5	91.2	95.8	70.8	92.9	**96.6**	64.0	93.9
SCS	98.9	96.5	**99.5**	98.9	74.8	98.6	99.0	62.2	97.2

Findings

We introduced an improved version of the original SBM clustering algorithm and investigated its performance by comparing it to the original version and other algorithms. Through the improvements presented here, the space complexity of SBM has been significantly improved. Moreover, the processing time has been further reduced for high-dimensional data, allowing the algorithm to be used on high-dimensional datasets. With the addition of these improvements, the linear scalability of SBM with regard to the number of samples has not been changed and its performance on neural data has been increased, being able to outperform other methods on multiple datasets and on almost all the metrics we have used.

The first improvement presented here tackles the space and time complexity regarding the number of dimensions, but without affecting the linear time complexity regarding the number of samples. The space complexity of SBM has been reduced from $O(PN^N)$ to only $O(n)$, where PN is the partitioning number, N is the number of dimensions, and n is the number of samples. The overall time complexity of the algorithm has been reduced to $O(n + (V+E))$, where n is the number of samples, V is the number of nodes, and E is the number of edges between the nodes of the graph. This complexity is still exponential because, as the number of dimensions increases, the number of edges increases exponentially. However, this second improvement has achieved the goal of increasing the accuracy of the algorithm for neural data. Thus, we have improved the complexities of the algorithm while also increasing its accuracy on overlapping and imbalanced clusters.

3.1.4. Conclusions

The methods presented in this chapter bring improvements to the spike sorting pipeline that allows in time a better understanding of brain functions. The steps of spike detection, feature extraction and clustering have all been addressed and new methods proposed for each. Each of the proposed methods has been thoroughly validated through analyses using several performance metrics and datasets.

Several studies have been conducted within the domain of spike sorting. Spike detection was improved by combining classifiers with threshold-based approaches enhances performance but requires retraining for each recording, limiting real-world applications. A method based on Self-Organizing Maps (SOM) was proposed to address both the feature extraction and clustering steps of spike sorting that proved highly effective for datasets featuring numerous clusters with imbalanced and overlapping characteristics. We have proposed the Superlet Transform as a feature engineering approach for spike sorting and its analysis in spike sorting demonstrates that it can outperform traditional methods as its features enhance cluster separability. The best performance is achieved by using minimum values for Superlet parameters, and the performance is consistent for datasets with different cluster numbers. Autoencoder variants are proposed as a feature extraction method in spike sorting and have been shown to consistently outperform other commonly used feature extraction algorithms. Although training autoencoder models incurs additional costs, their improved performance justifies this expense. Regarding the clustering step, we have proposed an approach called ISBM which outperforms other clustering algorithms in neural data clustering. ISBM is an improved version of a simpler proposed clustering algorithm called SBM, which enhances performance and scalability, making it practical for large datasets. Throughout this work, we have demonstrated that evaluation metrics are not perfect and in line with this, we propose our own metric developed specifically for spike sorting. Moreover, we propose a new approach for distance computation that is applicable in both clustering and performance evaluation that has shown improved performance for non-convex clusters and similar performance for convex clusters in comparison to traditional approaches.

3.2 Burst detection

3.2.1. Introduction

Neurons within the communicate with each other through action potentials, also known as spikes. Neurons have two firing patterns: tonic mode, characterised by individual spikes occurring at relatively distant time intervals, and burst mode, where neurons tend to discharge sequences of spikes in rapid succession. Gaining an understanding of the dynamics underlying neuronal burst activity is essential to understand the mechanisms involved in neural information processing. However, the detection of bursts in extracellularly recorded neural data poses significant challenges. Thus, the process of burst detection is closely tied to that of spike sorting, the filtering and the spike detection steps are also used in the detection of bursting activity.

A neuronal burst is defined as a segment of signal representative of brief time period in which there is heightened occurrence of spikes from a single source neuron. They differ from high-frequency tonic activity in which the activity is produced by multiple source neurons. However, intervals between bursts are juxtaposed with intervals of low-frequency tonic activity. Burst detection algorithms can be categorised into two primary types, yet all these methods rely solely on the analysis of spike timings to discern the presence of bursts. Rate-threshold-based approaches employ a predetermined threshold of firing rates to identify whether the neuronal activity surpasses the threshold. If this criterion is met, the activity is classified as a burst. Inter-spike interval (ISI)-based methods examine the duration between consecutive action potentials. If the duration falls below a predefined threshold, it is classified as a burst. Other methods use a combination of these first two. Hence, these techniques are founded upon the computation of statistical properties inherent in the data and the utilisation of either manually determined or calculated thresholds to identify bursts.

A key feature of bursts is the time interval between two consecutive spikes (action potentials) within the burst. This interval is utilised in ISI-threshold algorithms, which set a range for this interval based on literature findings. Typically, the minimum interval ranges from 2-3ms, while the maximum interval ranges from 7-9ms [28]. Analysing the time interval between spikes is thus critical for the detection of bursts.

The literature discusses several other characteristics related to bursts. For example, it is observed that within a burst, the neuron is unable to generate spikes of the same amplitude, and there is only a gradual decrease in amplitude between consecutive action potentials [166]. During bursts, neurons do not return to the resting state between spikes. Bursts can be triggered by a slow depolarizing mechanism (such as the T-type calcium ionic current) that prevents the repolarization of the neuron and the initiation of the refractory period, allowing for the generation of consecutive spikes. Signals that do not exhibit these characteristics have not been considered as burst candidates.

Several approaches have been proposed for burst detection; however, none of them have gained widespread adoption within the field. This can be attributed to the inherent complexity of many algorithms, which often necessitate the adjustment of multiple parameters, the inclusion of additional criteria to yield satisfactory results and additional steps of postprocessing. Among the methods under consideration are ISIn [146], ISI Rank threshold (IRT) [147], Max Interval (MI) [148], Cumulative Moving Average (CMA) [149], Rank Surprise (RS) [150], and Poisson Surprise (PS) [151]. However, the disadvantage of these approaches is that they rely solely on the timing of

action potentials, thereby disregarding valuable information inherent in the signal. The objective was to devise a novel method that incorporates the established criteria for burst characteristics outlined in the literature; thus, enhancing detection by using as much available information as possible from the recorded data.

The primary challenge lies in effectively distinguishing between close action potentials originating from different neurons, a phenomenon referred to as superposition, and consecutive action potentials generated by the same neuron within a short timeframe, which are known as bursts. Addressing this challenge involves exploring approaches that aim to differentiate the waveform shapes of action potentials from distinct neurons through correlation in the time-frequency domain. Our proposed hypowork [167] was that the statistical analysis of the correlation coefficient could be used to differentiate between the sub-spikes of a burst and the tonic activity. This is based on the observation that sub-spikes within a burst share similar shapes as they originate from the same neuron. Therefore, it was expected that burst sub-spikes would exhibit a higher correlation compared to spikes from tonic activity. The aim of analysing the correlation coefficients was to establish a threshold that would enable the separation of bursting activity from tonic activity.

3.2.2. Data

Benchmark synthetic burst data

A part of the evaluation was made using synthetic burst data [168], consisting of labelled simulations spanning 300 seconds, enabling the assessment of method correctness. Different types of data were generated, including non-bursting and non-stationary data without bursts, as well as regular, long, high frequency, and noisy data containing bursts. Each data type consisted of 100 spike train examples. Figure 3.42 displays a representative spike train from each data type.

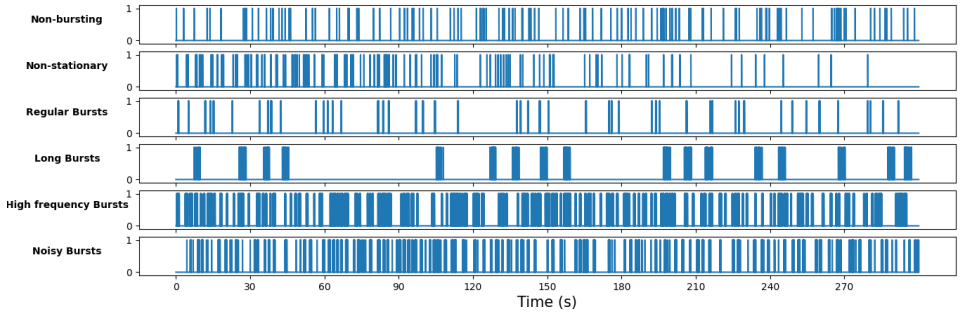

Figure 3.42 - Types of bursts, each subplot shows a different type of simulated data [168] indicated by the label on the left of each subplot. The data is composed of timestamps and as such 1s indicate activity and 0s no activity.

Real neural data

The data presented here was recorded by the Transylvanian Institute of Neuroscience with 32-channel probes from the visual cortex of mice (details about the data acquisition procedure can be found in the Appendices). In the analyses that can be found throughout this subchapter several real datasets have been used, as follows:

- M029 – recorded with a 32-channel probe with a sampling frequency of 32kHz for ~10 minutes resulting in a signal composed of 19,456,000 samples. The raw signal was filtered within the [300, 7000] frequency band, where spiking activity is

typically observed [2]. The identification of individual spikes is carried out using an amplitude threshold, typically set as a multiple (between 3 and 5 [2]) of the signal's standard deviation.

- M029-C23 – represents the 23-rd channel of the M029 dataset.

3.2.3. Methods

Proposed method

In our proposed approach [167], burst candidates are identified by iterating through the signal and finding peaks that exceed the threshold (usually in the negative potential direction). The distance between peaks is configurable. Based on the literature, we have chosen the 2 to 7ms interval. In Figure 3.46, the red dots are the peaks, and they were extracted by finding local minima on a neighbourhood of ~0.3ms. In addition, we imposed that the values of the peaks of each subsequent sub-spike within a burst must have a higher value than the preceding one (decreasing amplitude).

The Superlet Transform allows us to extract time and frequency information simultaneously and enables for a better characterization of a burst, burst candidate, or spike. To obtain the spectrum, the following parameters were used: order 2, 1.5 number of cycles, and a sampling frequency of 32000. The spectrum is visualized within the frequency range of spikes of 300 Hz to 7000 Hz. These parameters were found to be optimal for the classification of spikes [157]. Figure 3.43 shows an example of the spectrogram obtained by the Superlet transform and its corresponding burst signal.

To split the spectrograms, our initial approach was to divide the signal first and then generate spectrograms for each segment. However, this approach is flawed because abrupt transitions at the edges of the signal can generate border effects that distort the spectrogram's edge. One solution to mitigate the border effects is to pad the signal with zero values or gradually transition towards zero from the last value. However, this solution is computationally more expensive compared to calculating the spectrogram for the entire signal, where there are no border effects, and then selecting the sub-matrices of interest afterward.

Another problem we had to solve is that the separation of burst candidates into spikes based on local minima results in action potentials of different lengths, which consequently leads to spectrograms of different lengths, leading to a different number of columns.

To analyse individual sub-spikes within bursts and calculate correlations, the burst candidate spectrogram needs to be divided into smaller spectrograms specific to each spike in order to allow the analysis of individual sub-spikes within bursts and enable the calculation of the correlation. By examining sub-spikes from the same channel and comparing with sub-spikes from different channels located at a relatively large distance, we computed the distributions of the expected correlations between spectrograms of spikes from the same and from different neurons. Because at a large distance, it is very unlikely to record spikes from the same neuron, across channel correlations give us the expected value of correlation between spike spectrograms of different neurons. The distribution of correlations between spectrograms of spikes within channels should in principle have a larger expected value than that of correlations across distant channels.

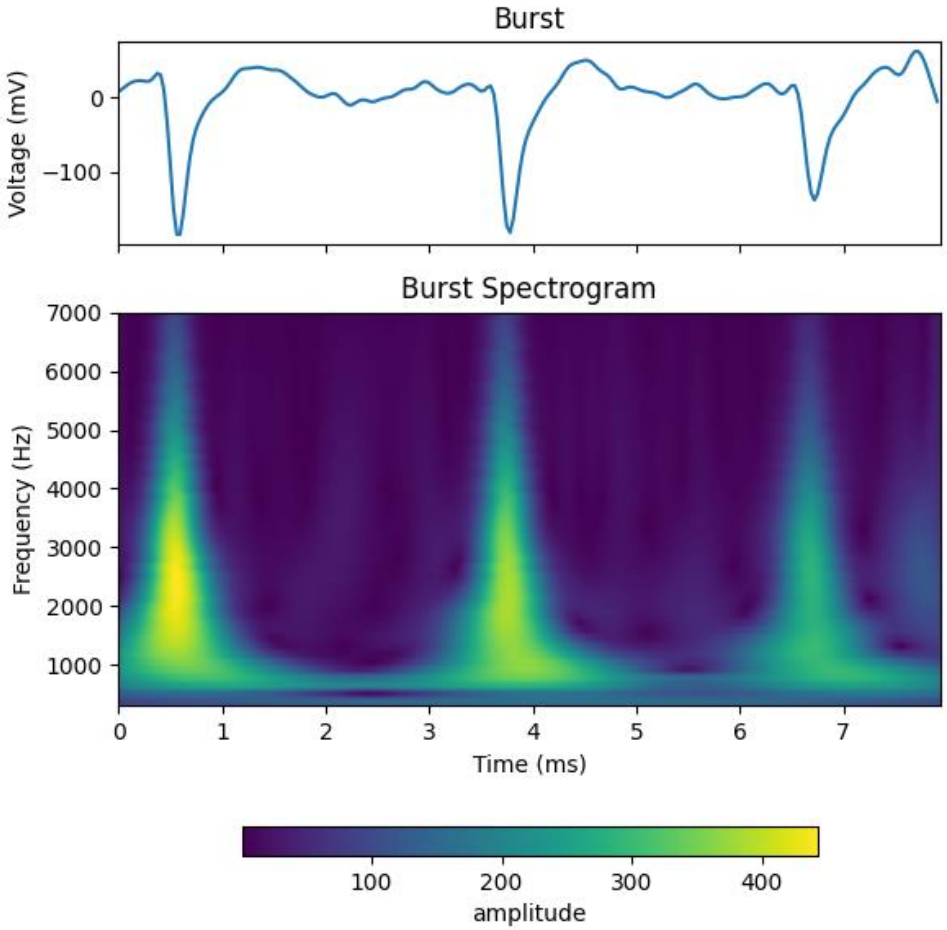

Figure 3.43 - Spectrogram of a burst candidate, extracted from the M029-C23 real dataset by the proposed method, containing three sub-spikes.

To calculate the correlation coefficient between two spikes, we need spectrograms of the same size so that the correlation will result in a single value. The method used to separate burst candidates based on the indices of action potentials results in spikes and spectrograms of different sizes. This problem is resolved by truncating the spectrogram of one or both action potentials. The duration from the beginning of the spectrogram to the column representing the peak of the action potential, as well as the duration from the peak column to the end, is calculated. The spectrogram(s) are then cropped accordingly, so that both have the shorter duration of either spectrogram before and after the peak. This ensures that the spectrograms being analysed are of equal size, facilitating the calculation of correlation coefficients.

Several types of correlation analyses have been made on the sub-spikes of burst candidates from single-channel and multi-channel perspectives. We have considered the correlation as adequate to indicate the correctness due to the characteristics of bursts. Spikes of the same neuron will have similar shapes [2] and as bursts are the spiking activity of a single neuron in a small timeframe, it is to be expected that the correlation of intra-burst spikes to be higher than the correlation between intra-burst spikes and tonic spiking activity. Furthermore, the correlation coefficients were calculated for sub-spikes

originating from bursts on the same channel, as well as from bursts on different channels at a distance. As distant channels will record the activity of different neurons, it is to be expected that the bursts found on a channel by any burst detection method will have higher correlation coefficients than the bursts of different channels as they present the activity of different neurons. After calculating the coefficients, the distribution of values is analysed using histograms that have been normalized into Probability Density Functions (PDF) through division by their sum. The PDF was chosen as it provides comparable distributions.

Performance analysis of burst detection methods

In order to compare the burst detection algorithms, two metrics were used: the true positive fraction (the number of action potentials correctly identified as part of bursts and labelled as bursts, divided by the total number of action potentials labelled as bursts) and the false positive fraction (the number of non-burst action potentials incorrectly identified as bursts, divided by the total number of action potentials in bursts). These metrics provide a quantitative measure of the algorithms' performance in detecting bursts and avoiding false positives as they assess the performance of the algorithms in detecting bursts (true positives) and incorrectly identifying non-bursts as bursts (false positives) on benchmark synthetic burst data [168].

Each method was analysed with the parameterizations suggested in the corresponding articles and the metrics were calculated for each type of data. In Figure 3.44, we present the true positives for each data type as a panel, and the x-axis indicates the method, the same layout is used for false positives presented in Figure 3.45.

Upon the examination of Figure 3.45, it can be observed that the methods do not, generally, misidentify non-bursting activity as bursts. For high-frequency bursts, long bursts, and regular bursts, the fraction of elements misidentified as bursts is 0. This suggests that the parameterization of the data for these algorithms is very good, as the elements identified as bursts are clearly bursts due to the lack of false positives.

The only case in which misidentification is present is for noisy bursts, which are the hardest to identify. The Rank Surprise, Poisson Surprise, and Cumulative Moving Average algorithms rarely misidentify a non-burst as a burst. ISI Rank Threshold falls in the middle, identifying more errors, while ISIn and MaxInterval misidentify bursts more frequently than the other methods. Although ISIn and MaxInterval have more false positive identifications, the overall error rate, as seen in the figure, is not very high, averaging around 22%.

Although Rank Surprise, Poisson Surprise, and Cumulative Moving Average algorithms have a very low rate of false positives, as seen in Figure 3.45, they also have a very low rate of true positives as can be seen in Figure 3.44. The ISI Rank Threshold algorithm performs slightly better in burst detection than the previous two. On the other hand, ISIn and MaxInterval, despite having an approximate 20% false positives rate, but only in the case of noisy bursts, also have the highest rates of correctly identifying bursts.

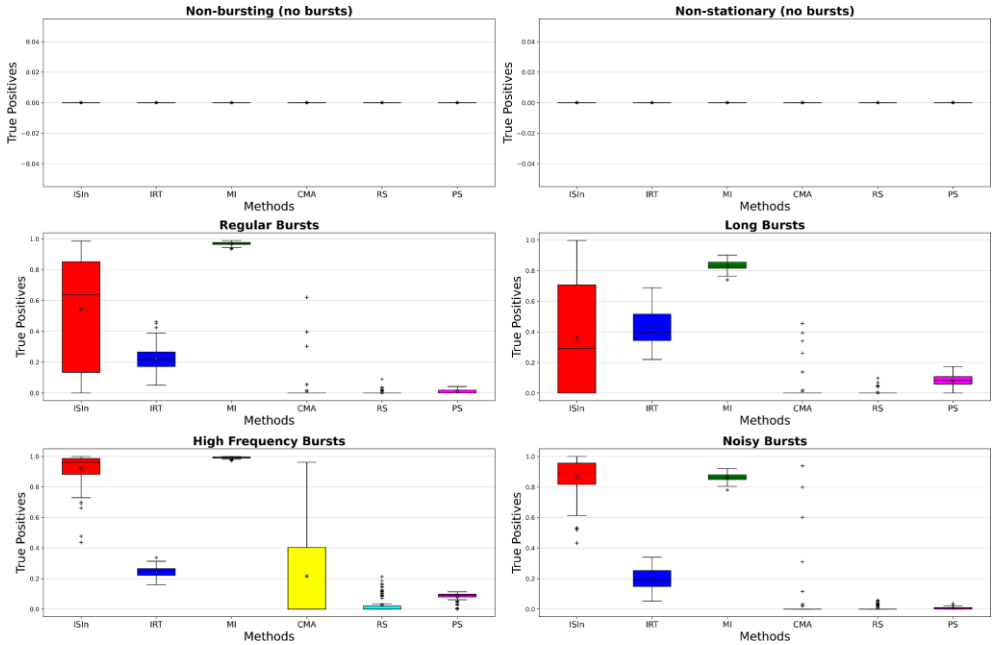

Figure 3.44 - True positive percentage (indicated on the y-axis labels), each subplot shows the evaluation on the percentage of true positives (ranging from 0 to 1) found in a specified data type (indicated by the subtitle) by each method (indicated on x-axis labels) on the benchmark synthetic burst data.

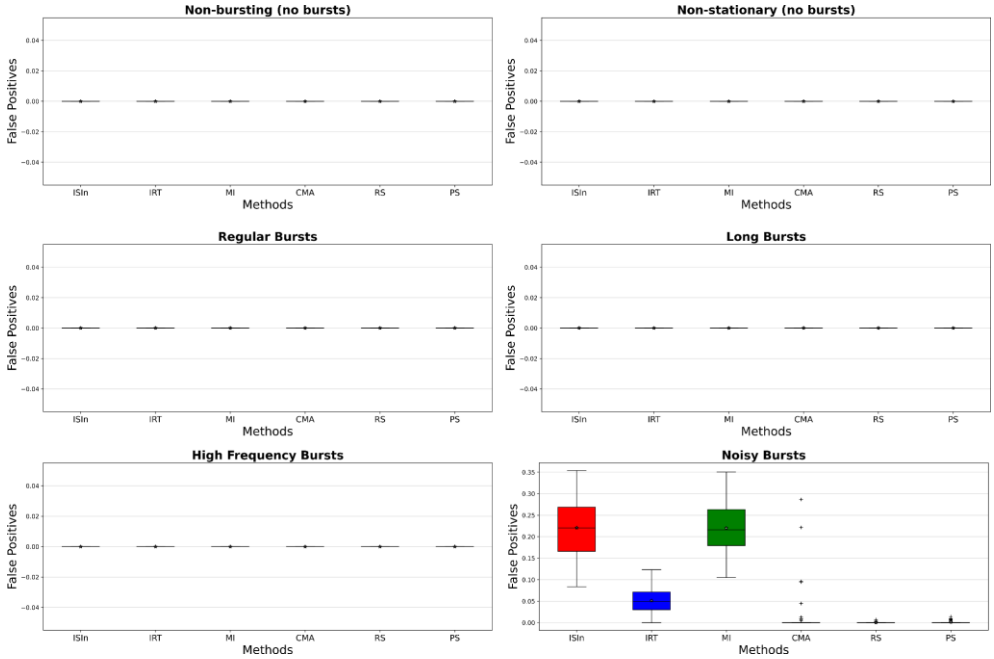

Figure 3.45 - False positive percentage (indicated on the y-axis labels), each subplot shows the evaluation on the percentage of false positives (ranging from 0 to 1) found in a specified data type (indicated by the subtitle) by each method (indicated on x-axis labels) on the benchmark synthetic burst data.

Analysing the perspective of correctly identified burst types by these two algorithms, for high-frequency bursts, MaxInterval has an average identification rate of almost 100%, compared to ISIn which has an identification rate of approximately 96%. From the perspective of long bursts, as seen in Figure 3.44, MaxInterval performs much better, correctly identifying approximately 82%, compared to ISIn which only identifies about 30%. This significant difference can also be observed for regular bursts, with MaxInterval

providing an identification rate of approximately 97%, compared to ISIn which has around 62%. However, in the context of noisy bursts, ISIn performs slightly better than MaxInterval, with the former having an approximate 90% identification rate, while the latter has approximately 85%.

Correlation analysis

The MI burst detection method offers promising results for the simulated data. This subsection analyses its performance on real electrophysiological data, that has no ground truth, through the correlation coefficient. We have analysed the performance of the MI method using the parameterization suggested by the authors of the method and also using empirically chosen parameters. We compare the proposed method [167] with the two aforementioned options for the MI method. Figure 3.46 displays a burst extracted using the MI method with the suggested parameterization.

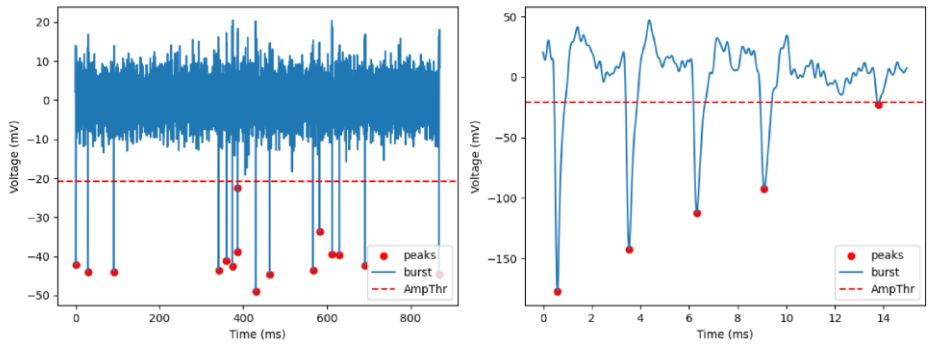

Figure 3.46 - Burst candidate extracted from the M029-C23 real dataset by the MI [148] burst detection method (left) and the proposed method (right), the blue line represents the signal, the red dots show the peaks of spikes, and the dashed red line represents the amplitude threshold used in the detection of spikes.

An initial hypowork was that the correlations would be higher for action potentials originating from bursts on the same channel compared to those from different channels at a distance. This would aid in the problem of burst detection, as the identification of the source of action potentials, whether from a burst or an individual discharge, could be based on a simple correlation threshold. If the action potentials originated from different neurons, they would not form a burst. This analysis is shown in Figure 3.47 for the both the proposed method (left) and the MI method with the suggested parameterization (right). The correlations resulted from the sub-spikes of bursts identified by the proposed method on a single channel are much more concentrated towards values of 1 than those of the MI method, indicating that the sub-spikes identified are more similar. Furthermore, there is a visible disparity between the distributions of the same channel versus different channels for the two methods, indicating that the bursts detected by the proposed method are more distinctive across channels. As mentioned previously, it is to be expected that the sub-spikes of bursts from different channels to have lower correlation values as they originate from different neurons. In Figure 3.48, we show the comparison between the correlation PDFs for the same channel across the three methods on the left and for distant channels on the right. The highest correlation values for the burst sub-spikes are given by the proposed method, however the suggested parameterization of MI results in the most skewed distribution towards

lower values for the burst sub-spikes of different channels. For this analysis, the correlations provided by the chosen parameterization of MI results in a distribution with its peak between the two methods.

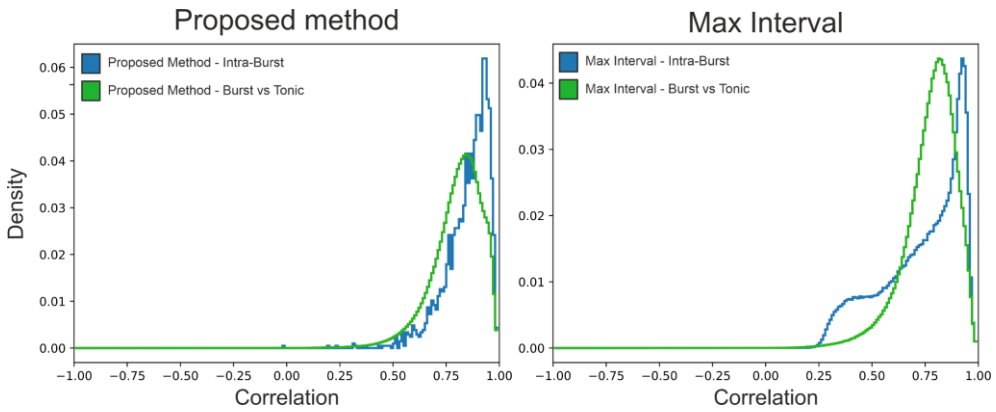

Figure 3.47 - The left panel shows a comparison between the correlation PDF between intra-burst sub-spikes of a single channel versus the burst sub-spikes and tonic spiking activity for the proposed method, the right panel shows the same analysis for the MI [148] method with the suggested parametrization.

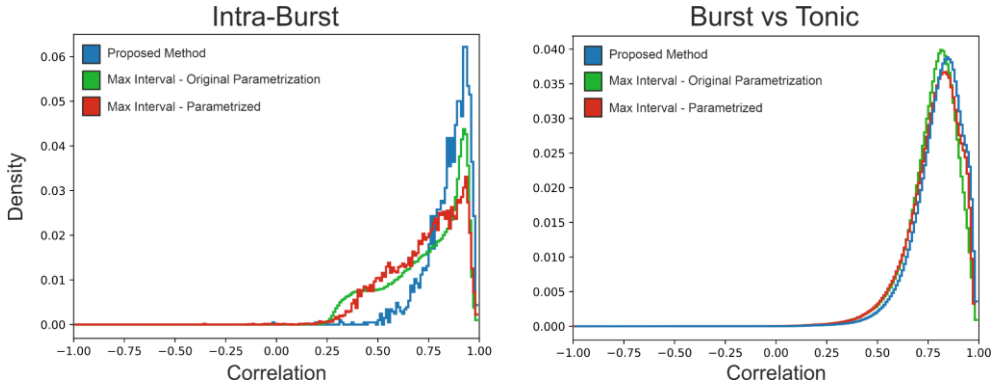

Figure 3.48 - The left panel shows the correlation PDF of intra-burst sub-spikes of a single channel for each of the methods, the right panel shows the correlation PDF between burst sub-spikes and the tonic spiking activity of a single channel.

Another hypowork was that the correlations between intra-burst sub-spike would be higher than the those between burst sub-spikes and the tonic activity of the same channel because burst sub-spikes will have more similar waveform shapes than tonic activity as the sub-spikes originate from the same neuron. This analysis is shown in Figure 3.49, for the both the proposed method (left) and the MI method with the suggested parameterization (right). The correlations resulted from the intra-burst sub-spikes identified by the proposed method are much more concentrated towards values of 1 than those of the MI method, indicating that the sub-spikes identified by the proposed method are more similar. Our hypowork was partially confirmed as the correlation values of intra-burst sub-spikes are indeed overall higher than those between burst sub-spikes and tonic activity, however not by a significant amount regardeless of the burst detection method. In Figure 3.50, we compare the correlation PDFs for the intra-burst sub-spikes across the three methods on the left and for bursting against tonic activity on the right. In spite of the noticeable difference between the distributions on intra-burst

sub-spikes of the three methods, the comparison across methods between burst sub-spikes and tonic activity shows no significant difference across the three methods.

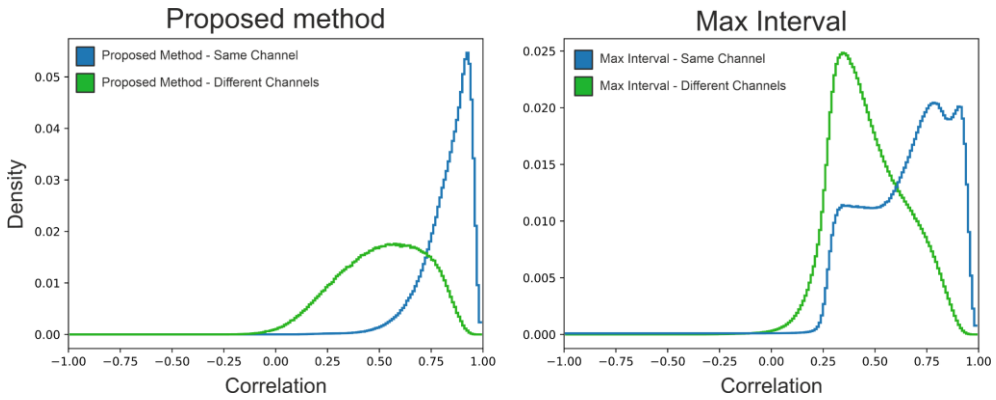

Figure 3.49 - The left panel shows a comparison between the correlation PDF between burst sub-spikes of a single channel versus the burst sub-spikes of distant channels for the proposed method, the right panel shows the same analysis for the MI [148] method with the suggested parametrization.

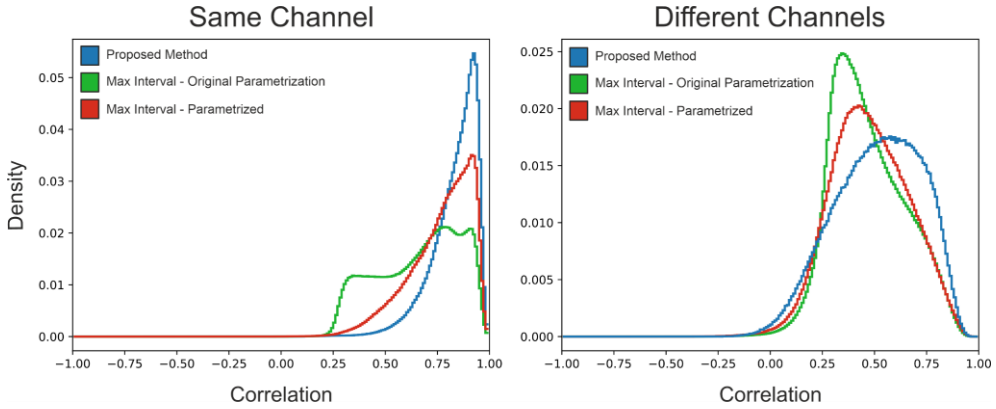

Figure 3.50 - The left panel shows the correlation PDF of all burst sub-spikes against each other of a single channel for each of the methods, the right panel shows the correlation PDF between burst sub-spikes of two different distant channels.

Upon these explorations, a clear differentiation point in the distribution peak could not be found for each case and for some cases the values are distributed quite similarly and close to the value of 1. Therefore, it is not possible to find a simple threshold that would produce separation for every situation. Consequently, a more suitable similarity metric on correlation is available for exploration and other correlation functions could be investigated. Nevertheless, the comparative analysis of correlation values can offer insights into the performance of detection methods for a variety of conditions.

3.2.4. Conclusions

Alongside a novel burst detection method, we propose correlation as a viable analysis tool for the comparison of performance of burst detection methods and for the validation of correctness of these methods. Regarding the ability of burst detection algorithms to identify bursts, although it is desirable to have as few incorrectly identified

bursts as possible, we also need a large number of correctly identified bursts. Analysing the overall results, we can say that the MaxInterval method offers the highest performance for all types of bursts on the analysed synthetic data compared with all other existing burst detection methods.

Our comparative analysis is based upon the evaluation of correlation values across a variety of conditions between the proposed method for burst detection and the MaxInterval method that obtained the highest results on synthetic data. In this study, we have analysed whether correlation can be used to differentiate between the bursting activity of different channels by comparing the results to the values obtained for the bursting activity of the same channel across these burst detection methods. We have also analysed whether the bursting activity detected by these methods can be separated from tonic activity through an evaluation of correlation values.

Our exploration of correlation as a potential measure for differentiation revealed certain limitations. Despite initial expectations, the correlation values obtained from various cases were remarkably similar, making it challenging to establish a clear threshold for separation. Consequently, correlation alone proved inadequate for distinguishing between burst and non-burst cases. It would be unfair to exclude from contemplation the possibility that correlation might be an adequate tool and it is the burst detection methods that hinder its efficiency. However, this is a question that can only be answered as a deeper understanding of the bursting phenomenon is achieved in the domain.

4. Methods for brain activity characterisation

The identification of brain activity is only the first step in understanding the brain. Once identified, this activity needs to be characterised. The most common and oldest method of brain recording is EEG which can now provide a high-volume data which requires efficient computer science methods to be analysed. One option is to encode the high-dimensional space of EEG data into a simpler format through symbolic analysis. The first subchapter proposes an original method, a SOM-based symbolic analysis pipeline, for the encoding of EEG data and several visualization techniques that can offer insights into the brain activity that is happening during experiments through the identification of recurring patterns of activity.

The second subchapter takes a slightly different avenue. Oscillatory activity has been found all throughout the brain, with each frequency band linked to various brain states and activities. To be able to correctly characterise the oscillatory activity, they must be first be precisely detected. We propose an original detection method for brain oscillations that is able to separate oscillatory activity from background noise with higher precision than other existing methods. Through the detection of these oscillations and their characterisation, by certain criteria as in location in time and frequency or power, links can between behaviours in tasks of experiments and these oscillations.

From a scientific perspective, the characterisation of EEG data, as it records the whole activity of the cortex by regions, can help us understand how different areas interact during an experiment, such as when the visual processing starts, when a decision has been made or when a conclusion has been reached. Moreover, anomalies in EEG data and oscillatory activity have been linked to various conditions such as schizophrenia, dementia, and many others.

4.1. Symbolic analysis

4.1.1. Introduction

The recent advancements in computer hardware and software have led to the recording of large datasets, making it challenging for traditional methods to efficiently analyse them and extract insights. One option is to summarise large datasets in a manner that results in a manageable format while preserving as much knowledge from the original dataset as possible [5]. Symbolic analysis is one of the many ways that data can be summarised into a more manageable format.

Through symbolic analysis, the task of knowledge discovery becomes easier to manage from a computational perspective and requires less execution time for the algorithms. Other various visualisation techniques applied as postprocessing operations allow for the interpretation of results. One of the many methods of symbolic analysis was developed for the analysis of EEG data, namely EEG microstates.

The main challenge of analysing EEG data comes from its high dimensionality introduced by using multiple electrodes for recording. Regardless of its placement on the scalp, an electrode records electrical signals from multiple brain regions due to the interference of signal sources, making it difficult to distinguish data from different parts of the brain. External factors such as eye blinks, body movements or electrical noise

introduce unwanted artifacts regarded as noise, hindering the analysis process. Since the electroencephalogram is a non- invasive technique, the electrical signal has to bypass several layers including the scalp and skull until it is picked up by the electrode. This results in the final signal being heavily attenuated and smeared.

The hypowork is that symbolic analysis can be applied to various types of neuronal data in order to transform the massive amounts of recorded data into a more manageable format for both the researcher and the algorithms. EEG recordings were analysed, and several techniques were used to visualise the transformed data and to detect patterns that were invisible to the naked eye from the raw or transformed data.

4.1.2. Data

The datasets found in this subchapter were recorded by the Transylvanian Institute of Neuroscience (details about the data acquisition procedure can be found in the Appendices):

- LFP data – recorded with a 32-channel probe at a sampling rate of 32kHz that was band-pass filtered in the 0.1-300Hz range and downsampled to 1kHz (additional band stop filter at 50, 100 and 150Hz to remove noise).
- EEG data – recorded with a high-density EEG cap of 128 electrodes at a sampling rate of 1024Hz that was band-pass filtered between 0.1-200Hz (additional band stop filter between 49.5 and 50.5 to remove noise).
    - EEG recordings were obtained during an experiment involving an object recognition task. Participants were presented with deformed two-dimensional lattices of points, where each represented an object; different visibility levels could be created through the adjustment of the level of deformation [169]. The subjects were given the freedom to visually explore the scene and their task was to identify the object and then press a specific key corresponding to their perception. They had three possible response options: "Nothing" if they did not recognize the object, "Certain" if they were able to name the object, and "Uncertain" if they identified an object but could not name it. The experiment consisted of seven blocks of trials (each composed of 30 trials with unique objects), with each block corresponding to a different visibility level where visibility levels ranged from 0 to 0.3 with increments of 0.05, thus creating a total of 210 trials. Due to the variable duration of trials resulted from the free exploration, several event timestamps were defined to identify important moments in the recorded neuronal activity. These timestamps included "stimulus on" to mark the onset of the stimulus, "key press" to indicate when the subject pressed a key, "message for verbal response" to note the moment when a verbal response was provided, and "stimulus off" for when the stimulus disappeared off the screen.

4.1.3. Methods

Proposed method

Our proposed pipeline [170] draws inspiration from the analysis methods for spike-trains introduced in paper [152]. A first question that emerges is whether such

analysis methods can be applied to EEG data resulting in relevant findings. Three-dimensional SOMs are proposed as a way to encode the EEG data into symbols.

After the training of the SOM, the distance map can be computed. In Figure 4.1 , we illustrate three types of 3D plotting techniques for the visualization of the distance map. The left side of Figure 4.1 represents a scatter plot, where each point represents a value of the distance map placed at its corresponding coordinates in the Cartesian system. In the middle of Figure 4.1, we show a voxel plot in which the values of the distance map are 1×1×1 cubic cells forming the three-dimensional volume. The right side of Figure 4.1 illustrates a volume slice plot, which displays the distance map as an interactive cube, whose x, y and z planes can be moved on their corresponding axes to showcase the values of the map at any coordinates.

Every sample from the data set has a corresponding best- matching unit in the map. If the BMU is close to its adjacent neurons, then the input sample is very similar to its neighbours, forming a cluster with related features. However, if the BMU is distant, having values close to 1 in the distance map, this means that it becomes differentiated from its neighbours, having distinct characteristics. In the same context, yellow regions on the plots usually act as cluster separators, while the input samples are mapped on the purple areas.

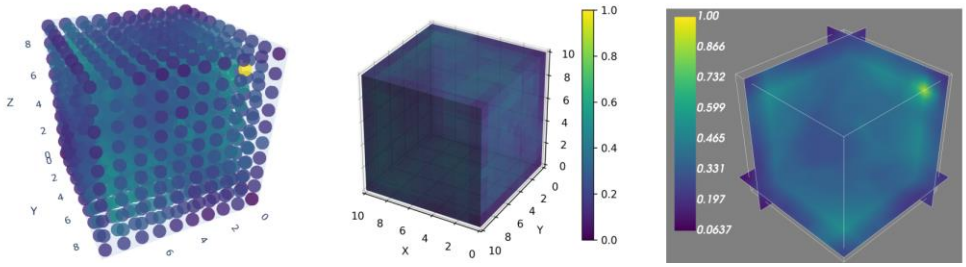

Figure 4.1 - Several ways of visualisation of the distance map: scatter plot (left), voxel plot (middle), and volume slice plot (right).

Clustering of the distance map

The subsequent phase in the proposed pipeline after visualizing the results of the SOM training is clustering the data set. The distance map plots have already given us an idea about how the possible clusters are distributed in the 3D SOM space, but we cannot indicate visually exactly how many groups the data can be divided in. Finding clusters in the SOM lattice implies identifying samples which resemble each other in some respects.

The method we propose [170] for cluster assignment iterates through the samples from the data set and assigns them cluster labels one by one. Initially, we set the number of found clusters to 0. Then, we take one sample from the input data set, we find its best matching unit from the trained SOM, and we assign it to a new cluster. At this point of the algorithm, the assumption is that the current sample will form a cluster on its own. Meanwhile, we also keep track of the samples and their best matching units that have already received a cluster number. The next step in the algorithm is to iterate through all the already clustered BMUs and compute the normalized distance between them and the current sample. If the minimum value from all the distances is also smaller than a predefined threshold, it means that the current sample is close enough to the cluster of that BMU to be considered part of it. This indicates that the initial assumption was incorrect, and we update the cluster label of the sample accordingly. If, however, we don't

find any distance within the threshold, the initial cluster assigned to the sample remains unchanged. If the cluster has indeed changed since the initial assignment, then the number of total clusters that have been found is incremented. These steps are repeated for all input vectors from the data set.

After having labelled every sample, we assign to each individual cluster a unique combination of a marker and a colour that will represent it for visualization purposes. Regarding the threshold used in the clustering method, it should be a number between 0 and 1, as all the distances are normalized in that range. Its value should be chosen depending on the data set by employing a trial-and-error principle. The resulting number of clusters that will be obtained using this method is influenced by the threshold as well as the values of the parameters used in training the self-organizing map.

For visualizing the clustered data, we have implemented a scatter plot similar to the one used for illustrating the distance map, replacing the points with specific markers. The plot is made by taking each sample, finding its best matching unit and placing the marker corresponding to the cluster of that sample on the BMU's coordinates in the 3D lattice. In Figure 4.2, the plot obtained after clustering the data set using a threshold of 0.21 is displayed. The number of clusters is 7.

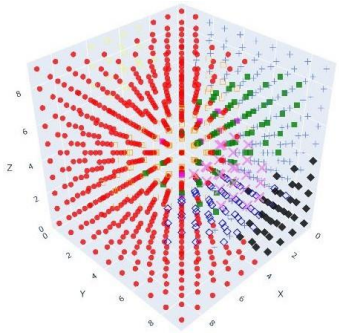

Figure 4.2 - Scatter plot with the clusters obtained from the clustering of the EEG data.

Colour sequences

Our proposed pipeline [170] uses the technique of colour sequences to build a visual representation of the brain activity acquired during EEG recording. Each trial will have its own colour sequence defined by the colours of the samples in the trial. The pipeline defines two approaches for the colour assignment of each sample.

In the first approach, the set of colours is obtained using the corresponding colours of the best matching units from the three-dimensional self-organised map of the EEG data set. The process of colour labelling of a sample implies finding its model vector in the self-organised map and then mapping its spatial coordinates to corresponding RGB values in the RGB colour subspace defined by the size of the map, as shown in Figure 4.3. The size of the self-organising map determines the range of RGB values that can be assigned to each sample.

Figure 4.3 - The process of colour assignment of a sample.

A second approach is to use the previously proposed clustering approach [170] prior to constructing colour sequences, where samples are grouped into distinct clusters based on their similarity. Once the clustering is completed, each sample is assigned the colour corresponding to the cluster it belongs to. This ensures that samples within the same cluster share the same colour in the resulting colour sequences.

Since the colour sequences are positioned in an adjacent and time-aligned manner, one below the other, an essential thing to consider is the grouping approach of the sequences. Different regularities and meaningful patterns can be detected according to the criteria by which the trials are organised into subsets. Based on the particularities of the EEG data set, we decided to group the trials considering three different key criteria: the response of the subject (Nothing/Certain/Uncertain), the stimulus and the visibility level of the stimulus. For example, 'Nothing', 'Certain' and 'Uncertain' are referred to as sub- groups within the grouping criteria 'group by response'.

As stated in the beginning, the colour sequences are visual representations of the EGG trials. Consequently, they encapsulate the structural properties of trials, including their length. Due to the exploratory manner of the object recognition task, the subject is allowed to respond when he has reached a decision, therefore the length of each trial will be different. Moreover, this temporal variation of events leads to the misalignment of colour sequences which implies that they cannot properly capture the regularities of a pattern possibly triggered by an event. The solution to this issue involved trimming the colour sequences representation between the timestamp corresponding to 'stimulus on' event and the timestamp corresponding to 'stimulus off' and then performing an alignment operation using these two events, identified as left alignment and right alignment. Figure 4.4 illustrates the left alignment of colour sequences of trials grouped by the response of the subject. In this figure for the "Certain" response, a pattern can be observed that appears approximately 150ms after the 'stimulus on' event. In contrast, Figure 4.5 shows the second option of visualising colour sequences created using clusters from the input dataset. However, this visualisation method fails to retain the patterns that were previously identified using the alternative.

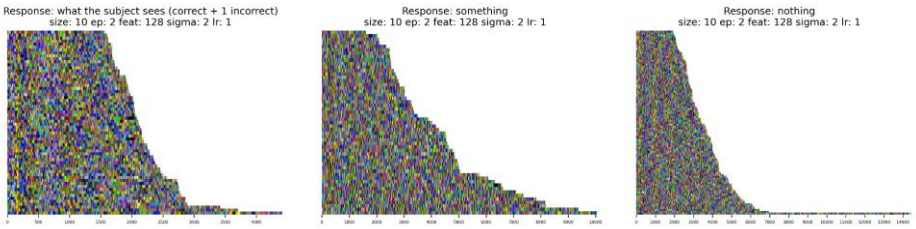

Figure 4.4 - Left-alignment of colour sequences of the "Certain" (left), "Uncertain" (middle) and "Nothing" (right) responses of subjects.

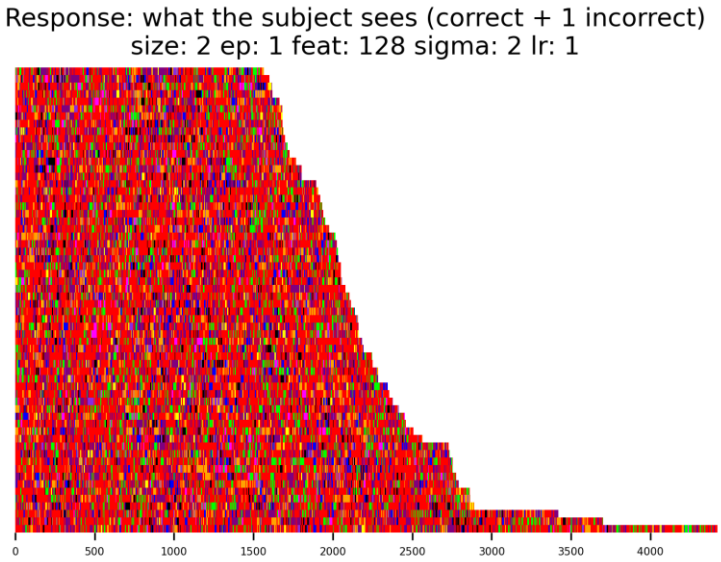

Figure 4.5 - Left-alignment of colour sequences of the "Certain" response of subjects.

The subsequent step, in the process of accomplishing a thorough analysis of the information depicted by the colour sequences, was to inspect the representation on a time-limited window given by the shortest duration among all trials in the group. After establishing a fixed length interval, the pipeline applies two rules for building the sequences: the former implies taking a number of samples equal to the length of the interval from the beginning of each sequence, while the latter comprises a visualization of the sequences on a same length window but aligned at the end. The result of these two approaches is illustrated by Figure 4.6 left and right, respectively. As it can be seen in both figures, the analysis on a time-limited window emphasises the presence of a pattern in the brain activity triggered by 'stimulus on' respectively by 'stimulus off'.

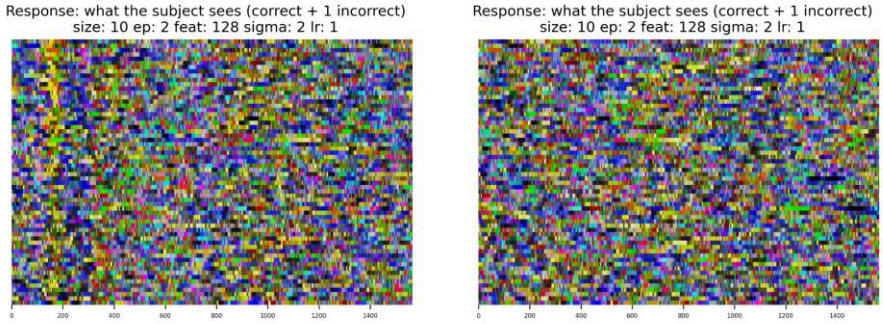

Figure 4.6 - Windowed colour sequences of the "Certain" response of subjects with left-alignment (on the left) and with right-alignment (on the right).

Pattern Specificity Index

The Pattern Specificity Index (PSI) serves as a basis for various analysis methods in the proposed pipeline. By assigning numerical values to the PSIs of patterns (colours) within the RGB subspace defined by the size of the self-organising map, thresholding techniques can be applied. This allows us to generate colour sequence plots that include only the patterns that surpass the threshold, thereby highlighting the most relevant and specific patterns in the data.

The PSI equation takes into account the number of occurrences of a pattern *p* in a specific stimulus *s*, as well as the total number of occurrences of that pattern across all stimuli in the experiment. However, the pipeline utilises alternative methods of grouping the data that go beyond solely relying on the stimulus of each trial. Instead, it considers conditions specific to the structure of the EEG dataset. When computing the PSI for a pattern belonging to a subgroup defined by the subject's response (Nothing/Certain/Uncertain), it is important to consider its occurrences not only within that subgroup but also in the other subgroups. This is necessary due to the structural characteristics of the EEG trials. However, there are other factors that need to be taken into account to ensure correct results, as the formula alone may not fully capture the complexities of the EEG trial structure.

The PSI computation becomes imbalanced when using the original equation to handle groups of colour sequences that have trials of varying lengths and an unequal number of trials. Specifically, by grouping the sequences based on the subject's response, the resulting groups have different trial counts. Additionally, the outcome is further affected by the observation that trials associated with the 'Nothing' response are considerably longer, containing a greater number of patterns, compared to subgroups of 'Certain' and 'Uncertain' responses.

Our pipeline proposes a new technique called "Weighted PSI" [170] to address the challenge of imbalanced distributions of structural properties in colour sequence groups. Its purpose is to weigh the contribution of PSI values from each subgroup within a condition with the intention of ensuring that each PSI has an equal impact on the final value. The final PSI formula for a pattern within a subgroup becomes a derivation of the original formula as it implies a multiplication of the initial PSI value with a weight $W_{subgroup}$, which corresponds to the specific subgroup it belongs to.

$$WeightedPSI_{p,s} = W_{subgroup} \cdot \frac{count(p \mid subgroup{=}s)}{\sum_j count(p \mid subgroup{=}j)} \tag{39}$$

Finding PSI weights of a group of trials can be viewed as a linear system of equations where the weights of all subgroups must add up to 1 and all multiplications of a subgroup weight and the probability of choosing a sample from that subgroup must be equal. Thus, the PSI remains in the 0 to 1 range.

Meaningful patterns, those appearing the most during the colour sequences of each group, are identified by thresholding the PSI values of all patterns. We identify a pattern as a meaningful or significant if its corresponding pattern specificity index value satisfies the condition:

$$PSI_{p,group} - \mu > coeff * \sigma$$

where μ is defined as the mean of all PSI values of patterns in a given grouping criterion, and σ expresses the standard deviation of the values, while the term coefficient (*coeff*) determines the degree of restrictiveness that determines the significance or relevance of a pattern within a particular group. This coefficient is closely linked to the size of the self-organised map. The patterns shown in Figure 4.7 were obtained by setting *coeff* equal to 3.

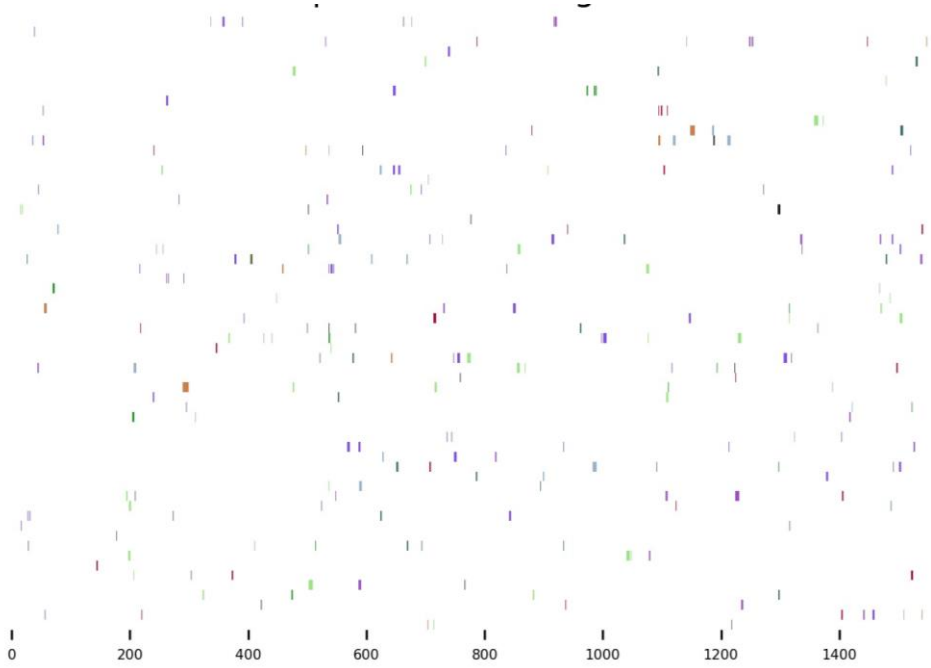

Figure 4.7 - Example of PSI with left-alignment for subject's response: "Certain", highlighting the most meaningful patterns.

Pattern Triggered Average

The computation of PTA starts from the PSI that has been determined in the previous step from the pipeline. We are interested in visualizing how the signal looks like only for those patterns that are specific for a group of trials. We carried out the PTA computation for every meaningful subgroup of trials (e.g. 'Nothing') that we have defined when constructing the colour sequences for a grouping criterion. For every pattern identified by PSI in such a group, the outcome of this process will be a figure containing a subplot for every trial in the subgroup. One such subplot will showcase the average of the EEG signal on a specific channel, offering a glimpse into how the signal behaves when that pattern occurs in the trial. If the pattern doesn't appear for all trials in the subgroup, those subplots are left empty. The computation of PTA starts with saving the timestamps at which each meaningful pattern appears within a trial. At the end of this step, each trial will be represented by a dictionary, having as keys the patterns and as values lists of corresponding positions at which the pattern appeared. This process is repeated for each subgroup generated by the grouping criterion used in the construction of the colour sequences.

Subsequently, the PTA is computed for every vector for each pattern in a trial. The PTA vector is obtained by averaging the signal values for each occurrence of the pattern in the trial. The values considered in the computation of the PTA vector are the values of the signal in a time interval centred at the point of occurrence of the pattern. The limits of the interval are computed by defining a fixed portion of time, referred to as window. The default window is of 100ms. For each pattern, we generate an image displaying the shape of the signal in every trial from the subgroup. All signal values considered in this computation correspond to a chosen channel. Besides this way of PTA visualization, we have also generated an average signal that encapsulates the overall behaviour of all the signals in a subgroup for a specific pattern.

On the right side of Figure 4.8, a pattern is chosen and a subset of its appearances in trials are shown (empty plots indicate that the pattern does not appear in that trial). While, on the right side of Figure 4.8, the average PTA for that pattern is shown. The shape of PTAs of high specificity patterns (found through PSI) in the occipital part of the brain feature a negative deflection, indicating the presence of synchronized activity of a large number of neurons for the visual processing of the stimuli. Due to the nature of EEG data, this method is rendered potent by its ability to capture such effects.

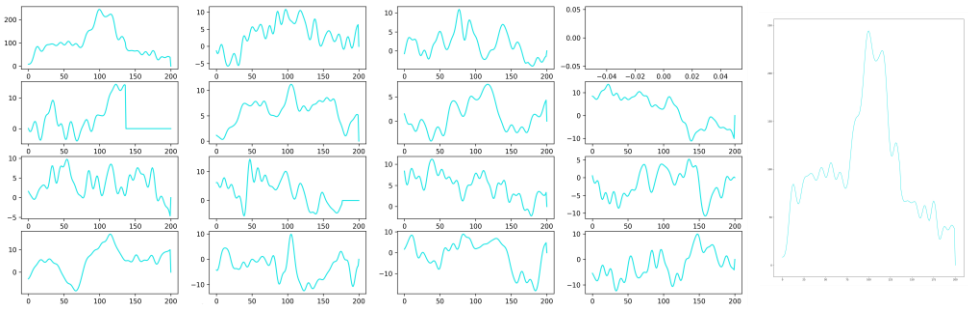

Figure 4.8 - PTA for a subset of trial in the subgroup "Certain", pattern (0.1, 0.9, 0.9) on channel D23 (left) and the Average PTA for subgroup 'Certain', pattern: (0.1, 0.9, 0.9) on channel D23 (right).

Peri-Stimulus Time Histogram

The pipeline generates the PSTH of each meaningful pattern discovered after PSI computation performed for each group of trials. The PSTH computation process is carried out during the reconstruction of colour sequences, which are time aligned on the stimulus onset, and it involves counting the occurrences of each colour pattern at a particular timestamp across all trials in a specific group. As a result, the histogram will provide information on the distribution of the meaningful pattern across each group of trials.

Considering the length of trials, the default bin size is 50ms and each bin contains the sum of occurrences of a pattern appearing over a time period of 50 timestamps. PSTH is a type of analysis that can only be performed on a time-limited window as it presents the distribution of a meaningful pattern across all trials in a subgroup. The PSTH presented in Figure 4.9 prove that these patterns are indeed meaningful as they are recurrent in the brain activity recorded during the experiment.

.

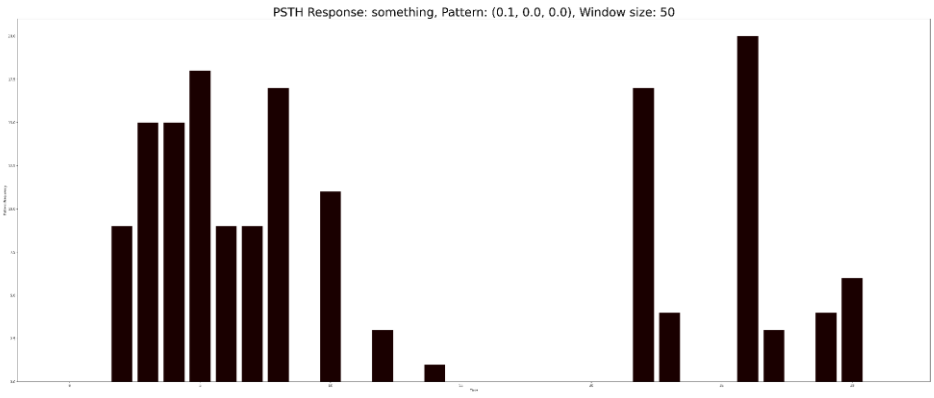

Figure 4.9 - PSTH of pattern (0.1, 0.0, 0.0) for subject's response: "Uncertain".

Findings

The proposed approach [170] is capable of identifying the onset of visual processing that ensues when the subject is presented with visual stimuli, as well as the disappearance of the stimuli. The ability to identify these phenomena, clearly associated with visual processing through their timings, indicates that the proposed approach is suitable for the task.

Several patterns with a fascinating behaviour have been identified in trials in which the response of the subject has been uncertain or nothing. It seems that some of the patterns found appear at the presentation of stimuli followed by a period of inactivity after which they emerge again. The reappearance of the pattern after its suppression can be certainly motivated by ocular movements, such as saccades and fixations. Nevertheless, this behaviour indicates a dualistic processing system for hard-to-identify visual stimuli. Immediately after the presentation of the stimulus there is a first attempt at identification, during which the pattern is suppressed. The reemergence could indicate another attempt of the subject to identify the stimulus and the pattern reappears as the search begins through fixations and saccades on different parts of the screen.

4.1.4. Conclusions

The techniques described here were able to extract meaningful patterns from multiple types of neuronal data that otherwise would have gone unnoticed. The depth of anaesthesia can be extracted through microstates even on LFP data that is positively different from EEG data although it requires a higher number of microstates.

The analysis of spiking activity through the creation of states by means of K-Means or SOM result in patterns that are recognizably related to the events of stimulus appearance and disappearance through visualisation of colour sequences. This indicates that these methods are capable of encoding complex information without loss of it. Visualisation techniques such as PSTHs, Tuning Curves and Correlograms offer information about the neuronal activity from different perspectives such as activity of a neuron throughout a trial or its attunement to a certain type of stimulus.

The same visualisation techniques can be applied to the case of EEG data as well with informative outcomes indicating that symbolic analysis is a powerful tool for the processing of such data into patterns that are meaningful from the perspective of neuroscience.

4.2. Detection of oscillation packets

4.2.1. Introduction

Neural computations and information transmission in the brain are accompanied by oscillations [171] embedded in rich time-frequency landscapes [95,172]. Oscillations often appear as events of finite duration and finite frequency span, called oscillation bursts or packets, intermixed with sustained oscillations and transient broadband events [173]. For instance, bursts of gamma and beta oscillations have been found to modulate attention, memory encoding and retrieval, and transiently couple distant areas in the brain. Oscillations have been found virtually in all relevant frequency bands, and even Berger [174] in his historical account depicted alpha oscillations as transient events.

Given the frequent transient expression of neural oscillations, it is crucial to develop tools that can precisely detect them in brain signals, enabling the quantification of their expression and statistical properties. Another difficulty in detecting oscillation packets in brain signals is related to the variety of patterns that such packets can take in time-frequency representations (TFRs), or spectrograms [95]. Different oscillation packets can have different shapes and extents, and this diversity is often not fully captured by simple detection algorithms that rely on thresholding or other straightforward measures. Instead, more complex algorithms that take into account the specific shapes of oscillation packets can be used to improve detection performance and specificity. In this context, the use of superlet TFRs and more complex detection algorithms that can capture the structure of oscillation packets, including their contours and sub-peaks, can be instrumental in probing single-trial oscillation dynamics.

4.2.2. Data

The datasets presented here were recorded by the Transylvanian Institute of Neuroscience (details about the data acquisition procedure can be found in the Appendices). In the analyses that can be found in this subchapter several real datasets have been used, as follows:

- EEG data – recorded with a high-density EEG cap of 128 electrodes at a sampling rate of 1024Hz that was band-pass filtered between 0.1-200Hz (additional band stop filter between 49.5 and 50.5 to remove noise).
- LFP data – recorded with a 32-channel probe at a sampling rate of 32kHz that was band-pass filtered in the 0.1-300Hz range and downsampled to 1kHz (additional band stop filter at 50, 100 and 150Hz to remove noise).

4.2.3. Methods

Proposed method

Our newly developed method for brain oscillation detection [175] is based on a clustering algorithm for neural spike sorting that was published in Ardelean et al., 2019 [163], one of the earliest research works in this work. The base method itself is actually a density-based clustering algorithm that quantizes the points to be clustered and estimates the extent of the clusters in this discretized space. For our ends in peak extent finding, since the power spectrum is already a discretized image (time- and frequency bins), we only need the extent finding part of the algorithm, that can be summarised as follows: using a threshold, normally set to a percentile of the distribution of the spectral

coefficients, all local maxima (peaks) are detected above that threshold. This is to ensure that the background does not produce too many peaks that would slow the algorithm and corrupt its detection capabilities.

Being based on the Space Breakdown Method (SBM) clustering algorithm, the proposed method has been denominated as Time-Frequency Breakdown Method (TFBM). TFBM identifies potential peaks of oscillations by traversing the discretized space and searching for local maxima, which serve as cluster centre candidates. The second parameter of the algorithm imposes a lower bound on these candidates, so only maxima above the threshold are considered. After identifying the peaks, a modified BFS expansion is applied to each candidate. However, for clustering and segmentation purposes, a stopping criterion is necessary to ensure that the BFS only expands to the extent of the cluster or oscillation packet. The expansion from one point to another must satisfy certain conditions, including being unvisited, having a lower value than the current point, and a higher value than the computed dropoff. The dropoff is determined through a formula similar to the root mean square of the neighbouring points of the cluster centre candidate. The algorithm iteratively expands each cluster centre until it can no longer expand, or it encounters a conflict with another assigned cluster. If a conflict arises, the algorithm initiates a disambiguation process, which evaluates the statistics of the conflict point, each of the cluster candidates, and the distance between them. Based on this evaluation, the disambiguation process concludes that the conflict point belongs to the current cluster, the old cluster, the whole old cluster should be included in the new expanding cluster, or the new cluster should be a part of the old cluster.

Although no map needs to be created, as SBM does, the algorithm still relies on the local statistics. Disproportionate time-frequency scales can skew the distance calculation, so we have included a homogenous scaling step as the first step. This step scales the power values between [0, 100] and modifies the Euclidean distance formula by multiplying each term with a factor determined by the resolution in time or frequency of the data. This ensures that the highest possible distance between two points in time, frequency, and power is 100, regardless of the number of samples.

To limit elevated plateaus to one point, we only consider local maxima as valid if they have no neighbouring local maxima. Additionally, these maxima must surpass a threshold parameter, as in the original version. For example, in Figure 4.10, the algorithm identifies three peaks. However, we have now incorporated an automated approach for calculating the threshold. By analysing the cumulative distribution of the power spectra, we determine the threshold as the value below which 90% of the power values lie. This method removes only a small portion of the power range and eliminates low-power, low-prominence local maxima candidates that would slow down execution and affect performance. It's worth noting that this threshold only affects valid local maxima, and the algorithm can continue expanding through such points. Once the valid local maxima have been identified, they are sorted in descending order, with the highest peaks given priority for expansion.

In Figure 4.11A, all local maxima discovered in a spectrogram generated using SLT and actual electrophysiology data are depicted. In contrast, Figure 4.11B illustrates only the local maxima that surpass the threshold. Power values below the threshold have been set to zero to emphasise the minimal impact of this restriction. The upper part of Figure 4.11C represents the distribution of power values divided into 100 bins. It is apparent

that most of the values are situated in the first percentile. The lower part displays the upper half of the cumulative distribution of local maxima.

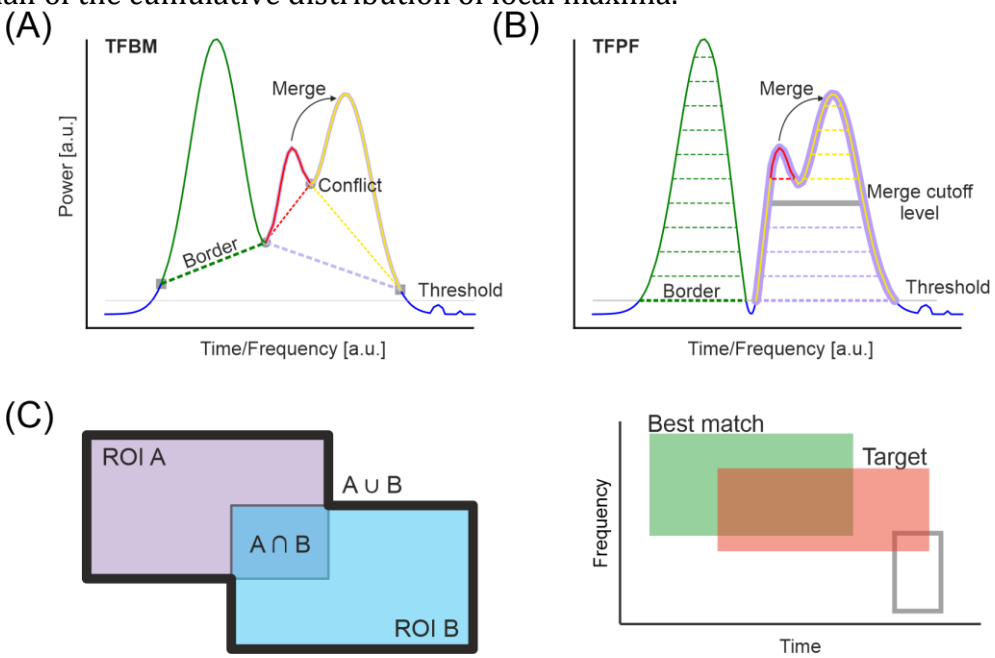

Figure 4.10 - The segmentation algorithms and the ROI matching metric. **(A)** The TFBM algorithm begins from the peaks of the TFR, and it expands until a conflict point (grey circle) or a point that satisfies the border condition (grey squares) is reached. Conflicts are resolved by assigning the conflicting points to one of the regions of interest (ROIs) or by merging the conflicting ROIs into a larger ROI, represented by dotted lines. Peaks below the threshold (thin grey line) are not considered as seed points for ROI expansion. **(B)** TFPF algorithm, which slices through the TFR from the highest power down to the threshold (grey line). ROIs, represented by dotted lines, expand as the slice level is lowered. When ROIs merge (thick grey line), the smaller peak is merged into the larger one. The algorithm stops at the threshold, and points below the threshold are not considered. **(C)** the matching metric between two ROIs is illustrated in the left pane. The right pane shows the best matching ROI (green) overlapping with the target ROI (red). The grey ROI has a lower match value to the target than the best matching ROI.

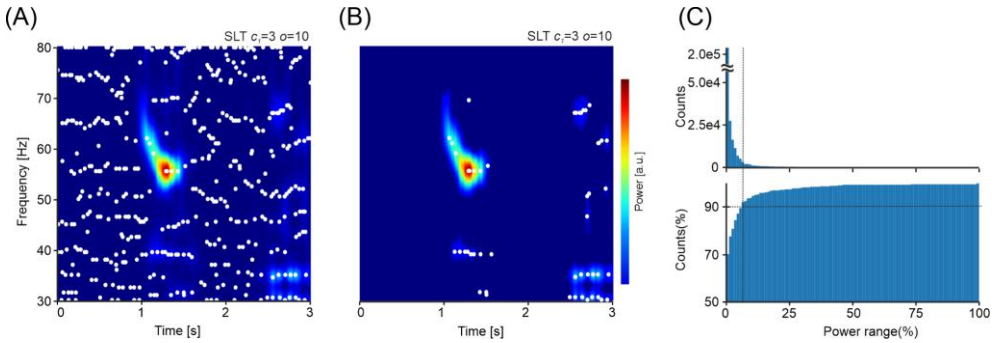

Figure 4.11 - Thresholding and rejection of low amplitude peaks of the TFR of a single-trial mouse LFP. **(A)** All local maxima of the TFR. **(B)** Only local maxima that are above the threshold, which is determined according to the method described in panel C. All power values below the threshold have been set to 0 to emphasise the low threshold while retaining the dynamic range of the TFR. **(C)** The power distribution in the TFR. The top panel shows the highly skewed distribution of the power values of all local maxima. The cumulative distribution shown in the bottom panel is used to set the threshold, covering 90% of all power values, which corresponds to less than 10% of the maximum power. The power values in the TFR have been scaled to the interval 0-100 for easier interpretation.

Similar to SBM, TFBM incorporates a disambiguation step. While in SBM, it was achieved during the expansion of clusters, in TFBM it is done as a subsequent step after

the expansion. Firstly, each peak can expand to its maximum limit before any modification is made to the layout, and each expansion can be stored as an isolated object for post-processing. This allows for hierarchical merging, where the merging process can be done at any required time while still being able to return to the unmerged segmentation. Secondly, the expansion requirements have been modified to recalculate the dropoff for each expansion point, delineating more precisely the oscillation packets. Figure 4.10 provides an example of the expansion process for each of the three local maxima found, with the extent of each peak delimited by colour. However, the disambiguation process is still necessary for the first two cases to redistribute the points. The BFS algorithm tends to spread greedily, and the redistribution of points by the disambiguation process ensures that every oscillation packet is spread to its maximum. The disambiguation step has also been separated from the expansion step as an intermediary between expansion and merging. This allows the expansion of packets to naturally stop, while points of conflict are recorded. Once all packets have been expanded, the points of conflict are reevaluated and distributed to the most appropriate packet. The most appropriate was chosen from the conflicting packets of said point that has the highest value of the division between its peak power value and the distance to the point of conflict. Points of conflict between the current cluster and any other are stored during the expansion process to be used later in the merging process.

The merging process iterates through all the packets in increasing order of their peak power value and their corresponding conflicting candidates. Two packets are merged if and only if the difference between their peak power values and the maximum power across the conflict points is below a threshold. Additionally, the packet with the higher power value of its peak assimilates the less prominent one. This threshold is exposed to the user as a parameter, called merge threshold. The merge threshold can be interpreted as the percentage (power values are normalized in the 0-100 interval) of difference allowed between conflicting peaks and their common maximum conflict point. In Figure 4.10, for example, modifying the merging threshold parameter would determine whether or not to merge the red and yellow peaks. As labels are removed during merging, a post-processing step can be performed to compact the labels and improve visualisation.

For each of the found packets TFBM stores the coordinates of the peak, maximum power, prominence as defined in the topological literature, coordinates of conflicting points, the contour of the packet, all the point coordinates that form the ROI, and the parent packet if merged. These characteristics completely define and expose the found packets for further analysis.

To summarize, TFBM is controlled by three parameters: i) the threshold which eliminates spurious noisy peaks in the TFR, ii) the aspect ratio which determines the resolution of the TFR for the calculation of the scaled distance, iii) the merge threshold that determines whether two packets will be merged into one or not. A summarised representation of the modifications brought to the SBM algorithm for the birth of the TFBM algorithm can be viewed in Figure 4.12.

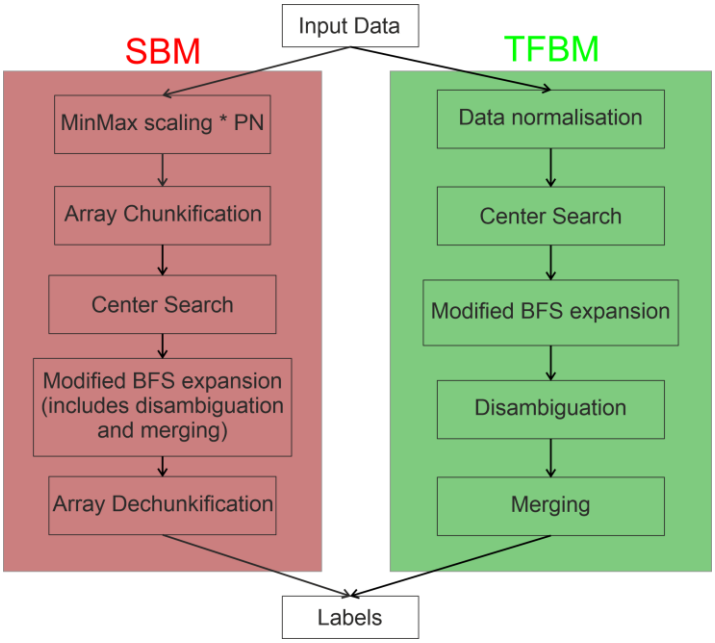

Figure 4.12 – A summarised comparation between the SBM and TFBM algorithms.

Performance analysis in oscillation detection

In order to evaluate and compare the performance of the algorithms, it is necessary to have an error function that can compare the regions of interest (ROIs) of the oscillation packets detected by the algorithm with the ground truth, which is the known ROI of the oscillation. To address this, we propose the match measure [175], denoted as m, which is calculated as the ratio between the intersection and union of the ROIs (as illustrated in Figure 4.10C on the left), where A and B are two ROIs. The formula can be written as follows:

$$m(A,B) = \frac{A \cap B}{A \cup B} \tag{40}$$

This equation offers a couple of advantages. It is inherently restricted to the [0,1] interval, where 1 represents perfect overlap, while no overlap results in 0. Using this formula, a matching error formula is proposed [175] and can be easily formulated as:

$$e(A,B) = 1 - m(A,B) \tag{41}$$

The comparison of the identified packets across the algorithms was performed first (see Figure 4.13). For the first batch of tests, the data was subjected to the superlet transform and all algorithms were applied to the superlet TFR. Although OEvents was originally used with a wavelet TFR, we wanted to eliminate the differences between TFRs in our evaluations. In principle, any TFR can be used in combination with all three algorithms, and all algorithms can benefit from the superior spectral representation of the superlets. In the first experiment, the aim was to assess the ability of the methods to differentiate between processes that are closely located in the frequency space. To achieve this, we created clusters of Gaussian atoms (20 cycles) that were equidistant in time but gradually closer in frequency (Figure 4.13A). The atoms, with an amplitude of 1, were added to random noise with double the amplitude. The superlet transform (with base cycles *c1 = 3* and order *o=10:10*) was used to analyse the signals, and all algorithms were applied to the same TFR. While TFBM and TFPF were able to identify all the atoms,

OEvents had difficulty detecting them due to its thresholding method that requires the power to be four times greater than the median for a peak to be detected [153]. The amplitude of the atoms and the noise made it difficult for the median to distinguish clear oscillations from the background. Atoms introduced a lot of power at low frequencies (20Hz), thus causing clear bursts to be missed as the median was set too high. Conversely, at higher frequencies where shorter atoms introduced less power, the normalisation process made even tiny levels of power seem significant, resulting in multiple atoms being grouped together under the same bounding box as a single event. This problem also led to many non-existing events being detected as oscillations in areas without atoms. We also performed the same analysis using OEvents, but the median used for normalisation was calculated across the entire TFR, and the same value was applied to all frequencies.

TFBM has a finer segmentation to separate packets amongst each other, but it can also cause over-segmentation. The Time-Frequency Peak Finder (TFPF), another proposed method, on the other hand, is a threshold-based method that works well when packets are clear. It performs similar over segmentations but with smaller satellite peaks. TFBM is more difficult to tune its threshold to detect faint packets and separate overlapping ones. Despite these challenges, neither method misses any packets, with some over-segmentations and spurious merges. These packets are faint parts of existing packets that can be eliminated by thresholding the representation at higher power values.

A single-trial local field potentials (LFP) recording from mouse visual cortex during a receptive field mapping trial is shown in Figure 4.13B. The adaptive SLT used was optimised to cover a wide frequency range (1-100Hz). The LFP reveals a wide array of frequencies with well-defined bursts of activity, such as the gamma burst at 55 Hz and the beta burst at 2.5s. Both have a complex shape with decreasing frequency and power modulations that appear as sub-peaks. TFBM and TFPF are able to delineate the most important packets of oscillations, with TFBM excelling at isolating both large and small packets. TFBM detects a larger number of structures that should be separated, while TFPF performs more aggressive merging. Both TFBM and TFPF present an unprecedented level of detail with respect to the complexity of the identified structure and the level of description. To identify such structures, it is important to employ the appropriate TFR. The comparison of algorithms on three different TFRs reveals that the crisp details in superlets allow the algorithms to overall separate better the structures.

The human EEG data shown in Figure 4.13C and Figure 4.13D presents a similar level of detail, in this case OEvents performs better than before in identifying prominent features of the TFR. Nevertheless, OEvents segmentation is greedier and merges larger areas in a coarser account of the structures, missing some low-power packets. TFBM is the algorithm that best delineates structures, although it merges some faint packets. TFBM identifies more structures than TFPF and OEvents, but only correctly delineates alpha activity and beta (20Hz) activity around 1.6s and 2.5s. TFBM segmentation is superior to TFPF but may be overly greedy in faint areas, while OEvents only captures a rough structure of the oscillations. TFPF finds contours nicely but suffers from a fixed threshold and poorly delineates smaller peaks.

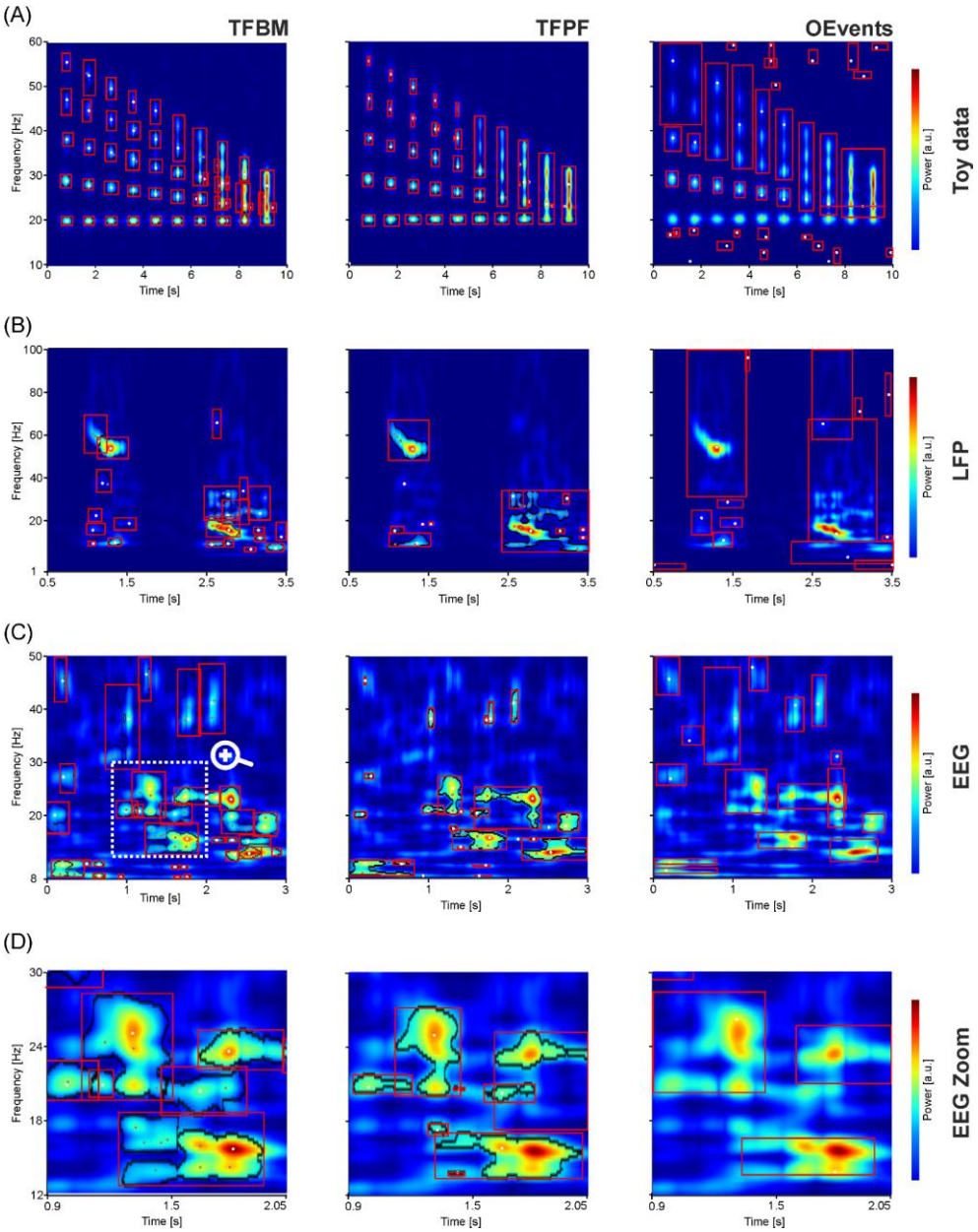

Figure 4.13 - Comparison of the segmentation provided by each algorithm. The algorithms are shown by columns: TFBM (left), TFPF (middle), and OEvents (right) on various single-trial TFRs computed with superlets on each row. The boundaries of the detected ROIs are depicted in black, and the bounding boxes are shown in red. White dots indicate the local maxima, while grey dots indicate the sub-peaks. **(A)** The TFR corresponds to a series of generated atoms in the gamma frequency range (20-55Hz) embedded in uniform noise. **(B)** The TFR corresponds to a wide frequency range (0-100Hz) from a single-trial LFP recording in the mouse visual cortex, stimulated with drifting gratings. **(C)** The TFR corresponds to a single-trial EEG with a rich spectrum covering the alpha, beta, and low gamma frequency bands. **(D)** Shows the segmentations of a zoomed-in area of the data from panel C.

As demonstrated in the examples shown in Figure 4.13B-D, neural oscillations in single trials are typically found amidst a complex backdrop of noise and background activity. In the following experiment, the effectiveness of the extraction of known oscillations that are hidden within a realistic background of these methods is evaluated. To accomplish this, 200 atoms were created with frequencies ranging between 35 and 95 Hz that lasted for 10 cycles. Each atom was randomly assigned a frequency, insertion

time, and a single trial from the available pool of 84 EEG trials to be inserted into. The creation of the atom also involves the choice of a level of SNR (0.1, 0.25, 0.5, 1, and 2) that modifies its amplitude. The insertion of atoms was made such that it ensured that the same configuration of atoms' frequency, time, and trial was repeated for each SNR level.

The measurement statistics for the detection performance of the three algorithms are presented in Figure 4.14A, based on the rectangular bounding boxes of the detected packets. An atom was considered detected if its known bounding box intersected with at least one of the packets detected by the algorithm. The error measurement against the ground truth atom was performed by considering the best matching packet. Three errors were calculated, namely the bounding box match error, the time match error, and the frequency match error. The time and frequency match errors represent the differences between the location of the atom and that of the highest peak in the bounding box.

In Figure 4.14A, it is evident that TFBM outperforms the other two algorithms in detecting atoms, as seen from the lower errors in bounding box match, time match, and frequency match. Subsequently, the time and frequency errors were assessed (shown in the second and third panels of Figure 4.14A, respectively). The calculation of time and frequency errors was limited to detected atoms only. Thus, undetected packets did not contribute to the statistics of time and frequency errors.

TFBM exhibits significantly lower errors in locating atoms compared to the other two algorithms (Figure 4.14A, top) for all SNRs. This superior performance, especially at low SNRs, is evident in TFBM having approximately half the misses of TFPF and OEvents at SNR 0.1 (Figure 4.14A, bottom). TFPF and OEvents collectively fail to detect about 7% of the atoms at this SNR. TFBM consistently demonstrates lower bounding box errors than TFPF and OEvents across all SNR levels, with the smallest proportion of missed packets. Beyond SNR = 1, neither TFBM nor TFPF miss any packets. At low SNRs (≤ 0.25), TFPF's performance is slightly inferior to OEvents, but at SNRs ≥ 1, TFPF exhibits lower bounding box errors and no misses.

The evaluation extends to time and frequency errors (Figure 4.14A, second and third panels). Time and frequency errors are computed only for detected atoms, meaning undetected atoms do not contribute to these error statistics. At SNR=0.1, TFPF and OEvents lose twice as many atoms as TFBM (Figure 4.14A, bottom). Although this favours the time/frequency distributions of TFPF and OEvents since the most challenging-to-detect packets are not included, TFBM still achieves smaller time and frequency errors. Overall, errors decrease with increasing SNR, and starting from SNR=1, the time and frequency errors are extremely low, with the three algorithms performing equally. However, OEvents continues to miss a small fraction of atoms, while TFPF and TFBM miss none. For SNR ≤ 0.5, TFPF encounters more difficulty in determining the correct timing and frequency of atoms compared to TFBM and OEvents. Significance levels were assessed using a paired t-test, assuming unequal variances, with Bonferroni correction for multiple comparisons (Figure 4.15).

The same evaluation procedure is applied to measurements shown in Figure 4.15B. However, instead of using rectangular bounding boxes, the assessments are conducted on the detected Regions of Interest (ROIs) (surface bounded by the detected packets' contour). Comparison is made only between TFPF and TFBM, as OEvents provides only bounding boxes. In terms of contour matching, TFBM significantly outperforms TFPF across all SNRs. At SNR = 0.1, TFBM misses only about 5% of packets, whereas TFPF fails to find approximately 9% of packets. From SNR 1 and above, when

atoms are more distinguishable from the background, TFBM and TFPF demonstrate nearly identical performance in terms of frequency and time errors, with no missed packets.

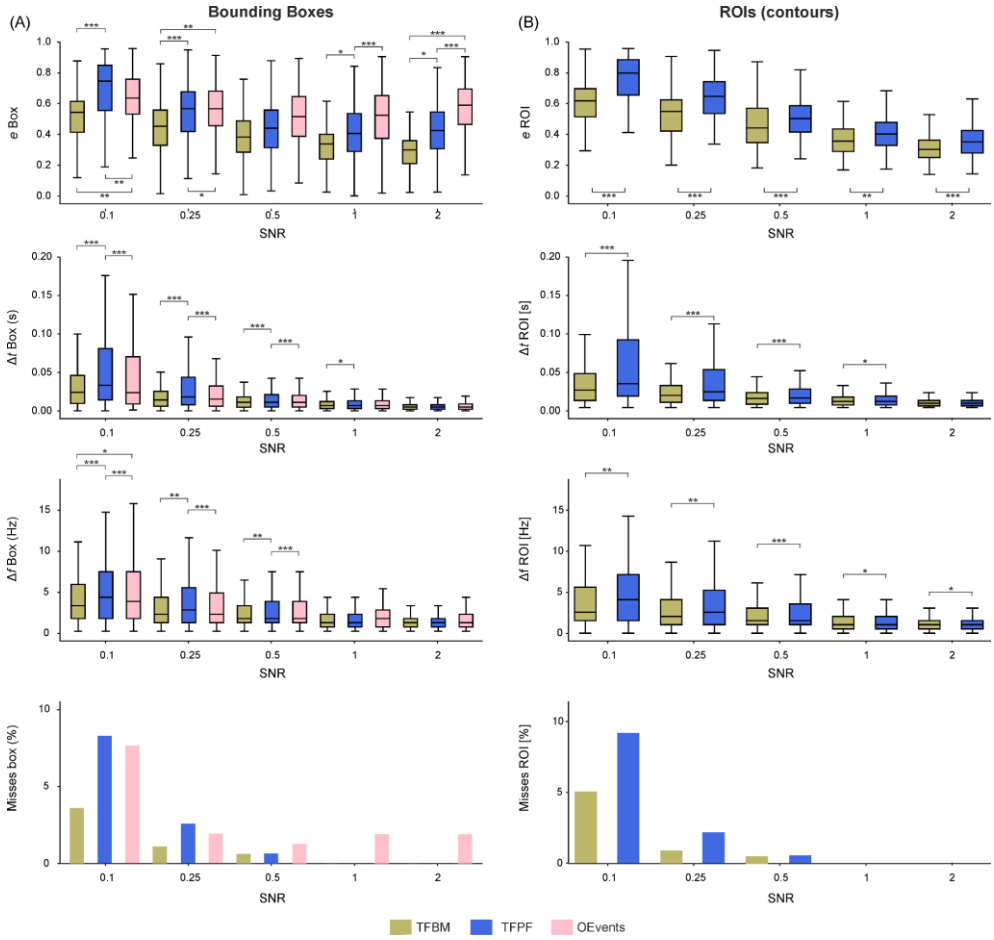

Figure 4.14 - A comparison of the segmentation of the TFBM, TFPF, and OEvents algorithms in terms of detection performance by SNR. Stars indicate significance levels, highlighting statistical differences. **(A)** The match between the ground truth (a generated atom introduced randomly in the data) and the detected packets is evaluated using the matching error of rectangular bounding boxes. The top panel displays the bounding-box detection errors, which improve with increasing SNR. The second panel shows the time errors, while the third panel presents the frequency errors. The bottom panel illustrates the percentage of missed atoms. **(B)** These panels have the same arrangement as those of panel A. Here, TFBM and TFPF are evaluated using fine-grained regions of interest (ROIs) represented by contours, instead of rectangular bounding boxes.

SLT was used as the preferred TFR, and atoms were embedded in a background of EEG in the previous evaluations. This raises two inquiries: How would the algorithms perform on other TFRs, and what is the impact of the background choice? To address these questions, we conducted the following additional evaluations.

In a second set of comparisons, the detection algorithm was applied to the same atoms embedded in EEG, but using CWT and STFT, the two most established TFRs. Figure 4.15A presents the comparative performance of TFBM, TFPF, and OEvents on the three TFRs generated by SLT, CWT, and STFT. As anticipated, across all representations, the algorithms exhibit improved performance at higher Signal-to-Noise Ratios (SNRs). TFBM generally has the smallest box errors and performs best in combination with SLT. It also

outperforms TFPF and OEvents for CWT. Overall, TFBM and TFPF benefit the most from the sharper superlet, while OEvents shows comparable performance on SLT and STFT, with more misses on CWT.

TFPF and TFBM encounter challenges on STFT at SNR ≤ 0.25, where they miss more atoms than OEvents. This issue might be attributed to the thresholding operation that eliminates small power values. The thresholds were set based on SLT representations, and STFT has different spectral characteristics. To test this hypowork, we lowered the power threshold, allowing more of the low part of the TFRs to contribute to packet detection. With a lower threshold (80%), the detection performances of TFBM and TFPF significantly improve, consistently surpassing OEvents. With this threshold favouring atom detection, TFBM only misses atoms on STFT and less than 2% at SNR = 0.1, while TFPF also minimally loses packets on SLT and CWT. These results suggest that SLT is the preferred TFR for TFBM and TFPF.

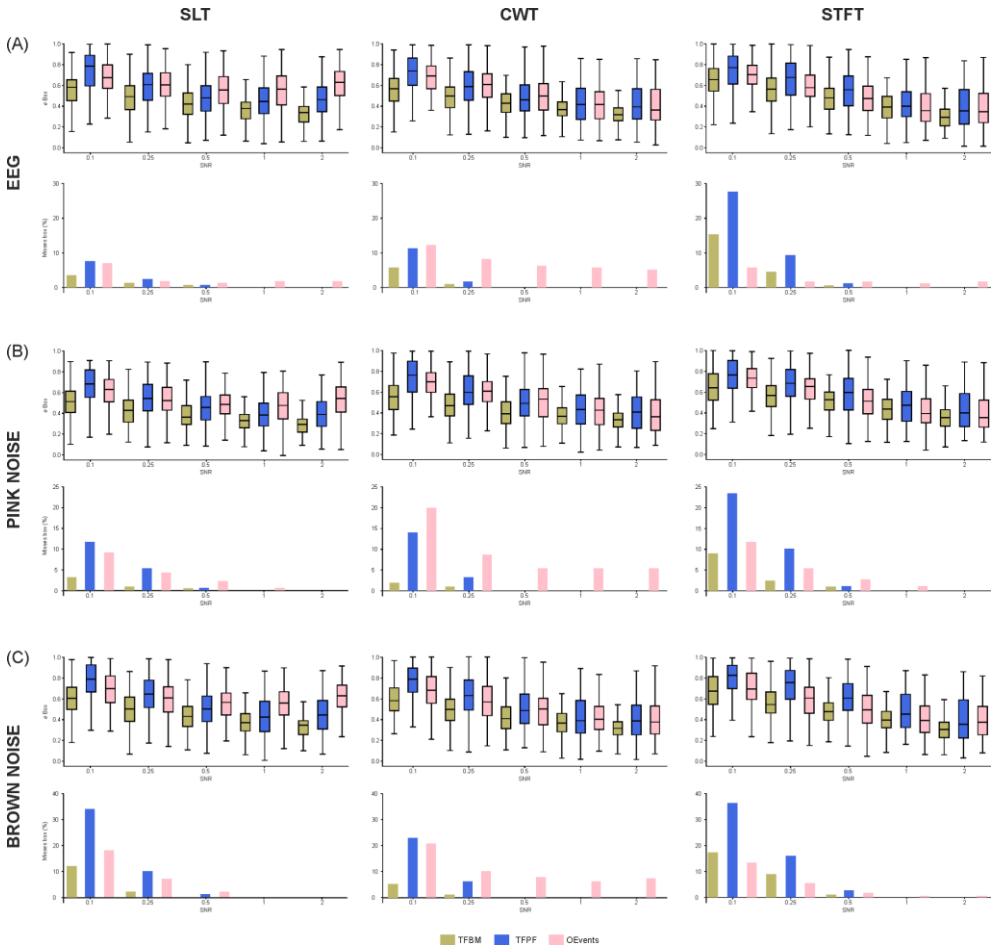

Figure 4.15 - A comparison of the detection algorithms performance for different background types and time-frequency representations. The detection performance of atoms within an EEG background is depicted in **(A)**. The upper panes display the box match error, while the bottom panes show the percentage of missed packets for three TFRs: SLT on the left, CWT in the centre, and STFT on the right. To facilitate comparison, the left panes in (A) recapitulate the top and bottom panes in Figure 4.15A. In **(B)** and **(C)**, the identical evaluation is presented for backgrounds of pink and brown noise, respectively.

Acknowledging that EEG might not be the most suitable background in all situations, we also tested atom detection with pink noise (Figure 4.15B) and brown noise

(Figure 4.15C) backgrounds. Each atom was embedded in a different instantiation (trial) of the noise, following the same embedding procedure as described for EEG background. Performance-wise, the algorithms demonstrate similar results on all backgrounds. TFBM generally performs best, with the exception of brown noise background with STFT. OEvents and TFPF show competitiveness at small SNRs (≤ 0.25), while at SNRs ≥ 0.5, TFPF has fewer overall misses than OEvents (except for brown noise with STFT at SNR = 0.5). Lowering the power threshold significantly enhances packet detection for TFBM and TFPF, even in the case of brown noise and pink noise backgrounds. TFPF and TFBM miss fewer packets compared to OEvents, although the box error is typically larger for TFPF than for OEvents. Similar to the EEG background case, TFPF and TFBM generally perform better on SLT and worse on STFT, with OEvents demonstrating poorer performance on CWT.

4.2.4. Conclusions

Our proposed method, referred to as the Time-Frequency Breakdown Method (TFBM) [175], employs the core principle of a clustering algorithm called the Space Breakdown Method (SBM) to identify peaks and expand regions around these peaks until certain stopping criteria are met. The second method, called the Time-Frequency Peak Finder (TFPF), is a threshold-based technique that divides the time-frequency landscape along the power axis, beginning from the power peaks and descending while merging successive peaks with the more prominent ones. Both methods were designed with the objective of identifying the precise contour of oscillation packets in time-frequency representations, rather than just their bounding box. Furthermore, it would be desirable to have techniques that can determine the hierarchical relationship between different, neighbouring time-frequency peaks. Requirements that are fulfilled by both TFBM and TFPF.

When comparing the two proposed methods, TFBM is more complex and computationally expensive but has better performance than TFPF or OEvents. Overall, both techniques outperform existing methods that can only extract bounding boxes of oscillation packets. The ability of the proposed methods to establish hierarchical relationships between components of the TFR allows for unprecedented analysis of underlying bursting processes compared with existing detection methods. It should be noted that there is no perfect method for detecting oscillation packets, but in general, more complex methods like TFBM tend to perform better. However, such methods come at a cost of increased complexity and parameters that need to be tuned by the user.

The effectiveness of super-resolution TFRs in detecting oscillation packets in neural data has been demonstrated, with the superlet transform performing particularly well. Furthermore, an innovative method for detecting oscillation packets has been presented, which enables precise isolation of the time-frequency components in the TFR and the determination of hierarchical relationships between peaks representing oscillation packets. These powerful tools for characterising oscillation bursts provide an opportunity for quantitative analysis of the properties of transient oscillations in brain signals. The true potential of these tools for detecting oscillation packets will be revealed through complex future studies of transient oscillation processes.

5. From brain activity to behaviour

The most natural and appropriate approach to understanding the brain is to analyse its activity during behaviour. Even though, experiments under anaesthesia have a role and can offer insights into brain function, they can alter brain states and as such do not show the brain activity that is found in real-life environments. This chapter dives into a more engineering type of approach in neuroscience. The first step for any experiment is the environment of the experiment itself and as such the first subchapter proposes experimental environments for the evaluation of vision, where the subject is conscious and able to move. And even more than that, for the second environment, the actions of the subject have effects in the environment. This approach allows the subject to enter a more natural state that provides the analysis with a clearer view of brain activity that is linked to the behaviour of the subject as the subject receives immediate feedback from the environment.

Having such a natural approach to experimental environments offers many advantages. Nevertheless, the behaviour of the subject must be monitored during the experiment to have the capability of linking the behaviour to the brain activity. As such, the second subchapter proposes a neural network-based approach to tracking behaviour based on predefined body parts from video recordings. Through the video recording and tracking of the subjects, their behaviour during the experiment can be detected in real-time. With these two approaches, the experimental environment and the tracking, the brain activity and behaviour of subjects can be easily linked while also reducing confound variables in order to understand how the brain functions in natural environments.

5.1. Experimental environment development for behaviour quantification

5.1.1. Introduction

Amblyopia, also known as “lazy eye”, is a malady of the eye that typically occurs during childhood and involves reduced acuity in one eye unrelated to structural abnormalities. If, during development, one of the eyes is significantly stronger than the other, the brain tends to favour that eye leading to amblyopia, a weakness in the other.

The most common treatment is to cover the stronger eye in order to force the development of the weaker as it was discovered that visual loss in the healthy eye results in improved function of the amblyopic eye [176]. The earlier this condition is found, the more chances of recovery. Left untreated, it may lead to permanent impairment of vision in the weaker eye. Amblyopia can lead to impairments in balance [177] and its prevalence only keeps on growing with estimated doubling of occurrence in the next decade [178]. Thus, the study of amblyopia and its effects on function is a valuable endeavour.

Sensory function measurements are critical for the appraisal of ability and its deterioration in research and in clinical practices. The scientific study of the stimulus-response dynamic is called Psychophysics. A sub-realm is the identification of a person’s ability to perceive changes in brightness, called Contrast Sensitivity.

The hypowork of this experiment is that amblyopia may also lead to impairments in prediction. As objects come towards you or go away from you, the speed and direction can be easily estimated by the brain. When the visual image does not overlap with the

prediction, for example an object suddenly changes direction, it is called a prediction mismatch, and it creates a significant amount of neuronal activity. It is hypoworked that this ability to detect mismatches is also impaired in amblyopic patients.

To establish whether this is the case or not, two experimental environments were developed. The first estimates the central and peripheral visual acuity of each eye, while only the second actually incorporates the mismatches. The whole setup includes three monitors for immersion, a device that simulates movement (such as a stationary bike for humans), an eye tracker to follow the eye movements of the subject and an EEG cap for the recording of neuronal activity.

For this experiment, multiple species are enrolled as subjects: mice, cats and humans. Humans are the goal of this study; therefore, it is essential that human subjects are a part of the experiments. Mice are one of the most prevalent and studied subjects in neuroscience, as such they are the baseline for this study, while cats have one of the best visual systems in the animal kingdom qualifying them to be part of this study.

An unambiguous approach to Contrast Sensitivity is to show the complete range with very small increments in order to assess the threshold at which the intensity is below visibility. Most commonly, adaptive procedures are used to reduce the testing time for both the experimenter and the patient. These procedures are called staircases, they adjust the stimulus dynamically based on previous responses to identify the participant's threshold [179].

Other methods, such as Quest [180] or Quest+ [181], have been developed based on statistics. They use maximum likelihood in order to estimate the probability of parameter values to be true. Thus, it allows the computation of the best stimulus, regarding information, for the next trial increasing the evaluation efficiency. Moreover, it allows for the evaluation of the entire psychometric function, not only the threshold, and the ability to assess the covariance of different parameters with distinct threshold values.

Traditional methods, such as Staircases, are considered more suitable than newer methods for children. Multiple factors are taken into consideration in this statement [182]. As children tend to exhibit a lack of attentiveness, a lack of patience, lapses in concentration and nonstationary behaviours, the efficiency of more complex methods decreases. Moreover, trials in Staircase methods can be separated into two categories: consolidation and motivational trials. Consolidation trials are the incipient trials that allow the subject to learn the task, while motivational trials are the effortless trials followed by an erroneous response to the stimulus. Another tendency of children is to relinquish the task temporarily when approaching the threshold as the difficulty increases. They stop responding correctly for several trials after such cases, generating "saw tooth" patterns [183]. Thus, the reversals provided by Staircase methods might increase the likelihood of children to regain interest in the task.

More complex methods might not be always necessary as the final results can be congruent with simpler methods, although complex methods have been found to produce results more quickly [182]. Furthermore, the intricacies of parameter choice increase with algorithm complexity increasing the possibility of errors being introduced by the experimenter rendering the measurements biassed or noisy.

5.1.2. Methods

The purpose of the software component is to evaluate the predictive acuity of the person participating in the experiment. The experiment consists of presenting visual

stimuli on a configurable number of monitors and estimating the participants' ability to identify these stimuli. The software component is divided into two: estimating the participant's ability to identify stimuli at varying contrast values, referred to as contrast sensitivity assessment, and estimating the ability to identify stimuli that do not meet expectations.

The Contrast Sensitivity Evaluation

The first component of the software, Contrast Sensitivity Function (CSF), is a configurable system that presents stimuli as Gabor filters, shown in Figure 6.2, against a background containing noise. The goal is to evaluate contrast sensitivity in multiple areas on the screen corresponding to areas of human frontal, median, and peripheral vision. Thus, the experiment setup is dependent on multiple monitors at a calculated angle to capture the field of view. Thus, the visual field represented by the monitors is divided into multiple zones, chosen by the experimenter. The participant interacts with the keyboard, the keys are chosen by the experimenter, to declare whether the presented stimulus is visible or not. The contrast of a stimulus is changed in accordance with the participant's response. A 'yes' answer reduces the contrast thus reducing the visibility of the stimulus. An answer of 'no' is considered as the inability of the participant to identify the existence of the stimulus in a chosen time period. Stimuli are presented in a random area, each area actually containing a chosen number of stimulus configurations, the presented configurations being also chosen at random. An example of a stimulus configuration is a stimulus with a spatial frequency of 2.5 measured in cycles/visual degrees, with an orientation of 45 degrees and a motion speed of 0.1 measured in visual degrees/second. The consistency of the presentation of stimuli is given by their contrast, a stimulus of a certain configuration that has been identified causes the next stimulus of the same configuration and from the same area to have reduced contrast.

To determine the contrast values, an estimation algorithm was chosen and implemented. Thus, the algorithm produces a contrast value based on all the participant's responses at all contrast levels up to that point. The visibility threshold is determined using the previously mentioned values using curve modelling. Thus, at the end of the experiment, each stimulus configuration in each area contains an individual visibility threshold.

Figure 6.1 presents the configuration options of the Contrast Sensitivity Function experiment. The system allows configuring the following parameters: number of screens, noise range as pixel values, the distance between the observer and the main screen in millimetres, the horizontal and vertical lengths of the screen in millimetres, the number of evaluated zones, which zones may be active at an instant and their size in percentages, stimulus size measured in visual degrees, stimulus spatial frequency measured in cycles per visual degrees, stimulus movement speed measured in visual degrees per second, stimulus orientation in degrees, the duration of the stimulus presentation in seconds and keyboard interaction keys. The measurement of the distance of the participant to the monitors and the size of the monitors in millimetres is required for the correct calculation of the size of the stimuli.

The first task of this study is to measure the CSF using a Gabor detection procedure. Gabor filters are ideal for the presentation of gratings as a stimulus in visual tasks. They can be viewed as a sinusoidal regulated by a Gaussian. The Gabor filter allows

for a high variability through its parameters thus it is suitable for stimulus presentation [184].

The stimuli used in this study are randomly oriented Gabor patches of variable spatial frequency and contrast. The ranges of the spatial frequency and contrast used can be viewed in Figure 5.1. The experiment was set up to cover the whole field of view with three monitors at different angles. Thus, even the peripheral view is evaluated. It is important to note that peripheral vision is acutely weaker than central vision for both optical and neural reasons [185].

config.ini

```
[SETUP_PARAMETERS]
SCREEN_NR = 1
OBSERVER_DISTANCE = 915
SCREEN_WIDTH = 596
SCREEN_HEIGHT = 336
TIMING = 50
NOISE_WIDTH = 0

[ZONE_PARAMETERS]
NR_ZONES_PER_ROW = 2
NR_ZONES_PER_COL = 2
ROW_ZONES_PERCENTAGES = 50
ACTIVE_ZONES = HORIZONTAL

[STIMULUS_PARAMETERS]
STIMULUS_SIZE = 7.5
SPATIAL_FREQUENCIES = 4
ORIENTATIONS = 0
SPEEDS = 1
FADE_IN_DURATION = 3
SPAWN_TIME_MIN = 2
SPAWN_TIME_MAX = 6

[KEYBOARD_INPUT]
TOP_LEFT = q
BOTTOM_LEFT = a
TOP_RIGHT = p
BOTTOM_RIGHT = l
PAUSE = enter
EXIT = [*]
```

Figure 5.1 - Configuration file for the CSF experiment.

The Gabor patches are evenly distributed along the field of view and move around the monitor with no possibility of overlap with another patch. The stimulus is presented for a given amount of time, 10 seconds, on the screen and traverses a certain section randomly. The participant must focus on the fixation point at the centre of the middle screen. At the start of such an experiment, full contrast Gabor patches are presented, as shown in Figure 5.2. A correct response is regarded as the sighting of the Gabor patch and the pressing of a key corresponding to that zone of the screen. The Gabor patch has a

lifespan of 10 seconds, if the participant is unable to see it and press the correct key during its lifespan, the patch disappears and is considered as the incorrect response.

The experiment can be configured to contain stimuli with multiple spatial frequencies, and they are presented randomly. Moreover, the contrast is linked to each individual stimulus that is determined by the zone it is presented in and its spatial frequency. Depending on the responses of the subject, the contrast increases or decreases to estimate the visibility threshold of the subject. A screenshot of the experiment for a three-monitor setup is shown in Figure 5.3.

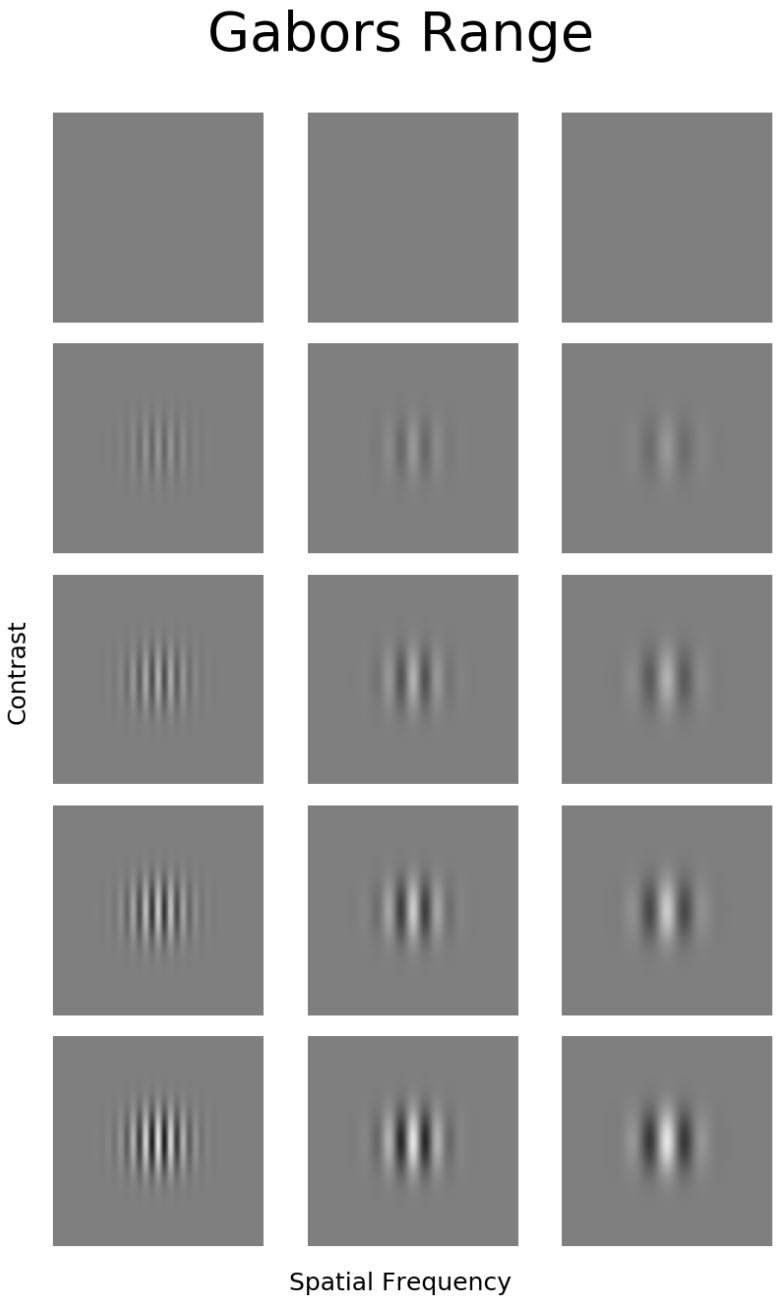

Figure 5.2 - Gabor patches with the varying contrasts and spatial frequencies used in the experiment.

Furthermore, because peripheral view is evaluated as well, the field of view has been divided into sections and each section contains all spatial frequencies. Thus, when

the subject has a correct or incorrect response it only affects the contrast of the presented patch of a certain spatial frequency in a certain zone. Through this outline, the evaluation of the participants' visual abilities was made.

Hence, multiple zones are defined with the possibility of multiple spatial frequencies being presented in each zone, at any time the field of view can contain anywhere between zero and a number of Gabor patches equivalent to the number of zones. Naturally, the number of patches present at one time is dependent upon the participant's responses. After a response has been attributed, the patch of a zone disappears, and another is generated within that zone in a pseudo-random amount of time dependent upon how many Gabor patches are present when the responses were assigned.

The length of the assessment of a spatial frequency per zone has been limited to a maximum of 30 trials. Because all zones can present Gabor patches at the same time, it does not influence the length of the experiment. In the worst case, when the participant is unable or unwilling to see or interact with the patches, given the lifespan of a Gabor patch, the experiment can extend only up to 18 minutes. Depending on the wishes of the experimenter, the distribution of Gabor patches into zones can also be configured, such that patches can simultaneously appear only on the vertical or horizontal lines.

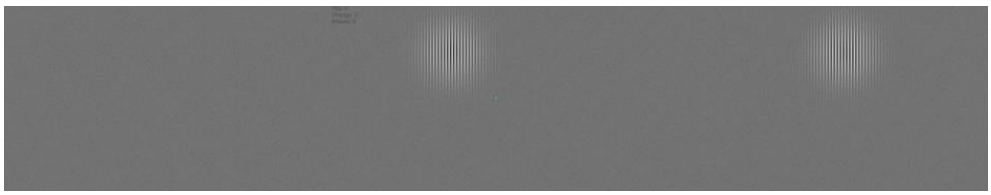

Figure 5.3 - Contrast sensitivity evaluation system configured for 3 monitors; two Gabor stimuli can be observed with an orientation of 0 degrees.

The Mismatch Tunnel

The second software component, called the Predictive Ability Assessment, is a system for evaluating the participant's ability to identify discrepancies in expectations. This experiment was modelled as a simulated walk through a tunnel where the walls were made of white or black squares. Passing through the tunnel triggers certain pieces of the walls to become immobile or to actually change direction, which produces an alarm signal, the expectation being that during movement the subject sees them pass. This system as well is configured to have 3 monitors, where the central monitor presents the frontal vision, and the other two are positioned at a 90-degree angle to the first one to cover the field of vision. A photo of the screen displays during the experiment is shown in Figure 5.4. While in Figure 5.5, a first setup for the testing of the environment is shown. Given that both systems are created for participant interaction through multiple monitors at preset angles, images are not the most representative to demonstrate functionality. Nevertheless, for this second experiment, the mismatches cannot be seen in an image as they are based on the movement of specific sections of the environment.

Just as the CSF system, this system is also configurable by multiple criteria as shown in Figure 5.6. The environment is composed of a multitude of objects, at the highest level it is composed of tunnel sections and their number, and the starting position of the player can be set from the configuration file. Each wall beside the floor is composed of plates and their numbers are again configurable, and each plate has a texture of black and white squares that is also configurable from the config file.

The experiment comprises two types of mismatches, local and global. The local mismatches appear as plates from the wall moving in the opposite direction than expected for a set amount of time. While global mismatches appear as the whole environment freezing for a set amount of time even though the subject is moving. The probability of both types of mismatches is configurable. The response to these mismatch stimuli is given through keys that are also configurable. The subject must specify on which wall the local mismatch appears and if a global mismatch happened.

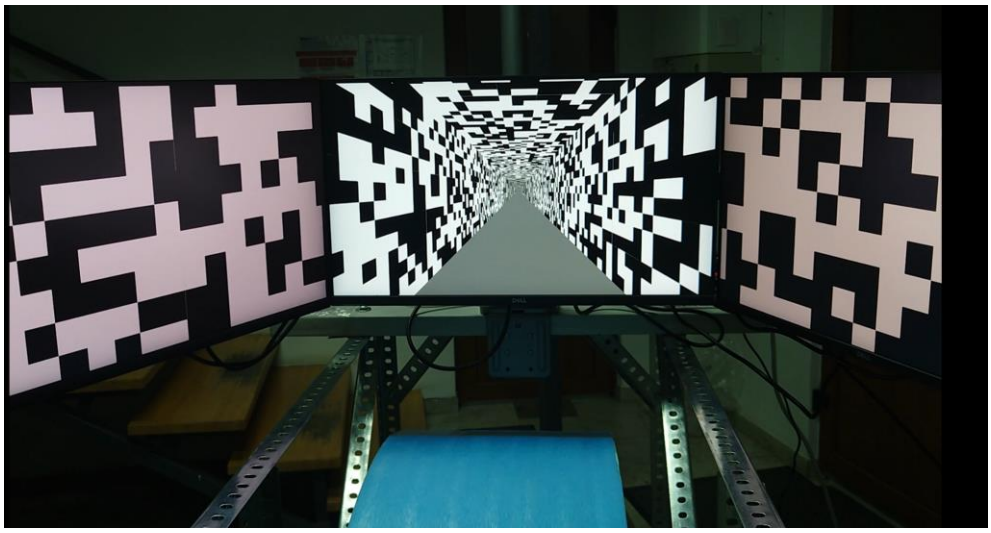

Figure 5.4 - Predictive Ability Assessment System.

Figure 5.5 - Example of a feline subject for a first setup for the testing of the environment.

Due to the high amount of training needed for mice and cats before the actual experiment can begin. The local mismatches have been made configurable to contain images of different objects instead for the training period. The mismatches as wall plates for the actual experiment can be incorporated only after a long training time of the subject.

Both systems have incorporated an exit and pause option that have configurable keys for exceptional cases.

config.ini

```
[SETUP]
TUNNEL_LENGTH = 50
PLAYER_POSITION = 10
WALL_PLATE_LENGTH = 5
WALL_PLATE_HEIGHT = 2
LATERAL_WALL_PLATES = 5
CEILING_WALL_PLATES = 5
PLATE_TILE_ROWS = 3
PLATE_TILE_COLS = 8
PHYSICS_TIMER = 0.02

[MOVEMENT]
PLAYER_SPEED = 5
PLAYER_MOUSE_SPEED = 0.1
PLAYER_SCROLL_SPEED = 0.5
DISABLE_BACKWARD_MOVEMENT = 0
SIGNAL_BACKWARD_MOVEMENT = 0

[MISMATCH]
LOCAL_MISMATCH_DURATION = 5
LOCAL_MISMATCH_TILE_RADIUS = 0
LOCAL_MISMATCH_PROBABILITY = 0.1
LOCAL_MISMATCH_PLATES_AHEAD_MAX = -1
LOCAL_MISMATCH_PLATES_BEHIND_MAX = 4
GLOBAL_MISMATCH_DURATION = 4
GLOBAL_MISMATCH_PROBABILITY = 0.0
EXCLUSIVE_MISMATCHES = 1
USE_MISMATCH_TEXTURES = 1

[DISPLAY]
ENABLE_VSYNC = 1
TARGET_FRAME_RATE = -1
ENABLE_FRAME_COUNTER = 1
FRAME_COUNTER_FPS_FLOOR = 100
FRAME_COUNTER_TRIAL_FRAMES_ONLY = 0
USE_FIXATION_CROSS = 1
USE_FLICKER_IMAGE = 0

[INPUT]
MOVEMENT_ENABLE_TREADMILL = 1
MOVEMENT_ENABLE_MOUSE = 1
MOVEMENT_ENABLE_SCROLL = 1
EXIT = escape
PAUSE = space
LEFT_MISMATCH = a
TOP_MISMATCH = w
RIGHT_MISMATCH = d
GLOBAL_MISMATCH = s

[ADDITIONAL_RESOURCES]
TEXTURE_FILE_COUNT = 4
TEXTURE_FILE_0_NAME = FISH-1
TEXTURE_FILE_0_FILE = ./utilities/fish1.png
TEXTURE_FILE_1_NAME = FISH-2
TEXTURE_FILE_1_FILE = ./utilities/fish2.png
TEXTURE_FILE_2_NAME = FROG-1
TEXTURE_FILE_2_FILE = ./utilities/frog1.png
TEXTURE_FILE_3_NAME = FROG-2
TEXTURE_FILE_3_FILE = ./utilities/frog2.png
FEEDBACK_GOOD_SOUND_FILE = wizardball-feedback-good
FEEDBACK_BAD_SOUND_FILE = wizardball-feedback-bad
```

Figure 5.6 - Configuration file for the Predictive Ability Assessment experiment.

5.1.3. Conclusions

Mice have been chosen as they have one of the longest experimental track records in neuroscience and because of the genetic mutations that can be induced with today's technology. Within this context, they can be considered as the perfect validation. Cats, even though they are hard to train, have one of the best visual systems and amblyopia has been previously induced in the eyes of cats which is why they have been chosen as subjects. Lastly, humans are the main point of this investigation.

The use of these systems requires the construction of particular physical environments that have to be approved by various ethics entities. The animal subjects require a long period of training before the actual experiments can commence.

Furthermore, these systems also incorporate eye tracking and the recording of EEG signals, parts that are yet to be implemented.

These systems are only the first step of a long endeavour attempting to answer the question of how amblyopia affects normal function. It is an engineering venture, with fuzzy requirements that may change with new discoveries and as the project advances forward, to create a new type of experiment in the form of an environment that is more realistic and natural for the subjects.

5.2. Behaviour quantification through tracking

5.2.1. Introduction

The unique biological characteristics of zebrafish make them suitable for study in the domain of neuroscience. They are transparent allowing for the direct observation of study of the development of the nervous system. They have a quick development allowing for the observation of the early neural circuit formation and its relationship to function and behaviour.

The first movements of zebrafish occur at 17 hours after fertilisation, but the behaviour remains limited until 3 days post fertilisation. Stable patterns, like active swimming and increased response to visual and acoustic stimulation, develop between 3 and 5 days. After 5 days, the brain development rate is reduced. By day 6-7, zebrafish require feeding for normal behaviour and social behaviour starts to develop. And by day 9, zebrafish show a preference to swim alongside similarly pigmented individuals, while larvae show no such preference. Predation seems to be improved by practice as its efficiency is increased in zebrafish that reside in environments with excess food [186].

Zebrafish have been found to inhibit environments with water temperature between 16 and 38C. The most appropriate temperature for development was found to be at around 28.5 Celsius, with lower temperatures resulting in a slower development. Besides temperature, the level of oxygen also impacts the development of the zebrafish. Low oxygen levels thwart normal development and result in abnormal behaviours, creating high density cultures with remaining unfertilized eggs [186].

The basic movements of the zebrafish can be categorised into 9 unique motor acts that stabilise between days 5 and 7. More complex movements can be deconstructed into these basic movements that last less than 50ms with movements being delimited by long intervals of inactivity. The movements of the adult zebrafish are continuous, increasing the difficulty of analysis [186]. The 9 basic movements of the zebrafish are: the slow swim which occurs spontaneously without need of a stimuli, the burst swim that occurs for looming stimuli as a predator escape, the capture swim and the j-turn are both predation sequences, the o-bend ensues in response to dark flashes, routine turns of about 40 degrees are used for orientation but can also occur spontaneously, the short (SLC) and long (LLC) latency C-bends are startle response that occur in response to acoustic or tactile stimuli after about 15 ms and 20-60 ms respectively, and lastly, the struggle manoeuvre which is an embedded stimulus [186].

Zebrafish respond to various stimuli: acoustic/vibrational, visual, tactile, vestibular, lateral line and chemical. Within the context of visual recordings, the acoustic and vibrational stimuli are not preferable because of the jitter that can appear. Zebrafish larvae respond with a fast C-bend to sudden stimuli of this type. As mentioned above, LLC occurs slower after the stimulus than SLC, therefore SLC are considered to be the faithful startle response, it has been theorised that LLC is a more calculated response that requires more processing time and may be a response for stimuli perceived as less threatening. Habituation has been observed to occur if the stimuli is repeated at short intervals and may be the reason for the loss of startle response, and consistent with this idea, the responsiveness is regained after no stimulus is presented for several minutes. Other possibilities might be fatigue or sensory adaptation [186].

The vestibular response has been seen after 4 days post fertilisation, as zebrafish are able to preserve the dorsal-up position through visual cues such as light. Zebrafish

have a high sensitivity to tactile stimuli and respond with escape manoeuvres. Approaching with a probe through water may also activate vibrational and lateral line stimuli as well. The lateral line is an organ used to detect changes in motion of the water and zebrafish can have startle responses with regards to lateral line stimulation [186].

By day 3 post fertilisation, visual stimuli can be used to produce responses in zebrafish. The looming stimuli produces innate escape responses, while light intensity variations also produce movements. The red and blue wavelengths have been found to be the most effective as the zebrafish tend to swim toward the light source. Also, within 3 days, the olfactory system is developed, and zebrafish start to respond to chemical stimuli acquired through gustation, olfaction or specialised skin cells. Certain stimuli have been observed to produce aversion through locomotor responses, such as the odour produced by the damaged skin of kin [186].

The activity of larvae and adults are modulated by light, extended immobility has been observed mostly at night while swimming occurs during daytime. The circadian clock can be scheduled, and zebrafish can swim even in darkness if it coincides with the daylight of the entrainment. New environments yield reduced motility that revert to normal after accustomization [186].

The hypowork is that through the tracking of the movements of zebrafish across several days in their development, the impact of pharmaceuticals can be inferred.

This project contains multiple integrated parts. From the recording made through an optic system including a high-resolution camera and mirrors, the microplate and its specific design for the pumping of pharmaceuticals to the software. Only the software design are presented here.

The goal of this application is to survey the zebrafish and their health during a period of time while pharmaceuticals are being pumped into their microenvironments. Thus, the zebrafish are constantly monitored including localization, orientation, speed and acceleration in order to infer the effects of pharmaceuticals. As the resolution of the images provided by the camera is 2592x1944, the classification is done per well in order to reduce the workload and allow for parallel processing.

Systems used for the analysis of visual recordings of zebrafish have been developed [187], even with neural networks (NN) for detection [188]. In general, Convolutional NNs (CNNs) are used in the identification of objects in videos, with background subtraction algorithms allowing for prelabelled data generation [188]. The complexity of the images may drive the architecture of the neural network, while certain requirements such as processing speed limits the complexity of the architecture. As the movements of the zebrafish are intertwined with long periods of immobility [186], data augmentation is recommended in order to increase the dataset size. Data augmentation can be achieved through translations, rotations and other image processing operations. Real time processing on the CPU with high accuracy is difficult to achieve with a high frame rate.

ImageNet [189] is a large-scale database of 3.2 million images that has become widely used for transfer learning [190]. The current segmentation networks are formed of two parts: an encoder and a decoder. The encoder is usually a pretrained network, such as ResNet[191], VGG-Net[192], AlexNet [193], Inception and Xception, sometimes called the backbone [194]. Transfer learning occurs by pretraining these networks using, for example, ImageNet. Among the backbones, Xception, which is an optimization of ResNet, has been found to have the best accuracy on ImageNet [194] with a fast convergence and

high accuracy for linear activation. One of the main differences between backbones is the size of the convolution kernel used. VGG has smaller convolution kernels (3x3) which result in improved fitting, and it uses the ReLU activation function for non-linearity, while ResNet (7x7) and AlexNet (11x11) have bigger convolution kernels.

U-Net [195] is a Convolutional Neural Network optimised for segmentation and composed of the two parts, where the first part extracts the features of the image by contracting and the second is up-sampling where it is expanded. U-net is based on Fully Convolutional Networks (FCN) [193]. On the contraction, the spatial information of the image is reduced while the number of features is increased. On the expansive part, the opposite happens by up-sampling and concatenating the features from the contractive part [194]. U-Net is trainable on all layers, whereas other architectures may use untrainable encoders and only train the decoder.

Increasing the complexity of deep learning models usually requires an increase in the amount of data used for training in order to avoid overfitting and produce a model that is able to generalise. Data augmentation has been used in order to increase the number of training samples and can prevent overfitting [192,196]. Using augmentation, the model is able to learn robustly. Examples of operations used in data augmentation are: translation, cropping, scaling, rotations, brightness/contrast change, noise addition, horizontal/vertical flips and many others [196].

Various toolboxes have implemented deep learning networks for image segmentation with high accuracy. These toolboxes have been used in current literature for the processing of videos. One such toolbox is DeepLabCut [197]. DeepLabCut allows the user to choose a desired network from a list containing ResNet [191], EfficientNet [198] and others that have been pre-trained using ImageNet [189]. Figure 5.7shows the result of DeepLabCut for one frame for the labelling of 3 zebrafish each in a separate well.

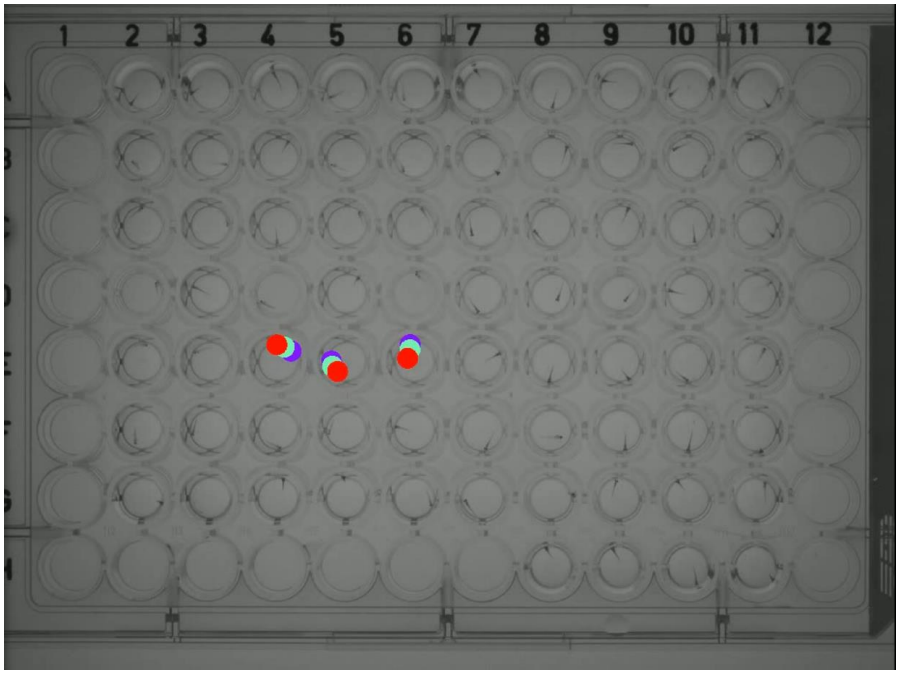

Figure 5.7 - Inference of a ResNet model through DeepLabCut

As of this moment, these methods do not allow for the detection of multiple animals at the same time. As such, a custom implementation of a CNN was chosen.

5.2.2. Methods

Several ML approaches have been applied attempting to track the movements of the zebrafish in real time. Current methods are based on background subtraction where the median, mean, or mode of a video is subtracted from each frame on the supposition that this statistic renders an image containing only the wells without any of the zebrafish. Multiple issues can become apparent, in some cases the fish does not move rendering this method unable to localise the fish at all. It requires the setting of a manual threshold as slight vibrations may appear or changes in light that heavily affect the results. Moreover, the movements of the zebrafish can be so quick that theyare not be detected at all through this method as it can be seen only as a blur. Even though this method may be one of the most efficient from the perspective of execution time, it remains lacking in the department of accuracy.

The disadvantage of classifiers is the requirement of labels meaning that a tedious process of manual labelling must be carried out. Depending on the method, thousands of images may need to be labelled by hand and the results of any method may be only as good as the labelling. Nevertheless, there are techniques, such as image augmentation, that can help enlarge the dataset or such as automated labelling, where only a small subset of the data is labelled and then the model is trained on this subset and used to label the rest.

Machine learning approaches

Two other machine learning methods that can be used for image segmentation are Support Vector Machines (SVM) [199] and RandomForest (RF) [200]. These methods can be taught to learn by generating different features through image processing of the image, such as Gabor features, edge detection, filtering and others. An example of image segmentation generated by an SVM model trained on a single image is shown in Figure 5.8, while Figure 5.9 shows the result of a RF model.

Figure 5.8 – Inference of a trained SVM model for image segmentation. The left and middle image are the input image and mask used to derive new features and to train the model and the right is the segmentation of SVM.

Figure 5.9 – Inference of a trained RandomForest Classifier for image segmentation. The left and middle image are the input image and mask used to derive new features and to train the model. The image from the right is the segmentation of the RandomForest Classifier.

The disadvantages of these methods are the high amount of training needed even for the training of a single image, while the robustness of its results decreases as more and more different images are added into the training set.

Pulse Coupled Neural Networks

For image segmentation, based on the iterative process, one possibility would be to stop once a certain threshold is reached. In the algorithm we used, the iterations stop when the number of pulses reaches the number of pixels in the image. This algorithm attempts to synchronise each region and desynchronize adjacent regions. Similar pixel values tend to fire synchronously and thus pixels of an object become correlated temporally while pixels of distinct object anti-correlated [81]. Figure 5.10 presents the image segmentation made by PCNN on a well.

Figure 5.10 – Image segmentation made by PCNN classic model, on the right is the input and on the left the segmented image. It is able to discern the head of the fish from the rest of the image.

The disadvantage of this method is that each image needs to be segmented by itself and it may require more time than is available in real time processing. Moreover, the method is only capable of distinguishing the head of the fish which limits the ability of this method to detect the orientation or the position of its body.

Proposed method based on Convolutional Neural Networks

As mentioned previously, the original images provided by the camera are high-resolution. Thus, the processing was reduced to that of a well and it was applied to all the wells of the micro-plate. An example of a well, an image of 208x208, is shown in Figure 5.12. Classification requires labels, in order to increase its accuracy, the images were labelled manually with three points.

As a first step of preprocessing, the labels have been transformed into masks containing 4 classes. Using the three points provided by the manual labelling, three body parts were defined in the image as circles of 5 pixels in diameter each with its own class, while all other points in the mask were given the background class. An example of a mask can be found in Figure 5.12, where each different colour represents a different class.

The model architecture used is a variation of U-Net. In contrast to the classical U-Net, this model applies a single convolutional layer before maxpooling in the encoder part, and a single transposed convolutional layer in the decoder part instead of applying convolutional layers after the transpose. Thus, the model is formed out of two sub-models: an encoder and a decoder. Several pre-trained encoder architectures (MobileNetV2, EfficientNetB2) containing convolutional layers along with batch normalisation and a ReLU activation have been tested. The encoder is based on transfer learning and its layers have been frozen. Therefore, the learning of the image features is done by the decoder. The decoder contains layers of concatenation receiving inputs from the encoder's corresponding-by-size layers. It includes layers of transposed convolution, batch normalisation, dropout and ReLU activation. The architecture is shown in Figure

5.11, in order to reduce its size only the main blocks are presented, not all actual layers. The network is provided with RGB images and the target which is a mask of the same size with only 1 channel. The output of the network is an image of the same size with 4 channels, each channel containing the network's output for each class.

An essential observation is that the number of pixels for the body part classes are extremely low when compared to the whole image, each body part class having under 1% of the total number of pixels. Thus, weighting has to be added for these classes, in order for the model to not fall into the local minima of predicting all pixels to be of the background class. An example of a prediction is shown in image Figure 5.12, along with the image and its true mask, while the loss and accuracy during training can be viewed in Figure 5.13. It can be seen from this figure, that the loss decreases across epochs, and the accuracy increases. The testing loss and accuracy values of 0.002 and 0.971, respectively, indicate that the model is able to generalize and has not been overfitted. As mentioned above, the network outputs an image with 4 channels. In order to transform it into a singular output as is presented in the figure, postprocessing has to occur. In this case, for each pixel in the mask, the class with the highest value of the four classes was chosen.

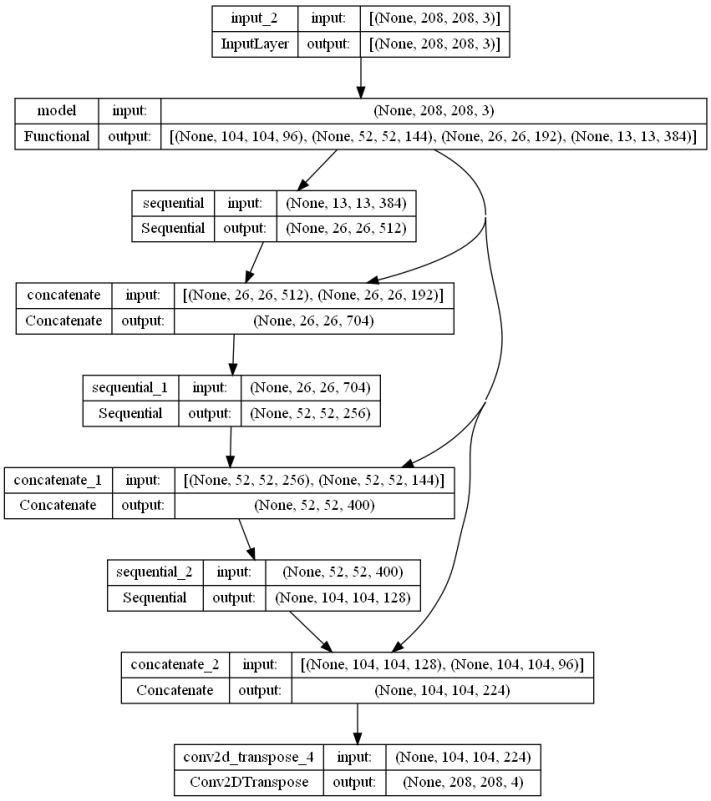

Figure 5.11 - Modified U-Net model architecture for Zebrafish Image Segmentation.

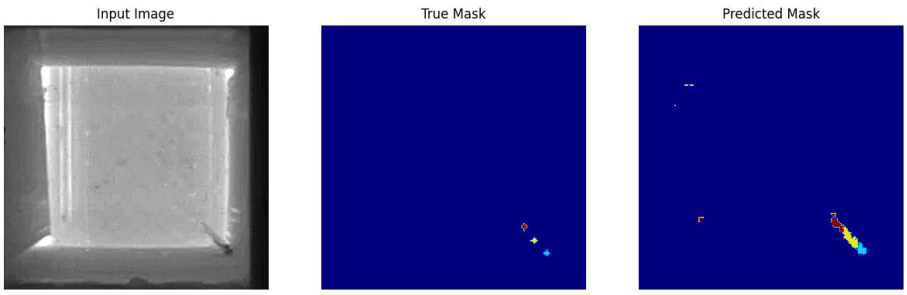

Figure 5.12 - The inputs of the proposed approach are presented in the left and middle images, while the right is the output obtained.

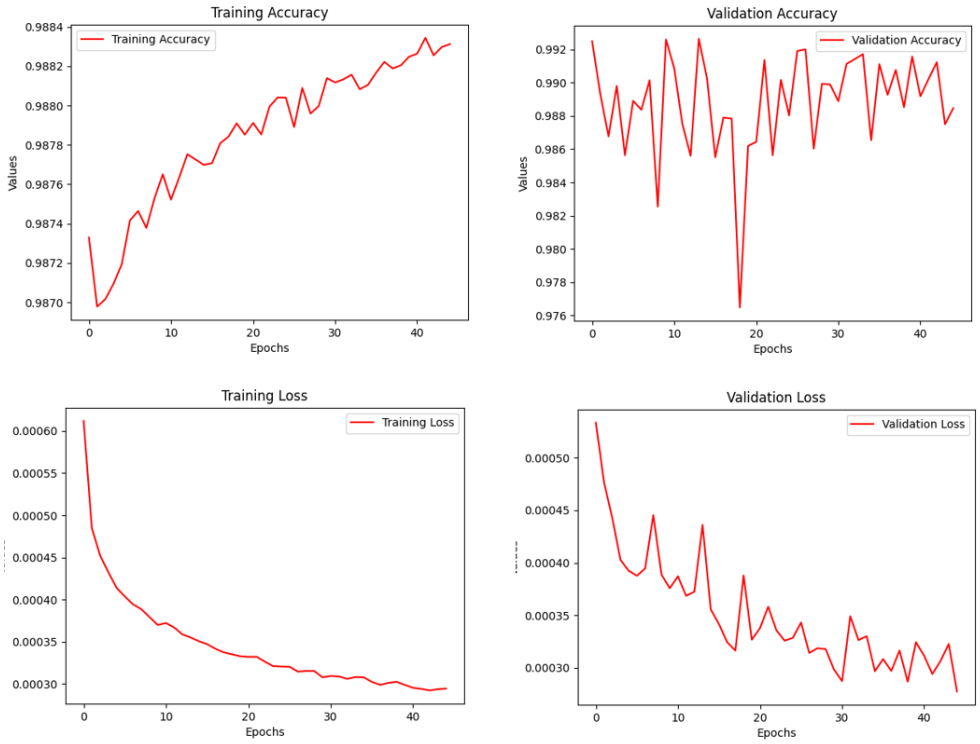

Figure 5.13 - Loss (bottom) and accuracy (top) of the model throughout both training (left) and validation (right).

Although our proposed CNN-based approach was developed for the tracking of zebrafish, it can also be used in the tracking of subjects for the previous direction described in Section 5.1, where the subjects' eyes need to be tracked in order to determine whether they are able to see the changes in the environment. This is hardest to do for animal subjects where they move their whole head, and the eyes leave the traditional eye tracking systems. In cases like these, the proposed approach can be useful to determine the exact position of parts of the head of subject in order to determine on which part of the environment the animal subject was focused on. An example of how the proposed approach would fare in such an endeavour is presented in Figure 5.14.

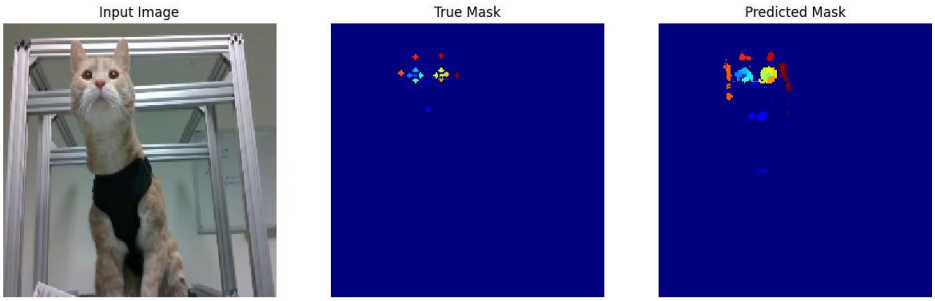

Figure 5.14 - The inputs of the proposed approach are presented in the left and middle images, while the right is the output obtained.

5.2.3. Conclusions

A CNN model is the most appropriate choice for a robust detection of zebrafish. However, there exists no universally accepted method or metric to allow for the evaluation of the performance of such a method.

The manual labelling is required for the training of the model which is a tedious and time-consuming procedure. The training itself takes hours upon hours even on a high-performance unit. A model can be created to detect multiple zebrafish in the same image however its robustness decreases and as such, a model for the detection of a single zebrafish in a well was chosen.

For the real-time processing of frames during recording, several models must be created such that the workload is balanced among these. Another approach may be taken, if there is not enough memory available, to choose when to apply the CNN model. One possibility is to use the background subtraction method to decide when movement occurred in one of the wells and only then apply the CNN to localise the zebrafish.

REFERENCES

1. Jun, J.J.; Steinmetz, N.A.; Siegle, J.H.; Denman, D.J.; Bauza, M.; Barbarits, B.; Lee, A.K.; Anastassiou, C.A.; Andrei, A.; Aydın, Ç.; et al. Fully Integrated Silicon Probes for High-Density Recording of Neural Activity. *Nature* **2017**, *551*, 232–236, doi:10.1038/nature24636.
2. Rey, H.G.; Pedreira, C.; Quian Quiroga, R. Past, Present and Future of Spike Sorting Techniques. *Brain Res. Bull.* **2015**, *119*, 106–117, doi:10.1016/j.brainresbull.2015.04.007.
3. Mishra, S.; Sarkar, U.; Taraphder, S.; Datta, S.; Swain, D.; Saikhom, R.; Panda, S.; Laishram, M. Principal Component Analysis. *Int. J. Livest. Res.* **2017**, 1, doi:10.5455/ijlr.20170415115235.
4. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. *Proc. Fifth Berkeley Symp. Math. Stat. Probab. Vol. 1 Stat.* **1967**, *5.1*, 281–298.
5. Diday, E.; Billard, L. Symbolic Data Analysis: Conceptual Statistics and Data Mining. *Symb. Data Anal. Concept. Stat. Data Min.* **2006**, doi:10.1002/9780470090183.
6. Hernández-Arteaga, E.; Ågmo, A. Seminatural Environments for Rodent Behavioral Testing: A Representative Design Improving Animal Welfare and Enhancing Replicability. *Front. Behav. Neurosci.* **2023**, *17*.
7. Azevedo, F.A.C.; Carvalho, L.R.B.; Grinberg, L.T.; Farfel, J.M.; Ferretti, R.E.L.; Leite, R.E.P.; Filho, W.J.; Lent, R.; Herculano-Houzel, S. Equal Numbers of Neuronal and Nonneuronal Cells Make the Human Brain an Isometrically Scaled-up Primate Brain. *J. Comp. Neurol.* **2009**, *513*, 532–541, doi:10.1002/cne.21974.
8. Cosgrove, K.P.; Mazure, C.M.; Staley, J.K. Evolving Knowledge of Sex Differences in Brain Structure, Function and Chemistry. *Biol. Psychiatry* **2007**, *62*, 847–855, doi:10.1016/j.biopsych.2007.03.001.
9. Fundamental Neuroscience for Basic and Clinical Applications - 5th Edition Available online: https://www.elsevier.com/books/fundamental-neuroscience-for-basic-and-clinical-applications/9780323396325 (accessed on 2 March 2023).
10. Ej, F.; Cc, I.; L, L. The History of the Development of the Cerebellar Examination. *Semin. Neurol.* **2002**, *22*, doi:10.1055/s-2002-36759.
11. Principles of Neuroanatomy: Angevine, Jay B., Cotman, Carl W.: 9780195028867: Amazon.Com: Books Available online: https://www.amazon.com/Principles-Neuroanatomy-Jay-B-Angevine/dp/0195028864 (accessed on 2 March 2023).
12. Saladin, K.S. *Human Anatomy*; 3rd Revised edition.; McGraw-Hill Medical Publishing: New York, 2011; ISBN 978-0-07-122207-5.
13. Uhlhaas, P.J.; Haenschel, C.; Nikolić, D.; Singer, W. The Role of Oscillations and Synchrony in Cortical Networks and Their Putative Relevance for the Pathophysiology of Schizophrenia. *Schizophr. Bull.* **2008**, *34*, 927–943, doi:10.1093/schbul/sbn062.
14. Herculano-Houzel, S. The Human Brain in Numbers: A Linearly Scaled-up Primate Brain. *Front. Hum. Neurosci.* **2009**, *3*, 31, doi:10.3389/neuro.09.031.2009.
15. Sapolsky, R.M. *Behave: The Biology of Humans at Our Best and Worst*; Illustrated edition.; Penguin Press: New York, New York, 2017; ISBN 978-1-59420-507-1.
16. Henson, K.T.; Eller, B.F. *Educational Psychology for Effective Teaching*; 2nd edition.; Kendall Hunt Publishing, 2012; ISBN 978-0-7575-9680-3.

17. Bear, M.; Connors, B.; Paradiso, M. *Neuroscience: Exploring the Brain: Fourth Edition*; 2015; p. 975;.
18. Cajal, S.R.Y. *Comparative Study of the Sensory Areas of the Human Cortex*; Andesite Press, 2017; ISBN 978-1-375-77722-3.
19. Tao, H.W.; Poo, M. Retrograde Signaling at Central Synapses. *Proc. Natl. Acad. Sci.* **2001**, *98*, 11009–11015, doi:10.1073/pnas.191351698.
20. Carter, M.; Shieh, J. Chapter 4 - Electrophysiology. In *Guide to Research Techniques in Neuroscience (Second Edition)*; Carter, M., Shieh, J., Eds.; Academic Press: San Diego, 2015; pp. 89–115 ISBN 978-0-12-800511-8.
21. Pedreira, C.; Martinez, J.; Ison, M.J.; Quian Quiroga, R. How Many Neurons Can We See with Current Spike Sorting Algorithms? *J. Neurosci. Methods* **2012**, *211*, 58–65, doi:10.1016/j.jneumeth.2012.07.010.
22. The Student's Guide to Cognitive Neuroscience: 9781138490543: Medicine & Health Science Books @ Amazon.Com Available online: https://www.amazon.com/Students-Guide-Cognitive-Neuroscience/dp/1138490547 (accessed on 25 February 2023).
23. Hebb, D.O. *The Organization of Behavior: A Neuropsychological Theory*; Psychology Press: New York, 2002; ISBN 978-1-4106-1240-3.
24. Langille, J.J.; Brown, R.E. The Synaptic Theory of Memory: A Historical Survey and Reconciliation of Recent Opposition. *Front. Syst. Neurosci.* **2018**, *12*.
25. Berger, H. Über das Elektrenkephalogramm des Menschen. *Arch. Für Psychiatr. Nervenkrankh.* **1929**, *87*, 527–570, doi:10.1007/BF01797193.
26. Buzsáki, G. *Rhythms of the Brain*; Oxford University Press: New York, 2006; ISBN 978-0-19-530106-9.
27. Buzsáki, G.; Draguhn, A. Neuronal Oscillations in Cortical Networks. *Science* **2004**, *304*, 1926–1929, doi:10.1126/science.1099745.
28. Mureşan, R.C.; Jurjuţ, O.F.; Moca, V.V.; Singer, W.; Nikolić, D. The Oscillation Score: An Efficient Method for Estimating Oscillation Strength in Neuronal Activity. *J. Neurophysiol.* **2008**, *99*, 1333–1353, doi:10.1152/jn.00772.2007.
29. Bazhenov, M.; Timofeev, I. Thalamocortical Oscillations. *Scholarpedia* **2006**, *1*, 1319, doi:10.4249/scholarpedia.1319.
30. Worden, M.S.; Foxe, J.J.; Wang, N.; Simpson, G.V. Anticipatory Biasing of Visuospatial Attention Indexed by Retinotopically Specific Alpha-Band Electroencephalography Increases over Occipital Cortex. *J. Neurosci. Off. J. Soc. Neurosci.* **2000**, *20*, RC63, doi:10.1523/JNEUROSCI.20-06-j0002.2000.
31. Rodriguez, E.; George, N.; Lachaux, J.P.; Martinerie, J.; Renault, B.; Varela, F.J. Perception's Shadow: Long-Distance Synchronization of Human Brain Activity. *Nature* **1999**, *397*, 430–433, doi:10.1038/17120.
32. Buzsáki, G.; Wang, X.-J. Mechanisms of Gamma Oscillations. *Annu. Rev. Neurosci.* **2012**, *35*, 203–225, doi:10.1146/annurev-neuro-062111-150444.
33. R, H. What Can Functional Neuroimaging Tell the Experimental Psychologist? *Q. J. Exp. Psychol. A* **2005**, *58*, doi:10.1080/02724980443000502.
34. de Bock, R.; Mackintosh, A.J.; Maier, F.; Borgwardt, S.; Riecher-Rössler, A.; Andreou, C. EEG Microstates as Biomarker for Psychosis in Ultra-High-Risk Patients. *Transl. Psychiatry* **2020**, *10*, 1–9, doi:10.1038/s41398-020-00963-7.

35. Koenig, T.; Lehmann, D.; Merlo, M.C.G.; Kochi, K.; Hell, D.; Koukkou, M. A Deviant EEG Brain Microstate in Acute, Neuroleptic-Naive Schizophrenics at Rest. *Eur. Arch. Psychiatry Clin. Neurosci.* **1999**, *249*, 205–211, doi:10.1007/s004060050088.
36. Lalley, P.M.; Moschovakis, A.K.; Windhorst, U. Electrical Activity of Individual Neurons in Situ: Extra- and Intracellular Recording. In *Modern Techniques in Neuroscience Research*; Windhorst, U., Johansson, H., Eds.; Springer: Berlin, Heidelberg, 1999; pp. 127–172 ISBN 978-3-642-58552-4.
37. Henze, D.A.; Borhegyi, Z.; Csicsvari, J.; Mamiya, A.; Harris, K.D.; Buzsáki, G. Intracellular Features Predicted by Extracellular Recordings in the Hippocampus in Vivo. *J. Neurophysiol.* **2000**, *84*, 390–400, doi:10.1152/jn.2000.84.1.390.
38. Mazzoni, A.; Logothetis, N.K.; Panzeri, S. The Information Content of Local Field Potentials: Experiments and Models 2012.
39. Buzsáki, G. Large-Scale Recording of Neuronal Ensembles. *Nat. Neurosci.* **2004**, *7*, 446–451, doi:10.1038/nn1233.
40. Herreras, O. Local Field Potentials: Myths and Misunderstandings. *Front. Neural Circuits* **2016**, *10*.
41. Sauer, I.; Doerr, C.; Schanze, T. Spike Sorting: The Overlapping Spikes Challenge. *Curr. Dir. Biomed. Eng.* **2015**, *1*, 42–45, doi:10.1515/cdbme-2015-0011.
42. Gray, C.M.; Maldonado, P.E.; Wilson, M.; McNaughton, B. Tetrodes Markedly Improve the Reliability and Yield of Multiple Single-Unit Isolation from Multi-Unit Recordings in Cat Striate Cortex. *J. Neurosci. Methods* **1995**, *63*, 43–54, doi:10.1016/0165-0270(95)00085-2.
43. Mokri, Y.; Salazar, R.F.; Goodell, B.; Baker, J.; Gray, C.M.; Yen, S.-C. Sorting Overlapping Spike Waveforms from Electrode and Tetrode Recordings. *Front. Neuroinformatics* **2017**, *11*.
44. Doerr, C.; Schanze, T. Are Heptodes Better than Tetrodes for Spike Sorting? *IFAC-Pap.* **2015**, *48*, 94–99, doi:10.1016/j.ifacol.2015.10.121.
45. Pedreira, C.; Martinez, J.; Ison, M.J.; Quian Quiroga, R. How Many Neurons Can We See with Current Spike Sorting Algorithms? *J. Neurosci. Methods* **2012**, *211*, 58–65, doi:10.1016/j.jneumeth.2012.07.010.
46. Constant, I.; Sabourdin, N. The EEG Signal: A Window on the Cortical Brain Activity. *Paediatr. Anaesth.* **2012**, *22*, 539–552, doi:10.1111/j.1460-9592.2012.03883.x.
47. Nunez, P.L. and Srinivasan, R. (1981) Electric Fields of the Brain The Neurophysics of EEG. Oxford University Press, Oxford. - References - Scientific Research Publishing Available online: https://www.scirp.org/(S(lz5mqp453edsnp55rrgjct55.))/reference/referencespapers.aspx?referenceid=2560918 (accessed on 2 March 2023).
48. da Cruz, J.R.; Favrod, O.; Roinishvili, M.; Chkonia, E.; Brand, A.; Mohr, C.; Figueiredo, P.; Herzog, M.H. EEG Microstates Are a Candidate Endophenotype for Schizophrenia. *Nat. Commun.* **2020**, *11*, 3089, doi:10.1038/s41467-020-16914-1.
49. Baldi, P. Autoencoders, Unsupervised Learning, and Deep Architectures. In Proceedings of the Proceedings of ICML Workshop on Unsupervised and Transfer Learning; JMLR Workshop and Conference Proceedings, June 27 2012; pp. 37–49.
50. Pinaya, W.; Vieira, S.; Garcia-Dias, R.; Mechelli, A. Autoencoders. In; 2019; pp. 193–208 ISBN 978-0-12-815739-8.
51. Wang, W.; Huang, Y.; Wang, Y.; Wang, L. Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction. In Proceedings of the 2014

IEEE Conference on Computer Vision and Pattern Recognition Workshops; June 2014; pp. 496–503.
52. Hinton, G.E.; Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507, doi:10.1126/science.1127647.
53. Wang, Y.; Yao, H.; Zhao, S. Auto-Encoder Based Dimensionality Reduction. *Neurocomputing* **2016**, *184*, 232–242, doi:10.1016/j.neucom.2015.08.104.
54. Radmanesh, M.; Rezaei, A.A.; Jalili, M.; Hashemi, A.; Goudarzi, M.M. Online Spike Sorting via Deep Contractive Autoencoder. *Neural Netw.* **2022**, doi:10.1016/j.neunet.2022.08.001.
55. Eom, J.; Park, I.Y.; Kim, S.; Jang, H.; Park, S.; Huh, Y.; Hwang, D. Deep-Learned Spike Representations and Sorting via an Ensemble of Auto-Encoders. *Neural Netw.* **2021**, *134*, 131–142, doi:10.1016/j.neunet.2020.11.009.
56. Nguyen, P.-M. Analysis of Feature Learning in Weight-Tied Autoencoders via the Mean Field Lens 2021.
57. Sagheer, A.; Kotb, M. Unsupervised Pre-Training of a Deep LSTM-Based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. *Sci. Rep.* **2019**, *9*, 19038, doi:10.1038/s41598-019-55320-6.
58. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780, doi:10.1162/neco.1997.9.8.1735.
59. Wang, W.; Yang, D.; Chen, F.; Pang, Y.; Huang, S.; Ge, Y. Clustering With Orthogonal AutoEncoder. *IEEE Access* **2019**, *7*, 62421–62432, doi:10.1109/ACCESS.2019.2916030.
60. Rifai, S.; Vincent, P.; Muller, X.; Glorot, X.; Bengio, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. 8.
61. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Advances in Neural Information Processing Systems; Curran Associates, Inc., 2014; Vol. 27.
62. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks 2016.
63. Goodfellow, I. NIPS 2016 Tutorial: Generative Adversarial Networks 2017.
64. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; Illustrated edition.; The MIT Press: Cambridge, Massachusetts, 2016; ISBN 978-0-262-03561-3.
65. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets 2014.
66. Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks 2018.
67. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*; 2nd edition.; Springer: New York, NY, 2016; ISBN 978-0-387-84857-0.
68. Metz, L.; Poole, B.; Pfau, D.; Sohl-Dickstein, J. Unrolled Generative Adversarial Networks 2017.
69. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN 2017.
70. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved Training of Wasserstein GANs 2017.
71. Li, K.; Malik, J. Implicit Maximum Likelihood Estimation 2018.
72. Kohonen, T. Self-Organized Formation of Topologically Correct Feature Maps. *Biol. Cybern.* **1982**, *43*, 59–69, doi:10.1007/BF00337288.

73. Kohonen, T.; Honkela, T. Kohonen Network. *Scholarpedia* **2007**, *2*, 1568, doi:10.4249/scholarpedia.1568.
74. Ponmalai, R.; Kamath, C. *Self-Organizing Maps and Their Applications to Data Analysis*; Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2019;
75. Liu, Y.; Weisberg, R.H.; Mooers, C.N.K. Performance Evaluation of the Self-Organizing Map for Feature Extraction. *J. Geophys. Res. Oceans* **2006**, *111*, doi:10.1029/2005JC003117.
76. Wendel, J.; Buttenfield, B.P. Formalizing Guidelines for Building Meaningful Self-Organizing Maps. 6.
77. Akinduko, A.A.; Mirkes, E.M. Initialization of Self-Organizing Maps: Principal Components Versus Random Initialization. A Case Study 2012.
78. Pölzlbauer, G. Survey and Comparison of Quality Measures for Self-Organizing Maps. *Proc. Fifth Workshop Data Anal. WDA04* **2004**.
79. Samsonova, E.V.; Bäck, T.; Kok, J.N.; IJzerman, A.P. Reliable Hierarchical Clustering with the Self-Organizing Map. In Proceedings of the Advances in Intelligent Data Analysis VI; Famili, A.F., Kok, J.N., Peña, J.M., Siebes, A., Feelders, A., Eds.; Springer: Berlin, Heidelberg, 2005; pp. 385–396.
80. Vesanto, J.; Alhoniemi, E. Clustering of the Self-Organizing Map. *IEEE Trans. Neural Netw.* **2000**, *11*, 586–600, doi:10.1109/72.846731.
81. Zhan, K.; Shi, J.; Wang, H.; Yuange, X.; Li, Q. Computational Mechanisms of Pulse-Coupled Neural Networks: A Comprehensive Review. *Arch. Comput. Methods Eng.* **2017**, *24*, 573–588, doi:10.1007/s11831-016-9182-3.
82. Nie, R.; He, M.; Cao, J.; Zhou, D.; Liang, Z. Pulse Coupled Neural Network Based MRI Image Enhancement Using Classical Visual Receptive Field for Smarter Mobile Healthcare. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, doi:10.1007/s12652-018-1098-3.
83. Harris, M.A.; Van, A.N.; Malik, B.H.; Jabbour, J.M.; Maitland, K.C. A Pulse Coupled Neural Network Segmentation Algorithm for Reflectance Confocal Images of Epithelial Tissue. *PLOS ONE* **2015**, *10*, e0122368, doi:10.1371/journal.pone.0122368.
84. Mureşan, R.C. Pattern Recognition Using Pulse-Coupled Neural Networks and Discrete Fourier Transforms. *Neurocomputing* **2003**, *51*, 487–493, doi:10.1016/S0925-2312(02)00727-0.
85. Lian, J.; Yang, Z.; Liu, J.; Sun, W.; Zheng, L.; Du, X.; Yi, Z.; Shi, B.; Ma, Y. An Overview of Image Segmentation Based on Pulse-Coupled Neural Network. *Arch. Comput. Methods Eng.* **2019**, *28*, doi:10.1007/s11831-019-09381-5.
86. Tao, Z.; Tang, X.; Zhang, B.; Tang, P.; Tan, Y. Image Segmentation Based on PCNN Model. In Proceedings of the 2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP); December 2014; pp. 230–233.
87. Vignesh, T.; Thyagharajan, K.K.; Balaji, L.; Kalaiarasi, G. Implementation and Performance Analysis of Various Models of PCNN for Medical Image Segmentation. In Proceedings of the Intelligent Computing and Innovation on Data Science; Peng, S.-L., Hsieh, S.-Y., Gopalakrishnan, S., Duraisamy, B., Eds.; Springer: Singapore, 2021; pp. 73–84.

88. Lindblad, T.; Kinser, J.M. *Image Processing Using Pulse-Coupled Neural Networks*; Perspectives in Neural Computing; Springer: London, 1998; ISBN 978-3-540-76264-5.
89. Kulkarni, A.; Chong, D.; Batarseh, F.A. 5 - Foundations of Data Imbalance and Solutions for a Data Democracy. In *Data Democracy*; Batarseh, F.A., Yang, R., Eds.; Academic Press, 2020; pp. 83–106 ISBN 978-0-12-818366-3.
90. Weiss, G.M. Mining with Rarity: A Unifying Framework.
91. Wegier, W.; Ksieniewicz, P. Application of Imbalanced Data Classification Quality Metrics as Weighting Methods of the Ensemble Data Stream Classification Algorithms. *Entropy Basel Switz.* **2020**, *22*, E849, doi:10.3390/e22080849.
92. Sun, Y.; Wong, A.K.C.; Kamel, M.S. Classification of Imbalanced Data: A Review. *Int. J. Pattern Recognit. Artif. Intell.* **2009**, *23*, 687–719, doi:10.1142/S0218001409007326.
93. Joshi, M.V.; Kumar, V.; Agarwal, R.C. Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements. In Proceedings of the Proceedings 2001 IEEE International Conference on Data Mining; November 2001; pp. 257–264.
94. Lyons, R. *Understanding Digital Signal Processing*; 3rd edition.; Pearson: Upper Saddle River, NJ, 2010; ISBN 978-0-13-702741-5.
95. Moca, V.V.; Bârzan, H.; Nagy-Dăbâcan, A.; Mureșan, R.C. Time-Frequency Super-Resolution with Superlets. *Nat. Commun.* **2021**, *12*, 337, doi:10.1038/s41467-020-20539-9.
96. Buzsáki, G.; Anastassiou, C.A.; Koch, C. The Origin of Extracellular Fields and Currents — EEG, ECoG, LFP and Spikes. *Nat. Rev. Neurosci.* **2012**, *13*, 407–420, doi:10.1038/nrn3241.
97. Toosi, R.; Akhaee, M.A.; Dehaqani, M.-R.A. An Automatic Spike Sorting Algorithm Based on Adaptive Spike Detection and a Mixture of Skew-t Distributions. *Sci. Rep.* **2021**, *11*, 13925, doi:10.1038/s41598-021-93088-w.
98. Yael, D.; Bar-Gad, I. Filter Based Phase Distortions in Extracellular Spikes. *PLoS ONE* **2017**, *12*, e0174790, doi:10.1371/journal.pone.0174790.
99. Quiroga, R.Q.; Nadasdy, Z.; Ben-Shaul, Y. Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering. *Neural Comput.* **2004**, *16*, 1661–1687, doi:10.1162/089976604774201631.
100. Lewicki, M.S. A Review of Methods for Spike Sorting: The Detection and Classification of Neural Action Potentials. *Netw. Bristol Engl.* **1998**, *9*, R53-78.
101. Quiroga, R.Q. Spike Sorting. *Scholarpedia* **2007**, *2*, 3583, doi:10.4249/scholarpedia.3583.
102. Borsos, Z.; Lemnaru, C.; Potolea, R. Dealing with Overlap and Imbalance: A New Metric and Approach. *Pattern Anal. Appl.* **2018**, *21*, 381–395, doi:10.1007/s10044-016-0583-6.
103. Laboy-Juárez, K.J.; Ahn, S.; Feldman, D.E. A Normalized Template Matching Method for Improving Spike Detection in Extracellular Voltage Recordings. *Sci. Rep.* **2019**, *9*, 12087, doi:10.1038/s41598-019-48456-y.
104. Jolliffe, I.T.; Cadima, J. Principal Component Analysis: A Review and Recent Developments. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2016**, *374*, 20150202, doi:10.1098/rsta.2015.0202.

105. Glaser, E.M.; Marks, W.B. ON-LINE SEPARATION OF INTERLEAVED NEURONAL PULSE SEQUENCES. In *Data Acquisition and Processing in Biology and Medicine*; Enslein, K., Ed.; Pergamon, 1968; pp. 137–156 ISBN 978-0-08-003543-7.
106. Abeles, M.; Goldstein, M.H. Multispike Train Analysis. *Proc. IEEE* **1977**, *65*, 762–773, doi:10.1109/PROC.1977.10559.
107. Adamos, D.A.; Kosmidis, E.K.; Theophilidis, G. Performance Evaluation of PCA-Based Spike Sorting Algorithms. *Comput. Methods Programs Biomed.* **2008**, *91*, 232–244, doi:10.1016/j.cmpb.2008.04.011.
108. Hyvärinen, A. Independent Component Analysis: Recent Advances. *Philos. Transact. A Math. Phys. Eng. Sci.* **2013**, *371*, 20110534, doi:10.1098/rsta.2011.0534.
109. Lopes, M.V.; Aguiar, E.; Santana, E.; Santana, E.; Barros, A.K. ICA Feature Extraction for Spike Sorting of Single-Channel Records. In Proceedings of the 2013 ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC); February 2013; pp. 1–5.
110. Tiganj, Z.; Mboup, M. Neural Spike Sorting Using Iterative ICA and a Deflation-Based Approach. *J. Neural Eng.* **2012**, *9*, 066002, doi:10.1088/1741-2560/9/6/066002.
111. Tharwat, A.; Gaber, T.; Ibrahim, A.; Hassanien, A.E. Linear Discriminant Analysis: A Detailed Tutorial. *Ai Commun.* **2017**, *30*, 169-190, doi:10.3233/AIC-170729.
112. Tenenbaum, J.B.; de Silva, V.; Langford, J.C. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* **2000**, *290*, 2319–2323, doi:10.1126/science.290.5500.2319.
113. Zhou, H.; Wang, F.; Tao, P. T-Distributed Stochastic Neighbor Embedding Method with the Least Information Loss for Macromolecular Simulations. *J. Chem. Theory Comput.* **2018**, *14*, 5499–5510, doi:10.1021/acs.jctc.8b00652.
114. Dasgupta, S. The Hardness of K-Means Clustering.
115. Veerabhadrappa, R.; Ul Hassan, M.; Zhang, J.; Bhatti, A. Compatibility Evaluation of Clustering Algorithms for Contemporary Extracellular Neural Spike Sorting. *Front. Syst. Neurosci.* **2020**, *14*.
116. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. *Proc. Fifth Berkeley Symp. Math. Stat. Probab. Vol. 1 Stat.* **1967**, *5.1*, 281–298.
117. Salganicoff, M.; Sarna, M.; Sax, L.; Gerstein, G.L. Unsupervised Waveform Classification for Multi-Neuron Recordings: A Real-Time, Software-Based System. I. Algorithms and Implementation. *J. Neurosci. Methods* **1988**, *25*, 181–187, doi:10.1016/0165-0270(88)90132-x.
118. Caro-Martín, C.R.; Delgado-García, J.M.; Gruart, A.; Sánchez-Campusano, R. Spike Sorting Based on Shape, Phase, and Distribution Features, and K-TOPS Clustering with Validity and Error Indices. *Sci. Rep.* **2018**, *8*, 17796, doi:10.1038/s41598-018-35491-4.
119. Pachitariu, M.; Steinmetz, N.; Kadir, S.; Carandini, M.; D, H.K. Kilosort: Realtime Spike-Sorting for Extracellular Electrophysiology with Hundreds of Channels 2016, 061481.
120. Bezdek, J.C.; Ehrlich, R.; Full, W. FCM: The Fuzzy c-Means Clustering Algorithm. *Comput. Geosci.* **1984**, *10*, 191–203, doi:10.1016/0098-3004(84)90020-7.
121. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Proceedings

of the Second International Conference on Knowledge Discovery and Data Mining; AAAI Press: Portland, Oregon, August 2 1996; pp. 226–231.
122. Campello, R.J.G.B.; Moulavi, D.; Sander, J. Density-Based Clustering Based on Hierarchical Density Estimates. In Proceedings of the Advances in Knowledge Discovery and Data Mining; Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G., Eds.; Springer: Berlin, Heidelberg, 2013; pp. 160–172.
123. Ackermann, M.R.; Blömer, J.; Kuntze, D.; Sohler, C. Analysis of Agglomerative Clustering. *Algorithmica* **2014**, *69*, 184–215, doi:10.1007/s00453-012-9717-4.
124. Carreira-Perpiñán, M.Á. A Review of Mean-Shift Algorithms for Clustering 2015.
125. Cheng, Y. Mean Shift, Mode Seeking, and Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *17*, 790–799, doi:10.1109/34.400568.
126. Magland, J.F.; Barnett, A.H. Unimodal Clustering Using Isotonic Regression: ISO-SPLIT 2016.
127. Shi, J.; Malik, J. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 888–905, doi:10.1109/34.868688.
128. von Luxburg, U. A Tutorial on Spectral Clustering 2007.
129. Steinley, D. Properties of the Hubert-Arable Adjusted Rand Index. *Psychol. Methods* **2004**, *9*, 386–396, doi:10.1037/1082-989X.9.3.386.
130. Hubert, L.; Arabie, P. Comparing Partitions. *J. Classif.* **1985**, *2*, 193–218, doi:10.1007/BF01908075.
131. Vinh, N.X.; Epps, J.; Bailey, J. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. 18.
132. Santos, J.; Embrechts, M. On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification.; September 14 2009; Vol. 2009, pp. 175–184.
133. Vinh, N.X.; Epps, J.; Bailey, J. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. 18.
134. Strehl, A.; Ghosh, J. Cluster Ensembles --- A Knowledge Reuse Framework for Combining Multiple Partitions. *J. Mach. Learn. Res.* **2002**, *3*, 583–617.
135. Vinh, N.X.; Epps, J.; Bailey, J. Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary? In Proceedings of the Proceedings of the 26th Annual International Conference on Machine Learning; Association for Computing Machinery: New York, NY, USA, June 14 2009; pp. 1073–1080.
136. Lazarenko, D.; Bonald, T. Pairwise Adjusted Mutual Information 2021.
137. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Illustrated edition.; Cambridge University Press: New York, 2008; ISBN 978-0-521-86571-5.
138. Rendón, E.; Abundez, I.; Arizmendi, A.; Quiroz, E.M. Internal versus External Cluster Validation Indexes. **2011**, *5*, 8.
139. Fowlkes, E.B.; Mallows, C.L. A Method for Comparing Two Hierarchical Clusterings. *J. Am. Stat. Assoc.* **1983**, *78*, 553–569, doi:10.2307/2288117.
140. Rosenberg, A.; Hirschberg, J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure.; January 1 2007; pp. 410–420.
141. Caliński, T.; JA, H. A Dendrite Method for Cluster Analysis. *Commun. Stat. - Theory Methods* **1974**, *3*, 1–27, doi:10.1080/03610927408827101.
142. Halkidi, M.; Batistakis, Y.; Vazirgiannis, M. On Clustering Validation Techniques. *J. Intell. Inf. Syst.* **2001**, *17*, 107–145, doi:10.1023/A:1012801612483.

143. Davies, D.L.; Bouldin, D.W. A Cluster Separation Measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *PAMI-1*, 224–227, doi:10.1109/TPAMI.1979.4766909.
144. Rousseeuw, P.J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65, doi:10.1016/0377-0427(87)90125-7.
145. Yger, P.; Spampinato, G.L.; Esposito, E.; Lefebvre, B.; Deny, S.; Gardella, C.; Stimberg, M.; Jetter, F.; Zeck, G.; Picaud, S.; et al. A Spike Sorting Toolbox for up to Thousands of Electrodes Validated with Ground Truth Recordings in Vitro and in Vivo. *eLife* **2018**, *7*, e34518, doi:10.7554/eLife.34518.
146. Bakkum, D.; Radivojevic, M.; Frey, U.; Franke, F.; Hierlemann, A.; Takahashi, H. Parameters for Burst Detection. *Front. Comput. Neurosci.* **2014**, *7*.
147. Hennig, M.H.; Grady, J.; van Coppenhagen, J.; Sernagor, E. Age-Dependent Homeostatic Plasticity of GABAergic Signaling in Developing Retinal Networks. *J. Neurosci. Off. J. Soc. Neurosci.* **2011**, *31*, 12159–12164, doi:10.1523/JNEUROSCI.3112-11.2011.
148. Kirillov, A. NeuroExplorer Manual.
149. Kapucu, F.E.; Tanskanen, J.; Mikkonen, J.; Ylä-Outinen, L.; Narkilahti, S.; Hyttinen, J. Burst Analysis Tool for Developing Neuronal Networks Exhibiting Highly Varying Action Potential Dynamics. *Front. Comput. Neurosci.* **2012**, *6*.
150. Gourévitch, B.; Eggermont, J.J. A Nonparametric Approach for Detection of Bursts in Spike Trains. *J. Neurosci. Methods* **2007**, *160*, 349–358, doi:10.1016/j.jneumeth.2006.09.024.
151. Legéndy, C.R.; Salcman, M. Bursts and Recurrences of Bursts in the Spike Trains of Spontaneously Active Striate Cortex Neurons. *J. Neurophysiol.* **1985**, *53*, 926–939, doi:10.1152/jn.1985.53.4.926.
152. Jurjuţ, O.F.; Nikolić, D.; Pipa, G.; Singer, W.; Metzler, D.; Mureşan, R.C. A Color-Based Visualization Technique for Multielectrode Spike Trains. *J. Neurophysiol.* **2009**, *102*, 3766–3778, doi:10.1152/jn.00758.2009.
153. Neymotin, S.A.; Tal, I.; Barczak, A.; O'Connell, M.N.; McGinnis, T.; Markowitz, N.; Espinal, E.; Griffith, E.; Anwar, H.; Dura-Bernal, S.; et al. Detecting Spontaneous Neural Oscillation Events in Primate Auditory Cortex. *eNeuro* **2022**, *9*, ENEURO.0281-21.2022, doi:10.1523/ENEURO.0281-21.2022.
154. Subakan, C.; Smaragdis, P. Generative Adversarial Source Separation. *ArXiv171010779 Cs Stat* **2017**.
155. Ardelean, E.-R.; Coporîie, A.; Ichim, A.-M.; Dînşoreanu, M.; Mureşan, R.C. A Study of Autoencoders as a Feature Extraction Technique for Spike Sorting. *PLOS ONE* **2023**, *18*, e0282810, doi:10.1371/journal.pone.0282810.
156. Dwork, C.; Kumar, R.; Naor, M.; Sivakumar, D. Rank Aggregation Methods for the Web. In Proceedings of the Proceedings of the 10th international conference on World Wide Web; Association for Computing Machinery: New York, NY, USA, April 1 2001; pp. 613–622.
157. Mureşan, D.B.; Ciure, R.-D.; Ardelean, E.R.; Moca, V.V.; Mureşan, R.C.; Dînş, M. Spike Sorting Using Superlets: Evaluation of a Novel Feature Space for the Discrimination of Neuronal Spikes. In Proceedings of the 2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP); September 2022; pp. 229–235.

158. Ardelean, E.-R.; Terec, R.-D.; Marieş, C.-M.; Moca, V.V.; Mureşan, R.C.; Dînşoreanu, M. Spike Sorting Using Superlets: Identifying Feature Importance through Perturbation. In Proceedings of the 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP); October 2023; pp. 357–362.
159. İncetaş, M.O. Image Interpolation Based on Spiking Neural Network Model. *Appl. Sci.* **2023**, *13*, 2438, doi:10.3390/app13042438.
160. Schober, P.; Boer, C.; Schwarte, L.A. Correlation Coefficients: Appropriate Use and Interpretation. *Anesth. Analg.* **2018**, *126*, 1763, doi:10.1213/ANE.0000000000002864.
161. Mitchell, J.F.; Sundberg, K.A.; Reynolds, J.H. Differential Attention-Dependent Response Modulation across Cell Classes in Macaque Visual Area V4. *Neuron* **2007**, *55*, 131–141, doi:10.1016/j.neuron.2007.06.018.
162. Ardelean, E.-R.; Grosu, G.F.; Terebeş, R.; Dînşoreanu, M. Exploiting the Self-Organizing Map for Spike Sorting. In Proceedings of the 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP); October 2023; pp. 363–369.
163. Ardelean, E.-R.; Stanciu, A.; Dinsoreanu, M.; Potolea, R.; Lemnaru, C.; Moca, V.V. Space Breakdown Method A New Approach for Density-Based Clustering. In Proceedings of the 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP); September 2019; pp. 419–425.
164. Ardelean, E.-R.; Ichim, A.-M.; Dînşoreanu, M.; Mureşan, R.C. Improved Space Breakdown Method – A Robust Clustering Technique for Spike Sorting. *Front. Comput. Neurosci.* **2023**, *17*.
165. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
166. Zeldenrust, F.; Wadman, W.J.; Englitz, B. Neural Coding With Bursts—Current State and Future Perspectives. *Front. Comput. Neurosci.* **2018**, *12*.
167. Ardelean, A.-I.; Ardelean, E.-R.; Moca, V.V.; Mureşan, R.C.; Dînşoreanu, M. Burst Detection in Neuronal Activity. In Proceedings of the 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP); October 2023; pp. 349–356.
168. Cotterill, E.; Eglen, S.J. Burst Detection Methods. *Adv. Neurobiol.* **2019**, *22*, 185–206, doi:10.1007/978-3-030-11135-9_8.
169. Moca, V.V.; Ţincaş, I.; Melloni, L.; Mureşan, R.C. Visual Exploration and Object Recognition by Lattice Deformation. *PLOS ONE* **2011**, *6*, e22831, doi:10.1371/journal.pone.0022831.
170. Moisa, O.M.; Pop, I.; Ardelean, E.-R.; Moca, V.V.; Mureşan, R.C.; Dînşoreanu, M. Symbolic Analysis Based Pipeline for EEG Data. In Proceedings of the 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP); October 2023; pp. 371–378.
171. Wang, X.-J. Neurophysiological and Computational Principles of Cortical Rhythms in Cognition. *Physiol. Rev.* **2010**, *90*, 1195–1268, doi:10.1152/physrev.00035.2008.
172. Bârzan, H.; Ichim, A.-M.; Moca, V.V.; Mureşan, R.C. Time-Frequency Representations of Brain Oscillations: Which One Is Better? *Front. Neuroinformatics* **2022**, *16*.

173. Tal, I.; Neymotin, S.; Bickel, S.; Lakatos, P.; Schroeder, C.E. Oscillatory Bursting as a Mechanism for Temporal Coupling and Information Coding. *Front. Comput. Neurosci.* **2020**, *14*.
174. Berger, H. Ueber Das Elektrenkephalogramm Des Menschen. *J. Für Psychol. Neurol.* **1930**, *40*, 160–179.
175. Ardelean, E.-R.; Bârzan, H.; Ichim, A.-M.; Mureşan, R.C.; Moca, V.V. Sharp Detection of Oscillation Packets in Rich Time-Frequency Representations of Neural Signals. *Front. Hum. Neurosci.* **2023**, *17*.
176. Rahi, J.S.; Logan, S.; Borja, M.C.; Timms, C.; Russell-Eggitt, I.; Taylor, D. Prediction of Improved Vision in the Amblyopic Eye after Visual Loss in the Non-Amblyopic Eye. *Lancet Lond. Engl.* **2002**, *360*, 621–622, doi:10.1016/S0140-6736(02)09775-1.
177. Brin, T.A.; Xu, Z.; Zhou, Y.; Feng, L.; Li, J.; Thompson, B. Amblyopia Is Associated with Impaired Balance in 3–6-Year-Old Children in China. *Front. Neurosci.* **2022**, *16*.
178. Fu, Z.; Hong, H.; Su, Z.; Lou, B.; Pan, C.-W.; Liu, H. Global Prevalence of Amblyopia and Disease Burden Projections through 2040: A Systematic Review and Meta-Analysis. *Br. J. Ophthalmol.* **2020**, *104*, 1164–1170, doi:10.1136/bjophthalmol-2019-314759.
179. Leek, M.R. Adaptive Procedures in Psychophysical Research. *Percept. Psychophys.* **2001**, *63*, 1279–1292, doi:10.3758/bf03194543.
180. Watson, A.B.; Pelli, D.G. Quest: A Bayesian Adaptive Psychometric Method. *Percept. Psychophys.* **1983**, *33*, 113–120, doi:10.3758/BF03202828.
181. Watson, A.B. QUEST+: A General Multidimensional Bayesian Adaptive Psychometric Method. *J. Vis.* **2017**, *17*, 10, doi:10.1167/17.3.10.
182. Farahbakhsh, M.; Dekker, T.M.; Jones, P.R. Psychophysics with Children: Evaluating the Use of Maximum Likelihood Estimators in Children Aged 4–15 Years (QUEST+). *J. Vis.* **2019**, *19*, 22, doi:10.1167/19.6.22.
183. Jones, P.R.; Kalwarowsky, S.; Braddick, O.J.; Atkinson, J.; Nardini, M. Optimizing the Rapid Measurement of Detection Thresholds in Infants. *J. Vis.* **2015**, *15*, 2, doi:10.1167/15.11.2.
184. Mihaylova, M.S.; Hristov, I.; Racheva, K.; Totev, T.; Mitov, D. Effect of Extending Grating Length and Width on Human Visually Evoked Potentials. *Acta Neurobiol. Exp. (Warsz.)* **2015**, *75*, 293–304.
185. Rosén, R.; Lundström, L.; Venkataraman, A.P.; Winter, S.; Unsbo, P. Quick Contrast Sensitivity Measurements in the Periphery. *J. Vis.* **2014**, *14*, 3, doi:10.1167/14.8.3.
186. Fero, K.; Yokogawa, T.; Burgess, H.A. The Behavioral Repertoire of Larval Zebrafish. In *Zebrafish Models in Neurobehavioral Research*; Kalueff, A.V., Cachat, J.M., Eds.; Neuromethods; Humana Press: Totowa, NJ, 2011; pp. 249–291 ISBN 978-1-60761-922-2.
187. Joo, W.; Vivian, M.D.; Graham, B.J.; Soucy, E.R.; Thyme, S.B. A Customizable Low-Cost System for Massively Parallel Zebrafish Behavioral Phenotyping. *Front. Behav. Neurosci.* **2021**, *14*.
188. Sun, M.; Li, W.; Jiao, Z.; Zhao, X. A Multi-Target Tracking Platform for Zebrafish Based on Deep Neural Network. In Proceedings of the 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER); July 2019; pp. 637–642.

189. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition; June 2009; pp. 248–255.
190. Huh, M.; Agrawal, P.; Efros, A.A. What Makes ImageNet Good for Transfer Learning? *ArXiv160808614 Cs* **2016**.
191. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *ArXiv151203385 Cs* **2015**.
192. Kebir, A.; Taibi, M.; Serradilla, F. Compressed VGG16 Auto-Encoder for Road Segmentation from Aerial Images with Few Data Training Available online: https://www.semanticscholar.org/paper/Compressed-VGG16-Auto-Encoder-for-Road-Segmentation-Kebir-Taibi/a368f5fe2745e4b854197d0350a8316f82518312 (accessed on 31 January 2022).
193. Zhang, R.; Du, L.; Xiao, Q.; Liu, J. Comparison of Backbones for Semantic Segmentation Network.; 2020.
194. Sultana, F.; Sufian, A.; Dutta, P. Evolution of Image Segmentation Using Deep Convolutional Neural Network: A Survey. *Knowl.-Based Syst.* **2020**, *201–202*, 106062, doi:10.1016/j.knosys.2020.106062.
195. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv150504597 Cs* **2015**.
196. Shorten, C.; Khoshgoftaar, T.M. A Survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60, doi:10.1186/s40537-019-0197-0.
197. Mathis, A.; Mamidanna, P.; Cury, K.M.; Abe, T.; Murthy, V.N.; Mathis, M.W.; Bethge, M. DeepLabCut: Markerless Pose Estimation of User-Defined Body Parts with Deep Learning. *Nat. Neurosci.* **2018**, *21*, 1281–1289, doi:10.1038/s41593-018-0209-y.
198. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv190511946 Cs Stat* **2020**.
199. Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. A Comprehensive Survey on Support Vector Machine Classification: Applications, Challenges and Trends. *Neurocomputing* **2020**, *408*, 189–215, doi:10.1016/j.neucom.2019.10.118.
200. Schroff, F.; Criminisi, A.; Zisserman, A. Object Class Segmentation Using Random Forests. In Proceedings of the Procedings of the British Machine Vision Conference 2008; British Machine Vision Association: Leeds, 2008; p. 54.1-54.10.

APPENDICES

List of figures

Figure 2.1 - The human brain from a lateral perspective presenting the coarse grouping into lobes. The image was taken from Wikimedia Commons (public domain) and adapted. 6
Figure 2.2 - One of Cajal's drawings of neurons using the Golgi staining technique from 1899 [17,18]..... 8
Figure 2.3 - The neuron and its main parts. The cell body, or soma, contains the organelles of the cells, while the axon and dendrites connect the neuron, by forming synapses, to other neurons. The image was taken from Wikimedia Commons (public domain) and adapted. 9
Figure 2.4 - The action potential and its structure. The time course of the membrane potential during the spiking of a neuron and various terminology used that determine its parts. The image was taken and adapted from Wikimedia Commons (public domain). The left and right panels present synonyms that are used within this domain. 10
Figure 2.5 - Particularities of action potentials on different types of recordings (left, extracellular from TINS and right, intracellular from [21]). 11
Figure 2.6 - The synapse and its inner workings. The image was taken and adapted from Wikimedia Commons (public domain)..... 13
Figure 2.7 - Neuron circuitry. The image was taken from Wikimedia Commons (public domain) and adapted. 14
Figure 2.8 - Looping in neuron circuitry. The image was taken from Wikimedia Commons (public domain) and adapted. 15
Figure 2.9 - Raw signal filtering resulting in Local Field Potentials and/or Spiking Activity. 20
Figure 2.10 - A simple diagram of the perceptron, where *w* indicates the weights, *b* the bias, and *f* the activation function. 24
Figure 2.11 - A simple example of an autoencoder architecture. 25
Figure 2.12 - A simple diagram of the Generator model inputs and outputs. 27
Figure 2.13 - A simple diagram of the Discriminator model inputs and outputs. 28
Figure 2.14 - The interaction between the Generator and Discriminator models creating the GAN. 28
Figure 2.15 - The flow of the PCNN model. 34
Figure 2.16 - Time vs Frequency Domain. The red trace is the sum of the light blue sinusoidals. The image was taken from Wikimedia Commons (public domain). 37
Figure 2.17 – The spike sorting pipeline [2]..... 41
Figure 2.18 – An example of feature extraction process. Spikes produced by three distinct neurons produce separable features in the PCA space. 45
Figure 2.19 - An illustration of clustering on simple data. 47
Figure 3.1 - (Left) Synthetic spikes of one neuron model show that even though some variability exists, they are very similar. (Right) Each synthetic spike is from taken

randomly from all spikes of a different neuron model showing that there is a high amount of variability between the spikes of different neuron models. 63
Figure 3.2 - The influence of spike alignment on feature extraction through PCA on synthetic data. Panel A shows the unaligned spikes on the left and the result of feature extraction on the right, the colours represent the ground truth, it is clearly visible that unaligned spikes created a feature space in which the white cluster is fragmented. Panel B shows the aligned spikes on left and the result of PCA on the right, in this case clusters are not fragmented. 64
Figure 3.3 - The proposed architecture. At the top a GAN component is shown having as input a random vector, called Hash, and an output, the generated signal. The source signal separation architecture uses two GANs, one that generates each type of source. In this case, the two sources are spikes and noise. This architecture attempts to generate by iterative updates, the two sources that estimate as well as possible the original signal. 67
Figure 3.4 - Loss and accuracy of GAN training. 68
Figure 3.5 - Signals produced by the Generator model trained on the FFT signal of spikes. 69
Figure 3.6 - The inverse Fourier Transform of the signals produced by the Generator model shown in Figure 3.5. 69
Figure 3.7 - A spike and its Fourier transform. 70
Figure 3.8 - Separation of a signal into spike and noise compared with the original signal with no preprocessing before insertion in the training of the GAN. 70
Figure 3.9 - Separation of a signal into spike and noise compared with the original signal when the signal was preprocessed using Fourier Transform before insertion in the training of the GAN. 70
Figure 3.10 - Generation of drowned signals by changing to the signal-to-noise ratio....71
Figure 3.11 - Performance evaluations of spike detection using GANs with a specifically designed dataset of half spikes and half noise on training data. Each plot shows the performance from the perspective of a different metric. Lines in plots indicate the method used to detect spikes, whether it is the proposed approach, or the threshold based on the standard deviation. The x-axis indicates the SNR applied to the dataset. 72
Figure 3.12 - Performance evaluations on the training data of spike detection using GANs on a synthetic dataset with a low percentage of spike windows in comparison with noise examples. 73
Figure 3.13 - Performance evaluations on the training data of Spike Detection using GANs on a synthetic dataset with a low percentage of spike windows in comparison with noise, with an increased number of training iterations. 73
Figure 3.14 - Training data of the classifier: (A) extracellular spikes, (B) noise. The spikes and the noise segments were extracted from the M022 real dataset. 75
Figure 3.15 - Loss and accuracy of the classifier during training on the M022 real dataset. 75
Figure 3.16 - The left side shows histograms of the distributions of spike amplitudes, while the right side shows the spike shapes. The top represents spikes found by the amplitude thresholding approach, the middle shows spikes found by the classifier approach and the bottom shows the sum of the two approaches. 76

Figure 3.17 - PCA analysis of spikes found by amplitude thresholding (red), new spikes found by classifier (blue), and noise (white). 77
Figure 3.18 - General architecture of an AE. 78
Figure 3.19 - Feature extraction of the Sim4 dataset containing 5 clusters, the colour coding shown represents the ground truth. 80
Figure 3.20 - Evaluation of performance for all approaches applied on all 95 synthetic simulations with regard to each performance metric. 81
Figure 3.21 - Feature extraction methods applied on the M045-C17 real dataset; colours indicate the labels obtained through K-Means. 83
Figure 3.22 - Feature extraction using the Superlet Transform and PCA on the Sim8 synthetic dataset with various values for the *cycles* parameter (c=1.5, left and c=3, right). 84
Figure 3.23 - Feature extraction using the Superlet Transform and PCA on the Sim8 synthetic dataset with various values for the *order* parameter (o=5, left and o=15, right). 85
Figure 3.24 - Spectrogram of a cluster from a simulated dataset (Sim8) with varying values for the parameters. The left image shows the best precision with *c*=1.5 and *o*=2, the minimum values. The middle image has a high *order* value resulting in a high frequency precision, while the right image has a high *cycle* value resulting in a high temporal precision. 85
Figure 3.25 - Evolution of loss and performance metrics (accuracy and F1 score, they are overlapped) during training of the Sim79 dataset with 21 clusters. 87
Figure 3.26 - The flow of the correlated feature set perturbation pipeline. 89
Figure 3.27 - Average spikes and their corresponding spectrograms for 3 different clusters (neurons) of a M045-C4. 89
Figure 3.28 - An example of a correlation matrix obtained from M045-C4 real dataset after the downsampling of spectrograms through bicubic interpolation. 90
Figure 3.29 - Perturbation matrix showing the effect of perturbation on the learning of a neural network on each feature. 90
Figure 3.30 - Bicubic interpolation for upsampling to obtain original size. 91
Figure 3.31 - Superlet parameterization impact on a single cluster of spikes from the M045-C4 real dataset, where the heatmap values represent the power values of the spectrogram. 92
Figure 3.32 - Superlet parameterization impact on the drop in performance, where the values of the heatmap represent the change in performance given by the perturbation of each characteristic at those positions [158] 92
Figure 3.33 - Plot of the quantization and topographic errors during the SOM training across 1000 iterations on the Sim4 synthetic dataset. 94
Figure 3.34 - (Left) Result of SOM on the Sim4 synthetic labelled dataset, each colour represents a different cluster in the ground truth. (Right) Plot of distance matrix obtained by the SOM from the training on the Sim4 dataset. 95
Figure 3.35 - (Left) The resulted U-matrix from SOM training based on the Sim11 synthetic dataset plotted as an image of higher and lower distances denoted by pixel intensity levels, marking the spatial structure of the clusters given by the organisation of

the SOM network. (Right) A plot of the resulted U-matrix obtained from the SOM training on the Sim11 synthetic dataset, with the addition of the corresponding ground truth as overlaid annotations over the U-matrix. Each annotation (the individual combinations of colour and shape) represents a distinct cluster. 95
Figure 3.36 - Example of resulted U-matrix local maxima-based segmentation on the same data as in Figure 2. (Left) The computed local maxima with binary closing overlaid as red pixels over the afferent U-matrix. (Right) The label annotations for the resulting clusters determined by the U-matrix local maxima separation method. 96
Figure 3.37 - (Left) The ground truth clusters labels mapped to the corresponding data projected in a 2D space by the PCA method. (Right) The resulted clusters' labels mapped to the same corresponding data projected in the PCA-based 2D space. 97
Figure 3.38 - A comparison of all methods on Sim11, on the left side, each clustering method with PCA is presented, while on the right side with Isomap, the result of the proposed approach can be found as the last panel. 98
Figure 3.39 - A comparison of the chunkification step of SBM and ISBM, the space presented is already normalised as the first step of the algorithms in the [0,5) interval. Every cell in the grid shown contains a number in the lower-left representing the value of the chunk in both SBM and ISBM chunkification. The difference appears in the final structure, while SBM saves the whole array, ISBM only retains the values that are encircled, where circles represent the nodes and the lines the edges of the final graph structure. 102
Figure 3.40 - (a). Simulation 4 (Sim4), a synthetic dataset with its ground truth (b). The clustering of K-Means on the dataset (c). The clustering of DBSCAN on the dataset (d). The result of MeanShift on the Sim4 dataset (e). The clustering of Agglomerative Clustering on the Sim4 dataset (g). The clustering of HDBSCAN on the dataset (f). The clustering of FCM on the dataset (h). The clustering of ISO-SPLIT on the dataset (i). The clustering of the original version of SBM on the dataset (j). The clustering of the improved SBM (ISBM) on the dataset. 105
Figure 3.41 – The M045-C1 real dataset that has been cleaned and filtered and recorded from the brain of a mouse (a). Ground truth generated through K-Means (b). The clustering of K-Means on the dataset (c). The clustering of DBSCAN on the dataset (d). The result of MeanShift on the Sim4 dataset (e). The clustering of Agglomerative Clustering on the Sim4 dataset (g). The clustering of HDBSCAN on the dataset (f). The clustering of FCM on the dataset (h). The clustering of ISO-SPLIT on the dataset (i). The clustering of the original version of SBM on the dataset (j). The clustering of the improved SBM (ISBM) on the dataset. 106
Figure 3.42 - Types of bursts, each subplot shows a different type of simulated data [168] indicated by the label on the left of each subplot. The data is composed of timestamps and as such 1s indicate activity and 0s no activity. 110
Figure 3.43 - Spectrogram of a burst candidate, extracted from the M029-C23 real dataset by the proposed method, containing three sub-spikes. 112
Figure 3.44 - True positive percentage (indicated on the y-axis labels), each subplot shows the evaluation on the percentage of true positives (ranging from 0 to 1) found in a

specified data type (indicated by the subtitle) by each method (indicated on x-axis labels) on the benchmark synthetic burst data. 114
Figure 3.45 - False positive percentage (indicated on the y-axis labels), each subplot shows the evaluation on the percentage of false positives (ranging from 0 to 1) found in a specified data type (indicated by the subtitle) by each method (indicated on x-axis labels) on the benchmark synthetic burst data. 114
Figure 3.46 - Burst candidate extracted from the M029-C23 real dataset by the MI [148] burst detection method (left) and the proposed method (right), the blue line represents the signal, the red dots show the peaks of spikes, and the dashed red line represents the amplitude threshold used in the detection of spikes. 115
Figure 3.47 - The left panel shows a comparison between the correlation PDF between intra-burst sub-spikes of a single channel versus the burst sub-spikes and tonic spiking activity for the proposed method, the right panel shows the same analysis for the MI [148] method with the suggested parametrization. 116
Figure 3.48 - The left panel shows the correlation PDF of intra-burst sub-spikes of a single channel for each of the methods, the right panel shows the correlation PDF between burst sub-spikes and the tonic spiking activity of a single channel. 116
Figure 3.49 - The left panel shows a comparison between the correlation PDF between burst sub-spikes of a single channel versus the burst sub-spikes of distant channels for the proposed method, the right panel shows the same analysis for the MI [148] method with the suggested parametrization. 117
Figure 3.50 - The left panel shows the correlation PDF of all burst sub-spikes against each other of a single channel for each of the methods, the right panel shows the correlation PDF between burst sub-spikes of two different distant channels. 117
Figure 4.1 - Several ways of visualisation of the distance map: scatter plot (left), voxel plot (middle), and volume slice plot (right). 121
Figure 4.2 - Scatter plot with the clusters obtained from the clustering of the EEG data. 122
Figure 4.3 - The process of colour assignment of a sample. 122
Figure 4.4 - Left-alignment of colour sequences of the "Certain" (left), "Uncertain" (middle) and "Nothing" (right) responses of subjects. 123
Figure 4.5 - Left-alignment of colour sequences of the “Certain” response of subjects. 124
Figure 4.6 - Windowed colour sequences of the “Certain” response of subjects with left-alignment (on the left) and with right-alignment (on the right). 124
Figure 4.7 - Example of PSI with left-alignment for subject’s response: “Certain”, highlighting the most meaningful patterns. 126
Figure 4.8 - PTA for a subset of trial in the subgroup “Certain”, pattern (0.1, 0.9, 0.9) on channel D23 (left) and the Average PTA for subgroup ‘Certain’, pattern: (0.1, 0.9, 0.9) on channel D23 (right). 127
Figure 4.9 - PSTH of pattern (0.1, 0.0, 0.0) for subject’s response: “Uncertain”. 127
Figure 4.10 - The segmentation algorithms and the ROI matching metric. **(A)** The TFBM algorithm begins from the peaks of the TFR, and it expands until a conflict point (grey circle) or a point that satisfies the border condition (grey squares) is reached. Conflicts are resolved by assigning the conflicting points to one of the regions of interest (ROIs) or

by merging the conflicting ROIs into a larger ROI, represented by dotted lines. Peaks below the threshold (thin grey line) are not considered as seed points for ROI expansion. **(B)** TFPF algorithm, which slices through the TFR from the highest power down to the threshold (grey line). ROIs, represented by dotted lines, expand as the slice level is lowered. When ROIs merge (thick grey line), the smaller peak is merged into the larger one. The algorithm stops at the threshold, and points below the threshold are not considered. **(C)** the matching metric between two ROIs is illustrated in the left pane. The right pane shows the best matching ROI (green) overlapping with the target ROI (red). The grey ROI has a lower match value to the target than the best matching ROI. 131
Figure 4.11 - Thresholding and rejection of low amplitude peaks of the TFR of a single-trial mouse LFP. **(A)** All local maxima of the TFR. **(B)** Only local maxima that are above the threshold, which is determined according to the method described in panel C. All power values below the threshold have been set to 0 to emphasise the low threshold while retaining the dynamic range of the TFR. **(C)** The power distribution in the TFR. The top panel shows the highly skewed distribution of the power values of all local maxima. The cumulative distribution shown in the bottom panel is used to set the threshold, covering 90% of all power values, which corresponds to less than 10% of the maximum power. The power values in the TFR have been scaled to the interval 0-100 for easier interpretation. 131
Figure 4.12 – A summarised comparation between the SBM and TFBM algorithms. 133
Figure 4.13 - Comparison of the segmentation provided by each algorithm. The algorithms are shown by columns: TFBM (left), TFPF (middle), and OEvents (right) on various single-trial TFRs computed with superlets on each row. The boundaries of the detected ROIs are depicted in black, and the bounding boxes are shown in red. White dots indicate the local maxima, while grey dots indicate the sub-peaks. **(A)** The TFR corresponds to a series of generated atoms in the gamma frequency range (20-55Hz) embedded in uniform noise. **(B)** The TFR corresponds to a wide frequency range (0-100Hz) from a single-trial LFP recording in the mouse visual cortex, stimulated with drifting gratings. **(C)** The TFR corresponds to a single-trial EEG with a rich spectrum covering the alpha, beta, and low gamma frequency bands. **(D)** Shows the segmentations of a zoomed-in area of the data from panel C. 135
Figure 4.14 - A comparison of the segmentation of the TFBM, TFPF, and OEvents algorithms in terms of detection performance by SNR. Stars indicate significance levels, highlighting statistical differences. **(A)** The match between the ground truth (a generated atom introduced randomly in the data) and the detected packets is evaluated using the matching error of rectangular bounding boxes. The top panel displays the bounding-box detection errors, which improve with increasing SNR. The second panel shows the time errors, while the third panel presents the frequency errors. The bottom panel illustrates the percentage of missed atoms. **(B)** These panels have the same arrangement as those of panel A. Here, TFBM and TFPF are evaluated using fine-grained regions of interest (ROIs) represented by contours, instead of rectangular bounding boxes. 137
Figure 4.15 - A comparison of the detection algorithms performance for different background types and time-frequency representations. The detection performance of atoms within an EEG background is depicted in **(A)**. The upper panes display the box

match error, while the bottom panes show the percentage of missed packets for three TFRs: SLT on the left, CWT in the centre, and STFT on the right. To facilitate comparison, the left panes in (A) recapitulate the top and bottom panes in Figure 4.15A. In **(B)** and **(C)**, the identical evaluation is presented for backgrounds of pink and brown noise, respectively. 138
Figure 5.1 - Configuration file for the CSF experiment. 143
Figure 5.2 - Gabor patches with the varying contrasts and spatial frequencies used in the experiment. 144
Figure 5.3 - Contrast sensitivity evaluation system configured for 3 monitors; two Gabor stimuli can be observed with an orientation of 0 degrees. 145
Figure 5.4 - Predictive Ability Assessment System. 146
Figure 5.5 - Example of a feline subject for a first setup for the testing of the environment. 146
Figure 5.6 - Configuration file for the Predictive Ability Assessment experiment. 147
Figure 5.7 - Inference of a ResNet model through DeepLabCut 151
Figure 5.8 – Inference of a trained SVM model for image segmentation. The left and middle image are the input image and mask used to derive new features and to train the model and the right is the segmentation of SVM. 152
Figure 5.9 – Inference of a trained RandomForest Classifier for image segmentation. The left and middle image are the input image and mask used to derive new features and to train the model. The image from the right is the segmentation of the RandomForest Classifier. 152
Figure 5.10 – Image segmentation made by PCNN classic model, on the right is the input and on the left the segmented image. It is able to discern the head of the fish from the rest of the image. 153
Figure 5.11 - Modified U-Net model architecture for Zebrafish Image Segmentation 154
Figure 5.12 - The inputs of the proposed approach are presented in the left and middle images, while the right is the output obtained. 155
Figure 5.13 - Loss (bottom) and accuracy (top) of the model throughout both training (left) and validation (right). 155
Figure 5.14 - The inputs of the proposed approach are presented in the left and middle images, while the right is the output obtained. 156

List of tables

Table 2.1 – A descriptive summary of various brain recording techniques. 19
Table 2.2 - Explanation of the Confusion Matrix [89]. 34
Table 2.3 - A descriptive summary of various clustering methods. 47
Table 2.4 – A descriptive summary of various clustering performance metrics. 50
Table 3.1 - Architecture of the Generator and Discriminator in the GAN. 68
Table 3.2 - Architecture of the classifier. 74
Table 3.3 - Alignment impact on the performance evaluation for PCA and AE on the Sim4 synthetic dataset. 79
Table 3.4 - Performance of different values of the number of epochs for the training process on the Sim4 synthetic dataset. 79
Table 3.5 - Performance of different values of learning rates for the training process on the Sim4 synthetic dataset [155]. 79
Table 3.6 - Evaluation of feature extraction methods on Sim4 containing 5 clusters. 80
Table 3.7 - Approaches ranked using the Borda ranking system with regard to each metric after applying them on all 95 synthetic simulations. 81
Table 3.8 - Performance analysis of all feature extraction methods on the M045-C17 real dataset. 82
Table 3.9 - Performance evaluation of Superlets features. 85
Table 3.10 - Evaluation of the neural network's learning from superlet features. 86
Table 3.11 - Comparative analysis between performance metrics applied to multiple channels with different thresholds on real data. 92
Table 3.12 - Comparative analysis between the performance metrics, applied to Superlet of orders 1, 2 and 5 on the M045-C17 real dataset. 93
Table 3.13 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim9 synthetic dataset. 98
Table 3.14 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim10 synthetic dataset. 99
Table 3.15 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim11 synthetic dataset. 99
Table 3.16 - Comparison through multiple metrics of various clustering algorithms combined with PCA or Isomap against the proposed method on the Sim12 synthetic dataset. 99
Table 3.17 - Evaluation of the number of chunks/nodes by varying the PN parameter on the Sim4 synthetic dataset. 103
Table 3.18 - Evaluation of the execution time (in milliseconds) for 100 runs for the number of dimensions on the Sim4 dataset using PCA to reduce dimensionality to the chosen values. 103
Table 3.19 - Evaluation of the execution time (in milliseconds) for 100 runs by varying the number of samples. 103

Table 3.20 - Evaluation of clustering algorithms using various performance metrics on Sim4. 105
Table 3.21 - Evaluation of clustering algorithms using various performance metrics on M045-C1. 107

List of abbreviations

AE	Autoencoder
AMI	Adjusted Mutual Information
ANN	Artificial Neural Network
ARI	Adjusted Rand Index
BFS	Breadth-First Search
BMU	Best-Matching Unit
CHS	Calinski-Harabasz Score
CMA	Cumulative Moving Average
CNN	Convolutional Neural Network
CNS	Central Nervous System
CSF	Contrast Sensitivity Function
DBS	Davies-Bouldin Score
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DFT	Discrete Fourier Transform
DI	Direction Index
DLC	DeepLabCut
DOA	Depth of Anaesthesia
ED	Edging Distance
ETA	Event Triggered Average
EEG	Electroencephalography
FCM	Fuzzy C-Means
FMI	Fowlkes-Mallows Index
FE	Feature Extraction
FN	False Negative
FP	False Positive
FT	Fourier Transform
GFP	Global Field Power
HDBSCAN	Hierarchical DBSCAN
IBI	Inter-Burst Interval
ICA	Independent Component Analysis
IRT	ISI Rank Threshold
ISBM	Improved Space Breakdown Method
ISI	Inter-Spike Interval
LDA	Linear Discriminant Analysis
LFP	Local Field Potential
LTP	Long-Term Potentiation
LSTM	Long Short-Term Memory
MEA	Multi-Electrode Array
MER	Microelectrode Recording
MEG	Magnetoencephalography
MI	Max Interval
MI	Mutual Information

ML	Machine Learning
MS	Mean Shift
MSE	Mean Squared Error
MUA	Multi-Unit Activity
NN	Neural Network
OI	Orientation Index
PCA	Principal Component Analysis
PCNN	Pulse-Coupled Neural Network
PDF	Probability Density Function
PS	Poisson Surprise
PSI	Pattern Specificity Index
PSTH	Peri-Stimulus Time Histogram
PTA	Pattern Triggered Average
QE	Quantization Error
RF	Random Forest
RI	Rand Index
RNN	Recurrent Neural Network
ROI	Region of Interest
RS	Rank Surprise
SBM	Space Breakdown Method
SCS	Spike Cluster Score
SD	Standard Deviation
SL	Superlet
SLT	Superlet Transform
SOM	Self-Organizing Map
SS	Silhouette Score
SVM	Support Vector Machine
STFT	Short-Time Fourier Transform
TE	Topographic Error
TFR	Time-Frequency Representation
TMS	Transcranial Magnetic Stimulation
TN	True Negative
TP	True Positive
T-SNE	t-Distributed Stochastic Neighbour Embedding
VM	V-Measure

Glossary

Biology

Axon - Slender projection of a neuron that conducts electrical impulses away from the cell body as a way to transmit information. It can be interpreted as the output of the system.
Cortex - Outermost layer of the brain shown to have a key role in higher cognitive functions such as perception, memory, language, and decision-making. It has a thickness of 1.5 to 4.5 millimetres, and it contains most of the neuronal cell bodies conferring it the name 'grey matter'.
Dendrite - Branched extension of a neuron that receives information in the form of electrical impulses from other neurons. It can be interpreted as the input of the system.
Gyrus - Ridge on the surface of the brain, together with the sulci they enhance the surface area.
Interneuron - A type of neuron that acts as a connector, relaying inhibitory signals within the nervous system.
Neurotransmitter - A chemical messenger released by neurons that transmits signals across synapses by binding to the receptors of connected neurons, facilitating communication between nerve cells by producing changes in the target neuron.
Pyramidal cell - A type of neuron with a pyramid-shaped cell body and an elongated axon that plays a crucial role in transmitting excitatory signals within the nervous system.
Sulcus - Furrow on the surface of the brain, together with the gyri they enhance the surface area.
Synapse - Gap between nerve cells through which information is transmitted in the form of electrical or chemical signals, enabling communication within the nervous system.

Computer science, machine learning

Artificial neural network (ANN/NN) - A biologically-inspired computational model based on the structure and function of the human brain. It consists of interconnected nodes (called neurons) organised in layers to process and learn from a set of given examples.
Classification - Data analysis technique that assigns objects to predefined categories based on their characteristics, enabling organisation and analysis of information.
Clustering - Data analysis technique that groups similar items together based on certain characteristics, enabling the identification of patterns and structures within the data.
Feature Extraction - Data analysis technique that uses the relevant and informative characteristics from raw data to transform them, simplifying complex information and facilitating more efficient analysis or modelling.
Symbolic Analysis - Data analysis technique that converts data into symbols (discrete representations), allowing for the exploration and extraction of meaningful patterns, relationships, or structures within the data from a more manageable format.

Electrophysiology

Action potential or spike - A rapid electrical impulse travelling along a neuron's axon, facilitating the transmission of information between neurons and enabling

communication within the nervous system. A spike occurs when the membrane voltage surpasses a certain threshold (-55 mV) and rapidly rises to positive values (+20 mV) followed by a return to the resting potential (-65 mV).
Burst - A pattern of rapid and repetitive firing of action potentials by a neuron, followed by a period of relative quiescence. It can play a role in transmitting information and synchronising network activity within the nervous system.
Depolarization - A process where the neuron's membrane potential becomes less negative, thus coming closer to the voltage threshold and increasing the likelihood of generating an action potential.
Electroencephalography (EEG) - Non-invasive extracranial technique for the recording and measurement of the electrical activity of the brain through electrodes placed on the scalp.
Hyperpolarization - A process where the neuron's membrane potential becomes more negative (typically lower than the resting potential), reducing the likelihood of generating an action potential and thus temporarily inhibiting the production of other action potentials.
Local field potential (LFP) - Intracranial measurement of electrical activity generated by groups of (thousands of) neurons near the recording electrode, giving insights into the collective behaviour and communication of neurons in a specific brain region.
Refractory period - A short time interval after an action potential during which a neuron cannot generate another action potential.
Resting membrane potential - The stable electrical charge across a neuron's membrane when it is not transmitting signals, maintaining its readiness to generate an action potential. Typically, the resting membrane potential voltage is within -65 to -80 mV.

Signal processing
Spectrogram – a two-dimensional visual representation of the power of a signal over a time interval and in a certain frequency range. It can be created through various time-frequency analysis methods such as the Wavelet Transform or the Fourier Transform.
Superlet Transform (SLT) - A time-frequency analysis method and it has been shown to provide a better time-frequency resolution than the Short-Time Fourier Transform or the Continuous Wavelet Transform.

Miscellaneous
Amblyopia - Vision disorder that occurs in early childhood, commonly known as "lazy eye". It manifests as reduced vision in one eye that cannot be fully corrected with glasses or contact lenses, even with no structural damage to the eye.
Brain oscillations - Rhythmic patterns of electrical activity produced by the synchronous firing of neurons. It plays a fundamental role in communication and coordination among different brain regions, and it has been shown to have a role in cognitive processes such as attention, memory, and perception.
Spike sorting - Neuroscientific process where the action potentials recorded from multiple neurons are separated and grouped into individual units or spikes. It allows researchers to study the activity of different neurons within a neural population.