DEPARTMENT OF ENGINEERING MATHEMATICS

The Foundational Development of a Modular Pipeline and Interactive Visualisation for Spike Sorting

A Path to Accelerating Innovation in Neuroscience

Samuel Harris

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Monday 2nd October, 2023

Supervisor: Dr. Martin Homer

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Samuel Harris, Monday 2nd October, 2023

Contents

List of Figures

List of Algorithms

Abstract

Spike sorting is an essential data processing technique in neuroscience that isolates signals from individual neurons in raw electrode recordings, enabling the study of single neural and population dynamics. However, the procedure encompasses numerous complex steps, each offering many algorithmic options. The procedure requires both technical expertise and field knowledge, creating barriers to entry for new researchers and slowing progress.

This project's underlying aim is to increase the rate of development in spike sorting, leading to an increase in productivity in the field of neuroscience. This problem is addressed by developing a set of tools: a modular and extensible pipeline facilitating a spike sorting procedure's swift initiation and iteration and an accessible web-based demonstration acting as a supplementary teaching resource for those new to the field.

The Python pipeline abstracts each step into interchangeable "blocks" with standardised inputs/outputs. This foundation enables researchers to rapidly build custom workflows by swapping modular blocks and making their own. However, further refinement through abstraction is needed before open-source release.

The pipeline was adapted into a Flask API integrated with a frontend Next.js application to create an interactive web demonstration that enables hands-on spike sorting exploration, supporting active learning. While not yet publicly hosted due to resource limitations, the frontend exhibits promising functionality. Through comprehensive user testing and subsequent optimisation, this tool has the potential to serve as a widely accessible and effective educational resource.

In summary, this project developed preliminary modular code and visualisation that display the potential to enhance accessibility and innovation in spike sorting through collaborative refinement.

The code developed for this project is available on GitHub:

- The Python spike sorting pipeline and Flask API - https://github.com/SamwelZimmer/DataScienceProjectAPI
- The Next.js frontend application - https://github.com/SamwelZimmer/DataScienceProjectFrontend

Supporting Technologies

- The code developed for the spike sorting process was written in Python (version 3.9.6) [79], utilising various common data science packages, such as NumPy [29], SciPy [83] and scikit-learn [62].
- The interactive web visualisation frontend leveraged Next.JS [82], a React-based framework, deployed serverlessly on Vercel [81]. This codebase relied on TypeScript [46] for static type checking, alongside Tailwind CSS [38] for styling.
- Both Unlighthouse [91] and the Web Vitals browser extension [2] were used to assess the web application's performance.
- The API connecting the pipeline to the web application was constructed using the Flask framework [27].
- This project's code was written in the Visual Studio Code [16] editor, with GitHub [25] used for cloud-based version control for both the Python API [30] and the web application [31].
- The report was written in LaTeXusing the Overleaf online editor [60].

Notation and Acronyms

Biological Terminology:

AP	:	Action Potential
EEG	:	Electroencephalogram
fMRI	:	Functional Magnetic Resonance Imaging
GRP	:	Gastrin Releasing Peptide
HH	:	Hodgkin Huxley
LIF	:	Leaky Integrate-and-Fire
LFP	:	Local Field Potential
MEA	:	Microelectrode Array / Multielectrode Array
TTX	:	Tetrodotoxin

Machine Learning (ML) Techniques:

CNN	:	Convolutional Neural Network
GMM	:	Gaussian Mixture Models
GTM	:	Generative Topographic Mapping
LSTM	:	Long Short-Term Memory
MLP	:	Multi-Layer Perceptron
PCA	:	Principal Component Analysis
UMAP	:	Uniform Manifold Approximation and Projection

Standard Engineering Principles:

BIC	:	Bayesian Information Criterion
MAD	:	Mean Absolute Deviation
ODE	:	Ordinary Differential Equation
SNR	:	Signal-to-Noise Ratio

Programming Terminology:

API	:	Application Programming Interface
AWS	:	Amazon Web Services
CLS	:	Cumulative Layout Shift
CSR	:	Client Side Rendering
FCP	:	First Contentful Paint
FID	:	First Input Delay
HTML	:	HyperText Markup Language
INP	:	Interaction to Next Paint
LCP	:	Largest Contentful Paint
OOP	:	Object-Oriented Programming
OS	:	Operating System
REST API	:	Representational State Transfer API
SEO	:	Search Engine Optimisation
SSR	:	Server Side Rendering
TTFB	:	Time to First Byte
UI	:	User Interface
UX	:	User Experience

Standard Mathematical Notation:

s.f.	:	significant figures
a.u.	:	arbitrary units

Terminology

Biological:

- *Action Potential* - A brief electrical impulse transmitted along the membrane of a neuron to communicate information. Also referred to as a "spike".
- *Bursting* - A spiking pattern where a neuron alternates between periods of rapid spiking and dormancy. Important for neural coding.
- *Closed-loop experiment* - Recording neural activity in response to real-time stimulation.
- *Leaky integrate-and-fire model* - A simplified computational model of neuron spiking behaviour. More abstract than detailed Hodgkin-Huxley model.
- *Local field potential* - Low frequency extracellular electrical signals reflecting coordinated neural activity over a region [11]. Removed from recordings as noise.
- *Microelectrode array* - A device containing multiple electrodes in a grid pattern used to record extracellular electrical signals from neural tissue.
- *Neural coding* - The encoding of information in the brain through precise spike patterns and timing. Understanding neural coding can reveal how sensory information is represented.
- *Tetrode* - A bundle of four electrodes used to record neural signals, usually implanted in the brain for *in vivo* recordings.
- *Threshold crossing* - A basic spike detection method that uses a threshold value to determine the occurrence of a spike.

Machine Learning / Data Analysis:

- *Autoencoder* - A type of artificial neural network often used for feature extraction in spike sorting. Encoder compresses inputs into a latent space and decoder reconstructs inputs.
- *Clustering* - Grouping spikes based on waveform features to categorise spikes by their neuron of origin. Algorithms like K-means and GMM are often used.
- *Convolutional neural network* - A deep learning model commonly used for end-to-end spike sorting.
- *Feature extraction* - Transforming spike waveforms into compact numerical representations that capture key features. Reduces data dimensionality. PCA is commonly used.
- *Filtering* - Removing noise and artefacts from raw neural recordings by attenuating frequencies outside a specific band. Common techniques include bandpass, notch, and median filters.
- *Overclustering* - A common problem in spike sorting where clustering algorithms mistakenly split waveforms from a single neuron into multiple clusters.
- *Spike sorting* - The process of distinguishing spikes from different neurons in extracellular electrical recordings. Involves steps like filtering, feature extraction, and clustering.
- *Simulation* - Generating synthetic neural spikes and recordings *in silico* to validate spike sorting algorithms when ground truth labels are unavailable in real data.
- *Triangulation* - Estimating a neuron's spatial location relative to recording electrodes by examining spike amplitude differences across channels.

Web Development / Computer Science:

- *Accessibility* - The practice of designing applications and devices to be usable by the widest range of users, considering factors such as disabilities and wealth.
- *Flask* - A Python web framework used to build the API for the spike sorting visualisation application. Enables request handling and responses [27].
- *React* - A JavaScript library for building user interfaces. Provides components for building interactive web applications [20].
- *Neuromorphic computing* - A model of computing where the infrastructure is inspired by the nervous system.
- *Next.js* - A React-based framework that enables server-side rendering for web applications. Used to build the spike sorting application [82].
- *Tailwind CSS* - A utility-first CSS framework that provides classes for styling web content. Used for application styling [38].
- *User testing* - Evaluating an application or interface by observing users completing tasks. Provides insights into usability issues and user comprehension.
- *Vercel* - A cloud platform optimised for Next.js applications. Provides hosting for the spike sorting visualisation web application [81].
- *Web worker* - JavaScript running in background without interfering with page, enabling parallel processing.

Acknowledgements

I would like to thank my supervisor, Dr. Martin Homer from the School of Engineering Mathematics at the University of Bristol, for his consistent guidance throughout this project. Additionally, a heartfelt thank you to Jake Ahern, a postgraduate at the University of Bristol, for his suggestions regarding the project's direction and for providing data.

Ethics statement: This project fits within the scope of the blanket ethics application, as reviewed by my supervisor Dr Martin Homer.

Chapter 1

Introduction

Neuroscience is the study of the brain and the nervous system, representing the only scientific field where the subject essentially studies itself. Despite our physical and emotional attachment to the brain, very little is known about it - as stated by physicist Emerson Pugh, "If the human brain were so simple that we could understand it, we would be so simple that we couldn't." [65].

In recent years, understanding the inner workings of the brain has become top of mind, with surging interest in brain-computer interfaces, prompted by rapidly advancing technologies like Neuralink [55] that promise restored mobility for paralysis patients and telepathy-like communication. Similarly, refinements in neuroimaging techniques have also enabled earlier diagnosis of neurological diseases such as Alzheimer's [63] while providing surgeons with greater anatomical precision. For this rate of development to persist, the field demands tools that enable swift iteration and promote a steady inflow of new practitioners and researchers.

1.1 The Problem

Spike sorting is an essential data processing technique in neuroscience that isolates signals from individual neurons in raw electrical recordings. This poses challenges since the signals overlap. Spiking occurs when a neuron activates, and ion flows prompt voltage fluctuations across its membrane to transmit information. Electrodes capture these changes in potential. Recording methods are classified as either *intracellular*, which measures internal membrane voltage, or *extracellular*, which measures voltage outside the membrane. Techniques are also classified as *in vitro* if the subject is isolated tissue or *in vivo* if within a living organism.

Recorded data comprises noise-ridden signals with intermittent spikes, known as action potentials (APs). Spike sorting converts these raw traces into informative numerical representations and clusters spikes according to their neuronal origin. This enables examining single neuron behaviours and network-wide brain communication. The procedure entails data collection, pre-processing to remove noise and extract AP waveform features, and clustering similar spikes. Each step offers numerous techniques spanning simple established methods to recent advanced developments.

This project addresses challenges arising from the vastness of algorithmic options for spike sorting. It aims to facilitate innovation by reducing iteration time and attracting new researchers through accessible educational tools. There are two main objectives: to create a modular and extensible spike sorting pipeline and to produce an interactive web-based demonstration for said pipeline.

An open-source, modular, and extensible Python module that abstracts each spike sorting step into interchangeable "blocks" could enable researchers to construct customisable spike sorting pipelines rapidly. Adhering to defined input and output formats for each block would allow creating or adapting algorithms, accelerating the iteration and integration of novel techniques. By contributing new methods back to the shared module, collaborative innovation could substantially accelerate the field.

This module can then be adapted into an API backend, facilitating an interactive web-based visualisation that supplements traditional spike sorting education. Spike sorting is a complex procedure, often requiring months of literature review for newcomers to grasp state-of-the-art techniques. An interactive tool can enable users to directly understand how algorithm choice and parameter adjustments impact results. Additionally, if the techniques and procedures in the Python module expand, this demonstration can follow suit - not only acting as a learning resource but also as a means of experimentation and testing for researchers without a technical background, removing a barrier to entry imposed upon the field. This

web demonstration must be suitable for mobile devices and low-powered commodity hardware so it is accessible to as many people as possible, no matter their equipment or economic position.

Both objectives are, in essence, the organisation of the surrounding literature into a set of usable tools with a common aim: increasing the rate of innovation and productivity within academic neuroscience and laying the foundation for technological innovation, which may lead to a widespread increase in the quality of life.

1.2 Preconceived Project Complexities

This project will undoubtedly encounter several challenges as it consists of many parts spanning multiple disciplines, such as algorithm design, full-stack web development and neuroscience. Some of the most likely challenges to appear are discussed below.

Simulating Neural Signals

Although spike sorting typically utilises real neural recordings, developing and evaluating algorithms relies on simulated data where ground truth labels are available. Accurately modelling complex dynamics like noise, attenuation, and spiking *in silico* requires tradeoffs balancing biological precision and computational efficiency. Furthermore, artefacts found in real signals may be inherently unpredictable and irreplicable. However, when creating these algorithms, it is important to use real data to ensure robustness to these unmodelled features. Some central issues include formulating valid synthetic neuron models and ensuring seamless integration of natural or synthetic data into the top of the pipeline.

Achieving Modularity

For the Python module to broadly impact spike sorting, it must encapsulate common processes into interchangeable blocks. This necessitates extensive upfront planning, as retroactive decoupling is long and tedious. Some techniques may span multiple steps or even bypass them altogether, such as end-to-end deep learning models [86], resisting simple abstraction. For the pipeline to flow seamlessly, each block must have a standardised input and output, which could impose limitations. However, following best software engineering practices for extensibility eases future expansion.

Effective Visualisation

As an educational tool, the web interface must engage users by distilling complex concepts into an intuitive interface. Tamara Munzner's task taxonomy [50], and principles of data visualisations, will guide the development to ensure success. However, compelling visuals alone may insufficiently retain learner attention. The central challenge lies in striking a balance where interactivity enhances rather than distracts from core lessons, as excessive customisable parameters risk cognitive overload and poor performance. However, insufficient interactivity will not provide enough customisation for useful experimentation. Operating system (OS) native applications, like an app made solely for Apple Macintosh, offer greater computational control, yet these are considerably less accessible than a web application.

Promoting Accessibility

An accessible tool can mean two things: it is widely available or it accommodates minority audiences, such as those with physical disabilities. Although different, in the context of this project, better accessibility means getting these spike sorting tools into the hands of more people. Within this project's scope, it is nigh on impossible to create an application usable by the entire population, but specific changes can be made to maximise the catchment demographic. For example, strong visual contrasts and legible fonts aid users with visual impairments, while clear, interactive element sizing assists those with limitations in their fine motor control. Accessibility remains an ongoing imperative for equitable educational tools.

1.3 Report Structure

The remainder of this report is structured as follows. Chapter 2 provides essential background on the concepts, techniques, and prior literature underpinning the project. This background contextualises the vast array of spike sorting algorithms; much of the decision-making processes have been offloaded to

this chapter. Chapter 3 details the execution of the Python spike sorting pipeline, explaining the specific procedures implemented in each step. Chapter 4 discusses how this pipeline connects to the visual demonstration, and how each pipeline step is represented in the interactive web-based environment. The development details of the frontend application are omitted from this report for two primary reasons. Firstly, the intricacy and technical nature of the process resonates poorly in a concise written format such as this report. Secondly, the end users only engage with the final product, making the development process unimportant to their experience. The code for building this application can be found on GitHub [31]. However, unlike the Python spike sorting pipeline, it has not been optimised for readability, as this code is simply a means to an end. Chapter 5 provides potential ways this project could be expanded upon and suggests concepts for future work. Chapter 6 concludes the project by reviewing its outcomes, achievements and shortcomings.

The aims, objectives and achievements of this project are listed below accordingly.

Aims

- Reduce the time for an experienced, technical user to initiate a spike sorting process and allow them to incorporate their own methods.
- Increase the ability of neuroscience students to learn about spike sorting through visuals and experimentation.
- Remove the technical barrier to entry to the spike sorting process.

Objectives

- Create an open-source modular and extensible Python module to accelerate spike sorting pipeline development.
- Produce a responsive web-based no-code demonstration accessible on common hardware to improve student learning.

Achievements

- Developed a Python pipeline with modular blocks to enable rapid innovation.
- Designed and built an interactive web-based educational demonstration compatible with smartphones and tablets. The demonstration is ready for online hosting, pending necessary funding.

Chapter 2

Background

This chapter provides the scientific context and technical concepts underpinning this project. First, a foundational neuroscience background on neural signalling, coding, and the need for spike sorting establishes motivation. Next, common techniques for simulating neuronal activity *in silico* are reviewed, given this project's simulation-based approach. The chapter ends with an overview of the standard steps, algorithms, and challenges in the spike sorting pipeline. Together, this section expresses the domain knowledge required for this project, and the literature presented in this chapter has greatly influenced many decisions made throughout the project.

2.1 Neuron Overview

Nerve cells, or neurons, are the primary cells of the nervous system, which specialise in transmitting information throughout the body using intricate mechanisms, enabling subconscious and complex cognitive processes alike. A human brain contains billions of neurons [6], each communicating through electrical and chemical signals. This section will use the literature to explore the fundamentals of neuron excitation, the mechanisms of their communication, and the reasons for processing their data.

2.1.1 Neurons and Neural Signalling

Information is transmitted by electrical signals, or action potentials (APs), within each neuron through a voltage change across the neuron's cell membrane. As the generation and propagation of these signals are metabolically demanding, "brains have evolved efficient ways of developing an energy-efficient neural code from the molecular level to the circuit level" [95].

Neurons communicate chemically through specialised junctions called synapses. The electrical signal transforms within the neuron into a chemical signal mediated by neurotransmitters. These neurotransmitters cross the synapse and bind to receptors on the receiving neuron, changing its membrane potential and initiating an action potential [47]. Unlike electrical synapses, neurons are not physically connected in chemical synapses, resulting in slower information transfer. However, they offer greater versatility and can connect a wider variety of neurons [40].

Neural Spiking

Action potentials (APs) only originate from specific parts of the neuron, typically the axon initial segment, propagating along the axon until it reaches the synaptic terminals (see Appendix Figure B.1).

In neuroimaging, APs are commonly referred to as "spikes" as they are a brief and rapid change to the membrane potential of a nerve cell, translating to a sharp spike-like feature in a recorded signal. During a spike, the voltage fluctuation from the resting state to peak depolarisation is typically 100mV due to the opening and closing of voltage-gated ion channels, allowing ions to flow across the cell membrane [53].

When a neuron experiences sufficient excitation, passing a threshold, its membrane potential begins to depolarise (see Appendix Figure B.2). At this point, the voltage-gated sodium channels open, allowing for a sudden influx of sodium ions, resulting in a spike in membrane potential. Following this, the sodium channels self-inactivate and voltage-gated potassium channels open, allowing potassium ions to exit the cell, repolarising the membrane. The exact value of these thresholds varies among neurons, and there are many influencing factors, such as recent neural activity and the presence of modulatory substances [35].

When measuring the APs of neurons, the spike direction isn't always positive. Various factors, including the recording technique, can account for this variation. Intracellular recordings show action potentials as rapid depolarisation. This happens because the neuron's interior becomes relatively more positive compared to the outside during a spike. Conversely, MEA and other extracellular recording methods often display spikes as negative deflections. This occurs because the region close to the neuron becomes relatively more negative as the potential inside the neuron rises. These variations necessitate slight modifications in the preprocessing pipeline when analysing a neuron's spiking activity.

2.1.2 Neural Coding

Neurons convey messages through these APs, with distinct spike patterns encoding diverse information. These patterns represent external stimuli, like sensory inputs, and internal states, like memories. Relevant patterns linked to particular tasks or stimuli can be discerned by analysing the activity of neuron clusters in the brain. For instance, a 2017 study by Trautmann et al. opted to study multi-unit threshold crossings rather than individual sorted neurons. Their findings suggest that by accurately estimating the geometry of low-dimensional manifolds, one can discern the neural dynamics related to specific tasks [77].

Spiking Behaviour

The firing or spike rate describes the number of spikes a neuron emits within a specific period. This rate indicates the intensity or magnitude of stimuli, such as light intensity or sound duration. However, not only does the spike rate matter, but also the precise timing of these spikes. This timing is crucial for information encoding, underpinning the concept of temporal coding. For example, a 2016 study by Srivastava et al. found that songbirds rely on precisely timed spike patterns for respiration [73]. Neurons do not operate in isolation. They belong to expansive networks with interconnected activities stemming from shared inputs and connections. Examining the correlated spike patterns of these neurons can shed light on how different brain parts synchronise to process information. For instance, neurons that fire simultaneously might belong to the same functional network.

Information Encoding

Decoding how neurons encode information is essential for understanding brain activity. This knowledge is crucial for advancing technologies like brain-computer interfaces and deepening our understanding of brain functions and disorders. Encoding models play a central role in predicting brain responses to sensory stimuli, illuminating how the brain represents sensory information [28].

2.1.3 Motivation for Spike Sorting

Spike sorting is a process used in neuroscience to classify recorded signals emitted from neurons. A single electrode can detect the signals from multiple neurons, and spike sorting aims to attribute these signals to individual neurons. Modern recording techniques, particularly those which employ high-density microelectrode arrays (MEAs), detect the signals from a vast number of neurons among many electrodes, resulting in a complex mixture of spike waveforms in the recorded data [22]. The sorting procedure distinguishes genuine neuronal activity from noise and other artefacts by isolating and analysing the neuron spikes. After segregating the spikes of individual neurons, researchers can study the response of each neuron to particular stimuli and further understand the neural coding of the brain [67]. Because of this, spike sorting has become an indispensable preprocessing step in the analysis of neural activity.

2.2 Simulated Neuron Spiking

The simulation of neurons and their spiking behaviour is a prominent tool in neuroscience. By replicating the patterns of neuronal activity *in silico*, the mechanisms that govern neural function, from the individual neuron level to entire neural circuits, are more accessible for study. Offering a controlled environment where variables can be isolated and manipulated allows for observations that are challenging to obtain through direct biological experiments.

2.2.1 Applications of Artificial Neuron Simulation

Simulating neuron spiking extends beyond pure scientific inquiry, and the ability to accurately model and predict neuronal behaviour becomes highly important in several technological fields.

Simulating neurons aids several technological fields. Electrodes record neural activity for spike sorting using machine learning, enabled by ground truth labels from synthetic signals [10]. Spiking neural networks (SNNs) offer alternate computing paradigms, like Sourikopoulos et al.'s energy-efficient artificial neuron [71]. SNNs also further artificial intelligence, as seen in Mozafari et al.'s SpykeTorch for simulating deep convolutional SNNs [48]. Self-adaptive models like Yang et al.'s SAM replicate working memory and learning [94]. And, SNNs can bridge neuroscience and machine learning through biologically realistic neuron models, as highlighted in Yamazaki et al.'s recent review [93].

2.2.2 Simulation Techniques

The Hodgkin-Huxley Model

The Hodgkin-Huxley (HH) model, introduced in 1952, is a foundational model in the study of neuron spiking describing the initiation and propagation of APs [32]. It represents the flow of ions (potassium and sodium) through the voltage-gated channels of the neuronal membrane using simple first-order ordinary differential equations.

The HH model, grounded in experimental data, describes the ionic mechanisms responsible for neuronal excitability. This model captures diverse neuronal behaviours, including APs and bursting. However, its depth of detail comes at a cost: the model relies on intricate differential equations, making it computationally demanding. While the HH model boasts the flexibility to emulate various neuron types or conditions by adjusting parameters, achieving biological precision necessitates meticulous calibration of these values [7].

Leaky Integrate-and-Fire Model

The Leaky Integrate-and-Fire (LIF) model is a simplified neuronal model that captures the essential dynamics of neurons. Instead of using the biophysically detailed HH model, the LIF model represents the neuron's behaviour through linear differential equations. This model tracks the neuron's membrane potential, accumulates incoming signals, and produces a spike once it hits a threshold. After the spike, the potential resets.

Contrasted with the HH model, the LIF model offers mathematical simplicity by abstracting many of the biophysical intricacies inherent in actual neurons. This abstraction, however, may hinder its precision in emulating specific neuronal behaviours. Yet, this simplified approach brings several advantages. The LIF model's computational efficiency facilitates real-time simulations of extensive neuronal networks. Its inherent flexibility empowers researchers to easily augment the model with features like adaptation [1] or stochasticity [2] without adding considerable complexity. Furthermore, its capacity for broad generalisation ensures that the LIF model encapsulates the fundamental dynamics of neuronal activity, making it a versatile model suitable for various neuron types. Although the LIF models effectively capture spike generation through threshold dynamics, they don't inherently exhibit complex behaviours, such as bursting.

Modern Approaches

The neural dynamics of a system are influenced by interacting spiking neurons [18]. A recent 2023 study by Olenin et al. introduces a new model that can simulate a neural population's spiking, bursting and chaotic temporal behaviours [57]. This model was developed to model the neuron–glial interaction and it is important in understanding the glial cells' role in neural dynamics.

Compute power is the most significant roadblock when simulating large interconnected neural populations. In 2022, Torti et al. leveraged a Multi-GPU System to run a LIF-based simulation to boast significantly improved processing times - reducing the simulation time of a network with "more than 1.8 million neurons from approximately 54 to 13" hours [76].

[1] Regarding neurons, adaptation is the common phenomenon of decaying neuronal activities in response to repeated or prolonged stimulation [8].

[2] Stochasticity refers to the neuron's propensity to activate spontaneously without an external stimulus or synaptic excitation that surpasses the activation threshold [49].

2.2.3 The Challenges of Neural Simulation

The dynamics of the brain are complex, and much about the gigantic neural network is unknown, making it nigh on impossible to produce an ideal representation *in silico*. However, the combination of artificially simplified conditions and mathematical modelling has allowed particular challenges of the simulation process to be overcome.

Modelling Neuron Behaviours

Neuronal spiking patterns and rates can vary drastically, making it difficult for a single model to capture all types of neural behaviour. These diverse spiking patterns encode information in unique ways. For example, tonic firing generates APs at a relatively constant rate without pauses. In contrast, phasic firing produces bursts of rapid spikes followed by periods of inactivity, allowing neurons to signal rapid changes in stimuli [85]. Bursting neurons exhibit alternating patterns of tonic firing and dormancy. Adaptive neurons decrease their firing rates in response to sustained stimuli, potentially due to changes in membrane channels. Meanwhile, some neurons are susceptible to network oscillations, firing rhythmically with the cyclic excitability of the neural network. Enlightening how different spiking modalities encode information remains an ongoing challenge due to the diversity of neuronal signalling behaviours.

Signal Attenuation and Noise

Ions moving across neuron membranes generate electrical fields due to neuronal activity, and electrodes record these fields. However, the signal they record is often a noisy and weaker version of the original signal from the neurons. When simulating these conditions, it is important to account for this suboptimal spiking behaviour.

Different parts of the brain, like grey matter, white matter, and cerebrospinal fluid, have distinct conductivities. Therefore, we should model the brain as a heterogeneous volume conductor. We can describe the behaviour of electrical signals in such a medium using the cable theory [3] for individual neurons and the volume conductor theory [4] for bulk tissue.

Signals attenuate with distance from the source neurons as they pass through various tissue types. Often, the Fourier transform and Green's functions are utilised to model this attenuation, representing the signal in the frequency domain, making it more straightforward to describe phase shift. For less complex models, such as those not representing a frequency change in the attenuated signal, simplifications are often employed, like modelling the signal strength proportional to the inverse square of the distance between the neuron and the recording site.

Another challenge when replicating neuronal signal recording is the incorporation of electrode characteristics. The electrode's type, size and material will all influence the recorded output. For example, larger electrodes that tend to average over larger tissue volumes report a smoother signal but at the cost of spatial resolution.

Unfortunately, electrode recordings do not solely register the signals emitted by neurons but also considerable amounts of noise. The sources of this noise vary based on the specific recording instrumentation, but we must model them to produce a realistic simulation. This noise can originate from electromagnetic interference from other electronic devices and electrodes; other physiological processes (e.g. muscle activity) and movement artefacts may arise from the subject or electrodes moving during the recording process. It is not uncommon to see these noise sources replicated crudely using a simple noise function, such as Gaussian or Brownian, as these methods may fare better as more generalisable models.

2.3 The Spike Sorting Process

Spike sorting enables the study of individual neuron behaviour and neural network dynamics. By distinguishing the spikes generated by different neurons, researchers can analyse the firing patterns of individual neurons and how they interact with each other. In turn, this can provide valuable insights into the functioning of neural networks and their responses to various stimuli or conditions.

[3] In the cable theory, a neuron's axons and dendrites are modelled as cylindrical conducting cables with resistance, capacitance, and longitudinal currents [33].

[4] In the volume conductor theory, the brain is treated as a conductive medium where currents from neuronal activity generate electric fields and potentials according to the properties of the medium [26].

Manual spike sorting is inadequate as it takes a researcher considerable amounts of time, and the outcome varies drastically depending on the individual. Wood et al. found wide variability in the number of neurons and spikes manually detected in real data, with average error rates of 23% false positive and 30% false negative for synthetic data [92]. However, technological developments have made it easier to record and analyse data from a large number of neurons. For example, high-density MEAs, combined with real-time spike sorting techniques, offer the possibility to study plasticity in neural networks consisting of several thousand neurons *in vitro* [22]. On the other hand, a study by Trautmann et al. investigated whether spike sorting is even necessary to estimate neural dynamics, finding that the scientific conclusions are quite similar when using multi-unit threshold crossings [5] in place of sorted neurons, suggesting that accurate estimation of neural population dynamics is achievable without spike sorting [77]. Despite this, spike sorting remains a staple procedure in most studies involving analysing neuron activity.

2.3.1 Neural Signal Acquisition

In the early 1960s, Hubel and Wiesel conducted groundbreaking experiments studying the visual system's development in kittens [90, 37]. To conduct such localised studies, Hubel and Wiesel developed tungsten microelectrodes, a significant advancement in neuroscience, allowing for a more detailed and precise understanding of neural activity.

Intracellular recordings like Hubel and Wiesel's tungsten probe [90] directly measure within neurons, offering high spatial and temporal precision (see Appendix Figure B.3), but inserting them into the brain is complex [51]. Intracellular recordings directly measure a neuron's electrical and biochemical activity.

Extracellular recordings detect external signals, enabling multi-neuron monitoring at the cost of lower resolution and signal quality. Modern techniques expand capabilities: wide-field calcium imaging captures large-scale activity [56], Neuropixels enable high-density multi-region recording [39] despite drawbacks like invasive damage [66]. Many of these extracellular recordings, like the tetrode, are invasive but can study the brain *in vivo*.

In vitro microelectrode arrays (MEAs) facilitate controlled neuronal network studies [22]. These are typically small rectangular arrangements of electrode but there are several alternatives, like Geramifard et al.'s 3D helical MEA [23]. While techniques differ, extracellular and intracellular signals can be processed similarly. This project will be using data from three sources: a tetrode, MEA, and simulated neurons.

Factors Affecting Signal Quality

Overlapping spikes are a common problem among many neural recording methods, and they can be challenging to resolve in the spike sorting process. Techniques with low spatial resolution, such as EEG, often result in the overlapping of spikes from multiple neuronal sources. Even techniques with many electrode recording sites suffer from spike overlap and collisions due to the large number of neighbouring neurons. This issue becomes more pronounced as the number of electrodes increases – tetrodes with their four channels see less overlap than MEAs, which in turn see less than Neuropixels with ∼1000 recording sites [74].

Unlike intracellular techniques, extracellular techniques often have low *signal-to-noise ratios* (SNRs). The SNR is a more apparent issue when using non-invasive methods, where the low spatial resolution increases the prominence of background network spikes, noise, and artefacts that can obscure neurons of interest.

The high temporal resolution and channel counts of modern tools like Neuropixels and MEAs, which can record gigabytes of data per second across their hundreds to thousands of channels, also require substantial computing power and data storage. This data must be stored and processed, often requiring simultaneous analysis of multiple signals, straining computational resources.

2.3.2 Neural Signal Preprocessing

After collecting the raw signal data, the next step is cleaning and formatting to become usable in downstream tasks. Usually, this step consists of several procedures, sequentially converting the uninformative raw data into a vector representation of neural spikes.

[5] A method in neuroscience for detecting collective neural activity, counting 'spikes' in electrical signals indicating a neuron's action potential crossing a set threshold.

Filtering

Filtering aims to remove noise and interference to help isolate the spike frequency bands in each signal. A raw electrode recording often hosts high-frequency and low-frequency noise caused by local field potential (LFP), although many software tools remove this during recording. Appendix Figure B.4 illustrates how a noisy, uninsightful raw signal from a tetrode can become useful with well-defined spikes after filtering.

The electrical wires delivering power to the electrodes may interfere with the recording at specific frequencies (e.g., 50 or 60 Hz), contaminating the signal. Fortunately, these frequencies are known prior to analysis and are removable using *notch filters*.

Bandpass filters allow signals within a specific frequency range to pass through while attenuating frequencies outside this range. Bandpass filters are well suited for emphasising the neural spikes in a recording as they usually occur within a specific frequency band [52]. The ideal frequency cut-off values for this band depend on the features of the electrode and the neuron itself.

A *median filter* is a non-linear type of filter which replaces each sample with the median of its neighbours. By using these filters, one can remove short-duration noise spikes without distorting the underlying signal significantly.

Python, and many other programming languages, allow for easy implementation of these filters using packages such as SciPy [83].

Spike Identification and Waveform Extraction

Once the signal has been filtered to accentuate the recording's important features, one can extract the APs. First, these spikes must be detected. If the signal recordings took place along many channels simultaneously, using tools such as a tetrode or MEA, this AP identification subprocess should also be done simultaneously. The most common method for spike identification is setting a voltage threshold value and only detecting the spikes that exceed this value. This choice of threshold is of critical importance, as poor differentiation between useful information and noise will result in inaccurate extraction. A common method of choosing this threshold constant is to take a multiple (usually between 3 and 5) of the signal's deviation. Due to its robustness to fluctuations, the mean absolute deviation (MAD) is often used over the standard deviation when calculating the threshold value [75]; see Appendix Equation A.2. However, such methods do not account for overlapping signals which occur when two spikes occur almost simultaneously. These compounded spikes are often recognised as a single instance, leading to the distortion of waveforms and an increased likelihood of misclassification [22].

Overcoming a low SNR is a great difficulty in the spike identification subprocess. To address this issue, Kim and Kim employed nonlinear energy operators and neural network classifiers rather than voltage threshold methods, finding that these advanced methods achieved a correct classification ratio higher than 90%, even when the signal-to-noise ratio is as low as 1.2 [41].

After spike detection, it is often helpful to extract and plot them. This extraction is trivial once the waveform peaks have been located, usually taking a fixed period preceding and trailing this point. Visualisation of these waveforms can be useful for ensuring the identification and extraction processes' quality and distinguishing between spikes from different neurons or sources, as shown in Appendix Figure B.5a. The waveforms' shape can provide insight into the type of neurons recorded and their synaptic connections [3].

After extraction, spike trains, a sequence of timestamps indicating when each spike occurred, are formed. Such a representation allows further study into the neuron(s), analysing its firing rate, the temporal dynamics of neural activity and the neuron's response to various stimuli [24].

Feature Extraction

The primary goal of feature extraction is to transform the extracted spike data into a compact, usually vectorised, representation which can then be clustered or classified in later tasks. Dimensionality reduction aims to isolate the distinguishing features of the spikes. Such a process also decreases the data's size, reducing the computational complexity for subsequent analyses [92].

The simplest and most rudimentary method of feature extraction is the characterisation of spikes from their *peak metrics*, for example, the amplitude and width of the spike's waveform. This method is swift and computationally inexpensive but does not effectively capture all of a spike's information.

There are many dimensionality reduction techniques available, such as Generative Topographic Mapping (GTM) and Uniform Manifold Approximation and Projection (UMAP), yet the most commonly used in the spike sorting process is Principle Component Analysis (PCA). PCA transforms the original

data into a new coordinate system where the variance of the data is maximised along the principal components. In spike sorting, PCA captures the waveforms' most significant features. The dominance of PCA over more advanced methods like GTM or UMAP in spike sorting is due to several factors. Firstly, data reduced using PCA is considerably more interpretable than the output of other methods.

Furthermore, the nonlinearity of techniques like UMAP is often misplaced when linear means can effectively model the primary variability of clean spike data, resulting in an unnecessary increase in computation. There is also something to be said about the novelty of other techniques. The spike sorting community has a long history with PCA; it is reasonably robust [75], with many established methods and pipelines built around it. Transitioning to new, less-proven methods requires rigorous validation and testing.

In recent years, it has become more popular to extract features from neural data using deep-learning methods. The convolution neural network (CNN) is particularly effective in the dimensional reduction of neural data, capturing the spatial-temporal patterns of spike trains.

Autoencoders are an artificial neural network consisting of two primary parts: the encoder, which maps the input into a latent representation, and the decoder, which reconstructs the input from latent space. A recent study by Ardelean et al. proposed a spike-sorting feature extraction method leveraging autoencoders, achieving higher performance than other state-of-the-art spike sorting techniques [5].

2.3.3 Neural Signal Clustering

The ultimate goal of spike sorting is to categorise the recorded spikes based on their neuron of origin. This categorisation is usually done by clustering the data output from the feature extraction process.

Clustering Techniques

In the past, it was relatively common to cluster spikes by hand, which is highly inefficient and inaccurate. Grouping APs this way would require the waveforms to be reduced to only 2 or 3 dimensions so humans can visualise and manually identify points. This incredibly low dimensionality meant much of the important characteristic information of the waveform would be lost. Although human-machine hybrid clustering is still used today, leveraging machine learning is more common.

K-means is a widely used unsupervised machine learning method that aims to group similar spikes into non-overlapping clusters, assuming spikes originating from a particular neuron will have a similar waveform. This clustering technique has been used in combination with several feature extraction methods. For example, Chah et al. experimented with K-means combined with a feature extraction technique based on Laplacian eigenmaps. They found that the proposed algorithm yielded "significantly improved performance" over a procedure combining K-means with PCA, achieving a mean sorting accuracy of 73% as opposed to PCA's 58% [14].

K-means is simple, efficient and fast, yet it is far from perfect in the context of spike sorting. Fundamentally, the algorithm assumes clusters to be spherical and of equal size, which is often not the case. Like K-means, Gaussian Mixture Models (GMM) approximates the distribution of spike waveforms as a combination of k Gaussian distributions. However, unlike K-means, the cluster size is not fixed.

Common Problems with Clustering

Both the K-means and GMM models face a non-trivial challenge: determining the optimal number of clusters (k). One solution to this problem is the Bayesian Information Criterion (BIC). In essence, the BIC quantifies the trade-off between model complexity, which penalises the number of parameters, and the log-likelihood of the data given the model. By minimising this criterion (see Equation A.3), we can determine a proposed optimal value of k, although, in practice, this is often not the most appropriate value.

Although such an event is easily recognisable by eye, clustering algorithms tend to get stuck on local minima. There are several ways to try and prevent this, such as the split and merge approach [78, 97], yet they come at the cost of computation time.

When analysing real data, there are often outlying waveforms, resulting in a distribution with heavy tails. As the components in the GMM are normally distributed in nature, the clusters are unable to fit such distribution. When considering the Mahalanobis distance, it is apparent that replacing the Gaussians with t-distributions fit the data considerably better, as seen in Appendix Figure B.6.

Another issue, which is more apparent when using an invasive recording technique, is waveform drift caused by the gradual shifting of the electrode throughout the recording process. This can misalign

waveforms resulting in a sharp reduction in clustering accuracy. The Kalman filter mixture model is an adaptation of the GMM, which can account for these drifts in waveform by adopting a time component in the underlying algorithm [12].

In 2017, Shan et al. proposed a nonstationary generative model for spike sorting that can track waveform drift whilst being robust to outliers [68], aptly named the "Drifting Mixture of t-Distributions".

Automatic Spike Sorting

Spike sorting is an involved process with several complex steps, and in recent years, researchers have begun experimenting with automatic spike sorting algorithms, bypassing several of these intermediate stages. In 2018, Diggelmann et al. developed an automatic spike-sorting algorithm capable of dealing with large volumes of data from high-density MEAs [17]. They found that by sorting local groups of electrodes independently, this algorithm could circumvent the "curse of dimensionality" - which arises from dealing with the high dimensional data of an MEA recording.

These end-to-end spike sorters often take a deep-learning "black box" approach, where the model's inner workings are unknown. In 2020, Li et al. proposed a one-dimensional CNN capable of achieving a clustering accuracy of 99%, outperforming state-of-the-art methods like "WMsorting" [36] and multilayer perceptron (MLP) models on simulated data [44]. A recent 2023 study by Wang et al. built up this idea, using a CNN in combination with a Long-Short Term Memory (LSTM) model to sort and classify neural spikes [86]. This model also achieved a clustering accuracy of over 99% on simulated whilst reaching 95% on real data.

Evaluation of Clustering Performance

We need methods to assess their performance to determine the effectiveness of spike sorting systems.

Acquiring ground truth labels for real neural spiking data is inherently difficult due to the unknown nature of the task. For this reason, the quality of an algorithm is often quantitatively assessed on simulated spike data where the true labels are available. However, when this is not the case, visual inspection of the feature spaces and the resulting clusters can provide qualitative insights into an algorithm's performance. Although subjective and highly dependent on the observer, well-separated and compact clusters in the feature space are usually apparent to an untrained eye - especially when projected in two dimensions.

Accuracy is one of the primary performance metrics used to assess a clustering algorithm, measuring the proportion of correctly identified spikes to the total spikes. A confusion matrix often complements this metric, showing the classifier's true positives, false positives, true negatives, and false negatives [80].

Validation indices are statistical measures for classification algorithms, where high-quality clusters consist of spikes originating from the same neuron (high intra-cluster similarity) and spikes from different neurons are well separated (low inter-cluster similarity) [80].

Consistency is also a vital characteristic of a spike sorting algorithm. A process which returns varying outputs on otherwise identical data is inconsistent and may indicate sensitivity to initial conditions or the presence of local optima.

In summary, this chapter has reviewed essential neuroscience and spike sorting fundamentals, including neuronal communication methods, applications of neural simulation, and standard procedures for processing spike data. This background provides important domain knowledge to contextualise the significance of this project and we can turn our attention to the practical side of the project.

Chapter 3

Constructing the Spike Sorting Pipeline

This chapter presents the methodological and practical approaches to constructing a modular, extensible, and flexible spike sorting pipeline, offering insights into the methods, designs, and limitations fundamental to the project's primary objectives. First, the project design, data acquisition, and planned performance analyses are described to provide context. Next, the implementation of each pipeline step, simulation, preprocessing, clustering, and triangulation are explained in detail. Each step has at least one implemented technique, which is sufficient to build an end-to-end spike sorting module. Particular methods, algorithms, tools, and design choices are made with the guidance of the literature discussed in Chapter 2 whilst adhering to the project's aims and constraints. Code structure and accessibility practices are also highlighted.

3.1 Project Design

This project has two primary objectives, both aiming to make the spike sorting process more accessible. The first objective is to develop a modular and adaptable spike sorting pipeline in Python, the most popular programming language according to IEEE [13]. This pipeline will provide a framework of interchangeable components or "blocks" representing different techniques for each stage of spike sorting, such as various filtering methods and dimensionality reduction models. The modularity will allow rapid iteration and comparison of spike sorting approaches. The second objective leverages this pipeline to create an interactive visualisation that enables users to experiment with different spike sorting blocks and immediately view the results. Inspired by the rising popularity of no-code tools (e.g., Webflow [88] and Zapier [96]), this interface aims to enable those without specialised technical skills to experiment with different spike sorting techniques, serving as an educational demonstration of spike sorting principles. Together, these objectives work to make spike sorting a more transparent, iterative, and accessible process. However, we need neural spike data to design and test this framework.

3.1.1 Data Acquisition

It is important to use both real and simulated neural recordings when developing a spike sorting pipeline to ensure it is robust to diverse inputs. This project utilises MEA and tetrode data as they are both multichannel recordings, yet one is *in vitro* and extracellular, whilst the other is *in vivo* and intracellular.

The MEA data originated from a regular ten by six grid of electrodes, positioned on a slice of a rat brain *in vitro*, with each electrode recording simultaneously at 25kHz. The recordings taken by the MEA are over an hour in duration and comprising of three distinct stages. The first stage consists of spontaneous neural activity, where the tissue is unmodified. During the second stage, Gastrin Releasing Peptide (GRP) is applied to the tissue at a concentration of 200nM, generally increasing the cell spiking rates. And the final stage involves the application of tetrodotoxin (TTX), abolishing almost all neural activity. As illustrated in Figure 3.1, these three stages provide a comprehensive view of the rat's neural activity at varying excitation levels.

Unlike the MEA data, the tetrode data (sourced from Berens et al. [9]) contains a single stage of unadulterated neural spikes. While real data provides critical robustness, simulated data enables

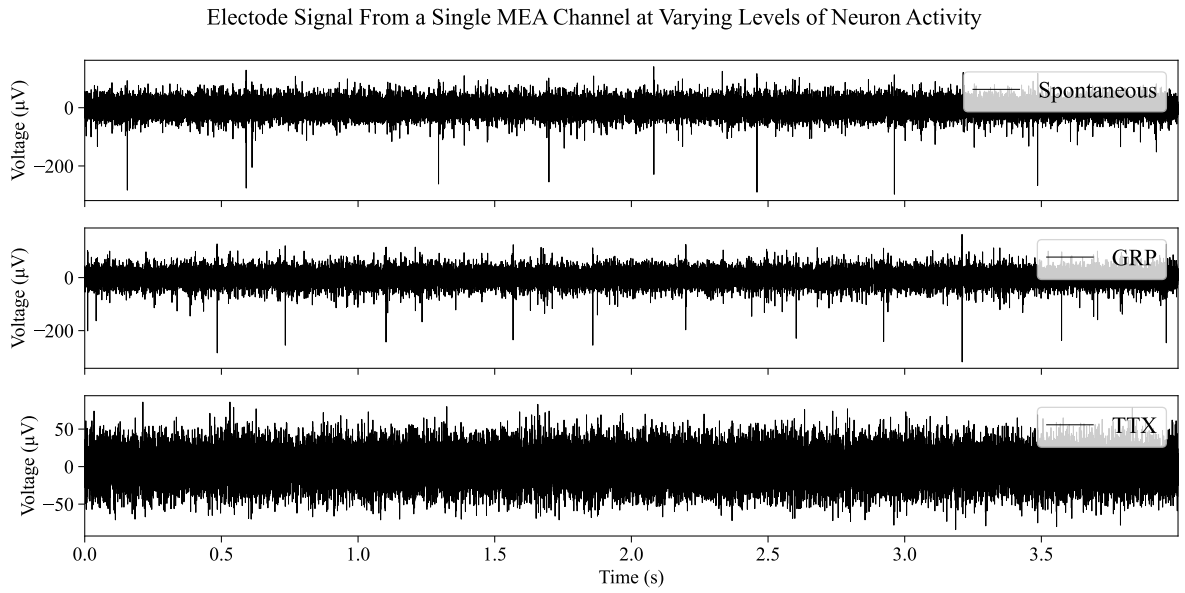


Figure 3.1: Electrode signal from an electrode in an MEA recording a slice of rat brain in vitro: during an unhindered state (upper), under chemical stimulation via Gastrin Releasing Peptide (GRP) (middle), and following neural activity reduction due to the application of tetrodotoxin (TTX) (lower).

quantitative pipeline evaluation by providing ground truth labels. Together, these diverse real and simulated datasets allow comprehensive testing of the spike sorting pipeline.

3.1.2 Performance Analysis

The focus of this project is not on the quality of the algorithms implemented into the blocks forming the pipeline but to provide an infrastructure so users can test their own configurations and algorithms. Implemented methods allow users to assess pipeline quality through end-to-end clustering metrics and triangulation errors on synthetic data.

Assessing the Python module's effectiveness should be based on modularity and extensibility. Modularity refers to the degree that pipeline blocks are self-contained and interchangeable. Extensibility refers to the ease of creating new customisable blocks. If the module facilitates swapping and adding techniques, it achieves its core aim of accessibility. While comprehensively evaluating these qualitative measures is non-trivial, it is essential for determining the pipeline's success as a flexible framework.

3.1.3 Limitations in the Project Design

This project has limitations stemming from its design and circumstances. The highly subjective goals make success challenging to judge quantitatively. While user surveys could aid assessment, time constraints preclude thorough evaluation. The tight schedule also necessitated focusing on computationally efficient, well-established algorithms rather than newer deep-learning approaches, limiting current options. However, the pipeline's modularity enables future expansion. Additionally, it is not feasible to conduct thorough unit testing, and despite it operating on a virtual environment set up using a list of Python module requirements, it is not certain that this pipeline will operate on all systems as intended.

3.1.4 Module Structure

The Python pipeline employs object-oriented programming (OOP) to enable modularity between blocks. Each process is a parent class that child classes inherit from, with standardised inputs and outputs. Type hints, docstrings, and comments explain the code's purpose and use. Adhering to best practices also improves accessibility and extensibility - version control via GitHub, virtual environments, and requirements.txt allow smooth forking and expansion. While designed for iterating on simulated data, real recordings can still be used by bypassing layout and simulation stages. Together, these coding principles empower users to understand, adapt, and enhance the pipeline for their research needs. The

focus on modularity, documentation, and extensibility aims to make the pipeline a flexible framework for spike sorting exploration.

3.2 Simulation

The first step in any spike sorting process is the collection of neural signal data. This data should be simulated in a controlled environment if a labelled data set is required. Several components are necessary to simulate neural signals: the synthetic neurons and electrodes themselves, their relative placements, and the medium, or matrix, connecting them.

3.2.1 Synthetic Neuron Modelling

The neuron block is designed so that each unique neuron type inherits the necessary methods from the parent class. So, long as the structure of the output is consistent, these methods can be tailored to specific neuron types.

The Leaky Integrate-and-Fire (LIF) model templates the neuron's spiking behaviour. The LIF model was chosen over the Hodgkin-Huxley (HH) model because it is less computationally expensive. This increase in computation speed is necessary when simulating a large array of neurons, especially when considering the visual demonstration (see Chapter 4) where users are likely to become disengaged or disinterested during long loading times.

The conditions under which a neuron will spike are modelled using the Poisson distribution,

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{3.1}$$

where k is the number of times a spike occurs in a given interval and λ is a measure of the neuron's spike rate. When this probability, $P(X = k)$, is greater than a random threshold (between 0 and 1), the neuron will spike. The refractory period (t_{ref}) between spikes is a tunable neuron parameter.

The characteristic differential equation of the LIF neuron is represented by,

$$v'(t) = \frac{-(v(t) - v_r) + R \cdot I_0}{\tau}, \tag{3.2}$$

where $v(t)$ is the membrane potential (mV), v_r is the resting potential (mV), R is the resistance of the membrane (Ω), I_0 is the initial current through the cell (A) and τ is the time it takes for a neuron to spike (ms). In models more advanced than the LIF used in this project, the membrane potential $v(t)$ is influenced by the potentials from neighbouring neurons.

The values of R, I_0 and τ were tuned to produce realistic spiking activity. These values are kept constant ($R = 10^4\Omega$, $I_0 = 0.01A$ and $\tau = 0.02ms$) within the blocks as a user's access to these would likely only lead to frustration and reduce the focus on the more important variables.Since these values typically represent constants of the recorded system, namely the brain and its cells, they are generally invariable. However, technical users can edit the source code to modify these values.

Solving this differential can be done relatively simply in code, and two different neuron blocks using different methods have been created. One uses the Runge-Kutta ODE solver from the SciPy [83] package. The other uses a custom implementation of the Euler method. The prior method is more reliable due to several under-the-hood optimisations.

The code used to simulate the excitation of these neuron types is very similar, both following the general process described in Algorithm 3.1.

3.2.2 The Connecting Medium

The signals generated by neurons propagate through the surrounding matrix, which envelops the simulated neurons and electrodes. The material properties of this medium significantly influence the distortion of the signals as they travel from their point of emission to the point of recording. In this simulated environment, the medium is also a source of ambient noise in the recorded signals. However, it is important to note that in real-world scenarios, such noise arises from various factors, including the activity of distant neurons and the electrical interference produced by the electrodes.

Data: resting potential, refractory time, time step
Result: array of membrane potentials over time
Initialise the neuron:
SET membrane potential to resting potential
SET time to 0
SET refractory counter based on refractory time
while *not reached end of the simulation* **do**
  DETERMINE if spike is probable based on random threshold and Poisson distribution
  if *no spike is probable* **then**
    SET potentials for a random duration (based on refractory interval) to resting potential
  else
    SOLVE the ODE to estimate potential up to the spike point
    ADD spike and post-spike events to the potential values
    SET potentials during refractory period to resting potential
  end
  UPDATE time by a time step
end
return array of membrane potentials over time

Algorithm 3.1: A pseudocode algorithm for the simulation of a neuron signal using the LIF model.

Signal Attenuation

As a signal propagates through space, the amplitude of the signal received by a sensor is typically lower than the amplitude at the signal's source - this phenomenon, known as attenuation, results from a loss of signal strength during transmission. The rate of attenuation is influenced by various factors, with the distance between the transmitter and receiver being one of the most significant. Additionally, the medium through which the signal travels also plays a role in the attenuation rate.

There are many ways to reasonably simplify and model the medium of the brain and its properties relevant to signal propagation. As discussed in Section 2.2.3, the brain is characterisable as a heterogeneous conductor composed of grey matter, white matter, and cerebrospinal fluid. The propagation of neural signals within this complex environment can be represented using approaches such as cable theory for individual neurons [33] or volume conductor theory for bulk tissue [26].

The spike sorting pipeline presented here does not allow for the differentiation of various types of neural matter. Due to the complexity this would introduce and the limited development time available for this project, more intricate models were not deemed necessary. As such, this pipeline represents the simulated surrounding matter as a homogeneous medium. The attenuation of signals in this model is conceptualised similarly to how the intensity of electromagnetic waves or sound diminishes with distance, following the inverse square law (see Appendix Equation A.4). In the context of the homogeneous medium through which neural signals propagate, the attenuation equation can be expressed as:

$$v_2(t) = v_1(t)\frac{k}{x^2}, \tag{3.3}$$

where $v_1(t)$ represents the signal emitted by a neuron, $v_2(t)$ denotes the signal recorded by the electrode, k is a parameter representing the transmissibility of the medium and x represents the distance between the transmitter and receiver.

However, this model is somewhat simplistic and has two significant limitations. First, it does not consider the temporal delay associated with signal transmission from a more distant neuron to the electrode. Such a delay could be modelled by incorporating a penalty term related to the medium's conductivity. Consequently, spikes originating from the same neuron could appear slightly shifted on different electrodes. Since this pipeline has undergone testing with actual data, the spike identification algorithm described in Section 3.3.2 accounts for this effect. Secondly, modelling voltage attenuation using the inverse square law is not entirely accurate, given the properties of electrical conductors. Nonetheless, this approach serves as a simplification. Future iterations of the pipeline could address these issues by incorporating more sophisticated solutions.

Modelling the attenuation of the neural signals is a vital step in this spike sorting process as it enables the triangulation of neurons using the relative amplitudes of signals incident on different electrodes (see Section 3.5).

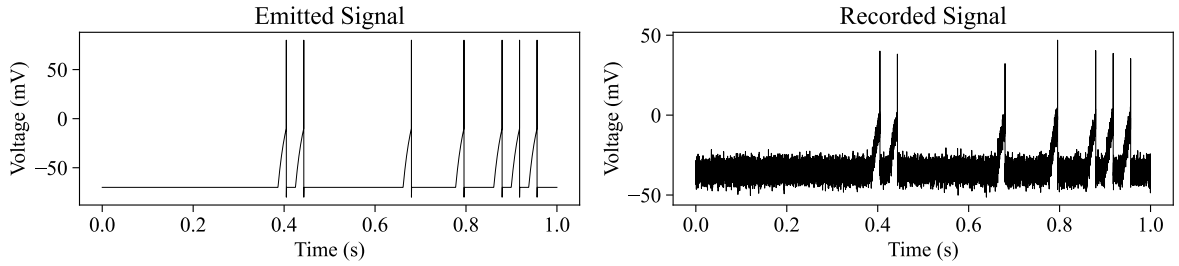


Figure 3.2: Simulated membrane potential of a leaky integrate-and-fire (LIF) neuron, solved using the Runge-Kutta ODE method (left). The simulation parameters include a firing rate of $\lambda = 14$, refractory period $t_{ref} = 20$ms, resting potential $v_r = -70$mV, and spiking threshold voltage $v_{thr} = 10$mV. The right plot depicts the potential recorded by a synthetic electrode located two arbitrary units away ($x = 2$) from the neuron. The signal attenuation between the neuron and the electrode follows an inverse square relationship, with the connecting matrix having a transmissibility constant of $k = 2$. Additionally, the electrode introduces Gaussian background noise with a standard deviation $\sigma_{noise} = 4$.

Noise

As discussed in Section 2.2.3, the electrodes register noise from multiple sources, including electromagnetic interference from the recording equipment, signals from distant neurons, and artefacts arising from muscular movements. These artefacts are unpredictable; hence challenging to model reliably. Electromagnetic interference, though present, can be effectively mitigated using methods such as notch filtering and thus has not been explicitly modelled. Gaussian noise appropriately represents the residual ambient noise, but other distributions, such as Brownian noise, are also suitable.

The principle of superposition allows for the addition of noise to a signal (see Appendix Equation A.5). This simple linear summation of the neuron's signal and the noise allows for the easy substitution of one noise distribution for another (e.g., from Gaussian to Brownian). The effects of signal attenuation and ambient noise are combinable into a single equation:

$$v_2(t) = v_1(t)\frac{k}{x^2} + n(t), \tag{3.4}$$

where $n(t)$ represents the noise at time t. Figure 3.2 illustrates the impact of the simulated matrix's characteristics on neural signal recording. Due to the superposition of noise, the average amplitude of a neural spike varies, making the process of spike identification (see Section 3.3.2) complex, even after filtering the raw signal. The maximum amplitudes of the signals emitted by the neuron and recorded by the electrode are 80.0 mV and 46.9 mV (3 s.f.), respectively.

Simulation Layout

The neurons and electrodes need to have some form of spatial structure. Using a grid-like system, their positions can be explained in cartesian coordinates, reducing many calculations to simple linear algebra and geometry. This, however, does not consider the brain's depth and, therefore, may not fully represent neural recording techniques such as the Neuropixel probe [39], which captures signals at varying depths.

Cartesian coordinates also make it simpler for users to choose the positions of the neurons and electrodes on the grid. In the Python implementation, there are checks to identify invalid component placements, such as when an electrode and neuron occupy the same position or a component lies outside the defined grid boundaries. The size of the grid is variable, allowing users to scale the resolution and vastness of the simulated brain region.

The implemented Python "Grid" class manages the simulation layout, the surrounding matrix's properties, the neurons' simulation, and the electrodes' recording activity.

3.3 Preprocessing

The signals recorded by the electrodes are noisy, which can obscure the neural spikes. Preprocessing takes this raw data and converts it into a more usable form. In spike sorting, preprocessing aims to produce a dimensionally reduced, feature-rich representation of the APs present in the signal. This

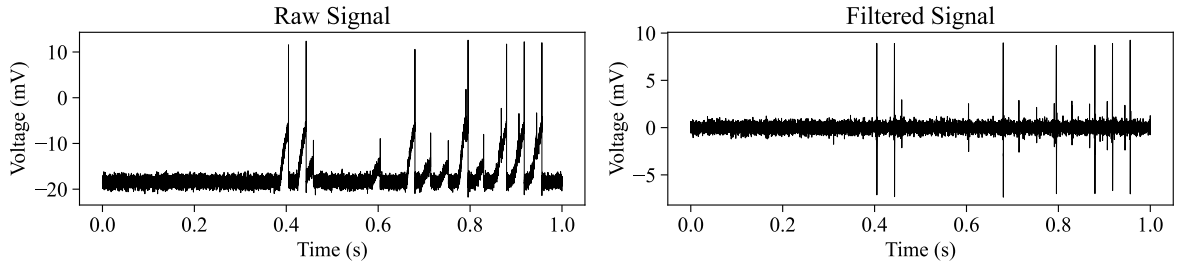


Figure 3.3: A raw electrode recording of two neurons situated at distances of 3.16 a.u. and 5.66 a.u. (3 s.f.) from the electrode. The simulation parameters for the neurons and matrix are as follows: $\lambda = 14$, $t_{ref} = 20$ms, $v_r = -70$mV, $v_{thr} = 10$mV, $k = 2$ and $\sigma_{noise} = 4$ (left). And the same signal when passed through a Butterworth bandpass filter with lower and upper frequency cut-offs of 500Hz and 4000Hz, respectively (right).

stage involves several sequential steps: signal filtering, spike identification, waveform extraction, and dimensionality reduction (or feature extraction). While the existing literature offers various approaches to each of these steps, as discussed in section 2.3.2, the initial version of this pipeline will focus only on the most foundational techniques.

3.3.1 Filtering

Generally, filtering is the first step in preprocessing neural signals. It aims to remove noise and artefacts originating from indistinct background neural signals, electromagnetic interference, and muscle movements.

The Butterworth bandpass filter is helpful for this purpose, as it allows a specified frequency range to pass through while attenuating frequencies above and below the designated thresholds. This is well-suited for neural signal processing, as neural spikes typically occur within a specific range of frequencies [52]. The filter is applied in both forward and backward directions to compensate for phase shifting introduced by the filtering process. Implementing this two-way combined Butterworth filter is straightforward, thanks to the SciPy module [83].

When considering digital signals, it is important to consider the Nyquist frequency, equal to half the signal's sample rate, representing the highest frequency that can be accurately captured without aliasing [58].

Figure 3.3 shows the outcome of a raw electrode subjected to a bandpass filter, accentuating the APs in the signal. Although the raw signal does not contain any low-frequency noise as the LFP is not simulated, the bandpass filter would account for such features if present (as seen in Figure B.4). Filtering is often unnecessary when handling synthetic data, as spikes are usually well-defined. In some cases, filtering may even introduce further complications.

3.3.2 Spike Identification

Once the raw electrode signals have been filtered, making the spikes more prominent, it becomes easier to identify the locations of these spikes for subsequent extraction.

A common method for spike detection is to set a threshold at a multiple of the signal's mean absolute deviation (MAD), categorising any points above this value as spikes (see Equation A.2). MAD is preferred over standard deviation due to its robustness against signal fluctuations. Although the simulated electrode recording here has positive-going spikes, spikes in real extracellular recordings (e.g., with tetrodes or MEAs) would be negative due to the relative potential of the surrounding matter. This polarity difference can be accounted for by using a negative threshold, thereby enabling the detection of both positive and negative spikes.

Since a spike is not just represented by a single point above the threshold, it is necessary to locate the peak of each spike. The process implemented (see Appendix Algorithm A.1) determines these peaks using the first differential of the signal to identify points where the direction of the signal's gradient changes. This algorithm also considers the possibility that some spikes are closely positioned within a signal, such as the two spikes at approximately 0.8s in Figure 3.3.

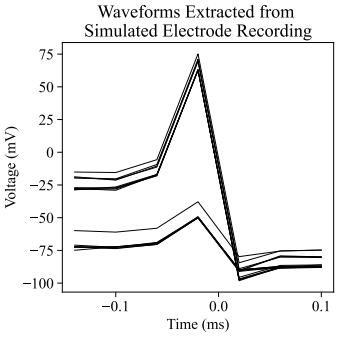


Figure 3.4: A plot of extracted waveforms from an electrode within a simulated multi-electrode array subject to the signals of two neurons.

3.3.3 Waveform Extraction

With the positions of the spike peaks located, extracting the waveforms is a relatively straightforward task. A spike waveform is essentially a small window of data before and after the peak, capturing the AP's defining characteristics. When simultaneously processing signals from multiple recording electrodes, if a spike is detected on any of these signals, the corresponding time window is captured on all channels. For example, if there are two signals (A and B), and a spike is identified on signal A at time $t = ts$, then a window of data is extracted from both A and B and time t_s. This data is useful when triangulating the source neuron position (see Section 3.5). This approach, however, may not be suitable for analysing data where recording sites are located at significant distances from one another, such as in an MEA. In such cases, it might be more appropriate to analyse only a local subset of recording channels at any given time.

In real data, the time it takes for a neuron's signal to reach the electrode varies depending on factors like distance. As a result, the identified positions may not correlate with the peak of the spikes across all channels. To account for this, the spike locations are merged to an average position, and the corresponding extracted waveforms are recentered so that the peak values remain in the same location within the extracted time window. This realignment is necessary to prevent the feature extraction step from becoming ineffective.

The extracted waveforms can be displayed with each superimposed on top of one another, as shown in Figure 3.4, or as a spike train. The layered waveform plot allows the user to visually determine the differences in the shape of the waveforms, segregating their neurons of origin. Figure 3.4 appears to show an electrode signal containing spikes from two distinct sources. However, this plot only shows the waveforms recorded by a single electrode, so this cannot be confirmed.

3.3.4 Feature Extraction

Feature extraction, or in this case, dimensionality reduction, attempts to capture the most significant characteristics of the waveforms, summarising them in a condensed numerical format that reduces computational load.

The number of dimensions the data is reduced into is a tuneable parameter. A greater dimensionality leads to a more detailed representation of the waveforms but at the cost of increased computation and decreased interpretability when visualising the data in a two-dimensional representation.

As discussed in Section 2.3.2, PCA is the most used feature extraction technique in the spike sorting process due to its high interpretability and ability to capture a large percentage of the waveform data quickly compared to other techniques. In short, PCA transforms the original data into a coordinate system where the variance of the data is maximised along the principal components.

Implementing this machine learning technique in Python, and others like it, is straightforward due to the functions provided by the scikit-learn library [62].

Figure 3.5 depicts the feature extraction of the waveforms shown in Figure 3.4, using both PCA and UMAP as dimensionality reduction techniques. The values of these waveforms were standardised before the feature extraction process. Generally, two-dimensional visualisations using PCA are easier to understand than non-linear methods like UMAP. This is evident in Figure 3.5, where PCA groups similar waveforms closely compared to UMAP's less defined representation.

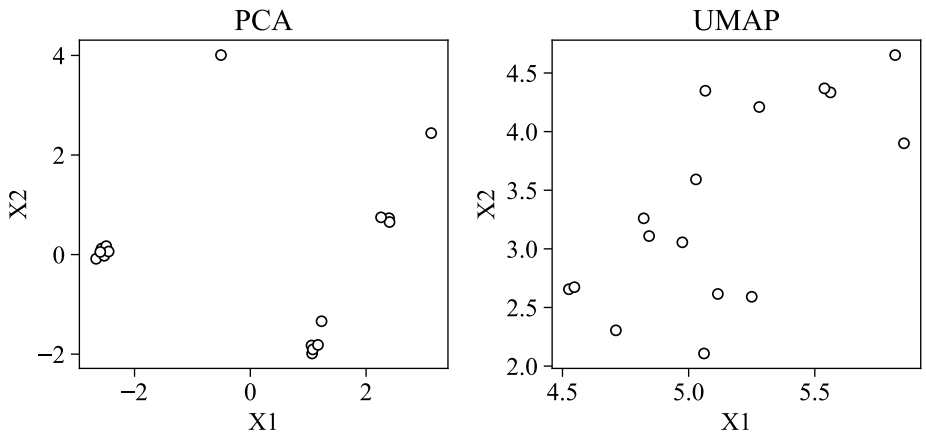


Figure 3.5: Two-dimensional representations of the waveforms from an electrode signal, reduced using PCA (left) and UMAP (right), with three components. In the PCA reduction, the first two components have explained variance ratios of 0.642 and 0.354 (3 s.f.), respectively.

3.4 Clustering

After feature extraction, the electrode signals are ready for clustering, which groups similar waveforms and identifies their source neurons. A good clustering algorithm takes the extracted, feature-rich representations of the waveforms across multiple electrodes and attempts to group them by similarity. The ideal outcome of such a process is well-defined clusters of waveforms. As discussed in Section 2.3.3, automatic clustering, using machine learning, is faster and generally more reliable than manual clustering. If the ground truth labels are available, clustering performance can be assessed using metrics such as accuracy and F1-score.

In this pipeline, we use GMM clustering, which groups the data into k clusters based on Gaussian distributions. The similarity of any two data points is represented by the distance between them in vector space. Clustering errors may arise from the algorithm itself or inaccuracies in previous spike sorting steps.

More advanced clustering methods are better suited to this specific task, like the "Drifting Mixture of t-Distributions" [68], as they account for outliers and the drifting nature of APs over time. However, this model is considerably more complex and beyond this project's scope.

The number of clusters (k) is a model hyperparameter. The BIC is a measure of model fit that considers both the goodness of fit and the number of parameters used [84]. Choosing the k value corresponding to the lowest BIC value is a reasonable heuristic. However, it is prone to overestimation. For example, the BIC may predicted 3 clusters where the actual value of k is 2 (see Appendix Figure B.7). Figure 3.6 shows over-segregation, with waveforms suggesting two neurons, but the BIC groups them into three clusters. Note that the algorithm visualises and compares only one electrode signal.

3.5 Triangulation

If the clustering process succeeds in grouping waveforms by their source neurons, we can determine the position of these neurons relative to the electrodes. This process is more appropriately called trilateration rather than triangulation, as it relies on distances rather than angles to pinpoint a neuron's location.

The spikes generated by each predicted neuron must be isolated for each electrode recording and combined to create an average waveform. This process would yield eight average waveforms if there were four electrodes and two predicted neurons. This step aims to determine the average signal strength from the neurons at each electrode. This is not as simple as comparing the membrane potential at the aggregated peaks, as the resting potential of each signal has also been affected by attenuation. Instead, the difference in potential between the resting state and the peak for each aggregated waveform serves as a measure of its signal strength.

When using actual neural signal datasets, the position and original strength of the neurons' signals are unknown. Therefore, this information will not be used when replicating the process with simulated data.

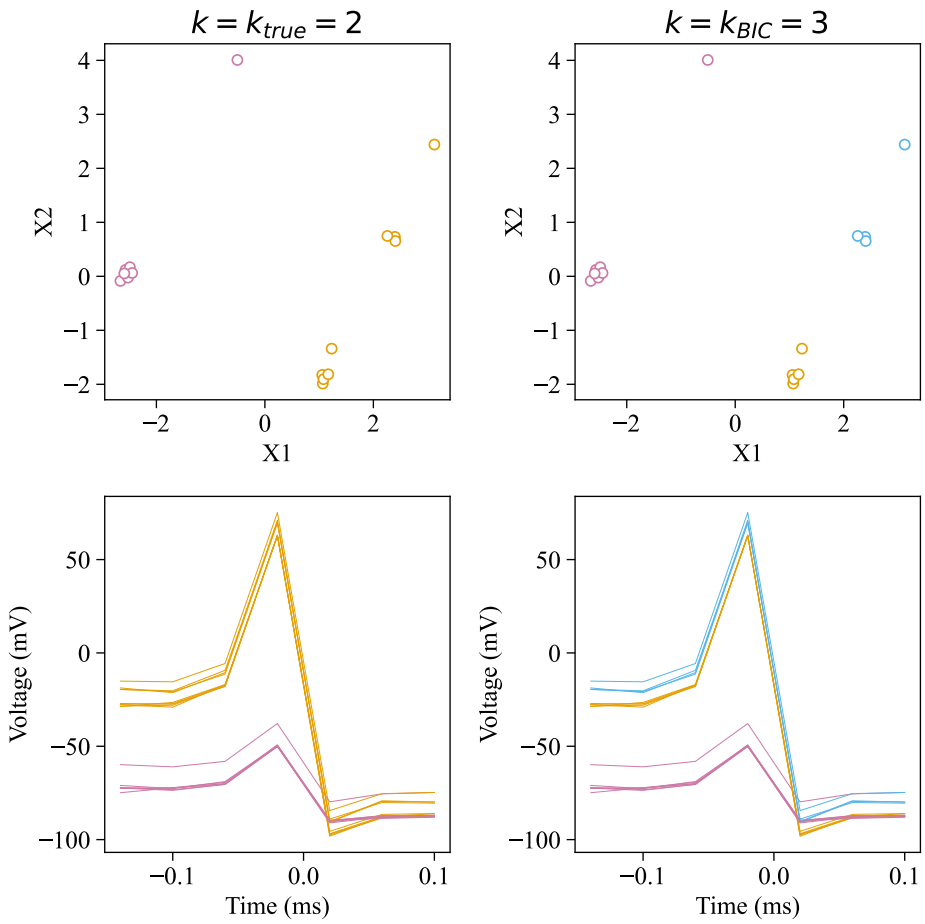


Figure 3.6: Two-dimensional representation of vectorised waveforms (upper) and waveform plots (lower). Colours indicate the clusters assigned to each waveform by a GMM clustering algorithm. The left panel shows clustering with the correct number of clusters, $k = k_{true} = 2$, while the right panel shows clustering with the number of clusters estimated by the BIC, $k = k_{BIC} = 3$.

This triangulation process must be done for each neuron individually. First, determine the three electrodes with the strongest signals, as they will provide the highest signal-to-noise ratio (SNR). In Figure 3.7, these electrodes are labelled B, A, and C in order of decreasing signal strength.

Then, use the attenuation rule (in this case, the inverse square law) to determine the relative distance between the neuron and the electrodes. Ignoring the noise term, Equation 3.4 becomes:

$$y_{2_A} = y_{1_A} \frac{k}{x_A^2}, \tag{3.5}$$

where y_{1_A} represents the strength of the signal emitted by the neuron, y_{2_A} is the signal strength recorded by the electrode A, k is a parameter representing the transmissibility of the medium and x_A is the distance between electrode A and the neuron. Both x_A and y_{1_A} are unknown.

Since the emitted signal strength from the neuron is constant regardless of the electrode ($y_1 = y_{1_A} = y_{1_B}$), we can use the signal strength ratios to estimate the distance ratios between the neuron and the electrodes. The equation for the relative distance ratio for electrodes A and B is:

$$\frac{x_A}{x_B} = \sqrt{\frac{y_{2_B}}{y_{2_A}}}. \tag{3.6}$$

A line perpendicular to the one connecting the two electrodes must exist where any point along it satisfies this ratio. For example, if $y_{2_A} = 4$ and $y_{2_B} = 9$ (a.u.), then $x_A = \frac{3}{2} x_B$. This implies that the line must be $\frac{3}{2}$ times further away from electrode A than B, placing it at a distance of $\frac{3}{5} d_{AB}$ from electrode A, as seen in Figure 3.7.

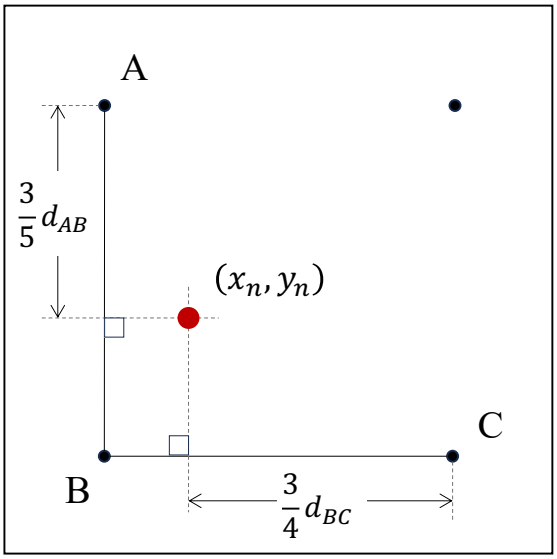


Figure 3.7: An example layout of a four-electrode array and the construction used to locate the position of a neuron. The average signal strengths of the neuron on each electrode are $y_{2_A} = 4$, $y_{2_B} = 9$ and $y_{2_C} = 1$ (a.u.).

We then repeat this step between another pair of electrodes, for example, between B and C. If $y_{2_C} = 1$ (a.u.), then $x_C = 3x_B$ placing a line at a distance $\frac{3}{4}d_{BC}$ from electrode C, where d_{BC} is the distance between neurons B and C. The intersection point of the two lines is the estimated position of the neuron, denoted as (x_n, y_n) in cartesian coordinates. In Figure 3.7, the neuron is represented by a red dot at position $(x_n, y_n) = (x_b + \frac{2}{5}d_{AB}, y_b + \frac{1}{4}d_{BC})$.

In summary, this chapter has explained the development process behind the preliminary spike sorting pipeline and demonstrated its capabilities. A modular codebase abstracting techniques into swappable blocks was constructed to enable rapid workflow iteration. However, further abstraction and testing could ready the pipeline for open-source publishing. Regardless, this codebase can be converted into an API, enabling the development of an interactive web-based spike sorting demonstration.

Chapter 4

Evaluating the Web Application

This chapter evaluates the interactive web-based spike sorting demonstration. First, Section 4.1 outlines the implementation process, describing how the modular Python pipeline was adapted into an API that communicates with the frontend. Section 4.2 details the rationale behind key design decisions. Subsequently, Section 4.3 analytically examines the site sections representing each spike sorting stage, identifying enhancements to improve the outcome. Finally, Section 4.4 assesses the web application's performance. This evaluation aims to critique the demonstration produced in this project and suggest how it can be refined into an optimal educational and experimental tool.

4.1 Implementation

The Python code implementing each pipeline step was restructured into a Flask [27] application programming interface (API). Flask is a popular framework for building REST APIs due to its simplicity and thorough documentation. The frontend web application was developed using Next.js [82], a React-based framework that supports both server-side rendering (SSR) and client-side rendering (CSR). The SSR feature was not used much due to the interactive nature of the application.

The Next.js frontend was written in TypeScript [46], a strongly typed JavaScript-based language. The Tailwind CSS framework [38], which provides utility classes to enable the fast development of responsive designs, was used for styling the site. Vercel, Next.js's parent organisation, provides a cloud platform specialised for Next.js applications built atop Amazon Web Services (AWS), meaning it is highly reliable [4]. This enabled free and efficient deployment of the frontend, though the Flask backend API has not yet been deployed due to time and resource constraints.

In short, the technology stack used to implement the application was a reasonable choice. Focusing on type safety, responsiveness, and cloud-based deployment aims to provide better accessibility to users and simplicity for future developers who may wish to pursue this project further.

4.2 Design Principles

The primary goal of the interactive visualisation is to serve as an educational demonstration for spike sorting, enabling users to experiment with parameters and techniques. It is important that this visualisation is accessible to as many people as possible, increasing the rate of learning in the field and, as a consequence, progressing it further. To succeed, the demonstration needs to be well-designed. Several design principles have been put in place to address this.

Emphasising Distinct Steps

The interface reinforces that spike sorting comprises sequential, discrete steps. Users navigate between stages using buttons and scrolling "snaps" to each step rather than scrolling continuously. The interface also prevents advancing until the prior stage is completed, conveying the hierarchical nature of the process (Appendix Figure C.1). This structured presentation aims to methodically guide users through the stages while preventing confusion regarding the order and purpose of each step.

Consistency

A consistent visual interface reduces cognitive load, enabling users to focus on relevant content. This follows the Gestalt principle of "common fate", where consistent elements are perceived as belonging together [42, 89]. For example, buttons that generate data share common stylings like borders and padding, making their purpose identifiable without instruction. In contrast, parameter selection buttons are identifiable as simple underlined text. When a method is not yet implemented, it is presented with crossed-out text to manage user expectations. Similarly, the presence of a dashed line indicates a scrollable element due to the purposeful overflowing of content from the screen, gesturing the user to explore further. The extent to which these components are consistent is visible in Appendix Figure C.2.

Simplicity in colour reinforces consistency; most elements are black on a white background, resulting in a luminance contrast exceeding 3:1 [87]. Items of less importance are muted with less contrast to the background to attract minimal attention [19]. The colours red and blue are reserved for denoting the positions of neurons and electrodes, respectively.

Each visualisation aligns with Munzner's task taxonomy, ensuring users extract the intended information [50]. The interface consistency aims to direct user attention and streamline information delivery, although further studies are needed to quantify the benefits for learner comprehension.

Interactivity

Interactivity is the primary method of delivering the desired user experience, allowing users to change parameters whilst getting instant feedback on their choices. This interactive environment also allows users to request "details on demand" [69]. For instance, sometimes, users can toggle views to extract complementary insights from the data (see Append Figure C.3). Although necessary constraints are imposed in this demonstration, such as the ranges of parameter values, interactivity is important as it makes the user feel as if they are in control. Further studies could quantify the learning gains enabled by interactivity versus passive observation in the spike sorting process. Tracking user interactions may reveal common exploration patterns and inform future design iterations.

Device Compatibility

The responsiveness of a site relates to its ability to change and function when viewed on devices with different screen sizes, as illustrated by the example website layout in Appendix Figure B.8. Responsiveness is important in this case, as one of the project's goals is accessibility, and with ∼51% of mobile users relying solely on phones for internet access [45], catering to this audience is essential. Preliminary testing on simulated (Google's Pixel 5) and physical (Apple's iPhone X) mobile devices confirmed horizontal responsiveness. However, comprehensive testing across diverse form factors was beyond this project's timeframe. Hence, vertical responsiveness for short screens may need improvement to maintain usability.

The performance testing on each device was informal and qualitative. Comprehensively quantifying a device's compatibility requires measuring refresh rates, load times, and hardware utilisation. Enabling equitable access for users on commodity hardware through comprehensive accessibility testing and subsequent optimisation would be an important avenue for future development.

4.3 The Spike Sorting Demonstration

Each step in the spike sorting process is represented in this web demonstration. The development process is omitted since only the end product matters to users, unlike the Python pipeline, where the code itself is the final outcome. This chapter presents screenshots of each site section as displayed on a mobile interface. Appendix C offers visuals of site sections across both larger screens and mobile devices.

4.3.1 Simulation

The simulation interface consists of four stages: neuron/electrode placement, neuron parameter selection, medium property configuration and signal generation.

Placement Grid

Users position electrodes and neurons on an adjustable grid through touch-based controls (Figure 4.1a). The user must place at least three electrodes and one neuron to progress - the minimum amount required

for triangulation. The grid is scalable from 4-by-4 to 12-by-12. The squares of finer meshes are too small for a user to select with their finger on a small mobile device. This system is functional, but there are fringe cases that cause errors. For example, placing the three electrodes in a straight line will not cause an immediate error but will cause issues for downstream tasks. Similarly, placing the neuron outside a triangle formed by any three electrodes will cause the triangulation algorithm (see Section 3.5) to fail - although this is a fundamental condition of triangulation. Future improvements should validate electrode and neuron formation and notify the user if otherwise.

Neuron Parameters

This section of the application allows the user to select neuron types and modify three key parameters: firing rate, resting voltage, and threshold voltage (Figure 4.1b). Others, like the refractory period, are not available. This leaves only the essential parameters for users to choose between, reducing the likelihood of them becoming overwhelmed. Fixing the random seed enables isolated testing of individual parameters, allowing users to determine the purpose of each.

To cater to a larger audience, it would be helpful to have two modes for this demonstration - one for those learning about the spike sorting process and the other for those more advanced with more available parameters allowing for true experimentation. Further iterations should expand neuron models beyond the LIF version and incorporate various neuron behaviours, like bursting, to better model neural relationships.

The Connecting Medium

Once the neuron parameters are confirmed, the simulation matrix's properties can be selected. As discussed in Section 3.2.2, this includes the type and amount of noise and attention experienced by the simulated signal. Filtering a signal to remove noise and emphasise the spikes is a preprocessing step, however, it was included in this demonstration section so the relationship between the noise and filter can be visualised. It is uncertain if this alteration in the spike sorting process would benefit learning or cause confusion. Future user testing could determine the productivity of this outcome. This demonstration section contains numerous variables, leading to a cluttered page. Which, in turn, diminishes clarity and reduces the size of the visualisation shown at the bottom of Figure 4.1c. This is not an issue when using a larger monitor, but perhaps, on mobile, this necessitates design changes like hiding settings or multi-page layouts. A/B testing UI variations could optimise usability.

Generation

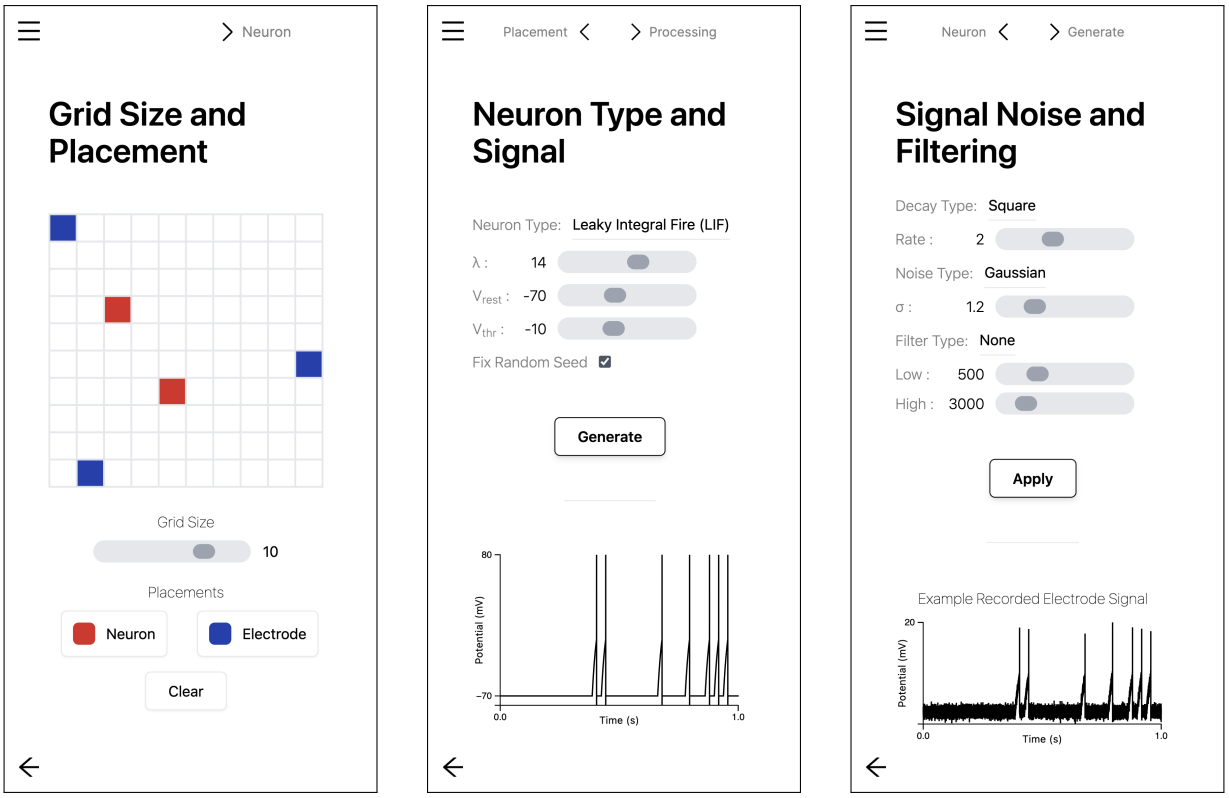
This simple section contains a button which initiates the recording simulation. Depending on the size and the number of components placed on this grid, this can take up to one minute. A loading animation is presented to the user, giving them visual feedback to confirm the operation. However, a user's concentration can drift during this period. Potential solutions include server-side processing, offloading operations to client web workers, precomputing or code optimisation. The first two solutions require capital for computational resources; the latter would entail a rewrite of the codebase.

This section acts as the bridge between the simulation's initialisation and the pipeline's outcome. Some text on the screen briefly explains the processes the user has completed and what is to come (see Appendix Figure C.8). This text also intends to occupy the user while awaiting the API's response.

In summary, the simulation section is functional, responsive and interactive, allowing for the visual testing of parameters. Enabling such real-time experimentation relies on the spike sorting pipeline's modularity with interchangeable algorithmic blocks. However, some blocks, like noise and attenuation, are less customisable. If this project were to continue past its current scope, further abstraction at the API level would be beneficial as it could improve extensibility.

4.3.2 Preprocessing

The application handled the filtering step when configuring the connecting medium in Section 4.3.1. Therefore, preprocessing consists of only two sections: spike and feature extraction. Section 3.3 details the underlying techniques used in these steps. These stages show users how to parse signals to isolate APs and how parameters influence these processes.



(a) (b) (c)

Figure 4.1: Screenshots from a mobile device displaying the grid layout (a), neuron parameters (b), and attenuation, noise, and filtering sections (c).

Electrode Recordings

The simulation concludes by visualising the electrode recordings generated based on configured parameters (see Figure 4.2a). As each recording is one second long, simulated at 25kHz, there can be over 75,000 data points on the screen at any one time, which may result in lag. This step is not strictly necessary, but it was implemented to enable the user to compare the recordings from each electrode more easily.

Spike Extraction

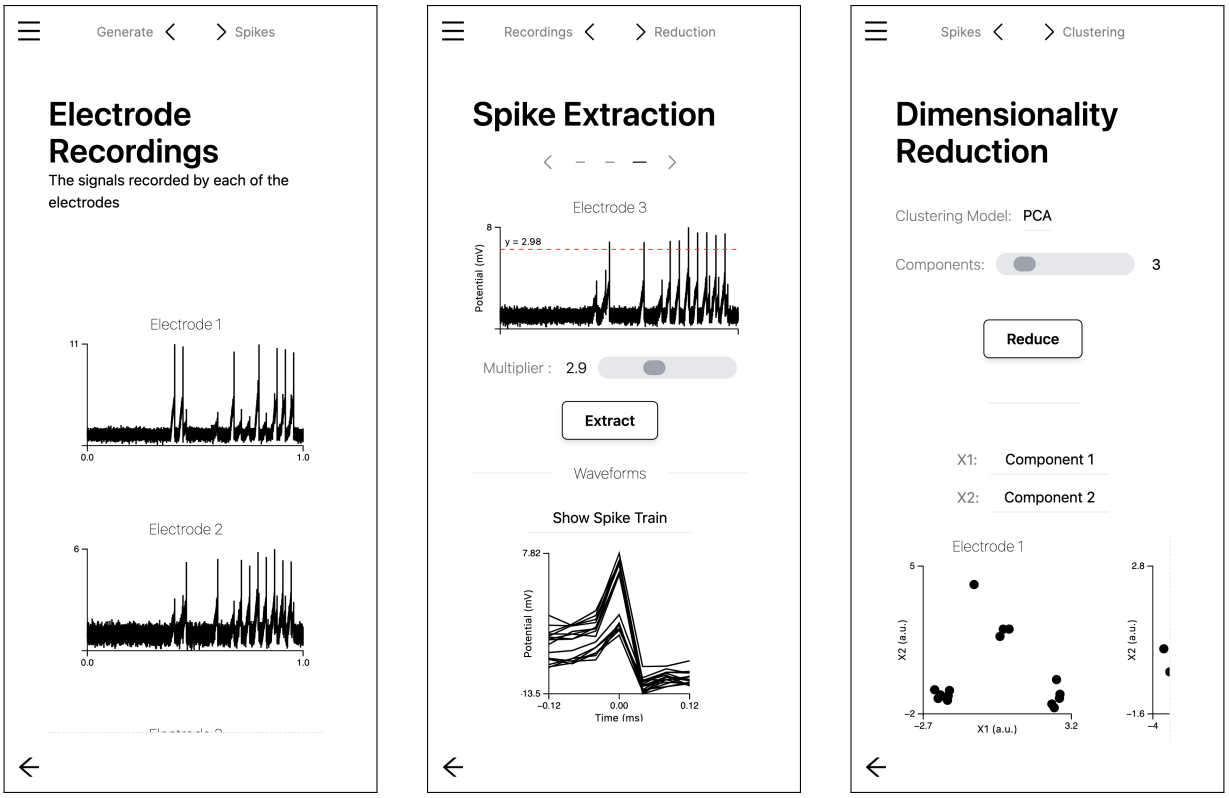
The section of the site aptly titled "Spike Extraction" lets users toggle between electrode signals to test threshold multipliers, visualising detected spikes. Although there are other methods of determining the spike locations, this demonstration applies the thresholding method using the MAD (see Section 3.3.2). The decision was made to permit only this method to maximise simplicity and reduce confusion, as other methods would require a more complex UI. However, this comes at the cost of limiting experimentation.

Users can extract and visualise waveforms as overlaid plots or spike trains, toggling between views using the button shown in Figure 4.2b.

Feature Extraction

Once satisfied with the extracted waveforms, users can vectorise features via dimensionality reduction. Only the PCA model is available, chosen over UMAP for its prevalence in the existing literature. More complex deep learning methods, like Autoencoders or CNNs, are unavailable. This is because Python packages used to implement these methods, such as PyTorch [61] or TensorFlow [1], would be expensive to host online due to their size and computational requirements. However, given adequate resources, inclusion of such methods could be valuable future work.

Users choose the number of dimensions for reduction. Since high-dimensional data is hard to visualise, two components representing the x- and y-axes can be selected. Each electrode produces one plot. The HTML container is scrollable when multiple exist (see Figure 4.2c).



(a) (b) (c)

Figure 4.2: Screenshots from a mobile device displaying the raw electrode recording (a), spike extraction (b), and feature extraction sections (c).

More interactivity, such as inspecting specific spike features, could improve these sections by providing further "details on demand" [69]. This feature has not been implemented for two main reasons. First, the project's timeframe did not accommodate the development of such an advanced feature. Second, achieving the desired control over visualisation typically necessitates using a lower-level language tailored for a specific OS, as in native applications. Given that web applications rely on higher-level languages like TypeScript and must be compatible across various devices and browsers, such control is unattainable.

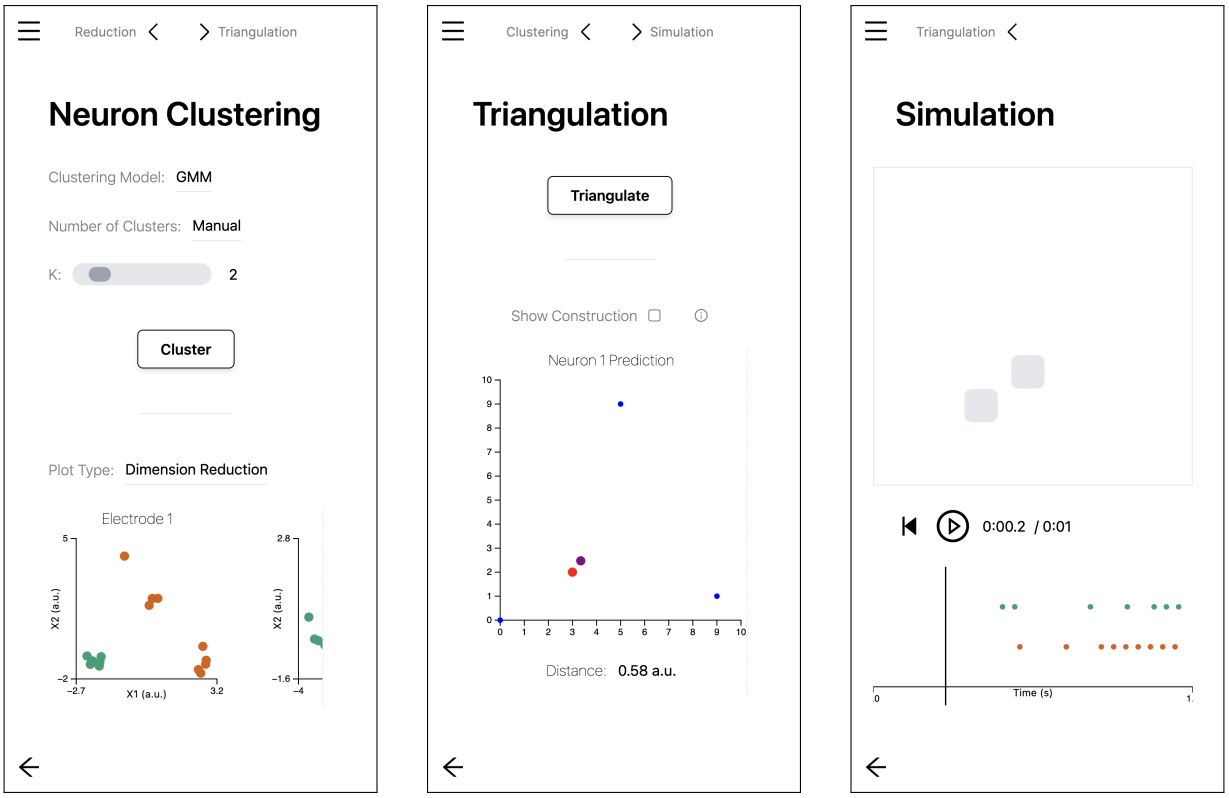
4.3.3 Clustering

The clustering section allows users to group the vectorised waveforms by their predicted neuron of origin. Currently, only Gaussian Mixture Models (GMM) are implemented, but the API's modularity enables adding alternatives like the "Drifting Mixture of t-Distributions" (see Section 2.3.3) [68]. The number of clusters, k, can be manually set or estimated via the BIC. However, as the BIC is prone to overestimation (see Section 3.4), future work could improve this heuristic. Figure 4.3a illustrates clustering with $k = 2$. The outcome of clustering is depicted with colour, and three different views are available via a toggle: dimensionality reduced scatter, waveform or spike train plot, each of which can be seen in Appendix Figure C.4.

The clustering performance metrics, which depend on earlier steps, are not available to the user. This is because these metrics require an equal number of true labels and predicted labels to calculate the classification quality, and depending on the prior procedures, this may not be the case. However, ensuring this condition is always met could be means for future development.

4.3.4 Triangulation

This section of the site is simple. As shown in Figure 4.3b, there are no parameters for the user to choose from, as there is only a single implemented method, which is discussed in Section 3.5. Comparing the robustness of alternative methods that predict the neurons' position would be an interesting direction for future studies. The electrodes and neurons are coloured blue and red, respectively, keeping consistent



(a) (b) (c)

Figure 4.3: Screenshots from a mobile device displaying the clustering (a), triangulation (b), and simulation player sections (c).

with their placements at the start of the demonstration. The purple marks represent the predicted neuron position. Users can toggle construction lines used in the underlying algorithm, similar to Figure 3.7.

If there is no noise, and all prior spike sorting steps are completely accurate, the positions of the true and predicted neurons are identical. That being said, it is straightforward for a user to assess the quality of the entire spike sorting process from the visual discrepancy between the red and purple marks. The Euclidean distance between them, shown below the graph in Figure 4.3b, quantifies this difference. This distance measure is undefined if the predicted neurons does not exist.

It should be noted that the graph displaying the triangulation is an inverted version of the grid at the start of the demonstration (see Figure 4.1a). This is because nested array indexing for the grid starts at the top, unlike the origin of the graphs in Figure 4.3b.

4.3.5 Simulation Player

In this section of the application, the user can visualise the simulation. Its purpose is to demonstrate the spatial and temporal neural population dynamics, providing context to the prior steps. It uses CSR without API communication, limiting recordings to one second (25 kHz sample rate) to minimise performance issues on basic hardware. As shown in Figure 4.3c, estimated neuron positions are denoted by squares that flash red during spikes. The one-dimensional spike train at the bottom of the interface displays spike timings, grouping the spikes by their predicted source neuron using the colours produced in the clustering step (see Section 4.3.3).

This rudimentary system hints at future potential for more sophisticated neural behaviour simulation. Advancing the simulation player could be an entire project in itself, especially with complex spiking models.

In summary, the web demonstration represents all spike sorting steps, allotting appropriate weight and logical flow to each section. While simple and requiring refinement to maximise its impact as a tool for teaching and experimentation, the existing framework provides a starting point for user testing.

4.4 Performance

The performance of the spike sorting process is no real concern to this project, as its desired outcome is a set of spike sorting tools; hence, a successful project is one that provides a solid foundation on which these tools can be extended. There are integrated methods for the users themselves to determine the quality of their constructed pipeline, such as the distance between the true and predicted neuron positions.

On the other hand, the performance of the frontend application is not within the users' control, and even though this project is producing a demonstration, its quality should be assessed to show areas needing improvement.

Website performance, influencing search engine optimisation (SEO), can be measured with Google's Web Vitals [21] using open source tools like Unlighthouse [91]. While SEO is unimportant here, the underlying metrics reflect user experience. When combined, the Core Web Vitals, consisting of the Largest Contentful Paint (LCP), the First Input Delay (FID), and the Cumulative Layout Shift (CLS), capture loading, interactivity, and visual stability. The LCP measures the time to display key content. The FID quantifies the time elapsed between the initial interaction and the start of processing. CLS tracks visual shifts as elements load. Poor scores often indicate usability issues, guiding iterative improvements to the demonstration. Other, more subjective factors, like aesthetics and ease of use, impact user engagement and educational value. These aspects could be assessed in the future by conducting user studies on the demonstration.

The application received perfect "Best Practices" and "Accessibility" scores from the Unlighthouse report by following security guidelines and inclusive design principles. It also boasts a perfect "SEO" score with excellent LCP, FID, and CLS metrics at 100ms, 8ms, and 0.06 (1 s.f.), respectively. Appendix Figure C.15 shows the reports awarding these scores.

However, "Performance" only scored 81% due to a high total blocking time of 790ms stemming from the large 5,041 KiB network payload. Currently processed client-side, this strains users with limited bandwidth. Offloading computation to a server and using SSR could alleviate demands on entry-level hardware. Identifying performance shortcomings highlights opportunities to optimise the learning experience through technical improvements.

While browser-based testing provides insights into the application's performance, it doesn't capture its functionality across a diverse range of devices. Currently, this comprehensive testing is not feasible since the API is not hosted online. Future efforts could delve into metrics like device frame rate and hardware strain while running the application. Such in-depth testing is necessary to truly gauge the demonstration's accessibility.

This critical evaluation has reviewed the web-based spike sorting application's design, demonstrative, and performance factors. While the interface suitably represents each spike sorting stage, current functionality remains limited in scope. Achieving the full vision as an educational and experimental tool will require refinements identified through rigorous user testing. In conclusion, ongoing iterative improvement guided by such analytical assessment will further enhance these tools, bolstering productivity in spike sorting.

Chapter 5

Future Work

This project establishes a foundation ripe for extension in several directions. This section proposes high-potential avenues to build upon the outputs of this project and alternative routes that may yield interesting and useful results.

Enhancing the Python Pipeline

Hosting the API on a cloud or local server would enable online availability and allow more complex algorithms by increasing computational capacity. Although simple, this requires capital investment. Additional improvements include refining component algorithms, incorporating 3D neuron placement modelling (more accurately representing the brain), and implementing the volume conductor theory to model signal attenuation (see Section 2.2.3) [26].

The pipeline requires further abstraction and rigorous software testing before open-source distribution. These enhancements facilitate community involvement in maintaining and expanding functionality.

Optimising the Web Demonstration

While functional, the interface may overwhelm users unfamiliar with spike sorting. Integrating more scaffolded explanations and contextual parameter details could aid novice learners. Including a greater number of models and parameters would provide a more comprehensive testing environment. However, too many options may overstimulate and confuse less advanced users. Two modes could be created: one for more advanced users, tailored towards experimentation, and the other more suited as a learning resource. However, the efficacy of these improvements, and many other aspects of the application, can be determined through extensive user testing (such as A/B testing), further guiding the future development of this project.

Alternative Approaches

During this project, opportunities arose to follow different avenues within the realm of spike sorting, which may have led to interesting conclusions.

Processing a live stream of spike data would require reinventing the pipeline for low-latency performance. Closed-loop experiments can be used to analyse pre- and post-synaptic neuron activity [22]. Replacing each pipeline step with deep neural networks [44, 86] and comparing their performance with more conventional techniques is another promising direction.

The triangulation method used in this project is limited to identifying only a few simulated neurons located in close proximity. It would be advantageous if a similar algorithm could be scaled effectively to locate the neurons detected by significantly more electrodes across a wider area, perhaps from devices such as the high-density MEA.

Although touched upon in the demonstration (Section 4.3.5), a playable simulation where users could move around components and determine their effects on surrounding neurons could provide insight into the temporal and relational dynamics within neuron populations.

To truly transform this process into a foundational resource, a public vetted vector database like Pinecone [64] or Chroma [15] would be invaluable. Such a database, storing vast amounts of vectorised spike data, could significantly accelerate technical advancements in the field, thanks to the availability of reliable data from diverse sources.

Chapter 6

Conclusion

The project succeeded in constructing a modular spike sorting pipeline, enabling rapid adaptation of algorithms using interchangeable blocks. Each step incorporates techniques standard in the literature, with ample flexibility for customisation and expansion. Standardised data structures between blocks empower seamless integration. This represents a substantial efficiency gain over developing such algorithms from scratch. Despite these achievements, several loose ends exist due to the limited time allotted for this project. The code requires further refinement and testing before an open-source public release. Some of the code is not abstracted to the level where it is ready for expansion, and many of the algorithms require rigorous testing. However, these goals are obtainable if this project were to be continued.

Some pipeline components were adapted into a Flask API, enabling the development of an interactive web visualisation using the Next.js framework. This application allows hands-on experimentation with pipeline parameters and algorithms to support active learning. Although the frontend is available online, it has not been possible to host the Python API due to capital constraints. Almost any device with access to the web could run this application due to its responsive layout and capability to run client-side processes on commodity hardware, meaning it would be a highly accessible resource if it were hosted. Despite this, it is operational in a development environment (like localhost) and meets the objective of a no-code demonstration for experimenting and testing various spike sorting pipelines. This web application could serve as a learning resource for those new to spike sorting, teaching through visuals and experimentation rather than the written word. With further refinement, the tool could allow non-technical domain experts to construct customised pipelines through a non-technical interface. Removing reliance on technical specialists would democratise innovation. Currently, this is not the case, so such an approach would remove one of the field's most considerable barriers to entry.

The outcomes of this project have the potential to impact both the scientific community and broader society positively. Within academia, developing more efficient spike sorting tools could accelerate discoveries and innovations in neuroscience. As researchers gain the ability to test techniques and construct custom pipelines swiftly, the rate of experimentation and hypothesis testing may rapidly increase. This promises new insights into neuronal encoding and communication, enriching our understanding of neuroscience. Beyond academia, enhancing spike sorting capabilities can ultimately lead to real-world advancements that improve quality of life. With greater knowledge of neural dynamics, brain-computer interfaces may be refined, restoring communication or mobility for paralysed individuals. Improved comprehension of neurological disorders could enable earlier diagnosis of diseases like Alzheimer's. The project's focus on accessibility aims to attract talent from diverse backgrounds, driving further innovation.

The next steps involve preparing the pipeline and web demonstration for public release, which includes finalising modules for open-source distribution on GitHub and conducting user studies to optimise the application as an educational tool.

By establishing a flexible spike sorting infrastructure, this project has laid the groundwork for increased productivity in neuroscience. With these foundational resources now in place, collaborative innovation can flourish, leading to the development of new technologies and therapies that enhance the quality of human life.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: https://www.tensorflow.org/.

[2] Addyosmani. Web vitals: Measure metrics for a healthy site. Chrome Web Store, 2023. Accessed: August 8, 2023. URL: https://chrome.google.com/webstore/detail/web-vitals/ahfhijdlegdabablpippeagghigmibma.

[3] Zane N. Aldworth, John P. Miller, Tomas Gedeon, Graham I. Cummins, and Alexander G. Dimitrov. Dejittered spike-conditioned stimulus waveforms yield improved estimates of neuronal feature selectivity and spike-timing precision of sensory interneurons. *The Journal of Neuroscience*, 25:5323 – 5332, 2005. URL: https://api.semanticscholar.org/CorpusID:9397038.

[4] Inc. Amazon Web Services. Availability, 2023. Documentation on the AWS Well-Architected Framework discussing the concept of availability, its metrics, and its significance in system design. URL: https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/availability.html.

[5] E.-R. Ardelean, A. Coporîie, A.-M. Ichim, M. Dînșoreanu, and R. C. Mureșan. A study of autoencoders as a feature extraction technique for spike sorting. *PLoS ONE*, March 2023. Retrieved June 18, 2023. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0282810.

[6] FA Azevedo, LR Carvalho, LT Grinberg, JM Farfel, RE Ferretti, RE Leite, W Jacob Filho, R Lent, and S Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *J Comp Neurol*, 513(5):532–541, Apr 10 2009. doi:10.1002/cne.21974.

[7] David Beeman. *Hodgkin-Huxley Model*, pages 1–13. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4614-7320-6_127-3.

[8] Jan Benda. Neural adaptation. *Current Biology*, 31:R110–R116, 02 2021. doi:10.1016/j.cub.2020.11.054.

[9] Philipp Berens and Alexander Ecker. Dataset for the lecture "Neural Data Science" at University of Tübingen, April 2021. doi:10.5281/zenodo.4704658.

[10] A.P. Buccino and G.T. Einevoll. Mearec: A fast and customizable testbench simulator for ground-truth extracellular spiking activity. *Neuroinform*, 19:185–204, 2021. doi:https://doi.org/10.1007/s12021-020-09467-7.

[11] G Buzsáki, CA Anastassiou, and C Koch. The origin of extracellular fields and currents–eeg, ecog, lfp and spikes. *Nat Rev Neurosci*, 13(6):407–420, May 2012. doi:10.1038/nrn3241.

[12] Ana Calabrese and Liam Paninski. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of Neuroscience Methods*, 196(1):159–169, 2011. doi:https://doi.org/10.1016/j.jneumeth.2010.12.002.

[13] Stephen Cass. Top programming languages 2022. *IEEE Spectrum*, 2022. Accessed: August 8, 2023. URL: https://spectrum.ieee.org/top-programming-languages-2022.

[14] E Chah, V Hok, A Della-Chiesa, J J H Miller, S M O'Mara, and R B Reilly. Automated spike sorting algorithm based on laplacian eigenmaps and k-means clustering. *Journal of Neural Engineering*, 8(1):016006, 2011. doi:10.1088/1741-2560/8/1/016006.

[15] Chroma. Chroma: the ai-native open-source embedding database. Website, 2023. Accessed: August 24, 2023. URL: https://www.trychroma.com.

[16] Microsoft Corporation. Visual studio code, 2023. URL: https://code.visualstudio.com/.

[17] Roland Diggelmann, Michele Fiscella, Andreas Hierlemann, and Felix Franke. Automatic spike sorting for high-density microelectrode arrays. *Journal of Neurophysiology*, 120(6):3155–3171, 2018. PMID: 30207864. doi:10.1152/jn.00803.2017.

[18] Grégory Dumont, Alexandre Payeur, and André Longtin. A stochastic-field description of finite-size spiking neural networks. *PLOS Computational Biology*, 13(8):1–34, 08 2017. doi:10.1371/journal.pcbi.1005691.

[19] H. J. Eysenck. A critical and experimental study of colour preferences. *The American Journal of Psychology*, 54(3):385–394, 1941. doi:10.2307/1417683.

[20] Facebook. React - a JavaScript library for building user interfaces, 2023. An open-source JavaScript library for building user interfaces or UI components. URL: https://reactjs.org/.

[21] Google for Developers. Understanding Core Web Vitals and Google search results, May 2023. Accessed: August 24, 2023. URL: https://developers.google.com/search/docs/appearance/core-web-vitals.

[22] Felix Franke, David Jäckel, Jelena Dragas, Jan Müller, Milos Radivojevic, Douglas Bakkum, and Andreas Hierlemann. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Frontiers in Neural Circuits*, 6, 2012. doi:10.3389/fncir.2012.00105.

[23] Negar Geramifard, Jennifer Lawson, Stuart F. Cogan, and Bryan James Black. A novel 3d helical microelectrode array for in vitro extracellular action potential recording. *Micromachines*, 13(10), 2022. URL: https://www.mdpi.com/2072-666X/13/10/1692, doi:10.3390/mi13101692.

[24] Felipe Gerhard, Gordon Pipa, Bruss Lima, Sergio Neuenschwander, and Wulfram Gerstner. Extraction of network topology from multi-electrode recordings: Is there a small-world effect? *Frontiers in Computational Neuroscience*, 5, 2011. doi:10.3389/fncom.2011.00004.

[25] GitHub. GitHub, 2020. URL: https://github.com/.

[26] Pierre Gloor. Neuronal generators and the problem of localization in electroencephalography: Application of volume conductor theory to electroencephalography. *Journal of Clinical Neurophysiology*, 2:1–15, 1985. URL: https://www.fuw.edu.pl/~suffa/SygnalyBioelektryczne/Gloor_Neural_Generators_JClinNeurphysiol_1985.pdf.

[27] Miguel Grinberg. *Flask web development: developing web applications with Python*. O'Reilly Media, Inc., 2018.

[28] Umut Güçlü and Marcel A. J. van Gerven. Modeling the dynamics of human brain activity with recurrent neural networks. *Frontiers in Computational Neuroscience*, 11, 2017. doi:10.3389/fncom.2017.00007.

[29] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2.

[30] Samuel Harris. Data science project API GitHub repository, 2023. URL: https://github.com/SamwelZimmer/DataScienceProjectAPI.

[31] Samuel Harris. Data science project frontend GitHub repository, 2023. URL: https://github.com/SamwelZimmer/DataScienceProjectFrontend.

[32] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–44, Aug 1952. doi:10.1113/jphysiol.1952.sp004764.

[33] W.R. Holmes. *Cable Theory: Overview*, pages 1–2. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4614-7320-6_757-1.

[34] G. Hong and C.M. Lieber. Novel electrode technologies for neural recordings. *Nature Reviews Neuroscience*, 20:330–345, 2019. doi:https://doi.org/10.1038/s41583-019-0140-6.

[35] Chao Huang, Andrey Resnik, Tansu Celikel, and Bernhard Englitz. Adaptive spike threshold enables robust and temporally precise neuronal encoding. *PLOS Computational Biology*, 12(6):1–25, 06 2016. doi:10.1371/journal.pcbi.1004984.

[36] Libo Huang, Bingo Wing-Kuen Ling, Ruichu Cai, Yan Zeng, Jiong He, and Yao Chen. Wmsorting: Wavelet packets' decomposition and mutual information-based spike sorting method. *IEEE Transactions on NanoBioscience*, 18(3):283–295, 2019. doi:10.1109/TNB.2019.2909010.

[37] D.H. Hubel and T.N. Wiesel. Effects of monocular deprivation in kittens. *Naunyn-Schmiedebergs Archiv für experimentelle Pathologie und Pharmakologie*, 248:492–497, 1964. doi:https://doi.org/10.1007/BF00348878.

[38] Tailwind Labs Inc. Tailwind CSS: A utility-first CSS framework for rapidly building custom user interfaces, 2023. Accessed: August 24, 2023. URL: https://tailwindcss.com/.

[39] J. Jun, N. Steinmetz, J. Siegle, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551:232–236, 2017. doi:https://doi.org/10.1038/nature24636.

[40] E.R. Kandel, J.H. Schwartz, T.M. Jessell, S. Siegelbaum, A.J. Hudspeth, and S. Mack, editors. *Principles of Neural Science*. McGraw-hill, New York, Jan 2000.

[41] Kyung Hwan Kim and Sung June Kim. Neural spike sorting under nearly 0-db signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier. *IEEE Transactions on Biomedical Engineering*, 47(10):1406–1411, 2000. doi:10.1109/10.871415.

[42] Kurt Koffka. *Principles of Gestalt psychology*. Harcourt, Brace & World, New York, 1935.

[43] Christophe Leterrier. The axon initial segment, 50 years later: A nexus for neuronal organization and function. *Current topics in membranes*, 77:185–233, 2016. URL: https://api.semanticscholar.org/CorpusID:14028120.

[44] Z Li, Y Wang, N Zhang, and X Li. An accurate and robust method for spike sorting based on convolutional neural networks. *Brain Sci*, 10(11):835, Nov 11 2020. doi:10.3390/brainsci10110835.

[45] James McDonald. Almost three quarters of internet users will be mobile-only by 2025. *WARC*, 2019. Accessed: August 8, 2023. URL: https://www.warc.com/content/paywall/article/warc-curated-datapoints/almost-three-quarters-of-internet-users-will-be-mobile-only-by-2025/en-gb/124845.

[46] Microsoft. TypeScript: JavaScript with syntax for types. Website, 2023. Accessed: August 24, 2023. URL: https://www.typescriptlang.org.

[47] Leonid L. Moroz, Daria Y. Romanova, and Andrea B. Kohn. Neural versus alternative integrative systems: molecular insights into origins of neurotransmitters. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 376(1821):20190762, 2021. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2019.0762, arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rstb.2019.0762, doi:10.1098/rstb.2019.0762.

[48] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 13, 2019. URL: https://www.frontiersin.org/articles/10.3389/fnins.2019.00625, doi:10.3389/fnins.2019.00625.

[49] Berndt Müller, Joachim Reinhardt, and Michael T. Strickland. *Stochastic Neurons*, pages 38–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. doi:10.1007/978-3-642-57760-4_4.

[50] T. Munzner. *Visualization Analysis and Design*. AK Peters Visualization Series. CRC Press, 2014. URL: https://books.google.co.uk/books?id=dznSBQAAQBAJ.

[51] K. Musick, J. Rigosa, S. Narasimhan, et al. Chronic multichannel neural recordings from soft regenerative microchannel electrodes during gait. *Scientific Reports*, 5:14363, 2015. doi:https://doi.org/10.1038/srep14363.

[52] Samuel Nason, Alex Vaskov, Matthew Willsey, Elissa Welle, Hyochan An, Philip Vu, Autumn Bullard, Chrono Nu, Jonathan Kao, Krishna Shenoy, Taekwang Jang, Hun-Seok Kim, David Blaauw, Parag Patil, and Cynthia Chestek. A low-power band of neuronal spiking activity dominated by local single units improves the performance of brain–machine interfaces. *Nature Biomedical Engineering*, 4:1–11, 10 2020. doi:10.1038/s41551-020-0591-0.

[53] Richard Naud and Henning Sprekeler. Sparse bursts optimize information transmission in a multiplexed neural code. *Proceedings of the National Academy of Sciences*, 115(27):E6329–E6338, 2018. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1720995115, arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1720995115, doi:10.1073/pnas.1720995115.

[54] Lucas Hermann Negri and Christophe Vestri. lucashn/peakutils: v1.1.0, September 2017. doi:10.5281/zenodo.887917.

[55] Neuralink. Neuralink: Creating a generalized brain interface. Website, 2023. Accessed: August 24, 2023. URL: https://www.neuralink.com.

[56] Angela K. Nietz, Laurentiu S. Popa, Martha L. Streng, Russell E. Carter, Suhasa B. Kodandaramaiah, and Timothy J. Ebner. Wide-field calcium imaging of neuronal network dynamics in vivo. *Biology*, 11(11), 2022. URL: https://www.mdpi.com/2079-7737/11/11/1601, doi:10.3390/biology11111601.

[57] Sergey M. Olenin, Tatiana A. Levanova, and Sergey V. Stasenko. Dynamics in the reduced mean-field model of neuron-glial interaction. *Mathematics*, 11(9), 2023. URL: https://www.mdpi.com/2227-7390/11/9/2143, doi:10.3390/math11092143.

[58] Enwenode Onajite. Chapter 8 - Understanding sample data. In Enwenode Onajite, editor, *Seismic Data Analysis Techniques in Hydrocarbon Exploration*, pages 105–115. Elsevier, Oxford, 2014. doi:https://doi.org/10.1016/B978-0-12-420023-4.00008-3.

[59] OpenStax. *How Neurons Communicate*, chapter 35.2. OpenStax, Rice University, n.d.

[60] Overleaf. Overleaf, 2023. URL: https://www.overleaf.com/.

[61] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Workshop Autodiff*, pages 336–343, 2017. URL: https://openreview.net/forum?id=BJJsrmfCZ.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[63] R. Perrin, A. Fagan, and D. Holtzman. Multimodal techniques for diagnosis and prognosis of Alzheimer's disease. *Nature*, 461:916–922, 2009. doi:10.1038/nature08538.

[64] Inc. Pinecone Systems. Pinecone: Vector database for vector search. Website, 2023. Accessed: August 24, 2023. URL: https://www.pinecone.io.

[65] George Edgin Pugh. *The Biological Origin of Human Values*. Basic Books, New York, 1977.

[66] Patrick Ruther and Oliver Paul. New approaches for CMOS-based devices for large-scale neural recording. *Current Opinion in Neurobiology*, 32:31–37, 2015. Large-Scale Recording Technology (32). doi:https://doi.org/10.1016/j.conb.2014.10.007.

[67] Melinda Rácz, Csaba Liber, Erik Németh, Richárd Fiáth, János Rokai, István Harmati, István Ulbert, and Gergely Márton. Spike detection and sorting with deep learning. *Journal of Neural Engineering*, 17(1):016038, Jan 2020. doi:10.1088/1741-2552/ab4896.

[68] Kevin Q. Shan, Evgueniy V. Lubenov, and Athanassios G. Siapas. Model-based spike sorting with a mixture of drifting t-distributions. *Journal of Neuroscience Methods*, 288:82–98, 2017. doi:https://doi.org/10.1016/j.jneumeth.2017.06.017.

[69] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996.

[70] Shy Shoham, Matthew R. Fellows, and Richard A. Normann. Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of Neuroscience Methods*, 127(2):111–122, 2003. doi:https://doi.org/10.1016/S0165-0270(03)00120-1.

[71] Ilias Sourikopoulos, Sara Hedayat, Christophe Loyez, François Danneville, Virginie Hoel, Eric Mercier, and Alain Cappy. A 4-fJ/spike artificial neuron in 65 nm CMOS technology. *Frontiers in Neuroscience*, 11, 2017. doi:10.3389/fnins.2017.00123.

[72] Martin Splitt. Introducing INP to Core Web Vitals, May 2023. URL: https://developers.google.com/search/blog/2023/05/introducing-inp.

[73] Kyle H. Srivastava, Caroline M. Holmes, Michiel Vellema, Andrea R. Pack, Coen P. H. Elemans, Ilya Nemenman, and Samuel J. Sober. Motor control by precisely timed spike patterns. *Proceedings of the National Academy of Sciences*, 114(5):1171–1176, 2017. doi:10.1073/pnas.1611734114.

[74] Nicholas A Steinmetz, Christof Koch, Kenneth D Harris, and Matteo Carandini. Challenges and opportunities for large-scale electrophysiology with Neuropixels probes. *Current Opinion in Neurobiology*, 50:92–100, 2018. Neurotechnologies. doi:https://doi.org/10.1016/j.conb.2018.01.009.

[75] Takashi Takekawa, Yoshikazu Isomura, and Tomoki Fukai. Spike sorting of heterogeneous neuron types by multimodality-weighted PCA and explicit robust variational Bayes. *Frontiers in Neuroinformatics*, 6, 2012. doi:10.3389/fninf.2012.00005.

[76] Emanuele Torti, Giordana Florimbi, Arianna Dorici, Giovanni Danese, and Francesco Leporati. Towards the simulation of a realistic large-scale spiking network on a desktop multi-GPU system. *Bioengineering*, 9(10), 2022. doi:10.3390/bioengineering9100543.

[77] Eric M. Trautmann, Sergey D. Stavisky, Subhaneil Lahiri, Katherine C. Ames, Matthew T. Kaufman, Stephen I. Ryu, Surya Ganguli, and Krishna V. Shenoy. Accurate estimation of neural population dynamics without spike sorting. *bioRxiv*, 2017. doi:10.1101/229252.

[78] Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey Hinton. Split and merge EM algorithm for improving Gaussian mixture density estimates. *VLSI Signal Processing*, 26:133–140, 08 2000. doi:10.1023/A:1008155703044.

[79] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[80] Rakesh Veerabhadrappa, Masood Ul Hassan, James Zhang, and Asim Bhatti. Compatibility evaluation of clustering algorithms for contemporary extracellular neural spike sorting. *Frontiers in Systems Neuroscience*, 14, 2020. doi:10.3389/fnsys.2020.00034.

[81] Vercel. Vercel: Develop. preview. ship. for the best frontend teams, 2023. URL: https://vercel.com.

[82] Inc. Vercel. Next.js: The React framework for the web. Website, 2023. Accessed: August 24, 2023. URL: https://nextjs.org.

[83] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

[84] Thomas W. Volscho. Minimum wages and income inequality in the American states, 1960–2000. *Research in Social Stratification and Mobility*, 23:343–368, 2005. doi:https://doi.org/10.1016/S0276-5624(05)23011-1.

[85] L Wang, PJ Liang, PM Zhang, and YH Qiu. Ionic mechanisms underlying tonic and phasic firing behaviors in retinal ganglion cells: a model study. *Channels (Austin)*, 8(4):298–307, 2014. doi:10.4161/chan.28012.

[86] Manqing Wang, Liangyu Zhang, Haixiang Yu, Siyu Chen, Xiaomeng Zhang, Yongqing Zhang, and Dongrui Gao. A deep learning network based on CNN and sliding window LSTM for spike sorting. *Computers in Biology and Medicine*, 159:106879, 2023. doi:https://doi.org/10.1016/j.compbiomed.2023.106879.

[87] C. Ware. *Information Visualization: Perception for Design*. Information Visualization: Perception for Design. Elsevier Science, 2013. URL: https://books.google.co.uk/books?id=qFmS95vf6H8C.

[88] Webflow. Webflow: Create a custom website — no-code website builder, 2023. Accessed: August 8, 2023. URL: https://webflow.com.

[89] M. Wertheimer. Gestalt theory. In W. D. Ellis, editor, *A source book of Gestalt psychology*, pages 1–11. Kegan Paul, Trench, Trubner & Company, 1938. doi:10.1037/11496-001.

[90] Torsten N. Wiesel and David H. Hubel. Single-cell responses in striate cortex of kittens deprived of vision in one eye. *Journal of Neurophysiology*, 26(6):1003–1017, 1963. doi:10.1152/jn.1963.26.6.1003.

[91] Harlan Wilton. Unlighthouse - site-wide Google lighthouse. Web Tool, 2023. Accessed: August 8, 2023. URL: https://unlighthouse.dev.

[92] F. Wood, M.J. Black, C. Vargas-Irwin, M. Fellows, and J.P. Donoghue. On the variability of manual spike sorting. *IEEE Transactions on Biomedical Engineering*, 51(6):912–918, 2004. doi:10.1109/TBME.2004.826677.

[93] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7), 2022. doi:10.3390/brainsci12070863.

[94] Shuangming Yang, Tian Gao, Jiang Wang, Bin Deng, Mostafa Rahimi Azghadi, Tao Lei, and Bernabe Linares-Barranco. SAM: A unified self-adaptive multicompartmental spiking neuron model for learning with working memory. *Frontiers in Neuroscience*, 16, 2022. doi:10.3389/fnins.2022.850945.

[95] Lianchun Yu and Yuguo Yu. Energy-efficient neural information processing in individual neurons and neuronal networks. *Journal of Neuroscience Research*, 95(11):2253–2266, 2017. doi:https://doi.org/10.1002/jnr.24131.

[96] Zapier. Zapier — automation that moves you forward, 2023. Accessed: August 8, 2023. URL: https://zapier.com.

[97] Zhihua Zhang, Chibiao Chen, Jian Sun, and Kap Chan. EM algorithms for Gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36:1973–1983, 09 2003. doi:10.1016/S0031-3203(03)00059-1.

Appendix A

Supplementary Equations and Algorithms

$$\mu_{\text{robust}}(x) = \text{median}(x) \tag{A.1}$$

$$\sigma_{\text{robust}}(x) = \frac{\text{median}(|x - \mu_{\text{robust}}|)}{0.6745} = \frac{\text{MAD}}{0.6745} \tag{A.2}$$

This equation (A.2) describes how the Mean Absolute Deviation (MAD) is used to form a measure of a signal's deviation ($\sigma_{\text{robust}}(x)$), which is more robust to outliers than the standard deviation.

$$\text{BIC} = -2 \ln p(\mathbf{x}|\mu_{\text{opt}}, \sigma_{\text{opt}}, \pi_{\text{opt}}) + P \ln N \tag{A.3}$$

In the equation (A.3), $\mathbf{x}$ represents the data, μ_{opt}, σ_{opt}, and π_{opt} are the optimal parameters for mean, standard deviation, and mixing coefficients, respectively. P is the number of parameters in the model, and N is the number of data points. The first term in the equations corresponds to the log likelihood and the second refers to the penalty of model complexity.

$$I \propto \frac{1}{x^2}, \tag{A.4}$$

Equation (A.4) illustrates the inverse square law, which states that the intensity I of a signal diminishes proportionally to the reciprocal of the square of the distance x from its source.

$$f(x_1(t) + x_2(t)) = f(x_1(t)) + f(x_2(t)), \tag{A.5}$$

The equation (A.5) denotes the principle of superposition, stating that the total response to multiple input signals is equal to the sum of the individual components in a linear system. In this equation, f represents a linear operator, while $x_1(t)$ and $x_2(t)$ are two input signals.

Data: y, *threshold*, *minimum_gap*
Result: *extrema*
Compute the differential:
LET dy = differentiate(y)
LET *zeros* = indices where $dy = 0$
if *length of zeros* = *length of* $y - 1$ **then**
  return *empty array*
end
Find the extrema:
FIND *extrema* where dy shifts from positive to negative, and y values are greater than *threshold*
if *number of extrema* > 1 *AND minimum_gap* > 1 **then**
  SORT extrema by the y values in descending order
  INITIALISE a boolean array *rem* of size y with all elements set to True
  SET elements at indices *extrema* in *rem* to False
  foreach *extremum in sorted_extrema* **do**
    if *rem*[*extremum*] *is False* **then**
      SET slice from (*extremum* − *minimum_gap*) to (*extremum* + *minimum_gap* + 1) in *rem* to True
      SET *rem*[*extremum*] to False
    end
  end
  UPDATE *extrema* to indices where *rem* is False
end
return *extrema*

Algorithm A.1: A pseudocode algorithm used to locate the positions of extrema in a signal. Algorithm adapted from the PeakUtils Python module [54].

Appendix B

Visual Aids

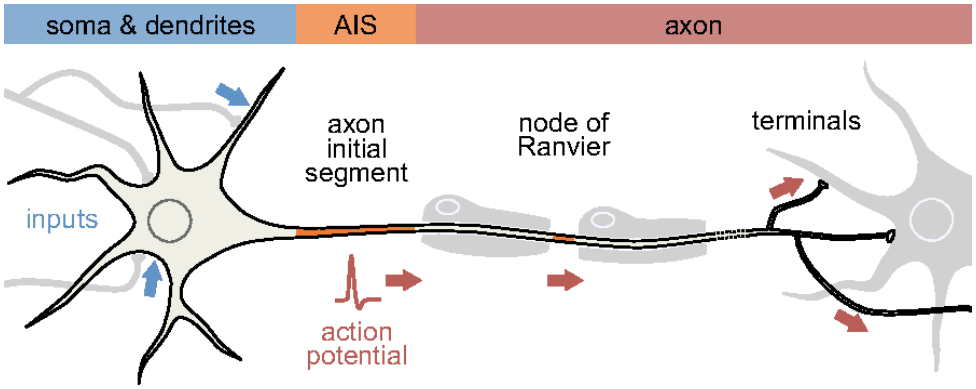


Figure B.1: Diagram depicting the initiation and propagation of action potentials along a neuron's axon, culminating at the synaptic terminals [43].

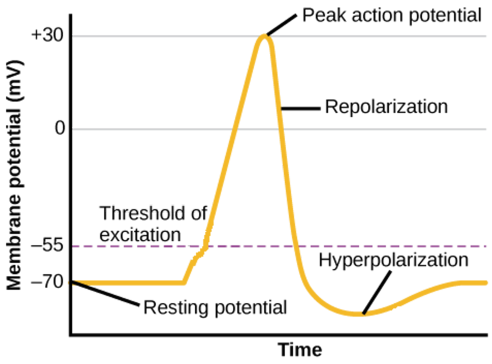


Figure B.2: Depiction of membrane potential dynamics as it surpasses the excitation threshold, causing a neuron spike. Image modified from "How Neurons Communicate: Figure 35.11" by OpenStax College, Biology [59].

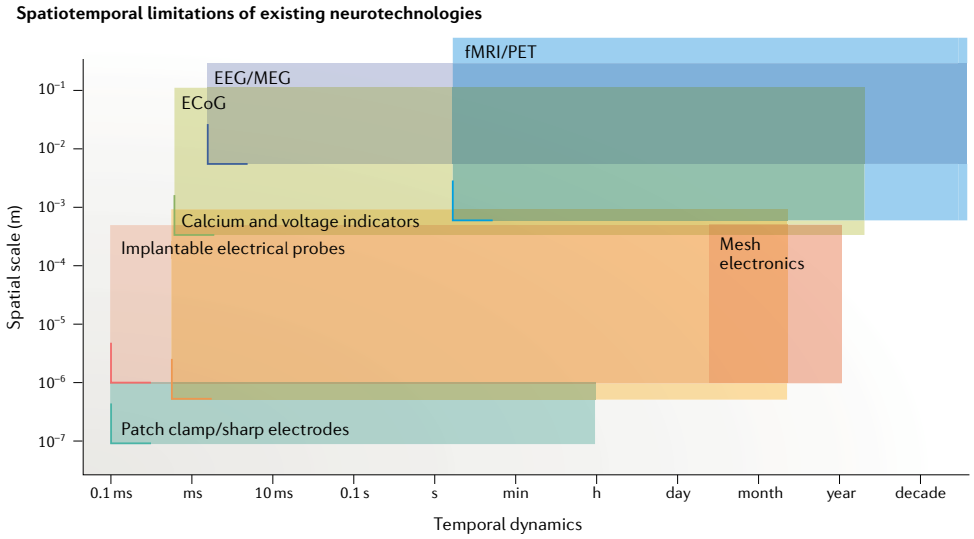


Figure B.3: The "spatiotemporal dilemma of various existing neurotechnologies." Each translucent box represents a neural recording technique, with its area and position encoding its temporal and spatial resolution, respectively. Image sourced from Hong and Lieber, 2019 [34]

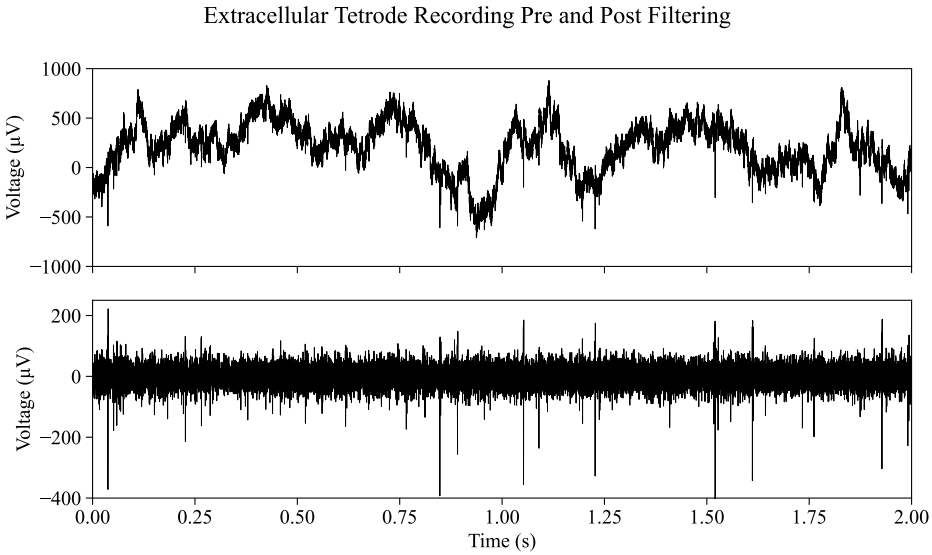
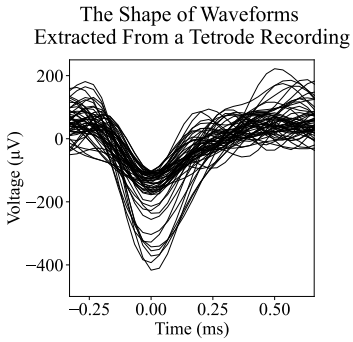
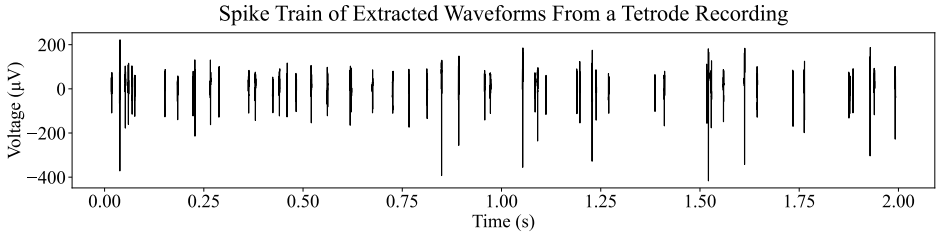


Figure B.4: The raw data from a single channel of a tetrode recording (upper panel) and the same signal when subject to a Butterworth bandpass filter (lower panel) with upper and lower frequency bounds of 500 Hz and 4000 Hz, respectively. Data sourced from Berens and Ecker, 2021 [9].



(a)



(b)

Figure B.5: The shape of the waveforms extracted from the first two seconds of a tetrode recording (upper) and the corresponding spike train (b). The length of each waveform window is 1 second, with the scale of the x-axis in (a) centred about the point of maximum amplitude. Data sourced from Berens and Ecker, 2021 [9].

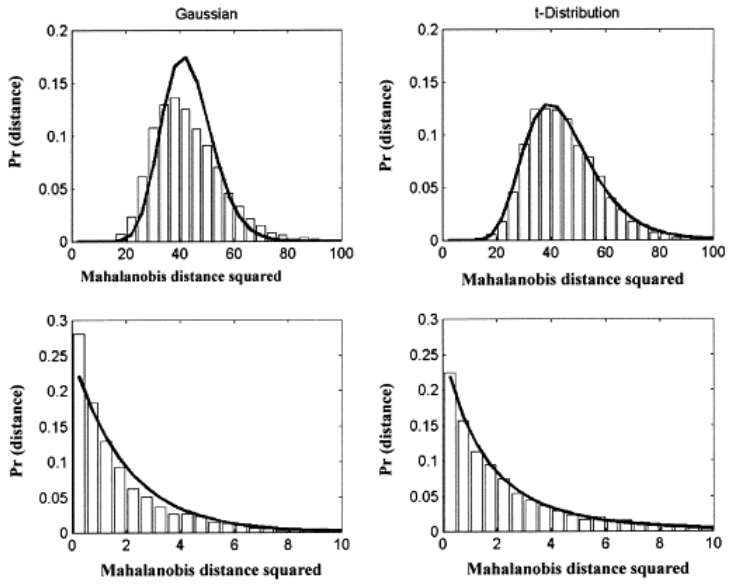


Figure B.6: The distributions for the predicted and actual Mahalanobis squared distances for Gaussian (left) and t-distribution models (right). Image sourced from Shoham et al., 2003 [70].

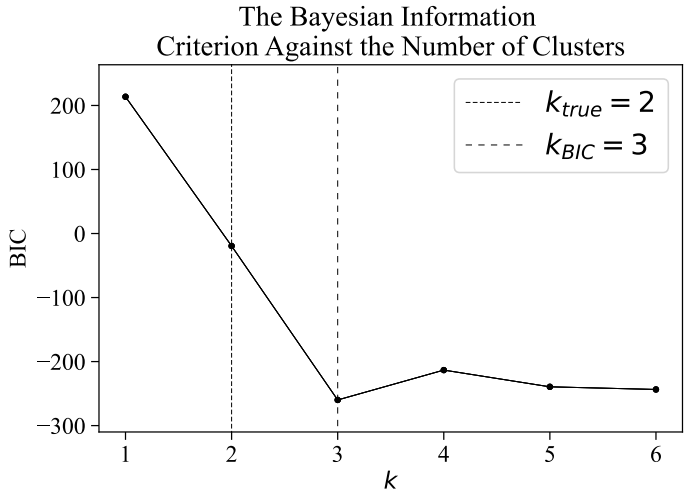
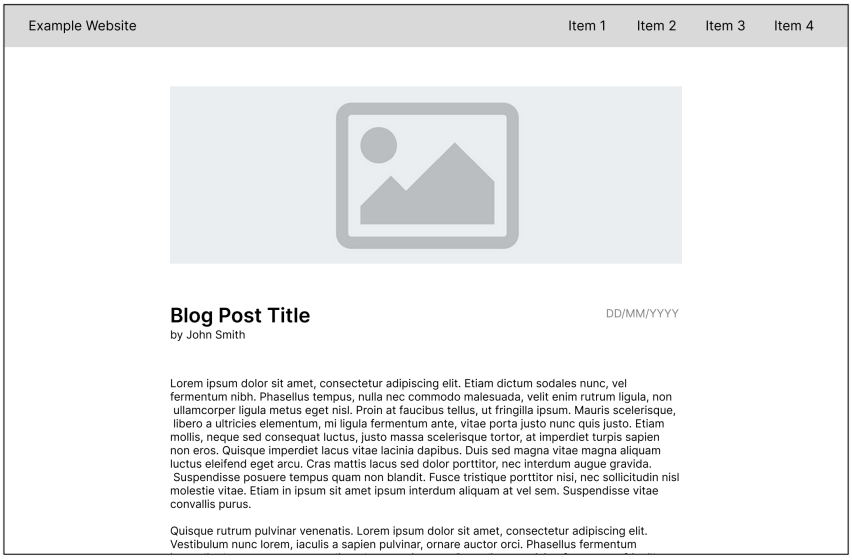


Figure B.7: The Bayesian Information Criterion (BIC) plotted against the number of clusters, k, for vectorised waveforms from two neurons.



(a)

(b) (c)

Figure B.8: A template design of a blog website page displayed on a standard 14.2-inch screen (a) and on a 6.1-inch mobile screen with both a responsive (b) and an unresponsive (c) layout.

Appendix C

Web Application Graphics

(a) (b)

Figure C.1: An example of a section being restricted until the previous step is complete, on both a large screen (a) and a small screen (b). The section shown would otherwise allow for the type of neuron, and its parameters, to be chosen (see Figure C.6).

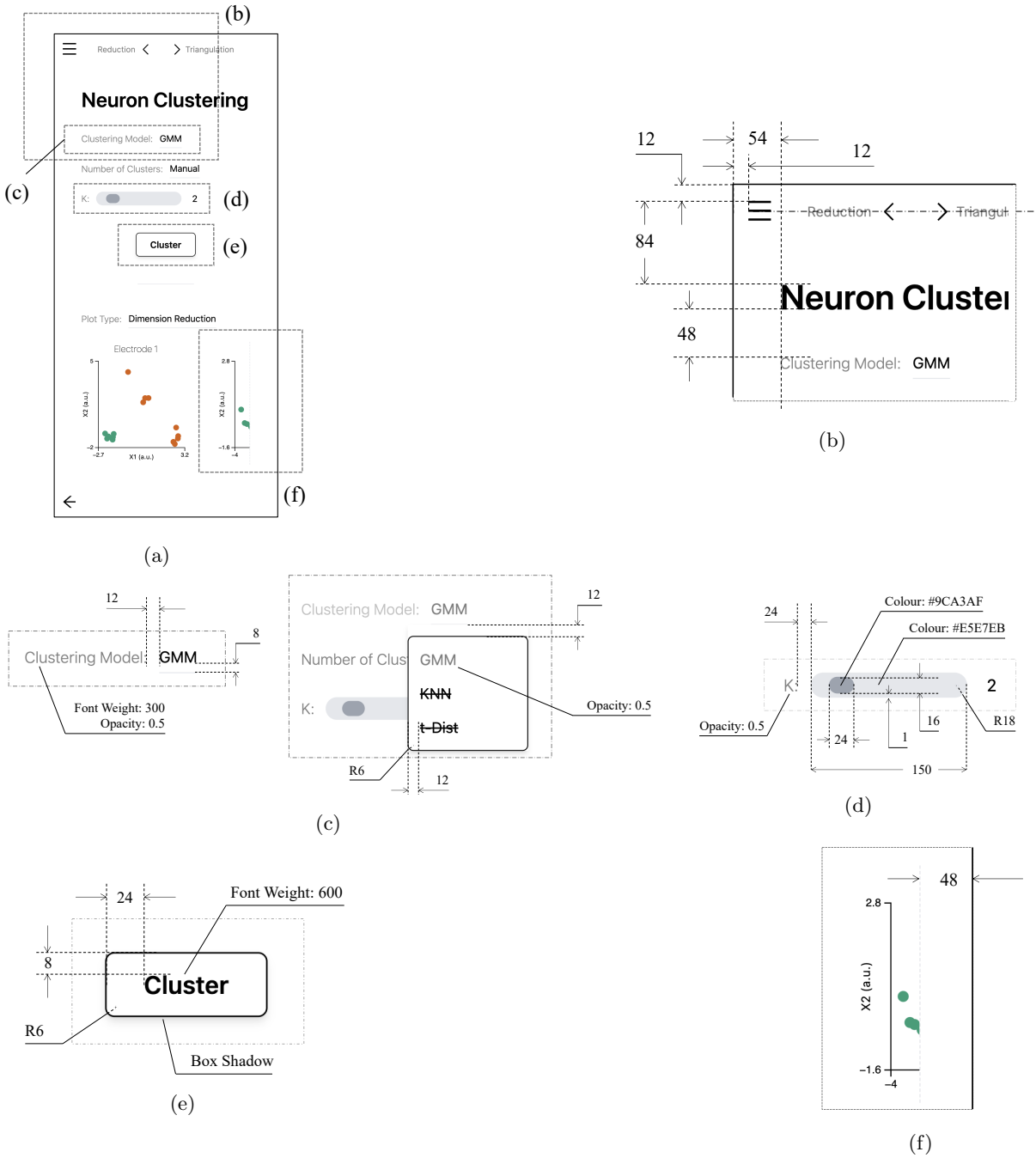
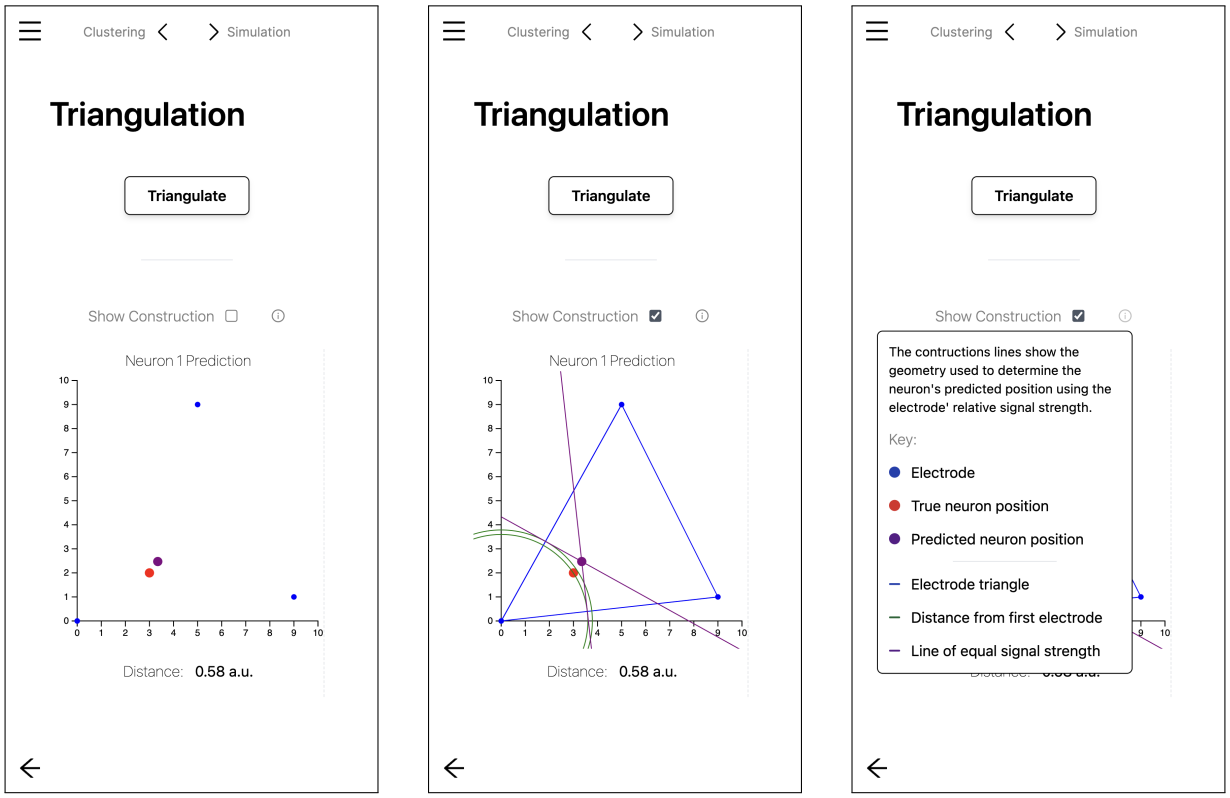
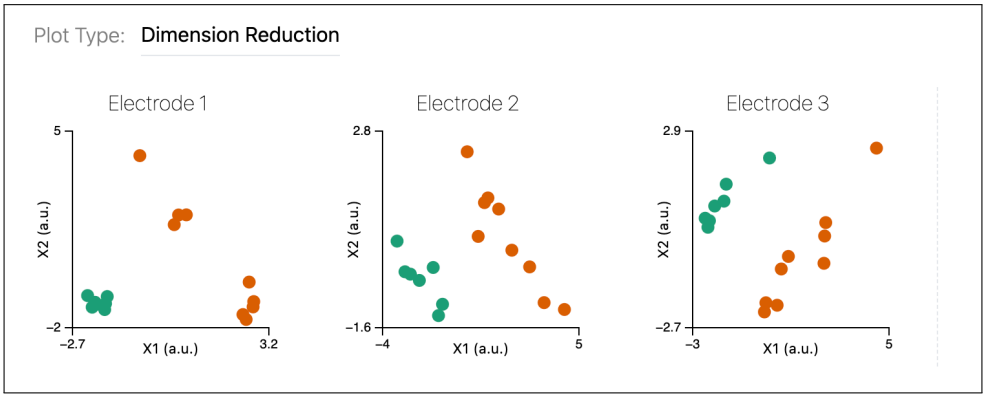


Figure C.2: The style guide for some of the web application's common components. The figure shows a section of the site with several superimposed boxes isolating each component (a), the navigation and header components (b), the dropdown selector when closed and open (c), the range slider (d), the action button (e) and the dashed line indicating a scrollable component (f).

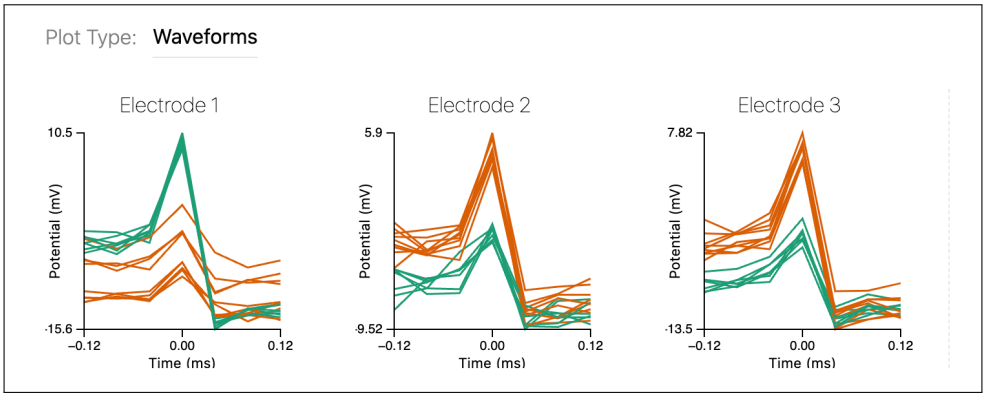


(a) (b) (c)

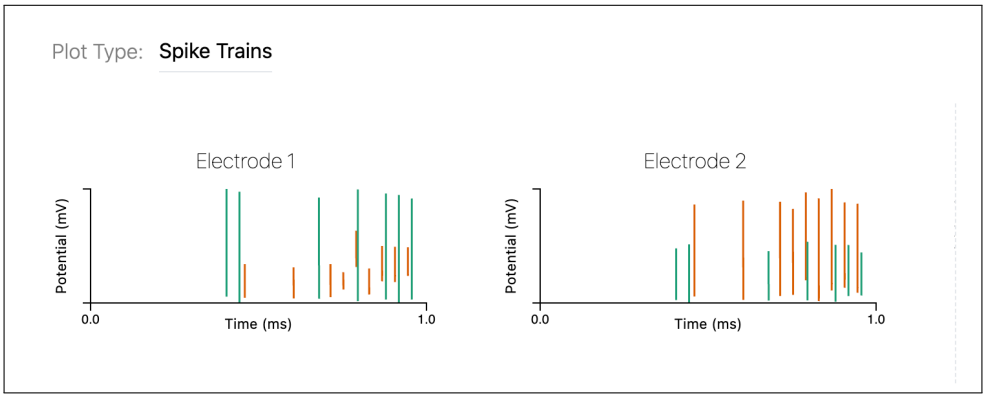
Figure C.3: Triangulation section of the web application showcasing: (a) the electrode positions alongside the actual and predicted neuron locations, (b) the construction lines used to determine the predicted neuron positions upon user request, and (c) information about these construction lines.



(a)

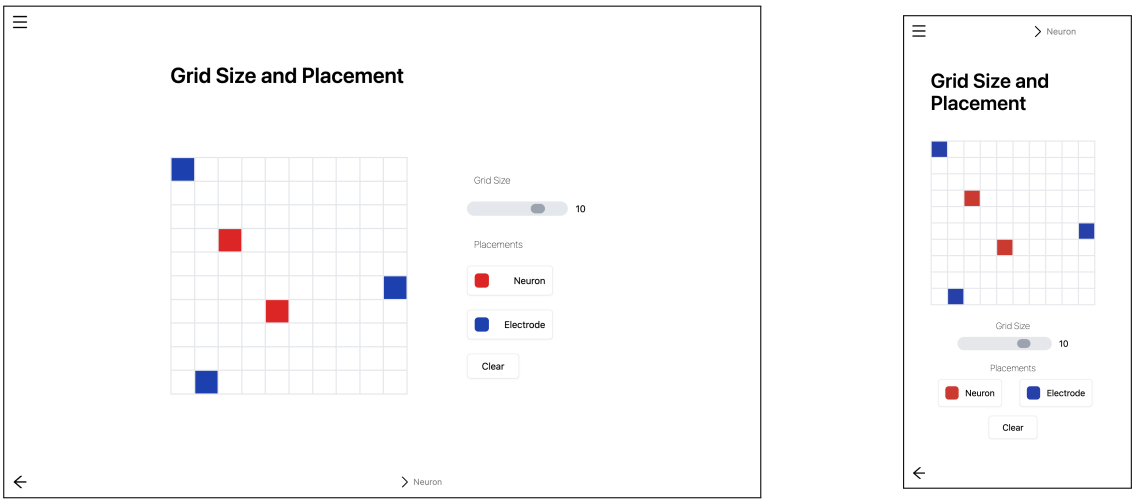


(b)



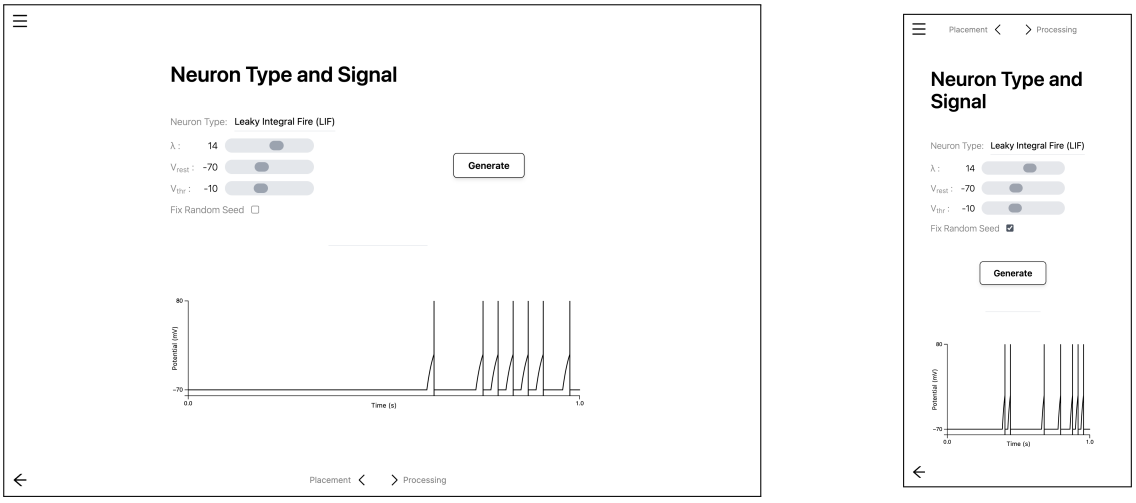
(c)

Figure C.4: The available clustering visualisations: (a) a two-dimensional representation of vectorised waveforms, (b) the extracted waveforms, and (c) a spike train plot for two predicted neurons. The spike train plot displays recordings from two electrodes; users must scroll to view the third electrode's data.



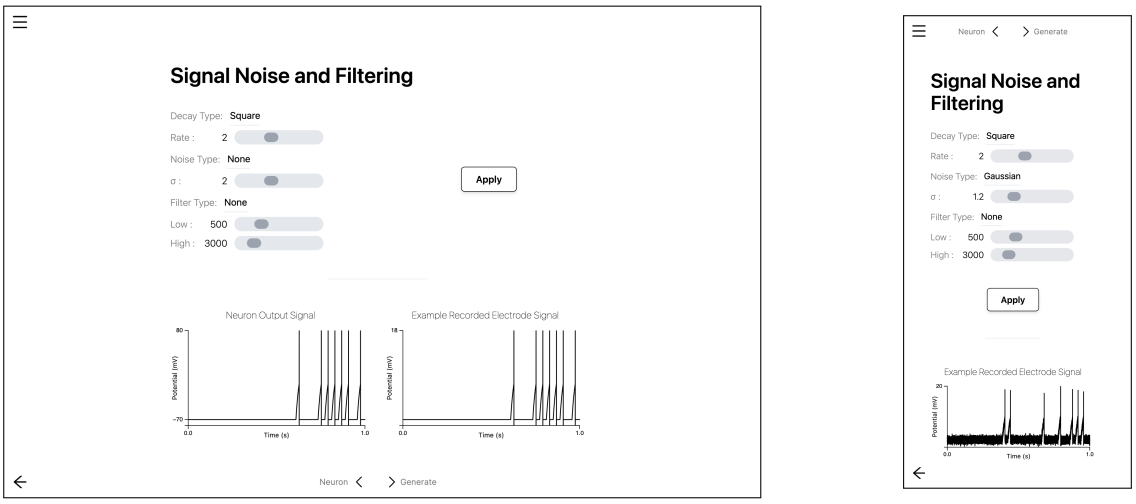
(a) (b)

Figure C.5: Comparative view of the web application's electrode/neuron placement section: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.



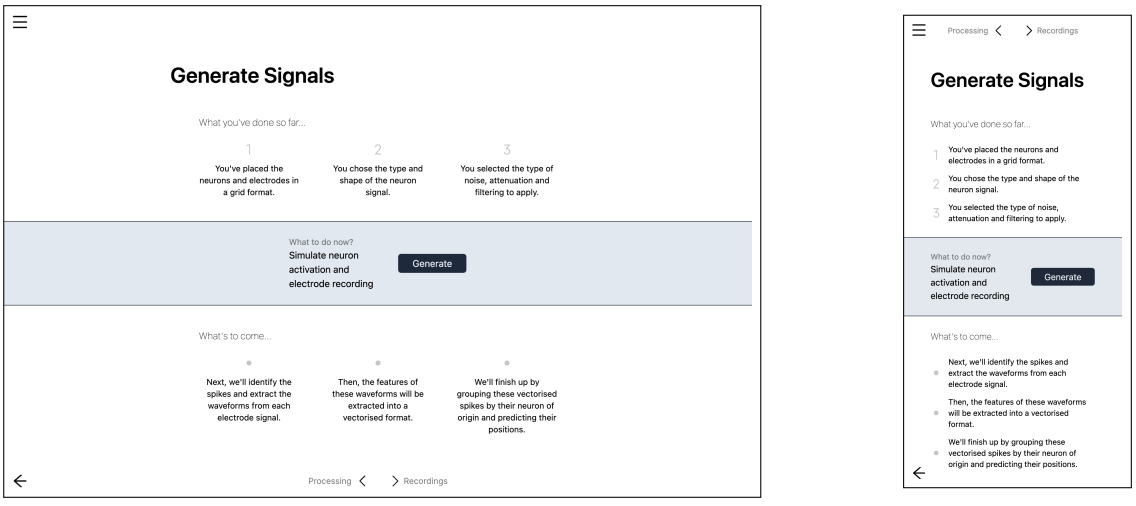
(a) (b)

Figure C.6: Comparative view of the web application's section for adjusting the type and parameters of the neurons: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.



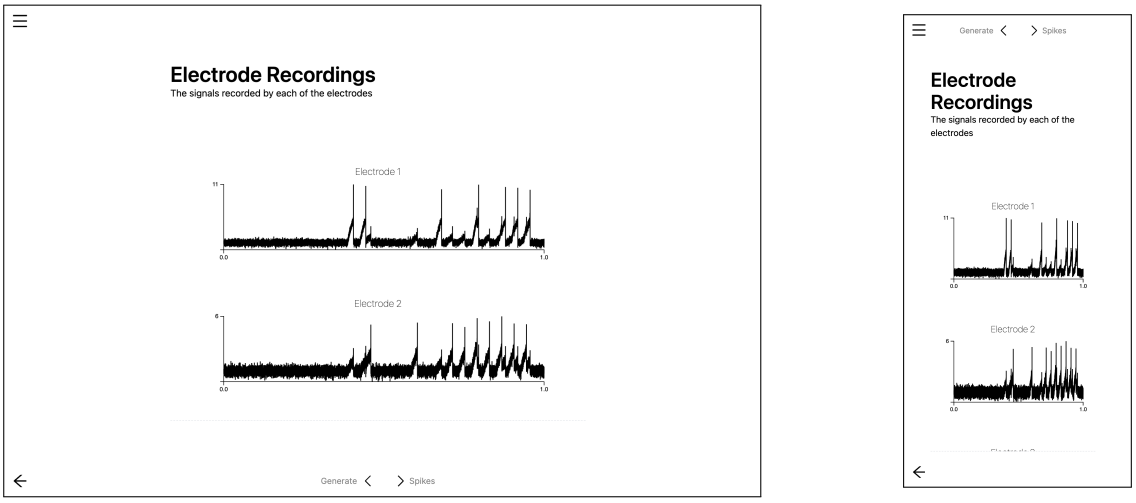
(a) (b)

Figure C.7: Comparative view of the web application's section for adjusting the parameters of the connecting medium and electrode signal filtering: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.



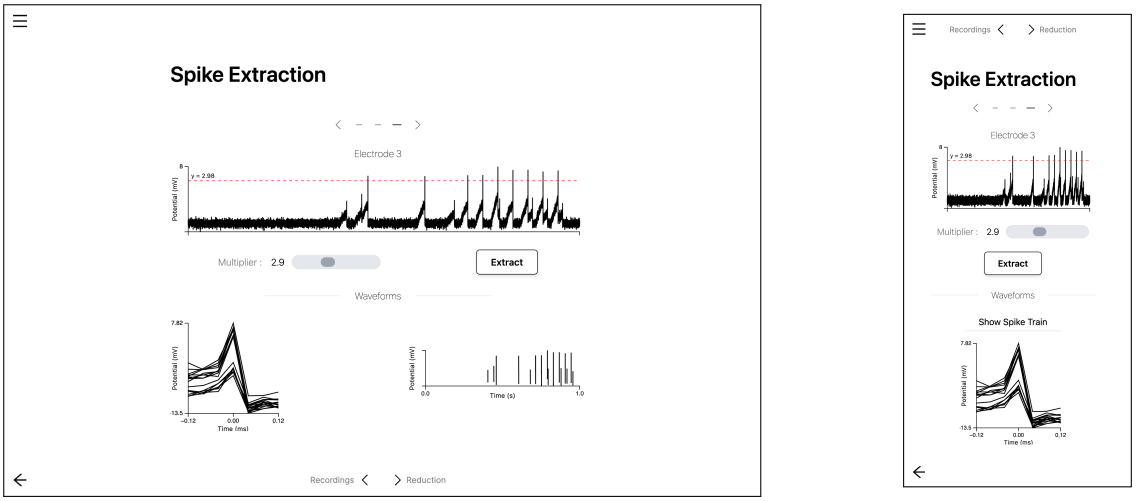
(a) (b)

Figure C.8: Comparative view of the web application's section for generating the simulated recordings: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.



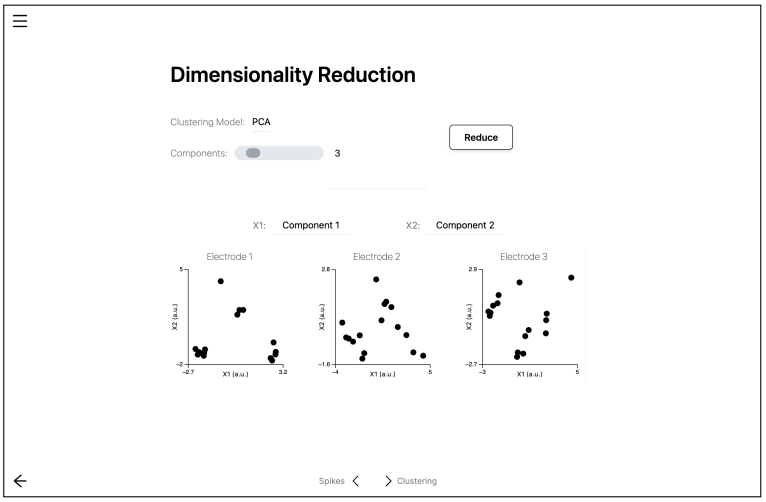
(a) (b)

Figure C.9: Comparative view of the web application's section for viewing the simulated recordings: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.

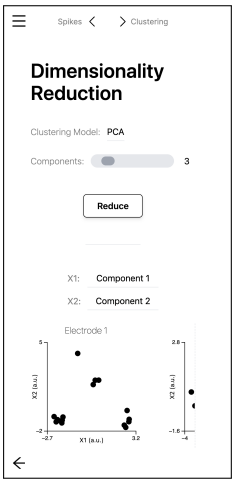


(a) (b)

Figure C.10: Comparative view of the web application's section for extracting the spikes from each electrode recording: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.

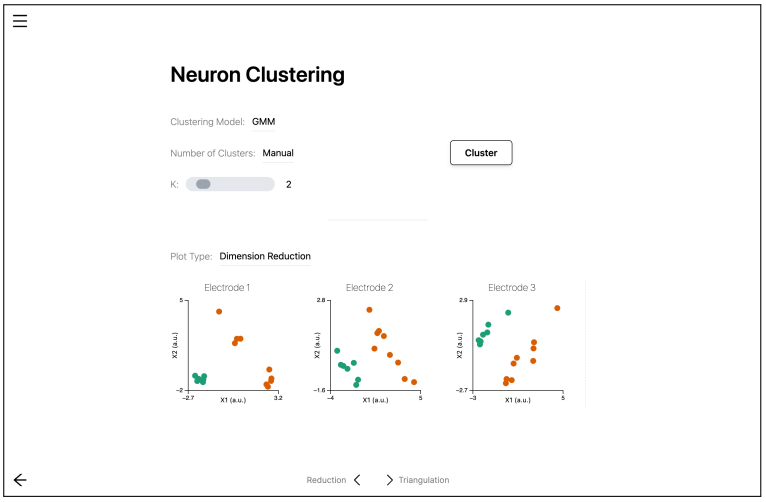


(a)

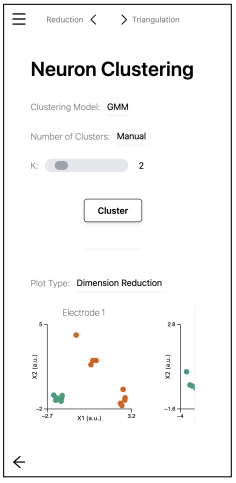


(b)

Figure C.11: Comparative view of the web application's section for extracting the features (dimensionality reduction) of the waveforms: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.

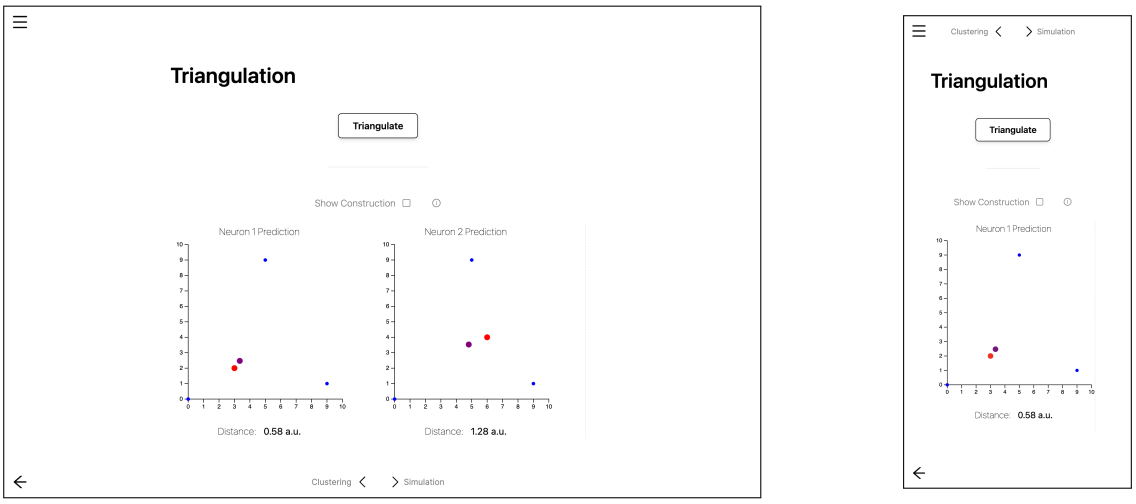


(a)



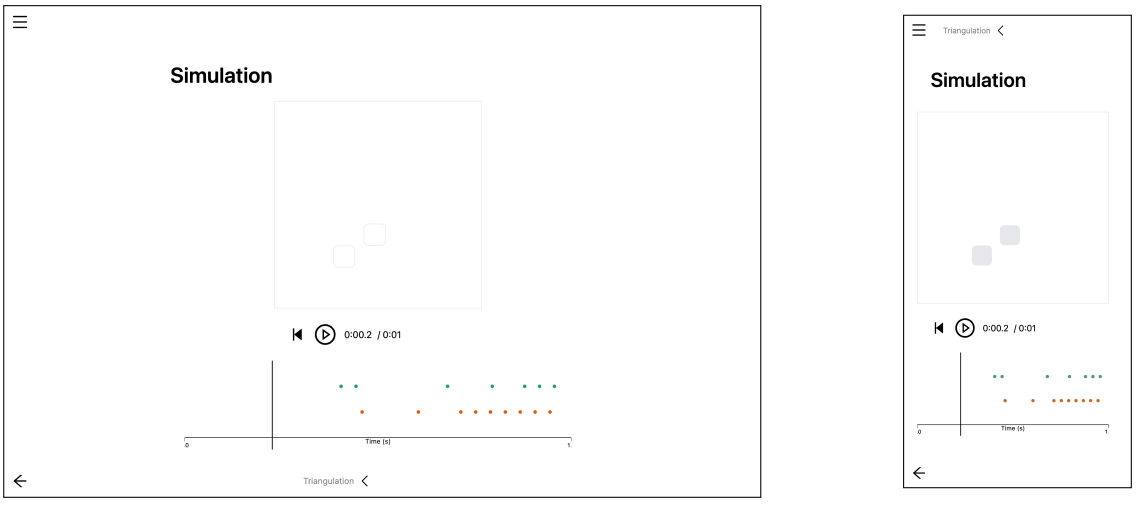
(b)

Figure C.12: Comparative view of the web application's section for clustering the vectorised waveforms by their predicted source neurons: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.



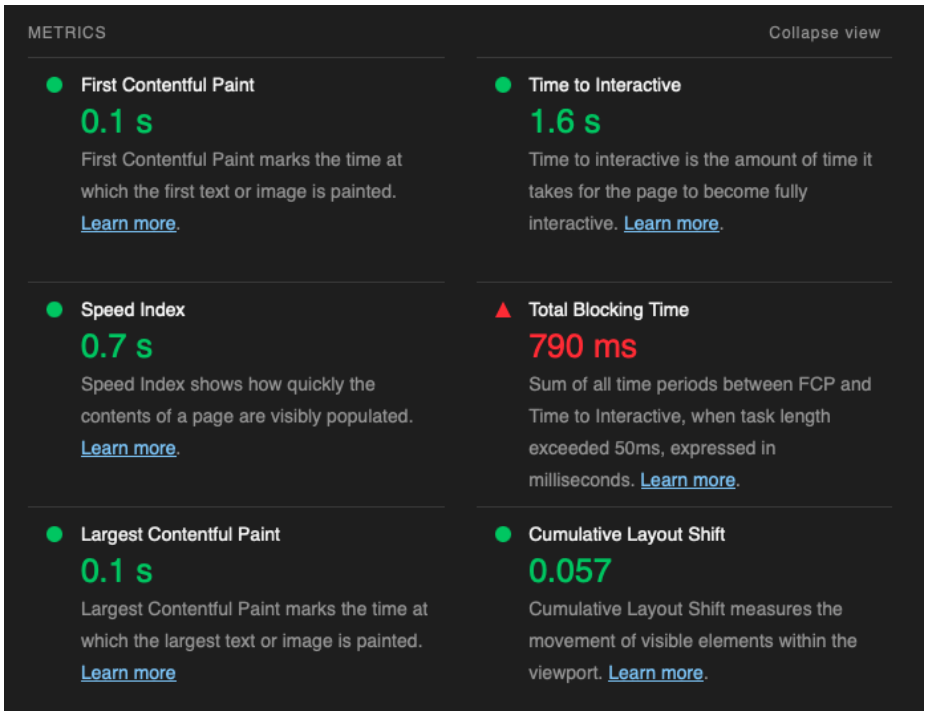
(a) (b)

Figure C.13: Comparative view of the web application's section for triangulating the position of the predicted neurons relative to the electrodes: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.

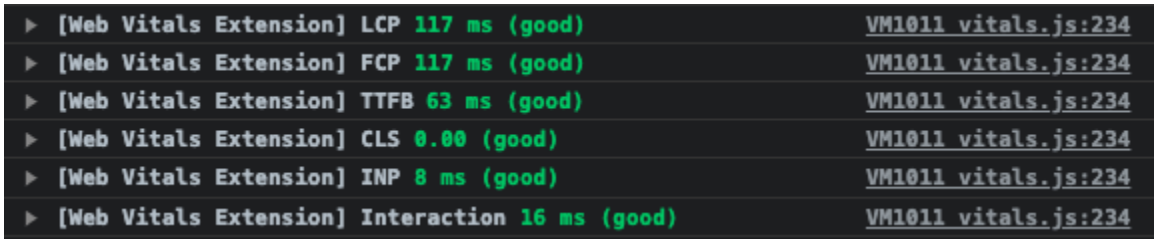


(a) (b)

Figure C.14: Comparative view of the web application's section for playing the simulated neuron signal: (a) on a large screen with dimensions 1261x828px, and (b) on a compact mobile screen sized 393x851px.



(a)



(b)

Figure C.15: The performance of the frontend application according to Google's web vitals with reports generated from (a) Unlighthouse [91] and (b) the Web Vitals browser extension [2]. Note that INP (Interaction to Next Paint) is essentially an updated version of FID (First Input Delay) [72].