

TDK-thesis

Dominika Darabos

Spike classification and decision analysis on microneurography data using Weighted Hermite VPNet

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF NUMERICAL ANALYSIS



Author:

Dominika Darabos

Computer Science MSc

Supervisor:

Dr. Péter Kovács

Habilitated Associate Professor

Budapest, 2024

Abstract

Microneurography data provides valuable insights into peripheral nerve responses critical for understanding sensory and motor functions. This thesis presents a unique approach by using a neural network solution to detect spike events in microneurography data, using the weighted Hermite Variable Projection Neural Network model to capture distinct neurophysiological patterns. Addressing the inherent challenges of high noise levels, low signal-to-noise ratio, and extreme class imbalance, the proposed pipeline integrates a knowledge-augmented model, data balancing techniques, model output visualization, and explainable AI methods to ensure both accurate and interpretable predictions. A sliding window technique is applied to the one-dimensional time series signal to detect spike events, where window-wise binary classifications are transformed back into the time series, resulting in a continuous detection signal suitable for visual interpretation. The model's decision-making process is analyzed post-hoc, uncovering the neural patterns and feature extraction mechanisms it has discovered. This detection-focused approach paves the way for future advancements in neural network approaches for spike sorting.

Contents

1	Introduction	3
1.1	Machine learning and signal processing	3
1.2	Explainable artificial intelligence	5
1.3	Neuroscience background	8
1.4	Scope of this work	9
2	Neural data	11
2.1	Nervous system	11
2.2	Electrophysiology and microneurography	13
2.3	Data characteristics	14
3	Problem statement and solution plan	18
3.1	Review of previous work in microneurography spike identification	18
3.2	Automated system plan	21
3.2.1	Data processing pipeline	22
4	Design decisions and implementation	26
4.1	Data Processing - From raw data to labeled samples	26
4.1.1	Handling imbalanced dataset during preprocessing	33
4.1.2	Dataset split structure	34
4.1.3	Preprocessing pipeline overview	35
4.2	Model design and architecture	35
4.2.1	Weighted-Hermite VPNet	35
4.2.2	Handling imbalanced dataset at the model level	38
4.2.3	Validation strategies and evaluation metrics	38
4.2.4	Visualization of model outputs	40
4.3	Explainability	43
4.3.1	Post-hoc interpretability	43

4.3.2	Algorithms for explainable artificial intelligence	44
5	Results	46
5.1	Outcomes of the classification model	46
5.2	Outcomes of the explainability analysis	51
6	Conclusion and improvement potentials	57
	Acknowledgements	59
	Bibliography	62

Chapter 1

Introduction

1.1 Machine learning and signal processing

A signal, technically, represents a phenomenon evolving over time, space, or any independent variable. In essence, a signal carries information that can be processed or utilized for various purposes. Signal processing refers to any manual or mechanical operation that modifies, analyzes, or otherwise manipulates the information contained in a signal in order to extract useful information. Serving as a bridge between Mathematics, Physics, and Informatics, signal processing is inherently multidisciplinary, involving digitalized forms of data such as images, audio, video, and sensor data. Mathematics is necessary to evaluate it, Informatics enables the implementation, and the physical world will generate the signals.

Signal processing traditionally relied on model-based mathematical or physical algorithms, involving manual data manipulation methods to extract useful information. Techniques like linear estimators, Fourier transforms, wavelet transforms, and adaptive filtering [1] served as essential tools for information extraction. While these methods differ in their fundamental principles, they share the common goal of effectively manipulating data to reveal desired features.

With the emergence of artificial intelligence, it became clear that data could serve a dual purpose — not only as a source of past evidence but also as a foundation for future predictions. By identifying patterns in past data, models could be trained to process current inputs and predict future outcomes. Machine learning integrates signal processing, computer science, and statistics to harness this predictive power, opening doors to innovative technologies. As before, these methods are built upon mathematical formulas,

forming powerful models that generate predictions and guide decisions based on input data.

Supervised learning algorithms offer an automated and optimized alternative to manual approaches for various tasks. For instance, linear and logistic regression are effective in modeling linear relationships and binary classification, respectively. Decision trees split the feature space based on feature values, making them intuitive for both classification and regression tasks, while random forests combine multiple decision trees to improve accuracy. Support vector machines find the optimal hyperplane to separate classes. Principal component analysis reduces data dimensionality by identifying orthogonal components that explain maximum variance. These mathematical algorithms continue to provide versatile, efficient, and automated solutions across diverse applications, surpassing the limitations of earlier manual techniques.

However, in light of the rapid development of machine learning tools, the complexity and quantity of data are also exponentially growing. To address this escalating demand, a shift toward deep learning, is gaining importance. Deep neural networks can automatically learn complex patterns and representations directly from raw data, and offer end-to-end learning pipelines, enabling the entire signal processing task to be learned in a single model, unlike traditional algorithms that involve separate stages for the processing. Furthermore, neural networks demonstrate adaptability to diverse signal types and noise levels, enabling them to generalize well across various signal processing tasks and adapt to changing environments. They offer scalability and flexibility to handle high-dimensional data and large datasets efficiently, making them valuable tools for more complex signal processing applications, thus meeting today's growing requirements.

Although neural networks show promising results, they come with their own challenges. The model learns from input data, but this learning is guided by hyperparameters, and finding the optimal settings for each parameter can be time-consuming and challenging. Another possible difficulties are, for example, overfitting and underfitting, which can also impact expected performance on unseen data. While these challenges are common, there are techniques to address them.

However, there is a bigger unresolved issue they face, the lack of interpretability. Paradoxically, the complexity that drives their success and accuracy is also what raises concerns. The end-to-end processing pipeline includes components learned by the network, yet these are not transparent or interpretable to outside observers. This so-called "black box" phenomenon limits the use of neural networks in critical fields where excel-

lent results alone are insufficient. Especially in safety-critical applications, the demand for interpretability has reached a point where the quality of explanations is nearly as important as accuracy itself, making the outcome truly valuable only when supported by a thorough explanation.

1.2 Explainable artificial intelligence

The demand for explanations in deep neural networks inspired an emerging new field, explainable artificial intelligence (XAI). XAI methodologies focus on analyzing the learning process of neural networks. They aim to 'open up the black box' of these networks, providing insights into how the model creates its decisions. It is a set of processes and methods that allows humans to understand, interpret, and trust the results and output created by machine learning algorithms. Explainable AI is crucial for building trust and confidence before deploying AI models in real-life settings. By offering insights into the inner workings of AI systems, it decreases the risk of blind reliance on network outputs, thereby promoting responsible AI usage. This approach fills the gap between the high performance of complex models like deep neural networks and the need for interpretability in critical applications. The development of XAI is crucial for extending the use of advanced machine learning models to domains where understanding the reasoning behind a decision is as important as the decision itself.

According to the article [2], there are two main directions of XAI methods, that aim to different levels and types of interpretability. Firstly, the total and most strict explanation of the processes is transparency, which is considered at the level of the whole network and can be extracted during the model training process. Transparency's rules express that the results are satisfactory enough if the decision-making process is so clear that it could be fully reproducible manually without the model itself. Transparency offers a clear, controlled approach to AI usage, but as the model gets more complex, these methods can not be implemented. Also, full reproducibility is not always necessary, the goal is to understand the model's inner workings rather than to replicate everything already achieved.

Secondly, post-hoc interpretability bypasses the strict requirements of full transparency. It does not necessitate knowing every single detail to the extent that the decision can be entirely reproduced anytime. Instead, the focus is on providing clarity about the reasons behind a final decision. These interpretations include natural language explanations, visualizations, and case-based reasoning. Language explanations resemble human

reasoning, where we describe our thoughts and how we reach a conclusion in words. Visualizations are generated from gradient maps computed from the input, illustrating which regions the model heavily relies on. Examples can be helpful when we want to clarify a decision based on a familiar previous case, like doctors referring back to case studies.

The comprehensive survey on XAI [3] identifies additional types of explanation methods, which align with those discussed in [2]. At the model level, two primary directions are model-specific and model-agnostic interpretability. Model-specific interpretability is connected to a particular model type, involving processes that may not be applicable or effective for other models. This approach often involves fully transparent methodologies, which can be limiting: if a certain level of interpretability is required, the model that provides it may sacrifice predictive accuracy, creating a trade-off between interpretability and performance.

To address this limitation, model-agnostic methods have been developed. These methods are not tied to any particular model, making them more versatile and suitable for comparison across various models. Model-agnostic interpretability is typically applied post-hoc. Within this approach, there are several methods, including visualization techniques, knowledge extraction (such as rule extraction or model distillation, where a simpler "student" model is used to analyze the decision-making process), influence methods (where explanations are derived by observing the effect of modified specific inputs), and example-based explanations.

The survey also introduces two main types of explanations, which can be applied within both model-specific and model-agnostic frameworks: global and local explanations. Global interpretability seeks to understand the overall logic of the model by providing insights into all possible outcomes, while local interpretability focuses on explaining individual predictions. Global explanations provide a broad understanding of the model's behavior, whereas local explanations build trust by offering detailed insights into specific decisions.

An explanation has to be understandable by humans, which is why post-hoc interpretability has become so popular. These methods mirror how humans naturally explain decisions. Just as we do not need to know every single step of another person's thought process to understand their decision, post-hoc interpretability highlights the most influential factors leading to a model's outcome. Explanations are inherently selective, typically focusing on one or a few key causes strong enough to be responsible for a significant part

of the outcome, rather than detailing every minor factor. [4] As stated in [5], post-hoc interpretability is considered effective to the extent that humans are themselves interpretable. Among the techniques, visualizations are the most human-centered, where the emphasis is not only on making the display visually appealing but also on ensuring it is visually understandable.

However, despite being based on mathematical equations, post-hoc interpretability algorithms are not universally applicable to all models, do not consistently produce fully accurate explanations, and may reveal noise rather than meaningful insights. In these scenarios, the danger lies in our desires, as we may accept the result of an algorithm even if they are misleading, simply because the outcome aligns with our prior expectations. Similar to interpreting statistical results, single outcomes alone do not always tell the whole story. Analytical results must be verified from multiple perspectives.

The need for trustworthy artificial intelligence depends greatly on the field in which AI is used. For fields with lower risks, where the credibility of AI can be established primarily through output results and experience over time, and where errors do not lead to unpleasant, significant, or life-threatening consequences, the extra effort to explain AI decision-making may not be justified—in terms of time, energy, or cost. However, in safety-critical fields the decision process becomes crucial. As shown in [3], interpretability is essential in fields such as healthcare, transportation, legal, finance, and military scenarios. These fields have historical examples demonstrating how outcomes differ depending on the presence of interpretability. In healthcare, for example, a model identified a counterintuitive pattern supported by data, but incorrect in real life, which led to misclassification of emergency cases, resulting in delayed care. In legal contexts, cases have shown that models made different decisions in otherwise identical cases where only race or age varied—an unacceptable bias in legal institutions. Transportation and military applications also rely on decisions in life-threatening scenarios, making interpretability equally critical in these fields.

Explanations provide greater control, drive improvements, and can lead to discoveries [3]. When a system’s behavior is better understood, it can reveal vulnerabilities and critical points. Identifying these allows for targeted improvements, enhancing the system’s stability and performance. Since neural networks detect patterns in data, there is also the potential for the model to identify previously undiscovered patterns. Running explanatory algorithms on these findings may reveal truly valuable insights that can be applied in the future.

1.3 Neuroscience background

Neuroscience is a multidisciplinary field focused on understanding the structure and function of the nervous system, encompassing neuroanatomy, neurochemistry, and neurophysiology. Neuroanatomy studies the structure and organization of the nervous system. It explores how different parts of the nervous system are interconnected and how these connections support functions like sensory processing, motor control, and emotion. Neurochemistry explores the chemical processes and activities of nervous tissue and how signals are transmitted throughout the body, examining the effects of neurotransmitters like dopamine, noradrenaline, and serotonin in the brain. Neurophysiology investigates how information is processed at the level of individual neurons to gain an understanding of how neural networks operate.

The data gathered in the field of neurophysiology are known as electrophysiological recordings - the recording of electrical activity in individual neurons and neural groups exhibiting coordinated activity. This process involves measuring voltage changes during active function or in response to manipulations. Electrophysiological recordings include several techniques, such as electroencephalography (EEG), microneurography, local field potential (LFP) recordings, and intra- and extracellular recordings. EEG measures electrical activity across the scalp to monitor brain activity patterns, often used in sleep studies, epilepsy research, and cognitive neuroscience. Microneurography captures single or multiunit nerve activity directly from human peripheral nerves outside the brain, providing insights into sensory and motor function. LFP recordings, typically collected from specific regions of the brain, measure the summed electrical activity of neural populations, offering valuable information about local processing. Intra- and extracellular recordings, conducted either inside (intracellular) or outside (extracellular) nerve cells, are crucial for studying individual neuron behavior, synaptic activity, and neuron-to-neuron interactions.

Neurobiologists gather vast amounts of data that vary across multiple types and modalities, including how the data is captured, the type of data collected, the species studied, the presence of any diseases, the environment in which the data is recorded, and so on. By applying data science and analytical techniques to these large-scale datasets, valuable information can be extracted to decode complex patterns of brain activity and uncover insights that bring the researchers one step closer to understanding underlying brain functions.

Many electrophysiological recordings are aimed at identifying responses to specific

stimuli or behaviors. A stimulus is anything that can trigger a physical or behavioral change. One approach to analyzing these recordings is to isolate the response to a particular stimulus. The challenge of detection largely depends on the data's characteristics. When responses are clearly distinguishable, spike detection algorithms, such as threshold-based methods, wavelet transforms [6], template matching [7], and machine learning-based classifiers, can be highly effective.

Threshold-based methods detect spikes by setting amplitude limits, and identifying signals that exceed a defined threshold as spikes. Wavelet transform decomposes a signal into a set of basis functions generated by contracting, expanding, and translating a primary function, known as the mother wavelet. Template matching involves creating a model of the expected spike shape and scanning the data for matches, while machine learning-based approaches use training data to detect spikes with greater adaptability.

However, when the signal-to-noise ratio is low, these methods begin to lose effectiveness, making analysis challenging without expert knowledge. While background noise reduction is a potential solution, it may also reduce the response signal itself, possibly causing data loss that impacts the results. The recording techniques for these responses also contribute to waveform instability, as even slight, inevitable movements of the recording probe can significantly alter the recording of these micro-scale elements. Consequently, developing robust and automated spike detection methods that minimize response loss remains a significant challenge.

1.4 Scope of this work

The main concept is based on previous research [8][9], where the microneurography recordings were recorded and labeled by Barbara Namer (Namer's microneurography lab at University Hospital of Aachen/Würzburg). With the ground truth spike labels, the recordings can be used for detection and sorting algorithms. In [9], Kutafina's research group (Data-driven medicine at University Hospital of Cologne) has used the VPNet[10] for spike sorting, with promising initial results.

Spike detection and sorting [11] is particularly challenging because the signals are recorded using an inserted probe, which can capture activity from multiple nerve fibers simultaneously, making the detection and separation of individual spikes complex. The conventional spike sorting pipeline relies heavily on manual detection, statistical filtering, shape matching, and clustering. However, variability in spike strength, shape, and timing

complicates the analysis and reduces the pipeline’s overall effectiveness. The complexity of the data has led to a debate around an accepted and effective method, having no overall fully automated pipeline for this problem.

The collaborative work of Namer’s and Kutafina’s lab [9] - the Digital C-Fiber group in Germany - has led to a demand for an automated detection part of the labeled microneurography data, that finds its basis within this project. The goal is to develop a framework with a unique preprocessing pipeline, with the training of a published neural network model [12], and a detection based output visualization to accurately detect spike occurrences, despite significant background noise and variations in spike shapes. The project begins with a recording containing responses from two stimulated nerves, and the methodology is intended to scale to recordings involving more types of responses. The framework’s input will consist of a 1D signal representing an electrophysiological recording. The output will be a visual representation of the input signal, color-coded to indicate the trained model’s predictions, making it immediately obvious in which areas the model detected spike occurrences. The automated detection phase will significantly reduce manual effort, allowing researchers to focus on revising and classifying the detected spikes instead of examining every detail of the whole input data.

Once valuable detection metrics are achieved, the decision-making process will undergo XAI analysis to uncover patterns generated by the model. These insights can then be used to adjust the model for even better performance, and for an overall understanding of the underlying processes.

Chapter 2

Neural data

2.1 Nervous system

The nervous system is divided into two primary components: the central nervous system (CNS) and the peripheral nervous system (PNS). The CNS, consisting of the brain and spinal cord, is the main processing center. It receives sensory information from the PNS, processes it, formulates responses, and sends them back to the PNS as commands. The PNS, involving all nerves outside the CNS, acts as a communication network, feeding information to the CNS and carrying out its commands.

The PNS has two main subsystems: the autonomic nervous system (ANS) and the somatic nervous system (SNS). The ANS governs automatic processes essential for survival, like heart rate, blood pressure, and digestion, without conscious input. In contrast, the SNS handles optional movements and sensory perception, allowing us to consciously interact with and respond to our environment.

The basic building block of the nervous system is the neuron, a specialized cell designed for communication, using both electrical and chemical signals. The three main parts of a neuron are the cell body, the axon, and the dendrites. The cell body is essential for the neuron's maintenance and function, processing signals received from other cells. Dendrites are branching extensions that receive incoming signals from neighboring neurons and direct them toward the cell body. If enough input is received, an action potential, or spike, is generated, which the axon then carries away from the cell body to other neurons, muscles, or glands, facilitating communication throughout the nervous system. Nerves are composed of many neurons bundled together with their axons, forming nerve fibers. These fibers carry electrical impulses between the brain and the rest of the

body, enabling sensation, muscle movement, and the regulation of autonomic functions like breathing, sweating, and digestion. Information flows in two directions: some nerves send sensory data from the body to the brain, while others transmit motor commands from the brain to muscles and organs.

Nerves can be categorized into three types based on their function:

- Sensory nerves pass on information from sensory receptors to the CNS. These nerves originate in the skin and other sensory organs, sending information to the brain via the spinal cord in response to external stimuli. These stimuli might include sensations of touch, pressure, temperature, and pain, allowing the body to respond appropriately to its environment.
- Motor nerves carry instructions from the brain to the muscles, enabling movement and physical responses.
- Autonomic nerves control the automatic functions in various organs and systems, regulating activities such as breathing, heartbeat and digestion.

The ability to perceive sensory information accurately from the external environment relies on various specialized peripheral nerve fibers. These fibers are categorized into four subclasses based on characteristics such as diameter, propagation speed that refers to the speed at which electrical impulses travel along a nerve fiber, and level of myelination which is the process of coating the axons of nerve cells with a fatty substance called myelin, which helps to increase the speed at which electrical signals travel along the nerve fibers. One subclass, the C-type or unmyelinated fibers [13] are responsible for transmitting sensations like temperature, burning pain, and itch from the periphery to the central nervous system. C-type fibers have the slowest conduction velocity among nerve fibers and make up the majority part of the peripheral nerves. In electrophysiology, these fibers are often studied using a technique called microneurography, which helps researchers gain a deeper understanding of how sensory information is transmitted to the central nervous system. Due to their slow conduction, C-type fibers generate distinct signals that can be detected and analyzed in electrophysiological recordings, offering valuable insights into pain perception, thermal sensation, and other sensory experiences.

2.2 Electrophysiology and microneurography

In neuroscience, electrophysiology refers to the techniques for recording electrical activity in neurons and neural ensembles. One electrophysiological recording technique is microneurography [14, 15], which enables the recording of neural traffic from peripheral nerves. Microneurography has significantly advanced the understanding of C-fiber complexity, revealing potential nerve deficits and demonstrating how certain interventions can influence C-fiber responses.

Microneurography is a technique that involves inserting a microelectrode needle into a peripheral nerve to record individual signals (spikes and potentials) from peripheral receptors, allowing the examination of signal transmission from the periphery to the brain [16]. Although it requires inserting a microelectrode through the skin, microneurography is considered minimally invasive compared to previous techniques, which involved surgical procedures. This approach provides valuable insights into fundamental aspects of somatosensation, enabling the measurement of afferent (sensory, input) neuron responses to various stimuli and their correlation with subjective sensations. Notably, microneurography is the only method that directly assesses the function of both afferent and efferent (motor, output) C-fibers — the latter controlling smooth muscle, glands, and other tissues, and helping to regulate processes like sweating and blood vessel dilation — in awake human subjects.

A major challenge in this field stems from the fact that microneurography is an extracellular recording technique, meaning the recording probe is positioned near multiple nerve fibers rather than directly inside a single one, as in intracellular recording [8]. Precisely inserting the needle into these microstructures to isolate a single neuron fiber is nearly impossible. Consequently, multiple nerve fibers from the same small skin area may produce overlapping signals, and if the signal-to-noise ratio is low or the amplitudes of different nerve signals are similar, distinguishing individual nerve responses becomes challenging [15]. Clear, isolated recordings are rare, with only a few examples available.

In *in vitro* recordings - conducted outside of living organisms - certain tools, such as template matching or clustering, can assist in identifying distinct neural signals [11]. However, these methods rely on consistency in spike shapes. In recordings from awake humans or animals, even small micromovements can alter the shape of nerve responses, leading to unreliability in spike shapes.

To address this in microneurography data, a technique known as the marking method

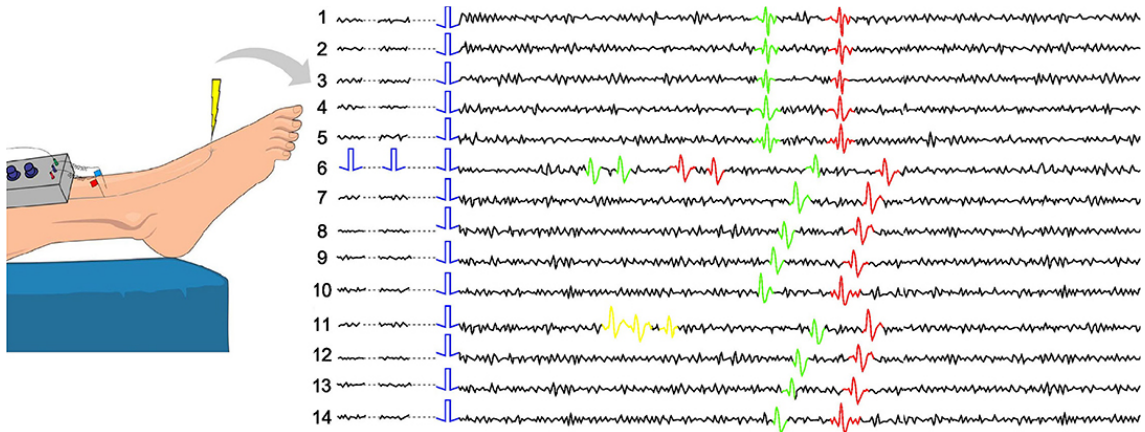


Figure 2.1: Microneurography data characteristics and marking method. The source of the figure is from article [8]. The blue marks correspond to the stimuli, the green and red colored action potentials are the responses of two distinct nerve fibers to the stimuli. The yellow mark is an instance of a spontaneous action potential.

[17] is used. This method involves applying repeated, controlled low-frequency background stimulation to the nerve endings to evoke consistent responses from specific nerve fibers near the electrode. By using such stimuli, signals from fibers farther from the electrode are less likely to interfere, effectively isolating responses from nearby fibers. Furthermore, because neural responses can be influenced by various external factors, for example, the response speed can slow due to repetitive stimulation — a phenomenon known as activity-dependent slowing (ADS) -, the marking method helps create a controlled environment where the response pattern is predictable, and the activity-dependent changes are clearly attributed to the manipulation or intended stimulation rather than external factors. However, a drawback of this background stimulus is that smaller responses may fall below the detection threshold, potentially leading to data loss.

2.3 Data characteristics

The data used to develop the detection framework is described in [8]. The microneurography recordings were made on isolated peripheral nerve fibers from mice and rats, as well as human peripheral C-fibers. In the *in vitro* animal data, the noise level is very low, in contrast to the *in vivo* data, which often faces the challenge of multi-fiber recordings. However, in rare cases where needle insertion is exceptionally precise, the recording can capture an individual fiber with greater clarity.

All datasets were recorded with the marking method [17], where background electrical stimulation is typically applied at frequencies between 0.25 and 0.125 Hz. This technique

is illustrated in Figure 2.1, where the sequential signal is segmented into smaller parts, and these parts are placed vertically in a way, that the background stimuli times are aligned in one vertical line. The paper [8] specifies that “main latencies” refer to the time between an individual background stimulation and the response to that stimulation (as shown in the first five rows). When additional stimuli (e.g., mechanical or electrical, as seen in the sixth row) are applied without allowing enough time for the nerve to recover, “extra latencies” occur due to ADS.

The background stimulation is applied at a constant frequency, while an additional stimulus is occasionally applied between instances of the background stimulus. The action potentials, or spikes, represent responses to either the background stimulation or the additional stimulus. Ideally, these spikes are distinguishable from the background noise by differences in amplitude and shape. Additionally, the main latency serves as a guide for identifying the location of the responses with extra latencies. Although spontaneous action potentials can occur (indicated by the yellow mark in Figure 2.1), they should be rare in healthy fibers. Therefore, in theory, the number of these spikes should approximately correspond to the number of stimulation pulses.

The data fed to the model is labeled based on expert knowledge. For animal recordings, which are less noisy than human recordings, analysis is performed using Spike2 [18], with templates, thresholds, and visual expert matching. In human recordings, manual tracking was conducted as follows.

As previously mentioned, the shape and intensity of action potentials can vary over time. However, since the number of spikes is expected to roughly match the number of stimuli, a manual "waterfall" technique is applied based on expert knowledge. This involves segmenting the 1D signal by each background stimulus and aligning these segments vertically, as seen in Figure 2.1. Once aligned, the unambiguously detected spike occurrences corresponding to a specific nerve fiber are connected by a vertical line, forming a "waterfall" pattern. This line connection helps to identify and reveal the faint spikes - that are weakly formed and fall below the noise level. From a machine learning perspective, this approach adds complexity. Faint spikes labeled as valid may introduce ambiguity, potentially confusing the model during training and in real-world applications.

For initial results, one noisy in-vivo human recording will be used, where a two-fiber marking method was applied. Two electrodes were placed close to each other on the skin, stimuli were delivered to capture by the receptors, and responses were recorded from both fibers. The recording was captured at a sampling rate of 10 kHz, resulting in

approximately 8 million data points over a 13-minute duration. The ground truth includes a total of 267 stimulation instances resulting in 536 action potentials from the two fibers. From the point of detection, the number of action potentials in this amount of data points reveal that class imbalance is strongly present.

For the raw data and labeled spikes, Figure 2.2 provides a deeper insight. The first window presents a zoomed-out view, clearly showing the raw data as the lower-ranged gray data mass, as well as the emerging spikes of stimuli or action potentials from the noisy data. The next three windows provide insights at progressively different zoom levels. Across all of them, the challenges in handling this data are evident: the stimulus timestamps appear earlier than the corresponding spike events in the recording, and the second type of action potential is often indistinguishable from the background due to the faint spikes identified by the "waterfall" technique.

The article [8] demonstrates that, due to ADS affecting all fibers similarly, the order of responses from different fibers remains consistent. If fiber A responds before fiber B, this sequence will always hold, as any latency introduced by stimulation will impact both fibers approximately proportionally.

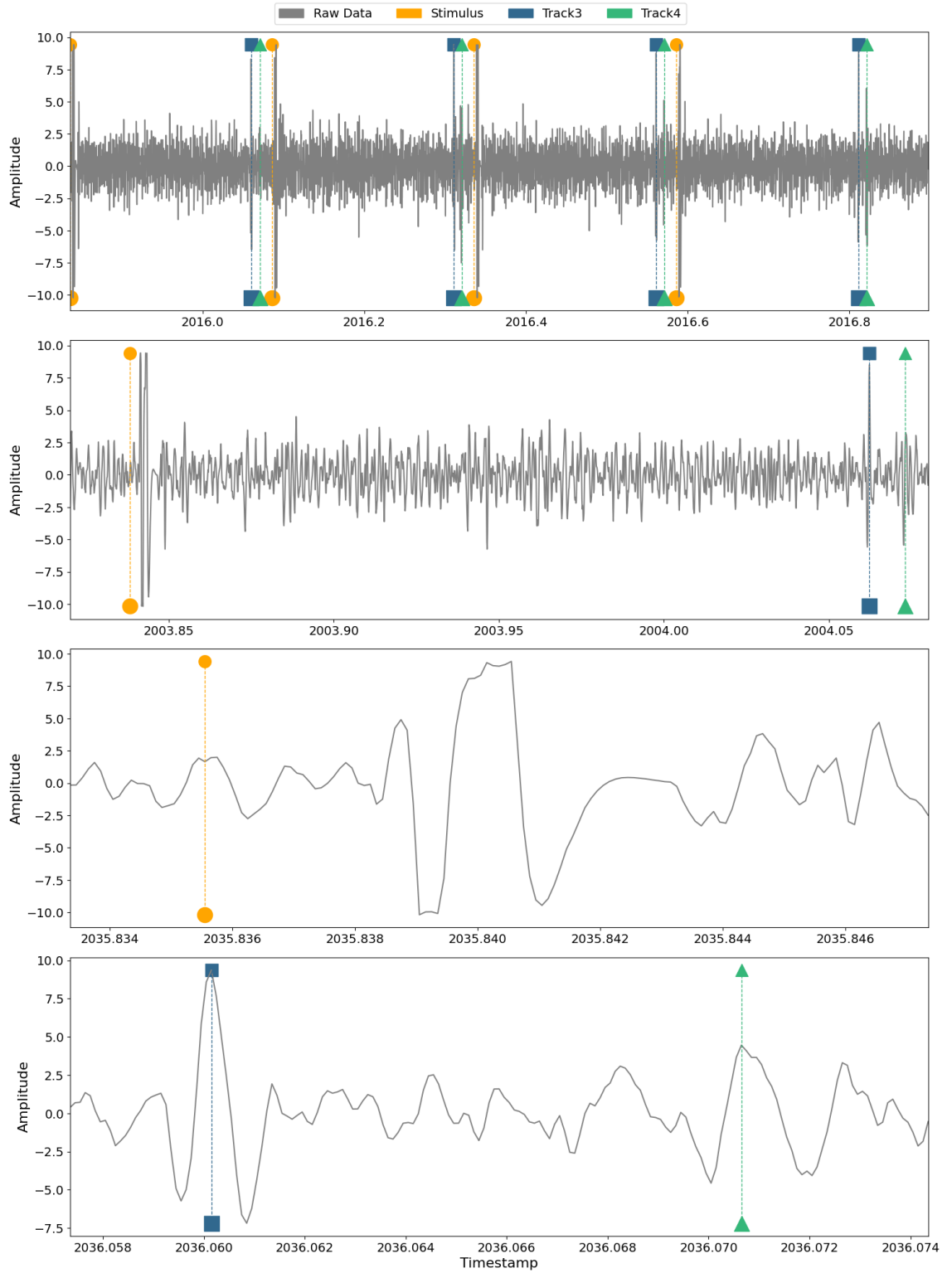


Figure 2.2: Raw data and ground truth labels at different zoom levels.

Chapter 3

Problem statement and solution plan

3.1 Review of previous work in microneurography spike identification

Since the technique of microneurography came into use, its analysis methods have undergone the same evolution as those for other types of data. Initially, analysis was conducted manually. Over time, new methods were developed to automate certain parts of the process, eventually leading to machine learning-based approaches. As previously mentioned, labeled microneurography data can be used for detection or spike-sorting. For most spike-like data, the detection phase involves straightforward manual filtering. However, in microneurography, due to its unique characteristics, filtering does not always serve as a universal solution, necessitating a return to labor-intensive manual efforts.

The literature indicates that there is currently no state-of-the-art solution for the detection phase of microneurography data analysis using deep learning. Detection is still performed manually, and only once the spike occurrences are identified can spike-sorting begin.

In contrast, spike-sorting has a broader range of examples documented in the literature. Since both detection and sorting methods rely on the nature of the spikes, exploring this may help with the lack of automated detection solutions with insights into how spike shapes are analyzed.

For spike-sorting techniques, [19] provides a comprehensive overview, outlining the core steps and the challenges faced at each stage. Spike-sorting generally begins with filtering the raw data using a thresholding bandpass filter to isolate relevant frequency bands, followed by spike detection, typically based on an amplitude threshold. Once spikes are

detected, the next step is feature extraction to characterize each spike's properties, enabling clustering into distinct groups. Ideally, each cluster should represent the action potentials from a single nerve fiber.

Feature extraction methods vary widely. For instance, some techniques use local maxima and minima as features. However, spikes with similar peak amplitudes can still belong to different nerve fibers due to shape differences. Template matching is another approach, though it can require significant manual input. The number of possible features is high, and to get the most important ones and to reduce complexity, dimensionality reduction techniques such as Principal Component Analysis (PCA) are frequently applied. However, studies indicate that wavelets outperform PCA by capturing more detailed spike shapes. Despite their potential, wavelet-based methods require selecting specific wavelets that match spike shapes, making this process less adaptable. Furthermore, action potential shapes can change over time, necessitating frequent template adjustments, which adds to the manual workload.

The final stage of spike-sorting involves clustering. Clustering methods range from manual techniques - which are limited by human perception to two or three dimensions - to more automated approaches, like Bayesian clustering, hierarchical clustering, nearest neighbors, neural networks, and support vector machines. However, no single method is universally accepted, as each technique's effectiveness depends heavily on the specifics of the data. This variability underscores ongoing debates within the community regarding the most effective approach to spike-sorting.

Only a small number of articles have demonstrated working algorithms specifically for microneurography data and their effectiveness, and all of them are based on statistical or traditional feature extraction methods, which can later be used as input to automated systems.

The uniqueness of [20] lies in a new modified wavelet transform that is used in a de-noising procedure, followed by an action potential detector and a classifier based on waveform characteristics. The method is tested on simulated signals, with comparisons drawn between different parameterized approaches. The analysis is conducted in MATLAB using built-in functions, and wavelets are chosen based on their similarity to the target waveform shape. The de-noising technique applies a threshold to the wavelet transform to separate noise from signal components. The results show promising performance at high signal-to-noise ratios (SNR), though accuracy decreases as SNR lowers.

The work [21] also includes with wavelets, however, the novelty lies in the construc-

tion of a new wavelet instead of a modification of a previously existing one. The new wavelet matches the shape of the mean of the detected action potential templates. After applying the wavelet transform, thresholding and local maximum analysis are used to identify spikes. This approach has been tested on both real and simulated data, where the newly constructed wavelet demonstrates improved performance over previously existing wavelets.

In other studies, different approaches were conducted. In [22], the authors highlight the limitations of spike-sorting algorithms that rely solely on spike shape, as they tend to have high error rates when spikes overlap, change in shape, or vary in strength. They compare their new technique with Hammarberg's approach [23], where the "waterfall" display of microneurography data is processed in three steps as detection, tracking, and parameter estimation. Hammarberg's method uses template matching focused on spike shape and relies on thresholding in the detection phase, making it sensitive to SNR and threshold selection. This method also struggles when spikes run closely parallel or cross each other. In contrast, the new technique presented in [22] distinguishes itself by making no assumptions about spike shape during detection. Instead, it assumes that stimulus-correlated spikes have more strength along linearly aligned sub-signals, considered locally. Once tracks are detected, they serve as templates for further spike discrimination. This approach enhances signal contrast and enables accurate detection even in noisy, multi-unit recordings. By focusing on local linear alignment and the detected spikes are compared to each other on time, this method effectively filters out spontaneous neural firings that are not correlated with the stimulus.

There is also an open-source framework built especially for microneurography data analysis. OpenMNGlab [24] is using Spike2 and Dapsys - the same two frameworks that were used for the ground truth labeling in this TDK thesis' data. It provides opportunities for data visualization, fiber tracking, and a database for the features of machine learning.

In this work [9] by the Digital C-Fiber group in Germany, with whom we collaborated during the TDK project, they demonstrated that by using various feature extraction techniques and a support vector machine (SVM), they achieved 96% accuracy. The test data is the same as in this study, with a 10,000 Hz sampling rate and recordings of action potentials from two fibers. From the raw signal, they extracted a 3 ms (30-sample) time window around each identified action potential. Several features were then extracted, including spike amplitude and width; shape, phase, and distribution features from the spike-sorting method SS-SPDF using first and second derivatives; PCA on SS-SDFA features;

raw waveform features (the 30 extracted data points); and PCA of the raw components. These features were computed for all spikes and subsequently used as input for classification by SVM. An individual classifier was created for each feature type using the default radial basis function (RBF) kernel, and the results were compared to determine which feature provided the best performance based on the metrics of the fiber class predictions. Among all features, the raw waveform features yielded the highest performance, achieving 96% accuracy, precision, recall, and F1-score. The second-best results were for PCA of raw waveforms, with 92% accuracy, indicating that the waveform features had the strongest performance overall.

Additionally, the Digital C-Fiber group reported promising initial outcomes for spike-sorting using the Variable Projection Neural Network (VPNet) model [10] on their data, even without hyperparameter optimization [25]. They highlighted that these results are particularly unique due to the minimal amount of data required for the model to learn effectively. This early success suggests that VPNet may serve as a flexible approach for microneurography data analysis, not only reducing manual workload but also achieving good results with significantly less data, while maintaining interpretability — in contrast with traditional deep learning solutions.

3.2 Automated system plan

Building on the results of the Digital C-Fiber group, this project addresses the demand for a solution to detect action potential instances in microneurography recordings with a neural network. The approach is based on the weighted Hermite Variable Projection Neural Network (WHVPNet), an improved version of VPNet. Both VPNet and WHVPNet have already shown effectiveness in signal processing tasks. VPNet was applied to classify ECG recordings to distinguish between normal and abnormal signals [10]. WHVPNet has been used with EEG recordings for binary classification to differentiate brain responses to visual stimuli, categorizing signals based on whether the patient viewed red or green flashing lights [12]. In this work, the WHVPNet is used again for binary classification, providing a detection solution where positive instances correspond to spike occurrences.

The WHVPNet is implemented in PyTorch [26]. The data processing, training, and decision analysis fall under the scope of this project and are implemented in Python. For the explainability analysis, the iNNvestigate framework [27] is utilized. iNNvesti-

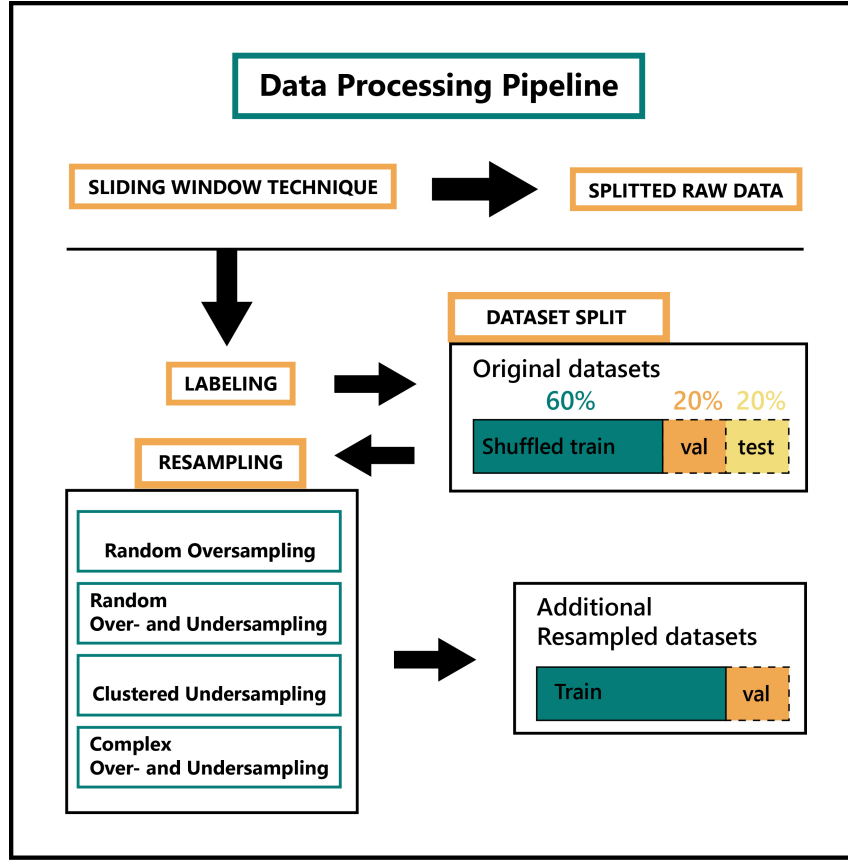


Figure 3.1: Steps of the data processing pipeline.

gate provides a standardized interface for commonly used XAI algorithms, simplifying their evaluation and comparison; however, it does not include methods for interpreting the results of these algorithms. The explanation figures were designed using examples from [28] and custom methods. Since iNNvestigate supports only Keras [29] models, the WHVPNet was translated into TensorFlow [30]. After verifying that the results are consistent with the PyTorch implementation, the decision analysis is conducted using the TensorFlow model.

3.2.1 Data processing pipeline

The logic of data processing (Figure 3.1) and loading was created by considering both the characteristics of local data segments and the structure of the whole dataset. In this setup, the data from Namer’s microneurography lab consists of timestamp-amplitude pairs, each representing the measured amplitude at a given time instance. Additional timestamps are listed for stimulus events and expert-identified action potentials.

The recording should be divided into overlapping windows using a sliding window technique, producing subarrays from the original sequence. Both the window size and

overlap amount are parameters to be fine-tuned for optimal performance. The spikes are not really close to each other compared to the amount of data points, so with a reasonable window size, in one window there should be maximum one spike.

The model classifies each window of the signal as either 'background' (Class 0) - indicating the raw signal or background noise - or 'spike' (Class 1) - representing the stimulus spikes and the action potential responses. The final output of the framework will be a plot of the original recording (without overlapping), where evaluated segments are color-coded based on the model's predictions indicating the likelihood of a spike occurrence. Prediction percentages will be grouped into bins and assigned colors, with each range representing a different likelihood level. This visualization will highlight sections where the model detects a spike, making it easy to see where each segment falls on the likelihood scale.

To make the model interpretable, the integrated gradients algorithm [31] - a widely accepted post-hoc interpretation method is used. XAI algorithms produce relevance scores — outputs with the same shape as the input — demonstrating how specific parts of the input contributed to the model's final decision. These relevance scores, when visualized independently or alongside the original inputs, reveal important aspects of the model's decision-making process.

Although the model itself does not distinguish between different types of nerve fiber responses, the dataset includes this information. By generating multi-class labeling, it can help to reveal whether the model behaves differently for various types of spikes — even though they share a common class label — by analyzing relevance scores grouped according to the multi-class labels.

Dataset balancing

The drawback of the rare spike events is that there are far fewer windows containing actual spike instances compared to those with only background noise. Depending on the window size and overlap settings, the proportion of spike-related windows is around $\sim 0.3\%$. To improve model performance, multiple strategies commonly used for imbalanced datasets should be considered.

First, downsampling or upsampling could be used. Downsampling is the process where not all samples from the majority class are used, but only a subset, in order to match the number of samples from the minority class. On the other hand, oversampling focuses on the minority class, increasing its representation in the dataset. This can be done

by duplicating the minority class samples either in their original form or with transformations, such as horizontal or vertical flipping, noise addition, color manipulation (in case of images), etc. The drawback of downsampling is the potential loss of data, as it is held that in lucky cases when more data is available, the best practice is to use as much of that as possible. In contrast, oversampling can result in a significant increase in the dataset size, which may lead to heavier training processes.

To overcome the drawbacks of simple downsampling and oversampling, several variations can be applied. For instance, to minimize data loss during downsampling, k-nearest clustering can first be performed on the majority class. Then, from each cluster, an equal number of samples can be selected for the undersampled dataset, thus preserving the variability among the clusters. Alternatively, a combination of downsampling and upsampling can be used, where the majority class is downsampled while simultaneously upsampling the minority class. This approach can help balance the dataset by adjusting both classes effectively. Additionally, during the selection process — whether for downsampling or upsampling — probabilistic methods can be applied to determine the number of instances to sample from each subgroup. For example, if the positive class could be subdivided based on different nerve fiber responses, a parameter could be introduced to increase the multiplication ratio of the first type of nerve fiber signal compared to the second type of nerve fiber signal in the final Class 1 representations.

The batch size becomes a more critical factor in the case of imbalanced datasets compared to balanced ones. This is because the batch size must be chosen such that each batch contains representations of all possible outcome classes. With a smaller batch size - despite some studies suggesting that smaller batches can often have better results - there is a high likelihood that many batches may not contain any samples from the minority class, which can lead to poor model performance.

Few-shot learning [32] is a branch of machine learning that addresses scenarios where a model must be trained on a dataset with only a few positive instances. The most extreme case is one-shot learning, where only one single instance of the positive class is available, such as in face recognition. Few-shot learning typically handles this problem at the model level by employing a support and query set. By comparing the support set (containing known examples) to the query set (containing new, unseen samples), the model can determine whether the current sample is similar to the example(s) in the support set or not. The prediction outcomes will be the percentages generated by the model, based on the similarities between the support and query sets. On the data level, few-shot learning often relies

on the common solutions for imbalanced datasets - even the sliding window technique is accepted as a step to handle imbalanced datasets effectively.

At the model level, the loss function can be further optimized. For instance, in PyTorch's implementation of cross-entropy loss, a weighting parameter allows for assigning greater importance to one class over another. Additionally, focal loss [33] — an extended and parameterized version of cross-entropy loss — was specifically designed for imbalanced datasets.

All of these methods have proven effective for imbalanced datasets; however, only through experimentation can it be determined which approach is best suited for a specific case.

Chapter 4

Design decisions and implementation

4.1 Data Processing - From raw data to labeled samples

The dataset provided by Namer's microneurography lab is organized into three distinct components:

- Sensor measurements:

The primary data consists of sensor recordings, arranged in two columns: timestamps, and amplitude values corresponding to the recorded neural signals at the timestamp's moment. The data was sampled at 10 kHz, resulting in 10 samples per millisecond.

- Stimulus evaluation timestamps:

The second component has a single column listing timestamps when background or additional stimuli were triggered.

- Action potentials (responses to stimulus):

The third data has two columns. The first column records the timestamps of detected action potentials (around the middle of the spike) that were identified as responses to the applied stimuli, and the second column indicates the nerve type responsible for each specific response. This information was determined by expert analysis.

The detailed steps of the data processing pipeline is visible in Figure 4.1. The first step is the implementation of the sliding window technique, with fine-tuning the window size and overlap ratio parameters. The original sequence to which the sliding window technique is applied should result in a 2D matrix, where consecutive rows share samples

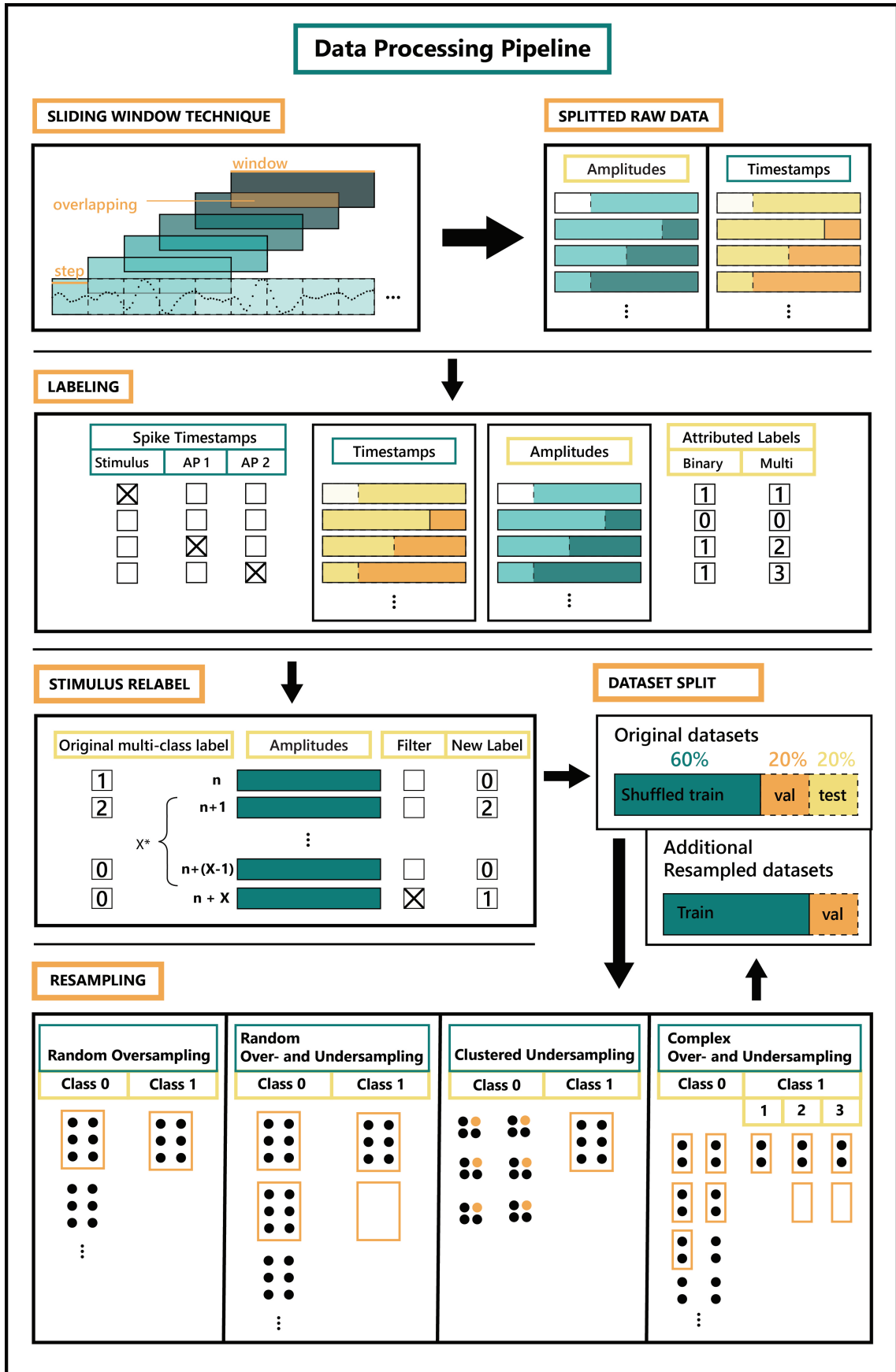


Figure 4.1: Data processing steps from raw data to model-ready input.

based on a specified overlap ratio. In previous work by the Digital C-Fiber group [9][25], the data was divided into 30-sample windows (without using a sliding window technique or overlap), based on the observation that action potentials could be fully captured within 30 sample points. Additionally, in the first dataset that was used for training and testing, the minimum distance between two spikes is 48 samples — though, in practice, this distance is mostly larger. Based on this information, training was tested with window sizes ranging from 15 to 48 samples. Given how rare the action potentials in the dataset are, higher overlap was necessary to ensure that potential occurrences were adequately captured. Consequently, overlapping ratios of 50% and 75% were tested with all window sizes.

Although the raw data consists of timestamp-amplitude pairs, only the amplitude values are used as input to the model. However, the timestamps are still essential in generating ground truth labels. The sliding window split is applied separately - but with the same parameters - to both the amplitude and timestamp data. This separation allows the amplitude windows to be prepared for model input, while the timestamp windows can be used in the next step for generating ground truth labels.

The detection focuses on action potentials, however, the high amplitude of the stimuli cannot be ignored by labeling them as background, as this could confuse the model. Therefore, the labeling process was as follows:

Binary labeling approach:

- Class 1 : If any timestamp (action potential or stimulus) is present within a timestamp window, the corresponding amplitude window is labeled as Class 1
- Class 0 : If no relevant timestamps are found

This labeling process is extended to create an additional label dataset, as multi-class labeling, which can be useful for interpreting model outputs.

Multi-class labeling for interpretability:

- Class 1: When a stimulus timestamp falls within a timestamp window.
- Class 2: If an action potential timestamp is detected for one type of nerve fiber.
- Class 3: If an action potential timestamp is detected for another type of nerve fiber.

These multi-class labels are only used for interpretability purposes, they are not included as inputs during model training.

For labeling, another issue has been identified that needs to be addressed. Although action potentials are manually detected, the stimulus timestamps are automatically generated based on when the stimulus was triggered. However, there is a slight delay as the signal takes time to arrive at the probe inserted into the nerve. At a 10 kHz sampling rate, the difference between the trigger time and the artifact — when the signal reaches the nerve and forms a spike in the recording — becomes significant. As a result, the signal corresponding to the high stimulus appears slightly delayed in the data compared to the marked timestamps.

On visual examination, the stimulus amplitude stands out, representing the largest range among all signals, with the highest values in both positive and negative directions. However, a statistical analysis of the absolute maximum values across labeled samples (in the multi-label scenario) has the following results:

Label	Min Value	Max Value
0	-10.243	9.476
1	-4.705	5.057
2	-9.929	9.464
3	-8.474	8.723

Table 4.1: Minimum and maximum values for each label

These results indicate that Label 1, which corresponds to the stimulus, has the lowest absolute values, while Label 0, corresponding to the background noise, has the highest. This finding led to further investigation, revealing that, on average, the actual signal appears around 40 samples - equivalent to 4 milliseconds - after the marked stimulus timestamps. Given the small window sizes, this delay significantly impacts the correctness of the labeling.

As a result, a stimulus relabeling procedure was applied and tested. Based on previous observations, the filtering criteria for individual windows were set as follows:

- There is at least one value in the window that is less than or equal to -10, and at least one value in the window that is greater than 9.
- The current window's index is less than or equal to the last original stimulus window's index + X .
- The window is previously labeled as background or 'stimulus' (not as action potential from either nerve fiber)

If all these conditions are met for the current window, it is labeled as a stimulus. For windows initially labeled as stimulus that do not meet these criteria, the label is changed to background. The threshold values of -10 and 9, along with the 'and' operator, capture the middle of the spike, as seen in Figure 4.3, where the -10 and 9 values appear in 3 data points. This is similar to the manual detection of action potentials, where timestamps are selected from around the middle of the signal. The parameter X depends on the window and overlap size, as well as the arrival time of the artifact. For instance, with 15-sample windows and an 11-sample overlap, $X = 10$, as actually each window processes 4 new data points and the stimulus spike is around 40 samples away from the marked timestamp.

During relabeling, the program checks to ensure no window originally marked as an action potential is mistakenly relabeled, although no instances of this occurred. In the original dataset, the number of stimuli and action potentials for both nerves is nearly balanced, and this ratio is expected to remain approximately the same after relabeling. This ratio is checked after relabeling to confirm the filtering method's reliability, and the method with these parameters satisfies these conditions.

Figure 4.2 shows the first four originally stimulus-labeled windows. The y-axis limits are set between -10 and 10 so that the relabeled stimulus in Fig. 4.3 can be compared effectively. The plots clearly show that the originally labeled stimulus windows have a much lower range than the relabeled ones and that there are no spikes in the windows originally labeled as stimulus.

These windows represent 15-sample segments with 75% overlap, meaning that, in the best-case scenario, a single stimulus is captured up to four times, as observed in these cases. Verifying the timestamps shows that the third timestamp of the first window aligns with the first timestamp of the next window, and so on. This pattern is consistent across all four windows in both the original and relabeled stimulus timestamps, confirming the accuracy of the relabeling process. Additionally, the difference between the original timestamp and the detected occurrence is 3.3 ms, also aligning with the statement of up to ~ 4 ms artifact arrival time. The algorithm effectively identified the actual spike events in the same manner as they were originally marked with timestamps.

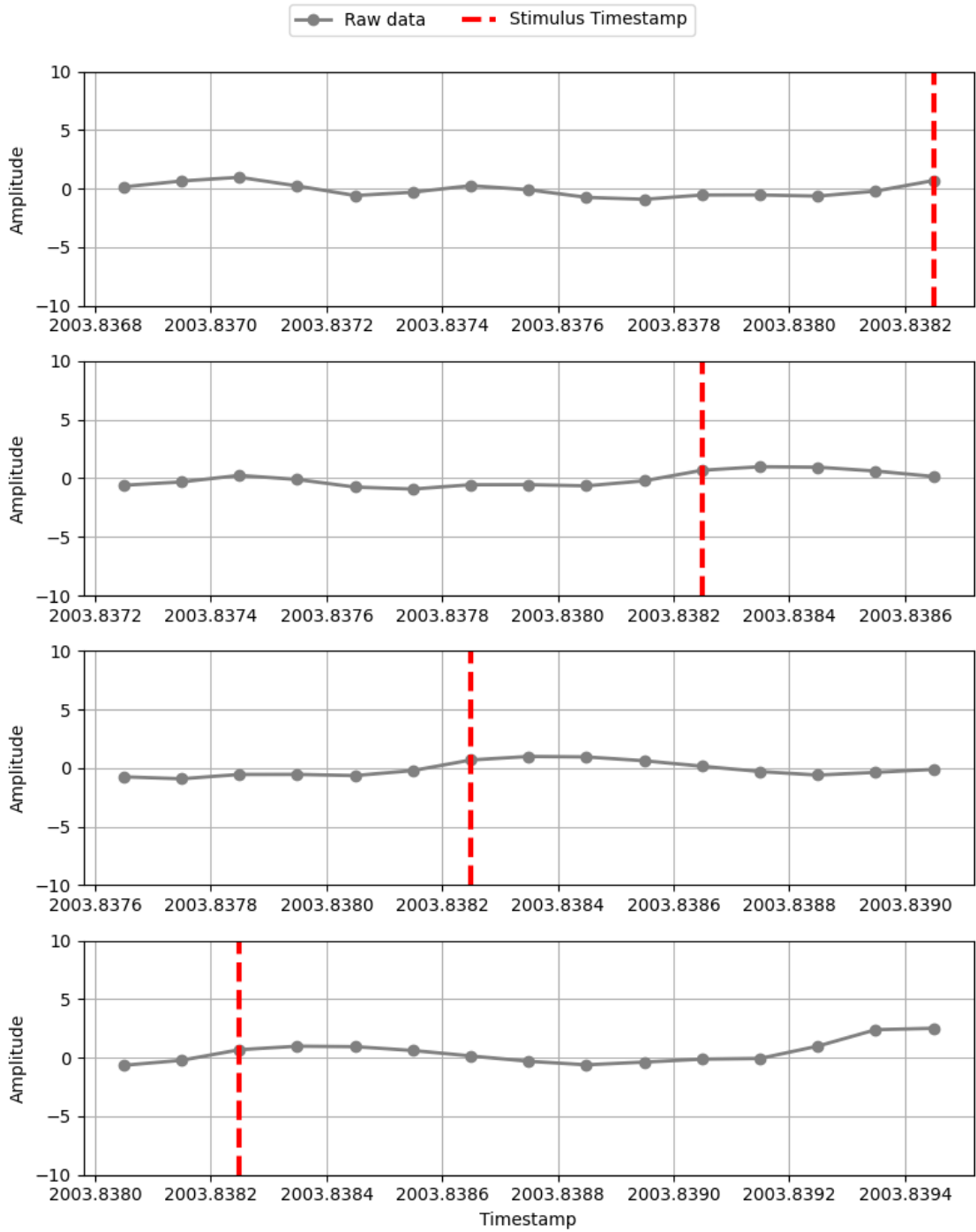


Figure 4.2: The first four windows originally labeled as stimulus windows.

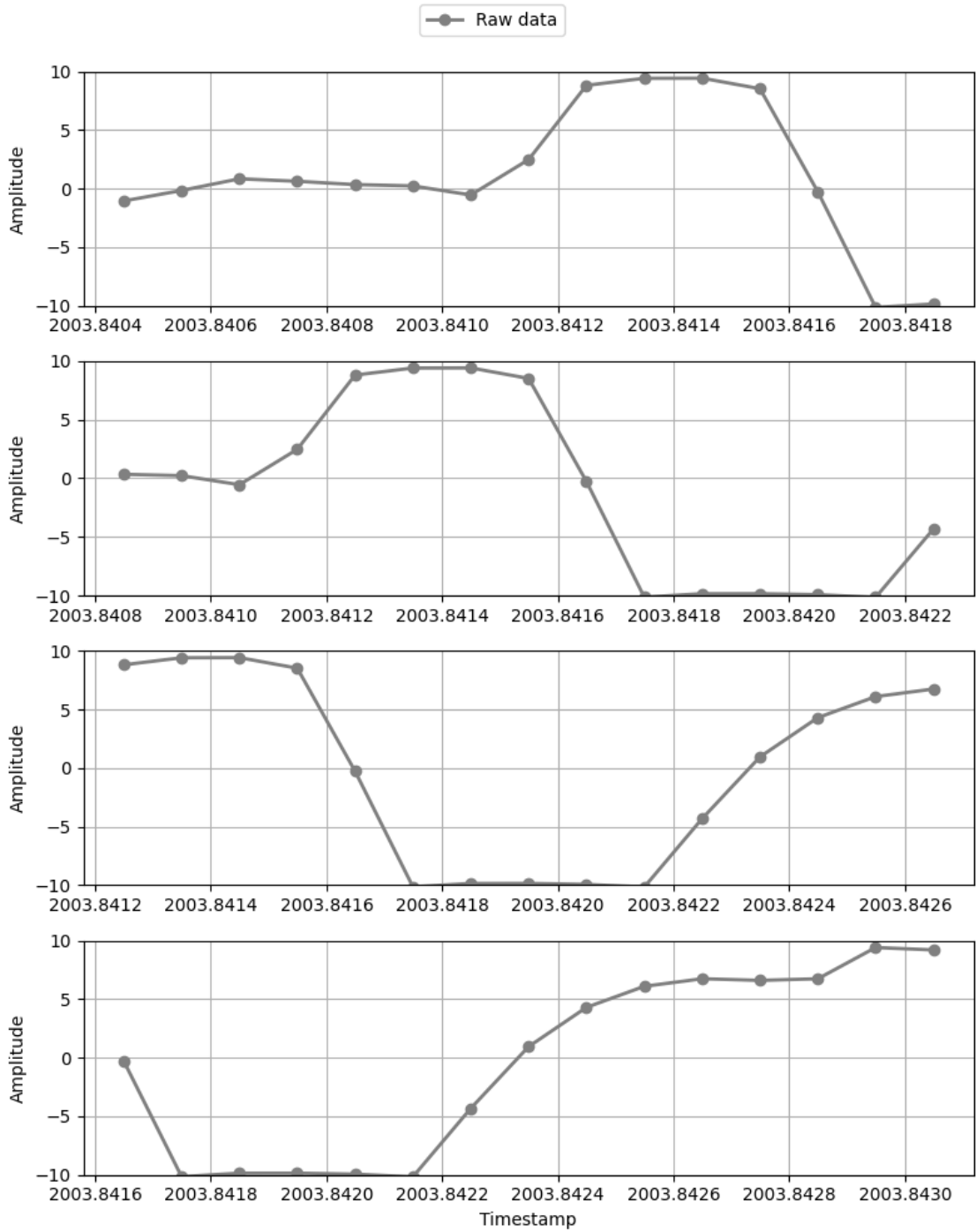


Figure 4.3: The first four windows labeled as stimulus after relabeling.

4.1.1 Handling imbalanced dataset during preprocessing

In an imbalanced dataset, the majority class is the predominant label (Class 0), while the minority class represents the less common label - in this case the action potentials (Class 1). There are three levels of imbalance based on the ratio of minority class occurrences. A dataset is considered mildly imbalanced when the minority class constitutes 20–40% of instances, moderately imbalanced for ratios between 1 and 20%, and extremely imbalanced when the minority class is less than 1%. Given that reasonably sized and overlapped windows alone cannot increase the original dataset's minority class ratio above 1%, the data is an extremely imbalanced dataset here.

As outlined in Section 3.2, various combinations of oversampling and undersampling are possible and implemented in the framework:

- **Random undersampling:**

From the indices of Class 0, a number of indices equal to the number of samples in Class 1 are randomly selected.

- **Random oversampling and undersampling:**

params: `upsample_count`, `downsample_count`

Class 0 is undersampled to the amount of `downsample_count`, and Class 1 is oversampled to the amount of `upsample_count`. The indices are drawn or permuted randomly.

- **Complex oversampling and undersampling:**

params: `upsample_count`, `downsample_count`

Class 0 is undersampled to the amount of `downsample_count` samples, and Class 1 is oversampled to the amount of `upsample_count` samples. The Class 0 indices are drawn randomly, while the Class 1 indices are chosen as follows: all original samples are included once. Additional samples are selected only from the action potential samples (excluding stimulus labels). Optionally, probability values can be set for the two action potentials to control the likelihood of selecting indices from the different nerve fiber samples.

- **Clustered undersampling:**

param: `cluster_num`

The Class 0 samples are clustered into `cluster_num` clusters. An equal number of indices is drawn from each cluster to match the number of indices in Class 1.

Due to the large imbalance between the two classes, basic oversampling is not feasible in the framework, as the resulting data size would be rather unmanageable.

4.1.2 Dataset split structure

Given that the window sizes are small relative to the entire recording and result in a larger amount of data, the dataset is divided into training, validation, and test sets in a 60-20-20 ratio.

Since this is a time series analysis, careful handling of shuffling is essential. Shuffling the training data enhances the model's robustness and improves its generalization across diverse patterns. However, for the validation and test sets, it is critical to preserve the original sequence, as it reflects the natural timing of real-world data. Based on this, the validation and test sets should be continuous data with sequential indexes, and not randomly selected samples with shuffling.

However, this case is more complicated, because the class imbalance should be handled. The training data needs to be rebalanced so that the model can learn from it more effectively. Additionally, a rebalanced validation set could be useful at first, to examine if the model is able to perform well on a dataset similar to the one it was trained on. Then, validating it on a real-life dataset could show the model's performance in its final intended application.

To accommodate these dual requirements, both the original and resampled sets are made available for validation, while for testing, only the original time-series sequence is available. This setup allows for flexibility in experimenting with various imbalance-handling techniques while ensuring the model's performance is evaluated on data that closely resembles real-world conditions.

All in all, the available datasets:

- Shuffled training data, without resampling.
- Shuffled, resampled training data.
- Resampled validation data.
- Unshuffled validation data with preserved timing.
- Unshuffled test data with preserved timing.

4.1.3 Preprocessing pipeline overview

To summarize the entire preprocessing pipeline:

- **Sample generation:** The samples are created using the sliding window technique, with specified parameters for window size and overlap.
- **Label assignment:** Labels are generated with or without the stimulus relabeling option. Both binary labels (for model training) and multi-class labels (for data examination) are assigned to each sample.
- **Data splitting:** The data is divided in 60-20-20 for training, validation and testing.
- **Rebalancing and shuffling:** The selected rebalancing technique is applied to the training and validation sets.
- **DataLoader creation:** DataLoaders are created for each set with three components: samples, binary labels, and multi-class labels.

4.2 Model design and architecture

4.2.1 Weighted-Hermite VPNet

Model-agnostic deep learning models only rely on the provided data to learn the relationships between input samples and output labels, without extracting valuable domain knowledge from the data. While these models have shown promising results, enabling the widespread adoption of deep learning, they tend to produce inconsistent predictions on unseen data, and additionally, their decision-making processes remain largely unexplained. To address these limitations, the knowledge-augmented deep learning paradigm [34] was developed, where the model is not solely dependent on data, but also integrates domain knowledge into its pipeline.

The Variable Projection Neural Network (VPNet) [10] instantiates a knowledge-augmented approach, providing a framework for solving layer-wise separable nonlinear least squares (SNLLS) problems using variable projection. Variable projections simplify and speed up the solution of nonlinear least squares problems by projecting the nonlinear components of the problem onto a lower-dimensional space, thus reducing the dimensionality of the problem and making it easier to solve. SNLLS enhances this process by

separately optimizing linear and nonlinear parameters, improving convergence through complexity reduction.

VPNet employs a feedforward neural network architecture, where the first layer, called the VPLayer, applies the variable projection operator. This layer orthogonally projects whole input data into linear combinations of adaptive Hermite functions.

The adaptive Hermite functions are considered as follows:

The fundamental parts are the Hermite polynomials $h_k(x)$, $k \in \mathbb{N}$, and are recursively defined as:

$$h_0(x) = 1, \quad h_1(x) = 2x, \quad h_{k+1}(x) = 2x \cdot h_k(x) - 2k \cdot h_{k-1}(x) \quad (4.1)$$

whose derivatives are given by:

$$h'_k(x) = 2k \cdot h_{k-1}(x) \quad (4.2)$$

These polynomials are pairwise orthogonal with respect to the weight $w(x) = e^{-x^2}$:

$$\langle h_k, h_j \rangle_w = \delta_{kj} \cdot \sqrt{\pi} \cdot 2^k k! \quad \text{for all } k, j \in \mathbb{N} \quad (4.3)$$

where δ_{kj} corresponds to the Kronecker-delta symbol. The Hermite polynomials, along with $\sqrt{w(x)}$ and the normalization factor, define the Hermite functions:

$$\phi_k(x) := \frac{1}{\sqrt{\pi^{1/2} 2^k k!}} \cdot h_k(x) \cdot e^{-x^2/2} \quad (4.4)$$

that form a complete, orthogonal system in $\mathbf{L}^2\mathbb{R}$, enabling signal approximation with arbitrary accuracy.

To make the Hermite functions adaptive, dilation (λ) and translation (τ) parameters can be introduced, that can modify the function in frequency and time localization.

$$\phi_k^{\lambda\tau}(x) := \sqrt{\lambda} \cdot \phi_k(\lambda(x - \tau)) \quad (x, \tau \in \mathbb{R}, \lambda > 0) \quad (4.5)$$

The transformed data - the coefficients of the Hermite components - then serves as input for a fully connected neural network. In classification tasks, the VPLayer acts as a trainable feature extractor, allowing the built-in structural interpretability of the model.

Although shifting and scaling can adjust the function to the data's pattern, it does not offer enough flexibility for data with complex morphological changes.

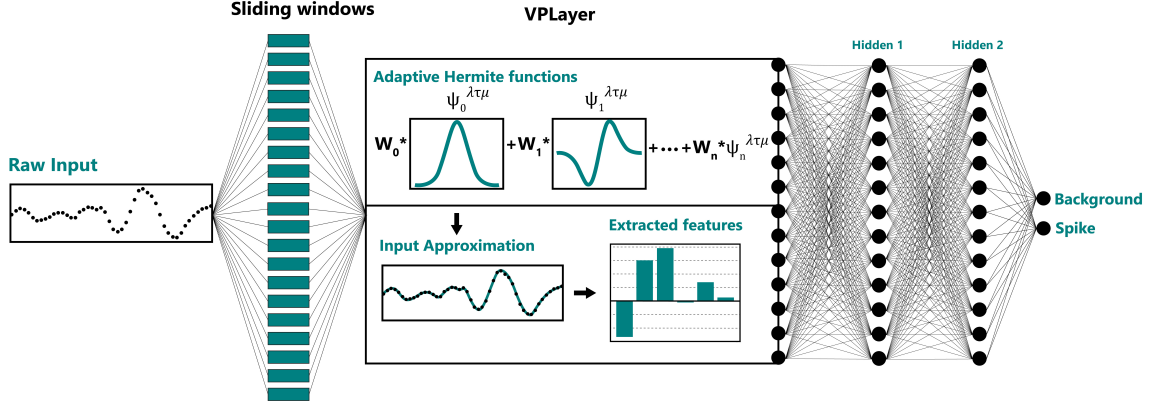


Figure 4.4: Weighted Hermite VPNet architecture

The weighted Hermite variable projection neural network (WHVPNet) [12] is built on VPNet by introducing nonlinear weight parameters for classical Hermite functions, leading to an even more generalized modification that ensures more flexibility by the ability to change to shape of the function. The architecture of the network is shown in Figure 4.4, where λ and τ are the affine parameters, and μ corresponds to the newly introduced trainable parameter of the function system.

Here, the adaptive Hermite functions are defined as:

$$\psi_k^{\lambda\tau\mu}(x) = \sqrt{\lambda} \cdot \hat{h}_k(\lambda(x - \tau)) \cdot \sqrt{Q(\lambda(x - \tau))} \cdot e^{-\lambda^2(x - \tau)^2/2} \quad (4.6)$$

where $\lambda > 0$ and $\tau \in \mathbb{R}$ are the affine parameters as previously, and Q is the weight modifier, a positive function. The polynomials $\hat{h}_k$ are orthogonal with respect to the inner product (in Eq. 4.3) $\langle \cdot, \cdot \rangle_v$ generated by the modified weight function $Q(x) \cdot e^{-x^2}$, where

$$Q(x) = Q^\mu(x) := \prod_{j=0}^{d-1} (x - \mu_j)^2, \quad (\mu_j \in (\mathbb{R} \setminus \beta)) \quad (4.7)$$

with $\beta \subset \mathbb{R}$ being the roots of the adaptive Hermite polynomials in Eq. (4.5).

Hermite functions are particularly effective at representing spike-like waveforms due to their compact nature. These nonlinear weights enable a basis function expansion (BFE), a technique that represents complex functions as linear combinations of predefined basis functions. Similar to how vectors are expressed in terms of basis vectors in linear algebra, functions in a space can be represented using basis functions. The Hermite-based BFE preserves orthogonality while enhances the representation power of the classical function, enabling WHVPNet to achieve more accurate predictions in tasks involving spike-like waveform classification.

4.2.2 Handling imbalanced dataset at the model level

In machine learning, loss functions are essential for evaluating how effectively a model predicts the correct outcomes. A commonly used loss function for binary or multi-class classification tasks is `CrossEntropyLoss`, that computes the error by comparing the predicted probabilities with the true labels.

In certain scenarios, such as in some fields or for class imbalance, assigning more importance to one class over others becomes essential. For example, in medical applications, the positive class often requires more emphasis because false negatives may have more severe consequences than false positives.

PyTorch's [26] `CrossEntropyLoss` function allows the use of a weight parameter to address class imbalance. This parameter is a 1D tensor containing weights for each class, where the length of the tensor matches the number of classes.

In the context of microneurography recordings, where class imbalance is a significant challenge, and minimizing false negatives is a priority over false positives, the weight parameter in `CrossEntropyLoss` becomes an essential tool for rebalancing the model's focus on the minority class.

By fine-tuning these weights, the model was optimized to achieve high true positive rates, however, at the cost of an increase in false positives. When a slightly higher weight was assigned back to the majority class, false positives indeed decreased, but this adjustment caused an undesirable rise in false negatives.

To address this trade-off, the distribution of prediction confidences was analyzed: For correct predictions, the model exhibited high certainty, with an average confidence of 98%. For incorrect predictions, the confidence is measured on a wider spread of range. Based on this analysis, the decision boundary for classification was shifted from the conventional 0.5 (used in binary classification) to a higher value - a technique often used for imbalanced datasets. This adjustment reduced the likelihood of false positives without compromising the accuracy of correct predictions, leading to improved overall precision.

4.2.3 Validation strategies and evaluation metrics

In model evaluation, a variety of metrics are used to assess performance, with details provided in the appendix (Section 6).

Accuracy is a common metric for evaluating prediction performance; however, in imbalanced datasets, where the minority class makes up less than 1% of the data, accuracy

can be misleading. For instance, if the model predicts only negative cases, accuracy might still exceed 99%, which falsely suggests near-perfect performance. In such cases, alternative metrics are more suitable.

Common metrics

Minimizing false negatives is particularly important in this project, as they are more detrimental than false positives. As a result, maximizing recall is a key objective. Another important metric used is the false positive rate, which represents the proportion of false positives among all negative samples. These two metrics are especially useful because they are less affected by an imbalanced dataset and allow for handling positive and negative cases distinctly.

Additionally, when recall is prioritized, the F2-score is a more relevant metric compared to the F1-score, as it places greater emphasis on recall over precision.

For imbalanced datasets, a modified version of accuracy, known as balanced accuracy, is often used. This metric calculates the average of sensitivity and specificity. In cases of class imbalance, balanced accuracy provides a more effective measure, as it evaluates the prediction accuracy of each class separately, unlike traditional accuracy, which equally weighs all predictions.

Merged metrics metrics

Since the input data is split into overlapping windows, one way to evaluate the model is to merge neighboring positive predicted windows into a single prediction and calculate the TP, FP, and FN instances accordingly. This merging technique is also applied to the ground truth labels, resulting in positive ranges for both the predictions and the ground truth. For example, one merged instance might start at a specific index and last for four consecutive windows.

If a merged range from the predictions overlaps in any way with a merged range from the ground truth, it is counted as a TP prediction. If a merged range from the predictions does not overlap with any range in the ground truth, it is counted as an FP. Conversely, if a merged range from the ground truth does not overlap with any range in the predictions, it is counted as an FN.

The lengths of these merged ranges are also statistically informative. Therefore, the output of the evaluation includes a breakdown of prediction quality, showing how many instances correspond to distinct range lengths.

Multiclass Label Analysis for Binary Predictions

Since the model only recognizes binary labels, while multiclass labels are in the raw data and further distinguish among Class 1 samples, it provides an opportunity for additional statistical analyses. These analyses evaluate how often multiclass labels were correctly or incorrectly classified. For multiclass labels 1, 2, and 3, a correct prediction corresponds to Class 1, while incorrect predictions are classified as Class 0.

This analysis revealed that the model struggled most with the stimulus labels prior to relabeling - as the original labeling was misleading - by incorrectly classifying them as Class 0 for almost all scenarios. Also, the second type of action potential is harder to distinguish than the first. In such cases, the resampling probability for the second type of action potential could be adjusted, assigning it a higher likelihood of being oversampled in the Class 1 dataset to improve model performance.

Summary of used metrics

The framework is adaptable to resampling of the input data, therefore all relevant metrics — including classic accuracy when the resampling results in a balanced dataset — are calculated during training and testing to provide comprehensive performance statistics for the model in various scenarios. The first focus, however, was on recall, as detecting rare events was the most critical aspect of this project, and the second focus was on FPR.

For the multi-class labels, counted values are represented for the different prediction qualities, to identify if the model struggles with one type of positive instance more than with another.

Additionally, the model creates statistics for the merged positive windows in ground truth, TP, FP, and FN groups. By comparing these results to the merged ground truth information, it becomes evident how many actual spike events the model successfully detected.

4.2.4 Visualization of model outputs

The output visualization of the spike detection model provides a detailed view of the model's performance in identifying spike events over time. It combines the model's predicted probabilities with the actual ground truth spike annotations (when available), allowing for an in-depth evaluation of the model's behavior.

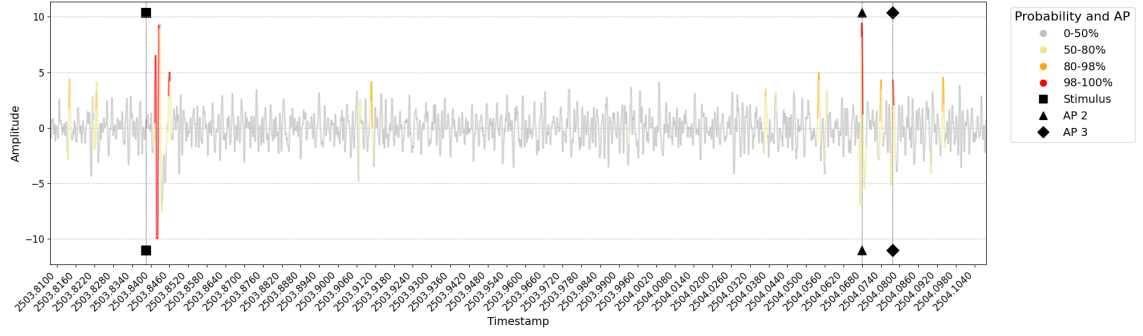
The data is initially divided into segments using a sliding window approach, with each window covering a fixed number of samples. Due to the overlapping nature of the win-

dows, each data point may appear in multiple windows, each with a different probability outcome. Therefore, all data point is assigned with the average probability it received from its assigned windows. These probabilities are then categorized into different levels of certainty by grouping them into predefined bins. Each bin is assigned a unique color to visually differentiate various confidence levels in the model's predictions.

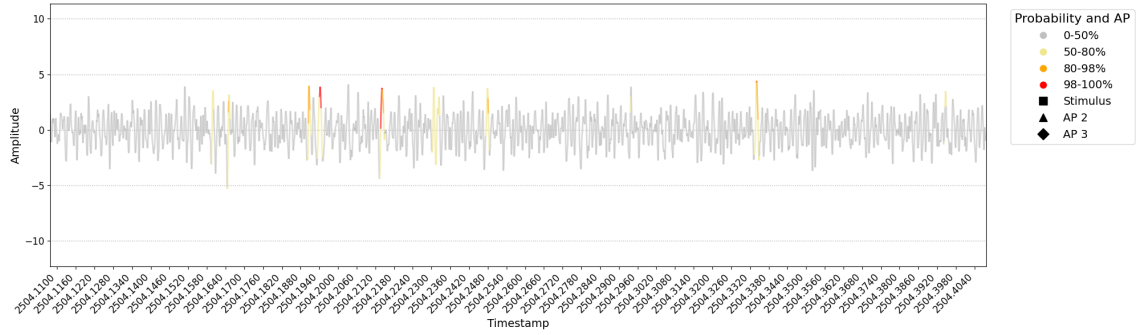
The primary visualization is a line plot, where the X-axis represents the timestamps and the Y-axis the recorded amplitudes. One line segment's color indicates the spike probability, based on the distinct colors detailed on the legend. Vertical lines mark the stimulus and actual action potential events, based on the ground truth data. Since these events have a specific annotated timestamp, these exact timestamps are represented on the plot instead of a whole spike area. By displaying the predicted spike probabilities alongside the true spike annotations, the visualization offers immediate insights into the model's performance.

Furthermore, when evaluating in the real-world scenario where the ground truth data is not available, this output allows immediate interpretation of the model output, indicating around which timestamp it is advisable to examine the signal for possible action potentials. By displaying the predicted probabilities on the raw signal, the visualization ensures an intuitive understanding of the model's outputs.

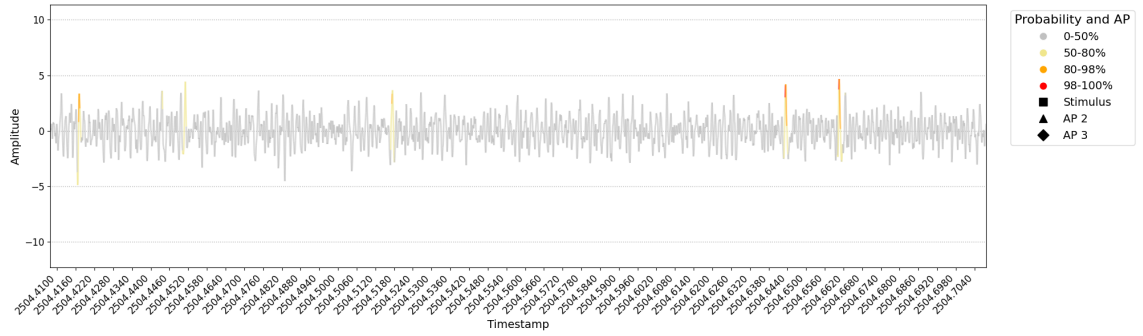
Figure 4.5 displays three consecutive segments of the output visualization on the test set of the best-performing model. The first plot illustrates the model's performance on actual spike events, accurately capturing the ground truth signals. In the second segment, only background noise is present, with some false positive instances. In the third plot, constant background noise is present again, but there are no strong predictions of spike events suggesting a spike check.



(a) Stimulus, and the two types of action potentials ground truths present on one plot.



(b) Some false positive instances on background noise.



(c) Clear, well predicted background noise.

Figure 4.5: Output visualization of three consecutive prediction segments.

4.3 Explainability

4.3.1 Post-hoc interpretability

Post-hoc interpretability has gained popularity among other XAI approaches due to its human-like method of interpretation. It is a model-agnostic approach, meaning its effectiveness is not limited to a specific type of model. It has proven to be effective across a variety of models with different types of inputs and offers a broad set of techniques, such as visual explanations, textual justifications, and case-based reasoning, as detailed in Section 1.2.

XAI algorithms, when applied to pre-trained models, aim to decode the inner workings of these models, primarily using gradient-based or backpropagation methods, depending on the algorithm. A wide range of algorithms is available, but evaluating all of them for a specific problem can be time-consuming. To address this, the iNNvestigate library [27] was developed to facilitate research and development by providing pre-implemented algorithms for analyzing neural network decisions. The goal of iNNvestigate is to simplify the use of these methods without necessarily explaining their underlying concepts and assumptions.

These algorithms take the model and its input as parameters. Running an XAI algorithm produces an output object with the same shape as the input, revealing how each component of the input contributed to the final prediction. These relevance scores act as contribution maps on the original input. Negative relevance scores indicate that a specific component discouraged the model from classifying the input as belonging to a certain class, while positive scores indicate components that contributed to the classification. Relevance scores close to zero suggest that the model largely ignored those components in its decision-making process.

These relevance maps can provide meaningful insights for various input types. For example:

- Images: Relevance maps highlight pixel-wise importance, showing which regions of the image were most influential.
- Text: Individual words or phrases can be emphasized, offering insights into how the model processes language.
- Signals: Every amplitude point in the signal can be assigned a relevance score, enabling analysis of time-series data.

By applying visualization techniques to these maps, either in isolation or alongside the input data, key patterns become evident. For example, sequences of important data points over time can be immediately identified, allowing to draw conclusions. These conclusions can be further supported by additional plots that illustrate the connection between the relevance maps and the input data from different perspectives. When hypotheses are supported from multiple angles, clear textual explanations can be created.

This three-component pipeline — relevance mapping, visualization, and explanatory synthesis — enables valuable discoveries. Areas can be identified where the model needs improvement or uncover patterns that the model detects, which may not have been forecasted by humans.

4.3.2 Algorithms for explainable artificial intelligence

The algorithms can be grouped based on the principles of sensitivity and implementation invariance [31]. Sensitivity ensures that if two inputs differ by a single feature and result in different predictions, the differing feature should receive a non-zero relevance score. Implementation invariance, on the other hand, states that if two models have identical output functions for the same input, their relevance scores for the respective inputs should also be identical.

In a referenced study [28] that analyzed a model for classifying ECG signals, the Integrated Gradients (IG) and Layer-wise Relevance Propagation (LRP) algorithms were found to be particularly effective for signal processing tasks.

Integrated Gradients (IG) [31] calculates the importance of each input feature by approximating the integral of gradients of the model’s output with respect to its inputs. IG evaluates the entire input space, capturing how variations in individual features influence the model’s output. It adheres to both the sensitivity and implementation invariance axioms, making it a robust method for interpreting model predictions.

The iNNvestigate library introduces the concept of neuron indexing, a parameter offering three weighting options for network analysis: no special weighting (all), prioritizing neurons with the highest activation during training (max activation), and selecting a neuron corresponding to the final class index and tracing its path (index).

Across analyses of VPNet and WHVPNet for ECG, EEG, and microneurography data, Integrated Gradients with the max activation consistently produced meaningful and interpretable results in all cases, demonstrating its efficiency for these applications and

possibly for the VPNet model and its variations. Therefore, in this project, only the IG algorithm was utilized to provide relevance score values for further analysis.

Integrated gradients algorithm

For the Integrated Gradients algorithm, the authors [31] define the need for a baseline (or reference input) as the possibility of represent the absence of a cause, thus being able to compare it with values where the cause is present. For example, in an image recognition problem, the baseline is a black image; and in signal processing, the baseline is zero.

IG is a combination of the implementation invariance gradients and the sensitive techniques and computed as follows.

For a function F representing a deep network, let x be the input and x' the baseline. IG considers a straight-line path p from the baseline x' to the input x , computes the gradients at all points along p , and cumulates them. It behaves like the line integral of the gradients along p , which is defined as $x' + \alpha(x - x')$ for $\alpha \in [0, 1]$.

The integrated gradient along the i^{th} dimension for an input x and baseline x' is :

$$\text{IG}_i(x) := (x_i - x'_i) \cdot \int_0^1 \frac{\partial F(x' + \alpha \cdot (x - x'))}{\partial x_i} d\alpha$$

where $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i^{th} dimension.

IG satisfies completeness. This means that the attributions sum up to the difference between the output and the baseline, thereby approximating the prediction score. This can be proved by the fundamental theorem of calculus for line integrals. If F is differentiable everywhere, then

$$\sum_{i=1}^n \text{IG}_i(x) = F(x) - F(x')$$

For computing IG, the integral is approximated with the Riemann sum, where the points occurring at sufficiently small intervals along the p are summed.

$$\text{IG}_i(x)^{\text{approx}} := (x_i - x'_i) \cdot \sum_{k=1}^m \frac{\partial F(x' + \frac{k}{m} \cdot (x - x'))}{\partial x_i} \cdot \frac{1}{m}$$

where m is the number of steps in the Riemann approximation.

Based on the calculations, the algorithm needs two parameters. The baseline or reference input, and the number of steps.

Chapter 5

Results

5.1 Outcomes of the classification model

Prior to achieving the best outcome, parameter fine-tuning, validation, and testing strategies were applied.

The preprocessing pipeline has two tunable parameters: window size and overlapping size; the VPLayer has two: the number of features to extract and the number of Hermite system weights; the fully connected layer has one: the number of neurons; and the model as a whole has three: learning rate, epochs, and batch size. Grid searches were conducted for each group: 98 runs for the preprocessing pipeline, 50 runs for the VPLayer, 8 runs for the fully connected layer, and 60 runs for the learning rate, epochs, and batch size.

Sliding window parameters

The first tested parameter was the sliding window configuration for input preparation. Window sizes were tested in a range starting from 15 up to 45, with a step size of 5. For all window sizes, overlapping ratios of 50% and 75% were tested, rounded to the nearest integer for overlap size.

The results revealed a clear trend: smaller window sizes consistently resulted in better model performance. Furthermore, a 75% overlapping ratio outperformed the 50% ratio across all window sizes. Therefore, the 15-sampled window with 11-sample oversampling was the final decision, as it constantly proved to have the best outcomes.

Resampling

The next phase of testing involved exploring various resampling methods to address the data imbalance using the techniques described in Section 4.1.1. Class 0 instances correspond to raw data or background noise, while Class 1 represents stimulus and action potential instances. The following test cases are sorted, with the next test case always having the same or better results than the previous one:

1. Balanced Dataset: Random undersampling of Class 0 to match the size of Class 1. Results showed improvement compared to the original dataset.
2. Balanced Dataset: Oversampling Class 1 to M , and undersampling Class 0 to M .
3. Balanced Dataset: Clustered undersampling of Class 0 to match the size of Class 1. The results were the same as in previous cases.
4. Balanced Dataset: Oversampling Class 1 to M , and undersampling Class 0 to M , but repetitive samples for Class 1 were only taken from action potentials and not from stimulus labels.
5. Mildly Balanced Resampling: Class 1 constitutes 20%-40% of the dataset, and repetitive samples are only from action potentials. Cross-entropy loss weighting is necessary.
6. Mildly Balanced Resampling: Class 1 constitutes 20%-40% of the dataset, with all Class 1 instances sourced from action potentials, excluding stimulus-labeled samples. Cross-entropy loss weighting is applied. The results are the same whether stimulus-labeled samples are excluded or not, indicating these windows do not contribute to learning.
7. Mildly Balanced Resampling: Class 1 and Class 0 are in a 25%-75% ratio. Cross-entropy loss is weighted, and stimulus windows are relabeled. This approach had the best overall results for data modifications.

In the over- and undersampling method, the amount of class instances can be selected, therefore more ranges were tested. In the training dataset, there were ~ 2250 windows labeled as the positive class. Oversampling it to 10 times, the model performed significantly better than in any previous scenarios. To maintain exactly the 25:75 ratio, the sample size of the minority class was set to 25000, and for the majority class, to 75000, reducing it to $\sim 6\%$ of the original size. Increasing these numbers did not lead to further improvement.

Stimulation relabel

The stimulation relabeling process was tested alongside the sliding window and re-sampling phases. Table 5.1 presents the validation results for models with window sizes between 15 and 30, with 75% oversampling ratio, comparing scenarios with and without stimulus relabeling. The models were always re-trained with the respective configuration, and their validation was performed using the same window sizes on which they were trained. Larger window sizes were excluded due to their poor performance in prior tests.

Window Size	Overlapping Size	Stimuli Corrected?	AUC Value
15	11	FALSE	0.79
15	11	TRUE	0.94
20	15	FALSE	0.75
20	15	TRUE	0.90
25	17	FALSE	0.75
25	17	TRUE	0.87
30	22	FALSE	0.72
30	22	TRUE	0.88

Table 5.1: Results for different window sizes and stimuli correction.

The table shows that smaller window sizes consistently performed better. Additionally, when comparing the same window sizes with or without relabeling, the relabeled scenario improved model performance in all cases.

Secondly, to support the statement that the original stimulus labels did not contribute to the learning process, a comparison of the predictions for class 1 with respect to the multi-class labels before and after stimulus relabeling is presented in table 5.2.

Multi-class label	Before Relabel	After Relabel
Label 1	TP: 26.0 FN: 208.0	TP: 254.0 FN: 5.0
Label 2	TP: 248.0 FN: 7.0	TP: 250.0 FN: 5.0
Label 3	TP: 234.0 FN: 19.0	TP: 235.0 FN: 18.0

Table 5.2: Comparison of TP and FN values before and after relabeling for each label.

The results clearly demonstrate that while action potential detections (Label 2 and Label 3) already showed acceptable performance, the stimulus-labeled windows (Label 1) were consistently misclassified before relabeling. From the model’s perspective, these labels all correspond to Class 1, but the evaluation metrics were significantly impacted due to the misclassification of stimulus labels. After relabeling, a substantial improvement was observed in the classification of stimulus labels (Label 1).

Cross-Entropy Loss Weighting

The purpose of weighting the cross-entropy loss function is to emphasize the minority class. The weights should ideally be inversely proportional to the ratio of the imbalanced classes. For instance, if the ratio of Class 0 to Class 1 is 75:25, the weights should be approximately 0.25 for Class 0 and 0.75 for Class 1. However, small adjustments, such as using weights of 0.2 and 0.8 or 0.3 and 0.7, can result in variations depending on how FPR and recall metrics are prioritized.

Since Class 1 (positive events) is more important in this context, initial weights of [0.2, 0.8] and [0.25, 0.75] were used, producing good results.

Upon analyzing the decision certainty for predictions (detailed in Section 4.2.2), it was evident that when the model correctly predicted the positive class, it exhibited high confidence. To take advantage of this insight, the decision boundary was adjusted to 0.8, allowing for greater uncertainty freedom for Class 0. Following this adjustment, and the realization that Class 1 predictions are confident enough, using weights of [0.3, 0.7] slightly improved the outcomes.

Fine-Tuning WHVPNet Parameters

The WHVPNet introduces two critical parameters for fine-tuning:

- Number of coefficients: Determines the number of features the VPLayer [10] extracts for signal decomposition.
- Number of weights in the Hermite system: A new parameter unique to WHVPNet [12], that corresponds to the number of trainable parameters for weighting the Hermite system.

In all prior test cases, the model was configured with 3 coefficients and 2 weights. Under these settings, the best performance achieved was a 0.94 ROC-AUC, 0.91 recall, and 0.04 FPR.

To optimize these parameters, a grid search was conducted:

- Coefficients: Tested values ranged from 3 to 12. Results showed that coefficients between 6 and 11 provided comparable performance.
- Weights: Tested values ranged from 2 to 6. Among these, 2, 3, and 4 had the best results.

Fine-tuning these parameters led to a 6% improvement in recall, achieving 0.97 in most cases. Models configured with coefficients in the range of 6–11 and weights in the range of 2–4 consistently produced recall values between 0.96 and 0.97. The slightly best-performing model is constructed with 6 coefficients and 4 weights for the Hermite system. Its output analysis is seen in Table 5.3.

Metric	Validation Set	Test Set
Label 0 (TN)	387,548	392,453
Label 0 (FP)	15,258	10,655
Label 1 (TP)	259	158
Label 1 (FN)	0	0
Label 2 (TP)	255	153
Label 2 (FN)	0	0
Label 3 (TP)	236	135
Label 3 (FN)	17	20
Recall	97.78%	95.71%
FPR	3.7%	2.6%

Table 5.3: Comparison of validation and test set results for the best performing model.

This model also demonstrates why a decision boundary of 0.8 is better. On the validation set, the model using a 0.5 decision boundary results in 15 false negatives but significantly more false positives — 22,848 compared to 15,258 with the 0.8 decision boundary.

When applying the merging windows evaluation metric, as shown in Table 5.4, the results indicate that the model successfully captured almost all action potential instances. The number of false positive occurrences ranges from 6,227 to 8,455. However, comparing these values directly to the negative windows is challenging, as negative windows cannot be merged in the same way.

The original validation and test sets, without overlapping windows, contain approximately 1.6 million data points. Dividing this by the average number of windows per spike (4 in the ground truth) results in approximately 400,000 sections. Using this value, 8,455 false positives correspond to maximum 2% of all sections. Moreover, a significant proportion of these merged false positive instances are only one window long, and not continuous misclassification windows.

	Validation	Test
Every Window Counted as One Decision		
Recall	0.97	0.95
FPR	0.037	0.024
Neighbouring Positive Windows Merged into One Prediction		
Original Number of Action Potentials	201	123
Predicted Number of Action Potentials	200	121
Number of False Positives (Out of 400,000)	8,455	6,227

Table 5.4: Metrics for validation and test sets using merging window evaluation.

Overall, the best model achieved a 97.78% recall and 3.7% FPR on the validation set. On the test set, it reached a 95.71% recall and 2.6% FPR, and in both test sets, almost every action potentials were found. Example output visualizations of this model are shown in Figure 4.5. In the following, this model has been selected for further evaluation in the explainability analysis.

5.2 Outcomes of the explainability analysis

First, indices are generated based on prediction outcomes, categorized as true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). This categorization allows for the creation of distinct plots for each prediction quality, enabling more detailed analyses.

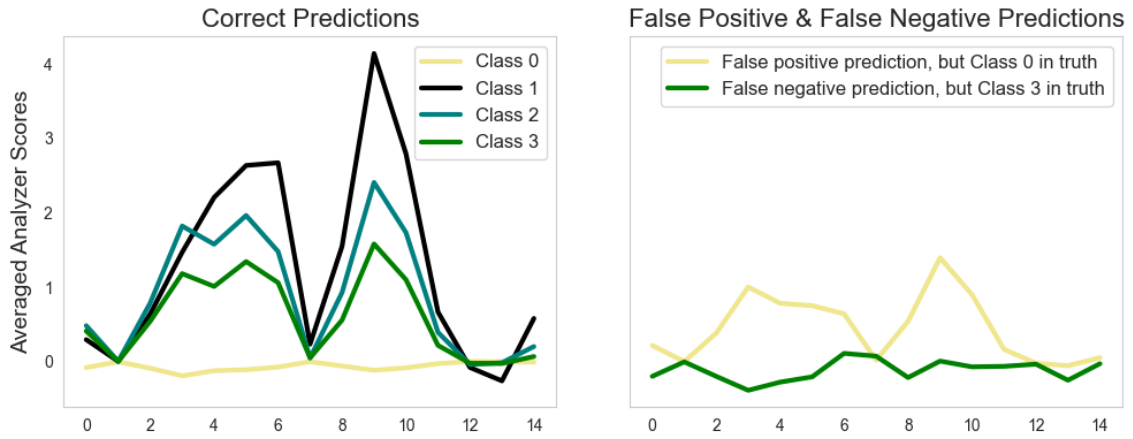


Figure 5.1: Averaged relevance scores grouped by the prediction outcomes.

With the multi-class labeling, for the false negative and true positive categories, where the original spike type is available, the prediction outcomes can be further subdivided. This allows for an analysis of whether the model's behavior varies for different types of spikes, even though they are grouped into a single class for binary classification. The post-hoc visualizations were performed in three types.

Averaged relevance scores plot for comparing different prediction qualities

The plot (Figure 5.1) visualizes the average relevance score signals across different prediction qualities. By displaying these scores together, it becomes easier to compare how the model's interpretation varies between different outcomes. This comparison provides insight into the factors that may have led to differing prediction qualities.

The relevance scores for the true negative predictions remain near zero, indicating that no part of the signal strongly contributes to these outcomes. This seemingly paradoxical observation gets its basis by comparing the relevance score averages for the positive class, where two distinct and highly significant areas in the signals consistently stand out.

The detailed multi-labels for the positive class reveal a consistent trend in the relevance scores' shape, with variations only in range. These ranges closely correlate with the actual spikes' ratios, suggesting that higher ranges in the raw signal help the model to make a difference in its decision. The class 0 relevance scores show a tendency for the signal to shift slightly negatively precisely at points where positive signals begin to emerge, like a sudden response or a mirror for the positive class relevances.

This mirroring effect is also evident in false positives versus false negatives. When the model identifies positive-worthy values in the two significant areas, even if these values

are weaker than those in the correct predictions, it still predicts them as positive, falsely. These weaker contributions in false positive predictions align with the prediction outcome statistics, where it became evident that the model exhibits very high confidence when making correct positive predictions, and far less confidence when making a false one.

The explanation for no values strongly contributing to the classification of the background noise is that the model seems to treat the background as the default decision. It does not extract particular observable common patterns or shapes from these signals. Instead, it only "changes its mind" when strong fluctuations or distinct signal shapes emerge, indicating a potential positive class instance.

Single sample plot colored by its relevance score

For a more detailed view, a single randomly selected recording can be plotted (Figure 5.2). For a unified colorization, the relevance scores are normalized between -1 and 1 for this type of plot, and then used to color the segments of the original signal, with a color bar indicating the correspondence between score values and colors. This visualization highlights specific parts of the input that influenced the prediction, offering a local interpretation and increasing trust in the model by providing insights into detailed outcomes.

In Figure 5.2, it is visible that the model takes signal shapes into account. For the true action potential instances and the false positive instance, it is evident that both the first and second arches of the spikes contribute significantly to the final decision, with the second arch having slightly more influence. If the model only relied on extreme positive and negative values, the stimulus signal's values below -5 and above 4 would have already been highlighted in red, but that is not the case. This observation confirms that the entire arch is considered, rather than just isolated absolute highest values.

For the true negative prediction, the same wide arch is strongly colored in blue. The negative relevance scores in this case indicate that these regions suggested the model the opposite class. Additionally, for the false negative prediction — which happened only for the second type of action potential — a similar arch is highlighted in blue, showing a negative contribution. Although the model mostly found negative contribution for the negative class, these samples were still classified as negative ones, which may suggest that the relevance scores lacked sufficient magnitude - as seen in Figure 5.1 - to tip the classification toward the positive class, and stayed at the default negative cases.

As for the false negative signal, even from visual examination, the shape suggests that this indeed does not look like an action potential, but was labeled so because of the

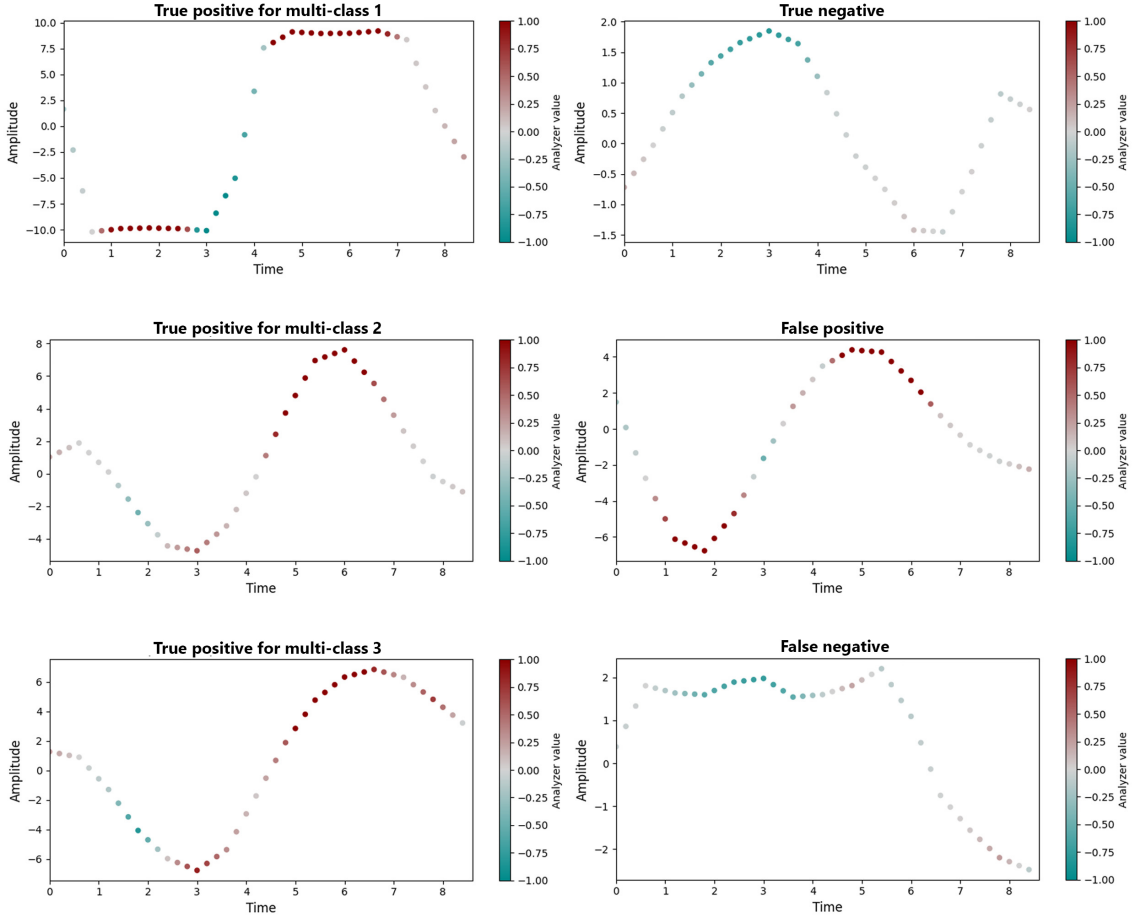


Figure 5.2: Randomly selected simple input signals with varying prediction qualities. The coloring reflects their relevance scores, as indicated by the colorbar. The relevance scores and colorbar values are normalized to facilitate easier comparison.

waterfall technique.

Further evidence of the model’s ability to detect arches is shown in how it handles the stimulus sample, so the multi-class 1 instance. While high or low amplitude values strongly correlate with the positive class, a sharp directional change in the lower part of the signal breaks the arch pattern and marked immediately as opposite signs of the positive class. These values are still very low, but this shape change was enough for the model to detect strong negative contribution to the positive class. However, since the model found high correlation to the positive class based on the extremely high and low values, these areas were strong enough to a positive predictions.

Feature-specific analysis in VPNet

The VPNet architecture enables separate analysis of its feature extraction component. The first unique VPLayer handles feature extraction, and its outputs are forwarded to a

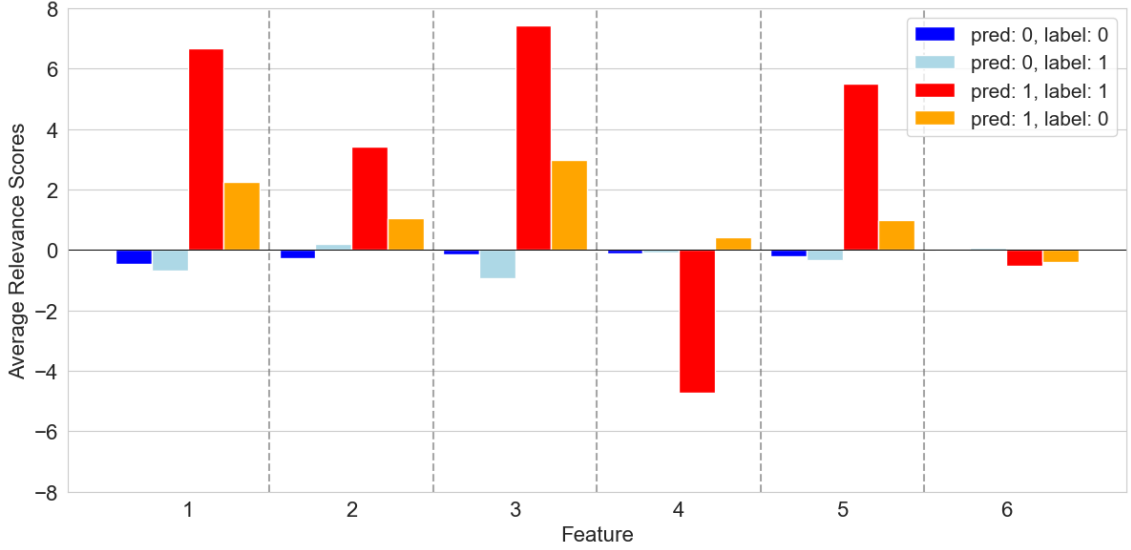


Figure 5.3: Averaged relevance scores of the six features of the model after the VPLayer, grouped by the prediction qualities. Dark blue for TN, light blue for FN, red for TP, and orange for FP predictions.

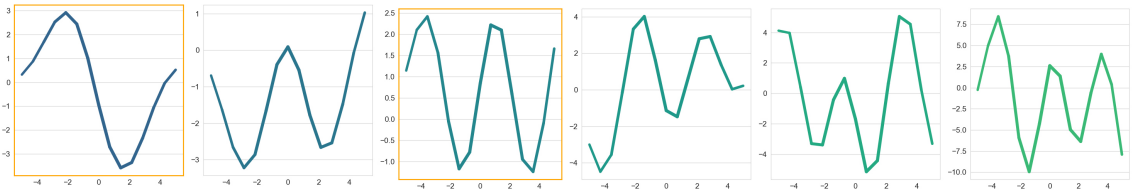


Figure 5.4: The six weighted Hermite basis functions that the model uses in the VPLayer for input approximation.

fully connected neural network. By applying the IG algorithm to the VPLayer's outputs, a diagnostic histogram can be generated (Figure 5.3). This histogram visualizes how individual features contributed to the final decisions.

The feature contributions reveal that for negative predictions, the model rarely depends on any features, as their relevance scores consistently remain near zero. In contrast, true positive predictions demonstrate highly positive absolute values for the first five features, indicating that the model becomes significantly "activated" in the presence of positive predictions.

For false positive predictions, the relevance scores are higher than those of the negative class but not as high as in the true positive cases. This aligns with the confidence statistics, which revealed that the model is highly confident in its true positive predictions compared to its false predictions.

The modified Hermite functions generated in the VPLayer are visible in Fig. 5.4. The highest relevance values are for the first bimodal, and third W-shaped spikes, with those

being mostly responsible for the false positive predictions as well.

Summary on interpretation

The XAI analysis from three main perspectives collectively suggests that the model does not attempt to extract specific features or patterns from the background noise, treating it instead as a default classification. However, when a potential spike instance appears, the model identifies it as distinct from the background noise. Firstly, if it exhibits extremely high or low positive values for the stimulus, or secondly, in the absence of such extreme absolute values, the model recognizes well-defined, regular spike peak arches — closer to the noise level than the stimulus — as positive instances, resulting in correct classifications.

Across most of the trained, validated, and tested models, the stimulus and first type of action potentials are classified with high accuracy, mostly perfectly, with maximum a few false negative instances. Almost all false negatives are attributed to the second type of action potentials. The model evaluated for explanation has no false negative predictions for that stimulus or first type of action potential neither in the validation nor in the test set. This second type of action potential often exhibits characteristics similar to background responses and is labeled because of a time-sequenced proof of manner. Due to these occurrences, the false negative predictions are likely to occur for spikes that lack a clear spike-like arch shape and remain below the background level — as verified through single sample interpretation.

On the other hand, false positive predictions occur in background regions where responses resemble action potential-like shapes, often with distinct arches.

These visualizations also hint at the model’s behavior: in true positive classifications, the model demonstrates significantly higher confidence in its predictions compared to false predictions. The decision-boundary shift already contributes to the handling of this phenomenon. Additionally, on the model’s output visualization, most of the time, only the highest percentage predictions correspond to actual spike instances.

Chapter 6

Conclusion and improvement potentials

During this project, a unique, interpretable deep neural network requiring fewer parameters and less data was successfully trained for microneurography spike detection. Previously, deep neural networks had not been effectively applied to this field due to the low signal-to-noise ratio, the highly imbalanced nature of the dataset, and the high labeled data demand of deep neural networks. Achieving a recall of 0.97 with FPR of 0.037 on the validation set, and a recall of 0.95 with FPR of 0.027 on the test set, the model demonstrated its first practical applicability and has been accepted by the Digital C-Fiber group in Germany, which provided the initial dataset. Building on this success, the Digital C-Fiber group plans to apply the model to other microneurography datasets recorded in different clinical environments at the Uniklinik RWTH Aachen University, offering more detailed feedback on its performance.

The interpretability analysis revealed that false negative and false positive predictions primarily occurred when the shape of the waveforms did not contribute meaningfully to their respective classes, often due to the consequences of high noise responses. While the current approach of performing classification independently for each window yields strong results, one important piece of information in the microneurography data is not utilized in this implementation. Specifically, the latency times, which provide high-level information about spike occurrences — similar to how manual labeling is performed — are excluded. Despite this, the model still achieves competitive results due to its ability to effectively capture spike-like waveforms. However, the remaining misclassifications could potentially be addressed by implementing this time-based information into the model architecture. This could be achieved by replacing the fully connected layers in the WHVPNet with recurrent neural networks, such as LSTM, which can handle time

dependencies in sequential input samples. In cases where the spike shapes are indistinguishable from the noise responses, the latency times could still provide valuable insight, indicating possible spike occurrences. By combining the current approach of detecting specific magnitude and arched spike shapes with latency information, the already good results could be even further improved.

Before the publication of the WHVPNet article [12], the manuscript's reviewer requested additional details about the model's decision-making process. Consequently, the interpretability technique detailed in this TDK thesis was also utilized in the article as a final addition, and this contribution is mentioned in the article's acknowledgments section. Furthermore, the results of this thesis are summarized in a conference paper, which has been submitted to the IEEE EMBC for the 2025 conference under the title [35].

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Dr. Péter Kovács from Eötvös Loránd University, for his expertise, reliable guidance and support, as well as for the unique classification model that enabled the results of this TDK paper. I also extend my gratitude to Barbara Namer from the University Hospital of Aachen/Würzburg for providing the real-life microneurography recordings and ground truth labeling, as well as to Ekaterina Kutafina and Alina Troglia from the University Hospital of Cologne for their foundational work in spike sorting. Additionally, I am grateful to Bence Barabás, PhD candidate in neurobiology at Semmelweis University, for his precise guidance in the writing of the neurological sections of this paper.

Appendix

Evaluation Metrics

This section provides the definitions and formulas for common model evaluation metrics.

TP— True Positive

TN— True Negative

FP— False Positive

FN— False Negative

Accuracy: The proportion of all correct classifications:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Recall or Sensitivity: The proportion of all actual positives that were correctly classified as positives:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

Precision: The proportion of true positives among all positive predictions:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

False Positive Rate (FPR): Measures the proportion of false positives out of all actual negative samples:

$$\text{FPR} = \frac{FP}{TN + FP} \quad (4)$$

Specificity: The proportion of true negatives correctly classified:

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (5)$$

Balanced Accuracy: The average of sensitivity and specificity:

$$\text{Balanced Accuracy} = 0.5 \cdot (\text{Sensitivity} + \text{Specificity}) \quad (6)$$

Confusion Matrix: A table summarizing classification outcomes:

		Predicted Values	
		Negatives	Positives
Actual values	Negatives	TN	FP
	Positives	FN	TP

Figure 1: Confusion matrix

F1-score: The harmonic mean of precision and recall:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

F2-score: Modified F-score, where the recall is prioritized over precision:

$$\text{F2-score} = \frac{(1 + 2^2) \cdot \text{Precision} \cdot \text{Recall}}{2^2 \cdot \text{Precision} + \text{Recall}} \quad (8)$$

ROC Curve and PR-AUC

The ROC (Receiver Operating Characteristic) curve plots the true positive rate (recall) against the false positive rate at various thresholds. The Area Under the Curve (AUC) represents the model's ability to distinguish between classes; a higher AUC indicates better performance.

PR-AUC (Precision-Recall Area Under the Curve) evaluates the relationship between precision and recall across thresholds. It is particularly useful for imbalanced datasets, where the positive class is rare or false negatives are more critical than false positives.

Bibliography

- [1] Sanjit K. Mitra. *Digital Signal Processing : A Computer-Based Approach*. 2nd Edition. McGraw-Hill Higher Education, 2001. ISBN: 0072321059.
- [2] Zachary C. Lipton. “The mythos of model interpretability”. en. In: *Communications of the ACM* 61.10 (Sept. 2018), pp. 36–43. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3233231. URL: <https://dl.acm.org/doi/10.1145/3233231> (visited on 03/21/2024).
- [3] Amina Adadi and Mohammed Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.
- [4] Tim Miller. *Explanation in artificial intelligence: Insights from the social sciences*. [Online]. Available: <https://arxiv.org/abs/1706.07269>. 2017.
- [5] Zachary C. Lipton. “The mythos of model interpretability”. In: *Commun. ACM* 61.10 (Sept. 2018), 36–43. ISSN: 0001-0782. DOI: 10.1145/3233231. URL: <https://doi.org/10.1145/3233231>.
- [6] Z. Nenadic and J.W. Burdick. “Spike detection using the continuous wavelet transform”. In: *IEEE Transactions on Biomedical Engineering* 52.1 (2005), pp. 74–87. DOI: 10.1109/TBME.2004.839800.
- [7] Katherine J. Laboy-Juárez, Soyeon Ahn, and Daniel E. Feldman. “A normalized template matching method for improving spike detection in extracellular voltage recordings”. In: *Scientific Reports* 9 (2019), p. 12087. DOI: 10.1038/s41598-019-48456-y. URL: <https://doi.org/10.1038/s41598-019-48456-y>.
- [8] Ekaterina Kutafina et al. “Decoding Neuropathic Pain: Can We Predict Fluctuations of Propagation Speed in Stimulated Peripheral Nerve?” In: *Frontiers in Computational Neuroscience* 16 (2022). ISSN: 1662-5188. DOI: 10.3389/

- fncom.2022.899584. URL: <https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2022.899584>.
- [9] Alina Troglio et al. *Automated Spike Sorting in Microneurography: A Proof-of-Concept through Classification on Ground Truth Data*. Mar. 2024. URL: https://www.researchgate.net/publication/379120276_Automated_Spike_Sorting_in_Microneurography_A_Proof_of_Concept_through_Classification_on_Ground_Truth_Data.
- [10] Péter Kovács et al. “VPNET: Variable Projection Networks”. In: *International Journal of Neural Systems* 32.01 (2022). PMID: 34651549, p. 2150054. DOI: 10.1142/S0129065721500544. eprint: <https://doi.org/10.1142/S0129065721500544>. URL: <https://doi.org/10.1142/S0129065721500544>.
- [11] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. “Past, present and future of spike sorting techniques”. In: *Brain Research Bulletin* 119 (2015). Advances in electrophysiological data analysis, pp. 106–117. ISSN: 0361-9230. DOI: <https://doi.org/10.1016/j.brainresbull.2015.04.007>. URL: <https://www.sciencedirect.com/science/article/pii/S0361923015000684>.
- [12] T. Dozsa et al. “Weighted Hermite Variable Projection Networks for Classifying Visually Evoked Potentials”. In: *IEEE Transactions on Neural Networks and Learning Systems* PP (Oct. 2024). Epub ahead of print. DOI: 10.1109/TNNLS.2024.3475271.
- [13] C.K. Arcilla and P. Tadi. *Neuroanatomy, Unmyelinated Nerve Fibers*. 2024 Jan–. StatPearls [Internet]. PMID: 32119348. Treasure Island, FL: StatPearls Publishing, 2023. URL: <https://www.ncbi.nlm.nih.gov/books/NBK557700/>.
- [14] Å.B. Vallbo and K.-E. Hagbarth. “Activity from skin mechanoreceptors recorded percutaneously in awake human subjects”. In: *Experimental Neurology* 21.3 (1968), pp. 270–289. ISSN: 0014-4886. DOI: [https://doi.org/10.1016/0014-4886\(68\)90041-1](https://doi.org/10.1016/0014-4886(68)90041-1). URL: <https://www.sciencedirect.com/science/article/pii/0014488668900411>.
- [15] H.E. Torebjörk and R.G. Hallin. “Identification of afferent C units in intact human skin nerves”. In: *Brain Research* 67.3 (1974), pp. 387–403. ISSN: 0006-8993. DOI:

- [https://doi.org/10.1016/0006-8993\(74\)90489-2](https://doi.org/10.1016/0006-8993(74)90489-2). URL: <https://www.sciencedirect.com/science/article/pii/0006899374904892>.
- [16] Rochelle Ackerley and Roger Holmes Watkins. “Microneurography as a tool to study the function of individual C-fiber afferents in humans: responses from nociceptors, thermoreceptors, and mechanoreceptors”. In: *Journal of Neurophysiology* 120.6 (2018). PMID: 30256737, pp. 2834–2846. DOI: 10.1152/jn.00109.2018. eprint: <https://doi.org/10.1152/jn.00109.2018>. URL: <https://doi.org/10.1152/jn.00109.2018>.
 - [17] M. Schmelz et al. “Delayed responses to electrical stimuli reflect C-fiber responsiveness in human microneurography”. In: *Experimental Brain Research* 104.2 (1995), pp. 331–336. ISSN: 0014-4819. DOI: 10.1007/BF00242018.
 - [18] Spike2. *CED Spike2: System Introduction*. Available online at: <https://ced.co.uk/products/spkovin> (accessed March 2022). Cambridge Electronic Design. 2022.
 - [19] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. “Past, present and future of spike sorting techniques”. In: *Brain Research Bulletin* 119 (2015). Advances in electrophysiological data analysis, pp. 106–117. ISSN: 0361-9230. DOI: <https://doi.org/10.1016/j.brainresbull.2015.04.007>. URL: <https://www.sciencedirect.com/science/article/pii/S0361923015000684>.
 - [20] Arne Diedrich et al. “Analysis of raw microneurographic recordings based on wavelet de-noising technique and classification algorithm: wavelet analysis in microneurography”. In: *IEEE Transactions on Biomedical Engineering* 50 (Jan. 2003). DOI: 10.1109/TBME.2002.807323.
 - [21] A. Salmanpour, L. J. Brown, and J. K. Shoemaker. “Spike detection in human muscle sympathetic nerve activity using a matched wavelet approach”. In: *Journal of Neuroscience Methods* 193.2 (2010), pp. 343–355. DOI: 10.1016/j.jneumeth.2010.08.035. eprint: 2010Sep8. URL: <https://doi.org/10.1016/j.jneumeth.2010.08.035>.
 - [22] B. Turnquist, B. RichardWebster, and B. Namer. “Automated detection of latency tracks in microneurography recordings using track correlation”. In: *Journal of Neuroscience Methods* 262 (2016), pp. 133–141. DOI: 10.1016/j.jneumeth.

- 2016.01.004. eprint: 2016Jan8. URL: <https://doi.org/10.1016/j.jneumeth.2016.01.004>.
- [23] B. Hammarberg, C. Forster, and E. Torebjork. “Parameter estimation of human nerve C-fibers using matched filtering and multiple hypothesis tracking”. In: *IEEE Transactions on Biomedical Engineering* 49.4 (2002), pp. 329–336. DOI: 10.1109/10.991160.
- [24] F. Schlebusch et al. “openMNGlab: Data Analysis Framework for Microneurography - A Technical Report”. In: *Studies in Health Technology and Informatics* 283 (2021). DOI: 10.3233/SHTI210556.
- [25] A. Troglio et al. “Advanced Spike Sorting on Microneurography Data: Proof-of-Concept of VPNet as a Universal Approach”. In: Poster. GMDS Cooperations Conference, Dresden, Germany, Sept. 2024.
- [26] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* 32. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [27] Maximilian Alber et al. “iNNvestigate Neural Networks!” In: *Journal of Machine Learning Research* 20.93 (2019), pp. 1–8. URL: <http://jmlr.org/papers/v20/18-540.html>.
- [28] Theresa Bender et al. “Analysis of a Deep Learning Model for 12-Lead ECG Classification Reveals Learned Features Similar to Diagnostic Criteria”. In: *IEEE Journal of Biomedical and Health Informatics* 28.4 (2024), pp. 1848–1859. DOI: 10.1109/JBHI.2023.3271858.
- [29] Francois Chollet et al. *Keras*. 2015. URL: <https://github.com/fchollet/keras>.
- [30] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [31] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, Aug. 2017, pp. 3319–3328.

- [32] Yisheng Song et al. “A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and Opportunities”. In: *ACM Comput. Surv.* 55.13s (July 2023). ISSN: 0360-0300. DOI: 10.1145/3582688. URL: <https://doi.org/10.1145/3582688>.
- [33] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.
- [34] Zijun Cui et al. “Knowledge-Augmented Deep Learning and Its Applications: A Survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–21. DOI: 10.1109/TNNLS.2023.3338619.
- [35] D. Darabos et al. “Explainable spike detection algorithm on microneurography data using weighted Hermite VPNet”. In: Submitted. 47th IEEE EMBC, 2025.