UNIVERSITY
OF OSLO

Master's thesis

Using Machine Learning to Predict the Effects of Deep Brain Stimulation on Axonal Pathways

Karianne Strand

Physics
60 ECTS study points

Department of Physics
Faculty of Mathematics and Natural Sciences

Spring 2023

Karianne Strand

Using Machine Learning to Predict the Effects of Deep Brain Stimulation on Axonal Pathways

Supervisors:
Gaute T. Einevoll
Torbjørn V. Ness

Abstract

Deep brain stimulation (DBS) is a technique used in treatment of various brain disorders, where electrodes releasing constant stimuli are implanted within the brain. As a way to study DBS, we explored the potential of machine learning in predicting action potentials within axons in response to stimuli. Using the NEURON simulator and the LFPy 2.0 Python module, we modeled straight and curved axon structures to simulate their response to different stimuli with varied axon and stimulus parameters. Through these simulations, we generated extensive datasets where the occurrence and absence of an action potential defined a binary outcome. Subsequently, the datasets were used to train a feedforward neural network (FNN). Based on parameters, such as minimum distance between the axon and electrode, stimulus amplitude, stimulus duration, and spatial coordinates of axon endpoints, our machine learning model was able to predict the occurrence of action potentials in response to stimuli with great accuracy. However, the predictive accuracy of our FNN decreased when including additional spatial coordinates along the axon as features.

Contents

List of Figures

List of Tables

Abbreviations

DBS: Deep brain stimulation
PD: Parkinson's disease
AI: Artificial intelligence
ML: Machine learning
OCD: Obsessive-compulsive disorder
STN: Subthalamic nucleus
GPi: Globus pallidus, internus
HH: Hodgkin-Huxley
FNN: Feedforward neural network
ANN: Artificial neural network
ReLU: Rectified linear activation function
ADAM: Adaptive moment estimation
BCELoss: Binary cross-entropy loss
TP: True positives
TN: True negatives
FP: False positives
FN: False negatives

Acknowledgements

I would like to thank my supervisors, Torbjørn V. Ness and Gaute T. Einevoll, for their inspiration and guidance. A special thanks to Torbjørn for motivating this project and for your invaluable help throughout the course of this thesis.

Lastly, I would like to thank my family for their encouragement and support, and my fellow physics students and friends at the University of Oslo for the shared fun, frustrations and accomplishments.

Chapter 1

Introduction

In 1987, Alim Louis Benabid discovered that continuous electrical stimulation via implanted electrodes in specific areas of the brain could alleviate symptoms of movement disorders, such as Parkinson's disease (PD) [4]. This revolutionary approach paved the way for the development of deep brain stimulation (DBS), which has since become a highly effective treatment for multiple neurological disorders [27, 29]. Moreover, DBS continues to be explored as a promising treatment option for nearly any brain disorder [5].

Despite extensive research, the therapeutic mechanisms of deep brain stimulation are yet to be understood [2]. At its core, DBS is thought to operate by evoking action potentials in axons, which in turn affects the flow of neuronal information. However, the precise timing and location of these action potentials depend on the characteristics of both the axon and the stimulation parameters. Typically, any change in the DBS parameters, or the location and orientation of the DBS electrode, can alter the therapeutic outcomes [14].

To further improve DBS as a treatment, it is essential to better our understanding of these therapeutic mechanisms. Unfortunately, research progress can be slow due to ethical reasons that limit access to necessary data [2, 13, 31]. However, advances in technology are opening up new possibilities for computational research on DBS, which can help overcome some of these challenges.

On a microscopic scale, the biophysics involved in DBS are fundamentally simple and can be studied through simulations [28]. To perform such simulations, neurons must be modeled with detailed biophysical properties that capture both their morphological and electrophysiological features, as both factors are critical for determining the neuronal response to DBS. This can be done with simulation environments such as the NEURON simulator and the LFPy 2.0 Python module [15].

1.1 Research Objectives

Within the field of artificial intelligence (AI), machine learning (ML) has become a powerful tool in identifying patterns and classifying information beyond human capabilities. ML allows computers to learn from extensive datasets containing various parameters, also known as features. Through training, computers use this data to create, enhance and update models with the purpose of predicting outcomes based on the relationship

between these features.

In this thesis, we aim to simulate deep brain stimulation (DBS) using the NEURON simulator and the LFPy 2.0 Python module. Our goal is then to generate extensive datasets by varying stimulation parameters, and record the occurrence of action potentials in axons. Utilizing these datasets, our primary objective is to explore the potential of artificial neural networks[1] in predicting these action potentials when using arbitrary axon and stimulation parameters.

Ultimately, we seek to demonstrate the applicability of machine learning to DBS research. Successfully training a network to accurately predict the location and timing of axonal firing based on DBS parameters, becomes a valuable tool when combined with point neuron network models of neural activity in the presence of DBS. This combination holds great potential for studying and understanding how DBS is affecting neural network activity.

1.2 Thesis Framework

The thesis follows a systematic structure containing four subsequent chapters. In chapter 2, we elaborate on the theory and background needed to understand the underlying approach of our research, further exploring topics such as deep brain stimulation, axonal function, neural modeling and machine learning. Following that, in chapter 3, we offer a detailed description of the methods employed in conducting our study, covering the entire process from neural modeling and data collection to the development of our machine learning algorithm. The results derived from these applied methods are then presented in chapter 4. Finally, in chapter 5, we provide a detailed discussion of these results and conclude our research.

[1] In this thesis, it is important to note that when we mention "artifcial neural networks" we are specifically referring to the neural networks used in machine learning, not neural networks designed to simulate biological neural activity. Similarly, when we mention an "artificial neuron," we are referring to a node within an artificial neural network, not a computational model of a biological neuron.

Chapter 2

Background & Theory

In this chapter, we aim to provide the necessary background information and theory required to understand the motivation and methods of our research. To begin, we give an introduction to deep brain stimulation (DBS) and its applications in neuroscience. Subsequently, we will discuss neural modeling and simulation, particularly in utilizing simulation environments such as the NEURON simulator and the LFPy 2.0 Python module. Finally, we will outline the key concepts of machine learning within the context of binary classification and neuroscience.

2.1 Introduction to Deep Brain Stimulation (DBS)

As previously mentioned, DBS involves surgical implantation of electrodes delivering persistent stimulation to certain parts of the brain. Primarily used as a treatment for movement disorders such as Parkinson's disease, essential tremor and dystonia, DBS is also being explored as a potential treatment for a variety of other conditions [27, 29]. These include Alzheimer disease, epilepsy, and psychiatric disorders such as major depression, obsessive–compulsive disorder (OCD) and Anorexia nervosa. The dysfunction of regions within the basal ganglia, including the subthalamic nucleus (STN) and globus pallidus, internus (GPi), is known to be a cause of various neurological and psychiatric disorders [6]. Therefore, these regions are commonly selected as primary targets for DBS [3, 29, 34].

It is apparent that DBS operates by affecting the electrical activity of local neurons within the targeted regions. However, the precise way in which these neurons are affected remains unknown [2]. Instead of a singular function of action, DBS is believed to work by a variety of mechanisms. Currently, there are a few selected hypotheses aimed at explaining its effects.

Contrary to initial belief, studies have shown that the mechanisms of DBS extends beyond mere excitation and inhibition of local neurons [2, 11, 12, 16, 21]. Instead, it affects the entire network being stimulated, involving processes such as neural and synaptic plasticity. Moreover, the timing of therapeutic outcomes varies significantly. While some effects are seen within seconds, others may take several months [2]. The mechanisms and response times rely not only on the specific neurological disorder being treated but also on the type of brain region undergoing stimulation [12, 16].

Another key aspect regarding DBS is that stimulation primarily targets local axons rather than the soma [18]. This is because the soma typically requires a higher threshold for activation. As a result, when stimulation is applied in a particular area, it is possible for a soma located further away to be activated instead of the nearest soma. This can occur if the distant soma has axons and dendrites positioned closely to the electrode. Consequently, understanding the impact of DBS on axonal pathways are of particular importance.

Furthermore, the complex nature of DBS lies in the multitude of stimulation parameters available for selection [14, 25]. Currently, testing and determination of stimulation parameters, such as electrode position, stimulation amplitude, duration and frequency, are performed during the surgical procedure itself, often demanding several visits [8]. Both patients and medical staff face considerable difficulties with this approach. The relationship between the chosen stimulation parameters and their impact on neuronal activity remains unknown [39], and establishing a correlation between these parameters and neuronal response would help alleviate some of these difficulties.

2.2 Neural Modeling and Simulation

Neural modeling and simulation are crucial in advancing our understanding of the brain, especially when empirical research can be slow due to reasons such as ethical considerations or lack of accessibility [2, 8, 13, 31]. The fundamental aim of computational models is to realistically replicate the biophysical properties of the brain, providing us with a controlled and accessible research environment to explore the mechanisms underlying various brain functions. Nevertheless, computational methods have yet to reach their potential and are subject to their own limitations. These limitations include insufficient computational power and resources [35] or oversimplified models that deviate from realistic standards [38]. However, combining empirical and computational research can benefit the field of neuroscience as a whole.

The Hodgkin-Huxley (HH) Model

The Hodgkin-Huxley (HH) model stands out as the most used and well-known mathematical model to describe the electrical properties involved in axons [36]. Their model provided the first quantitative description of the mechanisms involved in the generation of action potentials, and demonstrated the significance of ion channels within the axon membrane [20].

In the HH model, the action potential is described in terms of the membrane potential, which is influenced by the behavior of voltage-gated ion channels. The model involves a set of differential equations which describe the conductance of sodium channels, potassium channels and leak channels. Moreover, the model considers the probability of these channels being open or closed at a given membrane potential. Although additional voltage-gated ion channels have been discovered since [22], the HH model accurately replicates the characteristics of action potentials observed experimentally [20].

Multicompartmental Models

As most neurons are non-isopotential, meaning the membrane potential varies spatially along the axon, it is of high interest to explore how action potentials propagate through axons with complex geometries. Such considerations has led to the development of multicompartmental models, where the axon is seen as being divided into distinct cylindrical compartments. Each compartment is characterized by an area (a) equal to πdl, representing the product of the diameter (d) and the length (l) of the compartment.

Multicompartmental models utilize circuit theory to characterize the electrical properties of each compartment [37], while cable theory is employed to describe the collective behavior of the axon [24]. In this context, a patch of cell membrane is seen as an RC circuit. The cell membrane is regarded as capacitor with a specific capacitance (C_m), and a resistor with a specific resistance (R_m). Furthermore, current can flow both extracellularly and intracellularly along the axon, where the intracellular medium holds a specific axial resistivity (R_a).

Let us examine a compartment referred to as j, and the neighboring compartments denoted $j+1$ and $j-1$. As a result of Kirchhoff's current law, combined with the capacitive current and Ohm's law, we are left with the fundamental equation of a compartmental model [37]. Specifically

$$C_m \frac{dV_j}{dt} = \frac{E_m - V_j}{R_m} + \frac{d}{4R_m}\left(\frac{V_{j+1} - 2V_j + V_{j-1}}{l^2}\right) + \frac{I_{e,j}}{\pi dl}. \tag{2.1}$$

Here, V denotes the membrane potential of the specified compartment, E_m represent the resting membrane potential, or equilibrium potential, and $I_{e,j}$ signifies a current injected from an electrode into compartment j.

NEURON and LFPy

The NEURON simulator and the LFPy 2.0 Python module are powerful tools for neural modeling and simulation, providing environments for exploring detailed neural structures and their electrical properties.

The NEURON simulator employs multicompartmental models to allow for creation of neurons with complex geometries [10, 17]. Specifically, NEURON performs integration methods, such as the backward Euler or Crank-Nicholson method, to numerically solve equation (2.1). The primary goal of NEURON is to calculate the transmembrane voltage and current of each individual compartment within the neural structure being modeled.

On the other hand, LFPy 2.0 relies on the calculated transmembrane voltages and currents of NEURON to effectively compute the extracellular potential [15, 26]. This allows for easy modeling of stimulation and recording devices, enabling us to study the effects of externally applied electrical potentials [26]. These features are especially relevant in deep brain stimulation (DBS) research.

2.3 Machine Learning in Neuroscience

Machine learning has emerged as a valuable tool across various fields, enabling recognition of complex patterns from extensive datasets. The field of Neuroscience is no exception, as it provides new ways to understand brain function and make predictions regarding various aspects of neural activity [40].

As mentioned in chapter 1, the main objective of this thesis is to use machine learning techniques to make predictions of axon spiking behavior in response to stimuli. This approach involves training algorithms on known patterns of neural activity, also known as supervised learning. Specifically, supervised learning revolves around situations where the outcome of the input data is already known. The aim of this learning framework is to learn from the existing information to make predictions based on arbitrary input data. Supervised learning includes both classification and regression tasks. Classification refers to the task of predicting outcomes that belong to distinct categories, while regression involves predicting continuous numerical values. In this thesis, we are faced with a binary classification problem, where the outcome reduces to the occurrence (1) or absence (0) of an action potential.

Feedforward Artificial Neural Networks (FNNs)

Feedforward artificial neural networks (FNNs) are highly effective in solving binary classification problems. They consist of nodes arranged in layers, including an input layer, one or more hidden layers, and an output layer. FNNs operate by letting information flow forward from the input layer, through the hidden layers, eventually reaching the output layer. In the case of a binary classification problem, the output layer consists of one node.

Each node carries an unknown parameter, also known as weights, which for the nodes in subsequent layers are the weighted sum of the input weights. These weights are then passed through an activation function, which essentially determines the degree of emphasis assigned to each node.

The rectified linear unit (ReLU) activation function and the Sigmoid activation function are two examples of commonly used activation functions. The ReLU activation function is defined such that

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise.} \end{cases} \tag{2.2}$$

This function returns zero for any negative input value, and the input value itself otherwise.
On the other hand, the Sigmoid activation function states that

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

which returns any value between 0 and 1. The Sigmoid activation function is particularly useful in binary classification problems.

Optimization and Training

The training of an artificial neural network (ANN) involves the concept of optimizing its performance by employing an optimization algorithm, such as the adaptive moment estimation (ADAM) optimizer [23, 32]. By utilizing a method known as backpropagation [33], this algorithm calculates the gradients and updates the weights of each node during training. The aim of the optimization algorithm is to minimize the loss function, which calculates the differences between predicted and actual outcomes. In the context of binary classification, the binary cross-entropy loss (BCELoss) function can be employed to calculate such loss [19].

Regularization

In machine learning, regularization techniques are included as a way to prevent overfitting, a phenomenon in which the model performs well on the training data but fails to generalize to the test data. To address this issue, regularization techniques such as early stopping and dropout are employed. Early stopping terminates the training process when the performance of the model fails to improve for a certain amount of iterations. On the other hand, dropout prevents overfitting by ignoring the contributions of a fraction of nodes during each iteration. These regularization techniques improves the ability of the model to generalize, thereby promoting genuine learning rather than memorization.

Evaluation Metrics

Evaluation metrics are crucial in assessing the performance of machine learning (ML) algorithms. When it comes to binary classification, the four most well-known metrics include accuracy, precision, recall, and the F1 score. These evaluation metrics are reliant on predictions made by the machine learning model, specifically taking into account the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

Accuracy provides an overall ratio of correct predictions made by the model, i. e.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2.4}$$

where a higher accuracy indicates better performance. Furthermore, precision is defined such that

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.5}$$

This equation indicated the model's ability to avoid false positives. Additionally, recall is an evaluation metric that reflects the model's ability to avoid false negatives, where

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.6}$$

Lastly, the F1 score combines precision and recall to provide the model's ability to avoid both false positives and false negatives. It is defined such that

$$\text{F1 score} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.7}$$

These evaluation metrics offer important information about the performance of ML models, allowing for assessment of their accuracy, precision, recall, and overall ability to make reliable predictions.

Chapter 3

Methods

In this chapter, we will discuss the methods utilized to accomplish our main objective; exploring the potential of machine learning in predicting the occurrence of action potentials in axons in response to different stimuli. We begin by explaining how we modeled and simulated the axon, followed by an overview of the procedure used to generate the datasets. Lastly, we discuss the development of our machine learning algorithm and present our approach for evaluating its performance.

3.1 Neural Modeling and Simulation

Using the NEURON simulator, we construct an axon composed of numerous segments. The shape, length and diameter of the axon are determined by lists of Cartesian coordinates and diameters of each segment. Specifically, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ represent lists of the x-, y- and z-coordinates of each segment, while $\mathbf{d}$ represents a list of segment diameters. Thus, we have that

$$\begin{aligned}\mathbf{x} &= [x_1,\ x_2,\ \ldots,\ x_{n_{\text{seg}}}],\\ \mathbf{y} &= [y_1,\ y_2,\ \ldots,\ y_{n_{\text{seg}}}],\\ \mathbf{z} &= [z_1,\ x_2,\ \ldots,\ z_{n_{\text{seg}}}]\end{aligned}$$

and

$$\mathbf{d} = [d_1,\ d_2,\ \ldots,\ d_{n_{\text{seg}}}],$$

where n_{seg} denotes the total number of segments.

Straight axons are simply created with Cartesian coordinates that define a linear path. However, to better mimic the shapes of real axons and their meandering trajectories, we perform cubic spline interpolation in the x- and y-direction. This is done by using the CubicSpline class from the SciPy library.

Furthermore, we employ the axon morphology with LFPy 2.0 to define the axial resistivity, membrane capacitance, initial membrane potential, as well as the start and end times of our simulations. Any other optional parameters related to the axon are kept to their default values. Subsequently, the LFPy.Cell class is used to create the cell object representing our axon. To define the position of the axon, we utilize the LFPy.set_pos and LFPy.set_rotation methods, which allow us to specify the Cartesian coordinates of

the axon endpoint and the rotation of the axon around the Cartesian axes, respectively.

Having completed the setup of our axon, we are now ready to model the electrical stimulation. Let us begin by assuming that the medium surrounding the axon is the same in all positions (homogeneous), and that the extracellular conductivity, denoted σ, remains the same in all directions (isotropic). Under these assumptions, the extracellular potential from a stimulus point source is given as

$$\phi = \frac{I}{4\pi\sigma R}, \tag{3.1}$$

Here, I represents the stimulus current, σ denotes the extracellular conductivity, and R indicates the distance between the stimulus point source and the location at which the extracellular potential is being evaluated [30]. This equation is employed to compute the extracellular potential resulting from the stimulus point source to each segment of the axon.

We are now ready to simulate the axon-stimulus interaction. To incorporate the extracellular potential, derived from equation (3.1), as a boundary condition in the NEURON model, we employ the LFPy.insert_v_ext method. Subsequently, we initiate the simulation by using the LFPy.simulate method, allowing us to record the membrane voltage and membrane current of each segment.

To ensure the correct setup of our model, we create a plot that illustrates the position of the axon and electrode, the current pulse of the stimulus, the extracellular potential resulting from the stimulus, and the membrane potential. This plotting process is conducted for both straight and curved axons using the Matplotlib library.

Thresholds

Using the same plotting setup, we want to investigate how axons respond to stimuli at their endpoints compared to the midsections. Specifically, we aim to examine the stimulus amplitude threshold for triggering an action potential in the axon under three different scenarios.

In the first scenario, the stimulus point source is placed beside the endpoint of a straight axon. In the second scenario, the axon is shaped with two perpendicular lines of equal length, and the stimulus point source is positioned next to the right angle. Lastly, in the third scenario, the stimulus point source is placed beside the midpoint of a straight axon. In all three scenarios, the distance between the stimulus point source and the closest point on the axon remains constant.

3.2 Data Collection

We utilize our axon model to conduct a multitude of simulations using various axon and stimulus parameters. The aim is to extract valuable information about the axonal activity triggered by stimuli, ultimately generating an extensive dataset suited for machine learning training. In our thesis, we will direct our focus towards three key parameters

that significantly impact axonal activity in response to stimuli; axon positions, stimulus amplitude, and stimulus duration [7].

The axon positions and stimulus durations are randomly generated from a uniform distribution, while the stimulus amplitudes are drawn from a log-uniform distribution. This process generates a set of axon positions, stimulus amplitudes and durations. Simulations are then performed using different combinations of axon positions and stimuli based on these parameters.

Any recorded membrane potential that reaches or exceeds 0 mV are classified as an action potential. During each simulation, we record whether or not an action potential is evoked. Additionally, we calculate the Euclidean distance from the stimulus point source to each segment, and determine the minimum distance among them.

Subsequently, we create a large dataset that includes the parameters of each simulation. The dataset is organized such that each column corresponds to a specific parameter; the stimulus amplitude, stimulus duration, the calculated minimum distance, and the Cartesian coordinates of the axon endpoints. Additionally, the dataset includes a binary column that indicates the occurrence of an action potential, also known as a spike, where a value of zero indicates that no spike was recorded, and a value of one indicates that a spike was recorded.

We are also interested in creating datasets that provide more information about the axon positions. Specifically, we want to create a selection of datasets that incorporate additional coordinates along the axon as parameters, rather than solely including the endpoints. The aim of this approach is to evaluate the optimal parameters required for the machine learning algorithm to achieve its highest performance, as well as to examine the amount of information necessary. For this purpose, we generate datasets with a uniform distribution of 2, 5, 10, 20, 40, 60, 80, 100, 120, 140, 160, 180 and 200 Cartesian points along the axons, which we will denote df_2, df_5, df_{10}, df_{20}, df_{40}, df_{60}, df_{80}, df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200}, respectively. This is done for both straight and curved axons.

To visualize our simulations, we use the Matplotlib library to make a three-dimensional plot showing the position of the electrode alongside some of the axons. Furthermore, we create a scatter plot illustrating the relationship between the minimum distance, stimulus amplitude and the occurrence of spikes. This is also done for both straight and curved axons.

3.3 The Machine Learning Algorithm

We aim to explore the potential of our datasets in the development of a machine learning algorithm that predicts spiking activity in axons. The main objective is to determine the possibility of such predictions, potentially contributing in the field of deep brain stimulation research.

Artificial Neural Network Architecture

In this pursuit, we employ the PyTorch library to create a feedforward neural network model consisting of three linear layers. These layers comprised an input layer with 128 nodes, a hidden layer with 64 nodes, and an output layer with a single node. Our specific challenge concerns a binary classification problem, for which we apply the rectified linear unit activation function (ReLU) to both the initial and hidden layer, and the Sigmoid activation function to the final layer.

Data Preprocessing

It is assumed that the dataset follows a specific format, with columns representing various input parameters as well as the output parameter. In our datasets, these parameters consist of stimulus amplitude, stimulus duration, minimum distance, axon coordinates, and the spike indicator. Once extracted, these parameters are used as features in the machine learning algorithm.

The dataset is divided into training, validation and testing subsets using the StratifiedShuffleSplit class from the Scikit-learn library. This ensures that the distribution of the outcome variable is preserved in all subsets. The size of our validation and test subsets are set to 0.1, indicating that 10% of the datasets are used for validation, and 10% is used for testing.

The input features of the dataset are normalized by applying min-max normalization, which scales the values between 0 and 1. As a result, the feature values become balanced within equal range, preventing certain features from overpowering others solely based on their magnitudes.

Training

The model is trained by the ADAM optimizer, utilizing backpropagation to adjust the model parameters based on computed gradients. The predictions of the model are then compared to the actual outcomes by calculating the loss using the binary cross-entropy loss (BCELoss) function. Furthermore, the loss of the validation data is calculated as a way to implement early stopping in the training process. Training involves a series of iterative updates referred to as epochs. If there is no improvement in the validation loss for a certain amount of epochs, the training is stopped. To prevent overfitting, dropout is also incorporated as a regularization technique, which ignores a fraction of nodes during training by setting their outputs to zero. Additionally, to check for signs of overfitting, we plot the calculated loss of each epoch, considering both the training and validation data.

Evaluation

The performance of our machine learning model is tested with the unseen data from the dedicated test subset. To assess the performance of the model, we employ evaluation metrics based on the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). Specifically, we calculate the accuracy of the predictions

by employing equation (2.4), and the precision utilizing equation (2.5). Furthermore, we calculate the recall using equation (2.6), and lastly, we utilize equation (2.7) to compute the F1 score.

To identify any patterns in the incorrect predictions made by the model, we are interested in visualizing these predictions. This is achieved by generating a scatter plot showing the relationship between minimum distance, stimulus amplitude, false positives (FP), false negatives (FN), true positives (TP) and true negatives (TN).

All evaluation metrics are calculated for the datasets df_2, df_5, df_{10}, df_{20}, df_{40}, df_{60}, df_{80}, df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200}. Subsequently, we generate two plots to visualize the accuracy and F1 score obtained through each dataset. The first plot illustrates the results for the datasets involving simulations of straight axons, while the second plot shows the results for the datasets involving simulations of curved axons.

3.4 Code Availability

The code developed throughout this thesis can be found publicly available on GitHub at https://github.com/kariannestrand/FYS5960.git.

Chapter 4

Results

This chapter provides a presentation of the results obtained through our methods outlined in chapter 3. Firstly, we will examine the outcome of modeling and simulating both the straight and curved axon. Following that, we will present the generated datasets and the associated visualizations. Finally, we deliver an overview of the outcomes derived from assessing the performance of our machine learning algorithm.

4.1 Overview of the Model

In our simulations, we modeled each axon with a length of 3000 µm, consisting of 200 compartments. Each segment within the axon was created with a diameter of 1.0 µm, a membrane capacitance of 1.0 $\mu F/cm^2$, and an axial resistivity of 150 Ωcm. Additionally, we chose -65 mV as the resting membrane potential, and kept a constant extracellular conductivity of 0.3 Siemens per meter. The total simulation time was set to 120 ms, using a time step size of 2^{-5} ms, with the electrode positioned at the origin.

Figure 4.1 and 4.2 illustrate our simulation of an electrode giving stimulus to a straight and curved axon, respectively. These figures clearly show the rapid change in the membrane potential, thereby imitating the characteristics of an action potential in response to stimuli.

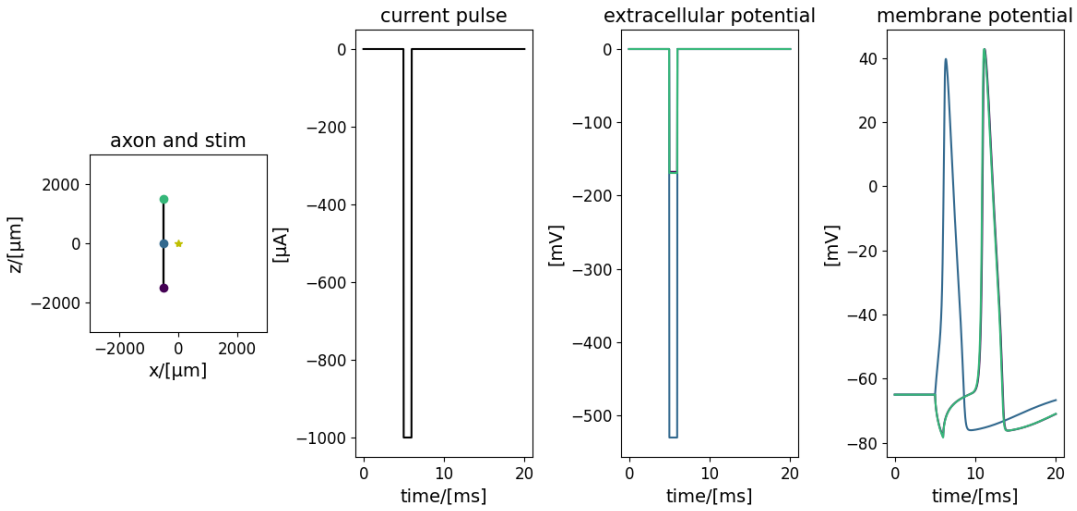


Figure 4.1: The first panel shows the position of the electrode (star) and the straight axon (line) with the three dots marking the midpoint and the endpoints of the axon. The next panels show the current pulse of the stimulus, the extracellular potential resulting from the stimulus, and the membrane potential of each compartment marked by dots, respectively. Here the stimulus amplitude was set to −1000 µA, and stimulus duration was set to 1.0 ms.

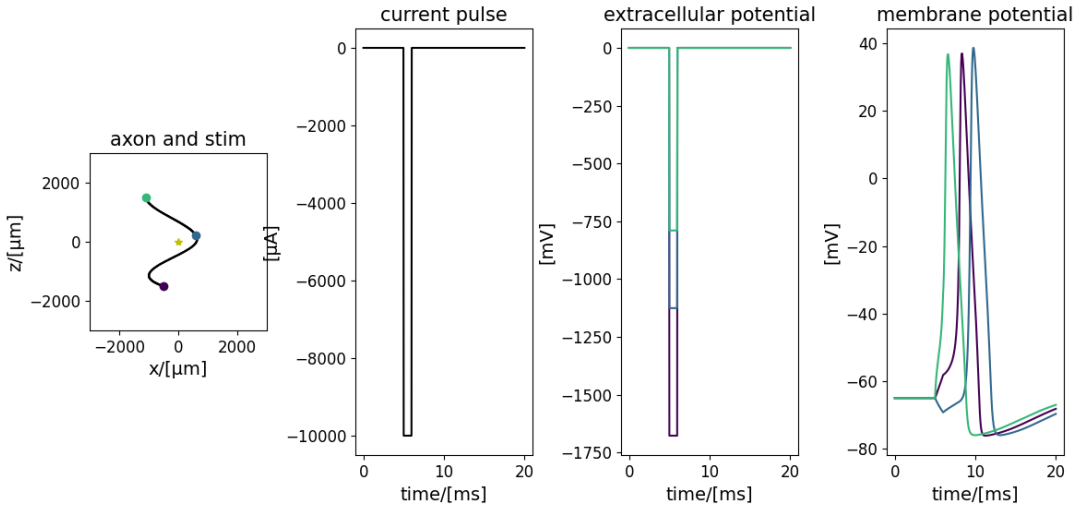


Figure 4.2: The first panel shows the position of the electrode (star) and the curved axon (line), with the three dots marking the midpoint and endpoints of the axon. The next panels show the current pulse of the stimulus, the extracellular potential resulting from the stimulus, and the membrane potential of each compartment marked by dots, respectively. Here the stimulus amplitude was set to −10000 µA, and stimulus duration was set to 1.0 ms.

Thresholds

To develop a better understanding of how extracellular current stimulation is influenced by the shape of the axon, we examined three different scenarios.

The threshold for evoking an action potential in the first scenario, where the electrode was positioned beside the endpoint, was found to require a stimulus amplitude of approximately −158 µA. The simulation illustrating this scenario is showed in figure 4.3.

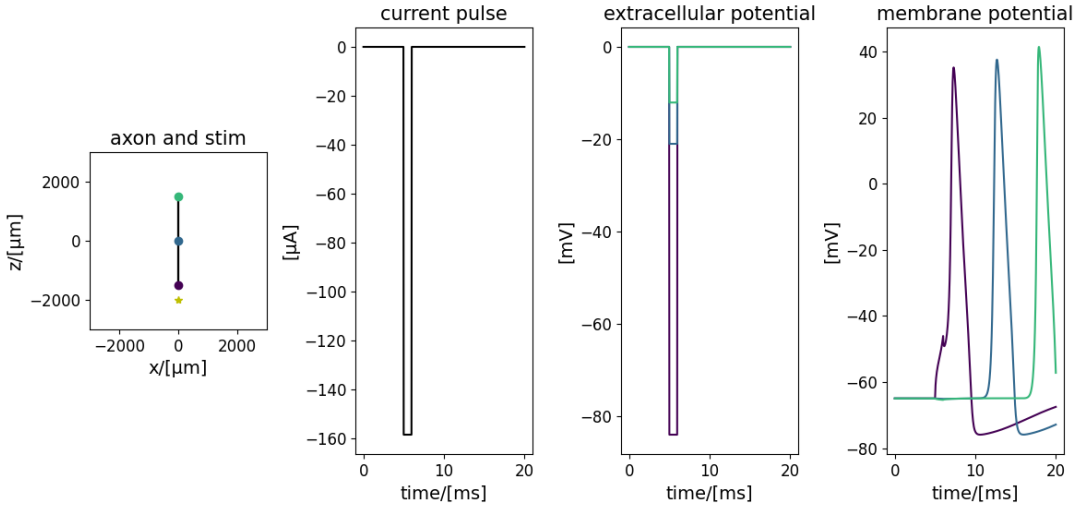


Figure 4.3: The first panel shows the position of the electrode (star) and the axon (line), with the three dots marking the midpoint and endpoints of the axon. The next panels show the current pulse of the stimulus, the extracellular potential resulting from the stimulus, and the membrane potential of each compartment marked by dots, respectively. Here the stimulus amplitude was set to approximately −158 µA, marking the threshold for evoking an action potential in the axon, while the stimulus duration was set to 1.0 ms.

In the second scenario, where the electrode was positioned beside the midpoint of an axon shaped as a right angle, the threshold was determined as a stimulus amplitude of approximately −200 µA. The simulation showing this scenario is illustrated in figure 4.4.

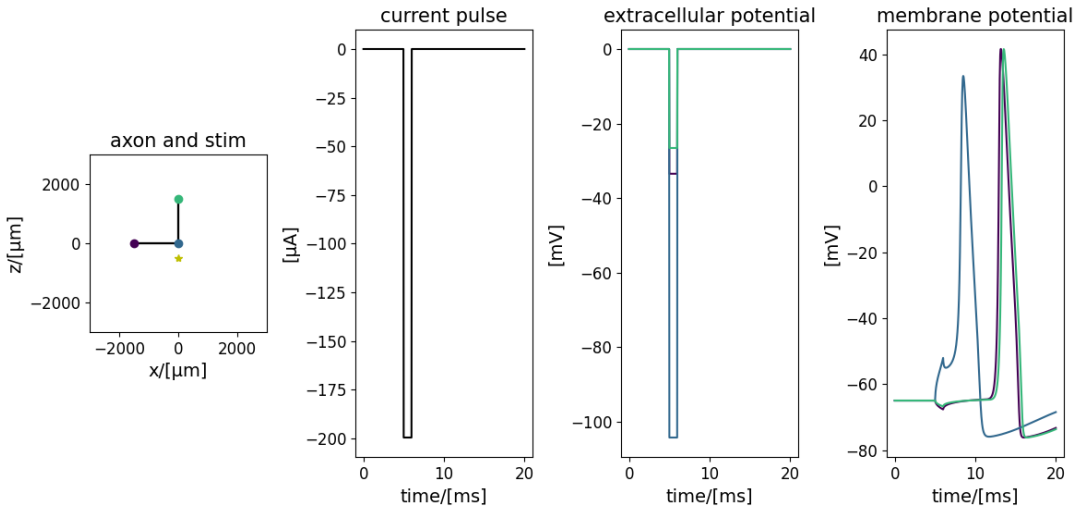


Figure 4.4: The first panel shows the position of the electrode (star) and the axon (line), with the three dots marking the midpoint and endpoints of the axon. The next panels show the current pulse of the stimulus, the extracellular potential resulting from the stimulus, and the membrane potential of each compartment marked by dots, respectively. Here the stimulus amplitude was set to approximately -200 µA, marking the threshold for evoking an action potential in the axon, while the stimulus duration was set to 1.0 ms.

Lastly, a stimulus amplitude of approximately -398 µA was found to be the threshold in the third scenario, where the electrode was positioned beside the midpoint of a straight axon. The simulation of this scenario can be seen in figure 4.5.

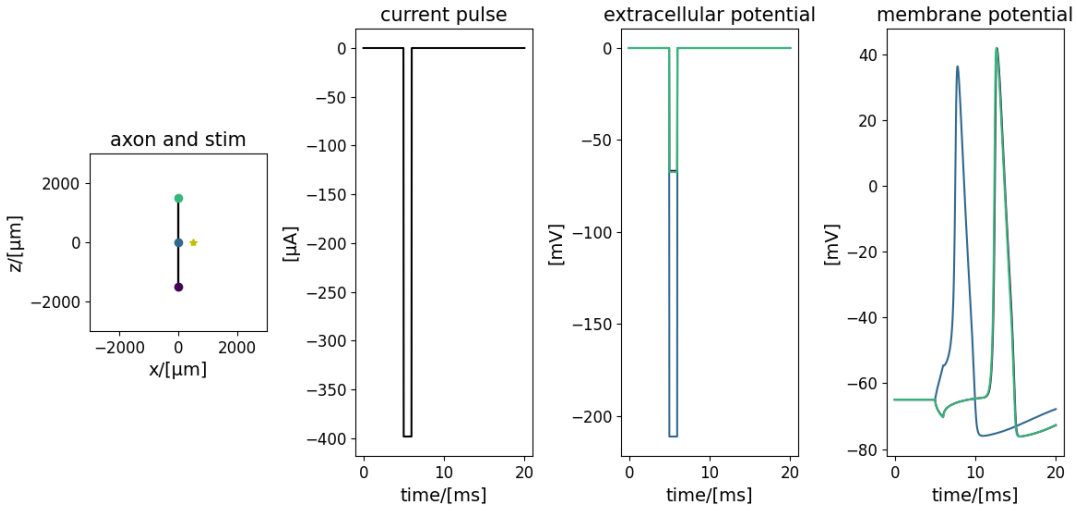


Figure 4.5: The first panel shows the position of the electrode (star) and the axon (line), with the three dots marking the midpoint and endpoints of the axon. The next panels show the current pulse of the stimulus, the extracellular potential resulting from the stimulus, and the membrane potential of each compartment marked by dots, respectively. Here the stimulus amplitude was set to approximately -398 µA, marking the threshold for evoking an action potential in the axon, while the stimulus duration was set to 1.0 ms.

We see that an axon with the electrode placed at its endpoint demanded the lowest amplitude, while both a right-angled axon and a straight axon with the electrode placed beside their midpoints required larger amplitudes, as shown in previous literature [1, 9]. Predicting the conditions under which an action potential occurs is a significant challenge, as it depends on the morphological details of the cell and various stimulation parameters. As a result, we will turn to machine learning as a means to solve this problem.

4.2 Overview of the Data

The stimulation amplitude varied between -10000 µA and -10 µA, while the stimulation duration ranged from 0.5 ms to 2.0 ms. Axon positions were defined based on the Cartesian coordinates of the endpoint, and were adjusted between -3000 µm and 3000 µm in each direction. Additionally, axon positions were also determined by the rotation of the axon around the Cartesian axes, ranging from 0 to 2π.

By employing these specified boundaries, we successfully generated datasets containing columns representing each parameter. For both straight and curved axons, we created datasets with 2, 5, 10, 20, 40, 60, 80, 100, 120, 140, 160, 180 and 200 Cartesian points along the axon, which resulted in a total of 26 datasets. Each of these datasets was constructed with a size of 100 000 samples. The 100 000 simulations were shared across the 13 datasets of the straight axons, while an additional 100 000 simulations were conducted for generating the 13 datasets of the curved axons.

Figure 4.6 illustrates a selection of the simulated axon positions surrounding the electrode, which were used to generate the datasets of straight axons. Similarly, figure 4.7 shows a selection of the simulated axon positions used to create the datasets of curved axons.

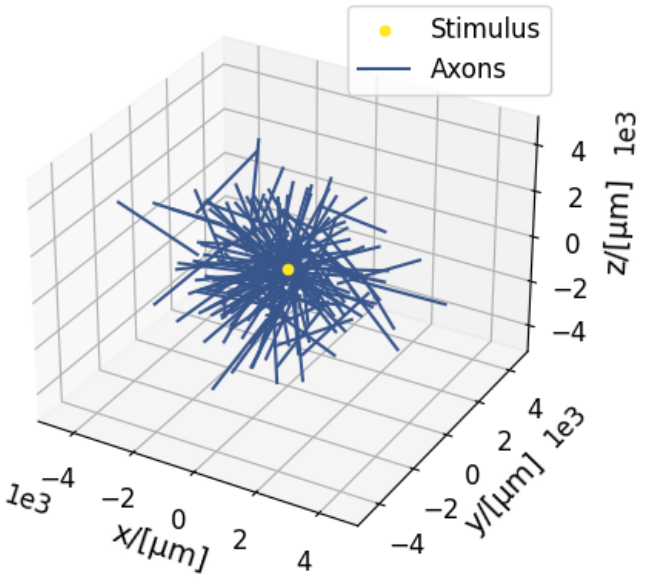


Figure 4.6: 3D representation showing the positions of the electrode (dot) and 150 straight axons (lines).

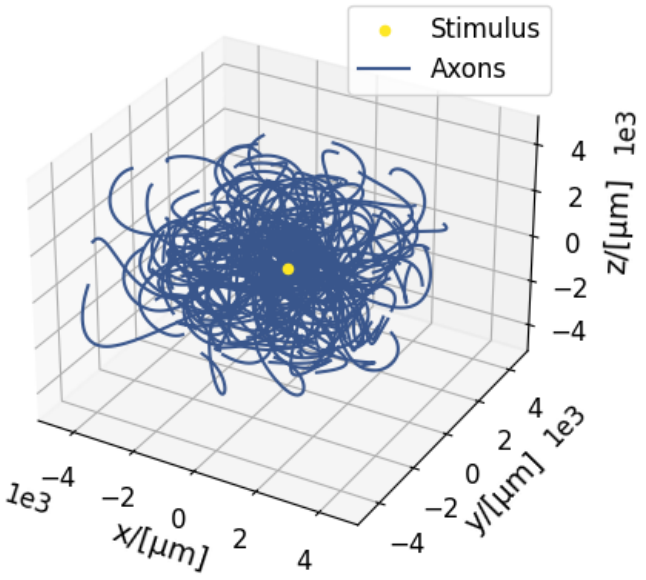


Figure 4.7: 3D representation showing the positions of the electrode (dot) and 150 curved axons (lines).

Furthermore, the scatter plot showing the relationship between the stimulus amplitude,

minimum distance and the occurrence of spikes can be seen in figure 4.8 across 100 000 simulations of straight axons. Additionally, figure 4.9 displays the corresponding scatter plot for the 100 000 simulations of the curved axons.

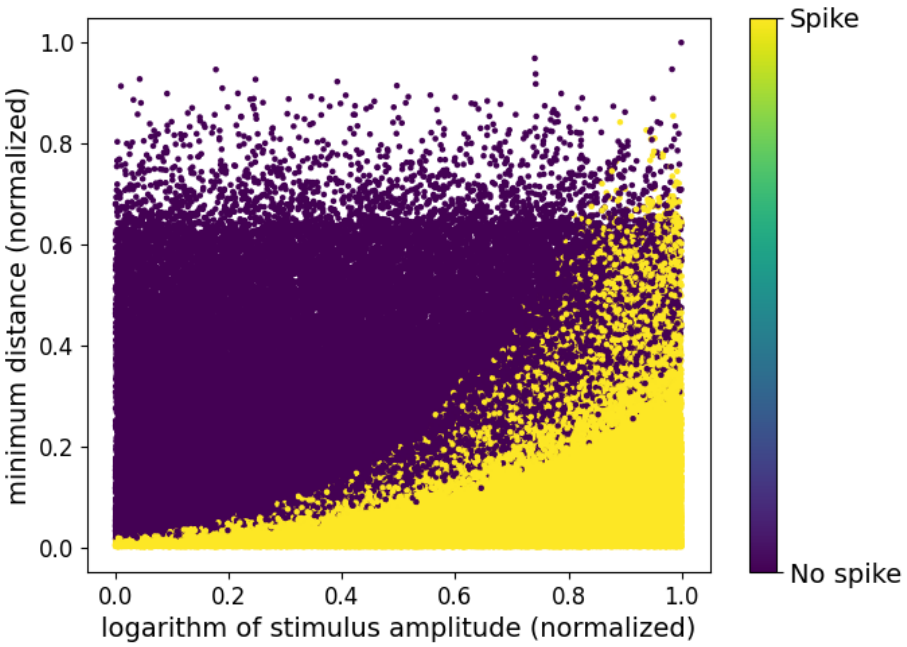


Figure 4.8: Scatter plot showing the relationship between the normalized minimum distance, logarithm of the stimulus amplitude and the occurrence of spikes across 100 000 simulations of straight axons. Each dot represents a single simulation, with yellow dots indicating the occurrence of a spike, and purple dots indicating the absence of a spike. There are a total of 49121 yellow dots representing the occurrence of a spike, and 50879 purple dots representing the absence of a spike.

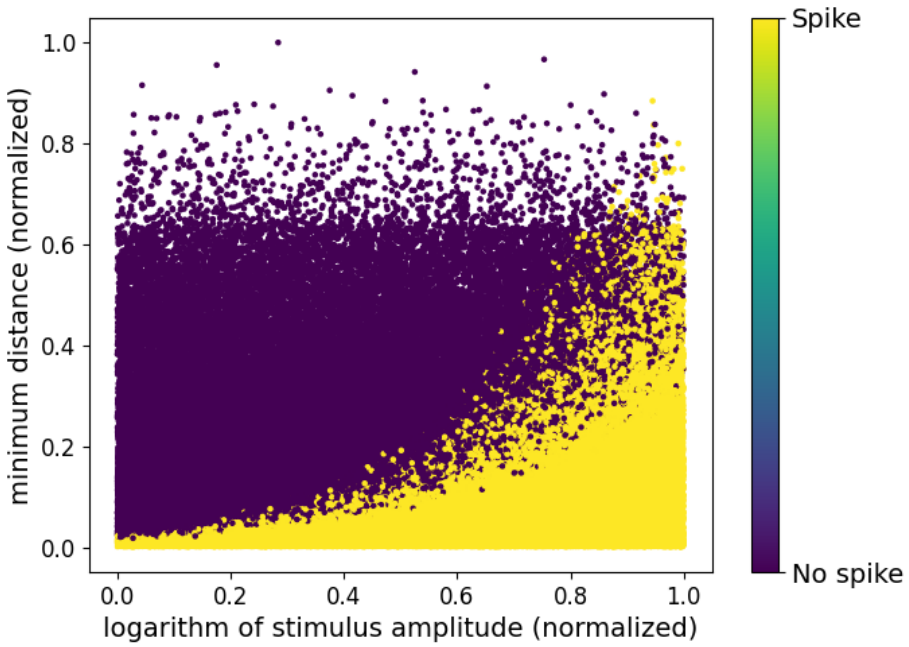


Figure 4.9: Scatter plot showing the relationship between the normalized minimum distance, logarithm of the stimulus amplitude and the occurrence of spikes across 100 000 simulations of curved axons. Each dot represents a single simulation, with yellow dots indicating the occurrence of a spike, and purple dots indicating the absence of a spike. There are a total of 51322 yellow dots representing the occurrence of a spike, and 48678 purple dots representing the absence of a spike.

These scatter plots show a clear pattern between the spiking activity, minimum distance, and the stimulus amplitude. This relationship suggests that the ANN is likely to predict spike occurrences with ease in regions exclusively consisting of positive or negative spikes. However, it may struggle with predictions in the boundary region, where positive and negative spikes overlap.

The datasets were divided into training, validation and test subsets, with equal distribution of outcomes. Moreover, each parameter was appropriately normalized as described in section 3.3.

4.3 Performance Evaluation of the Machine Learning Algorithm

The model of the machine learning algorithm was trained using each generated dataset individually. In this section, we will present the results obtained by training the model with df_2 as its dataset, encompassing simulations of both straight and curved axons. Additionally, the comparison of the performance across all datasets will be presented.

During training, a learning rate of 0.01 was employed. Furthermore, dropout was utilized as a regularization technique, with a rate of 0.1. Lastly, early stopping was employed, terminating the training process if there was no observed improvement in validation loss after 10 consecutive epochs.

Figure 4.10 shows the result of plotting the calculated loss of each epoch, covering both the training and validation data, when training the model using df_2 with straight axon simulations as its dataset.

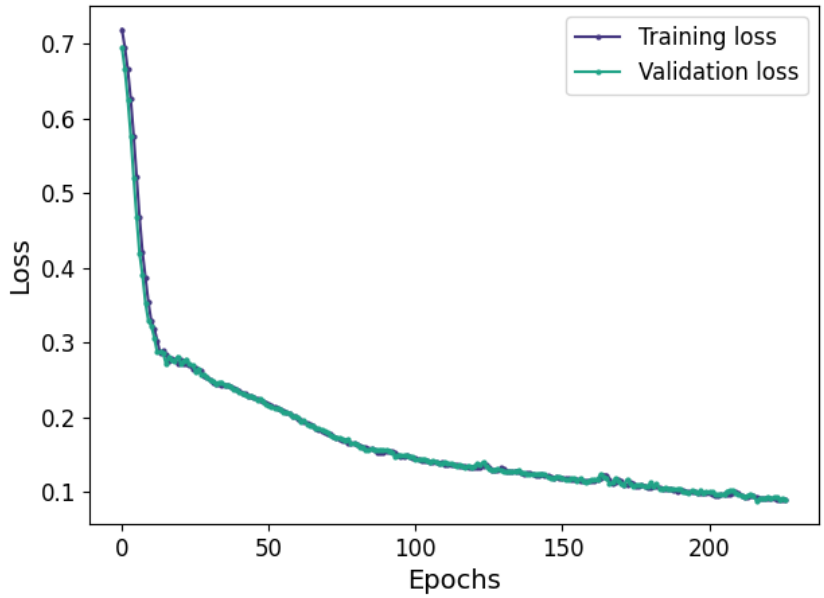


Figure 4.10: Plot showing the calculated training loss and validation loss for each epoch throughout the training process. The model is trained using df_2 with straight axon simulations as its dataset.

Similarly, figure 4.11 shows the corresponding result of the loss plot when training the model using df_2 with curved axon simulations as its dataset.

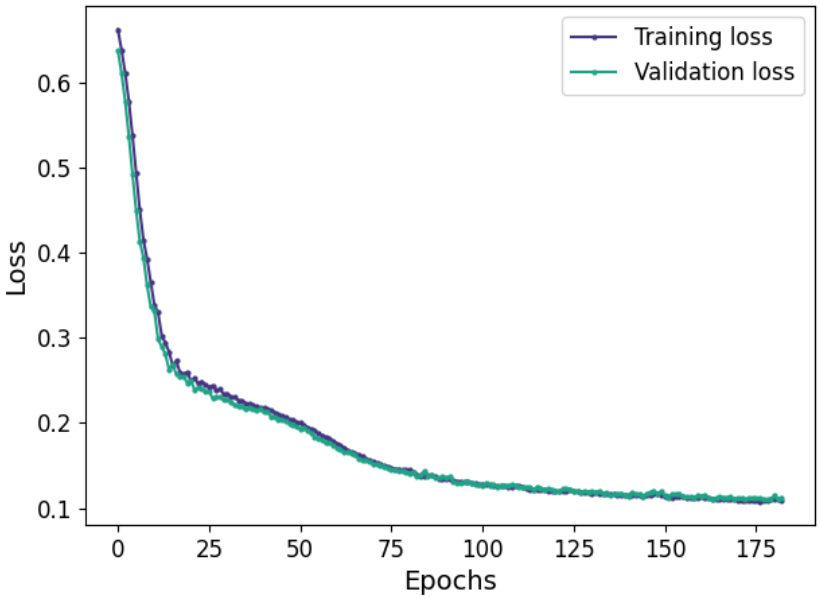


Figure 4.11: Plot showing the calculated training loss and validation loss for each epoch throughout the training process. The model is trained using df_2 with curved axon simulations as its dataset.

The plots within figures 4.10 and 4.11 clearly demonstrate a decline in the computed loss. Furthermore, the training and validation loss appears remarkably similar, seemingly showing no sign of overfitting.

Section 7.1 in the appendix contains the plots of the training and validation loss computed from training the model using the remaining datasets that involve simulations of straight axons. Specifically, figure 7.1 presents the loss plots obtained when training the model with df_5, df_{10}, df_{20}, df_{40}, df_{60} and df_{80}, while figure 7.2 shows the loss plots when training the model with df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200}. Correspondingly, the equivalent loss plots obtained through the curved axon simulations can be found in figures 7.3 and 7.4.

In figure 4.12, we see the scatter plot showing the relationship between the minimum distance, stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN) when training the model using df_2 with straight axon simulations as its dataset.

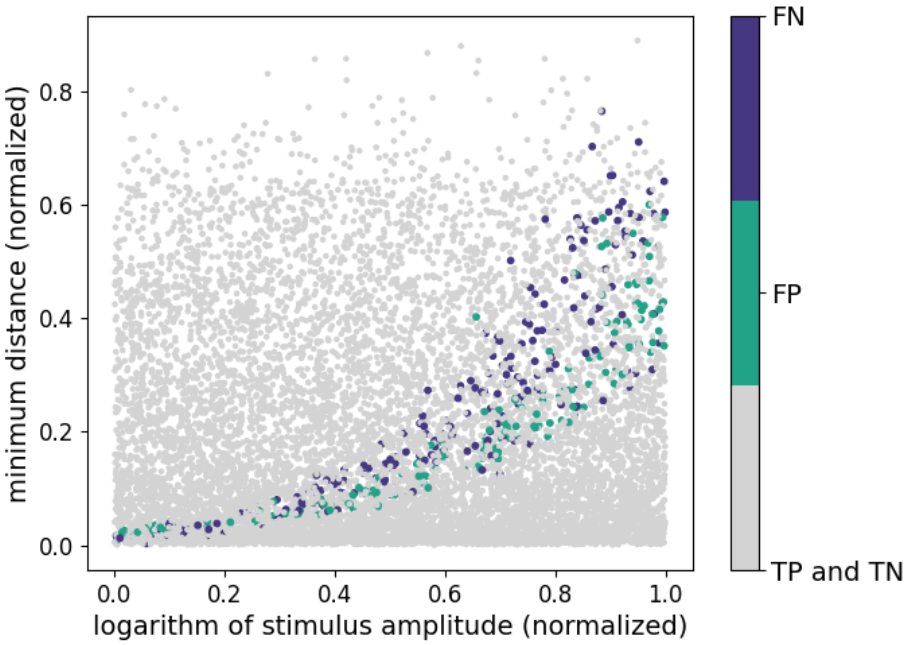


Figure 4.12: Scatter plot showing the relationship between the normalized minimum distance, normalized stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). The model is trained using df_2 with straight axon simulations as its dataset. Each dot represents a single straight axon simulation, with gray dots indicating the correct predictions of the model (TP and TN), teal dots indicating false positives (FP), and purple dots indicating false negatives (FN).

The corresponding scatter plot using df_2 with curved axon simulations can be seen in figure 4.13.

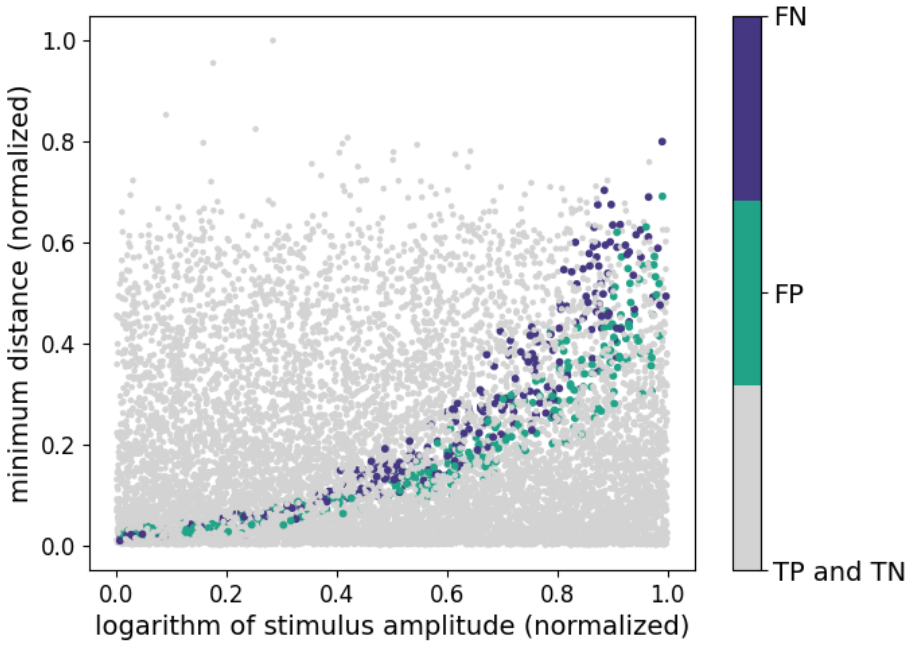


Figure 4.13: Scatter plot showing the relationship between the normalized minimum distance, normalized stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). The model is trained using df_2 with curved axon simulations as its dataset. Each dot represents a single straight axon simulation, with gray dots indicating the correct predictions of the model (TP and TN), teal dots indicating false positives (FP), and purple dots indicating false negatives (FN).

As predicted, we can see that the errors of these scatter plots are concentrated near the boundary region observed in the relationship between the minimum distance and stimulus amplitude seen in figures 4.8 and 4.9.

Likewise, when training the model with the remaining datasets that involve simulations of straight axons, we obtained the scatter plots showing the relationship between the minimum distance, stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). Section 7.2 in the appendix contains these scatter plots, where figure 7.5 presents the scatter plots obtained when training the model with df_5, df_{10}, df_{20}, df_{40}, df_{60} and df_{80}. Furthermore, figure 7.6 shows the scatter plots obtained when training the model with df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200}. The equivalent scatter plots acquired through the curved axon simulations can be found in figures 7.7 and 7.8.

In table 4.1, we have collected the number of true positives (TN), true negatives (TN), false positives (FP) and false negatives (FN), along with the computed accuracy, precision, recall and F1 score for all datasets involving simulations of straight axons.

Data	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 score
df_2	4716	4910	145	229	0.9626	0.9702	0.9537	0.9619
df_5	4780	4598	457	165	0.9378	0.9127	0.9666	0.9389
df_{10}	4620	4861	194	325	0.9481	0.9597	0.9343	0.9468
df_{20}	4812	4596	459	133	0.9408	0.9129	0.9731	0.9421
df_{40}	4575	4455	600	370	0.9030	0.8841	0.9252	0.9042
df_{60}	4007	4933	122	938	0.8940	0.9705	0.8103	0.8832
df_{80}	3824	4969	86	1121	0.8793	0.9780	0.7733	0.8637
df_{100}	3875	4949	106	1070	0.8824	0.9734	0.7836	0.8683
df_{120}	4724	3706	1349	221	0.8430	0.7779	0.9553	0.8575
df_{140}	4546	3838	1217	399	0.8384	0.7888	0.9193	0.8491
df_{160}	3583	4990	65	1362	0.8573	0.9822	0.7246	0.8339
df_{180}	3559	4895	160	1386	0.8454	0.9570	0.7197	0.8216
df_{200}	4286	2942	2113	659	0.7228	0.6698	0.8667	0.7556

Table 4.1: The number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), and the related accuracy, precision, recall and F1 score for each dataset with the straight axon simulations.

Similarly, table 4.2 contains the corresponding values for all datasets involving simulations of curved axons.

Data	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 score
df_2	4992	44460	278	270	0.9452	0.9472	0.9486	0.9480
df_5	5080	4296	442	182	0.9376	0.9200	0.9654	0.9421
df_{10}	4750	4601	137	512	0.9351	0.9720	0.9027	0.9361
df_{20}	5017	4292	446	245	0.9309	0.9184	0.9534	0.9356
df_{40}	5126	4002	736	136	0.9128	0.8744	0.9742	0.9216
df_{60}	4895	4042	696	367	0.8937	0.9705	0.9303	0.9021
df_{80}	4953	3793	945	309	0.8746	0.8398	0.9413	0.8876
df_{100}	4932	3868	870	330	0.8800	0.8501	0.9373	0.8915
df_{120}	4012	4593	145	1250	0.8605	0.9651	0.7624	0.8519
df_{140}	5094	3523	1215	168	0.8617	0.8074	0.9681	0.8805
df_{160}	4189	4424	314	1073	0.8613	0.9303	0.7961	0.8580
df_{180}	4505	2640	2098	757	0.7145	0.6823	0.8561	0.7594
df_{200}	3960	2732	2006	1302	0.6692	0.6638	0.7526	0.7054

Table 4.2: The number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), and the related accuracy, precision, recall and F1 score for each dataset with the curved axon simulations.

Figure 4.14 shows the result of plotting the accuracy and F1 score of the model for each dataset involving simulations of straight axons.

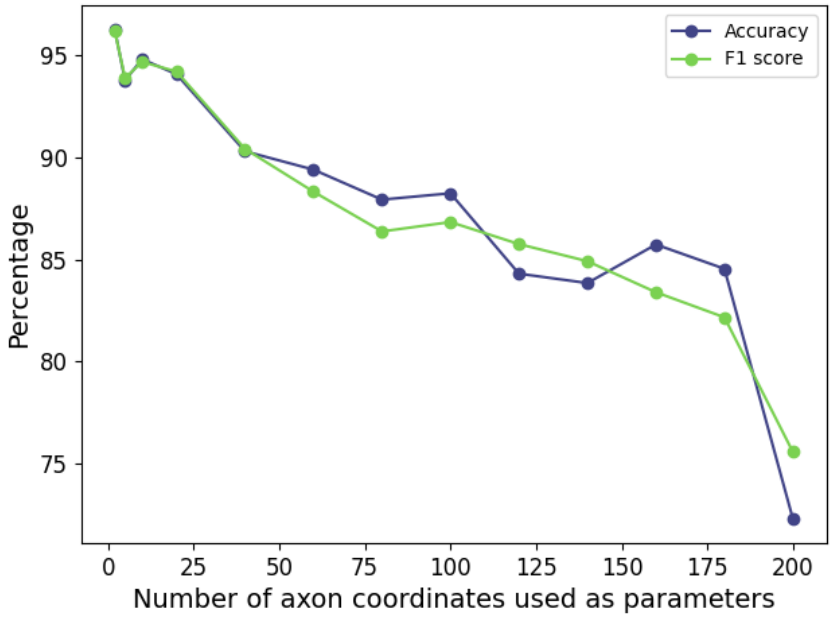


Figure 4.14: The accuracy and F1 score plotted for all datasets with straight axon simulations.

Correspondingly, figure 4.15 shows the result of plotting the accuracy and F1 score for each dataset involving simulations of curved axons.

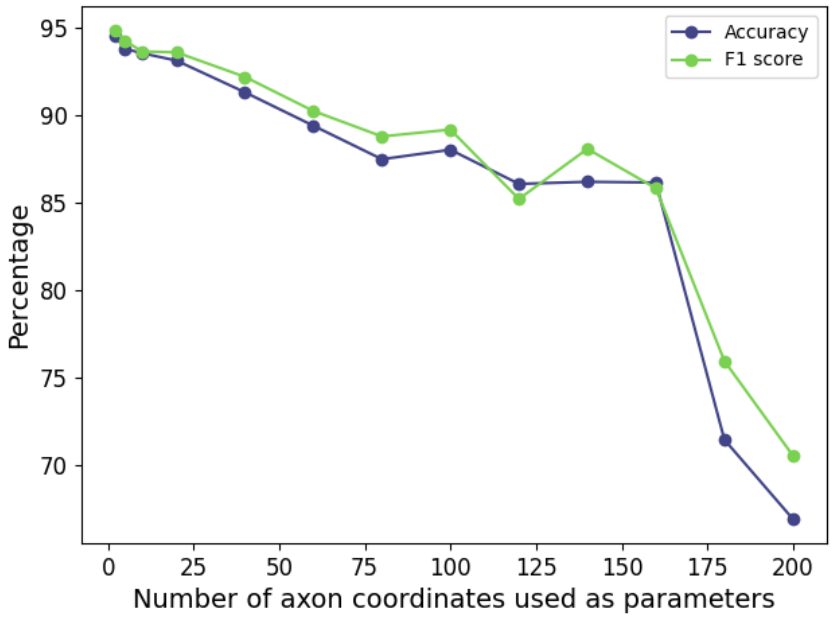


Figure 4.15: The accuracy and F1 score plotted for all datasets with curved axon simulations.

These figures illustrate a declining trend in performance when using datasets with additional axon points as parameters. Additionally, we observe that the accuracy and F1 score are of similar values for each dataset. This trend is further supported by the loss plots in figures 7.1, 7.2, 7.3 and 7.4, and the scatter plots in figures 7.5, 7.6, 7.7 and 7.8. In these plots, it is evident that the loss function loses its shape, and the scatter

plots no longer exhibit distinct patterns with an increasing number of axon points used as parameters. This unexpected decrease in performance, despite the increasing availability of information about the shape of the axon, came as a surprise. In chapter 5, we will discuss the potential reasons behind this observation.

Chapter 5

Discussion & Conclusion

In this chapter, we aim to provide a detailed discussion of the results presented in chapter 4. We begin by giving a summary of our findings, and continue with an analysis of these results. Lastly, we discuss future work and conclude our thesis in relation to the research objectives.

5.1 Summary of Findings

As evident by figures 4.1 and 4.2, we successfully employed the NEURON simulator and LFPy 2.0 to simulate both straight and curved axons in the presence of a stimulus point source. The simulated axons showed their electrical properties in response to stimuli through the prominent spikes observed in the membrane potentials, effectively mimicking the occurrence of action potentials. Additionally, we found that positioning electrodes near the midpoints of axons required higher amplitudes to generate a spike, compared to placing the electrode near the endpoints. These results are in accordance with previous findings [1, 9]. Notably, the straight axon with the electrode placed beside its midpoint revealed the highest threshold amplitude.

During the training process, it was observed that the loss computed for both the test and validation data effectively decreased, and remained virtually identical throughout. These findings are evident by the loss plots in figures 4.10 and 4.11, and strongly suggest the absence of overfitting. Furthermore, the data showed a clear relationship between spiking activity, the normalized minimum stimulus distance, and the log-normalized stimulus amplitude, as seen by the scatter plots in figures 4.8 and 4.9. The prediction errors of the artificial neural network (ANN) were shown to lie within the boundary area of this relationship, as observed in figures 4.12 and 4.13.

Lastly, when plotting the accuracy and F1 score of the model obtained with each dataset, a clear and similar decrease was observed for both straight and curved axon simulations when incorporating additional axon points as parameters. These findings can be observed in figures 4.14 and 4.15. Moreover, the F1 score, a metric that provides the ability of the model to avoid false negatives and false positives, and the accuracy of the model was found to have similar values for each dataset. The fact that this metric aligns with the overall accuracy of the model indicates a balanced performance, meaning that there is no preference in predicting false positives over false negatives, or vice versa.

5.2 Analysis of Results

In determining spike thresholds, our results showed the significance of electrode placement in relation to the axon endpoints compared to midsections, as previous research has shown [1, 9]. This led to the initial belief that the ANN would need additional information about the location and structure of the axon in order for it to effectively distinguish between endpoints and midsections, thereby influencing its performance. However, our findings revealed that the machine learning model achieved the highest performance when employing the least amount of axon points as features, seemingly treating additional points as noise. This unexpected finding indicates that ANNs can predict spiking behavior to a high degree by using less complex machine learning architecture than expected. Nevertheless, it is important to note that this level of accuracy likely arises from the simplicity of our axon-stimulus model, and may not hold true for more realistic computational models. Consequently, further development of the machine learning algorithm is necessary to enhance its performance in later use.

The errors of the ANN were found to lie within the boundary region of the relationship between the normalized minimum distance and log-normalized stimulus amplitude. It is important to note that even though the ANN performs well overall, it does not undermine the importance of factors such as the mentioned thresholds or the additional information regarding the location and shape of the axon. In our model, the overlapping spiking behavior observed in the boundary region is likely due to the effects of these factors. However, they may drown in the noise of more dominant factors, such as the minimum distance and stimulus amplitude. A first approach in improving the performance in this boundary region would be to increase the test data size within its range, although the primary focus should be on improving the machine learning algorithm to account for increased model complexity.

5.3 Future Work

In future work, the neural model should be improved to more accurately simulate both the axon and electrode. One approach is to include increasingly complex morphological properties of the axon, as well as one or more electrodes with cylindrical shapes. Although previous work has explored more realistic neural models [28], they still hold great potential for being explored within machine learning.

Furthermore, the concepts of this thesis can be applied to point neuron network models, allowing us to study and predict the electrical activity in response to stimuli within neural networks as a whole.

Notably, these neural models will require more complex sets of features, thus requiring the need for more advanced machine learning algorithms.

5.4 Conclusion

As a way to study deep brain stimulation (DBS), our research explored the potential of machine learning techniques in predicting action potentials within axon-stimulus inter-

actions. Through simulations using the NEURON simulator and LFPy 2.0, we employed different axon and stimulus parameters to generate extensive datasets where the occurrence or absence of an action potential were defined as a binary outcome. These datasets were then used to train a feedforward neural network as a way to solve the binary classification problem.

The results showed that the machine learning model achieved a high level of accuracy in predicting the occurrence of action potentials when using features including the minimum distance between the axon and electrode, stimulus amplitude, stimulus duration, and spatial coordinates of axon endpoints. However, when additional spatial coordinates along the axon were included as features, the performance of the machine learning algorithm decreased. This suggests the need for further development of the machine learning algorithm in handling model complexity.

To address our main research objective, we have shown that machine learning can be used to predict action potentials in response to stimuli when using arbitrary axon and stimulation parameters. Although further work is needed to advance the methods presented in this thesis, the proof of concept still remains. This demonstrates the potential of machine learning in DBS research, ultimately leading to improved treatment of brain disorders.

Chapter 6

Bibliography

Bibliography

[1] Aman S. Aberra, Angel V. Peterchev and Warren M. Grill. 'Biophysically realistic neuron models for simulation of cortical stimulation'. In: *Journal of neural engineering* 15.6 (2018), pp. 066023–066023. DOI: 10.1088/1741-2552/aadbb1.

[2] Keyoumars Ashkan et al. 'Insights into the mechanisms of deep brain stimulation'. In: *Nature Reviews Neurology* 13.9 (2017), pp. 548–554. DOI: 10.1038/nrneurol.2017.105.

[3] Alim Louis Benabid et al. 'Acute and Long-Term Effects of Subthalamic Nucleus Stimulation in Parkinson's Disease'. In: *Stereotactic and functional neurosurgery* 60 (1994), pp. 76–84. DOI: 10.1159/000098600.

[4] Alim Louis Benabid et al. 'Combined (Thalamotomy and Stimulation) Stereotactic Surgery of the VIM Thalamic Nucleus for Bilateral Parkinson Disease'. In: *Stereotactic and Functional Neurosurgery* 50 (1987). DOI: 10.1159/000100803.

[5] Hagai Bergmann. *The hidden life of the basal ganglia: at the base of brain and mind*. The MIT Press, 2021. Chap. 18, p. 215.

[6] Hagai Bergmann. *The hidden life of the basal ganglia: at the base of brain and mind*. The MIT Press, 2021. Chap. 1, p. 4.

[7] Hagai Bergmann. *The hidden life of the basal ganglia: at the base of brain and mind*. The MIT Press, 2021. Chap. 18, p. 216.

[8] Alexandre Boutet et al. 'Predicting optimal deep brain stimulation parameters for Parkinson's disease using functional MRI and machine learning'. In: *Nature communications* 12.1 (2021), pp. 3043–3043. DOI: 10.1038/s41467-021-23311-9.

[9] Kelsey L. Bower and Cameron C. McIntyre. 'Deep brain stimulation of terminating axons'. In: *Brain stimulation* 13.6 (2020), pp. 1863–1870. DOI: 10.1016/j.brs.2020.09.001.

[10] Nicholas T. Carnevale and Michael L. Hines. *The NEURON book*. Cambridge University Press, 2006.

[11] Satomi Chiken and Atsushi Nambu. 'Mechanism of Deep Brain Stimulation: Inhibition, Excitation, or Disruption?' In: *The Neuroscientist* 22 (2016), pp. 313–322. DOI: 10.1177/1073858415581986.

[12] Jean-Michel Deniau et al. 'Deep brain stimulation mechanisms: beyond the concept of local functional inhibition'. In: *European Journal of Neuroscience* 32 (2010), pp. 1080–1091. DOI: 10.1111/j.1460-9568.2010.07413.x.

[13] Joseph J. Fins et al. 'Ethical guidance for the management of conflicts of interest for researchers, engineers and clinicians engaged in the development of therapeutic deep brain stimulation'. In: *Journal of Neural Engineering* 8.3 (2011). DOI: 10.1088/1741-2560/8/3/033001.

[14] Ulrike Gimsa et al. 'Matching geometry and stimulation parameters of electrodes for deep brain stimulation experiments—Numerical considerations'. In: *Journal of Neuroscience Methods* 150.2 (2006), pp. 212–227. DOI: 10.1016/j.jneumeth.2005.06.013.

[15] Espen Hagen et al. 'Multimodal Modeling of Neural Network Activity: Computing LFP, ECoG, EEG, and MEG Signals With LFPy 2.0'. In: *Frontiers in Neuroinformatics* 12 (2018). DOI: 10.3389/fninf.2018.00092.

[16] Todd M. Herrington, Jennifer J. Cheng and Emad N. Eskandar. 'Mechanisms of deep brain stimulation'. In: *J Neurophysiol* 115 (2016), pp. 19–38. DOI: 10.1152/jn.00281.2015.

[17] M.L. Hines and N.T. Carnevale. 'The NEURON Simulation Environment'. In: *Neural Computation* 9.6 (1997), pp. 1179–1209. DOI: 10.1162/neco.1997.9.6.1179.

[18] Mark H. Histed, Vincent Bonin and R. Clay Reid. 'Direct Activation of Sparse, Distributed Populations of Cortical Neurons by Electrical Microstimulation'. In: *Neuron* 63 (2009), pp. 508–522. DOI: 10.1016/j.neuron.2009.07.016.

[19] Yaoshiang Ho and Samuel Wookey. 'The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling'. In: *IEEE access* 8 (2020), pp. 4806–4813.

[20] A. L. Hodgkin and A. F. Huxley. 'A quantitative description of membrane current and its application to conduction and excitation in nerve'. In: *The Journal of physiology* 117.4 (1952), pp. 500–544. DOI: 10.1113/jphysiol.1952.sp004764.

[21] Erwin B. Montgomery Jr and John T. Gale. 'Mechanisms of action of deep brain stimulation (DBS)'. In: *Neuroscience and Biobehavioral Reviews* 32 (2008), pp. 388–407. DOI: 10.1016/j.neubiorev.2007.06.003.

[22] James Kew and Ceri Davies. *Ion Channels: From Structure to Function*. Oxford University Press, 2009. DOI: 10.1093/acprof:oso/9780199296750.001.0001.

[23] Diederik P. Kingma and Jimmy Ba. 'Adam: A Method for Stochastic Optimization'. In: *arXiv.org* (2017).

[24] Christof Koch. *Biophysics of computation: information processing in single neurons*. Oxford University Press, 1999.

[25] Sydnei Lewis et al. 'Pilot Study to Investigate the Use of In-Clinic Sensing to Identify Optimal Stimulation Parameters for Deep Brain Stimulation Therapy in Parkinson's Disease'. In: (In press). DOI: 10.1016/j.neurom.2023.01.006.

[26] Henrik Lindén et al. 'LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons'. In: *Frontiers in neuroinformatics* 7 (2014), pp. 41–41. DOI: 10.3389/fninf.2013.00041.

[27] Andres M. Lozano et al. 'Deep brain stimulation: current challenges and future directions'. In: *Nature Reviews Neurology* 15 (2019), pp. 148–160. DOI: 10.1038/s41582-018-0128-2.

[28] Svjetlana Miocinovic et al. 'Computational Analysis of Subthalamic Nucleus and Lenticular Fasciculus Activation During Therapeutic Deep Brain Stimulation'. In: *Journal of Neurophysiology* 96.3 (2006), pp. 1569–1580. DOI: 10.1152/jn.00305.2006.

[29] Svjetlana Miocinovic et al. 'History, Applications, and Mechanisms of Deep Brain Stimulation'. In: *JAMA Neurol.* 70 (2013), pp. 163–171. DOI: 10.1001/2013.jamaneurol.45.

[30] Paul L. Nunez and Ramesh Srinivasan. *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, 2006. Chap. 3, p. 104.

[31] Peter Rabins et al. 'Scientific and Ethical Issues Related to Deep Brain Stimulation for Disorders of Mood, Behavior, and Though.' In: *Arch Gen Psychiatry* 9 (2009), pp. 931–937. DOI: 10.1001/archgenpsychiatry.2009.113.

[32] Setareh Rafatirad. *Machine learning for computer scientists and data analysts: from an applied perspective*. Springer, 2022.

[33] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature (London)* 323.6088 (1986), pp. 533–536.

[34] J. Siegfried and B. Lippitz. 'Bilateral chronic electrostimulation of ventroposterolateral pallidum: a new therapeutic approach for alleviating all parkinsonian symptoms'. In: *Neurosurgery* 35 (1994), pp. 1126–1129. DOI: 10.1227/00006123-199412000-00016.

[35] David Sterratt et al. *Principles of computational modelling in neuroscience*. Cambridge University Press, 2011. Chap. 11, pp. 315–316.

[36] David Sterratt et al. *Principles of computational modelling in neuroscience*. Cambridge University Press, 2011. Chap. 3, p. 47.

[37] David Sterratt et al. *Principles of computational modelling in neuroscience*. Cambridge University Press, 2011. Chap. 2, pp. 37–41.

[38] Christoph Teufel and Paul C. Fletcher. 'The promises and pitfalls of applying computational models to neurological and psychiatric disorders'. In: *Brain (London, England : 1878)* 139.10 (2016), pp. 2600–2608. DOI: 10.1093/brain/aww209.

[39] Matteo Vissani, Ioannis U. Isaias and Alberto Mazzoni. 'Deep brain stimulation: a review of the open neural engineering challenges'. In: *Journal of Neural Engineering* 17.5 (2020). DOI: 10.1088/1741-2552/abb581.

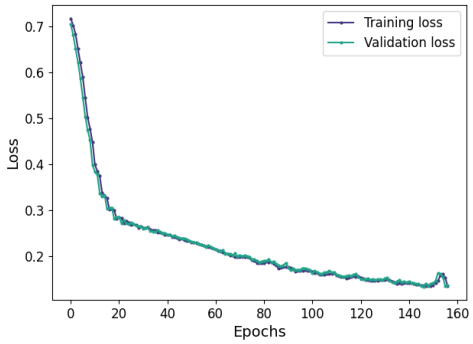
[40] Mai-Anh T. Vu et al. 'A Shared Vision for Machine Learning in Neuroscience'. In: 38.7 (2018), pp. 1601–1607. DOI: 10.1523/JNEUROSCI.0508-17.2018.

Chapter 7

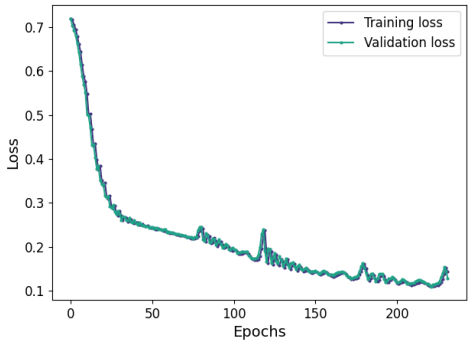
Appendix

7.1 Loss Plots

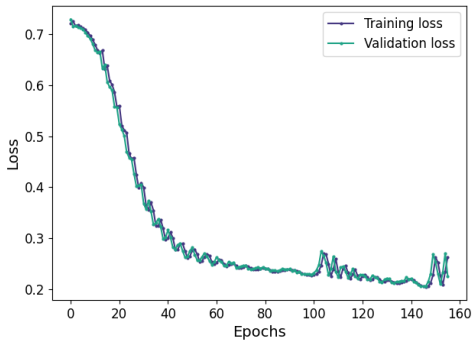
Straight Axon Simulations



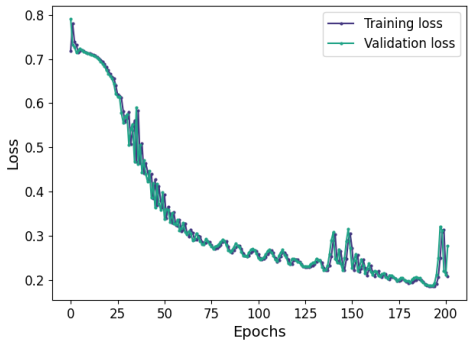
(a) Model is trained using df_5 with straight axon simulations as its dataset.



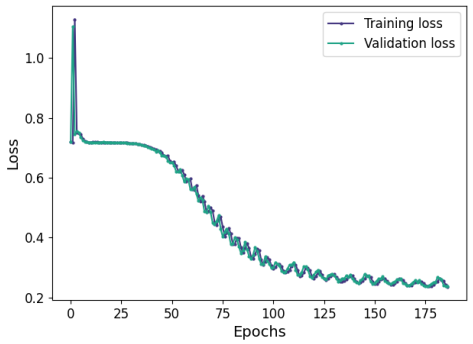
(b) Model is trained using df_{10} with straight axon simulations as its dataset.



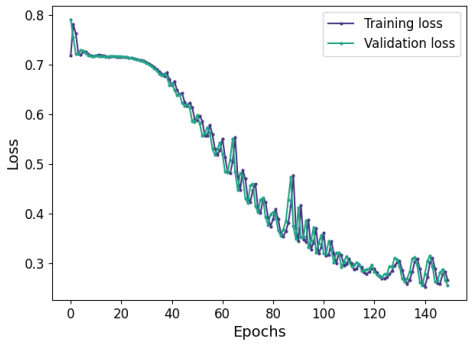
(c) Model is trained using df_{20} with straight axon simulations as its dataset.



(d) Model is trained using df_{40} with straight axon simulations as its dataset.

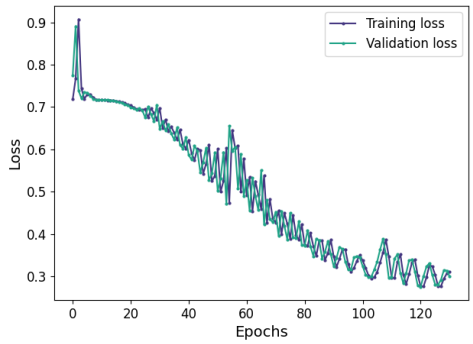


(e) Model is trained using df_{60} with straight axon simulations as its dataset.

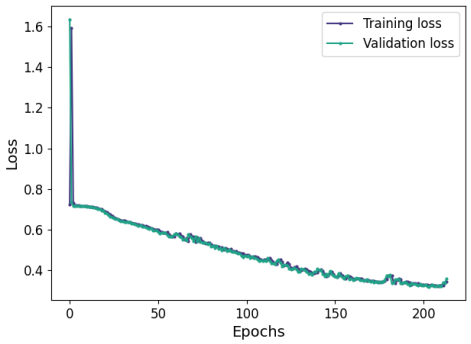


(f) Model is trained using df_{80} with straight axon simulations as its dataset.

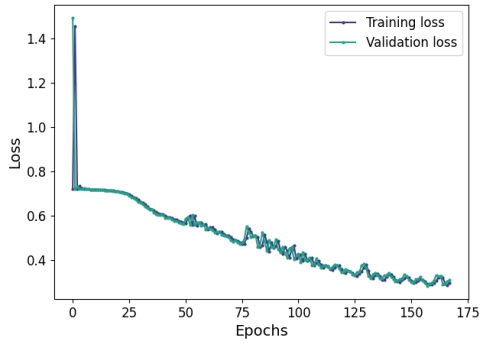
Figure 7.1: Plots showing the calculated training loss and validation loss for each epoch throughout the training process, each plot showing the loss with different datasets used to train the model. In subfigures a, b, c, d, e and f, the model is trained using df_5, df_{10}, df_{20}, df_{40}, df_{60} and df_{80} with straight axon simulations as its datasets, respectively.



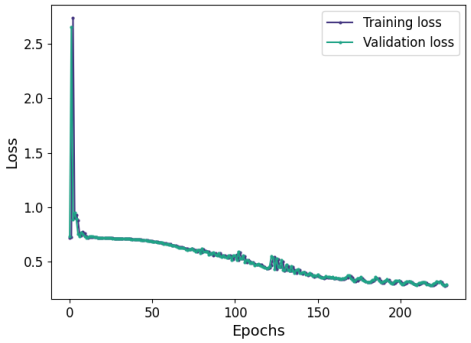
(a) Model is trained using df_{100} with straight axon simulations as its dataset.



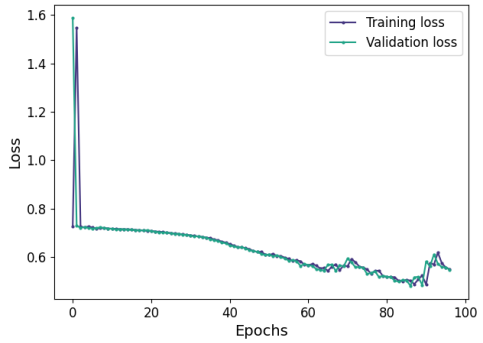
(b) Model is trained using df_{120} with straight axon simulations as its dataset.



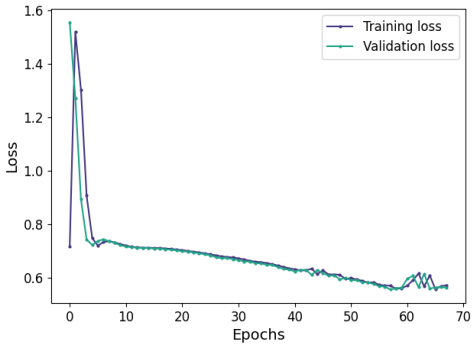
(c) Model is trained using df_{140} with straight axon simulations as its dataset.



(d) Model is trained using df_{160} with straight axon simulations as its dataset.



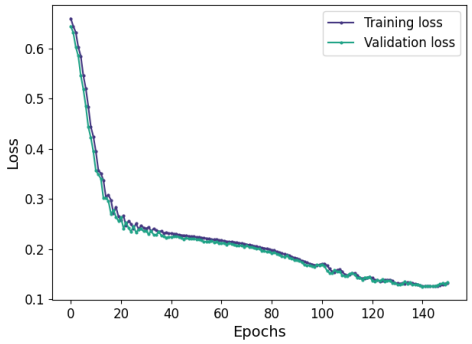
(e) Model is trained using df_{180} with straight axon simulations as its dataset.



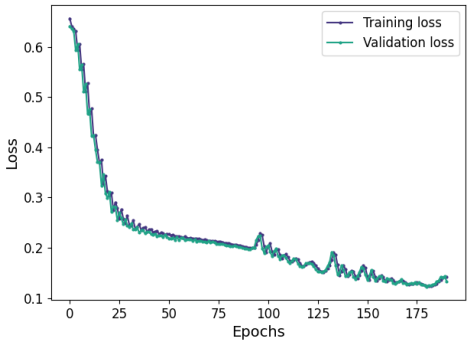
(f) Model is trained using df_{200} with straight axon simulations as its dataset.

Figure 7.2: Plots showing the calculated training loss and validation loss for each epoch throughout the training process, each plot showing the loss with different datasets used to train the model. In subfigures a, b, c, d, e and f, the model is trained using df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200} with straight axon simulations as its datasets, respectively.

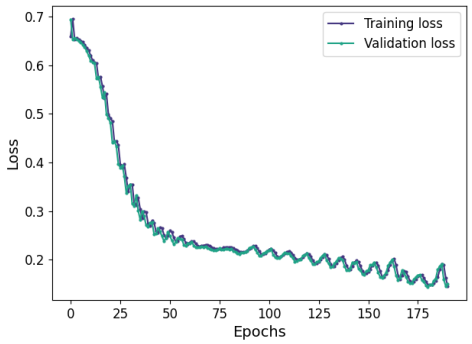
Curved Axon Simulations



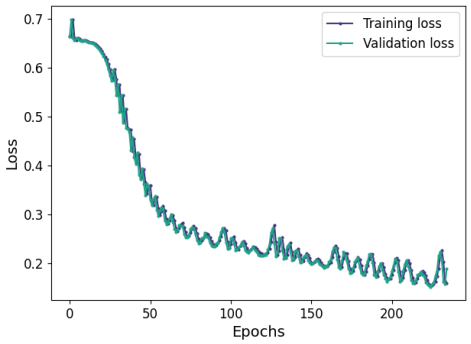
(a) Model is trained using df_5 with curved axon simulations as its dataset.



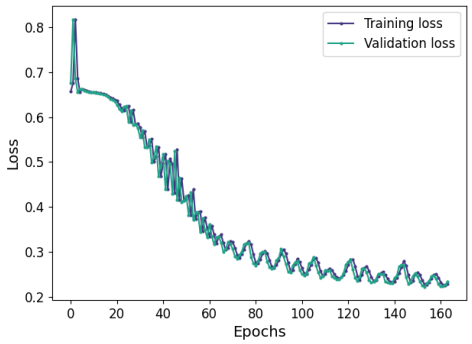
(b) Model is trained using df_{10} with curved axon simulations as its dataset.



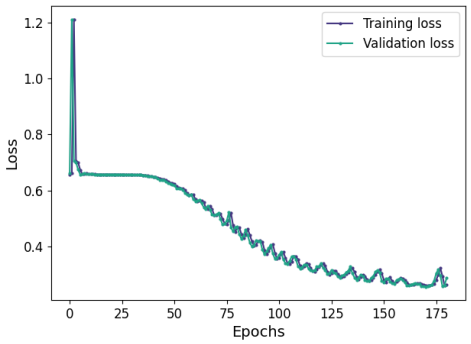
(c) Model is trained using df_{20} with curved axon simulations as its dataset.



(d) Model is trained using df_{40} with curved axon simulations as its dataset.

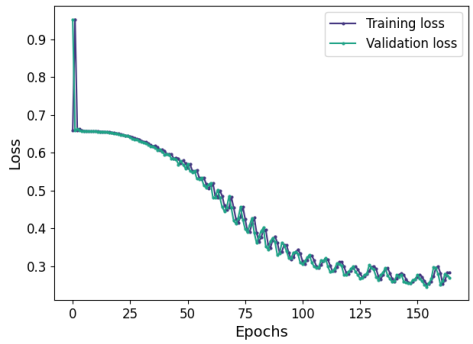


(e) Model is trained using df_{60} with curved axon simulations as its dataset.

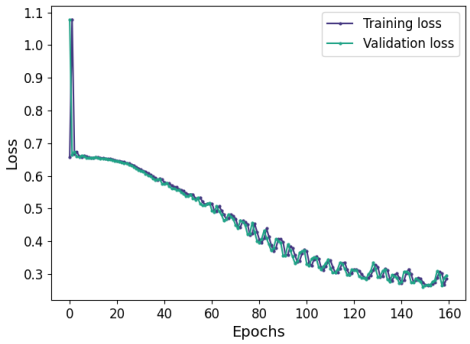


(f) Model is trained using df_{80} with curved axon simulations as its dataset.

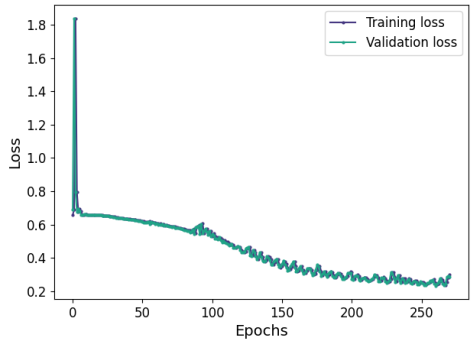
Figure 7.3: Plots showing the calculated training loss and validation loss for each epoch throughout the training process, each plot showing the loss with different datasets used to train the model. In subfigures a, b, c, d, e and f, the model is trained using df_5, df_{10}, df_{20}, df_{40}, df_{60} and df_{80} with curved axon simulations as its datasets, respectively.



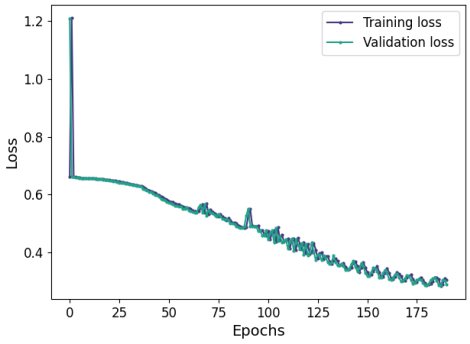
(a) Model is trained using df_{100} with curved axon simulations as its dataset.



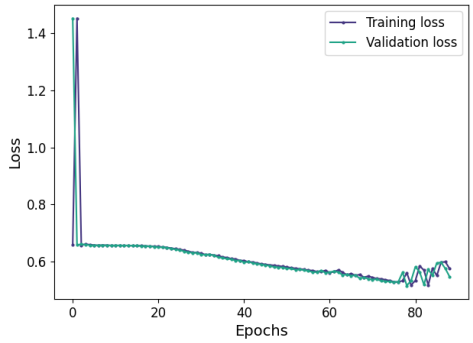
(b) Model is trained using df_{120} with curved axon simulations as its dataset.



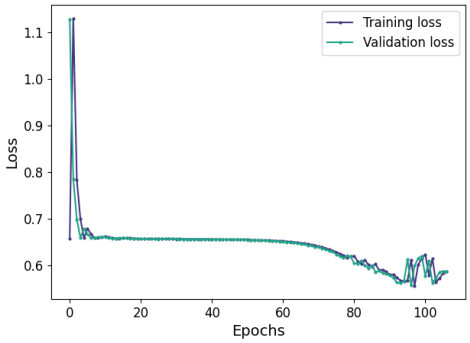
(c) Model is trained using df_{140} with curved axon simulations as its dataset.



(d) Model is trained using df_{160} with curved axon simulations as its dataset.



(e) Model is trained using df_{180} with curved axon simulations as its dataset.

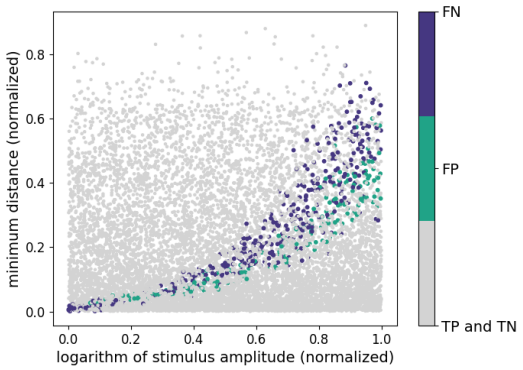


(f) Model is trained using df_{200} with curved axon simulations as its dataset.

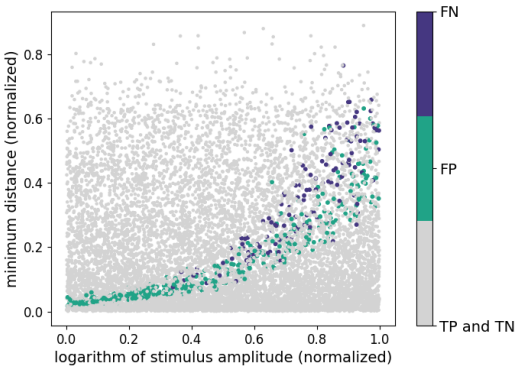
Figure 7.4: Plots showing the calculated training loss and validation loss for each epoch throughout the training process, each plot showing the loss with different curved axon simulation datasets used to train the model. In subfigures a, b, c, d, e and f, the model is trained using df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200} with curved axon simulations as its datasets, respectively.

7.2 Scatter Plots

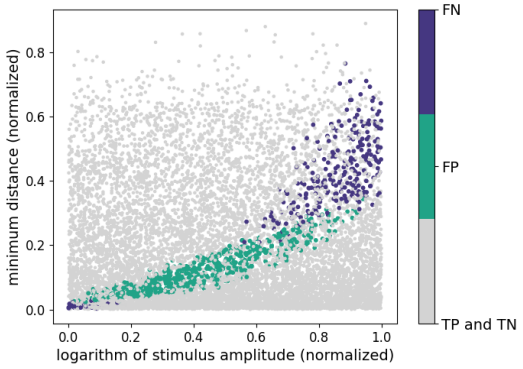
Straight Axon Simulations



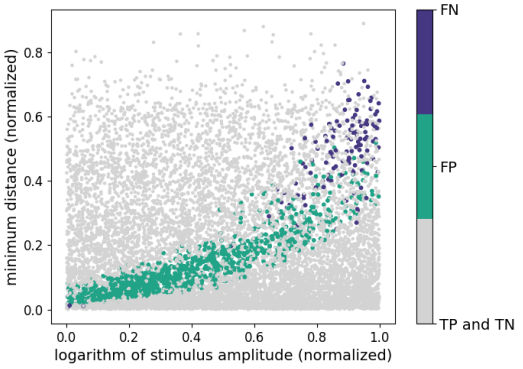
(a) The model is trained using df_5 with straight axon simulations as its dataset.



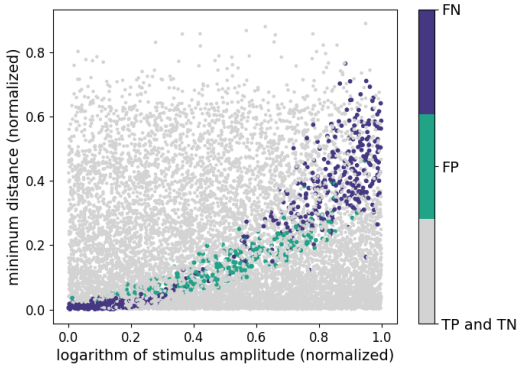
(b) The model is trained using df_{10} with straight axon simulations as its dataset.



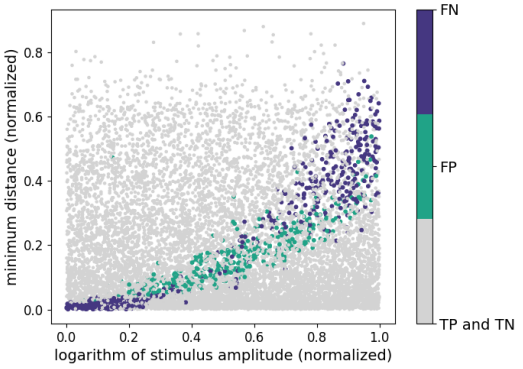
(c) The model is trained using df_{20} with straight axon simulations as its dataset.



(d) The model is trained using df_{40} with straight axon simulations as its dataset.

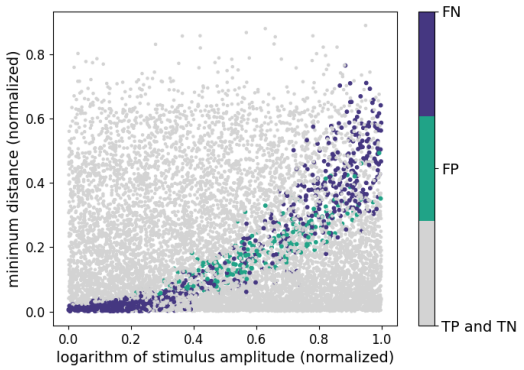


(e) The model is trained using df_{60} with straight axon simulations as its dataset.

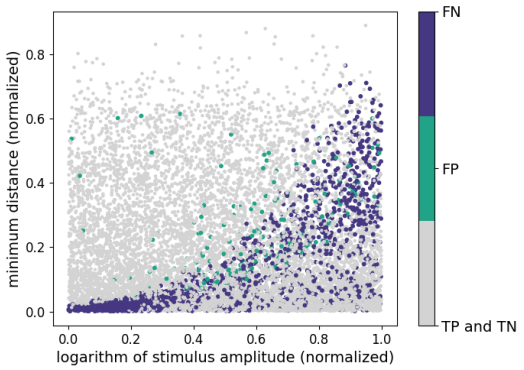


(f) The model is trained using df_{80} with straight axon simulations as its dataset.

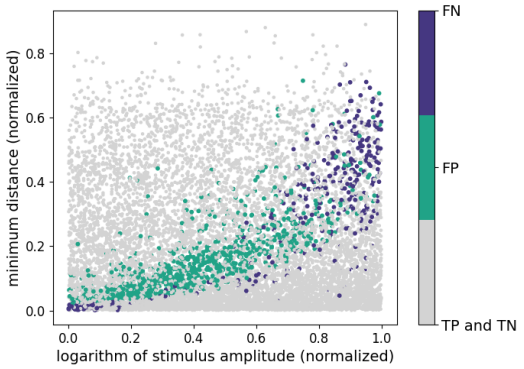
Figure 7.5: Scatter plots showing the relationship between the normalized minimum distance, normalized stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). Each dot represents a single straight axon simulation, with gray dots indicating the correct predictions of the model (TP and TN), teal dots indicating false positives (FP), and purple dots indicating false negatives (FN). In subfigures a, b, c, d, e and f, the model is trained using df_5, df_{10}, df_{20}, df_{40}, df_{60} and df_{80} with straight axon simulations as its datasets, respectively.



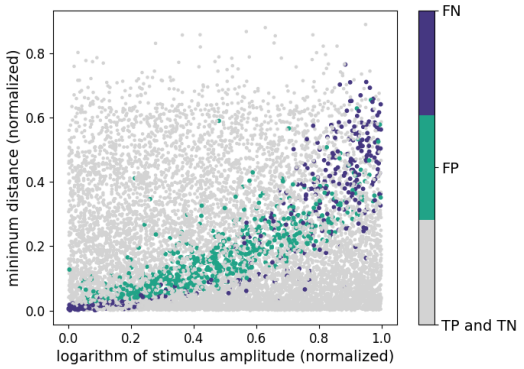
(a) The model is trained using df_{100} with straight axon simulations as its dataset.



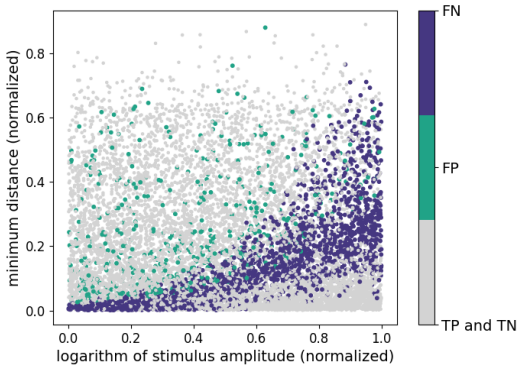
(b) The model is trained using df_{120} with straight axon simulations as its dataset.



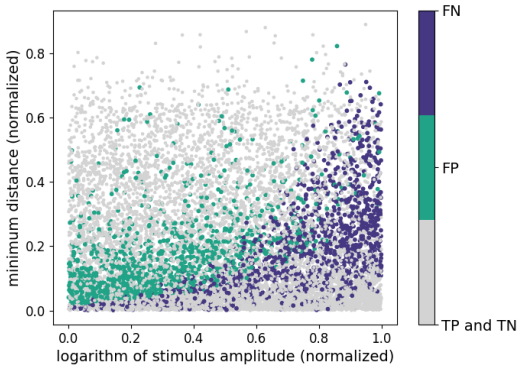
(c) The model is trained using df_{140} with straight axon simulations as its dataset.



(d) The model is trained using df_{160} with straight axon simulations as its dataset.



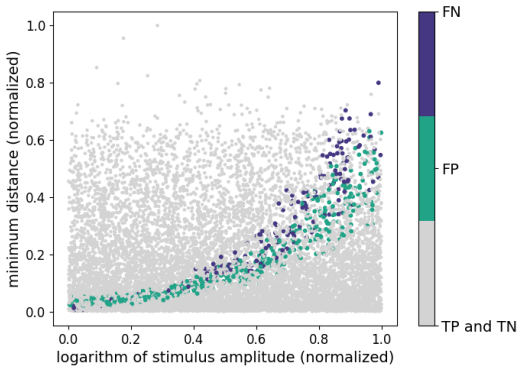
(e) The model is trained using df_{180} with straight axon simulations as its dataset.



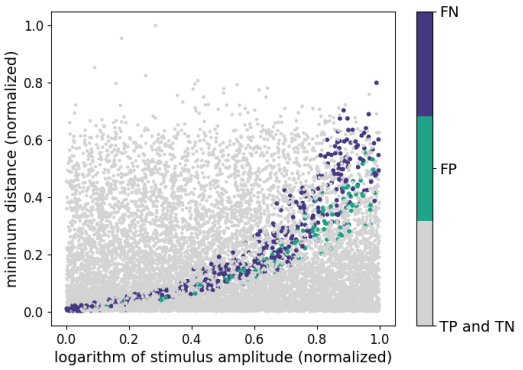
(f) The model is trained using df_{200} with straight axon simulations as its dataset.

Figure 7.6: Scatter plots showing the relationship between the normalized minimum distance, normalized stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). Each dot represents a single straight axon simulation, with gray dots indicating the correct predictions of the model (TP and TN), teal dots indicating false positives (FP), and purple dots indicating false negatives (FN). In subfigures a, b, c, d, e and f, the model is trained using df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200} with straight axon simulations as its datasets, respectively.

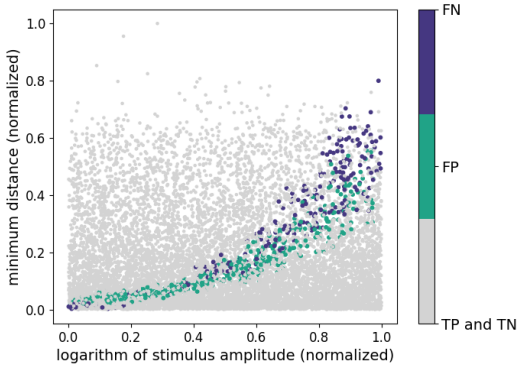
Curved Axon Simulations



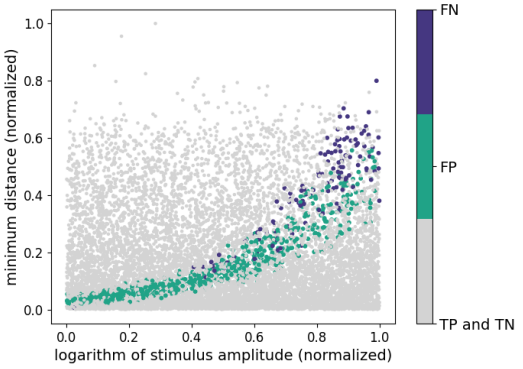
(a) The model is trained using df_5 with curved axon simulations as its dataset.



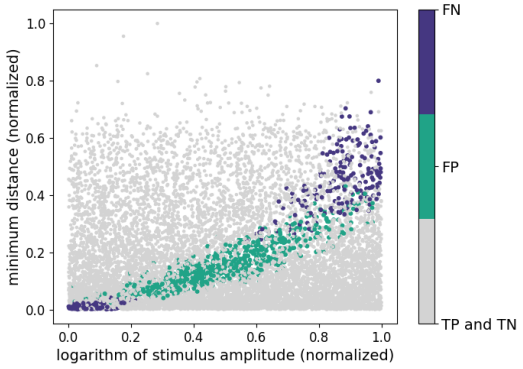
(b) The model is trained using df_{10} with curved axon simulations as its dataset.



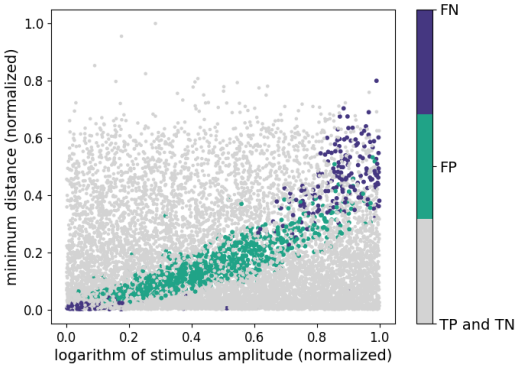
(c) The model is trained using df_{20} with curved axon simulations as its dataset.



(d) The model is trained using df_{40} with curved axon simulations as its dataset.

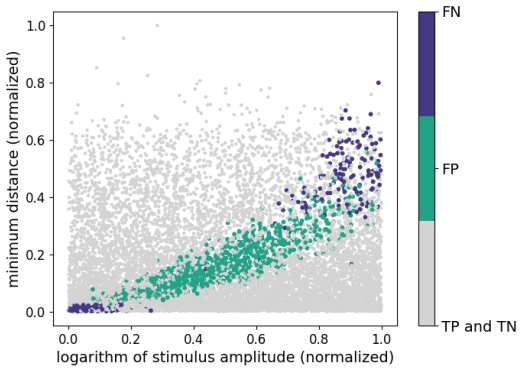


(e) The model is trained using df_{60} with curved axon simulations as its dataset.

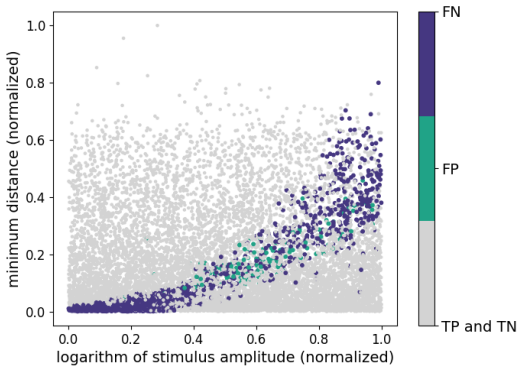


(f) The model is trained using df_{80} with curved axon simulations as its dataset.

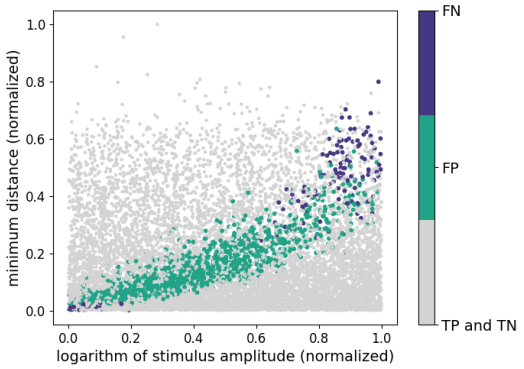
Figure 7.7: Scatter plots showing the relationship between the normalized minimum distance, normalized stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). Each dot represents a single curved axon simulation, with gray dots indicating the correct predictions of the model (TP and TN), teal dots indicating false positives (FP), and purple dots indicating false negatives (FN). In subfigures a, b, c, d, e and f, the model is trained using df_5, df_{10}, df_{20}, df_{40}, df_{60} and df_{80} with curved axon simulations as its datasets, respectively.



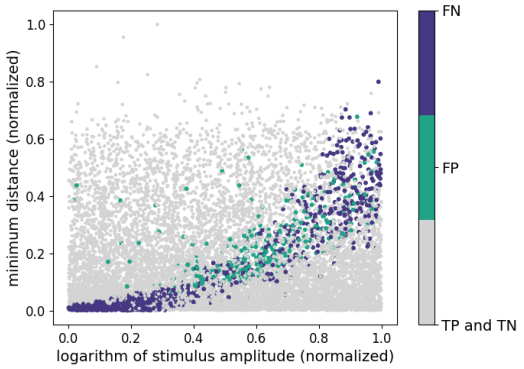
(a) The model is trained using df_{100} with curved axon simulations as its dataset.



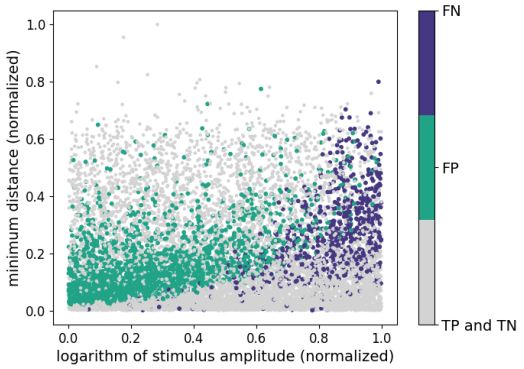
(b) The model is trained using df_{120} with curved axon simulations as its dataset.



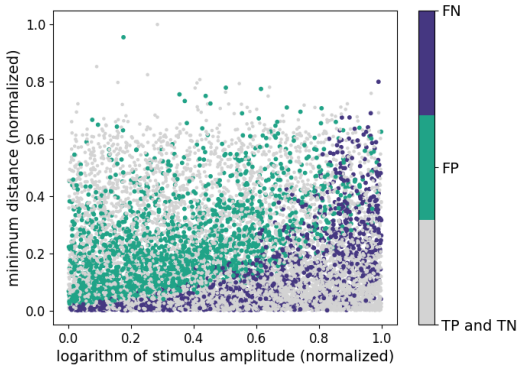
(c) The model is trained using df_{140} with curved axon simulations as its dataset.



(d) The model is trained using df_{160} with curved axon simulations as its dataset.



(e) The model is trained using df_{180} with curved axon simulations as its dataset.



(f) The model is trained using df_{200} with curved axon simulations as its dataset.

Figure 7.8: Scatter plots showing the relationship between the normalized minimum distance, normalized stimulus amplitude, false negatives (FN), false positives (FP), true positives (TP) and true negatives (TN). Each dot represents a single curved axon simulation, with gray dots indicating the correct predictions of the model (TP and TN), teal dots indicating false positives (FP), and purple dots indicating false negatives (FN). In subfigures a, b, c, d, e and f, the model is trained using df_{100}, df_{120}, df_{140}, df_{160}, df_{180} and df_{200} with curved axon simulations as its datasets, respectively.