Eline Stenwig

Neural Response Analysis of Head Direction Cells

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Ilangko Balasingham
June 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Eline Stenwig

Neural Response Analysis of Head Direction Cells

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Ilangko Balasingham
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Abstract

Theoretical analysis and computational modeling of neurons are tools used to get a better understanding of the nervous system — an understanding which hopefully shortens the path to finding treatments, cures, and ways to prevent problems related to nervous disorders.

This study aims to understand and explain how neural cells function and communicate by studying head direction (HD) cells. These cells are cells that assist with spatial orientation and are selected due to the availability of real data sets, collected from freely moving mice. From a selection of HD cells, specific criteria were placed on the data set, extracting parts for further inspection. A search for a common model capable of describing the response of an HD cell was conducted, using patterns describing the discharge of said HD cell. In addition to trying to find a model using already existing spiking neuron models, it was tested to see if it is possible to classify the angle of the head by looking at the same response recordings as for the model. The last task was explored using a machine learning approach with long short-term memory networks, commonly used for classifying sequences.

The Izhikevich spiking neuron model was chosen as the model of investigation because of the combination of being biologically plausible as well as computationally simple. Additionally, this model requires little to no alterations to the model or data set prior to the analysis. It was possible to approximate the model to parts of the neural responses using the model, but a general model describing the complete response of an HD cell was not found.

When using long short-term memory networks to classify spiking sequences to the direction of the head according to the data set, some results show promise. The response from a single HD cell alone does not contain enough information to use the method described in this report, while the neural net can distinguish between some directions when combining the response from several HD cells; about $10^\circ \pm 30^\circ$.

Sammendrag

Teoretisk analyse og modellering av nevroner er verktøy brukt til å gi en bedre forståelse av nervesystemet — en forståelse som forhåpentligvis forkorter veien til å finne behandlingsmetoder og kurer, samt måter å forhindre problemer relatert til sykdommer i nervesystemet.

Ved å studere hoderetningsceller ønsker denne studien å gi forståelse, samt beskrive hvordan nerveceller fungerer og kommuniserer. Hoderetningsceller er celler som bistår med romlig orientering. Disse er valgt grunnet tilgjengelighet av ekte data samlet fra frittgående mus. På et utvalg av hoderetningscellene ble bestemte kriterier benyttet for å velge ut data for videre bearbeidelse. Det ble gjort forsøk på å identifisere en generell modell som kan beskrive utladningene av hoderetningscellene. I tillegg ble det samme datasettet testet for å se om det er mulig å klassifisere hoderetningen. Det siste problemet ble undersøkt ved hjelp av en maskinlæringstilnærming med 'long short-term memory' nettverk, ofte brukt til klassifisering av sekvenser.

Izhikevich nevronmodell ble valgt som utgangspunkt for videre undersøkelser ettersom den er biologisk plausibel og kan beregnes enkelt. I tillegg kreves det få tilpasninger av både modell og datasett før videre analyse. Det var mulig å modellere deler av nevronresponsen, men det var ikke mulig å lage en generell modell som gir en komplett beskrivelse av hoderetningscellene.

Klassifisering av fyringssekvensene til hoderetningene ved å bruke 'long short-term memory' nettverk på datasettet gir noen lovende resultater. Responsen fra en hoderetningscelle inneholder alene ikke nok informasjon til å benytte metoden beskrevet i denne oppgaven, men ved å kombinere responsen fra flere hoderetningsceller kan man bestemme hoderetningen med noe avvik, circa $10° \pm 30°$.

Preface

This Master's thesis is submitted in partial fulfillment of the five year Master's degree programme Electronics Systems Design and Innovation.

I would like to thank my supervisor Ilangko Balasingham and co-supervisor Mladen Veletić for all help. I would also like to thank Finn, Katzy, Noha and Hemin for help with understanding the large, confusing world of machine learning. For help with motivation and for having someone to complain to when things did not go my way, thank you Maja and Kristina and the people hanging out at A491. An additional thanks to Nabiul Islam for the code that highly simplifies the work with the data set used.

And to Stig: Thank you for everything.

Contents

Abstract i

Sammendrag iii

Preface v

Table of Contents ix

List of Tables xi

List of Figures xiii

Abbreviations xv

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 2
- 1.3 Previous Work . . . 2
- 1.4 Method . . . 3
- 1.5 Organization of This Report . . . 3

2 Background 5
- 2.1 Neural Prostheses . . . 5
- 2.2 Neurons . . . 6
- 2.3 Head Direction Cells . . . 8
- 2.4 Recording of Neural Responses . . . 9
- 2.5 Neural Coding . . . 11
- 2.6 Spiking Neuron Models . . . 13
- 2.7 Spike Train Metrics . . . 14
- 2.8 Machine Learning . . . 14
- 2.9 Artificial Neural Networks . . . 17
- 2.10 Long Short-Term Memory . . . 21

3 Theory 23
- 3.1 Complete Data Set . . . 23
- 3.2 MATLAB Code Extracting HD Cells . . . 24

3.3 The Victor-Purpura Distance 25
3.4 The van Rossum Distance 26
3.5 Izhikevich Neuron Model 27
3.6 MATLAB Deep Learning Toolbox 28

4 Methods 31
4.1 Notation 31
4.2 Delimitations 31
4.3 Initial Analysis 31
4.4 Time Between Spikes 33
4.5 Comparison of Sessions 33
4.6 Finding a Model 34
4.7 Machine Learning 36
4.8 One HD Cell 39
4.9 Multiple HD Cells 40
4.10 Combining Models and Neural Nets 40

5 Results 43
5.1 Delimitations 43
5.2 Initial Analysis 43
5.3 Time Between Spikes 50
5.4 Comparison of Sessions 51
5.5 Finding a Model 53
5.6 Machine Learning 55
5.7 One HD Cell 57
5.8 Multiple HD Cells 57
5.9 Combining Models and Neural Nets 60

6 Discussion 65
6.1 Initial Analysis 65
6.2 Time Between Spikes 67
6.3 Comparison of Sessions 68
6.4 Finding a Model 69
6.5 Machine Learning 70
6.6 One HD Cell 70
6.7 Multiple HD Cells 71
6.8 Combining Models and Neural Nets 71

7 Conclusion 75

8 Future Work 77

Bibliography 79

A Hardware 87

B MATLAB Deep Learning Toolbox **89**
B.1 Layers . . . 89
B.2 Hyperparameters . . . 90

C Initial Analysis **91**
C.1 Head Directions Represented . . . 91
C.2 Sessions per Angle Bin Width . . . 93
C.3 Session Lengths . . . 94

D Code **95**
D.1 MATLAB Code Extracting HD Cells . . . 95
D.2 Overview Functions . . . 100

List of Tables

3.1 Head direction cells inclusion criteria. 25

4.1 Parameter values for the Izhikevich model. 36

5.1 Length of wake phase. 43
5.2 Number of head direction cells extracted from data set. 44
5.3 Directional firing range and peak firing rate found by visual inspection of tuning curves. 45
5.4 Number of extracted sessions for day 120806. 48
5.5 Angle bin width. 49
5.6 Minimum number of spikes. 50
5.7 Minimum session length. 50
5.8 Mean and deviation for different angle bins. 50
5.9 Number of extracted sessions. 52
5.10 Comparison of sessions. 53
5.11 Optimal model parameter values. 54
5.12 Victor-Purpura distance for different optimal model parameter values. 55
5.13 Hyperparameter values and number of hidden units. 55
5.14 Hyperparameter values and alterations. 60
5.15 The number of HD cells per angle bin with at least one model. . . 61

C.1 Number of extracted sessions for 120807. 93
C.2 Number of extracted sessions for 120808. 93
C.3 Number of extracted sessions for 120809. 93

List of Figures

2.1 The three main components of a neuron. 6
2.2 Cell membrane components and construction. 7
2.3 Firing of an action potential [1]. 8
2.4 Tuning curve of an HD cell with properties. 9
2.5 Main components of a probe. 10
2.6 Artificial neural network with one hidden layer with four nodes. . . 17
2.7 Overview over a perceptron. Weights are applied to the input before the weighted sum is passed through a function that produces the output. 18
2.8 Common types of activation functions. 19
2.9 Feedforward network. 20
2.10 An unfolded basic recurrent neural network. 20
2.11 Long short-term network architecture. 21

3.1 Recording setup using camera and probes to find head direction and spike times. 24
3.2 Tuning curves from code in Appendix D.1. 26

4.1 Ideal model for describing response of head direction cells. 35
4.2 Architecture of a simple LSTM network. 38

5.1 Tuning curves of HD cell 120806.3.8. 44
5.2 Imperfect tuning curves. 45
5.3 Time spent with head in each direction and PFDs of HD cells for 120806. 46
5.4 Comparison of tuning curves. 47
5.5 Tuning curve of head direction cell 120806.3.8 with angle bin width of 10°. 48
5.6 Length of extracted sessions for day 120806. 49
5.7 Time between spikes for HD cell 120806.3.8 for three different angle bin widths. 51
5.8 Comparisons of sessions and models. 54
5.9 Comparison of recordings from session $S2$ and model from session $S1$ and $S3$. 55
5.10 Number of sessions per angle bin for 120806. 56

5.11 Training accuracy. 57
5.12 Confusion matrix; one HD cell. 58
5.13 Confusion matrices; one HD cell, with sifted data. 58
5.14 Confusion matrices; multiple HD cells. 59
5.15 New input in neural net NN_1. Input based on sessions with more than five spikes. 62
5.16 New input in neural net NN_1. Input based on sessions with a minimum length of 250 ms. 63

C.1 Time spent with head in each direction and PFDs of HD cells for 120807. 91
C.2 Time spent with head in each direction and PFDs of HD cells for 120808. 92
C.3 Time spent with head in each direction and PFDs of HD cells for 120809. 92
C.4 Length of extracted sessions for angle bin width of 10°. The bar above 300 ms represents the number of sessions longer than 300 ms. 94

Abbreviations

ANN	=	Artificial Neural Network
AP	=	Action Potential
BMI	=	Brain-Machine Interface
HD	=	Head Direction
HDCEC	=	HD Cell Extracting Code
LSTM	=	Long Short-Term Memory
ML	=	Machine Learning
NN	=	Neural Network
NP	=	Neural Prostheses
PFD	=	Preferred Firing Direction
PoS	=	Postsubiculum
RNN	=	Recurrent Neural Network
SD	=	Standard Deviation
SGD	=	Stochastic Gradient Descent

1 | Introduction

1.1 Motivation

Neurological disorders are diseases of the nervous system, which includes the brain, spinal cord, and nerves [2, 3]. The term covers disorders such as Alzheimer's and Parkinson's disease, epilepsy, and injuries due to head trauma [4]. Hundreds of millions of people worldwide are affected by neurological disorders throughout their lives, which pose a substantial burden of disease globally [4]. Some of these disorders have currently no cure, and in many cases, available treatments are also poor. In cases where parts of the nervous system are unable to communicate correctly, artificial stimulation of the nerves mimicking the natural behavior of the nerves can help to restore functions such as bladder control [5], limb movement [6], and memory [7].

Neural prostheses (NPs) are external or implanted devices with the goal of restoring neurological functions as the ones mentioned above. For NPs to be able to replace neurons in the brain that no longer function properly, models able to explain the input and output relationship of neurons are essential. The knowledge about this relationship can then be exploited by Brain-Machine Interfaces (BMIs), which allows the brain and a machine to exchange information [8]. The success of the NPs and the BMIs depends on the understanding of the principles of neural signal processing, wiring, and communication.

As well as using the knowledge of how cells interact to create machines able to communicate with neurons, this knowledge can also be used to investigate how diseases such as Parkinson's disease progress. Which in turn can lead to better treatments and potential cures.

Neurons can be divided into different types, depending on their function, which can be everything from responding to various stimuli to controlling muscles. An example is head direction (HD) cells, which are cells that respond to the direction of the head, and assist with spatial orientation. There are several reasons why HD cells are interesting to study. Due to the one-dimensional HD sense, the HD cells offers a simpler analysis than cells responding to several types of stimuli at the same time, which can be used to develop models for more complicated cells later. Head direction cells are found in multiple areas of the brain [9, 10, 11, 12, 13, 14], and can provide valuable knowledge that can be used to understand diseases such as Alzheimer's [15]. Head direction cells are also chosen as cells of interest due to the availability of real data sets [16].

1.2 Objectives

The scope of this project is investigating the possibilities of finding a model using existing spiking neuron models and artificial neural network models to describe the firing patterns that can be classified to a set of directions. The main objectives are as follows:

- To study the possibility of adapting an already existing spiking neuron model to the response of HD cells.
- To study if it is possible to classify spike sequences from HD cells to corresponding angles with the use of machine learning, and in particular long short-term memory networks.

1.3 Previous Work

There are done several studies on HD cells, ranging from their location and topography [17, 12], to specific aspects related to the activity such as relation to behaviour [18], and computational [19, 20] and sensory properties [21].

There exist multiple models able to describe the neural response. Some of them are quite general, adaptable to several types of cortical neurons [22, 23, 24], while others are more targeted, related to more specific types of neurons such as place and grid cells [25], and retinal cells [26].

Multiple studies are done on the modeling of neurons in the brain related to spatial orientation. Many of these studies use machine learning methods such as recurrent neural networks [27, 28, 29] and reinforcement learning [30, 29, 31, 32] to look at either grid cells, place cells or a combination, sometimes including head direction cells. None of the studies found are directly comparable to the work done in this study, as they all use other approaches towards other objectives.

A pre-master project was written on the same topic, looking at the possibilities of finding a model describing the discharging of HD cells. First, common patterns describing the discharge of each HD cell was attempted to be extracted, before the possibilities of finding a common model were explored. Due to limitations in the results, neither common patterns nor models were determined to be achievable in the scope of the project.

1.4 Method

One part of this work is to continue the investigation from the pre-master project described in Section 1.3. This is done by conducting a more thorough analysis of the available data set, before improving, adapting, and further develop the methods used to investigate possible models able to describe the neural response of the cells.

The second part of this project is to investigate if machine learning can be used as a tool for finding a model describing the spiking patterns of neurons. Here, the long short-term memory (LSTM) network is used with the adam optimizer in MATLAB.

1.5 Organization of This Report

This report is divided into seven chapters. The chapters are organized as follows: **Chapter 2** introduces some background information. If the reader already has adequate knowledge on the topics covered in the sections of this chapter, either sections or the whole chapter may be skipped. The first part covers some general information on neurons, while the second part contains information on the main topics of machine learning relevant for this project.

Chapter 3 contains specific theoretical background necessary for an understanding of the topics covered in this paper. The methods considered for use as well as the implemented methods are given in **Chapter 4**. There is an initial analysis of the data set before a part on finding a model using already existing spiking neuron models. The last part investigates how machine learning can be used as a tool to find a way to describe the spiking of neurons.

The results obtained are given in **Chapter 5**, and are discussed in **Chapter 6**. The chapters on methods, results, and discussion of the results are divided into similar subsections. This is done for easier readability, as well as making it possible to read the method, the results, and the discussion of a specific topic separately, without having to read the whole text.

Chapter 7 includes both a conclusion of the report, as well as possible future work. The future work is both different paths to investigate, in addition to the possible next steps, should someone wish to continue using the methods outlined in this paper.

The **Appendix** includes some additional figures and tables and information about important parameters, as well as code excerpts used for preprocessing, with an explanation for some of the functions used.

2 | Background

2.1 Neural Prostheses

Neural prostheses are devices, either external or internally implanted, using electrical stimulation on nerves to restore nerve function [33]. Examples include devices with the goal of restoring senses, such as cochlear [34] and retinal implants [35], devices that can substitute a loss in motor modality such as controlling a hand prosthesis[36] and regaining functional upper-limb movement after paralysis [37, 38], and implants helping with cognitive modality i.e. hippocampal prosthesis restoring memory [7].

Loss of normal nerve function can be a result of multiple traumatic injuries, and the prognosis after such injuries can be quite poor [39]. Initially, NPs were used in connection with severe disabilities such as amyotrophic lateral sclerosis (ALS), spinal cord injuries and stroke, but have later been introduced for other types of disabilities as well, such as amputees. In the future, NPs might also be used for other purposes, such as restoration of locomotion and speech [40].

In order to translate signals between neurons and a robotic prosthesis, brain-machine interfaces are used. Some of these BMIs can translate raw neural signals to commands that the prosthesis can interpret, while others translate signals from a device to something the brain understands. There are several bottlenecks associated with BMIs, ranging from biocompatibility issues for implantable devices, development of real-time algorithms, the possibility to provide the brain with sensory feedback from actuators, and to the knowledge of specific neurons, their functions and the communication used [40].

Since 1963, more than 40 000 devices have been implanted to restore functions like bladder control and respiration [33]. Early experimental demonstrations with real-time BMIs were conducted in the late 1990's [41] and has since become a rapidly growing field of research.

2.2 Neurons

Cells are the basic building blocks of living tissue, and the total number of cells in the human body is estimated to be $3.72{\cdot}10^{13}$ [42]. A neuron, or a nerve cell, is a type of a highly specialized cell that holds the ability to communicate rapidly over large distances. This is done electrically inside neurons and chemically between them through complex membrane junctions called synapses [43].

Nerve cells can be found throughout the body and are the fundamental units of the nervous system [44]. The nerve cells found in the brain are closely packed, and highly interconnected in intricate patterns. Per cubic millimeter there can be as many as 10^4 cortical neurons, with several kilometers of neuron “arms” [43]. The estimated number of cells in the human brain is between 10^{10} and 10^{11}, and each of these neurons can have as many as 10^3 to 10^5 connections to other neurons [45]. In addition to the neurons, the brain also consists of several other cells, which mainly help with structural stabilization and supply of energy. These cells, called glia cells, are important for the function of the brain but are not directly involved in the information processing [46].

Anatomy of Neurons

A typical nerve cell, depicted in Figure 2.1, can be divided into three main parts. The first part is the cell body, or the soma, which is structurally similar to other cells. The soma can be considered as the “central processing unit” of the cell. The protrusions of the soma are called dendrites, and the long fiber extending from the body is the axon [45]. Together they send and receive signals through synaptic connections to other neurons [43]. The most common type of communication is the axodendritic synapse, where the communication flows from an axon extending from one neuron to a dendrite on another neuron [45]. Other types of synapses also exist but are less common [47].

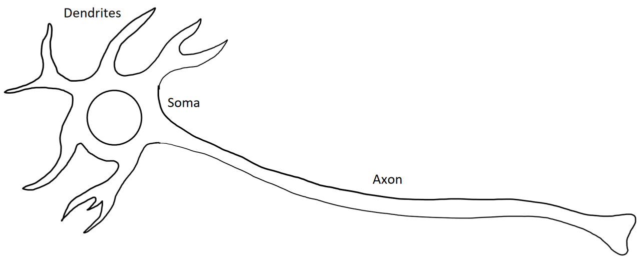
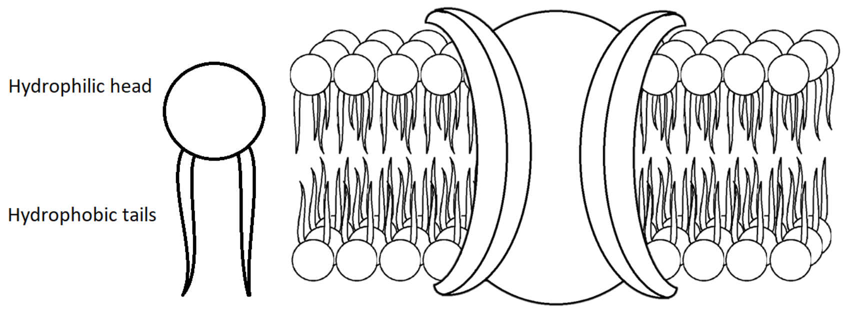


Figure 2.1: The three main components of a neuron.

One important component of cells is the cell membrane, which is depicted in Figure 2.2. The cell membrane consists of phospholipid molecules (Figure 2.2a), which have a hydrophilic head and hydrophobic tails. These lipids form a bilayer with the tails turned inwards (Figure 2.2b) [45]. An important property of the



(a) Phospholipid molecule. (b) Cell membrane consisting of a bilayer of phospholipid molecules and ionic channel.

Figure 2.2: Cell membrane components and construction.

cell membrane is the membrane voltage, which is the voltage difference exhibited across the membrane due to differences in ion concentration inside and outside the cell. Ion channels are found throughout the cell membrane, and allow specific ions to flow in and out of the cell [45].

Electrical Properties of Neurons

The membrane exhibits a voltage difference across the membrane,

$$V_m = \Phi_i - \Phi_o, \tag{2.1}$$

where V_m is the membrane voltage, Φ_i is the intracellular potential, and Φ_o is the extracellular potential. The resting voltage of the cell is typically between -70 and -90 mV and deviates from this value when exposed to stimuli. The stimuli can be excitatory, where the membrane voltage changes towards more positive values (called depolarising), or inhibitory, where the membrane voltage changes towards more negative values (called hyperpolarising). The voltage difference returns to its original state, the resting voltage, after stimulation [45].

If the excitatory stimulus is large enough for the transmembrane voltage difference to cross a threshold, typically around -60 mV, the neuron discharges or “fires” an action potential (AP) [45]. This firing is a short voltage pulse with a duration of 1-2 ms and an amplitude of about 100 mV [46]. The firing of an AP is depicted in Figure 2.3.

Strong stimulus holds the ability to cross the threshold alone, while weaker stimulus needs to be added together within a time window in order to cross the threshold [46]. The size and shape of APs are independent of the stimuli but dependent on the cell type [45].

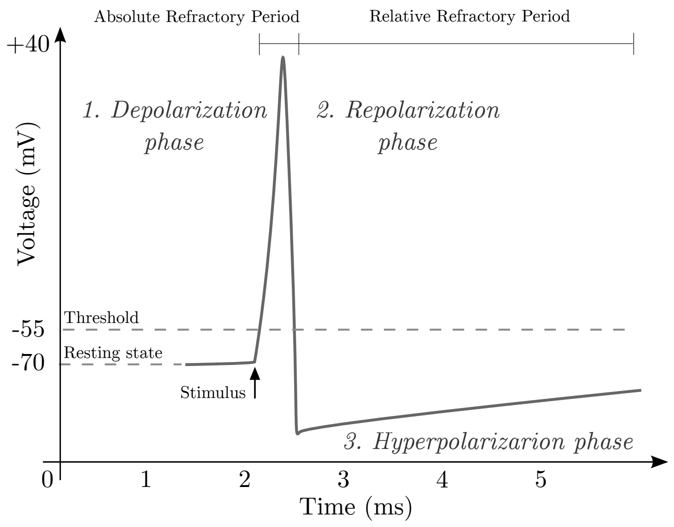


Figure 2.3: Firing of an action potential [1].

Additionally, the frequency of the spikes varies. The maximum firing rate depends on the rate at which the cell can be depolarized. The smallest time between two APs is the absolute refractory period of the neuron. This period is followed by the relative refractory period, where it is hard, but not impossible to excite the neuron [46].

2.3 Head Direction Cells

Head direction cells are neurons in the brain discharging with a firing rate dependent on the direction of the individual's head in the horizontal plane, relative to the environment. They have been identified in animals as mice [48], rats [11] and monkeys [49], but are thought to be present in all mammals [48].

Head direction cells were first identified in the postsubiculum (PoS) [9, 10], and have later been found in several areas of the brain, including retrosplenial cortex [11], the anterior dorsal thalamic nuclei[12], the lateral mammilary nuclei [13] and enthorinal cortex [14]. Many of them situated within the Papez circuit. It is not well understood why HD cells can be found in so many brain areas, but a hypothesis is that it is because the head direction is needed for several different brain functions [48].

There are three main properties of HD cells; the peak firing rate, the preferred firing direction (PFD) and the directional firing range [50]. A cell's peak firing rate is the maximum firing rate of the cell, which occurs when the head is in the PFD of the cell. The directional firing range is the angular range where the firing rate is above the cells baseline firing rate. When it comes to the PFDs, all directions are equally represented in the total population of HD cells [48].

Figure 2.4 is an example of a tuning curve of an HD cell. Here, the direction of the head is represented by the x-axis, while the y-axis represents the firing rate in spikes per time unit, often per second. The firing rate of the cell is zero, or almost zero for angles far away from the PFD, but increases quickly when the head direction is turning towards the cell's PFD. The directional firing range varies from cell to cell, from around 60° to 150°, but with an average of ∼90°. The peak firing rate also varies from cell to cell, usually from 5 to >120 spikes per second. It is not known if there is a specific reason behind these differences [48].

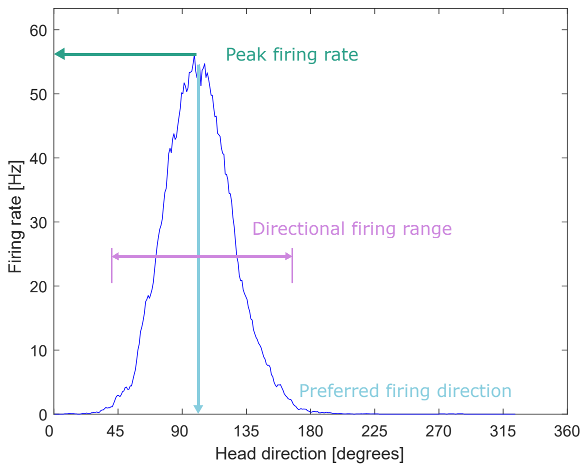


Figure 2.4: Tuning curve of an HD cell with properties.

The response of the HD cells is constant even when the head is not moving, which indicates that HD cells are independent of motion input [50]. An HD cell can be thought of as a sort of compass that only reacts when facing north, or nearly north. Instead of being affected by the Earth's geomagnetic field, it is dependent on landmarks and self-motion cues like the vestibular system and motor/proprioceptive information [48]. The HD cells are therefore considered a part of the allocentric system together with grid cells and place cells [51], which help with spatial orientation [48]. During sleep the tuning curves are similar to the tuning curve during wake periods [52].

2.4 Recording of Neural Responses

The neurons transmit information by electric signals. In order to understand this communication it is necessary to measure these signals. Fortunately, there are several methods to measure both intracellular and extracellular neural responses, which can be either electrical or optical. Intracellular recordings show both the firings of APs and the subthreshold membrane potentials. This is opposed to the extracellular recordings, whose signal is weaker and depends more on the geometrical contact with the neuron. Extracellular recordings record response from several cells at the same time, and are mostly used for detecting whether or not an AP has occurred. Intracellular recordings are usually done in vitro, while the extracellular ones are more often done in vivo since these recordings can be made by placing the electrode near the neurons without penetrating them [43].

By recording neural responses in vivo, it is possible to better understand basic brain functions. In trials where animals move around freely, electrodes inserted directly into a neuron pose a challenge of possibly damaging the neuron. Therefore, larger electrodes are used to provide better mechanical stability. However, a disadvantage of using these larger electrodes is that the activity of several neurons are recorded simultaneously, and more post-processing of recordings is needed in order to extract the responses of single neurons [53]. An example of a probe inserted into the brain is shown in Figure 2.5.

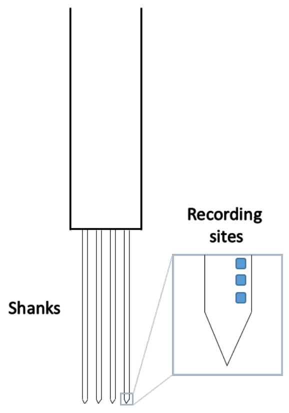


Figure 2.5: Main components of a probe.

The probes have one or more shanks, the one in the figure with four, each with several recording sites. The site layout varies depending on the application.

Spike Detection

The disentanglement of the response from a single neuron from the complex recordings can be done in three steps [53]:

1. **Spike detection:** Spike detection is done to extract the spiking times from the electrical recordings. The spike detection is usually done with high-pass filtering and thresholding and can be done through either hardware or software.

2. **Feature extraction:** For every detected spike, an array of quantitative parameters, called feature vector, is calculated. These features can be am-

plitudes and wave patterns, and the idea behind this is that spikes from the same neuron hold the same characteristics.

3. **Clustering:** The process of grouping spikes with similar feature vectors is called clustering and is often done manually with a graphical tool.

Describing Neural Behavior

Stimuli and response can be expressed as spike trains. A spike train is a sequence of APs originating from a single neuron, and is one way of describing neural response. Even though APs vary slightly when it comes to amplitude, duration, and shape, they are usually treated as all-or-none events. The signals are therefore expressed by the timing of the firings alone. For that reason, spike trains are described by listing the spike times. For n spikes, the times are denoted by t_i, where $i = 1$, 2, ..., n. For all i, t_i will be in the interval $0 \leq t_i \leq T$, where 0 is the start time of the recording of the neural response, and T is the end time [43]. This way of describing neural response is suitable for single recordings but gets quite impractical when describing the results of multiple recordings, and impossible to use for generalization. It is possible to describe the firings probabilistically, but this also gets quite problematic for a large number of spikes. Instead, statistical models are used to express the firings of neurons. It is not possible to predict spike sequences based on the probabilities of one spike occurring since the spikes most likely are not independent occurrences [43].

2.5 Neural Coding

Neurons transmit messages via spatio-temporal pulse patterns. This means that the messages are communicated over both space and time. “What is the information contained in these patterns?” “How is it encoded?” “Are we able to decode it?” “Can we use this information to mimic the stimuli and response?” These are some of the many questions arising when studying neurons and the communication between them.

Neural coding comprises neural encoding, the mapping of stimulus to response, and neural decoding, the mapping of response to stimulus [43]. “What will the response be for a given stimulus, and given a specific response, what was the stimulus applied?” Finding this relationship has proven to be difficult. Neural responses are complex and variable and are usually products of both the intrinsic dynamics of the neuron and the temporal characteristics of the stimulus. To make it even more challenging, the neural response can also provide different outputs for the same stimuli [43]. This trial-to-trial variation of neural response is quite significant and can originate from a different level of arousal and attention, randomness associated with various biophysical processes that may affect the neural firing, and other cognitive processes taking place [43]. Trial-to-trial variations are often considered as “noise”, but it is still not known if this instead is part of some rich neural coding that is not yet fully understood [54].

There are several different coding schemes describing the firings of the neurons, often falling into one of four categories; rate coding, temporal coding, population coding, and sparse coding.

Rate Coding

As the name suggests, rate coding takes the rate of the spikes into consideration when transmitting information. It is a “simple” method and assumes that most, or all, of the information is contained in the firing rate. One limitation of this method is that it does not capture the information in the timing of the spikes. An indication that rate coding is not the only code used is that it is known that the human reaction time can be as short as a few hundred milliseconds, which is too short for the brain to do a temporal average before reacting to the stimuli [46].

Mean firing rate
One common way of describing the firings is by the mean firing rate. However, this is not an unambiguous term. In literature, there are at least three different types of averages used when it comes to describing spikes; average over time, average over trials, and average over neurons. The mean firing rate over time has been used to describe the firing rate since the 1920s [55] and is the most common definition. The average over time of a firing rate is given by the following equation:

$$v = \frac{n_{sp}(T)}{T}, \tag{2.2}$$

where v is the average firing rate, T is a time window, and $n_{sp}(T)$ is the number of spikes contained in that time window [46].

Temporal Coding

Temporal coding takes the timing of the individual spikes into consideration when transmitting information, instead of just the rate.

Time-to-first-spike
In the time-to-first-spike coding scheme, it is thought that the time between the stimuli and the first spike in the response holds the transmitted information. Studies have shown that most of the information from new stimuli is given in the first 20 to 50 ms after the onset of the response [56, 57], but only considering the first spike when looking at the transmitted information is considered to be highly simplified [46].

Phase
In this coding scheme, the time-to-first-spike scheme is applied on a periodic signal. In some areas of the brain, oscillations are quite common, and the phase is seen in relation to these oscillations [46]. One of the best studied temporal coding models is the theta-phase precession in hippocampal place cells [58].

Population Coding

With population coding, it is not possible to look at singular neurons individually, but the response of several neurons must be considered as a whole [43]. Just like coding for single neurons, neural population codes at multiple scales; the diversity of the neural responses, the spatial and temporal properties, cross-correlations and state-dependence of cortical activity [59].

Sparse Coding

Studies have shown that information can be represented by a small number of active neurons out of a larger population. This is referred to as sparse coding. An advantage of this form of coding is the effectiveness when it comes to storage and energy due to the low number of active neurons [60].

2.6 Spiking Neuron Models

There are several different models for simulating spiking neurons, both singular neuron models and population models. The type of coding used has an impact on the model, as some coding schemes require more detailed descriptions of the neural dynamics than others. Some of them being detailed, conductance-based models, while others are simpler and more calculation friendly.

The Integrate-and-Fire Models

The different integrate-and-fire models are simple, computational friendly equations. When the sum of stimuli added together (or integrated) reaches a certain threshold, an AP is triggered. Integrate-and-fire models are models where APs are described as all-or-none events, and consist of two separate components; an equation that describes the changes in the membrane potential, and a mechanism to generate spikes [61].

The Hodgkin-Huxley Model

The Hodgkin-Huxley membrane capacitance model is a model describing the creation and propagation of APs. The model describes the nerve cell with an equivalent electric circuit, and from this circuit arises a set of differential equations. The model is quite simple but manages to explain several of the membrane properties quite accurately [45].

2.7 Spike Train Metrics

An important aspect of neural coding is the ability to compare spike trains to identify similarities and dissimilarities in both stimulation and response. As opposed to most signal processing techniques, spike train metrics operate on binary sequences. The way the data is described will influence the methods used in analyzing the data.

A challenge that becomes more apparent when comparing spike trains compared to other types of signals is the large signal-to-signal variations. Neural signals have a trial-to-trial variability, and the challenge is not finding two patterns that are exactly the same, but rather to see if they are similar enough to most likely come from the same source or be responses to the same type of stimuli with statistical significance. A metric is needed in order to quantify this similarity.

There are several ways to measure this similarity, or dissimilarity, between two spike trains. One of the simplest ones is to compare the number of spikes in each spike train. However, this method misses the temporal features of the spike trains. Several metrics have been proposed to resolve temporal structures. One is cost-based metrics, exemplified by the Victor-Purpura distance [62]. Here, a cost is added for operations such as removing, adding and moving spikes for transforming one spike train into another one. The distance between two spike trains is given as the minimum total cost of transforming one of the spike trains into the other one.

Another type of metrics is kernel-based metrics, which is a group of metrics that map the spike trains into the vector space before calculating the distance. The van Rossum distance [63] is an example of this, which calculate the Euclidian distance between the spike trains after mapping them into vector space [64]. In addition to being sensitive to the timing of the spikes, there are also metrics that are sensitive to the distance between spikes, or the temporal pattern of the spikes [62].

2.8 Machine Learning

Machine learning (ML) is an area of study on algorithms and statistical models used by computers to efficiently perform tasks without explicit instructions. In classical programming, the machine relies on static program instructions, while the basis of ML is mathematical models that can be “trained” on specific data, and later use the information extracted from patterns and interferences. ML is used in cases where it is infeasible to create algorithms capable of performing the specific task, and there are innumerable of real-world applications relying upon this type of programming, ranging from fraud detection [65] to diagnosing cancer [66].

ML is ordinarily divided into three main categories: supervised, unsupervised, and reinforcement learning, dependent on the input and output, and the problem to be solved.

- **Supervised learning** is when using the algorithms on unknown or new data, but the pattern is known. A function learns how to map inputs to outputs by examining assigned input-output pairs called a training set [67]. Supervised learning is often divided into classification or regression problems. Classification is the task of classifying something and is used when the outputs are restricted to specific alternatives. An example is email spam filtering [68]. The system would need to correctly recognize spam, although the distinct pattern may not have been identified during training. Regression is used to predict something. Here the output is usually a numerical value within a range. An example here is predicting stock market prices [69].
- **Unsupervised learning**, on the other hand, is when the data is known, but the patterns are not. This method is used in cases where the specific knowledge on what is essential information contained in the input is unknown. As opposed to supervised learning where patterns in the input are provided as labeled data, the patterns are learned by the machine without any explicit feedback [67]. The most common task is clustering. Here, the input data is divided into homogeneous subsets. This method is particularly useful in the examination of large data sets [70]. An example is crime data mining where an ML approach is used to understand patterns in criminal behavior by finding patterns in crimes committed by the same offender or group [71].
- In **reinforcement learning** the system learns from a series of reinforcements. The system then tries to perform the tasks that maximize the cumulative reward. Some applications for reinforcement learning are optimization of chemical reactions [72] and playing Atari games [73].

For most practical reasons, the raw input data is transformed into a new representation where the pattern recognition problem is simpler to solve. This set of new, useful attributes are called features and are considered an important part of ML systems [70]. The choice of features hugely impacts the performance of the model. While a good selection of features improves the performance, a poor choice can negatively affect the accuracy, as well as cause overfitting. The extraction of features is normally done in a preprocessing step, before the training of the program. The raw data given to the model is not always sufficient for performing ML tasks as the raw data may not contain sufficient features. In such cases, a solution could be to create new features and feed those to the model. These are often created by adding, subtracting, or multiplying existing features.

For many applications, a training set is used to tune the parameters of an adaptive model. The training set usually consists of a set of input values, $\mathrm{X}_{\mathrm{TRAIN}}$, and a target vector, $\mathrm{Y}_{\mathrm{TRAIN}}$, containing the labeling of the input values. After training, the model can be expressed as $\mathrm{Y}_{\mathrm{PRED}}(\mathrm{X}_{\mathrm{TEST}})$. Here, $\mathrm{X}_{\mathrm{TEST}}$ is an input not found in the training set, $\mathrm{X}_{\mathrm{TRAIN}}$, and $\mathrm{Y}_{\mathrm{PRED}}$ is the output, which is encoded in the same fashion as the target vector. The precise form of the function $\mathrm{Y_{PRED}}(\mathrm{X_{TEST}})$ is decided during the training phase or the learning phase. The model is usually more precise if the training set is more extensive and with an ample variety of inputs. However, the accuracy is normally better when the training set corresponds to data that the model is going to be implemented with.

The task of generalization is an important aspect of ML, which typically occurs when a model has been trained using a limited number of samples, and this model should be able to react well to previously unseen data. If an exceedingly complex model is chosen to describe only a few samples, the model might not respond well to new types of input. While a too complex model leads to overfitting, a likewise simple model leads to underfitting [70].

Gradient Descent

Gradient descent is an iterative optimization algorithm for finding a (local) minimum of a function and is one of the most popular and widely used ML algorithms [74]. The algorithm operates by taking a step along the negative of the gradient of the function at the current point to converge to a minimum possible low [75]. The size of the step taken is determined by a learning rate, α, which can be either constant or decaying over time. A high learning rate may lead to overshooting the minimum, while a small learning rate reaches the minimum very slowly. In artificial neural networks (ANNs), the algorithms use training data sets to update internal parameters of the model (weights and biases) with the goal of finding a set of parameters that perform well against some performance measure. The error gradient to minimize is calculated by comparing the prediction of the model with the training set provided to the ANN. The loss is given by

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) \tag{2.3}$$

where l is the iteration number, $\alpha > 0$ is the learning rate, θ is the parameter vector, and $E(\theta)$ is the loss function. The internal model parameters are updated after a certain number of samples have been worked through. This number of samples is called the batch size, and influence the trade-off between the accuracy of the parameter update, and the update time.

If the batch number is the same as the number of samples in the training data set, the learning algorithm is called a batch gradient descent. This form of gradient descent can be quite slow since the gradients for the whole data set is calculated before updating the parameters, and it is unmanageable for data sets that do not fit in memory. For large data sets, it performs redundant computations since it recomputes gradients for similar examples before each parameter is updated [74].

If the batch size is equal to one, the algorithm is a stochastic gradient descent (SGD). Here, the parameters are updated for each training example and are usually much faster than batch gradient descent due to the lack of redundant computations. A downside with the rapid fluctuations of SGD that may occur is that it complicates the convergence to a minimum since it keeps overshooting. An upside with these fluctuations is that it might jump to a newer, better local minimum. However, it has been shown that when slowly decreasing the learning rate, the SGD converges in the same way as the batch gradient descent [74].

The last type of gradient descent is the mini-batch gradient descent, where the batch size is somewhere between one and the number of samples in the training set. An advantage is that it reduces the variance of the parameter updates, which can lead to more stable convergence. The optimal number of samples in each mini-batch is application dependent [74]. One completion of the training data set is called an epoch.

2.9 Artificial Neural Networks

An artificial neural network, or simply neural network (NN), is a computational learning system inspired by biological neurons, with the goal of translating data input to output of a desired form, usually different from the input. ANNs are a framework for many different ML algorithms working together to process complex data inputs.

An illustration of an ANN is given in Figure 2.6. Each node represents an artifi-

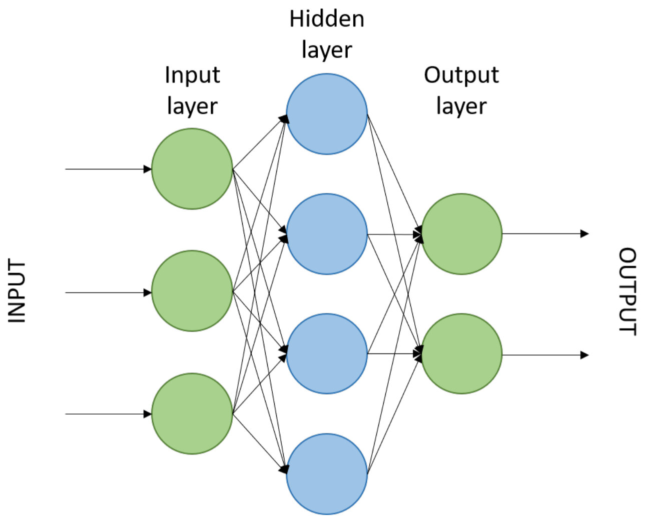


Figure 2.6: Artificial neural network with one hidden layer with four nodes.

cial neuron while the arrows represent the connections or synapses. These artificial neurons loosely model the neurons in the brain. The synapses transmit the signal from one neuron to another one. A signal received is processed by the artificial neuron before it is transmitted to the connected neurons. Typically, neurons are bundled in layers, and different layers perform different types of transformations on the data. Signals travel from the first layer, the input layer, through the hidden layer to the last layer, the output layer [75]. The shape of the input and output layer depends on the shape of the input and output data, and the hidden layer(s) are often dependent on the purpose of the NN. A deep neural network is an ANN with multiple layers between the input and the output layer [76]. The nodes in a hidden layer are often called hidden units.

Perceptrons are binary classifiers and are the same thing as a single layer network [77]. A perceptron consists of four parts: input values, weights and biases, summation function, and an activation function. These are shown in Figure 2.7. All inputs $\mathrm{x_n}$ are weighted with a corresponding weight $\mathrm{w_n}$. These weighted in-

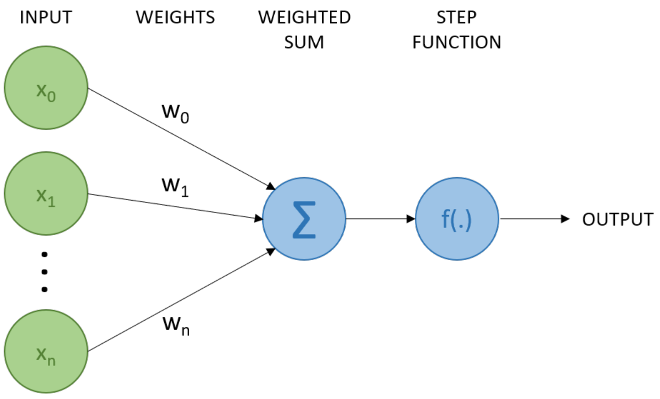


Figure 2.7: Overview over a perceptron. Weights are applied to the input before the weighted sum is passed through a function that produces the output.

puts are then added together, and the weighted sum is applied to the activation function, which determines the binary output of the perceptron. The choice of activation function depends on the application but is used to map the input between requires values. Adding a bias shifts the activation function curve. The activation function can be divided into three main types: binary step, linear, and non-linear, where the non-linear is the most common type. The most common types of these non-linear activation functions are given below.

Sigmoid function
The sigmoid activation function, or the logistic activation function, is given by

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.4}$$

and is plotted in Figure 2.8a. This function is useful when the probability is being predicted, since the function is limited between 0 and 1. [78]

Hyperbolic tangent function
The hyperbolic tangent function, often denoted tanh has the same shape as the sigmoid function, but is limited between -1 and 1 as shown in Figure 2.8b. An

advantage of this is that the negative inputs will be strongly negative, and values close to zero inputs will be mapped near zero in the output [78].

Rectified linear unit function

The rectified linear unit (ReLU) function is currently the most used activation function [79]. The function is given by the function

$$f(x) = max(x, 0), \tag{2.5}$$

and is depicted in Figure 2.8c. The ReLU is half rectified, which maps all negative input values to zero.

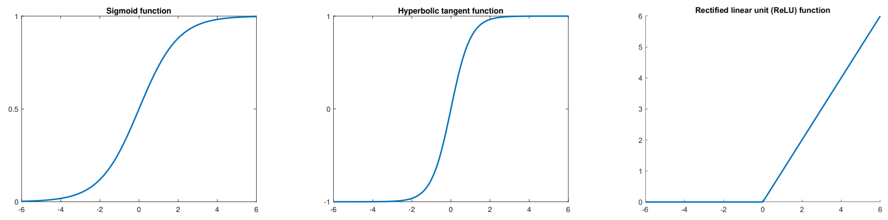
Softmax

The softmax function, or the normalized exponential function, turns a real numbered vector of K-dimensions into probabilities; a vector of numbers between 0 and 1 that sums to one. The function is given by

$$S(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}, \tag{2.6}$$

for $j = 1, 2, ..., K$ [80].

An important property of activation function is if they are derivable. The derivative is used to know how much to change the curve, and in which direction.



(a) Sigmoid function. (b) Hyperbolic tangent function. (c) Rectified linear unit function.

Figure 2.8: Common types of activation functions.

Types of Neural Networks

Feedforward networks

Feedforward networks is a group of NNs where the connections between the nodes do not form a cycle [76]. A graphical representation of a feedforward network can be seen in Figure 2.9. The figure shows a compact, general representation with $\mathrm{x_t}$ inputs, and $\mathrm{h_t}$ outputs, while A represents the hidden layers. The information moves from the input nodes, through the hidden nodes, and to the output nodes. This flow moves only in one direction, without any loops. Figure 2.6 used to illustrate a neural network shows an example of a feedforward NN with three inputs x_t with $t = 3$, two outputs h_t with $t = 2$ and one hidden layer, A, containing four nodes.

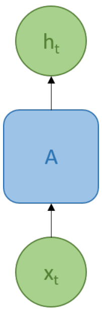


Figure 2.9: Feedforward network.

Recurrent neural networks

Recurrent neural networks (RNNs) belongs to a group of neural networks that processes sequential data. As opposed to feedforward networks, the connections in RNNs do form cycles. This allows information to persist and the networks to have an internal memory which is needed in order to exhibit temporal dynamic behavior [76].

An RNN is depicted in Figure 2.10. The diagram on the left shows the RNN

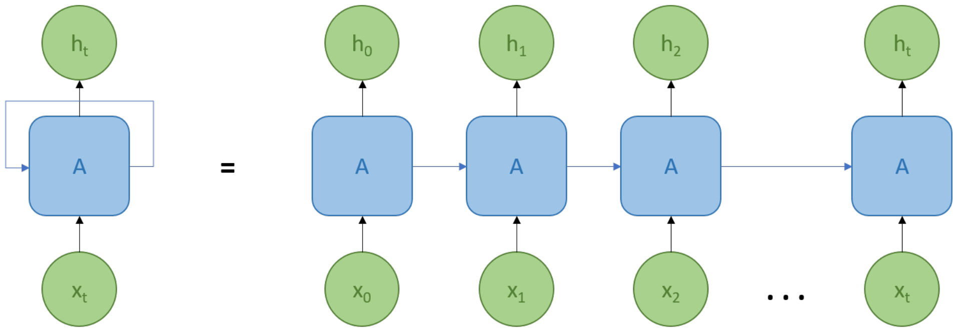


Figure 2.10: An unfolded basic recurrent neural network.

with x_t as input values, h_t as output values, and the cell itself A. The arrow from A to A indicates the feedback loop. The diagram on the right shows the unrolled network. Here, the result is passed from one network to its successor.

2.10 Long Short-Term Memory

The long short-term memory architecture is a type of RNN and is widely used in applications such as acoustic modeling of speech [81], handwriting recognition [82], and prediction of protein secondary structure [83]. The LSTM architecture was initially developed to deal with problems related to long-term dependencies common in other types of RNNs [84].

The LSTM Cell

A typical LSTM architecture is depicted in Figure 2.11. Here, the four gray boxes are NN layers, while the red circles inside the cell are pointwise operations. The

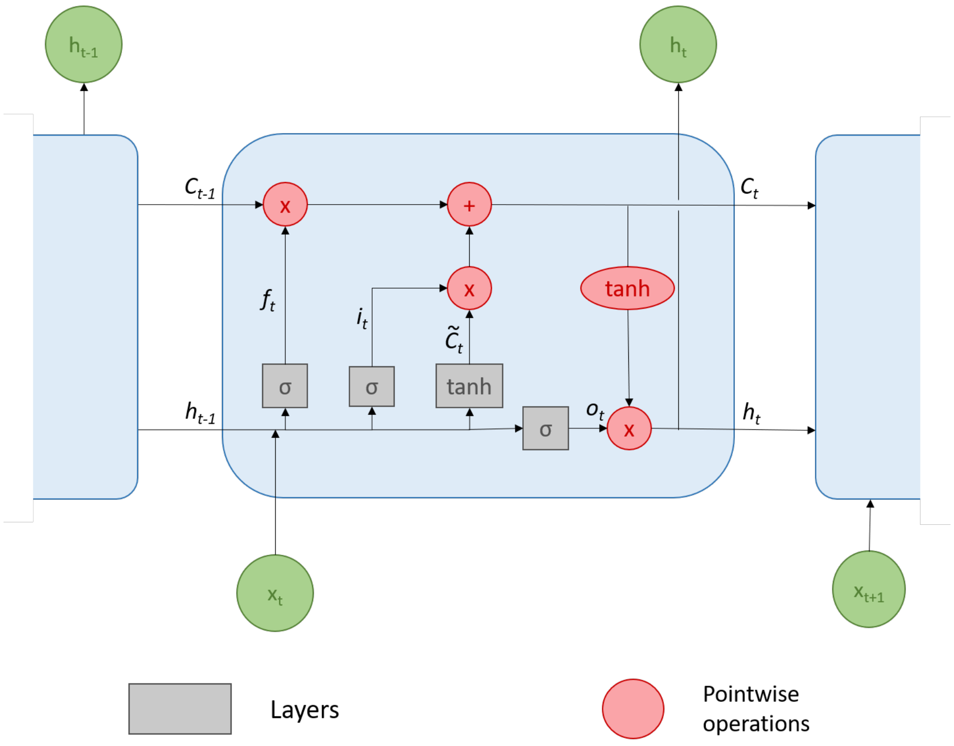


Figure 2.11: Long short-term network architecture.

unit consists of a cell with three gates: input, output, and forget, which regulate the flow of information in and out of the cell. The task of the cell is to keep track of dependencies in the input data. The function of the input gate is controlling to which extent a new value is allowed into the cell. A standard RNN usually contains a single layer, but the LSTM network contains four interacting layers.

The key element of the LSTM is the cell state. This can be seen as the horizontal line at the top of the cell, with C_{t-1} as input and C_t as output. This is the main flow of information and is regulated by the gates.

The forget gate describes how much information to keep from the cell state and how much to forget, and can be described with the equation

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.7}$$

where W_f are the weights, and f_t is the amount to keep, where 0 is nothing and 1 is everything. The input gate layer decides which values to update with the help of a sigmoid layer:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i). \tag{2.8}$$

Here, the tanh layer finds possible elements to add, while $\tilde{C}_t$ decides which ones to add to the state.

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{2.9}$$

The cell state is then updated from C_{t-1} to C_t given by the equation

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{2.10}$$

Here, the old state is multiplied by the activation function of the forget gate, f_t, before adding the new scaled version of the new candidate values. The output of the cell is given by the equations

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.11}$$

and

$$h_t = o_t * tanh(C_t). \tag{2.12}$$

Here, the values are first filtered by the sigmoid layer to decide which parts to output. The cell state is then mapped to values between -1 and 1 with the tanh function, before being multiplied by the output of the sigmoid gate. This last step is used to filter out unwanted information [85, 86].

There also exists a bidirectional version of the LSTM, BiLSTM, where the sequences is fed through the LSTM twice; once from the beginning to end, and once from the end to the beginning. This can be useful in cases where one wants the network to learn from the complete time series at each time step [87].

3 | Theory

3.1 Complete Data Set

The data set used is publicly available through The Collaborative Research in Computational Neuroscience (CRCNS) [16], which is a joint program for theoretical and experimental neuroscience. A paper on the internally organized mechanisms of the head direction cells was published along with the data set [52].

The complete data set consists of 42 recordings from seven different mice; five male, and two female. In total, recordings from 1077 cells were collected. Out of these, 353 were categorized as HD cells.

In 41 out of 42 sessions, the wake phase of the mouse included foraging for 35-40 minutes, proceeded and followed by two hours of sleep. In the last session the mouse followed a radial maze after the second sleep phase instead of looking for feed, with no sleep phase afterward.

In all seven mice, electrodes were placed in the anterior thalamus. In three of the seven mice, electrodes were also placed in the PoS. In the remaining four mice, electrodes were placed in the hippocampal CA1 pyramidal layer for accurate sleep scoring. The probes used consisted of either 4, 6 or 8 shanks, each shank with either 8 or 10 recording sites [52]. The recording setup is shown in Figure 3.1. Electrode recordings were made when the mice were searching for feed in an open environment (I); a box of 53x46 cm, with 20 cm high walls. The box was black and included two visual cues.

The neural response is recorded (II), spike sorting is performed on the waveforms, and the responses of the individual cells are obtained (III). Video recordings are made (IV), and the coordinates (V) and directions (VI) of the head of an individual mouse were extracted from these recordings and saved in separate files. The angles were measured with the help of two LED lights mounted on top of a headstage.

Information about the head angle and spikes from the individual neurons are combined, and the HD cells are classified based on the criteria given in Table 3.1 (VII).

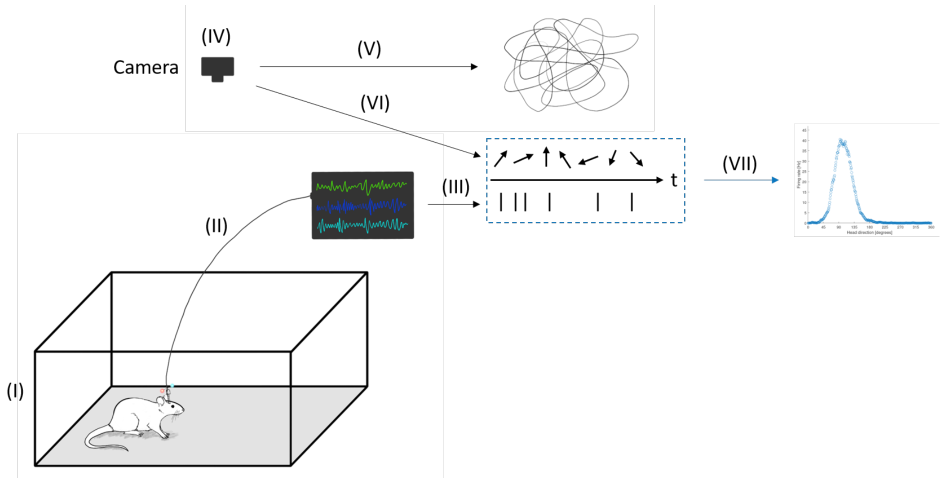


Figure 3.1: Recording setup using camera and probes to find head direction and spike times.

The following are included in the data set:

- Times and waveforms of detected potential spikes.
- Results of spike sorting.
- Local field Potentials from the hippocampus.
- The coordinate and direction of the mice head.
- Video files from which the head coordinates and directions are extracted.
- Metadata tables giving properties of the neurons and characteristics of the recording sessions.

The extraction of HD cells from the total population of recorded neurons is not included in the publicly available data set. Step (VII) is therefore done separately with code described in Section 3.2.

3.2 MATLAB Code Extracting HD Cells

The MATLAB functions provided, given in Appendix D.1, locates HD cells from the data set described in Section 3.1. The function extracts the spike times and the corresponding angles and combine them in a predefined structure for each cell. The function only considers spikes during wake phase. The data is then transposed such that one measurement is given for each awake millisecond.

Detection of HD cells

The identification of HD cells are done by smoothing the data with a Gaussian kernel and fitting it with von Mises distribution before calculating a concentration parameter and the peak firing rate. The inclusion criteria for HD cells are given in Table 3.1. The MATLAB code provides two types of tuning curves; one in cartesian

Table 3.1: Head direction cells inclusion criteria.

Concentration parameter	>1
Peak firing rate	>1
Probability of non-uniform distribution	<0.001

coordinates, and one in polar coordinates. These two types are illustrated in Figure 3.2 which shows the curves for one HD cell and one cell not classified as an HD cell. This code is through the rest of the text referred to as the HD Cell Extracting Code (HDCEC).

3.3 The Victor-Purpura Distance

One type of cost based metric mentioned in Section 2.7 is the Victor-Purpura distance.

The distance d between two points, A and B, is given by

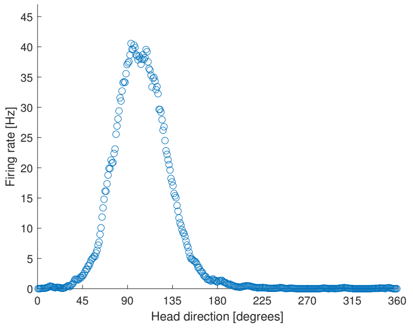
$$d(A,B) = min\left\{ \sum_{j=0}^{n-1} c(X_j, X_{j+1}) \right\}, \tag{3.1}$$

where $X_0, X_1, ..., X_n$ is a path from $A = X_0$ to $B = X_n$, and $c(X_j, X_{j+1})$ is a cost function intended to capture basic biological functions [62].

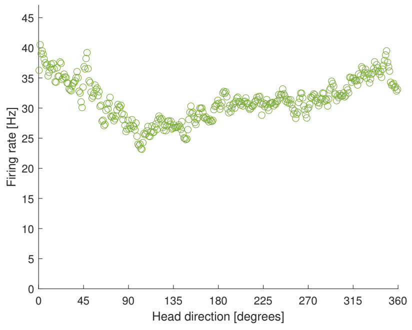
A cost of one is added to the operations of removing and adding spikes, while the cost of shifting spikes in time is given as a product of a cost parameter, q and the number of time units moved. The value of q sets the time scale of the analysis. For $q = 0$, the total cost will be the number of spikes that differs between the two sequences. For large q, the distance is transformed from rate distance to temporal distance since it is "cheaper" deleting all the spikes and then add them again than shifting them [88]. If two spike trains are identical, apart from a single spike that occurs in t_a for A and t_b for B, the cost function will be $c(A,B) = q|t_a - t_b|$, where q is in the unit of sec^{-1} [62].

An indicative scale of the cost is given by $|t_a - t_b| = 2/q$. Two spikes are considered comparable if they occur within an interval of $2/q$. The code used can be found at [89].

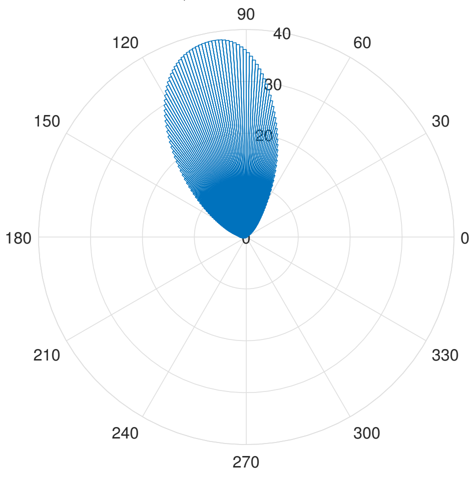
The Victor-Purpura distance is applicable on both single and multineuronal spike trains [64].



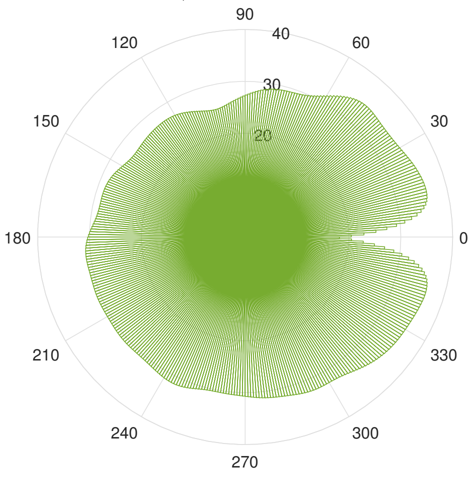
(a) Tuning curve of HD cell from Mouse12-120806, Shank 3



(b) Tuning curve of cell from Mouse12-120806, Shank 1



(c) Polar plot tuning curve of HD cell from Mouse12-120806, Shank 3.



(d) Polar plot tuning curve of cell from Mouse12-120806, Shank 1.

Figure 3.2: Tuning curves from code in Appendix D.1.

3.4 The van Rossum Distance

The van Rossum distance also calculates the distance between two spike trains, but as opposed to the Victor-Purpura distance, the van Rossum distance first maps the spike train into a vector space of functions before calculating the distance. The van Rossum distance is more computational friendly than the Victor-Purpura distance [63].

Given a spike train,

$$f(t) = \sum_{i}^{M} \delta(t - t_i), \tag{3.2}$$

where t_i denotes the spike times, $t_i > 0$, and $M > i$. The delta function is replaced with a kernel $h(t)$:

$$h(t) = \begin{cases} 0, & t < 0 \\ e^{-t/\tau}, & t \geq 0. \end{cases} \tag{3.3}$$

This is equivalent to adding an exponential tail to all the spikes.

The Euclidean distance, D, between two spike trains f and g is defined as

$$D^2(f, g)_\tau = \frac{1}{\tau} \int_0^\infty [f(t) - g(t)]^2 dt. \tag{3.4}$$

The distance uses a time constant, τ, which sets the time scale of the comparison. When $\tau \to \infty$, the metric acts as a rate different counter, while it for $\tau \to 0$ counts non-coincident spikes [63]. This is the opposite as for the Victor-Purpura distance, where a cost of $q = 0$ gives the difference is spike count between the spike trains. The code used can be found at [90, 91].

3.5 Izhikevich Neuron Model

The model, described in [24], simulates large-scale networks of spiking neurons. It reproduces spiking and bursting behavior of eight known types of cortical neurons, categorized according to patterns found in intracellular recordings. The model combines the Hodgkin-Huxley model with an Integrate-and-fire model to create a biologically plausible model as well as being effective for computations.

The following differential equations are implemented

$$v' = 0.04v^2 + 5v + 140 - u + I, \tag{3.5}$$

$$u' = a(bv - u), \tag{3.6}$$

and the following after-spike resetting criteria are also included.

$$\text{if } v \geq 30 \text{ mV}, \text{ then} \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3.7}$$

Here, v' and u' are derivatives of v and u with respect to time. A brief overview of the different variables and parameters is given below. A more detailed explanation can be found in [24].

Variables

- v: Membrane potential
- u: Membrane recovery

Parameters

- a: Time scale of recovery variable u.
- b: Sensitivity of the recovery variable u to subthreshold fluctuations of the membrane potential v.
- c: After-spike reset value of the membrane potential v.
- d: After-spike reset of the recovery variable u.

After a spike, when v reaches 30 mV, the membrane potential and membrane recovery variable reset according to Equation (3.7). By changing the parameters mentioned above, the spiking sequence from the simulation changes. Typical values for different spiking patterns of different types of cortical neurons are given in [24].

Changing the parameters causes various intrinsic firing patterns. By tweaking the model parameters, the model returns moderately different spiking pattern for the neuron.

Equation (3.5) is found by looking at intracellular recordings, and is a "one-type-fits-all" kind of equation for simulating large-scale networks. When simulating single neurons, other variations of the function may be used to better adapt to the effects of single neurons [24].

3.6 MATLAB Deep Learning Toolbox

The MATLAB Deep Learning Toolbox is a set of tools providing a framework for designing and implementing deep learning neural networks in MATLAB. The toolbox includes algorithms, pretrained models and apps for some commonly known types of neural network architectures such as convolutional neural networks and LSTMs.

Layers

The toolbox includes several types of layers for different tasks. The relevant layers are mentioned in Appendix B.1 [92].

Hyperparameters

Hyperparameters control many aspects of the behaviour of the algorithm, and is values that are used to control the learning process. Some of the most relevant hyperparameters is mentioned in Appendix B.2 [93].

Optimizers

When working with LSTMs, MATLAB have three different gradient descent solver options; stochastic gradient descent with momentum (SGDM), root mean square propagation (RMSProp) and adaptive moment estimation (adam). Different solvers have different approaches to deal with the different problems related to the types of gradient descent, and there are different pros and cons related with each solver [93].

`sgdm`
A problem with the stochastic gradient descent is that the algorithm may oscillate along the path of steepest descent towards the minimum. The SGDM tries to reduce this oscillation by adding a momentum term to Equation (2.3):

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}), \tag{3.8}$$

where γ determines how much of the previous steps to include.

`rmsprop`
While the SGDM reduces the oscillation by adding a momentum, it only uses one single learning rate for all the parameters. RMSProp differs from this by using different learning rates for different parameters which automatically adapt to the loss function. This is done by using a moving average of element-wise squares of the parameter gradients,

$$v_l = \beta_2 v_{l-1} + (1-\beta_2)[\nabla E(\theta_l)]^2, \tag{3.9}$$

where β_2 is the decay rate of the moving average. This moving average is used to normalize the updates of each parameter individually,

$$\theta_{l+1} = \theta_l - \frac{\alpha \nabla E(\theta_l)}{\sqrt{v_l} + \epsilon}, \tag{3.10}$$

where the division is performed element-wise.

`adam`
Adam is similar to the RMSProp in the way of updating parameters but adds a momentum term. An element-wise moving average is used for both the parameter gradients and their squared values,

$$m_l = \beta_1 m_l + (1-\beta_1)\nabla E(\theta_l), \tag{3.11}$$

and

$$v_l = \beta_2 v_{l-1} + (1-\beta_2)[\nabla E(\theta_l)]^2, \tag{3.12}$$

where β_1 and β_2 are the decay rates named '`GradientDecayFactor`' and '`SquaredGradientDecayFactor`' respectively. The network parameters are updated with the moving average

$$\theta_{l+1} = \theta_l - \frac{\alpha m_l}{\sqrt{v_l} + \epsilon}. \tag{3.13}$$

Empirical results prove that adam compares favorably to other stochastic optimization methods [94].

4 | Methods

The specifications of the hardware used is given in Appendix A. MATLAB R2019a on Windows was the software used. Descriptions of the main functions can be found in Appendix D.2.

4.1 Notation

For practical reasons, specific HD cells are referred to using the following notation: *day.shank.HDcell*. Here, *day* is the day of recording, which for this study is 12080X, where X is substituted with either 6, 7, 8 or 9. Furthermore, *shank* is the shank that recorded the response of the HD cell, and *HDcell* is the number of the HD cell according to the code described in Section 3.2.

4.2 Delimitations

From the complete data set, the mouse named 'Mouse12' is investigated, and four of five recording days are considered in this study. Mouse12 had electrodes inserted into the thalamus, as well as the hippocampus for accurate sleep scoring. The recording days are the following: 120806, 120807, 120808, and 120809.

4.3 Initial Analysis

When analyzing data, the type of analysis done affects the results found. The starting point of the analysis can be the difference between useful and less useful results. Therefore, this starting point should be chosen carefully. Also, it is not always clear where to start looking when examining types of data that have been less explored beforehand. A valuable tool when receiving new data is visualization. Visualization often helps with the understanding of complex problems, which in turn helps to find better solutions and descriptions for said problems. Not only does graphical representations of data ease the understanding of complex problems, but it can also highlight deviations and errors in the data sets. For the given data sets, multiple ways of visualization have been used in order to correlate different possible dependencies easily. Most of these have not panned out, but some have hinted of correlation between variables and spurred further investigation.

Extraction of HD Cells

The HD cells are classified by using the HDCEC in Appendix D.1 described in Section 3.2 on the data set.

Extraction of Sessions

The free head movement of the mouse is taken into consideration by dividing the spike recordings into different "sessions". A session consists of the spikes recorded inside an angle bin without the head moving outside of the angle bin between spikes. If the head is inside an angle bin, moves out of that angle bin and then back again, the spikes recorded are considered as two different sessions. The sessions are given as sequences of ones and zeros, one value for each millisecond, where one represents a spike and zero represents no spike.

Size of angle bin

The size of the angle bin is essential. Ideally, the bins should be quite narrow to be able to account for small changes, one degree for instance, but this is not feasible with limited data material since the number of sessions per angle bin becomes too low to conduct a proper analysis. With wide angle bins, the sessions are longer, but the analysis and comparison of sessions and angle bins are more complicated. It is not known if the head, during the time spent in the angle bin, is held quite still or if it moves back and forth inside the bin. Spike characteristics, at least the mean, varies from one "side" of the bin to the other one, and have to be taken into consideration. This is especially evident in large bins compared to the directional firing range. The optimal size of the angle bin also depends on which HD cell is being analyzed, since the directional firing range and the peak firing rate varies from cell to cell.

Spikes per session

The firing rate and standard deviation (SD) per session are calculated to see if there are any differences between the sessions within the same angle bin. The average firing rate is calculated using Equation (2.2). A problem with this method is that it also includes all the very short sessions that consist of only a few milliseconds. If a spike happens to occur in one of these short sessions, the firing rate for that particular session will be artificially high. In addition, many short sessions with no spikes will contribute to a lower mean.

There are several ways of dealing with this problem, but all come with different drawbacks. One way is to remove all sessions with a certain number of spikes, e.g. one spike, since the probability that two spikes occur inside the short sessions is lower than one occurring. However, by doing this, one removes much potentially useful information, and the ratio between spikes and no-spikes is skewed.

Another approach is removing all sessions shorter than a certain length. Here, as opposed to only removing sessions with a certain number of spikes, the ratio is sustained. A problem here is that much data is lost from an already sparse data set.

4.4 Time Between Spikes

After dividing the spikes into sessions, the time between the spikes in each session is found for sessions where there are minimum two spikes. Both the mean and the SDs are calculated for all angle bins. Since there need to be at least two spikes in each session, the bins should be of a larger width than for calculating tuning curves in order to get enough sessions to be able to give meaningful results.

A problem with this method of finding the sessions before finding the time between spikes is that one “loses” quite a large amount of the data. The time between two spikes in different angle bins are not included in the analysis, and with rapid changes of angle bins, and many sessions, this adds up to be a significant amount of data excluded: the narrower bins, the more loss of data. With large bins, one suffers less data loss since the switching of sessions happens less often, although fine details may be lost.

4.5 Comparison of Sessions

By comparing different sessions from the same HD cell, it is possible to see if sessions from the same angle bin follow the same pattern. Sessions from within the same angle bin are compared, and the spike train distances are found. These distances are then compared with the distances found when comparing sessions from different angle bins to see if there are any significant differences between the results. This is done to see if it is possible to say if two arbitrary sessions from the same HD cell originate from the same angle bin or not.

Sessions with less than a certain number of spikes are removed. Sessions with few spikes are more likely to be similar by chance than by an actual common pattern, and are therefore omitted from the analysis.

The directional firing ranges are symmetric around the PFDs. To account for possible symmetry on a spike train level, the same comparison methods are used on the angle bins inside half of the directional firing range to see if there are any differences.

Victor-Purpura distance

A problem with using the Victor-Purpura metric (Section 3.3) for this comparison is that the metric does not account for the length of the compared spike trains. Due to this limitation, only sessions with approximately the same length are compared. This pose quite large limitations on the selection from the data set. Sessions with lengths within five milliseconds of each other are compared. Here, the sessions are considered “similar enough”, and allows comparisons of more sessions than only the sessions of the same length. With a short minimum distance between the spikes, the number of possible comparisons is reduced.

The cost value is looped through to see if one cost value provides better results than the others.

van Rossum distance

The van Rossum distance (Section 3.4) have some of the same limitations as the Victor-Purpura distance when it comes to the length of the spike trains compared; thus only sessions with approximately the same length are compared.

The time scale parameter τ is looped through possible values the same way as the cost in the Victor-Purpura distance.

Average firing rate

The same comparison method is conducted with the average number of spikes as a similarity measure instead of the Victor-Purpura and van Rossum distances. In this case, the requirement that the difference in length of the sessions compared has to be less than a specific limit is removed. This is because the average firing rate takes the length of the spike sequence into consideration. The “distance” between two spike trains is here the difference in average spike rates for the two sessions. A problem with the average firing rate comparison is that it misses temporal structures in the spike trains.

4.6 Finding a Model

Since the HD cells all have different directional firing ranges, PFDs, and peak firing rates, this is something that needs to be taken into account when creating a model describing the response of the cells. Ideally, a single model is able to describe the response of all HD cells, with only a few changes in parameter values taking the properties of each HD cell into consideration. The input should be given as a function taking the head direction as an input parameter, and should be general for all HD cells. An example is depicted in Figure 4.1. Here, the input is a function of the head direction, while the model is a function, H, of the three parameters α, β and γ, representing the properties directional firing range, PFD and peak firing rate. The output is the spike times. Ideally, it should also be possible to

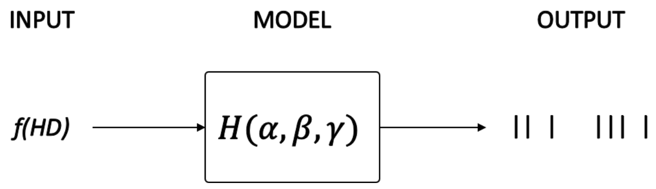


Figure 4.1: Ideal model for describing response of head direction cells.

reverse the model flow with spike times as inputs and the head direction as output.

First, an already existing model is used on one cell to see if it is possible to replicate the response for one specific direction. Here, the input stays constant. After a model is found for describing one direction, the same model should be tried to see if it can replicate the response of the HD cells for other directions, only by changing the input. If that works, models for different cells should be compared to see if there are any ways of creating a general model.

The Izhikevich Model

The Izhikevich model is chosen due to the combination of being biologically plausible as well as being computational effective. In addition, with the form of the output of the model being close to the form of the recordings, little adaption of the model or preprocessing of the recordings is needed in order to use the model.

The code for the spiking neuron model found in [24] is adapted to the current use. The code found is for modeling large-scale networks. The function is modified to return the spiking times and take the duration of the simulation as an input parameter. The duration of the simulation is set to the duration of the session which it is going to be compared with. The problem with the Victor-Purpura distance only being applicable on spike trains with different lengths is therefore no longer an issue, and the method proposed is applicable on all sessions.

The original code includes several elements of randomness. This is done to mimic different dynamics for the different cells and different weights for the different synaptic connections, in addition to varying input. When modeling one neuron, the first two factors are not relevant. The randomness of the input is removed to see if it is possible to find a model.

Some possible values of the model parameters making the model exhibit the same properties as known cortical neurons are provided [24]. The upper and lower boundaries for the parameter values looped trough are based on the values from the models describing these known cortical neurons. Step sizes are chosen relative to the start and end value.

Table 4.1: Parameter values for the Izhikevich model.

Parameter	**Start value**	**End value**	**Step size**
a	0.02	0.1	0.01
b	0.2	0.26	0.01
c	-65	-50	5
d	0.05	8	0.05
I	1	15	1

All possible combinations of the values in Table 4.1 are looped through, and the parameter values that give the shortest Victor-Purpura distance are chosen. This is done for all the sessions for a specific shank, HD cell, and angle bin.

Due to uncertainties regarding the optimal value of the cost parameter q, different values of this parameter are also tested. The cost parameter influences how much shifting spikes compared to removing and inserting spikes is prioritized, and have an influence on the "optimal" parameter values of the model.

When a model has been found for each session in an angle bin, these models should be compared to each other in order to find a common model capable of describing all sessions in an angle bin. After finding a model able to describe firings inside an angle bin, the model needs to be altered so that it is able to replicate the response from all directions, but with different input. The next step is then to generalize the model so that, by changing some parameter values, the model is able to describe all HD cells.

4.7 Machine Learning

There are several possible questions to ask when looking at ML and HD cells. Both classification and prediction problems can be formulated with the goal of getting a better understanding of HD cells and on the way of finding possible models for modeling them.

Possible Problems

For a spike sequence, from a given HD cell, is it possible to tell from which angle or angles the sequence is recorded?
This question can be regarded as a classification problem, with different angle bins being the different categories possible. Here, the spike sequence is the known factor, while the direction is the unknown one. The HD cells are being analyzed separately, and it is theoretically possible to solve this problem by looking at one cell alone. If a suitable model is found, it would be interesting to see if it is possible to replicate the response, this time with predefined inputs instead of recordings. It is also possible to do a sequence to sequence classification, where each data point, i.e. millisecond, is classified to an angle. However, this may be quite difficult, or

impossible, as the sequences only consisting of ones and zeros.

When looking at sessions from different HD cells combined, what is the angle or angles from which the sequence originates?
This problem is quite similar to the one above, but instead of looking at only one HD cell, the response from several HD cells are combined. The results are highly dependent on the selection of HD cells, especially the dispersion of PFDs. It would also be interesting to see if it is possible to find input for the NN, capable of replicating the same response as when using the recordings. Also here could it be interesting to investigate the possibilities for doing a sequence to sequence classification.

For a given sequence (from a random HD cell), is it possible to tell from which HD cell the sequence is collected?
This question is also a classification problem since the number of possible classes is limited, and in this case, equal to the number of HD cells considered. By looking at one sequence from a random HD cell; do this sequence contain enough information to say anything about which cell it originates from? An additional problem could here be to find out the direction of the head from which the sequence was recorded, or add the direction as additional information, together with the sequence. Some challenges related to this problem revolve around the available selection of HD cells. If the characteristics of the HD cells are quite different, the task would be easier than if they were quite similar.

For a given HD cell, what will the sequence be for a given angle?
This problem is a prediction problem, as the number of possible sequences are innumerable. From the described machine learning problems, this is the one closes to the main goal of finding a model being able to describe the neural responses of a cell.

From these possible problems, the first two are chosen as areas of focus, with the sequence classification, and not sequence to sequence classification. The problems seem more manageable than the last problem, and the results can give valuable insight into finding a model.

Layers and Hyperparameters

MATLAB, with the Deep Learning Toolbox, is used as a framework for the ML. Some of the relevant features and settings explained in Section 3.6.

A simple LSTM network architecture for classification is given in Figure 4.2, and is chosen since LSTM networks are proven suitable for sequential data. The sequential input layer inputs the features, i.e. spike times, of the different sessions. This is followed by one or more LSTM or BiLSTM layers. For the prediction of labels, there are one or more fully connected layers, a softmax layer, followed by a classification output layer. If more LSTM or BiLSTM layers are used, dropout layers (Appendix B.1) are used after the LSTM layers to avoid overfitting.

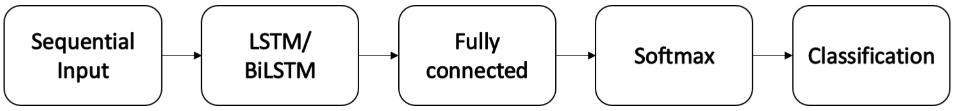


Figure 4.2: Architecture of a simple LSTM network.

The architecture in Figure 4.2, with one LSTM/BiLSTM layer and one fully connected layer is the one used as a basis in the training of both one and multiple HD cells.

The hyperparameters are codependent, and it is not possible to tweak only one at a time to get a "most optimal result". The most relevant hyperparameters that are changed to find the most optimal NN are described in Appendix B.2. From the three available optimizers, adam is used.

The execution environment hyperparameter '`ExecutionEnvironment`' is set to 'CPU'. Other options were tried but were unsuccessful.

Creating Training and Test Sets

The responses of the HD cells are split into sessions as described in Section 4.3. Each session is labeled with the corresponding angle bin.

The training is done on 80% of the data set, while the testing is done on the remaining 20 %.

By default, the software divides the training data into mini-batches and pads them so all sequences in a mini-batch have the same length. If the padding is too large, it can have a negative impact on the network performance. Both the training and test data set are sorted on length to avoid excess padding.

Angle bin size

While too narrow angle bins lead to short sessions, too wide angle bins are also problematic since they remove important nuances. Due to differences in direction firing range and peak firing rate, different HD cell have different "optimal" angle bin widths. When analyzing one HD cell at the time, this does not pose too many problems, while combining several HD cells in the same study, more challenges are emerging. The "optimal" angle bin width will then be a trade-off between the properties of the individual HD cells in the analysis.

Omit short sessions

Short sessions are a problem when looking at both one HD cell as well as multiple HD cells. Even for HD cells with a high peak firing rate and with the head in the PFD, the short sessions of a few milliseconds often contain only zeros, and the few that contain one or two ones do not have enough information to say anything specific about the session by itself. These short sessions may also pose the problem that there are several equal sessions extracted from different angle bins. Even when combining the response of several cells, there are still sessions with no spikes.

One “solution” to this problem could be to remove all sessions shorter than a certain length. The chances that the sessions are equal becomes smaller with increasing length. Unfortunately, this removal of sessions decreases the usable data drastically due to the large number of short sessions.

Balancing data sets

All data sets have angle bins that contain more sessions than others. This causes an imbalance in the NNs, as they train more on these angle bins. Under-sampling is done by removing sessions from the angle bins with the most sessions to account for this imbalance. The number of sessions per angle bin can vary some, and still be considered “balanced enough”. For simplicity, sessions are removed from angle bins such that all angle bins contain the same number of sessions equal to the one with the lowest number of sessions originally. The sessions removed are the shortest ones, so that the long sessions, containing most information, are kept. The removal of excess sessions is done first, before the data is shuffled. The data set is then divided into new, smaller data sets used for training and testing.

Evaluating Performance

The performance is evaluated using accuracy as a measurement. The accuracy is the average of the number of correctly predicted angle bins. A confusion matrix is created for each training and plotted for visual inspection.

4.8 One HD Cell

It is investigated whether or not it is possible to classify the direction of the head solely based on the response from one HD cell. This is done to see if the information from one HD cell alone contains enough information to describe the head direction, and if it is possible to get some more knowledge about the way HD cells communicate.

Only Recordings Inside Directional Firing Range

Outside the directional firing range, the HD cells fire with a frequency below a baseline threshold. This baseline is in many cases quite low, which results in several sessions only including zeros, or in some cases a few spikes. Intuitively, several angle bins will contain similar sessions, which makes it hard to distinguish between the different directions outside of this directional firing range. One solution to this problem could be to treat all directions outside the directional firing range as one class, or only look at the recordings when the head is held inside the directional firing range.

4.9 Multiple HD Cells

By combining the response of several HD cells, more information is available. Instead of only detecting angles within the directional firing range, a good selection of HD cells provide the possibility of classifying spike sequences from all head directions.

Feature Creation

As mentioned in Section 2.8, features are an important part of ML. The features for the ML problem with multiple HD cells are given by the number of HD cells, where the spike times for each cell are the features. By combining these already existing features, it is possible to create new ones. One way of doing this is by combining responses from two and two HD cells, to see if the accuracy improves. The multiplication operator should be avoided as the sequences consist of ones and zeros, and a lot of the information contained would be removed by multiplying the sequences.

4.10 Combining Models and Neural Nets

After an NN is found, it is interesting to see if it is possible to create input that yields results similar to the recordings, as this could help with the creation of artificial HD cells, which in turn could be interfaced with machines.

As a starting point, the Izhikevich model is used to create input to the NN. This is done using the method described in Section 4.6. A model is found for each session fulfilling a criterion, for all HD cells in a specific day. Possible criteria are sessions with a minimum number of spikes, and sessions longer than a minimum length.

When using a minimum number of spikes as the criterion, this number should be chosen such that there are enough sessions available for the generation of the models. A low number of spikes may lead to problems since the available sessions may include noise, which then is introduced in the models. High numbers may lead to very few models being generated.

If using a minimum session length as the criterion, this minimum needs to be short enough such that model creation is possible for all the different angle bins. At the same time, the minimum should be long enough such that the sessions include as many spikes as possible.

After the models are generated, the input is created by choosing a random model for each angle bin and HD cell which then are combined the same way as the input in the training of the NN. A random offset is added to the input current of each model to ensure variance in the models. If no model is available for a given HD cell and angle bin, the generated sequence consists of only zeros.

All sessions are given the same length for simplicity. The number of sessions per angle bin is equal for all angle bins.

5 | Results

5.1 Delimitations

Not all results are given in this paper. A representative selection has been chosen to keep the paper within its scope. An additional selection is available in the appendix, while a lot of similar results have been excluded.

The results given here are deemed to be the most representative ones for the complete data set, and the analysis conducted, and are in many cases used to give an insight or ground for discussion.

For the reader to be able to see connections between the different types of analysis conducted and methods used, the results are given for the same day, or HD cell where applicable, as far as possible. In most cases, this is day 120806, and HD cell 120806.3.8.

5.2 Initial Analysis

Recording Days

The length of the wake phase for each day is shown in Table 5.1. The length varies between 15 and 36 minutes.

Table 5.1: Length of wake phase.

Day	Awake time [ms]
120806	2 188 901
120807	2 079 901
120808	910 000
120809	2 103 800

Extracting HD cells

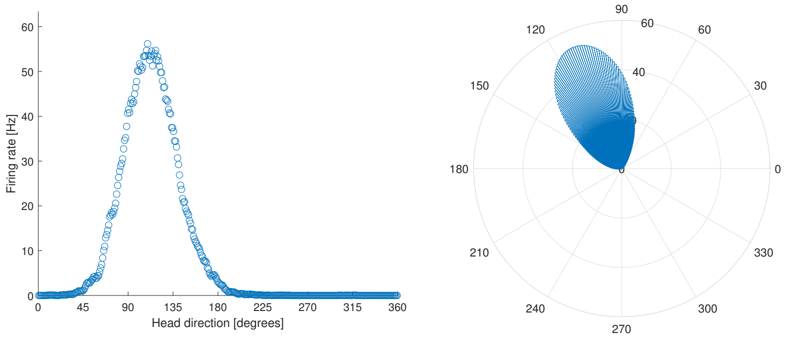
The HD cells are separated from the rest of the cells in the data set by the HDCEC. The number of HD cells is given in Table 5.2. About 15% of the detected cells are classified as HD cells.

Table 5.2: Number of head direction cells extracted from data set.

Day	#cells	#HD cells
120806	60	21
120807	62	13
120808	66	13
120809	271	20
Total	459	67

Tuning curves

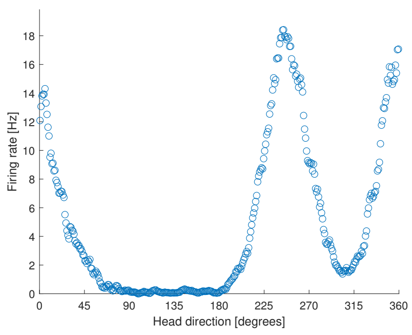
In addition to classifying which cells are HD cells, the code also plots the tuning curves of the different cells, both in cartesian and polar coordinate systems as seen in Figure 5.1. When looking at the tuning curves, there are some that deviates



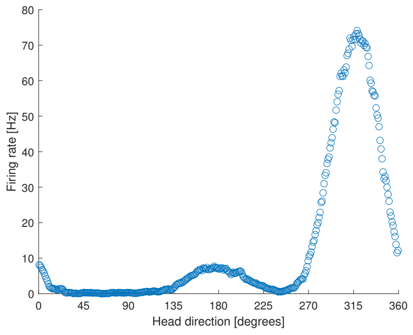
(a) Tuning curve. (b) Polar plot tuning curve.

Figure 5.1: Tuning curves of HD cell 120806.3.8.

from the "typical" form. Examples are given in Figure 5.2. In Figure 5.2a, two peaks are present, instead of one. When comparing it to the PFD given by the HDCEC, the PFD in the code is calculated to be approximately 291°, which is somewhere between the two large peaks. In Figure 5.2b, there is a second, smaller peak in addition to the "normal" one found in most tuning curves for HD cells, but the PFD of 312° does here match quite well with the peak of the tuning curve. Due to this offset, an additional criterion is imposed; any second peak has to be below 30% of the tallest peak. Two of the HD cells from 120806 and two from 120807 are therefore omitted from the results. In these cases, the PFDs found given by the HDCEC do not match the direction at which the largest peak's peak



(a) Tuning curve of cell day 120806.4.8.



(b) Tuning curve of cell 120806.4.5.

Figure 5.2: Imperfect tuning curves.

firing rate occurs.

When moving towards 0° in the polar plots, the length of the bars decreases towards zero, as can be seen in Figure 3.2d. This problem does not seem to impact the rest of the polar plot in any other way, and the HD cells with a directional firing range that do not contain 0° are not affected.

Visual Inspection of Tuning Curves

The directional firing range and the peak firing rate can be found by visual inspection of the tuning curves. The results for HD cell 120806.3.8 are given in Table 5.3. The peak firing rates vary between 3 and 60 spikes per second, and the directional firing range ranges between 90° and 150°. The peak firing rate is also given by the

Table 5.3: Directional firing range and peak firing rate found by visual inspection of tuning curves.

HD cell	**Directional firing range**	**Peak firing rate [spikes/sec]**
1	90°	3
2	100°	3.5
3	90°	55
4	120°	25
5	120°	35
6	100°	45
7	150°	15
8	110°	60
9	100°	16
10	100°	14
11	90°	5
12	90°	9

HDCEC. This peak firing rate is usually a bit lower than the one found by visual inspection of the tuning curves.

When looking at the peak firing rate for the four days in this study, it varies between 1.4 and 125 spikes per second. The directional firing range varies between 85° and 180°.

Head Directions Represented

The PFDs are given from the HDCEC. These are plotted against the time spent with the head in each direction, with angle bins of width 1°. The results are shown in Figure 5.3 as well as Appendix C.1. The radial axis denotes the time

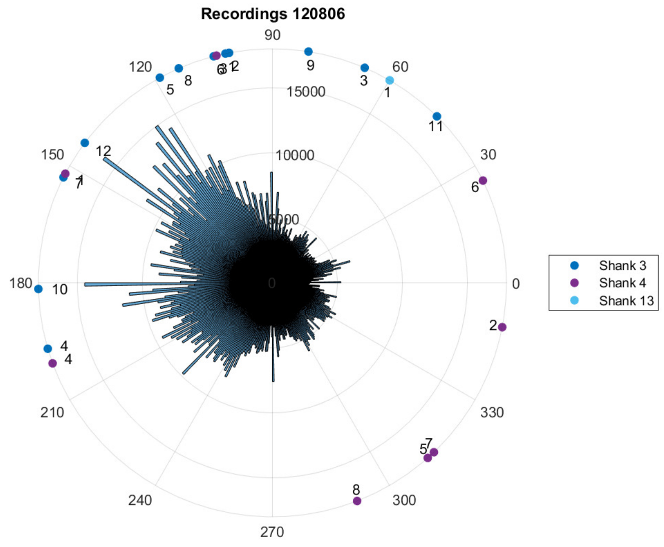


Figure 5.3: Time spent with head in each direction and PFDs of HD cells for 120806.

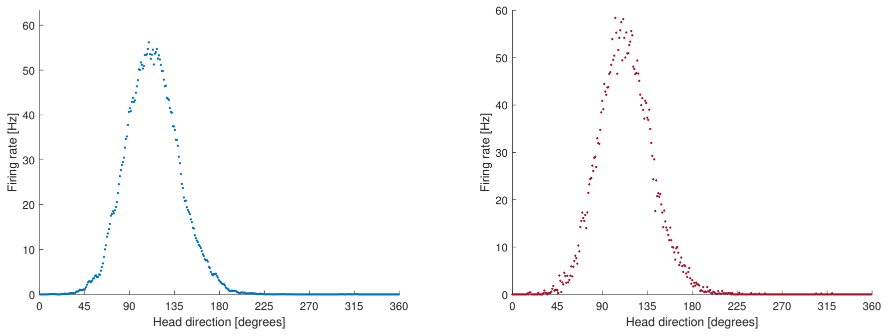
in milliseconds. The bars give the total time spent in each direction, but do not give explicit information about the continuous duration of the head in a certain direction. From the figure, it can be seen that the mouse held its head more in the directions between 100° and 240° degrees than the rest of the directions. As the figure also shows, the PFDs do not always correspond to the time spent with the head in each direction. This means that the amount of data well suited for analysis is sub-optimal.

Extraction of Sessions

The angle bins represent the surrounding angles, rounded to the closest angle bin, i.e. with an angle bin width of 10°, the angle bin named 90° includes angles in the range 85° to 94°. This way of calculating angle bins is done for simplicity when it comes to the coding. The angle bin 0° includes the angles from 0° to 4°, and 355° and 360°.

Comparison of methods

Though tuning curves of the HD cells can be plotted with the HDCEC, the tuning curves are also plotted from the extracted sessions to check if the two methods yield similar results visually. Here, angle bins of 1° is used, and for each angle bin, the number of spikes is divided by the time spent in each angle bin before being plotted. Figure 5.4 shows the tuning curve of 120806.3.8 created with the two methods.



(a) HDCEC.

(b) Based on sessions.

Figure 5.4: Comparison of tuning curves.

Size of angle bins

When looking at the impact of the angle bin width when it comes to the firing rate, a simple calculation is used as an example. For HD cell 120806.3.8 in Figure 5.4, the slope of the curve is approximately 1.1 spikes per second for the first half of the angle bin. An angle bin width of 10° creates a difference in firing rate of more than ten spikes per second between the two extremities of the angle bin.

The tuning curve of HD cell 120806.3.8 using an angle bin width of 10° is shown in Figure 5.5. When comparing this to the tuning curves using 1° (Figure 5.4), the main properties are preserved. Table 5.4 shows the number of sessions extracted for day 120806 for three different-sized angle bin widths. The results for rest of the days are given in Appendix C.2. All days experience a large drop in number of sessions when increasing the angle bin width.

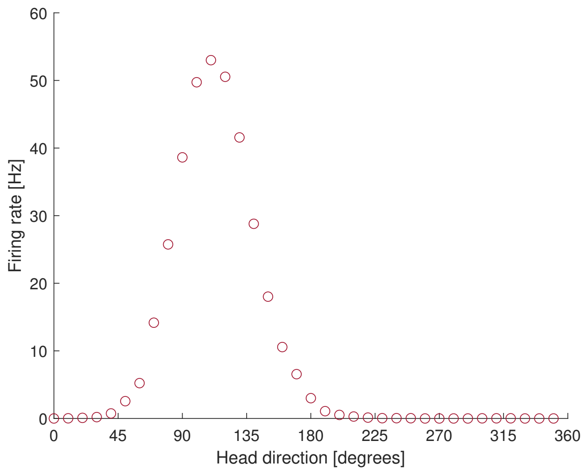


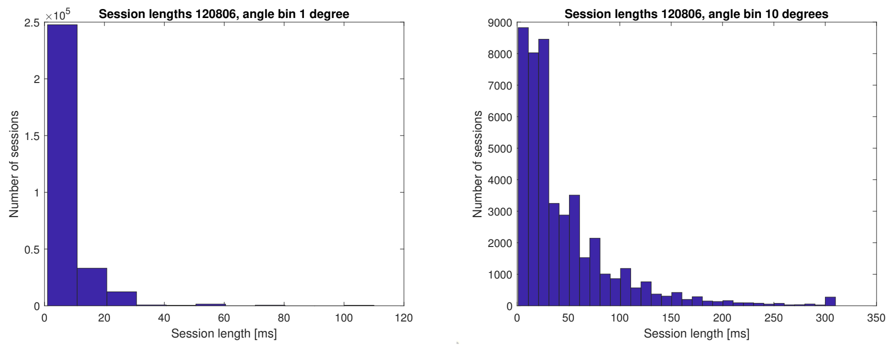
Figure 5.5: Tuning curve of head direction cell 120806.3.8 with angle bin width of 10°.

Table 5.4: Number of extracted sessions for day 120806.

Angle bin width	**#sessions**
1°	296 983
5°	84 841
10°	45 866

Length of extracted sessions

Sessions are extracted as described in Section 4.3, with angle bins with width of 1° and 10°. Figure 5.6 shows the length of the sessions extracted. The bar furthest to the right represents all sessions longer than either 100 ms or 300 ms, depending on the width of the angle bins. The sessions extracted range in length from 1 ms to the low thousands. As can be seen from both the histograms; short sessions dominate. The head of the mouse moved a lot during the recordings and rarely stayed in the same position for a longer amount of time. For an angle bin width of 1°, most sessions are shorter than 10 ms, while for an angle bin width of 10°, many of the sessions are shorter than 30 ms. The session lengths for the other days of recordings for angle bin widths of 10° are given in Appendix C.3. Here, similar results are found as for day 120806; sessions shorter than 30 ms dominate for angle bin widths of 10°, while there are few sessions longer than 10 ms when dividing the recordings into angle bins of 1°.



(a) Angle bins of 1°. (b) Angle bins of 10°.

Figure 5.6: Length of extracted sessions for day 120806.

Spikes per session

The Tables 5.5-5.7 all include results for HD cell 120806.3.8 for angle bin 120°, and each present the impact of tweaking a single premise in this selection of the data set. The angle bin 120° is one of the angle bins in the middle of the directional firing range, and is chosen due to the high concentration of spikes.

Table 5.5 includes the mean and SD for the different angle bin widths 1°, 5° and 10°. The smaller the angle bin, the larger the SD. The number of available sessions for analysis increases with increasing angle bin width. Generally, when calculating the average firing rate for each session in every angle bin individually, the average firing rate differs a lot, even within the same angle bin. The SD is in many cases larger than the mean itself.

Table 5.5: Angle bin width.

Bin width	Mean	SD	#sessions
1°	55	101	1140
5°	52	67	1622
10°	50	48	1785

The results in Table 5.6 and 5.7 are both given for an angle bin width of 5°, and are similar to results for other angle bins and HD cells. The angle bin width of 5° is chosen to ensure a sufficient number of sessions for analysis. When removing sessions with spikes with fewer spikes than a certain number, both the mean and the SD substantially decrease.

Table 5.7 includes the mean and SD when removing sessions shorter than a minimum length. By removing sessions shorter than a minimum length, the mean stays approximately the same, while the SD decreases with an increasing minimum length. The number of sessions also decrease with increasing minimum length.

Table 5.6: Minimum number of spikes.

#spikes	Mean	SD	#sessions
≥2	39	52	1107
≥4	18	37	734
≥10	1	11	574

Table 5.7: Minimum session length.

Length [ms]	Mean	SD	#sessions
≥2	53	64	1580
≥10	52	40	1239
≥100	51	19	55

Some of the results from Table 5.5, 5.6 and 5.7 are given in Table 5.8 for five different angle bins. The average spike rates for sessions were calculated for several cells. The results showed here for HD cell 120806.3.8 are representable for the results found for the rest of the tested HD cells.

Table 5.8: Mean and deviation for different angle bins.

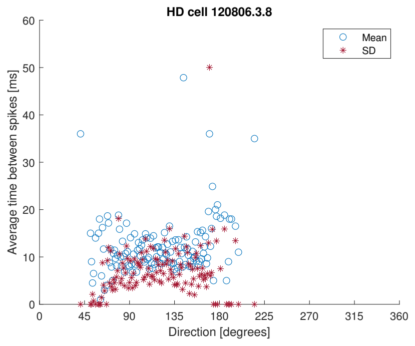
Angle bin	Mean[a]	SD[a]	Mean[b]	SD[b]	Mean[c]	SD[c]
110°	56	68	20	40	48	21
115°	53	65	21	37	46	22
120°	52	67	18	37	51	19
125°	49	65	20	38	42	24
130°	41	61	12	30	34	20

[a]All sessions included.
[b]Without sessions with 1-3 spikes.
[c]Without sessions shorter than 100 ms.

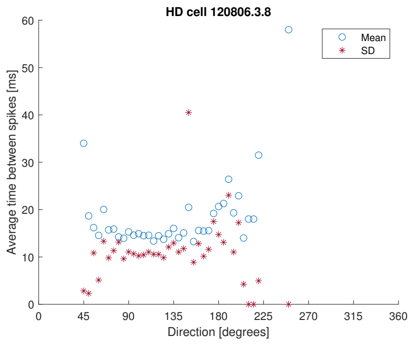
5.3 Time Between Spikes

A visualization of the mean and SD of the time between spikes is given in Figure 5.7. Here, the mean and the SD for each angle bin are plotted in the same way as the frequency in the tuning curves. The y-axis shows the time between spikes in milliseconds, while the x-axis gives the angle bins.

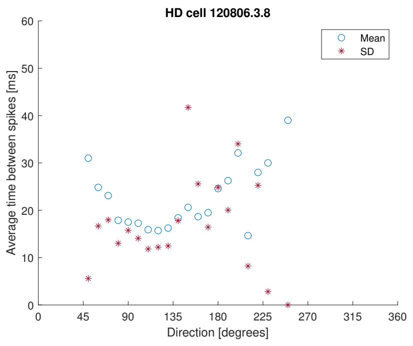
A visualization of the results with angle bins of width 1° for HD cell 120806.3.8 can be seen in Figure 5.7a. One SD outlier with an y-value >100 is removed. As opposed to the tuning curves, the plotting of the time between spikes does not appear to form a specific pattern. For some HD cells, there are some angle bins without any data points. This is most likely due to the fact that the time spent in the angle bins were too short for more than one spike to occur. The SD is also quite high compared to the average in many cases.



(a) Angle bin width of 1°.



(b) Angle bin width of 5°.



(c) Angle bin width of 10°.

Figure 5.7: Time between spikes for HD cell 120806.3.8 for three different angle bin widths.

The results with angle bin width 5° and 10° are given in Figure 5.7b and 5.7c, respectively. Even though all the plots originate from the same HD cell, there are large differences. One could argue that the time between spikes is a bit lower around 100° and that the SD is lower around 45° and 225°.

5.4 Comparison of Sessions

The results from the extraction of sessions from 120806.3.8 can be seen in Table 5.9. Here, the number of sessions found is given for angle bin widths 5° and 10°, and the minimum number of spikes is set to 5, 10 or 15. The total number of sessions without a lower limit on the number of spikes is given in Table 5.4. From these alternatives, the results from 10° angle bin and 10 spikes (in bold-face) is used for the results in the rest of this section. This is chosen due to the large number of sessions available as well as having a sufficient number of spikes per

Table 5.9: Number of extracted sessions.

Angle bin width	#spikes/session	#sessions
1°	5	202
1°	10	14
1°	15	3
5°	5	1043
5°	10	104
5°	15	18
10°	5	1727
10°	**10**	**406**
10°	15	131

session for analysis.

Only comparisons where it is possible to compare a session with at least one session from the same angle bin and one from a different angle bin are included. Sessions where several comparisons yield the same shortest distance are registered as indeterminable, unless all the comparisons giving the shortest distance are from the same angle bin.

Victor-Purpura Distance

For a maximum distance of 5 ms between the sessions for the comparisons, the number of total comparisons is 3037. For the 406 sessions found, only 224 of them were able to be compared with sessions of similar length from both same and different angle bins. The cost giving the highest percentage of comparisons originating from the same angle bin, yielding the shortest distance is $q = 0.03$, with 22.3%. The different values of q looped through are 0.001 to 0.01 with a step size of 0.001, and 0.01 to 1 with a step size of 0.01.

van Rossum distance

The same 5 millisecond maximum distance between the spikes to be compared is used with the Victor-Purpura distance is used for the van Rossum distance as well. The number of compared sessions are, therefore, the same. The time scale parameter found to return the highest percentage of comparisons originating from the same angle bin, yielding the shortest distance is $\tau = 0.28$, with 24.6%. The values looped through for τ are 0.001 to 0.01 with a step size of 0.001, and 0.01 to 1 with a step size of 0.01, as well as 1 to 10 with a step size of 0.5.

Average Firing Rate

When using the average spiking rate instead of the Victor-Purpura metric as comparison criteria, 82 215 comparisons are made instead of 3037. The number of sessions compared with both sessions from the same angle bin and different angle bins is 395. The percentage of comparisons originating from the same angle bin, yielding the shortest distance, is 15.4%.

For all comparison methods, the number of comparisons of sessions from different angle bins is significantly higher, varying from about four to five times larger than the number of comparisons of sessions from the same angle bin.

When only looking at one half of the tuning curve, the percentage of comparisons originating from the same angle bin, yielding the shortest distance, increases to about 30%.

Using an angle bind width of 5°, and including sessions with either a minimum of 5 or 10 spikes, yielded similar or worse results.

Some of the distances found when comparing sessions in 120806.3.8 are given in Table 5.10. The first column indicate whether the two sessions compared originate from the same angle bin, or different ones. The proceeding three columns give the Victor-Purpura distance, the van Rossum distance, and the difference in average firing rate for the sessions. The bold-faced numbers are the lowest distance for the specific metric and session combination.

Table 5.10: Comparison of sessions.

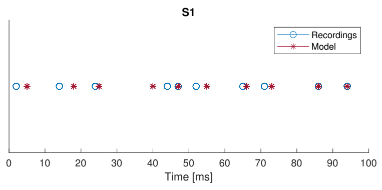
Same/ different	**Victor-Purpura ($q = 0.6$)**	**van Rossum ($\tau = 0.28$)**	**Difference in average firing rate**
Same	3.02	**106**	8.2
Same	2.78	276	**2.2**
Different	**2.08**	235i	9.4
Same	**3.08**	**211i**	**2.4**
Different	3.90	371	23.2
Different	3.90	252	13.4
Same	4.72	191i	14.1
Different	**2.26**	282	20.8
Different	**2.26**	**50i**	**1.2**

5.5 Finding a Model

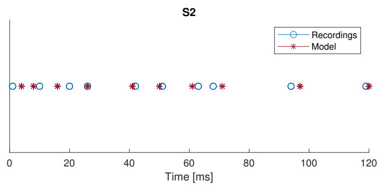
The Table 5.11 shows the “optimal” parameter values, i.e. values yielding the shortest Victor-Purpura distance when comparing the recordings and model, for the first five sessions, $S1$, $S2$, $S3$, $S4$ and $S5$ from HD cell 120806.3.8 for cost value $q = 0.01$. The cost value is found by iterating through different values, and choosing the most optimal value. The session recordings and models using the optimal model parameter values for each session are plotted in Figure 5.8. Several of the comparisons are visually quite similar. The number of spikes are often similar, and some are merely slightly shifted. For some comparisons, the models match

Table 5.11: Optimal model parameter values.

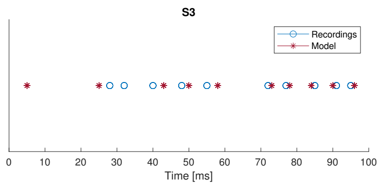
	a	***b***	***c***	***d***	***I***	**Distance**
$S1$	0.1	0.23	-65	0.3	11	0.18
$S2$	0.03	0.26	-65	1.1	12	0.2
$S3$	0.07	0.26	-60	0.15	10	0.43
$S4$	0.06	0.25	-55	3.5	11	0.26
$S5$	0.08	0.21	-50	2.25	9	0.9



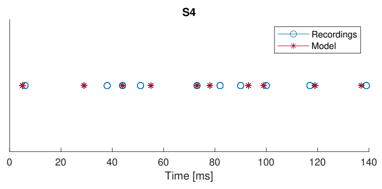
(a)



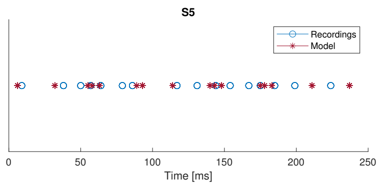
(b)



(c)



(d)



(e)

Figure 5.8: Comparisons of sessions and models.

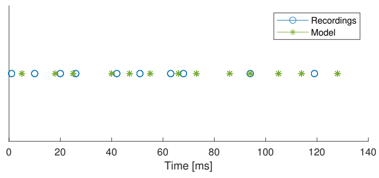
the recordings well apart from a few spikes, like $S3$, shown in Figure 5.8c.

Table 5.12 shows the Victor-Purpura distance for the first three sessions when the model with optimal parameter values are used. The bold-faced numbers are the distance for the sessions where optimized values are used. The distances between session recordings and a model found using another session's optimal model values are significantly larger than the distance between a session recording and the model found using the optimal values for that session. In Figure 5.9, the recordings of session $S2$ are plotted against the model found using the optimal model parameter values for session $S1$ and $S3$. Here, one can see that these models are a poorer match than the one in Figure 5.8b.

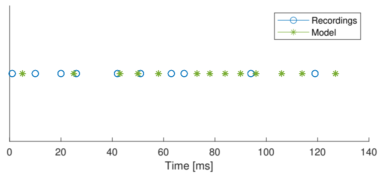
Table 5.12: Victor-Purpura distance for different optimal model parameter values.

Session	Distance[a]	Distance[b]	Distance[c]
$S1$	**0.18**	1.49	1.03
$S2$	3.53	**0.2**	4.42
$S3$	1.14	2.62	**0.43**

[a] $\boldsymbol{a}$ = 0.1, $\boldsymbol{b}$ = 0.23, $\boldsymbol{c}$ = -65, $\boldsymbol{d}$ = 0.3, $\boldsymbol{I}$ = 11.
[b] $\boldsymbol{a}$ = 0.03, $\boldsymbol{b}$ = 0.26, $\boldsymbol{c}$ = -65, $\boldsymbol{d}$ = 1.1, $\boldsymbol{I}$ = 12.
[c] $\boldsymbol{a}$ = 0.07, $\boldsymbol{b}$ = 0.26, $\boldsymbol{c}$ = -60, $\boldsymbol{d}$ = 0.15, $\boldsymbol{I}$ = 10.



(a) Model $S1$.



(b) Model $S3$.

Figure 5.9: Comparison of recordings from session $S2$ and model from session $S1$ and $S3$.

5.6 Machine Learning

Layers and Hyperparameters

Due to time constraints, not all possible layer and hyperparameter value combinations (Appendix B.2) could be tested. A selection of the different values of used hyperparameters is given in Table 5.13. The upper and lower limit gives the largest and lowest value used. Most training and test were done with the simple

Table 5.13: Hyperparameter values and number of hidden units.

Hyperparameter	
Initial learn rate	0.0001 - 0.05
Mini-batch	32 - 512
Max epoch	30-200
Drop factor	0.01-0.5
Drop period	5-30
No. of hidden units	100 - 1000

network architecture shown in Figure 4.2, which includes only one LSTM/BiLSTM layer as well as one “Fully connected” layer. The number of hidden units in the LSTM/BiLSTM-layer were between 100 and 1000. Deeper networks did not give notably improved results, while the training time increased drastically with the number of layers. The LSTM and BiLSTM layers yielded similar results. This applies for both one cell and multiple cells combined. Results specific for either one cell or multiple combined is given in Section 5.7 and 5.8.

Balancing Data Sets

An overview over the number of sessions per angle bin for an angle bin width of 10° is given in Figure 5.10. Here, the angle bins are presented along the x-axis while

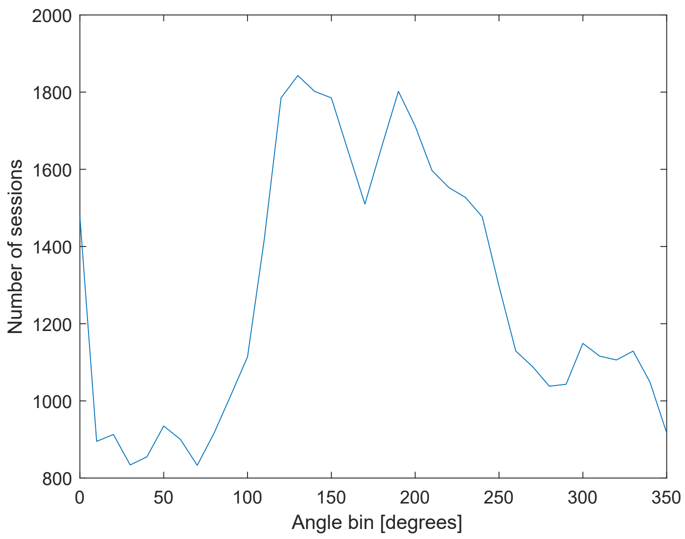


Figure 5.10: Number of sessions per angle bin for 120806.

the y-axis gives the number of sessions per bin. As can be seen in the figure, the number of sessions in each angle bin is quite imbalanced. The difference between the angle bin with most sessions and the one with least sessions is 1010 sessions.

If the data set was not balanced beforehand, the angle bins with the most sessions available had a tendency to be predicted more often than other angle bins.

Training

A screenshot of the accuracy during a training process is shown in Figure 5.11. Each gray and white column is one epoch. For most of the training done with different NNs and hyperparameters, the accuracy increases during an epoch, before dropping at the beginning of the next epoch. This rapid drop in accuracy is not present when the sessions are not sorted by length.

Confusion Matrix

The output confusion matrix visualizes the performance of the neural net. Along the x-axis are the predicted angle bins for the sessions, while the y-axis denotes the true angle bins. In the intersections are the number of sessions predicted as

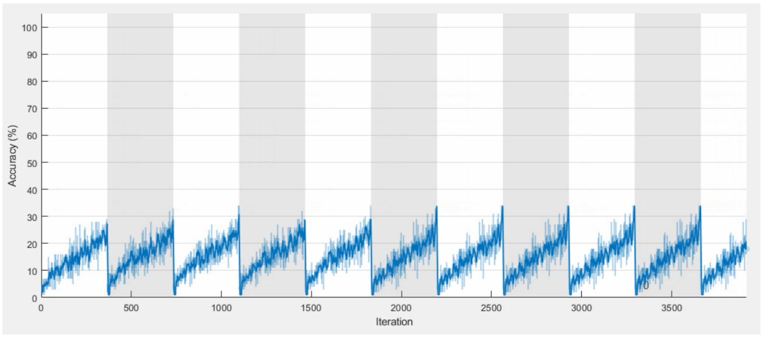


Figure 5.11: Training accuracy.

belonging to the x-value, while actually being from the y-value e.g. the intersection of x-value 140 and y-value 180 is the number of sessions from angle bin 180° classified as from angle bin 140° by the NN. This is for ease visualized with colors. Orange/red is used for false predictions, while blue visualizes the correct predictions. The darker the color, the more predictions there are for that specific tile. The correct predictions go as a diagonal from the upper left corner to the lower right corner.

5.7 One HD Cell

None of the combinations of layers and hyperparameters provided any acceptable results, as no specific patterns can be observed when looking at the confusion matrices. An example is shown in Figure 5.12. When only looking at angle bins within the directional firing range, as shown in Figure 5.13a, or for half of the directional firing range, as shown in Figure 5.13b, to account for symmetry, the results were similarly non conclusive as no clear patterns were found. The results presented are from HD cell 120806.3.8, but all HD cells yielded similar results.

5.8 Multiple HD Cells

When looking at the number of HD cells available for each recording day, as well as the distribution of PFD from Figure 5.3 and Appendix C.1, the recordings from day 120806 appears to be most suitable. This is partly because of the large number of HD cells available compared to 120807 and 120808, as well as the distribution of the cells compared to 120809. The length of the wake phase is also longest for this day, as seen in Table 5.1.

Results from two different NNs are given below. The hyperparameters and specifications are given in Table 5.14, along with the training time and accuracy. Values not mentioned in the table are given their default values. Both the NNs consist of one BiLSTM layer as well as one fully connected layer. The confusion matrices are

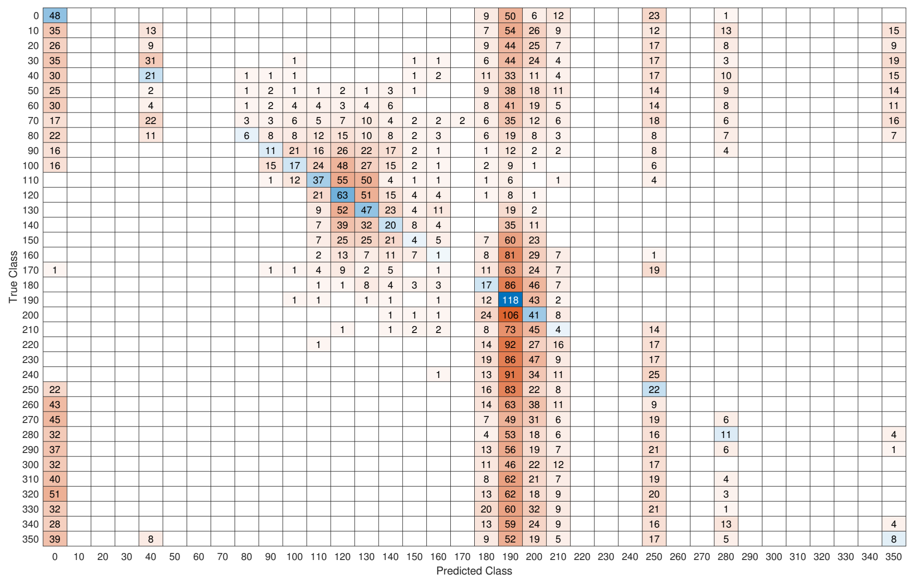
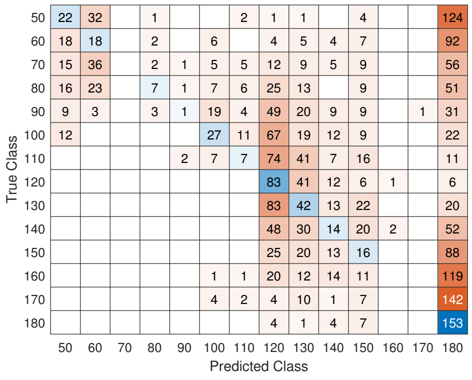
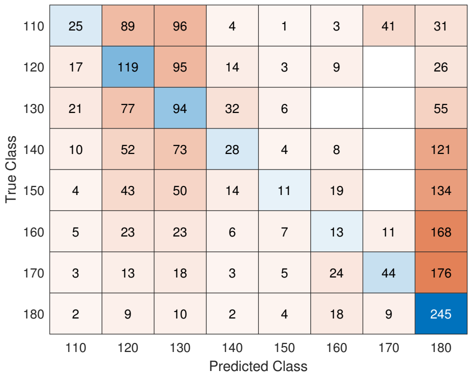


Figure 5.12: Confusion matrix; one HD cell.



(a) Direction firing range.



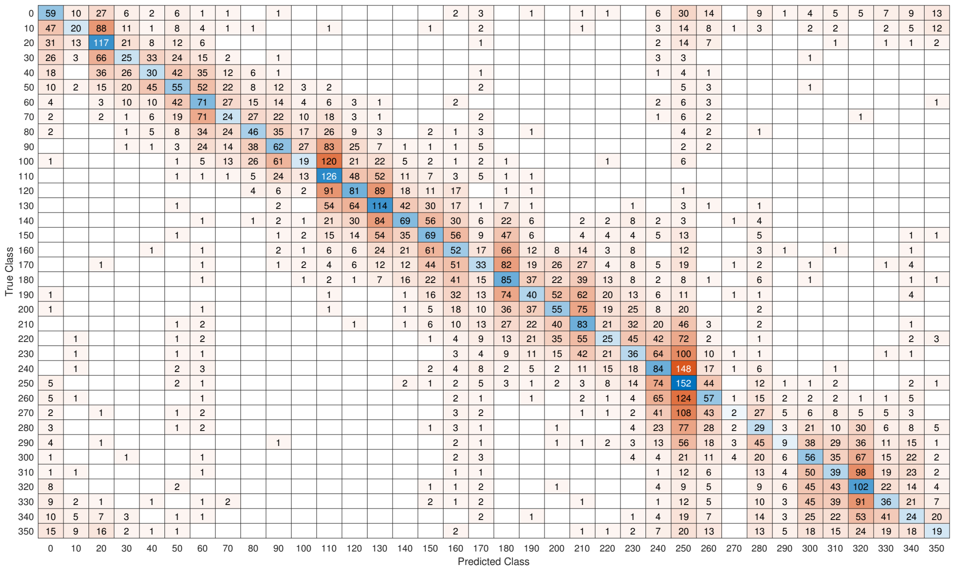
(b) Half direction firing range.

Figure 5.13: Confusion matrices; one HD cell, with sifted data.

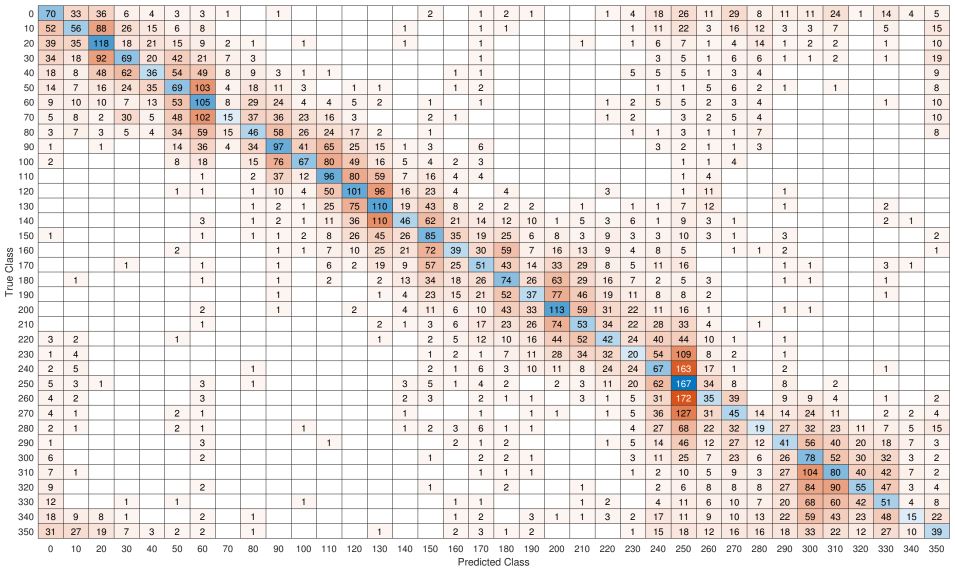
given in Figure 5.14. Both the confusion matrices outline a distribution around the true value as a blue and orange band from [0,0] to [360,360], with most values contained within the true angle bin ±30°.

For both confusion matrices, there are some values that are predicted more frequently. Some angle bins, e.g. the 250°, are predicted more often quite independent of the hyperparameters chosen.

The band are low (often three or less per tile) for both. However, NN_2 contain more of these tiles. If the angle bin is chosen completely random when classifying,



(a) NN_1



(b) NN_2

Figure 5.14: Confusion matrices; multiple HD cells.

the accuracy would be 2.8%. Which is found by taking 1/36. The accuracy of the NN giving the confusion matrix in Figure 5.14a and 5.14b are 2.9% and 2.8% respectively, which is only marginally larger. Some of the other NNs tested gave an accuracy up to around 3.5%, but when looking at the confusion matrix, there are less clear patterns.

Table 5.14: Hyperparameter values and alterations.

Hyperparameter	NN_1	NN_2
Initial learn rate	0.01	0.01
Mini-batch	32	128
Max epoch	100	100
Drop factor	0.5	0.5
Drop period	30	20
Neural net		
No. of hidden units	200	300
LSTM/BiLSTM	BiLSTM	BiLSTM
Data set alterations		
Sorted?	Yes	Yes
Balanced?	Yes	Yes
Short sessions removed?	<20 ms	No
Results		
Accuracy [%]	2.9	2.8
Training time [min]	343	373

Feature creation

The response from two and two cells are combined to create new features. They are combined using the OR operator, and they are combined in the order 1|2, 3|4, ..., N|N+1. For uneven numbers of available HD cells, the last HD cell is not combined with any. The combined features are added to the existing spike trains. The results do not have any notable differences when adding new features.

5.9 Combining Models and Neural Nets

Table 5.15 shows an overview over the number of HD cells per angle bin with at least one session fulfilling the criterion placed on the sessions to contain either five or ten spikes, i.e. the number of HD cells per angle bin with at least one model. The NN used is NN_1 from Table 5.14.

When looking at the number of sessions fulfilling the criteria for each HD cell, there are large differences between HD cells. Some HD cells have a large number of sessions, while others have almost none. An example is HD cell 17, which have sessions fulfilling the criterion of a minimum of five spikes for 25 angle bins. In several cases, HD cell 17 is the only HD cell that has sessions fulfilling the criterion, e.g. angle bin 260° to 330°.

The models are created from sessions with minimum five spikes, as opposed to ten, to have more models available. The input current to the models varies with ±0.5A from the "optimal" value. The number of sessions created with the models

Table 5.15: The number of HD cells per angle bin with at least one model.

Angle bin	**0°**	**10°**	**20°**	**30°**	**40°**	**50°**	**60°**	**70°**	**80°**	**90°**	**100°**	**110°**
≥5 spikes	2	4	4	3	5	6	6	8	8	6	12	9
≥10 spikes	1	2	2	3	3	3	4	4	6	5	7	3

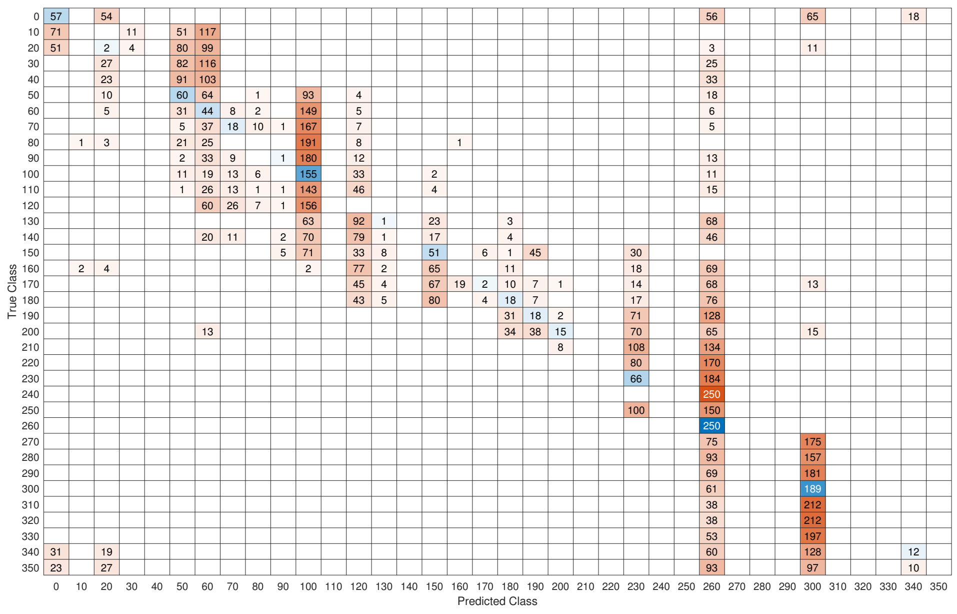
Angle bin	**120°**	**130°**	**140°**	**150°**	**160°**	**170°**	**180°**	**190°**	**200°**	**210°**	**220°**	**230°**
≥5 spikes	11	11	13	12	11	9	12	8	6	7	5	4
≥10 spikes	5	6	9	5	3	3	2	3	2	2	2	1

Angle bin	**240°**	**250°**	**260°**	**270°**	**280°**	**290°**	**300°**	**310°**	**320°**	**330°**	**340°**	**350°**
≥5 spikes	3	2	1	1	1	1	1	1	1	1	2	2
≥10 spikes	1	0	0	1	1	1	1	1	1	1	1	1

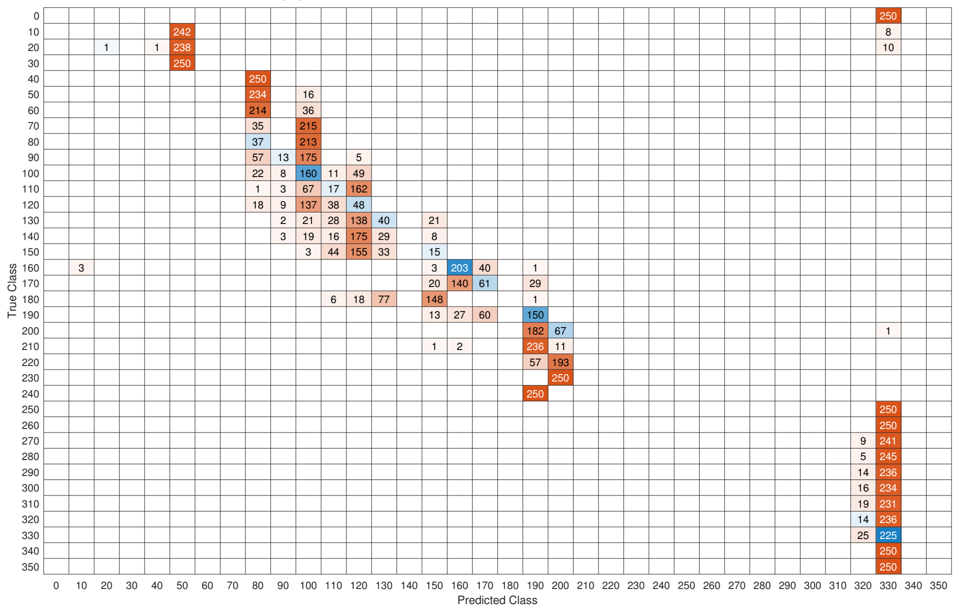
is 250 per angle bin. This number is chosen to make them somewhat comparable to the ones in Section 5.14, which are tested on 9173 sessions when short sessions are kept. Different session lengths were tried. The confusion matrices for sessions with lengths of 5 ms and 500 ms are shown in Figure 5.15. A length of 5 ms was the minimum length to be able to get results that did not consist of only one prediction. In this case, the angle bin 260° is predicted several times, for almost all true angle bins. There are several other angle bins that are predicted more often than others, such as 100° and 300°. These angle bins are mostly predicted when the true angle bin is within ±50° of the predicted angle bin.

For a session length of 500 ms, not all angle bins are predicted, but there are no angle bins that are predicted for all different true angle bins, as 260° in Figure 5.15a. For both confusion matrices, almost all predictions are within ±50° from the true angle bin. The same trends are visible for other session lengths than 5 ms and 500 ms.

Figure 5.16 shows the confusion matrix when using input from models based on recorded sessions with a length of minimum 250 ms. The number of sessions used for testing is 250 per angle bin. The length of the input sessions created from the models is 500 ms. There are more sessions longer than 250 ms than sessions with more than five spikes, and longer sessions take more time to compare with the Victor-Purpura distance than short sessions. Due to time limitations, only two models were found from each HD cell per angle bin. As one can see from the confusion matrix, the results are somewhat similar to the ones seen in Figure 5.15b. The main difference between the two confusion matrices is that there are more different angle bins predicted in Figure 5.16. When testing the network on shorter sessions, like 250 ms, the results are similar to the ones seen in Figure 5.16.



(a) Session length of five milliseconds.



(b) Session length of 500 ms.

Figure 5.15: New input in neural net NN_1. Input based on sessions with more than five spikes.

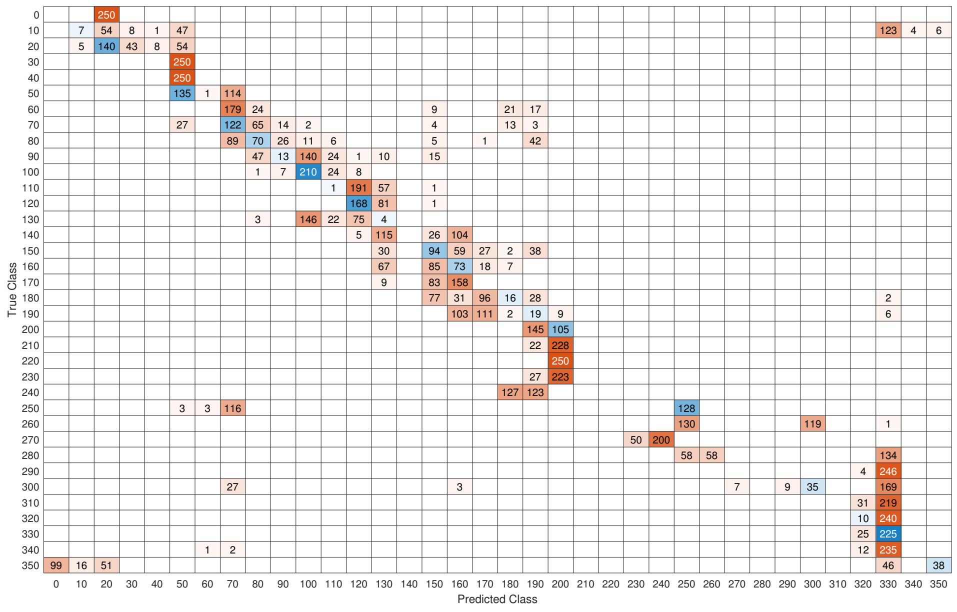


Figure 5.16: New input in neural net NN_1. Input based on sessions with a minimum length of 250 ms.

6 | Discussion

6.1 Initial Analysis

Extracting HD Cells

The percentage of HD cells found is approximately half of the percentage found in the original paper [52] (15% versus 33%). Due to restrictions in the data set, the extraction method described in this paper is not possible to use on the rest of the data set without further preprocessing steps. Since not all mice and days are analyzed, it is not possible to say whether or not the code classifies the same number of cells as HD cells as the one in the original study.

Tuning curves

After a brief inspection of the HDCEC, no explanation is found why some of the cells with more than one high peak were classified as HD cells. Given more time, the code should probably be more thoroughly investigated to see if an explanation can be found and if there are any other potential issues with the code. This could, for instance, be if any HD cells are not classified as HD cells.

Visual Inspection of Tuning Curves

The directional firing range and the peak firing rate of the HD cells found in Table 5.3 are consistent with the literature. The directional firing range is similar to published literature discussed in Section 2.3, while the peak firing rate is a bit lower for two of the HD cells, with a firing rate below 5. When it comes to the results for the rest of the days, some of the directional firing ranges are a bit larger than the ones found in literature, and some of the peak firing rates are a bit lower, with peak firing rates down to below 1.5 spikes per second.

The exact values given in the results should, however, be interpreted with caution. The directional firing range and peak firing rate were only estimated from a visual inspection of the tuning curves. This was determined to be accurate enough for the simple, initial analysis conducted. For a more thorough analysis, more accurate values should be used. One method for finding these values could be to estimate a fitted curve, before calculating the directional firing range and peak firing rate from this.

Extraction of Sessions

Comparison of methods

The tuning curves from the HDCEC and the ones based on the sessions extracted are quite similar. There are some differences between them, but the ones from the HDCEC are smoothed. This shows that dividing the recordings into sessions do not have a large impact on the properties.

Size of angle bins

The significance of the angle bin width with regards to the variations of firing rate (and other possible types of coding) inside the angle bin becomes evident when looking at the difference in firing rate between the two extremities of an angle bin. Even though HD cell 120806.8.3 is only one example of the firing rates for each extremity, it illustrates that the difference is large enough to pose a problem. How wide an angle bin is depends on the directional firing range and the peak firing rate, and what is wide for one HD cell may not be of relevance to another HD cell. When looking at sessions originating from the same angle bin, the results over time may not be that different when using wide angle bins compared to narrow ones. This can be seen when looking at the tuning curve in Figure 5.5 based on an angle bin width of 10°. When looking at sessions individually, these differences may be substantial.

As seen in the results, in Table 5.4 and Appendix C.2, the number of sessions available is drastically reduced when increasing the angle bin width. This is as expected since larger head movements more often stay within the same angle bin.

Length of extracted sessions

A large portion of the extracted sessions are short. Many of these sessions are shorter than 30 ms even for “large” angle bins with a width of 10°. Individually, these sessions do not contain much information. When taking an HD cell with a peak firing rate of 60 spikes per second, which is larger than the majority of the HD cells found, 30 ms will on average contain less than two spikes. This is quite problematic when it comes to both the analysis of single sessions and combinations of sessions using machine learning. Looking at sessions individually, almost all sessions from 1° angle bins are too short for analysis, while for 10° angle bins, some sessions are sufficiently long. The problem with larger angle bins is, as mentioned before, that it is harder to pinpoint the exact angle of the head, and it is not possible to detect small variations in head direction or types of coding used.

For future work, an attempt if combining sessions from different angle bins to make longer sessions available for analysis should be explored.

Spikes per session

Several factors are contributing to the number of spikes in the different sessions. This becomes evident from Tables 5.5 through 5.8, and will be discussed further.

When it comes to the choice of angle bin width, the narrower the angle bins, the higher the SD. This can be explained by more short sessions, with more spikes occurring in short sessions and thus increasing the SD.

By removing all sessions with spikes less than a minimum number of spikes, the sessions with no spikes dominate, and the mean decreases drastically. The SD also decreases. This is most likely due to the short sessions with artificially high average spiking rates are removed. Neither the mean nor the SD is representative for the actual mean and SD for the angle bin.

Exempting sessions shorter than a minimum length, as exemplified in Table 5.7 and 5.8, some of the problems with including all sessions disappear, such as an artificially large SD. However, by removing short sessions, the number of sessions also reduces drastically due to the large number of short sessions. The mean is close to the average firing rate that can be read from the tuning curves. Even by removing all sessions less than 100 ms, the SD is quite large compared to the mean.

The large SD could also originate from large angle bins, for example, if sessions are found from different "parts" of the angle bin. For an angle bin between 5° and 15°, one session may originate only from recordings close to 5°, and another from angles close to 15°. If the width of the angle bin is large compared to the directional firing range, the average firing rate will differ significantly between the two sessions.

All methods used, apart from removing short sessions, shifts the ratio between spikes and non-spikes. Additionally, the longer the sessions, the closer the average is to the "true" average spike rate. The drawback with using only long sessions is that the number of sessions available diminishes fast with increasing minimum length.

The large SD may suggest that HD cells also communicate with a different coding scheme than only rate coding.

6.2 Time Between Spikes

As mentioned in Section 4.4, a lot of information about the time between spikes is excluded when dividing the recordings into angle bins. One distance between spikes is lost with every change of angle bin. The narrower the angle bins, the more sessions there are, and the more information is lost. Still, the results found give some indication on whether or not the distance between spikes could be a type of coding used by HD cells.

If the HD cells only communicate with rate coding, and with the spikes evenly spaced, the curves of time between spikes should have the form of the inverse of the tuning curves. When looking at the plots in Figure 5.7, although far from conclusive, there are hints of such an inverse tuning curve in both Figures 5.7b and 5.7c.

The immediate difference between SD and mean around 45° and 225°, as opposed to around 100°, could be that the time between spikes varies more around 45° and 225°. Another explanation is that the number of measurements of time between spikes is lower, due to fewer spikes at the outskirts of the directional firing range, so that the larger variations have more impact on the SD.

6.3 Comparison of Sessions

By looking at several arbitrary sessions; some from the same angle bin, and some from different angle bins, it is not possible to say which ones originate from the same angle bin by looking at the Victor-Purpura distance, the van Rossum distance or the average firing rate.

There are several possible explanations for this: Spike train metrics are often used on repeated trials to see the trial-to-trial variation. The problem with the data set used is that there are no repeated trials, and the lack of these trials creates the need for breaking the recordings into comparable pieces. However, since these sessions varies more than "normal" trial-to-trial variations, metrics such as the Victor-Purpura distance and the van Rossum distance are in these cases not the most optimal to use.

The size of the angle bins is crucial for further analysis. The bins have to be of a minimum width in order to contain enough data for proper analysis, but too wide bins provide a problem if the HD cells use a "smaller resolution". If an angle bin width of 10° is used, but the HD cells code directions that differ with 1° differently, sessions from the same angle bin are not directly comparable if the movement inside the angle bin differs.

Both the Victor-Purpura distance and the van Rossum distance use a parameter to decide the time scale. As seen in Section 5.3, the average between spikes is not constant. Thus, there is no single optimal value for the parameters q and τ. Looking at recordings from all angle bins for a specific HD cell poses several problems with this method. One of these problems is that recordings from outside the directional firing range contain less information than from the ones inside the directional firing range. If these are coded in the same way, the percentage of sessions classified as from the same angle bin will decrease. When looking at the tuning curve, the peak is quite symmetric. If the same holds for the spike patterns, the percentage may also drop since a session may be classified as originating from the "symmetric" angle bin. By looking only at recordings from angle bins from half the directional firing range, it can give some indications if this is true. As seen in the results, the percentage of comparisons with the shortest distance originating from the same distance increases when only looking at half of the directional firing range. The percentage is still not large enough to be able to say anything about which sessions are from the same angle bins when looking at a random selection of sessions. A possible explanation of the increase in percentage is that the number of comparisons from other sessions compared of sessions from the same session is reduced with the reduced number of possible angle bins.

It is curious that the average firing rate also yields poor results compared to the Victor-Purpura and the van Rossum distances, as it is known that rate coding is one of the types of coding used for HD cells. These explanations also hold when comparing sessions using average firing rate, but there are additional explanations for this. After removing half of the angles, the percentage does not increase to more than about 30%. When looking at this result in comparison to the results in Table 5.7 and 5.10, this is not very surprising due to the large SD.

The spike train metrics used are metrics that take the spikes as the main method of measurement. This is done since HD cells are known to communicate with rate coding, if not more types of coding as well, and thus provide a good starting point. There are also time-based metrics, which is a possible area of investigation for the future.

6.4 Finding a Model

When looking at the plotting of session recordings and the model given by the model parameter values yielding the shortest Victor-Purpura distance, some of the plots are visually quite similar. This shows that the Izhikevich model can reproduce, or mimic, some of the spike patterns reproduced by the HD cells. However, this does not mean that the Izhikevich model is suitable for describing the neural response of HD cells. How many sessions the model is able to describe well has not been fully investigated. Similarities between recordings and the model output has not been compared using other metrics, which should be done in the future.

As seen in Tables 5.11 and 5.12, the model parameter values that yield the shortest distance for one session are different for other sessions within the same angle bin. The optimal choice of parameter values for one session changes throughout the angle bin. The large difference in distance when using the optimal model parameter values for one session for other sessions in the same angle bin indicates that the differences are more than just trial-to-trial variations. It could also be an indication of that the combination of metrics and models is unsuitable for this application. It is not possible to prove that the Izhikevich model cannot be used, but the results so far are not very promising. If the Izhikevich model is to be used, the method used to optimize the model parameters needs to change.

Another approach to finding a model is to calculate the distance between the model and all sessions available for a specific angle bin and choose the one with the lowest total distance. This way, the model found is the one best approximated to the "average" model. However, since no clear connection between the sessions from the same angle bin is found, as seen in Section 6.3, a thorough analysis should be conducted to see if this is a feasible approach to finding a model.

6.5 Machine Learning

Training

The rapid decrease in training accuracy at the beginning of each new epoch is due to the sorting of the sessions by length. Short sessions are notably harder to classify than the longer sessions, and the input is not shuffled before each epoch.

Evaluating Performance

The accuracy is not necessarily the best way to measure the performance of the NNs. The accuracy was low for all tests, and for several of the tests, the accuracy was almost as low as a random distribution. When looking at the confusion matrix, a pattern around the true value is often evident for multiple HD cells. There was no visible correlation between the accuracy and the patterns in the confusion matrix.

For future work, a measurement for an additional accuracy for predictions within the "band" should be incorporated in the code. Thus, reducing the need for visual inspections.

6.6 One HD Cell

For all confusion matrices found when looking at a single HD cell, it quickly becomes evident that there is a lack of data for the ML process. As exemplified in Figures 5.12, 5.13a and 5.13b, no pattern appear, and the accuracy is extremely poor.

When looking at the length of the sessions used for training and testing, visualized in Figure 5.6b, the length of the sessions is in most cases less than 30 ms. These sessions do not contain more info than their own randomness, and when excluding them, the sessions left are too few for training a NN.

It is not possible to tell the direction of the head by looking at one HD cell alone. However, it is also not possible to dismiss the possibility of being able to tell the direction by considering the spike trains from only one cell, for example, by looking at data sets containing longer sessions. Another possibility could be to combine sessions from similar angle bins, but this may introduce new problems, which should be considered.

6.7 Multiple HD Cells

When combining the response from multiple HD cells, the accuracy is still poor, but a pattern is often evident around the true values when looking at the confusion matrices. The band illustrates that the NNs can distinguish between angles with an accuracy poorer than the angle bin width.

Some of the angle bins are predicted more often than others, even when changing the hyperparameter values. The number of sessions per angle bin is balanced before division into training and test sets. This is therefore not the reason for the number of predictions, as it also occurs for training with several different hyperparameters. Due to the persistence, a reason can be the combination of HD cells available.

One explanation for NN_2 having more different predictions outside of the band is the choice of hyperparameter values. This is evident from the direct impact hyperparameter values have on the predictions. Also, NN_1 excluded sessions less than 20 ms, which means that the number of sessions used in testing is less than for NN_2. For future work, the number of predictions inside such a band should be calculated and compared to the number outside.

From the results, it is not possible to say which angle bin the sequence originates from. Whether or not these results are acceptable depends on the application. If these, or similar, results are going to be used in applications for finding a "general" direction, or distinguish between four main directions, these findings may be sufficient, while it is not usable for an application requiring higher accuracy. It could be that the method works better with a more extensive data set, with more or better distributed PFDs.

Even if the classification was perfect, one is still dependent on dividing angle bins into sessions. It is possible to investigate use of sequence-to-sequence classification, which enables distinguishing between multiple angle bins within a session.

Feature Creation

Feature creation was not explored to the necessary extent, and should be investigated further in the future. The results did not improve notably when combining the response from two and two consecutive HD cells, but it could be that another selection of combined HD cells provide better results. This could be done by considering the properties of the individual HD cells available in the data set, combining response from two or more HD cells with complementing properties.

6.8 Combining Models and Neural Nets

When considering sessions with minimum five, opposed to ten, spikes as the criterion when finding sessions, the number of sessions to base the models on is higher, and therefore yields more models. Still, out of 36 angle bins, only seven include models for more than half of the 19 HD cells. This means that the number of sessions to test on, that consists of only zeros, is quite high.

It is expected that not all HD cells have sessions with more than five spikes for all angle bins, as some HD cells have low peak firing rates. Combined with short sessions, the number of spikes per sessions is often less than the criterion of a minimum of five spikes. For angles outside the directional firing range, the sessions need to be longer than inside the directional firing range to include a sufficient number of spikes. When looking at Figure 5.3 in combination with Table 5.15, one can observe that the angle bins with the largest number of HD cells, with at least one model, are the ones in the directions where the mouse held its head the most. These are also the directions with a higher density of PFDs of HD cells.

Compared to the testing of the NN with recordings, the number of predictions per predicted value is higher. This is visualized with darker colors in the confusion matrices. A reason for this could be the low number of non-zero sessions available, which means that the precise form of the spiking sequences has less impact on the classification than the number of spikes.

The issues with classifying short sessions can explain the larger number of predicted values for sessions with a length of five milliseconds.

There are several problems and challenges related to using the Izhikevich model to find applicable inputs for the NN. One is that the models are based on some of the sessions that were used for training the NN, which may lead to the results being better than they would have otherwise. Another problem is the length of the sessions. It is not tested whether or not the models found preserve the properties for session lengths longer or shorter than the sessions they were derived from.

When it comes to the many predictions of angle bin 330° for a session length of 500 ms, this can in part be explained by looking at the models used to create the sessions. For angle bin 250° through 350° plus 0°, there are models from either one or two HD cells available.

When it comes to the numerous models for HD cell 17, it is not very surprising that the number of sessions fulfills that criterion, as the HD cell has a peak firing rate of nearly 80 spikes per second. However, being the lone HD cell with sessions fulfilling the criterion is somewhat problematic. One reason is that the angle bins are classified solely on the models from the response of one cell, which earlier has been shown not to contain enough information. Almost all of the models are similar enough for the NN to classify them as originating from angle bin 330°.

Using a minimum session length of 250 ms as the criterion for sessions used to find a model, more different angle bins are predicted. This can be explained by the number of models available when creating the input to the NN. When using a minimum number of spikes as a criterion, only a few HD cells had sessions fulfilling this, while all of them have at least two sessions longer than 250 ms. Based on the results from Table 5.15, most of the models found are based on sessions with less than five spikes, which means that the models are based on spikes that could originate solely from random noise.

When it comes to angle bin 330°, there are still many predictions. The explanation behind this is not completely clear, but a reason could be the low number of models available per angle bin and HD cell. If the two models available yield low spike numbers for all sessions except 17, and HD cell 17 have models with more spikes, the explanation for the high number of predictions could, therefore, be the same as above.

7 | Conclusion

The scope of this project was to investigate the possibilities of finding a way to describe the response of HD cells using existing spiking neuron models, and artificial neural network models to describe the firing patterns that can be classified to a set of directions. The latter one using long short-term architecture commonly used for classifying sequences.

By dividing the data set into sessions, some of the problems related to the free movement of the mouse are removed. However, this partitioning places several limitations on the methods used and affects all parts of the work. Even with these limitations, this extraction of sessions provide a way to look at the neural responses of the HD cells at a spike train level and investigate their properties.

The Izhikevich model shows promise for describing individual spiking sequences, but has not yet been proven useful for more generalized models describing the firings of HD cells. The lack of a generalized model is not due to the Izhikevich model itself, but due to limitations in the methods. The possibility of describing the neuronal response of HD cells using the Izhikevich model still exists, but another approach should be taken to investigate this potential further.

For the investigated data, one HD cell does not contain enough information to distinguish between different angles when using the machine learning methods described. When combining the response from multiple HD cells, it is possible to distinguish between some angles with a resolution of $10° \pm 30°$, and future work may improve this further.

In general, the data in the data set is not sufficient when using the methods in this study. If the same, or similar methods is to be used again, either more data is needed, or the data available needs to be used differently, for example by combining sessions or looking at the waveforms. The latter does, however, require more preprocessing.

8 | Future Work

Mentioned below are a few possible next steps in the work of finding a model capable of describing the response of HD cells.

The method of finding a neural network able to classify the head direction based on spike patterns is quite time consuming, and have not yet yielded results good enough for further processing. Before attempting several more combinations of hyperparameters, other days in the data set should be considered for analysis instead, as some of them may provide more available HD cells, more and longer sessions and/or a better distribution of the PFDs of the HD cells.

It would be interesting to investigate if the periods of sleep in the data set could be used for the same analysis as conducted for the awake phases. Being able to use these sleep periods, the amount of data available greatly increases. In addition to more data due to a longer period of recordings (four hours instead of 35-40 minutes), there would also most likely be less rapid changes of the head direction during sleep. It could be that not all 360° are represented, but sessions extracted would potentially be much longer than the ones currently available.

Another idea to explore, which was not tested due to time limitations, is combining shorter sessions into longer ones instead of omitting them from further processing. This way, no data is lost. However, it should be investigated if multiple short sessions combined possess similar properties as long sessions from the same angle bin.

In addition to spike times, the data set includes waveform of the spikes. Including this information in the training and test data set could give new correlations or insight.

Bibliography

[1] Mladen Veletić. *On the Neural Communication for Data Transmission in Nano-Networks*. PhD thesis, NTNU, 2017.

[2] Miller-Keane Encyclopedia, Nursing Dictionary of Medicine, and Seventh Edition Allied Health. neurological disorder. (n.d.). https://medical-dictionary.thefreedictionary.com/neurological+disorder (Last accessed 06.06.19), 2003.

[3] Miller-Keane Encyclopedia, Nursing Dictionary of Medicine, and Seventh Edition. Allied Health. nervous system. (n.d.). https://medical-dictionary.thefreedictionary.com/nervous+system (Last accessed 06.06.19), 2003.

[4] World Health Organization. Neurological Disorders: Public Health Challenges - World Health Organization. *World Health Organization*, 2006.

[5] Marc Possover. The laparoscopic implantation of neuroprothesis to the sacral plexus for therapy of neurogenic bladder dysfunctions after failure of percutaneous sacral nerve stimulation. *Neuromodulation*, 13(2):141–144, 2010.

[6] Dennis J. McFarland and Jonathan R. Wolpaw. Brain-computer interface operation of robotic and prosthetic devices. *Computer*, 41(10):52–56, 2008.

[7] Theodore W. Berger, Robert E. Hampson, Dong Song, Anushka Goonawardena, Vasilis Z. Marmarelis, and Sam A. Deadwyler. A cortical neural prosthesis for restoring and enhancing memory. *Journal of Neural Engineering*, 8(4), 2011.

[8] Karen A. Moxon and Guglielmo Foffani. Brain-machine interfaces beyond neuroprosthetics. *Neuron*, 86(1):55–67, 2015.

[9] Jeffrey S. Taube, Robert U. Muller, and James B. Ranck Jr. Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420–35, 1990.

[10] Jeffrey S. Taube, Robert U. Muller, and James B. Ranck Jr. Head-direction cells recorded from the postsubiculum in freely moving rats. II. Effects of environmental manipulations. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 10(2):436–47, 1990.

[11] Longtang L. Chen, Lie Huey Lin, Edward J. Green, Carol A. Barnes, and Bruce L. McNaughton. Head-direction cells in the rat posterior cortex - I. anatomical distribution and behavioral modulation. *Experimental Brain Research*, 101(1):8–23, 1994.

[12] Jeffrey S. Taube. Head Direction Cells Recorded in the Anterior Thalamic Nuclei of Freely Moving Rats. *The Journal of Neuroscience*, 15(January):70–86, 1995.

[13] Robert W. Stackman and Jeffrey S. Taube. Firing Properties of Rat Lateral Mammillary Single Units: Head Direction, Head Pitch, and Angular Head Velocity. *The Journal of Neuroscience*, 18(21):9020–9037, 2018.

[14] Francesca Sargolini, Marianne Fyhn, Torkel Hafting, Bruce L. McNaughton, Menno P. Witter, May-Britt Moser, and Edvard I. Moser. Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science*, 312(MAY):758–762, 2006.

[15] Faculty Research Development Office (FRDO) University of New Mexico. Psychology Professor Wins Alzheimer's Association Research Grant (AARG). http://frdo.unm.edu/?q=content/psychology-professor-wins-alzheimer%E2%80%99s-association-research-grant-aarg (Last accessed 19.06.19).

[16] A. Peyrache and G. Buzsáki. Extracellular recordings from multi-site silicon probes in the anterior thalamus and subicular formation of freely moving mice. CRCNS.org. http://dx.doi.org/10.6080/K0G15XS1, 2015.

[17] Lisa M. Giocomo, Tor Stensola, Tora Bonnevie, Tiffany Van Cauter, May-Britt Moser, and Edvard I. Moser. Topography of head direction cells in medial entorhinal cortex. *Current Biology*, 24(3):252–262, 2014.

[18] Gary M. Muir and Jeffrey S. Taube. The neural correlates of navigation: do head direction and place cells guide spatial behavior? *Behavioral and cognitive neuroscience reviews*, 1(4):297–317, 2002.

[19] John C. Baird, Jeffrey S. Taube, and Damen V. Peterson. Statistical and information properties of head direction cells. *Perception and Psychophysics*, 63(6):1026–1037, 2001.

[20] Patricia E. Sharp, Hugh T. Blair, and Jeiwon Cho. The anatomical and computational basis of the rat head-direction cell signal. *Trends in Neurosciences*, 24(5):289–294, 2001.

[21] Sidney I. Wiener, Alain Berthoz, and Michaël B. Zugaro. Multisensory processing in the elaboration of place and head direction responses by limbic system neurons. *Cognitive Brain Research*, 14(1):75–90, 2002.

[22] Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304, 1999.

[23] L F Abbott and Thomas B Kepler. Model neurons: from Hodgkin-Huxley to Hopfield. In *Statistical mechanics of neural networks*, pages 5–18. Springer, 1990.

[24] Eugene M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.

[25] Neil Burgess and John O'Keefe. Models of place and grid cell firing and theta rhythmicity. *Current opinion in neurobiology*, 21(5):734–744, 10 2011.

[26] P Vance, S A Coleman, D Kerr, G P Das, and T M McGinnity. Modelling of a retinal ganglion cell with simple spiking models. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[27] Christopher J. Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *International Conference on Learning Representations (ICLR)*, 2018.

[28] Ardi Tampuu, Tambet Matiisen, H Freyja Ólafsdóttir, Caswell Barry, and Raul Vicente. Efficient neural decoding of self-location with a deep recurrent network. *PLOS Computational Biology*, 15(2), 2019.

[29] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018.

[30] Xiaomao Zhou, Tao Bai, Yanbin Gao, and Yuntao Han. Vision-based robot navigation through combining unsupervised learning and hierarchical reinforcement learning. *Sensors*, 19(7), 2019.

[31] Nicholas J. Gustafson and Nathaniel D. Daw. Grid cells, place cells, and geodesic generalization for spatial reinforcement learning. *PLoS Computational Biology*, 7(10), 2011.

[32] A. Arleo, F. Smeraldi, S. Hug, and W. Gerstner. Place Cells and Spatial Navigation based on Vision, Path Integration, and Reinforcement Learning. *Advances in Neural Information Processing Systems 13*, pages 89–95, 2001.

[33] Arthur Prochazka, Vivian K Mushahwar, and Douglas B Mccreery. Neural prostheses. (November 2000):99–109, 2001.

[34] Fan-Gang Zeng. Trends in Cochlear Implants. *Trends In Amplification*, 8(1), 2004.

[35] Gislin Dagnelie. Retinal implants: Emergence of a multidisciplinary field. *Current Opinion in Neurology*, 25(1):67–75, 2012.

[36] Leigh R. Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y. Masse, John D. Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S. Cash, Patrick Van Der Smagt, and John P. Donoghue. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

[37] D. J. Guggenmos, M. Azin, S. Barbay, J. D. Mahnken, C. Dunham, P. Mohseni, and R. J. Nudo. Restoration of function after brain damage using a neural prosthesis. *Proceedings of the National Academy of Sciences*, 110(52):21177–21182, 2013.

[38] Jonas B. Zimmermann and Andrew Jackson. Closed-loop control of spinal cord stimulation to restore hand function after paralysis. *Frontiers in Neuroscience*, 8((MAY)), 2014.

[39] Max O. Krucoff, Shervin Rahimpour, Marc W. Slutzky, V. Reggie Edgerton, and Dennis A. Turner. Enhancing nervous system recovery through neurobiologics, neural interface training, and neurorehabilitation. *Frontiers in Neuroscience*, 10(DEC), 2016.

[40] Mikhail A Lebedev and Miguel A L Nicolelis. Brain – machine interfaces : past , present and future. *Trends in Neurosciences*, 29(9), 2006.

[41] John K Chapin, Karen A Moxon, Ronald S Markowitz, and Miguel A L Nicolelis. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature Neuroscience*, 2(7):664–670, 1999.

[42] Eva Bianconi, Allison Piovesan, Federica Facchin, Alina Beraudi, Raffaella Casadei, Flavia Frabetti, Lorenza Vitale, Maria Chiara Pelleri, Simone Tassani, Francesco Piva, Soledad Perez-Amodio, Pierluigi Strippoli, and Silvia Canaider. An estimation of the number of cells in the human body. *Annals of Human Biology*, 40(6):463–471, 11 2013.

[43] Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* MIT Press, 2001.

[44] Eric R. Kandel, James H. Schwartz, Thomas M. Jessell, Steven A. Siegelbaum, and A. J. Hudspeth. *Principles of neural science*, volume 4. McGraw-hill New York, fifth edition, 2000.

[45] Jaakko Malmivuo and Robert Plonsey. *Bioelectromagnetism - Principles and Applications of Bioelectric and Biomagnetic Fields.* 1 1995.

[46] Wulfram Gerstner and Werner M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity.* Cambridge University Press, 2002.

[47] Miller-Keane Encyclopedia, Nursing Dictionary of Medicine, and Seventh Edition. Allied Health. axosomatic synapse. (n.d.). https://medical-dictionary.thefreedictionary.com/axosomatic+synapse (Last accessed 12.05.19), 2003.

[48] Jeffrey S. Taube. The Head Direction Signal: Origins and Sensory-Motor Integration. *Annual Review of Neuroscience*, 30(1):181–207, 2007.

[49] E T Rolls, Robertson R G., and P Georges-François. Head Direction Cells in the Primate Pre-Subiculum. *Hippocampus*, 9:206–219, 1999.

[50] Michael E. Shinder and Jeffrey S. Taube. Self-motion improves head direction cell tuning. *Journal of Neurophysiology*, 111(12):2479–2492, 2014.

[51] Edvard I. Moser, Emilio Kropff, and May-Britt Moser. Place Cells, Grid Cells, and the Brain's Spatial Representation System. *Annual Review of Neuroscience*, 31(1):69–89, 2008.

[52] Adrien Peyrache, Marie M. Lacroix, Peter Petersen, and György Buzsáki. Internally-organized mechanisms of the head direction sense. *Nature Neuroscience*, 18(4):569–575, 2015.

[53] Darrell A. Henze, György Buzsáki, Kenneth D. Harris, Hajime Hirase, and Jozsef Csicsvari. Accuracy of Tetrode Spike Separation as Determined by Simultaneous Intracellular and Extracellular Measurements. *Journal of Neurophysiology*, 84(1):401–414, 2017.

[54] A. Scaglione, K. A. Moxon, J. Aguilar, and G. Foffani. Trial-to-trial variability in the responses of neurons carries information about stimulus location in the rat whisker thalamus. *Proceedings of the National Academy of Sciences*, 108(36):14956–14961, 2011.

[55] E. D. Adrian. The impulses produced by sensory nerve endings: Part I. *The Journal of physiology*, 61(1):49–72, 1926.

[56] Martin J. Tovee and Edmund T. Rolls. Information Encoding in Short Firing Rate Epochs by Single Neurons in the Primate Temporal Visual Cortex. *Visual Cognition*, 2(1):35–58, 1995.

[57] Lance M. Optican and Barry J. Richmond. Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. III. Information theoretic analysis. *Journal of Neurophysiology*, 57(1):162–178, 1987.

[58] John O'Keefe and Michael L. Recce. Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus*, 3(3):317–330, 1993.

[59] Stefano Panzeri, Jakob H. Macke, Joachim Gross, and Christoph Kayser. Neural population coding: Combining insights from microscopic and mass signals. *Trends in Cognitive Sciences*, 19(3):162–172, 2015.

[60] Bruno A. Olshausen and David J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, 2004.

[61] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition.* Cambridge University Press, 2014.

[62] Jonathan D. Victor. Spike train metrics. *Current Opinion in Neurobiology*, 15(5):585–592, 2005.

[63] M. C. W. van Rossum. A novel spike distance. *Neural Computation*, 13(4):751–763, 2001.

[64] Conor Houghton and Jonathan Victor. Measuring representational distances – the spike train metrics approach. *Visual Population Codes–Toward a Common Multivariate Framework for Cell Recording and Functional Imaging*, pages 391–416, 2010.

[65] Tom Fawcett and Foster Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[66] Joseph A. Cruz and David S. Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer Informatics*, 2:59–77, 2006.

[67] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, Inc., third edition, 2010.

[68] Enrico Blanzieri and Anton Bryl. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1):63–92, 2008.

[69] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka. Stock market prediction system with modular neural networks. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 1–6, 1990.

[70] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning.* The MIT Press, second edition, 2018.

[71] Tong Wang, Cynthia Rudin, Daniel Wagner, and Rich Sevieri. Learning to Detect Patterns of Crime BT - Machine Learning and Knowledge Discovery in Databases. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 515–530, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[72] Zhenpeng Zhou, Xiaocheng Li, and Richard N. Zare. Optimizing Chemical Reactions with Deep Reinforcement Learning. *ACS Central Science*, 3(12):1337–1344, 2017.

[73] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *NIPS Deep Learning Workshop*, 2013.

[74] Sebastian Ruder. An overview of gradient descent optimization algorithms. http://arxiv.org/abs/1609.04747, 2017.

[75] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[76] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[77] Yoav Freund and Robert E. Schapire. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277 – 296, 1999.

[78] B L Kalman and S C Kwasny. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581, 1992.

[79] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[80] Christopher M. Bishop. *Machine Learning and Pattern Recoginiton.* 2007.

[81] H. Sak, Andrew Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 338–342, 2014.

[82] P Doetsch, M Kozielski, and H Ney. Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 279–284, 2014.

[83] Søren Kaae Sønderby and Ole Winther. Protein Secondary Structure Prediction with Long Short Term Memory Networks. *arXiv preprint arXiv:1412.7828*, 2014.

[84] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2015.

[85] Christopher Olah. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Last accessed 06.06.19), 2015.

[86] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[87] Mike Schuster and Kuldip K Paliwal. Bidirectional Recurrent Neural Networks. 45(11):6757, 1997.

[88] J. D. Victor and K. P. Purpura. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of Neurophysiology*, 76(2):1310–1326, 2017.

[89] Daniel Reich. Matlab Code for Spike Time Distances Between Spike Trains. http://www-users.med.cornell.edu/˜jdvicto/spkdm.html (Last accessed 03.12.18).

[90] Thomas Kreuz, Conor Houghton, and Charles Dillon. Matlab code to calculate the bivariate van Rossum distance. http://wwwold.fi.isc.cnr.it/users/thomas.kreuz/images/vanRossum.m (Last accessed 10.02.19).

[91] Conor Houghton and Thomas Kreuz. On the efficient calculation of van Rossum distances. *Network: Computation in Neural Systems*, 23(1-2):48–58, 2012.

[92] MATLAB. List of Deep Learning Layers. https://se.mathworks.com/help/deeplearning/ug/list-of-deep-learning-layers.html (Last accessed 04.06.19).

[93] MATLAB. trainingOptions. https://se.mathworks.com/help/deeplearning/ref/trainingoptions.html (Last accessed 31.05.19).

[94] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, pages 1–15, 2014.

A | Hardware

Name	**Processor**	**Memory**
Private laptop	Intel i5-8250U @ 1.60GHz	8 GB
Office workstation	Intel i7-6700 CPU @ 3.40 GHz	16 GB
Calcfarm	Intel Xeon E5-2690 v4 @ 2,6 GHz	96 GB

The different computers were used for the following:

- **Private laptop:** Initial analysis and simple calculations.
- **Office workstation:** Training of neural networks.
- **Calcfarm:** Some training of neural networks for single HD cells and finding model parameter values for the Izhikevich model.

The MATLAB license limits the usable hardware on Calcfarm. The office workstation was used in favour of Calcfarm with the training of the neural networks due to this limitation.

B | MATLAB Deep Learning Toolbox

B.1 Layers

A complete overview over available layers can be found in [92].

sequenceInputLayer	Inputs the sequence into the network.
fullyConnectedLayer	Multiplies the input by a weight before adding a bias vector.
lstmlayer	Learn long time dependencies in sequential data.
bilstmlayer	Learn bidirectional long time dependencies in sequential data.
dropoutLayer	Randomly sets input elements to zero with a given probability. Is used between layers in deep networks to reduce overfitting.
softmaxLayer	Applies a softmax function to the input of the layer.
classificationLayer	Computes the cross entropy loss for multi-class classification problems with mutually exclusive classes.

B.2 Hyperparameters

A complete overview over available hyperparameters can be found in [93].

`InitialLearnRate`	The variable decides the initial learn rate of the training. A too low value will have the training take too long, while a too large value have the training get stuck on a sub optimal result.
`LearnRateSchedule`	This specifies whether the learn rate should drop during training. If the parameter is set to 'piecewise', the drop in training is decided by the parameters LearRateDropFactor and LearnRateDropPeriod.
`LearnRateDropFactor`	How much the learn rate will drop every LearnRateDropPeriod epoch.
`LearnRateDropPeriod`	Number of epochs before the LearnRate drops.
`MaxEpochs`	Gives the maximum number of epochs used for training.
`MiniBatchSize`	How many observations to include in each mini-batch.
`Shuffle`	Controls whether the training data is shuffled or not, and if they are how often. The options are '`never`', '`once`' and '`every-epoch`'. By choosing '`once`', the data is shuffled once before training, while the '`every-epoch`' alternative shuffles the data before every epoch.
`ExecutionEnvironment`	Sets the execution environment for the network and determines the hardware resources used in training. The different options are '`auto`', '`cpu`', '`gpu`', '`multi-gpu`' and '`parallel`'.
`SequenceLength`	Pads or truncates the sequences in the mini-batches to a specific length.

C | Initial Analysis

C.1 Head Directions Represented

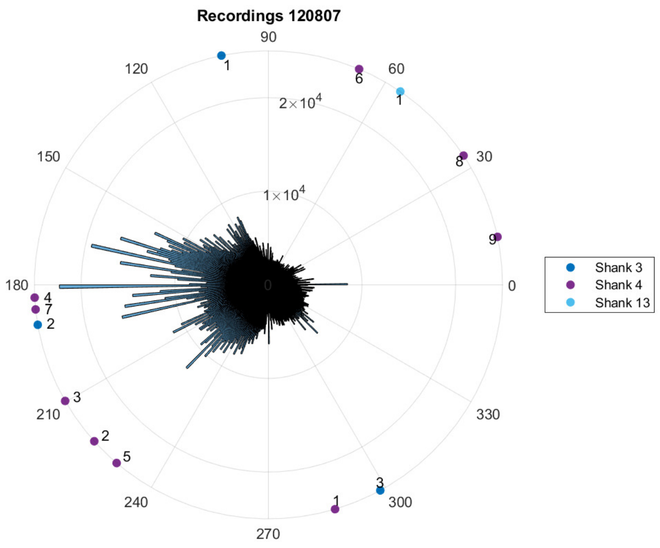


Figure C.1: Time spent with head in each direction and PFDs of HD cells for 120807.

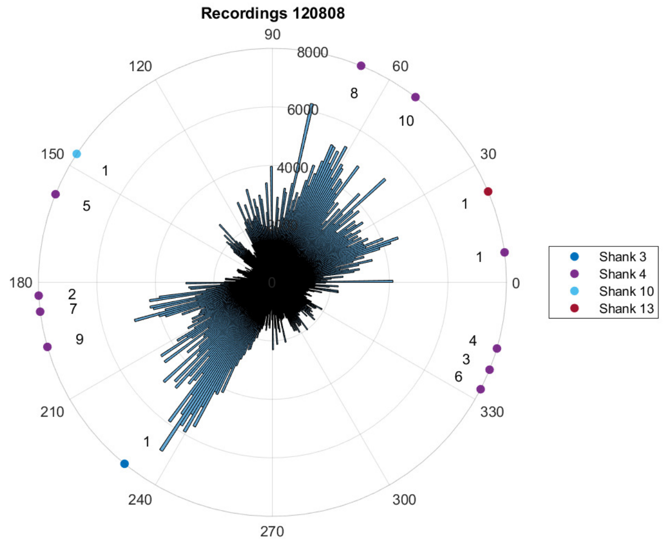


Figure C.2: Time spent with head in each direction and PFDs of HD cells for 120808.

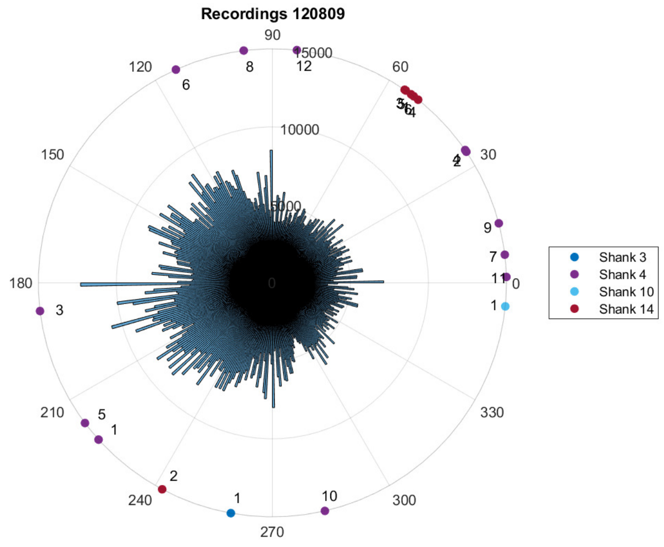


Figure C.3: Time spent with head in each direction and PFDs of HD cells for 120809.

C.2 Sessions per Angle Bin Width

Table C.1: Number of extracted sessions for 120807.

Angle bin width	**#sessions**
1°	257 770
5°	74 860
10°	40 684

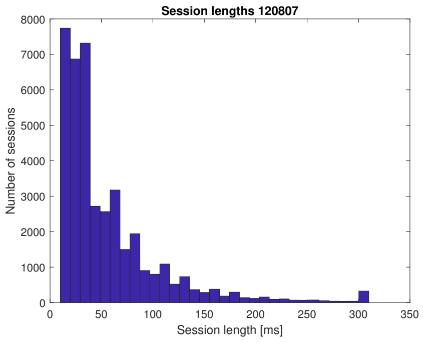
Table C.2: Number of extracted sessions for 120808.

Angle bin width	**#sessions**
1°	103 845
5°	30 256
10°	16 054

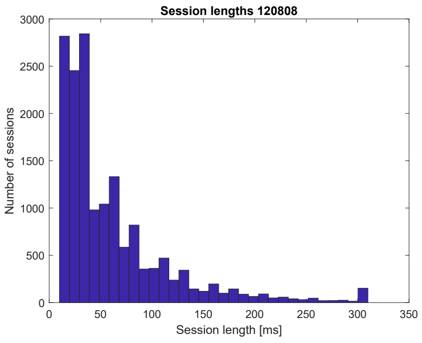
Table C.3: Number of extracted sessions for 120809.

Angle bin width	**#sessions**
1°	286 371
5°	82 112
10°	44 396

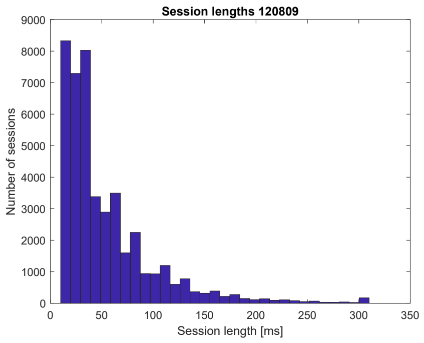
C.3 Session Lengths



(a) Day 120807.



(b) Day 120808.



(c) Day 120809.

Figure C.4: Length of extracted sessions for angle bin width of 10°. The bar above 300 ms represents the number of sessions longer than 300 ms.

D | Code

D.1 MATLAB Code Extracting HD Cells

```
function [totalCells, HDcells, totalHDcells,awakeTimeMs,
    binnedAwakeAngleData, meanFiringRate]=HDTuningCurves(session)

switch session
    case 1
        myDirName = 'path';
        filesWanted=3;
    case 2
         myDirName = 'path';
         filesWanted=13;
    case 3
         myDirName = 'path';
        filesWanted=13;
    case 4
         myDirName = 'path';
        filesWanted=14;
    case 5
         myDirName = 'path';
         filesWanted=14;
end

[angleData, resclu]=getData(myDirName,filesWanted);
nrpoints=360;

%Important relations from session. Relation between vectors.
fileName=sprintf('%s.states.Wake',myDirName);
awakeFile=load(fileName);
if numel(awakeFile)>2
    awakeFile=awakeFile';
end
awakeStart=awakeFile(1);
awakeEnd=awakeFile(2);
awakeTimeMs=awakeStart*1000:awakeEnd*1000;

angleFreq=1250/32;     %angle measurement rate
electrodeFreq=20000;   %ephysRate

%change -1 (no angle measurement) to be represented by NotANumber
tmpIndex=find(angleData==-1); % Finds indices where the array has
    values -1
angleData(tmpIndex)=NaN;
```

```

indexAwakeStart=round(awakeStart*angleFreq);
indexAwakeEnd= round(awakeEnd*angleFreq);
awakeAngleData=angleData(indexAwakeStart:indexAwakeEnd);

%Creating binnedAwakeAngleData one angle for each awake ms.
binnedAwakeAngleData=zeros(1,ceil((awakeEnd-awakeStart)*1000));
for i=1:length(binnedAwakeAngleData)
    ind0=ceil(i/ (1000/angleFreq) );
    ind1=ind0+1;
    if ind1 > length(awakeAngleData)
      continue
    end
    y0=awakeAngleData(ind0);
    y1=awakeAngleData(ind1);
    binnedAwakeAngleData(i)=interpolateBetweenPoints(i,ind0,ind1,y0,y1,angleFreq);
end

%occupancy part needed to compute tuning curve
occupancy=zeros(1,nrpoints);
plottingangles=zeros(1,nrpoints);
for i=1:nrpoints
    AA = pi*2/nrpoints*(i-1);
    BB = pi*2/nrpoints*i;
    plottingangles(i) = 0.5*(AA+BB)*360./(2.*pi);
    occupancy(i)=length(find(AA<binnedAwakeAngleData & binnedAwakeAngleData<BB));
end


totalHDcells=0;
totalCells=0;
for tetrode=1:filesWanted
    %tetrode (multiple electron measurement device)
    cellTimestamps=1000*resclu{2,tetrode}/electrodeFreq;  % Time of firings in all cells measured by electrode in milliseconds (ms) (freq 1000Hz).
    cluster=resclu{1,tetrode};  % CellNr in the cluster measured by the electrode.
    nrCellsInTetrode=cluster(1)-1;  % first element in cluster is overview of nr of Cells measured by electrode.
    HDcells(tetrode).val=[];
    HDcells(tetrode).time={};
    HDcells(tetrode).angle={};
    HDcells(tetrode).PFD=[];
    for cell=2:nrCellsInTetrode  % use nrCellsInTetrode if you want to plot all cells
        thisCellIndex=find(cluster==cell)-1;  % -1 as clusters first element is an extra telling number of cells in cluster.
        cellTimestampsThisCell=cellTimestamps(thisCellIndex);  % like cellTimestamps but only for one of the cells.

        awakeTimestampsIndexThisCell=find((cellTimestampsThisCell>(awakeStart*1000)) & (cellTimestampsThisCell<(awakeEnd*1000))); % find Index
        awakeCellTimestampsThisCell=cellTimestampsThisCell(awakeTimestampsIndexThisCell); % now only firings when the mouse is awake.
        firingTime=awakeCellTimestampsThisCell;

        %Interpolate to find angle at firingTime.
        nrFirings=length(firingTime);
        anglesAtFiring=zeros(1,nrFirings);
        for i=1:nrFirings
            %Find index of the angles surrounding firingTime(i)
            ind0=floor(firingTime(i)*angleFreq/1000);
            ind1=ind0+1;
            y0=angleData(ind0);
            y1=angleData(ind1);
            xx=firingTime(i); %angle at firingTime(i) is between y0
                and y1 angle
            anglesAtFiring(i)=interpolateBetweenPoints(xx,ind0,ind1,
                y0,y1,angleFreq);
        end

        if length(firingTime) < 100
            continue
        end

        tuningcurve=zeros(1,nrpoints);
        for i=1:nrpoints
            AA = pi*2/nrpoints*(i-1);
            BB = pi*2/nrpoints*i;
            numspikes=length(find(AA<anglesAtFiring & anglesAtFiring<
                BB));
            tuningcurve(i)=1000. * numspikes/occupancy(i); % convert
                to firing rates
        end

         figure
         plot(plottingangles, tuningcurve, 'o')
         title(sprintf('T%dC%d', tetrode, (cell-1)))
         set(gca,'XtickLabel',0:360/8:360)
         xlabel('Head direction (degrees)')
         ylabel('Firings rate (Hz)')
         ylim([0,max(tuningcurve)])

%%%% added by Nabiul %%%%
   cellIndex=cell-1;
   meanFiringRate{cellIndex}=tuningcurve;
    theta=pi/180:pi/180:2*pi;
    %%% plotting the circular histogram
    thetaT=zeros(1,4*numel(theta));
    for i=1:numel(theta)
        if i==1
         thetaT((i-1)*4+2)=0;
         thetaT((i-1)*4+3)=theta(i);
        else
         thetaT((i-1)*4+2)=theta(i-1);
         thetaT((i-1)*4 +3)=theta(i);
        end
    end

    %%% smoothing with gaussian kernel %%%
    sigma=6*pi/180;
    edges=-3*sigma:pi/180:3*sigma;
    kernel=normpdf(edges,0,sigma);
    kernel=kernel*pi/180;
    afterConv=conv(meanFiringRate{cellIndex},kernel);
    center=ceil(length(edges)/2);
    meanFirRateConv=afterConv(center:360+center-1);

    %%%% plotting the smoothed data
    rhoConv=zeros(1,4*numel(meanFirRateConv));
    for i=1:numel(theta)
      rhoConv((i-1)*4 +2)=meanFirRateConv(i);
      rhoConv((i-1)*4 +3)=meanFirRateConv(i);
    end
       %figure(totalCells+cellIndex)
       %title(sprintf('T%dC%d', tetrode, cell)) %
       %polarplot(thetaT, rhoConv);
    %%%%% calculating the mean resultant vectoral length
    vectorR{cellIndex}=circ_r(theta,meanFirRateConv,pi/180,2);
    meanDir{cellIndex}=circ_mean(theta,meanFirRateConv,2);


    [PFD{tetrode, cellIndex}, kappa{tetrode,cellIndex}]= circ_vmpar(
        theta,meanFirRateConv,pi/180);
    pval{tetrode, cellIndex}=circ_rtest(theta,meanFirRateConv,pi/180)
        ;

   %%%% HD cells inclusion criteria
    %%% (1). concentration parameter (kappa) >1 (2). peak firing rate
         >1 and
    %%% (3). probability of non-uniform distribution smaller than
        0.001
        if kappa{tetrode,cellIndex}>1
            if PFD{tetrode, cellIndex}>0
                  peakFirRate=meanFirRateConv(ceil(PFD{tetrode,
                      cellIndex}/(pi/180)));
            else
                  peakFirRate=meanFirRateConv(ceil((PFD{tetrode,
                      cellIndex}+2*pi)/(pi/180)));
            end
            if peakFirRate > 1 & pval{tetrode, cellIndex} <.001
                  HDcells(tetrode).val(end+1)=cell;
                  HDcells(tetrode).time{end+1}=firingTime;
                  HDcells(tetrode).angle{end+1}=anglesAtFiring;
                  HDcells(tetrode).PFD(end+1)=PFD{tetrode, cellIndex};
                  totalHDcells=totalHDcells+1;
            end
        end
    end
  totalCells=totalCells+nrCellsInTetrode-1;
end
end
```

```matlab
function [angleData, resclu]=getData(dirName,numfiles)

% Loads all res and clu files into a cell array.
dir_nm = dirName;

% Create cell array to store res and clu data from each "electrode"
resclu = cell(2,numfiles);

for i = 1:numfiles
     clufile = sprintf('%s.clu.%d',dir_nm,i);
     resfile = sprintf('%s.res.%d',dir_nm,i);
     clusters = load(clufile);
     timestamps = load(resfile);
     resclu{1,i} = clusters; resclu{2,i} = timestamps;
end



angleData=sprintf('%s.ang',dir_nm);
angleData=load(angleData);


end
```

```matlab
function [yy]= interpolateBetweenPoints(xx,ind0,ind1,y0,y1,angleFreq)
    %ind0 and ind1 is the two closest indices of angleData

yd=y1-y0;
if(isnan(y0) || isnan(y1))   %if firingTime is close to a nan angle
     yy=nan;
else
     if(yd>pi)
          y1=y1-2*pi;
     elseif(yd<-pi)
          y1=y1+2*pi;
     end

     x0=1000*ind0/angleFreq;
     x1=1000*ind1/angleFreq;

     yy=y0 + (xx-x0)*(y1-y0)/(x1-x0); %linear interpolation
end

yy=mod(yy,2*pi);

end
```

D.2 Overview Functions

analyseSessions	Divides the responses from HD cells from one day into sessions and presents key values for each angle bin and HD cell such as number of spikes an time spent in the angle bin.
balanceSessionsPerBin	Takes sessions and labels as input, and balances the data set by removing the shortest sessions from the angle bins with the most sessions.
compareSessionAndModel	Takes the model parameter values from `findOptimalModelParameterValues` and compares the spike sequence of one session with the given model parameter values. The function returns the distance between the session and model, and plots the session with the model.
compareSessions	Calculates the distance between all sessions for an HD cell. The functions returns a struct with some key number such as the percentage of comparisons from the same angle bin yielding the shortest distance and number of comparisons.
createAndTestNewInput	Creates new sessions from models from the createInputModel function and test the model in a pre-trained neural net.
createInputModels	Find optimal parameter values for Izhikevich model for `minModels` sessions fulfilling a given criterion. The model parameters are all zero if no sessions are available.
createXY	Takes head angles and spike times for a specific day, shank and HD cell, and extract sessions and labels them. The angle bin width is decided by the variable `angleBinWidth`. The function returns a cell array, `X`, with spikes, and a categorical array, `Y`, with angle bin labels.

createXYall	Takes the head angles and spike times for all HD cells recorded in a single day and extract sessions and labels the. The angle bin width is decided by the variable `angleBinWidth`. The function returns a cell array, `X`, with spikes of all cells, and a categorical array, `Y`, with angle bin labels.
findOptimalModelParameterValues	Returns the model parameters yielding the shortest Victor-Purpura between session recordings and the model for all sessions for one HD cell. The function also plots the recordings and the model yielding the shortest distance.
removeCells	Takes the sessions from `createXYall` as input, and returns the sessions for all the cells except the ones given in the array `cellsToRemove`.
removeCellsOutsideDFR	Takes sessions and labels as input, and removes all sessions from angle bins outside `dfrStart` and `dfrEnd`.
removeSessionsWithFewSpikes	Takes the sessions, `X` and the labels `Y` as an input, and removes all sessions with less spikes than `minSpikes`.
removeSessionsWithXSpikes	Takes the sessions, `X` and the labels `Y` as an input, and removes all sessions with `Xspikes` number of spikes.
removeShortSessions	Takes the sessions, `X` and the labels `Y` as an input, and removes all sessions shorter than `minLength`.
shuffleSessions	Takes sessions and labels as input, and shuffles them.
sortByLength	Takes sessions and labels as input and sort the sessions by length.

testLSTM	Classifies the input `XTest` with the input neural net `net` and calculates the accuracy. The confusion chart is created. The size of the mini-batch `miniBatchSize` should be the same as when training the network.
timeBetweenSpikes	Divides the response from one HD cell into sessions and finds the average time between spikes and standard deviation. The results are plotted with the angle along the x-axis and the time in milliseconds along the y-axis. The function returns a struct with some key results such as mean and SD for all angle bins.
trainLSTM	Trains the LSTM network. Input is `XTrain` and `YTrain` created with either `createXY` or `createXYall`.
twoTwoOR	Combines two and two cell responses in the `XTrain` using the OR operator to create new features.

Eline Stenwig

Neural Response Analysis of Head Direction Cells

NTNU
Norwegian University of
Science and Technology