Norges miljø- og biovitenskapelige universitet

Master's Thesis 2018 60 ECTS

Faculty of Science and Technology
Gaute T. Einevoll

Predicting model parameters of a spiking neuron network through its local field potential by using deep learning

Jan-Eirik Welle Skaar

Biotechnology
Faculty of Chemistry, Biotechnology and Food Science

Abstract

Electrophysiological measurements provide a relatively easy method of probing the brain at the scale of networks. One example of such measurements is the extracellular potential that can be obtained by inserting microelectrodes directly into neural tissue. The low frequency part of this signal is called the local field potential (LFP), and despite having a solid understanding of the underlying physical principles determining these electric potentials, the enormous complexity of neural networks makes the interpretation of the signal difficult. Little is known about exactly what information can be extracted from it.

Convolutional neural networks, a form of machine learning using deep feed-forward artificial neural networks, have been successfully applied to a wide array of tasks, most notably image recognition tasks and natural language processing. In this project, we apply convolutional neural networks to simulated local field potentials. We use a newly developed hybrid scheme to simulate the LFP generated by a spiking point-neuron network model. To explore what information the LFP signals contain, we address the following question: *Can a convolutional neural network be trained to make predictions about the parameters of a spiking point-neuron network by using only the LFP it generates?*

We use a relatively simple model consisting of one excitatory population and one inhibitory population, and systematically vary three parameters: the excitatory synaptic strength, the relative inhibitory synaptic strength and the amount of external input. We then simulate the LFPs generated by this model network, and use them to train convolutional neural networks to make predictions about the values of each parameter. We find that for this network, it is indeed possible to make fairly accurate predictions about the network parameters by using only the LFP it generates. These results demonstrate that our approach shows promise for extracting non-trivial information from brain signals, and therefore has the potential to play an important role in taking full advantage of these signals in the future.

Sammendrag

Elektrofysiologiske målinger gir en forholdsvis enkel måte å måle aktiviteten til nettverk av nevroner i hjernen. Et eksempel på en slike målinger er det ekstracellulære potensialet som kan måles ved å sette elektriske elektroder direkte i hjernevev. Den lavfrekvente delen av dette kalles lokale feltpotensialer (LFP), og selv om vi har en solid forståelse av de underliggende fysiske prinsippene bak de lokale feltpotensialene, gjør kompleksiteten til nevrale nettverk at det vanskelig å tolke dem, og man vet lite om hva slags informasjon de inneholder.

Konvolusjonelle nevrale nettverk er en form for maskinlæring som har blitt mye brukt til blant annet bildegjenkjenning og prosessering av naturlige språk. I dette prosjektet blir konvolusjonelle nevrale nettverk anvendt på simulerte lokale feltpotensialer. En nylig utviklet hybridmetode for å simulere de lokale feltpotensialene som blir generert av nettverker av punktnevroner blir benyttet, og følgende spørsmål blir belyst: *Kan konvolusjonelle nevrale nettverk bli trent opp til å anslå verdien til spesifikke parametere til et nettverk av punktnevroner?*

Vi benytter en relativt enkel modell som består av en eksitatorisk populasjon og en inhibitorisk populasjon, og varierer tre parametere: den eksitatoriske synaptiske styrken, den relative inhibitoriske synaptiske styrken og mengden ekstern input. LFP-ene fra dette nettverket med de ulike parameterverdiene blir simulert, og blir brukt til å trene opp det konvolusjonelle nevrale nettverket til å anslå parameterverdiene. Det viser seg at det konvolusjonelle nevrale nettverket er i stand til å anslå alle tre verdiene med ganske stor treffsikkerhet. Disse resultatene antyder at metoden er i stand til å hente ut informasjon fra hjernesignaler, og har potensial til å spille en viktig rolle i å dra full nytte av dem i fremtiden.

Contents

1 Introduction **1**

2 Theory **5**
2.1 Neurons and communication 5
2.2 Point neuron networks 7
2.2.1 The Brunel network 8
2.3 LFP generation and the hybrid scheme 9
2.3.1 The origin of local field potentials 9
2.3.2 Forward modeling of the LFP 11
2.3.3 LFP generated by point-neuron network 13
2.3.4 LFP prediction by population activity 13
2.4 Artificial neural networks 14
2.4.1 Convolutional neural networks 15

3 Methods **19**
3.1 Point neuron activity 19
3.2 LFP approximation by population activity 19
3.3 Convolutional neural network 22
3.4 Statistical methods 23

4 Results **25**
4.1 Point network activity 25
4.2 LFP approximation by population activity 29
4.3 Parameter effects on LFP 31
4.4 Training convolutional neural networks 34
4.4.1 Classification accuracies 36
4.4.2 Inspecting the first filters 39

5 Discussion **41**

List of Figures

1.1 Electrophysiological measurements 2

2.1 Schematic neuron 6
2.2 Schematic synapse 7
2.3 Excitatory postsynaptic potential 8
2.4 Brunel phase diagram 9
2.5 Network activity states 10
2.6 Compartment RC circuit 12
2.7 Dense connections 15

3.1 Column and morphologies 21
3.2 CNN architecture 24

4.1 Example network activity 26
4.2 Firing rates and CVs 27
4.3 Pairwise spiketrain correlations 28
4.4 Population approximation to the LFP 29
4.5 Combined kernels 30
4.6 Population activity approximation to the LFP 31
4.7 Example LFPs 32
4.8 LFP power spectral densities 33
4.9 Training curves 34
4.10 Training curves 35
4.11 Prediction accuracies by parameter 36
4.12 Prediction accuracies 37
4.13 First-layer filters 40

Chapter 1

Introduction

The human brain is composed of roughly 86 billion neurons [1]. Each neuron is connected to thousands or tens of thousands of others [2], to which it can send and receive signals. The nature of signalling between individual neurons is well understood, but how information is processed in the vast network that is the brain, or how higher features of the mind arise, such as thought and consciousness, is still largely unknown.

Neocortex, a part of the outermost layer of the mammalian brain, is where higher-order functions of the brain take place, such as sensory perception, cognition and motor control [3]. Vertically, it has six layers, labelled I to VI from the outside inward, and is functionally partitioned across its surface. The most basic organization of the cortex is believed to be *minicolumns*, a vertical chain comprising roughly 100 neurons in primates [4]. Minicolumns form *cortical columns*, or *modules*, by short-range horizontal connections. These vary between 300 and 600 µm in all species, regardless of the size of the brain [4], suggesting a very fundamental structure that can be thought of as a functional unit.

Investigating the details of cortical networks experimentally is difficult. The large number of cells they are composed of make it very challenging to physically examine them, necessitating more indirect methods of investigation. The most widely used are measurements of electric potentials, which can be done at multiple levels: Electroencephalography (EEG) measures the potential on the scalp, electrocorticography (ECoG) measures the potential on the brain surface, and by inserting microelectrodes directly into the brain, the potential can be measured in the extracellular medium [5]. Figure 1.1 shows an illustration of the scale of these measurements. Recordings such as these have been carried out for many decades, but a true understanding of what information they carry is still lacking. The recorded potentials arise from electric transmembrane currents generated by the signalling of neurons, and although we have a solid understanding of the physics involved at the single cell scale, the complexity of large networks has made their connection to the measurable electric potentials difficult to establish [6].

Since EEG and ECoG measure potentials far from their source and contain contributions from many different areas of the brain, their resolution will be limited and they will contain less information about local networks [5]. Extracellular recordings within the brain, however, will mainly consist of contributions from its closer surroundings, making it a better source of information on more detailed network activity. The extracellular potential can be split into two parts: the multi-unit activity

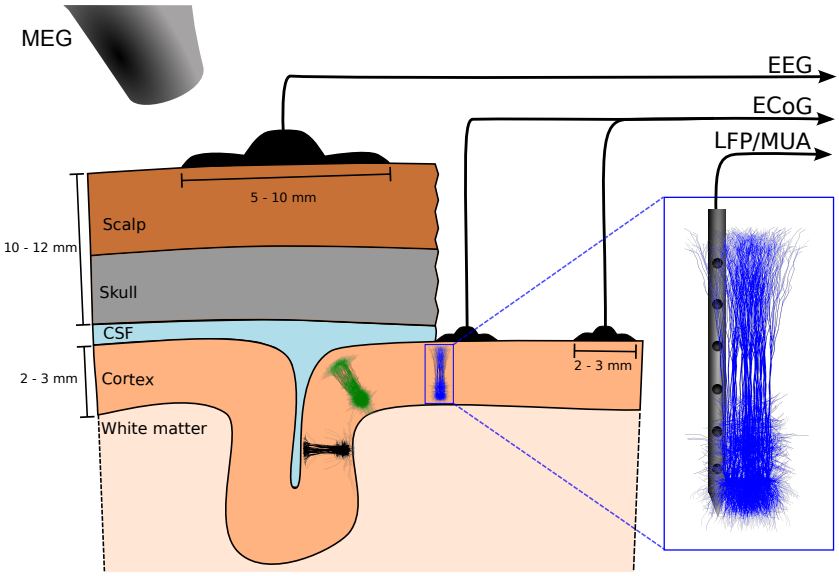


Figure 1.1: An illustration of electrophysiological measurements at different levels. The distance of the recording position will affect the recording in multiple ways. The farther away from the current sources, the weaker the signals become, and it contains contributions from a larger number of neurons. The LFP/MUA is the most local extracellular measurement of activity. Courtesy of Torbjørn V. Ness.

(MUA) and the local field potential (LFP). MUA is the high frequency part of the signal (> 500 Hz), and reflects the firing of neurons in the immediate vicinity of the electrodes [7]. However, the signal decays rapidly with distance [8], so while it can say much about the activity of individual neurons close to the recording electrode, it is of limited use when investigating larger networks. The LFP is the low frequency part (≤ 500 Hz), and reflects the synaptic input and processing in a greater area [7]. The LFP will have sizeable contributions from neurons hundreds of micrometers away [9, 10], and reflects the activity of a much larger network, on the scale of tens of thousands of neurons [7]. This potentially makes it a good source of information on local cortical circuitry, but the large number of neurons making contributions also makes it very difficult to interpret, and little is known about exactly what sort of information it carries.

Convolutional neural networks (CNN) [11] are a type of deep feed-forward artificial neural networks, a form of machine learning that has been successfully applied to a wide array of tasks, most notably image recognition tasks [12] and natural language processing [13]. In this project, we apply convolutional neural networks to simulated LFPs. To investigate what information can be obtained from the LFP, we ask the following question: *can a convolutional network be trained to make predictions about the parameters of a spiking point-neuron network by using only the LFP it generates?*

A relatively simple point neuron network model is used, consisting of two populations, one excitatory population of 10,000 neurons, and one inhibitory population of 2,500 neurons. We focus on three parameters of the network: J, the excitatory synaptic strength, g, the relative inhibitory synaptic strength, and η, the amount of external input. These are key parameters in determining the activity of the network [14]. Using the newly developed hybridLFPy tool [15], the LFPs generated by this

model network is simulated for 8 different values of each parameter, giving a total of 512 unique parameter combinations. These LFPs are then used to train convolutional neural networks to make predictions about the value of each parameter. We find that for this model network, it is indeed possible to fairly accurately predict the values of each parameter.

Chapter 2

Theory

This chapter gives some biological background on neurons, before describing the formalism used to model the activity of our network, the theory used to make the forward calculations of the LFP, and an introduction to convolutional neural networks.

2.1 Neurons and communication

Neurons come in many different shapes and sizes. With few exceptions, they have three distinct parts: the soma, the axon, and the dendrites. The soma is the "'heart'" of the neuron, and contains the nucleus and most of the other organelles. Long, cable-like extensions project out from the soma: axons and dendrites. It is through these projections the neuron communicates with others. For communication to happen, a signal must pass the cell membranes of both neurons. This happens through special interfaces called synapses, from the axon on the *presynaptic* neuron to a dendrite on the *postsynaptic* neuron. Figure 2.1 shows an illustration of a neuron and contact points with other neurons.

Cell membranes consist of a lipid bilayer about 5 nm thick [16], where phospholipids are oriented with their charged head-groups pointing outward into the intracellular and extracellular medium, and their lipid tails pointing inward, meeting each other at the centre. While ions move freely in both the intracellular and extracellular medium, the highly resistive membrane makes it almost completely impermeable to ions and acts as a capacitor, being able to store charge that is transported across the membrane. Special ion pumps set up an electrochemical gradient across the membrane. Sodium, chloride and calcium is pumped from the intracellular to the extracellular space, while potassium is pumped in the opposite direction. Protein channels are embedded in the membrane that can let specific ions through. Some channels are open all the time, while others only open when activated by either a neurotransmitter or a voltage threshold. When the channels are open, the electrochemical gradient will drive a current through the channels, either inward or outward depending on which channels are open [16].

There are two main types of synapses: chemical and electrical. Through electrical synapses, an electrical signal is passively transmitted from one neuron to the other through gap junctions, small structures creating a hole in the membranes, connecting the cytoplasm of the two neurons allowing small molecules and ions to pass [17]. Chemical synapses are more complex. When a presynaptic neuron fires an action

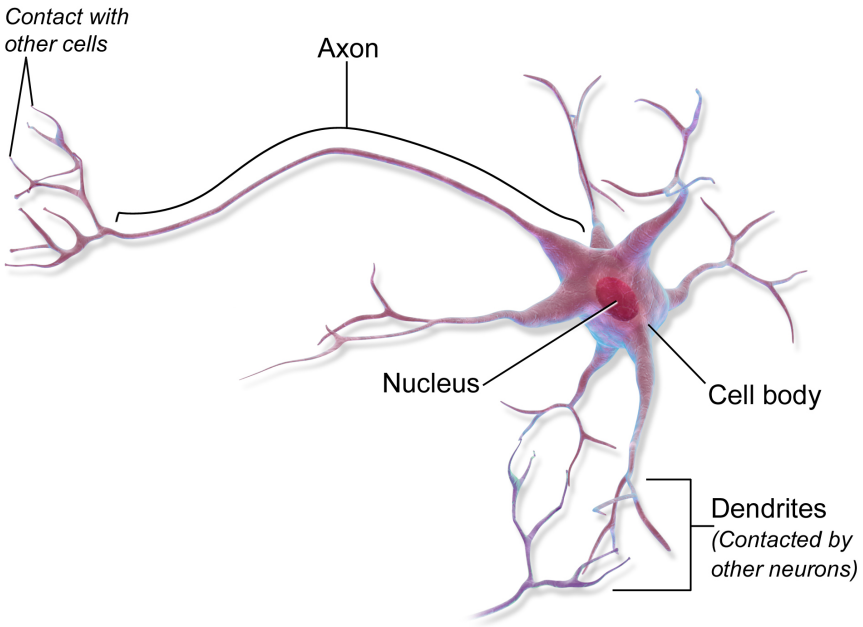


Figure 2.1: Illustration of a neuron. Signals can be sent to other neurons through the axon, and signals can be received from other neurons through the dendrite. The cell body, or soma, contains the nucleus and is the site of most metabolic activity. Adapted from Blausen.com staff (2014). "Medical gallery of Blausen Medical 2014". WikiJournal of Medicine 1 (2). DOI:10.15347/wjm/2014.010. ISSN 2002-4436

potential, all of the synapses on its axon are activated. When a chemical synapse is activated, neurotransmitters are released from vesicles inside the axon terminal into the synaptic cleft, where they interact with and open ion channels on the membrane of the postsynaptic neuron. If the neurotransmitters open channels that carry a positive inward current, an excitatory postsynaptic potential (EPSP) occurs in the dendrite, depolarizing the postsynaptic neuron. If channels that carry a positive outward current are opened, an inhibitory postsynaptic potential (IPSP) occurs, polarizing the cell. The strenghts of the postsynapic potentials vary from synapse to synapse, and can change with time, called synaptic plasticity. This is believed to be the mechanism by which we can learn and form memories [18]. Figure 2.2 shows an illustration of a synapse.

The postsynaptic potentials are propagated through the dendrite, with some of it leaking through channels in the membrane. Since the neuron receives input from many synapses at different locations and times, the membrane potential is the spatial and temporal summation of all the postsynaptic potentials the dendrite receives. All incoming signals to a neuron is processed in this manner in the dendrite, and will ultimately decide when the neuron is going to fire. If the neuron receives enough excitatory input to drive the potential in the soma above a certain threshold, the neuron will fire, creating an action potential that is actively propagated through the axon, activating its synapses in an all-or-none manner.

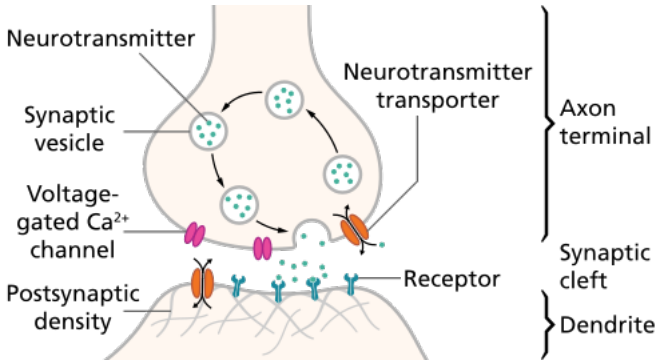


Figure 2.2: Schematic of a chemical synapse. When an action potential reaches a synapse, neurotransmitters stored in vesicles are released into the synaptic cleft, where they interact with receptors opening ion channels on the dendrite, allowing ions to flow through. The neurotransmitters are subsequently transported back into the neurons, allowing the ion channels to close and are ready to repeat the process. By Thomas Splettstoesser (www.scistyle.com) [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], from Wikimedia Commons

2.2 Point neuron networks

There are several different frameworks for simulating the activity of a neuronal network. From highly detailed neuron models calculating all transmembrane and intracellular currents in detailed morphologies, to very simple rate-based models where the entire neuron is modelled by only its firing rate. In this project, leaky integrate-and-fire (LIF) point neurons are used to model the activity of the network. These are simplified neuron models which do not take into account the spatial complexities of the neurons, but rather assumes that the neuron is confined to a single point. Although simplified, these point neurons still display interesting dynamics and are able to capture many important features of neuronal networks [14]. The mathematical formalism used is described here.

Formalism

The subthreshold dynamics of the LIF neurons obey the differential equation

$$\tau_{\mathrm{m}} \frac{dV(t)}{dt} = -V(t) + R_{\mathrm{m}} I(t) \tag{2.1}$$

where τ_{m} is the membrane time constant, V the membrane potential, R_{m} the membrane resistance and I the input current [14].

When the membrane potential reaches a threshold θ, the neuron fires and its potential is clamped to the reset potential V_{r} for the refractory period t_{ref}.

There are N_{E} excitatory neuron and N_{I} inhibitory neurons, and each of them receive local input from $C_{\mathrm{E}} = \epsilon N_{\mathrm{E}}$ excitatory neurons and $C_{\mathrm{I}} = \epsilon N_{\mathrm{I}}$ inhibitory neurons, i.e. from a share ϵ of both populations. Every neuron also receives external excitatory poisson distributed input with a rate ν_{ext}. The input currents are modelled as delta functions, causing an instantaneous change in the the membrane potential with an amplitude J. Figure 2.3 shows an example of a single postsynaptic

potential for an excitatory spike. For neuron i,

$$R_{\mathrm{m}} I_i(t) = \tau_{\mathrm{m}} \sum_j J_{ij} \sum_k \delta(t - t_j^k - t_{\mathrm{d}}) \tag{2.2}$$

where the first sum is over all the presynaptic neurons j, including the external ones, and the second sum is over the spike times of those neurons. J_{ij} is the voltage amplitude of the postsynaptic potential, δ is the Dirac delta function, t_j^k are the times at which neuron j fires, and t_{d} is the delay period.. For excitatory synapses, including the ones receiving external input, $J_{ij} = J > 0$, and for inhibitory synapses $J_{ij} = -gJ$, where g is the parameter determining the relative strength of the inhibitory synapses compared to the excitatory synapses [14].

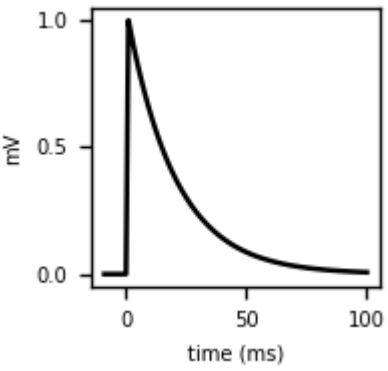


Figure 2.3: A single excitatory postsynaptic potential from a spike with $J = 1$ mV and $\tau_{\mathrm{m}} = 20$ ms

The external poisson rate ν_{ext} will be given in terms of a new parameter $\eta = \nu_{\mathrm{ext}}/\nu_{\mathrm{thr}}$, where $\nu_{\mathrm{thr}} = \theta/(JC_{\mathrm{E}}\tau_{\mathrm{m}})$, the minimum constant rate input that would drive the membrane potential to firing threshold; i.e. if the external input was constant over time with an η of 1, the membrane potential would converge to θ as $t \to \infty$. Note however that with random poisson distributed input, the neurons will still occasionally fire when $\eta < 1$.

2.2.1 The Brunel network

Networks with the same formalism as described above has been mathematically analysed by Brunel (2000) [14]. In his article, he analysed the activity generated from this network mathematically, and was able to derive the particular activity states that arise by varying different parameters of the network, including g, η and J.

Figure 2.4 shows a phase diagram for the network derived from the mathematical analysis by Brunel, adapted from Figure 2 in Brunel (2000) [14]. Four different activity states arise by varying η and g. Two characteristics of the activity are defined, synchrony and regularity. Synchrony is in this context defined as a time dependent global firing rate, and regularity is defined as a coefficient of variation (CV) of the inter-spike intervals close to zero in the individual neurons. The four different activity states are synchronous regular, fast and slow oscillating synchronous irregular, and asynchronous irregular.

Examples of the activity from each of these states are shown in Figure 2.5. The synchrony of each synchronous state is driven by different factors. Brunel found that the fast oscillating synchrony for high η and high g was driven by the negative feedback loop caused by the strong inhibitory neurons. When the network activity rises due to the external input, the inhibitory neurons cause a decrease in activity again at a time proportional to t_{d}, which again causes an increase in activity at

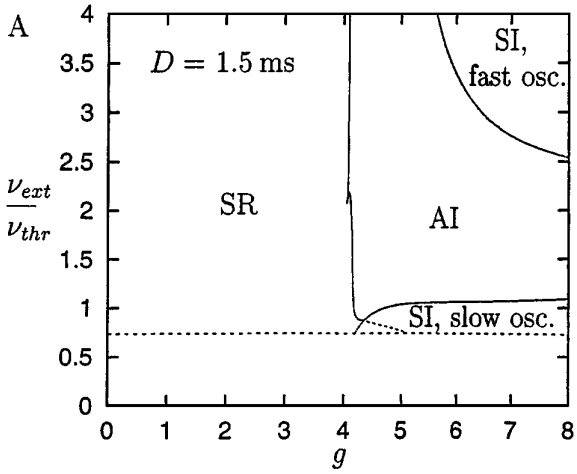


Figure 2.4: Phase diagram showing the boundaries of four different network activity states depending on parameters g and η. J is 0.1 mV, and the synaptic delay 1.5 ms. The four different states are synchronous regular, synchronous irregular with fast oscillations, synchronous irregular with slow oscillations and asynchronous irregular, denoted by SR, SI and AI. Aapted from Figure 2 in Brunel (2000) [14].

a time proportional to $2t_{\mathrm{d}}$, driving a rapid cycle of high activity followed by low activity. This can be seen in the top right panel of Figure 2.5.

The slow oscillating synchronous activity at $\eta \approx 1$ and $g > 4$ is driven by the external input keeping the neurons close to their firing threshold, making the network very sensitive to any fluctuations in activity. They can be rapidly excited, before the strong inhibitory neurons kills the activity again, causing the network to oscillate between being in a state of quiescence and state of activity. This can be seen in the lower left panel of Figure 2.5.

Lastly, when $g < 4$, the network will be in an excitation dominated regime, since there are four times as many excitatory neurons as inhibitory neurons. A very high frequency oscillation appears, controlled by the refractory and delay period, and the neurons fire at very regular intervals.

Note that due to finite size effects, the transitions between the states are not sharp, and there are clearly some oscillations in the activity in the asynchronous state, as can be seen in the lower right panel of Figure 2.5, but it is clearly less synchronous than in the other three panels. Instead, there is a gradual transition from one state to the other [14]. The phase diagram in Figure 2.4 is assuming J to have a value of 0.1 mV, and will not be equal for other values of J.

In this project, we have chosen to focus on the transition from the synchronous regular regime to the asynchronous regular, in the area $4 \leq g \leq 5.4$, $1.6 \leq \eta \leq 3.0$ and $0.06 \leq J \leq 0.2$.

2.3 LFP generation and the hybrid scheme

2.3.1 The origin of local field potentials

The local field potential is generated by transmembrane current [6], acting as *current sources* and *current sinks* in the extracellular medium. Within the framework of

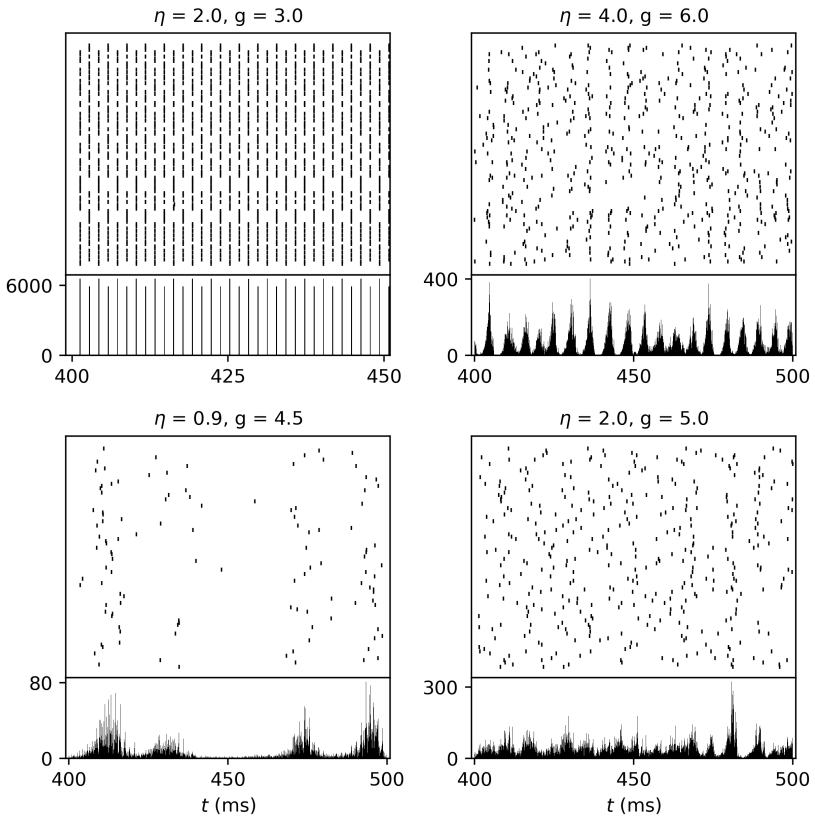


Figure 2.5: Four different network states, from simulations with 10,000 excitatory neurons and 2,500 inhibitory neurons. $J = 1$, and the the synaptic delay is 1.5. A raster is shown on top and full spike histograms below, with bins of 0.1 ms. Top left shows the excitation dominated asynchronous regular regime, where close to half of the neurons fire simultaneously in regular intervals. In the bottom right panel, g is increased to 5, and the activity becomes asynchronous irregular. Top right and bottom left shows the synchronous irregular states, fast and slow respectively. This figure shows the same as figure 8 in [14], but is recreated with own data.

volume conduction theory [5, 19], the electric potential generated by such current sources and sinks can be derived.

In a uniform medium of inifinite extent, the current from a point source would flow uniformly in all radial directions. Since the current is conserved, the current density at a distance r would be

$$\mathbf{J} = \frac{I(t)}{4\pi r^2}\mathbf{a_r}\,, \tag{2.3}$$

where $\mathbf{J}$ is the current density vector, $I(t)$ is the magnitude of the point current source and $\mathbf{a_r}$ is the radial unit vector [19]. Assuming the medium is *linear*, *isotropic*, *homogeneous*, *ohmic* and *frequency-independent*, the current density $\mathbf{J}$ is related to the electric field $\mathbf{E}$ by Ohm's law,

$$\mathbf{J} = \sigma\mathbf{E}\,, \tag{2.4}$$

where σ is the conductivity of the medium, a constant real scalar [19, 20, 7]. Using the quasi-static approximation of Maxwell's equations, i.e. neglecting the time derivatives, the electric field $\mathbf{E}$ is related to the scalar potential ϕ by

$$\mathbf{E} = -\nabla\phi\,. \tag{2.5}$$

Applying equations 2.4 and 2.5 to Equation 2.3 yields

$$\frac{I}{4\pi r^2}\mathbf{a_r} = -\sigma\nabla\phi\,. \tag{2.6}$$

Due to the symmetry of the system, ϕ can only change radially, so integrating along $\mathbf{a}_r$ we end up with a formula for the potential a distance r away from the current source [19],

$$\phi(r) = \frac{I}{4\pi\sigma r}\,. \tag{2.7}$$

If many different currents are injected at different points, the potential adds linearly as [5, 19]

$$\phi(\mathbf{r}, t) = \sum_n \frac{1}{4\pi\sigma}\frac{I_n(t)}{|\mathbf{r} - \mathbf{r}_n|}\,, \tag{2.8}$$

where $I_n(t)$ is the point source at position $\mathbf{r}_n$.

The assumptions made above will generally hold for the extracellular medium of the cerebral cortex [7, 20, 21], and this is the framework used to make the forward calculations of the local field potential.

2.3.2 Forward modeling of the LFP

Calculating the LFP generated by a neuron requires two steps: first, all transmembrane currents in the neuron must be calculated in a morphologically detailed neuron model. These transmembrane currents can then be treated as current sources and current sinks in the extracellular medium, and can be used to calculate the LFP [20, 7].

To model the transmembrane currents, multicompartment neuron models are used, where the neurons are divided into compartments small enough that the potential can be assumed to be constant within the entire compartment [16]. The dendrites are divided into cylindrical compartments, while the soma is modelled as a single sphere. The compartments can be modelled as an RC circuit, shown in Figure 2.6. The cell membrane, separating the intracellular space from the extracellular space, acts as a resistor in parallel with a capacitor. The passive membrane current is determined by the membrane resistance R_n and the membrane potential $V_n - E$, where E_n is the equilibrium potential. Note that a passive cable formalism [15] is used, where E_n is constant, and the entire system is linear. E_n can therefore be set to 0 mV and be ignored. The extracellular space is assumed to have an infinitely large conductivity, leading to a constant potential in the extracellular space. In the intracellular space, however, there is a finite conductivity between each compartment, leading to different membrane voltages throughout the neuron, and axial currents flowing between compartments.

Kirchhoff's current law states that the net current entering a node in the circuit must equal 0, and the following equation governing the development of the

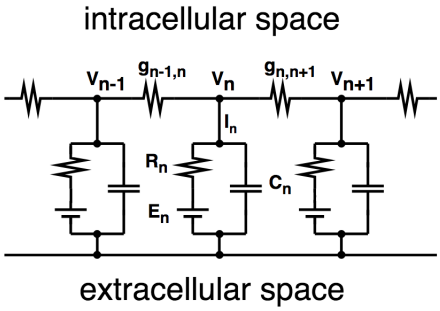


Figure 2.6: The equivalent RC-circuit of multicompartment models. In compartment n, R_n is the membrane resistance, C_n is the membrane capacitance, E_n is the equilibrium potential, I_n is the transmembrane current, V_n is the membrane potential and $g_{n,n+1}$ is the axial conductance between compartments n and $n+1$.

compartment potentials can be derived [16, 20]

$$g_{n,n+1}(V_{n+1} - V_n) - g_{n-1,n}(V_n - V_{n-1}) = C_n \frac{dV_n}{dt} + \sum_j I_n^j \,, \tag{2.9}$$

where $g_{n,n+1}$ is the conductance between compartments n and $n+1$, proportional to the cross section between the compartments. $\sum_j I_n^j$ is the transmembrane currents through the j channels in compartment n, in this case encompassing synaptic input and leak currents. The left-hand side represents the intracellular axial currents to the neighbouring compartments. The first term on the right hand side is the capacitive current, while the second term is the currents due to other membrane processes such as passive leak currents and synaptic input.

Current-based synapses are used in this project, meaning that the synapse input current is of a fixed shape, independent of the compartment potential. Since a passive neuron model is used, this makes the entire system linear.

Once the transmembrane currents are calculated, they can be used to predict the LFP. Since the dendritic compartments are cylindrical, instead of the point source formula in Equation 2.8, a *line-source* equation is used to calculate the LFP from the dendritic transmembrane currents [20]. The currents through the membrane are assumed to be spread out evenly in each section and a line-source current is used for each compartment. By integrating eq. 2.4 along the center-line axis of the compartment the equation [20]

$$\phi(\mathbf{r}, t) = \frac{1}{4\pi\sigma} \sum_n I_n(t) \int \frac{dr_n}{|\mathbf{r} - \mathbf{r_n}|} = \frac{1}{4\pi\sigma} \sum_n I_n(t) \frac{1}{\Delta s_n} \log \left| \frac{\sqrt{h_n^2 + \rho_n^2} - h_n}{\sqrt{l_n^2 + \rho_n^2} - l_n} \right| , \tag{2.10}$$

is obtained, where Δs_n is the length of compartment n, ρ_n is the distance perpendicular to the compartment, h_n is the longitudinal distance from the start of the compartment, and $l_n = \Delta s_n + h_n$ is the longitudinal distance from the end of the compartment. The soma is assumed to be spherical, and a point source is used to calculate its LFP contribution, leading to the full equation for a single neuron [20]

$$\phi(\mathbf{r}, t) = \frac{1}{4\pi\sigma} \left(\frac{I_0(t)}{|\mathbf{r} - \mathbf{r}_0|} + \sum_{n=1}^{N} I_n(t) \frac{1}{\Delta s_n} \log \left| \frac{\sqrt{h_n^2 + \rho_n^2} - h_n}{\sqrt{l_n^2 + \rho_n^2} - l_n} \right| \right) \tag{2.11}$$

where the first term is the point-source contribution from the soma, and the remaining sum is the line-source contributions from all dendritic compartments.

2.3.3 LFP generated by point-neuron network

Point neurons are not spatially extended and cannot be directly used in the forward modelling of the LFP described above. Instead, the hybrid scheme developed by Hagen et al. (2016) [15] is used, where the spikes of the point neuron network are mapped onto morphologically detailed neurons in a consistent manner. A one-to-one mapping is made from point neurons to morphological neurons, and for each connection in the point neuron network, a corresponding synapse is placed in a compartment on its morphological counterpart. The synapses are placed in random compartments with a probability weighted by the compartments surface area, within given depth boundaries. Current-based synapses are again used, but the delta-function shaped synapses used in the point neuron network cannot be used when calculating the LFP, since they produce an infinitely large current. Instead, alpha-function shaped synapses are used, described by the function

$$I(t) = JCte^{1-t/\tau_{\mathrm{s}}}\,, \tag{2.12}$$

where τ_{s} is the synaptic time constant, J the synaptic strength, and C a constant. Due to the linearity of the system, scaling C will scale the resulting LFP by the same proportion, and is ultimately irrelevant for the purpose of this project.

Alpha-function shaped synaptic currents could also have been used in the point-neuron network model, but delta-function shaped currents are used instead because it was used by Brunel in his analysis [14], so in order to make use of his results, the same is used here. Alpha synapses with very small synaptic time constants would have produced similarly shaped postsynaptic potentials to delta synapses, but in order to get a more realistic low-pass filtering effect, a higher synaptic time constant was used when calculating the dendritic currents, so this discrepancy was chosen over better self-consistency.

The neuron somas are randomly placed within a cylinder of given dimensions, and are randomly rotated around specified axes. For each spike in the point-neuron network, the corresponding synapses on the postsynaptic morphological neurons are activated after the delay period, and the currents are calculated as described in the previous section. The transmembrane currents are then used to calculate the LFP.

A more detailed description of the hybrid modelling scheme is given by Hagen et al. (2016) [15].

2.3.4 LFP prediction by population activity

Instead of explicitly simulating all the dendritic currents, which is a computationally intensive task, Hagen et al. (2016) [15] found that it is also possible to make a good approximation the LFP by the instantaneous population firing rates. Due to the current-based synapse and passive dendritic currents, every synapse creates a unique contribution to the LFP which does not change with time. The LFP generated by the firing of a single point neuron i can be found by the convolution

$$\sum_k K_i(t) * \delta(t - t_i^k)\,, \tag{2.13}$$

where K_i is some kernel specific for neuron i and t_i^k are its spike times. The kernel K_i would depend on which neurons it is connected to, their morphologies and parameter values. Finding the kernel can be done by simply calculating the LFP generated by the spike. Doing this for all neurons in a network by summing over all neurons,

$$\sum_i \sum_k K_i(t) * \delta(t - t_i^k) \tag{2.14}$$

would produce equal results to calculating the LFP as described above. However, finding and using all the individual kernels would still be a laborious process. Instead of a kernel specific for each neuron, one could use the average kernel for each population,

$$\frac{1}{N} \sum_{i=0}^{N} K_i(t) \tag{2.15}$$

that could be convolved with the instantaneous population firing rate to predict an approximate LFP. This kernel would encompass the full connectivity pattern, neuron placements, morphologies and parameters. Using the population firing rate means that the information on exactly which neurons are firing is lost, but it turns out that the approximation is still very good. This kernel can be found in an analogous way to the single neuron kernel, by calculating the LFP created by all the the neurons of the population firing simultaneously. Due to the random connectivity, the exact neuron positioning within the cylinder does not affect the kernel much, and it will create a good approximation regardless of the seeds used to place synapses, neurons and to set rotations.

2.4 Artificial neural networks

In the field of machine learning, artificial neural networks consist of *nodes* or *neurons*, connected to each other in some fashion. These connections represent a single multiplication operation. One of the simplest types of network is one consisting only of fully connected layers.

Node i in layer k, n_i^k, receives input from all nodes in the previous layer, and outputs to all nodes in the following layer. The *activation* of the node is $a_i^k = \sigma(\sum_j a_j^{k-1} w_{ji}^k + b_i^k)$, where a_j^{k-1} is the activation of node j in layer $k-1$, w_{ji}^k is the *weight* of the connection between node j in layer $k-1$ and node i in layer k, and b_i^k is the *bias* of node i in layer k. The weights and biases are the parameters of the network. σ is some nonlinear *activation function*. Figure 2.7 shows an example of the connectivity in a small dense network. From the point of view of a single node, it collects the activations from all nodes in the previous layer, multiplies each of them by their specific weight, adds them together, adds its bias on top, and finally applies the activation function on the resulting number. This nodes activation is the output of the activation function, and will be collected by all nodes in the following layer in the same manner. An arbitrary number of layers and nodes can be connected in this manner, and the final layers is called the *output layer*, whose size will depend on what the network is trying to learn.

In *supervised learning*, a network such as this would receive some input where the desired output is known. As an example, if a network were to be trained to

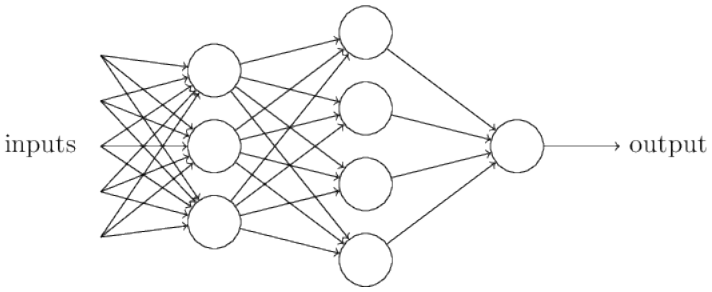


Figure 2.7: A small, fully connected network. Each node takes as input the activations of all nodes in the previous layer, multiplied by it, and broadcasts its activation to all nodes in the subsequent layer. Figure taken from Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015.

recognize cats in images, labeled training data would consist of images that may or may not contain cats and a label specifying whether or not cats are actually present in the image. The label could for instance be 1 if a cat is present, and 0 if not. This allows you to define a *loss function* to determine how well the network is performing. One example of such a function could be the mean squared difference between the labels and outputs for all images in the dataset,

$$L(\theta) = \frac{1}{N}\sum_{n=0}^{N}(y(x_n, \theta) - y'_n)^2 \tag{2.16}$$

where θ are the parameters, x_n the input image, $y(x_n, \theta)$ the network output and y'_n its label. The key to training a network such as this is an algorithm called *backpropagation*, which computes the partial derivatives of the loss function with respect to all parameters in the network, i.e. its gradient. By making a small adjustment to the weights and biases in the opposite direction of the gradient, the loss function should decrease, and the output should be closer to the label [22].

Typical datasets used to train neural networks contain thousands or tens of thousands of images and labels, and ideally the weights should be changed as to minimize the loss function for all of them. However, doing so would be very computationally expensive. What is done instead is to randomly partition the datasets into *mini-batches*, or simply batches, and using the gradient of the loss function over single batches instead of the full dataset. The size of the batches are typically on the order of one or a few hundreds. This is called stochastic gradient descent. For each pass over the dataset, new random batches are created so the network is trained on different combinations each time.

2.4.1 Convolutional neural networks

The problem with fully connected networks is the huge number of parameters, making it unfeasible to create deep networks consisting of multiple hidden layers. Another type of networks, convolutional neural networks, greatly reduces the amount of parameters, and don't have this issue [11]. CNNs typically consist of convolutional layers, normalization layers, max pooling layers and fully connected layers. The workings and function of each layer will be briefly explained here.

In convolutional layers, *filters*, also called kernels, are slided across the input, and the scalar product is calculated at each position. The width and height of the

filters are chosen, but they always include all input channels. The strides, or step size, that are taken as the filters move is also chosen, and if it is larger than one, the input will be downscaled. If the strides prevent the filters to reach the boundaries without 'overshooting', zero-padding is used to allow them to reach the end. Zero-padding can also be used to prevent downscaling due to the filters themselves, by allowing them to start and end 'outside' the image. A single bias is added to all of the scalar products from a single filter, and an activation function, usually the rectified linear unit (ReLU)

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

is applied. The resulting output is called a *feature map*. Each filter produces its own feature map, so the number of filter determines how many feature maps are created. The feature maps from one layer is considered the channels in the next. The trainable parameters of convolutional layers are the filters themselves. For a more detailed introduction to convolutional layers, see, e.g. [22].

Batch normalization [23] is a technique used to help the training process. The idea of batch normalization is to normalize the input to a layer so that its mean is 0 and variance 1, and is typically used after every or every few layers in the network. It consists of the following steps [23]:

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{and} \quad \sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(\mathrm{x_i})$$

where μ_B is the batch mean and σ_B^2 the batch variance over dimensions dependent on the type of layer. x_i is example i of m in the batch, and $\hat{x}_i$ the normalized example, ϵ is some small number for numerical stability. The first steps consist of finding the batch mean and batch variance, over dimensions depending on the preceding layer, explained below. The mean and variance is used to normalize the input, and the final step consists of multiplying by and adding two *trainable* parameters. The reason for this final step is to give the network the choice of undoing the normalization. Forcing the activities to have certain statistics would also put constraints on what the network can learn. By adding the final operation, the batch normalization layer could perform the identity function if needed, by setting $\gamma = \sigma_B$ and $\beta = \mu_B$.

Before fully connected layers, the mean and variance is taken over the batch dimension, so that each neuron receives normalized input. After convolutional layers, however, we want to preserve the convolution property that every activity in the same feature map is normalized in the same way. Therefore, the normalization is done over all locations, and one pair of parameters γ and β is trained per feature map. When training the network, the batch mean and variance is used to normalize. During inference, however, these are not available. Instead, a moving average of the batch means and variances is taken during training, providing a global mean and average that are used for normalization when doing inference.

Max pooling layers move a window of given size across each feature map and outputs the largest value within the window at each step. This serves to down-sample the input and make it more translationally invariant. If a particular feature is detected its filter will output a high number to the feature map, and by max pooling only the highest number from the feature map in each max pooling window will be passed on to the next layer, making its exact position less important.

There are multiple techniques to avoid *overfitting* the model to the training data, i.e., called *regularization*, including *weight decay* [24], which adds the L1-norm or L2-norm of the weights as an additional term to the loss function, *dropout* [25], which randomly removes a share of the neurons in a fully connected layer during the training stages, and batch normalization also provides some degree of regularization [23].

These are the basic building blocks of the convolutional neural networks. There are no exact rules for determining the best architecture for a particular task, and they are usually determined by trial and error. There are, however, some features shared by most architectures. The first part of the network consists of convolutional layers, followed by batch normalization layers and max pooling layers every so often. Batch normalization can be applied after every layer or every few layers, and max pooling is typically less frequently, especially in deeper networks, to avoid too much loss of information. After these layers usually follows one or two fully connected layers before the output layer. The exact architectures used in this project is described in the next section.

Chapter 3

Methods

3.1 Point neuron activity

The network studied in this project is a two-population network consisting of one excitatory population of 10,000 neurons, one inhibitory population of 2,500 neurons. Each neuron also receives excitatory input from some external population, modelled as poisson processes. The simulations follow the formalism described in section 2.1.2, and was carried out using the NEST simulator v. 2.12.0 [26]. All parameters used, with a short description, are given in table 3.1.

A total of eight different values were used for each of the parameters η, g and J, giving of 512 combinations. 12 simulations were run for 2001 ms for each parameter combination with different seeds. The spiking histograms for each population were saved to be used for predicting the LFP. An additional simulation was run for each parameter combination for 3000 ms, where the spikes were saved for analysis purposes. The simulations were performed on Stallo, a high performance computing cluster consisting of 2.80 Ghz Intel Xeon E5 2680 CPUs.

3.2 LFP approximation by population activity

The kernels used for predicting the LFP by population activity were created by the method described in 2.1.3.

Figure 3.1 shows a schematic of the column and neuron morphologies. There are two layers, an upper one in which half of the excitatory synapses are placed on the excitatory neurons, and a lower layer where the other half of the excitatory synapses are placed on the excitatory neurons, and all inhibitory synapses are placed. The inhibitory neurons have synapses placed only in the lower layer. The neuron somas are placed randomly within the bounds of -450 µm $< z < -350$ µm and $r < 564$ µm, and are not placed closer than 1 µm to other somas. The excitatory neurons are rotated randomly about their apical axis, and the inhibitory neurons are rotated randomly about all axes. The electrode is placed vertically down the middle of the column, with six contacts, starting at $z = 0$ µm, separated equally by 100 µm.

A short simulation of 400 ms was run, where all excitatory neurons fire at t = 100 ms, and all inhibitory neurons fire at t = 300 ms. The kernels for the excitatory population is the recorded LFP between t = 0 ms and t = 200 ms, divided by 10,000, the number of excitatory neurons, and the kernels for the inhibitory populations is

Table 3.1: Point neuron simulation parameters

Neuron parameters		
Parameter	Value	Description
τ_{m}	20 ms	membrane time constant
C_{m}	1 pF/µm^2	specific membrane capacitance
V_{r}	10 mV	reset potential
E_{L}	0 mV	resting potential
θ	20 mV	spike threshold
τ_{ref}	2 ms	refraction period
t_{d}	1.5 ms	synaptic delay
J	$0.06, 0.08, .., 0.2$ mV	excitatory synaptic strength
g	$4.0, 4.2, .., 5.4$	relative inhibitory strength
η	$1.6, 1.8, .., 3.0$	relative external input
C_{E}	1000	number of incoming excitatory connections
C_{I}	250	number of incoming inhibitory connections
Connection rules		
Connection rule		fixed indegree
Self-connections		yes
Multiple connections allowed		yes
Simulation parameters		
Simulation time		2001 ms
Simulation time step		0.1 ms
Integration method		exact
Initial membrane potentials		uniformly distributed between 0 and θ

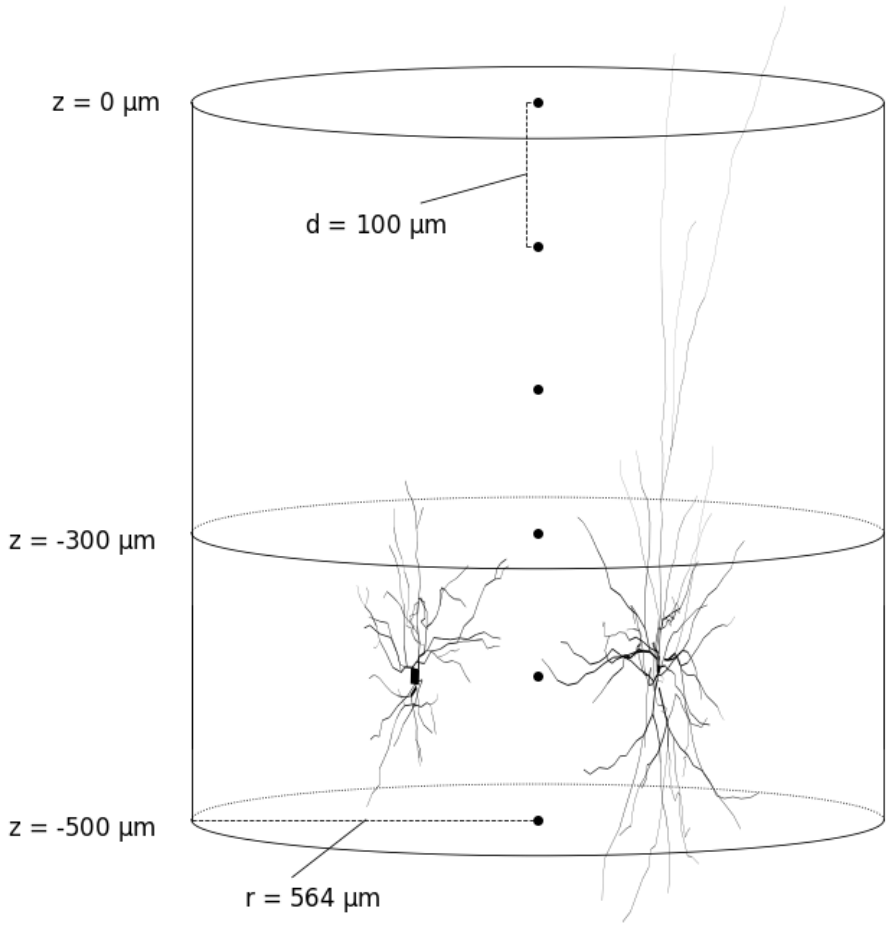


Figure 3.1: The cylinders show the upper and lower layers, as well as the lateral bounds for soma placements. Vertically, the somas are confined between $z = -350$ and $z = -450$, and not the layer boundaries. The black dots are the six locations at which the LFP is recorded, starting at z = 0 µm, equally separated by 100 µm, ending at -500 µm The neuron on the left shows the inhibitory morphology, and the neuron on the right shows the excitatory morphology.

Table 3.2: Hybrid model parameters

Parameter	Value	Description
		Neuron parameters
R_{m}	20000 Ω/mm^2	membrane resistance
C_{m}	1 $\mu\mathrm{F}/\mathrm{cm}^2$	specific membrane capacitance
V_{init}	0 mV	initial membrane potential
R_{a}	150 Ωcm	axial resistance
λ_{f}	100 Hz	frequency of AC length constant for d_lambda rule
r	564 μm	population radius
h	100 μm	soma layer thickness
E_{L}	0 mV	resting potential
		Electrode parameters
σ	0.3 Sm^{-1}	extracellular conductivity
N_{contacts}	6	number of electrode contacts
d_{e}	100 μm	distance between contacts
r_{e}	5 μm	electrode radius

the recorded LFP between t = 200 ms and t = 400 ms, divided by 2,500, the number of inhibitory neurons. Although both the activity and LFP simulation is run with a time resolution of 0.1 ms, the LFP signal is down-sampled to 1000 Hz. These kernels are used to make the LFP prediction for all the point network simulations by convolving it with their spike histograms with time bins of 1 ms.

3.3 Convolutional neural network

The LFPs generated were cut into lengths of 300 ms with overlaps of 150 ms to increase the amount of training data. The first 150 ms of each signal was cut to avoid any abnormalities occuring at the start of the simulations. Each simulation created 11 LFPs. Two unique simulations for each parameter combination were separated from the rest to serve as test data, and another was separated as validation data. A total of 50688 unique training signals were created, and 11264 test signals, and 5632 validation signals.

The architecture of the network is described in figure 3.2. For all convolutional layers, the ReLU activation function is used, and is immediately followed by batch normalization, as described in section 2.4. The convolutions are same-padded, and only the max pools alter the output lengths. In the fully connected layers except the output layer, batch normalization is performed on the input, and the ReLU activation function is used. The output layer consists of 8 neurons, one for each parameter value, where one-hot encoding of the label is used. L2-regularization is performed on all weights in the network, and dropout on the first of the fully connected layers.

One network was trained to for each of the parameters, i.e. three networks were trained to make the full classification of all three parameters. The cross-entropy loss function was used, and different optimizers were tried.

The training was done using TensorFlow v1.6 with cuDNN v7.1 on a single

NVIDIA GeForce GTX 1070. The CNN training and evaluation scripts, along with the simulation scripts can be found at the following github: github.com/janskaar

3.4 Statistical methods

To analyse and compare the activities of the point-neuron network across the entire parameterspace, some key statistics have to be extracted from each simulation. The regularity of firing of a neuron is measured by the coefficient of variation (CV) of its inter-spike intervals. The CV is defined as the ratio of the standard deviation to the mean. A neuron firing at the exact same intervals has a CV of 0. The pairwise correlations of spike timings are also measured by the Pearson correlation coefficient, which calculated for all pairwise spiketrains of 500 randomly chosen neurons, with a bin size of 3 ms. The mean firing rate of all neurons in the network is also measured.

To analyse the simulated LFPs, the power spectral density is calculated. Since 12 simulations of the LFP are run per parameter combination, the average PSD over all simulations for the particular parameter combination is calculated. Welch's method is used, with segments of length 256 and overlaps of 50% on the LFP of each simulation, before they are averaged.

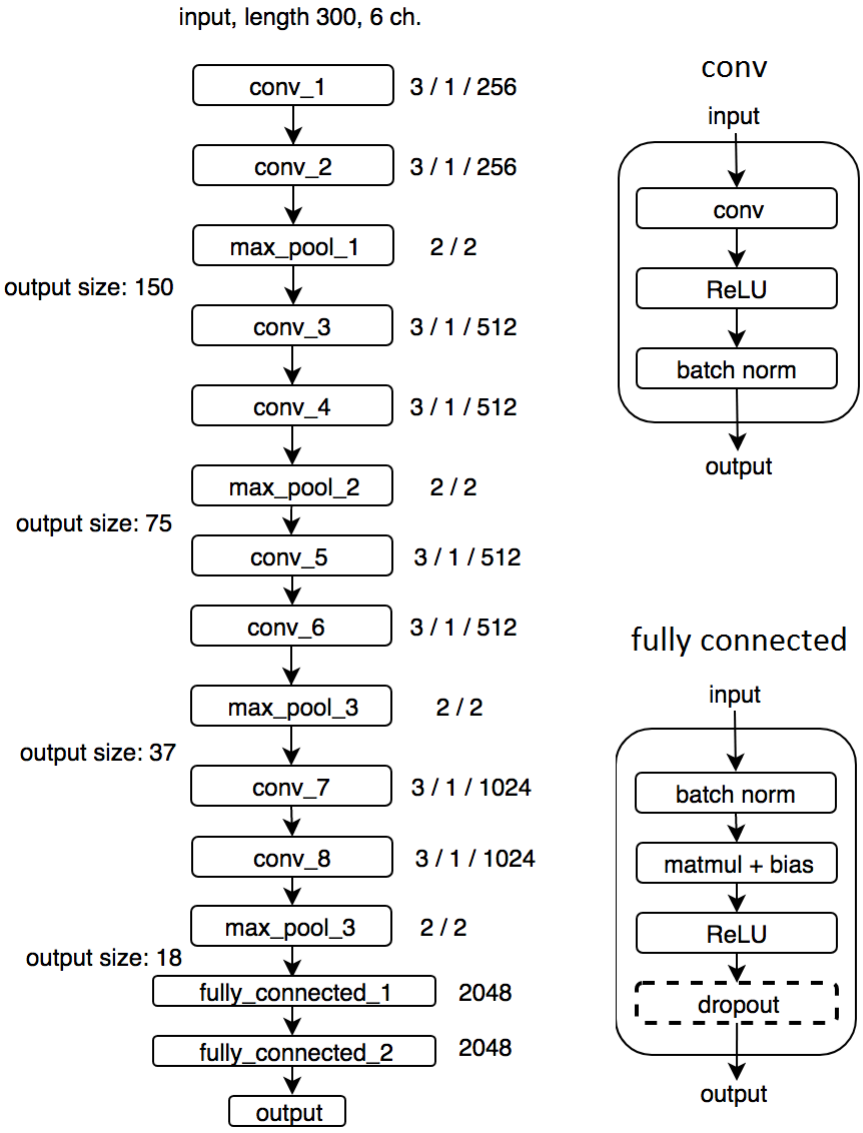


Figure 3.2: Architecture of the convolutional neural network. On the left side, the number and order of convolutional blocks, max pooling layers and fully connected blocks. The numbers to the right of the convolutional blocks are the filter sizes, strides, and output channels. The number to the right of the fully connected blocks are the number of neurons, and the numbers to the right of the max pooling layers are the size and strides. On the right, the details of each block is shown. The convolutional layers are followed by ReLU activations, and batch normalization, while the fully connected layers are preceded by batch normlization, and followed by ReLU activation. Dropout is applied only on the first fully connected layer. Matmul + bias represents the weight multiplication and addition of biases in the fully connected layer, as explained in the previous chapter.

Chapter 4

Results

4.1 Point network activity

To give an idea of what the network activity looks like, three rasters and full spike histograms are shown in Figure 4.1, from simulations in different areas of our parameter space. The rasters are from 50 neurons randomly chosen regardless of population. Recall that both population have the same number of synapses coming from each population, so their activity will on average be the same.

The upper raster is from a simulation with the lowest values of g and J, where the neurons fire with high regularity and there is a strong oscillation of the global activity. It is this low-J regime that is closest to the synchronous regular state, although it is quite far from completely the synchronous regular regime, which can be observed by comparing it to the top left panel in Figure 2.5.

The middle raster is from a simulation where all parameter values are larger. One can observe that the regularity has largely disappeared, and the synchrony has decreased, although there is some degree of oscillating global activity, as there always will be in finite sized networks. It is clearly further into the asynchronous irregular regime. The activity is still quite high, as η has increased, but it's still lower than in the one in the first panel, due to the larger g and J. The lower raster is from a simulation with the highest values of g and J, giving much sparser firing.

Figure 4.2 shows the mean firing rates and CVs over the entire parameter space. Each heat map shows the η-g-plane for a single value of J. The firing rates generally decrease as g increases, and as η decreases. This is of course to be expected, since a higher η means more external excitatory input, and a higher g means stronger inhibitory input. The effect of g on the firing rate is more marked for higher values of J, causing a more rapid decline than for lower values of J. Apart from on the $g = 4$ line, the activity also decreases with J. This might seem puzzling, as J increases both the excitatory and inhibitory input by the same proportion, but can be explained by considering the absolute amount of current entering the neurons from local connections, $J_I + 4J = (4 - g)J$. Although the ratio between excitatory and inhibitory input is determined only by g, their absolute difference is also determined by J, except for when $g = 4$. When $g > 4$, increasing J also increases the amount of inhibition in the network. Below the line $g = 4$, it would have the opposite effect, increasing the amount of excitation.

While the effect of J on the firing rate is relatively modest compared to the effect of g, its effect on the CV is large. A low value of J leads to very regular firing, and a

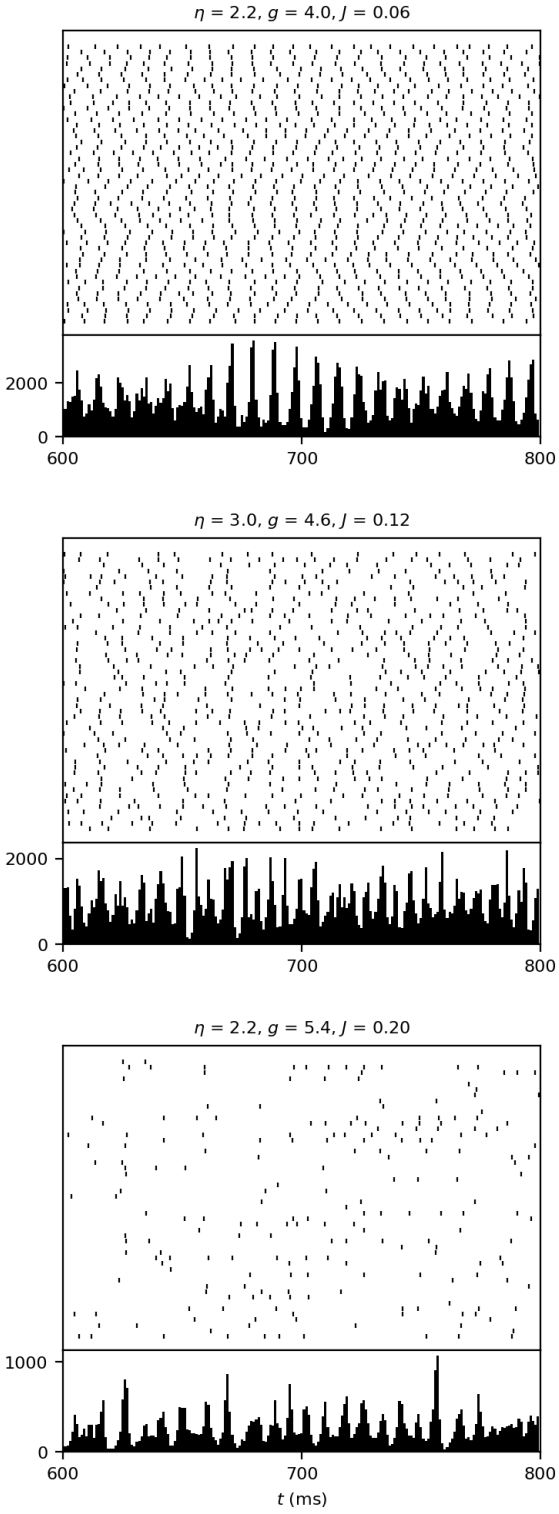


Figure 4.1: Rasters and firing histograms from three simulations with different parameter combinations. Each raster contains spikes from 50 randomly chosen neurons regardless of population. The raster time resolution is 0.1 ms and the histogram time resolution is 1 ms.

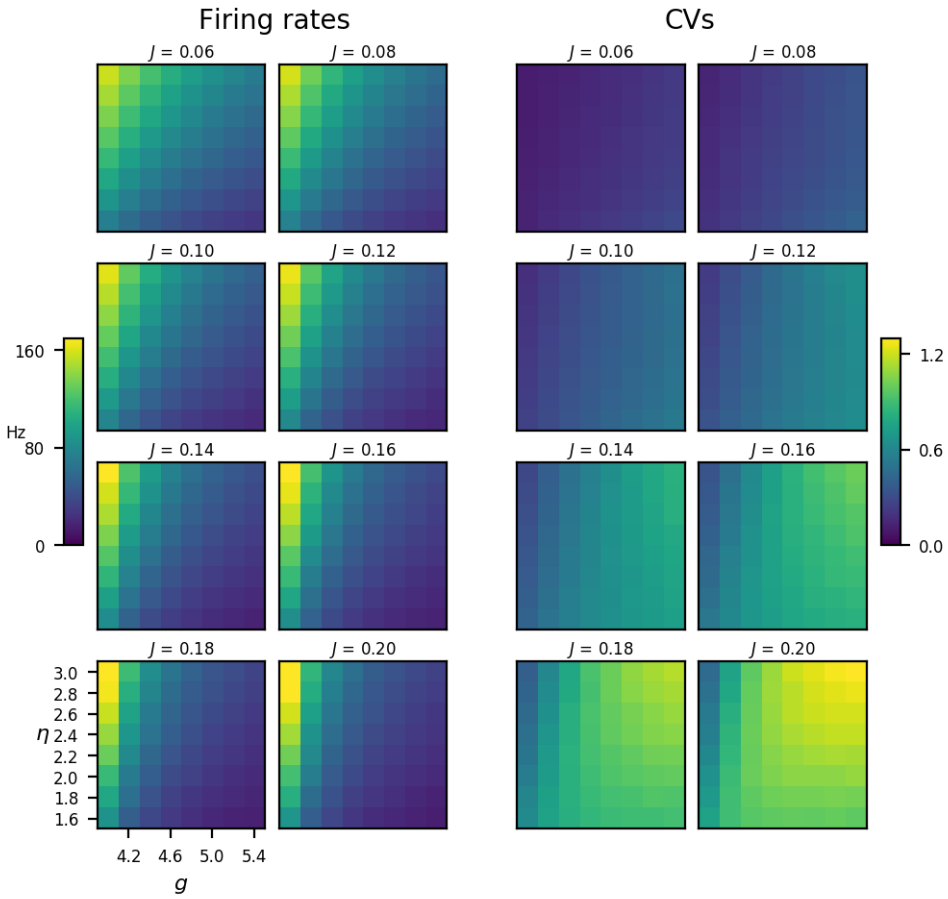


Figure 4.2: CVs and firing rates for all parameter combinations. Each heat map shows the η-g-plane for the value of J indicated above. The left pane shows the mean firing rates averaged over all neurons in the network, and the right pane shows the mean CV averaged over all neurons in the network.

high value leads to irregular firing. The effect can be observed in Figure 4.1. Recall the definition of $\eta = \nu_{\text{ext}} J C_{\text{E}} \tau_{\text{m}} / \theta$. For a constant value of η, if J increases, the external firing rate ν_{ext} must decrease by the same proportion. The incoming spikes from the external population are modelled as a poisson process, where the number of spikes in a given interval is poisson distributed. Since current based synapses are used, a fixed amount of charge enters the neuron every time a synapse receives input. The total amount of charge entering the neuron in some time interval will therefore be XJ, where X is the poisson distributed number of incoming spikes in the interval and J is the synaptic strength. Consider what happens to the variance of the input current as J is scaled by some factor α. The mean external firing rate will decrease by the same factor, $\nu_2 = \nu_1/\alpha$, and the variances will be $\text{Var}(X_2) = \text{Var}(X_1)/\alpha$. The ratio of the variances of the incoming currents will therefore be

$$\frac{\text{Var}(\alpha J X_1)}{\text{Var}(J X_2)} = \frac{\alpha^2 J^2 \text{Var}(X_1)}{J^2 \text{Var}(X_2)} = \frac{\alpha^2 \text{Var}(X_1)}{\alpha \text{Var}(X_1)} = \alpha \,.$$

i.e. although the mean external input current stays the same, its variance increases linearly with J.

Figure 4.3 shows the mean pairwise Pearson correlation coefficients of the spike

trains from 500 randomly chosen neurons. The pairwise correlations are generally very small, but there is still a clear tendency for them to decrease with g as the network enters the asynchronous irregular state. Not shown in the figure is the standard deviations, which are roughly 0.035 everywhere except at the lowest value of J and g, where it is double that.

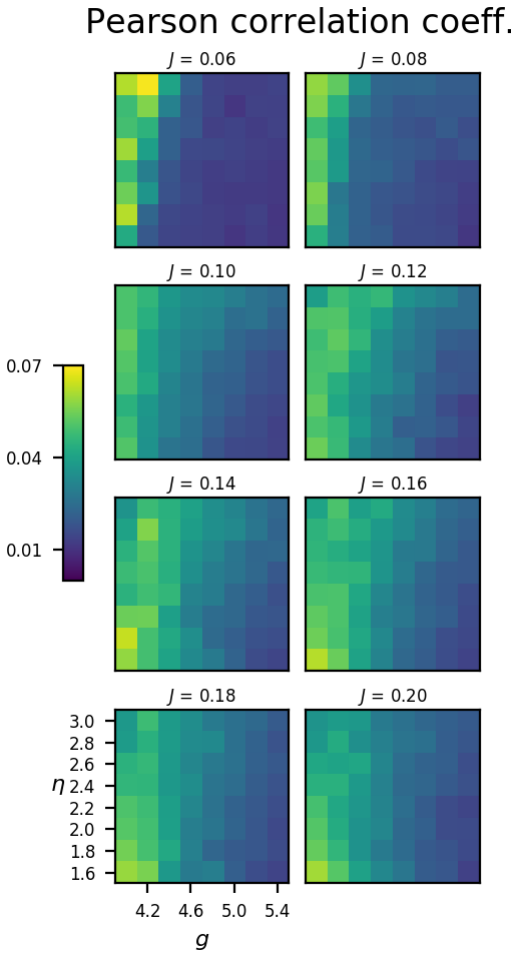


Figure 4.3: Mean Pearson correlation coefficients for all pairwise spiketrains of 500 randomly chosen neurons regardless of population, with a bin size of 3 ms.

4.2 LFP approximation by population activity

The kernels for predicting the LFP from the population activity were produced as described in section 2.2. They are shown in the bottom right panel in Figure 4.4. Each kernel represent the average contribution to the LFP by a single spike of its population. Due to the linearity of the current based synapses and passive dendrites, scaling the synaptic strengths will scale the LFP by the same proportion, i.e. scale the respective kernels by the same proportion. The kernels in the figure are shown with $g = 4$ and $J = 0.1$ mV. As g increases, the inhibitory kernels will become larger compared with the excitatory ones. Note however also that since the excitatory population is four times larger than the inhibitory population, so the effect of the excitatory population on the LFP is larger than what the kernels might suggest.

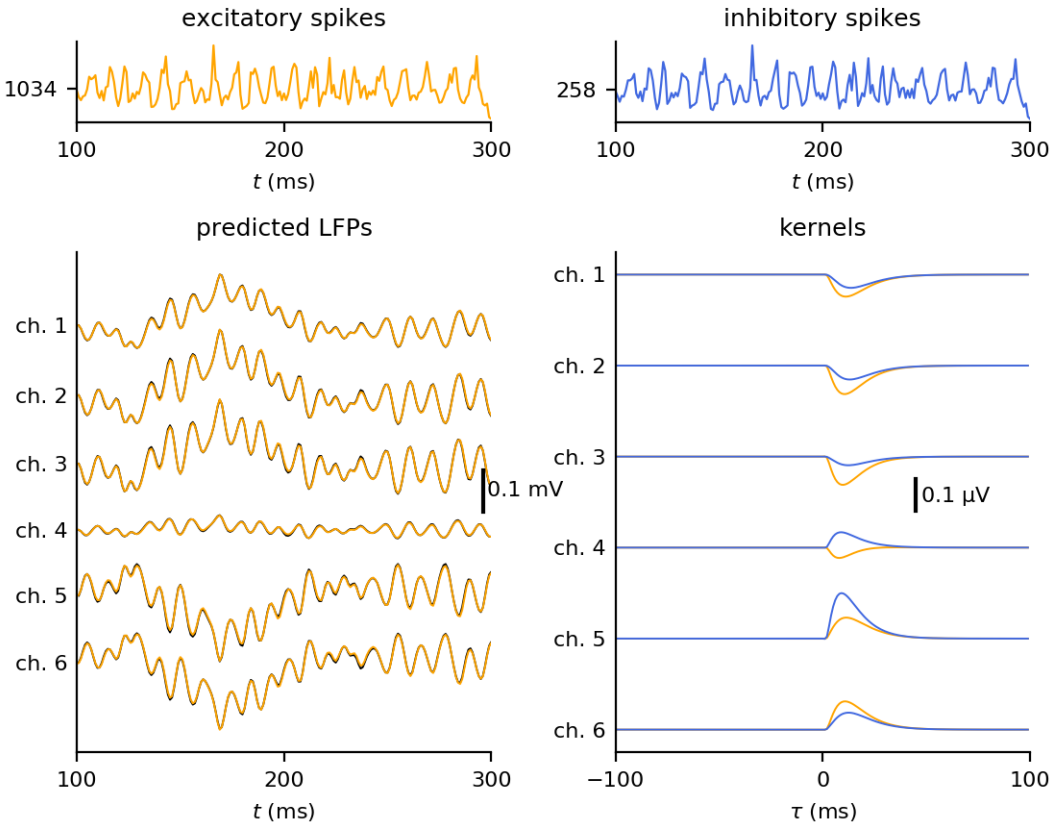


Figure 4.4: The bottom right part of the figure shows the kernels used for the population activity approximation to the LFP. The inhibitory kernels are shown in blue, and the excitatory kernels in orange. The top part shows both the excitatory and inhibitory population activity for a single simulation, and the bottom left part shows the approximated LFP in orange superimposed on the fully simulated LFP in black.

The figure also shows the approximated LFP from as single simulation in orange superimposed on the fully simulated one in black. Apart from some very slight deviations, the approximation for this particular simulation is near perfect. The spike

histograms with which the kernels are convolved with are shown on the top. The oscillation in the population activities are clearly visible in the LFP. The approximation is far better for this network than what it was for the 8-population network analysed by Hagen et al. [15]. This could be due to the higher spatial complexity of their network and sparser firing leading to contributions that are not as well approximated by some population average.

Since the neurons of each population have the same average input, and therefore also roughly the same population activity, the inhibitory and excitatory kernels can be combined to a single kernel per channel encompassing the activity of both population. Scaling the parameter g will have slightly different effects on the channels. Since scaling g will only affect the inhibitory kernels, whose contribution to each channel relative to the excitatory contribution varies for each channel, scaling g will have larger effects on some channels than others.

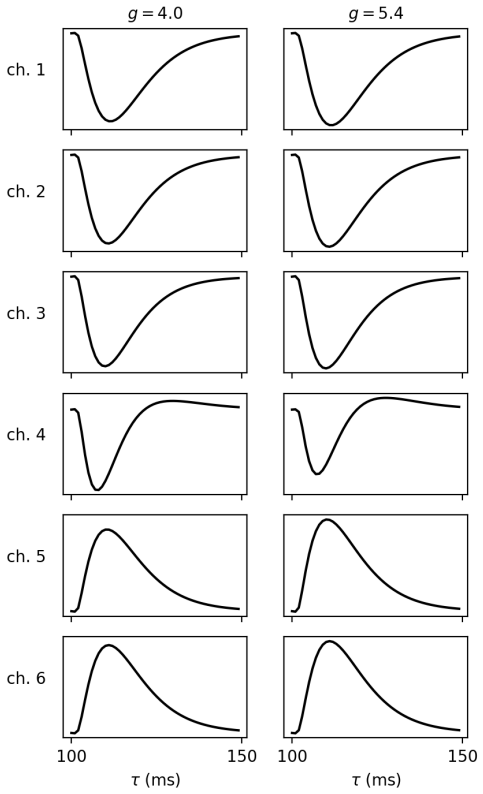


Figure 4.5: The shapes of the combined kernels of both populations for two values of g, weighted by the respective population sizes are plotted for each channel. The y-axes is equal row-wise, illustrating the different effect g has on the ch. 4 and ch. 5.

Figure 4.5 shows the combined kernels, weighted by their respective population sizes, for the lowest and highest values of g. As can be seen, g has little effect on channels 1-3 and 6, but on channel 4 and 5 the effect is more noticeable. Note that channel 4 is the only one where the inhibitory kernel has the opposite effect of the excitatory one, where increasing g decreases the amplitude of the combined kernel. Of course, the impact it has on the population activity will be much larger than the one it has directly on the LFP calculations.

The population approximation to the LFP for this network is not equally good for all parameter values. Figure 4.6 shows three plots of the LFP generated by the full simulation and the population activity approximation for three different lower values of g. When the network becomes almost fully synchronous, as it is when $g = 3.2$, a high frequency component in the approximation appears that is not present in the fully simulated LFP. This is due to the fact that there are some very small contributions still present at the time delay at which the kernels are cut, which add up when almost all the neurons fire simultaneously. This is irrelevant for the parameter ranges we are interested in. The two lower plots also show some deviations in the approximated LFP, but qualitatively they are more or less the same.

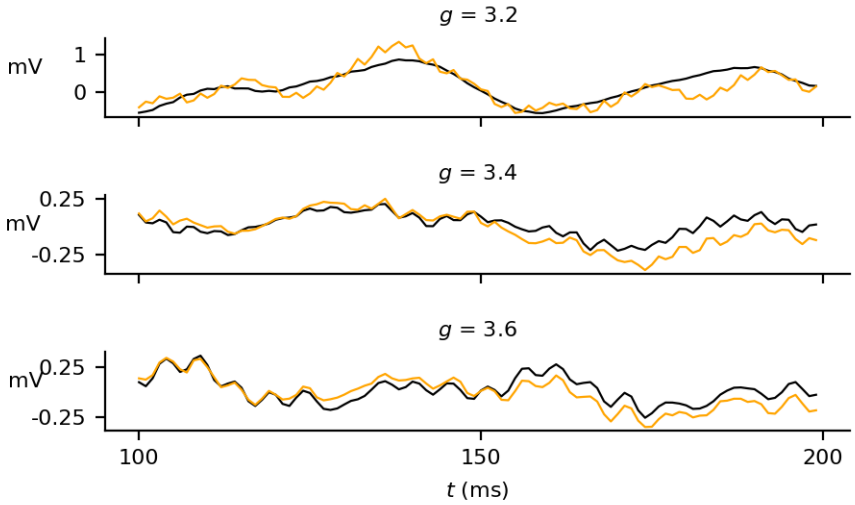


Figure 4.6: The population activity approximation of the ch. 5 LFP for three different simulations with increasing values of g. The orange lines show the approximations and the black lines are the fully simulated LFPs. The other parameters for these simulations are $J = 0.1$ mV and $\eta = 2.0$.

4.3 Parameter effects on LFP

To first give an idea of what the LFP signals looks like, Figure 4.7 shows three examples in the time domain. The upper one is from the synchronous regime, oscillating rapidly with the global activity. The two lower ones are both from different parts of the asynchronous regime, lacking any single dominating frequency.

The dimensionality of the parameter space and resulting LFPs make their visualization across parameters difficult. With three changing parameters and two-dimensional LFP signals, ideal figures would require five dimensions. Instead, to make things manageable, a single channel of the LFPs through a line in the parameter space are plotted to try to elucidate the effects each parameter has on the LFP, with the other two kept constant at low, intermediate and high values. The LFPs are shown in the frequency domain, and is averaged over all simulations done with each parameter combination. Channel 5 is chosen, as it is the one with the largest contributions from both the inhibitory and excitatory population.

In Figure 4.8 the PSD of all 8 LFPs arising from the varying values of g are plotted for low, intermediate and high values of η and J, indicated at the top and right hand side. The shade of gray indicates the value of g, with darker shades meaning higher values. This gives an idea of the diversity of the LFPs generated by these parameters, and the effect each parameter has on the LFP. For low values of J, shown in the leftmost column, the LFPs have a clear peak frequency, corresponding to the global oscillation of activity, which shifts toward lower frequencies as g increases. These are the ones closest to the synchronous regular state. As η increases, the peaks shift toward higher frequencies. These are all the same effects we saw in the network activities in Figure 4.2, and for low values of J, the LFP is dominated by the frequency of the synchronous global activity. These peak frequencies gradually disappear as J increases. As the value of J increases, the effect of η also decreases,

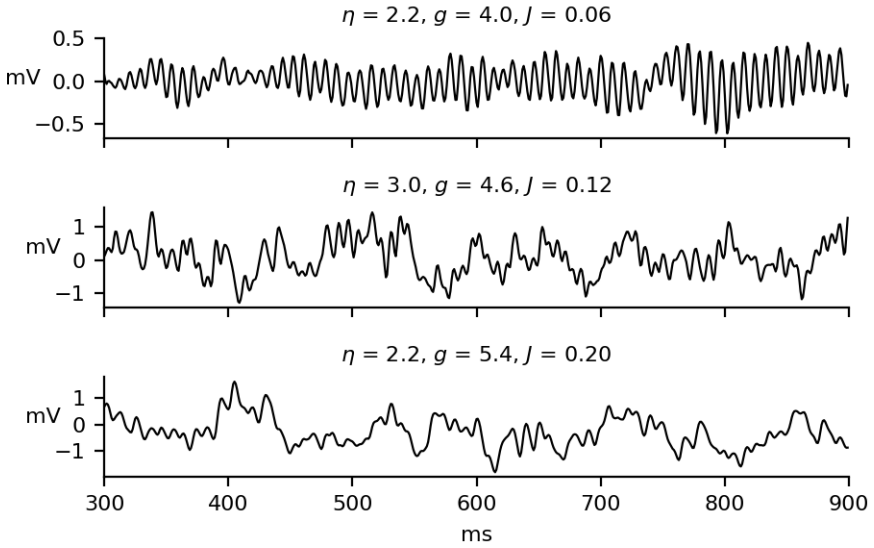


Figure 4.7: Three example LFPs from ch. 5 generated with different parameters. The parameter values are the same as the ones used in figure 4.1.

and for high values of J, the LFPs look very similar regardless of the value of η. The additional effect J has on the LFP is to increase the amplitude of the lower frequencies. This is a very marked effect, and for the three values shown here, they do not even overlap.

The parameters g and J change the resulting LFP in a more fundamental manner that η does. There are two reasons for this. Firstly, for the parameter g and J change the activity of the network in a more fundamental sense, in that they both facilitate the transition from the synchronous regular state to the asynchronous irregular state, as opposed to η, which mainly shifts the peak frequencies where they are present. Secondly, J and g have a direct effect on the generation of the LFP itself, in that they both directly affect the synaptic currents determining the LFPs, whereas η can only affect the LFP through the network activity. Recall that when mapping the population activities to the LFP, increasing J will scale all kernels by the same proportion, directly increasing the amplitudes of the LFP. As shown above in figure 4.5 changing g, however, will only scale the inhibitory kernels, which gives slightly different effects on each channel. The effect is quite small, however, and for channel 5 it should only slightly increase the amplitude of the LFP, so generally the direct effects of g on the calculation of the LFP is small compared to the effect it has through the network activity.

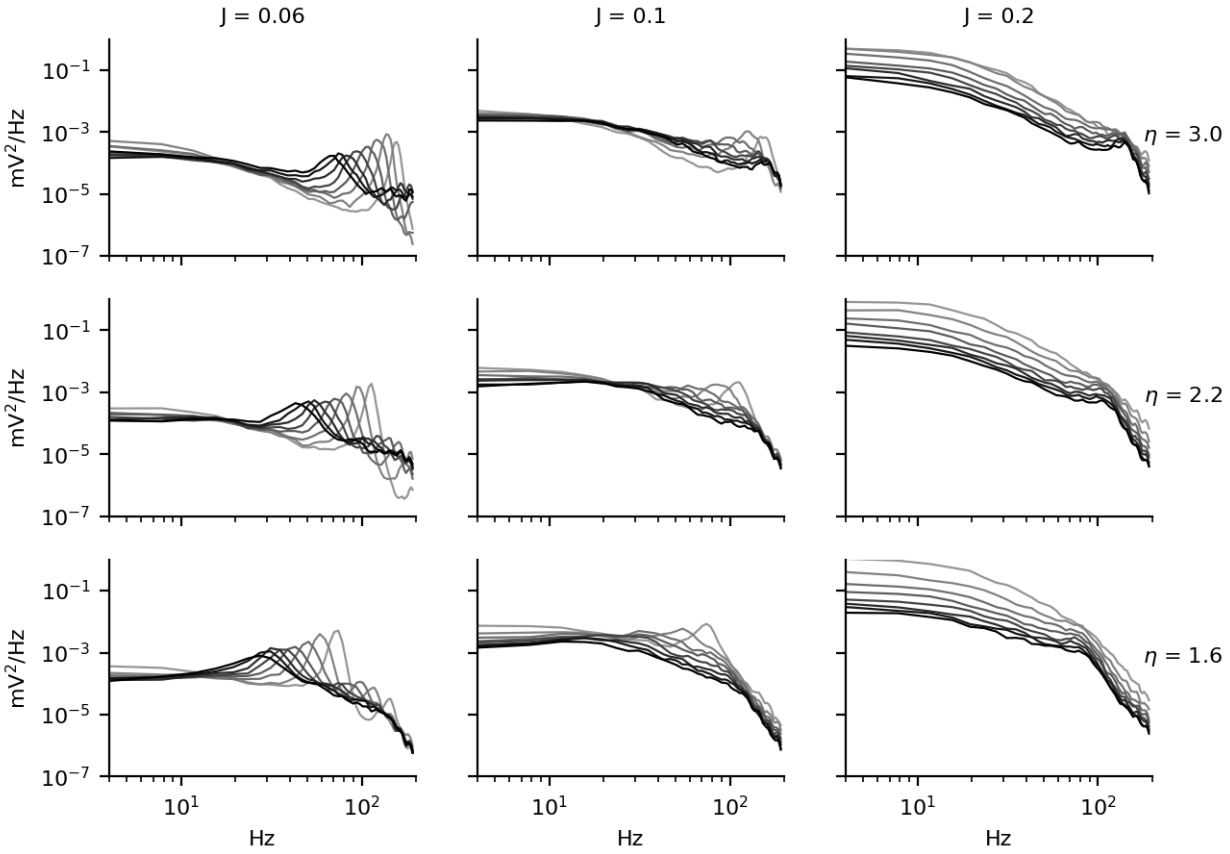


Figure 4.8: Each subplot shows the PSD of channel 5 of the LFPs for all 8 values of g, with fixed values of η and J incidated at the top and on the right. The value of g is indicated by the shade of gray, with darker meaning a higher value.

4.4 Training convolutional neural networks

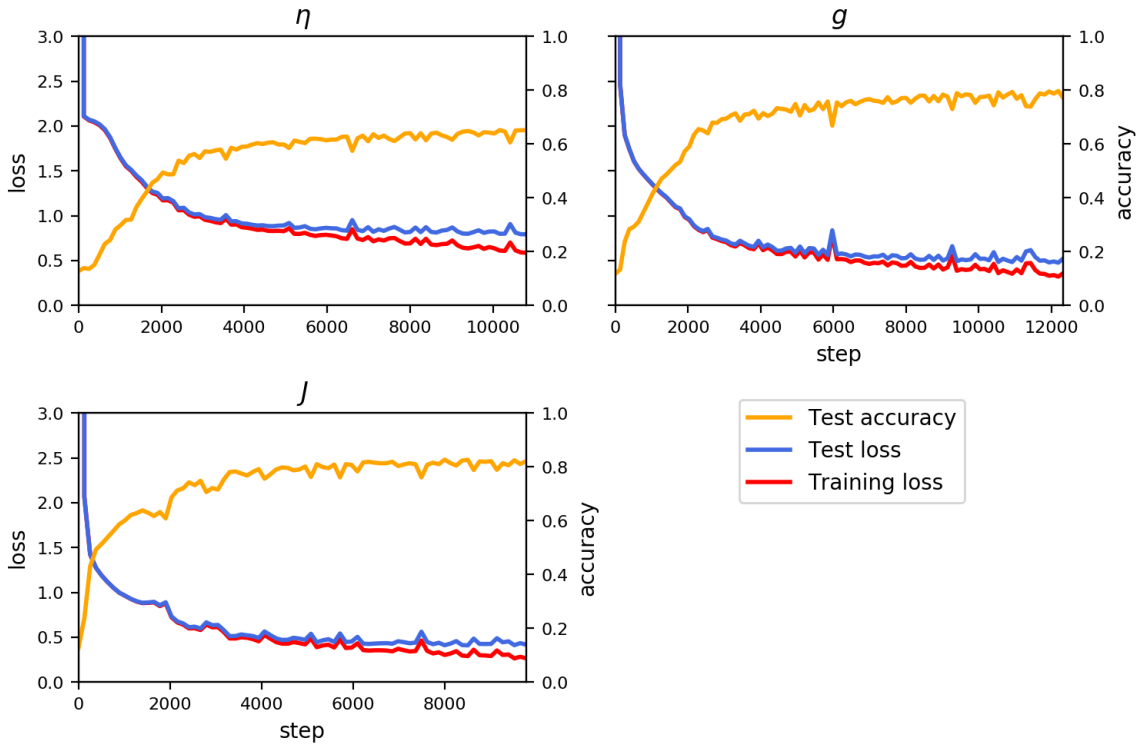


Figure 4.9: Classification training curves where the loss function smoothly converges to a local minimum. Orange lines are accuracies, measured as LFPs correctly binned to their parameter values. Blue lines are test and validation losses, and red lines are training losses. One step constitutes weight updates after a single batch. The Adam optimizer with the default parameters is used here, and a fixed learning rate of 10^{-4}. For the network classifying by value of η, a dropout of 0.6 was applied, but for the other two networks, dropout was not applied.

First, to give a better idea of the problem the network is faced with, consider Figure 4.8 again. For each parameter value, the network is trying to correctly bin all 64 possible parameter combinations containing that specific parameter values, while excluding all others. As an example, the leftmost column shows the PSDs of 24 of the 64 LFPs the network trained to classify J-values has to recognize as being in the same category, while all LFPs in the other two columns must be excluded. Note that these are very smooth curves, as they are the average LFPs over 12 different simulations, where each PSD has been obtained by Welch's method, resulting in much less noise than what the network is faced with. It will nevertheless serve as a guide to understanding the classification task.

Various network architectures were tried, and the one used for the training presented here is shown in Figure 3.2. A thorough and systematic architecture search was not performed due to limited resources, but it was generally found that with eight convolutional layers the network performed well, and simply adding more layers did not improve performance. Four max pooling layers with a window of 2 kept the input size to the fully connected layers at 18 per filter, allowing for relatively large batch sizes and number of filters without exceeding the memory limits. Dif-

ferent optimizers and weight initializations were also tried, leading to quite different results. Some difficulties were had avoiding getting trapped in local minimas. Typically, when the loss was minimized in an smooth fashion, as can be seen in Figure **??**, the network slowly converged to a local minimum. That particular training was done with the Adam optimizer, with the default parameter values, a learning rate of 10^{-4} without decay, and a batch size of 400. In contrast to this, the best performance was generally achieved during training in which the loss fluctuated a lot. An example of this is seen in Figure 4.10. Both weight initializations and optimizer parameters had a large effect on the stability.

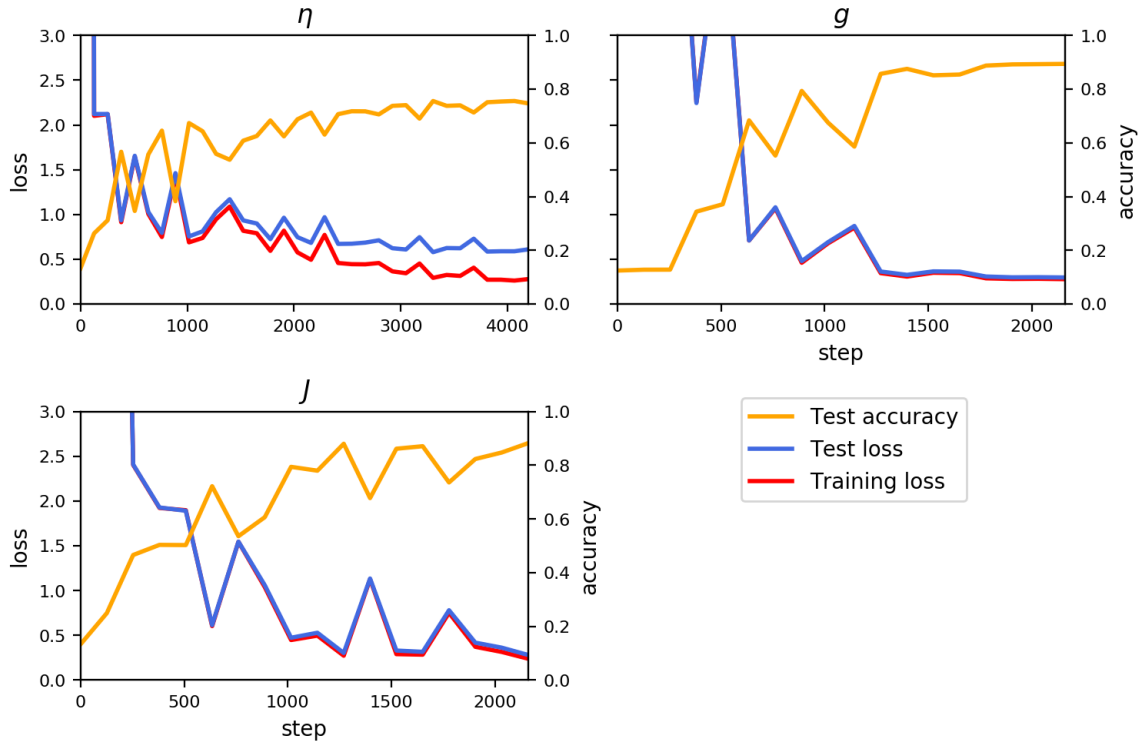


Figure 4.10: Training curves of the best achieved results. The network is the same as in figure 4.9, but with a different optimizer and initialization.

The best results were gained with a mini-batch stochastic gradient descent optimizer with Nesterov momentum [27]. With a momentum of 0.9, batch size of 400 and an exponentially decreasing learning rate, starting at 10^{-5}, decreasing by a factor 0.9 every 200 steps, i.e. slightly more often than every other epoch. The training curves can be seen in Figure 4.10. A strongly oscillating loss and accuracies can be observed, particularly for g and J, but they quickly reach a much higher accuracy and lower loss than the more stable counterpart in Figure 4.9, achieving an accuracy of around 75% for η predictions, and an accuracy of around 90% for g and J, at slightly more than 4,000 and 2,000 steps respectively. Overfitting is still present in the network classifying η values, but for g and J, there is no overfitting. This indicates that there is likely much room for improvements in the tuning of the optimizer and in the initialization of the network. Particularly, a more fine-grained hyperparameter search would likely have been fruitful. It could also be the case that the loss function is highly chaotic around the area giving the best results, and other architectures would be better suited, for instance introducing skip connections [28],

which typically gives a much smoother loss function [29], although it's difficult to see why this should be necessary with a relatively shallow network such as the one used here. It could also be that a smaller network ultimately could have provided better results with better tuned hyperparameters and initializations.

4.4.1 Classification accuracies

The accuracies shown in Figure 4.9 and 4.10 is the mean over the entire test dataset. Figure 4.11 shows the accuracies of all three networks as a function of each parameter. In other words, the upper plot shows the accuracies of η predictions as a function of the η, g and J.

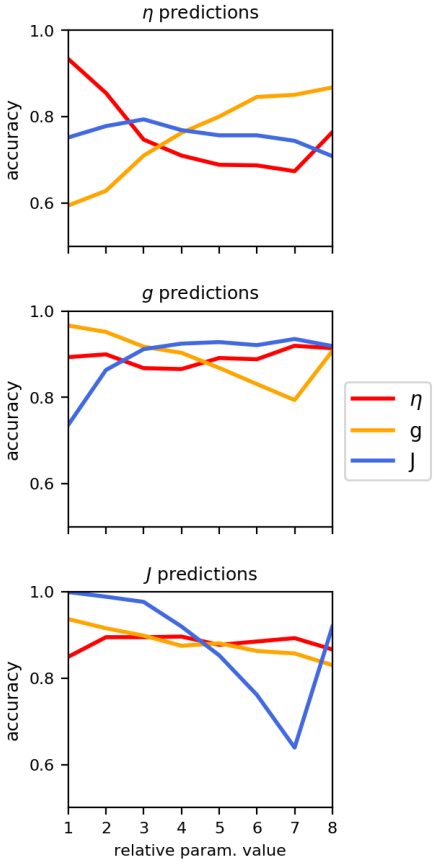


Figure 4.11: For each of the three networks classifying η, g and J, the accuracies are plotted as functions of the three parameters.

The parameter η is overall the most difficult to classify. Since η doesn't change the network activity in a fundamental way in the ranges of parameters used here, this is perhaps to be expected. For η predictions, the network performs far better in classifying the lower values than the higher. A similar effect can be seen in the network classifying g and J, where each network performs best at low values of the parameter it is trying to classify, and a sudden jump in performance for the very highest value. The η predictions are also more accurate for higher values of g, but it doesn't depend much on the value of J, so it's difficult to say why this should be, as both g and J play an important role in determining the activity state.

The difficulty of classifying the different parameters are dependent on different aspects of the LFP. While value of J has a clear effect on the accuracy of classifying g and J, it doesn't affect the predictions of η much. The accuracy of η predictions are strongly dependent on both η and g, while the accuracy of J predictions are mostly dependent on the value of J.

Generally, the difficulty of classification doesn't seem to be much dependent on where the network is on the synchronous regular-asynchronous irregular axis, as the networks doesn't show a clear tendency to perform better or worse for increasing values of both g and J, but rather some other factors seem to determine the difficulty. The accuracies broken down by each parameter combination can be seen in Figure 4.12.

When the network makes a prediction error, it almost always predicts one of the immediately neighbouring values. Specifically, for g, out of the 1183 erroneous classifications made, all but two were classified as being in an immediately neighbouring bin. For η, the corresponding numbers were 124 out of 2699, and for J, 4 out of

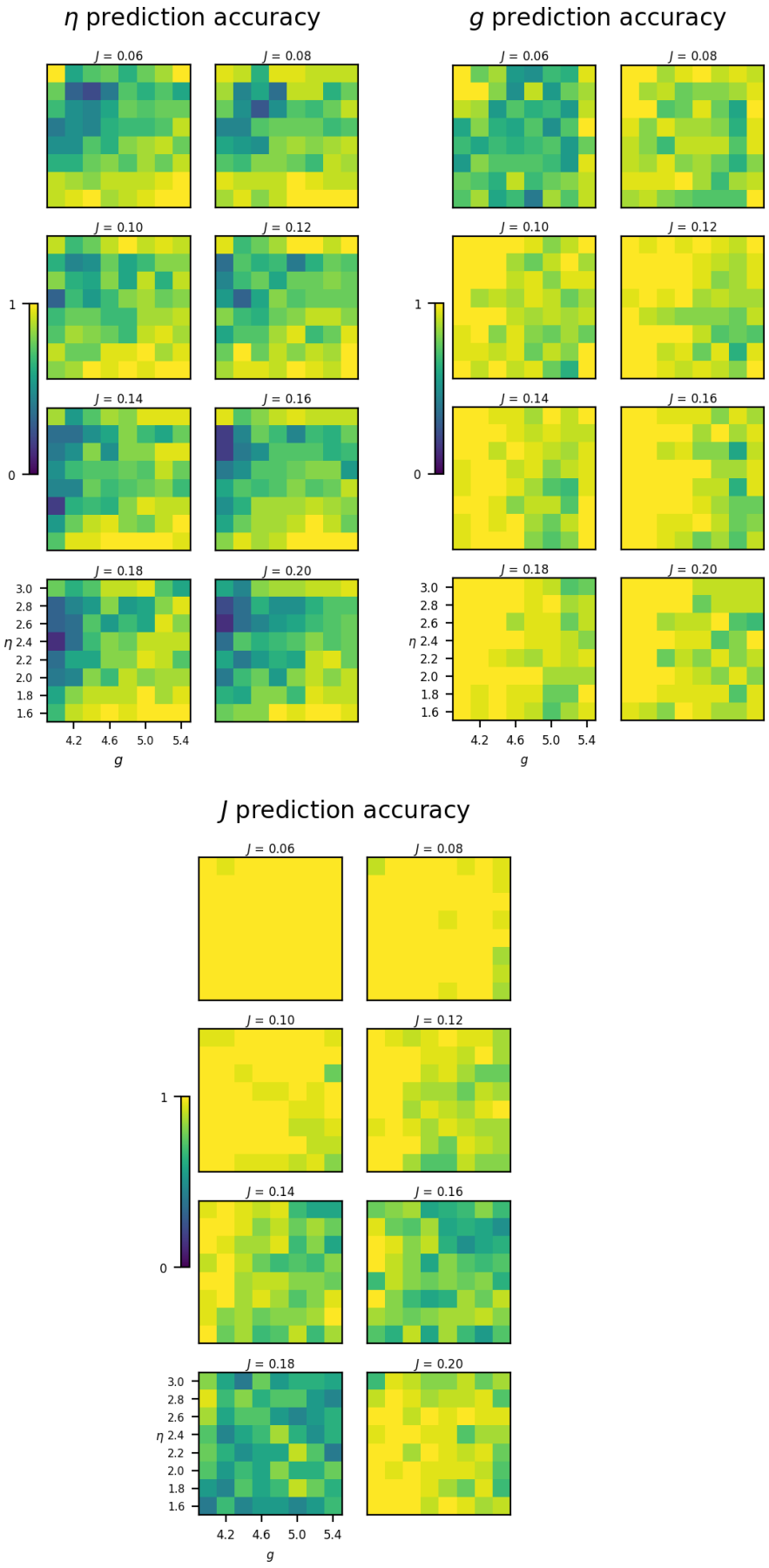


Figure 4.12: Prediction accuracies broken down by each parameter combination.

1328. Unsurprisingly, the most difficult task is separating the most closely related LFPs.

4.4.2 Inspecting the first filters

Trying to figure out how a convolutional neural network works is no easy task. In the deeper layers, the filters are looking for features in the features detected in the previous layers, so inspecting them directly doesn't give much intuition. In the very first layer, the filters can be directly inspected to get an idea of what features they pick up. The upper part of Figure 4.13 shows channel 3 from 121 out of the 256 filters in the very first layer in the network classifying values of g. In other words, these are some of the kernels that are cross-correlated with channel 3 of the input LFP.

It is immediately clear that there are a variety of different frequencies present in the filters, and it seems that different filters pick up different frequencies. Some of the filters seem to pick up a single frequency where others pick up combinations of frequencies. Many of them look similar, which could suggest that there are redundancies and perhaps the number of filters could be reduced, but the different input channels are not taken into account.

The bottom part of the figure shows the all 6 channels of 10 filters. Since the frequency content of the channels are largely the same, one would perhaps expect the channels of single filters in the first layer to be similarly shaped, but this looks largely not to be the case. What could perhaps be the case instead is that the channels of the first filters act as logical gates, enabling a single filter to determine the presence of certain features and absence of others. If the filter channels did indeed have the same shape, the different channels wouldn't provide much new information, but it is ultimately difficult to say by inspecting the filters alone.

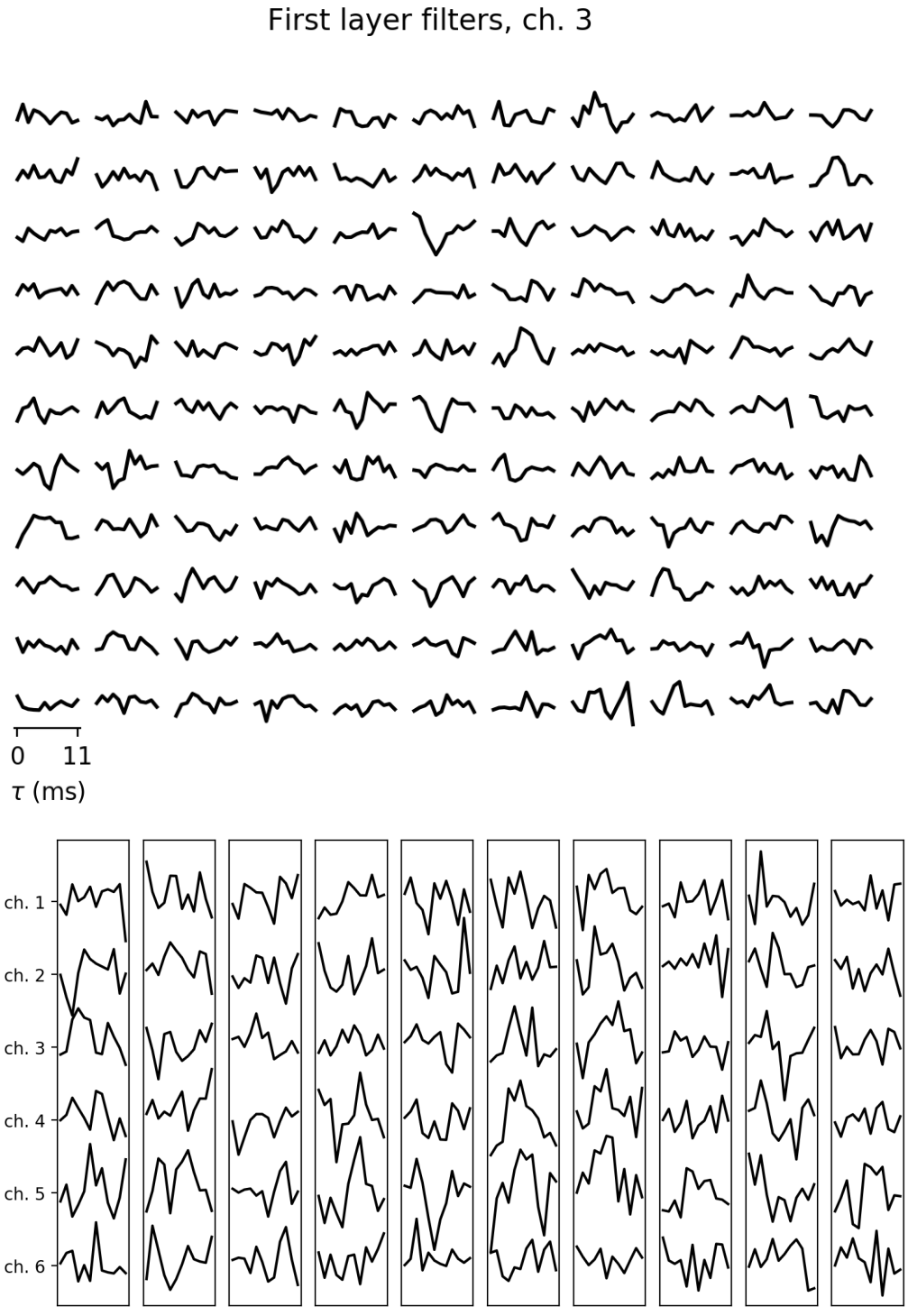


Figure 4.13: The upper part shows channel 3 of 121 of the 256 filters in the first convolutional layer. The lower part shows all 6 channels from 10 of the filters.

Chapter 5

Discussion

The aim of this project has been to explore the possibilities of utilizing convolutional neural networks on local field potentials, or brain signals in general. To this end, we constructed a simple point neuron network for which we simulated the LFP generated by its activity for different values of some key parameters of the network. The task of extracting parameter values of a spiking point-neuron network by using the LFP was chosen as a starting point to test the applicability convolutional neural networks on this type of signal. The simplicity of the model network allows a thorough understanding of the link between the parameters of the network and the resulting network activity and LFP. The CNN was able to quite accurately classify the LFPs based on their parameter values. Each CNN was trained to classify a single parameter having eight possible values, regardless of the value of the other two parameters. The network classifying the parameter η achieved an accuracy of 75%, while the networks classifying g and J achieved an accuracy of 90%. This implies that for this particular network, the LFP contains very specific information about these parameters. There are, however, many parameters of the network that we did not vary, for instance the synaptic time delay and synaptic time constant, and for the three parameters we did vary, they were still constant for all neurons in the network. If more parameters were varied, a more diverse set of LFPs would be generated, and would make the classification task much more difficult, so the results achieved here is specific to this particular network and parameters used. Nevertheless, the results imply that convolutional neural networks are able to extract information from the LFP signals, which indicates that they can potentially play an important role in interpreting this type of signals.

The model used in this project was a simple one, and has served as a proof of concept showing that for this network, very specific information about the spiking network is conserved in the LFP, and that convolutional neural networks are able to extract that information. A more thorough analysis of the features used by the convolutional neural networks to make the classifications could provide additional information about the nature of the signature of the parameters on the LFP. Particularly, an analysis of the activations in the network when presented with different LFPs could be done. This could potentially allow a mapping between each particular parameter value and the subset of filters its resulting LFP activates, giving insight into what particular features of the LFP are actually used to make the classifications, and give an idea of how the parameter values are 'encoded' in the LFP,

so to speak.

The work done in this project has laid some groundwork for further investigations of utilizing deep learning on brain signals, and a natural extension of the work done here would be to apply the same methods on larger, more realistic network models, such as the one used by Hagen et al. (2016), which encompasses an entire cortical column, and shows much more realistic firing statistics and LFPs [15]. An extension of the approach taken here to more realistic network models such as that one could potentially give insights on the LFPs generated by real cortical networks.

Bibliography

[1] Suzana Herculano-Houzel. "The human brain in numbers: a linearly scaled-up primate brain". In: *Frontiers in Human Neuroscience* (2009).

[2] Javier Defelipe, Lidia Alonso-Nanclares, and Jon Arellano. "Microstructure of the neocortex: Comparative aspects". In: 31 (2002), pp. 299–316.

[3] Simona Lodato and Paola Arlotta. "Generating Neuronal Diversity in the Mammalian Cerebral Cortex". In: *Annual Review of Cell and Developmental Biology* 31.1 (2015), pp. 699–720.

[4] V B Mountcastle. "The columnar organization of the neocortex. Brain 120(Part 4):701-722". In: 120 (Pt 4) (1997), pp. 701–22.

[5] Paul L. Nunez and Ramesh Srinivasan. *Electric Fields of the Brain: The Neurophysics of EEG*. Oxford University Press, USA, 2006.

[6] Gaute T. Einevoll et al. "Local Field Potentials Biophysical Origin and Analysis". In: *Principles of Neural Coding*. 2013.

[7] Klas H. Pettersen et al. "Extracellular spikes and CSD". In: *Handbook of Neural Activity Measurement*. 2012.

[8] Klas H. Pettersen and Gaute T. Einevoll. "Amplitude Variability and Extracellular Low-Pass Filtering of Neuronal Spikes()". In: *Biophys J* 94.3 (2008), pp. 784–802.

[9] Gaute T. Einevoll et al. "Modelling and analysis of local field potentials for studying the function of cortical circuits". In: *Nature Reviews Neuroscience* 14 (2013). Review Article.

[10] Henrik Lindén et al. "Modeling the Spatial Reach of the LFP". In: *Neuron* 72 (5 2011), pp. 859–872.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521 (2015), pp.436–444.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Curran Associates Inc., 2012, pp. 1097–1105.

[13] Ronan Collobert and Jason Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. 2008, pp. 160–167.

[14] Nicolas Brunel. "Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons". In: *Journal of Computational Neuroscience* 8 (2000), pp. 182–208.

[15] Espen Hagen et al. "Hybrid Scheme for Modeling Local Field Potentials from Point-Neuron Networks". In: *Cerebral Cortex* 26.12 (2016), pp. 4461–4496.

[16] David Sterratt et al. *Principles of Computational Modelling in Neuroscience*. Cambridge University Press, 2011.

[17] E.R. Kandel et al. *Principles of Neural Science, Fifth Edition*. McGraw-Hill Education, 2012. ISBN: 9780071810012.

[18] Tomonori Takeuchi, Adrian J. Duszkiewicz, and Richard G. M. Morris. "The synaptic plasticity and memory hypothesis: encoding, storage and persistence". In: *Philos Trans R Soc Lond B Biol Sci* 369.1633 (2014).

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

[20] Henrik Lindén et al. "LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons". In: *Frontiers in Neuroinformatics* 7 (2014).

[21] Stéphanie Miceli et al. "Impedance Spectrum in Cortical Tissue: Implications for Propagation of LFP Signals on the Microscopic Level". In: *eNeuro* 4.1 (2017).

[22] Michael A. Nielsen. *Neural Networks and Deep Learning*. 2015.

[23] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015).

[24] Anders Krogh and John A. Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS'91. Denver, Colorado, 1991, pp. 950–957. ISBN: 1-55860-222-4.

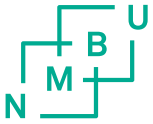
[25] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958. ISSN: 1532-4435.

[26] Susanne Kunkel et al. "NEST 2.12.0". In: (2017).

[27] I Sutskever et al. "On the importance of initialization and momentum in deep learning". In: (2013), pp. 1139–1147.

[28] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015).

[29] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2018.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway