NTNU
Det skapende universitet

Kunstige nevrale nettverk for deteksjon av aksjonspotensialer i kalsiumbildedata

Mikkel Antonsen

Master i informatikk
Innlevert: mars 2016
Hovedveileder: Keith Downing, IDI
Medveileder: Benjamin Adric Dunn, KAVLI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Artificial Neural networks for Spike Inference in Calcium Imaging Data

Mikkel Antonsen

March 2016

MASTER THESIS

Department of Computer and Information Science

Norwegian University of Science and Technology

Supervisor 1: Professor Keith Downing

Supervisor 2: Post Doctoral Researcher Benjamin Dunn

Abstract

Calcium imaging is an important tool for systems neuroscience to study large populations of neurons. For calcium imaging to be useful one has to uncover the neuronal activity that caused the signal one observes. Many techniques have been proposed to solve this problem. Typically they exploit a mathematical model of how intracellular calcium in neurons behaves or by detecting the sharp rise in intracellular calcium concentration associated with neuronal activity. This thesis explores if artificial neural networks (ANNs), a model free approach, can do spike inference better by not assuming anything about the underlying dynamics of the signal.

The task was solved by setting the ANN up as an autoencoder and learning how to detect the spikes as an unsupervised task. By regularizing the activity of the hidden layers of the encoder part of the network we show that it is able to learn how to represent the input as a spike train.

The spike train reconstruction efficiency for Fast-OOPSI and the model free ANN was compared. Fast-OOPSI was found to perform slightly better on simulated data while the ANN performed slightly better on real data.

Preface

This document constitutes the Master's thesis of Mikkel Antonsen . It was written in the period March 2015 to April 2016.

The thesis assumes that the reader is familiar with Computer Science, basic Artificial Intelligence techniques, and preferably basic Neuroscience.

Trondheim, 2016-03-01

Mikkel Antonsen

Acknowledgment

I would like to thank my the following people my brother for invaluable and unrelenting support during this ordeal. My parents for everything, without which I would not be where I am today. Jonathan Whitlock for graciously inviting me into his group to do this thesis and Benjamin Dunn for suggesting the topic. Everyone at the Kavli Institute for Systems Neuroscience for welcoming me so warmly. Dromedar Kaffebar for keeping me caffeinated and inspired.

I also want to acknowledge the contributions of my supervisors Keith Downing and Benjamin Dunn.

M.A.

Chapter 1

Acronyms

ANN Artificial Neural Network

CNN Convolutional Neural Network

AE Autoencoder

DAE Denoising Autoencoder

SDAE Stacked Denoising Autoencoder

SVM Support Vector Machine

PCA Principle Component Analysis

MSE Mean Squared Error

AP Action Potential

TPR True Positive Rate

FDR/FPR False Detecation Rate/False Positive Rate

WTA Winner Takes All

Fast-OOPSI Fast Optimal OPtic Spike Inference

FP False Positive

TP True Positive

FN False Negative

TN True Negative

Contents

Chapter 2

Introduction

2.1 Motivation

The number of neurons neuroscientists have been able to record activity from has followed a curve similar to that of Moore's law. From Hodkin and Huxley recording a single squid axon in 1952 to now where we can record hundreds of neurons concurrently. Recording the activity of this many neurons has been made possible by a technique called calcium imaging.

Unfortunately, the quality of the measured activity has decreased proportionately with the number of neurons recorded. A considerable effort has therefore been put into developing methods to accurately infer the underlying neuronal activity from calcium imaging data.

This work will investigate how well Artificial Neural Networks (ANNs) do compared to state of the art methods.

Modern ANNs are very loosely inspired by how the real brain works, where nodes (corresponding to neurons) exchange information using edges (corresponding to dendrites and axons). Each node in the network integrates its input by multiplying the connected node's activity with the weight of the connection they share. By modifying the number of nodes, connectivity, and the weights of the edges, ANNs have been shown to be able to approximate any function Hornik (1991).

Modern ANNs have been applied successfully in a wide variety of problems where one tries to learn non-linear mappings between high-dimensional and/or highly variational input to output. Examples of successful applications include object recognition Krizhevsky

et al. (2012), speech recognition Hannun et al. (2014), and time-series analysis Dalto (2014), often achieving state-of-the-art performance.

The problem of inferring spikes from a fluorescent trace obtained through calcium imaging has been thoroughly explored in literature Ziv et al. (2013), Lütcke et al. (2013), Grewe et al. (2010), Kerr et al. (2005), Peron et al. (2015), Sato et al. (2007), Yaksi and Friedrich (2006), Vogelstein et al. (2009), Vogelstein et al. (2010).

Attempts to solve this problem broadly fall into the one of the following categories. Template matching, deconvolution, or various machine learning approaches. None of these methods are optimal. Their weaknesses usually include one or more of the following. They often require hand tuning of parameters, are not resistant to noise, unable to deal with the highly non-linear relationship between neuronal spikes and the resulting fluorescent trace, or make a lot of assumptions about the underlying dynamics of the signal causing them to underperform on real data.

In order to further complicate the issue, there are a lot of possible variations with the system being studied, which all have big effects on how the final data looks. What type of neuron or what brain area is being studied, what animal, virus, and virus preparation changes with each experiment. The exact location of the injection relative to the cell and how the virus reaches the cell will vary. The fluorescence signal changes with where the cell is relative to the focal plane and changes naturally over time with the health of the cell.

It is hard to make a good model taking all these factors into account and template matching becomes very cumbersome to work with if you want to adapt it to all these features.

It is therefore interesting to investigate if ANNs are able explain the fluorescent imaging data better by letting the data speak for itself, not making any assumptions about the underlying dynamics of the signal.

Recording the activity of many neurons concurrently is one of the big challenges in neuroscience today. Calcium imaging allows recording hundreds of cells at the time, more than any other technique. Accurately infering the neuronal activity from the calcium data is therefore an important topic to study.

2.2 Problem formulation

Spike inference is the problem of finding exact spike times of neurons from a noisy fluorescent trace obtained by photographing the cell. This thesis will explore how ANNs can be used for spike inference as an unsupervised task. The performance will be compared to existing techniques.

The motivation for solving this as an unsupervised task is that it is impossible to obtain ground truth in real experimental conditions.

This thesis will try to answer the following research questions.

- Can an ANN infer spike times in an unsupervised fashion?
- How does this method compare to previously established approaches described in literature?

2.3 Context

Available datasets The real data used comes from the publications Chen et al. (2013a). The data is provided by Collaborative Research in computational Neuroscience where researchers share data for the betterment of all.

Contents of datasets The datasets consists of *in-vitro* imaging data of three different GCaMP calcium indicators. In addition to imaging data there is simultaneous loose-seal cell-attached recordings of the cells, which accurately report the exact membrane potential, serving as ground truth.

2.4 Approach

The goal for this work is to investigate if a data-driven, model free machine learning technique can outperform other approaches for spike inference.

By allowing data to speak for itself, we might be able to do better spike inference by making fewer assumptions about the underlying dynamics of the system.

We will evaluate different ANN architectures and see how well they solve the spike inference problem. To evaluate the effectiveness we will also compare how well the ANN does on

spike inference versus other established methods. We will test against both real data where the ground truth is known and simulated data.

Chapter 3

Background

This chapter will serve as an introduction to some of the electrophysiology of a neuron and calcium imaging techniques. Familiarity with these subjects is important as they constitute the domain of this thesis. Furthermore, previously proposed spike inference techniques will be presented along with relevant methods for unsupervised learning with artifical neural networks.

3.1 A Short Introduction to a Neurons Electrophysiology

A resting neuron has a membrane potential of around -40 to -90mV. When the neuron is stimulated this membrane potential changes. If the membrane potential is depolarized to a certain threshold an action potential occurs.

An action potential, also referred to as a spike, is a fast depolarization of the cell followed by a repolarization. At its peak it will reach 40mV and then return back to the resting membrane potential. A spike is the smallest unit of information in the nervous system. For example, a neuron spiking exclusively in response to certain stimuli, such as an organism's position , allows the organism to code for that stimuli.

When the cell membrane reaches threshold potential, voltage-dependent Na^+ gates open. An influx of positively charged ions causes the membrane potential to rise. At a certain membrane potential voltage-dependent K^+ gates open, pushing K^+ out of the cell. This brings the cell back to resting membrane potential.

Other ions do not drive the membrane potential with their charge, but act as secondary messengers. One such ion is Ca^{2+}, the hero of this story.

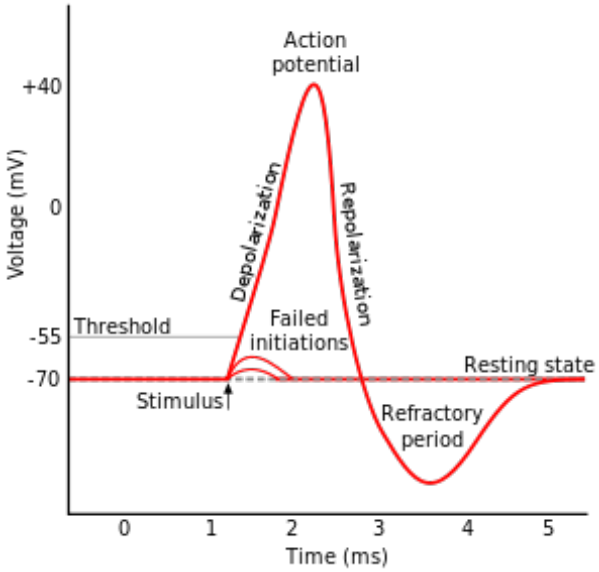

Figure 3.1: The stereotypical action potential. This whole event happens so fast for our purposes we look at it as an instantaneous event. This picture was obtained from the wikipedia page on action potentials.

Ca^{2+} has a myriad of functions to keep a neuron working normally. What these functions are do not concern us particularly, the important point is that a resting neuron has almost no intracellular calcium. The Ca^{2+} concentration is around 10^{-9} Mol, compared to the extracellular environment where it is around 10^{-3}. Right after a spike a lot of this extracellular calcium will enter the cell. We will see later how this influx of calcium right after a spike can be detected and in turn allow us to infer when a neuron has spiked.

These action potentials are thought to be the driving force behind our cognitive function. Finding the spike trains (how a neuron spikes through time) and how they relate to behavior is one of main efforts of Neuroscience today.

3.2 A Slightly Longer Introduction to Calcium Imaging

This thesis is about machine learning in a neuroscientific context. Therefore it is important we have a sound understanding of how the data is obtained and how it is currently analysed. After reading this chapter the reader will know what imaging techniques are used in neuroscience, how they are used, hurdles to overcome while using them and how the data is analysed.

3.2.1 Calcium Imaging at a Glance

Calcium imaging is a technique that allows recording the activity of a cell indirectly by looking at the intracellular calcium concentration. One way to measure the calcium concentration is by inserting a calcium indicator into the cell. The calcium indicator binds free intracellular calcium and emits detectable photons when exposed to light. Since the amount of light emitted by the cell is proportional to the calcium concentration inside the cell, it is possible to infer the calcium concentration with the light that is detected. Knowing the intracellular calcium concentration, it is possible to infer when a spike has happened.

When it comes to inserting the calcium indicator into the cell there are two main approaches. A virus can transport the indicator into a cell. Alternatively you can make transgenic animals, which are animals that have had foreign DNA injected into their genome. These transgenic animals express the indicator genetically in the cells.

There are several ways of doing calcium imaging each with their own advantages and disadvantages. They differ mostly in practicality and signal-to-noise ratio, and will be described below.

For all techniques, you need access to the tissue of interest. Usually this can be be obtained by removing the top of the cranium of a live animal replacing it with a hat that contains a small camera. The emitted light is recorded by the camera and one tries to infer the spike train that caused the signal one observes. In the context of this thesis though, the brain cells were recorded *in-vitro*.

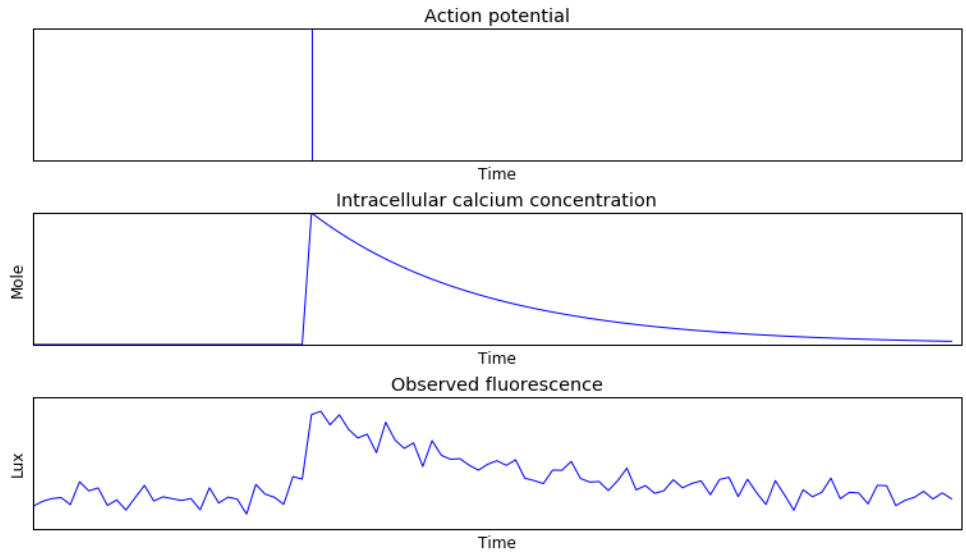

Figure 3.2: Top: An imagined neurons spike train through time. Each bar corresponds to an event. Middle: This is what we refer to as the the unitary calcium trace or stereotypical calcium transient. When describing this shape we are interested in the rise amplitude and decay time. Bottom: The light signal the camera sensor might register for the event.

3.2.2 Mathematical model of calcium imaging

I will now present a simple mathematical model that describes the relationship between the spikes, intracellular calcium, and the light that is detected when doing calcium imaging. We will be using the model presented in Vogelstein et al. (2010). This model will be used to make the simulated data.

First we relate the Ca^{2+} concentration at time t with the concentration at time $t-1$ by the following equation.

$$[Ca^{2+}]_t - [Ca^{2+}]_{t-1} = -\frac{\Delta}{\tau}([Ca^{2+}]_{t-1} - [Ca^{2+}]_b) + An_t + \sigma_c\sqrt{\Delta}\epsilon_{c,t} \tag{3.1}$$

where Δ is the frame rate at which the signal is recorded, τ is the decay rate, $[Ca^{2+}]_b$ is the baseline fluorescence, A is the spike amplitude, n_t is a boolean that indicates whether a spike happened at time t, σ scales the noise, and $\epsilon_{c,t}$ is a Gaussian with mean zero and standard deviation 1.

In words, this equation states that the difference in intracellular calcium from time $t-1$ to time t can be explained by an exponential decay by constant τ and adding the effect of any spikes that happened. So in the absence of a spike the concentration will be slightly lower and in the presence of a spike, the concentration will jump according to the A parameter.

In figure 3.2 three time series are presented. The topmost series indicates when in time the spike happened. In the middle series we see how the calcium concentration inside the cell rises and decays exponetially in response to the spike. The bottom shows the time series that the camera sensor actually detects.

This model assumes that the intracellular calcium level jumps instantaneously after a spike, that every spike jump is the same size, and models the extremely rich calcium extrusion and buffering dynamics with a single average time constant. Noise is obviously assumed to be Gaussian.

Furthermore we have to relate the intracellular calcium dynamics to the observed fluorescent traces. The following equation describes observed fluorescence F as a product of the intracellular calcium $[Ca^{2+}]$. β can be interpreted as the baseline fluorescence in the tissue.

$$F_t = \alpha[Ca^{2+}]_t + \beta + \sigma_F \epsilon_{F,t} \tag{3.2}$$

Where α scales the signal, β decides the offset, and σ_F scales the noise, and $\epsilon_{F,t}$ is the unit Gaussian. This equation states that the observed fluorescence is proportional to the intracellular calcium concentration with some scaling and added noise.

It is important to mention that, as all models, this one fails to capture a lot of the nuance of the actual underlying mechanics of the cell.

3.2.3 Recording Cell Activity with Calcium Imaging

Depending on the nature of the experiment being conducted there are a few alternatives in how you register the emitted light and what fluorophore to use.

For registering the light, the choice is between confocal, two-photon, and widefield microscopy. The tradeoff between the three lies in how noisy the resulting pictures are and how big the microscope is. The two first are a lot less noisy but animals are unable to move freely because of the size of the microscope.

The fluorophore is a chemical compound that re-emits light upon excitation. The re-emitted light is detectable because it has a different wavelength than the photon that caused the excitation. Different fluorophores have different properties. The most important properties for this thesis are the rise and decay time. These correspond to how fast the fluorphore binds to and releases intracellular calcium. It is important to mention that the rise is so sud-

den, it is usually modelled as being instantanious. The decay time is modeled as exponential decay, as seen in middle of figure 3.2.

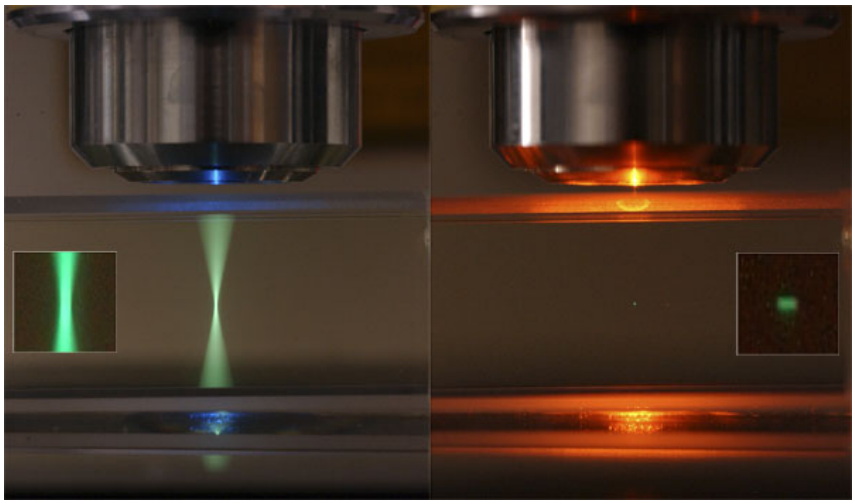

Figure 3.3: The green light indicates the area which will be registered by the camera. We see on the right the place with the highest density of photons, the middle of the X, is the only part being registered. This means that the activity of the tissue foreground or background is left out. Left: Confocal microscope. Right: two-photon microscope.

To obtain the ground-truth of what the membrane potential was exactly, one can perform patch-clamp experiments. The patch-clamp measures the membrane potential directly by inserting a needle into the cell. This will also destroy the cell afterwards.

3.2.4 Two-photon microscopes

Two-photon microscopes take advantage of a phenomenon described by Maria Goeppert-Mayer where exactly two photons of similar frequencies can excite a molecule from a ground state to a higher energy state. The excitation only happens if both photons hit the molecule at the same time. The molecule is excited to a higher energy state equal to the sum of the energy of both photons. The excitation is similar to that of a single photon, except two photons are absorbed instead of one.

Since the probability of two photons exciting the same fluorophore at the same time is extremely low, it will only happen in the focal plane where the photon density is highest. This means that only a very small part of the tissue is imaged at a time. A full picture is made by moving the focal point around in a plane allowing different sections of the brain to be pictured individually.

Another advantage is that photobleaching (which occurs when a fluophore is excited for

a long time and loses its ability to fluoresce) does not happen as only a small volume is excited at the time. A comparison between what is in focus in a two-photon and regular confocal microscope can be seen in figure 3.3. Here we see the two-photon confocal microscope on the right, where the green part is the total volume being excited by the microscope.

3.2.5 Analysis of Calcium Imaging Data

When you have the video recording of cells firing, subsequent analysis can be described as a two step process.

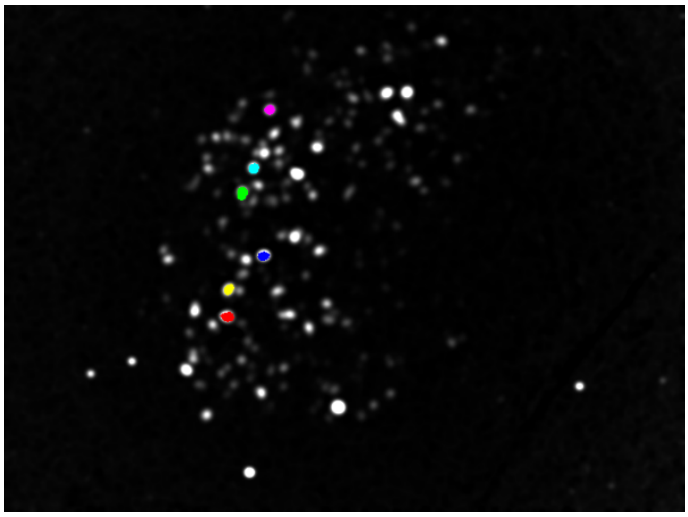

Figure 3.4: A projection of all the calcium activity of an entire movie onto one frame.

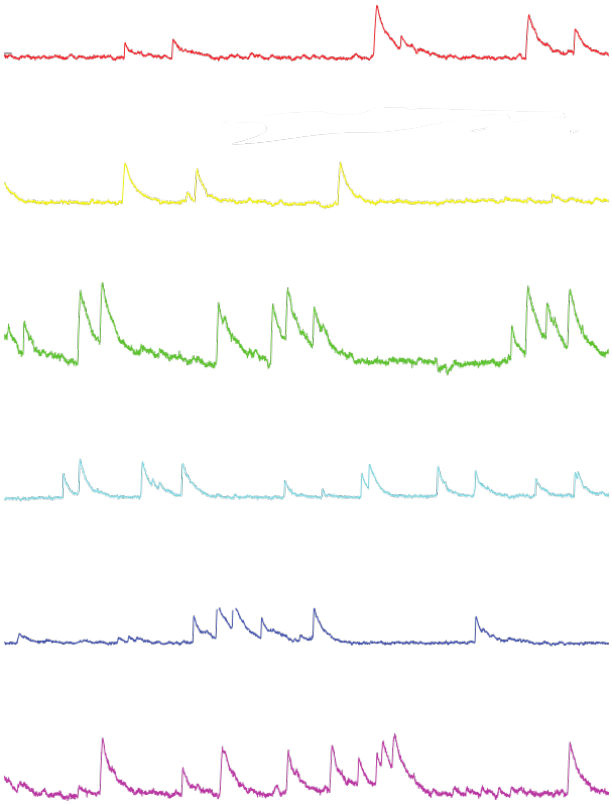

Figure 3.5: The calcium traces through time corresponding to the colored cells in figure 3.4.

First we want to reduce each cell in the recording into a one-dimensional vector. Each entry in this vector F_t should correspond to the amount of light being emitted by the cell at frame t from the movie. Six examples of what this F_t vector would look like can be seen in figure 3.5.

To find F_t we need to know the spatial extent of individual neurons within the movie. The location of a neuron can be described as binary mask over the picture, where 1 indicates where the neuron is. For example, each of the colored areas in figure 3.4 could be a spatial mask for a single neuron.

With these filters we reduce each cell in the movie to a vector F_t. Take one filter corresponding to one cell as an example. The binary filter is multiplied with each frame in the movie. This will render all pixels that do not belong to the cell zeroed out. For each frame t in the movie, the average light intensity for the all the remaining pixels is assigned to F_t, which is a vector of length t. All subsequent analysis is done on these F_t vectors.

The second part of the analysis is finding individual spikes underlying F_t. This is the problem this thesis will attempt to solve. A wide variety of previously proposed solutions to this problem will be discussed in section 3.3.

The raw F_t traces are almost never used directly for analysis. Typically one calculates the $\Delta F/F$, which is the change in fluorescence divided by the baseline fluorescence. If the baseline fluorescence is unknown, it is estimated as the average of the entire trace.

Alternatively one can use an average of a moving time window. This measure it more useful because it is the change in intracellular calcium that indicates activity not total fluorescence.

3.2.6 Practical Issues with Calcium Imaging

The brain is a volume with the x and y axis denoting the imaging plane and the z axis the depth of the brain. This means that two neurons can overlap spatially, completely or partially, in the picture. They share the same xy coordinates but have different z coordinates. This is a form of noise as we want the signal of one neuron not the conflation of two or more. This problem is extremely prevalent in widefield microscopy as out-of-focus light from the entire brain volume will hit the lens at all times.

Other sources of noise include neurpil (unmyelinated axons, dendrites and glial cell processes). They show up in pictures because the fluorophore will not exclusively be loaded into neurons. Different types of vascularities, such as blood vessels, might also be present and block or scatter light.

In addition there is also shot noise. The exact number of photons hitting the lens per time bin will vary from frame to frame, even if the light intensity is constant. This is because light is quantal, being spread probabilistically, arriving as discrete photons on the lens. This is a larger problem in the two-photon or confocal because the total number of photons recorded is lower.

One of the most important jobs a neuron does to stay healthy is to regulate its intracellular calcium concentration properly. Inside a neuron the fluorphore acts as a buffer for calcium. This buffer disturbs the natural calcium concentration which over time will kill the cell. This effect is called phototoxicity and will in time make cells useless to study.

It is also possible to have a working neuron where the fluorophore inside is broken down making it unable to fluoresce. This is called photobleaching and can occur if the fluorophore is stimulated for too long.

3.3 Related work

This section will describe previous work done on spike inference in calcium imaging data. Similar papers will be presented together and I will highlight where they differ from each other. I will also try to assess the quality of the approaches, but this is hard because more often than not the ground truth is not known or the method is solely tested on synthetic data.

3.3.1 Spike inference

Thresholding Thresholding approaches, as seen in Ziv et al. (2013), are fairly simple. A threshold is chosen, usually a some multiple of the standard deviation of the signal. Any activity that exceeds this threshold is counted as an event. After an event has been detected one has to find the event onset, which is usually done by finding the lowest point with a reasonably fast rise following it. The paper does not make any attempt test how well the spike inference works.

Template matching methods Various template matching methods have been described, Lütcke et al. (2013), Grewe et al. (2010), Kerr et al. (2005), Peron et al. (2015) Sato et al. (2007). These methods do some variation on matching the calcium trace with the following template.

$$\text{template(t)} = \text{offset} \qquad (t \leq 0) \tag{3.3a}$$

$$\text{template(t)} = \text{offset} + A_F \cdot e^{\frac{-t}{\tau}} \qquad (t > 0) \tag{3.3b}$$

where τ is the decay constant, A_F is the amplitude of the spike, and offset is the baseline of the signal. The first two have to be decided when running the algorithm, the latter is fit by minimizing the squared error.

In Grewe et al. (2010) and Lütcke et al. (2013) a so called "peeling-algorithm" is used. It is called as such because after a spike is detected, the prototypical calcium shape is subtracted from the signal. By iteratively removing all the spikes this way only the noise will be left, at which point all spikes have been detected. Since a single calcium event can last for some time the observed signal can often become a conflation of several different spikes. This mix

of several calcium events does not necessarily look anything like a single event. In this regime template matching will often fail. The peeling algorithm solves this problem by removing the influence of previous spikes.

Peron et al. (2015) uses a variation of the peeling algorithm but uses a bank of several templates with varying decay and rise times. This is one way to deal with the multitude of different shapes a calcium event can have, but it is not very elegant.

Lütcke et al. (2013) puts the peeling-algorithm to the test under a variety of different conditions, but on simulated data. While they went to great lengths to make the data as realistic as possible, it should still be taken with a grain of salt. They report a True Positive Rate (TPR) of 95.5% and a False Detection Rate (FDR) of 1.5%. This is comparable to what Grewe et al. (2010) reported with doing simultanous patch-clamp for ground truth.

Kerr et al. (2005) did patch-clamp on a subset of the cell that were optically recorded. They were able to detect 97% of individial spikes and 100% of spike bursts. FDR is not mentioned.

Peron et al. (2015) also did patch-clamp on a subset of the cells. They report 54% TPR of individual action potentials, which is drastically lower than what the previous approaches reported. This difference may be due to specifics of where in the brain the recording took place. The FDR of this approach is given as 0.01 Hz, which makes it hard to compare the other approaches.

Deconvolutional approaches The most notable deconvolutional approaches are introduced in Yaksi and Friedrich (2006) and Vogelstein et al. (2010).

Yaksi and Friedrich (2006) noticed that the observed fluorescent trace looks like a spike train convolved with the stereotypical calcium transient shape (as seen in the middle of figure 3.2). Inspired by this they attempt to reconstruct the firing rate of the cell by convolving the fluorescent trace with a filter that is the inverse of the stereotypical calcium shape. This approach does not resolve individual spikes, but instead yields a reconstruction of the firing rate. The authors compare the reconstructed firing rate with the original firing rate by their correlation. They report a reconstruction correlation of approximately 0.9 for three different datasets. It is hard to compare how this correlation compares to the TPR and FDR of other approaches.

Vogelstein et al. (2010) tries to infer spike times by deconvolution, but using a particle filter. The method is based on the mathematical model introduced in 3.2.2 and leverages

the fact that spikes can not be negative in order to outperform optimal filters that allow for negative spikes. The math is rather involved and will not be discussed here. The method was tested on simulated data and achieved a FPR of approximately 1 as FDR approached 0.3. For a FPR 0.97 they report a FDR of 0.1. From now on this method will be referred to as Fast-OOPSI, which stands for Fast Optimal OPtical spike inference.

Machine learning approaches Machine learning techniques have been used by and Sasaki et al. (2008).

In the former the most likely spike train is found by sampling. Spike trains are sampled and translated into a fluorescent trace using the forward model introduced in 3.2.2. The spike train that best fits the observed signal with this model is assumed to be correct.

The latter projects small windows over the calcium signal onto a two-dimensional space using principle component analysis (PCA). A support vector machine (SVM) is then conditioned to separate windows with and without spikes. An SVM works by finding the hyperplane that seperates the classes in the training data by the biggest margin. This approach is made possible because the authors did patch-clamp simultaneous with imaging which gives the ground truth of whether or not a spike happened. The authors report that for a signal to noise ratio greater than 10, the FPR and FDR is less than 10%.

3.3.2 Learning sparse representations through unsupervised Learning

The topic of this thesis is uncovering the underlying cause of the signal we observe. In our case we know that the underlying explanatory factor behind the data are sparse neuronal spikes. We want our algorithm to be able to uncover this underlying structure in the data in an unsupervised manner.

With this goal in mind we will investigate different approaches to unsupervised learning. We will also look at how we can help the algorithm to find the simplest explanation of the data we observe. In our case the simplest explanation would be where spikes happened, which is a binary vector. This is what would be called a sparse code, so we will in particular examine methods for that encourage sparse coding.

The autoencoder is an ANN architecture often used to learn from unlabeled data. It was first proposed in 1986 by David E. et al.. In the first successful examples of learning deep

architectures the network weights were initialized by doing unsupervised learning with autoencoders Bengio and Lamblin (2007). To do classification you have to learn the features of the data and how the features relate to the classes of interest. Unsupervised pretraining was an attempt to learn good features for doing classification later. Since then unsupervised pretraining for deep networks has since fallen out of favor and been replaced with with better initalization of network parameters, more effective nonlinearities and more data.

Learning features from data unsupervised, however, still remains an active research topic. Seeing that most data is unlabeled, effective unsupervised training would be a huge boon if one wants to scale up ANNs further.

The typical autoencoder consists of an encoder part and an decoder part. The encoder creates a latent representation z of input x. The decoder, on the other hand, tries to recreate the input x given the code z.

This problem of encoding and decoding has a trivial solution, the identity mapping. Therefore additional constraints are necessary for the network to learn useful features of the data. I will now present different ways to regularize autoencoders as proposed in literature.

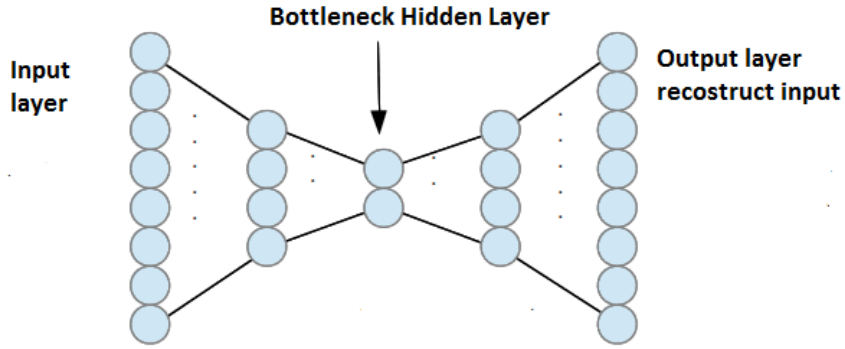

Figure 3.6: The autoencoder architecture. It is composed of two parts, the encoder and decoder. The encoder reduces the dimensionality of the data. The decoder recreates the original data from the hidden representation.

The autoencoder in Hinton (2006) was solely regularized by introducing a bottleneck to the architecture as seen in figure 3.6. That is, the middle layer of the network has a lower dimenionality than the input. Because there is a limited amount of information one can get past this bottleneck, only the most salient features of the input can be coded for.

The encoder in the bottleneck-scheme projects the input data on to a high-probability manifold in a lower dimensional space.

This is similar to principle component analysis, where each data point is projected onto

the principle components. In fact, an autoencoder with linear units is equivalent to principle component analysis Baldi and Hornik (1995).

The advantage of autoencoders is that the projections it performs do not have to be linear which allows a better representation of the data.

Two similar alternatives to just introducing a simple bottleneck is the denoising autoencoder introduced by Vincent et al. (2010) or the contractive autoencoder Rifai and Muller (2011).

While most regularizing of autoencoders is motivated by constraining the volume of hidden configurations accessible by the learner, these two try to make the hidden representation insensitive to changes in the input.

In the case of the denoising autoencoder the input x is turned into $\tilde{x}$ by corrupting it with some noise. This corruption can take several forms, but is usually Gaussian noise or multiplying the input with a Bernoulli random variable.

The objective is then to try to reconstruct the original input x given the noisy version$\tilde{x}$. This forces the autoencoder to group similar inputs to the same hidden representation, which is theorized to be a high probability manifold. The manifold hypothesis states that natural data in high dimensional space is concentrated around lower dimensional manifold. When using denoising as a training criterion the network learns to collapse input onto this high probability manifold, effectively modelling the distribution of the data.

The contractive autoencoder on the other hand imposes a penalty equal to the Frobenius norm on the encoder's Jacobian. This decreases the sensitivity to small variations in the input. The learned representation is shown empirically to capture the local directions of variations, which corresponds to the lower-dimensional manifold previously discussed.

These two methods have been used to win the Unsupervised and Transfer Learning Challenge, which makes them promising for our use-case. They do, however, tend to produce saturated rather than sparse representations which is exactly the opposite of what we want.

Masci et al. (2011) combined the ideas of using denoising as a training criterion and a CNN to extract features from the data. The network is set up like an autoencoder, where the encoder part makes an overcomplete representation of the input. The decoder in turn takes the overcomplete representation and linearly combines them to recreate the input. Interestingly they found that max-pooling, which retains the the maximal values of set of neighboring activations, was key to make robust features.

Ng (2011) introduces a 'life time sparsity' constraint which requires the average activation of a unit on the whole test set to be small. It is called a 'life time' constraint because it is calculated across the entire training set.

This constraint is formalized through the KL divergence between the target sparsity probability $\hat{\rho}$ and the hidden unit marginals ρ.

Let $a_j(x^{(i)})$ be the activation of unit j given input example $x^{(i)}$ then

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}[a_j(x^{(i)})] \tag{3.4}$$

is the average activation of a_j over all m training examples. They want to enforce

$$\hat{\rho}_j = \rho, \tag{3.5}$$

where ρ is some sparsity parameter that is chosen, typically a small value close to zero. The penalty term added to the cost function becomes

$$\beta\sum_{j=1} KL(\hat{\rho}\|\rho) \tag{3.6}$$

where β is the weighting of the penalty term. With this sparsity scheme new parameter is introduced, ρ, which has to be chosen. It is also not obvious how to make this work when using the rectified linear transfer function instead of a sigmoid.

In Makhzani and Frey they learn hiarchical sparse representations of data unsupervised. They regularize the autoencoder with of Winner-take-all pooling (WTA-pooling) and a lifetime sparsity which keeps the k% biggest activations in a minibatch. WTA is similar to max-pooling, but the non-maximal values are set to 0, instead of removed. This conserves spatial information which, as a side note, also is important for the task we want to solve. Instead of having WTA in all layers, only the last layer of the encoder is regularized. The decoder recreates the input by linearly combining the final features of the encoder.

Other strategies for regularizing the activity in the hidden layer is imposing an l1, l2, l1+l2, or student-t penalty on the hidden activation as proposed in Bengio et al. (2012). To remind the reader what the different regularization penalties are,

$$l1(x) = \sum |x| \tag{3.7}$$

$$l2(x) = \sum x^2 \tag{3.8}$$

$$\textit{student-t}(x) = \sum \frac{1}{\log(x+1)} \tag{3.9}$$

Chapter 4

Method

This chapter will describe how all the data, real and simulated, is treated. The artificial neural network architecture used for the task will be presented. Additionally, I will describe how the different spike inference methods effectiveness will be compared.

4.1 Data

4.1.1 Simulating Data

For the simulated data we create spike trains by having a simulated mouse do a random walk in a simulated box. The simulated mouse has simulated neurons that respond to simulated head direction. Each neuron has a tuning curve that is normally distributed around its preferred head direction, σ^2. The neurons follow a stochastic Poisson process where the spike rate for each neuron, λ_t, for each time t is given by the tuning curve and current head direction.

The spike trains are then translated a calcium trace according to the calcium model in section 3.2.2.

4.1.2 Real Data

The real data is simultaneous imaging and patch clamp of single cells from Chen et al. (2013b). The patch clamp of single cells give yields the exact membrane potential which makes it possible to find exact spike times.

Contrary to simulated data, real data is very finite. Usually one would need to split the

data into three different groups in order to avoid overfitting on training data. The data is split into one training, one validation and one test set. The model is trained on the training set and hyperparameters are tuned based on the performance on the validation set. When one has a model and hyperparameters that perform well on the training and validation set, the performance of the model is measured on the test set.

Since all training is unsupervised we can not overfit the data, so this split is not directly necessary. We will, however, show that the model generalizes to data it has never been shown before.

4.2 Experimental Setup

4.2.1 Spike Inference Methods

The model free ANN was compared with a model based approach.

We want to explore if a model free approach can outperform an algorithm that makes a lot of assumptions about the underlying dynamics of the signal. To represent approaches that make a lot of assumptions we will use the nonnegative deconvolution, from here on referred to as Fast-OOPSI, as described in Vogelstein et al. (2010).

4.3 Artificial Neural Network Architecture Design

The overall architecture of the ANN was the same for both simulated and real data. The architecture was guided by the constraints of the problem at hand and inspired by solutions to similar problems in literature.

Because we want to do unsupervised learning the network is set up as an autoencoder. The idea here is that the activation of the encoder will end up being something equivalent to the original spike train. Then, the decoder can recreate the original signal by convolving the hidden activation with an appropriate learned kernel (which hopefully will look like the stereotypical calcium transient from figure 3.2).

Each layer of the network is a convolutional layer and there are several motivations for this. First of all, convolutional layers retain spatial information and where exactly spikes happen is important to this task. In contrast, a unit in a fully connected layer could code for a feature anywhere in the input making it hard to find out where exactly the event was.

Secondly, as we have seen in Yaksi and Friedrich (2006), the observed fluorescence looks like a spike train convolved with the stereotypical calcium event shape. It therefore makes sense that the decoder is a single convolutional layer.

Thirdly, spikes within one recording might be fairly similar so it would be redundant to learn a spike detector for every position in the input.

The observant reader will be wondering why the encoder part of the network should choose to encode a representation that resembles the original spike train. There is really no reason that it should, so we will add a few constraints such that this representation becomes more likely.

Based on the prior that a spike train should be sparse, we can encourage a spike train representation in the hidden activation by regularizing activity. Possible regularizing schemes that were explored to promote sparse coding included l1, l2 or l1 + l2, and the student-t.

Other ways to regularize the activity that were explored, that forces sparse coding, include WTA-pooling and lifetime k-WTA.

Note that it is not an assumption we make that the spike train is sparse, it is a fact.

With this we have introduced 3 hyperparameters that have to be chosen, λ which is the scaling of the activation penalty, the WTA pool size, and the k value for the global k-WTA.

Each valid convolution will make the input (kernel size - 1) smaller. The input will be padded accordingly on both sides such that the output becomes the same size as the original input. This is to preserve the spatial information, which is important to get the spike times correct.

The activity in the hidden layer are thresholded in order in order to produce our inferred spike train. This threshold is chosen such that TPR and FDR is minimized. While this uses information that would not be accessible in a real setting, we do it in order to compare the methods in their best case. The threshold for Fast-OOPSI is chosen the same way.

Additionally, the activity in the hidden layer will be laterally, but consistently, shifted. This is due to how network is padded for each layer, how convolutions work and where in the output kernel the network decides to put the fast rise time.

This is not a huge problem, since the detected spikes are in the right position relative to each other. This means that in data with ground truth one can shift the position of the inferred spikes such that the first spike coincides with the first spike in the ground truth. In real data one could do something similar, but instead chose the shifting such that the first

spike coincides with the activity complex that probably caused it.

4.3.1 Neural Network Architectures for Real and Simulated Data

The input to the network was a 1×14400 vector. In words this means that the input has one feature plane and is 14400 long. The simulated data was made the same length as the real data for convenience sake.

The output of the network was a 1×14400 vector, which is the reconstructed signal.

Each valid convolution generates output with size input shape - kernel shape + 1. To retain spatial information padding is applied.

The first layer of the encoder is padded with zeros on each side. The total number of zeros being equal to the sum of the size of the all the kernels minus the number of layers. This ensures that the output of the encoder is the same size as the input. The rectified linear function was used as a transfer function for all layers in the encoder.

The decoder is a single layer and input is padded with the size of the kernel - 1. No transfer function was applied to the decoder activation.

Before activation is propagated to the last layer of the encoder WTA-pooling is applied. The output of the encoder is subjected to global k-WTA.

All weights in the encoder were initialized from the Glorot uniform distribution Glorot and Bengio (2010), while the decoder was initialized from a zero mean, 0.0001 sigma normal distribution.

The loss function was mean squared error and updates were performed using RMSprop. Both networks were trained end to end rather than layer by layer.

On simulated data the network consisted of 3 layers in total, 2 for the encoder and 1 for the decoder. On real data an additional layer was added for the encoder.

During training the network compares the original signal with its reconstructed signal. When doing spike inference, how ever, we disregard the decoder and look at the output of the encoder.

4.3.2 Implementation

The network was implemented with Lasagne on top of Theano Bastien et al. (2012) Bergstra et al. (2010).

Theano is a python library that allows you to define mathematical expressions as graph. While the process of expressing computation as a graph can be cumbersome, particularly in the beginning, the benefits you receive include optimizations, automatic differentiation of functions, and allows utilizing a GPU at the flick of a switch.

Lasagne is a library developed on top of Theano that allows for rapid prototyping of ANNs. It has predefined layers that can be connected like Lego, but still allows you use Theano for use cases not covered by Lasagne.

4.4 Evaluating Spike Inference Performance

It is not immediately obvious what the best scheme for measuring spike inference performance should be. There are several issues to address

The first problem relates to getting the exact spike timing correct. One algorithm could reconstruct the exact right spike train correctly but get each spike exactly 1/30th of a second too late and end up getting zero correctly identified spikes. This is obviously something we would want our scoring function to address.

The second problem is what to do if an approach gives out the wrong number of spikes. It could get all the target spikes correct by a shotgun approach, but we would not consider this a satisfactory solution.

To deal with these issues we will compare spike inference performance with two different scoring schemes, each described in turn below.

4.4.1 Comparing spike trains directly

We have seen that comparing spike trains directly is insufficient because inferring the exact spike time down to the millisecond is unrealistic. We will adopt an approach that matches actual and inferred spikes greedily, allowing a 0.5 s difference in spike times, as described in Lütcke et al. (2013).

This allows us to define True Positives, True Negatives, False Positives and False Negatives (TP, TN, FP, FN) such that not everything is registered as false positives or false negatives.

Let X_i be the i'th actual spike time, Y_j the j'th inferred spike time, both sorted in ascending order. Then the algorithm can be summed up the following way.

for all X_i **do**

for all Y_i **do**
if $X_i - 0.5 < Y_i < X_i + 0.5s$ **then**
Register a True positive
Remove Y_i from Y
Remove X_i from X
Break
end if
end for
end for
Register remaining elements in Y as False Positives
Set True Negatives to $(Y.length - FalsePositives - TrueNegatives$
Register elements left in X as False Negatives

With TP, TN, FP, and FN defined this way and letting N and P be the total number of negative and positive samples respectively, we can look at the following performance measures.

$$\text{True Positive Rate} = \frac{\text{TP}}{P} \tag{4.1a}$$

$$\text{False Discovery Rate} = \frac{\text{FP}}{FP + TP} \tag{4.1b}$$

$$\text{Precision} = \frac{\text{TP}}{TP + FP} \tag{4.1c}$$

$$\tag{4.1d}$$

4.4.2 Firing Rate as a Measurement of Spike Inference Efficiency

Another way to compare the actual and inferred spike train is by comparing firing rates.

The reconstructed spike train and the actual spike train will be translated to their respective firing rates. The MSE of the resulting firing rates will be a measure of how well the reconstruction captures the dynamics of the original spikes.

The firing rate is number of spikes divided by some time unit, for example spikes per second. It is not obvious what the length in time each bin should be though. Additionally, the exact bin size can have a big effect on the resulting firing rate.

In order to achieve a smooth firing rate that is impartial (that is, keeping me from choos-

ing the bin size that makes my approach look better), we will treat this as a kernel density estimation problem with an Epanechnikov kernel.

The bin size will in this case be the breadth of the kernel (or bandwidth). The kernel breadth will be decided by doing a 20 fold cross-validation on the original spike train data. This way the size of the kernel is chosen such that the optimal representation of the firing rate is achieved.

Since the bandwidth is decided using the original data, it it not biased towards any of the spike train reconstruction approaches, yielding a fair comparison.

Chapter 5

Results and Discussion

In this chapter I will present the results of the performance measures discussed in the methods section.

5.1 Results

In all plots and tables in this chapter parameters for the different algorithms have been optimized by looking at the actual spike train. For Fast-OOPSI this is the threshold level for what counts as a spike. For the network this was mainly the k value for the global k-WTA and the threshold for what counts as a spike.

5.1.1 Distribution of Mean Squared Reconstruction Error after Training

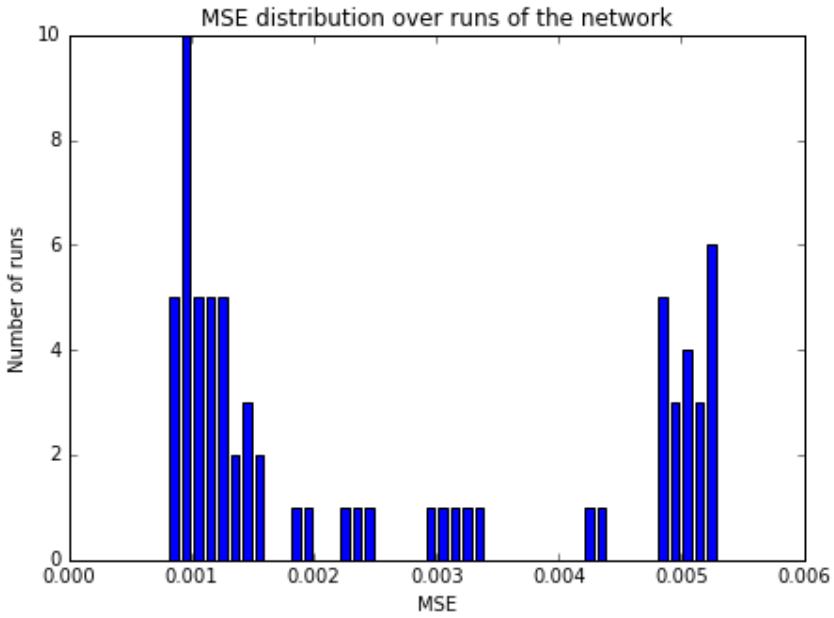

Figure 5.1: The distribution of the final MSE for 70 runs of the algorithm for varying k values.

In figure 5.1 the result of training 70 different ANNs is plotted. The networks were trained on 1 of the calcium traces from the dataset. The first 10 networks were trained using k equal to 20, the next group of 10 used 25 and so on ending in 55.

In total there were 14 actual spikes in the sample the algorithm was trained on. The k value seems to have little effect on the resulting reconstruction MSE. The five best results, in terms of MSE, were found in both the earlier and the later groups. While the k value is not particularly important at train time, it is at test time.

In figure 5.2 the spikes underlying the signal is shown in the top plot. The 5 subsequent rows show what the hidden activation of the 5 best runs in terms of MSE. All the networks use k equal to 20 in these plots.

Figure 5.2: The top plot shows the actual target spikes underlying the observed signal. The five plots under show what the ANN thought the spike train looked like given the mean squared error it achieved. These five ANNs are the ones who got the lowest training error, corresponding to the left-most bar, in figure 5.1

5.1.2 Firing Rate Reconstruction in Simulated Data

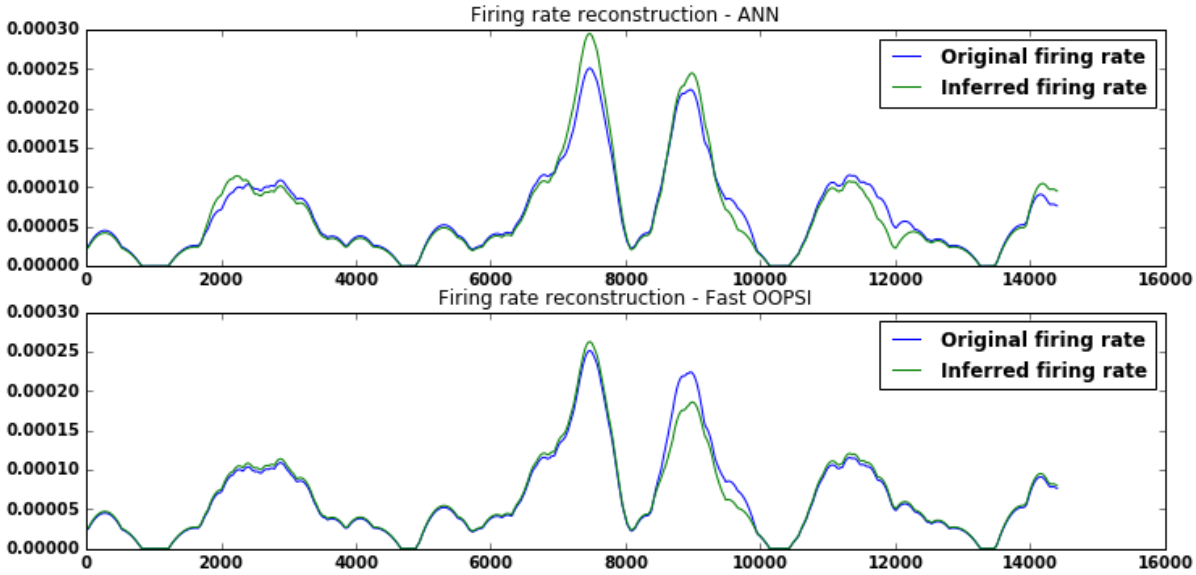

Figure 5.3: Top: The original firing rate compared to the ANN firing rate reconstruction. The reconstruction achieved an MSE of $1.97 * 10^{-6}$ Bottom: The original firing rate compared to the Fast-OOPSI firing rate reconstruction. The reconstruction achieved an MSE of $1.13 * 10^{-6}$

In figure 5.3 can see that the methods have fairly perform similarly in terms of reconstruction the firing rate. The discrepancy is primarily in areas of high activity. In these hot areas the ANN tends to produce more false positives, while Fast-OOPSI on the other hand tends to produce more false negatives.

5.1.3 Confusion Matrix for Simulated Data

The confusion matrix is a visual representation of what kind of mistakes the method does. The statistics were found using the method described in 4.4.1.

For the following plots and figures, while each algorithm was given 0.5 seconds leeway in getting the spike time correct, only the ANN really benefitted from this. Fast-OOPSI performed as good with 0 seconds leeway.

The following tables were made according to the method described in section 4.4.1.

Table 5.1: Confusion matrix for ANN - simulated data

	Actual Spike	No Spike
Inferred Spike	67	7
Inferred No Spike	2	14924

Table 5.2: Confusion matrix for Fast-OOPSI - simulated data

	Actual Spike	No Spike
Inferred Spike	68	0
Inferred No Spike	1	14931

Table 5.3: Some relevant statistics

	True Positive Rate	False Discovery Rate	Precision
Fast-OOPSI	0.98	0	1
ANN	0.97	0.1	0.91

5.1.4 Firing Rate Reconstruction in Real Data

In the following sections concerning real data neither of the algorithms have seen the data they are tested on before. Fast-OOPSI measures its parameters using the calcium trace it is presented to. The ANN has its k parameter tuned so that the reconstruction looks reasonable.

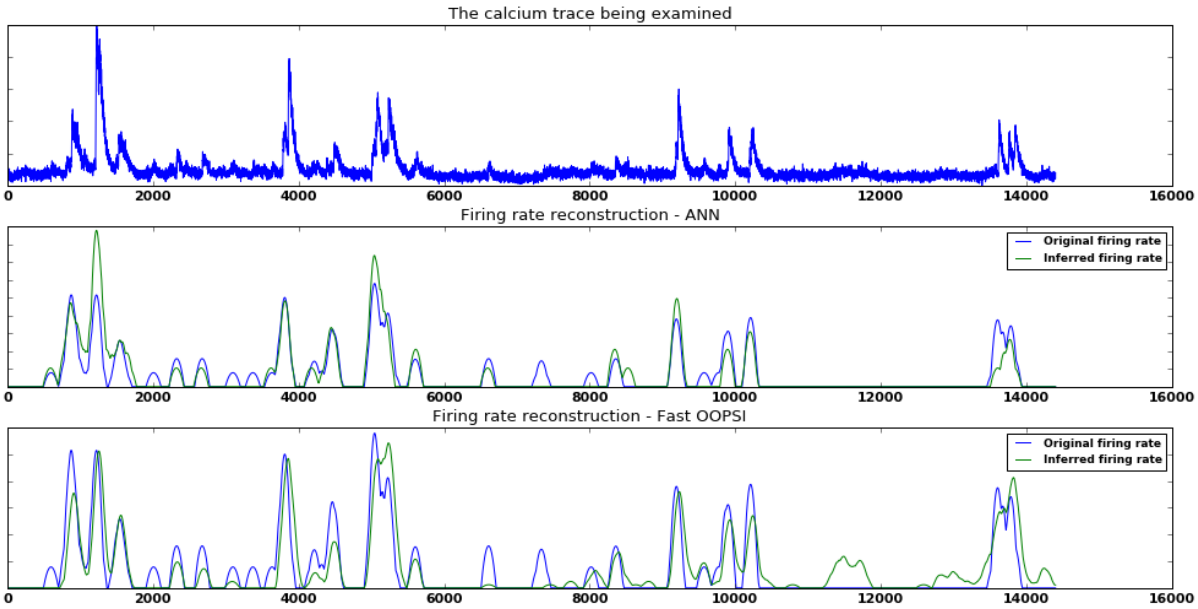

Figure 5.4: Top: The original firing rate compared to the ANN firing rate reconstruction. Bottom: The original firing rate compared to the Fast-OOPSI firing rate reconstruction

In figure 5.4 can see that both methods do capture the trend of the firing rate pretty accurately. Fast-OOPSI, for some reason, struggles with a part with low activity towards the end right before the 12000 mark. While the noise could be removed by increasing the threshold the overall quality of the firing rate reconstruction suffered for it. The ANN on the other

hand struggles with the spike complex in the beginning and towards the end. The ANN and Fast-OOPSI end up with an MSE of $3.95 * 10^{-9}$ and $4.56 * 10^{-9}$ respectively.

5.1.5 Confusion Matrix for Real Data

The confusion matrix is a visual representation of what kind of mistakes the method does. The statistics were found using the method described in 4.4.1.

All the following graphs and tables were made using one test example from the dataset containing 92 spikes in total. Neither of the algorithms had been exposed to the data before making these graphs.

For the real data, both methods benefitted from being given half a second of leeway. Fast-OOPSI does get the timing more accurately than the ANN and as a consequence needs less leeway before the number of true positives is maxed out.

In table 5.5 the threshold et such that Fast-OOPSI got more true positives than the ANN, while minimizing the number of false positives.

The following results were aquired with the method described in section 4.4.1.

Table 5.4: Confusion matrix for ANN - real data

	Actual Spike	No Spike
Inferred Spike	60	19
Inferred No Spike	32	14205

Table 5.5: Confusion matrix for Fast-OOPSI - real data

	Actual Spike	No Spike
Inferred Spike	66	103
Inferred No Spike	26	14205

Table 5.6: Some relevant statistics

	True Positive Rate	False Discovery Rate	Precision
Fast-OOPSI	0.72	0.61	0.39
ANN	0.65	0.24	0.76

5.1.6 Examining what the Network Network has Learned

The following figure is an example of what the network has learned about the simulated data.

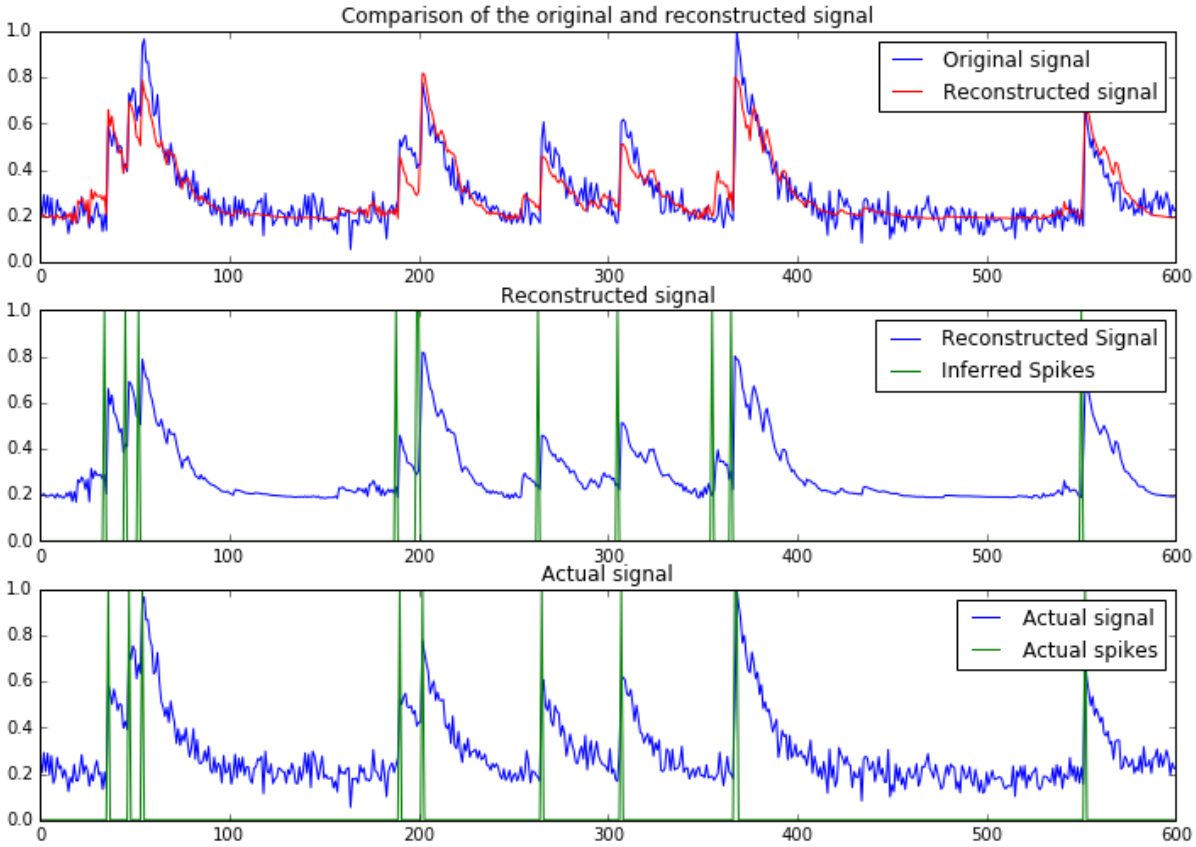

Figure 5.5: Top: Comparison of the input given to the ANN and the output. Middle: The reconstructed activity with detected spikes overlaid. Bottom: The original signal with the actual spikes overlaid.

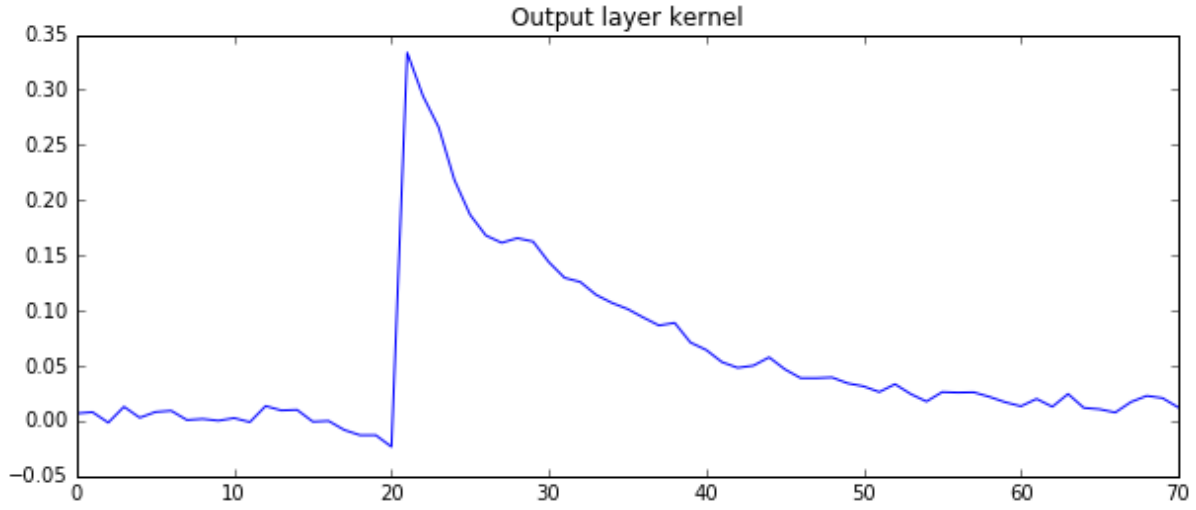

Figure 5.6: The kernel the decoder learned. This is essentially what the network thinks a single calcium event looks like.

The following figure is an example of what the network has learned about the real data.

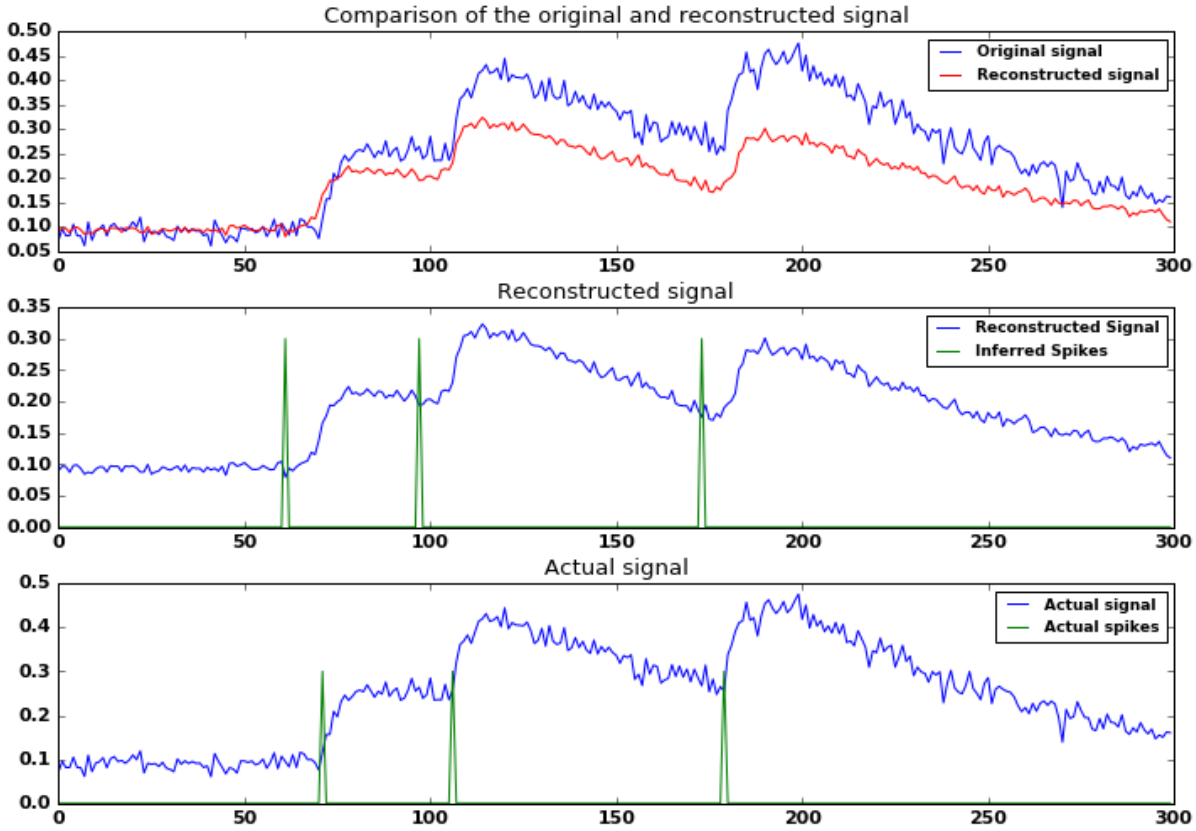

Figure 5.7: Top: Comparison of the input given to the ANN and the output. Middle: The reconstructed activity with detected spikes overlaid. Bottom: The original signal with the actual spikes overlaid.

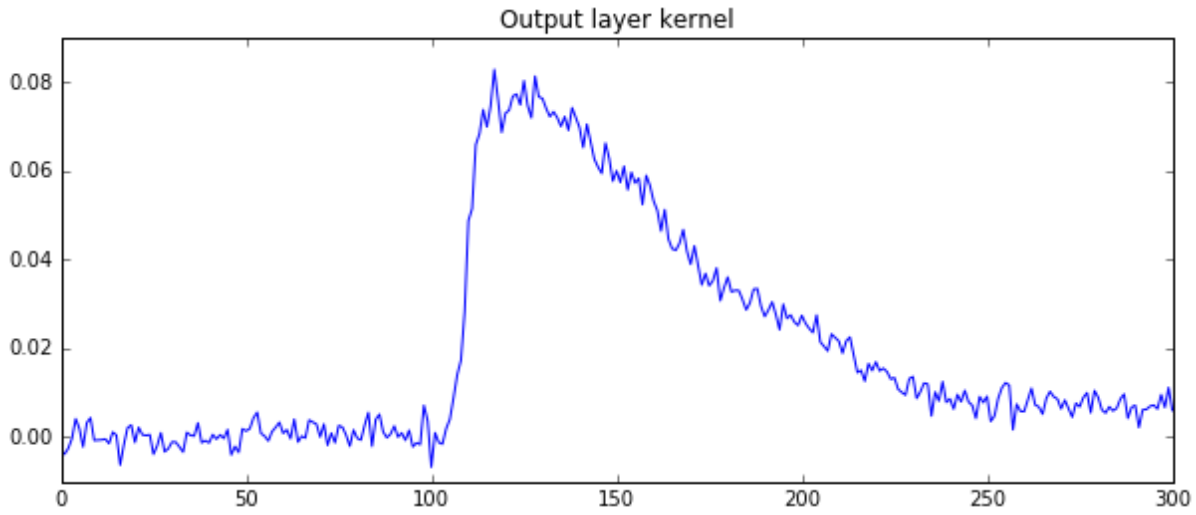

Figure 5.8: The kernel the decoder learned. This is essentially what the network thinks a single calcium event looks like.

5.1.7 Side by Side Comparison of the ANN and Fast-OOPSI Output

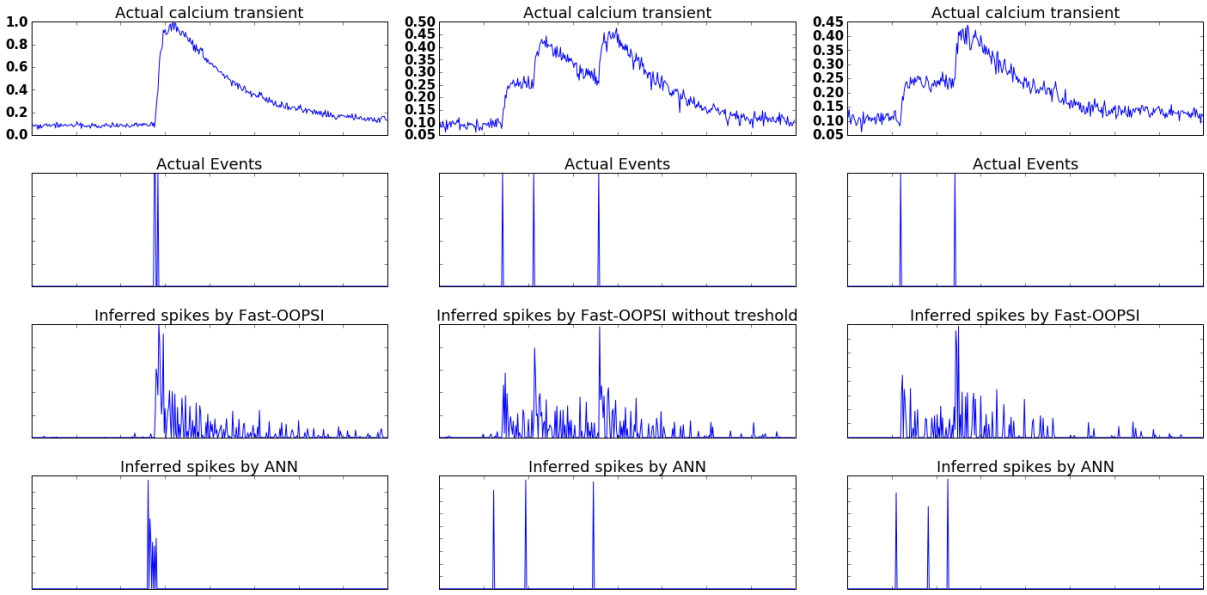

Figure 5.9: In this plot we see what spikes Fast-OOPSI and the ANN sees given the calcium transients in the top row. The calcium transients are excerpts from a fluorescent trace containing 14 spikes. The Fast-OOPSI output has not been tresholded, while the ANN output has had global k-WTA applied. The k parameter was 20 for this particular output.

5.2 Discussion

5.2.1 Performing Spike Inference on Simulated Data

For simulated data the architecture of alternating convolutional layers and doing WTA-pooling with a penalty on the activity in the encoder's hidden layers managed to solve the task of spike inference unsupervised.

It would fairly consistently reach a desired solution but would sometimes suffer from the dead filter problem. That is, the algorithm would output a vector where all the elements were the same. This is particularly common if the hidden activity regularized too harshly and having little hidden activity becomes more important than having a good reconstruction.

Most of the difficulty training these networks was balancing the regularization so that it did not suffer the dead filter problem but also did not learn a useless identity function. But once that balance was found, the algorithm would converge to a satisfactory solution most of the time.

The way the ANN learned to solve the task was by detecting places where the signal has a sharp rise time and counting them as spikes. It did not consider the overall shape of the

signal.

This can be explained by the fact that each spike will have a sharp visible onset, but not necessarily have the classic exponential decay. That is, it will have exponential decay, but an other spike might happen right after which changes the shape of the decay drastically. Examples of this can be seen in the top plot in figure 5.5. Because of this the statistically most important feature of the data for the network is the sharp rise.

The problem with this learned representation is that noise can make sharp rises randomly without any spikes having happened. Artifacts of this can be seen in how the network reconstructs the input. Random noise with a sharp rise will register as small spikes. Notice how in the middle plot in figure 5.5 the reconstruction contains what appears to be tiny calcium transients. The most notable example is the seventh inferred spike in this plot which is random noise that was registered as a spike.

This happens because a convolutional network is essentially just doing template matching with learned kernels. Because the template it learns matches any spike proportionally to the magnitude of the spike, small spikes of noise can sometimes be enough to trigger the template sufficiently.

While the network performed well on the real data in this thesis, the data was recorded under near perfect conditions. The signal-to-noise ratio is well above what we could expect from real experimental conditions.

Experiments were done with both l1, l2, l1 + l2 and student-t regularization. With the simulated data l1 regularization did yield a hidden representation close to the underlying spike train on occasion. It was not very robust and would mostly produce a wide bump rather than a sharp spike.

What was found in the end was that regularization by doing local WTA-pooling and global k-WTA-pooling was sufficient and efficient.

When comparing the confusion matrix for the two methods, 5.2 and 5.5 we see that for simulated data Fast-OOPSI outperforms the ANN. This is not too surprising though, considering that the simulated data is based on the underlying model of the Fast-OOPSI algorithm.

5.2.2 Performing Spike Inference on Real Data

One of the issues with working on real data is that it is a lot more realistic than simulated data which is extremely inconvenient. In particular, the baseline fluorescence changes with time

and the popular assumption that spikes cause the same fluorescence amplitude is shown to be less than reasonable. These things make it harder to find a good solution because fitting the baseline also becomes a good way to reduce the MSE.

One thing that seems extremely self-evident in hindsight is that the network should be fed the entire trace as a training sample. At first the trace was cut into smaller time windows and fed to the network. A problem with this is that in a lot of windows only the tail of a spike will be present. This poses a problem for the network because it wants to minimize the error but the spike that caused the transient it sees is outside the window. If the encoder has what we would consider the right representation, which is zero spikes for the current window, the network would get a poor reconstruction error.

One of the trickiest aspects of doing unsupervised learning is the difficulty of knowing whether the solution you have found is good or not. In the context of our autoencoder, a low reconstruction error might either mean that a good representation has been found or that it has too many degrees of freedom and has found a useless identity or delta function.

Using k-WTA solves this problem for us because we know that the output of the encoder is at maximum a vector with k values. The autoencoder can not make the identity function with this amount of regularization. The algorithm did, however, find a work around by making all the values the same size which we will discuss next.

The network did quite often find a local minima where the network would output a vector with a single value. This value is the constant c that minimizes the function

$$MSE = \frac{1}{m}\sum_{i}^{m}(y_i - c)^2 \tag{5.1}$$

which obviously is the mean of Y.

When you see the MSE that corresponds to this c value it either means that the network is regularized too hard or that it poorly intialized. We will call particular value MSE_{dead}.

One key difference between real and simulated data is that if the MSE of the run is lower than MSE_{dead} it has found the underlying representation we are looking for. On real data, however, any $MSE < MSE_{dead}$ can have found a myriad of different representations. It might represent each actual spike with several spikes or only code for the biggest event.

In the end the choice of penalty function on the hidden activation or having a penalty function at all was not important on the real data. Experiments were done with both l1, l2, l1 + l2 and student-t regularization. Out of several hundred training attempts with various λ

values, the network only succeeded once using l1 regularization.

The network did not benefit tangibly from using any of these regularization schemes and subsequently only WTA-pooling and global k-WTA was used in the end.

In the training sample used to make figure 5.1 MSE_{dead} is approximately 0.005 and we see a big cluster of runs around this value. It is important to note that spike in fluorescence in this example is twice as big as the second largest, and approximately three times bigger than the average event. For the bars where $MSE < MSE_{dead}$, most only code for the biggest event. Where lower error means that it captures the activity of the bigger event better. It is not until MSE is around 0.001 that all spikes are captured properly.

The most successful network, which was trained on an example from the dataset consisting of 14 spikes in total, generalized well on other examples with 6 times as many spikes, as long as the k parameter was changed appropriately. If the k parameter was too low, only the most prominent spikes would be reported. Too high and the network yield too many false positives.

This arbitrary setting of the k parameter remains a huge problem if the method is to employed in a scenario where ground truth is not known.

One way to mitigate this would be looking at the reconstructed signal for each trace as the k value is changed. That way you can find the lowest number that still responds to most of the spikes you see. I was not able to find a satisfactory solution to this problem.

Furthermore one could argue that when the kernel of the decoder, as shown in figure 5.6 and 5.8, looks like the stereotypical calcium transient then the output of the encoder has to be spikes.

Chapter 6

Conclusion

6.1 Summary

This thesis was motivated by the need for analysis techniques that can capture the rich dynamics of calcium imaging data better.

Based on surveyed related work, a artificial neural network was constructed to learn how to infer spike trains in an unsupervised fashion.

The network ended up being a convolutional autoencoder with WTA-pooling and global k-WTA on the hidden activation of the encoder.

The new model free network was shown to work on both simulated and real data and performing better than a model based approach on the real data.

Based on the results presented results the ANN seems like a promising avenue of further researchs.

6.2 Evaluation

We set out to examine whether an ANN could infer spike times unsupervised and how this compares to an established method.

It was shown that an ANN was able learn to infer spike times in an unsupervised manner.

The ANN did however sometimes diverge, unable to find a proper solution. Also, when it converged it did not always capture all the spikes in the spike train. But even so, as shown in 5.2 it was able to find something close to the real spike train on several occasions.

While the algorithm has flaws, it is still promising as a proof of concept. The most damn-

ing flaw being that it is hard to argue that the algorithm has found the underlying spike train on real data when ground truth is not available, given that we know it does not always converges to the right answer.

As for how the ANN compares to established methods, it looks like the ANN somewhat outperforms Fast-OOPSI on real data as can be seen in table 5.4, 5.5, and 5.6. While Fast-OOPSI does identify spikes more reliably, it also has a false positive rate that is a lot higher than the ANN.

We see in figure 5.4 that the ANN achieves a slightly better MSE when reconstructing the firing rate on real data than Fast-OOPSI.

While the resultsce are encouraging, there are some issues with parameter selection and training robustness that need to be addressed.

6.3 Future Work

The most important aspect of spike inference algorithms is that they are easy to use. As an example, I originally aimed to include the peeling-algorithm from Lütcke et al. (2013) to the analysis in this thesis. I was however unable to make it work because I could not find proper values for the around 30 variables that have to be set manually.

While artificial neural networks are complicated models that can not be entirely plug and play, the algorithm should have a robust way of deciding key parameters such as the k value automatically if researchers are to adopt this method.

An other future work is convincing a neuroscientist that what the algorithm finds is the real spike train when ground truth is not available. Other algorithms, such as Fast-OOPSI, even if their performance was sub-optimal on the *in-vitro* data presented in this thesis, have a solid justification in theory. The ANN is a black box and will be distrusted as such.

Bibliography

Baldi, P. F. and Hornik, K. (1995). Learning in linear neural networks: a survey. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 6(4):837–858.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bengio, Y., Courville, A., and Vincent, P. (2012). Representation Learning: A Review and New Perspectives. (1993):1–30.

Bengio, Y. and Lamblin, P. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, (1):153—-160.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

Chen, T.-W., Chen, T.-W., Wardill, T. J., Wardill, T. J., Sun, Y., Sun, Y., Pulver, S. R., Pulver, S. R., Renninger, S. L., Renninger, S. L., Baohan, A., Baohan, A., Schreiter, E. R., Schreiter, E. R., Kerr, R. A., Kerr, R. A., Orger, M. B., Orger, M. B., Jayaraman, V., Jayaraman, V., Looger, L. L., Looger, L. L., Svoboda, K., Svoboda, K., Kim, D. S., and Kim, D. S. (2013a). Ultrasensitive fluorescent proteins for imaging neuronal activity. *Nature*, 499(7458):295–300.

Chen, T.-W., Wardill, T. J., Sun, Y., Pulver, S. R., Renninger, S. L., Baohan, A., Schreiter, E. R., Kerr, R. a., Orger, M. B., Jayaraman, V., Looger, L. L., Svoboda, K., and Kim, D. S. (2013b). Ultrasensitive fluorescent proteins for imaging neuronal activity. *Nature*, 499(7458):295–300.

Dalto, M. (2014). Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting.

David E., R., Hinton, G. E., and Williams, R. J. Backprop{_}Old.Pdf.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 9:249–256.

Grewe, B. F., Langer, D., Kasper, H., Kampa, B. M., and Helmchen, F. (2010). High-speed in vivo calcium imaging reveals neuronal network activity with near-millisecond precision. *Nature Methods*, 7(5):399–405.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition. *Arxiv*, pages 1–12.

Hinton, G. (2006). To Recognize Shapes , First Learn to Generate Images To Recognize Shapes , First Learn to Generate Images.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Kerr, J. N. D., Greenberg, D., and Helmchen, F. (2005). Imaging input and output of neocortical networks in vivo. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):14063–8.

Krizhevsky, A., Sulskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information and Processing Systems (NIPS)*, pages 1–9.

Lütcke, H., Gerhard, F., Zenke, F., Gerstner, W., and Helmchen, F. (2013). Inference of neuronal network spike dynamics and topology from calcium imaging data. *Frontiers in neural circuits*, 7(December):201.

Makhzani, A. and Frey, B. Winner-Take-All Autoencoders. pages 1–11.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *Lecture Notes in Computer Science (including*

subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6791 LNCS(PART 1):52–59.

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, pages 1–19.

Peron, S., Freeman, J., Iyer, V., Guo, C., and Svoboda, K. (2015). A Cellular Resolution Map of Barrel Cortex Activity during Tactile Behavior. *Neuron*, 86(3):783–799.

Rifai, S. and Muller, X. (2011). Contractive Auto-Encoders : Explicit Invariance During Feature Extraction. *Icml*, 85(1):833–840.

Sasaki, T., Takahashi, N., Matsuki, N., and Ikegaya, Y. (2008). Calcium Fluctuations. *J Neurophysiol*, pages 1668–1676.

Sato, T. R., Gray, N. W., Mainen, Z. F., and Svoboda, K. (2007). The functional microarchitecture of the mouse barrel cortex. *PLoS biology*, 5(7):e189.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11(3):3371–3408.

Vogelstein, J. T., Packer, A. M., Machado, T. a., Sippy, T., Babadi, B., Yuste, R., and Paninski, L. (2010). Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of neurophysiology*, 104(6):3691–3704.

Vogelstein, J. T., Watson, B. O., Packer, A. M., Yuste, R., Jedynak, B., and Paninski, L. (2009). Spike inference from calcium imaging using sequential Monte Carlo methods. *Biophysical journal*, 97(2):636–655.

Yaksi, E. and Friedrich, R. W. (2006). Reconstruction of firing rate changes across neuronal populations by temporally deconvolved Ca2+ imaging. *Nature Methods*, 3(5):377–383.

Ziv, Y., Burns, L. D., Cocker, E. D., Hamel, E. O., Ghosh, K. K., Kitch, L. J., El Gamal, A., and Schnitzer, M. J. (2013). Long-term dynamics of CA1 hippocampal place codes. *Nature neuroscience*, 16(3):264–6.