Unsupervised methods for large-scale cell-resolution neural data analysis

Gergő Bohner

A dissertation submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Gatsby Computational Neuroscience Unit

Department of Life Sciences

University College London

November 19, 2019

I, Gergő Bohner, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In order to keep up with the volume of data, as well as the complexity of experiments and models in modern neuroscience, we need scalable and principled analytic programmes that take into account the scientific goals and the challenges of biological experiments.

This work focuses on algorithms that tackle problems throughout the whole data analysis process. I first investigate how to best transform two-photon calcium imaging microscopy recordings – sets of contiguous images – into an easier-to-analyse matrix containing time courses of individual neurons. For this I first estimate how the true fluorescence signal gets transformed by tissue artefacts and the microscope setup, by learning the parameters of a realistic physical model from recorded data. Next, I describe how individual neural cell bodies may be segmented from the images, based on a cost function tailored to neural characteristics. Finally, I describe an interpretable non-linear dynamical model of neural population activity, which provides immediate scientific insight into complex system behaviour, and may spawn a new way of investigating stochastic non-linear dynamical systems.

I hope the algorithms described here will not only be integrated into analytic pipelines of neural recordings, but also point out that algorithmic design should be informed by communication with the broader community, understanding and tackling the challenges inherent in experimental biological science.

Impact statement

The problems discussed in this thesis are expected to be of interest first and foremost for experimental and computational neuroscientists, who are concerned with recording and analysis of neural data. The proposed algorithmic solutions are tailored towards neural characteristics, and evaluated on neural data. I aimed to make my implementations generally applicable, and all code is publicly available for direct application to novel data, incorporation into data analysis pipelines, or for extending the ideas.

Although not tested, the models and algorithms may well be applicable outside the neural setting. Firstly, the inverse model of the two-photon microscope is directly applicable to any kind of recording; furthermore, the ideas of non-uniform spatial gain and modelling the discrete nature of optical signal for low photon counts should be of interest for any type of microscopic application.

Secondly, I discuss the presence and reconstruction of higher order joint cumulants in multivariate time series data, in which the measurements have a spatial structure. Although in my imaging application this spatial structure is a multidimensional grid of pixels, this idea can be generalised, as shown during the derivation. Financial time series analyses are often concerned with the presence of higher order cumulants, associated with risk. I believe the proposed reconstruction algorithm could be applied to a low-dimensional embedding of stocks, in which the spatial distance would reflect the similarities of the stocks, whereas the joint variations of time series could be summarised by the higher order cumulants.

Finally, I describe algorithms that learn interpretable models of non-linear stochastic dynamical systems. These could be of interest to a large number of academic and engineering fields, as they can be used to provide immediate insight into qualitative changes of the behaviour of arbitrary systems, regardless of transition or observation noise. Applicable systems range from engineering control through chemical reactions to weather modelling.

Ultimately, I believe this thesis demonstrates the application of unsupervised machine learning as an effective tool to bridge complex theoretical models and limited experimental measurements, and thus will drive the advancement of more detailed models of the world, as well as more informed experiments to constrain those models, leading to faster scientific progress.

Acknowledgements

S még mindig nem tudom elmondani neked,
mit is jelent az nékem, hogy ha dolgozom,
óvó tekinteted érzem kezem felett.

– Radnóti Miklós, *Tétova Óda*, 1943. május 26.

I thank my parents and family. My mother has been incredibly supportive of all my choices, and raised me to be caring, but independent, to engage in deep analysis of events, while remembering to enjoy the moment. My father is not with us anymore, but his fascination with mathematics, geometry, riddles and even puns definitely influenced who I am today. My extended family introduced me to supporting and caring about each other, no matter the distance, and I value the time we get to spend together, discussing past and present, science and religion.

I thank my partner, Amaia, for her love, support and understanding in difficult times, and for making sure we enjoy and remember the joyful moments of our lives. After this thesis I now have to enter a new phase of my life, but I am optimistic about the future, knowing I get to form it together with a smart, caring and amazing person whom I love!

I thank my friends. With my childhood friends – Kovács Tibor, Matlák Péter, Töreki Balázs, Kozma Richárd, and Suhajda Balázs – we have known each other for nearly two decades, and I am grateful for the knowledge that I will be able to rely on them, and call them friends for the decades to come. Throughout the years since, I am thankful for having met many excellent people and communities. I remember especially fondly my time at the Pázmány Péter Catholic University, and the group of people I studied, hiked, played board games, or sang with; even at half 4 in the morning! In the past 5 years I have lived at Goodenough College in London, and its international community has been open, welcoming and enlightening: our inspired conversations opened my eyes to both problems and wonders of the world, and I engaged in my fair share of singing, dancing, and other pastimes; most importantly, it truly felt like a home.

I thank my PhD supervisor, Maneesh Sahani, who introduced me to the world of machine learning. I was fascinated by his structured view of it – understanding algorithms as inference in a generative model – which he passed on excellently through teaching and personal discussions. Moreover, he always made himself available to share his thoughts and guide his students. I admire

the breadth and depth of both his technical and general knowledge, and I hope to have a similarly complete understanding of the world one day! I am thankful that he showed it is possible.

I thank the members of the Gatsby Unit, forming a tight group of gifted yet approachable people, many of whom formed my academic thinking strongly. I started together with Joana Soldado-Magraner, Heiko Strathmann, Wittawat Jitkrittum, and Carsen Stringer, and I remember spending all our time together – mostly at the office, deliberating over the latest neuroscience or machine learning challenge handed to us by our demanding, but amazing teachers, Peter Dayan, Peter Latham, Arthur Gretton and of course Maneesh. Others that I remember particularly fondly from my years at Gatsby are Gaurav Venkataraman, Ritwik Niyogi, Laurence Aitchison, Vincent Adam, Mijung Park, Srini Turaga and Lea Duncker; we've had many excellent conversations about both academic and personal thoughts.

Finally, I'd like to thank the broader academic community, always moving the frontier of human knowledge forward, enabling individuals to build upon the current knowledge. Thanks to funding from the Gatsby Foundation, DARPA, the University of Warwick and Cancer Research UK over the years, I have had the chance to do research at the London Research Institute, the Alan Turing Institute in London, as well as Stanford University in the United States. I am grateful to have met and worked with Árpád Csurgay, Thomas Surrey, Sebastian Maurer, Franck Fourniol, Nicholas Cade, Sebastian Vollmer, Franz Király, Bilal Mateen, Louis Aslett, Catalina Vallejos, Dan O'Shea, Eric Trautmann and Krishna Shenoy, amongst others.

Contents

List of Figures

List of Tables

Thesis overview

Chapter 1 - Introduction

In this chapter I provide a brief overview of the history of extracting cell-resolution data from animal brains in vivo, and how the advances in experimental methods lead to novel data analysis algorithms, both for extracting signal from the raw data, and analysing the joint activity of increasing numbers of neurons. I then discuss the increasing role of machine learning in analysing biological data, and how it can be utilised to create physically inspired generative models, and estimate model parameters from the recorded data.

Chapter 2 - Spatial background equalisation of two-photon calcium imaging videos of neural tissue

In this chapter I describe an interpretable generative model of two-photon microscopy recordings of fluorescent calcium indicators expressed in neural tissue. I then propose and apply a scalable approximate algorithm, which fits the model parameters, and inverts the model in order to standardise both the data within a single field of view, as well as recordings of different brain areas from multiple animals, produced via various microscopes and settings.

Chapter 3 - Segmentation of neural cell bodies via Convolutional Higher Order Matching Pursuit (CHOMP)

In this chapter I show an extended generative model of calcium imaging, which describes the data as a superposition of signals from multiple sources with small spatial support. These stereotypically shaped sources are thought to be neural cell bodies, and I propose and apply an approximate algorithm, which identifies single cell bodies and activities via utilising neurally-inspired features and constraints.

Chapter 4 - Learning interpretable models of latent stochastic dynamical systems

In this chapter I propose two algorithms that can learn an interpretable non-linear stochastic dynamical picture from a large number of jointly observed time series - such as recordings of neural population activity. Non-linear models may be difficult to interpret directly; they are often analysed in terms of the number and stability of their fixed points. The two proposed algorithms – one for

data evenly sampled in time, such as image frames; the other for unevenly recorded events, such as neural spiking – incorporate this type of analysis, and learn dynamical models that are conditioned on fixed points and local linearisations around them. This conditioning enables the fitting process to estimate its own uncertainty of the number and locations of the fixed points, and importantly it provides direct scientific insight into the qualitative behaviour of a recorded system.

Chapter 5 - Conclusions

In this chapter I summarise the models and algorithms desribed in this thesis, discuss their role in the analysis of cell-resolution neural data, and elaborate on potential promising avenues for future work.

1

Introduction

Not always, dear Licinius, is it wise
On the main sea to ply the daring oar;
Nor is it safe, in dread of angry skies,
To keep too near to the perfidious shore.
To no excess discerning spirits lean;
They feel the blessings of the golden mean.

– Horace, *To Licinius Murena*

In this chapter I provide a brief history of extracting cell-resolution data from animal brains in vivo, and how the advances in experimental methods lead to novel data analysis algorithms, both for extracting signal from the raw data, and analysing the joint activity of increasing numbers of neurons. I then discuss the increasing role of machine learning in analysing biological data, and how it can be utilised to create physically inspired generative models, and estimate model parameters from the recorded data.

This chapter solely presents a generic overview of my understanding, aiming to pinpoint open questions and motivate the need for the models and algorithms described in this thesis. The further chapters each contain a more technical introduction of the particular problems and related fields, providing references to previous work.

1.1 Measuring and analysing neural data

Ever since Cajal, Waldeyer and others introduced the neuron doctrine – establishing neurons as discrete, functional units of the nervous system – neuroscientists have been attempting to glean insight into the information processing of multicellular organisms by measuring the electrical activity of single neurons. Starting by recording from single, easily accessible cells – such as the squid giant axon investigated by Hodgkin and Huxley in their pioneering work – experimental methodologies advanced a great deal, thanks to the contribution of many dedicated scientists and engineers. Ma-

jor improvements include decrease in invasiveness – now enabling recordings in vivo, from awake, behaving animals, while not affecting the cellular behaviour – and increase in the number of simultaneously recorded cells.

Intracellular recording. One way of gaining access to single neurons' activity is via intracellular recordings. Widely used techniques, called voltage clamp and current clamp, require a microelectrode penetrating the cell membrane, as well as disturbing the natural changes of membrane potential, or flow of ionic currents. Sharp electrode recordings affect the ionic constitution of the intracellular fluid less; whereas whole-cell patch clamp – developed by Neher and Sakmann – now enables precise recordings of single neuron activity without penetrating the cell membrane. Intracellular recordings provide a detailed picture of single neurons' electrical state, but they perturb the observed cell, and are also very difficult to scale to a large number of neurons.

Extracellular recording. These issues may be fixed by extracellular recordings, in which electrodes do not approach the cellular membrane, but record only spiking activity from a distance, without perturbing the cell. Since spiking activity of cells is thought to represent a major part of cell-to-cell communication, and for certain cells has also been proven to contain information about external stimuli or behaviour, recording it indeed provides valuable insight into neural information processing; for example, it was used by Hubel and Wiesel to investigate the cat visual system. Unlike intracellular recordings, however, the data recorded on an extracellular electrode may contain electrical information from multiple cells, called multi-unit activity. The identification of single-unit activities – spikes from a single neuron – given the extracellularly recorded data is non-trivial, and resulted in a major data analysis challenge. The problem of 'spike sorting' involves understanding how a cell's spike appears in the recording, and describing the recorded signal as the superposition of apparent electrical activities from multiple sources – each with their own distinct characteristics. The sorting procedure results in data that contains the spike event times from multiple neurons. Initially, with single – or spatially distant – electrodes, the sorting problem was not robustly solvable, and multi-unit activities were manually identified and rejected. With the development of microelectrode arrays (and recently polytrodes), spike sorting of multi-unit activities became a feasible inverse problem, as single spikes appeared on multiple electrodes, providing more information about the location and spike shape of individual neurons. However, the increasing amounts of data made it more difficult for humans to manage, and led to advance of several algorithmic solutions; first human-assisted ones, with now completely automated methods available. The size and electrode density of microelectrode arrays, as well as the reliability of spike sorting algorithms is constantly increasing, and nowadays it is possible to simultaneously record spiking activity from hundreds or even thousands of neurons.

Optical imaging. An alternative approach to accessing neural activity is based on optical imaging. Of course, electrical activity of cells does not result in an imageable optical signal, and thus imaging methods involve a proxy: a molecule or protein that changes its optical properties depending on the electrical state of the cell. Cells naturally express proteins that change their optical absorption cross section depending on local ionic environment (for example NADH / NADPH), but those proved to be a weak proxy of the neuron's overall membrane potential changes. To enable more reliable imaging, electrically sensitive dyes or fluorescent sensor proteins may be introduced to the cells. This, of course, perturbs them, but the perturbation is thought to be much less severe than intracellular recordings. This is exemplified by the existence of transgenic animals – behaviourally indistinguishable from wild-type in many ways – living with neurons constantly expressing such sensor proteins[1]. The most widespread imaging technique is based on the concentration of calcium ions as the proxy of neural spiking, recognising the opening of calcium channels associated with firing, and the subsequent influx of calcium ions. The sensor is a heavily engineered fusion of green fluorescent protein and calmodulin, with the former providing the optical reporting, and latter changing the optical absorption when binding calcium. This construct is excited by a tightly focused illuminating beam (with diameter less than a single cell body), and the emitted fluorescent photons are collected by a microscope. The resulting data is correlated with the electrical activity of the recorded neuron. To measure the activity of multiple neurons, the focus of the illuminating beam is changed repeatedly, resulting in time-shifted, near-simultaneous recording of potentially millions of locations. We, however, face an issue similar to spike sorting before – the locations and spatial extents of the neurons are unknown, and must be estimated from the data itself. Recognising this problem - as well as due to physical constraints – the focus is scanned through physical space, and data is recorded on an evenly spaced grid, similar to an image or a video frame[2]; the whole process is called 'in vivo scanning two-photon microscopy of neural tissue via fluorescent calcium sensors', often referred to as 'calcium imaging'. Again, just like extracellular recordings, calcium imaging also gave rise to significant data analysis challenges. Firstly, we are not directly measuring electrical activity, but rather a doubly disassociated proxy of it: near-instantaneous electrical spikes first result in calcium ion influx, which then bind to the sensor protein with slower kinetics, finally resulting in a change in the recorded data. The problem of recovering the precise time of electrical spikes from so-called calcium transients is 'spike deconvolution'. The second major challenge concerns the localisation of neural cell bodies in the recorded images, and sorting the measured signal to individual cells; this is one of the challenges I address in this thesis, and is discussed thoroughly in chapter 3. The third challenge is related to the imaging process. We are not directly accessing the binding state of sensor molecules, but via illumination and collection processes, which are highly influenced by the brain

[1] Noting that some cortical activity in transgenic mice was shown to be aberrant recently (Steinmetz et al. 2017), questioning the generalisability of results.

[2] Noting that the values are not recorded at the same time, but rather in a time-shifted manner due to the scanning process.

tissue and microscope elements. Although this problem is less recognised and not often thought of, it has a significant effect on the data; thus I first present the nature of the problem, then propose a solution to it in chapter 2.

Dimensionality reduction and dynamical modelling. Having recorded the joint activity of a large number of neurons, the next step is to try and understand the information their activity represents. This information is often probed in the context of an experimental question: in particular, how well does the neural activity represent some other piece of information we measured – for example sensory input, or behavioural output. For individual neurons, this is relatively simple, we only need to take into account the noise model, as informed by the recording method; as an example, spike-triggered average reveals linear associations of extracellular single-unit activity with an other measured signal. However, the brain is thought to store information in a distributed and dynamic manner – that is, individual neurons may represent only part of the contained information, and over time, the same piece of information may be represented by a different activity pattern. The association of neural activity with some other signal is called 'population coding', which essentially projects the information represented in neural activity onto the space of the measured signal. Although this technique is extremely useful in understanding what elements of a signal a particular neural measurement does *not* represent, it does not attempt to describe all the contained information. To try and understand the information processing recorded in the neural activity eo ipso, on its own account, we need to represent the system in a human-interpretable way. We have two general ways of understanding high-dimensional, noisy, time-varying systems: Dimensionality reduction methods aim to recover a small number of time-varying latent variables that retain most of the information in the recorded neural time series; whereas dynamical models attempt to represent the joint temporal changes of neural activities via a parametric description. Both are capable of providing valuable insight into complex systems, and they can even be applied together – ultimately giving rise to a large number of algorithms. In chapter 4, I propose two novel methods, which are uniquely tailored towards low-dimensional modelling of noisy neural systems, while retaining their information content and providing an easily interpretable dynamical description of their behaviour.

1.2 The role of machine learning in biological data analysis

My proposed models contain numerous unknown parameters. When faced with unknown parameters, there are three broad strategies to follow: 1. If the parameters represent properties of the physical world, one can design methods and experiments to measure them, and use the measurements to constrain the model in question. This strategy proved popular for constraining and evaluating theoretical models in physics – even when the experiments are extremely complicated or expensive, such as the LIGO interferometer for gravitational waves, or the LHC particle accelerator for the standard

model. Biological systems, however, are inherently difficult to measure due to both their adaptability and the lack of well-understood separation of scales; we often do not know, how to design the right experiment to measure a particular parameter. As such, biological sciences often turn to a different strategy: 2. We may admit, that with our current techniques, some of the model parameters are not directly accessible, and design an approximate, or different model, that is not concerned with the un-measurable parameters. Even though we may already be aware of the shortcomings of the simpler model[3], it may not be advantageous to deal with more complicated ones, as long as they cannot be constrained enough to provide additional insight. A neuroscientific example is the use of single-neuron receptive fields, measured via the aforementioned spike-triggered average. Even though it is clear that the neuron does not only represent the particular stimulus, and neither does it represent all the animal's knowledge about the stimulus, it is still a useful way of classifying single neurons or even brain regions; one just need to be aware that the result may not hold under different conditions. The last strategy involves inference: 3. Although it may be not known how to directly measure a particular parameter, given the model, we may well estimate the influence of the parameter on variables that we can measure. Given the data, we can then infer the maximum likelihood setting, or even the probability distributions jointly over all unknown parameters; the resulting knowledge can inform the design of future experiments and models. Unsupervised machine learning is concerned with developing algorithms to infer unknown parameters of arbitrary models, and as such, it allows for defining complex generative models that reflect our full understanding of the experimental technique and the system under study. Ultimately, unsupervised learning reduces the gap between complex theoretical models and the available measurements, and I believe it will play a major role in the future of biological modelling and data analysis.

[3]To quote George Box: "Since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad." I find this especially fitting, as most of our data indeed concerns mice; yet the problems we wish to solve are much broader.

2

Spatial background equalisation of two-photon calcium imaging videos of neural tissue

Where the telescope ends, the microscope begins;
and who can say which has the wider vision?

– Victor Hugo, *Les Misérables*

In this chapter I examine the popular two-photon scanning calcium microscopy, and construct an an algorithm to recover the signal we were setting out to measure in the first place - the changes in calcium ion concentration, which is thought of as a proxy for the electrical activity of neurons. Unfortunately the recovery of the absolute ion concentrations is impossible given the usual experimental setup, but we can attempt to recover the true optical signal emitted by and captured from the biological sample, by learning a model of the recording microscope, and inverting it. Furthermore, this estimated fluorescent signal can then be used to infer the relative ion concentrations across the field of view and on different frames, by correcting for biological and optical effects that are known to affect fluorescence measurements of ion concentrations – for example uneven illumination or varying expression levels of the fluorescent calcium indicator. Ultimately we aim to infer a signal that only depends on actual changes of calcium concentration, free of all distortions caused by the observation method. The current need for such an algorithm will hopefully become clear enough through the detailed description of the issue, as well as illuminating example images.

In this chapter, I first describe the raw data in section 2.1, and discuss the possible causes of how the underlying signal we wished to measure got transformed into the data through the observation process. Next, in section 2.2, I build a flexible probabilistic inference framework that is capable of both including parametric models of the known elements in experimental procedure, and describing previously unknown or unmodeled distortions supported by the data. In section 2.3, I test the framework on openly available experimental data. Finally, in section 2.4, I discuss the strengths and

shortcomings of the proposed algorithm, and importantly, based on the current study of the system, I suggest ways to render parts of the proposed algorithm unnecessary in the future, by measuring the transformation parameters as part of the experimental data collection. This could lead to better standardisation of calcium imaging recordings, and ultimately more reproducible and robust scientific conclusions drawn from such experiments.

2.1 Calcium imaging - from signal to data

Although my main contribution is the development of a novel preprocessing algorithm, I believe before discussing the *how*, it is always important to make the *what* very clear. The reason for the widespread use of scanning two-photon microscopy via fluorescent calcium sensors - which I will often call calcium imaging in the rest of this chapter - is that it provides an estimate of neuronal activity at the single neuron level, in alive, and often awake animals. This activity is measured via the increase of Ca^{2+} ion concentration that stereotypically follows electrical spikes in neuronal somas. The animals' cells are reprogrammed (either genetically or via viral injections) to produce a protein that acts as fluorescent calcium sensor. Fluorescence is the ability of the protein to absorb photons at a certain wavelength, enter a higher energy state and later emit a photon of a different wavelength; whereas calcium sensing means that the efficacy of this process is modulated by whether or not the protein is bound to calcium. After decades of still ongoing optimisation of both the fluorescent calcium sensor protein (Chen et al. 2013a; Whitaker 2010) and the active imaging process, in the past few years the neuroscience community has largely converged on using a GCaMP6 sensor - an ultrafast rise and decay time constant variant, based on green fluorescent protein and calmodulin.

The chosen fluorescent calcium sensor is then illuminated by a scanning two-photon microscope and the emitted fluorescent signal photons are collected in the epi direction through the objective, then redirected to a photomultiplier tube via a dichroic mirror. Finally this signal is digitised, and saved as a so-called grey value of an image pixel. The pixel location is determined by the angle of the scanning mirrors used to deflect the illuminating beam. The set of these pixels over one cycle of mirror movement forms an image, and repeated cycles result in a set of images - a video.

This established process has generated thousands of hours of video and correspondingly thousands of terabytes of data over the past few years, many of them exhibiting similar features in the recordings. As shown in figure 2.1, the data often exhibits strong background spatial non-uniformity, which is mostly caused by non-uniform distribution of the calcium sensors, uneven illumination intensities and collection efficiencies that depend on the position within the image. An other effect in the collected data is more subtle, but causes substantial difficulty in interpreting the data. This relates to what happens to the emitted fluorescence photons arriving at the photomultiplier. As demonstrated in figure 2.2 the relationship between observed data and the signal can be non-trivial. Modelling a realistic photomultiplier is a difficult problem; what's worse, here we are faced with

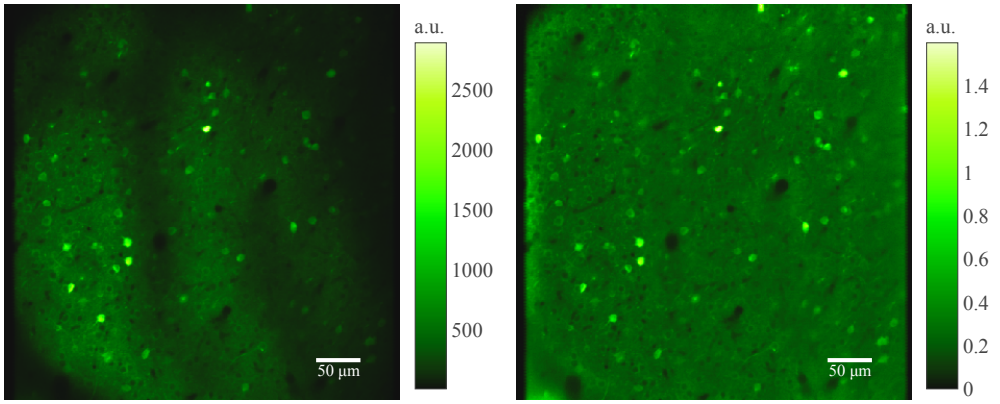


Figure 2.1: Neurofinder dataset 00.00. Original (left) and corrected (right) mean images, with the correction carried out by the algorithm proposed in this chapter, demonstrating its effect on the data. See section 2.3 for more details on the data and the correction.

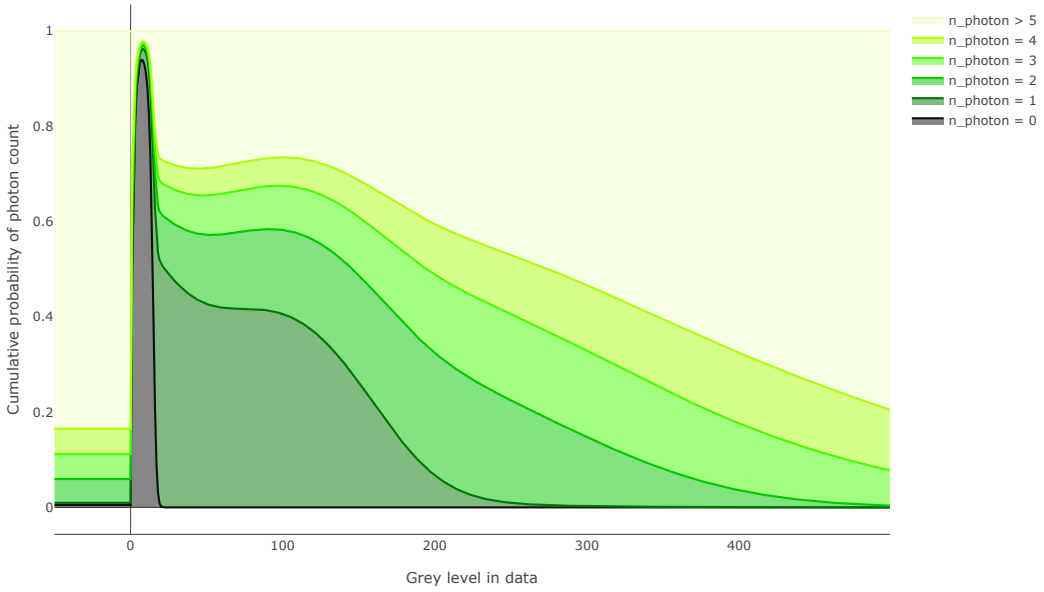


Figure 2.2: Demonstrating the conditional probability of the number of photons that reached the photomultiplier and generated a photoelectron; given we have observed a grey level in the raw data. Note that although the values less than 0 on the x axis serve purely visualisation purposes as the observations are strictly non-negative, this zero level is indeed a modifiable property of the observation system. This figure is the zoomed in version of figure 2.8a, see section 2.3 on how it was computed.

reverse-engineering one from data, as metadata about microscope specifications and settings are rarely shipped alongside published datasets. In order to attempt it, we need to clearly understand the individual parts of the system, and how they collectively result in the observed data.

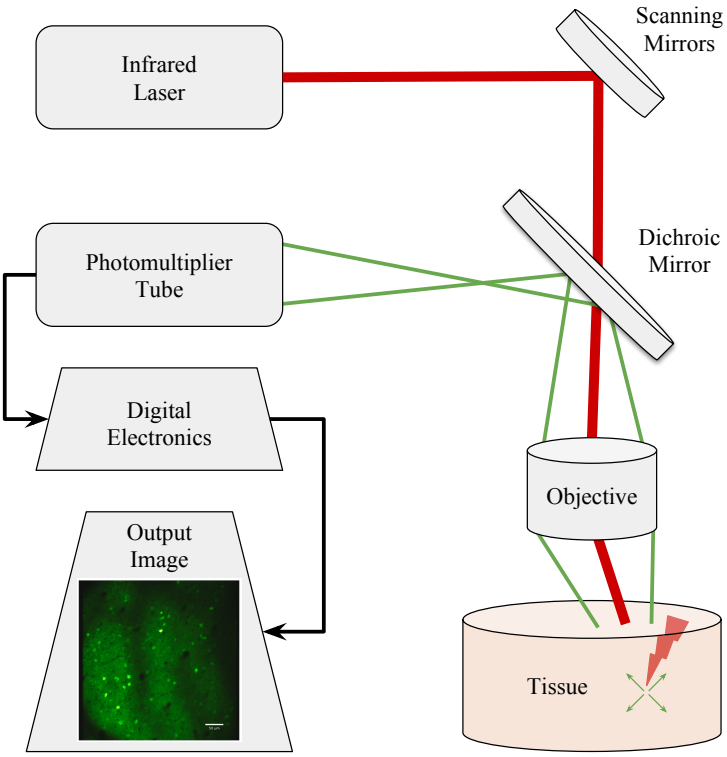


Figure 2.3: Scanning two photon microscopy schematic. The red light path indicates focused femto-second laser pulses from the illuminating laser. The green cones indicate the collection pathway of the fluorescently emitted highly scattered signal photons.

2.1.1 Scanning two-photon microscopy

In this section I describe how the illumination and collection processes result in an observed signal, given a latent true state of the calcium-bound fluorescent sensors. In order to record this state, one needs to probe the sensors via illumination, then collect the emitted fluorescent photons (figure 2.3). Here I give a brief introduction and focus on the elements that might introduce spatial distortions. For a detailed technical description of individual optical elements see Young et al. (2015).

Illumination. First the illumination pathway. The main elements we need to consider are the illuminating laser, the pair of scanning mirrors, the objective, and the tissue under examination. The infrared laser emits femtosecond pulses of long wavelength light. These light packets are deflected from the optical axis by the scanning mirrors, then focused by a high numerical aperture objective, ultimately creating a tight focal point that scans through the tissue as the mirrors rotate. In order to elicit fluorescent photons, two infrared photons need to excite the same sensor molecule concurrently, explaining the name of two-photon microscopy. This results in a much tighter effective point spread function around the focal spot, as the probability of evoking response scales quadratically with the field intensity, enabling two-photon microscopes to generate very little out-of-focus fluorescent signal. However, this quadratic dependence also causes small changes in the spatio-temporal

field strength around focal spot to be much more pronounced in the evoked signal.

Here I am interested in discovering spatial patterns in the changes of total effective illumination. There are three main causes for the existence of such spatial patterns. The first is introduced by the scanning mirrors. In order to achieve bidirectional deflection of the focal spot, two mirrors are required, one usually implementing fast X-directional (resonant) scanning, and another the slower orthogonal Y-directional one. As mentioned by major manufacturers (e.g. Olympus[1] and Leica[2]), ideally both mirrors should be in a plane conjugate to the back pupil of the objective lens, such that the laser beam can optimally fill the back pupil at all times using a so-called "4 f design", including a telescope and scan lens. For two mirrors, this is physically impossible, and compromises need to be made, usually resulting in the conjugate plane being in-between the two scan mirrors, and causing gradual loss of intensity as the scan angle increases and the focal spot moves further from the optical axis. If the mirrors are equidistantly placed from the conjugate plane, this type of aberration is radially symmetric around the optical axis, sometimes called vignetting; if the distances differ, we would notice decreasing field strength that differ about the X and the Y axes, called astigmatism. These effects are present in most datasets, as they are difficult to completely compensate for experimentally.

The second major cause for spatial aberrations is introduced by the objective itself. Although this is generally the optical element the most attention (and money) is paid towards, objective lenses are still not perfect. The X-Y tightness of the focus often widens as the focus moves away from the optical axis of the objective. In conventional microscopy this effect causes minor aberrations, but due to our quadratic dependence on field intensity in two-photon microscopy, this spatial dispersion results in an effective reduction of illumination in a radially symmetric manner about the optical axis.

Finally, the tissue above the focal plane itself attenuates the focused laser beam in a spatially unstructured way, due to the varying density of brain matter. Patterns of vasculature affect signals particularly strongly, well exemplified by their routine use in identifying the same field of view across multiple experimental sessions.

Collection. Once a small volume has been illuminated, the available fluorophores may get excited and subsequently emit a fluorescent photon. For fluorescent calcium sensors, the probability of such an absorption-emission event is modulated by whether or not the sensor molecule is bound to calcium, and thus serves as a proxy of the local calcium ion concentration. The generated fluorescent photon count therefore depends both on the number of sensors as well as calcium ions available. The way we can distinguish between signal caused by excess sensor count as opposed to more

[1] URL: `https://www.olympus-lifescience.com/en/microscope-resource/primer/techniques/confocal/confocalscanningsystems/`

[2] URL: `https://www.leica-microsystems.com/products/confocal-microscopes/technology/fov-scanner/`

calcium, is that calcium concentration varies rapidly in time, whereas the active sensor counts are essentially constant throughout the imaging, especially for modern non-bleaching sensor molecules. Nevertheless, the sensor molecule availability introduces another spatially unstructured effect, which is exacerbated for experiments utilising viral injections with non-uniform uptakes.

The emitted fluorescent photons then need to be collected (figure 2.3, green paths). Fluorophores emit photons uniformly with all directions, which are then scattered in the tissue. As we typically only collect photons in the epi direction through the objective, most of them are lost. Fortunately, even when a large portion of the photons is uniformly randomly lost, Poisson processes do not change in nature, rather they become a Poisson process of correspondingly lower intensity. This phenomenon is called thinning, and enables us to ignore the difficulties of estimating the absolute photon loss, and only focus on relative spatial variations. Although these variations of thinning during photon collection may again result from tissue density changes, this effect is much less detrimental for the scattered photons, than it was for the highly focused illuminating beam. Other effects include the radially symmetric reduction of collection efficiency due to the solid angle of the objective, as well as location- and angle-dependence of both the reflectivity of various optical pipeline elements, and the quantum efficiency of the photomultiplier, again causing non-uniform photon loss.

2.1.2 Recording the optical signal

The last stage of the data creation process is the digitisation of the signal that entered the photomultiplier tube (PMT)[3]. First, each photon may generate a photoelectron at a photocathode with some probability, which gets accelerated by a focusing electrode onto a successive chain of dynodes. Those exponentially amplify the signal via a cascade of secondary emissions. Finally, the resulting electron avalanche hits the anode, and generates a (temporally) sharp current pulse. The height of each pulse varies due to randomness in the amplification process, and the distribution of this variability may depend upon the location and angle of the photon incident on the photocathode[4]. There are two general ways of digitising the current that arrives at the anode. For low photon counts (up to $10^7 s^{-1}$, according to Hamamatsu), the individual current pulses are unlikely to overlap in time, and therefore it is possible to extract each event that passes a threshold, and obtain an integer number representing the number of generated photoelectrons. However, for higher photon counts the current pulses can not be resolved, and instead they are preamplified with a given input offset that thresholds DC components and low pulses. The resulting signal is then integrated in a time window corresponding to photons arriving from the same location in the sample. This integration time window is coupled with the scanning movement of the focal point in the sample with appropriate delays,

[3] Although PMTs are the most common detectors for two-photon microscopy, several new detector types are emerging. A different detector won't change the algorithm conceptually, but might require a different parametrisation of the likelihood function than the ones described in section 2.2.3 for PMTs

[4] For an extremely detailed description of characteristics and noise sources of photomultipliers, see the premier producer Hamamatsu's excellent PMT Handbook (edition 3), `URL: https://www.hamamatsu.com/resources/pdf/etd/PMT_handbook_v3aE.pdf`

and thus the integrated and digitised signal appears as the pixel's grey level in the raw output data. According to my estimated photon counts calculated later in this chapter (shown in section 2.3.2), calcium imaging recordings are generally below the resolution limit, and could be recorded in photon counting mode. Most published datasets, however, seem to have used amplification-integration process.

Unfortunately, the noisy amplification carried out by the photomultiplier – along with the thresholded integration by the subsequent electronics – significantly transform the incoming Poisson signal. The resulting digital grey level possesses an effective photomultiplier gain, which determines the average increase in pixel intensity for each added incident photon, and can be modified via changing the photomultiplier voltage. Furthermore, the preamplifier offset determines an arbitrary zero level, below which signals are not collected. Both these parameters are easily accessible to the experimenters to manually maximise the apparent signal to noise ratio, and ultimately they may change not only from microscope to microscope, but even from recording to recording, making the interpretation of raw data difficult.

The two stages of data generation described above in sections 2.1.1 and 2.1.2 are conceptually different. All processes before the photo-multiplication affect the mean parameter of a Poisson process, but do not change the nature of it, and therefore can be thought of as a purely multiplicative gain-like effect, applied to the calcium concentration changes. Conversely, the signal transformation carried out by the photomultiplier and subsequent electronics is a process that amplifies all entered photons. This important difference may be best summarised by saying that the pre-photomultiplier gain process changes the mean of the signal, but keeps the variance-mean ratio (the Fano factor) one; whereas the recording process carried out by the photomultiplier and electronics keeps the standard deviation-mean ratio (coefficient of variation) constant, but changes the Fano factor.

Ultimately we need to first infer the optical signal that entered the PMT by inverting the recording process, then strip it of any spatial dependence that is not caused by fast timescale changes in calcium concentration. This latter process may be called spatial signal equalisation.

2.2 Spatial signal equalisation via probabilistic inference

Having understood the underlying system, we need to translate it into a mathematical model that is capable of capturing the known complexities, but also has a reasonable behaviour when faced with unseen data. Furthermore, it needs to be able to incorporate the tens of millions of observations, that are both available and required to elucidate the latent signal. As the observations are inherently samples from an unknown probability distribution, my approach was to parametrise this probability

distribution in a flexible way. The inferred parameters then represent measurable parameters of the physical systems involved, and all other unknown effects are learned from the data in a semi-parametric fashion.

The explicit goals of the inference is to recover the optical signal that generated the amplified grey levels, and then to estimate and correct for spatial distortions introduced by the biological and optical elements. Therefore to sketch a generative model that achieves both aims, I assume we record repeatedly from a fixed set of locations[5], each of which exhibit changes in calcium ion concentrations. Those changes are then multiplied by a spatially non-uniform gain-like function, resulting from the effects summarised in section 2.1.1. Finally, the signals are observed probabilistically through a photomultiplier and other electronics, as described in section 2.1.2. I use the following notation:

$$\begin{array}{rl} \mathbf{x} & \text{Spatial location} \\ Z(\mathbf{x},\, t) & \text{True signal} \\ G(\mathbf{x}) & \text{Spatial gain} \\ Y(\mathbf{x},\, t) & \text{Observed pixel intensities} \end{array} \tag{2.1}$$

The definition of these individual elements already gives rise to structure in our probabilistic model:

$$\begin{array}{ll} p(\, Y \mid G,\, Z,\, X,\, \xi\,) & \text{is the Likelihood,} \\ p(\, G \mid X,\, \theta\,) & \text{is the Prior over the gain,} \\ p(\, G \mid Y,\, Z,\, X,\, \xi,\, \theta\,) & \text{is the Posterior over the gain,} \end{array} \tag{2.2}$$

where X is a fixed set of locations, while ξ and θ parameterise the likelihood and the prior, respectively. To place it in our physical model, the parameters of the prior may represent the lengthscales of typical distortions from off-axis lenses or chromatic aberration of objectives; whereas the parameters of the likelihood relate very directly to measurable or even manually changeable ones of the PMT, such as the quantum efficiency or the high voltage.

For compactness, I will often drop the explicit conditioning on X, Z, ξ and θ. The prior, $p(\, G\,)$, encodes our assumptions about the structure in the spatial gain, whereas the likelihood, $p(\, Y \mid G\,)$, is the probabilistic model of how the photomultiplier and subsequent electronics record the spatially distorted signal. The posterior, $p(\, G \mid Y\,)$, describes our belief about the gain function, having incorporated the observed data as evidence via inverting our current estimate of the likelihood.

In the following I first describe the generic parametric form of a variational inference model, then specify the forms of the prior and approximate posterior functions as an approximate Gaussian

[5]Throughout this thesis I make the assumption that the datasets are spatially two dimensional, as is most common in practice. However, both the model described and the code used to implement it should work for arbitrary dimensional datasets, with reasonable scaling.

Process. Finally, I detail a set of options for the prior and likelihood functions that correspond to our knowledge about the physical systems that implement them.

2.2.1 Variational inference

For models that describe a physical system in detail, the posterior is often non-conjugate and the normaliser is intractable. This is indeed the case for the likelihoods I discuss later in this chapter, and thus we have to turn to an approximation that we can compute for arbitrary priors and likelihoods. Due to its established track record, along with recent developments, I chose to employ a variational sparse approximation (Titsias 2009) to the posterior. This introduces a new parametrised distribution, called the variational distribution, which is in a tractable model class. This distribution is then used to construct a lower bound to the marginal log likelihood, that is called the variational lower bound:

$$q(G) = q(G \mid \psi) \approx p(G \mid Y) \qquad \text{is the Variational distribution}^6, \tag{2.3}$$

$$\log p(Y) = \log \int p(Y \mid G)\, p(G)\, dG \qquad \text{is the Marginal log likelihood.} \tag{2.4}$$

$$= \log \int p(Y, G)\, \frac{q(G)}{q(G)}\, dG$$

$$= \log \left(\mathbb{E}_q \left[\frac{p(Y, G)}{q(G)} \right] \right)$$

$$\geq \mathbb{E}_q \left[\log \frac{p(Y, G)}{q(G)} \right] \qquad \text{due to the Jensen inequality.} \tag{2.5}$$

$$\mathcal{L}(q) = \mathbb{E}_q [\log p(Y, G)] - \mathbb{E}_q [\log q(G)] \qquad \text{is the Variational lower bound.} \tag{2.6}$$

This lower bound approaches the marginal log likelihood as the KL-divergence between the true and the approximate posterior vanishes,

$$\mathcal{L}(q) = \log p(Y) - D_{\mathrm{KL}} [\, q(G) \parallel p(G \mid Y) \,], \tag{2.7}$$

therefore maximising the tractable variational lower bound is equivalent to minimising the KL divergence between the approximating and the true posterior. This approximation thus translates a difficult inference problem of finding an approximate posterior, into a generic optimisation one, maximising an objective function. A further advantage of this approach, is that the marginal log likelihood defined in equation 2.4 is in fact a function of not only the gain G, but also the parameters of the illumination or observation models. Reintroducing the conditioning on the latent signal and the parameters to the prior and the likelihood, shown in equation 2.2, we have

[6] Where ψ is the collection of so-called variational parameters.

$$\log p(\, Y \mid Z,\, \xi,\, \theta\,) = \log \int p(\, Y \mid G,\, Z,\, \xi\,)\; p(\, G \mid \theta\,)\; dG. \tag{2.8}$$

In fact, we may treat all parameter values as uncertain, as all physical measurements and calibrations are inherently noisy, and work with the joint distribution of the observations and parameters:

$$\log p(\, Y,\, Z,\, \xi,\, \theta\,) = \log \int p(\, Y \mid G,\, Z,\, \xi\,)\; p(\, G \mid \theta\,)\; p(Z)\; p(\xi)\; p(\theta)\; dG. \tag{2.9}$$

This results in an extremely flexible procedure, where knowledge about physical systems may easily be incorporated into the prior functions of parameters. The complete model therefore defines an objective that is a function of the latent signal, the parameters and the gain:

$$q(\, G \mid \psi\,) \approx\; p(\, G \mid Y,\, Z,\, \xi,\, \theta\,) \tag{2.10}$$

$$\log p(\, Y,\, Z,\, \xi,\, \theta\,) =\; \log\left(\mathbb{E}_{\mathrm{q}}\left[\frac{p(\, Y,\, G,\, Z,\, \xi,\, \theta\;)}{q(\, G \mid \psi\,)}\right]\right) \tag{2.11}$$

$$\begin{aligned}
\mathcal{L}(q) =\; & \mathbb{E}_{\mathrm{q}}[\log p(\, Y,\, G,\, Z,\, \xi,\, \theta\;)] - \mathbb{E}_{\mathrm{q}}[\log q(\, G \mid \psi\,)] \\
=\; & \mathbb{E}_{\mathrm{q}}[\log p(\, Y \mid\, G,\, Z,\, \xi\,) + \log p(G \mid\, \theta\,) + \log p(\, Z\,) + \log p(\xi) + \log p(\theta)] \\
& - \mathbb{E}_{\mathrm{q}}[\log q(\, G \mid\, \psi)]
\end{aligned}$$

The final objective function for our variational inference scheme is

$$\begin{aligned}
\mathcal{L}(q) =\; & \mathbb{E}_{\mathrm{q}}[\log p(\, Y \mid\, G,\, Z,\, \xi\,)] \\
& - D_{\mathrm{KL}}[\, q(\, G \mid\, \psi\,) \parallel p(\, G \mid\, \theta\,)] \\
& + \log p(\, Z\,) + \log p(\xi) + \log p(\theta).
\end{aligned} \tag{2.12}$$

To fully specify our variational inference model, we need to choose the prior parameter distributions $p(Z)$, $p(\xi)$ and $p(\theta)$, the prior and variational distributions over the gain, $p(G \mid \theta\,)$ and $q(\, G \mid \psi\,)$, and the likelihood $p(\, Y \mid\, G,\, Z,\, \xi\,)$. The prior distributions are generally either uninformative ones over a range of physically plausible values, if no measurements are available, or normal distributions around some measured values, with the variance arising from the reliability of the measurement process itself. To define the gain and the observation functions, I first derive an approximate Gaussian Process model in section 2.2.2, which sets the form of the prior and variational distributions over G. Afterwards, in section 2.2.3, I describe potential likelihood models.

2.2.2 Scalable approximate Gaussian Processes as prior functions

A highly scalable, yet extremely flexible model for the prior and variational distributions is a recently developed Gaussian Process approximation, that uses Structured Kernel Interpolation (Wilson and Nickisch 2015). It exploits the Kronecker structure inherent in imaging data, and the interpolation introduces additional smoothness structure to the modelled function. In order to describe this approximation step by step, I first define a full Gaussian Process model, introduce its sparse approximation, then choose the particular form of the approximation that is most suitable for our purposes.

A Gaussian Process over spatial locations is defined (Rasmussen and Williams 2006) by a mean function $m(\mathbf{x})$ and a positive definite covariance (or kernel) function $k(\mathbf{x},\, \mathbf{x}')$, and

$$f(\mathbf{x}) \;\sim\; \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x},\, \mathbf{x}')) \tag{2.13}$$

Given a matrix of spatial locations $\mathbf{X}_{\mathrm{nd}}$, these functions may be applied element- or pairwise, resulting in the mean vector $\mathbf{m}_{\mathrm{n}} = m(\mathbf{X}_{\mathrm{nd}})$ and the kernel matrix $\mathbf{K}_{\mathrm{n\mathring{n}}} = k(\mathbf{X}_{\mathrm{nd}}, \mathbf{X}_{\mathrm{nd}})$, respectively. This then defines a *prior* normal distribution of the function values at the given locations:

$$\mathbf{f}_{\mathrm{n}} \;\sim\; \mathcal{N}(\mathbf{m}_{\mathrm{n}}, \mathbf{K}_{\mathrm{n\mathring{n}}}) \tag{2.14}$$

Observing function values $\mathbf{f}_{\mathrm{n}} = f(\mathbf{X}_{\mathrm{nd}})$ defines a *posterior* distribution over new locations $\mathbf{X}^*_{\mathrm{md}}$, such that the joint distribution of the observed and predicted values is still normal:

$$\begin{bmatrix} \mathbf{f}_{\mathrm{n}} \\ \mathbf{f}^*_{\mathrm{m}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_{\mathrm{n}} \\ m(\mathbf{X}^*_{\mathrm{md}}) \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathrm{n\mathring{n}}} & k(\mathbf{X}_{\mathrm{nd}}, \mathbf{X}^*_{\mathrm{md}}) \\ k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}}) & k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}^*_{\mathrm{md}}) \end{bmatrix}, \right) \tag{2.15}$$

This then gives rise to a predictive distribution:

$$\begin{aligned} p(\mathbf{f}^*_{\mathrm{m}} \mid \mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}}, \mathbf{f}_{\mathrm{n}}) = \mathcal{N}\Big(& m(\mathbf{X}^*_{\mathrm{md}}) + k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}})\,[\mathbf{K}_{\mathrm{n\mathring{n}}}]^{-1}\mathbf{f}_{\mathrm{n}}\,, \\ & k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}^*_{\mathrm{md}}) - k(\mathbf{X}^*_{\mathrm{md}}, \mathbf{X}_{\mathrm{nd}})[\mathbf{K}_{\mathrm{n\mathring{n}}}]^{-1} k(\mathbf{X}_{\mathrm{nd}}, \mathbf{X}^*_{\mathrm{md}}) \Big) \end{aligned} \tag{2.16}$$

Unfortunately due to presence of the inverse $[\mathbf{K}_{\mathrm{n\mathring{n}}}]^{-1}$, this formulation is not capable of dealing with $N \gg 10000$ locations even on modern computers, and thus we need to employ approximations to scale to larger datasets. A popular way of doing so is the use of inducing points – a set of locations $\mathbf{X}^\dagger_{\mathrm{ud}}$ chosen (or optimised) in a way, such that they define an approximate kernel $\tilde{k}(\cdot,\cdot \mid \mathbf{X}^\dagger_{\mathrm{ud}}) \approx k(\cdot,\cdot)$, where the computational complexity of the predictive distribution using $\tilde{k}$ is reduced to to $\mathcal{O}(U^3 + U^2 N)$ from $\mathcal{O}(N^3)$. Popular examples are the subset of regressors (SoR) method (Silverman 1985) and the fully independent training conditionals (FITC) approximation (Snelson and Ghahramani

2006), where

$$\tilde{k}_{\mathrm{SoR}}(\mathbf{x},\mathbf{x}' \mid \mathbf{X}^{\dagger}_{\mathrm{ud}}) = k(\mathbf{x},\mathbf{X}^{\dagger}_{\mathrm{ud}})\left[k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1} k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{x}') \tag{2.17}$$

$$\tilde{k}_{\mathrm{FITC}}(\mathbf{x},\mathbf{x}' \mid \mathbf{X}^{\dagger}_{\mathrm{ud}}) = \delta_{\mathrm{xx}'}\,k(\mathbf{x},\,\mathbf{x}') + (1-\delta_{\mathrm{xx}'})\,\tilde{k}_{\mathrm{SoR}}(\mathbf{x},\mathbf{x}' \mid \mathbf{X}^{\dagger}_{\mathrm{ud}}). \tag{2.18}$$

In order for these types of approximation to result in large computational gains, $U \ll N$ is required, which limits predictive performance (Wilson 2014). This limitation may partially be resolved by exploiting existing Kronecker structure in a multidimensional kernel and by imposing Cartesian grid structure on inducing points, which results in an efficiently invertible circulant Toeplitz kernel matrix for stationary k covariance functions. Given product structure in the kernel and a D-dimensional grid with $U = V^D$ grid points, these reduce the computational complexity for learning and inference from $\mathcal{O}(U^3)$ to $\mathcal{O}(DU^{1+1/D})$ by using fast matrix decomposition, fast matrix-vector products and linear conjugate gradients. For further details see work by Wilson (2014), Wilson and Adams (2013), and Wilson and Nickisch (2015).

The next issue is that for N locations these approximations still require the computation of the cross-covariances $k(\mathbf{X}_{\mathrm{nd}},\mathbf{X}^{\dagger}_{\mathrm{ud}})$. That becomes the limiting step, with $\mathcal{O}(U^2N)$ complexity. However, this matrix may also be approximated efficiently, interpolating from the covariances already computed between inducing points; this is called Structured Kernel Interpolation (SKI, by Wilson and Nickisch (2015)). With $\mathbf{W}_{\mathrm{nu}}$ being a sparse interpolation weight matrix, we may approximate the cross-covariances as

$$k(\mathbf{X}_{\mathrm{nd}},\mathbf{X}^{\dagger}_{\mathrm{ud}}) \approx \mathbf{W}_{\mathrm{nu}}\,k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}}) \tag{2.19}$$

and thus

$$\begin{aligned} k(\mathbf{X}_{\mathrm{nd}},\mathbf{X}_{\mathrm{nd}}) &\approx k(\mathbf{X}_{\mathrm{nd}},\mathbf{X}^{\dagger}_{\mathrm{ud}})\left[k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1} k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}_{\mathrm{nd}}) \\ &\approx \mathbf{W}_{\mathrm{nu}}\,k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}})\left[k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1} k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}})^{\top}\mathbf{W}^{\top}_{\mathrm{nu}} \\ &= \mathbf{W}_{\mathrm{nu}}\,k(\mathbf{X}^{\dagger}_{\mathrm{ud}},\mathbf{X}^{\dagger}_{\mathrm{ud}})\mathbf{W}^{\top}_{\mathrm{nu}} \\ &= k_{\mathrm{SKI}}(\mathbf{X}_{\mathrm{nd}},\mathbf{X}_{\mathrm{nd}} \mid \mathbf{X}^{\dagger}_{\mathrm{ud}},\,\mathbf{W}_{\mathrm{nu}}). \end{aligned} \tag{2.20}$$

To finish the approximation, we need to specify the locations of the inducing points, $\mathbf{X}^{\dagger}_{\mathrm{ud}}$, and the interpolation scheme used to compute the sparse interpolation weights, $\mathbf{W}_{\mathrm{nu}}$. As discussed above, the inducing points need to be placed in a multidimensional grid over the span of the input data to retain scalability, thus one only needs to determine the bounds and the density of the grid.

These parameters are also influenced by our choice of interpolation scheme. In order to inherently impose a known lengthscale spatial smoothness on the gain function inferred, I employed an interpolation scheme, where the weights of the grid points are a finite-size Gaussian filter centred on the data points, then row-wise normalised:

$$w_{\mathrm{nu}}^{\mathrm{raw}} = \begin{cases} \dfrac{1}{\sqrt{2\pi\sigma_{\mathrm{filter}}^2}} \exp\left\{ \dfrac{-\sum_{\mathrm{d}} (x_{\mathrm{nd}} - x_{\mathrm{ud}}^{\dagger})^2}{2\sigma_{\mathrm{filter}}^2} \right\} & \text{if } d_{\mathrm{x}} < d_{\mathrm{max}} \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

$$w_{\mathrm{nu}} = w_{\mathrm{nu}}^{\mathrm{raw}} \;/\; \sum_{\mathrm{u}} w_{\mathrm{nu}}^{\mathrm{raw}}$$

The choices of d_{max} and σ_{filter} set the smoothness of our interpolation scheme. Finally, setting the grid spacing d_{grid} and the bounds b_{grid} fully defines a scalable model called Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP, by Wilson and Nickisch (2015)), which is used as the parametric form of the prior on the gain function. The parameters of the prior include the choice and parameters of the mean function m, the covariance function k, as well as the above parameters of the approximation scheme itself. In practice this latter is not optimised.

Given this definition, the prior, the posterior, and their KL-divergence can be computed. Let f be our modelled function, and we evaluate it at input points $\mathbf{X}_{\mathrm{nd}}$. The parametric prior over the function values is

$$\mathbf{F}_{\mathrm{n}} \;=\; f(\mathbf{X}_{\mathrm{nd}}) \tag{2.22}$$

$$p(\,\mathbf{F}_{\mathrm{n}} \;\mid\; \mathbf{X}_{\mathrm{nd}},\, \theta\,) = \mathcal{N}\Big(\mathbf{F}_{\mathrm{n}} - \mathbf{W}_{\mathrm{nu}}\, m(\mathbf{X}_{\mathrm{ud}} \;\mid\; \theta\,),\;\; k_{\mathrm{SKI}}(\,\mathbf{X}_{\mathrm{nd}},\, \mathbf{X}_{\mathrm{nd}} \;\mid\; \theta\,)\Big), \tag{2.23}$$

where $\theta = \{\, m,\, \theta_{\mathrm{m}},\, k,\, \theta_{\mathrm{k}},\, \theta_{\mathrm{interp}} \,\}$ is the collection of prior parameters. The approximate posterior uses variational parameters ψ to define the mean and covariance structure over the inducing points, then employs the interpolation scheme to approximate the posterior over the input locations:

$$\psi = \{\, \boldsymbol{\psi}^{\mathrm{m}}_{\mathrm{u}},\ \boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}} \,\} \tag{2.24}$$

$$q(\,\mathbf{F}_{\mathrm{n}} \mid \mathbf{X}_{\mathrm{nd}},\ \boldsymbol{\psi}\,) = \mathcal{N}\left(\mathbf{F}_{\mathrm{n}} - \mathbf{W}_{\mathrm{nu}}\boldsymbol{\psi}^{\mathrm{m}}_{\mathrm{u}},\ \mathbf{W}_{\mathrm{nu}}\boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}}\mathbf{W}^{\top}_{\mathrm{n\mathring{u}}}\right) \tag{2.25}$$

$$D_{\mathrm{KL}}(\,q \parallel p\,) = \frac{1}{2}\left[\begin{array}{l} -D + \log\dfrac{|k(\mathbf{X}^{\dagger}_{\mathrm{ud}}, \mathbf{X}^{\dagger}_{\mathrm{ud}})|}{|\boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}}|} \\ + \mathrm{Trace}\left\{\left[\mathrm{k}(\mathbf{X}^{\dagger}_{\mathrm{ud}}, \mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1}\boldsymbol{\psi}^{\mathrm{cov}}_{\mathrm{u\mathring{u}}}\right\} \\ + (\boldsymbol{\psi}^{\mathrm{m}}_{\mathrm{u}} - m(\mathbf{X}_{\mathrm{ud}} \mid \theta))\left[k(\mathbf{X}^{\dagger}_{\mathrm{ud}}, \mathbf{X}^{\dagger}_{\mathrm{ud}})\right]^{-1}(\boldsymbol{\psi}^{\mathrm{m}}_{\mathring{\mathrm{u}}} - m(\mathbf{X}_{\mathring{\mathrm{u}}\mathrm{d}} \mid \theta)) \end{array}\right]. \tag{2.26}$$

Although this defines the variational posterior and the KL-divergence required to computed the objective function in our variational inference scheme equation 2.12, the particular forms of the the mean and covariance functions are still unspecified.

Probabilistic models of spatial distortions

The next step therefore is to design mean and covariance functions of the prior that correctly encode our assumptions. First and foremost, all interactions that lead from the true signal $Z(\mathbf{x},\, t)$ to the optical signal arriving at the photomultiplier may be thought of as strictly multiplicative effects, changing the mean parameter of a Poisson distribution via thinning; therefore only the product of G and Z affects the likelihood of the data:

$$p(Y \mid G,\, Z) = p(Y(\mathbf{x},\, t) \mid G(\mathbf{x})Z(\mathbf{x},\, t)). \tag{2.27}$$

As both G and Z are strictly positive quantities, we may model their multiplicative interaction as a sum in logarithmic space, $GZ = \exp(\log G + \log Z)$. This observation enables us to model the numerous multiplicative interactions resulting in the product G – see section 2.1.1 – as a sum of functions, each representing the individual contribution of an optical or biological element. These functions then add up to $\log G$, with the premise that our prior and approximate posterior uncertainties of the gain function will now be log-normally distributed. This actually ensures the physically constrained positivity of G, and with a slight rewrite of the variational inference scheme in equation 2.12 results in the unconstrained scheme

$$
\begin{aligned}
F &= \log G \\
\mathcal{L}(q) = \; & \mathbb{E}_{\mathrm{q}}\left[\log p(\, Y \mid e^{F+\log Z},\, \xi \,)\right] \\
& - D_{\mathrm{KL}}\left[\, q(\, F \mid \psi \,) \,\|\, p(\, F \mid \theta \,)\right] \\
& + \log p(\, \log Z \,) + \log p(\xi) + \log p(\theta).
\end{aligned}
\tag{2.28}
$$

There are numerous ways of making assumptions about the form of the additive F, I describe examples here that are motivated by our knowledge about the illumination, the tissue and the collection system, discussed in section 2.1.1. Each distortion effect caused by the various elements enters additively into the final logarithmic gain: $F = \sum_{\mathrm{i}} F^{\mathrm{i}}$, where each F^{i} represents the logarithmic spatial gain distortion introduced by a single element. Having no explicit knowledge about the particular optics and settings of a microscope system, a reasonable prior assumption is that the distortion they introduce is unknown, and thus the prior mean function for each element is uniformly zero. However, this additive structure neatly lends itself to designing an additive covariance function, with the individual elements generally describing two types of spatial variations. Each element introduces one of these simple distortions, leading to a complex gain function model. One class is informed by the symmetries about the optical axes, and the other is well described by a characteristic lengthscale of spatial smoothness of the introduced distortion.

Symmetric distortions. In order to capture the symmetries present in the system, I introduce two covariance function transformations. Given a covariance function $k_{\mathrm{D}} : \mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{D}} \to \mathbb{R}$ and the location of the focal point along the optical axis in space $\mathbf{x}_{\mathrm{d}}^{\mathrm{o}} \in \mathbb{R}^{\mathrm{D}}$, we may define the linearly symmetrised covariance function

$$
k_{\mathrm{LinSymm}}(\, \mathbf{x}_{\mathrm{d}},\, \mathbf{x}'_{\mathrm{d}} \mid k_{\mathrm{D}},\, \mathbf{x}_{\mathrm{d}}^{\mathrm{o}} \,) = k_{\mathrm{D}}(\, \mathrm{abs}(\mathbf{x}_{\mathrm{d}} - \mathbf{x}_{\mathrm{d}}^{\mathrm{o}}),\, \mathrm{abs}(\mathbf{x}'_{\mathrm{d}} - \mathbf{x}_{\mathrm{d}}^{\mathrm{o}}) \,), \tag{2.29}
$$

where the absolute value is applied dimension-wise. Functions drawn from a Gaussian Process with a linearly symmetrised covariance function will exhibit reflection symmetries about each cardinal spatial axes, which are defined in scanning microscopes as a basis of the transverse plane of the optical axis[7] with the basis directions given by the scanning mirror orientations. Similarly, given a one-dimensional base function, $k_1 : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, we may define a radially symmetric covariance function as

[7] It is worth mentioning here that some microscope objectives introduce a curvature as the focal point is moved off-axis, and therefore the actually imaged manifold is not truly a transversal plane with respect to the optical axis. Therefore the point cloud $\mathbf{X}_{\mathrm{nd}}$ that is usually assumed coplanar may not be, and the calculated distances are slightly overestimated. Nevertheless, the symmetries are still present, and they do not truly depend on distance from optical axis, but rather the scan angles, and thus are not affected by objective-induced sample plane curvature.

$$k_{\text{RadSymm}}(\,\mathbf{x}_\text{d},\,\mathbf{x}'_\text{d}\;\mid\;k_1,\,\mathbf{x}^\text{o}_\text{d}\,)=k_1(\,\left\|\mathbf{x}_\text{d}-\mathbf{x}^\text{o}_\text{d}\right\|,\,\left\|\mathbf{x}'_\text{d}-\mathbf{x}^\text{o}_\text{d}\right\|\,). \tag{2.30}$$

Distortion lengthscales. Finally, we need to estimate the various spatial lengthscales of the distortions and capture them with the appropriate covariance functions. The Radial Basis Function covariance (known also as Squared Exponential, Gaussian or Exponentiated Quadratic kernel) models diminishing strictly positive interactions that lead to smooth functions, and thus is an excellent way of describing smooth distortions. It may be parametrised by a single lengthscale, in which case the interaction decay is isotropic; or by a vector of lengthscales, one for each dimension:

$$k_{\text{RBF}}(\,\mathbf{x}_\text{d},\,\mathbf{x}'_\text{d}\;\mid\;l\,)=\exp\left(-\frac{\left\|\mathbf{x}_\text{d}-\mathbf{x}'_\text{d}\right\|^2}{2l^2}\right) \tag{2.31}$$

$$k_{\text{RBF}}(\,\mathbf{x}_\text{d},\,\mathbf{x}'_\text{d}\;\mid\;\mathbf{l}_\text{d}\,)=\exp\left(-\sum_\text{d}\frac{\left(x_\text{d}-x'_\text{d}\right)^2}{2l_\text{d}^2}\right). \tag{2.32}$$

2.2.3 Photomultiplier likelihood models

As described in section 2.1.2, the photomultiplier and subsequent electronics translate the optical signal into a current, then finally a digitised voltage, represented by pixel grey values. Due to the limitations of photon counting, this digital signal usually represents an analog integration of the variable height current pulses induced by the incoming photons in a given time window, that corresponds to the dwell time of the optical focus on a pixel. The conversion of pulses includes a user-controllable offset of the preamplifier, that essentially thresholds the incoming signal pulses and thus sets the zero level for the Analog-Digital Converter (ADC), which then appears as zero pixel intensity in the recorded data. The effective gain from photon count to pixel intensity is the product of the user-configurable photomultiplier gain (set by the photomultiplier high voltage, and converts photoelectrons to mean current pulse height) and the usually fixed preamplifier gain (that converts current into voltage). The electronic components before digitisation also introduce additive zero-mean noise into the system. These are the three most important parameters of our likelihood function: $\xi=\{\,\xi^o,\xi^g,\,\xi^{\sigma_0^2}\,\}$ are the offset, the gain and the electric noise variance, respectively. These form the basis of any photomultiplier model.

Let λ be the instantaneous intensity of the thinned Poisson process that generates photoelectrons at the photomultiplier cathode; then we wish to parameterise the likelihood in terms of the photon flux:

$$\begin{aligned}\lambda&=e^{F+\log Z} && \text{Photon flux}\\ p(\,Y\mid\,\lambda,\,\xi\,) && \text{Likelihood.}\end{aligned} \tag{2.33}$$

It is also important to consider the analog to digital conversion, which introduces lower and upper thresholds (cut-off and saturation), and thus – given the likelihood functions defined in this section, which describe the signal incoming to the analog digital converter – the true likelihood of the digital signal is

$$
\begin{aligned}
p(\, Y_{\text{digital}} = 0 \mid \lambda,\, \xi \,) &= \int_{-\infty}^{0} p(\, Y \mid \lambda,\, \xi \,)\, dY \\
p(\, Y_{\text{digital}} = \text{ADC}_{\text{max}} \mid \lambda,\, \xi \,) &= \int_{\text{ADC}_{\text{max}}}^{\infty} p(\, Y \mid \lambda,\, \xi \,)\, dY \\
p(\, Y_{\text{digital}} = i \mid \lambda,\, \xi,\, 0 < Y_{\text{digital}} < \text{ADC}_{\text{max}} \,) &= \int_{\text{i}-1}^{i} p(\, Y \mid \lambda,\, \xi \,) dY \,,
\end{aligned}
\tag{2.34}
$$

where ADC_{max} is the number of channels on the ADC. This process is taken into account during any computations, but are omitted from the following description for brevity.

In the following I introduce three models of the photomultiplier and subsequent electronics, which best describe reality for distinct values of the light intensity λ.

Model 1 - Intensity amplification

The simplest possible model disregards the discrete nature of incoming light, and thus is only generally applicable to situations with very high incoming photon counts. Even though our data is not in this regime, this model is frequently applied, and thus worth discussing. It approximates the Poisson photon distribution with intensity λ as a normal distribution. The contribution of the photomultiplier is the gain that multiplies the random variable representing the observed intensity, whereas the ADC cut-off shifts its mean. Often a third parameter is also considered, $\xi^{\sigma_0^2}$ is the variance of the zero-mean noise introduced by the electronics after the amplification process. The whole model is

$$
\nu \sim \mathcal{N}(\, \lambda,\, \lambda \,) \tag{2.35}
$$

$$
p(\, Y \mid \nu,\, \xi \,) = \mathcal{N}_{\text{Y}}\Big(\, \xi^{g}(\nu - \xi^{o}),\, \xi^{\sigma_0^2} \,\Big), \tag{2.36}
$$

therefore

$$
\xi_1 = \{\, \xi^{o}, \xi^{g},\, \xi^{\sigma_0^2} \,\} \tag{2.37}
$$

$$
p_1(\, Y \mid \lambda,\, \xi \,) = \mathcal{N}_{\text{Y}}\Big(\, \xi^{g}(\lambda - \xi^{o}),\, \xi^{g^2}\lambda + \xi^{\sigma_0^2} \,\Big). \tag{2.38}
$$

Model 2 - Gaussian single electron response

For the lower photon counts observed in our data, it is important to take into account the discrete nature of the incoming signal. The photomultiplier and subsequent electronics then amplify each photoelectron into a pixel intensity according to the single electron response (SER). This is described by a probability distribution of the total charge generated by a single event, and is often approximated by a normal distribution. Multiple photoelectrons result in a signal that is the sum of the independent pulse charges generated.

$$\begin{aligned} \nu &\sim \text{Poisson}(\,\lambda\,) && (2.39)\\ p_{\text{SER}}(\,C \mid\, \nu = 1,\, \xi\,) &\approx \mathcal{N}_{\text{C}}\Big(\,\xi^{g},\, \xi^{\sigma_1^2}\,\Big) && (2.40)\\ p(\,C \mid\, \nu,\, \xi\,) &= \mathcal{N}_{\text{C}}\Big(\,\xi^{g}\,\nu,\, \xi^{\sigma_1^2}\,\nu\,\Big) && (2.41)\\ p(\,Y \mid\, C,\, \xi\,) &= \mathcal{N}_{\text{Y}}\Big(\,C - \xi^{o},\, \xi^{\sigma_0^2}\,\Big), && (2.42) \end{aligned}$$

where ν is a discrete count and C is the charge generated by the photomultiplier. Therefore

$$\begin{aligned} \xi_2 &= \{\,\xi^{o}, \xi^{g},\, \xi^{\sigma_0^2},\, \xi^{\sigma_1^2}\,\} && (2.43)\\ p_2(\,Y \mid\, \lambda,\, \xi\,) &= \sum_{\nu=0}^{\infty} p(Y \mid \nu)\, p(\nu \mid \lambda)\\ &= \sum_{\nu=0}^{\infty} \mathcal{N}_{\text{Y}}\Big(\,\xi^{g}\,\nu - \xi^{o},\, \xi^{\sigma_1^2}\,\nu + \xi^{\sigma_0^2}\Big) \cdot \text{Poisson}_{\nu}(\,\lambda\,). && (2.44) \end{aligned}$$

Ultimately this likelihood distribution is a Poisson-weighted mixture of Gaussians, where the mixture components' mean and variance scales with the number of observed photons. In practice the infinite sum may be replaced by a finite sum, where the components with diminishingly low Poisson mixture weights are omitted, and the distribution is renormalised accordingly.

Model 3 - Underamplified single electron response

Photomultipliers with low high-voltage settings may not amplify each photoelectron maximally, and thus the single electron response probability distribution is not well-approximated by a normal distribution anymore (Dossi et al. 2000; De Haas and Dorenbos 2010). We may describe the charge distribution of underamplified events as an exponential distribution, with a given rate ξ^{e}. The single electron response distribution is a Bernoulli mixture of fully amplified and underamplified events, with probabilities $1 - \xi^{p_{\text{e}}}$ and $\xi^{p_{\text{e}}}$, respectively. Fortunately, already for two photons within the same time window, the distribution of their added charges are reasonably approximated by a normal distribution, with the approximation proceeding similarly to the previous model. We can thus treat

the single observed photon as a special case:

$$\nu \sim \text{Poisson}(\lambda) \tag{2.45}$$

$$p_{\text{SER}}(C \mid \nu = 1, \xi) \approx \xi^{p_e} \cdot \text{Exponential}_{\text{C}}(\xi^{\text{e}}) + (1 - \xi^{\text{p}_\text{e}}) \cdot \mathcal{N}_{\text{C}}\left(\xi^{\text{g}}, \xi^{\sigma_1^2}\right) \tag{2.46}$$

$$p(C \mid \nu, \nu \geq 2, \xi) = \mathcal{N}_{\text{C}}\left(\mathbb{E}_{\text{p}_\text{SER}}[C] \cdot \nu, \text{Var}_{\text{p}_\text{SER}}[\text{C}] \cdot \nu\right) \tag{2.47}$$

$$p(Y \mid C, \xi) = \mathcal{N}_{\text{Y}}\left(C - \xi^{o}, \xi^{\sigma_0^2}\right). \tag{2.48}$$

In order to derive the resulting probability distribution over Y, given λ, we need to be able to describe the sum of an exponentially distributed random variable and the additive zero-mean noise with variance $\xi^{\sigma_0^2}$, which is an Exponentially Modified Gaussian (EMG) distribution, defined as

$$\begin{aligned} a &\sim \text{Exponential}(\lambda) \\ b &\sim \mathcal{N}(\mu, \sigma^2) \\ x &= a + b \\ \text{EMG}_{\text{x}}(\lambda, \mu, \sigma^2) &= \frac{\lambda}{2} \cdot e^{0.5\lambda(2\mu + \lambda\sigma^2 - 2x)} \cdot \frac{2}{\sqrt{\pi}} \int_{\text{x}}^{\infty} e^{-t^2} dt. \end{aligned} \tag{2.49}$$

Furthermore, we need to compute the mean and variance of the underamplified SER in order to evaluate the multi-photoelectron components:

$$\begin{aligned} \mathbb{E}_{\text{p}_\text{SER}}[C] &= \xi^{p_e} / \xi^{e} + (1 - \xi^{p_e}) \xi^{g} \\ \text{Var}_{\text{p}_\text{SER}}[\text{C}] &= 2\xi^{p_e} / \xi^{e^2} + (1 - \xi^{p_e})(\xi^{g^2} + \xi^{\sigma_1^2}) - (\mathbb{E}_{\text{p}_\text{SER}}[C])^2 \end{aligned} \tag{2.50}$$

As a result, the complete model is

$$\xi_3 = \{\xi^{o}, \xi^{g}, \xi^{\sigma_0^2}, \xi^{\sigma_1^2}, \xi^{e}, \xi^{p_e}\} \tag{2.51}$$

$$\begin{aligned} p_3(Y \mid \lambda, \xi) = {} & \text{Poisson}_0(\lambda) \cdot \mathcal{N}_{\text{Y}}\left(-\xi^{\text{o}}, \xi^{\sigma_0^2}\right) \\ & + \text{Poisson}_1(\lambda) \cdot \Big(\xi^{\text{p}_\text{e}} \, \text{EMG}_{\text{Y}}(\xi^{\text{e}}, -\xi^{\text{o}}, \xi^{\sigma_0^2}) + \\ & \qquad (1 - \xi^{p_e}) \, \mathcal{N}_{\text{Y}}\left(\xi^{g} - \xi^{o}, \xi^{\sigma_1^2} + \xi^{\sigma_0^2}\right)\Big) \\ & + \sum_{\nu=2}^{\infty} \Big(\text{Poisson}_{\nu}(\lambda) \cdot \\ & \qquad \mathcal{N}_{\text{Y}}\left(\mathbb{E}_{\text{p}_\text{SER}}[C] - \xi^{o}, \text{Var}_{\text{p}_\text{SER}}[\text{C}] + \xi^{\sigma_0^2}\right)\Big). \end{aligned} \tag{2.52}$$

2.2.4 Heuristics for data subselection

Given a model of the optical signal intensity described in section 2.2.2, and a likelihood model from section 2.2.3, we are now equipped to carry out variational inference (section 2.2.1) and estimate the parameters of a realistic generative model of the data. There is, however, one latent variable, that we have not yet discussed in detail; the true signal $Z(\mathbf{x},\, t)$. If no assumptions are made about this latent variable, it introduces an inherent degeneracy to the spatial model, as it would be capable of explaining any structure, which we want to represent via G, the spatial gain. Therefore the first assumption we wish to make, is that there is no spatial structure in the signal. This assumption only stands when looking at the background, not when we consider the whole dataset, which contains neuronal structures generating informative signal. In order to make this assumption correct, we need to reject locations that are thought to have spatially structured activity – essentially we wish to find the background, and learn the model on the background pixels only.

Most neural structures have an apparent spatial extent of multiple pixels in our recordings, and those pixels are correlated by the joint underlying source, a phenomenon I explore in detail in chapter 3. As a result, having a low cross-correlation coefficient with its neighbours turned out to be a robust heuristic measure of identifying background pixels. Given a small neighbourhood, and an averaging filter f, we may estimate the local expected activity, and compute the cross-correlation of the pixel's signal with the local activity:

$$\begin{aligned} \mu(\,\mathbf{x},\, t\,) &= \mathbb{E}_{\mathring{\mathbf{x}}\in\mathrm{Neigh}(\mathbf{x})}\left[f(\mathring{\mathbf{x}})\cdot Y(\,\mathring{\mathbf{x}},\, t\,)\right] \\ \mathrm{CrossCorrCoeff}(\,\mathbf{x}\,) &= \frac{\mathbb{E}_{\mathrm{t}}\Big[\left(Y\,(\mathbf{x},\, t\,) - \mathbb{E}_{\mathrm{t}}[Y\,(\mathbf{x},\, t\,)]\right)\cdot\left(\mu(\,\mathbf{x},\, t\,) - \mathbb{E}_{\mathrm{t}}[\mu(\,\mathbf{x},\, t\,)]\right)\Big]}{\sqrt{\mathrm{Var}[\mathrm{Y}\,(\mathbf{x},\,\mathrm{t})]\mathrm{Var}[\mu(\,\mathbf{x},\,\mathrm{t}\,)]}}, \end{aligned} \tag{2.53}$$

where μ essentially represents the result of a convolution of f with the mean of Y.

Low cross-correlation (and information loss) in the signal may also be caused by the ADC thresholding. This means that values that are equivalent to 0 or $\mathrm{ADC}_{\mathrm{max}}$ could have represented any signals below or above these thresholds (see equation 2.34), and thus shift the mean and lower the spread during computation of the cross-correlation coefficient. Pixels with a high portion of thresholded signals were excluded due to the unreliability of these statistics.

Another significant effect of the experimental data collection process is the introduction of missing observations. During recording of the data, the raw, digitalised observations – generated in real time – are stored in finite-size memory buffers. These are then read and flushed by the controlling computer asynchronously. This process is less reliably than other operations in the microscope, which leads to presence of missing observations. Unfortunately, these are often represented by 0 values in the dataset – the same as low, thresholded signals – rather than being marked as missing. We

may attempt to tell apart missing observations from low signals by their different spatial structures. Provided that the data has been recorded with recommended gain and offset settings, thresholded signals should be very rare, unlikely to appear in neighbouring pixels. Long aligned streaks of zero values – whose spatial alignment is due to the temporal continuity of scanning – are significantly more likely to have been caused by missing data, than by truly observing multiple very low signals. The detection of aligned zeros may be carried out by convolution with a set of neighbourhood filters. For detecting any x-direction streaks of length at least k, I generate filters

$$f^i = [\, \mathbf{0}_{\mathrm{min(-i,0)}},\, \mathbf{1}_{\mathrm{k}},\, \mathbf{0}_{\mathrm{min(i+1,0)}} \,], \quad i \in [-k+1, k-1], \tag{2.54}$$

where the vectors of 0s and 1s are concatenated in the direction of fast scanning. If any of the filters return a zero value, that means that there has been a streak of 0s that the current pixel is part of, and thus it should be treated as missing data, rather than a true 0 valued observation. We may also notice and detect similar streaks of 0s in the slow scanning direction – especially near the edges of the field of view – which may be due to slight overlap of memory buffering and flushing. See figure 2.5 for an example of identifying missing values in real data. Once missing observations have been flagged, they may either be removed from the dataset, or imputed. Imputation is a common statistical operation in data preprocessing, and it means replacing the missing values with its expectation according to some metric. As a number of algorithms cannot deal with missing observations, I will refer to the imputed datasets that had each of its missing values successively replaced by the mean of its 3x3x3 spatio-temporal neighbourhood.

A final, computationally crucial assumption we wish to make about the selected background, non-thresholded pixels, is that all their apparent temporal activity is purely due to noise. If this is true, our model may be significantly simplified, as not only will $Z(\mathbf{x},\, t)$ lack spatial structure (by subselecting background pixels via equation 2.53), but also temporal one. This enables us to model the latent true signal with a single constant value, $Z(\mathbf{x},\, t) \,=\, z$. Again, in some datasets this assumption definitely cannot be made, as there are clear brightness changes over time, even in the background. However, these brightness changes are often due to either a non-spatially specific general increase in signal; or a slight movement of the whole sample relative to the microscope, as the animal carries out certain behaviours. Lateral movement artefacts are usually corrected before publishing the data and should be done by the user before applying this method (see work by Greenberg and Kerr (2009), Chen et al. (2012) or Pnevmatikakis and Giovannucci (2017), for examples of this extensively studied operation), but these do not correct for the induced overall temporal changes. As the change affects all background pixels similarly, these behaviour-elicited temporal changes may be readily identified and removed from the data via singular value decomposition. Therefore as a final heuristic preprocessing step, we may apply the following:

$$\mathbf{Y}(\mathbf{x},\, t) = \sum_{\mathrm{i}=1}^{K} s_{\mathrm{i}}\, \mathbf{U}_{\mathrm{i}}(\mathbf{x}) \mathbf{V}_{\mathrm{i}}(t) \tag{2.55}$$

$$\mathbf{Y}_{\mathrm{corr}}(\mathbf{x},\, t) = \mathbf{Y}(\mathbf{x},\, t) - \sum_{\mathrm{i}=1}^{K_{\mathrm{shared}}} s_{\mathrm{i}}\, \mathbf{U}_{\mathrm{i}}(\mathbf{x}) \mathbf{V}_{\mathrm{i}}(t), \tag{2.56}$$

where $K_{\mathrm{shared}} \ll K$ is the number of globally shared temporal components, and is usually set to 1 or 2, based on the singular value spectrum $\{s_{\mathrm{i}}\}_{\mathrm{i}=1}^{K}$.

To summarise, after applying the above defined heuristics, we may assume that the data Y only contains background pixels. The remaining spatial distortions are purely due to the effective spatial gain we wish to model, and any temporal variation of the observed data is only due to the various sources of noise present in the system. As a result, we can further simplify our variational inference scheme of equation 2.28, assuming that the temporal samples at the same location are independent and identically distributed, and they are fully described by a single intensity value, which only depends on the modelled spatial gain function G, and a constant latent signal z.

$$\begin{aligned}
F(\mathbf{x}) &= \log G(\mathbf{x}) \\
\lambda(\mathbf{x}) &= e^{F(\mathbf{x}) + \log z} \\
\log p(\, Y \mid \lambda,\, \xi\,) &= \sum_{\mathrm{t}} \log p(\, Y(\mathbf{x},\, t) \mid \lambda(\mathbf{x}),\, \xi\,) \\
\mathcal{L}(q) &= \mathbb{E}_{\mathrm{q}}[\log p(\, Y \mid \lambda,\, \xi\,)] \\
&\quad - D_{\mathrm{KL}}[\, q(\, F \mid \psi\,) \,\|\, p(\, F \mid \theta\,)] \\
&\quad + \log p(\, \log z\,) + \log p(\xi) + \log p(\theta)
\end{aligned} \tag{2.57}$$

Equipped with this objective function, using the spatial gain function model described in section 2.2.2 and a likelihood model from section 2.2.3, we may optimise all parameters $(\, \psi, \xi, \theta, z\,)$ for a typical dataset of 512x512 spatial extent and hundreds of temporal frames within hours on modern computer GPU.

2.3 Results

In this section I fit the proposed model to real calcium imaging recordings, and show that it is capable of recovering gain variations within the field of view, and standardising the observation process across different fields of view, by estimating the parameters of the microscopy-inspired observation model.

2.3.1 Datasets and model initialisation

In order to show the general applicability of the method, as well as to produce easily verifiable and reproducible results, I chose to use the dataset openly published as part of the Neurofinder[8] project. As part of an effort for establishing a set of calcium imaging related data analysis challenges, recordings were submitted from a number of established labs. They used two-photon microscopy with undisclosed settings to image various areas of the mouse brain in vivo during a behavioural task[9]. In all instances the mice were awake and head-fixed, and the fluorescent calcium sensor has been a variant of GCaMP6. See table 2.1 for further details of the individual datasets.

Code	Contributors	Brain region	Data Size (XxYxT)	Pixel size	Frame frequency
00.00	Simon Peron (Svoboda Lab)	vS1	512x512x3024	0.87μm	7 Hz
01.00	Adam Packer, Lloyd Russell (Hausser Lab)	V1	512x512x2250	1.25μm	7.5 Hz
02.00	Nicholas Sofroniew (Svoboda Lab)	vS1	512x512x8000	0.87μm	8 Hz
03.00	Jeff Zaremba (Losonczy Lab)	dHPC CA1	512x512x3024	0.59μm	7.5 Hz
04.00	Matthias Minderer (Harvey Lab)	Hindlimb S1	512x512x3000	0.87μm	6.75 Hz

Table 2.1: Basic characterisation of Neurofinder datasets

Before describing the model fits, I first carry out a more extensive characterisation of dataset 00.00, that explains the statistics I compute for each dataset, summarised in table 2.2. These statistics are part of the heuristic data exclusion step, and are used to initialise the model for fitting. I only used the first 500 frames of each dataset to fit the model; this both demonstrates data efficiency and reduces run times. The statistics computed here will be reflective of only those frames as well. The remainder of the frames were held out, and served as a validation set for investigating the generality of results.

Figure 2.4 shows the distribution of observed intensities over the whole imaging area and recording time. The distinct characteristics include the presence of a pedestal peak, which represents time windows with no photoelectrons observed. The minor fluctuations around the pedestal are solely due to the electronic noise on, and after the photomultiplier anode. The photoelectric

[8] URL: `https://github.com/codeneuro/neurofinder`

[9] Although a small note of the task types is provided, the associated stimuli, or behavioural data is not published alongside the calcium imaging recordings.

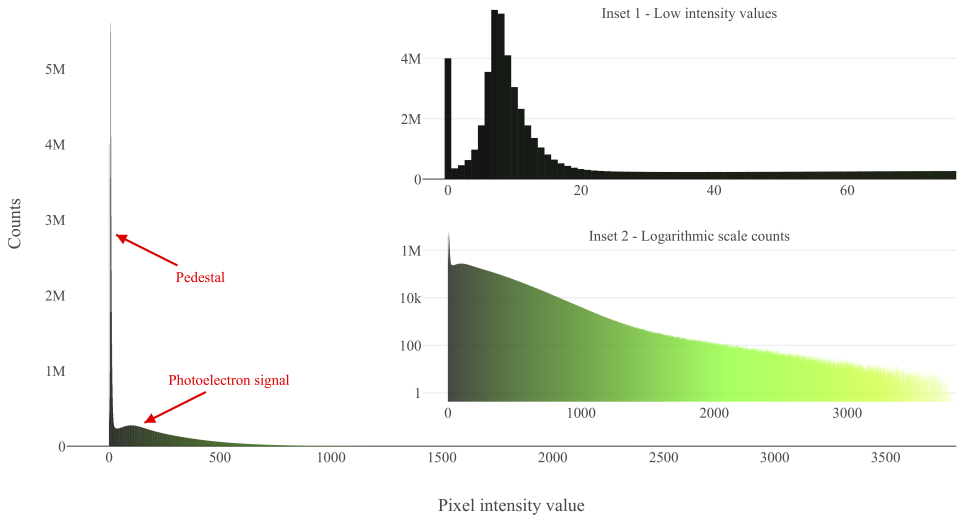
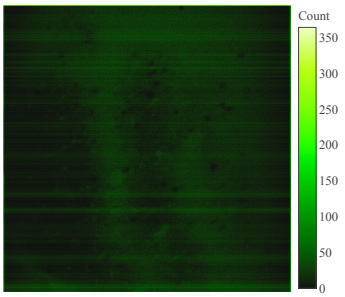


Figure 2.4: Pixel intensity histogram for the first 500 frames of dataset 00.00. The insets provide different visualisations of the same data. The separation of pedestal and signal pixels are clearly visible in the main figure. Inset 1 points out the separation of threshold and pedestal, whereas inset 2 shows the presence of high intensity observations. The colors of the individual bars were set to match those of image visualisations and to draw attention to the changes in axis ranges, but do not convey additional information.

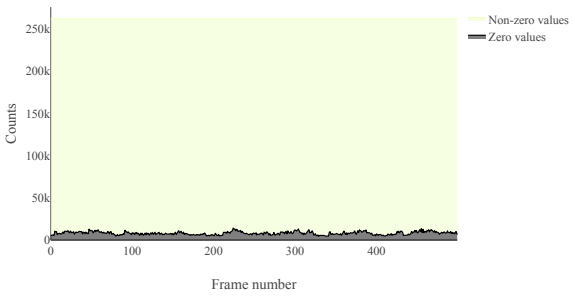
signal peak is much wider due to the large noise in the amplification process itself, as well as the fact that it includes additive signals from multiple photons within the same gating time window. The third characteristic is the peak exactly at 0, which is caused by the ADC cutoff – every signal below a certain threshold results in a 0 value. The excessive presence of 0s in most datasets is not only caused by thresholded real signals, however, these zeros also contain missing data. In figure 2.5 I display typical examples of zero observations, and discuss results from my proposed method for identifying missing data (described in equation 2.54). In some other recordings, one may need to worry about saturated pixels as well, that appear as an overpopulation of the histogram's maximum-valued bin.

The next step is to select the locations of background pixels in the dataset, that are expected to act as time-homogeneous Poisson source of photons, whose intensity depends only on their location. As described in section 2.2.4, we may identify neural structures via the shared signal in their multi-pixel extent, as evidenced by the local cross-correlation measure (figure 2.6). Rejecting pixels with high local cross-correlation results in the identification of the desired background pixels. As the time courses of individual Poisson sources recorded via extremely noisy machinery is impossible to identify without a model, the assumption of homogeneity can only be examined after the fact, once the model has been fit, which is why I call this background selection a heuristic procedure. In some other datasets, the presence of temporally structured low-dimensional signal is clear, which violates

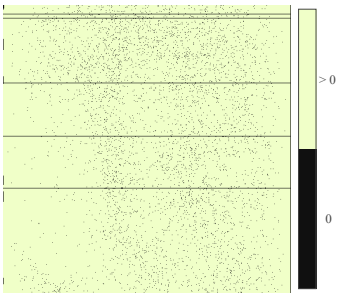
Figure 2.5: Identifying missing values and true observations.
(a-b) showcase the raw data over space and time, (c-d) provide examples and motivate the missing data identification described in equation 2.54. (e-f) demonstrate the efficacy of the identification and the effects of flagging the missing data.



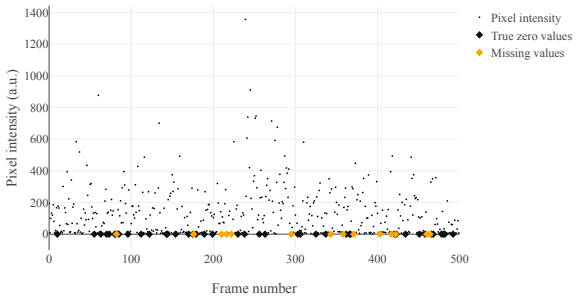
(a) Total number of zero values at each pixel, over 500 frames. Note the high counts at the edges, and the x-direction streaks.



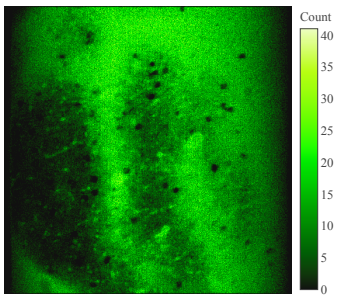
(b) The total number of zero values in each frame. Note the apparent high number, and the large fluctuations.



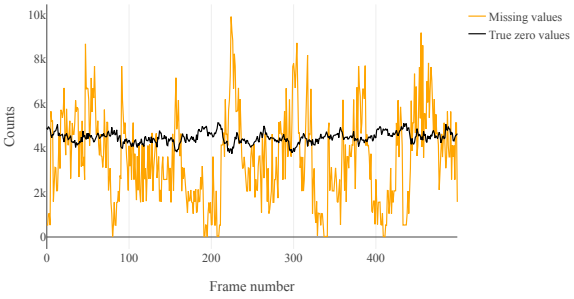
(c) Typical example frame, chosen as the one with the median number of zero observations (frame 44, 7959 zeros). Note the completely missing line-scans, as well as the missing values in the first and last columns.



(d) Example pixel, chosen as the one with the highest number of true zero observations (pixel 307x-61y, 55 total zeros). Without neighbourhood information, it would be impossible to tell apart true 0s from missing data.



(e) Number of true zero observations at each pixel,after removing all zeros flagged as missing data.



(f) Number of true zero and missing data observations on each frame. Note, that the number of true zeros per frame has much smaller fluctuation, more consistent with sampling variance.

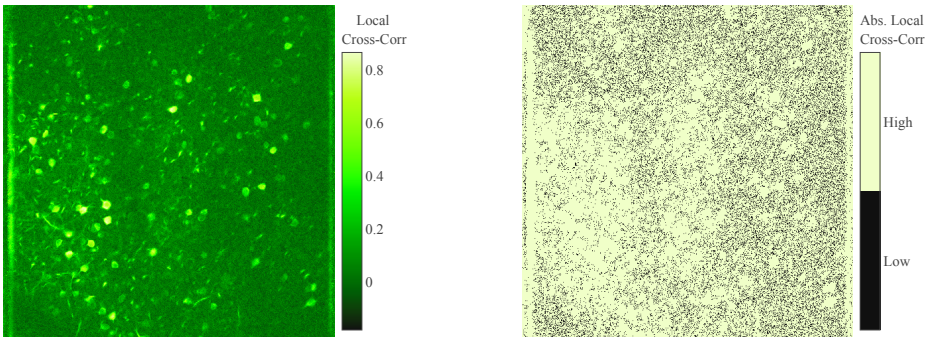


Figure 2.6: (Left) Local cross-correlation computed according to equation 2.53. This measure is robustly high for active neural structures*, and thus may be used as a means to identify background pixels.
(Right) Thresholded absolute signal – at 0.014 as the threshold, chosen as the 0.15 quantile of absolute local cross-correlations over all pixels. The black pixels constitute our accepted background locations, tiling the entire field of view.
*High values can also result from increased spatial correlation in illumination at the left and right edges, caused by the resonant scanning mirror being in the less linear regime of its sinusoidal curve. Further, our imputation process with local spatiotemporal mean also increases this measure, meaning we are more likely to reject locations with missing data. Lastly, neural structures with high mean activity may contain pixels with low cross-correlation, due to saturation – these are rejected by detecting this saturation.

the temporal homogeneity assumption. This is often a behaviour-induced side-effect, sometimes an undesired one of physical brain movement relative to the microscope, or a desired one of increased brain activity at certain experimental triggers, which affects the background due to the imperfect optical sectioning and the presence of fine neuropil. As described earlier, such strong global signals should be checked for, and removed via SVD – or other matrix decomposition methods – before background selection or model fitting takes place; neither of those variations is what we model, or want to correct the data for.

Finally, we wish to get simple estimates of certain model parameters, that we may use to initialise the model fits. A common method of gain estimation, given Poisson inputs is to chart the variance against the mean for each observation. This Fano factor should be 1 for the input Poisson process, therefore any other slope must be caused by the photomultiplier and subsequent electronics. Fit results for dataset 00.00 are shown in figure 2.7 and reported for each dataset in table 2.2.

2.3.2 Model fits and dataset correction

After carrying out the model fits, we need to examine how well it solved the two somewhat distinct problems: Firstly, we need to ensure that we estimated the parametric model of the photomultiplier well. Inverting the model enables us to transform the recorded data into estimates of the fluorescent

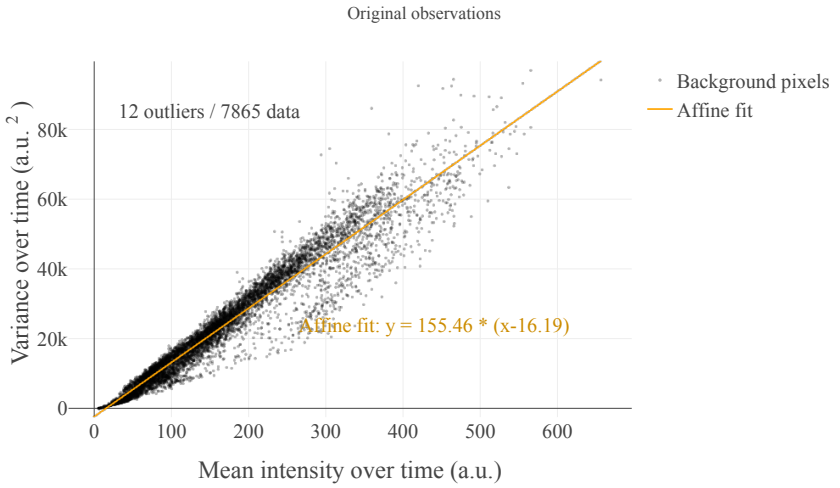


Figure 2.7: Estimation of the photomultiplier and electronics gain and offset using an affine least squares fit to observed variance against mean of each background pixel, excluding missing data. On the scatter plot, only a random subset of the background pixel statistics are shown to reduce figure size, but the affine fit was carried out using all 39322 selected pixels.

Code	Missing%	# Training locations	Pedestal Peak±width	Linear offset	Linear gain
00.00	1.346%	39322	7.59 ± 8.27	16.19	155.46
01.00	1.036%	39322	135.11 ± 207.78	1137.17	1524.11
02.00	32.15%	39322	0.63 ± 12.84	35.27	331.94
03.00	0.112%	36603	34.20 ± 223.27	-246.82	1024.69
04.00	6.48%	39322	2.12 ± 9.06	19.29	27.89

Table 2.2: Derived statistics of the datasets used

optical signal, which change linearly with respect to the calcium reporter signal, as desired. Secondly, we aimed to learn a useful description of the spatial gain non-uniformity, which can then be applied as as a divisive correction to the optical signal. The resulting corrected dataset should then have a spatially uniform background signal level. Importantly, this correction should remove differences not only within a single field of view, but even across datasets from different microscopes, brain regions, or animals; ultimately making subsequent data analysis methods more standardised, ubiquitously applicable, and the scientific conclusions drawn more robust.

To show the extent to which these two goals are achieved by the proposed model – as well as the necessary fitting process – I fitted the model to the first 500 frames of each dataset, and evaluated the correction on frames 1001-2000, to minimise the effects of temporal noise correlation. First, I describe the resulting model fit, and the following correction process in detail for dataset 00.00. Afterwards, in table 2.3, I show the fitted likelihood parameter values for all datasets, as well as the

effect of the corresponding inversion of the photomultiplier likelihood in table 2.4. Finally, I show a set of figures for each dataset, that showcase the effects of the grey level to photon flux conversion, display the learned spatial gain function, apply the correction and discuss its results in figures 2.8 to 2.12.

The correction itself ends up being relatively simple due to using maximum a posteriori estimates of both the likelihood and the the spatial gain non-uniformity, but keep in mind that during learning we propagate the full noise distributions through the whole model, leading to a robust fitting procedure. To illustrate this, figure 2.8a shows that the conditional probability $p(\nu \mid Y(\mathbf{x},t))$ over the number of photons incident at the photomultiplier, as a function of the grey level (see equation 2.45 for details of the likelihood model). Given this discrete distribution over photon numbers, we find the maximum a posteriori estimate of the rate of the Poisson distribution that generated the photons, $\nu \sim \text{Poisson}(\lambda)$. This λ rate I will now call 'photon flux', to indicate it is not a discrete photon count, but rather our rate estimate of the process that generated the photons $\nu(\mathbf{x},t)$, and ultimately the observation $Y(\mathbf{x},t)$. This grey level to photon flux estimate is shown in figure 2.8b, where the errors were assumed additive Gaussian in the output space (representing the amplification and electrical noise), and were also propagated through a MAP inverse of the likelihood.

Code	Pedestal $\pm$ Pedestal Noise std	Under-amplification probability * Amplitude	Gain per photon $\pm$ Noise std	Saturation grey value (photon flux)
00.00	8.67 ± 4.87	$0.30 * 51.56$	94.89 ± 53.23	3822 (40.19)
01.00	137.30 ± 64.44	$0.30 * 142.00$	699.02 ± 561.91	8191 (11.52)
02.00	2.77 ± 9.64	$0.32 * 91.00$	180.45 ± 85.82	5872 (32.53)
03.00	36.45 ± 69.12	$0.05 * 128.24$	777.39 ± 353.17	8191 (10.49)
04.00	4.13 ± 7.37	$0.03 * 22.26$	36.11 ± 10.11	6418 (177.64)

Table 2.3: Learned likelihood parameters for underamplified Poisson likelihood

Next, I investigated the effect of the resulting non-linear transformation on the data, mainly whether or not the resulting process became more Poisson-like, having a Fano factor of 1 – despite being rate estimate rather than actual photon count ones. I show the mean-variance plots for the original and the transformed data in figure 2.8c-d, and also show the parameters of the affine fits in table 2.4, which shows that we can achieve reasonable standardisation across multiple datasets, that were created in different labs by different equipments and settings.

Lastly, I show the need for spatial gain normalisation within a single field of view, by showing the mean image in figure 2.8e, and the mean photon flux image in figure 2.8f. The latter already reveals darker areas better, by learning the strong non-linearity present near the photomultiplier pedestal. These low light levels are typical in biological scanning two-photon imaging application, and thus most datasets are affected by this non-linearity. However, there usually still remains a lot of gain variation present across the field of view, the strongest of which is often induced by non-homogeneity of the scattering tissue (vasculature). This is well illustrated in the learned spatial gain non-uniformity of dataset 00.00, shown in figure 2.8g. An other important effect is the circularly symmetric illumination and collection fall-off away from the optical axes, towards the edges of the image, which is better shown by figures 2.9 to 2.11g. Finally, further strong non-uniformity is often aligned with the scanning direction, shown in figure 2.12e-g. We wish to design priors that are capable of identifying non-uniformities that we do wish to correct (such microscope artefacts or vasculature shadowing), without inadvertently explaining away features of the data that we did not want to. As discussed earlier in section 2.2.2, our main computational tools for prior design for Gaussian Processes include the choice of kernel type, inducing symmetries and setting strong hyperpriors for spatial lengthscales.

Code	Orig Offset	Orig Gain	Photon offset	Photon Gain
00.00	16.19	155.46	0.00	1.35
01.00	137.17	1524.11	-0.29	1.49
02.00	35.27	331.94	0.16	1.62
03.00	-246.82	1024.69	-0.47	1.14
04.00	19.29	27.89	0.26	0.71

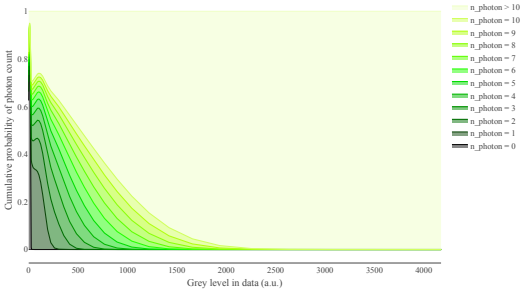
Table 2.4: Original and likelihood-corrected (photon) gain and offset estimates

The mean images of the spatially gain corrected and photon flux transformed datasets are shown in figures 2.8 to 2.12h, and they demonstrate the strengths and potential weaknesses of the proposed method. Note that I applied the exact same hyperparameter settings, initialisation, and priors to all five vastly different datasets. The too long spatial lengthscales used were unable to perfectly capture the strong edge effect in dataset 04.00, resulting in overamplification near the scan-aligned edges, but otherwise the results are fairly uniform.

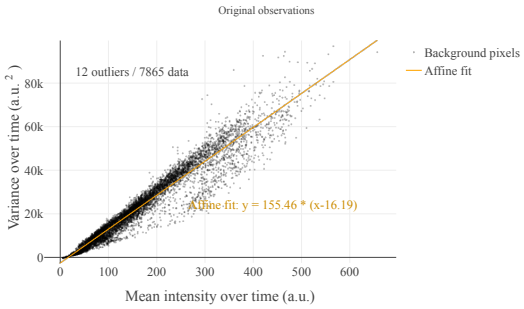
To more precisely describe the effects of the spatial gain normalisation, I also evaluated it at the background pixel locations, on the held out time frames. Furthermore, as the single-observation photon-flux transform induces a difficult-to-understand noise distribution, I instead estimated a single $\lambda(\mathbf{x})$ rate per background location, calculating the maximum a posteriori estimate $\hat{\lambda}(\mathbf{x})$ given all validation frames, $Y(\mathbf{x}, t \in [1001, 2000])$. This is consistent with our original modeling assump-

tion, that the only variation in background observations are due to Poisson and additive noise, and the signal is generated at a fixed rate, $\lambda(\mathbf{x})$. The histogram of the estimated $\hat{\lambda}(\mathbf{x})$ values over all $\mathbf{x}$ background locations, is shown in figure 2.8i. To illustrate the effects of the spatial gain correction, I also overlay the histogram of the gain-corrected $\hat{\lambda}(\mathbf{x})/G(\mathbf{x})$ values, where I matched the mean of the histograms to enable direct comparison. We can see that in all datasets the spatial gain correction reduces the spread of the histogram, showing that the spatially gain-corrected background is indeed much more uniform.

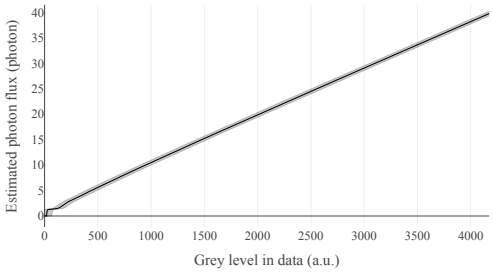
Figure 2.8: Evaluating model-based data correction for dataset 00.00.
(a) Conditional distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level, transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data, excluding missing data.
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image, excluding missing data.
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all validation frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matched histograms are shown.



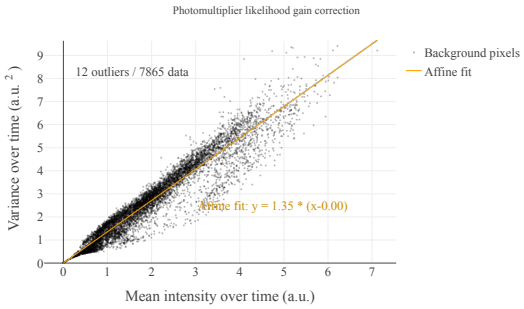
(a) Conditional distribution over photons at grey levels.



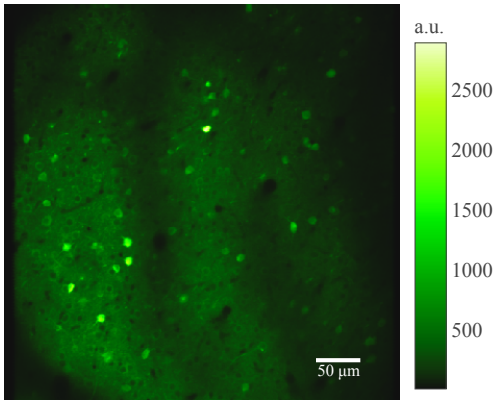
(c) Mean-variance plot in original data, per background pixel.



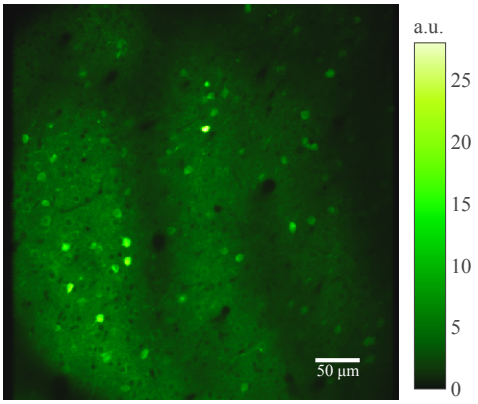
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



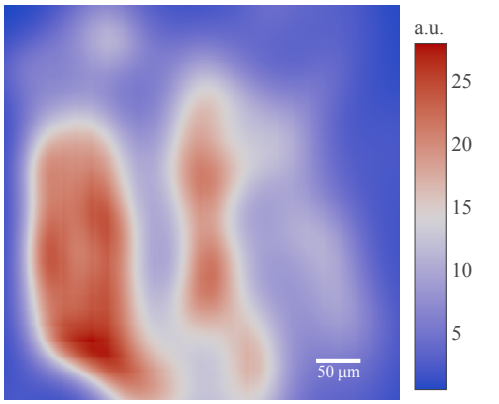
(d) Mean-variance plot of photon flux estimates, per background pixel.



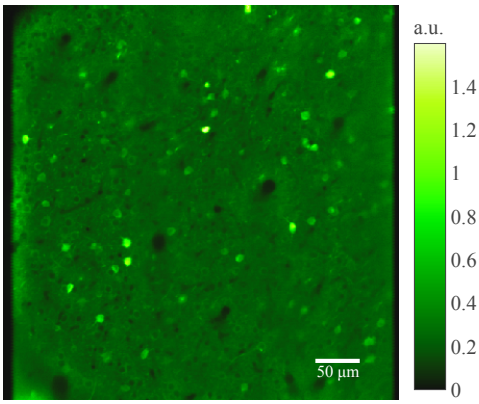
(e) Original mean image over frames, excluding missing data.



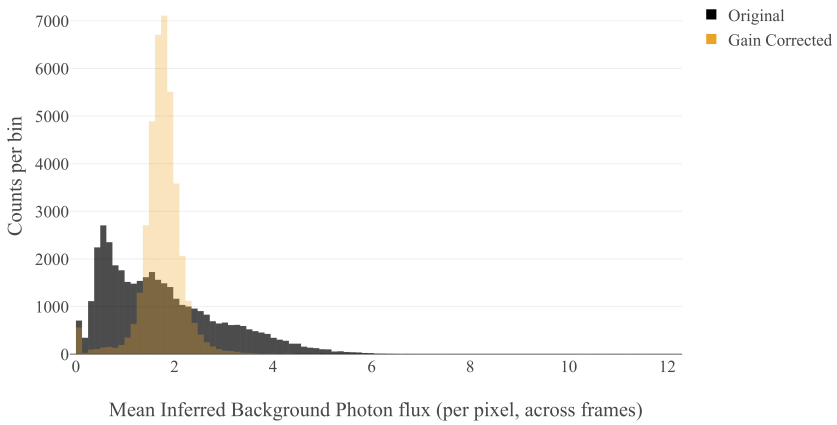
(f) Temporal mean after transformation into photon flux estimates.



(g) Learned spatial gain non-uniformity.

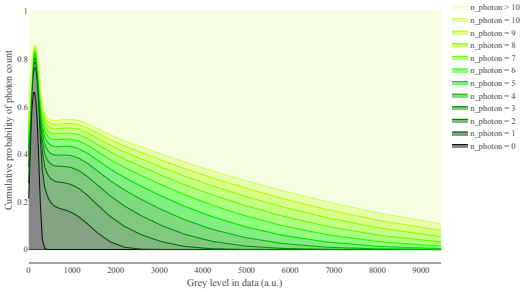


(h) Temporal mean after both likelihood and spatial gain correction.

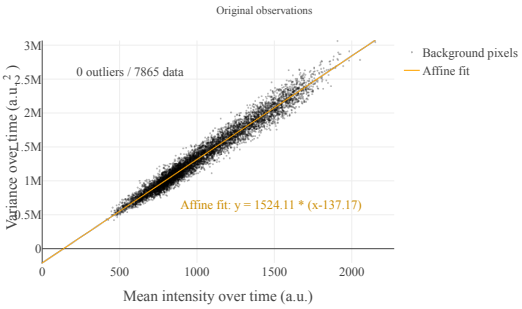


(i) Mean-matched histograms of original and corrected pixel-wise Poisson intensity, showing the effect of spatial gain correction.

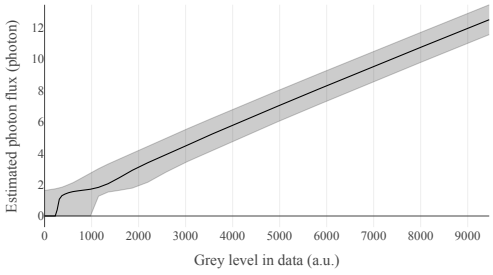
Figure 2.9: Evaluating model-based data correction for dataset 01.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level, transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data, excluding missing data.
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image, excluding missing data.
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all validation frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matched histograms are shown.



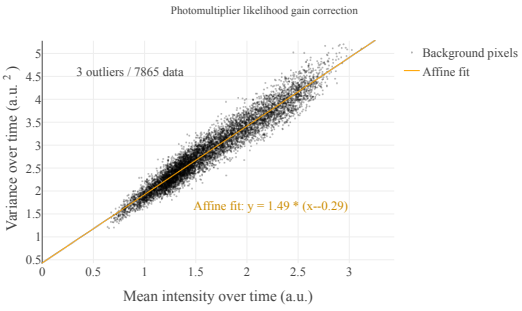
(a) Marginal distribution over photons at grey levels.



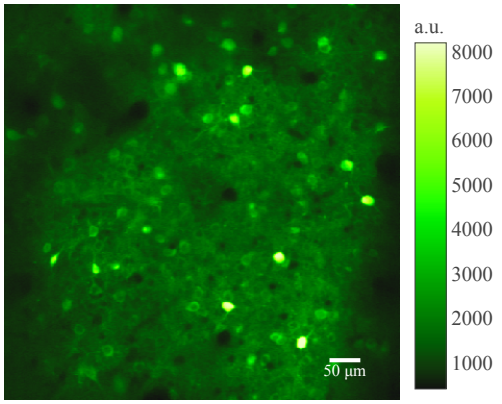
(c) Mean-variance plot in original data, per background pixel.



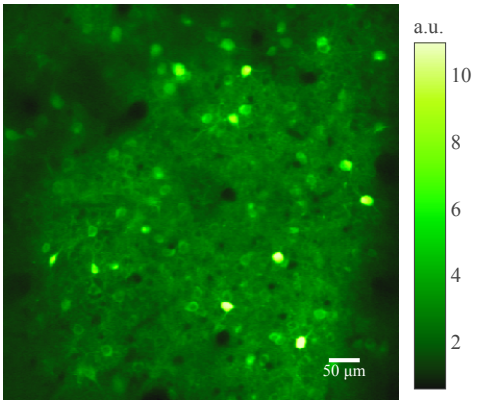
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



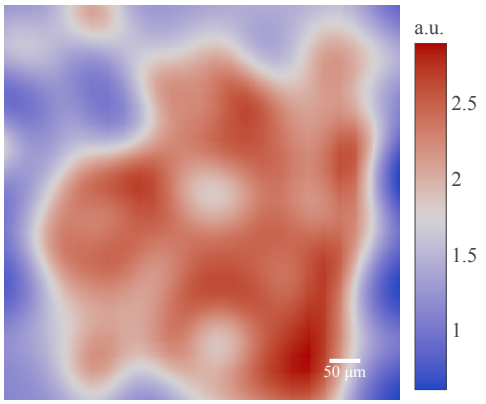
(d) Mean-variance plot of photon flux estimates, per background pixel.



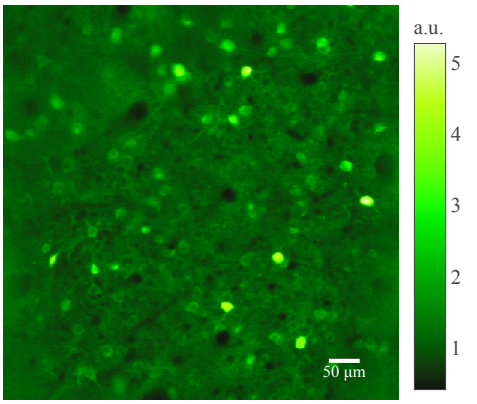
(e) Original mean image over frames, excluding missing data.



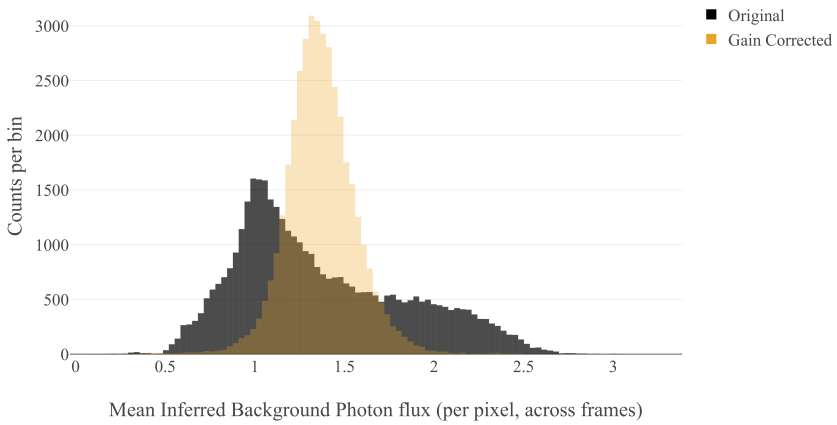
(f) Temporal mean after transformation into photon flux estimates.



(g) Learned spatial gain non-uniformity.

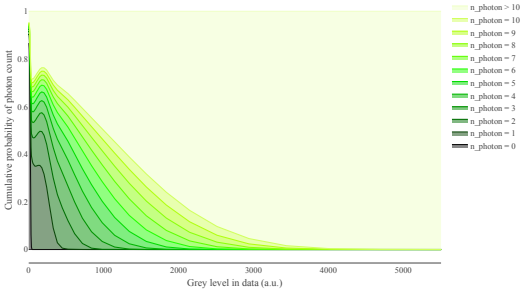


(h) Temporal mean after both likelihood and spatial gain correction.

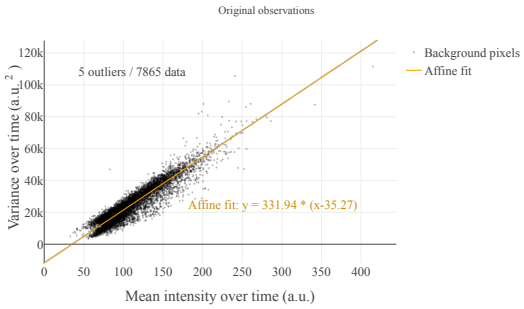


(i) Mean-matched histograms of original and corrected pixel-wise Poisson intensity, showing the effect of spatial gain correction.

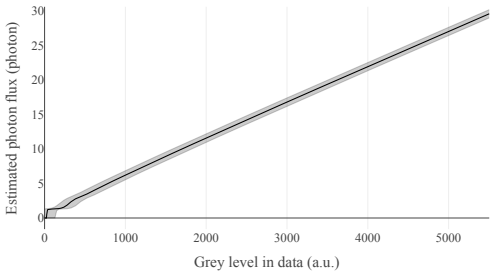
Figure 2.10: Evaluating model-based data correction for dataset 02.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level, transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data, excluding missing data.
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image, excluding missing data.
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all validation frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matched histograms are shown.



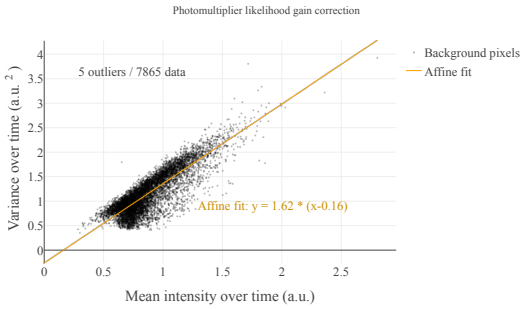
(a) Marginal distribution over photons at grey levels.



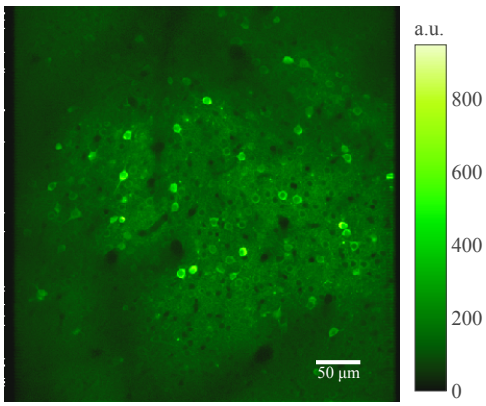
(c) Mean-variance plot in original data, per background pixel.



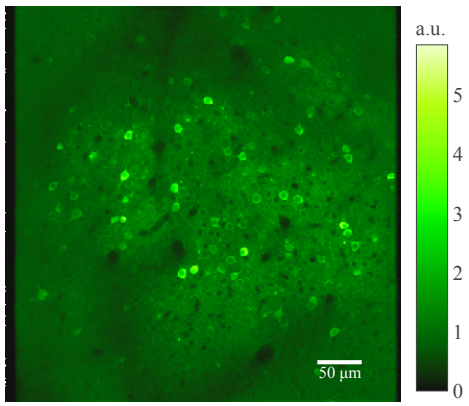
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



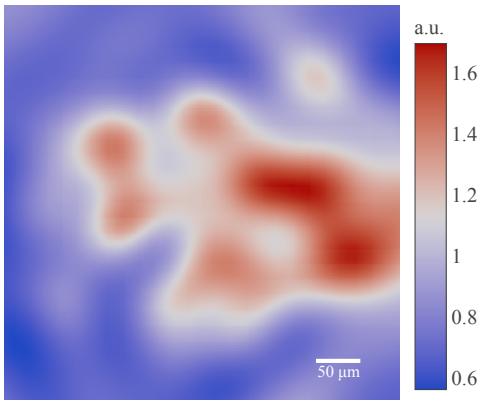
(d) Mean-variance plot of photon flux estimates, per background pixel.



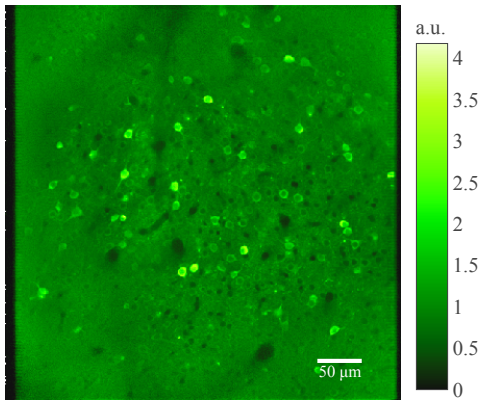
(e) Original mean image over frames, excluding missing data.



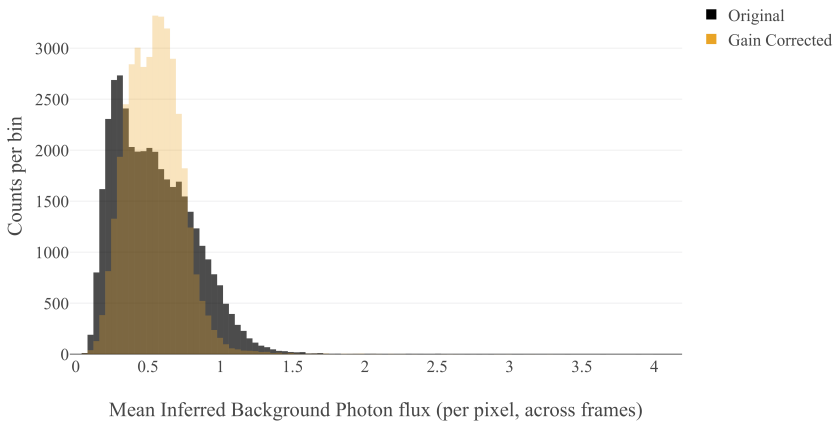
(f) Temporal mean after transformation into photon flux estimates.



(g) Learned spatial gain non-uniformity.

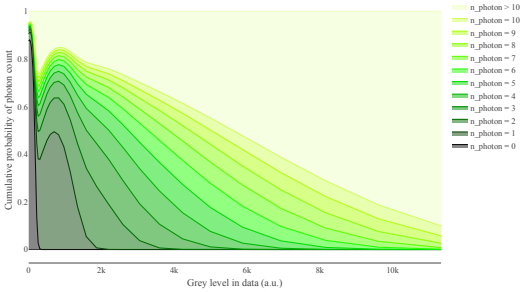


(h) Temporal mean after both likelihood and spatial gain correction.

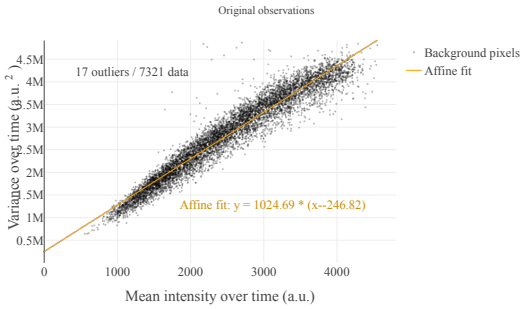


(i) Mean-matched histograms of original and corrected pixel-wise Poisson intensity, showing the effect of spatial gain correction.

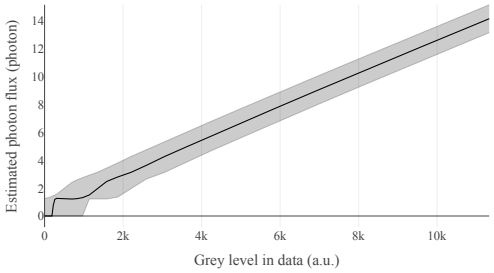
Figure 2.11: Evaluating model-based data correction for dataset 03.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level, transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data, excluding missing data.
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image, excluding missing data.
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all validation frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matched histograms are shown.



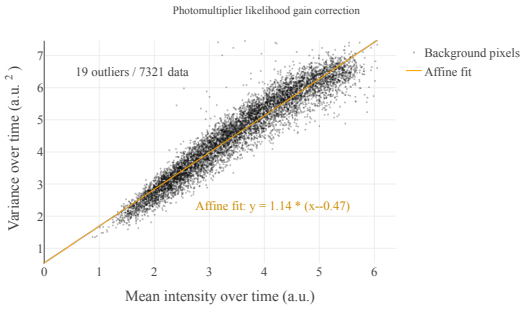
(a) Marginal distribution over photons at grey levels.



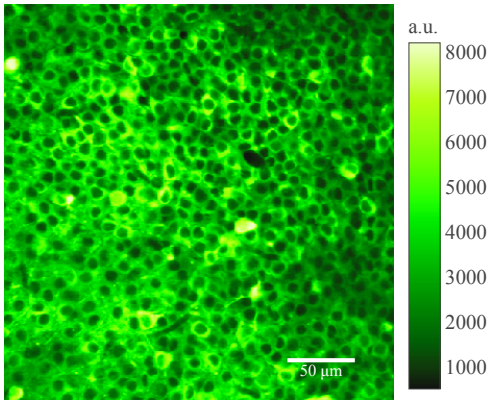
(c) Mean-variance plot in original data, per background pixel.



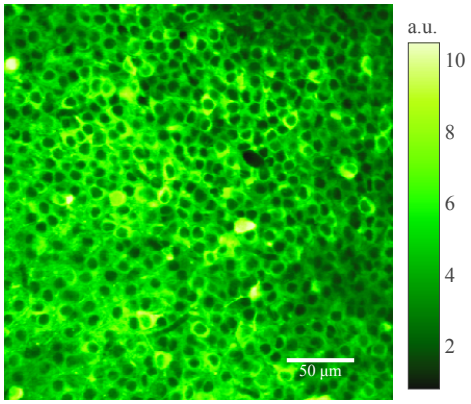
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



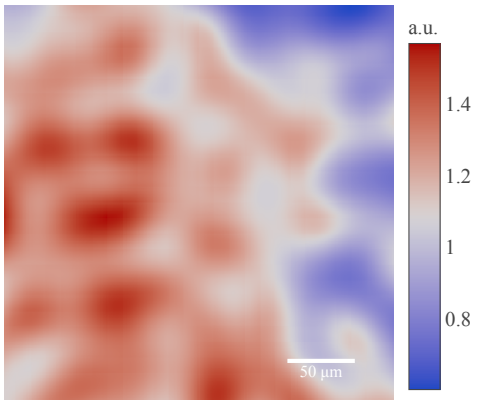
(d) Mean-variance plot of photon flux estimates, per background pixel.



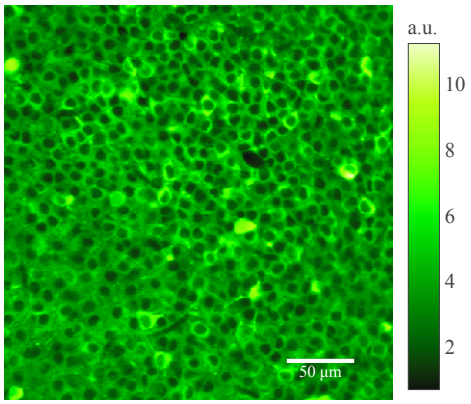
(e) Original mean image over frames, excluding missing data.



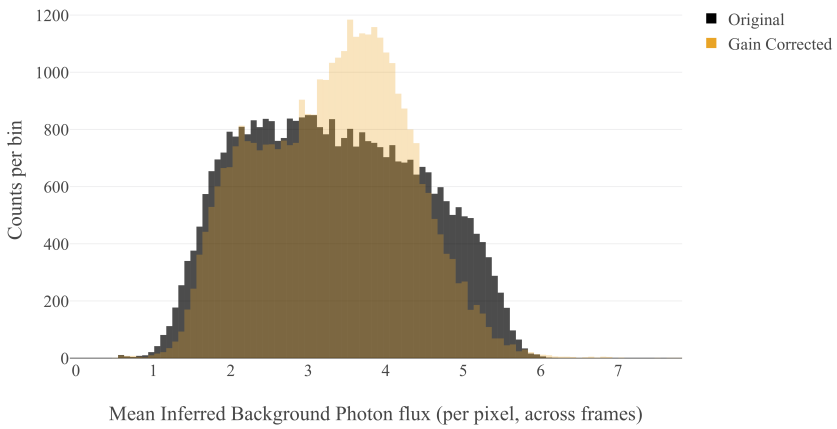
(f) Temporal mean after transformation into photon flux estimates.



(g) Learned spatial gain non-uniformity.

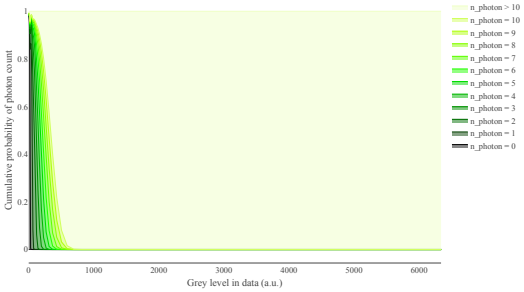


(h) Temporal mean after both likelihood and spatial gain correction.

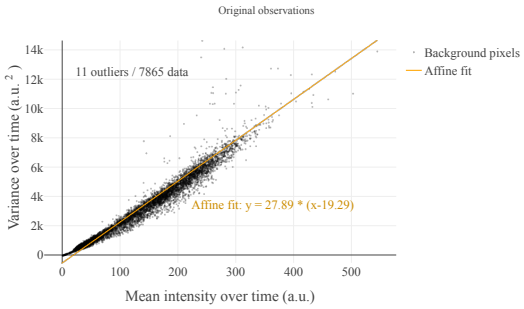


(i) Mean-matched histograms of original and corrected pixel-wise Poisson intensity, showing the effect of spatial gain correction.

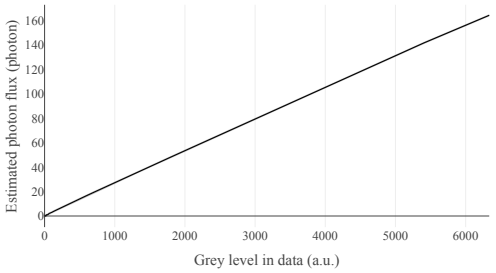
Figure 2.12: Evaluating model-based data correction for dataset 04.00.
(a) Marginal distribution over potential incident photon count at each observed output grey level.
(b) Maximum a posteriori estimate of the photon flux, given a single output observation. Error bars indicate one sigma output noise level, transformed through the estimated likelihood non-linearity.
(c) Mean-variance plot of original data, excluding missing data.
(d) Mean-variance plot after the likelihood transform, with the signals representing photon flux.
(e) Original mean image, excluding missing data.
(f) Mean over estimated photon flux per frame.
(g) Learned spatial gain non-uniformity.
(h) Mean over estimated spatially gain corrected photon flux per frame.
(i) Histogram of maximum likelihood photon flux per pixel, with the maximum likelihood photon flux computed across all validation frames. The spatial gain-correction is then applied to these stationary photon flux estimates, and the mean-matched histograms are shown.



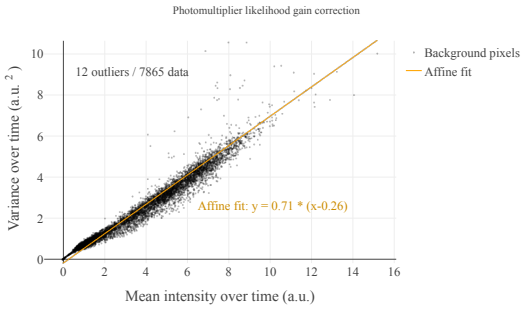
(a) Marginal distribution over photons at grey levels.



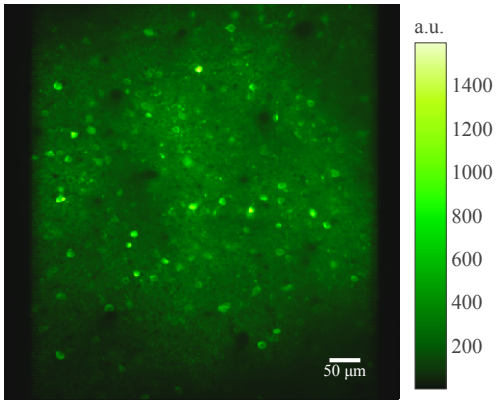
(c) Mean-variance plot in original data, per background pixel.



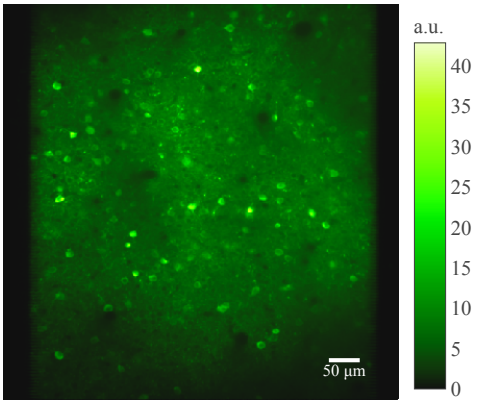
(b) Maximum a posteriori estimate of incident photon flux at a single output observation.



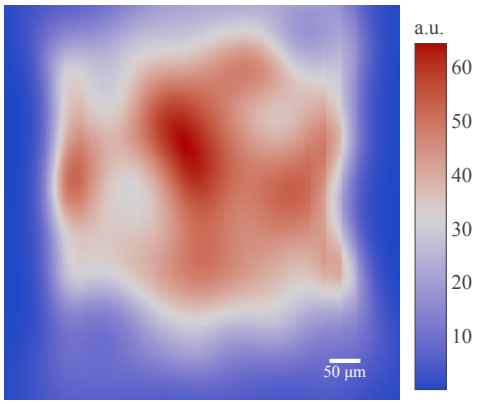
(d) Mean-variance plot of photon flux estimates, per background pixel.



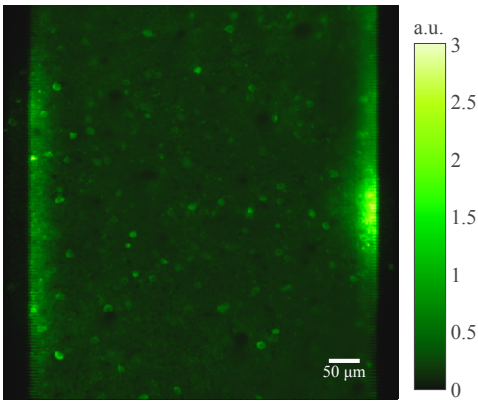
(e) Original mean image over frames, excluding missing data.



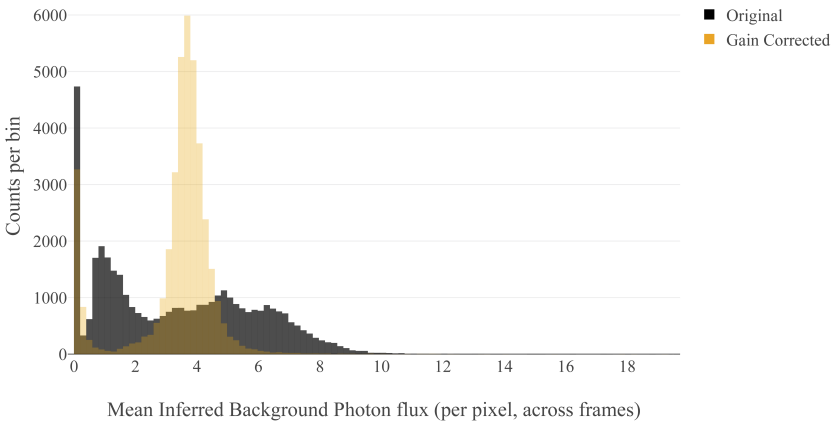
(f) Temporal mean after transformation into photon flux estimates.



(g) Learned spatial gain non-uniformity.



(h) Temporal mean after both likelihood and spatial gain correction.



(i) Mean-matched histograms of original and corrected pixel-wise Poisson intensity, showing the effect of spatial gain correction.

2.4 Discussion

In this chapter I described a simplified, but mechanistic generative model of datasets recorded by two-photon calcium imaging of neural tissue, and derived an algorithm capable of estimating the model parameters directly from the observed data. I aimed to construct the model in such a way, that the parameters directly correspond to particular biological or physical processes, which makes it possible to interpret and potentially validate the recovered estimates. More importantly, if some of these parameter values are measured by the experimenter, they can easily be incorporated into the algorithm as fixed values or strong priors, making the estimation of other, unknown parameters more precise and robust.

I then applied this algorithm to multiple recordings released publicly as part of the Neurofinder challenge, to showcase its versatility and applicability, as a general method of standardising data from different fields of view, brain regions or laboratories. The standardisation procedure has two distinct parts. Firstly, the algorithm estimates the parameters of the photomultiplier and subsequent electronics, that transform photon counts into the observed digital signal. These parameters can then be used to standardise the signal properties of different field of views, and to correct for non-linearities introduced by the observation process. Secondly, it attempts to correct for uneven gain within a single field of view. The spatial unevenness results from the multiplicative contribution from many processes, both biological and physical in nature. I do not try to distinguish their individual contributions, merely acknowledge the fact that this unevenness exists, and algorithmically estimate its extent. Given the learned models for each dataset, I then inverted them to get standardised datasets, in which observed values represent the maximum a posteriori instantaneous photon fluxes, and the background signal is indeed equalised across the field of view.

My preprocessing method performed well in equalising signal characteristics and gain across the datasets, but there are still a few outstanding issues. Firstly, to avoid the intractable modelling of signal variations over time, my fitting process assumes that all variations in the measurements of a single pixel are not reflective of changes in the signal, but are solely due to the contributions of the several noise processes in place: the inherent Poissonity, the photomultiplier noise and the subsequent electric digitalisation circuits. This is of course a very strong assumption, and is definitely not true for the regions of interest, in which we do expect to have a time-varying signal. Thus, to try and comply with this assumption, the preprocessing model is fit only to a heuristically selected subset of the pixels, which I deemed ‘background’, and treat them as having a single Poisson mean across all frames. Often even the background – thought to be composed of fine neuropil – has clear temporal variations which may need to be reduced, for example via matrix decomposition methods. An alternative, experimental way of providing training data for the preprocessing method is to include unstimulated period during experiments, in which synchronised background variations are often reduced. A second, smaller issue results from the approximate Gaussian Process nature of

our estimated uneven spatial gain function. Gaussian Processes, especially with the added KISS-GP kernel interpolation for their scalability, excel at describing spatially smooth functions, with uniform smoothness. The data we need to describe here, however, contain sharp changes at the edges of the scanning direction, as well as shadowing from vasculature across a range of diameters. For the examples shown in the chapter, I used a single setting of the approximations and kernel function hyperparameters, to show the method is quite generally applicable for vastly different datasets. However, it does also clearly fail in some situations, and that can mostly be contributed to the interpolation introduced by KISS. Although omitting the grid interpolation and using a large number of inducing points is possible, it would make the method significantly less scalable. This difficulty will be reduced with the ongoing development of hardware and algorithmic tools, and does not represent an inherent problem in the model itself.

The preprocessing method described here addressing a problem that has been identified before. Numerous image normalisation algorithms exist, often in the context of histogram equalisation and contrast enhancement (Bonnier and Simoncelli 2005; Dijk et al. 2007; Huang, Cheng, and Chiu 2013; Jin, Syu, and Jou 2017; Kim, Paik, and Kang 1998). My proposed solution is novel in the sense of addressing the multiple types of noise present in the recorded data, and utilising multiple samples of the same field of view. Others created image model based image standardisation, for example based on human vision (Lyu and Simoncelli 2008) or via detecting salient regions (Cheng et al. 2015), but these models and the resulting algorithms do not take into account the particular characteristics of two-photon calcium imaging. For neural data analysis in particular, similar normalisation is sometimes used less explicitly, by forming part of a complex model or data analysis pipeline. These complex models attempt to identify smaller regions of interest, and often treat the spatially non-uniform background signals as an additive element (e.g. Pachitariu et al. (2016)), but do not describe its generative model thoroughly. To my knowledge, my idea of representing the discrete nature of the recorded optical signal in calcium imaging microscopy, and discussing its effect on the data, is novel. The joint identification of the observation model parameters, the regions of interest and their activity time courses would be, of course, the ideal solution to avoiding the need for heuristic selection of background pixels, as done by my preprocessing; however, it would significantly increase model complexity and computational costs. I believe a potential implementation would involve approximating a hierarchical model, which includes preprocessing, ROI identification, and activity extraction. In this setting, the model parameters learned by preprocessing could also be influenced by subsequent algorithms. Such integrated hierarchical models could well be the most reproducible, interpretable and robust solution to carrying out data analysis on neural calcium imaging recordings, but for now they are somewhat out of reach.

Lastly, I want to emphasise the point, that parameterisation of the proposed model corresponds to measurable physical parameters. Although estimates of these are indeed recoverable from the data – as shown in this chapter – I think the more important message is that often these parameters are

accessible from the microscope control software, or could be measured via calibration slides prior to experiments. Instead of then publishing and distributing the recordings standalone – as currently is the practice – such metadata could be attached to aid subsequent analyses. Furthermore, as revealed by the preprocessing algorithm, many datasets contain very low photon counts, which are greatly amplified, resulting in high levels of noise. According to the photon flux estimates in this chapter, I believe many of the currently used microscopes would be able to record the same datasets in photon counting mode, completely eliminating the need for the non-linear inversion of photomultiplier amplification, making the spatially uneven gain – introduced by biological and optical elements prior to the photomultiplier – the only problem needing correction.

To summarise, in this chapter I introduced a preprocessing method that standardises signals across and within field of views, and showed that it works in a variety of settings, applicable to numerous already published datasets. Furthermore, I hope that the success of my described model highlighted characteristics of the generative model of the observed data, especially how various elements of the two-photon microscope transform the signal we originally intended to measure. These transformations can be partially recovered by my proposed model, but for moving towards future standardisation of calcium imaging, the transformations could be characterised during the data collection process, leading to more comparable recordings within fields of view, and across different brain regions, animals and laboratories.

3

Segmentation of neural cell bodies via Convolutional Higher Order Matching Pursuit (CHOMP)

A mathematician, however, who could back his prophecy with mathematical formulas and terminology, might be understood by no one and yet believed by everyone.

– Isaac Asimov, *Prelude to Foundation*

3.1 Introduction

A major goal at the interface of computational and experimental neuroscience is to gain direct and robust access to single neurons' activity, in the hopes of testing existing models of neural information processing, constraining the space of possible computational models, and potentially leading to the development of new ones that explain novel phenomena embedded in the recorded activities. Providing such access has long been a joint effort between developing novel experimental techniques, and the matching data analytic tools that can robustly extract single neurons' activity from the raw datasets provided. Rey, Pedreira, and Quiroga (2015) review an excellent example of such a collaboration, describing the developments in extracellular electrophysiological recordings as well as the corresponding analytic technique of spike sorting. Here I focus on a more recent, but now widespread experimental technique: two-photon imaging of genetically encoded fluorescent calcium reporters, or *calcium imaging* for short. Similarly to the spike sorting problem, calcium imaging has spawned its own set of data analysis challenges. A recent review by Stringer and Pachitariu (2019) discusses the current algorithmic needs, including motion correction and spike deconvolution; in this chapter, I address the question of localising single neurons and extracting their activity[1] over time,

[1] It should be noted that this activity is not necessarily a direct result of electrical activity, but instead reflects the concentration of calcium ions in the cytoplasm. Although there indeed is strong correlations between electrical activity, the amount of free calcium and the emitted fluorescent signal, cells are also known to strongly regulate calcium concentrations via in-

assuming the datasets have already been sufficiently motion corrected, and potentially the various non-uniformities have been mitigated, as proposed in chapter 2.

This process is generally called the identification of regions of interest (ROIs). A standard procedure that converts a video of a field of view into activity estimates of individual neurons, is the following: small binary masks are placed onto the field of view by a human marker, each reflecting the spatial extent of a neuron; pixel values within these ROIs are then summed to reflect the neural activity. There are two main concerns about this procedure: a.) observed pixel values within ROIs often contain signal from other neurons, or out-of-focus fluorescence due to the imperfect optical sectioning of the microscope; and b.) human markers processing datasets have various learned biases, and can also be strongly influenced by changing visualisations – the receiver operating characteristic curve of manual segmentation across multiple human markers is shown by Pachitariu et al. (2013). These issues led to the ongoing development of various algorithms, that attempt to automatically decompose the observed fluorescence into sums of signals generated by individual – potentially overlapping – neurons, taking into account fine neuropil-induced background or other contaminations; these are called 'demixing' algorithms in the field. The need for widely applicable algorithms for extracting ROIs from calcium imaging recordings is well illustrated by the existence and popularity of the Neurofinder[2] challenge. This is a collaborative effort from multiple experimental laboratories; each lab made calcium imaging datasets publicly available, and provided a segmentation of ROIs for each field of view[3]. The challenge provides an online interface for collecting implementation of proposed algorithms; it also validates and ranks the submissions automatically, using held out data. In the future of calcium imaging, such quick automated ROI extraction algorithms may play an even greater role: with the advancement of non- or partially-scanned imaging techniques (Schuck et al. 2018; Schultz et al. 2017), we are able to record from significantly more cells – provided we know and target their locations. This requires identifying ROIs rapidly and accurately as part of the closed experimental loop, giving an even greater role to robust methods that do so; especially considering that their results would now affect the data collected, rather than just the analyses of already fixed datasets.

In this chapter, I first provide an overview of existing methods that aim to identify single neurons; I discuss their main ideas, strengths and weaknesses in section 3.1.1. Next, I review matching pursuit methods in section 3.1.2, as they form the basis of a new algorithm I propose in section 3.2: Convolution Higher Order Matching Pursuit, or CHOMP for short, is a demixing algorithm tailored to identify the apparent characteristics of neural activity in calcium imaging. I presented the computational method at the 2016 Machine Learning in Signal Processing conference (Bohner and Sahani

tracellular stores. Therefore the insights gained from calcium imaging may differ from those obtained via direct electrical recordings, and should be interpreted with care.

[2] `http://neurofinder.codeneuro.org/`

[3] It is unknown how these segmentations were created. Known techniques include a.) segmentation by multiple human markers and approaching a consensual result; and b.) imaging a nuclear marker molecule alongside the calcium sensor, to aid cell localisation.

2016), and I showcased its neural application at the 2016 Gatsby Tri-Center Meeting. Here, in section 3.3, I apply it to both simulated data, and real recordings released by the Neurofinder challenge, then evaluate the results. Finally, in section 3.4, I discuss the merits of the proposed model, required extensions, and the future of algorithmic processing of neural calcium imaging recordings.

3.1.1 Overview of ROI extraction algorithms for calcium imaging

The algorithms presented here generally agree upon the fact that one needs to simultaneously estimate the spatial extent as well as the activity time courses of hundreds of neurons, in the face of significant noise and signal contamination. However, they also all acknowledge the fact that this problem is both computationally intractable, and the solution is not well regularised in the number of neurons inferred, making it a difficult problem. What these algorithms differ in, is the types of approximations they make to arrive at a tractable problem, and the types of constraints they put on the potential spatial extents, as well as the time course characteristics of individual neurons.

Spatio-temporal constraints. The constraints placed on spatio-temporal characteristics fit into three broad categories. Some methods place no constraints (such as PCA, SVD), but they tend to arrive at mixtures of sources rather than separating them. Placing spatial and temporal sparsity constraints leads to the family of independent component analysis methods (see Mukamel, Nimmerjahn, and Schnitzer (2009), based on Hyvärinen and Oja (2000)); whereas non-negativity constraints result in non-negative matrix factorisation based segmentation (Maruyama et al. 2014). These ideas help with separating signals from neurons with strongly differing activities; however, the resulting algorithms do not separate signals from correlated sources well – even spatially distant ones – as they do not take into account the physical shape of neurons. One may add more constraints to matrix factorisation that further limit the potential spatial filters' shapes (Pnevmatikakis et al. 2016). Alternatively, the use of templates – basis functions with limited spatial support – lead to convolutional methods; see Szlam, Kavukcuoglu, and LeCun (2010) for the original algorithm, and Pachitariu et al. (2013, 2016) for neural applications. Methods based on normalized cut or level set segmentation (Jianbo Shi and Malik 2000; Kaifosh et al. 2014; Reynolds et al. 2017; Smith and Häusser 2010), rely on finding spatially connected, but otherwise non-specifically shaped regions, that minimise the sum of within-cell and within-background discrepancies; it is worth noting that these methods often rely on heuristic background equalisation such as CLAHE in Kaifosh et al. (2014) or locally adaptive filters in Smith and Häusser (2010), and could thus benefit from the method described in chapter 2. Finally, a popular form of temporal constraint is to represent only certain features of the individual pixels' activity, such as the maximum, the mean (Pachitariu et al. 2013), or the cross-correlation with nearby pixels (Kaifosh et al. 2014; Reynolds et al. 2017; Smith and Häusser 2010).

Tractability. Such feature representations are not only useful to reduce noise or represent desirable neural characteristics, but in fact they significantly reduce the computational complexity of the methods. Indeed, separating the problems of spatial source localisation and temporal reconstruction of activities into two distinct steps is a common way of achieving tractability. As such, source localisation is often carried out on a reduced feature set; nevertheless it remains a difficult problem to simultaneously detect all sources at once, such that they all adhere to the spatio-temporal constraints. Level set methods are naturally intractable, and are typically solved by so-called active contours, which make an initial guess and then greedily evolve the individual regions, numerically solving partial differential equations; given good initial guesses and a simple correlation-based feature representation, these can then converge rapidly. In factorisation style algorithms, the source detection problem may be solved by minimising a cost function that combines reconstruction loss with a penalty for the number of sources used; essentially an L_0-regularised regression. This is a difficult cost function to optimise exactly, and therefore we turn to approximate inference methods. Two general types exist: one, the so-called L_1-relaxation results in solvable problems via replacing the L_0 cost with an L_1 one on reconstruction weights, and is applicable to both matrix factorisation methods (Pnevmatikakis et al. 2016) and template-based convolutional methods (via group lasso, introduced by Yuan and Lin (2006) and improved by Boyd et al. (2011). Another way of approximating the L_0-regularised regression problem is via greedy iterative reconstruction of sources that maximally decrease the reconstruction loss; these are called matching pursuit methods (Pachitariu et al. 2013, 2016). The algorithm I propose in this chapter – Convolutional Higher Order Matching Pursuit – belongs to this family of methods; thus I now review in more detail matching pursuit, as well as its various extensions that make it more applicable to the neural source detection problem.

3.1.2 Introduction to matching pursuit

The idea of L_0-regularised regression is that given a signal $\mathbf{y} \in \mathbb{R}^{|\mathcal{I}|}$ recorded at the set of $\mathcal{I}$ locations as the dependent variable, and multiple templates $\{\mathbf{b}^{\mathrm{k}} \in \mathbb{R}^{|\mathcal{I}|}\}_{\mathrm{k}=1}^{\mathrm{K}}$ as the independent variables, we can find the optimal sparse regression coefficients $\hat{\mathbf{x}} \in \mathbb{R}^{\mathrm{K}}$ – also called reconstruction coefficients, and activations – as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\|\mathbf{y} - \mathbf{x}^{\top}\mathbf{B}\|^2 + \lambda \|\mathbf{x}\|_0 \right), \tag{3.1}$$

where λ is the regularisation penalty, and $\mathbf{B}$ collects the $\mathbf{b}^{\mathrm{k}}$ templates as columns – the individual templates are also called basis functions or dictionary elements, and their collection can be referred to as 'bases', or 'dictionary'.

Matching pursuit (Mallat and Zhang 1993) provides a greedy approximate solution to this regression problem, by initialising $\mathbf{x} = \mathbf{0}$ and activating its x_{k} elements one by one: on each iteration, it first marginally solves for each non-active element x_{k}, then selects the one that maximally reduces

the residual reconstruction cost:

Algorithm 3.1 Matching Pursuit algorithm pseudocode

1: Let $\hat{\mathbf{x}}^0 = \mathbf{0}$ be the empty vector of basis coefficients, $j = 0$ the current iteration, and $\mathbf{B}$ the template matrix.
2: **repeat**
3: $j \leftarrow j+1$.
4: Calculate the current residual $\tilde{\mathbf{y}}^{\mathrm{j}} = \mathbf{y} - (\hat{\mathbf{x}}^{\mathrm{j}-1})^\top \mathbf{B}$.
5: For each unused k, solve for the best reconstruction of the residual via $x_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}$, and compute potential reconstruction cost change $\Delta\mathcal{C}^{\text{reconst}}_{\mathrm{jk}}$, if we were to use the k-th basis function.

$$\begin{aligned} \forall_{\mathrm{k}} \text{ where } \hat{x}^{\mathrm{j}-1}_{\mathrm{k}} &= 0 \\ x^{\mathrm{j}}_{\mathrm{k}} &= \arg\min_{\mathrm{x_k}} \|\tilde{\mathbf{y}}^{\mathrm{j}} - x_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}\| \\ \Delta\mathcal{C}^{\text{reconst}}_{\mathrm{jk}} &= \|\tilde{\mathbf{y}}^{\mathrm{j}} - x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}\|^2 - \|\tilde{\mathbf{y}}^{\mathrm{j}}\|^2 \\ &= (x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}})^\top (x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}}) - 2(x^{\mathrm{j}}_{\mathrm{k}}\mathbf{b}^{\mathrm{k}})^\top \tilde{\mathbf{y}}^{\mathrm{j}} \end{aligned} \tag{3.2}$$

6: Let $\hat{x}^{\mathrm{j}}_{\mathrm{k}} = \arg\min_{x^{\mathrm{j}}_{\mathrm{k}}} \Delta\mathcal{C}^{\text{reconst}}_{\mathrm{jk}}$ be the newly activated basis function coefficient on iteration j
7: **until** $\Delta\mathcal{C}^{\text{reconst}}_{\mathrm{jk}} + \lambda > 0$ for all k, and thus adding a source does not decrease the total cost $\mathcal{C}$.
8: Return $\hat{\mathbf{x}}^{\mathrm{j}-1}$ as the approximate solution to equation 3.1.

Although a rather general algorithm, this basic matching pursuit formulation is not immediately applicable to the neural source segmentation problem. Firstly, this formulation assumes we have a single sample $\mathbf{y}$ to reconstruct, whereas in neural applications multiple measurements are available over time: $\mathbf{y}(t)$, with $t \in [0, T]$. This issue may be addressed by either extending the templates $\mathbf{b}$, or altering both the coefficients $\mathbf{x}$ and the cost function $\mathcal{C}$ to account for the repeated samples. For large T, these solutions lead to expensive algorithms, and often a feature representation of the data is extracted instead: Let $\mathbf{Y} \in \mathbb{R}^{|\mathcal{I}| \times \mathrm{d_f}}$ be d_{f} features of the data, with the templates $\mathbf{b}$ being elements of this feature space, $\mathbf{b} \in \mathbb{R}^{|\mathcal{I}| \times \mathrm{d_f}}$, while still having a single activation coefficients per template, $\mathbf{x} \in \mathbb{R}^{\mathrm{K}}$.

The main difficulty of applying matching pursuit to neural calcium imaging comes from defining a suitable collection of templates, $\mathbf{B}$. This dictionary needs to be able to sufficiently represent the data (or features); moreover, to segment neurons, each activated $\mathbf{b}^{\mathrm{k}}$ template needs to correspond to exactly one neuron. However, in calcium imaging, the field of view contains a large, and underdetermined number of neurons, whose locations and shapes are also unknown. Therefore to represent every possible shape and location, and to infer the neural identities via the activations $\hat{\mathbf{x}}$ resulting from algorithm 3.1, we would need an extremely large number of templates, leading to an intractable algorithm. This is addressed by two successive extensions to the basic matching pursuit algorithm, *convolutional* (Szlam, Kavukcuoglu, and LeCun 2010) and *convolutional block matching*

pursuit (Pachitariu et al. 2013). The convolutional extension assumes that the size of the templates – now denoted $\mathcal{P}$ for patch size – is smaller than the observation space $\mathcal{I}$, and the full dictionary $\mathbf{B}$ is effectively created as a convolution of the small templates with all potential locations. The idea behind this extension is that we never need to represent the full $\mathbf{B}$ convolutional dictionary, but rather organise the reconstruction coefficients $\mathbf{x}$ such that $\mathbf{x} \in \mathbb{R}^{|\mathcal{I}|\times K}$, ie. we represent the basis function activations at every potential location[4]. The reconstruction in step 4 of algorithm 3.1 is now computed as $\sum_k \mathbf{b}_k * [\hat{\mathbf{x}}^{j-1}]_{\cdot k}$, and we need to accordingly adjust computing x_k^j and $\Delta C_{jk}^{reconst}$ in step 5; otherwise the algorithm remains the same. Block matching pursuit is a both mathematically and computationally straightforward addition, which changes the L_0 penalty term of equation 3.1 to $\|\sum_k \mathbf{x}_{\cdot k}\|_0$. Conceptually, however, it allows us to further reduce the dictionary size K by enabling us to use multiple basis activations at the same location without additional cost. This means that potential cell shapes at a location are not described by a single $\mathbf{b}^k$ template (requiring many templates to account for all possible shapes), but rather as a linear combination of the templates, $\sum_k \mathbf{x}_{\cdot k}\mathbf{b}^k$, capable of describing numerous shapes with only a few templates. The complete cost function of convolutional block matching pursuit is thus

$$\mathcal{C} = \left(\left\| \mathbf{y} - \sum_k \mathbf{b}_k * [\mathbf{x}]_{\cdot k} \right\|^2 + \lambda \left\| \sum_k \mathbf{x}_{\cdot k} \right\|_0 \right) \tag{3.3}$$

and the approximate matching pursuit solution can still be computed by algorithm 3.1, with small modifications to step 4 and 5.

My algorithm, Convolutional Higher Order Matching Pursuit – introduced next, in section 3.2.1 – uses this idea in essence, with a feature set $\mathbf{Y}$ specifically designed to match interesting neural characteristics, while enabling us to exploit the computational advantages of Convolutional Block Matching Pursuit. The requirement of a – now much simpler – dictionary of basis functions $\mathbf{B}$ as input still remains, and this I discuss in section 3.2.2, how one might initialise and update it via dictionary learning.

3.2 Convolutional Higher Order Matching Pursuit and dictionary learning for segmenting neurons

3.2.1 Convolutional Higher Order Matching Pursuit

The aim of Convolutional Higher Order Matching Pursuit (CHOMP) is to carry out sparse signal decomposition of multidimensional, repeated signal measurements, where the measured signal dimensions have some built-in structure; in the case of our main application – calcium imaging –

[4]It is worth mentioning that the set of potential source locations need not coincide with the observation locations $\mathcal{I}$ for convolutional matching pursuit to be applicable. To my knowledge this has not been derived before, and thus I introduce this novel possibility in section 3.2.

this would be pixels or voxels organised in a 2 or 3 dimensional grid. Therefore, from here onwards I will refer to this structure as measurement 'space' or 'spatial structure' and the individual measured dimensions as being 'locations', although the structure of measurements may well be of spatio-temporal nature, in which repeated measurements would be multiple time series.

In this section I first discuss the algorithm rather generally, without explicitly specifying the spatial structure. However, as our intention is to analyse data from calcium imaging – and thus evenly spaced grids as our measurement structure – I will describe in detail the computational simplifications that can be made to significantly speed up the algorithm for grid data.

Generative model

In order to understand the problem this algorithm solves, I first describe the generative model. This is based on the idea of measurements that are generated by a set of sources, which have fixed locations and small spatial extents, but varying activity across samples. To clarify the language used in this section: the maximum spatial extent of a source I call a 'patch', the apparent signal generated by a source on a single frame is a 'signal mode', assumed to be described by a linear combination of fixed basis functions (templates).

Each measurement at an observation location thus represents the superposition of spatially shifted samples of the apparent signal modes from multiple sources, with the shifts corresponding to the vector difference of the observation location and the source location. This measurement process is illustrated in figure 3.1. These measurements – corrupted by additive noise – are taken at each observation location repeatedly. The resulting samples are assumed to be independent from one-another[5].

Formally, let the data be generated by a set of S sources, $\mathsf{O} = \{O^{\mathrm{s}}\}_{\mathrm{s}=1}^{\mathrm{S}}$, located within a D-dimensional space. Each source has a location, $\mathbf{l}^{\mathrm{s}} \in \mathbb{R}^{\mathrm{D}}$, and generates a measurable signal in a finite extent D-dimensional patch around its location. This signal is described by a linear combination of K known basis functions, $\mathsf{B} = \{b^{\mathrm{k}}(\cdot)\}_{\mathrm{k}=1}^{\mathrm{K}}$, with activations $\mathbf{x} \in \mathbb{R}^{\mathrm{K}}$ for each basis. The bases, $b^{\mathrm{k}} : \mathbb{R}^{\mathrm{D}} \mapsto \mathbb{R}$, are each stationary functions with finite cuboidal support $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times \ldots \times \mathcal{P}_{\mathrm{D}}$, where $\mathcal{P}_{\mathrm{i}} = [-m_{\mathrm{i}}, m_{\mathrm{i}}]$ are closed intervals around zero; therefore $b^{\mathrm{k}}(\mathbf{l}') \neq 0, \iff \mathbf{l}' \in \mathcal{P}$, where $\mathbf{l}'$ often represents a vector difference, $\mathbf{l} - \mathbf{l}^{\mathrm{s}}$, essentially centring the basis function at a source location.

The source locations and their corresponding time-varying[6] activations at times $\mathcal{T}$, $\{\mathbf{x}^{\mathrm{s,t}}\}_{\mathrm{t}\in\mathcal{T}}$, are collected in $O^{\mathrm{s}} = \{\mathbf{l}^{\mathrm{s}}, \mathbf{x}^{\mathrm{s}}\}$. Let $\mathcal{T} = [1, \ldots, T]$ for simplicity. Our goal is to infer these source locations and activations, given T noisy measurements, $\{\mathbf{y}^{\mathrm{t}}\}_{\mathrm{t}=1}^{\mathrm{T}}$. We write $\mathbf{y} \in \mathbb{R}^{|\mathcal{I}|\times\mathrm{T}}$ for the entire collection of measurements, where $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 \times \ldots \times \mathcal{I}_{\mathrm{D}}$ is a set of D-tuples indexing discrete

[5] This is not generally the case in time series measurements, such as our calcium imaging videos. However, if the time-varying signal is stationary over time and sampled less frequently than its auto-correlation length, our independence assumption is fairly satisfied. We can achieve this by sufficiently downsampling high-frequency time-series measurements.

[6] From here onwards I will refer to the signals changing over samples as them having time courses, as they do in calcium imaging, although this need not generally be the case.

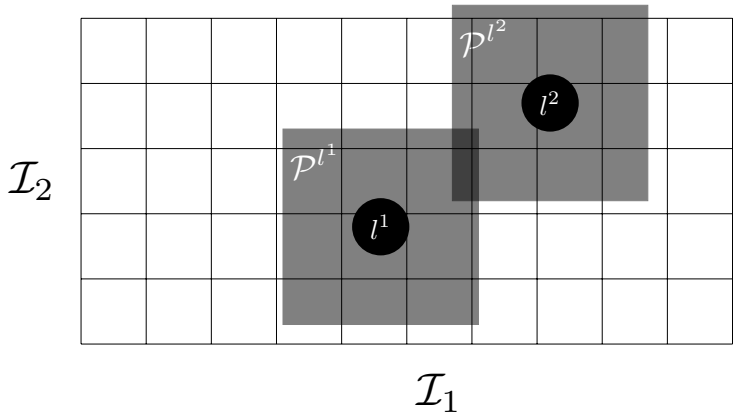


Figure 3.1: Source locations and patches embedded in a grid measurement space. The patches might overlap, or extend over the edges of the measurement space, as illustrated. Black circles indicate the source locations, the grey boxes show the patches that the individual sources affect and the dark grey area indicates overlap between the patches, where the signals from the two sources are added. The grid represents a measurement grid, and the observed data is the signal collected within each square, corrupted by additive noise.

measurement locations in the D-dimensional data space[7]. The generative model of our measured data $\mathbf{y}$, given the bases B and the sources O, is thus:

$$\begin{aligned} &\forall\, \mathbf{l} \in \mathcal{I}, \quad t \in \{1 \dots T\} \\ &\tilde{y}^{\,\mathbf{l},\mathrm{t}}(\mathsf{O},\mathsf{B}) = \left(\sum_{\mathrm{s}=1}^{\mathrm{S}} \sum_{\mathrm{k}=1}^{\mathrm{K}} b^{\mathrm{k}}(\mathbf{l}-\mathbf{l}^{\mathrm{s}}) x_{\mathrm{k}}^{\mathrm{s,t}} \right) \\ &\mathbf{y}^{\mathrm{t}} = \tilde{\mathbf{y}}^{\mathrm{t}}(\mathsf{O},\mathsf{B}) + \boldsymbol{\varepsilon}^{\mathrm{t}}, \end{aligned} \tag{3.4}$$

where $\boldsymbol{\varepsilon}^{\mathrm{t}} \sim \mathcal{N}(\mathbf{0},\ \mathbf{C} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|})$ is additive Gaussian noise, potentially correlated across measurement locations $\mathcal{I}$, but independent over samples t.

Cost function

Based on the generative model described, the inference problem can be described as two joint inferences: finding the source locations, and estimating the corresponding signal time courses at each location. An added complication is that the number of sources is unknown, and need to be estimated as part of the inference.

As discussed earlier in section 3.1.2, matching pursuit methods introduce a penalty for each additional source used in reconstructing the data. They thus find a balance between the reconstruction error and the number of objects used, by starting with 0 sources and maximum error, and adding

[7] Note that in experimental setups the measurement locations themselves would have an effective spatial extent within the space, rather than being point measurements. However, in current technical description, the effective spatial area of signal collection can be represented within the basis elements $b^{\mathrm{k}}(\cdot)$ themselves – effectively a convolution of the true signal mode with the measurement extent – and thus we can safely think of the measurements, as if collected at point-like locations.

sources iteratively, until the penalty for adding a new source would be larger than the reduction in reconstruction cost. Let our cost function thus be the sum of the reconstruction error and the cost of the used sources:

$$\mathcal{C} = \mathcal{C}^{\text{reconst}}(\mathbf{y}, \hat{\mathbf{y}}) + \mathcal{C}^{\text{sources}}(\hat{\mathsf{O}}). \tag{3.5}$$

Notice, that in this formulation the inference of the locations and activations of sources have to be done jointly each time we add a new source, making this a rather complicated, and often intractable problem. As discussed in section 3.1.1, this can be resolved by separating the inference into two steps, first finding source locations in a simpler feature representation of the full data, then reconstructing the full activity time courses given the source locations – both potentially tractable problems. The main task is now to select features that enable precise localisation, even considering the sparse neural activity, and the significant noise. Previous algorithms use the mean (Pachitariu et al. 2013), or the local cross-correlation (Pachitariu et al. 2016) as features, however they suffer from missing neurons with low activity, or lack the additive representation of signal from overlapping sources, respectively.

Convolutional Higher Order Matching Pursuit (CHOMP) extends and encompasses these earlier feature representations, and creates a flexible framework that takes into account the characteristics of neural activities recorded via calcium imaging. We expect the cytoplasmic calcium signal in active neurons to exhibit higher variance than background signals, as well as being more skewed towards higher values, and more kurtotic due to the sparse activity. CHOMP thus extracts these features from the provided samples, and finds source locations that display the expected neuron-like properties.

We also wish to retain the additivity of the individual signals for spatially overlapping sources, therefore our feature representation needs to operate in the space of cumulants. Furthermore, due to the non-pointlike nature of our sources, we need to retain not only the pointwise cumulants at each observation location, but also the joint cumulant tensors (such as the covariance matrix) for observations that are within the same patch $\mathcal{P}$, and thus could be jointly influenced by a single source. The unbiased, minimum-variance empirical cumulant estimator is the multivariate K-statistics, defined by Kendall, Stuart, and Ord (1987). In order to compute it, we first need to define the unnormalised, non-central vector moments of the observed data, as

$$\mathbf{S}_{\mathrm{r}} = \sum_{\mathrm{t}=1}^{\mathrm{T}} (\mathbf{y}^{\mathrm{t}})^{\otimes \mathrm{r}}, \tag{3.6}$$

where $(\cdot)^{\otimes \mathrm{r}}$ is the rth generalised (tensor) outer product $(\cdot) \otimes (\cdot) \otimes \ldots \otimes (\cdot)$. We use the first four K-statistics – see Kendall, Stuart, and Ord (1987) for higher-order expressions, and Di Nardo, Guarino, and Senato (2008) for a general discussion – for T samples:

$$
\begin{aligned}
\mathbf{Y}^1 &= \frac{\mathbf{S}_1}{T} \\
\mathbf{Y}^2 &= \frac{T\mathbf{S}_2 - \mathbf{S}_1^{\otimes 2}}{T(T-1)} \\
\mathbf{Y}^3 &= \frac{T^2\mathbf{S}_3 - 3T\mathbf{S}_2 \otimes \mathbf{S}_1 + 2\mathbf{S}_1^{\otimes 3}}{T(T-1)(T-2)} \\
\mathbf{Y}^4 &= \frac{T^2(T+1)\mathbf{S}_4 - 4T(T+1)\mathbf{S}_3 \otimes \mathbf{S}_1 - 3T(T-1)\mathbf{S}_2^{\otimes 2} + 12\mathbf{S}_2 \otimes \mathbf{S}_1^{\otimes 2} - 6\mathbf{S}_1^{\otimes 4}}{T(T-1)(T-2)(T-3)}.
\end{aligned}
\tag{3.7}
$$

Note that this retains the spatial structure of the original data, meaning if our observations are on a plane ($D = 2$), then the mean, $\mathbf{Y}^1$, will be a 2nd order tensor, the variance, $\mathbf{Y}^2$, is a 4th order one, and so on. Similarly for $D = 3$ dimensional datasets the relevant scaling is 3rd, 6th, etc. dimensional tensors. Although the formulas given above compute these tensors for the complete observation space, due to our generative model and small basis functions, we will only be able to reconstruct elements of these tensors that are within our interaction length, as defined by the patch size $\mathcal{P}$. Therefore, we do not need to compute or store most tensor elements, significantly reducing both computational and storage costs; specifically $\mathbf{Y}^{\mathrm{k}}_{\mathbf{l}^1\mathbf{l}^2\ldots\mathbf{l}^{\mathrm{k}}}$ only need to be stored if $\forall_{\mathrm{i,j}}\ \mathbf{l}^{\mathrm{i}} - \mathbf{l}^{\mathrm{j}} \in \mathcal{P}$. Added to the fact that these cumulant tensors – as generalisations of covariance matrices – are symmetric[8], we can increase efficiency by storing and computing only reconstructable and unique tensor entries. The feature extraction procedure is illustrated in figure 3.2, which shows the extracted cumulant feature tensors for two 1-dimensional sources with identical shapes, but different activities.

Having defined and extracted these neurally relevant features from the data that we wish to reconstruct, we have two tasks left to define the feature reconstruction cost, $\mathcal{C}^{\mathrm{reconst}}_{\mathbf{X}}$. First, we need to compute the reconstruction of the features $\hat{\mathbf{Y}}$, given a current setting of the inferred sources, $\hat{\mathsf{O}}$. Secondly, we need to define a distance metric between the reconstructed and original features.

Starting with the feature reconstruction, we need to remember that our main purpose with the feature representation was not having to compute the basis function activations, $\{\mathbf{x}^{\mathrm{s,t}}\}$, for each source and time point. This, however, means we do not have access to the reconstruction of the original data $\hat{\mathbf{y}}(t)$, as we are only to reconstructing the extracted features $\mathbf{Y}$. To indicate this difference, let us define a new set of basis function activations, $\{\mathbf{X}^{\mathrm{s,r}} \in \mathbf{R}^{\mathrm{K}}\}^{\mathrm{R}}_{\mathrm{r}=1}$, for each source s, and each r cumulant order. These are r-th order tensors of size $K \times K \times \ldots_{\mathrm{r}} \times K$, with the $X_{\mathrm{k}_1\mathrm{k}_2,\cdots\mathrm{k}_{\mathrm{r}}}$ entries being the activation of a higher order basis function, which is the result of the tensor product of respective bases, $\otimes^{\mathrm{r}}_{\mathrm{i}=1} b^{\mathrm{k}_{\mathrm{i}}}(\cdot)$; see equation 3.9 for the uses of $\mathbf{X}^{\mathrm{r}}$ in the feature-based generative model. Although each $\mathbf{X}^{\mathrm{r}}$ has K^{r} entries, these coefficient tensors themselves need to be symmetric tensors, which significantly reduces their degrees of freedom compared to the number of entries: $\binom{K+r-1}{r} \ll K^{\mathrm{r}}$. To illustrate the tractability achieved through this feature representation, it is helpful

[8] This is also called supersymmetric in some communities. See the work by Comon et al. (2008) for a detailed introduction to symmetric tensors and multilinear calculations.

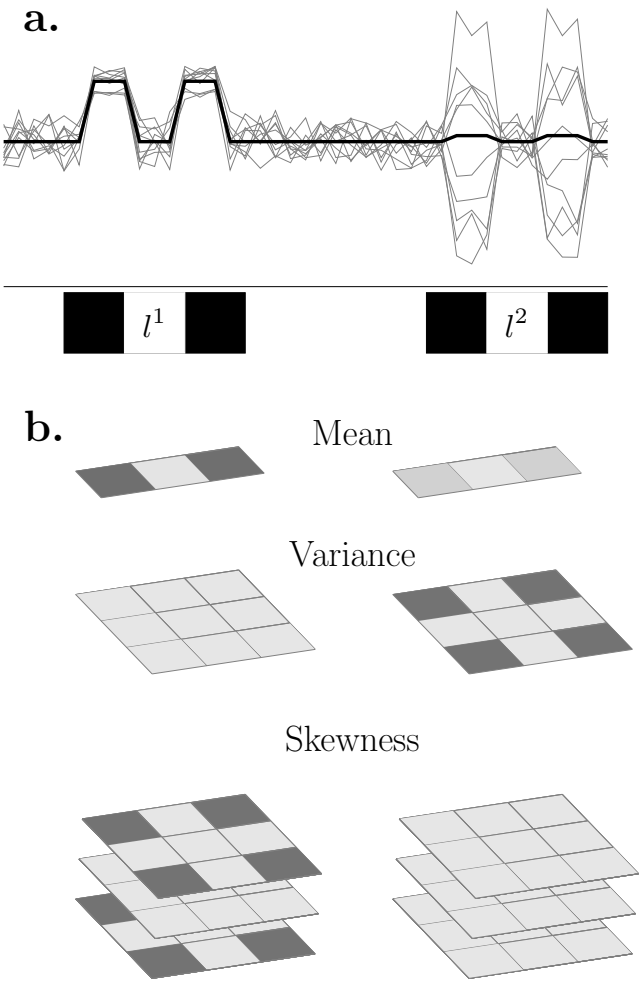


Figure 3.2: Feature extraction procedure in CHOMP for two sources with identical shapes. **a.** Two sources in a 1D space, l^1 with high mean and skewness, l^2 with high variance. The thick line is the true mean across the whole space, the thin ones are individual samples. The squares below indicate observation locations in 1D and the black colour shows the shape of the sources at those locations. **b.** The first three resulting cumulant tensors. Darker colour indicates higher value. The source at l^1 would have been found by an algorithm reconstructing mean values only, but at l^2 the variance feature is likely necessary for successful source localisation.

to compare the number of free parameters in CHOMP to the number needed to reconstruct the full time course. In a realistic scenario of $K = 4$ bases, $R = 4$ cumulant orders and $T = 1000$ samples:

$$\sum_{\mathrm{r}=1}^{\mathrm{R}} \binom{K+r-1}{r} = 69 \ll 4000 = T * K \tag{3.8}$$

Note that this nearly 100-fold reduction is to be understood per source, and in calcium imaging we often expect hundreds of sources within a single dataset. Therefore CHOMP needs to estimate on the order of 10000 times less free parameters during inference of source location, compared to a method that would reconstruct the complete time course at the same time. On the other hand,

CHOMP retains much more information about the time course itself than a method that operates only on the mean of all samples, admittedly with only $K = 4$ free parameters per source. Note that in fact CHOMP directly encompasses the mean-only method; we can simply set $R = 1$, and ignore higher order cumulants. Having defined this new higher order feature reconstruction, using $\{\mathbf{X}^{s,r}\}$ as coefficients, we can compute the full reconstruction of each tensor of order r; this is similar to equation 3.4, but now we reconstruct feature tensors of multiple orders, rather than multiple temporal samples:

$$\begin{aligned} &\forall\, \mathbf{l}^{i} \in \mathcal{I}, \quad r \in \{1 \dots R\} \\ &\widehat{Y}^{r}{}^{\mathbf{l}^{1},\cdots,\mathbf{l}^{r}}(\mathsf{O}_{\mathbf{X}}, \mathsf{B}) = \left(\sum_{s=1}^{S} \sum_{k_1,k_2,\cdots,k_r}^{K,K,\cdots,K} \left(X^{s,r}_{k_1 k_2 \cdots k_r} \prod_{i=1}^{r} b^{k_i}(\mathbf{l}^{i} - \mathbf{l}^{s}) \right) \right), \end{aligned} \tag{3.9}$$

where $\mathsf{O}_{\mathbf{X}}$ represents the set of source feature reconstruction estimates, by storing their locations and feature activation coefficient tensors:

$$O^{s}_{\mathbf{X}} = \left\{ \mathbf{l}^{s} \in \mathbb{R}^{D}, \quad \left\{ \mathbf{X}^{s,r} \in \mathbb{R}^{K \times K \times \ldots_{r} \times K} \right\}_{r=1}^{R} \right\}. \tag{3.10}$$

Furthermore, note again, that each reconstructed feature tensor, $\widehat{\mathbf{Y}^{r}}$, is a $\mathcal{I} \times \mathcal{I} \times \ldots_{r} \times \mathcal{I}$ space, indexed by r 'locations' $\{\mathbf{l}^{i} \in \mathcal{I}\}_{i=1}^{r}$; whereas the reconstruction coefficient tensors, $\mathbf{X}^{s,r}$, are indexed by r integers, $\{k_i\}_{i=1}^{r}$, which also select the appropriate basis functions for the product in equation 3.9.

The final step in fully defining the feature reconstruction cost $C^{\text{reconst}}_{\mathbf{X}}$, is to set a distance metric between the reconstructions and the observations. I give here a generic definition that showcases the flexibility of CHOMP. Afterwards, I show what hypotheses various choices represent, then briefly discuss potential extensions.

Firstly, in our model, the reconstruction orders are completely independent due to the use of cumulant feature representation, and thus the most important choice is balancing the weights of individual feature order reconstruction errors; we can do this with R weights, $\mathbf{c} \in \mathbb{R}^{R}_{\geq 0}$. Secondly, we need to define R distance metrics, $f^{r}(\cdot,\cdot) : \mathcal{I}^{r} \times \mathcal{I}^{r} \mapsto \mathbb{R}_{+}$, acting on the observed and reconstructed features. Then simply

$$C^{\text{reconst}}_{\mathbf{X}} = \sum_{r=1}^{R} c_r f^{r}\left(\mathbf{Y}^{r}, \widehat{\mathbf{Y}^{r}}\right), \tag{3.11}$$

but this simple form gives us a wealth of options. If only one c_i is non-zero, then this results in simple pursuit algorithms, with $c_1 \neq 0$ reconstructing the mean signal as done previously by Pachitariu et al. (2013), whereas $c_2 \neq 0$ favouring locations with higher spatially structured covariance, and when only $c_4 \neq 0$, we carry out pure kurtosis pursuit, pinpointing sparsely firing cells. Having multiple non-zero coefficients in $\mathbf{c}$ essentially interpolates between these options.

The other choice is of course the distance metric itself, with which one needs to be more careful. The simplest option is an entrywise norm, such as the Frobenius norm, for each r order. However, it should be noted that for a given dataset $\mathbf{y}$, where noise is independent and identically distributed across samples, the feature noise levels we expect in the individual entries vary both with the order of the feature tensor, as well as whether or not the entry is a diagonal or an off-diagonal entry within the tensor. This may be further complicated by having spatial autocorrelation within individual $\mathbf{y}^{\mathrm{t}}$ samples, which should also be taken into account in the error metric used to compute the reconstruction cost. A generally applicable procedure, assuming additive Gaussian noise on samples, is to estimate noise levels and spatial autocorrelation lengths using the $\mathbf{y}^{\mathrm{t}}$ samples, applying Isserlis' theorem (Isserlis 1918; Vignat 2011) to compute the expected noise in the various tensor entries, then defining the appropriately weighted entrywise Frobenius norm:

$$f^{\mathrm{r}}\left(\mathbf{Y}^{\mathrm{r}}, \widehat{\mathbf{Y}^{\mathrm{r}}}\right) = \frac{1}{N^{\mathrm{r}}(\mathcal{P})} \left\| \frac{1}{\boldsymbol{\sigma}^{\mathrm{r}}(\mathbf{y})} \odot \left(\mathbf{Y}^{\mathrm{r}} - \widehat{\mathbf{Y}^{\mathrm{r}}}\right) \right\|_{\mathrm{F}}^{2}, \tag{3.12}$$

where $\boldsymbol{\sigma}^{\mathrm{r}}(\mathbf{y})$ is the expected entrywise standard deviations based on the noise levels in $\mathbf{y}$, and $N^{\mathrm{r}}(\mathcal{P})$ is the number of elements in a feature patch of order r, which ensures higher reconstruction orders are not overrepresented in the cost function simply because of their higher number of reconstructable entries.

Having written down the reconstruction cost, we now only need to define the cost of sources used, $\mathcal{C}_{\mathbf{X}}^{\mathrm{sources}}(\hat{\mathsf{O}})$, to arrive at a full description of the cost function we wish to minimise. In basic matching pursuit algorithms, this is simply a weighted function of the number of sources used in the reconstruction (the cardinality of $\hat{\mathsf{O}}$), but again, CHOMP provides more options due to its convolutional nature. Importantly for neural applications, as we believe our sources occupy physical space, we may wish to discourage the algorithm from finding highly overlapping sources. We don't want to completely disable this behaviour, especially in $D = 2$ dimensions, as overlapping sources can be the result of imperfect optical sectioning during two-photon calcium microscopy. Generally a good sources cost is

$$\mathcal{C}_{\mathbf{X}}^{\mathrm{sources}}(\hat{\mathsf{O}}_{\mathbf{X}}) = \lambda \|\hat{\mathsf{O}}_{\mathbf{X}}\| + \sum_{\mathrm{s}=1}^{\mathrm{S}} \sum_{\mathrm{s}'=1}^{\mathrm{s}} g(\hat{O}_{\mathbf{X}}^{\mathrm{s}}, \hat{O}_{\mathbf{X}}^{\mathrm{s}'}), \tag{3.13}$$

where λ is the cost per source and $g(\hat{O}_{\mathbf{X}}^{\mathrm{s}}, \hat{O}_{\mathbf{X}}^{\mathrm{s}'})$ is an overlap penalty between sources s and s'. In its simplest form, it operates on distances between source locations $g(\cdot,\cdot) \propto \|\mathbf{l}^{\mathrm{s}} - \mathbf{l}^{\mathrm{s}'}\|$, and is decreasing with distance, being zero for sources further away than the patch size $\mathcal{P}$. However, one may also wish to take into account the reconstructions of nearby sources via the coefficients $\mathbf{X}^{\mathrm{s}}$ and $\mathbf{X}^{\mathrm{s}'}$, penalising nearby sources with complementary reconstructions less.

To conclude, the complete cost function we wish to minimise is built as the sum of the recon-

struction cost, and the cost for sources used, and is in the following parametric form:

$$\mathcal{C}_{\mathbf{X}} = \mathcal{C}_{\mathbf{X}}^{\text{reconst}}\left(\mathbf{Y}^{\text{r}}, \widehat{\mathbf{Y}^{\text{r}}}\right) + \mathcal{C}_{\mathbf{X}}^{\text{sources}}(\hat{\mathsf{O}}_{\mathbf{X}})$$

$$\begin{aligned} \mathcal{C}_{\mathbf{X}}^{\text{reconst}} &= \sum_{\text{r}=1}^{\text{R}} c_{\text{r}} f^{\text{r}}\left(\mathbf{Y}^{\text{r}}, \widehat{\mathbf{Y}^{\text{r}}}\right) \\ f^{\text{r}}\left(\mathbf{Y}^{\text{r}}, \widehat{\mathbf{Y}^{\text{r}}}\right) &= \frac{1}{N^{\text{r}}(\mathcal{P})} \left\| \frac{1}{\boldsymbol{\sigma}^{\text{r}}(\mathbf{y})} \odot \left(\mathbf{Y}^{\text{r}} - \widehat{\mathbf{Y}^{\text{r}}}\right) \right\|_{\text{F}}^2 \end{aligned} \tag{3.14}$$

$$\mathcal{C}_{\mathbf{X}}^{\text{sources}} = \lambda |\hat{\mathsf{O}}_{\mathbf{X}}| + \sum_{\text{s}=1}^{\text{S}} \sum_{\text{s}'=1}^{\text{s}} g(\hat{O}_{\mathbf{X}}^{\text{s}}, \hat{O}_{\mathbf{X}}^{\text{s}'}),$$

where the main user-adjustable parameters are $\mathbf{c}$, the weights of individual reconstruction orders, and λ, the cost for each source. On a lower level, one can further customise the designs of the metrics for reconstruction quality $f^{\text{r}}(\cdot,\cdot)$ and for source overlap penalty $g(\cdot,\cdot)$.

Inference via matching pursuit

The main advantage of matching pursuit based source localisation is that we turn an intractable simultaneous inference problem of an unknown number of sources into an iterative process. We add sources one-by-one, selecting the new source location greedily, such that it maximally decreases the cost function, until no potential source would decrease the cost anymore. Computing the potential decrease of the cost at a proposed new location in the above defined cost function is a computationally cheap procedure, as the reconstruction cost change depends only on a few of the observations – those within the patch size – and we can solve for all $\mathbf{X}^{\text{r}}$ coefficient tensors in closed form. Furthermore, the change in source cost only depends on a simple function of the previous sources, and even though the computational cost of the overlap-cost increases linearly throughout the inference with each additional source, it is still negligible for the typical number of a few hundred sources.

However, in order to select where the next source should be placed, we essentially need to carry out global optimisation at every iteration across all possible source locations (that is approximately the convex hull of observation locations, padded by the patch size, $\text{Conv}(\mathcal{I} \pm \mathcal{P})$). Although this would of course be computationally extremely expensive and thus intractable, we can exploit two characteristics of the cost function to approximately carry out this global optimisation procedure. Firstly, the reconstruction cost function itself is as smooth as the basis functions $b^{\text{k}}(\cdot)$ are, meaning we can first compute the cost decrease on a grid, and use a few promising locations to initialise local optimisations, ultimately finding the next source location with maximum cost decrease[9]. Secondly, all reconstruction cost changes are local around a proposed location. This means two things: *a.)* we

[9] This is of course not necessarily the global optimum of cost decrease, but it still results in a reproducible greedy algorithm

can massively parallelise computations on modern computational clusters, spreading both data and computation across nodes at will and *b.)* we only need to carry out the computation on the full grid once, since in subsequent iterations we can update this grid only around the previously added source.

After we have selected where to add the next new source, updating the current reconstruction and the residual is simply a byproduct of the selection process, and computationally free. The only extra computational cost is updating the grid that serves to provide initialisation for the next set of local optimisations. The full pseudocode for the procedure is shown in algorithm 3.2.

Algorithm 3.2 High level inference pseudocode for Convolutional Higher Order Matching Pursuit

1: Let $\hat{\mathsf{O}}_{\mathbf{X}} = \{\}$ be the empty set of inferred sources

2: At a set of locations (L) spanning the observation space $\text{Conv}(\mathcal{I} \pm \mathcal{P})$ compute the optimal reconstruction coefficients $\mathbf{X} = \{\mathbf{X}^{\mathrm{r}}\}_{\mathrm{r}=1}^{\mathrm{R}}$ that (locally) minimise the reconstruction error:

3:

$$\begin{aligned} &\forall\, \mathbf{l} \in \mathsf{L}: \\ &\quad \tilde{\mathbf{Y}}(\mathbf{l}) = \hat{\mathbf{Y}}(\hat{\mathsf{O}}_{\mathbf{X}} \cup O^{\mathrm{l}}_{\mathbf{X}}), \quad \text{where } O^{\mathrm{l}}_{\mathbf{X}} = \{\mathbf{X}, \mathbf{l}\} \\ &\quad \tilde{\mathbf{X}}(\mathbf{l}) = \arg\min_{\mathbf{X}} \mathcal{C}^{\text{reconst}}_{\mathbf{X}}\left(\mathbf{Y}, \tilde{\mathbf{Y}}(\mathbf{l})\right)) \end{aligned} \tag{3.15}$$

4: **repeat**

5: At each location $\mathbf{l} \in \mathsf{L}$ compute $\Delta\mathcal{C}_{\mathbf{X}}$ based on $\tilde{\mathbf{X}}(\mathbf{l})$

6: At a few 'promising' locations, initialise gradient descent jointly in $\mathbf{l}$ and $\mathbf{X}$ on $\mathcal{C}_{\mathbf{X}}$ to find the local minima, and subsequently select the global minimum location and reconstruction $\tilde{O}^{\text{new}}_{\mathbf{X}}$.

7: If adding $\tilde{O}^{\text{new}}_{\mathbf{X}}$ decreases the cost, expand the solution set, $\hat{\mathsf{O}}_{\mathbf{X}} := \hat{\mathsf{O}}_{\mathbf{X}} \cup \tilde{O}^{\text{new}}_{\mathbf{X}}$.

8: Update $\tilde{\mathbf{X}}(\mathbf{l})$ affected locally by the new source, where $\mathbf{l} - \mathbf{l}^{\text{new}} \in 2\mathcal{P}$

9: **until** $\tilde{O}^{\text{new}}_{\mathbf{X}}$ does not decrease the cost $\mathcal{C}_{\mathbf{X}}$.

10: Given the locations in the solution set $\hat{\mathsf{O}}_{\mathbf{X}}$, calculate the individual sample reconstruction coefficients $\mathbf{x}^{\mathrm{s,t}}$, and thus the full reconstruction of the original data, $\hat{\mathsf{O}}$.

Observations on a grid

As discussed above, an efficient solution to finding the global optima of the cost function involves first computing the cost surface on a grid of potential source locations L. From here onwards, I am also going to assume that my observations also lie on a grid, which is typically the case in two-photon calcium recordings (be it 2 or 3 dimensional), my main application of interest. This assumption allows for a further set of simplifications when computing $\tilde{\mathbf{X}}(\mathbf{l})$ in step 3 of algorithm 3.2.

Let a local patch around an observation at location $\mathbf{l}$ be $\mathcal{P}_{\mathbf{l}}$, and let the observation grid's step size be such that a patch covers a total of M observations. Therefore, we can represent a local patch of observations as a D-th order tensor with M elements, let us denote this patch as $\mathbf{y}^{\mathrm{t}}_{\mathbf{l}}$. Similarly, we may represent the individual basis functions $b^{\mathrm{k}}(\cdot)$ as D-th order tensors with a total of M elements,

let $\mathbf{b}^{\mathrm{k}} = b^{\mathrm{k}}(\mathcal{P})$ denote this tensor[10].

When computing the cumulant features $\mathbf{Y}^{\mathrm{r}}$, we can now similarly represent local patches of them as D^{r}-th order tensors with M^{r} elements, denoting such a local feature patch tensor as $\mathbf{Y}^{\mathrm{r}}_{\mathbf{l}}$. In order to write down local reconstructions – following equation 3.9 – in a tensor format now available due to the grid structure, we can define outer products of the basis functions tensors, letting $\mathbf{b}^{\mathrm{k}_1\mathrm{k}_2\ldots\mathrm{k}_{\mathrm{r}}} = \mathbf{b}^{\mathrm{k}_1} \otimes \ldots \otimes \mathbf{b}^{\mathrm{k}_{\mathrm{r}}}$ (where $\forall i \;\; k_{\mathrm{i}} \in \{1 \ldots K\}$) indicate a D^{r}-th order tensor with M^{r} elements. Let us define the respective flattened (vectorised) tensors as $\vec{\mathbf{Y}}^{\mathrm{r}}{}_{\mathbf{l}}$ and $\vec{\mathbf{b}}^{\mathrm{k}_1\mathrm{k}_2\ldots\mathrm{k}_{\mathrm{r}}}$ being M^{r} long vectors, with $\mathbf{B}^{\mathrm{r}} \in \mathbb{R}^{\mathrm{M}^{\mathrm{r}} \times \mathrm{K}^{\mathrm{r}}}$ being the matrix collecting all possible variations of the vectorised basis function outer products as columns. Then our local reconstruction problem can be written as

$$\forall\, r: \quad \hat{\mathbf{X}}^{\mathrm{r}}_{\mathbf{l}} = \underset{\mathbf{X}^{\mathrm{r}}}{\arg\min}\, f^{\mathrm{r}}\left(\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}},\, \mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}}\right). \tag{3.16}$$

For clarity, let's assume here that $f^{\mathrm{r}}(\cdot,\cdot)$ is simply the Frobenius norm, but this easily generalises for the entrywise weighted versions suggested above as well. Therefore the solution is

$$\begin{aligned} \hat{\mathbf{X}}^{\mathrm{r}}_{\mathbf{l}} &= \underset{\mathbf{X}^{\mathrm{r}}}{\arg\min} \left\| \vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} - \mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}} \right\|^2_{\mathrm{F}} \\ &= \left(\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\right)^{-1}\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} \\ &\text{where} \quad \mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} = \mathbf{B}^{\mathrm{r}\top}\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}}\,. \end{aligned} \tag{3.17}$$

Here $\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} \in \mathbb{R}^{\mathrm{K}^{\mathrm{r}}}$ represents the vectorised projection of the r-th order local patch cumulant tensor onto the r-th order basis function tensors. Having solved for the optimal reconstruction coefficients $\hat{\mathbf{X}}_{\mathbf{l}}$, we can compute the maximum decrease in the reconstruction cost, given a new source at $\mathbf{l}$, as

$$\begin{aligned} \Delta C^{\mathrm{reconst}}_{\mathbf{X}}(\mathbf{l}) &= \sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}} \left(\left\| \vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} - \mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}} \right\|^2_{\mathrm{F}} - \left\| \vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} \right\|^2_{\mathrm{F}} \right) \\ &= \sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}} \left(-2\vec{\mathbf{X}}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}\top}\vec{\mathbf{Y}}^{\mathrm{r}}_{\mathbf{l}} + \vec{\mathbf{X}}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\vec{\mathbf{X}}^{\mathrm{r}} \right) \\ &= -\sum_{\mathrm{r}=1}^{\mathrm{R}} c_{\mathrm{r}} \left(\mathbf{Z}^{\mathrm{r}\top}_{\mathbf{l}} \left(\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\right)^{-1} \mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} \right). \end{aligned} \tag{3.18}$$

This means that we do not even need to store the feature reconstruction coefficients $\mathbf{X}$ throughout the steps 3-5 of algorithm 3.2, as using only the projections of the data onto the expanded bases enables us to compute the potential reconstruction cost change at all locations $\mathbf{l} \in \mathsf{L}$. Furthermore,

[10] This description is applicable when the approximating grid L is chosen with a grid size that is an integer multiple of the observation grid size. This is typically a good choice for calcium imaging applications, where neurons span multiple pixels/voxels. For approximation grid sizes that are rational multiples of the original, e.g. $1/n$ step sizes, one would need n different sets of $\mathbf{b}^{\mathrm{k}}$ tensors, applying the appropriate one at each sub-grid offset value during the convolutional procedure described in the following. This finer initial approximation increases the computational complexity n^{D}-fold initially, and n^{Dr}-fold during the local updates, while the procedure remains highly parallelisable, and the gradient descent optimisation will likely converge faster due to the better initialisations.

these projections can be computed very efficiently by higher order convolutions as shown in equation 3.17, and then to calculate the cost change, we only need to compute the inverse $\left(\mathbf{B}^{\mathrm{r}\top}\mathbf{B}^{\mathrm{r}}\right)^{-1}$ once, resulting in fast quadratic form computations.

To carry out steps 6 and 8 of algorithm 3.2, the optimisation and the update, we can similarly complete the whole procedure in the projection space of $\mathbf{Z}^{\mathrm{r}}$, noting that both of these steps again require evaluation the $b^{\mathrm{k}}(\cdot)$ basis functions at arbitrary locations, rather than using their $\mathbf{b}^{\mathrm{k}}$ tensor representations.

Solutions on a grid

In order to further simplify the inference process, we may wish to sacrifice a bit of localisation accuracy, and decide that our source locations may only lie on the grid L. Note that this represents an approximation to the original generative model described in equation 3.4, where source locations were in a D dimensional Euclidean space, rather than in a finite discrete subset thereof.

However, with a fine enough grid we lose little in localisation, but completely eliminate the expensive gradient descent optimisation in step 6 of algorithm 3.2, and may also significantly simplify the local coefficient update of step 8. Specifically, with this approximation we never need to evaluate the bases $b^{\mathrm{k}}(\cdot)$ at arbitrary locations, and therefore we do not even need to represent the complete basis functions, only the M-element tensors $\mathbf{b}^{\mathrm{k}}$.

The local update after adding a new source to the solution set $\hat{\mathsf{O}}_{\mathbf{X}}$ generally requires computing the residual in feature space $\mathbf{Y}$ by subtracting the reconstructed $\mathbf{BX}$, as the cost function change associated with the next source depends on the proposed source's ability to fit this residual. However, as data and residuals only enter the cost change calculations via the basis projections $\mathbf{Z}$ (see equation 3.18), we do not need to explicitly represent the residual, we may just compute the change in the projections directly, after adding a new source $O^{\mathrm{s}} = \{\hat{\mathbf{X}}^{\mathrm{s}}, \mathbf{l}^{\mathrm{s}}\}$. For clarity, I first show the changes for $r = 1,2$ using the basis function formulation – that is applicable for arbitrary $\mathbf{l}^{\mathrm{s}}$ source locations – then discuss how the tensor representation may be used to speed up the computations. For arbitrary source locations, the update is

$$\begin{aligned}
\Delta_{\mathrm{s}}[\mathbf{Z}^1_{\mathbf{l}}]_{\mathrm{k}} &= \int_{\mathbf{l}'} b^{\mathrm{k}}(\mathbf{l}'-\mathbf{l}) \left(\sum_{\mathrm{k}'} b^{\mathrm{k}'}(\mathbf{l}'-\mathbf{l}^{\mathrm{s}}) \, [\hat{\mathbf{X}}^{\mathrm{s},1}]_{\mathrm{k}'} \right) \\
\Delta_{\mathrm{s}}[\mathbf{Z}^2_{\mathbf{l}}]_{\mathrm{k}_1\mathrm{k}_2} &= \int \int_{\mathbf{l}'_1,\mathbf{l}'_2} b^{\mathrm{k}_1}(\mathbf{l}'_1-\mathbf{l}) \, b^{\mathrm{k}_2}(\mathbf{l}'_2-\mathbf{l}) \left(\sum_{\mathrm{k}'_1,\mathrm{k}'_2} b^{\mathrm{k}'_1}(\mathbf{l}'-\mathbf{l}^{\mathrm{s}}) \, b^{\mathrm{k}'_2}(\mathbf{l}'-\mathbf{l}^{\mathrm{s}}) \, [\hat{\mathbf{X}}^{\mathrm{s},1}]_{\mathrm{k}'_1\mathrm{k}'_2} \right).
\end{aligned} \tag{3.19}$$

Therefore, in order to carry out the update directly in $\mathbf{Z}$ space, we essentially need to compute the pairwise convolutions of the basis functions with each other in every order, then weight the convolutions by the reconstruction coefficients, that are linear functions of $\mathbf{Z}$ themselves (see equation 3.17).

Therefore we can define a single linear operator, $U^{\mathrm{r}}(\Delta\mathbf{l} \mid \{b^{\mathrm{k}}(\cdot)\}_{\mathrm{k}=1}^{\mathrm{K}})$, that is a function of distance and the basis functions, and maps the projection vector $\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}^{\mathrm{s}}}$ at newly added s source to the projection vector at nearby locations $\mathbf{l}$:

$$\Delta_{\mathrm{s}}\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}} = U^{\mathrm{r}}(\mathbf{l}-\mathbf{l}^{\mathrm{s}} \mid \{b^{\mathrm{k}}(\cdot)\}_{\mathrm{k}=1}^{\mathrm{K}}) \cdot \mathbf{Z}^{\mathrm{r}}_{\mathbf{l}^{\mathrm{s}}} \tag{3.20}$$

When the solution locations $\mathbf{l}^{\mathrm{s}}$ are restricted to the L grid, this linear update operator can be represented by a $M_* \times K^{\mathrm{r}} \times K^{\mathrm{r}}$ tensor $\mathbf{U}^{\mathrm{r}}$, and only needs to be computed once. Here M_* is the number of grid points within a $2\mathcal{P}$ patch, which is the area where $\mathbf{Z}$ projections are affected by adding a new source. Therefore updating all projections around the newly added location can be written as:

$$\Delta_{\mathrm{s}}\mathbf{Z}^{\mathrm{r}}_{2\mathcal{P}(\mathbf{l}^{\mathrm{s}})} = \mathbf{U}^{\mathrm{r}}\,\mathbf{Z}^{\mathrm{r}}_{\mathbf{l}^{\mathrm{s}}}, \tag{3.21}$$

where $2\mathcal{P}(\mathbf{l}^{\mathrm{s}})$ selects the appropriate M_* locations affected. A visual summary of algorithm 3.2, using the vectorised projections $\mathbf{Z}$, is shown in figure 3.3.

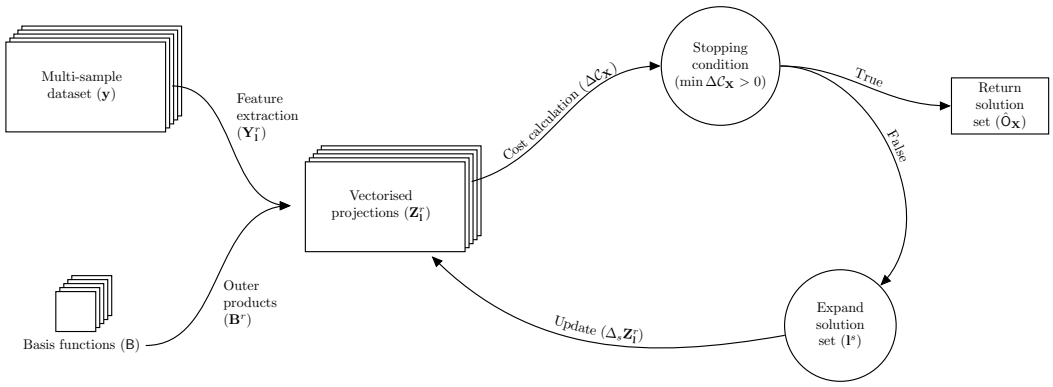


Figure 3.3: Algorithm flowchart for Convolutional Higher Order Matching Pursuit. Showcases the initial computation of projecting the data features onto the basis functions, then the iterative update loop, which involves only spatially local computations.

3.2.2 Dictionary learning

The inference algorithm proposed in the previous section relies on a fixed set of basis functions B – often called a dictionary – that collectively are capable of describing the temporal changes of the various signal modes of cells of interest. Although in some signal processing applications these modes may indeed be known, or at least stationary over multiple recordings, which use the same observation process and similar signal generators, it is nevertheless useful to discuss how one may initialise a set of basis functions, as well as how to iteratively update them based on the inference results. I first discuss the update procedure, as it is generally applicable to arbitrary types of data, then provide potential initialisations that are more specific to two-photon calcium imaging datasets.

In order to update the basis functions, we have to select appropriate training data that admits

to sparse decomposition, resulting in the new bases. As the basis functions' support is the patch $\mathcal{P}$, our first task is to select patches around the inferred source locations; let these locations be $\mathcal{P}(\mathbf{l}^{\mathrm{s}})$. In the general case, where $\mathbf{l}^{\mathrm{s}}$ are arbitrary locations, these extracted patches will result in location-value pairs ($\mathbf{u} \in \mathcal{P}$ and $v \in \mathbb{R}$), where the values are scalar observations $y^{\mathrm{t}}_{\mathbf{l}}$ (where $\mathbf{l} \in \mathcal{I}$), and the locations are shifted relative to the current source location, with $\mathbf{u} = \mathbf{l} - \mathbf{l}^{\mathrm{s}}$ spanning the patch $\mathcal{P}$. We then use these $\{\mathbf{u}, v\}$ pairs as training data to any generalised additive model capable of learning K components that explain the training data well, and represent basis functions that can be evaluated at arbitrary new locations within the patch $\mathcal{P}$. Such models are described by Adam, Durrande, and John (2018). One difficulty during the preparation of this training data is that multiple inferred sources may affect the same observation location owing to overlapping patches. In these cases care must be taken to either omit these affected locations from the training data, or – when affected by n different sources – represent them as n separate samples, with values $\{v^*_{\mathrm{i}}\}_{\mathrm{i}=1}^{\mathrm{n}}$, where v^*_{i} is the observed value v, minus the sum of reconstructions of all but the i-th source affecting that location[11].

When both observations and solutions lie on a grid we do not have to represent the bases as functions, but we can rather use their D-th order tensor representation, $\mathbf{b}^{\mathrm{k}}$. Similarly, the patches around inferred sources, $\mathbf{y}^{\mathrm{t},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})}$, are D-th order tensors, with the same size as the bases, with the $*$ again suggesting to explain away other sources at overlaps. Updating this tensor representation of the bases is a much more standard problem, as we can vectorise the patches, collect them into a matrix as columns, and apply standard matrix factorisation methods to recover the top K (singular) vectors that best explain the data, according to some metric and satisfying given constraints. Applicable matrix factorisation methods are Singular Value Decomposition (SVD), Independent Component Analysis (ICA) or Non-negative Matrix Factorisation (NMF). That is,

$$\begin{aligned} [\mathbf{V}]_{\cdot,\mathrm{s}} &= \vec{\mathbf{y}}^{\mathrm{t},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})} \qquad \text{for all } O^{\mathrm{s}} \in \mathsf{O} \\ \left\{\mathbf{b}^{\mathrm{k}}_{\mathrm{new}}\right\}_{\mathrm{k}=1}^{\mathrm{K}} &= \text{MatrixFactorisation}_{\mathrm{K}}\left(\mathbf{V}\right). \end{aligned} \tag{3.22}$$

Decomposing feature tensors. Alternatively – especially when we are trying to ignore some orders of signal cumulants during source localisation, and not even aiming to represent the full temporal signal – we may wish to update the basis functions based on the feature tensors, rather than based on the original observation samples. In this case, we can similarly collect the feature tensors $\mathbf{Y}^{\mathrm{r},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})}$, which are now D^{r}-th order tensors computed from the noisy data. Learning sparse decomposition from tensors of multiple orders is still an active research area, especially when we have multiple noisy samples from each order. Two methods that are capable of implementing tensor-based learning are Higher Order SVD (HOSVD, by Lathauwer, Moor, and Vandewalle (2000)) and Multi-Tensor Factorisation (MTF, by Khan, Leppäaho, and Kaski (2016)).

The former, HOSVD, relies on averaging across the decompositions of all standard factor-k

[11] This is similar to the idea of the cavity distribution in expectation propagation

flattenings of the tensors. Although this is very simply applicable for symmetric tensors – as we only have to do a single flattening – to my knowledge it has only been derived for a single tensor of a single given order. My attempt at extending HOSVD to multiple tensor samples of multiple orders proceeded as follows. For each tensor order r, and each sample s flatten the symmetric feature tensor $\mathbf{Y}^{\mathrm{r},*}_{\mathcal{P}(\mathbf{l}^{\mathrm{s}})}$ into an $M \times M^{\mathrm{r}-1}$ matrix, and stack the S samples as columns, resulting in an $M \times S * M^{\mathrm{r}-1}$ matrix, that I call $\mathbf{V}^{\mathrm{r}}$. If we now stacked tensors of different orders as columns again, the lower orders would be significantly underrepresented, as each additional tensor order yields M-times as many samples as the previous one. Therefore, we need to ensure we are representing each tensor order sufficiently well, before doing the decomposition. I found the following heuristic options: a.) One might randomly subsample the columns gained from higher order tensors to reduce the number of columns to M, the same as the first order tensor, noting this is extremely lossy. Alternatively, one can renormalise the columns of these $\mathbf{V}^{\mathrm{r}}$ matrices. Two options seem to work in practice, but was yet unable to prove their domain of applicability. b.) The first renormalisation option is to simply divide each entry in each matrix by M^{r}/M, the number of entries in the tensor divided by M, the number of elements in the basis functions. I found however, that this could now lead to under-representing the effect of higher order tensors. c.) The second option comes from my earlier discussion of the number of free elements in a symmetric tensor (see equation 3.8), and thus I use the number of free entries in each symmetric tensor, rather than the total number of entries. This penalises higher orders significantly less, as $\binom{M+r-1}{r} \ll M^{\mathrm{r}}$ for the typical large M and $r > 2$.

The other method, Multi-Tensor Factorisation (MTF), defines a principled joint Bayesian prior for the weighing between multiple tensors orders, but I found the resulting computational complexity and running times of the provided implementation prohibitive to use with the typical parameters in calcium imaging applications with $M > 50$, $R > 2$ and $S > 100$. Therefore, if dictionary updates are required, I for now recommend the use of equation 3.22 with well-understood matrix factorisation methods, until multi-sample, multi-order tensor factorisation methods enter a more maturely applicable phase.

Application to neural data. Next, I discuss the specific application of CHOMP inference and dictionary learning to neural two-photon calcium imaging datasets. CHOMP relies on basis functions capable of describing neurons well. One way of initialising the CHOMP dictionary is to use the source locations indicated by human markers as training data, then apply the dictionary learning shown in equation 3.22. Although as discussed previously, this does introduce unnecessary human bias, typically the most prominent 10-50 cells per field of view are agreed upon by all human markers, and the bases that describe them do not significantly change in repeated experiments – provided those use the same microscope with similar settings, and probe the same brain areas in the same

species. These human-segmented datasets are typically available or easy to collect in most experimental laboratories, and thus can be used as an initialisation to CHOMP. The patch size $\mathcal{P}$ of basis functions is chosen to ensure including the segmented cells' full extent.

That said, I both prefer reducing the human bias in data analysis, and does not necessarily have access to previously segmented data of interest; thus I describe here an initialisation that is purely driven by a model of neural signal generation. Firstly, the most popular type of recordings, and thus the ones I mainly focused on, are two-dimensional optical cross-sections of superficial cortex layers containing pyramidal neuron cell bodies. The typical shapes that correspond to the cell bodies in such recordings are filled circles and annuli ('donuts'). As the fluorescent calcium sensors are designed to mainly be expressed in the cytoplasm, but not the nucleus, the filled shape corresponds to cells for which the optical cross-section does not include the nucleus, or in which the calcium sensor protein is overexpressed also in the nucleus (these latter cells are typically dead). Conversely, donuts correspond to cells in which the darker middle section is the nucleus, with the halo of cytoplasm around it. Other active neural elements in the field of view are neural processes (axons or dendrites); the ones that are perpendicular to the field of view show up as speckles in the image, whereas the non-perpendicular ones appear as long thin lines. Lastly, non-neural elements typically correspond to vasculature, either perpendicular to the field of view, appearing as larger 'holes', or non-perpendicular vasculature above the image creates 'shadows', affecting a larger area in the field of view (see e.g. figure 3.12a for exemplars of each of these elements).

To initialise a set of basis functions capable of describing such data, we first need to determine the maximum cell size. This is probably best done by visual inspection (as slightly overestimating only increases computational cost, but is not otherwise detrimental), but one may utilise D-dimensional discrete Fourier transform to find the spatial frequencies present in the data, or finding the typical circle size via circle Hough Transform. Once this has been done, a good initialisation for the bases includes a small disk with $K-1$ increasing radii donuts, the linear combination of which is capable of describing speckles, donuts and filled circles of various sizes. After this initialisation, one may proceed iteratively with inference and dictionary updates, and can discover non-centrally symmetric signal modes as well.

Basis function symmetries. The symmetries of the basis functions both during initialisation and the updates is an interesting question. As discussed in chapter 2, we can easily design Gaussian Process function descriptions with arbitrary symmetries, and similarly restricting matrix decomposition methods to find basis modes only with pre-designed symmetries is a matter of reparameterising the locations $\mathbf{u}$ of the training data, then arranging the vectorised values $\vec{\mathbf{v}}^*$ to reflect the symmetrising transformation done in location space. In my experience, however, a number of interesting signal modes may not be symmetric about the centre of the source, and thus artificially limiting the dictionary updates to discover only symmetric modes may render the bases unable to explain interesting

neural behaviour, and thus potentially miss sources during inference. This can especially be the case for large apparent cell sizes and higher frame rates: in these circumstance, some cells – whose processes are not perpendicular to the 2-dimensional optical section – display a calcium release wave propagating through the cell body. Similarly, in high frame-rate 3-dimensional datasets, we would always expect such waves to be present, and should not forcefully introduce symmetries into basis functions.

3.2.3 Iterative inference and dictionary learning across datasets

In order to end up with a good reconstruction of the dataset, we generally need to carry out inference and dictionary update steps iteratively, until $C_{\mathbf{X}}$ does not decrease anymore at the end of a full inference step with further dictionary updates, meaning the iterative inference-update procedure has converged.

However, this may be an expensive procedure and one would end up with slightly different sets of basis functions for each dataset. If the sole purpose of the bases is to provide good source localisation, and in subsequent data processing the only output used will indeed be the source locations, then this is not a problem. However, if the bases are to be used also in determining the full extent of single-cell regions of interest (ROIs) or even during the extraction of the neural signal[12], it is desirable that the learned bases remain stable over multiple data analysis sessions.

We can achieve this by performing the inference simultaneously across multiple datasets (recorded by the same microscope with the same settings, but potentially in different animals, tasks or even brain regions), then doing the dictionary update on all patches collected from all datasets. When this iterative procedure on the sum of all costs from individual datasets, $C_{\mathbf{X}}^{\text{total}} = \sum_{\text{datasets}} C_{\mathbf{X}}^{\text{dataset}}$, converges, we can ensure that further data processing steps may rely on this fixed, stable set of basis functions, regardless of the dataset. This also has the advantage that once such a fixed dictionary is learned, one only needs to perform a single inference step on new, similar datasets, significantly reducing computational needs. Furthermore, as the update tensors $\mathbf{U}^{\text{r}}$ in equation 3.21 are a function of the bases only, they can be computed once and stored for subsequent inferences, leading to further speedup.

3.3 Validation and experimental results

3.3.1 Validation of CHOMP-based location inference

In order to thoroughly evaluate the impact of incorporating higher order cumulants in convolutional matching pursuit, I created a simulation using a broad range of signal distributions, evaluating the

[12] For example, instead of summing up all signal within a binary ROI to represent the signal of that ROI, we can use the optimal reconstruction of the signal that both reduces additive noise and rejects signals resulting from source overlaps

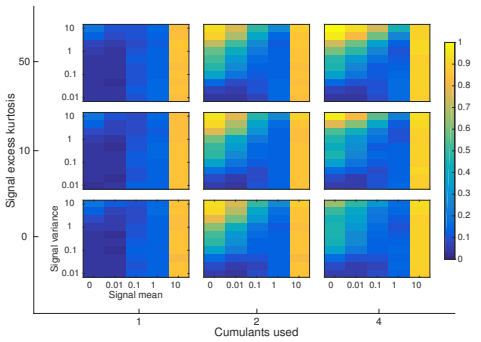
localisation accuracy, and comparing to a group lasso implementation by Boyd et al. (2011). These results are published in Bohner and Sahani (2016), and reproduced here.

Datasets were simulated from the generative model (equation 3.4) in a $D = 1$ dimensional space, with $|\mathcal{I}| = 512$ observations on an equidistant grid, and true source locations restricted to the observation grid, thus $\mathbf{l}^{\mathrm{s}} \in \mathsf{L}$ and $\mathsf{L} = \mathcal{I}$, with 26 sources per field of view (≈ 0.05 occupancy rate, ensuring overlapping sources). Each simulation was based on a new random set of basis functions – provided as input to the tested inference algorithms – with $M = 11$, $K = 2$, entries chosen from a unit variance normal distribution, then the bases are symmetrised about the centre and normalised to unit norm. Signal distributions were modelled as mixtures of Gaussians, with mixture parameters selected by non-linear least-squares to match an intended set of cumulants. I explored symmetric distributions with means and variances spanning multiple orders of magnitudes ($\mu \in [0, 10], \sigma^2_{\mathrm{signal}} \in [0.01, 10]$), zero skewness, and a number of excess kurtosis values $(3, 10, 50)$. For each source, basis activation coefficients, $\mathbf{x}^{\mathrm{s}}_{\mathrm{k,t}}$, were sampled ($T = 1000$ samples) from a distribution with given cumulants, and I explored three scenarios for sources placed within the same field of view. In the *no mixing* case, all sources within the same field of view share signal coefficients sampled from the same distribution, whereas in the *uniform mixture*, the sources' signal distributions were sampled uniformly randomly from all possible settings. Lastly, in the *realistic mixture* case, I first sample a single set of cumulant settings, then the individual sources' distribution parameters are distributed log-normally around the single setting, with one order of magnitude standard deviation. This is called the realistic case, as generally we expect sources' signal distributions to vary slightly within the same field of view, but not over multiple orders of magnitude.

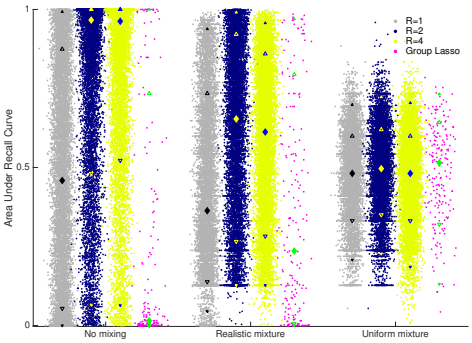
The source locations were chosen uniformly randomly within the field of view – only ensuring no exact co-localisation, overlaps are however common – and the final observations were created as the centred basis functions multiplied by the appropriate source signal coefficients – summed additively for overlapping sources – and corrupted by zero-mean, unit-variance ($\sigma^2_{\mathrm{noise}} = 1$) additive Gaussian noise. For all three signal distribution mixture types within the field of view, I simulated and analysed $n = 10000$ fields of view.

Inference of source locations was carried out as described in figure 3.3, with a stopping condition modified to correspond to the true number of sources. The values of $\boldsymbol{\sigma}^{\mathrm{r}}$ in the reconstruction cost distance metrics $f^{\mathrm{r}}(\cdot)$ (see equations 3.11 and 3.12) were set as described there, resulting in $(\boldsymbol{\sigma}^{\mathrm{r}})^2 = (M^{\mathrm{r}} - K^{\mathrm{r}}) * (\sigma^2_{\mathrm{noise}}/T)$. For comparability I used the true value for $\sigma^2_{\mathrm{noise}} = 1$, instead of estimating it from the data, as suggested for real datasets.

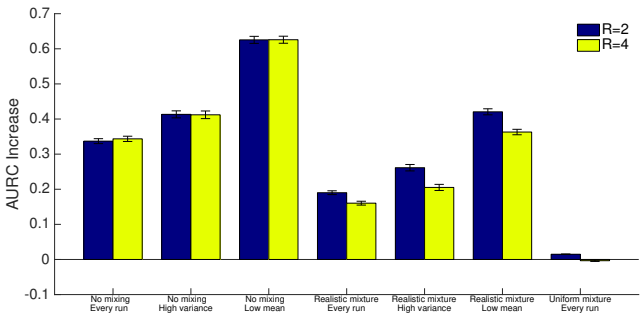
A natural evaluation metric is the frequency with which the algorithm correctly locates the sources. I applied the algorithm using increasing orders of cumulants as features, and found that as long as the signal distributions contain significant higher order structure, it is indeed feasible to attempt to reconstruct those tensors, and they do increase localisation performance (see figure 3.4a). A further feature of greedy algorithms in general, including the current one, is that they provide a



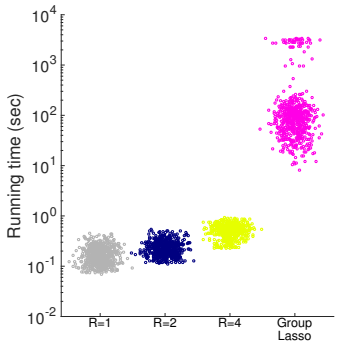
(a) Localisation performance for varying average signal cumulants, while $\sigma^2_{\text{noise}} = 1$ kept constant.



(b) Performance levels of source signal distribution mixtures within a single field of view



(c) The use of higher order features substantially increases localisation performance



(d) Substantial gain in running time compared to matrix factorisation

Figure 3.4: a. CHOMP incorporating higher order cumulants offers substantial gain in localisation performance when the corresponding structure is present in the signal distribution. Each cell shows the fraction of sources correctly localised in $n = 10000$ runs for different signal parameters (exact parameters were distributed normally with a standard deviation of one order of magnitude around the specified value). **b.** Localisation performance expressed as AURC for all runs ($n = 3 \times 10000 + 600$) with varying mixtures of sources within a single run. *No mixing*: All sources share the same signal distribution. *Realistic mixture*: as in (**a**). *Uniform mixture*: Signal distributions may vary up to 4 orders of magnitude within the same field of view. Means, 1σ and 2σ quantiles indicated. **c.** Assessing the improvement within a single field of view gained by incorporating higher order cumulants. Bars are the mean gains in AURC over the first order method, runs selected by signal distribution criterion ($n <= 10000$). *High variance*$>= 1$, *Low mean*$<= 0.1$. Error bars are SEM. **d.** Comparison of running times ($n = 4 \times 600$).

natural ordering of the sources found. We can thus define the Area Under the Recall Curve as

$$\begin{aligned} \mathrm{AURC}(\mathsf{O},\hat{\mathsf{O}}) &= \frac{1}{S}\sum_{\mathrm{s}=1}^{\mathrm{S}} \frac{\mathrm{NumCorrect}[\hat{\mathsf{O}}_{1:\mathrm{s}}] - \mathrm{Chance}_{\mathrm{s}}}{s - \mathrm{Chance}_{\mathrm{s}}} \\ \mathrm{Chance}_{\mathrm{s}} &= \frac{1}{\binom{|\mathcal{I}|}{s}} \sum_{s'=1}^{\mathrm{s}} s' \binom{s}{s'} \binom{|\mathcal{I}|-s}{s-s'} \end{aligned} \tag{3.23}$$

and estimate the performance using this metric, which weights the correctness of earlier sources higher. This is of interest especially in the case, when the field of view contains signals from multiple distributions, such as the realistic or the uniform mixtures (see figure 3.4b). Finally, I looked at how much higher order features offer on a case-by-case basis (figure 3.4c) and found that for all practical cases, higher order estimators are substantially beneficial.

I also compared the proposed CHOMP method to a group lasso implementation by Boyd et al. (2011), that corresponds to the L_1-relaxation of equation 3.3. To this group lasso, I provided as input the full $|\mathcal{I}| \times K|\mathcal{I}|$ spatial design matrix, as well as the grouped time courses belonging to the corresponding locations. Estimated time courses were sorted by their norms to obtain the ordered $\hat{\mathsf{O}}$ for the group lasso, so that AURC could be evaluated as above. CHOMP outperformed the group lasso (figure 3.4b) in source localisation while being over two orders of magnitude faster[13] (see figure 3.4d).

3.3.2 Applications to neural data

Having validated CHOMP on simple data simulated from its generative model, we need to evaluate it on real neural recordings. To this end I used the Neurofinder datasets – carefully characterised in chapter 2 – and unless otherwise stated, the data used as input to CHOMP has been transformed into photon flux estimates and divided by the non-uniform spatial gain, as the preprocessing described in chapter 2 suggests. Finally, before passing to CHOMP, the resulting non-integer photon flux data was stretched to the uint16 range and represented by integers, as that is the expected format of most neural recordings. Note, that although these transformations indeed equalise the expected background signal level, as shown in section 2.3.2, they cannot fundamentally change the signal-to-noise ratios across the images, which should be taken into account during the reconstruction of higher order cumulants.

I first examine the characteristics of neural data in section 3.3.2.1, whether it indeed includes detectable co-cumulants generated by single neurons of small spatial extents, as the CHOMP generative model suggests. Next, in section 3.3.2.2, I show that CHOMP is indeed capable of reconstructing both the mean signal and higher order cumulants, and these can be used in localising single neural cell bodies. Finally, I discuss the results of CHOMP on the Neurofinder challenge in section 3.3.2.3.

[13] Note that I could evaluate the group lasso method only on smaller sample of $n = 600 \ll 10000$ due to prohibitively slow running times.

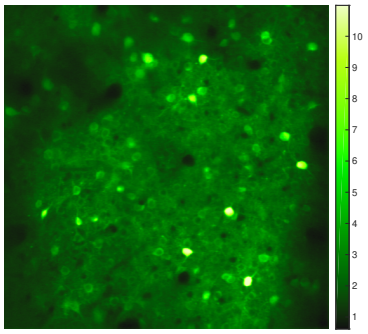
3.3.2.1 Higher order co-cumulants are present in the data

For a detailed example on what higher order cumulants and joint cumulants are present in the data, I will use the 01.00 dataset, that contains clearly overlapping cells. Therefore, first in figure 3.5, I characterise how the dataset was transformed using the preprocessing methods described in chapter 2 to achieve uniform background fluorescence over the field of view. Furthermore, in figure 3.5b and e, one may see that there is indeed structured pixel-wise variance present, with visible shapes clearly matching cell-like objects in the mean image. Similar structures are present in the standardised 3rd and 4th pixel-wise univariate cumulants – skewness and kurtosis – as well, shown in figure 3.6. These pixel-wise higher order cumulants I will call *diagonal cumulants*, as they represent the diagonal of the joint cumulant tensors.

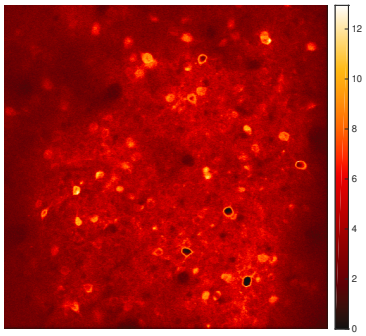
Although these diagonal higher order cumulants already show cell-like spatial structures, and thus may be beneficial to use as features during segmentation, the reason for the spatial structure in these pixel-wise statistics is the underlying Poisson distribution generating the signals: The variance, skewness and kurtosis are all directly related to the (instantaneous) mean. Therefore, the main idea of using higher order cumulants contains representing how activities of pixels vary jointly, using the joint cumulants of nearby pixels, rather than only the pixel-wise statistics. Such joint variations then in principle enable one to distinguish seemingly overlapping cells, and segment them correctly. To illustrate this, let us first zoom in on the mean and variance images, and see how covariance tensors may be visualised, in figure 3.7. The ultimate difficulty that CHOMP attempts to solve, but is hard for humans to interpret, is to retain a sense of spatial closeness of pixels, while still representing their joint cumulants. CHOMP achieves this by using higher order tensors; for example, the covariances of a patch of pixels in $D = 2$ dimensional data are a $D \times D = 4$-th order tensor, whereas their joint kurtosis could only possibly be represented by a $D \times D \times D \times D = 16$-th order one. Although as discussed earlier, these tensors are symmetric, with few unique elements, that doesn't help understanding what they represent, or to visualise them. Visualisation is best done in 2D via flattening the tensors[14], and getting used to the interpretation. As we are more familiar with covariance matrices over vectors, which do retain spatial closeness, I first show in figure 3.7 (e-h) the covariance matrix along a single vertical line. Firstly, (g) shows that the covariance matrix has a strong diagonal, due to the independent Poisson noise for each pixel; and although off-diagonal structure is present, it is difficult to interpret. In order to visualise the cell-induced covariances, I indicate the presumed cell locations along the line in (f), and show a low-pass Gaussian-filtered ($\sigma_{\text{filt}} = 1.2$ pixels) view of the covariance matrix, from which the diagonal was removed before filtering, and treated as missing data in the visualisation. The two overlapping cells now indeed have visible and distinguishable covariance structures, despite the high degree of overlap, and also illustrate the reason we decided to use cumulants. The overlapping area has clearly higher covariances than the individual cells, but

[14] In this chapter all flattening was carried out in Matlab ®, therefore using Fortran-style (column first) reshaping.

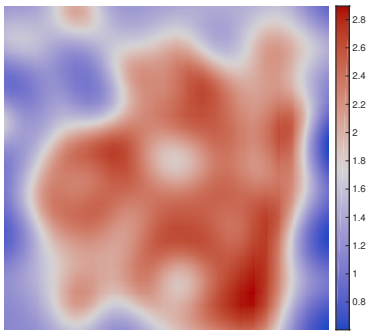
Figure 3.5: How spatial correction affects mean and variance of dataset 01.00. Although the background mean signal is indeed correctly equalised, as shown by the difference between (a) and (d), the signal-to-noise ratio does not change by divisive normalisation, and thus originally darker areas - generally with lower signal to noise ratios - now appear to have higher pixel-wise variance in (e), compared to (b). The numeric differences are caused by (a) and (b) representing actual photon flux estimates, whereas (d) and (e) have been stretched, such that the integer dataset utilises the uint16 range optimally; the stretch factor was 3747.



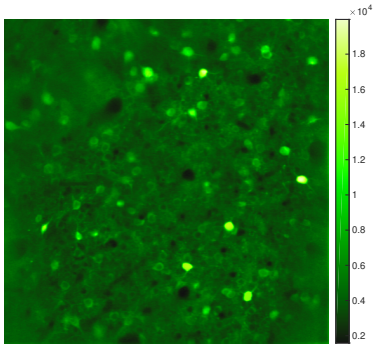
(a) Mean over time of photon fluxes estimated at each pixel and frame.



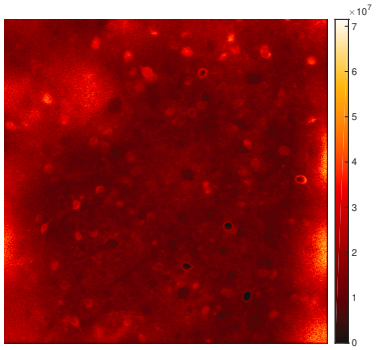
(b) Variance over time of photon fluxes estimated at each pixel and frame.



(c) Spatial gain non-uniformity, used for correction.

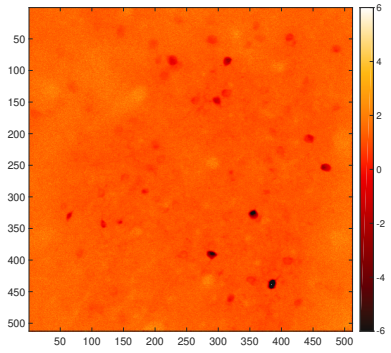


(d) Mean over time of photon fluxes estimated at each pixel and frame and corrected for spatial non-unformity.

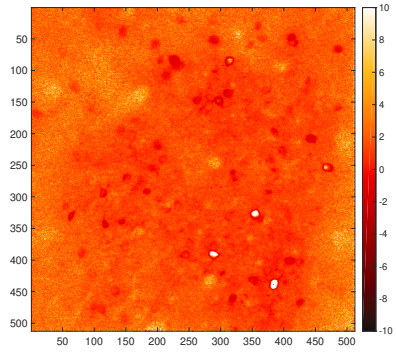


(e) Variance over time of photon fluxes estimated at each pixel and frame and corrected for spatial non-unformity.

Figure 3.6: How spatial correction affects skewness and kurtosis of dataset 01.00. Both images represent standardised cumulants, that is the raw n-th cumulant divided by σ^n, where σ is the standard deviation of the same pixel. Standardised cumulants are unaffected by spatial gain correction.



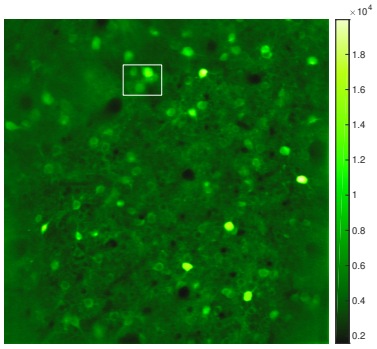
(a) Standardised 3rd cumulant (skewness) over time of photon fluxes estimated at each pixel and frame.



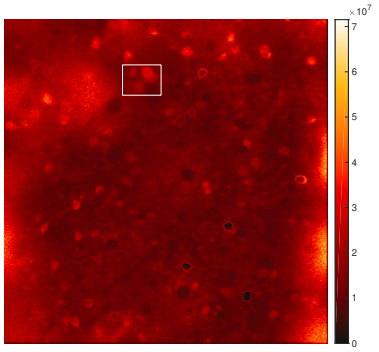
(b) Standardised 4th cumulant (kurtosis) over time of photon fluxes estimated at each pixel and frame.

covariances are additive, and thus ideal for matching pursuit type iterative reconstruction, such as implemented by CHOMP. Others have also recognised the importance of such off-diagonal joint variation in cell segmentation applications, using features like cross-correlation, to find co-varying pixels (Smith and Häusser 2010). I do believe other metrics are generally less robust due to their lack of additivity, and would advise the use of joint cumulants for separation of additive overlapping sources.

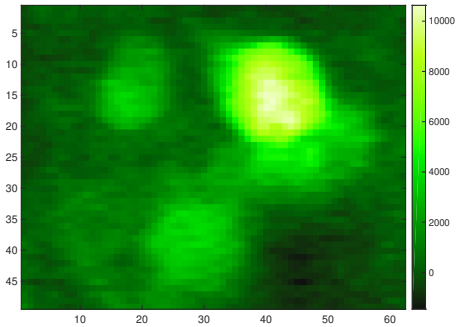
Figure 3.7: Covariances are present in the data. The mean and variance images in (c)-(d) are the patch indicated in (a) and (b). (e)-(h) illustrate covariances along a vertical line, (i)-(l) along a horizontal line, and (n)-(q) in a small patch. (m) shows spatial autocorrelation.



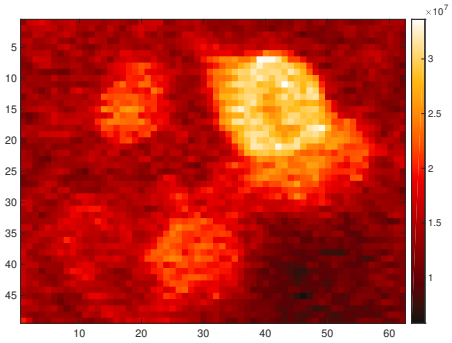
(a) Mean over time of photon fluxes estimated at each pixel and frame.



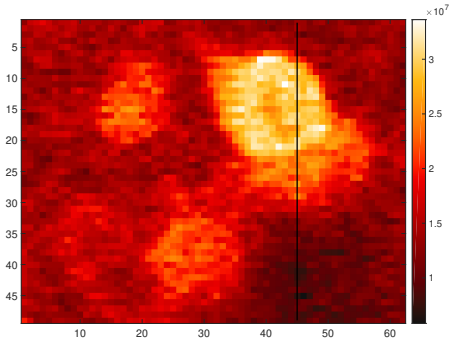
(b) Variance over time of photon fluxes estimated at each pixel and frame.



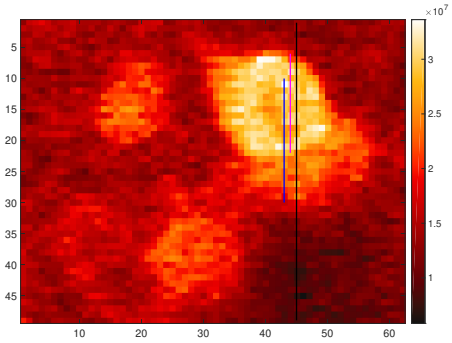
(c) Mean in a smaller area.



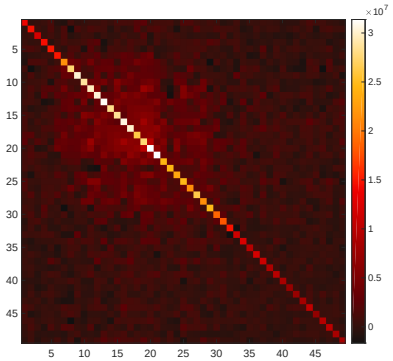
(d) Variance in a smaller area.



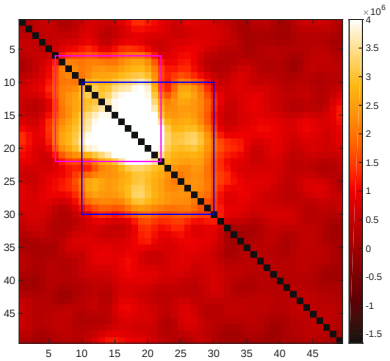
(e) Black line indicates selected pixels for (g).



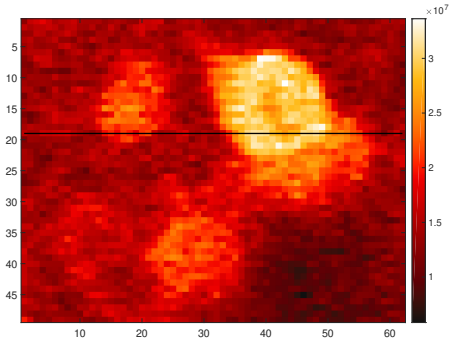
(f) Colored lines indicate highlighted regions in (h).



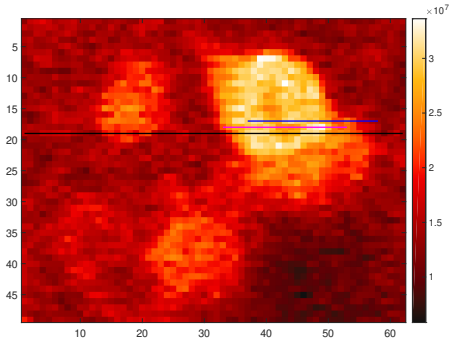
(g) Covariance matrix of the selected pixels, diagonally dominant and noisy.



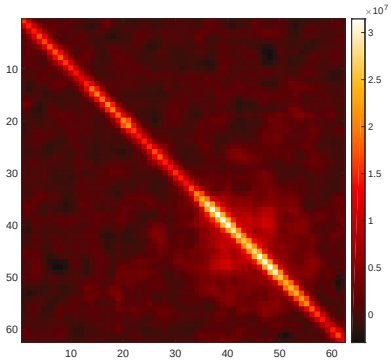
(h) Smoothed covariance matrix with diagonal removed, cell covariances are indicated.



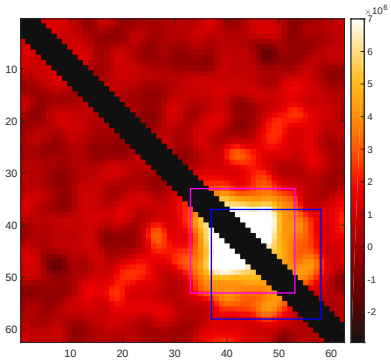
(i) Black line indicates selected pixels for (k).



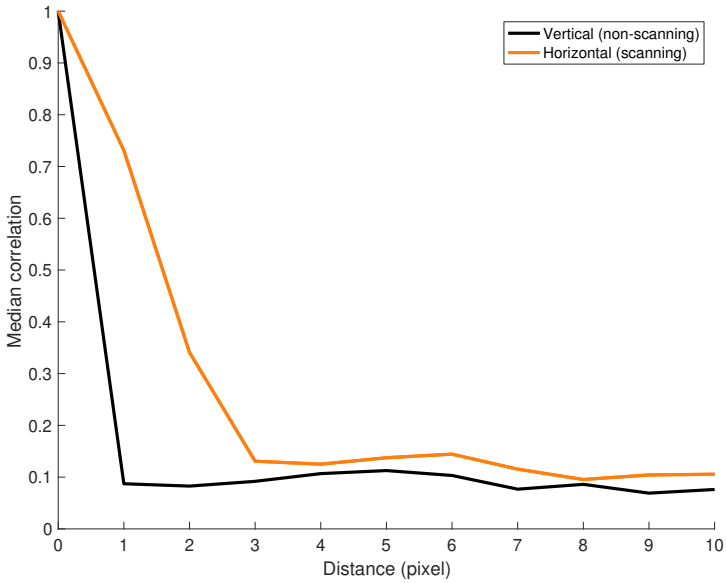
(j) Colored lines indicate highlighted regions in (l).



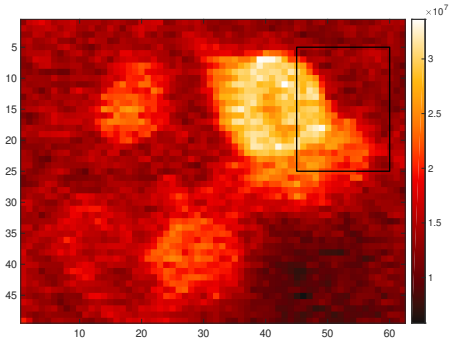
(k) Covariance matrix of the selected pixels, shows spatial autocorrelation (multiple strong diagonals).



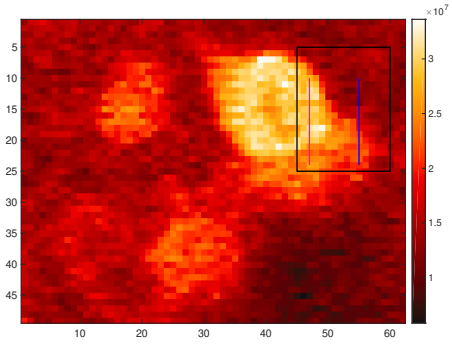
(l) Smoothed covariance matrix with diagonals removed, cell covariances are indicated.



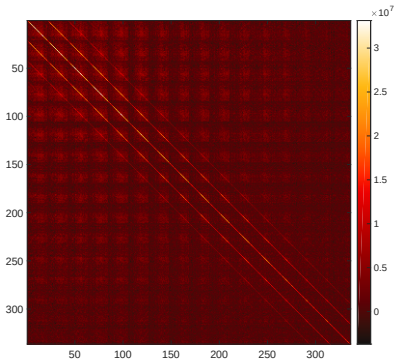
(m) Median correlation of pixel activities decays with distance. In the scanning direction we can see significant spatial auto-correlation, caused essentially by temporal autocorrelation of the two-photon imaging setup. In the non-scanning (vertical) direction, autocorrelation is completely absent.



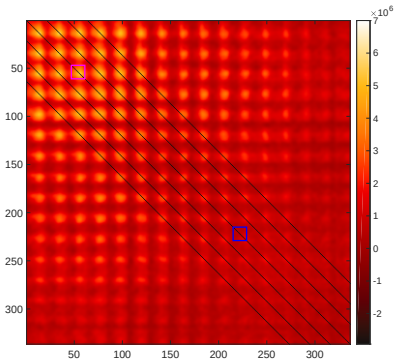
(n) Black box indicates selected pixels for (p). All pixels within the box contribute.



(o) Colored lines representing pixels, whose covariances are highlighted in (q) as blocks.



(p) Covariance matrix of the selected pixels. The blocked structure arises from the fact that this covariance matrix no longer retains spatial closeness in the horizontal direction. This is well illustrated by the fact that the strong off-diagonals (the results of scanning autocorrelation) are offset by 21 pixels – the height of the box in (n).



(q) Smoothed covariance matrix with diagonals removed. The highlight boxes indicate the small covariance boxes for the pixels sub-selected in (o), but also note the co-covariance blocks between those sets of pixels, due to both lines overlapping with the rightmost cell.

Regardless of metric, one must be careful about potentially unexpected artefacts in the data, which is introduced by the experimental system. One such common artefact, that strongly corrupts our mental model of cell-induced covariances, is shown in figure 3.7 (i-l). By simply examining the covariance matrix along a horizontal direction, we find that instead of a single strong diagonal representing independent pixel variances, there are multiple strong diagonals in (k), indicating spatial auto-covariance. The fact that this only shows up in the horizontal direction, but not the vertical one, is a strong indicator that this is in fact not a spatial effect, but rather the temporal auto-correlation induced by the two-photon microscope system, strictly appearing in the scanning direction. In order to discover truly cell-induced covariance structures in the scanning direction, we thus need to estimate this autocorrelation length (m), and remove the appropriate diagonals to reveal the structure of off-diagonal covariances (j,l).

Thus far, we did not need to worry about retaining the sense of spatial closeness, as we were

examining covariances along a single line. My last example highlights the difficulty of visualisation and interpretation in the realistic case, in which one attempts to reveal and reconstruct covariances within an image patch. In figure 3.7 (n-q) I first indicate a patch of pixels and its covariance matrix in (n,p), revealing a blocked structure due to the flattening process disturbing the sense of spatial closeness in the horizontal direction. This is well illustrated by the fact that the strong off-diagonals – the results of scanning autocorrelation – are offset by 21 pixels, which is the height of the selected box in (n). Lastly, to aid with the understanding of this blocked structure, two coloured vertical lines are indicated in (o), whose covariance blocks are shown in (q). Note the existing cross-covariances between the two lines of pixels (found at the off-diagonal intersection of coordinates of individual blocks), which is likely induced by the rightmost cell contributing signal to both short lines of pixels.

Finally, as an illustration, that such co-cumulant tensors may be computed for higher order cumulants as well, the resulting co-cumulants of different orders are shown in figure 3.8 for 5 pixels along the vertical line shown in figure 3.7e. It is difficult to see structure in this by eye, it merely demonstrates the exponential growth of tensor size. This is the reason why CHOMP never explicitly represents these co-cumulant tensors, merely their projections onto the basis functions, and even then it exploits the tensor symmetricity to reduce both storage and computation costs; without these reductions, the algorithm would be significantly slower and potentially impossible to run without using out-of-memory storage.

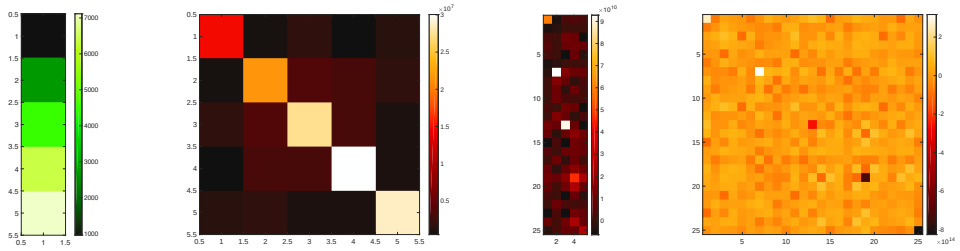


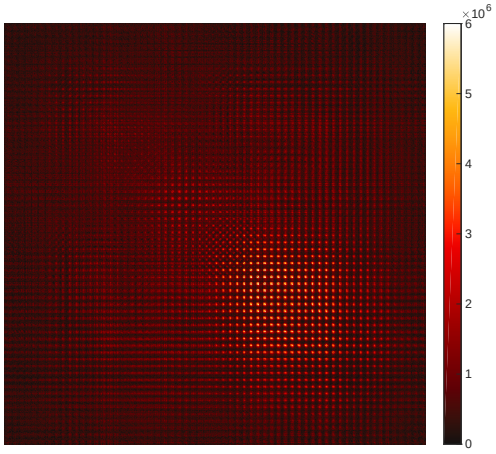
Figure 3.8: Illustration of co-cumulants of different orders. From left to right we see flattenned representations of 5 pixels' co-cumulants in orders 1 through 4. Although structure is present, it is progressively more difficult to understand it as the tensor order is increasing.

3.3.2.2 CHOMP reconstructs higher order co-cumulants

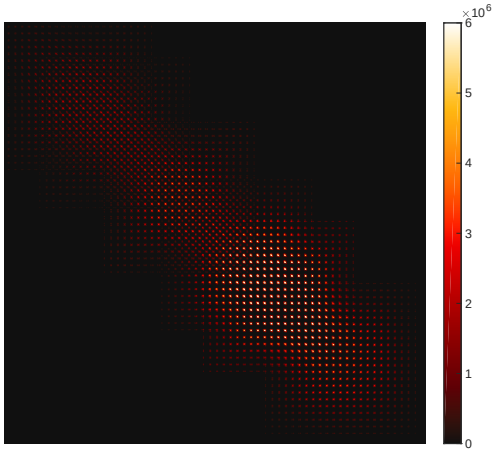
Given that the data indeed contains interesting higher order structure, I now show that CHOMP is capable of reconstructing such structure, and thus these features can be used to localise regions of interest that resemble active neurons. In figure 3.9, I first show the full flattened covariance tensor (a) of the zoomed-in region discussed in figure 3.7, and display the reconstruction achieved by CHOMP (b), as well as the residual covariance (c). Note, that to ensure the clarity for this example, CHOMP was ran in a pure off-diagonal covariance pursuit mode, ignoring all other cumulant orders (including

the mean) as well as the diagonal of the covariance matrix (highly correlated with the mean, due to the Poisson nature of the signal). The full area displayed is a 49x62 pixel region, whereas CHOMP was using a basis function size of 23x23 pixels, which results in the reconstruction being restricted to the block-diagonal subset of the full covariance matrix (b). The nature and quality of the reconstruction is difficult to judge in the 3038x3038 covariance matrix shown in (a-c), and therefore I show the reconstruction along the same horizontal and vertical lines of pixels as analysed above, in figure 3.9 (d-g) and (h-k), respectively. These indeed showcase that CHOMP recognised and reconstructed existing covariance structures, noting that the imperfect residuals result from model mismatch, and the fact that CHOMP did not only reconstruct these particular slices of the full covariance tensor, but attempts to minimise the residual over the complete tensor, as shown in (a-c).

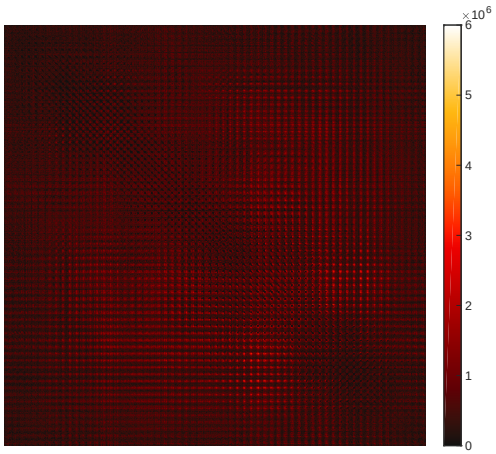
Figure 3.9: CHOMP reconstructs the full off-diagonal patch covariance. (a-c) show the full covariance for the whole zoomed region in figure 3.7 with the diagonals removed, but are difficult to interpret. Therefore (d-g) and (h-k) show more interpretable subsets of the CHOMP reconstruction, along the horizontal and vertical lines used before.



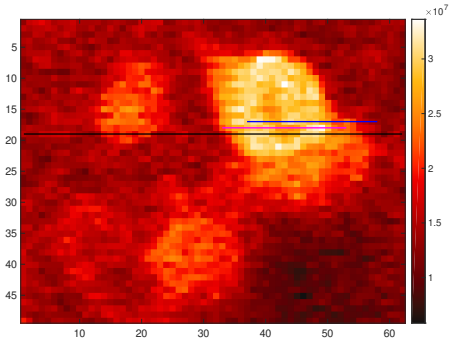
(a) Full flattened covariance tensor of the 49x62 pixel region shown in figure 3.7c-d.



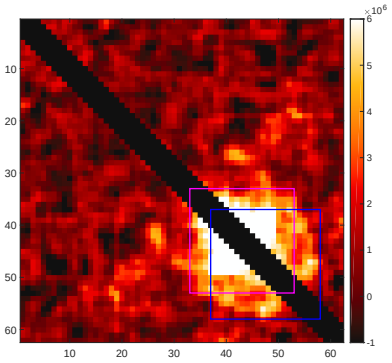
(b) Reconstruction of (a) by CHOMP, using 7 sources with a patch size of 23x23. The localisation and reconstruction was based on the off-diagonal covariances only, no diagonals or other cumulant orders were used.



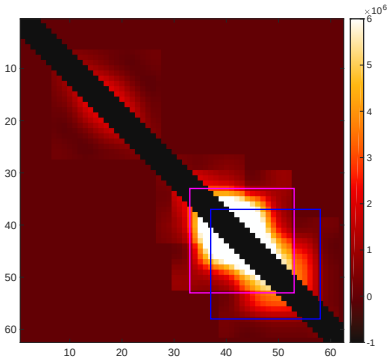
(c) The residual shows that covariance was indeed reduced.



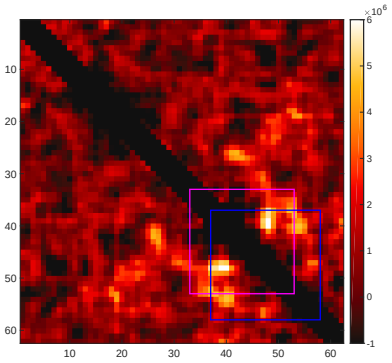
(d) Pixel-wise variance, with black line selecting pixels for the covariance matrix in (e) and colored lines indicate boxes in (e-g). Reproduction of figure 3.7j.



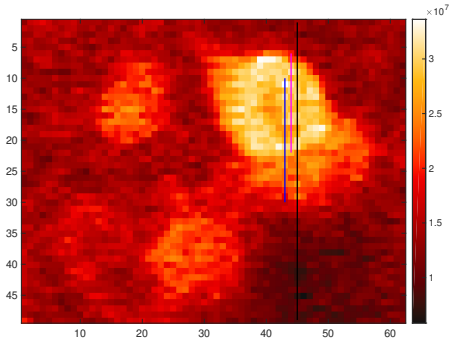
(e) Original covariance along line of pixels.



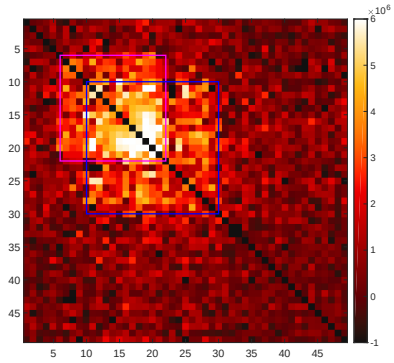
(f) CHOMP reconstruction of covariance.



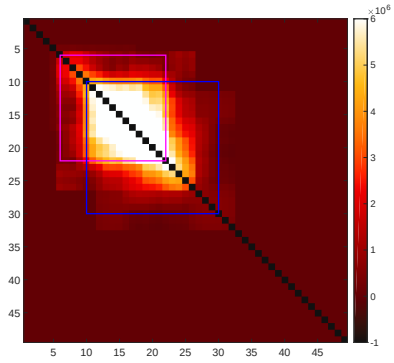
(g) Residual covariance.



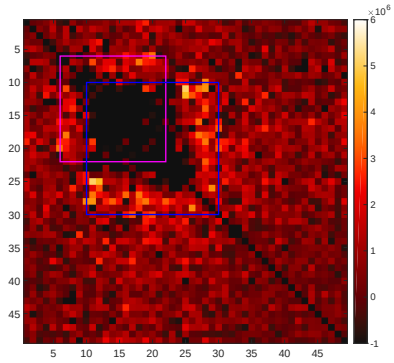
(h) Pixel-wise variance, with black line selecting pixels for the covariance matrix in (i) and colored lines indicate boxes in (i-k). Reproduction of figure 3.7f.



(i) Original covariance along line of pixels.



(j) CHOMP reconstruction of covariance.



(k) Residual covariance.

3.3.2.3 Results on the Neurofinder challenge

Finally, to show the applicability of preprocessing and CHOMP across numerous datasets, I discuss the results on the Neurofinder challenge. The main training datasets were introduced and thoroughly characterised in chapter 2, but the aim of the challenge was not. Neurofinder consist of 5 distinct sets of two-photon calcium imaging videos of various areas of the mouse brain, recorded by different labs from multiple distinct fields of view; see table 2.1 for a summary. The aim is to identify small regions of interest (ROIs) in the data, corresponding to individual neural cell bodies. Each field of view has been labelled by the challenge organisers, with the labels indicating the location and spatial extent of individual ROIs. The methods for generating these *Neurofinder labels* are unknown, but for some of the datasets they have been publicly released, to serve as training data for supervised algorithms, or as calibration and validation for others. For the remaining datasets the labels have been withheld, and are used as a test set to evaluate the performance of algorithms, by submitting sets of automatically identified binary ROIs for each field of view.

The performance is evaluated by the provided 'neurofinder evaluate' script[15]. It first computes the centre of mass for both Neurofinder-labelled and algorithm-identified ROIs. Next, it carries out a greedy matching procedure between the two sets of centres, matching nearest neighbours first, and up to a maximum distance of 5 pixels – noting that the general diameter of a cell in these datasets are 10-25 pixels, all fields of view are approximately 512x512 pixels, and contain several hundred cells according to the Neurofinder labels. After the nearest neighbour matching, two key metrics are computed: *recall* is the proportion of Neurofinder-labelled cells co-localised with algorithmically proposed ones, whereas *precision* is the proportion of algorithm-labelled cells co-localised with Neurofinder ones. The final, *combined*, performance metric is then computed as their harmonic mean:

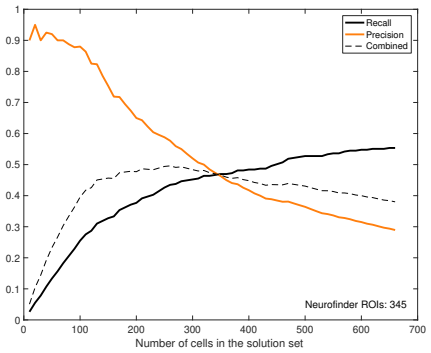
$$\text{combined} = 2 * \text{recall} * \text{precision} / (\text{recall} + \text{precision}), \tag{3.24}$$

and the sum of combined performances across test fields of view is used for ranking the submitted

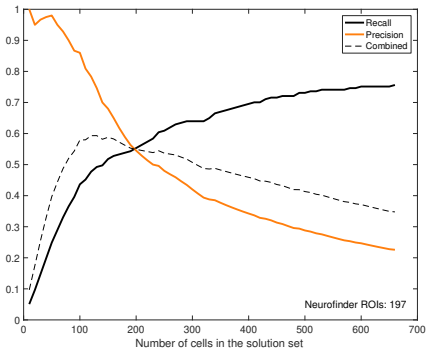
[15] `https://github.com/codeneuro/neurofinder-python/blob/master/neurofinder/commands/evaluate.py`

Figure 3.10: CHOMP Neurofinder test results, showing key metrics on all held out test datasets.

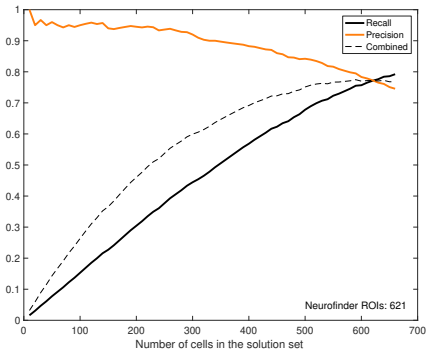
Figure 3.11: CHOMP Neurofinder training results; showing the changes in Neurofinder performance metrics when selecting the first N cells from the CHOMP solution, for increasing values of N. The number of Neurofinder-labelled ROIs are shown on the plots, and due to the metric chosen it coincides with the crossing of the 3 curves. Dataset 00.00 is discussed in more detail later, its training results are shown in figure 3.12f.



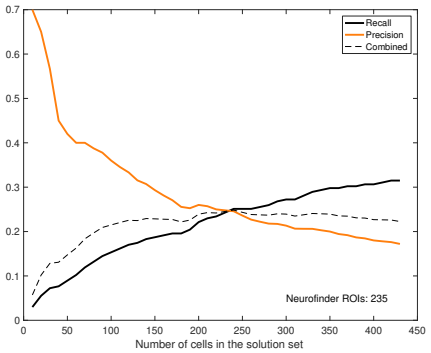
(a) Dataset 01.00



(b) Dataset 02.00



(c) Dataset 03.00



(d) Dataset 04.00

algorithms, collected on the challenge website[16]. The achieved performance metrics of my proposed algorithm on the test datasets are shown in figure 3.10.

CHOMP, being a greedy method, naturally ranks the identified ROIs according to how much they contribute to the reduction of the cost function (see equation 3.18). The number of submitted ROIs for each test dataset were chosen such that combined performance on the corresponding training dataset was maximised. As the Neurofinder labels are available for the training datasets, we can in fact trace the changes of performance metrics as we take the first N cells from the CHOMP solution set for various values of N, see figure 3.11. Generally, the first locations identified by CHOMP agree with the Neurofinder labels, resulting in high precision initially. Later on, as CHOMP identifies more regions of interest with significant residuals to reconstruct, they are less likely to agree with the Neurofinder labels. The recall however does keep increasing with added sources[17], al-

[16] `http://neurofinder.codeneuro.org/`

[17] This is partly a random effect. As discussed in section 3.3.1, adding an extra source, even at a random location always increases the recall chance.

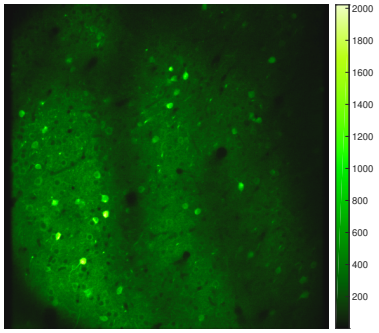
though it does not reach 1, meaning CHOMP does not identify some of the cells that are labelled by the Neurofinder challenge.

As a more detailed example, I showcase the Neurofinder and CHOMP labels of dataset 00.00, examine both false positives and false negatives (taking the Neurofinder labels as the ground truth for now), and discuss the reasons for the discrepancies. Both sets of labels are shown in in figure 3.12c, with one obvious difference arising from the preprocessing revealing cells in darker regions (a-b,e). Another difference is that although the cardinalities of the label sets shown in (c-e) are equal in the full field of view, CHOMP is capable of suggesting a significantly larger number of regions of interest (f-h). Of course adding sources not labelled by Neurofinder decreases the precision metric (f,h), and even then CHOMP misses some of the Neurofinder-labelled ROIs (g).

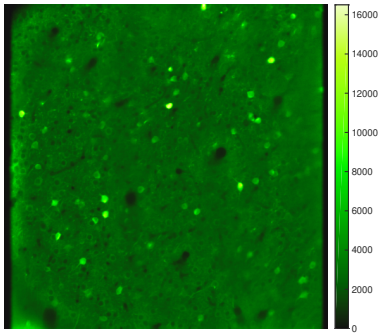
In figure 3.13, I thus investigate these differences by looking at the activity time courses of the CHOMP-identified and missed ROIs. In (a-d) I show the locations and activity – extracted from the preprocessed dataset via binary masks – of the top 10 CHOMP-identified ROIs in the two regions, showing that most indeed have cell-like activity, even those not amongst the Neurofinder ROIs (b,d cells 2-10). A potential explanation as to what enables the identification of these extra ROIs, is that preprocessing equalises the noise and signal levels across the field of view; see the difference between (e) and (f) compared to (c) and (d). Furthermore, CHOMP attempts to find cell shapes and corresponding cumulants that are typical across the field of view, enabling it to pinpoint co-varying regions of restricted shapes that may have little activity (as discussed earlier, see figure 3.9), but also limiting its ability to reconstruct less typical cells. Although for highly active cells this generally is not a problem, as enough of the activity falls into the reconstructable subspace, which drives CHOMP's localisation, this model mismatch does affect the localisation of nearby, overlapping regions of interest, that are less active. The greedy algorithm first reconstructs the active cell, somewhat incorrectly, and then the remaining residual of nearby cells does not resemble cell-like activity anymore. This problem is shown in figure 3.13g-h, where the activity of Neurofinder-labelled ROIs are shown, showing examples of strangely shaped active and nearby inactive regions (cells $\{8,11,12\}$, $\{2,22,24\}$ and $\{1,4,15,19\}$), or where the labelled regions are largely inactive (cells 12, 24, 15, 26).

Lastly, to further investigate the reasons for discrepancy between the Neurofinder-labelled ROIs and those identified algorithmically based on the cell activity, I compare CHOMP to the currently most popular activity-based neural segmentation pipeline, Suite2p (Pachitariu et al. 2016). Suite2p's segmentation relies on an EM-style algorithm, iteratively assigning pixels to ROIs in the E-step and estimating the activity of each ROI in the M-step; both the cost function and the approximation differ from CHOMP's approach. Although Suite2p's ROI extraction algorithm does not naturally offer an ordering of the proposed ROIs as CHOMP's greedy algorithm does, nevertheless it applies a post-processing step: a classifier that determines for each ROI its probability of being a cell – this is the ordering used in the following plots, when comparing the proposed ROIs from the two algorithms to

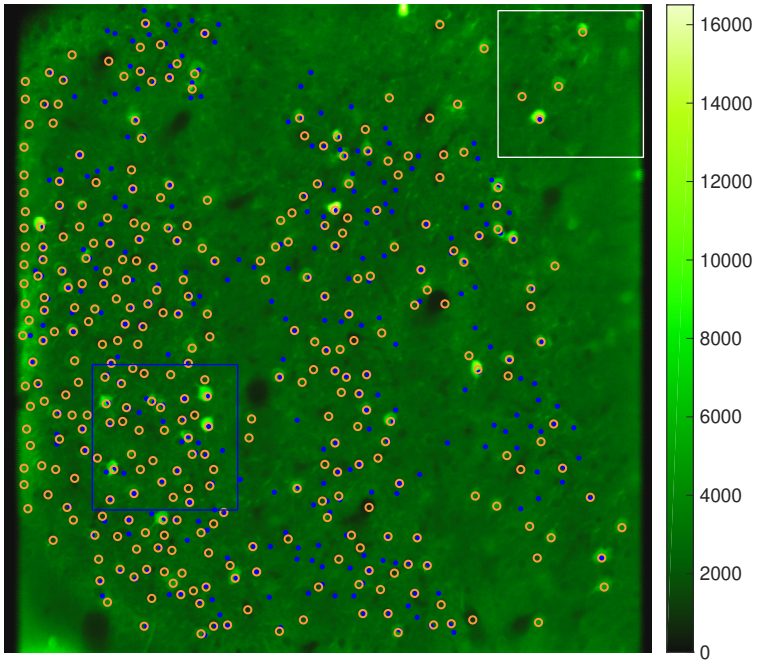
Figure 3.12: CHOMP results on dataset 00.00.



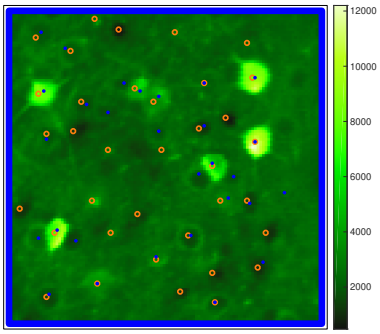
(a) Original mean image



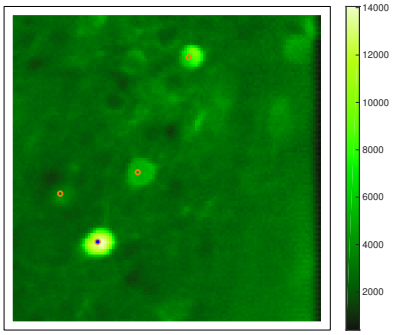
(b) Preprocessed mean image



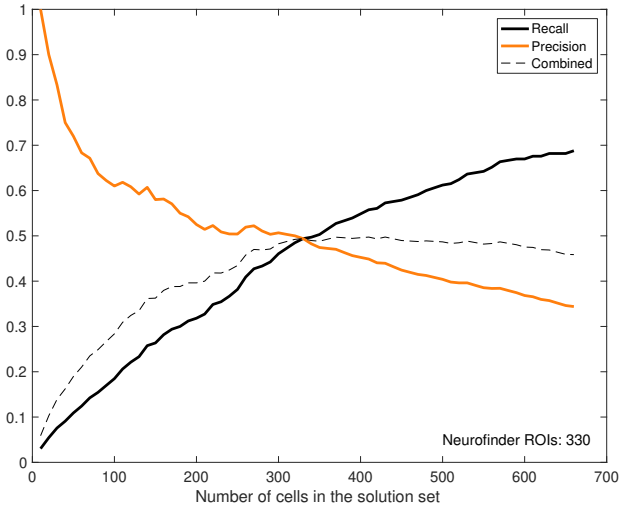
(c) Preprocessed mean image. Neurofinder-labelled (blue) and CHOMP-identified (orange) cell locations are superimposed. The markers represent the centre of mass of individual binary region of interests. The 330 ground truth locations, and the first 330 CHOMP identified locations are shown.



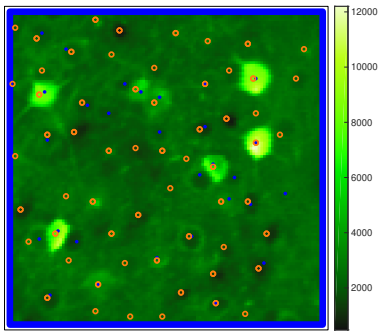
(d) Zoomed in block, indicated by blue outline in (c).



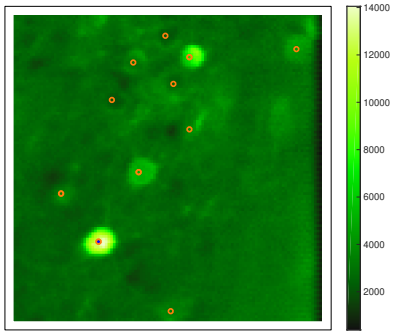
(e) Zoomed in block, indicated by white outline in (c).



(f) Precision-recall curve calculated by the external neurofinder script, given the 330 ground truth locations and the first N locations identified by CHOMP.



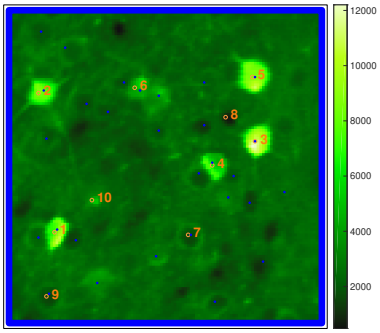
(g) As in (d), but using double the number of cells in the CHOMP solution set.



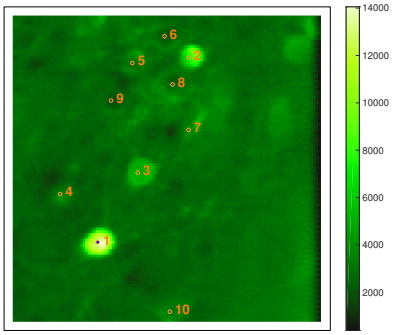
(h) As in (e), but using double the number of cells in the CHOMP solution set.

each other, as well as to the Neurofinder labelled ones. Briefly, Suite2p (version 0.6.16) was installed via pip, then ran on the raw Neurofinder data with largely default settings, except registration was omitted due to the input data being pre-registered. Furthermore, the 'sparse˙mode' option was set to True, as per explicit instructions from the author. This resulted in a total of 166-4686 ROIs proposed by Suite2p in the 5 datasets, although some with low confidence values according to its built-in classifier – the number of ROIs passing Suite2p's default cell-likeness threshold is indicated on the plots. Figure 3.14 shows that although both algorithms first identify Neurofinder-labelled ROIs with high confidence (as indicated by high precision values early on), these sets of ROIs proposed by the two algorithms differ substantially from one another (as shown via the initially low co-localisation). Furthermore, they both also suggest numerous ROIs that are not part of the originally labelled set (indicated by the subsequent decline in precision), but these extra ROIs are not necessarily the same for the two algorithms (as co-localisation never reaches 1). To investigate the types of differences

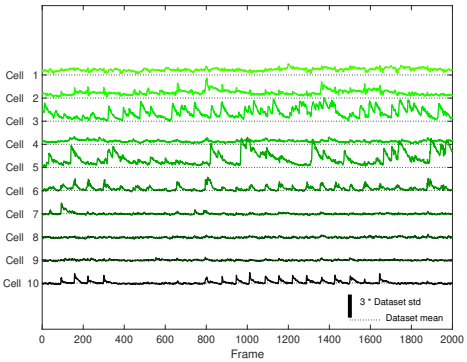
Figure 3.13: Discussing CHOMP results on dataset 00.00. CHOMP identifies additional ROIs that have cell-like activity (b,d). The preprocessing makes signals more comparable across the field of view (a-f), revealing these additional cells (d,f). CHOMP misses some inactive ROIs close to highly active ones (g-h). Supplemental videos are available for the zoomed in areas in (a) and (b), see appendix A for details.



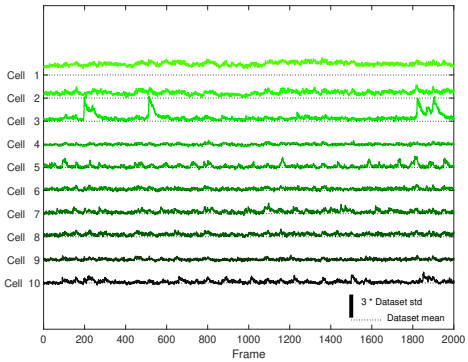
(a) As in figure 3.12d, but showing only the top 10 ROIs suggested by CHOMP.



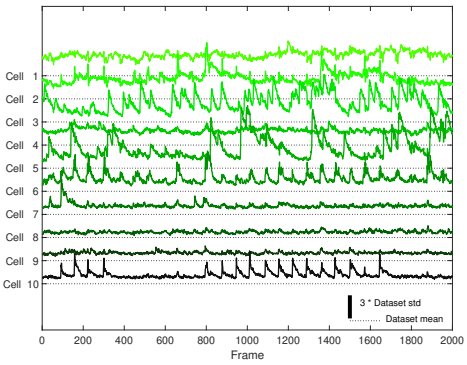
(b) As in figure 3.12e, but showing only the top 10 ROIs suggested by CHOMP.



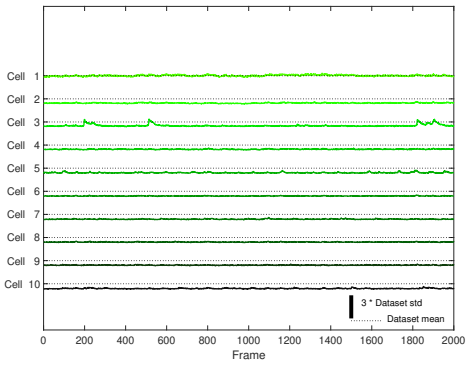
(c) Activity time courses from the 10 numbered ROIs in (a), identified by CHOMP.



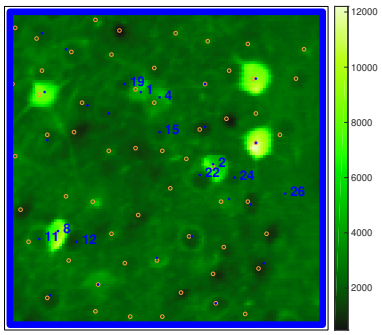
(d) Activity time courses from the 10 numbered ROIs in (b), identified by CHOMP.



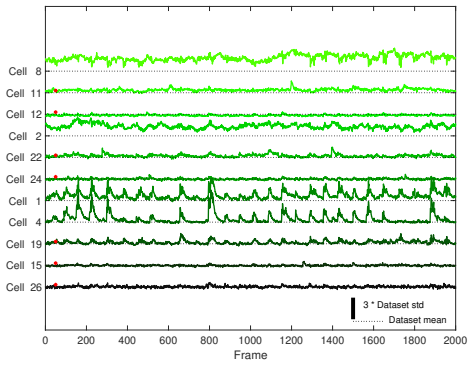
(e) As in (c), but extracting from the original dataset instead of the preprocessed one.



(f) As in (d), but extracting from the original dataset instead of the preprocessed one.



(g) Numbering Neurofinder ROIs that were missed by CHOMP, or are near the missed ROIs.

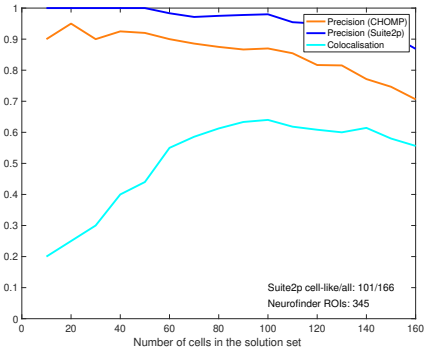


(h) Time courses of ROIs in (g), with the red asterisk indicating cells that were not found by CHOMP.

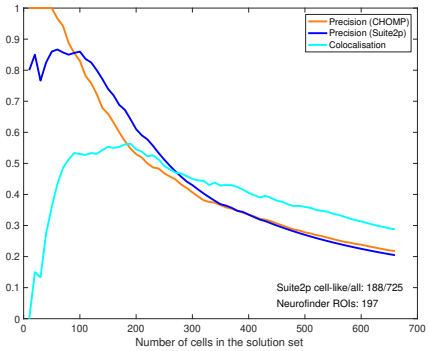
further, I compare and contrast the set of labelled ROIs at various numbers of proposed ROIs for both algorithms in figure 3.15. As evidenced in (a) and (c), both algorithms produce good co-localisation with the Neurofinder-labelled ROIs, but not with each other, meaning they focus on different features. With the default parameter settings of both algorithms it seems Suite2p strongly prioritises sparse activity, and fails to label even very bright cells (e,g); whereas CHOMP labels only bright regions first and foremost (c,e,g). When the 'correct' number of ROIs is supplied (b), both algorithms show reduced co-localisation with the ground truth labels – see precision curves in (a) – but the co-localisation of the algorithmic ROIs have increased (a,b), suggesting that they indeed find novel regions of interest that weren't part of the original Neurofinder-supplied label set, yet are agreed upon by two substantially differing algorithms. This is exemplified even stronger when taking into account the algorithms' capabilities of suggesting more ROIs, as shown in (d,f,h), especially the three co-localised novel candidate ROIs in (h). Examining (f) and (h) in more detail suggests that when asked to propose a large number of ROIs, Suite2p and CHOMP differ substantially in their approach. Overall, due its strong spatial priors, CHOMP identifies only larger regions of interest, thus spaced away from one another; conversely Suite2p uses the increased degrees of freedom to identify smaller ROIs closer to one another (f,h) largely typical of axonal or dendritic activity. Therefore CHOMP may be the better choice to look for cell bodies in single-cell approaches, whereas Suite2p is better *suited* towards discovering the total information encoded in the calcium imaging recording, be it by cell bodies or neurites. That said, the best solution may be to exploit their differences and use both methods in tandem, and ROIs proposed by both represent strong candidates for real cell-like behaviour.

To conclude, CHOMP is capable of identifying a large number of potential regions of interest in the various datasets provided by the excellent Neurofinder resource. The correctness of the proposed regions is generally difficult to evaluate, without direct access to the particular brain region to confirm findings via electrophysiological recordings. This would of course be a huge experi-

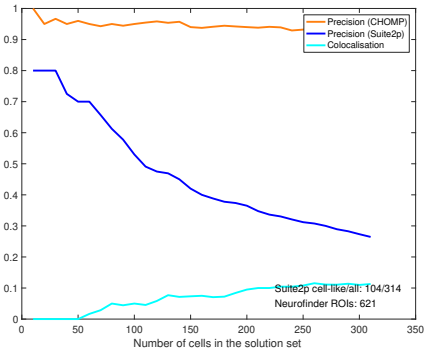
Figure 3.14: CHOMP comparison with Suite2p; the plots show the precision of each algorithm against the set of Neurofinder-labelled ROIs given the N most cell-like ROIs proposed by each algorithm independently, for various values of N. Furthermore, at each N value I show the fraction of co-localisation between the two algorithms. Interestingly, although both algorithms find largely Neurofinder-labelled ROIs first (at low values of N), they are honing on on different features first, as indicated by the initially low co-localisation (that is independent of the Neurofinder labels). Dataset 00.00 is discussed in more detail later, the results are shown in figure 3.15a.



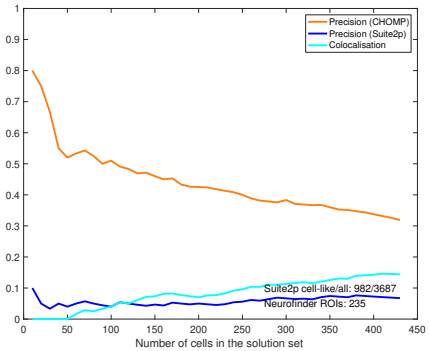
(a) Dataset 01.00



(b) Dataset 02.00



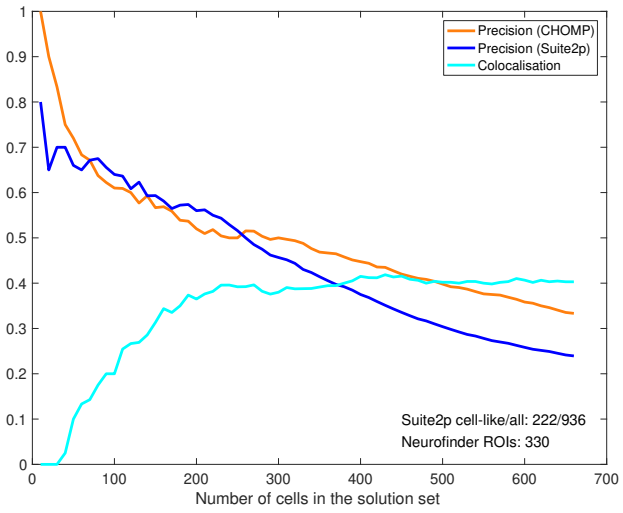
(c) Dataset 03.00



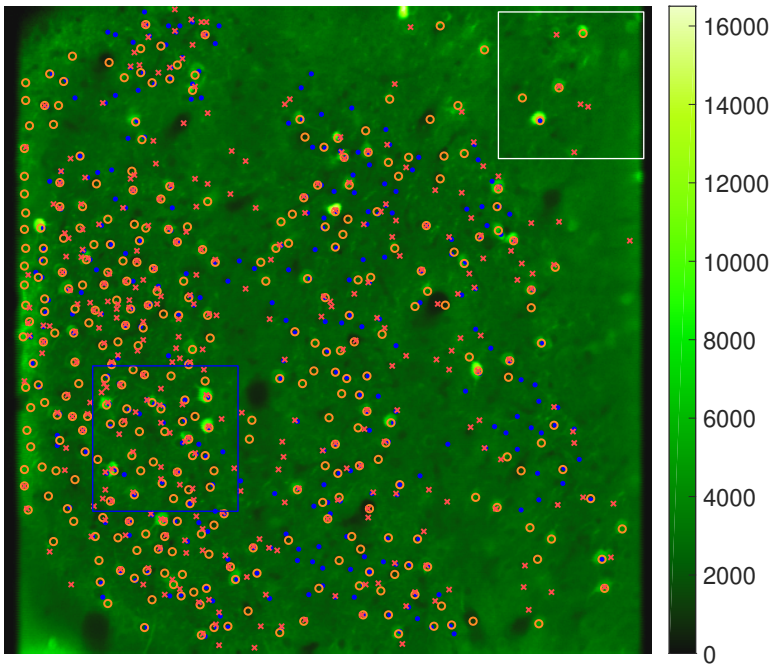
(d) Dataset 04.00

mental burden, with questionable usefulness, as the experimental conditions change across brain regions, animals, microscopes, labs and even people carrying out the experiment; thus even such clear validation would be of limited generalisability. Other techniques to achieve ground truth for cell localisation include multicolor imaging with nuclearly localised markers (e.g. Peron, Chen, and Svoboda (2015)), or matching the activity of extracted ROIs with electrophysiological data from the same area (Chen et al. 2013b; Schultz et al. 2009). Such methods are generally capable of identifying false negatives (ie. cells that are missed by segmentation algorithms such as CHOMP), although it is sometimes difficult to argue whether those missed cells simply display little-to-no activity in the calcium recording itself. Furthermore, with any method, it is extremely difficult to dismiss false positives, which would only potentially be possible via post-mortem tissue staining or electron microscopy reconstructions of the same area (Vishwanathan et al. 2017), that rely less on random effects such as viral uptake or gene expression levels in individual cells.

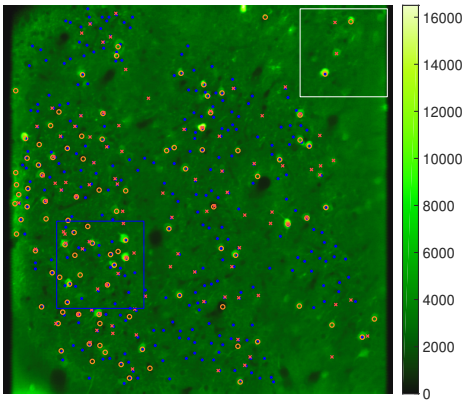
Figure 3.15: Comparison of CHOMP and Suite2p on dataset 00.00.



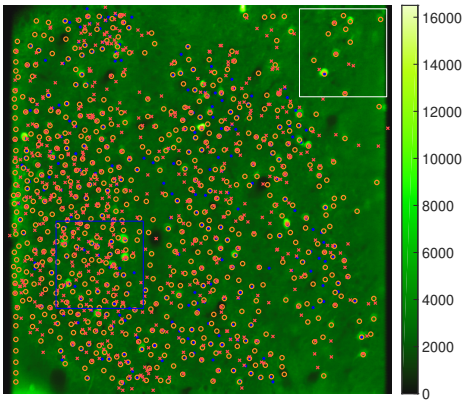
(a) Comparing the precision of CHOMP and Suite2p given the Neurofinder-labelled ROIs and increasing numbers of ROIs proposed by each algorithm, sorted by their internal confidence metrics. The co-localisation of the ROIs proposed by the two algorithms is shown for each number, demonstrating that CHOMP and Suite2p focus on different features.



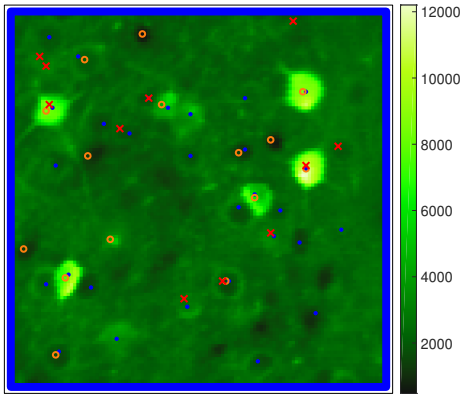
(b) Preprocessed mean image. Neurofinder-labelled (blue dot), CHOMP-identified (orange circle) and Suite2p-proposed (red cross) cell locations are superimposed. The markers represent the centre of mass of individual binary region of interests. In all subfigures the 330 Neurofinder-labelled ROIs are shown, however the number of ROIs shown from the two algorithms varies. Here 330 algorithmic ROIs are shown from each pipeline.



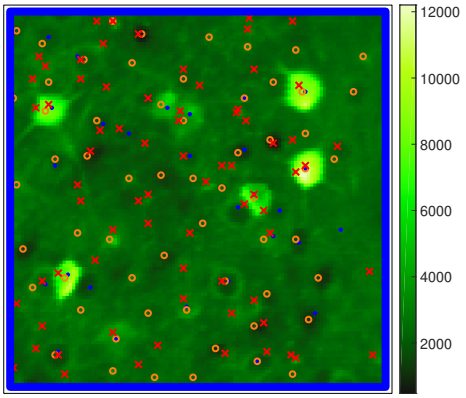
(c) As (b), but with only 100 algorithmic ROIs.



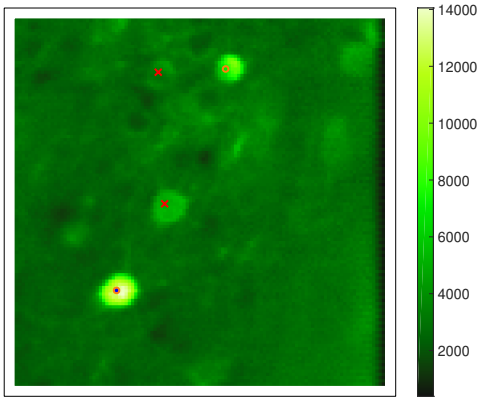
(d) As (b), but with 660 algorithmic ROIs.



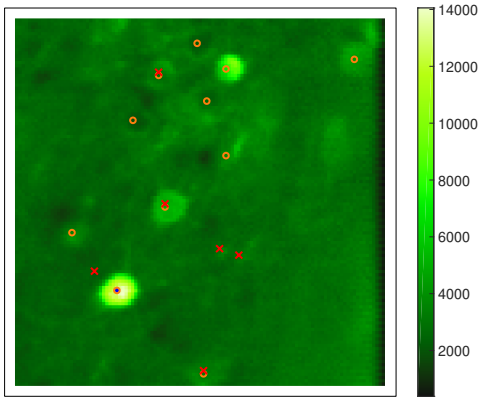
(e) Zoomed in block, indicated by blue outline in (c).



(f) Zoomed in block, indicated by blue outline in (d).



(g) Zoomed in block, indicated by white outline in (c).



(h) Zoomed in block, indicated by white outline in (d).

A potentially better way to analyse the performance of cell segmentation algorithms is to try and understand the usefulness of their output towards answering a particular scientific question – the one the recorded data was supposed to answer in the first place. Most often calcium imaging recordings are associated with both visual, auditory or tactile inputs provided or measured by the experimenter, and also behavioural outputs generated by the animal. Calcium imaging recordings serve as insight into how the brain represents and processes information, and are much more easily interpreted in context. Therefore, the performance of segmentation algorithms is also best analysed in such context, for example by understanding how correlated is the activity of individual ROIs with measured inputs or outputs, or how well the collection of all ROI activities represent latent computational variables, that we expect to find in the brain area. This latter may also provide a metric for the usefulness of individual ROIs, in context of all other ones within the same field of view, by analysing the experimentally relevant information gain associated with them.

3.4 Discussion

In this chapter, I introduced a novel algorithm, Convolutional Higher Order Matching Pursuit (CHOMP), that is capable of localising and segmenting regions of interest that correspond to single neural cell bodies in neural calcium imaging recordings. It achieves this by extending Convolutional Block Matching Pursuit, and adds the ability to represent joint higher order cumulants of small image patches. These joint cumulants are useful and tractable features in identifying neurons, due to their confined spatial extent, in which pixels are jointly influenced by the underlying cellular activity. Furthermore, the cumulant representation is well suited towards two-dimensional two-photon microscopy recordings, in which there are numerous apparent overlaps due to the imperfect optical sectioning. These overlaps contain summed signal from multiple sources, and thus cumulants – which are additive, unlike many other potential features, such as cross-correlation – are a correct representation of the typical single-plane two-photon calcium imaging microscopy recordings.

CHOMP, a matching pursuit based algorithm, proceeds to localise regions of interest by using basis functions with small spatial support, and iteratively selects a single local region, in which the current residual features can be best reconstructed. It is an extremely flexible algorithm, as its cost function can be tailored towards various expected noise characteristics, or be focused towards particular orders or parts of cumulants. In fact, it encompasses previous algorithms; for example Convolutional Block Matching Pursuit can be recovered by putting all reconstruction weight towards the mean, whereas reconstructing only the off-diagonal covariance essentially results in an additive version of cross-correlation based segmentation algorithms.

CHOMP of course possesses several limitations as well. Firstly, the greediness of the algorithm sometime prevents finding the optimal solution for nearby or overlapping sources. Although this is a recognised problem in convolutional matching pursuit algorithms, it is significantly reduced by

the use of full higher order co-cumulant tensors. Nevertheless, other similar algorithms often use a post-processing step that jointly solve for the best reconstruction within a small area, given a fixed number of sources; a difficult, but potentially tractable algorithmic improvement, that CHOMP is currently lacking. A more fundamental limitation is the finite cardinality dictionary of basis functions. These basis functions essentially describe allowed signal modes, which limit our definition of how signal from a neuron should appear in the recording. This is a desired feature in rejecting non-typical signals, and works well to that effect. The issue comes from the greedy iterative nature of CHOMP, where signals at identified locations are reconstructed, and the algorithm proceeds anew on the residual feature set. Due to the limited set of basis functions, the reconstruction is necessarily imperfect, and it sometimes results in a difficult-to-reconstruct residual, even in cases where the remaining residual indeed contains signal from another, overlapping cell. I believe a potential solution would be to use a constrained reconstruction in the style of non-negative matrix factorisation, in which both the basis functions and the reconstruction weights are constrained to be positive. This idea works well in the mean and variance, however it is less obvious, how to extend it to higher order cumulants, or off-diagonal elements; a potentially valuable avenue to pursue, merging two popular approaches of neural segmentation algorithms. Finally, it should be noted that higher order cumulants of two-photon microscopy data differ significantly from the typical application of matching pursuit – instead of the typically assumed iid. Gaussian noise, most of the noise is in fact Poisson distributed, due to the nature of the recording. This means that the diagonals – and sometimes even certain off-diagonals – of higher order cumulants are significantly correlated with the mean signal. As the current version of CHOMP was not in fact designed with correlations across different orders of cumulants in mind (that are generally assumed independent), I had the most success by combining the mean with only the off-diagonal elements of the higher order features, to avoid weighting high mean activity too highly. Of course the diagonal higher order cumulants also provide useful information about the mean, and may be used to get more stable estimates of it.

I applied CHOMP to the datasets published as part of the Neurofinder challenge, to demonstrate its capabilities. The datasets come from multiple brain areas, behavioural tasks, microscopes and labs, therefore they showcase the wide applicability of the algorithm. In fact, these datasets differ from one another in a number of important ways, as well as their is significant signal variation within the same field of view. These differences hurt CHOMP's – and likely other methods' – ability to identify regions of interest. This recognition lead to my development of the preprocessing algorithm described in chapter 2, which resulted in a more robust segmentation across the different datasets. These datasets also contain an existing segmentation for each field of view, created by the challenge organisers by undisclosed means. I evaluated the performance of CHOMP with the aid of both the publicly released as well as the withheld test label sets, and found that the cells highly weighted by CHOMP tend to agree with the provided segmentation. The publicly released labels helped me identify certain limitations of CHOMP, especially regarding false negatives, the labelled regions that

were missed by CHOMP. However, subsequent analysis revealed that a number of 'false positives' – regions of interest identified by preprocessing and CHOMP, but not amongst the provided labels – in fact may be active cells, suggesting that my proposed methods are capable of finding novel regions of interest, difficult to identify by previously agreed upon methods.

To summarise, I introduced a novel segmentation algorithm for neural calcium imaging recordings, showcased its capabilities on numerous datasets, and discussed its strength and weakness, suggesting ways of diminishing the latter. Convolutional Higher Order Matching Pursuit, or other algorithms building on its main ideas can be a useful addition to neural data analysis pipelines, and along with the previously proposed preprocessing method, they may result in a reproducible way of transforming raw calcium imaging recordings into standardised single-neuron activity time series, ultimately making data from different cells, fields of view or even labs more comparable, leading to more robust scientific conclusions.

4

Learning interpretable models of latent stochastic dynamical systems

Einstein's thinking is always on the ontological level traditional in physics; trying to describe the realities of Nature. Bohr's thinking is always on the epistemological level, describing not reality but only our information about reality.

– E. T. Jaynes on the Bohr-Einstein debates, *Probability in Quantum Theory*

4.1 Understanding the dynamics of neural recordings

When a neuroscience experiment is set up, we often aim to investigate the computation that a brain carries out in order to successfully solve an experimenter-defined task. Examples include probing premotor cortical areas for information relating to planned movement (Churchland et al. 2012) or looking for integration of evidence for decision making within the prefrontal cortex (Mante et al. 2013). Having recorded neural activity underlying the execution of the task, we seek to identify the 'computational variables' represented by the neurons, and thus to understand the process of the computation. However, most individual neurons appear to fire unreliably, and so these computational variables must be reflected in the collective patterns of many neurons. They are, in effect, 'latent' in the population activity, and so must be inferred. To ensure that our understanding of the system's complexity is well-captured by our inferred computational variables, we want them to explain much of the recorded signal; but we also want to ensure that they represent a stochastic, time-homogeneous Markovian state space - that is, given the current state of the system, the next state will not depend on the past history of states. This makes our set of recovered computational variables in some sense *complete*: such a set of variables enables us to reason about the individual system states, as well as paths through the state space, without having to consider non-(temporally)-local effects. Furthermore, it lets us investigate the dynamics of the state space, so that we can make probabilistic predictions of future states of the system and its representation within the recorded neurons.

Sketching the dynamical portrait of even noiseless, simple systems is no easy task. In order to glean insight into the noisy, subsampled data recorded from an extremely complicated system carrying out an experimenter-defined task, we need to make certain choices and compromises.

One approach investigators often take is to use a simple model to fit to available data, and examine how this simplified description changes over time, or as we vary experimental parameters. Examples include fitting linear dynamical systems to monkey evidence integration data, where the change in dynamics is parameterised by the context (Soldado Magraner 2018); or our work, in which the parameters of linear dynamics are allowed to slowly vary over time, via building a hierarchical model with a Gaussian Process prior (Park, Bohner, and Macke 2015). Although linear dynamics can be learned efficiently in closed form and their changes can be thoroughly interpreted (Soldado Magraner 2018), their use for explaining medium timescale evolution of systems is open to criticism, because such descriptions do not adequately capture the data itself; and one has to be very careful when interpreting results. However, allowing for linear dynamics' parameters to vary more rapidly in time (such as Duncker et al. (2017)), make their interpretations difficult.

An alternative, more recent approach has been to fit models with complex, non-linear Markovian dynamics[1]to multivariate time series data. Examples are latent recurrent neural networks (see e.g. LFADS by Pandarinath et al. (2018a) and Sussillo et al. (2016)) and Gaussian Process State Space Models (GPSSM, see e.g. Damianou, Titsias, and Lawrence (2011) and Eleftheriadis et al. (2017), without neural data applications). Although these methods are powerful at inferring the complex and noisy trajectories of the computational variables encoded by the recorded neural population activity, and they do contain an explicit description of the dynamical transition function, the results only really provide insight into 'what' the system does (following those inferred trajectories), rather than the 'how' it does it (via implementing interpretable dynamics that lead to those inferred trajectories). These problems have been recognised by others too, and they turned to the classic method of describing the learnt model dynamics by 'opening the black box', and finding the fixed points of the model (Golub and Sussillo 2018; Sussillo and Barak 2013).

The idea of utilising fixed points and locally linearised dynamics as human-interpretable descriptions of complex dynamical systems is indeed a powerful one, and has a long history in physics. Bifurcation theory concerns itself with identifying the number and type of fixed points in a system. Its application to deterministic systems is part of the standard curriculum, however identifying - or even defining - fixed points or attractors in stochastic dynamical system is in its early stages (Arnold and Crauel 1991; Diks and Wagener 2006; Wang, Chen, and Duan 2018), and thus random fixed

[1]There also seems to be a lot of confusion in the terminology of what people call a 'dynamics' or 'dynamical model'. In my definition, also stated as above, a latent dynamical system is described by not only a latent state that evolves over time, but I put a further requirement of having a well-defined Markovian state transition function as part of the model. In my view, this requirement disqualifies several methods that implement arbitrary non-Markovian time correlation structures, such as GPFA (see Duncker and Sahani (2018) and Yu et al. (2009)), including extensions such as Gaussian Process Latent Variable Models (e.g. Wu et al. (2017)) and Variational Latent Gaussian Processes (Zhao and Park 2016). Although these are powerful at inferring latent trajectories, they do not provide explicit descriptions of the latent dynamics, and depending on the GP kernel choices, the trajectories likely do not exhibit first-order Markovian dynamics.

point or stochastic bifurcation theory is not easily applicable to experimental data recorded from unknown systems. Therefore the approach used by the methods above (e.g. Golub and Sussillo (2018)) was to simulate trajectories from the learned systems (either noiseless, or averaging noisy ones), and find deterministic descriptions of the fixed points of the system.

I felt that this methodology - fitting a stochastic dynamical system to data, then finding fixed points using the model only - could be improved by directly including deterministic fixed point descriptions as parameters of the model, and fitting the observed data while simultaneously learning point estimates or posteriors over the fixed points and local linearisations. This directly results in a human-interpretable description of an unknown, noisy dynamical system, while ensuring that our uncertainty about the number, location or properties of the fixed points is propagated correctly, and taken into account as part of the model fit.

The main idea of conditioning the transition dynamics on fixed points is described in section 4.2; then I discuss the case when the transition dynamics function has a Gaussian Process prior, leading to a conditioned Gaussian Process State Space Model, in section 4.2.1. My first specific implementation – submitted to ICML 2018 (Bohner and Sahani 2018) and presented at COSYNE 2018 (Sahani 2018) – is described in section 4.2.2, whereas a subsequent, more flexible algorithm – the result of a collaboration with Lea Duncker and Julien Boussard and published at ICML 2019 (Duncker et al. 2019) – is shown in section 4.2.3. Finally, the two described algorithms are applied to simulated systems in section 4.3, to demonstrate their ability to both infer latent trajectories, and provide correct and interpretable descriptions of the dynamics given moderate datasets, including ones mimicking typical neural recordings in sections 4.3.4 and 4.3.5.

4.2 Modeling latent stochastic dynamics with fixed points

In order to build up towards the idea of finding the fixed points within dynamical maps fitted to repeated time-series recordings, we need to first describe the individual elements of a model that could implement such a task, then find a way of combining them together. I will discuss a very general model at first, and show the particular choices I made gradually - keeping in mind that there are many such choices, which all lead to a different specific algorithm, some potentially better or more broadly applicable than mine. My aim here was not to explore the complete space of all such potential algorithms, or argue my choices are the best possible ones; but rather to describe and share a working implementation of an idea that I hope will spark a novel way of looking into dynamical systems fit to neural recordings.

First, let us define the data we wish to model. It consists of repeated measurements (trials) of a time series

$$\mathbf{y}(t) \in \mathbb{R}^{\mathrm{N}}, \;\; t \in [1, T] \,, \tag{4.1}$$

where the output dimensionality N often corresponds to the number neurons recorded from, and T could be the number of video frames in calcium imaging, or time bins in spike sorted electrical recordings. In the simplest case, each trial will consist of the same number of neurons and time points, and thus the collection of trials may be described as either the set $\mathcal{Y} = \{\mathbf{y}^{\mathrm{m}}(t)\}_{\mathrm{m=1...M}}^{\mathrm{t=1...T}}$ or simply the tensor $\mathbf{Y}_{\mathrm{ntm}}$. Note that for the algorithm described here, we do not require the trials to have the same number of time points in trials, but it greatly simplifies the mathematical description. We do - for now - require the time points to be equally spaced in time (which is usually the case with microscopy and time-binned data), but in an extension described later – which was part of a collaborative effort to improve my initial model – that requirement will also be dropped (see section 4.2.3).

Our first modeling step involves dimensionality reduction of the time series data, by assuming there is a lower dimensional latent dynamical space, which is capable of capturing the interactions between the often large number of recorded neurons. Let us denote timeseries within this latent space as $\mathbf{x}(t) \in \mathbb{R}^{\mathrm{D}}$, where D is the latent dimensionality. Therefore the first modelling step of dimensionality reduction is

$$\mathbf{y}(t) = g(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\mathrm{y}}(t) \,, \tag{4.2}$$

where our goal is to infer latent time series $\mathbf{x}$ and learn an output mapping function $g : \mathbb{R}^{\mathrm{D}} \mapsto \mathbb{R}^{\mathrm{N}}$, such that the discrepancies $\boldsymbol{\varepsilon}^{\mathrm{y}}(t)$ are minimised (according to some metric and aggregation[2]).

Next, we wish to learn about the dynamical evaluation of the latent time series. At our current level of understanding most real biological process are 'inherently' noisy, therefore we need to describe our dynamical evaluation as a stochastic differential equation (SDE), $d\mathbf{x} = f'(\mathbf{x})\,dt + \sqrt{\boldsymbol{\Sigma}^{\mathrm{x}}}\,d\mathbf{w}$, where $\mathbf{w}(t)$ is a Wiener noise process. However, given we are assuming our data is on a fixed step time grid, we can use the significantly simpler language of stochastic state-space models, a discrete time analogue of SDEs. Therefore we can write

$$\mathbf{x}(t+1) = f(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\mathrm{x}}(t) \,, \tag{4.3}$$

where we need to learn a dynamical transition map $f : \mathbb{R}^{\mathrm{D}} \mapsto \mathbb{R}^{\mathrm{D}}$ and the level of transition noise $\boldsymbol{\varepsilon}^{\mathrm{x}}(t)$.

Finally, to enhance our insight into the dynamical process, we wish to explicitly model the locations of the fixed points in the transition map $f(\cdot)$ and the behaviour of the system around them through local linearisations. Let $\mathbf{s}^{\mathrm{l}} \in \mathbb{R}^{\mathrm{D}}$ be a fixed point in f, and $\mathbf{J}^{\mathrm{l}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$ the Jacobian matrix of f evaluated at that point $\left(\text{ie. } \mathbf{J}^{\mathrm{l}} = \nabla_{\mathrm{x}} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{s}^{\mathrm{l}}}\right)$. If we then choose a parametrisation of f that depends explicitly on these fixed points and local linearisations, we can learn these parameters as

[2] In the simplest case the squared error summed over all neurons, time points and trials, $\sum_{\mathrm{n,t,m}} \varepsilon^{\mathrm{y}}_{\mathrm{nm}}(t)^2$. Minimising this results in the optimal solution if we assume the observations have uniform isotropic Gaussian noise.

part of our model fitting process.

The full model is then

$$\begin{aligned} \mathbf{x}(t+1) &= f(\,\mathbf{x}(t)\,) + \boldsymbol{\varepsilon}^{\mathrm{x}}(t) \\ \mathbf{y}(t) &= g(\mathbf{x}(t)) + \boldsymbol{\varepsilon}^{\mathrm{y}}(t) \end{aligned} \tag{4.4}$$

$$\text{such that } \forall\, \mathbf{s}^{\mathrm{l}}, \mathbf{J}^{\mathrm{l}}$$

$$f(\mathbf{s}^{\mathrm{l}}) = \mathbf{s}^{\mathrm{l}} \text{ and } \nabla_{\mathrm{x}}\, f(\mathbf{x})\,|_{\mathbf{x}=\mathbf{s}^{\mathrm{l}}} = \mathbf{J}^{\mathrm{l}}\ .$$

In such a model, we succeed to fit the data, and gain direct scientific insight into the behaviour of a system that could have generated the observations, in a single step. What parameterisation of f may enable us to include the fixed point parameters and still fit datasets with various output and noise models? Recent work on Gaussian Process State Space Models (GP-SSMs) has shown great promise in modelling stochastic latent non-linear discrete time dynamical systems in a variety of applications (e.g. Frigola, Chen, and Rasmussen (2014) and McHutchon (2014)); furthermore, functions drawn from a Gaussian Process also lend themselves to be conditioned on their fixed points and derivatives. As it uniquely combines these strengths, I chose to use a Gaussian Process parameterisation for the transition map function f. This choice informs both the inference and learning steps of any algorithm, and thus will now be discussed in detail.

4.2.1 A Gaussian Process prior with fixed point parameters

Let the D-valued transition map function be a collection of D independent scalar-valued functions, $f(\cdot) = [\,f^1(\cdot), f^2(\cdot), \ldots, f^{\mathrm{D}}(\cdot)\,]$, a common technique when representing multiple-output functions via Gaussian Processes.

A Gaussian Process prior over a scalar-valued function $f^{\mathrm{d}}(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ with given mean and covariance functions, $m(\cdot) : \mathbb{R}^{\mathrm{D}} \mapsto \mathbb{R}$ and $k(\cdot,\cdot) : \mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{D}} \mapsto \mathbb{R}$, defines a joint distribution of function values for an arbitrary set of input points $\mathbf{X} \in \mathbb{R}^{\mathrm{D}\times\mathrm{P}}$, where m is applied column-wise, and k is applied to all pairwise combinations of columns. The joint distribution of outputs is Normal:

$$f^{\mathrm{d}}(\mathbf{X}) \sim \mathcal{N}\,(\,m(\mathbf{X})\,,\, k(\mathbf{X},\mathbf{X})\,) \tag{4.5}$$

In addition to the mean and covariance functions[3], we now wish to incorporate the fixed points $\mathbf{S} = \left[\mathbf{s}^1, \ldots, \mathbf{s}^{\mathrm{L}}\right] \in \mathbb{R}^{\mathrm{D}\times\mathrm{L}}$ and the local linearisations $\mathbf{J} = \left[\mathbf{J}^1, \ldots, \mathbf{J}^{\mathrm{L}}\right] \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}\times\mathrm{L}}$ where L is the number of fixed points. A Gaussian Process conditioned on these implies a joint distribution of not only the function values at given inputs, but also the fixed point and derivative parameters (which are mathematically equivalent to observations in a GP, but their values may change during the model

[3] that may differ for each output dimension, resulting in a set of m^{d} and k^{d} functions

estimation algorithm). In order to describe this joint distribution, we first need to define the derivatives of the mean and covariance functions, as they couple the Jacobians (essentially equivalent to derivative observations) to function value observations.

Fortunately, the derivative of a Gaussian Process is a Gaussian Process by itself, with the new mean function being the derivative of the mean function $\nabla m(\cdot)$, and the covariance function being the gradient of the covariance function with respect to both inputs $\nabla_1 \nabla_2 k(\cdot,\cdot)$ (see in appendix C). Furthermore, the cross-covariance of a Gaussian Process and its derivative can also be computed, by taking the gradient of the original covariance function with respect to only one of its inputs, $\nabla_1 k(\cdot,\cdot)$ or $\nabla_2 k(\cdot,\cdot)$. Therefore the complete joint normal distribution of the $f^{\mathrm{d}}(\cdot)$ function values at given locations $\mathbf{X}$ induced by a Gaussian Process that includes fixed points $\mathbf{S}$ and their local linearisations $\mathbf{J}$ can be written as

$$\begin{bmatrix} \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \\ f^{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m^{\mathrm{s}} \\ \nabla m^{\mathrm{s}} \\ m^{\mathrm{x}} \end{bmatrix}, \begin{bmatrix} k^{\mathrm{ss}} & \nabla_2 k^{\mathrm{ss}} & k^{\mathrm{sx}} \\ \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} & \nabla_1 k^{\mathrm{sx}} \\ k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} & k^{\mathrm{xx}} \end{bmatrix} \right) \tag{4.6}$$

where $\mathbf{S}_{\mathrm{d}\cdot}$ and $\mathbf{J}_{\mathrm{d}\cdot\cdot}$ selects the d-th entry along the first dimension, and the superscripts indicate the inputs to the functions m and k, with s representing the fixed point locations $\mathbf{S}$, whereas x corresponds to the locations of interest $\mathbf{X}$. Note that the locations of the Jacobians always correspond to that of a fixed point, therefore for example $\nabla_1 k^{\mathrm{sx}} = \nabla_1 k(\mathbf{S},\mathbf{X})$ indeed represents the cross-covariance of the (mean-centred) Jacobians and potential function value observations, with the Jacobians and gradient function outputs appropriately reshaped[4].

An equivalent, alternative view of this prior is instead of examining the joint distribution of fixed-point related parameters and function values as in equation 4.6, we rather define a conditional Gaussian Process on the function values:

$$p(\, f(x)\,) = \mathcal{GP}(\, f(x) \mid \forall_1 f(\mathbf{s}^{\mathrm{l}}) = \mathbf{s}^{\mathrm{l}}\,,\, \nabla f(\mathbf{s}^{\mathrm{l}}) = \mathbf{J}^{\mathrm{l}}\,), \tag{4.7}$$

where the m mean and k covariance functions are omitted from the notation for clarity, but are implied by any GP. Note that the algebraic computations (including the conditioning itself) is generally done based on the joint view in equation 4.6, but this conditional view of GPs may provide additional insight.

As we are establishing a powerful prior over a completely unknown function that we wish to

[4]The gradient operation increases the tensor order of the function, which then need to be flattened to represent the corresponding block of the covariance matrix

estimate from data, we need to insert a little more flexibility into it. The main concern is a priori determining the number of fixed points. In order to automatically select the number of fixed points during the inference and learning processes described later, I chose the strategy of setting the number of fixed points large, but augmenting the prior with an extra learnable variance parameter σ^{l} for each fixed point, such that it doesn't constrain the function value exactly, but $f(\mathbf{s}^{\mathrm{l}}) = \mathbf{s}^{\mathrm{l}} + \sigma^{\mathrm{l}}\boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim (N)(0, I)$. This then results in Automatic Relevance Determination (ARD) style procedure (see Snelson and Ghahramani (2012) for a GP example), in which the system is allowed to learn large σ^{l} values for fixed points that are 'unnecessary', therefore not only learning the location of true fixed points, but also pruning the extra ones. Note that there a number of alternative options available for determining the cardinality of a set of unknown parameters (such as life-death processes as in Pnevmatikakis et al. (2013) or standard model selection via a held out validation dataset), but they tend to be computationally more expensive than the ARD procedure used here.

Mathematically, these extra variance parameters enter the joint prior through the k^{ss} block of the covariance matrix in the joint (equation 4.6) , such that $k^{\mathrm{ss}} = k(\mathbf{S}, \mathbf{S}) + \mathrm{diag}([\sigma^1, \ldots, \sigma^{\mathrm{L}}])$. In the conditional prior view of equation 4.7, these added variance parameters enter much more clearly, by specifying the function value at a fixed point as a normally distributed random variable $p(\tilde{\mathbf{s}}^{\mathrm{l}}) = \mathcal{N}(\mathbf{s}^{\mathrm{l}}, \sigma^{\mathrm{l}} I)$ and integrating over these intermediate random variables:

$$p(f(x)) = \int \cdots \int \mathcal{GP}(f(x) \mid \forall_{\mathrm{l}} f(\mathbf{s}^{\mathrm{l}}) = \tilde{\mathbf{s}}^{\mathrm{l}}, \nabla f(\mathbf{s}^{\mathrm{l}}) = \mathbf{J}^{\mathrm{l}}) \prod_{\mathrm{l}} p(\tilde{\mathbf{s}}^{\mathrm{l}})\, d\tilde{\mathbf{s}}^{\mathrm{l}}. \tag{4.8}$$

Having established how our parametric Gaussian Process prior affects the distribution of functions via the fixed points and local linearisation, we now need to understand how this idea may be applied to infer latent time series $\mathbf{x}^{\mathrm{m}}(t)$ from the data $\mathcal{Y}$, and learn the various model elements: the output mapping g, the output noise $\boldsymbol{\varepsilon}^{\mathrm{y}}$, the transition map function f along with the transition noise $\boldsymbol{\varepsilon}^{\mathrm{x}}$, and of course the parameters of interest, the fixed point locations $\mathbf{S}$, their 'certainties' $\boldsymbol{\sigma}^{\mathrm{s}}$ and the linearisations around each fixed point, $\mathbf{J}$.

4.2.2 Estimating latent Gaussian Process transition map models of time series

The main difficulty of simultaneously inferring latent space trajectories $\mathbf{x}(t)$ and learning the transition function $f(\cdot)$ in the GP framework, is that exact Gaussian Process posteriors are represented by their input-output relationships; however, if these inputs and outputs are themselves latent, and inferred using $f(\cdot)$, a complex dependence structure arises, making exact inference and learning intractable.

This issue of the dependence structure of latent time series models with a GP transition function is discussed at length by McHutchon (2014), here I merely state their results. The first, and easiest way of breaking this dependence structure is to just make the assumption that the dependence structure does not exist to begin with, and treat the time series data as merely a regression problem from

time t to time $t+1$, essentially ignoring the propagation of uncertainty over time. This is simple and can be effective in certain situations, but in the case of investigating neural system, the estimation and representation of uncertainty is often a key question, and as such, I decided to find an alternative approach.

Another way that is widely used in scaling up Gaussian Process, is that of inducing points, essentially a set of extra input-output pairs that exclusively describe the function $f(\cdot)$, without direct dependence on the data. These can either be thought of extra parameters of the complete model leading to the family of methods called sparse GPs, which include FITC and PITC; or can be thought of as variational parameters and are to be integrated over when predicting from an estimated posterior. Here we utilise both approaches in the description of two distinct methods for estimating the transition function. I will first describe in detail the sparse GP version and the algorithm it led to, and later also include the main ideas from the variational approach, which was joint work with Lea Duncker and Julien Boussard and is described in section 4.2.3.

4.2.2.1 Sparse approximation

We introduce inducing point locations $\mathbf{z} \in \mathbb{R}^{\mathrm{D}}$ and uncertain function values at those locations $f^{\mathrm{d}}(\mathbf{z}) = \tilde{u}^{\mathrm{d}}$, where $\tilde{u}^{\mathrm{d}} \sim \mathcal{N}(u_{\mathrm{d}}, \sigma^{\mathrm{u}})$ is a normal random variable with mean u_{d} and uncertainty σ^{u}. The reasons for this uncertainty parameter are both that our measurements are noisy, and that we are learning a function in an unknown, transformed latent space, so we do not expect to have the same uncertainty (noise variance) everywhere in the latent space. This representation for a sparse approximation of a heteroscedastic function follows the idea by Snelson and Ghahramani (2012).

We can also collect the u_{d} scalar values into a vector, and write $f(\mathbf{z}) = \tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}} \sim \mathcal{N}(\mathbf{u}, \sigma^{\mathrm{u}})$, where σ^{u} is diagonal, if the output dimensions of $f(\cdot)$ are independent. Whether or not to use a different uncertainty level σ^{u} for each output dimension is a choice one can make, I decided to use a single value per inducing point location, therefore it represents how much the $\mathbf{u}$ should constrain the function at that location, rather than some form of non-isotropy in latent space transitions (for which we later introduce a different parameter that is location-independent).

Of course, we need more than one inducing point to sufficiently represent the function, and thus we denote the collections of inducing point locations, mean values and uncertainties as $\mathbf{Z} \in \mathbb{R}^{\mathrm{D}\times\mathrm{K}}$, $\mathbf{U} \in \mathbb{R}^{\mathrm{D}\times\mathrm{K}}$ and $\boldsymbol{\sigma}^{\mathrm{u}} \in \mathbb{R}^{\mathrm{K}}$, respectively. These are treated as additional parameters of the model, and are to be estimated given the data, along with parameters described previously. Our "posterior" Gaussian Process will then really mean finding a setting of these inducing point parameters, such that the predictive distribution given the true prior and the inducing point parameters as "noisy training data" results in a GP distribution over functions that optimises the exact marginal log likelihood of the time series data. The joint over the posterior function values $\hat{f}^{\mathrm{d}}(\mathbf{X})$, the prior and the inducing values resemble the form of equation 4.6, noting the added location-wise uncertainty terms $\boldsymbol{\sigma}^{\mathrm{u}}$ and

$\boldsymbol{\sigma}^{\mathrm{s}}$ in the covariance, and the added superscripts z representing the kernel's input being $\mathbf{Z}$, e.g. $\nabla_1 k^{\mathrm{sz}} = \nabla_1 k(\mathbf{S}, \mathbf{Z})$.

$$\begin{bmatrix} \mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \\ \hat{f}^{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m^{\mathrm{z}} \\ m^{\mathrm{s}} \\ \nabla m^{\mathrm{s}} \\ m^{\mathrm{x}} \end{bmatrix}, \begin{bmatrix} k^{\mathrm{zz}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{u}}) & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} & k^{\mathrm{zx}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} & k^{\mathrm{sx}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} & \nabla_1 k^{\mathrm{sx}} \\ k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} & k^{\mathrm{xx}} \end{bmatrix} \right) \tag{4.9}$$

To simplify notation for further calculation, from this point onwards we assume (without loss of generality) that the mean function $m(\cdot)$ is uniformly zero, and we introduce the matrices $\mathbf{K}^{\mathrm{post}} \in \mathbb{R}^{(\mathrm{K}+\mathrm{L}+\mathrm{KL})\times(\mathrm{K}+\mathrm{L}+\mathrm{KL})}$ and $\mathbf{K}^{\mathrm{pred}} \in \mathbb{R}^{\mathrm{P}\times(\mathrm{K}+\mathrm{L}+\mathrm{KL})}$, representing the posterior and the predictive parts of the covariance matrix above:

$$\mathbf{K}^{\mathrm{post}} = \begin{bmatrix} k^{\mathrm{zz}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{u}}) & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} \end{bmatrix}$$

$$\mathbf{K}^{\mathrm{pred}} = \begin{bmatrix} k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} \begin{bmatrix} \mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \\ \hat{f}^{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}^{\mathrm{post}} & (\mathbf{K}^{\mathrm{pred}})^{\top} \\ \mathbf{K}^{\mathrm{pred}} & k^{\mathrm{xx}} \end{bmatrix} \right)$$

As these posterior predictions $\hat{f}$ are going to be the ones that enter into our optimisation scheme, it is useful at this point to define the mean and variance $\hat{f}^{\mathrm{d}}(\mathbf{x})$ given a single test input location $\mathbf{x}$, and the current setting of all model parameters. Note that due to our assumption of independence of

each $f^{\mathrm{d}}(\cdot)$ function, the covariances of the predictions are 0. This trivially generalises to all d output dimensions, and also to multiple input locations[5].

$$\mathbb{E}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\mathbf{x})] = \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} \begin{bmatrix} \mathbf{U}_{\mathrm{d}\cdot} \\ \mathbf{S}_{\mathrm{d}\cdot} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \tag{4.11}$$

$$\mathrm{Var}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\mathbf{x})] = k(\mathbf{x},\mathbf{x}) - \mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} (\mathbf{K}^{\mathrm{pred}})^{\top}$$

Having established how the sparse approximation can be used to describe a "parametric" GP that disconnects the learned transition function from the dependence structure of the time series data, we have two tasks left. First I will discuss how we infer the latent path given the data and a current setting of all parameters (for the prior, the inducing points, the kernel and the output mapping), Secondly, once we have inferred the latent path's distribution, we can see how the parameters may be changed to maximise the exact marginal log likelihood via gradient ascent.

4.2.2.2 Inference and learning

There are numerous inference algorithms available for latent time series data, but they all generally follow the scheme of passing massages forward and/or backward in time in the latent space, and also collecting messages from the observed outputs at each time point, by (approximately) inverting the output model. Message passing schemes that only propagate information forward - and thus may be used in real-time applications - are called filtering algorithms (examples include work by Ko et al. (2007), Poyiadjis, Doucet, and Singh (2011), and Wan and Van Der Merwe (2000)), whereas on already recorded data we may carry out 'smoothing', which also incorporates messages from the future to determine the most likely latent state distribution at a given time (for example Deisenroth and Mohamed (2012) and Deisenroth et al. (2012)).

Given the large number of inference schemes available, it is fortunate that Andrew McHutchon previously compared a large number of these methods on various simulated datasets (McHutchon 2014) that are close to our problem setting, and found that the so-called "Direct method" performed best both in one-step-ahead predictive performance and recovering true model parameters. Furthermore owing to its relative simplicity it is a very stable method, unlikely to find spurious trajectories. The inference scheme of this Direct method is Assumed Density Filtering (ADF, see Deisenroth, Huber, and Hanebeck (2009) or Ramakrishnan, Ertin, and Moses (2011)), which I describe here in more detail.

ADF proceeds via a single forward filtering for each time series in our dataset, and propagates

[5] Note that in the time series setting we generally only have to deal with single input locations at a time, representing our current location in latent space

beliefs through the current estimate of the transition function, incorporates messages from the observations, and projects any resulting non-Gaussian distributions onto a Gaussian via moment matching of the means and variances. This results in a Gaussian belief over each latent state at the end of the inference, enabling us both to represent these beliefs via only means and variances, as well as making it a convenient input to any learning scheme, regardless of the output mapping[6].

Let $\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}$ represent our belief of the latent state at time $t-1$ as a normally distributed random variable, with

$$\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1} \sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathrm{t}-1|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}-1|\mathrm{t}-1}\right) \tag{4.12}$$

The superscripts $t-1|t-1$ indicate that we represent our belief of $\mathbf{x}$ at time $t-1$, with evidence from $\mathbf{y}$ observations incorporated up to time $t-1$ as well. In ADF, we first propagate this belief through the posterior transition function $\hat{f}(\cdot)$, and approximate the resulting distribution again as a Gaussian:

$$\begin{aligned}\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} &\approx \hat{f}(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}) + \boldsymbol{\varepsilon}^{\mathrm{x}}(t)\\ \tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} &\sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\right)\end{aligned} \tag{4.13}$$

These moments may be computed via taking the expectations with respect to $\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}$ of the means and variances of the GP predictions shown in equation 4.11. The posterior part of the GP covariance matrix, $\mathbf{K}^{\mathrm{post}}$, does not depend on the test input $\mathbf{x}$, and thus the expectation with respect to $\mathbf{x}$ only affects $k(\mathbf{x},\mathbf{x})$ and $\mathbf{K}^{\mathrm{pred}}$. Therefore the mean and variances of $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}$ can be computed by

$$\begin{aligned}\mu_{\mathrm{d}}^{\mathrm{t}|\mathrm{t}-1} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}} \mathbb{E}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1})]\\ &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[\mathbf{K}^{\mathrm{pred}}\right]\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\begin{bmatrix}\mathbf{U}_{\mathrm{d}\cdot}\\ \mathbf{S}_{\mathrm{d}\cdot}\\ \mathbf{J}_{\mathrm{d}\cdot\cdot}\end{bmatrix}\end{aligned} \tag{4.14}$$

$$\begin{aligned}\Sigma_{\mathrm{dd}}^{\mathrm{t}|\mathrm{t}-1} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}} \mathrm{Var}_{\hat{\mathrm{f}}}[\hat{f}^{\mathrm{d}}(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1})]\\ &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[k(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}, \tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}) - \mathbf{K}^{\mathrm{pred}}\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}(\mathbf{K}^{\mathrm{pred}})^{\top}\right]\\ &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[k(\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}, \tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1})\right] - \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t}-1|\mathrm{t}-1}}\left[(\mathbf{K}^{\mathrm{pred}})^{\top}\mathbf{K}^{\mathrm{pred}}\right]\mathrm{Tr}\left\{\left[\mathbf{K}^{\mathrm{post}}\right]^{-1}\right\}\end{aligned} \tag{4.15}$$

However, taking the expectation with respect to an uncertain input does introduce co-variances in the prediction, and thus we need to calculate the off-diagonal terms of $\Sigma^{\mathrm{t}|\mathrm{t}-1}$, unlike in equa-

[6] As non-conjugate output mappings can still make use of efficient Gaussian Quadrature integrators due to the Gaussianity of the input

tion 4.11, where those off-diagonal terms were zero. These calculations were originally carried out by Deisenroth, Huber, and Hanebeck (2009), but I have adapted them to include the fixed points and local linearisation terms, and written in terms of expectations rather than integrals, for clarity.

$$\Sigma^{\mathrm{t|t-1}}_{\mathrm{d}_1\mathrm{d}_2} = \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}} \mathbb{E}_{\hat{\mathrm{f}}} \left[\hat{f}^{\mathrm{d}_1}(\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}) \hat{f}^{\mathrm{d}_2}(\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}) \right] - \mu^{\mathrm{t|t-1}}_{\mathrm{d}_1} \mu^{\mathrm{t|t-1}}_{\mathrm{d}_2}$$

The expectation with respect to $\hat{f}$ is then expanded and rearranged as follows:

$$\begin{aligned}
&= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}} \left[\left(\mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} \begin{bmatrix} \mathbf{U}_{\mathrm{d}_1 \cdot} \\ \mathbf{S}_{\mathrm{d}_1 \cdot} \\ \mathbf{J}_{\mathrm{d}_1 \cdot\cdot} \end{bmatrix} \right) \left(\mathbf{K}^{\mathrm{pred}} \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} \begin{bmatrix} \mathbf{U}_{\mathrm{d}_2 \cdot} \\ \mathbf{S}_{\mathrm{d}_2 \cdot} \\ \mathbf{J}_{\mathrm{d}_2 \cdot\cdot} \end{bmatrix} \right) \right] - \mu^{\mathrm{t|t-1}}_{\mathrm{d}_1} \mu^{\mathrm{t|t-1}}_{\mathrm{d}_2} \\
&= \begin{bmatrix} \mathbf{U}_{\mathrm{d}_1 \cdot} \\ \mathbf{S}_{\mathrm{d}_1 \cdot} \\ \mathbf{J}_{\mathrm{d}_1 \cdot\cdot} \end{bmatrix}^{\top} \left[\mathbf{K}^{\mathrm{post}}\right]^{-\top} \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t-1|t-1}}} \left[(\mathbf{K}^{\mathrm{pred}})^{\top} \mathbf{K}^{\mathrm{pred}} \right] \left[\mathbf{K}^{\mathrm{post}}\right]^{-1} \begin{bmatrix} \mathbf{U}_{\mathrm{d}_2 \cdot} \\ \mathbf{S}_{\mathrm{d}_2 \cdot} \\ \mathbf{J}_{\mathrm{d}_2 \cdot\cdot} \end{bmatrix} - \mu^{\mathrm{t|t-1}}_{\mathrm{d}_1} \mu^{\mathrm{t|t-1}}_{\mathrm{d}_2}
\end{aligned} \tag{4.16}$$

These calculations completely determine the approximate belief at time t given data up to time $t-1$:

$$\begin{aligned}
p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}}) &\approx p(\mathbf{x}(t) \mid \mathbf{y}(1:t-1)) \\
\tilde{\mathbf{x}}^{\mathrm{t|t-1}} &\sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathrm{t|t-1}}, \boldsymbol{\Sigma}^{\mathrm{t|t-1}} \right)
\end{aligned} \tag{4.17}$$

The final inference step is to incorporate the observation at time t, and compute the updated belief

$$\begin{aligned}
p(\tilde{\mathbf{x}}^{\mathrm{t|t}}) &\approx p(\mathbf{x}(t) \mid \mathbf{y}(1:t)) \\
\tilde{\mathbf{x}}^{\mathrm{t|t}} &\sim \mathcal{N}\left(\boldsymbol{\mu}^{\mathrm{t|t}}, \boldsymbol{\Sigma}^{\mathrm{t|t}} \right)
\end{aligned} \tag{4.18}$$

In order to do this, we apply Bayes' rule to the likelihood computed given our current belief:

$$\begin{aligned}
p(\mathbf{x}(t) \mid \mathbf{y}(1:t)) &= \frac{p(\mathbf{y}(t) \mid \mathbf{x}(t))\, p(\mathbf{x}(t) \mid \mathbf{y}(1:t-1))}{p(\mathbf{y}(t) \mid \mathbf{y}(1:t-1))} \\
&\approx \frac{p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t|t-1}})\, p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})}{\int p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t|t-1}})\, p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})\, d\tilde{\mathbf{x}}^{\mathrm{t|t-1}}} \\
&= \frac{p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t|t-1}})\, p(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})}{\mathcal{L}(t)}
\end{aligned} \tag{4.19}$$

where $\mathcal{L}(t) \approx p(\mathbf{y}(t) \mid \mathbf{y}(1:t-1))$ is the approximate marginal likelihood contribution at time

t. As the above is a valid normalised probability distribution (that approximates the true posterior belief at $\mathbf{x}(t)$), we can find its first two moments and thus update our beliefs accordingly:

$$\begin{aligned}\boldsymbol{\mu}^{\mathrm{t|t}} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t|t-1}}}\left[\tilde{\mathbf{x}}^{\mathrm{t|t-1}}\,\frac{p(\mathbf{y}(t)\mid\tilde{\mathbf{x}}^{\mathrm{t|t-1}})}{\mathcal{L}(t)}\right]\\ \boldsymbol{\Sigma}^{\mathrm{t|t}} &= \mathbb{E}_{\tilde{\mathbf{x}}^{\mathrm{t|t-1}}}\left[\tilde{\mathbf{x}}^{\mathrm{t|t-1}}\,(\tilde{\mathbf{x}}^{\mathrm{t|t-1}})^{\top}\,\frac{p(\mathbf{y}(t)\mid\tilde{\mathbf{x}}^{\mathrm{t|t-1}})}{\mathcal{L}(t)}\right]-\boldsymbol{\mu}^{\mathrm{t|t}}(\boldsymbol{\mu}^{\mathrm{t|t}})^{\top}\end{aligned} \tag{4.20}$$

Given that our belief representation $\tilde{\mathbf{x}}^{\mathrm{t|t-1}}$ is Gaussian, for conjugate likelihoods (such as affine or GP transformations with additive Gaussian noise) these expectations may be available in closed form (e.g. Deisenroth, Huber, and Hanebeck (2009)). However, even for non-conjugate likelihoods, the resulting integrals over $\tilde{\mathbf{x}}^{\mathrm{t|t-1}}$ can be computed by Gaussian quadrature, and thus these moments can be approximated efficiently using numerical methods for arbitrary observation models.

To summarise, in order to infer the latent time series $\mathbf{x}(\cdot)$, we chose to represent our beliefs approximately via normally distributed random variables $\tilde{\mathbf{x}}^{\mathrm{t|t}}$. Given an initial belief $\left\{\boldsymbol{\mu}^{0|0},\,\boldsymbol{\Sigma}^{0|0}\right\}$, we can successively for each new observation time 1.) update the belief by propagating it through the GP transition function (see equations 4.14 to 4.16) and 2.) incorporate the new observation into our belief (see equations 4.18 to 4.20).

These methods and equations are generally applicable for any choice of kernel function or observation model. However, it must be stated that the kernel expectations which arise in step 1 are available in closed form only for linear, polynomial and Exponentiated Quadratic / Squared Exponential kernel functions. As the Exponentiated Quadratic kernel function is the most applicable one, I derived analytic expressions for the derivatives as well as all required expectations for it in appendix C, which enables closed form forward propagation of the belief. The expectations in step 2 are only computable in closed form for linear or GP transformations with additive Gaussian noise.

Fortunately the Exponentiated Quadratic kernel over the latent space, and linear output mapping with additive Gaussian noise defines a rather wide class of models. It restricts only the noise model, the smoothness of the transition map $f(\cdot)$ – as defined by the lengthscale(s) of the kernel(s) – and constrains the output mapping $g(\cdot)$ to be monotonic[7]. I derive the filtering step for the linear output mapping in section 4.2.2.3 for this widely applicable setup.

Learning During the inference process described above, we already compute the approximate marginal likelihood terms at each observation, $\mathcal{L}(t)$, and therefore we can compute the log marginal likelihood as a function of all parameters θ .

[7] Any non-linearity in a monotonic output function can be represented by transformation of the latent space itself, and the corresponding change of the transition function. In fact, allowing for and learning complex output mappings reduces the interpretability of the inferred latent trajectories and the learned non-linear transition map, and therefore should be avoided.

$$
\begin{aligned}
l(\theta) &= \log p(\, y(1:T) \mid \theta) \\
&= \log \prod_{\mathrm{t}} p(\mathbf{y}(t) \mid \mathbf{y}(1:t-1)) \\
&\approx \sum_{\mathrm{t}} \log \int p(\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})\; p(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1})\; d\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} \\
&= \sum_{\mathrm{t}} \log \mathcal{L}(t)
\end{aligned}
\tag{4.21}
$$

As all operations described in this section are differentiable (or the derivatives can numerically be approximated), if one keeps track of the computation graph and the individual derivatives[8], all parameters can be updated via gradient ascent on $l(\theta)$ as the objective. In practice it is useful to only update certain parameters after an inference step, such as keeping the output mapping $g(\cdot)$ (and thus the structure of the latent space) more stable while initially learning the transition map $f(\cdot)$ via its parameters $\theta_{\mathrm{f}} = \{\mathbf{S}, \mathbf{J}, \boldsymbol{\sigma}^{\mathrm{s}}, \mathbf{Z}, \mathbf{U}, \boldsymbol{\sigma}^{\mathrm{u}}\}$ and kernel hyperparameters θ_{k}. Inference and learning steps are iterated until convergence (or until some early stopping threshold based on a metric on held out validation data).

4.2.2.3 Linear Gaussian observation model

As stated above, for a linear Gaussian observation model, we can obtain closed form updates for incorporating a new observation into our belief, meaning we can fully specify equations 4.18 to 4.20. Let

$$
\begin{aligned}
p(\,\mathbf{y}(t) \mid \tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}) &= \mathcal{N}_{\mathbf{y}}\,(g(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}), \boldsymbol{\Sigma}^{\mathrm{y}}) \\
g(\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1}) &= \mathbf{C}\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} \\
\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} &\sim \mathcal{N}\,(\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1})
\end{aligned}
\tag{4.22}
$$

where $\tilde{\mathbf{x}}^{\mathrm{t}|\mathrm{t}-1} \in \mathbb{R}^{\mathrm{D}}$ is our belief at time t given data up to time $t-1$, $\mathbf{C} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is the linear output mapping, $\boldsymbol{\Sigma}^{\mathrm{y}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is the observation noise variance. We wish to calculate the approximate marginal likelihood given our current belief, as well as update our belief in light of the new observation. We can calculate the conditional mean and variance of y given our belief as

$$
\begin{aligned}
p(\mathbf{y}(t) \mid \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}) &= \mathcal{N}_{\mathbf{y}}\left(\boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}, \boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right) \\
\boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}} &= \mathbf{C}\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1} \\
\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}} &= \boldsymbol{\Sigma}^{\mathrm{y}} + \mathbf{C}\boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\mathbf{C}^{\top}
\end{aligned}
\tag{4.23}
$$

Therefore the approximate marginal likelihood contribution at time t is

[8] for which automated methods are now widely available

$$\begin{aligned}\log \mathcal{L}(t) &= p(\mathbf{y}^{\mathrm{t}} \mid \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1}, \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}) \\ &= -\frac{1}{2}\left(N\log(2\pi) + \log\|\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\| + (\mathbf{y}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}})^{\top}\left(\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right)^{-1}(\mathbf{y}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}})\right),\end{aligned} \tag{4.24}$$

and from the conditional $x|y,b$ we can update our belief via

$$\begin{aligned}\boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}} &= \boldsymbol{\mu}^{\mathrm{t}|\mathrm{t}-1} + \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\mathbf{C}^{\top}\left(\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right)^{-1}(\mathbf{y}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}) \\ \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}} &= \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1} - \boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}\mathbf{C}^{\top}\left(\boldsymbol{\Sigma}^{\mathrm{t}}_{\mathrm{y}|\mathrm{b}}\right)^{-1}\mathbf{C}\boldsymbol{\Sigma}^{\mathrm{t}|\mathrm{t}-1}.\end{aligned} \tag{4.25}$$

When this linear Gaussian observation model is used alongside an Exponentiated Quadratic kernel function, the inference is completely analytically computable with closed form expressions, and we can use the approximate marginal likelihood $l(\theta) = \sum_{\mathrm{t}} \log \mathcal{L}(t)$ as an objective function for parameter optimisation via gradient ascent.

In accordance with previous terminology, I will refer to this algorithm as *Fixed-point Gaussian Process Assumed Density Filtering* (FP GP-ADF) with linear Gaussian observations in further sections, and show applications in sections 4.3.2 and 4.3.5.

4.2.3 Continuous time modelling of transition flows via Variational Sparse Gaussian Process inference and learning

This section presents joint work with Lea Duncker and Julian Boussard, published in Duncker et al. (2019), resulting in the *Fixed-point Gaussian Process latent Stochastic Differential Equation* (FP GP-SDE) algorithm. I summarise here the main ideas, please refer to the paper for further details[9].

An alternative, more flexible approach to modeling dynamical systems is to identify the transition flows in the latent space, rather than relying upon discrete time transition maps. As mentioned in section 4.2.2, we may use stochastic differential equations (SDEs) to model the evolution of the latent state $\mathbf{x}$

$$d\mathbf{x} = f'(\mathbf{x})\,dt + \sqrt{\boldsymbol{\Sigma}}\,d\mathbf{w}, \tag{4.26}$$

where the function $f'(\cdot)$ now represents a transition flow rather than the transition map itself, and $\boldsymbol{\Sigma}$ is the incremental noise covariance shaping the Wiener noise process $\mathbf{w}(t)$.

The latent state is observed indirectly through noisy measurements $\mathbf{y} \in \mathbb{R}^{\mathrm{N}}$ at unevenly spaced time points t_{i}.

[9] Mind the notation and definitions in the paper is slightly different from present thesis chapter, but both are self-consistent. Furthermore, the notation within this subsection uses subscripts for naming objects, and not to be understood as indicial notation.

$$\mathbb{E}_{\mathrm{y}|\mathrm{x}}[\mathbf{y}(t_\mathrm{i})] = g(\mathbf{C}\mathbf{x}(t_\mathrm{i}) + \mathbf{d}) \tag{4.27}$$

We consider here a general, but particular form of measurement model, in which the measurements are distributed with a known parametric form and generalized linear dependence; that is the expected value is $g(\mathbf{C}\mathbf{x}+\mathbf{d})$ with a given inverse-link function g and parameters $\mathbf{C} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and $\mathbf{d} \in \mathbb{R}^{\mathrm{N}}$. We seek to infer latent paths $\mathbf{x}(t)$ along with the dynamical parameters and an interpretable representation of the dynamical transition flow $f'(\cdot)$.

In the following I describe the two main differences compared to the previous FP GP-ADF algorithm. Firstly, we chose to use a different treatment of how inducing points were incorporated in the model and what objective we use to learn model parameters. In the above FP GP-ADF algorithm I treat the inducing points (represented by locations, mean values and uncertainties) as extra parameters of the model, and found maximum likelihood point estimates for all parameters, based on an approximate marginal likelihood as the objective function. Here, as shown in section 4.2.3.1, we used an alternative option for learning model parameters based on treating both the distribution over the latents $\mathbf{x}$ as well as the inducing points $\mathbf{u}$ as variational parameters following Titsias (2009). We then maximise the variational free energy (a well-defined lower bound to the marginal likelihood rather than an approximation to it as previously) by integrating over the variational parameters[10]. Secondly, in order to make use of the continuous time representation based on SDEs, we need to employ a different inference algorithm (derived in section 4.2.3.2), one that uses the dynamical flow representation to propagate latent beliefs and can thus incorporate observations at arbitrary time points.

4.2.3.1 Variational sparse approximation

Although mathematically the introduction of fixed and inducing points to a GP prior over the transition flow function $f'(\cdot)$ looks very similar to what we derived in equation 4.9, there are some significant differences in what the parameters mean and how we proceed from there. I first state the updated representation, then discuss the differences.

[10] Note that the parametric and variational treatments of model learning are easily interchangeable and thus we could similarly derive a variational version of FP GP-ADF algorithm or a parametric version of the algorithm described here.

$$\begin{bmatrix} \tilde{\mathbf{U}}_{\mathrm{d}\cdot} \\ \mathbf{0} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \\ f'_{\mathrm{d}}(\mathbf{X}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k^{\mathrm{zz}} & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} & k^{\mathrm{zx}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} & k^{\mathrm{sx}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} & \nabla_1 k^{\mathrm{sx}} \\ k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} & k^{\mathrm{xx}} \end{bmatrix} \right) \tag{4.28}$$

Firstly, as we are modelling the transition flow rather than the map, we expect the function values at fixed points to be zero, rather than identity, therefore $f'(\mathbf{s}^{\mathrm{l}}) = \mathbf{0}$ for each fixed point. Similarly the Jacobians $\mathbf{J}^{\mathrm{l}}$ now represent derivatives of the flows rather than the map. Secondly, the inducing point values used in the prior are themselves random variables, and their prior distribution is defined by the kernel governing the GP, therefore for each latent dimension d we have $\tilde{\mathbf{U}}_{\mathrm{d}\cdot} \sim \mathcal{N}(\mathbf{0}, k^{\mathrm{zz}})$ a priori, and thus there is no extra diagonal term added to k^{zz} in the joint covariance matrix. Given these changes, we can again simplify the notation for the joint, then calculate the conditional prior on each $f'_{\mathrm{d}}(\cdot)$ function, similarly to equations 4.10 and 4.11, as follows:

$$\begin{aligned} \mathbf{K}^{\mathrm{prior}} &= \begin{bmatrix} k^{\mathrm{zz}} & k^{\mathrm{zs}} & \nabla_2 k^{\mathrm{zs}} \\ k^{\mathrm{sz}} & k^{\mathrm{ss}} + \mathrm{diag}(\boldsymbol{\sigma}^{\mathrm{s}}) & \nabla_2 k^{\mathrm{ss}} \\ \nabla_1 k^{\mathrm{sz}} & \nabla_1 k^{\mathrm{ss}} & \nabla_1 \nabla_2 k^{\mathrm{ss}} \end{bmatrix} \\ \mathbf{K}^{\mathrm{pred}} &= \begin{bmatrix} k^{\mathrm{xz}} & k^{\mathrm{xs}} & \nabla_2 k^{\mathrm{xs}} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_{\mathrm{d}\cdot} \\ \mathbf{0} \\ \mathbf{J}_{\mathrm{d}\cdot\cdot} \end{bmatrix} \\ f'_{\mathrm{d}}(\mathbf{X}) \end{bmatrix} &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}^{\mathrm{prior}} & (\mathbf{K}^{\mathrm{pred}})^{\top} \\ \mathbf{K}^{\mathrm{pred}} & k^{\mathrm{xx}} \end{bmatrix} \right) \end{aligned} \tag{4.29}$$

The form of the predictive mean and variance remains the same, noting that the inducing point

values are random variables, and thus this represents a true conditional prior GP over $f'(\cdot)$ rather than a parametrised posterior.

$$f'_{\mathrm{d}} \,|\, \tilde{u}_{\mathrm{d}} \sim \mathcal{GP}\left(\mathbb{E}_{\mathrm{f}'\,|\,\tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})], \mathrm{Var}_{\mathrm{f}'\,|\,\tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})]\right)$$

$$\mathbb{E}_{\mathrm{f}'\,|\,\tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})] = \mathbf{K}^{\mathrm{pred}}\left[\mathbf{K}^{\mathrm{prior}}\right]^{-1}\begin{bmatrix}\tilde{\mathbf{U}}_{\mathrm{d}\cdot}\\ \mathbf{0}\\ \mathbf{J}_{\mathrm{d}\cdot\cdot}\end{bmatrix} \tag{4.30}$$

$$\mathrm{Var}_{\mathrm{f}'\,|\,\tilde{\mathrm{u}}}[f'_{\mathrm{d}}(\mathbf{x})] = k(\mathbf{x},\mathbf{x}) - \mathbf{K}^{\mathrm{pred}}\left[\mathbf{K}^{\mathrm{prior}}\right]^{-1}(\mathbf{K}^{\mathrm{pred}})^{\top}$$

We derived an alternative sparse GP prior over the transition flow $f'(\cdot)$, conditioned on the inducing point values $\tilde{\mathbf{U}}$ and parametrised by the inducing point locations $\mathbf{Z}$, as well as fixed point locations $\mathbf{S}$, the local linearisations around them $\mathbf{J}$, and the kernel hyperparameters θ_{k}. We now need to understand how to represent the latent process $\mathbf{x}(t)$, incorporate observations $\mathbf{y}(t_{\mathrm{i}})$ recorded at arbitrary times t_{i}, propagate beliefs about $\mathbf{x}$ through the transition flow function $f'(\cdot)$ and thus infer the posterior $p(\mathbf{x}(t)\,|\,\mathcal{Y})$, and finally learn the parameters of both the transition flow $f'(\cdot)$ and the output mapping $g(\mathbf{C}\mathbf{x}+\mathbf{d})$.

4.2.3.2 Inference and learning

The problem of performing approximate inference in continuous-time SDE models has been considered previously, with the two main approaches being Expectation Propagation (Cseke et al. 2016) and variational inference (Archambeau et al. 2007, 2008). We first review the latter approach in this section, then derive our Variational Bayes algorithm that extends this work.

Review of Archambeau approximation

Archambeau et al. (2007, 2008) consider the latent SDE model in equations 4.26 and 4.27 under linear Gaussian observations. The authors derive an approximate inference algorithm based on a variational Gaussian approximation to the posterior process on $\mathbf{x}(t)$ under the constraint that the approximate process has Markov structure, as is the case for the true posterior process. The most general way to construct such an approximation is via a linear time-varying SDE of the form

$$d\mathbf{x} = (-\mathbf{A}(t)\mathbf{x}(t) + \mathbf{b}(t))\,dt + \sqrt{\boldsymbol{\Sigma}}\,d\mathbf{w} \tag{4.31}$$

The instantaneous marginal distributions of this approximation at any time t are Gaussian, with

means $\mathbf{m}_{\mathrm{x}}(t)$ and covariances $\mathbf{R}_{\mathrm{x}}(t)$ that evolve in time according to the ordinary differential equations (ODEs):

$$\begin{aligned}\frac{d\mathbf{m}_{\mathrm{x}}}{dt} &= -\mathbf{A}(t)\mathbf{m}_{\mathrm{x}} + \mathbf{b}(t) \\ \frac{d\mathbf{R}_{\mathrm{x}}}{dt} &= -\mathbf{A}(t)\mathbf{R}_{\mathrm{x}} - \mathbf{R}_{\mathrm{x}}\mathbf{A}(t)^{\top} + \mathbf{\Sigma}\end{aligned} \tag{4.32}$$

Archambeau et al. (2007, 2008) derive a lower bound to the marginal log-likelihood – often called the variational free energy or evidence lower bound – whose maximisation with respect to q_{x} is equivalent to minimising the Kullback-Leibler (KL) divergence between the approximate and true posterior process. The free energy has the form

$$\mathcal{F} = \sum_{\mathrm{i}} \left\langle \log p(\mathbf{y}_{\mathrm{i}}|\mathbf{x}_{\mathrm{i}}) \right\rangle_{q_{\mathrm{x}}} - \mathsf{KL}[q_{\mathrm{x}}(\mathbf{x}) \| p(\mathbf{x})] \tag{4.33}$$

The first term is the expected log-likelihood under the approximation and only depends on the marginal distributions $q_{\mathrm{x}}(\mathbf{x}(t))$. The second term is the KL-divergence between the continuous-time approximate posterior process and the prior process. Archambeau et al. (2007) show that this term can be written as

$$\mathsf{KL}[q_{\mathrm{x}}(\mathbf{x}) \| p(\mathbf{x})] = \int_{\mathcal{T}} dt \left\langle (\mathbf{f} - \mathbf{f}_{\mathrm{q}})^{\top} \mathbf{\Sigma}^{-1} (\mathbf{f} - \mathbf{f}_{\mathrm{q}}) \right\rangle_{\mathrm{q}} \tag{4.34}$$

where $\mathbf{f}$ is the output of the true stationary posterior transition flow $\hat{f}'(\cdot)$ evaluated at $\mathbf{x}(t)$, $\mathbf{f} = \hat{f}'(\mathbf{x}(t))$, and $\mathbf{f}_{\mathrm{q}} = \mathbf{f}_{\mathrm{q}}(t, \mathbf{x}(t)) = -\mathbf{A}(t)\mathbf{x}(t) + \mathbf{b}(t)$ is our time-varying approximation to the current flow, with the inputs omitted for both. Note that the noise covariance $\mathbf{\Sigma}$ is deliberately chosen to be equal for the SDEs in q_{x} and p, as this term would diverge otherwise.

To maximise $\mathcal{F}$ with respect to $\mathbf{m}_{\mathrm{x}}(t)$ and $\mathbf{R}_{\mathrm{x}}(t)$, subject to the constraint that the approximate posterior process has Markov structure according to equation 4.31, one can find the stationary points of the Lagrangian

$$\mathcal{L} = \mathcal{F} - \mathcal{C}_1 - \mathcal{C}_2 \tag{4.35}$$

with

$$\begin{aligned}\mathcal{C}_1 &= \int_{\mathcal{T}} dt \; \mathsf{Tr}\left[\Psi\left(\frac{d\mathbf{R}_{\mathrm{x}}}{dt} + \mathbf{A}\mathbf{R}_{\mathrm{x}} + \mathbf{R}_{\mathrm{x}}\mathbf{A}^{\top} - \mathbf{\Sigma}\right)\right] \\ \mathcal{C}_2 &= \int_{\mathcal{T}} dt \; \boldsymbol{\lambda}^{\top}\left(\frac{d\mathbf{m}_{\mathrm{x}}}{dt} + \mathbf{A}\mathbf{m}_{\mathrm{x}} - \mathbf{b}\right),\end{aligned} \tag{4.36}$$

where Ψ and $\boldsymbol{\lambda}$ are Lagrange multipliers. Archambeau et al. (2007, 2008) derive a smoothing algorithm that involves iterating fixed point updates of this Lagrangian. These are either closed form, or require solving ODEs forward and backward in time, thus achieving linear time complexity. In the next section, we will modify this original algorithm in order to improve its numerical stability, and show how to incorporate it in an efficient Variational Bayes scheme.

Novel Variational Bayes extension

We can derive an efficient Variational Bayes (VB) algorithm (Attias 2000) for variational inference and learning in the model defined in equations 4.26 and 4.27, by maximising a variational free energy. We assume that our full variational distribution factorises as

$$q(\mathbf{x}, \mathbf{f}, \tilde{\mathbf{u}}) = q_{\mathrm{x}}(\mathbf{x}) q_{\mathrm{f,u}}(\mathbf{f}, \tilde{\mathbf{u}}) \tag{4.37}$$

Following Titsias (2009), we choose $q_{\mathrm{f,u}}(\mathbf{f}, \tilde{\mathbf{u}}) = \prod_{\mathrm{k}=1}^{\mathrm{K}} p(f_{\mathrm{k}} | \tilde{\mathbf{u}}_{\mathrm{k}}, \theta) q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}})$. The variational approximation of the posterior over the inducing points are chosen to be of the form $q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}}) = \mathcal{N}\left(\tilde{\mathbf{u}}_{\mathrm{k}} | \mathbf{m}_{\mathrm{u}}^{\mathrm{k}}, \mathbf{R}_{\mathrm{u}}^{\mathrm{k}}\right)$. The marginal variational distribution $q_{\mathrm{f}}(\mathbf{f}) = \prod_{\mathrm{k}} \int d\tilde{\mathbf{u}}_{\mathrm{k}} p(f_{\mathrm{k}} | \tilde{\mathbf{u}}_{\mathrm{k}}, \theta) q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}})$ is also a Gaussian Process. The resulting expression for the variational free energy is of the form:

$$\mathcal{F}^* = \langle \mathcal{F} \rangle_{\mathrm{q_f}} - \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathsf{KL}[q_{\mathrm{u}}(\tilde{\mathbf{u}}_{\mathrm{k}}) \| p(\tilde{\mathbf{u}}_{\mathrm{k}} | \theta)] \tag{4.38}$$

The VB algorithm then iterates over an inference step, where the distribution q_{x} over the latent path is updated, a learning step where $q_{\mathrm{f,u}}$ and the parameters in the affine output mapping are updated, and a hyperparameter learning step where the kernel hyperparameters, and fixed point locations are updated.

Inference Our inference approach follows directly from the work by Archambeau et al. (2007, 2008), though we consider a wider class of observation models and include a nonparametric Bayesian treatment of the dynamics $\mathbf{f}$ under the conditioned sparse GP prior introduced in section 4.2.3.1.

After using integration by parts on the Lagrangian in equation 4.35 (exchanging $\mathcal{F}$ for $\mathcal{F}^*$), we take variational derivatives with respect to $\mathbf{m}_{\mathrm{x}}(t)$ and $\mathbf{R}_{\mathrm{x}}(t)$. Since our model has a rotational non-identifiability with respect to the latents $\mathbf{x}$, we fix $\boldsymbol{\Sigma} = I$ without loss of generality. We arrive at the following set of fixed point equations:

$$\frac{d\boldsymbol{\Psi}}{dt} = \mathbf{A}(t)^{\top} \boldsymbol{\Psi}(t) - \boldsymbol{\Psi}(t) \mathbf{A}(t) - \frac{\partial \mathcal{F}^*}{\partial \mathbf{R}_{\mathrm{x}}} \odot \mathbb{P} \tag{4.39}$$

$$\frac{d\boldsymbol{\lambda}}{dt} = \mathbf{A}^{\mathrm{T}}(t) \boldsymbol{\lambda}(t) - \frac{\partial \mathcal{F}^*}{\partial \mathbf{m}_{\mathrm{x}}} \tag{4.40}$$

$$\mathbf{A}(t) = \left\langle \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right\rangle_{\mathrm{q_x q_f}} + 2\boldsymbol{\Psi}(t) \tag{4.41}$$

$$\mathbf{b}(t) = \langle \mathbf{f}(\mathbf{x}) \rangle_{\mathrm{q_x q_f}} + \mathbf{A}(t) \mathbf{m}_{\mathrm{x}}(t) - \lambda(t) \tag{4.42}$$

with $\mathbb{P}_{\mathrm{ij}} = \frac{1}{2}$ for $i \neq j$ and 1 otherwise; and $\odot$ denotes the Hadamard product. In contrast to previous work, we explicitly take the symmetric variations of $\mathbf{R}_{\mathrm{x}}(t)$ into account, which leads to a slightly modified equation 4.39 compared to the work by Archambeau et al. (2007, 2008), and seems to im-

prove the numerical stability of the algorithm. As a result, we can work with the fixed point updates in equations 4.41 and 4.42 directly, without introducing a learning rate parameter that blends the updates with the previous value of the variational parameters $\mathbf{A}$ and $\mathbf{b}$, as was done by Archambeau et al. (2007, 2008).

The inference algorithm involves solving the set of coupled ODEs in equation 4.32 and equations 4.39 to 4.42. using the conditions $\mathbf{m}_{\mathrm{x}}(0) = \mathbf{m}_{\mathrm{x},0}$, $\mathbf{R}_{\mathrm{x}}(0) = \mathbf{R}_{\mathrm{x},0}$ and $\boldsymbol{\lambda}(T) = 0$, $\boldsymbol{\Psi}(T) = 0$. In principle, it is possible to use any ODE solver to do this. In this work, we chose to solve equation 4.32 using the forward Euler method with fixed step size Δt to obtain $\mathbf{m}_{\mathrm{x}}$ and $\mathbf{R}_{\mathrm{x}}$ evaluated on an evenly spaced grid. Similarly, we then solve equations 4.39 and 4.40 backwards in time to obtain evaluations of $\boldsymbol{\lambda}$ and $\boldsymbol{\Psi}$. The solutions from the ODEs can then be used with equations 4.41 and 4.42 to obtain evaluations of $\mathbf{A}$ and $\mathbf{b}$ on the same time-grid used for solving the ODEs.

Evaluating the expectations of the terms involving $\mathbf{f}$ with respect to q_{x} and q_{f} only involves computing Gaussian expectations of covariance functions and their derivatives. These can be computed analytically for choices such as an exponentiated quadratic covariance function. We update the initial state values $\mathbf{m}_{\mathrm{x},0}$ and $\mathbf{R}_{\mathrm{x},0}$ using the same procedure as described by Archambeau et al. (2008). Given the function evaluations on the inference time-grid, we use linear interpolation to obtain function evaluations of $\mathbf{m}_{\mathrm{x}}$ and $\mathbf{R}_{\mathrm{x}}$ at arbitrary time points. Further details on the inference algorithm are given in the appendix of Duncker et al. (2019).

Learning Dynamics The only terms in equation 4.38 that depend on parameters in $\mathbf{f}$ are the expected KL-divergence between the prior and approximate posterior processes and the KL-divergence relating to the inducing points for $\mathbf{f}$, which are jointly quadratic in the inducing points and Jacobians. Thus, given $\mathbf{m}_{\mathrm{x}}(t)$, $\mathbf{R}_{\mathrm{x}}(t)$, $\mathbf{A}(t)$ and $\mathbf{b}(t)$, we can find closed form updates for the Jacobians and variational parameters relating to $\mathbf{f}$. These updates require tractable linear algebraic operations, and the computation of one-dimensional integrals, which can be carried out efficiently using Gauss-Legendre quadratures. Detailed derivations are given in the appendix of Duncker et al. (2019); there we also provide closed form updates for the sparse variational GP approach for modelling $\mathbf{f}$, without further conditioning on fixed points and Jacobians.

Learning Output Mapping The only term that depends on the parameters $\mathbf{C}$ and $\mathbf{d}$ in equation 4.38 is the expected log-likelihood. Whether or not our algorithm admits for closed form solutions depends on the choice of observation model. In the case of a Gaussian likelihood, we can find the optimal updates as

$$\mathbf{C}^* = \left(\sum_{\mathrm{t}} (\mathbf{y}_{\mathrm{t}} - \mathbf{d})\mathbf{m}_{\mathrm{x,t}}^\top\right)\left(\sum_{\mathrm{t}} (\mathbf{R}_{\mathrm{x,t}} + \mathbf{m}_{\mathrm{x,t}}\mathbf{m}_{\mathrm{x,t}}^\top)\right)^{-1} \tag{4.43}$$

$$\mathbf{d}^* = \frac{1}{T}\sum_{\mathrm{t}} \left(\mathbf{y}_{\mathrm{t}} - \mathbf{C}^*\mathbf{m}_{\mathrm{x,t}}\right), \tag{4.44}$$

where the subscript t denotes a function evaluation at t. For other choices of observation model a closed form solution may not be available, but parameter updates can again be found by maximising the free energy using standard optimisation approaches.

Learning Hyperparameters The kernel function hyperparameters θ_k and fixed point locations $\mathbf{S}$ are learnt by direct optimisation of the variational free energy. The inducing point locations $\mathbf{Z}$ can also be included here, though we chose to hold them fixed on a chosen grid for all examples shown.

4.3 Applications

Having defined two novel algorithms, FP GP-ADF and FP GP-SDE that incorporate fixed points and their local linearisation as directly learnable parameters, I wish to demonstrate that these ideas and the resulting algorithms are indeed applicable for a variety of problem settings, including neural data analysis. Although truly understanding an unknown real-world dynamical system is extremely difficult, requiring years of careful analysis and probably repeated experiments and data collection to test the hypotheses arising from initial analyses, these algorithms may serve as one of the tools researchers can apply to glean further insights into systems, and generate truly testable hypotheses based on the crude dynamical portraits we can recover computationally.

Due to the difficulties in evaluating whether a learned portrait accurately describes an unknown system, we chose to validate the algorithms via a range of carefully simulated experiments, rather than applying them to experimental data and attempting to draw superficial conclusions based on the resulting fits.

This section describes the simulated systems and shows that the algorithms are indeed capable of recovering the true dynamical portraits in a variety of settings, including non-uniform observation times (for FP GP-SDE) and non-Gaussian noise models. The latent dynamics were restricted to 1 and 2 dimensional systems in these examples, as it is difficult to visualise and interpret the dynamical portrait as well as the type and stability of stochastic fixed points in higher dimensions. However, the implemented algorithms are readily applicable to $D > 2$ latent dimensions too. There is no such restriction put on the dimensionality of the observed space, N, as in both electrical and imaging type neural recordings we often get $N \gg 50$ simultaneously recorded neural timeseries, and thus in the experiments below we demonstrate that the algorithms can indeed make use of such data.

I first show the basic capabilities of both algorithms, FP GP-ADF and FP GP-SDE, via two similar one-dimensional systems, one defined by its transition flow function and sampled irregularly in time (section 4.3.1), and the other sampled on a time grid with a dynamical map function gowerning its evolution (section 4.3.2). Afterwards, I describe a life-like simulation of complex chemical reactor, with three stable and two unstable fixed points. It was observed in high dimensions via a 'spectroscopic' measurement process, and showcases that our system is capable of learning very

complex dynamical portraits (section 4.3.3).

Next, I discuss neural applications: first via simulating a generically parametrised known two-dimensional SDE observed through a multivariate point process emulating electrical population recordings (section 4.3.4); finally, I simulate from a high-dimensional spiking neural model observed through a calcium microscopy like process on a regular time grid representing video frames. The parameters of the spiking system were optimised by others to fit real neural recordings which implement multi-stage 2-dimensional 'memory and decision making' dynamics, regulated by external input and experiencing bifurcation (section 4.3.5).

In all examples, the Exponentiated Quadratic kernel function is used, due to the analytical solutions it provides. For both methods the inducing point locations $\mathbf{Z}$ are chosen on an equidistant grid, and these locations are not optimised, thus reducing computational complexity and improving stability. For the GP-ADF method we initialise parameters via GP regression from time t to $t+1$. For the GP-SDE method, the inducing point means and the Jacobians are initialised as zero and the ODEs in equations 4.39 to 4.42 are solved with the forward Euler method with $\Delta t = 1\text{ms}$.

4.3.1 Double well flow

We first demonstrate the FP GP-SDE method on a classic one-dimensional double-well example, where the latent SDE evolves with drift $f(x) = 4x(1-x^2)$, and is observed through $y(t) = \mathbf{C}x(t) + \mathbf{d}$. We simulate data on 20 trials with multivariate Gaussian outputs of dimensionality $N = 15$ with unknown variances 0.25, and observe the output process at 20 randomly sampled time-points per trial. We chose 8 evenly spaced inducing points in $(-3, 3)$ for f. While the true dynamics have three fixed points, we condition the prior on f on four fixed points, initialise their standard deviations as 0.1 and use the ARD method to automatically adjust those to select the correct number of fixed points. The results are summarised in figure 4.1, demonstrating that the algorithm can successfully perform inference and interpretable learning of the SDE path and dynamics, respectively, and does not move away from the good initial location for the model parameters $\mathbf{C}$ and $\mathbf{d}$.

4.3.2 Double well map and pitchfork bifurcation

Next I demonstrate that not only can we employ these ideas to find fixed points in a single experimental dataset, but rather that FP GP-ADF applied to a set of datasets, which differ in the setting of a single parameter (often the case in experimental data), can discover qualitative changes in the dynamics as the parameter varies. This qualitative change is called bifurcation, and is identified by FP GP-ADF by correctly detecting that the number and stability of fixed points differ between datasets. Here I demonstrate this ability through a classic one-parameter system that exhibits pitchfork bifurcation, and strongly resembles the one in section 4.3.1. Instead of investigating the performance at a single setting of the parameter, I generated a set of datasets with varying parameter values.

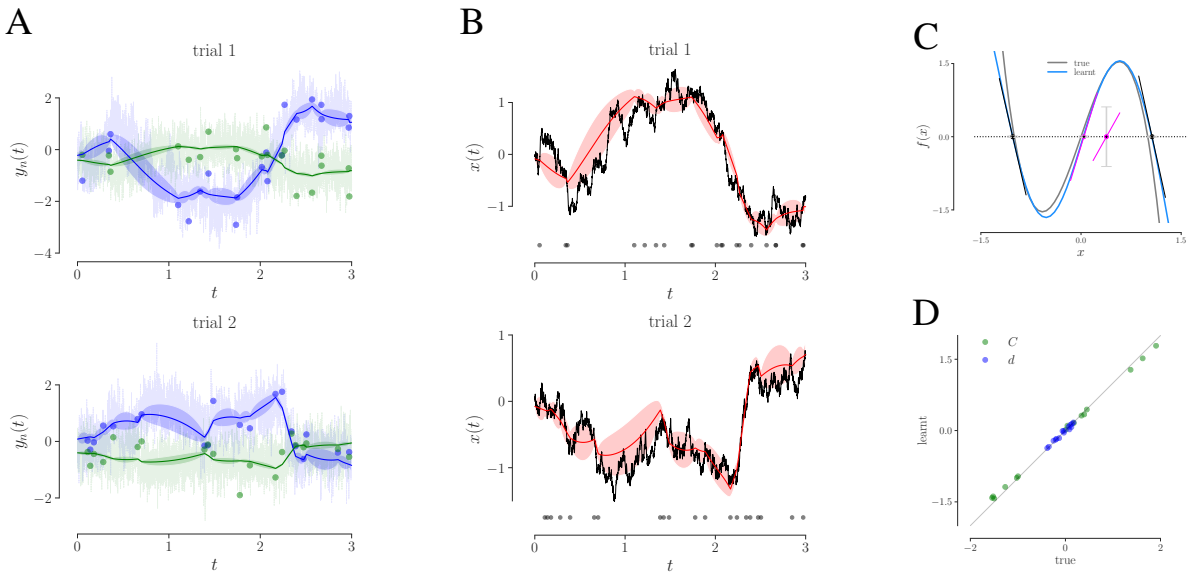


Figure 4.1: Double-well dynamics. A: Two example dimensions of the output process on two different trials. The dots represent the observed data-points of the noisy output processes plotted in faint lines. The solid blue/green traces are the inferred posterior means with ± 1 posterior standard deviation tubes around them. B: True and inferred latent SDE trajectory for the same example trials as in A. The red traces represent the posterior means with ± 1 posterior standard deviation tubes around them, black traces show the true latent SDE path. The black dots indicate the times when observations of $\mathbf{y}$ were made. C: True and learnt dynamics together with the learnt fixed-point locations and tangent lines. Stable fixed points are shown in black, unstable ones in magenta. The uncertainty about the fixed point observation is illustrated using grey error bars representing ± 1 standard deviation. Only the additional fourth fixed point is associated with high uncertainty. D: True vs. learnt model parameters $\mathbf{C}$ and $\mathbf{d}$.

The dynamics are defined by the stochastic map and the output is simply a noisy identity mapping, making inference simple.

$$x(t+1) = rx(t) - x^3(t) + \varepsilon_t^x \tag{4.45}$$

$$y(t) = x(t) + \varepsilon_t^y, \tag{4.46}$$

where $\varepsilon_t^x \sim \mathcal{N}\left(0, \sigma_x^2\right)$ and $\varepsilon_t^y \sim \mathcal{N}\left(0, \sigma_y^2\right)$ iid. I then examine how varying the parameter r affects the learned fixed points. For each parameter setting in the range $0.5 \leq r \leq 2.25$ I trained the system using 32 trials, lasting 20 time steps each, with $\sigma_x = 0.2$ and $\sigma_y = 0.05$, and the initial condition close to zero $x(0) \sim \mathcal{N}(0, 0.001^2)$. I then fit the FP GP-ADF model to the data with 16 inducing points, with locations initialised automatically to equidistantly span the data for each dataset; and using 5 fixed point as parameters (overestimating the true numbers of 1 or 3 depending on the value of r), letting the ARD formulation determine the number of fixed points present in the system.

I first confirmed, that the method indeed captures the available data well for various values of

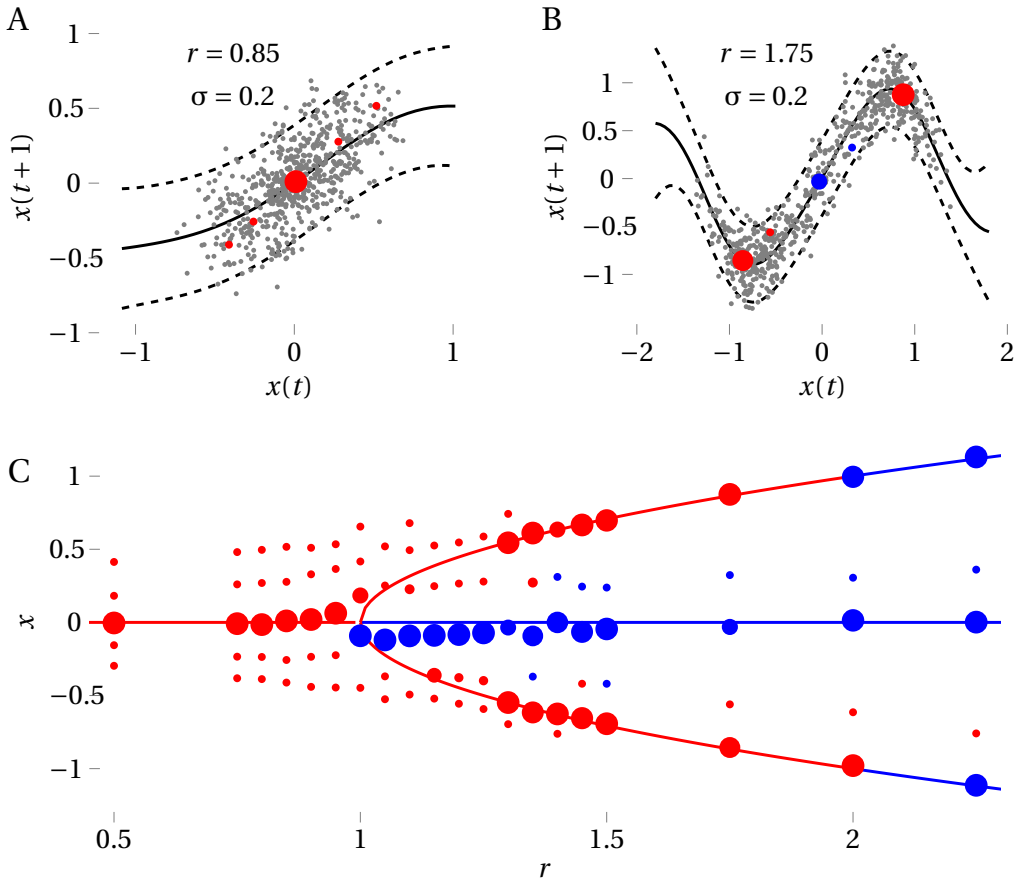


Figure 4.2: Stochastic bifurcation experiment. *A.* The posterior fit of the Fixed Point Sparse Gaussian Process on simulated data from equation 4.45. Small grey points represent the training data. The solid line is the posterior mean, with the dashed lines indicating the posterior standard deviation at $2\sigma_{\text{post}}$. The red and blue circles are the learned fixed point parameter locations. Red colour indicates stable fixed points with the learned Jacobian $|J| < 1$, and blue ones are unstable with $|J| > 1$. The circle size indicates the strength of belief in the fixed point, the larger the size, the smaller the learned σ^{s} uncertainty is. *B.* Same as *A*, with $r = 1.75$, after the bifurcation. *C.* Bifurcation plot of the stochastic system. The solid lines represent the analytic solution for fixed point locations in the noiseless system, when $\sigma_{\text{x}} = 0$. The noiseless fixed points are at $x = 0$ and when $r > 1$ at $x = \pm\sqrt{r-1}$, with the color indicating their stability. The circles represent the learned fixed point locations [y axis position], for a dataset simulated at a particular value of r [x axis position].

the bifurcation parameter r, as shown in figure 4.2 A and B. I then create the so-called bifurcation plot, figure 4.2 C, based on the learned parameter values. The fixed points identified by FP GP-ADF truthfully track the expected location and stability, while successfully recovering the true number of fixed points. Consistently with previous findings on similar systems (Diks and Wagener 2006), we indeed find that noise shifts the bifurcation towards larger values of r, and when the distances of the noiseless fixed points are on the order of the transition noise σ_{x}, the random fixed points are not detectable from data.

4.3.3 Chemical reactor dynamics

To test the ability of the FP GP-SDE algorithm to recover realistic and complex dynamics from well-understood real life systems, we turned towards chemistry, in particular the dynamics of the iodate-AS(III) system under imperfect mixing, described by Ganapathisubramanian (1991). This system is well-approximated by two coupled first-order ODEs describing the time evolution of the concentrations of two iodate species, I^- and IO_3^- in a two-compartment reaction chamber. For certain parameter settings this system exhibits complex multi-stable dynamics, giving rise to three stable and two unstable fixed points. To simulate realistic data, we took the absorption spectra of the two iodate species provided by Kireev and Shnyrev (2015), and the simulated output thus resembles an absorption spectrum that is a linear mixture of the individual spectra, weighted by the concentrations, and corrupted by low levels of Gaussian measurement noise.

The complete system is described in its parametric form[11] as

$$
\begin{aligned}
\mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_\mathrm{A}}{dt}\right] &= \left(k_\mathrm{a}\lfloor \mathrm{I}^- \rfloor_\mathrm{A} + k_\mathrm{b}\lfloor \mathrm{I}^- \rfloor_\mathrm{A}^2\right)\left(S_0 - \lfloor \mathrm{I}^- \rfloor_\mathrm{A}\right) \\
&\quad + \frac{F_1\lfloor \mathrm{I}^- \rfloor_0}{V_\mathrm{A}} - \frac{(F_3+F_4)\lfloor \mathrm{I}^- \rfloor_\mathrm{A}}{V_\mathrm{A}} + \frac{F_4\lfloor \mathrm{I}^- \rfloor_\mathrm{D}}{V_\mathrm{A}} \\
\mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_\mathrm{D}}{dt}\right] &= \left(k_\mathrm{a}\lfloor \mathrm{I}^- \rfloor_\mathrm{D} + k_\mathrm{b}\lfloor \mathrm{I}^- \rfloor_\mathrm{D}^2\right)\left(S_0 - \lfloor \mathrm{I}^- \rfloor_\mathrm{D}\right) \\
&\quad + \frac{F_4\lfloor \mathrm{I}^- \rfloor_\mathrm{A}}{V_\mathrm{D}} - \frac{F_4\lfloor \mathrm{I}^- \rfloor_\mathrm{D}}{V_\mathrm{D}} \\
\mathbf{x} = \begin{bmatrix} \lfloor \mathrm{I}^- \rfloor_\mathrm{A} \\ \lfloor \mathrm{I}^- \rfloor_\mathrm{D} \end{bmatrix} &= \begin{bmatrix} S_0 - \lfloor \mathrm{IO_3}^- \rfloor_\mathrm{A} \\ S_0 - \lfloor \mathrm{IO_3}^- \rfloor_\mathrm{D} \end{bmatrix} \qquad f(\mathbf{x}) = \begin{bmatrix} \mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_\mathrm{A}}{dt}\right] \\ \mathbb{E}\left[\frac{d\lfloor \mathrm{I}^- \rfloor_\mathrm{D}}{dt}\right] \end{bmatrix} \\
d\mathbf{x} &= f(\mathbf{x})dt + \boldsymbol{\sigma}_\mathrm{x} d\mathbf{w} \\
\mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \boldsymbol{\varepsilon}_\mathrm{y} \qquad \boldsymbol{\varepsilon}_\mathrm{y} \sim \mathcal{N}(0, \sigma_\mathrm{y}\mathbf{I})
\end{aligned}
\tag{4.47}
$$

where stochasticity arises due to small volumes and low concentrations, and thus low numbers of molecules. Fl and F2 correspond to inflow into the main reactor of I^- and IO_3^-, respectively. F3 is the total flow rate out of the main reactor. F4 is the volume flow rate at which the contents of the major reactor are pumped into the minor reactor, and vice versa. V_A is the volume of the major reactor, and V_D is the volume of the minor one. k_a and k_b are kinetic rate constants., and $S_0 = \frac{\lfloor \mathrm{I}^- \rfloor_0 + \lfloor \mathrm{IO_3}^- \rfloor_0}{2}$ is the average of ionic concentration in the inflow. The parameter settings of the simulation for the results shown in figure 4.3 were as follows

[11] Please refer to Ganapathisubramanian (1991) for the meaning of the individual parameters and how the below system was derived.

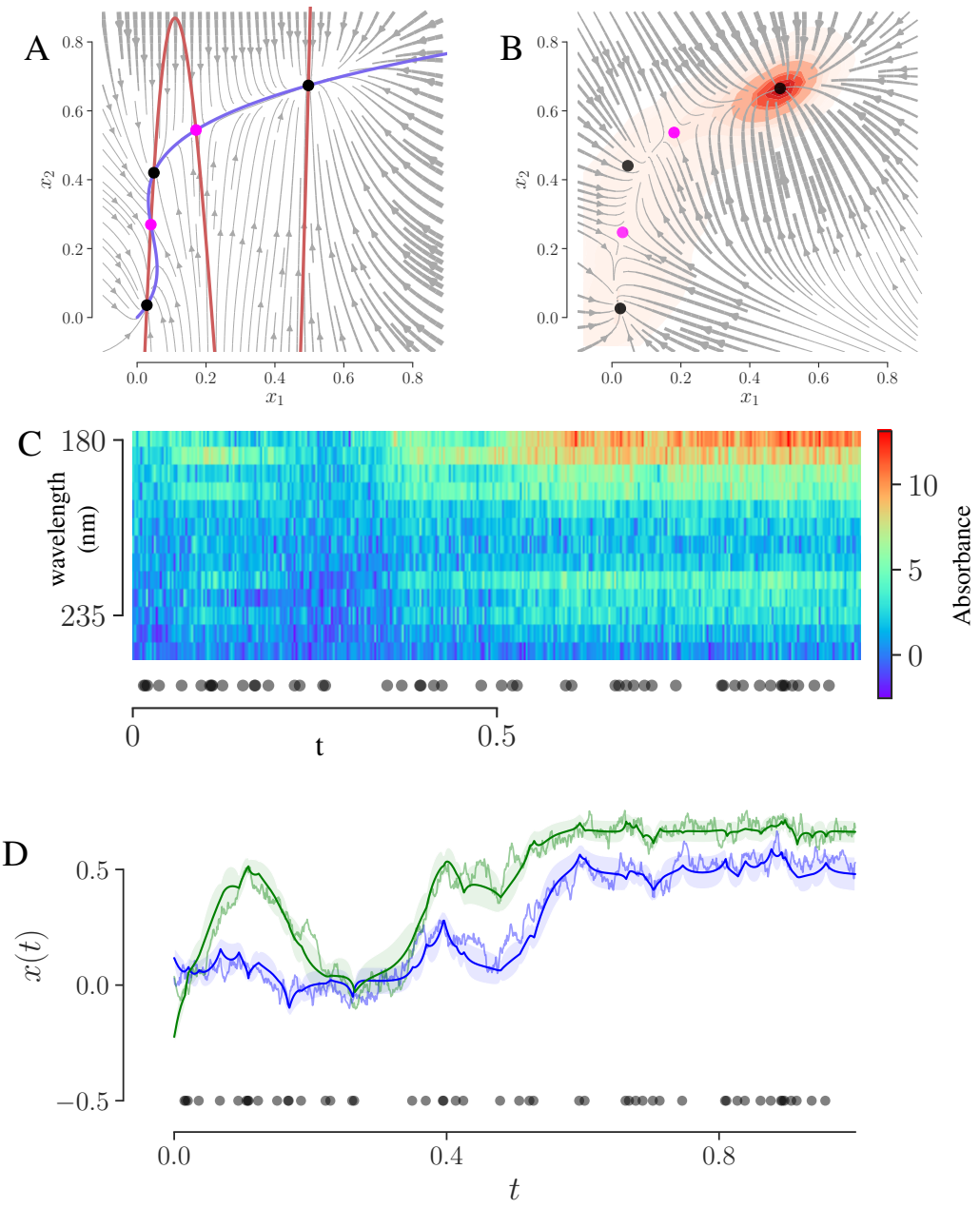


Figure 4.3: Multistable chemical reaction dynamics. A: Streamline plot of the concentration dynamics for two species of iodine, together with the nullclines and fixed points. Stable fixed points are black, unstable ones are magenta. B: Learnt dynamics and fixed points with stability determined by eigenvalues of learnt Jacobian matrices. Increasing uncertainty in the fixed-point observation is indicated by higher transparency of the dot. The red contour plot illustrates the density of latent path locations across all trials used for training. C: Example spectroscopy measurements (output process) across light wavelengths (nm). D: Example true latent path together with the inferred posterior mean and ± 1 standard deviation tubes for each latent dimension on the same trial as C. The black dots indicate the time points at which measurements were taken.

Noise free dynamics	$\lfloor \mathrm{I}^- \rfloor_0 = 4.4 \times 10^{-5}$	$k_0 = 2.7 \times 10^{-3}$
	$V_\mathrm{A} = 4 \times 10^1$	$F_4 = 3.25 \times 10^{-3}$
	$V_\mathrm{D} = 1$	$F_3 = k_0 V_\mathrm{A}$
	$k_\mathrm{a} = 2.1425 \times 10^{-1}$	$F_1 = \frac{1}{2} F_3$
	$k_\mathrm{b} = 2.1425 \times 10^4$	$F_2 = \frac{1}{2} F_3$
	$S_0 = \frac{1}{2}\left(\lfloor \mathrm{I}^- \rfloor_0 + 1.42 \times 10^{-3}\right)$	
Stochastic simulation	$dt = 1 \times 10^{-3}$	$\sigma_\mathrm{x} = 0.3 \times 10^{-3}$
	$\sigma_\mathrm{y} = 1 \times 10^{-3}$	

We finally fit the FP GP-SDE model to simulated data of 20 trials with different initial conditions, but the same parameter settings, and sampled a 13 dimensional spectroscopic observation at 50 random time points on each trial (see figure 4.3 C), resulting in a $20 \times 50 \times 13$ dimensional dataset. As a result of the fit, we recovered the correct dynamical portrait, fixed point locations and stability as shown in figure 4.3 A and B as well as inferred stochastic latent dynamics correctly, even ones that switch between the various fixed points within a single trial, as shown in figure 4.3 D.

4.3.4 Neural population dynamics

This example demonstrates the FP GP-SDE algorithm under multivariate point-process observations. We model the intensity functions of the nth output process as $\eta_\mathrm{n}(t) = \exp(\sum_{\mathrm{k}=1}^{\mathrm{K}} C_{\mathrm{nk}} x_\mathrm{k}(t) + d_\mathrm{n})$, where $\mathbf{x}(\cdot)$ is a K-dimensional stochastic dynamical process, $\mathbf{C}$ and $\mathbf{d}$ represent an affine mixing of the underlying process (information) into a high dimensional observation space (neurons that encode the information), a typical setup in dynamical system models of neural population observations. Conditioned on the intensity function, the $\phi(n)$ observed event-times $\mathbf{t}^{(\mathrm{n})}$ are generated by a Poisson process – representing the neural spiking – with log likelihood

$$\log p(\mathbf{t}^{(\mathrm{n})} | \eta_\mathrm{n}) = -\int T \eta_\mathrm{n}(t) dt + \sum_{\mathrm{i}=1}^{\phi(\mathrm{n})} \log \eta_\mathrm{n}(t_\mathrm{i}^{(\mathrm{n})}) \tag{4.48}$$

In contrast to the Gaussian observation case, the first term in the log-likelihood above is continuous in $\eta_\mathrm{n}(t)$ and the absence of events is also informative towards the underlying intensity of the process.

This observation process is common in neural data analysis, where electrophysiological recording results in a set of spike-times of a population of simultaneously recorded neurons jointly embedded in a circuit involved in performing a computation. In fact, studying neural population activity as a dynamical system has gained increasing traction in the field of neuroscience in recent years –

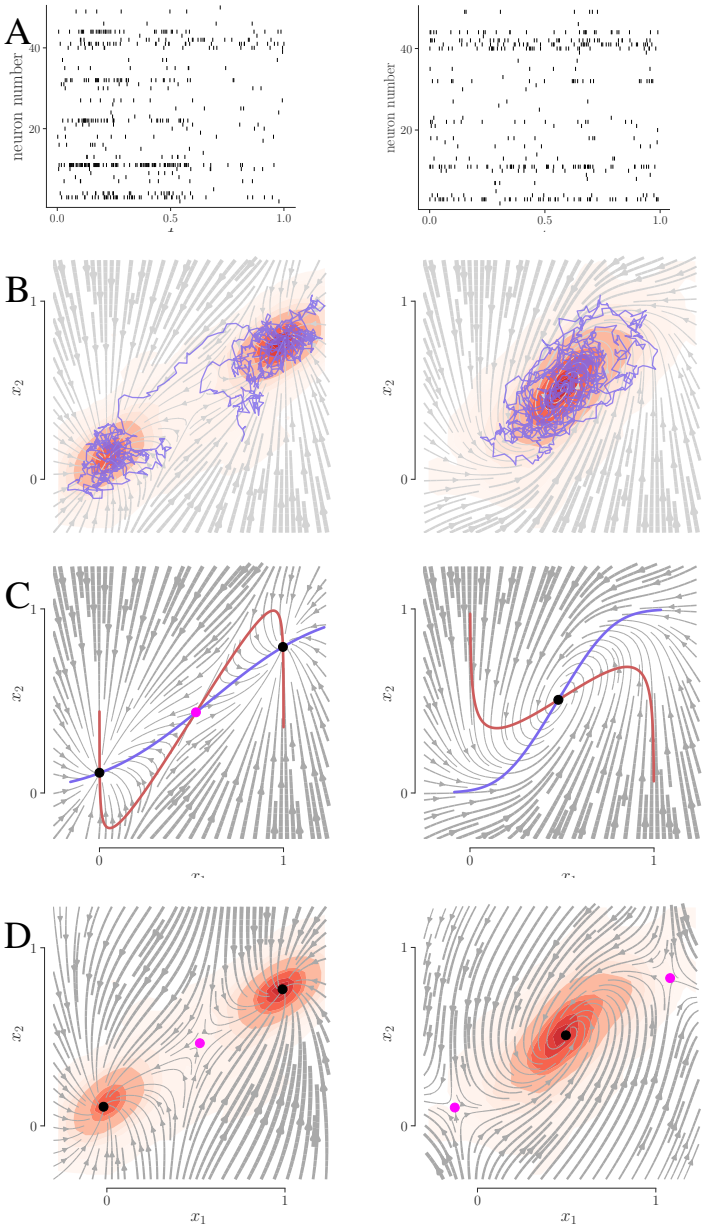


Figure 4.4: Neural population dynamics. Left: simulations with parameter settings $b_1 = 1.9, b_2 = 0.5, z_1 = 3, z_2 = 3.9, w_{11} = 10, w_{12} = 5, w_{21} = 9, w_{22} = 3$. Right: simulations with parameter settings $b_1 = 0.4, b_2 = 0.6, z_1 = 1.7, z_2 = 7, w_{11} = 20, w_{12} = 16, w_{21} = 21, w_{22} = 6$. A: Raster plot of the observed spike times for a population of 50 neurons for an example trial. B: Example paths through the two-dimensional latent space on the same trial as A, together with a density plot of latent locations visited across all trials that were used for learning the dynamics, shown in red. C: Streamline plots of the true dynamics together with their fixed points and nullclines for each latent dimension. Stable fixed points are black, unstable ones are magenta. D: Same density plots as in B together with streamline plots of the learnt dynamics and learnt fixed points. The fixed point stability is shown as indicated by the eigenvalues of the learnt Jacobian matrices.

see Macke et al. (2011), Pandarinath et al. (2018b), and Shenoy, Sahani, and Churchland (2013), including my own contribution in Park, Bohner, and Macke (2015) – and data analysis methods that can obtain such descriptions are thus of great interest.

We simulate a two-dimensional ($K = 2$) latent SDE using the dynamics $f_k(\mathbf{x}) = -x_k + \sigma_k(w_{k1}x_1 - w_{k2}x_2 - z_k)$ for $k = 1, 2$, where $\sigma_k(x) = (1 + \exp(-b_k x))^{-1}$. Depending on the choice of parameters b_k, w_{kj} and z_k the dynamical system will exhibit different properties. We explore the two regimes where the system either has two stable and one unstable fixed points (figure 4.4 C left) or exhibits a single stable spiral (figure 4.4 C right).

We simulate data from 50 neurons on 25 trials for each of the two parameter regimes for b_k, w_{kj} and z_k. Figure 4.4 A shows example neural spike trains under the two regimes. Figure 4.4 B illustrates sample paths through the latent space under the different dynamical regimes, together with the density of latent locations visited across all trials. In both settings, we initialise our algorithm with three fixed points and inducing points placed on an evenly spaced 4×4 grid in $(-0.25, 1.25)$, and hold the parameters relating to the output mapping constant. Figure 4.4 C and D show the true and the estimated flow fields in both settings, together with the location of the fixed points and their stability as indicated by the eigenvalues of the Jacobian matrices. In both settings, the FP GP-SDE method successfully recovers the main qualitative distinguishing features of the dynamics. In the regime where the dynamics are conditioned on three fixed points but the generative system only contains one, the two additional fixed points will either be associated with higher uncertainty or move to regions where no or little data was observed, as indicated by the superimposed density plots.

4.3.5 Mutually inhibiting neural populations observed via calcium imaging

Lastly, I demonstrate potential use of the FP GP-ADF algorithm on the type of data that we discussed in chapters 2 and 3, recorded from a set of neurons via calcium imaging. To simulate such data that could potentially be gathered with techniques becoming available within the next couple years, I chose a model of mutually inhibiting neural population by Machens, Romo, and Brody (2005). My choice was motivated by the fact that not only does the model exhibit interesting parameterised dynamics, similarly to the examples above, but it was also carefully set up, such that individual simulated neurons' characteristics matched those recorded from monkey pre-frontal cortex.

They created a simplified spiking simulation model with realistic integrate-and-fire neurons, structured inhibitory connectivity between two populations, and strictly external excitation[12] (see figure 4.5 A). This network was parametrised to be consistent with observed neural characteristics from the Rhesus monkey brain, with original neurophysiological data and the original working memory-decision making task published by Romo et al. (1999). Considering the increasing effort

[12] One might think of these as two groups of interneurons

for making calcium imaging possible in primate brains (O'Shea et al. 2017; Seidemann et al. 2016), and that I participated in the effort (Trautmann et al. 2019) for analysing the initial results of monkey calcium imaging, I decided to simulate the observed data as if the spiking network was viewed through a calcium reporter and subsequent fluorescent imaging.

As in our study monkeys were injected with an AAV1-CamKIIA-GCaMP6f virus construct, I used the GCamp6f kinetics as identified in vivo[13] as 45 ms rise time, 142 ms decay time and 19% $\Delta F/F_0$ fluorescence increase (see Chen et al. (2013a) Supplementary table 3), and thus I use the so-called fast rise, exponential decay (FRED, by Norris et al. (2005)) equation to describe the calcium response after a spike (similarly to Lütcke et al. (2013), and see the inset of figure 4.5 C for the resulting shape):

$$f^{\text{FRED}}(t - t_{\text{spike}} \mid \tau_{\text{rise}}, \tau_{\text{decay}}, A) = A * \left(e^{2\frac{\tau_{\text{rise}}}{\tau_{\text{decay}}}} \right)^{1/2} \left(e^{-\frac{\tau_{\text{rise}}}{t - t_{\text{spike}}} - - \frac{t - t_{\text{spike}}}{\tau_{\text{decay}}}} \right). \tag{4.49}$$

For the purposes of this simulation, non-linear saturation effects were not considered, to enable us focusing more precisely on discovering the latent dynamics, however for real recordings they may be taken into account both for the simulation and during model inference/learning (e.g. by Speiser et al. (2017)). I simulated 30-30 spiking neurons from both populations A and B, then convolved their spike trains with the FRED curve, to obtain the 'noiseless' calcium traces shown in figure 4.5 C.

The simulation then takes noisy snapshots of the instantaneous fluorescence every 125 ms (8 Hz), as is typical in calcium imaging to create the final dataset (see figure 4.5 C, markers). The additive Gaussian observation noise level was set heuristically to match typical effective noise characteristics after the intracellular variations, and various data recording and preprocessing steps (see chapter 2 for a detailed description of the real noise process). For exact details of the simulation, see the original paper (Machens, Romo, and Brody 2005), their detailed supplementary explanation and code, and my simulation and model fitting code available online[14].

The model fit results are shown in figure 4.6, and were fit to 40 simulated trials of 60 neurons' fluorescence, observed at 17 time points over a 2 second trial, with initial conditions set to be at $(1, 1)$. The model behaves differently at the two shown external excitation (E_{ext}) levels, in one case implementing a 'decision making' scheme (although here I used unbiased inputs, and only noise was integrated to serve as the basis of such decision) and in the other a 'loading' scheme, with all traces converging to a single stable fixed point (see Machens, Romo, and Brody (2005) about the naming of these behaviours). FP GP-ADF, initialised with 5 potential fixed points, successfully recovers the dynamics, the correct number of fixed points (as identified by the fixed point uncertainty parameter σ^{s}) and their stability (including learning a saddle node in figure 4.6 A), demonstrating its potential

[13] In mice, monkey data is unavailable, some population average hints in Seidemann et al. (2016)

[14] `https://github.com/gbohner/gpdm-fixpoints`

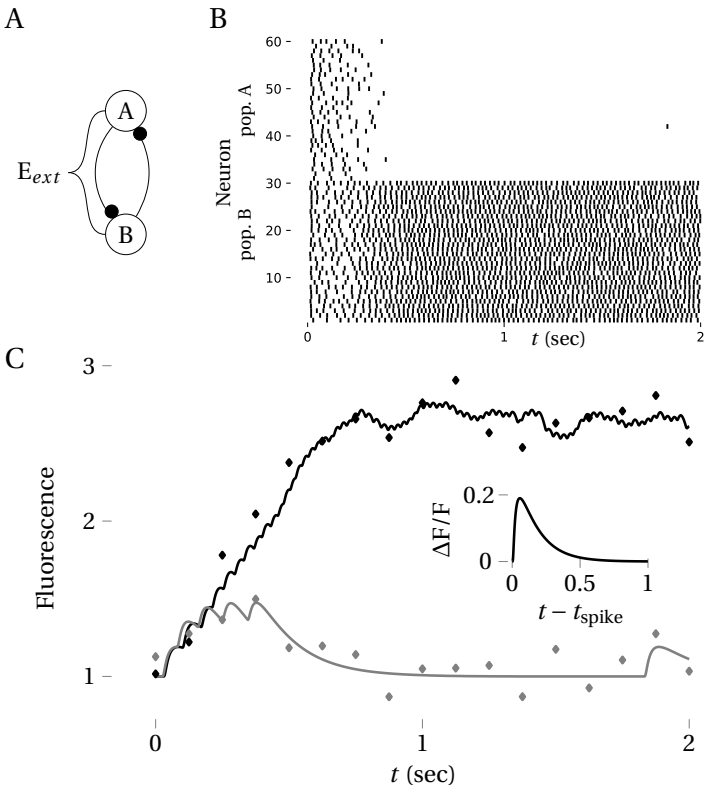


Figure 4.5: Simulation of mutually inhibiting neural populations with calcium observations. A: Circuit diagram of two mutually inhibiting neural populations under external excitation E_{ext}. B: Example raster plot during 'decision making'. Due to the mutual inhibition, in this regime we observe winner-take-all dynamics, and thus population A is silenced. C: Same trial as in B, showing two example neurons. Solid lines show the instantaneous fluorescence as a result of spikes being convolved with GCamp6f observation kinetics. The noisy and sparse observed data is shown by the markers. Inset: The fast rise exponential decay model output used as the convolution kernel.

usefulness in a calcium imaging setting.

4.4 Discussion

In this chapter I described two novel algorithms, FP GP-ADF and FP GP-SDE that learn interpretable models of latent stochastic dynamical systems. The varied applications showcase the promise of the proposed idea of conditioning the learned transition functions on their fixed points and also demonstrate the first applications of learning GPSSM models of realistic neural-like data. Although I do hope that these initial efforts will lead to a new class of interpretable machine learning models applied widely to real neural recordings, there are still a few hurdles that need to be overcome for the widespread usage of these type of algorithms, and thus will need to be subject to further research.

Firstly, for now we assumed the dynamical systems under investigation are autonomous; there

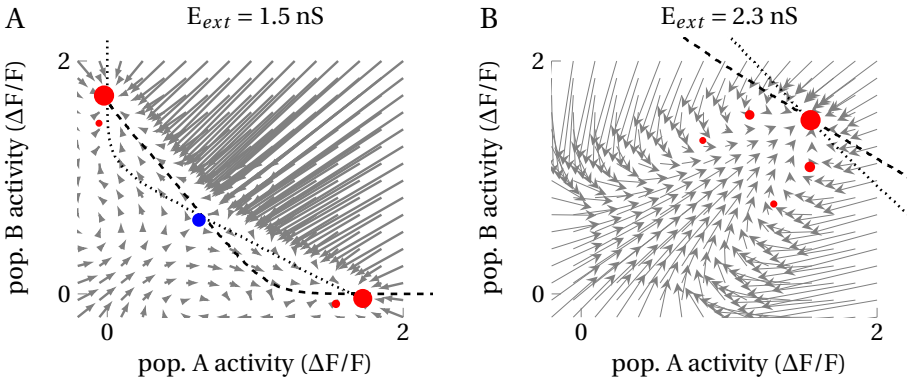


Figure 4.6: Model fits of FP GP-ADF to the simulated data. Dashed line are the numerically computed true nullclines of the simulated model. Grey arrows are the learned dynamics, and markers show the location of learned fixed points. Marker size indicates the inverse of σ^s (the large the point is, the more likely it is to be a fixed point). A: Under low external inputs the system exhibits winner take all dynamics. FP GP-ADF correctly identifies the 3 real fixed points, as well as their stability (red is stable, blue here is learned as a saddle node, with one stable and one unstable eigenvalue of the associated $\mathbf{J}$ matrix.). B: Higher excitatory input causes a change in model behaviour, resulting in a single stable fixed point, that is indeed correctly identified.

is no input driving the system, or at least the input is stationary over many trials, allowing us to learn the input-driven dynamics and treat changes in input as qualitative changes of the dynamics themselves, such as in section 4.3.5. In this view, the input is changing the effective transition function $f(\cdot)$, which may be undesirable. In previous work on GPSSMs used in the non-linear control literature (e.g. McHutchon (2014)), they extend the transition function to map from the state and inputs jointly to the new state, $\mathbf{x}_{\text{new}} = f(\mathbf{x}, \text{inputs})$. Although we could have added this straightforwardly, this extension assumes that the inputs are known to the model at each point in time, as they are set by the experimenter. This is often the case in chemistry or robotics application of non-linear control. However, in many real systems, especially in neural recordings that usually investigate a small, local part of a highly interconnected network, the system that we observe is definitely non-autonomous, but we also do not have access to the inputs driving it. Simultaneously learning dynamics implemented by the measured system as well as inferring the inputs driving it is currently an interesting open research question (Soldado Magraner 2018; Sussillo et al. 2016). The major difficulty is in attempting to ensure that the inferred inputs do not explain away the dynamics that actually are implemented, and thus usually there are several constraints placed on the inferred inputs, such as they vary slowly in time or vary little across trials, or map into a particular subspace of linear dynamics (Duncker et al. 2017; Soldado Magraner 2018). A further issue with inferring inputs is the partitioning of the innovations noise. We do expect to have noise in the transitions due to inherent stochasticity and model mismatch, and allowing for the inference of inputs will explain away some of this noise. Overall, I believe inferring inputs along with learning the dynamical system

is a powerful idea, but it should be implemented with care, and with very specific considerations and existing insight to the system under investigation. A potentially more generally applicable approach would be to accept the fact that unknown inputs change the learnt dynamics transition function $f(\cdot)$, and allow for it to indeed vary in time, such that we learn a full description of the transition process across trials and time, $f^{\mathrm{m}}(\cdot,t)$. This is of course a very flexible description, and may be difficult to learn, but it can be constrained in several ways, either by learning a switching process with a few fixed transition functions and a Markov Chain switching between them, or by assuming that it varies slowly in time, placing another Gaussian Process over the parameters of $f(\cdot)$, ensuring they change smoothly in time. Both of these ideas have already been implemented for linear latent dynamics (Park, Bohner, and Macke 2015; Petreska et al. 2011), and in my view they show more promise than the attempt to infer the unknown inputs to the system. With regards to their interpretability, they would result in the learning of single-trial bifurcations as we follow the now time-varying fixed points, essentially showing qualitative changes in the dynamics, just as they are being changed by the unknown inputs, and resulting in pictures similar to figure 4.2 C, but with time within trial being on the x-axis.

A second set of concerns is regarding the application of the proposed algorithms to real world datasets. Although the examples shown in section 4.3 demonstrate the reliability of the algorithms on simulated data – the results match the ground truth fixed points extremely well, and approximate the global transition function well too – this is all reliant on the available trials sufficiently exploring the whole of the dynamical portrait. This is best illustrated in figure 4.4 B-D, where the density plots showing data availability correlate strongly with 'how well' the true transition flows are estimated in different parts of the state space. Of course this is a rather general issue, it is well known that we can only estimate functions where we have samples from them (unless we assume certain trends or periodicity), and Gaussian Processes are well-suited towards not giving overly confident function estimates away from data. The larger issue is that in order to learn more complete dynamical portraits, we ideally need rather noisy latent paths exploring large chunks of the state space, whereas current experimental philosophy aims to produce as noiseless and 'reproducible' single trial traces as possible, not ideal for our purposes. A related question is a more precise definition of the 'how well' we learn transition functions. For now we only provided visual comparisons of the true and estimated dynamical portraits, and have shown that the fixed point locations and stabilities are indeed what we expect them to be, based on the crossings of nullclines. However, I feel we need further clarifications on how to think of these results, especially when considering real world applications to unknown systems. The potential answer is two-fold, with both directions requiring further work. From a computational point of view, when the ground truth dynamics are known – such as in the simulated experiments above – one ideally needs to define a metric of how much the learnt dynamics differ from the ground truth. This should ideally be a metric that submits to hypothesis testing, so we can make statements about learning the 'correct' dynamics matching the ground truth with some

probability, given the system and the amount of data. To my knowledge such metrics are not currently available for learnt stochastic dynamical systems, and are highly non-trivial to come up with. A potential avenue would be defining similarity metrics over arbitrary dynamical systems, such as Smola, Vidal, and Vishwanathan (2004) did for linear dynamical systems, and should be explored in the future. Fortunately, from an experimental point of view, when we are applying our methods to real world data, the answer may be a bit clearer, although not necessarily easier to implement. I believe the correct procedure for using these computational methods would be to treat them as iterative hypothesis generators. That is, given an initial dataset, one would apply to algorithms to sketch a dynamical portrait and showing the locations and stabilities of proposed fixed points given multiple trials for each setting of experimentally available parameters (such as in sections 4.3.1 and 4.3.4). With this added information, the next set of experiments can be designed to confirm the locations and stabilities of the proposed fixed points, or to explore new areas of the latent space to get a more complete sense of the dynamics. Therefore, I believe that already in their current form, the proposed algorithms can play a strong role in assisting experimentalists to get more useful data out of their experiments, by pointing out dynamically interesting features in a very directly accessible and human-interpretable way.

Thirdly, I need to discuss scalability of both the computational costs as well as the ease of interpretability. The proposed methods scale very well with regards of the dimensionality of the observation space due to the instantaneous output mapping from latent state to outputs, therefore working with even thousands of simultaneously recorded – or even stiched, as in (Nonnenmacher, Turaga, and Macke 2017) – neural time series that are now becoming available due to improved experimental techniques, is not going to cause issues. However, scalability with regards to latent space dimensionality D is an open question. Computationally, the number of free parameters in the Jacobians scale quadratically with D, and could quickly lead to difficult-to-learn over-parametrised systems. What's worse, the number of inducing points required to tile a D dimensional space sufficiently densely scales exponentially with D, quickly leading to huge kernel matrices k^{zz}, although this could be well-mitigated by further approximations, such as the KISS-GP – introduced by Wilson and Nickisch (2015), and also described here in section 2.2.2 – for inducing point locations $\mathbf{Z}$ on an equidistant grid. The scaling with D with regards to interpretability is potentially more worrying. All our example applications were using only up to 2 dimensional latent spaces, as the descriptive language for local linearisations around fixed point is rather complete in this case, stable or unstable nodes, saddles and spirals are well understood and intuitive to many, and also the transition functions can easily be visualised by regular $x_t \to x_{t+1}$ plots in 1D or by quiver plots in 2D. Already in three dimensions the visualisations (and thus interpretation) become difficult, and the classification of fixed point types – aside from the stable and unstable nodes – is less clear. Therefore instead of attempting to use $D \gg 2$ latent dimensionalities for the proposed methods to capture more variability in the latent space, I believe one would want to implement some of the ideas described in the

paragraphs above and use a two-dimensional time-varying $f(\cdot,t) : \mathbb{R}^{2+1} \to \mathbb{R}^2$ transition function to retain the ease of interpretability.

Finally, a minor point regarding the applications of latent stochastic dynamical system to calcium-imaging data, as done in section 4.3.4. Although the proposed algorithms can indeed be applied to calcium imaging observations, by treating the mapping from latent space to observations as instantaneous, the identified latent dynamics are confounded by the time course of the calcium reporter, as shown in figure 4.5. This time-convolution introduces correlations over time in the neural observations within the output space, whereas most existing state space models – including the ones here – assume time-correlations are only present in the latent computational trajectories, not in the observations. This issue is fairly specific to calcium imaging of neurons, and thus has not been thoroughly explored by other communities. Computational neuroscientists have indeed started working on these correlated-output models (see Speiser et al. (2017)), and I do think they become very necessary for high frame-rate calcium imaging videos, where the time between frames is much less than the width of the calcium reporter's response peak, or $\Delta t \ll \tau_{\text{rise}} + \tau_{\text{decay}}$. In lower frame-rate applications, including the one presented here, this issue is much less severe, and should not affect the learnt dynamics or the inferred computational trajectories too much.

To summarise, I presented here a novel way of doing data-driven stochastic bifurcation analysis, and gaining previously unavailable scientific insight into various real-world-like systems. Although this type of analysis is still in its infancy, it can already be a useful tool for scientists to aid their iterative experimental design, and once some of the current limitations are overcome by solutions I pointed out – or even better ones – it has the potential to become an essential tool in analysing various types of real neural data.

5

Conclusions and further work

All theory is gray, my friend. But forever green is the tree of life.

– Johann Wolfgang von Goethe, *Faust*

Unsupervised machine learning is a powerful tool that bridges the gap between complex generative models and available biological data. It infers unknown parameters of the model by estimating their complex relationships with the recorded data and each other. This inference processes is a somewhat unusual way of scientific understanding, but it allows for defining and constraining models that were previously thought too complicated for practical use. This is demonstrated by the models presented in chapters 2 and 3, which sketch physically-inspired generative models of calcium imaging data. I then use machine learning to estimate the signal transformation introduced by our measurement method, infer the changes in calcium ion concentrations – the signal we wanted to measure – and demix superposed signals from multiple sources, to reveal the activity of single neurons. Chapter 4 shows a different use of unsupervised machine learning, one that focuses on the human understanding of neural data eo ipso, on its own account. In this case, the suggested models are known to be incomplete and wrong, but they provide an interpretable picture of the information in the data, without making assumptions on how it was generated. Ultimately, these models, algorithms, and applications showcase the usefulness of machine learning in the understanding of neural data; they may be used to analyse existing neural data, help in designing more informative experiments in the future, and inspire other computational and theoretical researchers to build more complex models, relying on the power of inference to evaluate their utility.

In terms of further work, the issues, shortcomings and potential extensions of the proposed models and methods are already discussed in the individual chapters, so here I focus on the more general challenges in the future of neural data analysis. As discussed in chapter 1, throughout the history of cell-resolution neural data recording, novel experimental techniques always create new data analysis challenges. Although the algorithmic solutions to these problems appear faster nowadays than they used to, there are still significant temporal gaps between the introduction of a new experimental technique, the implementation of reliable computational solutions, and the adoption

of automated analytic programmes into standard analyses. These time gaps could be significantly reduced by closer collaboration between the developers of experimental methods and software solutions. In particular, the adoption of excellent models and algorithms is often lacking, owing to either limitations of their existing implementation – such as insufficient scalability, and inadequate ease of use – or simply missing awareness from potential users. These issues could be reduced, if algorithms were co-developed with the experimental method, then validated, and shipped together with the recorded data. The latter is common practice in microscopy and in bioinformatics, and is now appearing in neuroscience via the excellent Neurodata Without Borders project that aims to standardise datasets, and include all transformations and analyses alongside the recorded data. This project could make neural data analysis more accessible and reproducible, improving the rate and robustness of scientific progress in the field. Therefore, an important task for the future for all developers – including me – is to help it succeed by reimplementing existing models and algorithms, such that they comply with the proposed standards; and also to engage in discussion to make those standards flexible, yet lasting.

Lastly, our understanding of the brain constantly improves and changes, and we need to remember the limitations of the cell-resolution neural data analyses described in this thesis. Many models treat the neuron as the atom of information processing in the brain, and these models have indeed been successful at describing numerous perceptual and behavioural phenomena. However, we must not forget that the cell itself is a living organism, and intracellular components likely play a major role in the storage and processing of information in the brain; furthermore, the electric spikes are only one type of communication between neurons, with others including gap junctions as well as the non spike-induced release and uptake of small molecules. Although memories were previously thought to be well represented by stable electrical activity patterns of neurons – and implemented by the strengths of their synaptic connections – it is likely that the brain stores and processes significantly more information, than apparent via electrical activity. In particular, recent experiments demonstrated that the transfer of learned behaviour – such as fear conditioning – can be passed down several generations, showcasing that learned information was somehow transferred to a single sperm cell; moreover, that cell was capable of sufficiently representing the information, evidenced by the adult animals in subsequent generations behaviourally expressing the particular memory. This process is largely not understood; the intracellular representation of memories is an open research question, and its investigation might be a major avenue in future neuroscience research. The particular representation might be of epigenetic, or even genetic nature – to quote Barbara McClintock's 1983 Nobel Prize lecture, the genome is "a highly sensitive organ of the cell", capable of "sensing the unusual and unexpected events, and responding to them".

Regardless of the level of investigation, models that best reflect a state-of-the-art understanding of complex systems are bound to have unknown, and experimentally inaccessible parameters, ensuring that the main ideas of this thesis – insight via inference – will continue to be relevant to scientific progress.

Appendices

Appendix A

Supplemental material

This chapter provides links to electronic material related to this thesis.

Chapter 3 - Supplemental videos. Videos of the full time course of zoomed in regions of the preprocessed data shown in figure 3.13 (a-b) are available as AVI video files to be downloaded from my website:

- Zoomed in region in figure 3.13a, (7.9 MB):
  `http://www.gatsby.ucl.ac.uk/~gbohner/Thesis/savedVideos/supplemental_video_figure.3.13.a.avi`
- Zoomed in region in figure 3.13b, (9.7 MB):
  `http://www.gatsby.ucl.ac.uk/~gbohner/Thesis/savedVideos/supplemental_video_figure.3.13.b.avi`
- Full field of view in figure 3.12c, (84 MB):
  `http://www.gatsby.ucl.ac.uk/~gbohner/Thesis/savedVideos/supplemental_video_figure.3.12.c.avi`

Computer code. All my computer code that implements the definition and estimation of models proposed in this thesis – as well as the code that creates visualisations of the results – is publicly available on Github. The code particularly relevant to each chapter was collected into repositories created for this purpose:

- Chapter 2 – `https://github.com/gbohner/thesis-code-chapter2-preprocessing`
- Chapter 3 – `https://github.com/gbohner/thesis-code-chapter3-chomp`
- Chapter 4 – `https://github.com/gbohner/thesis-code-chapter4-fpgp`

Appendix B

Notation

Throughout this work the reader encounters mathematical concepts of varying difficulty. As my aim is to communicate any given concept as concisely as possible without introducing any ambiguity into the equations, I felt like I need to adapt the descriptive power of the notation to the needs of the concept.

There I use a mixture[1] of the *common matrix-vector* notation for tensors of up to order two, as well as the *sum* and *indicial* notations for computations that involve higher order tensors. To maintain clarity of which notation is in use at any given time, the reader may use the following guidelines:

- Non-bold objects with or without indices always refer to scalars. For example $\{ a, a_{\mathrm{i}}, A_{\mathrm{ij}}, A_{\mathrm{ijkl}} \}$.
- Bold objects always refer to non-zero order tensors (vectors, matrices or higher order ones). In case indices are present along a bold object, they are used as in indicial notation (see below). For example $\{ \mathbf{a}, \mathbf{A}, \mathbf{a}_{\mathrm{i}}, \mathbf{A}_{\mathrm{ij}}, \mathbf{A}_{\mathrm{ijkl}} \}$.
- For clarity, it is sometimes easier to add a subscript to an object, indicating its "name", that it is different from another object sharing the same base symbol. To avoid confusion of names with indices, I indicate names with an underline, for example $\{ \mathbf{a}_{\underline{\text{first}}}, \mathbf{A}_{\underline{\text{second}}}, \mathbf{A}^{\text{previous}}_{\underline{\text{second}}} \}$. As naming objects via superscripts can not be confused with indicial notation, underlining of superscripts as names may be ommitted.

The main features of the indicial notation, and how they relate to the common and the sum notation most readers are familiar with, are as follows:

1. In a term, the number of free indices - denoted by (possibly indexed) lower case letters of the English alphabet - represent the order of the tensor, with dimensions ordered according to order the in which indices show up in the expression. The range of an index is often not explicitly indicated, but inferred from context.

[1] Each notational style is kept consistent within a particular concept.

Common name	Tensor order	Common notation	Indicial notation
Scalar	0	a	a
Vector	1	$\mathbf{a} \in \mathbb{R}^{\mathrm{D}}$	$\mathbf{a}_{\mathrm{i}}$ i=1...D
Matrix	2	$\mathbf{A}$	$\mathbf{A}_{\mathrm{ij}}$
		$\mathbf{A}^{\top}$	$\mathbf{A}_{\mathrm{ji}}$
Tensor	3	$\mathcal{A}$	$\mathbf{A}_{\mathrm{ijk}}$
	4	$\mathcal{A}$	$\mathbf{A}_{\mathrm{ijkl}}$

2. Repetition of an index within the same term is called a dummy index, and indicates summation over the range of the index. Common linear algebra operations are:

Common name	Common notation	Indicial notation	Sum notation
Scalar product	$\mathbf{c} = a\mathbf{b}$	$\mathbf{c}_{\mathrm{i}} = a\mathbf{b}_{\mathrm{i}}$	$c_{\mathrm{i}} = ab_{\mathrm{i}}$
Inner product	$c = \mathbf{a}^{\top}\mathbf{b}$	$c = \mathbf{a}_{\mathrm{i}}\mathbf{b}_{\mathrm{i}}$	$c = \sum_{\mathrm{i}=1}^{\mathrm{D}} a_{\mathrm{i}}b_{\mathrm{i}}$
	$\mathbf{c} = \mathbf{A}^{\top}\mathbf{b}$	$\mathbf{c}_{\mathrm{i}} = \mathbf{A}_{\mathrm{ji}}\mathbf{b}_{\mathrm{j}}$	$c_{\mathrm{i}} = \sum_{\mathrm{j}=1}^{\mathrm{D}} A_{\mathrm{ji}}b_{\mathrm{j}}$
Outer product	$\mathbf{C} = \mathbf{a}\mathbf{b}^{\top}$	$\mathbf{C}_{\mathrm{ij}} = \mathbf{a}_{\mathrm{i}}\mathbf{b}_{\mathrm{j}}$	$C_{\mathrm{ij}} = a_{\mathrm{i}}b_{\mathrm{j}}$
Matrix trace	$c = \mathsf{Tr}(\mathbf{A})$	$c = \mathbf{A}_{\mathrm{ii}}$	$c = \sum_{\mathrm{i}=1}^{\mathrm{D}} A_{\mathrm{ii}}$
Tensor product	-	$\mathbf{c}_{\mathrm{i}} = \mathbf{A}_{\mathrm{ijk}}\mathbf{B}_{\mathrm{jk}}$	$c_{\mathrm{i}} = \sum_{\mathrm{j,k}} A_{\mathrm{ijk}}B_{\mathrm{jk}}$
	-	$\mathbf{C}_{\mathrm{jl}} = \mathbf{A}_{\mathrm{ijk}}\mathbf{B}_{\mathrm{ikl}}$	$C_{\mathrm{jl}} = \sum_{\mathrm{i,k}} A_{\mathrm{ijk}}B_{\mathrm{ikl}}$

3. Functions applied to an element in indicial notation are understood as acting on each element seperately. When a function is to be applied to the whole tensor, we indicate it by square

brackets.

$$\mathbf{A}^{-1} = [\mathbf{A}_{\mathrm{ij}}]^{-1} \neq \mathbf{A}_{\mathrm{ij}}^{-1}$$
$$eA = e^{[\mathbf{A}_{\mathrm{ij}}]} \neq e^{\mathbf{A}_{\mathrm{ij}}}$$

4. We define the following special symbols:

   The all ones tensor $\mathbf{1}_{\mathrm{ijk...}}$. This may be useful to carry out so-called broadcasting: When the free indices do not agree in terms that are being summed, they may be multiplied by the smallest order all ones tensors, such that the sum can be carried out, e.g. $\mathbf{a}_{\mathrm{i}}$ and $\mathbf{B}_{\mathrm{jk}}$ can not be summed, but we may compute $\mathbf{a}_{\mathrm{i}}\mathbf{1}_{\mathrm{jk}} + \mathbf{1}_{\mathrm{i}}\mathbf{B}_{\mathrm{jk}}$.

   The Kronecker delta tensor

$$\boldsymbol{\delta}_{\mathrm{ijk...}} = \begin{cases} 1 & \text{if } i = j = k = \ldots \\ 0 & \text{otherwise} \end{cases}$$

   Using the Kronecker delta tensor, we may extract the diagonal of a matrix $\mathbf{A}$ as the vector $\mathbf{A}_{\mathrm{ij}}\boldsymbol{\delta}_{\mathrm{ijk}}$, or represent a vector $\mathbf{a}$ as a diagonal matrix via $\mathbf{a}_{\mathrm{i}}\boldsymbol{\delta}_{\mathrm{ijk}}$. We may also implement the element-wise product of vectors $\mathbf{a}$ and $\mathbf{b}$, as $\mathbf{a}_{\mathrm{i}}\mathbf{b}_{\mathrm{j}}\boldsymbol{\delta}_{\mathrm{ijk}}$.

   The single entry tensor, where the raised indices select a single entry from their corresponding dimension

$$\boldsymbol{\Omega}^{\mathrm{a,b,c,...}}_{\mathrm{ijk...}} = \begin{cases} 1 & \text{if } i = a,\ j = b,\ k = c,\ \ldots \\ 0 & \text{otherwise} \end{cases}$$

   This may be used to access the 3rd column of a matrix $\mathbf{A}$ as $\mathbf{A}_{\mathrm{ij}}\boldsymbol{\Omega}^{3}_{\mathrm{j}}$ or a slice from a tensor $\mathbf{A}$ as $\mathbf{A}_{\mathrm{ijkl}}\boldsymbol{\Omega}^{4,2}_{\mathrm{jl}}$.

   The n-dimensional Levi-Civita or permutation tensor

$$\boldsymbol{\varepsilon}_{\mathrm{a_1 a_2 ... a_n}} = \begin{cases} +1 & \text{if } (a_1, a_2, \ldots, a_{\mathrm{n}}) \text{ is an even permutation of } (1, 2, \ldots, n) \\ -1 & \text{if } (a_1, a_2, \ldots, a_{\mathrm{n}}) \text{ is an odd permutation of } (1, 2, \ldots, n) \\ 0 & \text{otherwise} \end{cases}$$

5. Special operators $\odot, \oplus, \ominus$ carry out entrywise operations on tensors of arbitrary, but equal orders and sizes, such as the Hadamard product:

$$c_{\mathrm{i}} = [\mathbf{a}_{\mathrm{i}} \oplus \mathbf{b}_{\mathrm{i}}]_{\mathrm{i}} = A_{\mathrm{i}} + B_{\mathrm{i}}$$
$$C_{\mathrm{ij}} = [\mathbf{A}_{\mathrm{ij}} \odot \mathbf{B}_{\mathrm{ij}}]_{\mathrm{ij}} = A_{\mathrm{ij}} B_{\mathrm{ij}}$$

Furthermore, these operators may be restricted to act only on certain slices of its inputs, and thus can be applied to two tensors of differing orders, as long as the given slice is of equal size in the two tensors. The result will be a tensor that contracts the inputs along the given slice, but keeps the other dimensions from both input tensors, effectively 'broadcasting' each tensor along the missing dimensions.

$$C_{\mathrm{dij}} = [\mathbf{A}_{\mathrm{di}} \oplus_{\mathrm{d}} \mathbf{B}_{\mathrm{dj}}]_{\mathrm{dij}} = A_{\mathrm{di}} + B_{\mathrm{dj}} = \mathbf{A}_{\mathrm{di}} \mathbf{1}_{\mathrm{j}} + \mathbf{B}_{\mathrm{dj}} \mathbf{1}_{\mathrm{i}}$$
$$C_{\mathrm{deijkl}} = [\mathbf{A}_{\mathrm{dije}} \odot_{\mathrm{de}} \mathbf{B}_{\mathrm{dekl}}]_{\mathrm{deijkl}} = A_{\mathrm{dije}} B_{\mathrm{dekl}} = (\mathbf{A}_{\mathrm{dije}} \mathbf{1}_{\mathrm{kl}}) \odot (\mathbf{B}_{\mathrm{dekl}} \mathbf{1}_{\mathrm{ij}})$$

As there is some ambiguity in the order of dimensions in the result, it is recommended to explicitly state the output dimensions of the resulting computation. Also note that

$$\begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ \dots \\ a_{\mathrm{I}} b_{\mathrm{I}} \end{bmatrix} = \mathbf{a}_{\mathrm{i}} \odot \mathbf{b}_{\mathrm{i}} \neq \mathbf{a}_{\mathrm{i}} \mathbf{b}_{\mathrm{i}} = \sum_{\mathrm{i}} a_{\mathrm{i}} b_{\mathrm{i}}$$

6. We may use capital letters of the English alphabet to denote an ordered set of indices. If $P = \{ijk\}$, then $\mathbf{A}_{\mathrm{ijk}} = A_{\mathrm{P}} \neq \mathbf{A}_{\mathrm{jki}}$.

7. We can denote derivatives as follows. Let $A : \mathbb{R}^{\mathrm{D}} \to \mathbb{R}^{\mathrm{D}_{\mathrm{i}_1} \times \mathrm{D}_{\mathrm{i}_2} \times \cdots \times \mathrm{D}_{\mathrm{i}_n}}$ be an n-th order tensor-valued function over a D-dimensional vector space. The gradient of A in the canonical basis is denoted by placing the coordinate dimension after a comma:

$$\nabla A \triangleq \mathbf{A}_{\mathrm{i}_1 \mathrm{i}_2 \dots \mathrm{i}_n, \mathrm{d}} = \frac{\partial \mathbf{A}_{\mathrm{i}_1 \mathrm{i}_2 \dots \mathrm{i}_n}}{\partial \mathbf{x}_{\mathrm{d}}} \quad d = 1, 2, \dots, D$$

Note that the resulting tensor is now of order $n+1$. Higher order derivatives are denoted by

placing multiple indices after the comma.

$$\mathbf{A}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_\mathrm{n},\mathrm{de}} = \frac{\partial^2 \mathbf{A}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_\mathrm{n}}}{\partial \mathbf{x}_\mathrm{d}\, \partial \mathbf{x}_\mathrm{e}} \quad d,e = 1,2,\ldots,D$$

Remember, that repetition of indices still induces summation over the index, and results in a lower order tensor. We can for example represent the divergence of a scalar-valued function f as

$$\nabla^2 f \triangleq f_{,\mathrm{ii}} = \sum_{\mathrm{i}=1}^{\mathrm{D}} \frac{\partial^2 f}{\partial \mathbf{x}_\mathrm{i}\, \partial \mathbf{x}_\mathrm{i}},$$

which is indeed a scalar.

Let $k : \mathsf{U} \otimes \mathsf{V} \to \mathbb{R}$ be a scalar-valued function defined over the tensor product of real finite dimensional vector spaces. In the derivative, we may denote the identity of vector space by a raised index:

$$\mathbf{k}_{,\mathrm{d}^\mathrm{U}\mathrm{e}^\mathrm{V}} = \frac{\partial^2 k}{\partial \mathbf{u}_\mathrm{d}\, \partial \mathbf{v}_\mathrm{e}}$$

$$\mathbf{u} \in \mathsf{U} \qquad d = 1,2,\ldots,\dim(U)$$

$$\mathbf{v} \in \mathsf{V} \qquad e = 1,2,\ldots,\dim(V)$$

Appendix C

Exponentiated Quadratic kernel function, derivatives and expectations

Our aim here is to define the Exponentiated Quadratic (also known as Radial Basis Function, Squared Exponential or Gaussian) kernel function, write down its derivatives with respect to its arguments, and to derive the various expectations in the scenario, where one of the input arguments is not precisely known, but rather treated as a normally distributed random variable.

The final formulas are then used to estimate the output distribution of a function that has a Gaussian Process prior with an Exponentiated Quadratic covariance function, the function is parametrised by inducing points as well as fixed points and Jacobians around them, and the input to the function is a normally distributed random variable.

C.1 The kernel function

Let $\mathbf{x}$ and $\mathbf{z}$ be D-dimensional vectors and $k : \mathbb{R}^{\mathrm{D}} \otimes \mathbb{R}^{\mathrm{D}} \to \mathbb{R}$ be a positive definite kernel function. The exponentiated quadratic kernel function is parametrised by a positve output scale ν and positive definite covariance matrix $\mathbf{\Lambda}$:

$$k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) = \nu e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^\top \mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})} \tag{C.1}$$

The covariance matrix of the kernel function is often diagonal to reduce the number of parameters. As our latent space models contain rotational non-identifiability due to the flexible output mapping, we can indeed assume this diagonality without loss of generality, thus we set $\mathbf{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ is the D dimensional vector of so-called 'kernel lengthscales'.

The Exponentiated Quadratic kernel function is closely related to the probability density function of the normal distribution in either of its entries, and this identity will be used extensively throughout the derivations:

$$\mathcal{N}_{\mathbf{x}}(\mathbf{z},\,\mathbf{\Lambda}) = p_{\underline{\text{Normal}}}(\mathbf{x} \mid \mathbf{z},\,\mathbf{\Lambda}) \tag{C.2}$$

$$= (2\pi)^{-\mathrm{D}/2}(|\mathbf{\Lambda}|)^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^{\top}\mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})} \tag{C.3}$$

$$= \frac{1}{\nu}(2\pi)^{-\mathrm{D}/2}(|\mathbf{\Lambda}|)^{-1/2} k(\mathbf{x},\mathbf{z} \mid \nu,\,\mathbf{\Lambda}) \tag{C.4}$$

Therefore also

$$k(\mathbf{x},\mathbf{z} \mid \nu,\,\mathbf{\Lambda}) = \nu(2\pi)^{\mathrm{D}/2}(|\mathbf{\Lambda}|)^{1/2}\mathcal{N}_{\mathbf{x}}(\mathbf{z},\,\mathbf{\Lambda}) \tag{C.5}$$

Furthermore, we are often required to compute the so-called kernel matrix between two sets of inputs. Let $\{\mathbf{x}^{\mathrm{n}}\}_{\mathrm{n}=1}^{\mathrm{N}} = \mathbf{X} \in \mathbb{R}^{\mathrm{DxN}}$ and $\{\mathbf{z}^{\mathrm{m}}\}_{\mathrm{m}=1}^{\mathrm{M}} = \mathbf{Z} \in \mathbb{R}^{\mathrm{DxM}}$ be collection of N and M inputs, then we can define the kernel matrix entry-wise, via common notation or via indicial notation as

$$K_{\mathrm{nm}} = k\left(\mathbf{x}^{\mathrm{n}}, \mathbf{z}^{\mathrm{m}} \;\middle|\; \nu,\,\mathbf{\Lambda}\right) \tag{C.6}$$

$$\mathbf{K} = k(\mathbf{X},\mathbf{Z} \mid \nu,\,\mathbf{\Lambda}) \tag{C.7}$$

$$\mathbf{K}_{\mathrm{nm}} = k(\mathbf{X}_{\mathrm{dn}},\mathbf{Z}_{\mathrm{dm}} \;\middle|\; \nu,\,\mathbf{\Lambda}_{\mathrm{d\dot{d}}})\,. \tag{C.8}$$

Although possible, computing functions of the kernel matrix is not always simple or intuitive using the entry-wise representation, and thus I first define operations assuming single inputs and using the common linear algebra notation, but then also show how the same operations may be carried out on collections of inputs using tensor calculations via the indicial notation. The single input derivatives and expectations were made available by McHutchon (2013), whereas for collection of inputs some of them were shown by Girard (2004) in an entry-wise fashion.

To my knowledge this is the first publicly accessible document that shows these derivatives and expectations in a tensor format that lends itself easily for computer implementations, unlike previous derivations. However, as the main ideas of these lengthy derivations follow those previously published, I will show only the end result in some cases, to avoid introducing unnecessarily complex notation. The intermediate algebraic manipulations rely mainly upon rearranging terms to produce a normalised normal probability density function under the integral sign (which then integrates to 1), and collecting terms such that we end up with sums and products of easily computable Exponentiated Quadratic kernel functions with modified output scales, lengthscales or inputs.

C.2 The derivative Gaussian Process

As we are using derivative observations as parameters for our fixed point models, we need to define the derivatives of the kernel function with respect to either or both of their inputs. For single inputs the derivatives are

$$k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) = \nu e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^\top \mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})} \tag{C.9}$$

$$\frac{\partial}{\partial \mathbf{x}} k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) = -\mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})\, k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) \tag{C.10}$$

$$\frac{\partial}{\partial \mathbf{z}} k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) = +\mathbf{\Lambda}^{-1}(\mathbf{x}-\mathbf{z})\, k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) \tag{C.11}$$

$$\frac{\partial^2}{\partial \mathbf{x} \partial \mathbf{z}} k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) = \mathbf{\Lambda}^{-1}(\mathbf{I} - (\mathbf{x}-\mathbf{z})(\mathbf{x}-\mathbf{z})^\top \mathbf{\Lambda}^{-1})\, k(\mathbf{x}, \mathbf{z} \mid \nu, \mathbf{\Lambda}) \tag{C.12}$$

With these one can compute all elements of the covariance matrix between observations and derivative observations of a Gaussian Process with an Exponentiated Quadratic kernel function. Note that although the derivatives in equations C.10 and C.11 seem anti-symmetric, when we form the cross-covariance blocks of the joint covariance matrix, we compute $\frac{\partial}{\partial \mathbf{x}} k(\mathbf{x}, \mathbf{z})$ and $\frac{\partial}{\partial \mathbf{x}} k(\mathbf{z}, \mathbf{x})$ as the blocks. As the $(\mathbf{x}-\mathbf{z})$ term also switches sign when we flip the order of kernel inputs, these terms are indeed transposes of one-another, and the covariance matrix is symmetric as expected.

We can see that the first derivatives of single inputs are D-dimensional vectors, whereas the second derivative is a $D \times D$ matrix, as derivatives of a tensor-valued function with respect to vectors increase the tensor order of the output by one. Although in the case of single inputs, $k(\cdot, \cdot)$ returns a zero-th order tensor, when the inputs are collections of points, the output is a second order tensor, the 'kernel matrix'. Therefore I now also define the derivatives for collections of points as the input, then also describe how one may flatten the resulting tensors to end up with a valid block-structured covariance matrix describing the joint distribution of observations and derivative observations.

$$\mathbf{K}_{\mathrm{nm}} = k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu, \boldsymbol{\lambda}_{\mathrm{d}}) \tag{C.13}$$

$$\begin{aligned} \mathbf{K}_{\mathrm{d^x nm}} &= \frac{\partial}{\partial \mathbf{x}_{\mathrm{d}}} k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu, \boldsymbol{\lambda}_{\mathrm{d}}) \\ &= -[\boldsymbol{\lambda}_{\mathrm{d}}^{-1} \mathbf{1}_{\mathrm{nm}}] \odot [\mathbf{X} \ominus_{\mathrm{d}} \mathbf{Z}]_{\mathrm{dnm}} \odot [\mathbf{1}_{\mathrm{d}}\, k(\mathbf{X}, \mathbf{Z})]_{\mathrm{dnm}} \end{aligned} \tag{C.14}$$

$$\begin{aligned} \mathbf{K}_{\mathrm{nd^z m}} &= \frac{\partial}{\partial \mathbf{z}_{\mathrm{d}}} k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu, \boldsymbol{\lambda}_{\mathrm{d}}) \\ &= +[\boldsymbol{\lambda}_{\mathrm{d}}^{-1} \mathbf{1}_{\mathrm{nm}}] \odot [\mathbf{X} \ominus_{\mathrm{d}} \mathbf{Z}]_{\mathrm{dnm}} \odot [\mathbf{1}_{\mathrm{d}}\, k(\mathbf{X}, \mathbf{Z})]_{\mathrm{dnm}} \end{aligned} \tag{C.15}$$

$$\begin{aligned}\mathbf{K}_{\mathrm{d}^{\mathrm{x}}\mathrm{n}\mathring{\mathrm{d}}^{\mathrm{z}}\mathrm{m}} &= \frac{\partial^2}{\partial \mathbf{x}_{\mathrm{d}}\, \partial \mathbf{z}_{\mathring{\mathrm{d}}}} k(\mathbf{X}_{\mathrm{dn}}, \mathbf{Z}_{\mathrm{dm}} \mid \nu,\, \boldsymbol{\lambda}_{\mathrm{d}}) \\ &= \left[\boldsymbol{\Lambda}_{\mathrm{d}\mathring{\mathrm{d}}}^{-1}\mathbf{1}_{\mathrm{nm}} - \boldsymbol{\lambda}_{\mathrm{d}}^{-1} \odot_{\mathrm{d}} (\mathbf{X}_{\mathrm{dn}} \ominus_{\mathrm{d}} \mathbf{Z}_{\mathrm{dm}}) \odot_{\mathrm{nm}} (\mathbf{X}_{\mathring{\mathrm{d}}\mathrm{n}} \ominus_{\mathring{\mathrm{d}}} \mathbf{Z}_{\mathring{\mathrm{d}}\mathrm{m}}) \odot_{\mathring{\mathrm{d}}} \boldsymbol{\lambda}_{\mathring{\mathrm{d}}}^{-1})\right]_{\mathrm{dn}\mathring{\mathrm{d}}\mathrm{m}} \\ &\qquad \odot \left[\mathbf{1}_{\mathrm{d}\mathring{\mathrm{d}}}\, k(\mathbf{X}, \mathbf{Z})\right]_{\mathrm{dn}\mathring{\mathrm{d}}\mathrm{m}} \end{aligned} \tag{C.16}$$

These equations are now directly implementable in modern computer languages with little effort, using the appropriate broadcasting operations to form tensors of correct order and size.

The last question is how to flatten these higher order tensors back to second order to form the block-structured covariance matrix of a Gaussian Process. Let $\mathbf{S}_{\mathrm{dn}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{N}}$ be the N locations of derivative observations $\mathbf{J}_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{n}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}\times\mathrm{N}}$, and $\mathbf{Z}_{\mathrm{dm}}$ the locations of regular observations $\mathbf{U}_{\mathrm{dm}}$. In such a case we obtain the tensors $\mathbf{K}^{\mathrm{zz}}_{\mathrm{mm}} = k^{\mathrm{zz}}$, $\mathbf{K}^{\mathrm{zs}'}_{\mathrm{m(dn)}} = \nabla_2 k^{\mathrm{zs}}$, $\mathbf{K}^{\mathrm{s}'\mathrm{z}}_{\mathrm{(dn)m}} = \nabla_1 k^{\mathrm{sz}}$, and $\mathbf{K}^{\mathrm{s}'\mathrm{s}'}_{\mathrm{(dn)}(\mathring{\mathrm{d}}\mathring{\mathrm{n}})} = \nabla_1\nabla_2 k^{\mathrm{ss}}$. The higher order tensors should be flattened as follows to obtain the joint distribution of regular and derivative observations $\mathbf{U}_{1\cdot}$ and $\mathbf{J}_{1\cdot\cdot}$ (setting the output dimension in question arbitrarily to $d = 1$):

$$\begin{bmatrix} U_{11} \\ \vdots \\ U_{1\mathrm{M}} \\ \hline J_{111} \\ J_{121} \\ \vdots \\ J_{1\mathrm{D}1} \\ J_{112} \\ \vdots \\ J_{1\mathrm{DN}} \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{ccc|cccccccc} & \mathbf{K}^{\mathrm{zz}}_{\mathrm{mm}} & & & & & \mathbf{K}^{\mathrm{zs}'}_{\mathrm{m(dn)}} & & & \\ \hline K^{\mathrm{s}'\mathrm{z}}_{(11)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(11)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(11)(11)} & K^{\mathrm{s}'\mathrm{s}'}_{(11)(21)} & \cdots & K^{\mathrm{s}'\mathrm{s}'}_{(11)(\mathrm{D}1)} & K^{\mathrm{s}'\mathrm{s}'}_{(11)(12)} & \cdots & K^{\mathrm{s}'\mathrm{s}'}_{(11)(\mathrm{DN})} \\ K^{\mathrm{s}'\mathrm{z}}_{(21)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(21)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(21)(11)} & \ddots & & & & & \\ & \vdots & & & & & & & & \\ K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{D}1)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{D}1)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{D}1)(11)} & & & \mathbf{K}^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{dn})(\mathrm{dn})} & & & \\ K^{\mathrm{s}'\mathrm{z}}_{(12)1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(12)\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(12)(11)} & & & & & & \\ & \vdots & & & & & & & \ddots & \\ K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{DN})1} & \cdots & K^{\mathrm{s}'\mathrm{z}}_{(\mathrm{DN})\mathrm{M}} & K^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{DN})(11)} & & & & & & K^{\mathrm{s}'\mathrm{s}'}_{(\mathrm{DN})(\mathrm{DN})} \end{array}\right]\right) \tag{C.17}$$

C.3 Expectations with respect to noisy input

Let $\mathbf{Z} \in \mathbb{R}^{\mathrm{D}\times\mathrm{M}}$ be a collection of known inputs and $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$ a noisy input, such that $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a random variable distributed according to a multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. As described in chapter 4, this $\mathbf{x}$ represents a belief, and we wish to propagate this belief through a transition function represented by a Gaussian process represented by inducing points, fixed points and derivative observations around the fixed points. In order to compute our updated belief representation, we need to calculate the expected mean and the co-variance of the output $f(\mathbf{x})$ with respect to the noisy input $\mathbf{x}$. These expressions (see equations 4.14 to 4.16) contain various expectations

of kernel matrices (defined equation 4.10), and in turn require us to calculate expectations of various functions involving the kernel function $k(\cdot,\cdot)$. Fortunately all these expectation are available in closed form for the Exponentiated Quadratic kernel function, and thus here I state the solutions that enable us to propagate latent beliefs forward analytically. The superscripts on the resulting tensor identify what did we take the expectation of, whereas the subscripts denote the correct order and size of the tensor.

$$\begin{aligned}\mathbf{E}_{\mathrm{m}}^{\mathrm{k}} &= \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right] \\ &= k\left(\boldsymbol{\mu},\mathbf{Z}\;\middle|\;\frac{|\boldsymbol{\Lambda}|^{1/2}}{|\boldsymbol{\Lambda}+\boldsymbol{\Sigma}|^{1/2}}\nu,\,\boldsymbol{\Lambda}+\boldsymbol{\Sigma}\right)\end{aligned} \tag{C.18}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{dm}}^{\mathrm{xk}} &= \mathbb{E}_{\mathrm{x}}\left[\mathbf{x}\,k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right] \\ &= \mathbf{E}_{\mathrm{m}}^{\mathrm{k}} \odot_{\mathrm{m}} \left[(\boldsymbol{\Lambda}^{-1}+\boldsymbol{\Sigma}^{-1})^{-1}_{\mathring{\mathrm{d}}\mathrm{d}}\left([\boldsymbol{\Lambda}^{-1}\mathbf{Z}]_{\mathrm{dm}} \odot_{\mathrm{d}} [\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}]_{\mathrm{d}}\right)\right]_{\mathrm{dm}}\end{aligned} \tag{C.19}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{dm}}^{\mathrm{dk}} &= \mathbb{E}_{\mathrm{x}}\left[\nabla_2\, k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right] \\ &= \boldsymbol{\lambda}_{\mathrm{d}}^{-1} \odot_{\mathrm{d}} \left(\mathbf{E}_{\mathrm{dm}}^{\mathrm{xk}} - \mathbf{Z}_{\mathrm{dm}} \odot_{\mathrm{m}} \mathbf{E}_{\mathrm{m}}^{\mathrm{k}}\right)\end{aligned} \tag{C.20}$$

$$E^{\mathrm{v}} = \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{x},\mathbf{x}\mid \nu,\,\boldsymbol{\lambda})\right] = \nu \tag{C.21}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{kk}} &= \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{Z},\mathbf{x}\mid \nu,\,\boldsymbol{\lambda})\;k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right] \\ &= k\left(\mathbf{Z},\mathbf{Z}\;\middle|\;\frac{1}{2^{\mathrm{D}/2}}\nu,\,2\boldsymbol{\Lambda}\right) \odot k\left([\mathbf{Z}\oplus_{\mathrm{d}}\mathbf{Z}]_{\mathrm{dm}\mathring{\mathrm{m}}},\boldsymbol{\mu}_{\mathrm{d}}\;\middle|\;\frac{|\boldsymbol{\Lambda}|^{1/2}}{\left|\frac{\boldsymbol{\Lambda}}{2}+\boldsymbol{\Sigma}\right|^{1/2}}\nu,\,\frac{\boldsymbol{\Lambda}}{2}+\boldsymbol{\Sigma}\right)\end{aligned} \tag{C.22}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{dm}\mathring{\mathrm{m}}}^{\mathrm{xkk}} &= \mathbb{E}_{\mathrm{x}}\left[\mathbf{x}\,k(\mathbf{Z},\mathbf{x}\mid \nu,\,\boldsymbol{\lambda})\;k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right] \\ &= \mathbf{E}_{\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{kk}} \odot_{\mathrm{m}\mathring{\mathrm{m}}} \left[\left(\left(\frac{\boldsymbol{\Lambda}}{2}\right)^{-1}+\boldsymbol{\Sigma}^{-1}\right)^{-1}\left(\left(\frac{\boldsymbol{\Lambda}}{2}\right)^{-1} \odot_{\mathrm{d}} \left([\mathbf{Z}\oplus_{\mathrm{d}}\mathbf{Z}]_{\mathrm{dm}\mathring{\mathrm{m}}} \oplus_{\mathrm{d}} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right)\right]_{\mathrm{dm}\mathring{\mathrm{m}}}\end{aligned} \tag{C.23}$$

$$\begin{aligned}\mathbf{E}_{\mathrm{m}(\mathrm{d}\mathring{\mathrm{m}})}^{\mathrm{kdk}} &= \mathbb{E}_{\mathrm{x}}\left[k(\mathbf{Z},\mathbf{x}\mid \nu,\,\boldsymbol{\lambda})\;\nabla_2\, k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right] \\ &= \boldsymbol{\lambda}_{\mathrm{d}}^{-1} \odot_{\mathrm{d}} \left(\mathbf{E}_{\mathrm{dm}}^{\mathrm{xkk}} - \mathbf{Z}_{\mathrm{dm}} \odot_{\mathrm{m}} \mathbf{E}_{\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{kk}}\right)\end{aligned} \tag{C.24}$$

$$\mathbf{E}_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}}^{\mathrm{xxkk}} = \mathbb{E}_{\mathrm{x}}\left[\mathbf{x}\,\mathbf{x}\,k(\mathbf{Z},\mathbf{x}\mid \nu,\,\boldsymbol{\lambda})\;k(\mathbf{x},\mathbf{Z}\mid \nu,\,\boldsymbol{\lambda})\right]$$

$$= \left[\mathbf{E}^{\mathrm{kk}}_{\mathrm{m}\mathring{\mathrm{m}}} \left(\left(\frac{\boldsymbol{\Lambda}}{2} \right)^{-1} + \boldsymbol{\Sigma}^{-1} \right)^{-1}_{\mathrm{d}\mathring{\mathrm{d}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} + \left[\mathbf{E}^{\mathrm{xkk}}_{\mathrm{dm}\mathring{\mathrm{m}}} \odot_{\mathrm{m}\mathring{\mathrm{m}}} \mathbf{E}^{\mathrm{xkk}}_{\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} \tag{C.25}$$

$$\begin{aligned} \mathbf{E}^{\mathrm{dkdk}}_{(\mathrm{dm})(\mathring{\mathrm{d}}\mathring{\mathrm{m}})} &= \mathbb{E}_{\mathrm{x}} \left[\nabla_1 \, k(\mathbf{Z}, \mathbf{x} \mid \nu, \, \boldsymbol{\lambda}) \, \nabla_2 \, k(\mathbf{x}, \mathbf{Z} \mid \nu, \, \boldsymbol{\lambda}) \right] \\ &= \left[\boldsymbol{\lambda}^{-1}_{\mathrm{d}} \boldsymbol{\lambda}^{-1}_{\mathring{\mathrm{d}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}} \\ &\quad \odot_{\mathrm{d}\mathring{\mathrm{d}}} \left[\mathbf{E}^{\mathrm{xxkk}}_{\mathrm{dd}\mathring{\mathrm{m}}\mathring{\mathrm{m}}} - \mathbf{Z}_{\mathrm{dm}} \odot_{\mathrm{m}} \mathbf{E}^{\mathrm{xkk}}_{\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} - \mathbf{Z}_{\mathring{\mathrm{d}}\mathring{\mathrm{m}}} \odot_{\mathring{\mathrm{m}}} \mathbf{E}^{\mathrm{xkk}}_{\mathrm{dm}\mathring{\mathrm{m}}} + \mathbf{Z}_{\mathrm{dm}} \mathbf{Z}_{\mathring{\mathrm{d}}\mathring{\mathrm{m}}} \odot_{\mathrm{m}\mathring{\mathrm{m}}} \mathbf{E}^{\mathrm{kk}}_{\mathrm{m}\mathring{\mathrm{m}}} \right]_{\mathrm{d}\mathring{\mathrm{d}}\mathrm{m}\mathring{\mathrm{m}}} \end{aligned} \tag{C.26}$$

Bibliography

Adam, Vincent, Nicolas Durrande, and S. T. John (2018). "Scalable GAM using sparse variational Gaussian processes". In: *CoRR* abs/1812.11106 (cit. on p. 87).

Archambeau, Cédric, Dan Cornford, Manfred Opper, and John Shawe-Taylor (2007). "Gaussian process approximations of stochastic differential equations". In: *Journal of machine learning research* 1, pp. 1–16 (cit. on pp. 134–137).

Archambeau, Cédric, Manfred Opper, Yuan Shen, Dan Cornford, and John S Shawe-taylor (2008). "Variational inference for diffusion processes". In: *Advances in Neural Information Processing Systems*, pp. 17–24 (cit. on pp. 134–137).

Arnold, Ludwig and Hans Crauel (Jan. 1991). "Random dynamical systems". In: pp. 1–22 (cit. on p. 118).

Attias, Hagai (2000). "A variational baysian framework for graphical models". In: *Advances in neural information processing systems*, pp. 209–215 (cit. on p. 136).

Bohner, Gergo and Maneesh Sahani (2016). "Convolutional higher order matching pursuit". In: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6 (cit. on pp. 70, 91).

Bohner, Gergo and Maneesh Sahani (2018). "Empirical fixed point bifurcation analysis". In: *arXiv preprint arXiv:1807.01486* (cit. on p. 119).

Bonnier, Nicolas and Eero P. Simoncelli (2005). "Locally adaptive multiscale contrast optimization". In: *Proceedings - International Conference on Image Processing, ICIP* 1, pp. 949–952 (cit. on p. 67).

Boyd, Stephen, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein (Jan. 2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Found. Trends Mach. Learn.* 3.1, pp. 1–122 (cit. on pp. 72, 91, 93).

Chen, Taoyi, Zhong Xue, Changhong Wang, Zhenshen Qu, Kelvin K. Wong, and Stephen T C Wong (2012). "Motion correction for cellular-resolution multi-photon fluorescence microscopy imaging of awake head-restrained mice using speed embedded HMM". In: *Computerized Medical Imaging and Graphics* 36.3, pp. 171–182 (cit. on p. 45).

Chen, Tsai-Wen, Trevor J. Wardill, Yi Sun, Stefan R. Pulver, Sabine L. Renninger, Amy Baohan, Eric R. Schreiter, Rex A. Kerr, Michael B. Orger, Vivek Jayaraman, Loren L. Looger, Karel

Svoboda, and Douglas S. Kim (2013a). “Ultrasensitive fluorescent proteins for imaging neuronal activity”. In: *Nature* 499.7458, pp. 295–300 (cit. on pp. 26, 147).

Chen, Tsai-Wen, Trevor J. Wardill, Yi Sun, Stefan R. Pulver, Sabine L. Renninger, Amy Baohan, Eric R. Schreiter, Rex A. Kerr, Michael B. Orger, Vivek Jayaraman, Loren L. Looger, Karel Svoboda, and Douglas S. Kim (July 2013b). “Ultrasensitive fluorescent proteins for imaging neuronal activity”. In: *Nature* 499, 295 EP – (cit. on p. 111).

Cheng, Ming-ming, Niloy J Mitra, Xiaolei Huang, Philip H S Torr, and Shi-min Hu (2015). “Global Contrast Based Salient Region Detection”. In: 37.3, pp. 569–582 (cit. on p. 67).

Churchland, Mark M., John P. Cunningham, Matthew T. Kaufman, Justin D. Foster, Paul Nuyujukian, Stephen I. Ryu, and Krishna V. Shenoy (June 2012). “Neural population dynamics during reaching”. In: *Nature* 487, 51 EP – (cit. on p. 117).

Comon, Pierre, Gene Golub, Lek-Heng Lim, and Bernard Mourrain (2008). “Symmetric Tensors and Symmetric Tensor Rank”. In: *SIAM Journal on Matrix Analysis and Applications* 30.3, pp. 1254–1279 (cit. on p. 78).

Cseke, Botond, David Schnoerr, Manfred Opper, and Guido Sanguinetti (2016). “Expectation propagation for continuous time stochastic processes”. In: *Journal of Physics A: Mathematical and Theoretical* 49.49, p. 494002 (cit. on p. 134).

Damianou, Andreas C., Michalis K. Titsias, and Neil D. Lawrence (2011). “Variational Gaussian Process Dynamical Systems”. In: pp. 1–9 (cit. on p. 118).

De Haas, J. T. M. and Pieter Dorenbos (2010). “Methods for accurate measurement of the response of photomultiplier tubes and intensity of light pulses”. In: *IEEE Nuclear Science Symposium Conference Record* 58.3, pp. 205–210 (cit. on p. 42).

Deisenroth, Marc and Shakir Mohamed (2012). “Expectation propagation in Gaussian process dynamical systems”. In: *Advances in Neural Information Processing Systems*, pp. 2609–2617 (cit. on p. 126).

Deisenroth, Marc Peter, Marco F. Huber, and Uwe D. Hanebeck (2009). “Analytic moment-based Gaussian process filtering”. In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML ’09*, pp. 225–232 (cit. on pp. 126, 128, 129).

Deisenroth, Marc Peter, Ryan Darby Turner, Marco F. Huber, Uwe D. Hanebeck, and Carl Edward Rasmussen (2012). “Robust filtering and smoothing with gaussian processes”. In: *IEEE Transactions on Automatic Control* 57.7, pp. 1865–1871 (cit. on p. 126).

Di Nardo, Elvira, Giuseppe Guarino, and Domenico Senato (2008). “A unifying framework for k-statistics, polykays and their multivariate generalizations”. In: *Bernoulli* 14.2, pp. 440–468 (cit. on p. 77).

Dijk, Judith, Richard J.M. den Hollander, John G.M. Schavemaker, and Klamer Schutte (2007). “Local adaptive contrast enhancement for color images”. In: *Visual Information Processing XVI* SPIE-6575, 65750A (cit. on p. 67).

Diks, Cees G. H. and Florian Wagener (2006). "A weak bifurcation theory for discrete time stochastic dynamical systems". In: *Tinbergen Institute Discussion Paper No. 06-043/1* June, pp. 1–30 (cit. on pp. 118, 141).

Dossi, R, A Ianni, G Ranucci, and O Ju Smirnov (2000). "Methods for precise photoelectron counting with photomultipliers". In: *Nuclear Instruments and Methods in Physics Research A* 451, pp. 623–637 (cit. on p. 42).

Duncker, Lea and Maneesh Sahani (2018). "Temporal alignment and latent Gaussian process factor inference in population spike trains". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 10466–10476 (cit. on p. 118).

Duncker, Lea, Daniel J. O'Shea, Werapong Goo, Krishna V. Shenoy, and Sahani Maneesh (2017). *Low-rank non-stationary population dynamics can account for robustness to optogenetic stimulation*. COSYNE talk (cit. on pp. 118, 149).

Duncker, Lea, Gergo Bohner, Julien Boussard, and Maneesh Sahani (2019). "Learning interpretable continuous-time models of latent stochastic dynamical systems". In: *Proceedings of the 36th International Conference on Machine Learning* (cit. on pp. 119, 131, 137).

Eleftheriadis, Stefanos, Tom Nicholson, Marc Deisenroth, and James Hensman (2017). "Identification of Gaussian Process State Space Models". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 5309–5319 (cit. on p. 118).

Frigola, Roger, Yutian Chen, and Carl E. Rasmussen (2014). "Variational Gaussian Process State-Space Models". In: pp. 1–9 (cit. on p. 121).

Ganapathisubramanian, N (1991). "Tristability in the iodate–As (III) chemical system arising from a model of stirring and mixing effects". In: *The Journal of chemical physics* 95.4, pp. 3005–3008 (cit. on p. 142).

Girard, Agathe (2004). "Approximate methods for propagation of uncertainty with Gaussian process models". In: *Ph.D. Thesis* October (cit. on p. 168).

Golub, Matt and David Sussillo (2018). "FixedPointFinder: A Tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks". In: *Journal of Open Source Software* 3.31, p. 1003 (cit. on pp. 118, 119).

Greenberg, David S. and Jason N D Kerr (2009). "Automated correction of fast motion artifacts for two-photon imaging of awake animals". In: *Journal of Neuroscience Methods* 176.1, pp. 1–15 (cit. on p. 45).

Huang, Shih Chia, Fan Chieh Cheng, and Yi Sheng Chiu (2013). "Efficient contrast enhancement using adaptive gamma correction with weighting distribution". In: *IEEE Transactions on Image Processing* 22.3, pp. 1032–1041 (cit. on p. 67).

Hyvärinen, Aapo and Erkki Oja (2000). “Independent component analysis: algorithms and applications”. In: *Neural Networks* 13.4, pp. 411 –430 (cit. on p. 71).

Isserlis, Leon (1918). “On a Formula for the Product-Moment Coefficient of Any Order of a Normal Frequency Distribution in Any Number of Variables”. In: *Biometrika* 12.1-2, pp. 134–139 (cit. on p. 81).

Jianbo Shi and J. Malik (2000). “Normalized cuts and image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8, pp. 888–905 (cit. on p. 71).

Jin, Yu-feng, Shen-sian Syu, and Ming-jong Jou (2017). “An Adaptive Contrast Enhancement of Image Using Multi-Scale Histogram Representation”. In: c, pp. 318–321 (cit. on p. 67).

Kaifosh, Patrick, Jeffrey D. Zaremba, Nathan B. Danielson, and Attila Losonczy (2014). “SIMA: Python software for analysis of dynamic fluorescence imaging data”. In: *Frontiers in Neuroinformatics* 8, p. 80 (cit. on p. 71).

Kendall, Maurice G., Alan Stuart, and J. Keith Ord, eds. (1987). *Kendall’s Advanced Theory of Statistics*. New York, NY, USA: Oxford University Press, Inc. (cit. on p. 77).

Khan, Suleiman A., Eemeli Leppäaho, and Samuel Kaski (2016). “Bayesian multi-tensor factorization”. In: *Machine Learning* 105.2, pp. 233–253 (cit. on p. 87).

Kim, Tae Keun, Joon Ki Paik, and Bong Soon Kang (1998). “Contrast enhancement system using spatially adaptive histogram equalization with temporal filtering”. In: *IEEE Transactions on Consumer Electronics* 44.1, pp. 82–87 (cit. on p. 67).

Kireev, Sergey V. and Sergey L. Shnyrev (2015). “Study of molecular iodine, iodate ions, iodide ions, and triiodide ions solutions absorption in the UV and visible light spectral bands”. In: *Laser Physics* 25.7, p. 075602 (cit. on p. 142).

Ko, Jonathan, Daniel J Klein, Dieter Fox, and Dirk Hähnel (2007). “GP-UKF: Unscented kalman filters with Gaussian process prediction and observationmodels”. In: *Iros*, pp. 1901–1907 (cit. on p. 126).

Lathauwer, Lieven De, Bart De Moor, and Joos Vandewalle (Mar. 2000). “On the Best Rank-1 and Rank-(R1,R2,. . .,RN) Approximation of Higher-Order Tensors”. In: *SIAM J. Matrix Anal. Appl.* 21.4, pp. 1324–1342 (cit. on p. 87).

Lyu, Siwei and Eero P. Simoncelli (2008). “Nonlinear image representation using divisive normalization”. In: *IEEE Conference on Computer Vision and Pattern Recognition (2008)*, pp. 1–8 (cit. on p. 67).

Lütcke, Henry, Felipe Gerhard, Friedemann Zenke, Wulfram Gerstner, and Fritjof Helmchen (2013). “Inference of neuronal network spike dynamics and topology from calcium imaging data”. In: *Frontiers in Neural Circuits* 7, p. 201 (cit. on p. 147).

Machens, Christian K., Ranulfo Romo, and Carlos D. Brody (2005). “Flexible control of mutual inhibition: A neural model of two-interval discrimination”. In: *Science* 307.5712, pp. 1121–1124 (cit. on pp. 146, 147).

Macke, Jakob H, Lars Buesing, John P Cunningham, M Yu Byron, Krishna V Shenoy, and Maneesh Sahani (2011). "Empirical models of spiking in neural populations". In: *Advances in neural information processing systems*, pp. 1350–1358 (cit. on p. 146).

Mallat, Stephane G. and Zhifeng Zhang (Dec. 1993). "Matching Pursuits with Time-frequency Dictionaries". In: *Trans. Sig. Proc.* 41.12, pp. 3397–3415 (cit. on p. 72).

Mante, Valerio, David Sussillo, Krishna V Shenoy, and William T Newsome (Nov. 2013). "Context-dependent computation by recurrent dynamics in prefrontal cortex". In: *Nature* 503.7474, pp. 78–84 (cit. on p. 117).

Maruyama, Ryuichi, Kazuma Maeda, Hajime Moroda, Ichiro Kato, Masashi Inoue, Hiroyoshi Miyakawa, and Toru Aonishi (2014). "Detecting cells using non-negative matrix factorization on calcium imaging data". In: *Neural Networks* 55, pp. 11 –19 (cit. on p. 71).

McHutchon, Andrew (2013). "Differentiating Gaussian Processes" (cit. on p. 168).

McHutchon, Andrew (2014). "Nonlinear Modelling and Control using Gaussian Processes". In: August (cit. on pp. 121, 123, 126, 149).

Mukamel, Eran A, Axel Nimmerjahn, and Mark J Schnitzer (Sept. 2009). "Automated analysis of cellular signals from large-scale calcium imaging data". In: *Neuron* 63.6, pp. 747–760 (cit. on p. 71).

Nonnenmacher, Marcel, Srinivas C Turaga, and Jakob H Macke (2017). "Extracting low-dimensional dynamics from multiple large-scale neural population recordings by learning to predict correlations". In: *Advances in Neural Information Processing Systems (accepted)* Nips (cit. on p. 151).

Norris, J. P., J. T. Bonnell, D. Kazanas, J. D. Scargle, J. Hakkila, and T. W. Giblin (2005). "Long-Lag, Wide-Pulse Gamma-Ray Bursts". In: *The Astrophysical Journal* 627.1, pp. 324–345 (cit. on p. 147).

O'Shea, Daniel J., Eric Trautmann, Chandramouli Chandrasekaran, Sergey Stavisky, Jonathan C. Kao, Maneesh Sahani, Stephen Ryu, Karl Deisseroth, and Krishna V. Shenoy (2017). "The need for calcium imaging in nonhuman primates: New motor neuroscience and brain-machine interfaces". In: *Experimental Neurology* 287. Bio-electronics and prosthetics for neurological diseases, pp. 437 –451 (cit. on p. 147).

Pachitariu, Marius, Adam M. Packer, Noah Pettit, Henry Dalgleish, Michael Hausser, and Maneesh Sahani (2013). "Extracting regions of interest from biological images with convolutional sparse block coding". In: *Advances in Neural Information Processing Systems*. Vol. 1, pp. 1745–1753 (cit. on pp. 70–72, 74, 77, 80).

Pachitariu, Marius, Carsen Stringer, Sylvia Schröder, Mario Dipoppa, L. Federico Rossi, Matteo Carandini, and Kenneth D. Harris (2016). "Suite2p: beyond 10,000 neurons with standard two-photon microscopy". In: *bioRxiv* (cit. on pp. 67, 71, 72, 77, 106).

Pandarinath, Chethan, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. (2018b). "Inferring single-trial neural population dynamics using sequential auto-encoders". In: *Nature methods*, p. 1 (cit. on p. 146).

Pandarinath, Chethan, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo (2018a). "Inferring single-trial neural population dynamics using sequential auto-encoders". In: *Nature Methods* 15.10, pp. 805–815 (cit. on p. 118).

Park, Mijung, Gergo Bohner, and Jakob H Macke (2015). "Unlocking neural population non-stationarities using hierarchical dynamics models". In: *Advances in Neural Information Processing Systems*, pp. 145–153 (cit. on pp. 118, 146, 150).

Peron, Simon, Tsai-Wen Chen, and Karel Svoboda (2015). "Comprehensive imaging of cortical networks". In: *Current Opinion in Neurobiology* 32. Large-Scale Recording Technology (32), pp. 115 –123 (cit. on p. 111).

Petreska, Biljana, Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V Shenoy, and Maneesh Sahani (2011). "Dynamical segmentation of single trials from population neural data". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., pp. 756–764 (cit. on p. 150).

Pnevmatikakis, Eftychios A. and Andrea Giovannucci (2017). "NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data". In: *Journal of Neuroscience Methods* 291, pp. 83–94 (cit. on p. 45).

Pnevmatikakis, Eftychios A, Josh Merel, Ari Pakman, and Liam Paninski (2013). "Bayesian spike inference from calcium imaging data". In: *2013 Asilomar Conference on Signals, Systems and Computers*, pp. 349–353 (cit. on p. 123).

Pnevmatikakis, Eftychios A, Daniel Soudry, Yuanjun Gao, Timothy A Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, Misha Ahrens, Randy Bruno, Thomas M Jessell, Darcy S Peterka, Rafael Yuste, and Liam Paninski (Jan. 2016). "Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data". In: *Neuron* 89.2, pp. 285–299 (cit. on pp. 71, 72).

Poyiadjis, George, Arnaud Doucet, and Sumeetpal S. Singh (Feb. 2011). "Particle approximations of the score and observed information matrix in state space models with application to parameter estimation". In: *Biometrika* 98.1, pp. 65–80 (cit. on p. 126).

Ramakrishnan, Naveen, Emre Ertin, and Randolph L Moses (2011). "Assumed density filtering for learning Gaussian process models". In: *Statistical Signal Processing Workshop (SSP), 2011 IEEE*. IEEE, pp. 257–260 (cit. on p. 126).

Rasmussen, C E and C K I Williams (2006). *Regression*. MIT Press, Chapter 2 (cit. on p. 35).

Rey, Hernan Gonzalo, Carlos Pedreira, and Rodrigo Quian Quiroga (2015). "Past, present and future of spike sorting techniques". In: *Brain Research Bulletin* 119. Advances in electrophysiological data analysis, pp. 106 –117 (cit. on p. 69).

Reynolds, Stephanie, Therese Abrahamsson, Renaud Schuck, P. Jesper Sjöström, Simon R. Schultz, and Pier Luigi Dragotti (2017). "ABLE: An Activity-Based Level Set Segmentation Algorithm for Two-Photon Calcium Imaging Data". In: *eNeuro* 4.5 (cit. on p. 71).

Romo, Ranulfo, Carlos D. Brody, Adrián Hernández, and Luis Lemus (1999). "Neuronal correlates of parametric working memory in the prefrontal cortex". In: *Nature* 399.6735, pp. 470–473 (cit. on p. 146).

Sahani, Maneesh (2018). "Describing non-linear latent dynamics". COSYNE Workshop - RNNs: What are we doing and why? (Cit. on p. 119).

Schuck, Renaud, Mary Ann Go, Stefania Garasto, Stephanie Reynolds, Pier Luigi Dragotti, and Simon R Schultz (2018). "Multiphoton minimal inertia scanning for fast acquisition of neural activity signals". In: *Journal of Neural Engineering* 15.2, p. 025003 (cit. on p. 70).

Schultz, Simon R., Kazuo Kitamura, Arthur Post-Uiterweer, Julija Krupic, and Michael Häusser (2009). "Spatial Pattern Coding of Sensory Information by Climbing Fiber-Evoked Calcium Signals in Networks of Neighboring Cerebellar Purkinje Cells". In: *Journal of Neuroscience* 29.25, pp. 8005–8015 (cit. on p. 111).

Schultz, Simon R., Caroline S. Copeland, Amanda J. Foust, Peter Quicke, and Renaud Schuck (2017). "Advances in Two-Photon Scanning and Scanless Microscopy Technologies for Functional Neural Circuit Imaging". In: *Proceedings of the IEEE* 105.1, pp. 139–157 (cit. on p. 70).

Seidemann, Eyal, Yuzhi Chen, Yoon Bai, Spencer C Chen, Preeti Mehta, Bridget L Kajs, Wilson S Geisler, and Boris V Zemelman (2016). "Calcium imaging with genetically encoded indicators in behaving primates". In: *eLife* 5.2016JULY (cit. on p. 147).

Shenoy, Krishna V, Maneesh Sahani, and Mark M Churchland (2013). "Cortical control of arm movements: a dynamical systems perspective". In: *Annual review of neuroscience* 36, pp. 337–359 (cit. on p. 146).

Silverman, Bernard W. (1985). "Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting". In: *Source: Journal of the Royal Statistical Society. Series B (Methodological)* 47.1, pp. 1–52 (cit. on p. 35).

Smith, Spencer L and Michael Häusser (2010). "Parallel processing of visual space by neighboring neurons in mouse visual cortex". In: *Nature Neuroscience* 13.9, pp. 1144–1149 (cit. on pp. 71, 96).

Smola, Alexander J., Rene Vidal, and Vishy Vishwanathan (2004). "Kernels and Dynamical Systems". find at http://alex.smola.org/papers/2004/unpubSmoVidVis04.pdf (cit. on p. 151).

Snelson, Edward and Zoubin Ghahramani (2006). “Sparse Gaussian processes using pseudo-inputs”. In: *Advances in Neural Information Processing Systems* 18, p. 1257 (cit. on p. 35).

Snelson, Edward and Zoubin Ghahramani (2012). “Variable noise and dimensionality reduction for sparse Gaussian processes”. In: (cit. on pp. 123, 124).

Soldado Magraner, Joana (2018). “Linear Dynamics of Evidence Integration in Contextual Decision Making”. In: (cit. on pp. 118, 149).

Speiser, Artur, Jinyao Yan, Evan Archer, Lars Buesing, Srinivas C Turaga, and Jakob H Macke (2017). “Fast amortized inference of neural activity from calcium imaging data with variational autoencoders”. In: *Advances in Neural Information Processing Systems (accepted)* Nips, pp. 1–11 (cit. on pp. 147, 152).

Steinmetz, Nicholas A, Christina Buetfering, Jerome Lecoq, Christian R Lee, Andrew J Peters, Elina A K Jacobs, Philip Coen, Douglas R Ollerenshaw, Matthew T Valley, Saskia E J de Vries, Marina Garrett, Jun Zhuang, Peter A Groblewski, Sahar Manavi, Jesse Miles, Casey White, Eric Lee, Fiona Griffin, Joshua D Larkin, Kate Roll, Sissy Cross, Thuyanh V Nguyen, Rachael Larsen, Julie Pendergraft, Tanya Daigle, Bosiljka Tasic, Carol L Thompson, Jack Waters, Shawn Olsen, David J Margolis, Hongkui Zeng, Michael Hausser, Matteo Carandini, and Kenneth D Harris (Sept. 2017). “Aberrant Cortical Activity in Multiple GCaMP6-Expressing Transgenic Mouse Lines”. In: *eNeuro* 4.5, ENEURO.0207–17.2017 (cit. on p. 21).

Stringer, Carsen and Marius Pachitariu (2019). “Computational processing of neural recordings from calcium imaging data”. In: *Current Opinion in Neurobiology* 55. Machine Learning, Big Data, and Neuroscience, pp. 22 –31 (cit. on p. 69).

Sussillo, David and Omri Barak (2013). “Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks”. In: *Neural computation* 25.3, pp. 626–649 (cit. on p. 118).

Sussillo, David, Rafal Jozefowicz, L. F. Abbott, and Chethan Pandarinath (2016). *LFADS - Latent Factor Analysis via Dynamical Systems* (cit. on pp. 118, 149).

Szlam, Arthur, Koray Kavukcuoglu, and Yann LeCun (2010). “Convolutional Matching Pursuit and Dictionary Training”. In: *CoRR* abs/1010.0422 (cit. on pp. 71, 73).

Titsias, Michalis (2009). “Variational learning of inducing variables in sparse Gaussian processes”. In: *Artificial Intelligence and Statistics*, pp. 567–574 (cit. on pp. 33, 132, 136).

Trautmann, Eric, Daniel O’Shea, Xulu Sun, Jim Marshel, Ailey Crow, Brian Hsueh, Sam Vesuna, Lucas Cofer, Gergo Bohner, William Allen, Isaac Kauvar, Sean Quirin, Matthew MacDougall, Dr. Yuzhi Chen, Matthew Whitmire, Maneesh Sahani, Eyal Seidemann, Charu Ramakrishnan, Stephen Ryu, Karl Deisseroth, and Krishna Shenoy (2019). “Dendritic calcium signals in rhesus macaque motor cortex drive an optical brain-computer interface” (cit. on p. 147).

Vignat, Christophe (2011). “A generalized Isserlis theorem for location mixtures of Gaussian random vectors”. In: *Statistics and Probability Letters* 82.1, pp. 67–71 (cit. on p. 81).

Vishwanathan, Ashwin, Kayvon Daie, Alexandro D. Ramirez, Jeff W. Lichtman, Emre R. F. Aksay, and H. Sebastian Seung (2017). "Electron Microscopic Reconstruction of Functionally Identified Cells in a Neural Integrator". In: *Current Biology* 27.14, 2137–2147.e3 (cit. on p. 111).

Wan, Eric A. and Rudolph Van Der Merwe (2000). "The unscented Kalman filter for nonlinear estimation". In: *Technology* v, pp. 153–158 (cit. on p. 126).

Wang, Hui, Xiaoli Chen, and Jinqiao Duan (2018). "A Stochastic Pitchfork Bifurcation in Most Probable Phase Portraits". In: pp. 1–9 (cit. on p. 118).

Whitaker, Michael (2010). "Genetically-encoded probes for measurement of intracellular calcium". In: *Methods in cell biology* 99, pp. 153–182 (cit. on p. 26).

Wilson, Andrew Gordon (2014). "Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes ". In: (cit. on p. 36).

Wilson, Andrew Gordon and Ryan Prescott Adams (2013). "Gaussian Process Kernels for Pattern Discovery and Extrapolation". In: 28 (cit. on p. 36).

Wilson, Andrew Gordon and Hannes Nickisch (2015). "Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)". In: 37 (cit. on pp. 35–37, 151).

Wu, Anqi, Nicholas A. Roy, Stephen Keeley, and Jonathan W Pillow (2017). "Gaussian process based nonlinear latent structure discovery in multivariate spike train data". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 3496–3505 (cit. on p. 118).

Young, Michael D., Jeffrey J. Field, Kraig E. Sheetz, Randy A. Bartels, and Jeff Squier (2015). "A pragmatic guide to multiphoton microscope design". In: *Advances in Optics and Photonics* 7.2, p. 276 (cit. on p. 28).

Yu, Byron M., John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani (2009). "Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity". In: *Journal of Neurophysiology* 102.1. PMID: 19357332, pp. 614–635 (cit. on p. 118).

Yuan, Ming and Yi Lin (2006). "Model selection and estimation in regression with grouped variables". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1, pp. 49–67 (cit. on p. 72).

Zhao, Yuan and Il Memming Park (2016). "Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains". In: pp. 1–25 (cit. on p. 118).