Two-Photon Calcium Imaging Sequence Analysis Pipeline: A Method for Analyzing Neuronal Network Activity

by

Raoul-Emil Roger Khouri

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author . . . .
Department of Electrical Engineering and Computer Science
May 12, 2018

Certified by. . . . .
Una-May O'Reilly PhD.
Principal Research Scientist, MIT CSAIL
Thesis Supervisor

Certified by. . . . .
Erik Hemberg PhD.
Research Scientist, MIT CSAIL
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chairman, Master of Engineering Thesis Committee

Two-Photon Calcium Imaging Sequence Analysis Pipeline: A Method for Analyzing Neuronal Network Activity

by

Raoul-Emil Roger Khouri

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Investigating the development of neuronal networks can help us to identify new therapies and treatments for conditions that affect the brain, such as autism and Alzheimer's disease. Two-photon calcium imaging has been a powerful tool for the investigation of the development of neuronal networks. However, one of the major challenges of working with two-photon calcium images is processing the large data sets, which often requires manual analysis by a skilled researcher. Here, we introduce a machine learning (ML) pipeline for the analysis of two-photon calcium image sequences. This semi-autonomous ML pipeline includes proposed methods for automatically identifying neurons, signal extraction, signal processing, event detection, feature extraction, and analysis. We run our ML pipeline on a dataset of two-photon calcium image sequences extracted by our team. This dataset includes two-photon calcium image sequences of spontaneous network activity from primary cortical cultures of Mecp2-deficient and wild-type mice. Loss-of-function mutation in the *MECP2* gene, causes 95% of Rett syndrome cases and some cases of autism. We evaluate our ML pipeline using this dataset. Our ML pipeline reduces the time required to analyze two-photon calcium images from over 10 minutes to about 30 seconds per sample. Our goal is to accelerate the analysis of neuronal network function to aid in our understanding of neurological disorders and the identification of novel therapeutic targets.

Thesis Supervisor: Una-May O'Reilly PhD.
Title: Principal Research Scientist, MIT CSAIL

Thesis Supervisor: Erik Hemberg PhD.
Title: Research Scientist, MIT CSAIL

Acknowledgments

I would like to thank Una-May O'Reilly PhD., for all of her guidance throughout this research project. I would also like to thank Erik Hemberg, PhD., not only his for his help in finding a project that interested me but also his strong guidance along the way. I would also like to thank Susanna Mierau, MD, DPhil. for help with collection of the data used in these experiments and with evaluation and expert opinions on the results produced by the techniques discussed in this paper. I would also like to thank Michael Delaus for his help in implementing the genetic neuron detection algorithm discussed in this paper.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Two-photon calcium imaging has been a powerful tool for the investigation of the development of neuronal networks. However, one of the major challenges of working with two-photon calcium images is processing the large data sets, which often requires manual analysis by a skilled researcher. In this paper, we propose a semi-automated ML pipeline that can help streamline the analysis of two-photon calcium images, and help us investigate the development of neuronal networks. This can help us to identify new therapies and treatments for conditions that affect the brain, such as autism and Alzheimer's disease.

1.1 Background

1.1.1 Autism

Autism is an extremely common (1 in 68 children in the U.S.)[1] and chronic neurological and developmental disorder with no known cure. Autism is defined by deficits in social communication and interaction. People with autism often have obsessive interests, repetitive behaviors, and altered sensory processing. There are more then 200,000 reported cases per year in the US [4]. Unfortunately, our current understanding of how autism affects the development of neuronal networks in the brain is very limited [4]. Two-photon calcium imaging provides an excellent tool for investigating the development of spontaneous network activity in neurons in an in vitro model of autism; however, the current manual pipelines employed by many neuroscientists do not allow for a large data set to be analyzed.

1.1.2 Neural Network Development

Brain cells communicate with each other using a combination of electrical and chemical signals. These communications are primarily done through action potentials, which are rapid changes of electrical charge across the cell membrane. These action potentials can propagate along to distant parts of the cell along axons, which at their terminals, create synapses with other neurons. When the action potential reaches the axon terminal, it can trigger the release of neurotransmitters into the synaptic cleft (space between the two neurons). When the neurotransmitters bind to specific receptors on the receiving neuron, ion channels open, which allow ions to flow into the cell, changing the voltage across the membrane and allowing calcium to enter the cell. Thus, transient changes in calcium levels inside a neuron can serve as an indicator of neuronal activity; however, the calcium transients do not have the temporal resolution to identify individual action potentials. Action potentials occur on a millisecond time scale; however, bursts of action potentials in an active neuron lead to calcium fluctuations that may last for seconds. Thus, calcium imaging can be used to compare neuronal activity in a population of neurons growing in culture. Neurons are harvested from neonatal mice brains and can be grown in a dish for months. As these neurons mature, they become spontaneously active and form functional connections with other neurons. Network development can be detected when some groups of neurons show synchronous activity [8].

1.2 Aims of This Project

Previous research in our group [5], discusses the use of two-photon calcium imaging of mouse cortical neurons to find clusters of cells with correlated activity. Unfortunately, the sample size of previous work was limited since we had to manually identify individual neurons for each image series. Each ten-minute long recording of calcium transients in a population of neurons consists of 1600 images. For each recording, using previous methodology, each neuron's cell body must be manually identified and the borders manually drawn, and the change in calcium fluorescence is calculated within these regions of interest.

Neuron Detection

In this project, we have streamlined the process for identifying neurons in two-photon calcium imaging recordings by using an unsupervised learning model. We researched and implemented multiple image recognition techniques and compared the results. With this unsupervised learning process we can now analyze a much large data set and begin to compare the effects of autism on the development of these clusters [8]. We also generated a graphical user interface for not only easily labeling neurons by hand, but also for evaluating and inspecting the neurons labeled by our models.

Signal Extraction and Processing

After identifying neurons in the recordings, we can extract signal intensity from the specific regions in the images that correspond to each neuron. We use signal processing techniques for finding the underlying signals and filtering out noise.

Signal Event Detection

With the extracted and denoised signals, we can begin to do event detection. Burst-activity and slow-oscillations are the two main types of activity analyzed. We explore and evaluate methods for detecting slow oscillations and burst events in our recording. The bursts events are thought to be due to bursts of activity potentials in a neuron, while the slow oscillations may be related to up-and-down states, which synchronize activity within neural networks. We also generated a graphical user interface for not only easily labeling bursts by hand, but also for evaluating and inspecting the bursts labeled by our algorithm.

Feature Extraction and Analysis

With events identified in the individual signals we can begin to analyze the inter-neuronal communication. Correlations in activity are associated with inter-neuronal network activity.

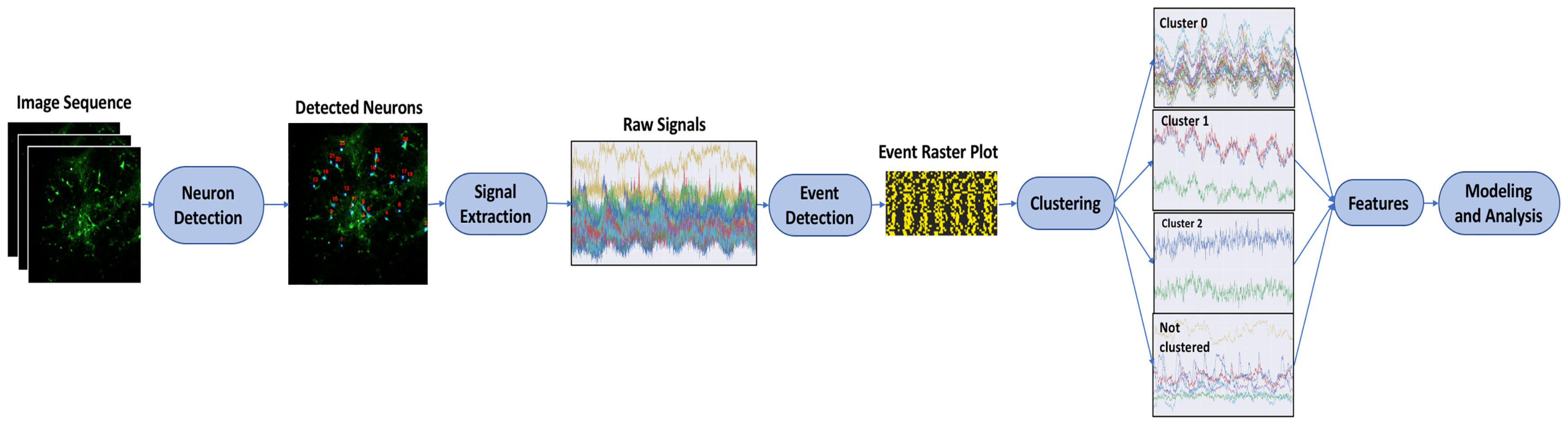

Figure 1-1: Overview of Proposed Pipeline. First an image sequence is captured from the mice brain cultures using two-photon calcium imaging. Next, our neuron detection algorithm identifies pixels that are part of the neuron cell bodies. Next, we measure the signals for each of the neurons by extracting the pixel intensity at the locations identified in each frame. Next, we use event detection to find specific events that correspond to neuronal activity in the signals. Next, we cluster signals that have similar event timings. Then, we extract useful features from the clusters. Finally, we analyze and model the features and compare differences between genotypes

We explore techniques for grouping signals and then evaluating similarities within and between clusters. These features can be used to observe and quantify neuronal activity within a population of signals. This process can be done on a full dataset to compare genotype or age related differences between the groups.

1.3 Structure of thesis

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the Mecp2 data, which we used in this project. This includes how the two-photon calcium images are generated. Chapter 3 presents six proposed solutions for neuron detection. This includes a quantitative evaluation of each solution. Chapter 4 describes how signals are extracted from the two-photon calcium imaging sequence once neurons have been detected. In Chapter 5, we explore methods for detecting different events in the signals extracted. Chapter 6 described our methods for finding similarities between neuronal signals and clustering neurons. Chapter 7 analyzes the data, and describes the features extracted as well as our methods of extraction. We list our conclusions, future works, and Implications in Chapter 8

Chapter 2

Data

2.1 MECP2 Data Set

Our data set is comprised of 62 10-minute timelapse images, with 1600 frames (just under 3 frames/second). We use two-photon calcium imaging to collect raw fluorescence intensities for each pixel in a frame (resolution 256x256). An example of one of these images can be seen in Figure 2-1. This dataset includes two-photon calcium imaging sequences of spontaneous network activity from mouse primary cortical cultures of 39 Mecp2-deficient and 23 wild-type mice.

2.2 Data Collection

Dr. Susanna Mierau performed the collection of data for this study. Cortical cultures for studying network activity were initiated by sacrificing mouse pups on the day of or one day after birth. To harvest the neurons for culture, the cortex was quickly dissected from the brain and dissociated first using the enzyme papain (1:1 in DPBS) at 37^oC for 25 minutes. The action of papain was stopped by adding 4% fetal bovine serum (FBS). Cells were then manually dissociated and collected by centrifuging for 10 minutes with relative centrifugal force (rcf) of 0.4. Afterwards, the cell pellet was re-suspended in 1 mL of Neurobasal (NB) media with the B27 supplement. The dissociated cells were then plated on coverslips and stored in the incubator at 37^oC with 5% CO_2 and 19% O_2. Cell cultures were maintained

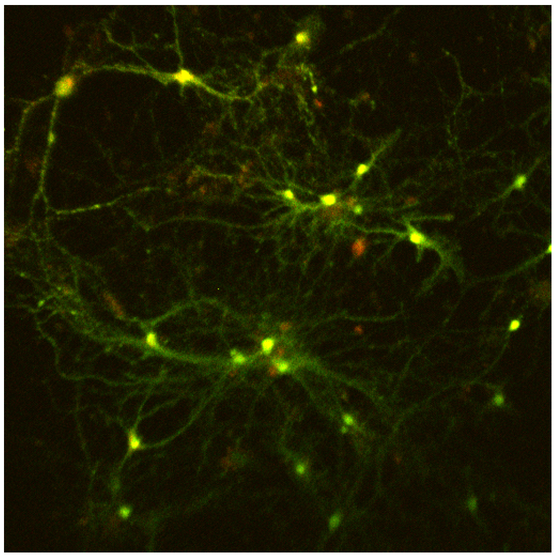

Figure 2-1: Two-Photon Calcium Image of Cortical Cultures From a Mouse. The green fluorescence is produced by a calcium indicator fluorescent dye. Transient changes in the calcium concentration occur in the neuron cell body and the extensions, axons and dendrites. The relative change in calcium concentration is proportional to the amount of activity.

for up to 12 weeks by exchanging one-third of the NB-B27 media (including L-glutamine) three times per week with fresh media. Calcium imaging was performed in cultures at 6 weeks (days-in-vitro, DIV, 42-44) and 8 weeks (DIV 56). The calcium indicator fluorescent dye, Oregon Green 488 BAPTA, was prepared by adding 8 μL of pluronic acid to dye. After 5 minutes mixing via sonification, 72 μL of artificial cerebral spinal fluid (ACSF) was added. Immediately before application to the cultured cells, the dye mix was diluted 1:8 in warm media. The coverslips were then transferred into a 35 mm petri dish and 40 μL of the dye in media was added. The cells were incubated for 5-6 minutes, after which the coverslip was gently washed three times with warm NBB27 media to remove excess dye. The coverslips were then transferred to the two-photon rig in 2 mL of warm NB-B27 media. Two-photon calcium imaging was performed using the Slidebook software (3i). For the recording, the coverslip was transferred to a warm bath (28-30^oC) with a circulating flow of HEPES-based ACSF. The position of the slide under the microscope was adjusted to maximize the number of cells visible within the frame (517.1 μm x 517.1 μm). A 1600 frame time lapse (approximately 10 minutes recorded with frequency of 3 Hz) recorded the changes in the green fluorescence (excitation wavelength 800-920 nm) from a single Z plane with resolution of 256 x 256 pixels [5]. Since the coverslips contain a single layer of neurons, no adjustments are required to eliminate interfering signals from other cells above or below the recorded cells that can be present in other in vitro or in vivo preparations.

Chapter 3

Neuron Detection

To begin analyzing large data sets of two-photon calcium images, the detection of neurons needs to be automated. Unfortunately, we do not have a large training set of labeled neurons. Also, labeling a single image is already a time consuming process for a trained neuroscientist, so generating a new training set is not a realistic option for our team. Therefore, our team was constrained to using unsupervised learning methods for neuron detection.

Thankfully, there has been similar work on detecting neurons in similar data. The Neurofinder competition [1] is an open source competition that invites data scientists from around the community to generate models for detecting neurons in two-photon calcium image sequences. However, the Neurofinder competition works on in vivo recordings. These rodents have a genetic mutation in which some cells express a calcium-sensitive fluorescent protein that changes its fluorescence depending on neural activity. The investigators open up a small window on the skull and use a similar microscope/laser set up, to record transient changes in calcium. Their images look grainy because, in contrast to the 2D layer of cells growing in cell culture, they have a 3D cellular architecture and must image through other (non-fluorescent) tissue. In addition to poorer resolution, they also have to do some addition subtraction or adjustment techniques to eliminate effects from fluorescence in cells above or below the plain where the image is taken for the putative cell. Previous work using unsupervised learning has been successful at identifying cells in these in vivo two-photon calcium imaging studies. [6] [7]. In these papers, they run a threshold and contour detection algorithm; thus, we have

[1] http://neurofinder.codeneuro.org

implemented and evaluated this method as well as new models on our data set.

3.1 Neuron Detection Methods

One of the main challenges of generating an accurate neuron detection algorithm is that we are constrained to unsupervised models since we do not have a large dataset of labeled neurons. In order to develop the unsupervised neuron detection part of our pipeline, we have developed six unsupervised methods and compared them. Our models all work by labeling the x and y of the center of a neuron and the radius, where the neuron is considered to be circular (this is not a perfect assumption; however, this works well in practice). We have also developed a graphical user interface for manually identifying neurons, which we use to generate a gold standard to quantitatively evaluate the different neuron detection models.

3.1.1 Threshold and Contour

The first method for neuron detection we tested came from the previous works on the Neurofinder competition. For the threshold and contour method, we start off by running an adaptive threshold through the first image in the sequence. The adaptive thresholding compares each pixel to the mean of the pixels in its local block. The pixel-in-question's intensity must be c greater then the local block mean for it to be assigned a value of 1, or 0 otherwise. The size of a block and the value of c are hyper-parameters that must be tuned. For our experiments we used a block size of 21 and 75 for c. We then run contour detection on the binary image outputted by the thresholding. This contour detection uses a union find algorithm to find all sets of connected ones. Then the center of each connected region is found using the center of mass, and the radius of the contour is estimated. We used the openCV [9] implementation for the contour detection. Figure 3-1 shows the affect of using the adaptive thresholding on a single frame.

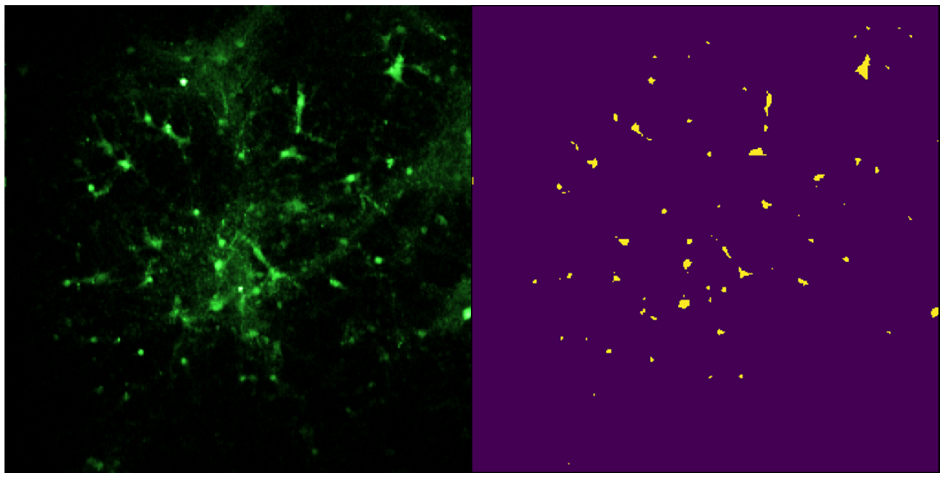

Figure 3-1: Example of Thresholded Image

Figure 3-2 shows neurons identified by the Threshold and Contour method. As one can see, this method performs well next to the neurons identified by the neurologist. This model was able to get 55.3% average recall and 77.3% average precision, when tested against our gold standard. Unfortunately, this model performs poorly in highly concentrated areas, this can be seen in Figure 3-2.

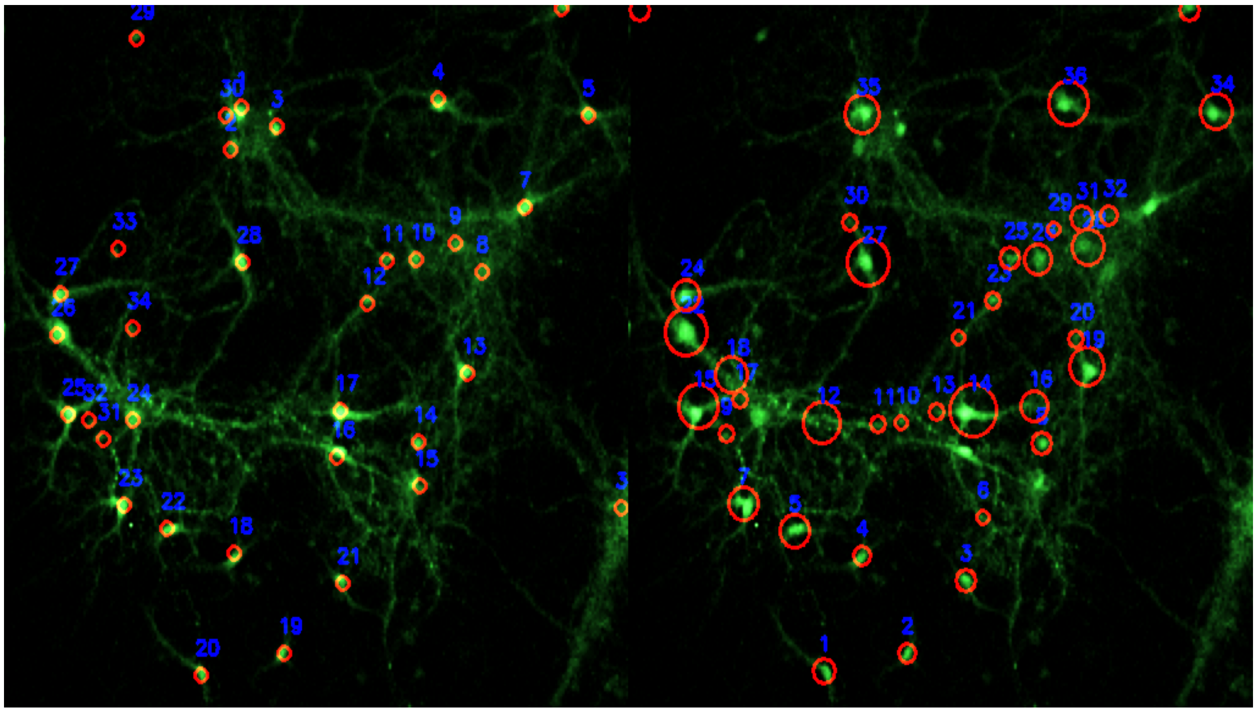

Figure 3-2: Neurons identified by neurologist on left. Neurons identified by Threshold and Contour method on right.

3.1.2 Template Matching

The team also explored running a template matching algorithm for finding neuron-like shapes in the two-photon calcium images. The template matching technique is used for finding areas in a given image that match (or are similar) to a template image[9]. In our implementation, we extracted a bag of templates (Figure 3-3) and used the openCV template matching algorithm with the correlation coefficient metric. The correlation coefficient equation which is used to evaluate the strength of a match can be seen in equation 3.1. Using the correlation coefficient equation we can generate a similarity image. We then threshold; through trial and error we found that a threshold value of 200,000 worked well. Finally we use Birch clustering to find the centers of the matched regions, and again through trial and error we found that a Birch threshold value of 10 worked well.

$$\begin{gathered} R(x,y) = \sum_{x',y'} \left(T'(x',y') \cdot I'(x+x',y+y')\right) \\ T'(x',y') = T(x',y') - \frac{\sum_{x'',y''} T'(x'',y'')}{w \cdot h} \\ I'(x+x',y+y') = I(x+x',y+y') - \frac{\sum_{x'',y''} I'(x+x'',y+y'')}{w \cdot h} \\ I_{i,j} = \text{pixel i,j of Image. } T_{i,j} = \text{pixel i,j of Template.} \end{gathered} \tag{3.1}$$

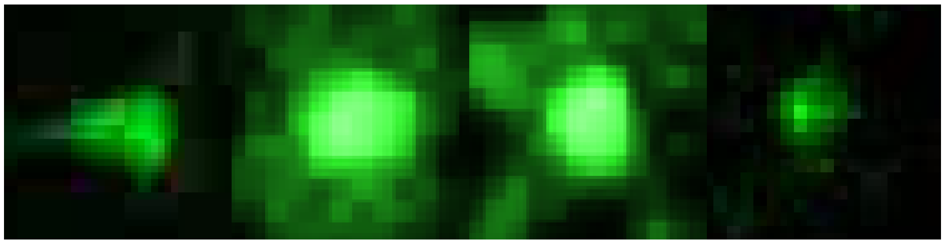

Figure 3-3: Examples of our Two-Photon Neuron Templates

Figure 3-4 shows the neurons identified by template matching. As one can see, this method performs well next to the neurons identified by the neurologist. This model was able to get 61.2% average recall and 75.4% average precision, when tested against our gold standard. The issues with this model is that we do not get an indication of the size of the neurons (this shows up as all the neurons being identified to have the same radius). This model is also highly sensitive to the bag of neuron templates used.

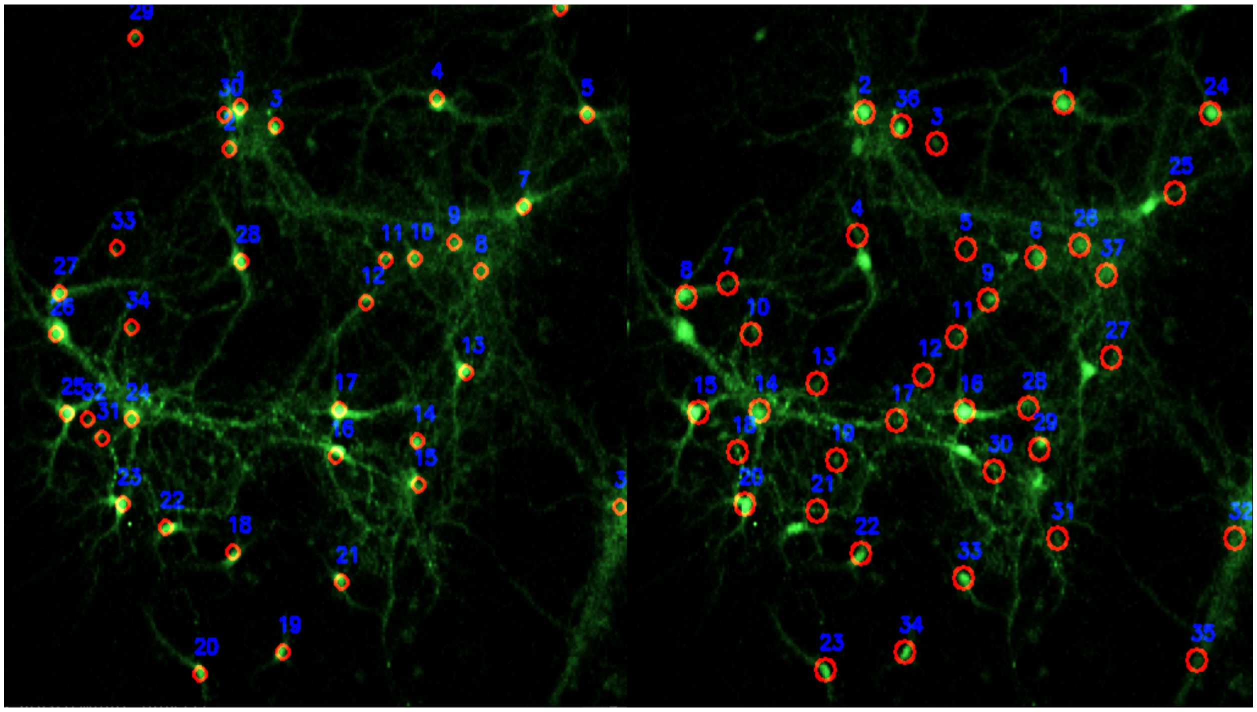
Figure 3-4: Neurons identified by neurologist on left. Neurons identified by Template Matching on right.

3.1.3 Gaussian Mixture Model

We realized that the neurons could be modeled as Gaussian sources that emit fluorescence in the area around them. We then decided to model a two-photon calcium image as a result of a Gaussian mixture model with a certain number of Gaussians. The big issue with this model is that users would have to know the number of neurons in the sequence in advance in order to initialize the model. With too few Gaussians in the initialization we would get a return that has some of the neurons well labeled and a few large Gaussians that tried to explain the remaining fluorescence. On the other hand, if we have too many Gaussians the model losses significant amounts of precision. In order to train a Gaussian mixture model, we needed to generate a binary image; thus, we run an adaptive threshold (see Section 3.1.1) with a block size of 75 and c of 75. For the analysis of the Gaussian mixture model we used the 20 for the number of Gaussians, since this was roughly the mean number of neurons. Then we considered the neuron cell body to be with the one standard deviation ring drawn.

However, as one can see in Figure 3-5 some of the Gaussians drawn can be very large (trying to explain all the noise). Thus, in order to control for this we cap the radius of a neuron to 20 pixels for evaluation.

Figure 3-5 shows the neurons identified by the Gaussian Mixture Model. As one can see, this method does not perform well next to the neurons identified by the neurologist. This model was able to get 43.8% average recall and 91.7% average precision, when tested against our gold standard. The issue here is that the number of Gaussians that the Gaussian Mixture Model was initialized with was too low. Because of this the model tried to explain a significant amount of the fluorescence with a large Gaussian over the entire image.

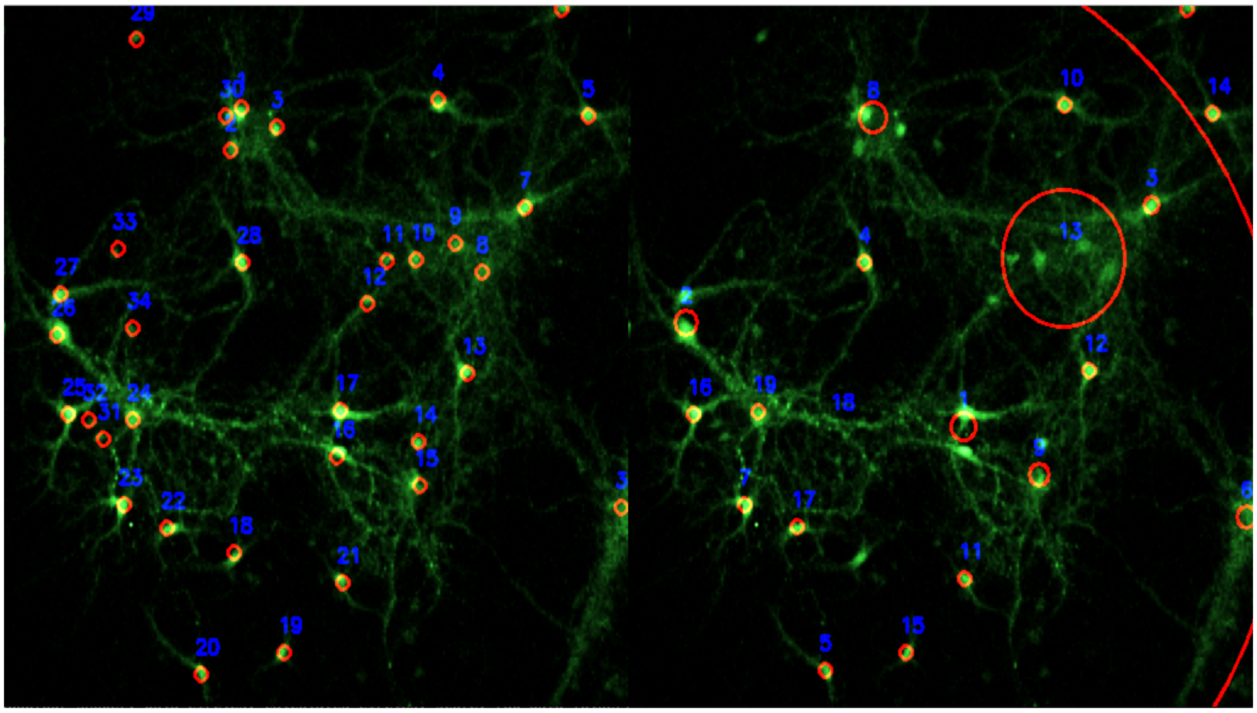

Figure 3-5: Neurons identified by neurologist on left. Neurons identified by Gaussian Mixture Model on right.

3.1.4 Genetic Neuron Detection

Genetic algorithms offer a potential new solution to finding and identifying neurons in two-photon calcium images. Genetic learning belongs to a larger class of evolutionary algorithms, which are inspired by the process of natural selection in nature. Genetic algorithms are used to find solutions to problems by using evolutionary methods such as reproduction, mutation, and selection. We have implemented a genetic algorithm for the detection of neurons.

The key components of the genetic algorithm are the representations of the candidate solutions, the operators which are used to evolve the solutions over generations, and the scoring function, which is used to evaluate the candidate solutions. The basic algorithm and diagram for the genetic neuron detection model are shown in Algorithm 1 and Figure 3-6, respectively.

Algorithm 1 Pseudo-code for Genetic Algorithm

```
1: INITIALIZE population of candidate solutions
2: EVALUATE initial population
3: for i = 0 to N do
4:     SELECT best candidates for reproduction
5:     BREED new candidates through reproduction and mutations
6:     EVALUATE new population
```

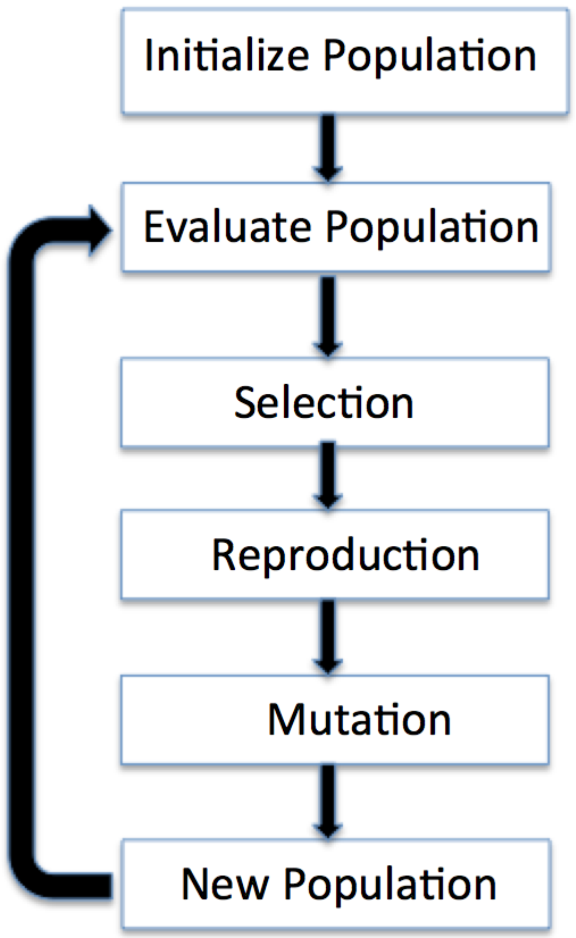

Figure 3-6: Diagram of Genetic Algorithm

An individual in the population is represented by a unique "bag" of circles, which vary in size. Each circle has a x-coordinate, a y-coordinate, and a radius. The initial population is comprised of 100 individuals. In order to intelligently choose the locations of the initial circles for a candidate, we average all the images in a sequence. Next, N (the size of the candidates bag) number of (x,y) coordinates are chosen, with brighter pixels having a higher probability of being selected. This ensures that a given candidate already has a set of circles that cover possible neurons, allowing for the genetic algorithm to converge quickly. These N circles are then each assigned a radius, with each radius chosen form a Gaussian distribution with a predetermined mean and standard deviation. A sample candidate's initial bag of circles is shown over the neuron image in Figure 3-7. As can be seen in Figure 3-7, the initial circles are well placed over the bright pixels in the image.

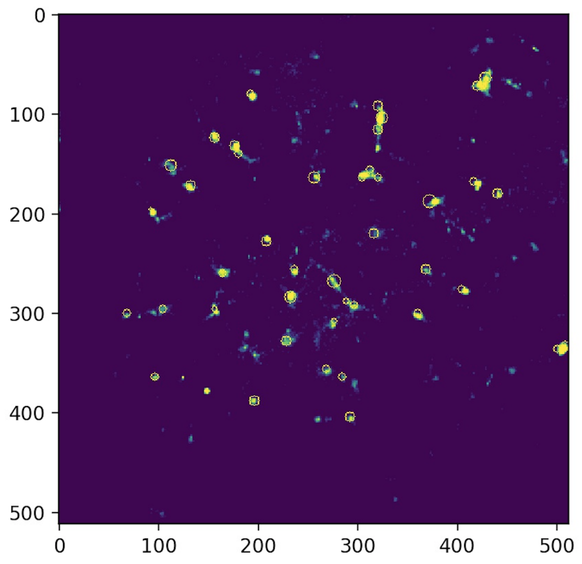

Figure 3-7: An Initial Genetic Neuron Candidate

The scoring function is very important to the genetic algorithm. The scoring algorithm tell us the fitness of an individual in the population. The score for a candidate can be written as:

$$\sum_{i=1}^{N} \frac{\sum_{j=x_i-r_i}^{x_i+r_i} \sum_{k=y_i-r_i}^{y_i+r_i} \text{image}(j,k)}{r_i} - N \tag{3.2}$$

With this scoring function, having too many neurons was a negative as well as too large of neurons. For reproduction and mutation, new candidates would be made from either two or one parent(s), respectively. In the case of reproduction, the new candidate would take a set of circles from both parents circles. In the case of a mutation, a new candidate would be created from a mutation of the parent. In Figure 3-8 shows, over 15 generations, the plot of the scores of the best candidate as well as the average of the population.

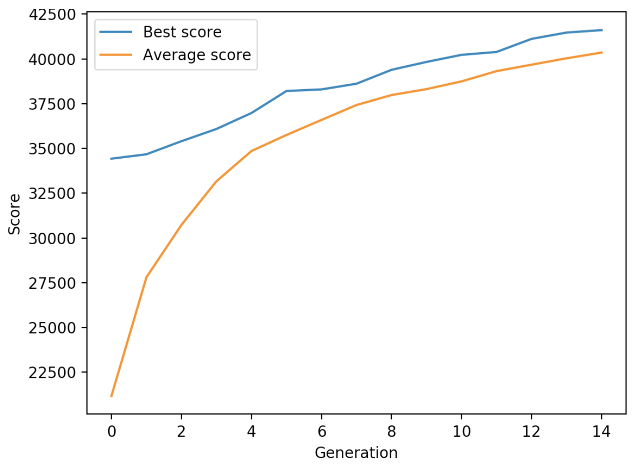

Figure 3-8: Plot of Best and Average fitness throughout the first 15 generation using a population of size 100.

Unfortunately, when compared with other models, the genetic neuron detection model did not perform well. As one can see in Figure 3-9, the circles created by genetic neuron detection are reasonable. However, the genetic neuron detection only had an average recall of 56.3% and an average precision of 68.4%. The runtime of the genetic neuron detection is also quite high. It takes roughly 30 seconds to detect the neurons of a single sample.

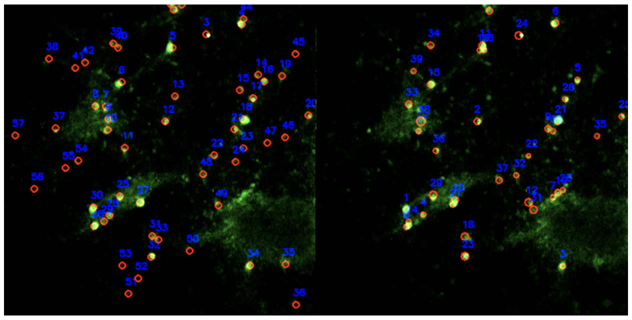

Figure 3-9: Neurons identified by neurologist on left. Neurons identified by genetic neuron detection method on right.

3.1.5 Threshold, Aggregate, and Contour

This model is very similar to the Threshold and Contour model; however, here we tried to incorporate all the images within the sequence. Using multiple images we gain information that helps remove more noise. In Figure 3-10 one can see a visualization of an image sequence.

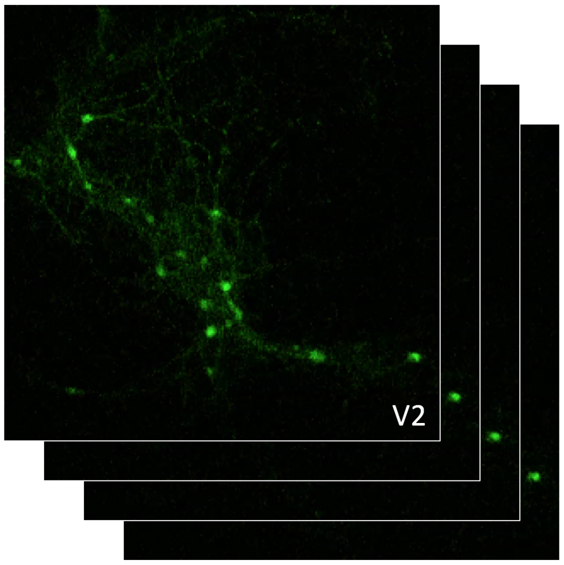

Figure 3-10: Two-Photon Neuron Image Sequence

The implementation of this method requires us to threshold each image in the sequence individually. We use the same adaptive thresholding method described in Section 3.1.1. For our experiments we used a block size of 21 and 75 for c; unfortunately, these hyper-parameters are not optimized, but we found a method for optimizing them which is discussed in Section 3.3.2. We then sum all images within a sequence and normalize the aggregate image. An example of such image can be seen in Figure 3-11.

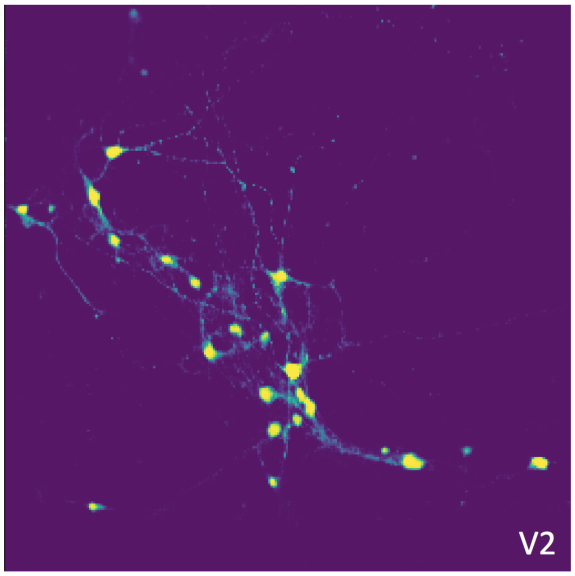

Figure 3-11: Two-Photon Calcium Image After Thresholding and Aggregating

We then take the average aggregate image and run a final adaptive thresholding to turn the average aggregate image into a binary image. We then run the contour detection on the thresholded binary image. The output of this Threshold, Aggregate, and Contour method can be seen in Figure 3-12 and Figure 3-13.

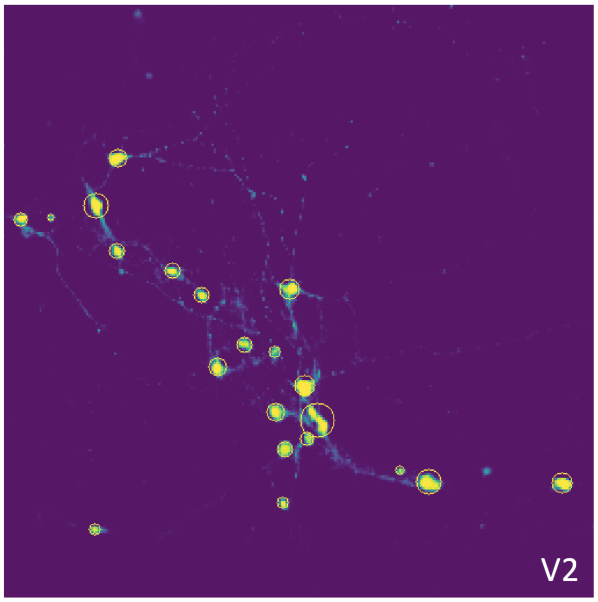

Figure 3-12: Two-Photon Calcium Image After Contouring

This technique gains a significant amount of improvement over the previous models since it is capable of taking in more data about the images within the sequence. This allows this model to remove a larger amount of noise. We can see this in Figure 3-11 where noise that survived individual image thresholding are shown as blue; however, the fluorescence that comes from the consistent locations (the neurons) survive the individual thresholds much more frequently and and shown in yellow. Unfortunately, it does take slightly longer to compute the aggregated image since all 1600 images in the sequence must be processed. However this process still runs in a relatively short time. It takes about 17 seconds on a 2.8 GHz Intel Core i7 laptop with 16 GB of RAM to run the entire process.

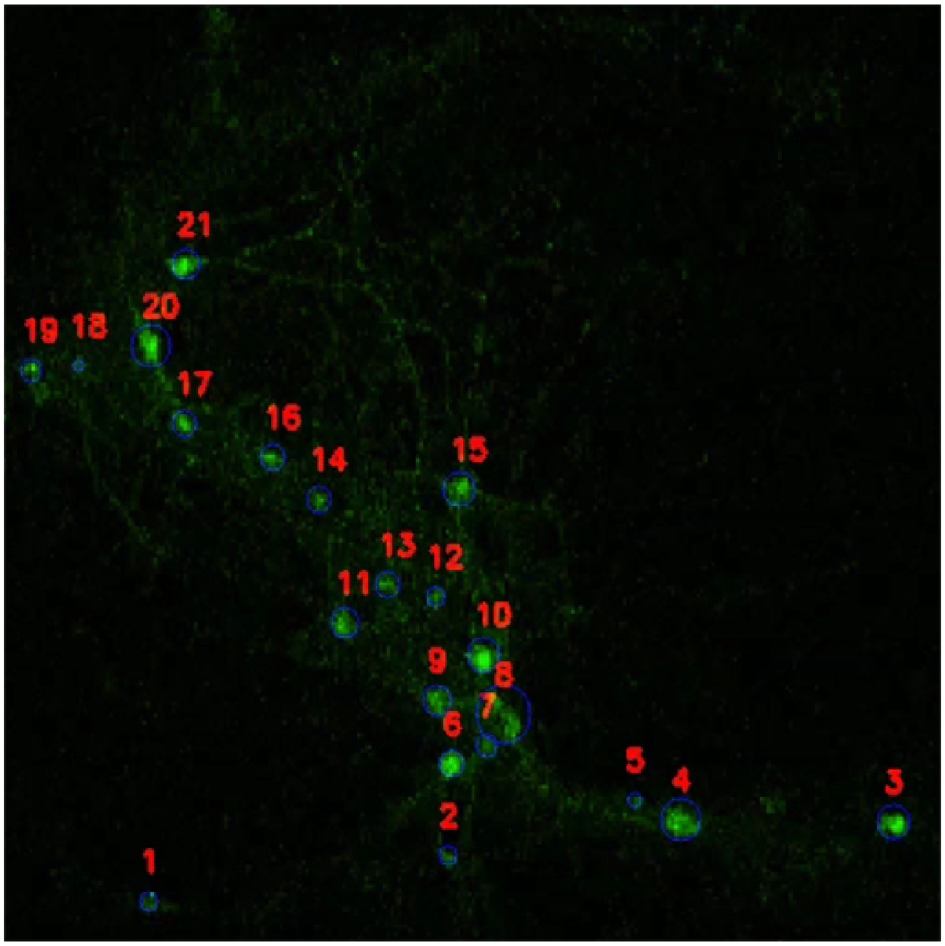

Figure 3-13: Two-Photon Calcium Image Labeled using Threshold, Aggregate, and Contour

Figure 3-14 shows the neurons identified by the Threshold, Aggregate, and Contour method. This method perform very well next to the neurons identified by the neurologist.

The Threshold, Aggregate, and Contour neuron detection method had an average recall of 56.3% and an average precision of 68.4%.

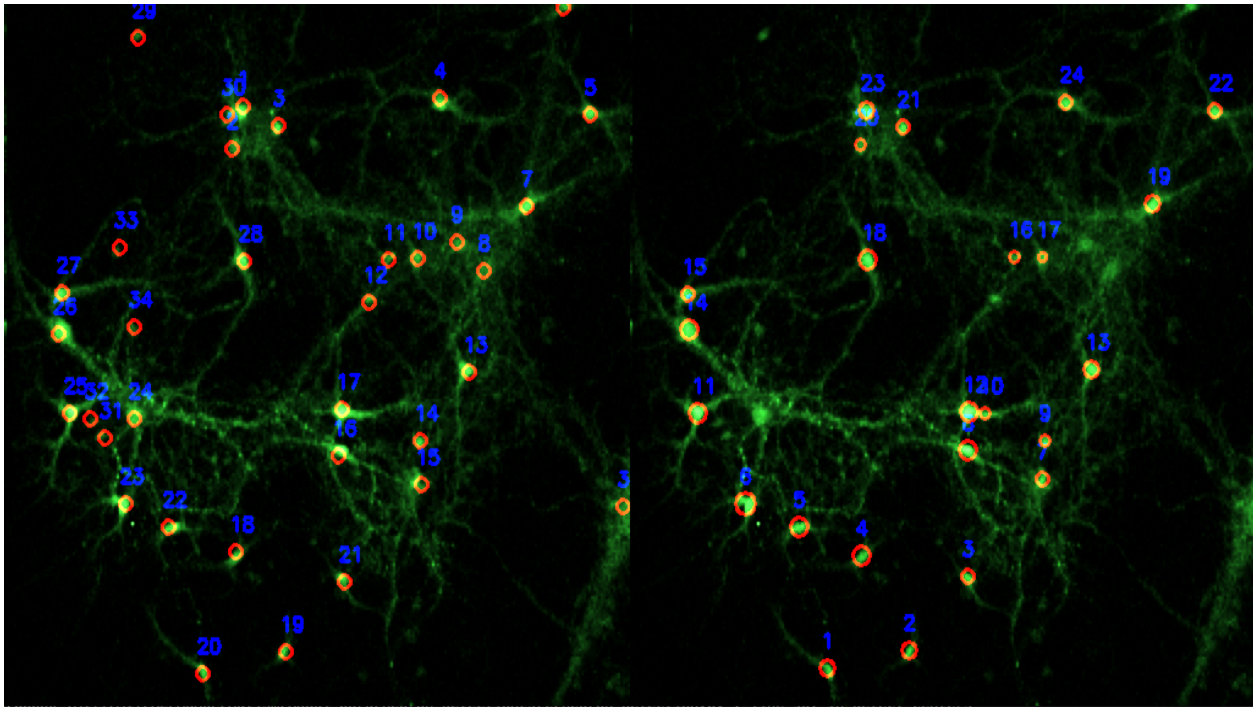

Figure 3-14: Neurons identified by neurologist on left. Neurons identified by Threshold, Aggregate, and Contour method on right.

3.1.6 Gaussian Threshold, Aggregate, and Contour

Since our algorithm is highly sensitive to the thresholding that is used we experimented with using Gaussian Mixture Model for thresholding our image. This is very similar to the work done by image segmentation using Gaussian Processes [2]; however, we only have background and neurons so it becomes a thresholding algorithm. We ran our Threshold, Aggregate, and Contour algorithm using a Gaussian Mixture Model for the thresholds.

Figure 3-15 shows the neurons identified by the Gaussian Threshold, Aggregate, and Contour method. Unfortunately, this method does not perform as well as the adaptive thresholding method. The issue being that the two-photon calcium images have a lot of noise pixels that are very similar to the pixels pertaining to the neurons. The GP model likes to keep the pixels that pertain to fluorescence from the background, as being part of a neuron. This means that large areas of noise are identified as being neurons.

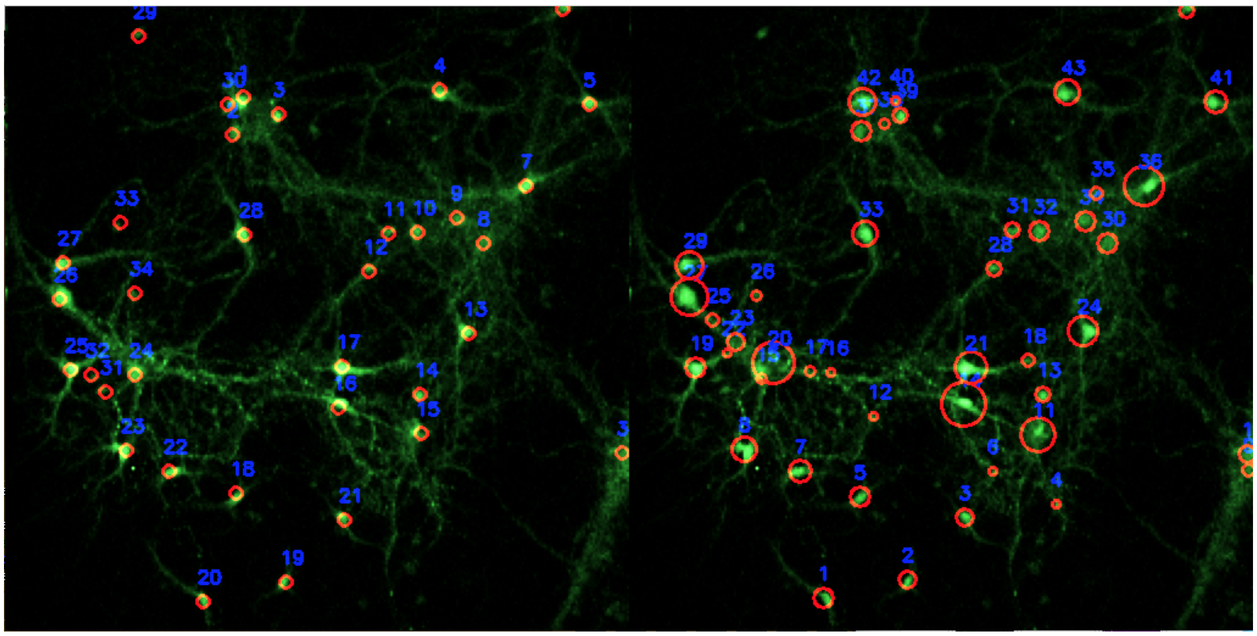
Figure 3-15: Neurons identified by neurologist on left. Neurons identified by Gaussian Threshold, Aggregate, and Contour method on right.

3.2 Image Drifting

One issue in the two-photon calcium image sequences is that over time the position of the camera relative to the glass cover slip can begin to drift very slightly (by 1 to 2) μm. This drift can be easily measured and corrected using the python imreg_dft [3] module, which compares images in the Fourier domain and can find the offset between the two. However, as mentioned in previous works, the amount of image drift is often negligible [5]. Figure 3-16 shows that in most of images the maximum amount of drift is less then 1 pixel. However, for our pipeline we do correct these drifts, since in severe cases it can cause problematic artifacts in the extracted signals.

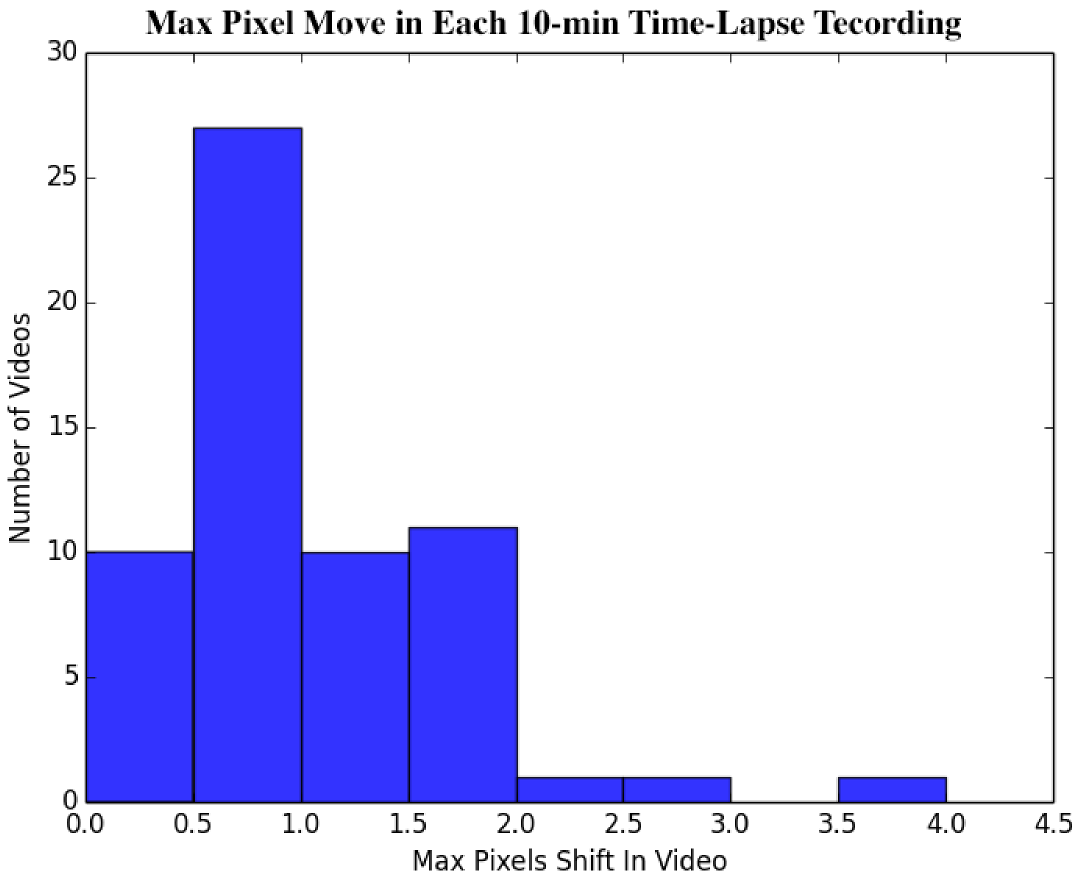

Figure 3-16: Maximum Realignment Required in Each Sequence

3.3 Evaluation of Neuron Detection Methods

For neuron detection we need to be able to not only find most of the neurons (high recall) we must also have a low false positive rate (high precision). The selected method must also be able to compute in a reasonable amount of time. In order, to evaluate each of the methods we create a graphical user interface for manually creating a gold standard, which we then use to calculate precision and recall of each of our models.

3.3.1 Neuron Find Tool

Unfortunately when starting this project, the team had no gold standard for evaluating the precision and recall of the system. In order to make a training set, I created a tool where a trained neuroscientist can manually identify neurons in two-photon calcium images and export the identified regions in csv format. Figure 3-17 shows the interface of the tool.

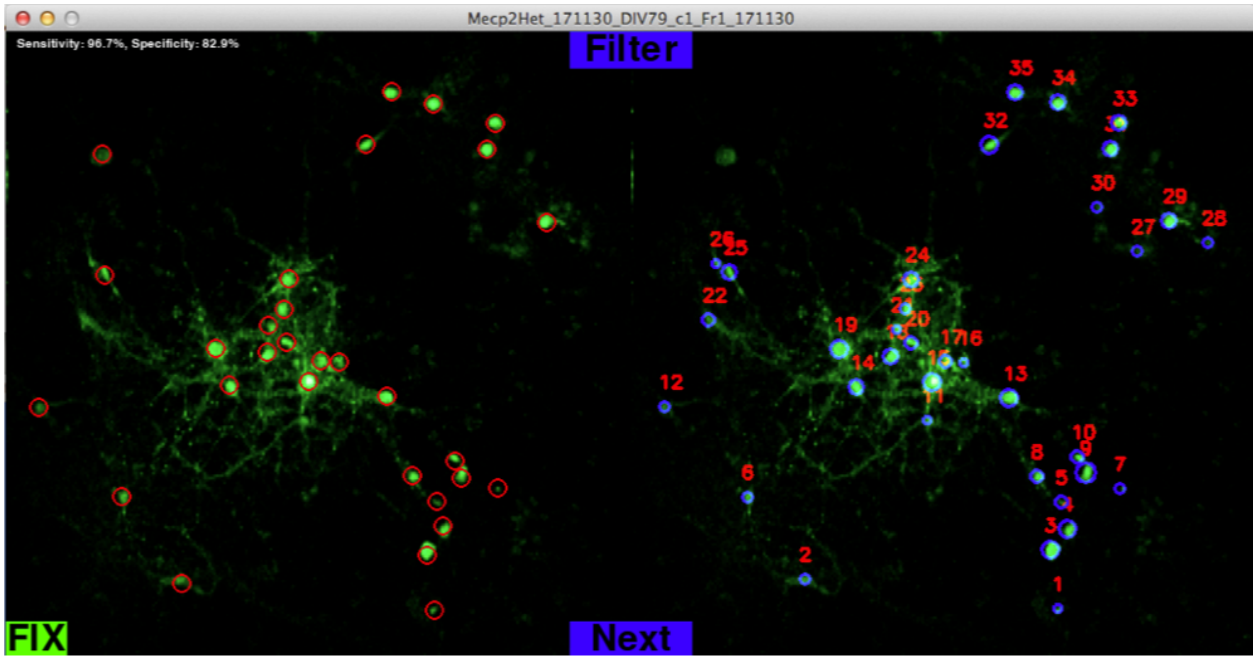
Mecp2Het_171130_DIV79_c1_Fr1_171130
Sensitivity: 96.7%, Specificity: 82.9%
Filter
FIX
Next

Figure 3-17: Neuron Find Tool Interface. On left image of neurons identified by neuroscientist. On the right, neurons identified by our automated neuron detection method.

With this tool we were able to generate a small dataset. We then used 80% of the generated data as training data for improving the hyper-parameters of our models and the remaining 20% of data for validation.

3.3.2 Evolutionary Parameter Optimization

We used an evolutionary parameter optimizer in order to refine the hyper-parameters of the Threshold, Aggregate, and Contour method. We used 80% of the gold standard data collected using the Neuron Find tool in order to evaluate the fitness function for the evolutionary parameter optimizer. Table 3.1 shows the slight improvement the model gained from this hyper-parameter optimization. In future work, the evolutionary parameter optimizer could be run on all the methods; however, we have only used it to optimize the hyper-parameters of our already best performing method.

3.3.3 Scoring Each Method

With the validation set of data we generated from the Neuron Find tool we were able to score the precision and recall of our methods. For the purposes of the data analysis pipeline, precision is weighted more then recall. This is because false positive neurons cause issues in the analysis of the entire sequence; where as false negatives simply miss some of the data from the sequence. We end up using an evaluation score of $2 * \text{Precision} + \text{Recall}$. Another possible scoring function is $\text{Precision}^2 * \text{Recall}$, this avoids the issue of having very low recall. However, we find that using $2*\text{Precision}+\text{Recall}$ gives us a good balance of false negatives and false positives, and thus we used this model for our analysis. Table 3.1 shows the runtime, precision, recall, score of each of the methods. We find that the Threshold, Aggregate, and Contour with the optimized parameters scored the best, and the data used later in the pipeline was extracted with this method.

Neuron Detection Model	Run Time	Recall	Precision	Score (2*P+R)
Threshold, Aggregate, and Contour + Evolutionary parameter Optimization	17s	μ= 58.6% σ= 18.2%	μ= 94.1% σ= 8.8%	247
Threshold, Aggregate, and Contour	17s	μ= 49.3% σ= 19.3%	μ= 96.0% σ= 6.4%	241
Gaussian Mixture Model	3s	μ= 43.8% σ= 16.7%	μ= 91.7% σ= 13.0%	227
Template Matching	0.2s	μ= 61.2% σ= 19.3%	μ= 75.4% σ= 6.4%	212
Threshold and Contour	0.05s	μ= 55.3% σ= 10.3%	μ= 77.3% σ= 19.5%	210
Genetic	30s	μ= 56.3% σ= 17.8%	μ= 68.5% σ= 15.1%	193

Table 3.1: Scores of Each Neuron Detection Method

Chapter 4

Neuronal Signal Extraction and Processing

4.1 Extraction

After the neurons on the images have been labeled, we extract the signals by taking our circles and summing all pixels inside of them at each frame within the sequence. On average a sample will contain 26 neurons from which signals will be extracted from. It takes about 8 seconds to extract the signals from each of the neurons in the image sequence. You can see an example of the fluorescence intensity per frame in Figure 4-1.

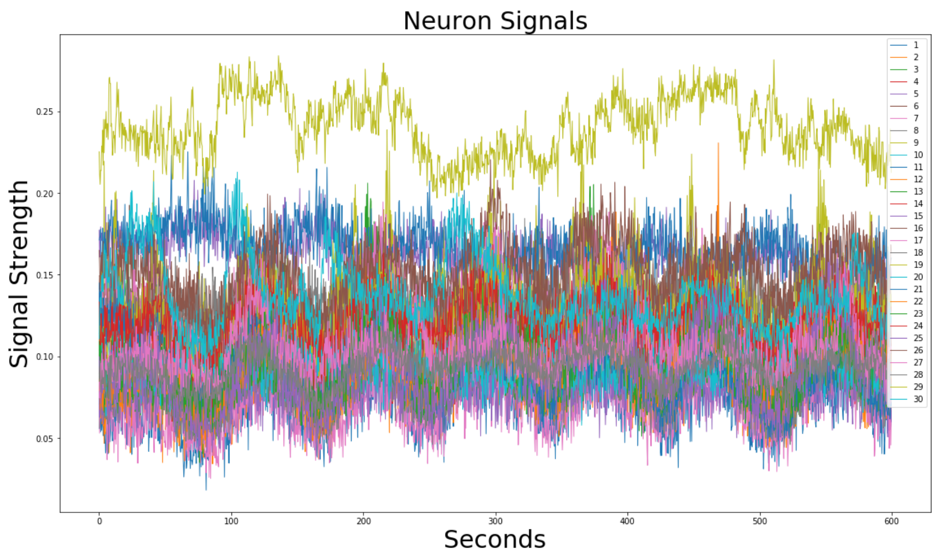

Figure 4-1: Signals Extracted from Two-Photon Calcium Image Sequence. Y axis is the total pixel intensity within an identified neurons area divide by the neurons surface area ($\frac{intensity}{pixels}$). This $\frac{intensity}{pixels}$ is measured for each of neurons for the 1600 frames: the X axis are these 1600 measurements on a seconds time scale.

These signals come in with a lot of noise. In order to isolate signals that come from neurons we want to make sure that the signal inside the circle is significantly brighter then the signal outside of the circle. To enforce this, we add an additional filter to assure that we do not have any false neurons in our analysis. For this filter we compare the neurons signal to its backgrounds average signal. We find this background average, by increasing the radius of the neurons circle by five and ignoring the original pixels. We compare this average to the mean of the neurons signal. If the difference is not more then 5 standard deviations of the signal we remove the neuron from the set. This filter usually will remove only one or two noisy neurons from the data.

4.2 Relative Fluorescence ($\frac{\Delta F}{F}$)

The extracted signal consists of a time series of measured fluorescence values. However, as pointed out by previous works[5], what is biologically relevant is the relative change in fluorescence. The equation for relative fluorescence is $\frac{\Delta F}{F_i} = \frac{F_i - F_0}{F_0}$, where F_i is the value of the fluorescence at time step i and F_0 is a baseline for the fluorescence [10]. However, in our pipeline we go one step further by making the baseline for the fluorescence be a low pass filter of the fluorescence. This reduces the risk that long-slow changes in the average fluorescence significantly disturb our measurements. In our team's prior works [5], we introduced this idea; however, only a Lowess filter was used for the low pass filter. We found that depending on the type of event we are trying to detect, different baseline functions filters would return better results.

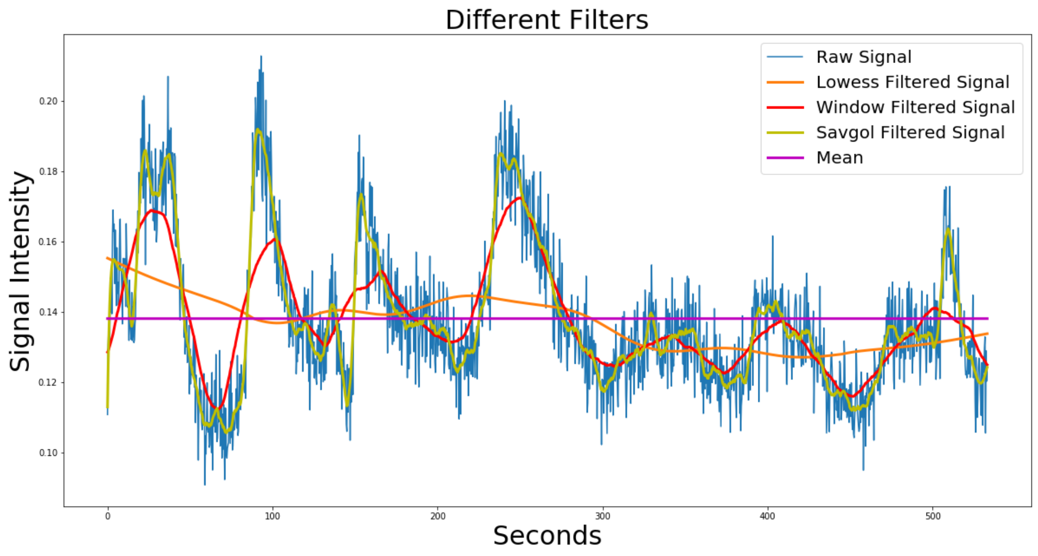

Figure 4-2: Single Signal with Different Filters. The original signal is in Blue. A mean filter is seen in magenta. A Lowess filter is seen in orange. Windowed average filter is seen in red. Savgol filter is seen in yellow

Figure 4-2 shows a single signal from a sequence, as well as a few filters applied to it. Here we show the mean, a Lowess filtered, a Savitzky-Golay (Savgol), and a windowed filtered signal. We use these different filters as the baseline for calculating $\frac{\Delta F}{F}$ depending on the type of event we are looking for. These filters act as different low pass filters. We found that subtracting these low-pass filters from the original signal when calculating $\frac{\Delta F}{F}$ these filters actually act as high-pass filters. For example, we find that the weak low pass filters, such as the windowed and Savgol filters, work as very strict high pass filters when calculating $\frac{\Delta F}{F}$. In contrast, strong low pass filters, such as mean and Lowess filters, act as relatively weak high pass filters when calculating $\frac{\Delta F}{F}$. The effects of this can be seen in Figure 4-3. The windowed and Savgol $\frac{\Delta F}{F}$ (red and orange plots) are acting as strict high pass filters (slow oscillations are removed), while the mean and Lowess $\frac{\Delta F}{F}$ (blue and green plots) are acting as weak high pass filters (allowing slow oscillations to persist).

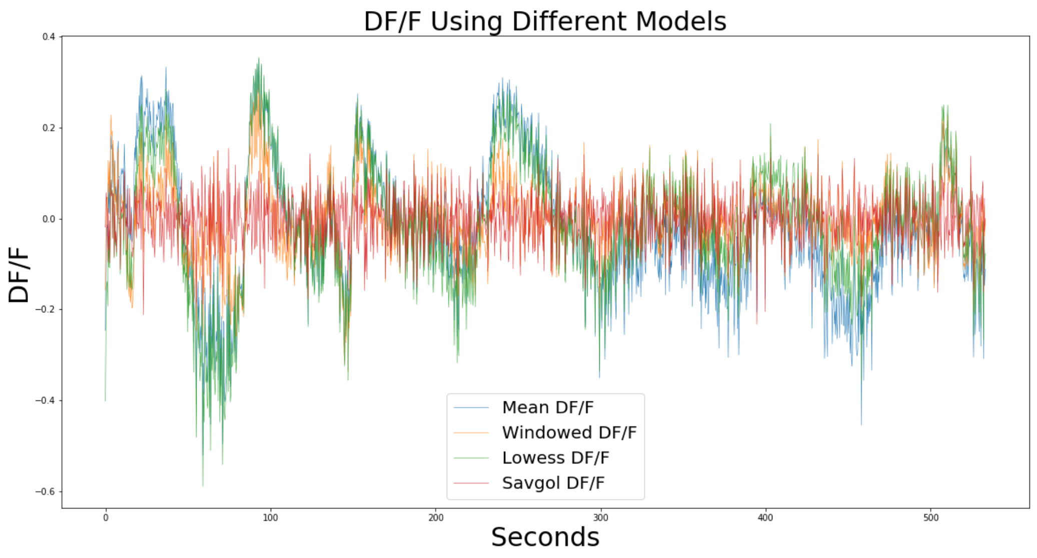

Figure 4-3: $\frac{\Delta F}{F}$ Calculated with Different Baseline Functions. Mean $\frac{\Delta F}{F}$ can be seen in blue. Lowess $\frac{\Delta F}{F}$ can be seen in green. Windowed average $\frac{\Delta F}{F}$ can be seen in orange. Savgol $\frac{\Delta F}{F}$ can be seen in red.

4.3 Denoising

Previous works have shown that auto-regressive models using a harmonic regression can be used to denoise two-photon calcium image sequence signals [10]. Unfortunately, calculating the auto-regressive model using a harmonic regression can be hard to compute. For our signals with 1600 data points, the model takes roughly 20 seconds to compute. However, as you can see in Figure 4-4. the Savitzky-Golay (Savgol) filter performs similar to the Auto Regressive model and takes 1.4 ms to compute. If we are processing 25 neurons in a sample the auto regressive denoising would take 8 min and 20 seconds, where as the Savgol filter performs very similar and would take 35 ms per sample. Additionally, the output of the auto-regressive harmonic regression may not represent the ground truth; thus, due to the high computation time, we do not use the harmonic auto-regression for denoising in our streamlined pipeline. For the purposes of our research, using the Savgol filter is a sufficient option for denoising our signals in our pipeline.

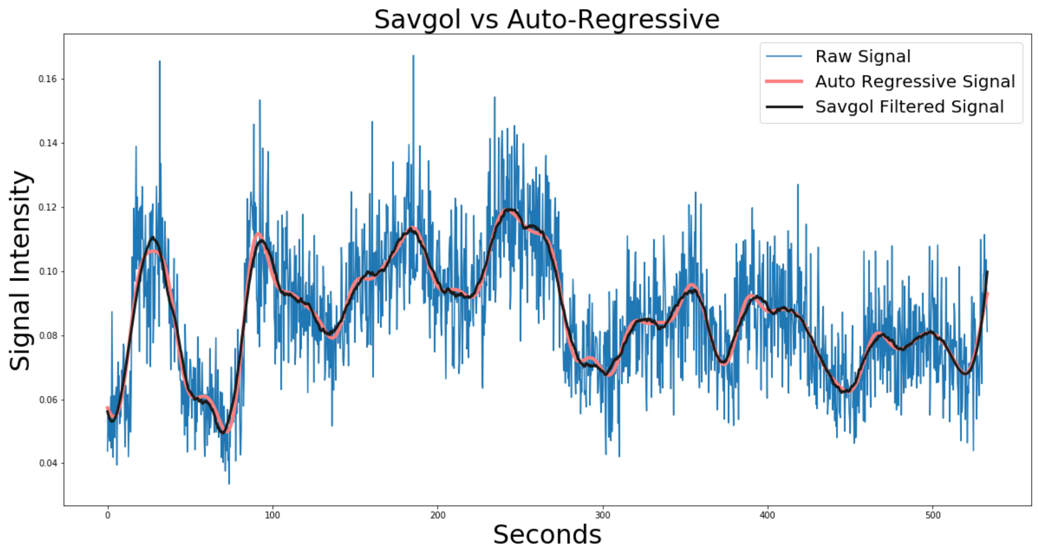

Figure 4-4: Harmonic Auto-regressive Denoising Vs Savgol Filter Denoising

Chapter 5

Neuronal Event Detection

Burst activity and slow oscillations are the two main types of activity in the neuronal cultures that we tried to detect. We focused on detecting correlations in activity that are associated with neuronal communication. In order to detect these correlations in activity, we must detect individual signal activity. We explore methods for detecting slow oscillations in neuron activity as well as burst activity In Figure 5-1 you can see example of such types of events.

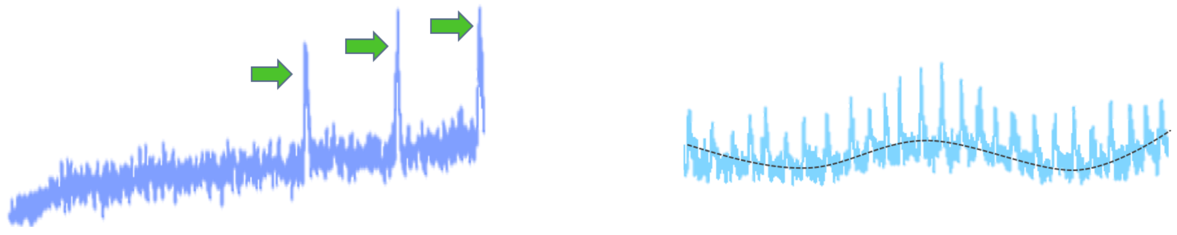

Figure 5-1: Example of burst activity on left, example of slow oscillations on right with (frequent burst activity as well).

The algorithms for detecting slow oscillations and burst activity are very similar. The pseudo-code can be seen below. For detecting slow oscillations activity we use the Lowess filter (strict low pass filter) as the low pass filter. In contrast, for detecting burst activity we use the windowed average filter (less strict low pass filter) as the low pass filter.

Algorithm 2 Pseudo-code for Event Detection

```
1: neurons = Neuron_Detection(Image Sequence)
2: signals = Signal_Extraction(neurons, Image Sequence)
3: activities = []
4: for signal in signals do
5:     base_signal = low_pass_filter(signal)
6:     $\frac{\Delta F}{F}$ = (signal - base_signal)/base_signal
7:     activity = threshold($\frac{\Delta F}{F}$)
8:     activities.append(activity)
```

5.1 Slow Oscillations Detection

To detect slow oscillations we start off with the $\frac{\Delta F}{F}$ calculated using the Lowess filter as the baseline. We then use the Savgol filter to denoise and find the underlying signal. The output of this process can be seen in Figure 5-2.

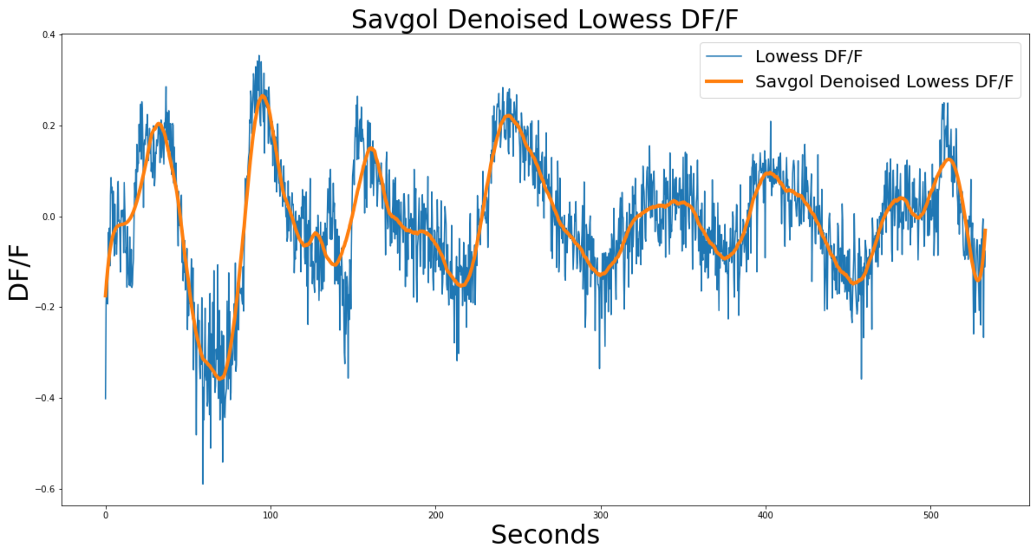

Figure 5-2: Savgol Denoised Lowess $\frac{\Delta F}{F}$. In orange you can see the estimated underlying slow oscillating signal extracted using the Savgol filter. The Lowess $\frac{\Delta F}{F}$ allowed slow oscillations to persist, then the Savgol filter removed the high frequency signals. The remaining signal (seen in orange) is the isolated slow oscillations we are looking for.

We then find when the signal is above a threshold for a long enough period of time (10 seconds). If the signal maintains high activity for long enough we consider the signal to be "active" in that period of time. In Figure 5-3 you can see the Savgol denoised Lowess $\frac{\Delta F}{F}$

going through the activity detection threshold.

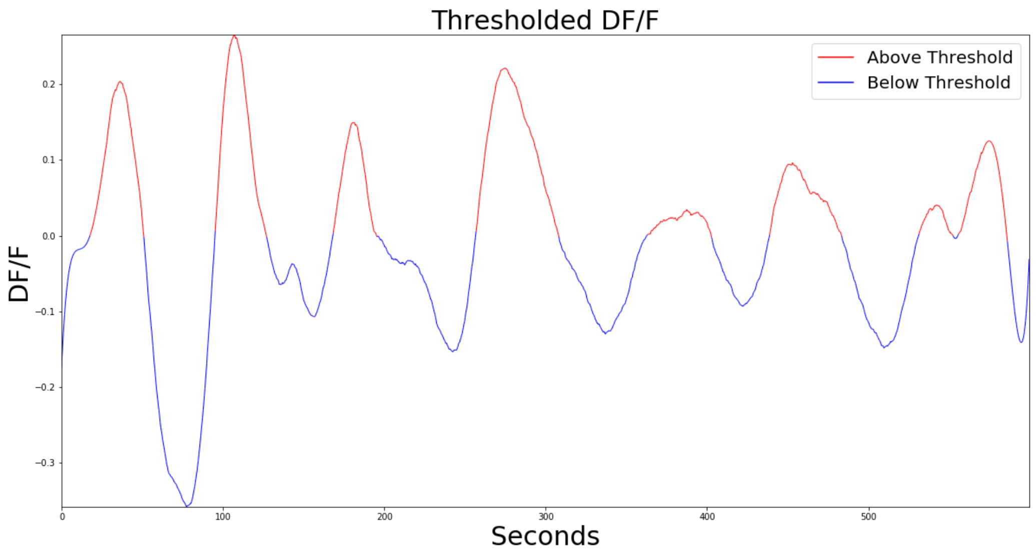

Figure 5-3: Thresholded Denoised Lowess $\frac{\Delta F}{F}$. Red marks the peaks of the slow oscillations and blue the troughs. By identifying the peaks, we can estimate the periodicity and compare that among cells.

The periods that survive the threshold are considered to be periods of peak slow oscillation activity. We then turn the activity into a binary signal which can be seen in Figure 5-4. A raster plot version of this signal can be seen in 5-5.

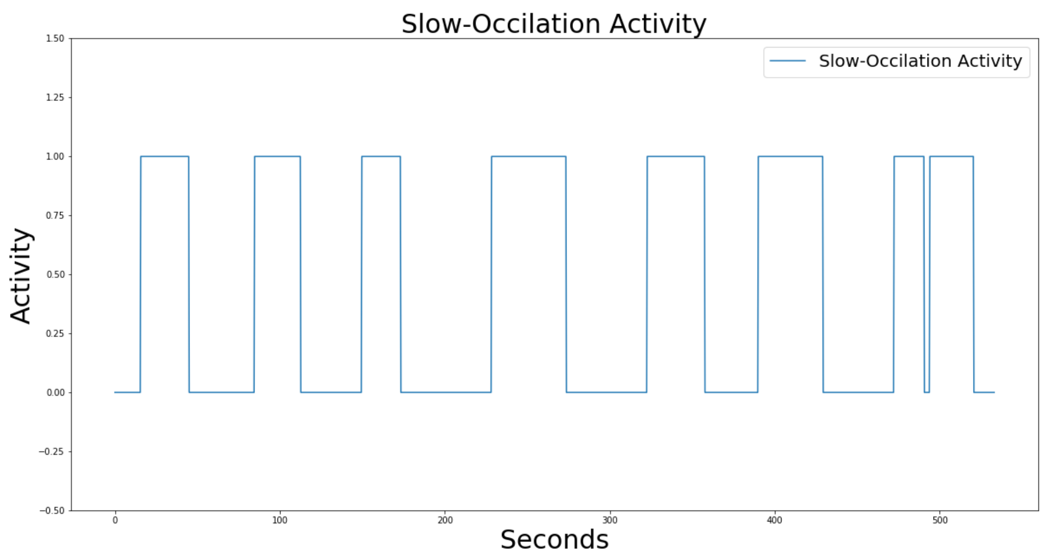

Figure 5-4: Slow Oscillation Activity Plot. When there is a peak detected in the slow oscillation activity the value is 1, otherwise the value is 0.

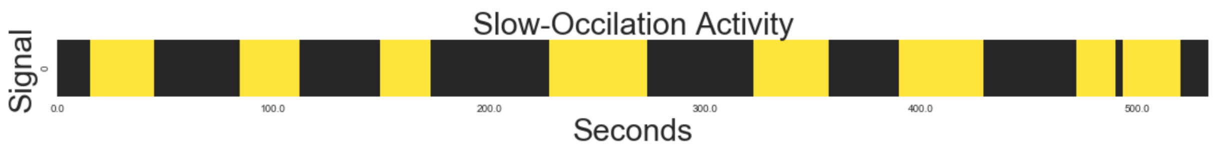

Figure 5-5: Raster Plot of Slow Oscillation Activity. The yellow sections represent the peaks of the slow oscillation activity. The black sections are trough periods in the slow oscillation activity.

Now the same process is performed on all the signals in a sequence. The result can be seen in figure 5-6.

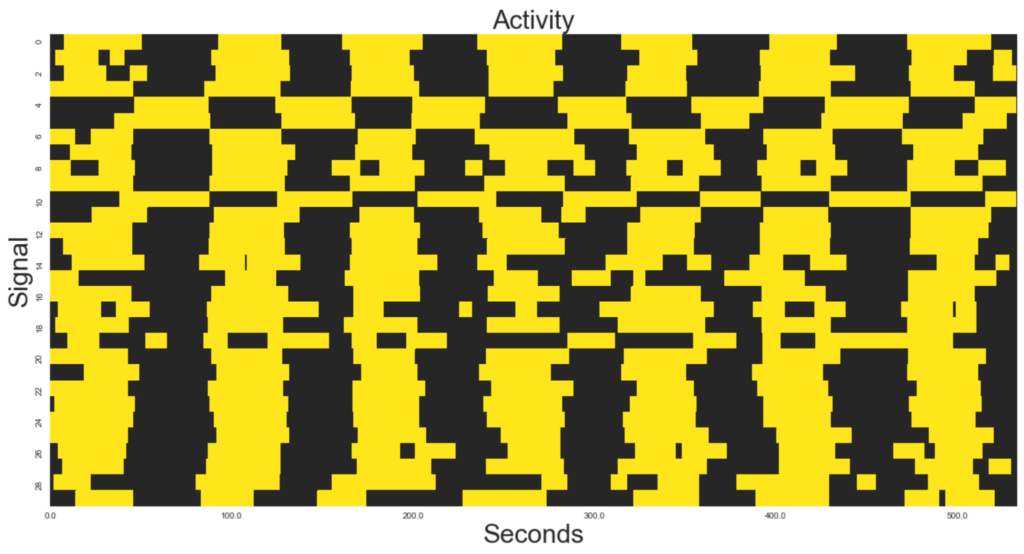

Figure 5-6: Raster Plot of Slow Oscillations Activities. The yellow sections are periods of slow oscillation activity. The Black sections are periods without slow oscillation activity. Each individual neurons raster plot is stacked on top of each other. One can see the correlation in activity.

5.2 Burst Activity Detection

To detect burst activity, the process is very similar to that of slow oscillations detection; however, now we want to isolate a higher frequency of signals. For slow oscillations detection we were isolating signals with periods in the range of 30-100 seconds, but for burst activity we are looking for signals with periods of 3-30 seconds. To do this we use a strict high-pass filter, so we use a weaker low-pass filter for the baseline of our calculations of $\frac{\Delta F}{F}$. For this we use the windowed $\frac{\Delta F}{F}$ method. Again we start off by denoising the windowed $\frac{\Delta F}{F}$ by using the Savgol filter. The output of this process can be seen in Figure 5-7.

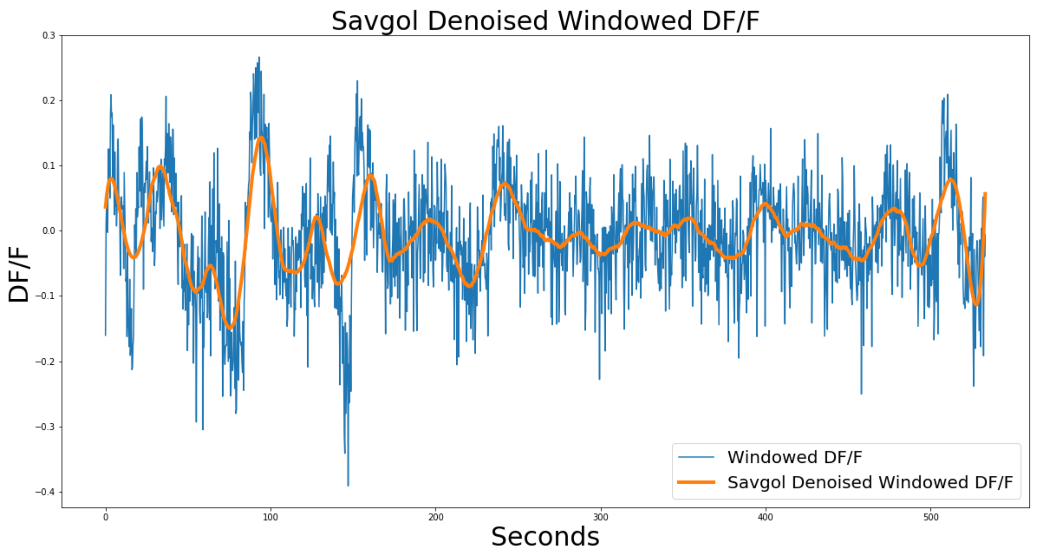

Figure 5-7: Savgol Denoised Windowed $\frac{\Delta F}{F}$. In orange you can see the estimated underlying burst activity signal extracted using the Savgol filter. The windowed $\frac{\Delta F}{F}$ removes the slow oscillations, then the Savgol filter removed very the high frequency signals. The remaining signal (seen in orange) is the isolated the burst activity we are looking for.

Again we want to find when the isolated signal is active for a long enough period of time. If the signal maintains high activity for long enough we consider the signal to be "active" in that period of time. Now that we are looking at the bursty frequencies of activity, the activity we label here is the bursty activity. In Figure 5-8 you can see the Savgol denoised Windowed $\frac{\Delta F}{F}$ going through the activity detection threshold.

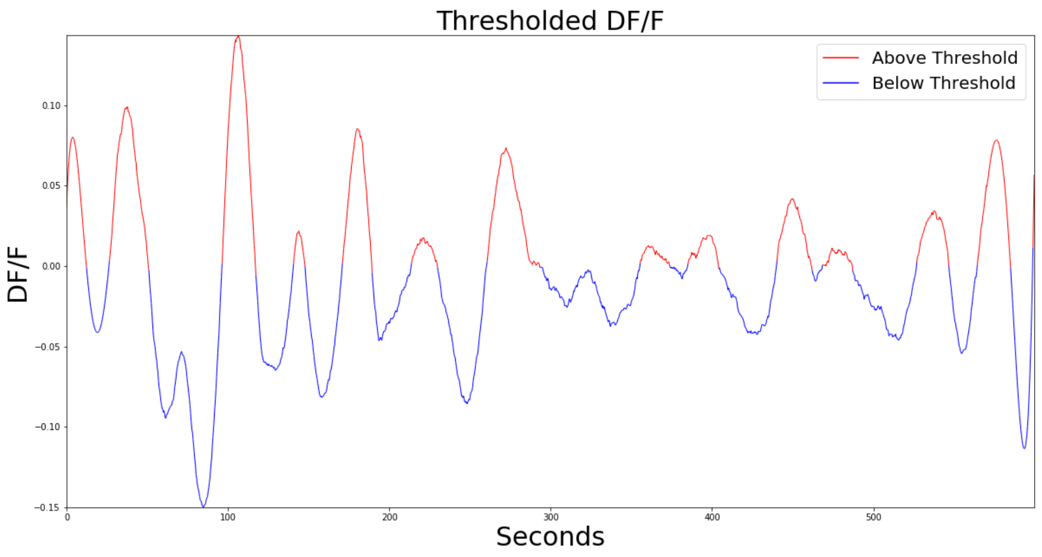

Figure 5-8: Thresholded Denoised Windowed $\frac{\Delta F}{F}$. Red is above the threshold and is considered to periods of burst-activity. Blue is below the threshold and is not considered to periods of burst activity.

The events that survive the threshold are considered to be bursty activity. We then turn the activity into a binary signal which can be seen in Figure 5-9. A raster plot version of this signal can be seen in 5-10.

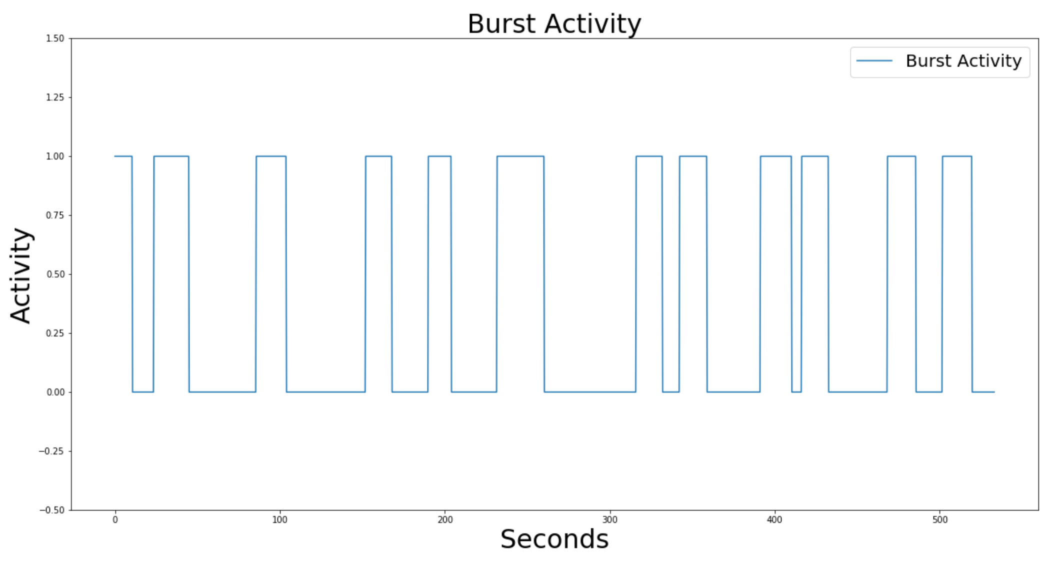

Figure 5-9: Burst Activity Plot. When there is burst activity the value is 1, otherwise the value is 0

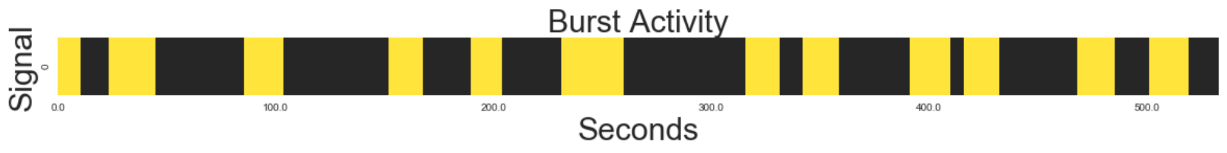

Figure 5-10: Raster Plot of Burst Activity. The yellow sections are periods of burst activity. The Black sections are periods without burst activity.

Now the same process is performed on all the signals in a sequence. The result can be seen in figure 5-11.

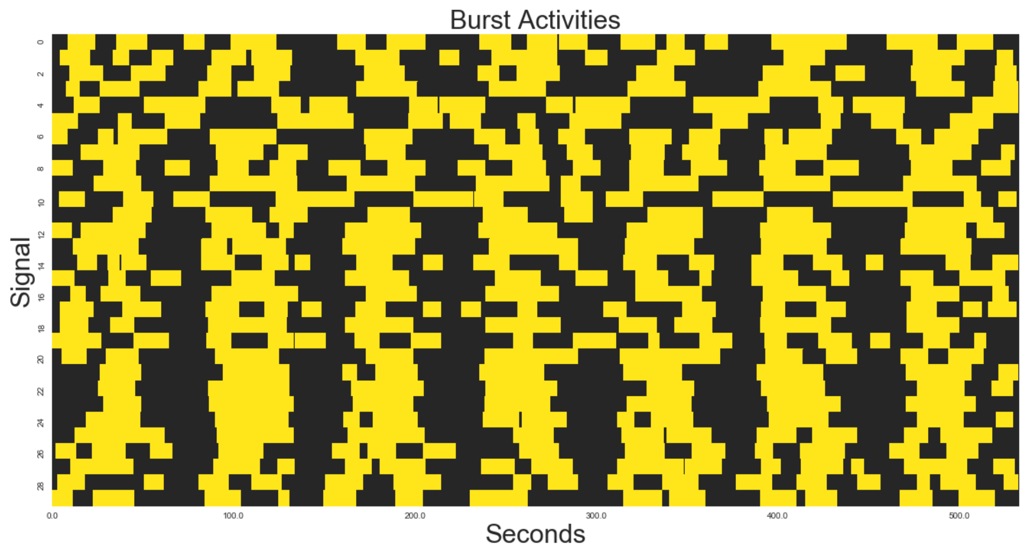

Figure 5-11: Raster Plot of Burst Activities. The yellow sections are periods of burst activity. The Black sections are periods without burst activity. Each individual neurons raster plot is stacked on-top of each other. One can see the correlation in activity.

5.2.1 Evaluation of Burst Detection

Similar to Neuron Detection, we did not have a gold standard in order to evaluate the performance of our algorithms, so we developed a graphical user interface. This graphical user interface can be seen in Figure 5-12.

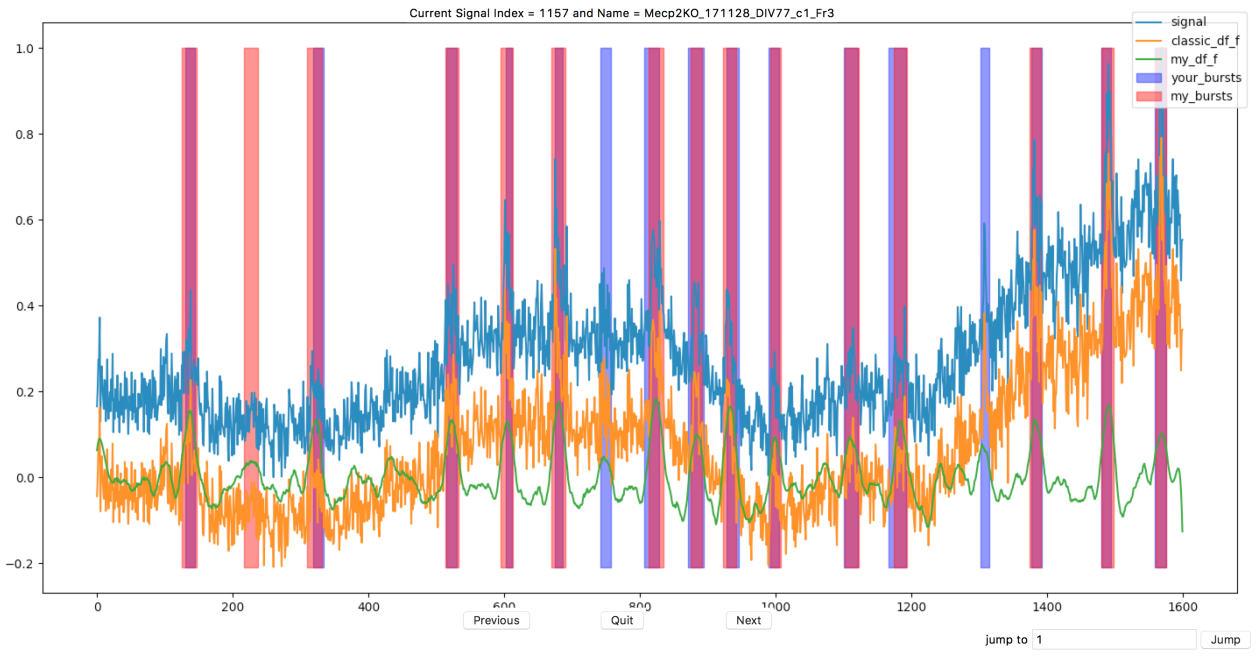

Figure 5-12: Burst Detect Interface. In green you can see the Savgol denoised windowed $\frac{\Delta F}{F}$. In blue plot is the original signal. The red sections are burst activity regions identified by the burst detection algorithm. The blue sections are burst activity regions identified by a user.

We had a neuroscientist use the interface we developed and inspect the signals. The neuroscientist first tried to label the burst events from the neurons in the recording on her own using the tool, then we showed her the burst events labeled by the algorithm. After using the tool, the neuroscientist noted that the false negatives were very rare as nearly all of the obvious signals were easily picked up by the burst activity detection algorithm. The neuroscientist also noted that the start and stop time points where marked more precisely then she could do by hand. Unfortunately, for our algorithm the rate of false positives was relatively high; however, this could be corrected in future work by optimizing the hyper parameters. Another issue was traces where there was very little change in the fluorescence, or a very high degree of noise, where the Savgol denoised windowed $\frac{\Delta F}{F}$ exaggerated the changes indicating events where there were none. The neuroscientist also noted that in many cases, she thought the burst-detection algorithm was likely more robust than her eye. More so, she believed that tuning the hyper parameters would likely improve the algorithms performance, as it did with the neuron detection. This will be the next step in further developing this analysis tool.

Chapter 6

Neuronal Activity Clustering

We know that correlations in activity are associated with functional neuronal communication. However, before we can quantify the correlations in activity we need to cluster similar signals to better understand the spontaneous network activity. Clusters of similar signals are thought to arise from network activity thus, we compare the correlation within clusters as well as correlation between clusters. In order to do these comparisons, we first needed to cluster the signals.

6.1 Signal Similarity Metric

To able to cluster signals we need to be able to calculate the similarity between two signals. The first obvious method is to use the cosine similarity (Equation 6.1) between the two signals. This has been used in previous works [5]. We find that signals that are out of phase get a very low cosine similarity value. However, this is an acceptable metric, given that signals that are out of phase are likly not to be in the same network cluster.

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{||A|| * ||B||} \tag{6.1}$$

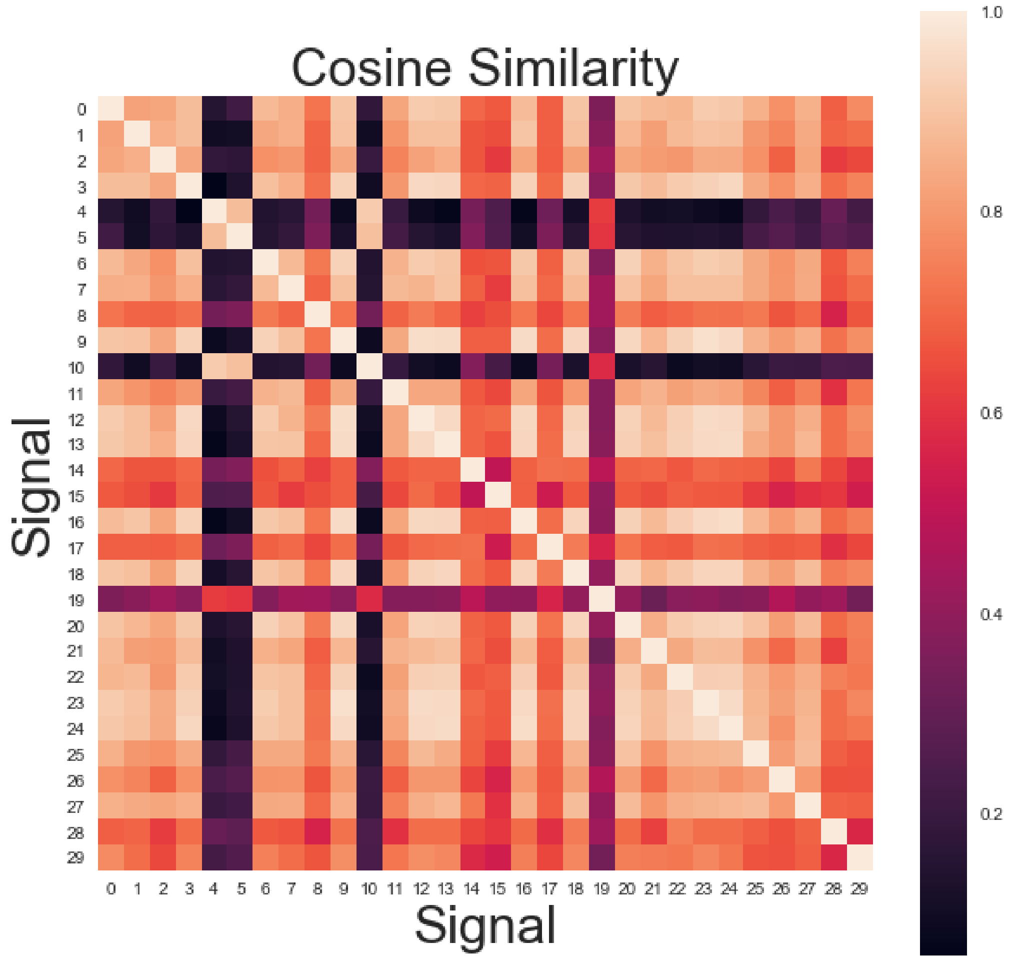

Figure 6-1: Cosine Similarity Matrix. The i, j term of this matrix is the Cosine Similarity(signal$_i$, signal$_j$). Darker areas show where the signals are not similar. Brighter areas show where the signals are similar. Note that this matrix is symmetric since Cosine Similarity is a commutative equation.

6.2 Neuronal Activity Clustering Algorithm

We explored a handful of clustering algorithms including K-means, spectral clustering, and Density Based Spacial Clustering Applications with Noise (DBSCAN) clustering. Unfortunately, most clustering algorithms require the number of clusters to be preset, where in this case we do not know the number of groups of signals beforehand. DBSCAN clustering offered the ability to not only have no preset number of clusters but also to identify unclustered signals. In Figure 6-2, one can see the first row pertains to an unclustered signal, followed by two clusters. The two clusters are clearly out of phase; however, have similar periods.

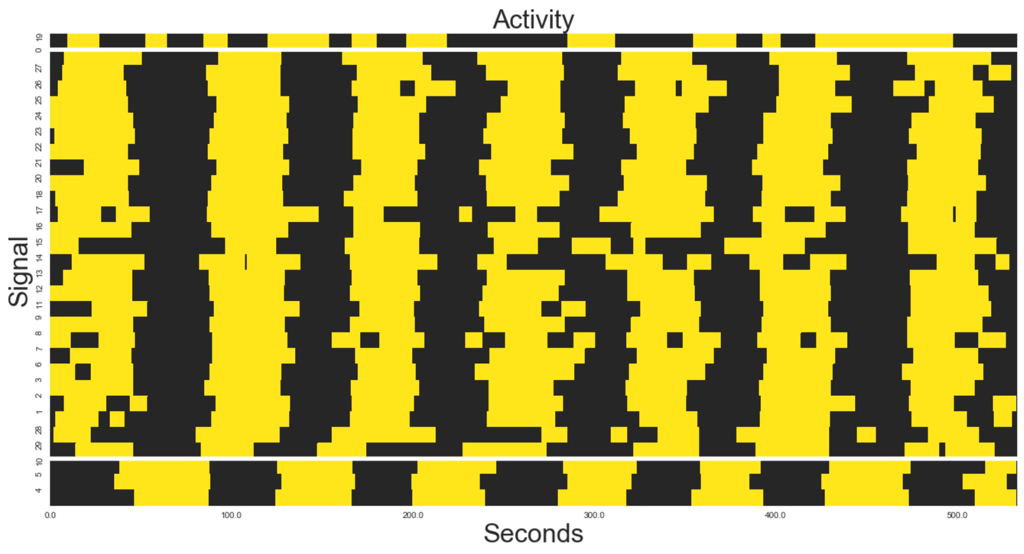

Figure 6-2: DBSCAN Clustered Raster Plot. The first row pertains to an unclustered signal, followed by two clusters. The two clusters are clearly out of phase; however, have similar periods.

In Figure 6-3 we can see the Cosine Similarity matrix with 3 discrete sections. There is the unclustered section, then the cluster 1 section, then the cluster 2 section. DBSCAN was able to successfully cluster the cells based on correlated activity.

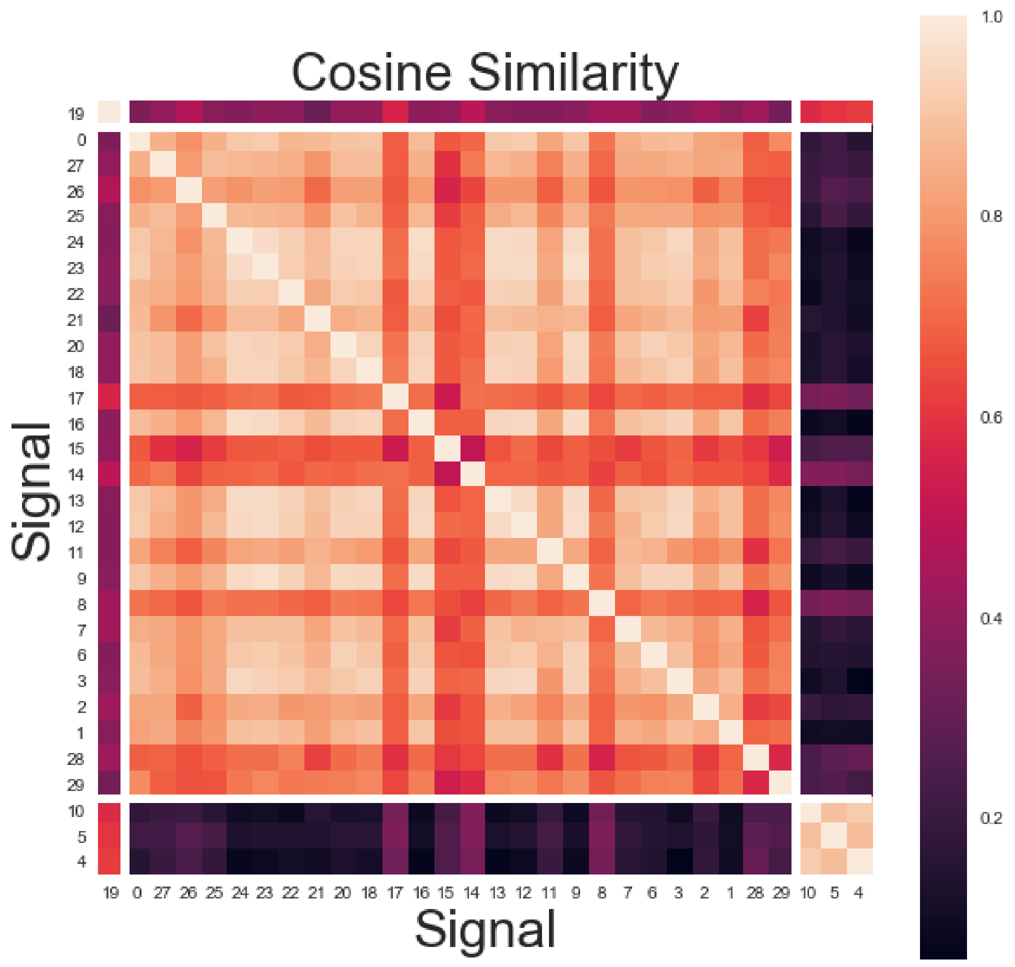

Figure 6-3: Clustered Cosine Similarity Plot. There are 3 discrete sections in this plot. There is the unclustered section with one signal. Then there is cluster 1 which has most of the signals. Then there is cluster 2 which contains signals from three brain cells.

In Figures 6-4 and Figure 6-5 we plotted the denoised $\frac{\Delta F}{F}$ of signals before and after DBSCAN clustering. DBSCAN was able to properly partition the neuronal signals.

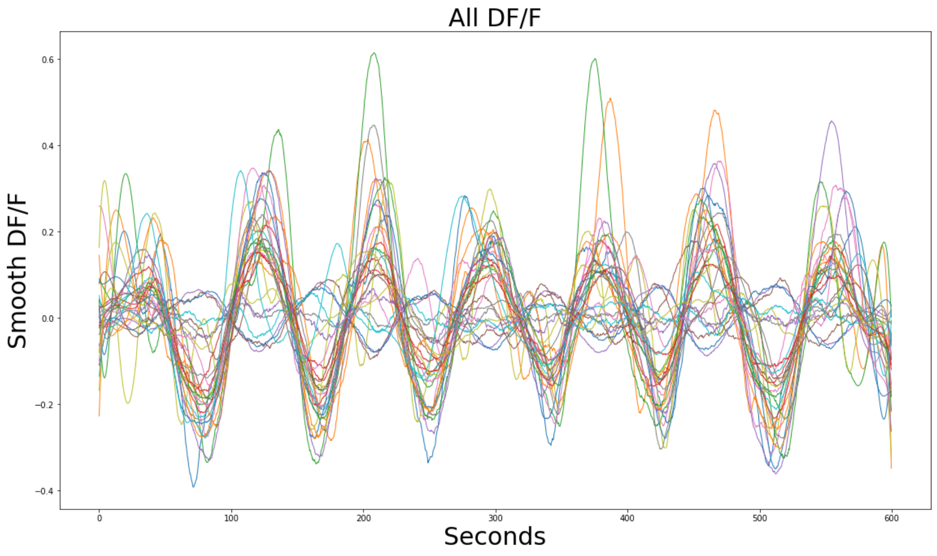

Figure 6-4: $\frac{\Delta F}{F}$ of Signals Before Clustering.

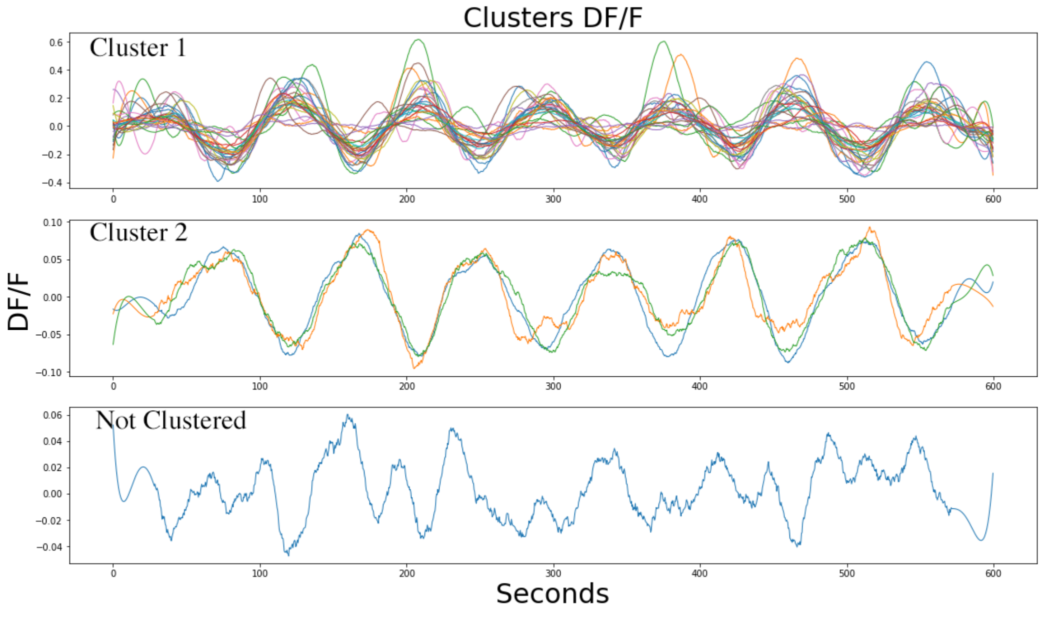

Figure 6-5: $\frac{\Delta F}{F}$ of Signals After Clustering. The top plot pertains to the cluster 1, which holds most of the signals. The second plot pertains to the cluster 2, which holds 3 of the signals. Finally the bottom plot pertains to an unclustered signal, which we can see is not similar to any of the other clusters.

We can do another sanity check to see where the locations of the clustered neurons come from. In figure 6-6 you can see the location of each neuron and what cluster it is in. As shown in past works [5] there is no strong correlation of location in image sequence. This is confirmed with our data.

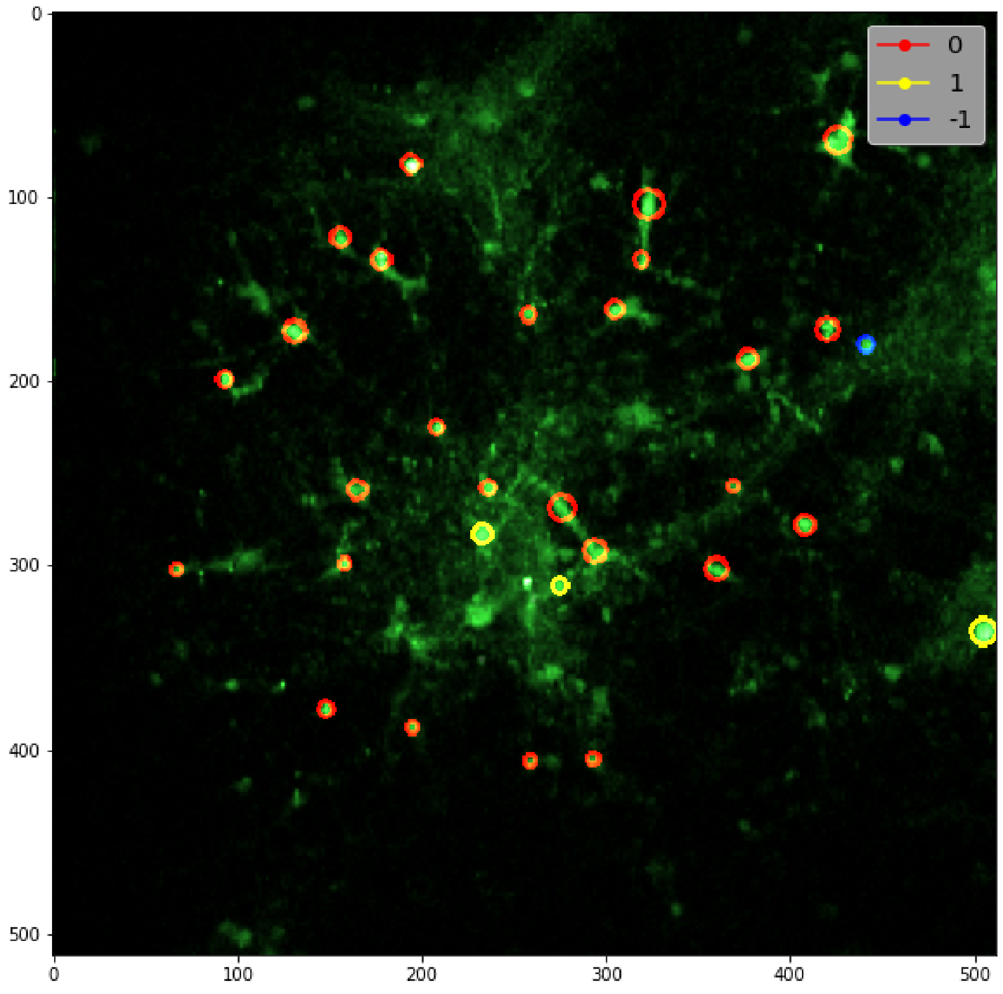

Figure 6-6: Location of Clusters in Two-Photon Calcium Image. Here we can see that the clustering of the yellow group appears independent of spatial location.

Chapter 7

Feature Extraction and Analysis

Now that we find neurons in an image, extract the signals, and find clusters of similar signals. We try to extract some features. This part of the pipeline relies on feature engineering. For the purposes of the dataset we are trying to analyze, we were looking for communication between neurons. Because of this we focus on extracting features such as the silhouette score of clusters, the number of clusters, and the percent of neurons that are unclustered.

7.1 Feature Extraction

To extract features we run the entire pipeline on each of the image sequences to identify both burst activity and slow oscillations and to cluster cells by both types of activity. Then with clustered groups we calculate the silhouette score, number of clusters, size of clusters, proportion unclustered, proportion in largest cluster, as well other features useful for our evaluations.

7.2 Analyzing

Table 7.1 shows the mean and standard deviation for all of the features that were extracted across each of the mouse genotypes. No significant differences were observed between the genotypes with the current data. One of the issues is that we have not controlled for is the days in vitro for each of these mice cultures. Unfortunately, our team does not have currently enough data to do such a study at the moment; however, this is important feature to control for in future experiments since the amount of spontaneous activity increases over development[8]. With more data we could train a logistical regression, and find features that are predictive of the genotypes. Unfortunately, we do not have enough data to detect any of the differences between the groups. Furthermore, the amount of noise in our data is high. This high level of noise may be masking any underlying differences.

Type	Number of Samples	Number of Neurons	Slow-Osccilations				Burst-Activity			
			% Unclustered	% In Largest Cluster	Number of Clusters	Silhouette Score	% Unclustered	% In Largest Cluster	Number of Clusters	Silhouette Score
KO	25	μ= 24 σ= 12	μ= 32% σ= 23%	μ= 47% σ= 24%	μ= 2.0 σ= 1.4	μ= 0.34 σ= 0.06	μ= 38% σ= 23%	μ= 39% σ= 24%	μ= 2.24 σ= 1.4	μ= 0.13 σ= 0.06
WT	23	μ= 30 σ= 17	μ= 36% σ= 23%	μ= 40% σ= 24%	μ= 2.4 σ= 1.4	μ= 0.33 σ= 0.06	μ= 50% σ= 23%	μ= 25% σ= 24%	μ= 2.39 σ= 1.4	μ= 0.14 σ= 0.06
HET	14	μ= 24 σ= 7	μ= 25% σ= 23%	μ= 53% σ= 24%	μ= 2.0 σ= 1.4	μ= 0.42 σ= 0.06	μ= 49% σ= 23%	μ= 36% σ= 24%	μ= 2.64 σ= 1.4	μ= 0.09 σ= 0.06

Table 7.1: Aggregate Features Extracted from Two-Photon Calcium Image Sequences. Mecp2-deficient mice that are either hemizygous for a deletion in Mecp2 (knock-out mice; KO) or heterozygous for the deletion (HET) are compared with mice with a wild-type (WT) copy (or copies) of Mecp2.

Chapter 8

Conclusion and Future Work

In this paper we have proposed a pipeline for analyzing two-photon calcium image sequences. This pipeline can significantly aid researchers in the analysis of these two-photon calcium images. Our ML pipeline reduces the time required to analyze two-photon calcium images from over 10 minutes to about 30 seconds per sample. In addition, signal extraction and analysis has not been standardized. The pipeline we propose solves these issues. The neuron detection algorithm in our pipeline, performed well in evaluation with a gold standard developed by our team. The signal extraction and event detection, in our pipeline has received positive feed by from a neuroscientist. The neuroscientist has also praised the clustering done by our clustering algorithm. Unfortunately, our pipeline did not find any significant differences between the neuronal communication between the different genotypes. However, this issue can be attributed to a relatively small dataset, and will be expanded on in future research.

8.1 Future Work

8.1.1 Supervised Neuron Detection

A labeled training set has been created for stacked two-photon images, and the results in the Neurofind competition (which evaluated performance in finding neurons in these types of images) have been good. We can create a training set for single sliced two-photon calcium images to improve our neuron detection performances. 3D convolutional neural networks achieved the highest performance in the Neurofind competition. We can collect a training set to train and evaluate a 3D CNN for neuron detection in two-photon calcium imaging sequences.

8.1.2 Controlling for Days-in-Vitro (DIV)

Our study was not able to control for the number of days the culture has been growing (days in vitro), since we had a very limited data set. However, it is known that DIV has a positive correlation with neuronal activity in the culture. A future study with more data should try to control for this.

8.1.3 Future Improvements to Genetic Neuron Detection

Currently the performance and the runtime of the genetic neuron detection model hinder it from being a feasible neuron detection method. However, we believe that this model could yield better results. First of all, the genetic model simply uses the mean of the image sequence rather then working through each frame. One improvement would be to evaluate the fitness function, through the image sequence. More research into an optimal fitness function should be done in the future work in order to promote better detection.

8.1.4 Event Detection Improvements and Future Research

One method for improving the performance of our event detection algorithms could be the use of an extended Kalman filter. This type of model has been used to detect events in signals such as earthquake detection in seismic data. Using this probabilistic approach for event detection could lead to improved performance. Using the burst-activity detection tool one could generate a labeled training set for training an extended Kalman filter. This supervised approach would likely receive a much higher accuracy.

An interesting observation that the neurologist who went through our burst-activity detection interface noticed, was that during slow-oscillations activity our algorithm revealed that there was a large increase in burst-activity. If this is true, which more controlled studies should observe, it would support the hypothesis that the slow-oscillations may be up-and-down states that help the neurons in the same network synchronize activity.

8.2 Clinical Implications

The pipeline proposed in this paper will help neuroscientists analyze a much larger dataset to create a developmental profile of the network activity in multiple situations. This includes the Mecp2-deficient and wild-type cortical cultures, which this paper has observed. This pipeline will help us understand neuronal networks better. With this tool we can see how neuronal networks develop and test the underlying mechanisms by recording the activity after treating the cultures with novel drugs that modulate network development. Moving forward, our pipeline can be used is to identify new therapies and treatments for conditions such as for Rett syndrome and autism.

Bibliography

[1] Van Naarden Braun K et al Christensen DL, Baio J. Prevalence and characteristics of autism spectrum disorder among children aged 8 years-autism and developmental disabilities monitoring network, 11 sites, united states, 2012, April, 2016. 1;65(3):1âĂŞ23.

[2] Soumya Ghosh and Erik B. Sudderth. Nonparametric learning for layered segmentation of natural images, 2012. http://soumyaghosh.com/publications/papers/GhoshSudderth12CVPR.pdf.

[3] imreg_dft Deveolopement Team. imreg_dft documentation, May, 2018. https://pythonhosted.org/imreg_dft/quickstart.html#quickstart.

[4] National Institute of Child Health and Human Development. Autism spectrum disorder, October, 2017. https://medlineplus.gov/autismspectrumdisorder.html.

[5] Jarzebowski P. Network development and its dysfunction in a genetic model of autism, July, 2017. (Master's thesis) University of Cambridge.

[6] Schroder S. Dipoppa M. Rossi L. F. Carandini M. Harris K. D. Pachitatiu M., Stringer C. Beyond 10,000 neurons with standard two-photon microscopy, 2016. https://doi.org/10.1101/061507.

[7] Roberto Asin-Acha Quico Spaen, Dorit S. Hochbaum. A novel combinatorial approach for cell identification in calcium-imaging, October, 2017. https://arxiv.org/abs/1703.01999.

[8] Kitzbichler MG Paulsen O Bullmore ET Schroeter MS, Charlesworth P. Emergence of rich-club topology and coordinated dynamics in development of hippocampal functional networks in vitro, April, 2015. 8;35(14):5459-5470.

[9] OpenCV Developement Team. Open cv api references, May, 2018. https://docs.opencv.org/2.4/modules/refman.html.

[10] Mriganka Sur Emery N. Brown Wasim Q. Malik1, James Schummers. Denoising two-photon calcium imaging data, June, 2011. http://web.mit.edu/surlab/publications/2011_MalikSchummersSurBrown.pdf.