Using Machine Learning for Analysis of Neuronal Network Activity

by

Srilaya Bhavaraju

S.B., Massachusetts Institute of Technology (2019)

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

September 2020

Author .....
Department of Electrical Engineering and Computer Science
July 2, 2020

Certified by .....
Una-May O'Reilly
Principal Research Scientist
Thesis Supervisor

Certified by .....
Erik Hemberg
Research Scientist
Thesis Supervisor

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

Using Machine Learning for Analysis of Neuronal Network Activity

by

Srilaya Bhavaraju

Submitted to the Department of Electrical Engineering and Computer Science
on July 2, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Analyzing neuronal activity in developing neuronal networks can improve our understanding of neuronal dysfunctions underlying conditions such as Rett syndrome. Two-photon calcium imaging is used to capture neuronal network activity over time. This method produces large sets of images that are typically manually analyzed by skilled neuroscientists. Because this process is both time-consuming and subject to error, discovery of therapies that ameliorate network dysfunction may be slowed. We improve an existing, semi-autonomous machine learning pipeline for two-photon calcium imaging sequence analysis. We introduce to the pipeline neuron detection methods using supervised learning models, heuristic filtering of pixels for signal extraction, and event detection using deconvolution. With these methods, we improve neuron detection performance, alter signal-to-noise ratio of extracted calcium signals, and allow for integration of methods that infer action potential firing underlying these signals.

Thesis Supervisor: Una-May O'Reilly
Title: Principal Research Scientist

Thesis Supervisor: Erik Hemberg
Title: Research Scientist

Acknowledgments

I would like to thank Erik Hemberg Ph.D. and Una-May O'Reilly Ph.D. for the opportunity to work on this project. Their strong guidance and mentorship have helped me gain confidence and become a better student and researcher. I am grateful for the ALFA group's positive and supportive work environment and am thankful for the time I have spent with them.

I would also like to thank Susanna Mierau, DPhil, M.D., and Rachael Feord Ph.D. for their consistent guidance and expert insights on the techniques and results discussed in this paper as well as the collection of the data used in this project. I am very thankful for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Analyzing neuronal activity in developing neuronal networks can improve our understanding of neuronal dysfunctions underlying Rett syndrome [1, 2]. Mutations in the *MECP2* gene have been found to cause these conditions [3]. Differences in network activity between Mecp2-deficient and wild-type cultures may help in discovering novel therapies that ameliorate network dysfunction [1]. Two-photon calcium imaging allows for the investigation neuronal network activity. This method produces large sets of images that are typically manually analyzed by skilled neuroscientists. Because the process is both time-consuming and subject to error, discovery of novel therapies that ameliorate network dysfunction may be slowed. Therefore, there is a need for reliable, automated methods.

Previous work by our group in [4] produced a semi-autonomous machine learning pipeline for accelerating the analysis of two-photon calcium imaging sequences ("imaging sequences"). The pipeline includes methods for automatically identifying neurons, signal extraction and processing, event detection, feature extraction, and analysis. This work produced promising results and provides a base for the overall pipeline and this work. However, the neuron detection, signal extraction, and event detection steps needed improvement before the pipeline could be used. Here, we improve this pipeline and make it usable for our group. We incorporate new neuron detection, signal extraction, and event detection methods to better analyze the large data sets. The signals outputted from this pipeline can be used for different analyses,

including the inference of functional connectivity in networks.

This chapter provides brief descriptions on two-photon calcium imaging, the effect of Mecp2 deficiency on neuronal network function, and the previous pipeline.

1.1 Two-Photon Calcium Imaging

Fluorescence microscopy takes advantage of "fluorescent objects that can be selectively excited and visualized" [5, p.823] [6]. "Two-photon excitation (2PE) laser scanning microscopy allows for high-resolution and high-sensitivity fluorescence microscopy in intact neural tissue" [5, p.823] that has "synthetic fluorophores that allow for labeling of cellular structures" or "aspects of cellular function", such as calcium ion concentration [5, p.823][7]. 2PE provides several advantages over one-photon techniques [8]. Due to its excitation wavelengths, 2PE microscopy is able to penetrate tissue better, limit excitation to a small focal volume, and as a result, produce better signals from detected fluorescent photons [5, p.824][8].

Calcium ions "control key functions in all types of neurons" [9, p.862]. Calcium indicators can be used "for monitoring the dynamics of cellular calcium signaling", and imaging calcium in neurons allows for investigation of calcium signals [9, p.862]. Scanning specific laser beams over a specimen generates images of "fluorescence values acquired for each pixel" [9, p.871][10]. Images collected over time in for cortical cultures act as a time-lapse sequence depicting activity in neurons (described in Chapter 2).

1.2 Effect of Mecp2 Deficiency on Neuronal Networks

Mecp2 is a protein that regulates whether genes are turned on or off in brain cells and plays an important role in brain development[11]. Mutations in the *MECP2* gene have been found to lead to Rett syndrome [3].

The experiments discussed in this work examine the effect of Mecp2 deficiency

in a mouse model of Rett syndrome. In mice, males with the mutant *Mecp2* gene (where exons 3 and 4 are deleted) live for up to 60 days [12]. Female mice with a single healthy copy of *Mecp2* survive but start to show symptoms after a few months [12].

In Mecp2 deficient networks, inhibitory PV cells[1] increase the number of boutons projecting onto the soma of excitatory cells in visual cortex, causing reduced spontaneous firing *in vivo* [2, p.6][13]. Furthermore, "reduced spontaneous firing in pyramidal cells is also seen in cortical slices" [2, p.6][14] "and in hippocampal autaptic neurons cultured *in vitro*" [2, p.6] [15]. It was found that *Mecp2*-deficiency alters the "maturation of NMDA receptors in two major classes of cortical neurons" [2, p.7][1].

1.3 Prior Work

In this section, we describe our previous research in our group [4] that helped motivate the research questions of this work. This work produced a pipeline which consists of neuron detection, signal extraction and processing, neuronal event detection, and feature extraction and analysis. An overview of this pipeline can be seen in Figure 1-1.

In the neuron detection step, neurons are identified in the two-photon calcium imaging recordings. Several different unsupervised learning models were implemented and evaluated. These methods include (1) Threshold, Aggregate, and Contour with Evolutionary Parameter Optimization, (2) Threshold, Aggregate, and Contour, (3) Threshold and Contour, (4) Template Matching, (5) Gaussian Threshold, Aggregate, and Contour, and (5) Genetic Neuron Detection. The Threshold, Aggregate, and Contour with Evolutionary Parameter Optimization obtained the highest score, with a mean precision of 94.1% and a mean recall of 58.6%. The Template Matching method achieved the highest mean recall of 61.2% and the Threshold, Aggregate, and Contour method achieved the highest mean precision of 96.0%.

In the signal extraction and processing step, signal intensity from the regions

[1] parvalbuminpositive interneurons [2]

detected as neurons are extracted over the sequence of images to produce signals. The signals are then denoised using a Savitzky-Golay filter.

The neuronal event detection step identifies both burst activity and slow oscillations in the signals. Signals of relative fluorescence were used, where the baseline is a low pass filter of fluorescence. For detecting slow oscillation activity, a Lowess filter is used as a strict low pass filter. For detecting burst activity, a windowed average filter is used as a less strict low pass filter. An appropriate threshold is applied to the resultant signals to identify times of neuronal events. Upon inspection by a neuroscientist, false negatives for burst activity detection were found to be very rare. Additionally, it was noted that almost all of the obvious burst activity events were identified by this method. However, it was found that the rate of false positives was relatively high.

Next, clusters of similar signals are created to quantify correlations in activity related to neuronal communication. Cosine similarity was used to measure similarity between two signals. Signals are then clustered using Density Based Spacial Clustering Applications with Noise (DBSCAN) clustering.

Finally, features are extracted and analyzed from these clusters of similar signals. The silhoutte score, number of clusters, size of clusters, proportion unclustered, proportion in largest cluster along with other features are extracted for each image sequence. The values obtained for the features are analyzed across the different mouse genotypes (described in Chapter 2). No significant differences were observed between the different genotypes.

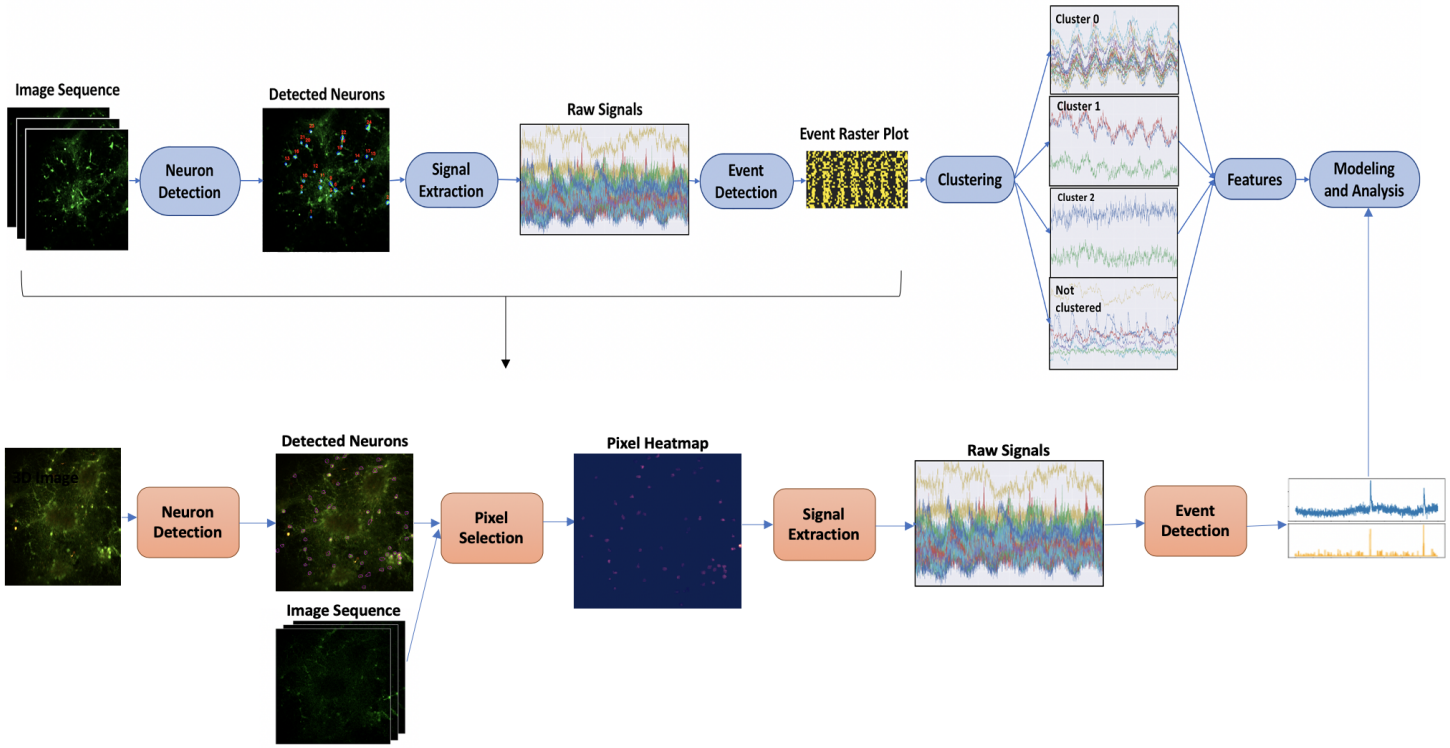

Figure 1-1: **Overview of Pipeline.** Previous Pipeline (top) and New Pipeline (bottom). The new pipeline improves the first few steps in the previous pipeline. First, an image sequence and 3D-image are captured from the mice brain cultures using two-photon calcium imaging. In the previous pipeline, the image sequence is used to identify regions of neuron cell bodies. In the new pipeline, the 3D image is used to identify neuron cell bodies. Next, signals are extracted from each of the neurons by finding the signal intensity at the identified locations in each frame. The previous pipeline uses all pixels in the identified regions to produce signal values. The new pipeline finds signal values from select pixels in each identified region. Next, specific events that correspond to neuronal activity in signals are detected. The previous pipeline uses a filter-based approach for event detection whereas the new pipeline leverages a deconvolution method. Afterwards, the previous pipeline clusters signals with similar event timings, extracts features from these clusters, and analyzes feature values for differences between genotypes. The new pipeline makes extracted signals available for various downstream analyses, including the clustering and feature extraction shown in the previous pipeline.

1.4 Research Questions

In this work, we both improve steps of the previous pipeline by leveraging new methods. Given a two-photon calcium imaging sequence, the existing pipeline identifies neurons in the sequence, extracts signals from the identified regions, detects neuronal events in the signals, clusters similar signals, and extracts and analyzes features from the clusters. Our research questions focus on how the existing pipeline can be improved. Specifically, we focus on the neuron detection, signal extraction, and event detection steps of the pipeline and investigate methods for each.

Research Question 1. In [19], it was noted that neuron detection methods were constrained to unsupervised learning methods as a training set of labeled neurons was not available. However, we now have annotated neurons for numerous image sequences (described in Chapter 2). Therefore, we investigate whether we can improve neuron detection using supervised learning. Specifically, we investigate the performance of a convolutional neural network.

Research Question 2. [19] also notes that the denoising method sometimes exaggerated noise in signals with very little change in fluorescence, causing false events. We aim to extract signals that are less noisy and present clearer burst activity events. We investigate whether using different heuristic filters to extract signals achieves higher signal-to-noise ratios.

Research Question 3. [19] claims that the rate of false positives was relatively high for burst activity detection. We investigate an alternate burst activity detection method that uses deconvolution and investigate whether it achieves better performance.

Contributions. In this work, we focus on three parts of the previous pipeline. We incorporate three-dimensional (3D) images into the neuron detection step and evaluate four different neuron detection methods. These methods include previous methods that are modified for the 3D images, a supervised learning method using Mask R-CNN [16], and a combination of these methods. In the signal extraction step, we incorporate pixel selection methods for signal extraction using different heuristic filters and

similarity measures. We also investigate an alternative event detection method based that uses the OASIS algorithm [17] in the event detection step. We evaluate both neuron detection and event detection against an annotated gold-standard data set.

1.5 Thesis Structure

The remainder of this is organized as follows. Chapter 2 provides an overview of the Mecp2 data set, including data collection and annotation. In Chapter 3, we describe and evaluate the methods used for neuron detection. Chapter 4 presents the methods used for signal extraction from the detected neurons. In Chapter 5, we describe and evaluate a burst activity detection method using OASIS [17]. Finally, Chapter 6 lists our conclusions, future work, and clinical implications.

Chapter 2

Data

2.1 Data Set

The collection of data for this study was performed by by Dr. Susanna Mierau and her group. The data set used in this work contains 16 10-minute time-lapse images of 1600 images and corresponding three-dimensional (3D) images of mouse neuronal cortical cultures. The variations in pixel values over the image sequence represent fluctuations of intracellular calcium that report neuronal activity. Of the 16 image sequences, there are eight Mecp2-deficient and eight wild-type cultures. In addition to these 16 sequences and corresponding 3D images, we have three additional 3D images for a total of 19 3D images.

2.2 Data Collection

We provide a brief description of the data collection process as described in previous work that studies the affects of *Mecp2* deficiency on mouse neuronal networks [2]. Cultures from the primary cortex are harvested from mouse pups sacrificed on the day of or one day after birth. The pups may have one of three genotypes: wild-type (WT), knockout (KO), or heterozygous (Het). Wild-type mice are healthy. KO mice do not make any functional Mecp2 protein due to an *Mecp2*-gene knockout. Het mice have one functional copy and one altered *Mecp2* copy.

[2, p.10] describes the following procedure for data collection:

> The cortex was dissected from the brain and the meninges removed in ice-cold Krebs solution or DPBS[1]. The cortices were transferred into chilled 200 μL of papain mixed with 200 μL of DPBS and incubated for 25 minutes in a 37 °C water bath to begin the process of dissociation. The reaction was terminated by the addition of 1 mL NB-B27 media containing 4% fetal bovine serum (FBS). Cells were then manually dissociated and then centrifuged for 10 minutes with relative centrifugal force (rcf) of 0.4. Afterwards, the supernatant was removed and the cell pellet was re-suspended in 350 μL of NB-B27 media. The cell count was estimated using a hemacytometer and the dissociated cells were plated ... onto ... coverslips for calcium imaging.

For plating the cells on coverslips [2, p.10-11] describes the following:

> The cells were further mixed with NB-B27 media to achieve concentration of $4.5 * 10^5$ cells in 200 μL. Next, the cells were aliquoted onto glass coverslips pretreated with poly-D-lysine and laminin (Neuvitro Corporation, USA) in 24-well plate. The plates with cells were incubated for 30 minutes, before 300 μL of NB-B27 media was added. On DIV1 [2], 500 μL of NB-B27 with 0.5 mM L-glutamine was added. One-third of the media was replaced with fresh media three times per week.

For calcium recordings, [2, p.13-14] describes the following steps:

> Oregon Green 488 BAPTA (Thermofisher) fluorescent dye was prepared by adding 8 μL of Pluronic acid (Thermofisher) and, after 5 minutes mixing via sonification, 72 μL of artificial cerebral spinal fluid [1]. Immediately before application to the cultured cells on the coverslip, the dye mix was diluted with 1:8 warm media. A coverslip was then transferred from the

[1] Dulbecco's phosphate-buffered saline [2, p.9]
[2] day in vitro

24-well plate into a 35 mm petri dish and immediately 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 NB-B27 media to remove eccess dye. The coverslip was transferred to the two-photon rig in 2 mf warm NB-B27 media.

Two photon calcium imaging (3i VIVO) was performed using the Slidebook software (3i). For the recording the coverslip was transferred to a warm circulating bath $28-30\,^{\circ}$C with a flow of HEPES buffer [pH 7.4, osmolality 309-319]. The position of the slide under the microscope was adjusted to maximize the number of cells visible within the frame (517 $\pm$ 1 μm $\times$ 517 $\pm$ 1 μm). A 1600 frame-long timelapse (approximately 10 minutes recorded with frequency 2.8 ± 0.1 Hz) of changes in the green fluorescence (excitation wavelength 800 nm) was recorded from a single Z plane with resolution of 256 $\times$ 256 pixels. Next, a 3D-stack image of the same recording area was taken to visualize the cell morphology with both green (for all cells) and red (for PV-Tdtomato positive cells) filters. The 3D stack is created with 10 Z-planes above and 10 Z-planes below the recorded timelapse with resolution 512 $\times$ 512 pixels.

Figure 2-1 shows an example of a 3D-stack image and Figure 2-2 shows an image from the corresponding imaging sequence. The sequences in the data set are named using the mouse genotype, the date the culture was initiated, which pup in the litter it came from, when the cortex was harvested (on the day of birth or the day after), days in vitro (DIV) of the culture, which coverslip it came from, and which section on the coverslip was recorded.

2.2.1 Annotated Regions of Interest

For each image sequence, neuroscientists manually annotated each neuron, or region of interest (RoI), found in the images. Each RoI is annotated as an ellipse. The

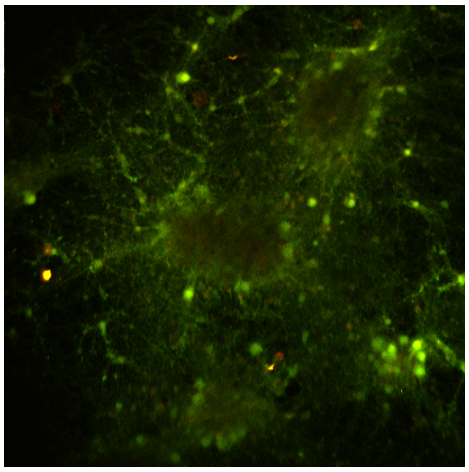

Figure 2-1: **Example of 3D-Image**

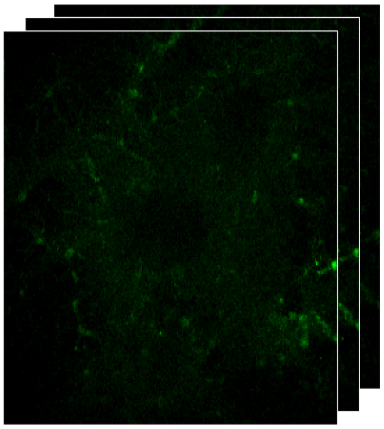

Figure 2-2: **Example of Image Sequence**

annotations include central coordinates, area, perimeter, and major and minor axis descriptions for each RoI.

2.2.2 Annotated Burst Activity Events

Neuroscientists also annotated times of burst activity for various neurons in each image sequence. The annotations include the beginning and end times of each burst, its peak value, and whether the signal displayed characteristics of slow oscillation.

Chapter 3

Neuron Detection

An important part for neuronal network analysis is knowing when and how frequently neurons fire action potentials, or are active. We refer to events related to firing action potentials as burst activity. Improving the ability to analyze neuronal networks depends on reliable detection of burst activity. The signals extracted from identified neurons affect the detection of burst activity. Therefore, we would like to extract signals with high signal-to-noise ratio from which burst activity events can easily be detected. In order to extract signals, however, we first need to detect regions of the images that are neuron cell bodies. Moreover, we would like to detect as many neuron cell bodies as possible without introducing too many false positives. This motivates improving the the first step of the pipeline: neuron detection. We also want to define good boundaries for each identified neuron cell body. Defining better boundaries can help assign more representative pixels to each neuron from which we can extract better signals.

While methods in previous work achieved promising results in detecting the locations of neuron cell bodies, they relied on unsupervised methods. Furthermore, most methods attempt to pinpoint the central coordinates of the cell bodies and draw approximate circular boundaries around these centers.

We first revisited some of these unsupervised methods to determine whether they could be modified to identify more neuron cell bodies. Following this, we also take advantage of the annotated RoIs data set to investigate a supervised learning method

for neuron detection.

The Neurofinder competition is a public competition for detecting neurons in calcium imaging data sets similar to ours and ranks submitted solutions [18]. Dr. Mierau noted that the data sets come from vivo recordings whereas our data set is composed of in vitro recordings. Additionally, it was noted that the images used in this competition appear to have a 3D cellular architecture whereas our imaging sequences consist of 2D layers of cells. Nonetheless, we thought that the methods submitted to this competition may still be applicable to our case. Several of the algorithms submitted make use of convolutional neural networks (CNN) [18]. One of the leading solutions is based on the Mask R-CNN model [16]. While details for this solution were not found, we found that another Mask R-CNN model performed well for detecting nuclei in images of cells [19]. Therefore, we thought using a Mask R-CNN model was promising and trained our own model using this architecture for neuron detection on our data.

Rather than use images in the time-lapse sequence itself to identify neurons as done in [4], we perform neuron detection on the 3D image associated with each sequence instead. We then use these detected regions to extract signals from the imaging sequence. Upon visual inspection of the 3D images, we thought that the neurons on these images were easier to identify for an untrained eye. Therefore, we thought the 3D images may provide an opportunity for better neuron detection for our methods as well.

3.1 Methods

3.1.1 Threshold and Contour

We first revisited the Threshold and Contour detection method from previous work [4]. Because we wanted to detect as many neurons as possible, we eliminated the adaptive threshold step before detecting contours. Instead, we simply threshold the 3D image using OpenCV's Otsu thresholding [20] and then run contour detection on

the outputted binary image . We run contour detection in two iterations. First, we limit the minimum enclosing circle of any particular contour to a radius of 40 pixels. Although this generally identifies most RoIs, larger contours would often be too large and enclose multiple neurons together while many small "noisy" contours also would be identified. In an attempt to eliminate these noisy contours, we experimented with using a smaller radius on this first iteration. However, we found that this would often eliminate smaller contours that actually contained neurons. Therefore, we maintained the larger radius of 40 and instead performed a second iteration. In this second iteration, we find all contours that are greater than 5 pixels in radius and then rerun contour detection on the section of the image described by the bounding box of the contour. However, this time we decrease the maximum radius allowed to 10 pixels. If we find smaller contours, we eliminate the original contour and add the new contours to our final list. Otherwise, we keep the original contour in the final list. Pseudocode for this algorithm is presented below.

In [4], the center of each contour was found and then a circle was drawn around this center to represent the neuron cell body region. Here, rather than use the minimum enclosing circle of each contour as the bound for each cell body, we kept the actual contours as the bounds. We thought this would be a more accurate representation of the neuron and may affect the signal extracted from it. While this method generally performed well in identifying most of the neuron cell bodies, we found that even after the second iteration, contours would often enclose two neurons together. This was evident particularly in cases where cell bodies were close to each other and the surrounding regions of the neuron cell bodies were lighter. Additionally, the 3D images often show regions of the network other than cell bodies clearly as well. These regions may include axons, dendrites, and small bits of debris. Therefore, many of the small, noisy contours identified these other regions. Figure 3-1 shows an example of contours found using this method.

Algorithm 1 Psuedocode for Threshold and Contour

```
max_radius = 40
initial_contours ← Find contours in image bounded by max_radius
max_radius = 10
final_contours = []
for contour in initial_contours
    if radius of contour > max_radius then
        crop_image ← Section of image defined by bounding box of contour
        new_contours ← Find contours in crop_image bounded by max_radius
        if length of new_contours > 0 then
            Add new_contours to final_contours
        else
            Add contour to final_contours
        end if
    end if
return final_contours
```

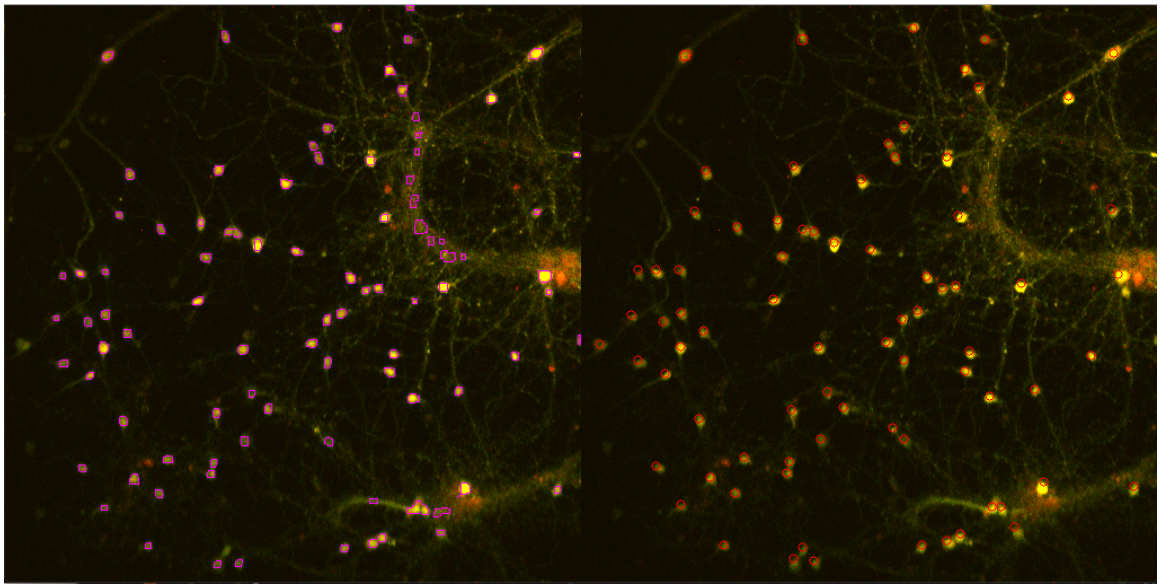

Figure 3-1: **Threshold and Contour Results.** Results of Threshold and Contour on left. Annotated cell bodies on right.

3.1.2 Template Matching

In addition to the Threshold and Contour method, we also revisited the Template Matching method. In previous work, it was found that the model was highly sensitive to the bag of neuron templates used [4]. Therefore, we increased the number of templates used from 4 to 29. We attempted to choose templates that we thought would help identify closely clustered cell bodies as well as cell bodies against lighter backgrounds. Examples of these templates can be seen in Figure 3-2. The template matching method finds areas in a given image that are similar to a template image [4, 20]. Similar to previous work, we combine our bag of templates with the OpenCV template matching algorithm with correlation coefficient metric [4, 20]. The correlation coefficient equation evaluates the strength of a match [20]. Using this equation, a similarity image is generated and then a threshold is applied to it. We maintained the threshold value of 200,000 from previous work, which appeared to perform well on the 3D images as well. Finally, Birch clustering from scikit-learn [21] is used to find the centers of the matched regions. We also maintained the Birch threshold value of 10 from previous work.

While we found this method works well in identifying central coordinates for most neuron cell bodies in an image, it is also unable to distinguish well between closely clustered neurons. In many cases, it will label the middle of multiple neuron cell bodies as the center of a single neuron cell body. Figure 3-3 shows an example of regions found from this method.

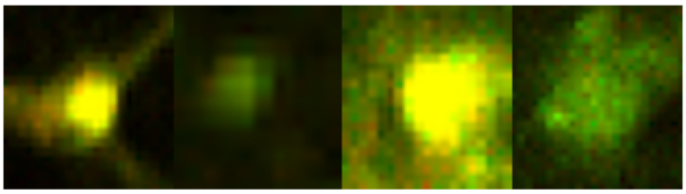

Figure 3-2: **Templates for Template Matching.** Examples of Neuron Cell Body Templates

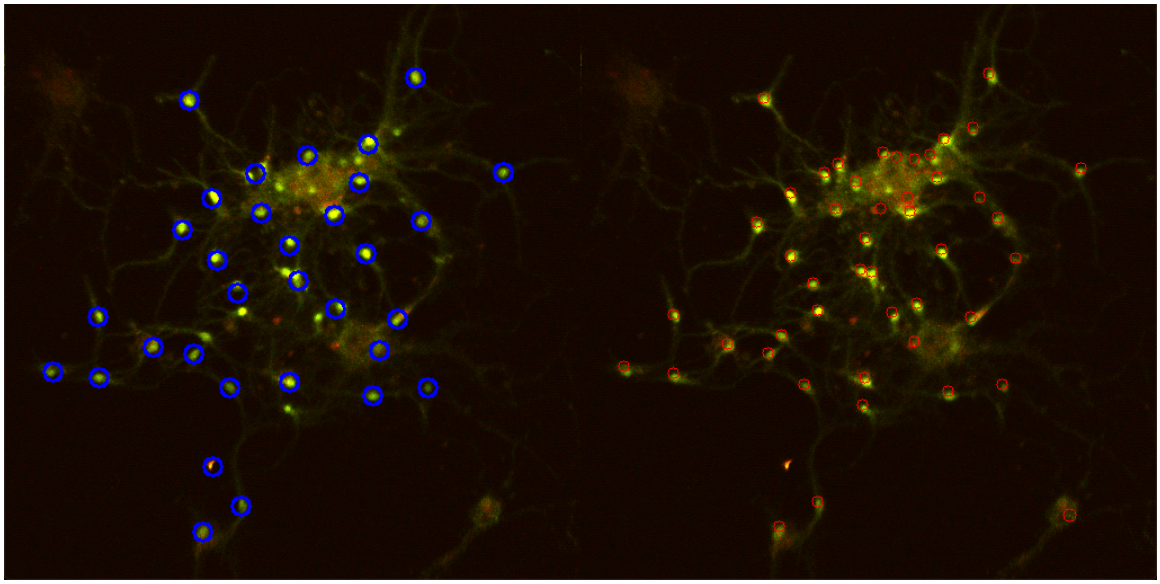

Figure 3-3: **Template Matching Results.** Results of Template Matching on Left. Annotated RoIs on right.

3.1.3 Mask R-CNN for Neuron Detection

While the unsupervised methods performed well, we wanted to investigate whether we could achieve better performance with a supervised learning method by taking advantage of the expertly annotated RoIs. Mask R-CNN has been shown to perform

well in general object detection and instance segmentation [16] and was also able to detect nuclei in images of cells well in [19]. Therefore, we thought it may also perform well on our data set. Furthermore, the model is easy to adapt without extensive modification [22].

Mask R-CNN

Mask R-CNN was developed by Facebook AI Research (FAIR) as a "conceptually simple, flexible, and general framework for object instance segmentation" [16, p.1]. The model not only efficiently detects objects in an image, but also generates high-quality segmentation masks for each instance [16]. We provide a brief overview of the model here.

[16, p.3] states that it outputs "for each candidate object, a class label and a bounding-box offset" and "the object mask".

Data Set Creation

In order to train a Mask R-CNN model, we had to first create an appropriate data set from our images and annotated RoI statistics. For each image, we had to create mask images, each of which depicted a single neuron in the image. [16, p.3] describes a mask as encoding "an input object's *spatial* layout". The annotated RoI statistics treat each RoI as an ellipse and provide the central coordinates, area, perimeter, major axis coordinates, and major and minor axis lengths. However, we thought that an ellipse was not a precise enough depiction of a neuron. Therefore, we used the central coordinates from these annotations combined with other methods to create masks for the 3D image.

For some of the images, we had an associated mask image ("total-mask image") that had contours of every neuron in the 3D image. An example of a total-mask image can be seen in Figure 3-4. We ran the Threshold and Contour detection method on this total-mask image to separate each RoI and create separate binary mask images for each RoI. While these masks generally depicted each RoI as the same shape, we thought that they still covered the neuron regions reasonably well.

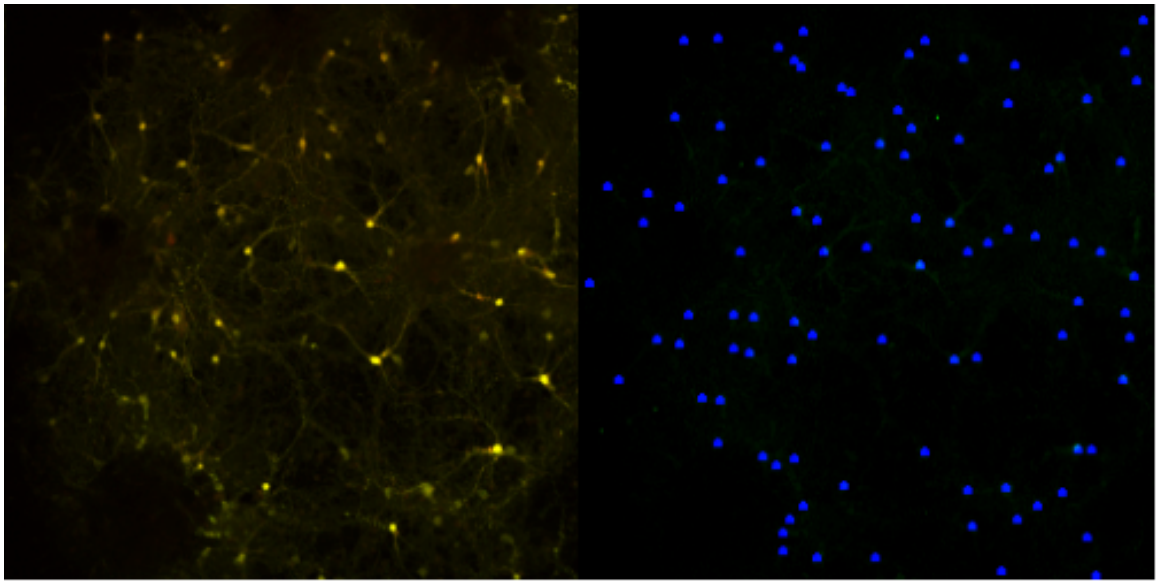

Figure 3-4: **Total-Mask Image.** Example of total-mask image (right).

For 3D images for which we did not have these total-mask images, we used our Threshold and Contour method to find contours of potential RoIs in the image. We then compared these contours to the central coordinates of the annotated RoIs. We selected contours that appeared to match up with the central coordinates of annotated RoIs to create mask images. Examples of these mask images can be seen in Figure 3-5.

Figure 3-5: **Masks Created from Contours.** Examples of masks created from contours (first, second, and third images from the right).

We note that this data set creation method may limit the performance of the model in certain ways. For example, when using the total-mask image, the contours are not entirely specific to the shapes of the neuron cell bodies. Therefore, the masks created may not be exactly true in shape. When using the threshold and contour detection

method to create masks, the masks may still be subject to the issues we presented with that method, such as having masks that encompass two closely clustered neurons as one.

Extension to Neuron Detection

Our implementation stems from an existing Mask R-CNN implementation by Matterport Inc. released under the MIT License [22].

We followed the training procedure outlined for nucleus detection in [22]. We initialized the model using pre-trained weights for the MS COCO data set [23] provided in [22] and trained the model in two stages. In the first stage we only trained "the RPN, classifier and mask heads of the network" [22] of the network with learning rate 0.001. Next, we reduce the learning rate by a factor of 10 and train all layers in the network.

We found that our supervised learning model based on Mask R-CNN was able to find good contours for the neurons it detected. Furthermore, at a minimum prediction confidence level of 0.5, the inference model was able to detect the majority of neurons in an image. Additionally, we found that this method was able to separate closely clustered neurons better than the previous methods in several cases. In some cases, the method detects one of the closely clustered neurons rather than encompass both into a single contour. Furthermore, we noticed that in some cases the model even picked up on neurons that were not annotated in the mask images. We think that this shows that the model is able to learn a good understanding of what a neuron looks like. While it still does enclose two neurons in one contour for some images, we think that this is a result of the masks it was trained with, as previously described. Therefore, we think that creating better masks may help the model avoid this problem.

3.1.4 Combining Methods

While the Mask R-CNN model detects the majority of neurons in an image, we wanted to extend the method to detect even more neurons. The Template Matching method,

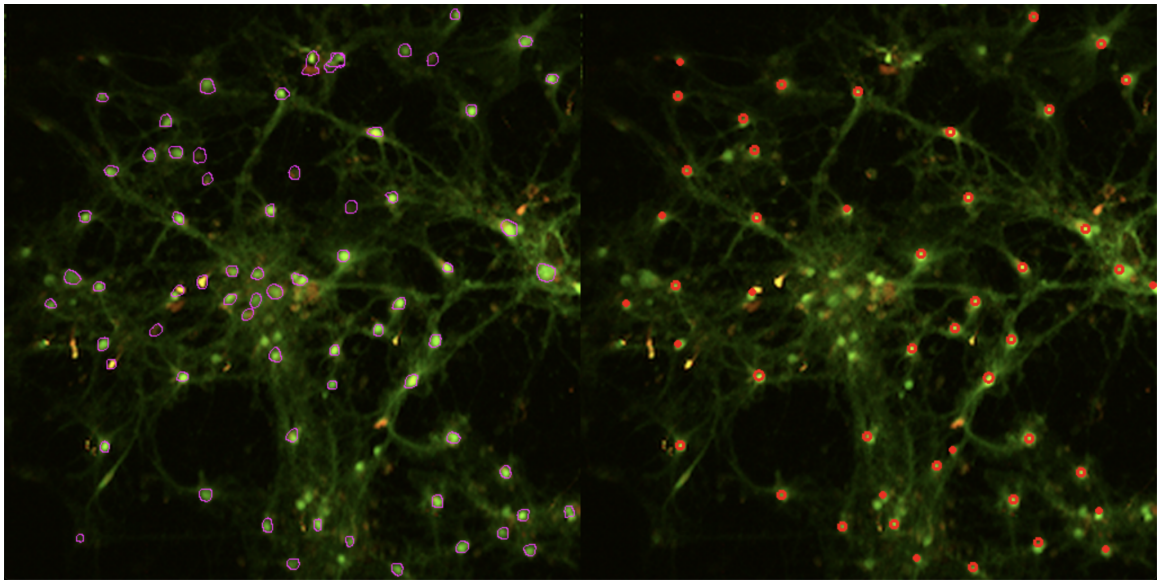

Figure 3-6: **Mask R-CNN Results.** Results of Mask R-CNN model on left. Annotated RoIs on right.

despite detecting a higher number of false positives, identified neurons that the Mask R-CNN model did not in several cases. However, the method detects the center of each neuron cell body and only an approximate boundary.

We decided to combine these methods in an attempt to take advantage of their different strengths and detect as many true neurons as possible. We first detect neuron contours using the Mask R-CNN model. We then detect neuron circles using the Template Matching method. We check to see whether each circle overlaps with the minimum enclosing circle of a contour found from Mask R-CNN. If a circle does not overlap with any contour, we add it as an additional neuron to our final list of detected neurons. Pseudocode for this method is shown below. Figure 3-7 shows an example of the results from this method.

Algorithm 2 Pseudocode for Combined Method

```
contours ← Find contours using Mask R-CNN
circles ← Find circles from Template Matching method
nonoverlapping_circles = []
for circle in circles
    if circle does not intersect any contour in contours then
        Add circle to final_circles
return contours, nonoverlapping_circles
```

We perform an ablation study to show the advantages of combining methods in the Combined Model. We evaluate the methods separately and compare their performances to that of the Combined Model. These results, along with those of the Threshold and Contour method, can be seen in Table 3.1.

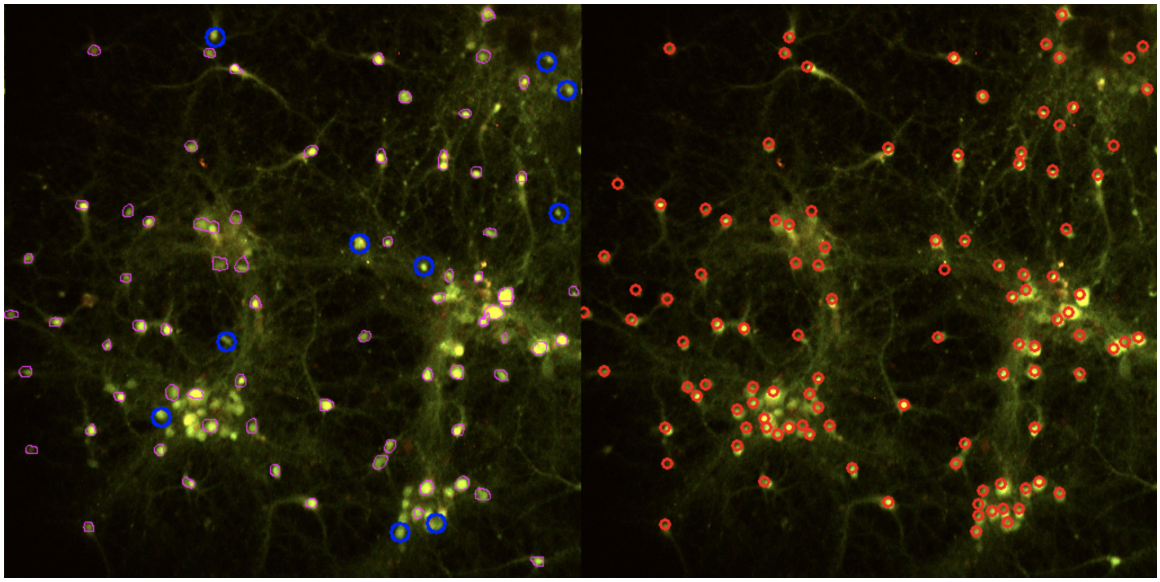

Figure 3-7: **Combined Model Results.** Results of Combined Model on left (Mask R-CNN model contours in pink and Template Matching circles in blue). Annotated RoIs on right.

3.2 Results and Discussion

We use the annotated RoIs data set to evaluate each neuron detection method. We calculate the precision and recall values by determining, for each annotated RoI, whether any of the detected neurons overlap with it. When considering overlap, each annotated RoI is approximated with a radius of 5 pixels. Each detected contour is represented by its minimum enclosing circle, where its radius is capped at 20 pixels. Similarly, for each detected circle, its radius is capped at 20 pixels. This evaluation does not penalize against a single detected region overlapping with multiple annotated RoIs. Even though this may affect results for closely clustered neurons, we thought this was still reasonable as we did not observe detected regions being too large and this allows to compare our results to previous work [4].

We evaluate the Threshold and Contour and Template Matching methods on the 19 3D images against the RoIs found in the annotated data set. For the Combined Model and Mask R-CNN Model, we evaluate the methods on using leave-one-out cross-validation. We trained a model on each training set on a compute-optimized instance with 16 vCPU and 64 GB memory. We trained for 4 epochs in each stage and evaluated each test set with a confidence level of 0.5. It took approximately 4-5 hours to complete training for each data set. The average recall, average precision, and average F1 scores for each method is shown in Table 3.1.

Table 3.1 also includes values reported for the highest scored detection method from previous work (as evaluated in [4]): Threshold, Aggregate, and Contour with Evolutionary Parameter Optimization. The F1 score presented in Table 3.1 for this method is calculated using the average recall and average precision values reported. We note that our highest performing methods have faster average run time, comparable precision, much higher recall, and higher average F1 score.

In addition to leave-one-out cross-validation for the combined and Mask R-CNN models, we created five additional data sets from the 19 3D images with less training examples in each. Each data set has 14 training images and 5 test images. We also train each model on the corresponding training set on a compute-optimized instance

with 16 vCPU and 64 GB memory. We trained for 4 epochs in each stage and evaluated each test set with a confidence level of 0.5. It took approximately 4-5 hours to complete training for each data set. The results of this evaluation in Table 3.2 show the average of the average values obtained from the five test sets.

Neuron Detection Model	Training Time (min)	Run Time (s)	Recall	Precision	F1 Score
Combined Model	$\mu = 262.8$ $\sigma = 4.4$	$\mu = 13.7$ $\sigma = 1.5$	$\mu = 76.6\%$ $\sigma = 9.3\%$	$\mu = 85.7\%$ $\sigma = 14.6\%$	$\mu = 79.8$ $\sigma = 8.0$
Mask R-CNN Model	$\mu = 262.8$ $\sigma = 4.4$	$\mu = 11.4$ $\sigma = 1.0$	$\mu = 63.9\%$ $\sigma = 12.2\%$	$\mu = 87.4\%$ $\sigma = 13.3\%$	$\mu = 72.4$ $\sigma = 8.9$
Template Matching	N/A	$\mu = 1.4$ $\sigma = 0.5$	$\mu = 66.0\%$ $\sigma = 13.0\%$	$\mu = 91.7\%$ $\sigma = 10.4\%$	$\mu = 75.7$ $\sigma = 9.3$
Threshold and Contour	N/A	$\mu = 0.04$ $\sigma = 0.01$	$\mu = 82.5\%$ $\sigma = 7.1\%$	$\mu = 69.3\%$ $\sigma = 21.8\%$	$\mu = 72.9$ $\sigma = 13.5$
Threshold, Aggregate, and Contour +Evolutionary parameter Optimization	N/A	$\mu = 17$	$\mu = 58.6\%$ $\sigma = 18.2\%$	$\mu = 94.1\%$ $\sigma = 8.8\%$	72.2

Table 3.1: **Evaluation of Neuron Detection Methods.** The table above shows the training time, run time, recall, precision, and F1 scores for the four methods described as well as the top-scoring method from previous work.

Neuron Detection Model	Run Time (s)	Recall	Precision	F1 Score
Combined Model	$\mu = 14.3$ $\sigma = 0.9$	$\mu = 83.6\%$ $\sigma = 9.0\%$	$\mu = 85.7\%$ $\sigma = 4.1\%$	$\mu = 81.4$ $\sigma = 2.8$
Mask R-CNN Model	$\mu = 11.9$ $\sigma = 3.5$	$\mu = 68.7\%$ $\sigma = 7.2\%$	$\mu = 87.4\%$ $\sigma = 3.5\%$	$\mu = 75.3$ $\sigma = 4.5$

Table 3.2: **Results of Evaluation Using Five Additional Data Sets.**. The table shows the run time, recall, precision, and F1 scores of the Combined Model and Mask R-CNN model obtained from training with fewer examples.

We conduct Wilcoxon rank-sum tests in order to determine whether there is a significant difference between the neuron detection methods. We conduct separate tests for precision, recall, and F1 score. Samples for the Combined Model and Mask R-CNN Model consist of 19 values from the leave-one-out cross validation. Samples for the Threshold and Contour and Template Matching methods also consist of 19 values for each of the 19 images. We use [24][1] to conduct the tests, where the null hypothesis is that "the distribution of x and y differ by a location shift mu and the alternative is that x is shifted to the right of y". x and y refer to the two samples and mu is 0 in our case. Table 3.3, 3.4, and 3.5 show these results[2]. Table 3.5 notes significant differences in F1 score when using the Combined Model. We show box and whisker plots for the values obtained from each method in Figures 3-8 through 3-11.

We also conduct a Wilcoxon rank-sum test between the leave-one-out Mask R-CNN models and the Mask R-CNN models obtained from training with the five additional data sets described earlier using the same procedure above. In this case, we take x to be the leave-one-out results. We conduct three separate tests: for precision, recall, and F1 score to determine if there is a significant difference between training the Mask R-CNN models with 14 images versus 18 images. Our first sample consists of 19 values from the leave-one-out cross validation. Our second sample consists of 5 values, each an average of the results obtained from the five additional data sets described earlier. The results of these tests can be seen in Table 3.6. The box and whisker plot for the five additional Mask R-CNN models can be seen in Figure 3-12.

[1]The function uses a normal approximation when an exact p-value cannot be found.

[2]The Wilcoxon rank-sum test assumes two independent samples [25]. We note that the performance of the Combined Model depends on both the Mask R-CNN model and the Template Matching method, so these samples may not be independent of each other. We still include these values in the table but realize that the dependence might affect results.

Method x y	Combined Model	Mask R-CNN	Threshold and Contour
Mask R-CNN Model	0.6	-	-
Threshold and Contour	0.007	0.004	-
Template Matching	0.9	0.9	1.0

Table 3.3: **P-values from Wilcoxon Rank-Sum Tests Between Methods for Precision Values.** The table shows the p-values obtained from the Wilcoxon rank-sum tests for precision values for each pair of methods.

Method x y	Combined Model	Mask R-CNN	Threshold and Contour
Mask R-CNN Model	0.001	-	-
Threshold and Contour	1.0	1.0	-
Template Matching	0.02	0.6	0.0003

Table 3.4: **P-values from Wilcoxon Rank-Sum Tests Between Methods for Recall.** The table shows the p-values obtained from the Wilcoxon rank-sum tests for recall values for each pair of methods.

Method x y	Combined Model	Mask R-CNN	Threshold and Contour
Mask R-CNN Model	0.009	-	-
Threshold and Contour	0.05	0.7	-
Template Matching	0.1	0.9	0.7

Table 3.5: **P-values from Wilcoxon Rank-Sum Tests Between Methods for F1 Scores.** The table shows the p-values obtained from the Wilcoxon rank-sum tests for F1 scores values for each pair of methods.

Test	P-Value
Precision	0.3
Recall	0.8
F1 Score	0.8

Table 3.6: **P-values from Wilcoxon Rank-Sum Tests between Leave-One-Out Mask R-CNN Model and Mask R-CNN Model with Less Training Images.** The table shows the p-values obtained from the Wilcoxon rank-sum tests between the values obtained from the leave-one-out Mask R-CNN models and the Mask R-CNN models using the additional five data sets.

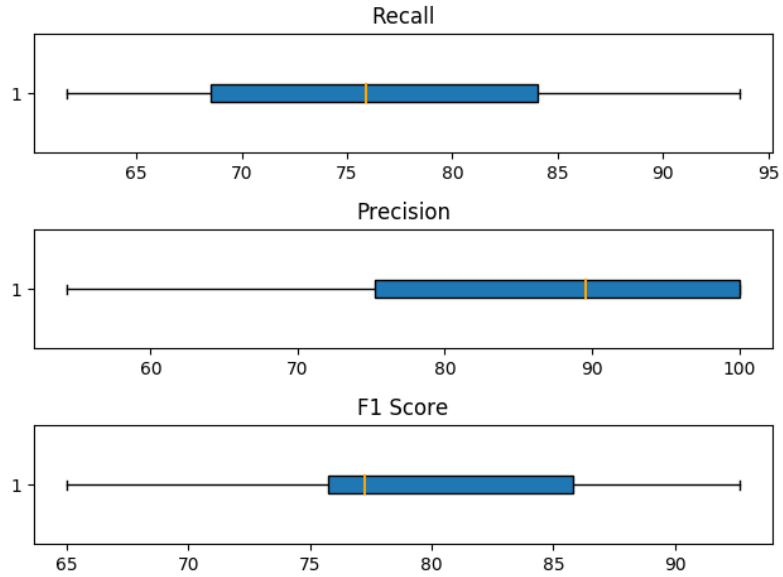

Figure 3-8: **Recall, Precision, F1-Score Box Plots for Combined Model.** This figure shows the recall, precision, and F1 scores obtained using the leave-one-out Combined Models.

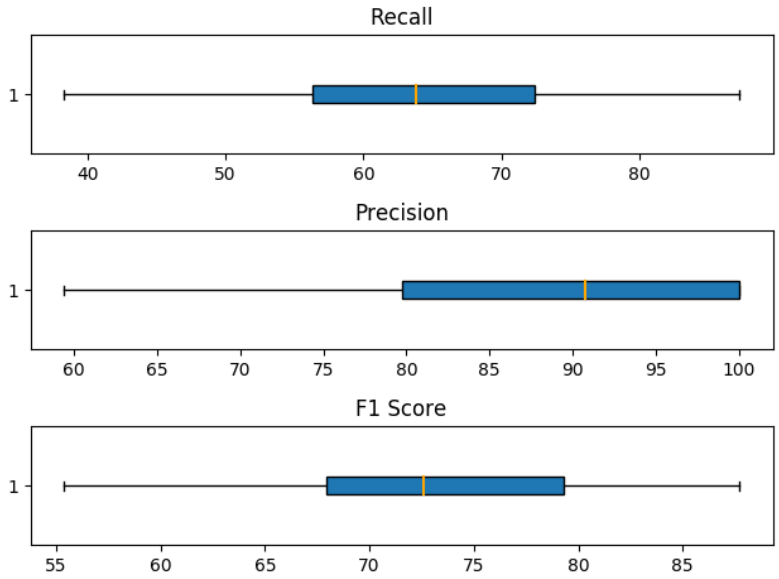

Figure 3-9: **Recall, Precision, F1-Score Box Plots for leave-one-out cross-validation Mask R-CNN Models.** This figure shows the recall, precision, and F1 scores obtained using the leave-one-out Mask R-CNN models.

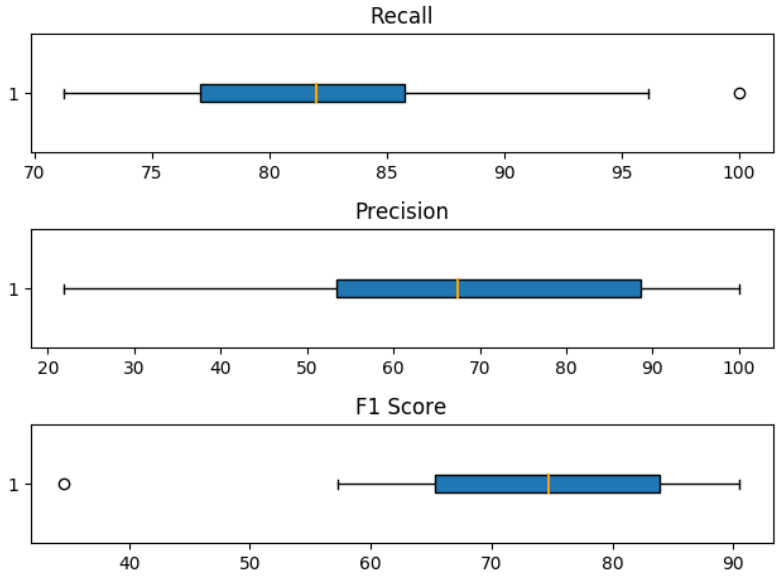

Figure 3-10: **Recall, Precision, F1-Score Box Plots for Threshold and Contour.** This figure shows the recall, precision, and F1 scores obtained using the Threshold and Contour method.

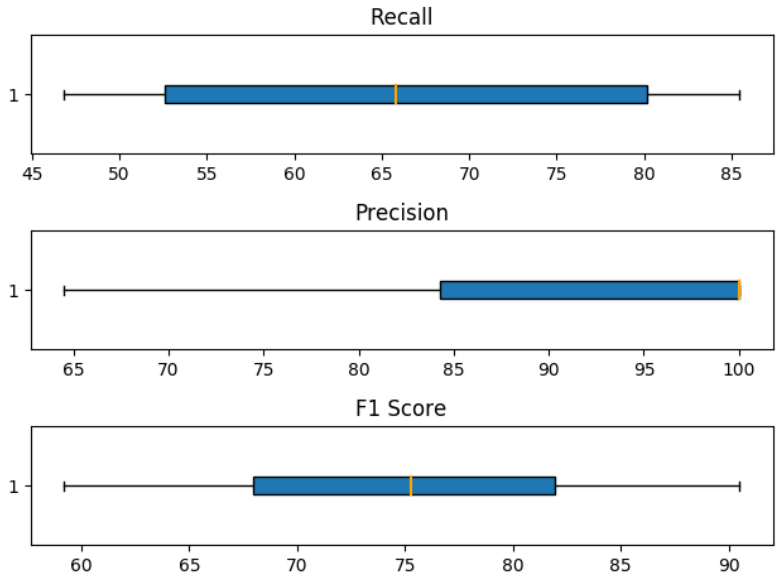

Figure 3-11: **Recall, Precision, F1-Score Box Plots for Template Matching.** This figure shows the recall, precision, and F1 scores obtained using the Template Matching method.

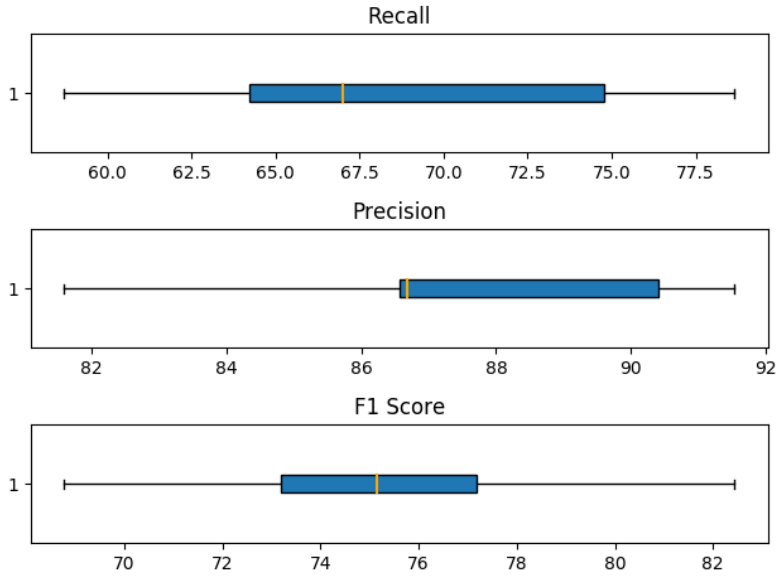

Figure 3-12: **Recall, Precision, F1-Score Box Plots for Mask R-CNN Models using Additional Data Sets.** This figure shows the recall, precision, and F1 scores obtained using the five additional data sets for the Mask R-CNN models.

Chapter 4

Neuronal Signal Extraction

After detecting neurons in the 3D image, we extract signals from the corresponding regions in the imaging sequence. We resize the images in the imaging sequences to match the dimensions of the 3D image. The extracted signals indirectly measurements of neuronal activity [26] and are used to detect burst activity. Ideally, we want to extract signals with minimal noise and clear burst activity. In this chapter, we describe the signal extraction methods using several different heuristic filters that we investigated. We provide a brief evaluation of these methods in the next chapter.

4.1 Original Signal Extraction Method

A single signal depicting neuronal activity over time is extracted from each identified neuron cell body. We extract this signal by summing all pixels within a neuron boundary and dividing by the number of pixels in the boundary at each frame within the 1600 images of a sequence. An example of an extracted signal is shown in Figure 4-1.

For every image sequence, we also extract a signal consisting of 3-5 pixels from the top left of the images. We subtract this signal from all other signals extracted from the sequence. Because this region is predominantly black, the signal extracted from these pixels may represent artefacts in the images which we can remove from neuronal signals. Generally, this signal predominantly consists of zeroes and very small values

and subtracting it from other signals does not produce a noticeable difference.

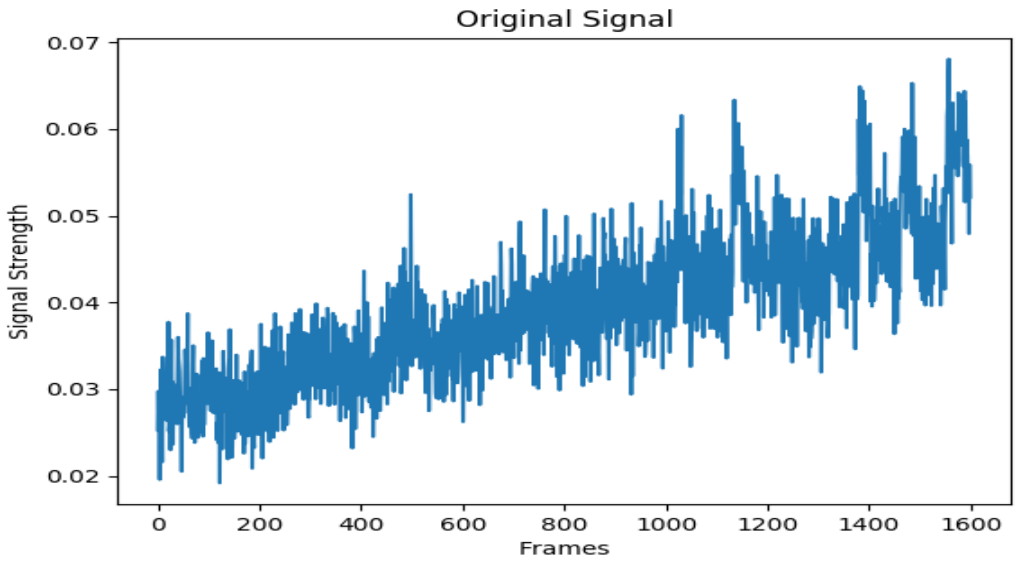

Figure 4-1: **Example of Signal Extracted from Identified Neuron Cell Body**

The extracted signals generally come with a high degree of noise. We would ideally like to extract the underlying "true" signal. It is possible that extracting signals from a limited number of pixels in each contour can help extract a signal that is closer to the ground truth signal. In the following sections, we describe different approaches for filtering pixels in an attempt to extract better signals.

4.2 Variance-based Signal Extraction

The first heuristic filter we explored was to filter pixels based on variance. We extracted a signal for a neuron using only the pixels in the contour whose individual signals had the highest variances. In other words, we find a raw signal for every pixel ("pixel signal") in the contour and calculate its variance. We then find pixels with the highest variance values and average their signals to produce the final signal for the neuron. Because burst activity is associated with increases in signal intensity values, we thought that using the variance as a heuristic filter for pixel selection may help extract a signal in which burst activity was clearer. A heat map showing variances

for each pixel in the detected neuron contours can be seen in Figure 4-2.

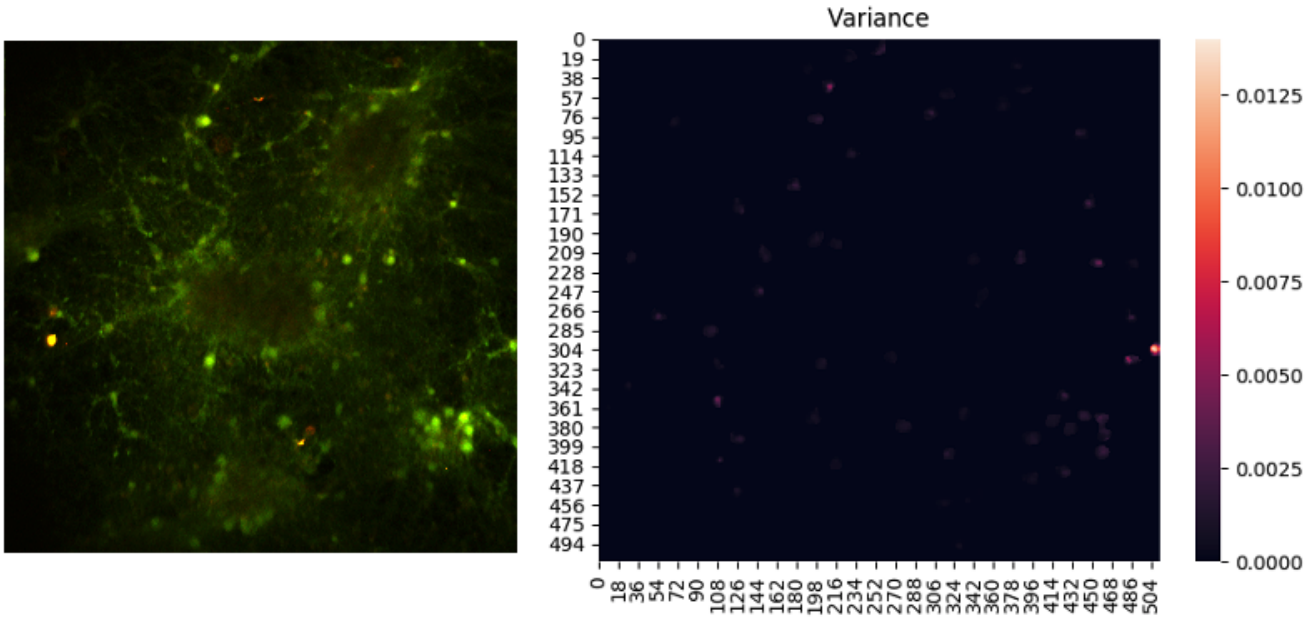

Figure 4-2: **Heatmap of Pixelwise Signal Variances for Detected Contours.** 3D image on right. Heatmap of detected contours on left.

First, we extracted signals from the five pixels with highest variance in the contours. However, we suspected that these signals may be too noisy, producing false burst activity events possibly. Therefore, we instead considered these five pixels as the core cluster of a neuron. We then extract a signal from only this core cluster, the "core cluster signal". Next, we add a limited number of pixels from the rest of the contour to this core cluster based on the pixel signals' correlations with the core cluster signal. We added the pixels with the highest correlations, creating the final cluster from which the neuronal signal is extracted.

We found that sometimes the five pixels with highest value were not adjacent to one another and thought that this might be due to noise or artefacts. Therefore, we enforce the core cluster to be connected. Once we extract the top five pixels, we check to see if the pixels are connected. If they are connected, they form the core cluster. Otherwise, we find the pixel with the highest value and add its neighboring pixels to form the core cluster still capped at five pixels.

We experimented with two different correlation metrics. We first experimented with cosine similarity between each pixel signal and the core cluster signal as the correlation metric ("cosine correlation"). We add the pixels whose pixel signals have with highest cosine similarity to the core cluster. We also investigated a similarity metric based on Hamming distance ("OASIS correlation"). Once we extracted the core cluster signal, we find points in the signal that correspond to increases in neuronal activity, or burst activity, using the OASIS deconvolution method (described in Chapter 5). We will refer to these points as bursts and the sequence of bursts for a signal as the burst train. This burst train is positive where the method detects an increase in burst activity in the signal. We convert this burst train to a binary list that is positive where the burst train is positive. From each of the remaining pixels, we extract pixel signals and find its binary burst train ("pixel burst train"). We compare the pixel burst train to the core cluster burst train and find the number of indices with positive burst train values that the pixel burst train and the core cluster burst train have in common. If the Hamming distance between the binary burst trains is defined as d, then this is also equivalent to $1 - d$. The proportion of matching indices is the correlation score for that pixel.

Because it was unknown how many pixels would be best for neuronal signal extraction, we experimented with the number of pixels allowed in the final cluster. Specifically, we experimented with 5, 10, 15, and 20 pixels. However, in order to be added to the final cluster, the correlation value must be greater than zero. For several instances, neuroscientists noted better signal-to-noise ratios. In other instances, they noted that signals extracted from all pixels within a contour produced clearer signals. However, it was also thought that finding the optimal number of pixels for each identified neuron cell body region may help alleviate this variability in signal-to-noise ratio. Examples of signals found using variance and these different correlation metrics can be seen in Figures 4-3 and 4-4.

Extracting signals per pixel in each contour can take between 20-70 minutes for a single sequence with multiprocessing on a compute-optimized 16vPU and 64 GB memory machine, depending on the number of contours detected and their sizes.

However, once these signals are extracted, the remaining pipeline does not have any major run time blocks.

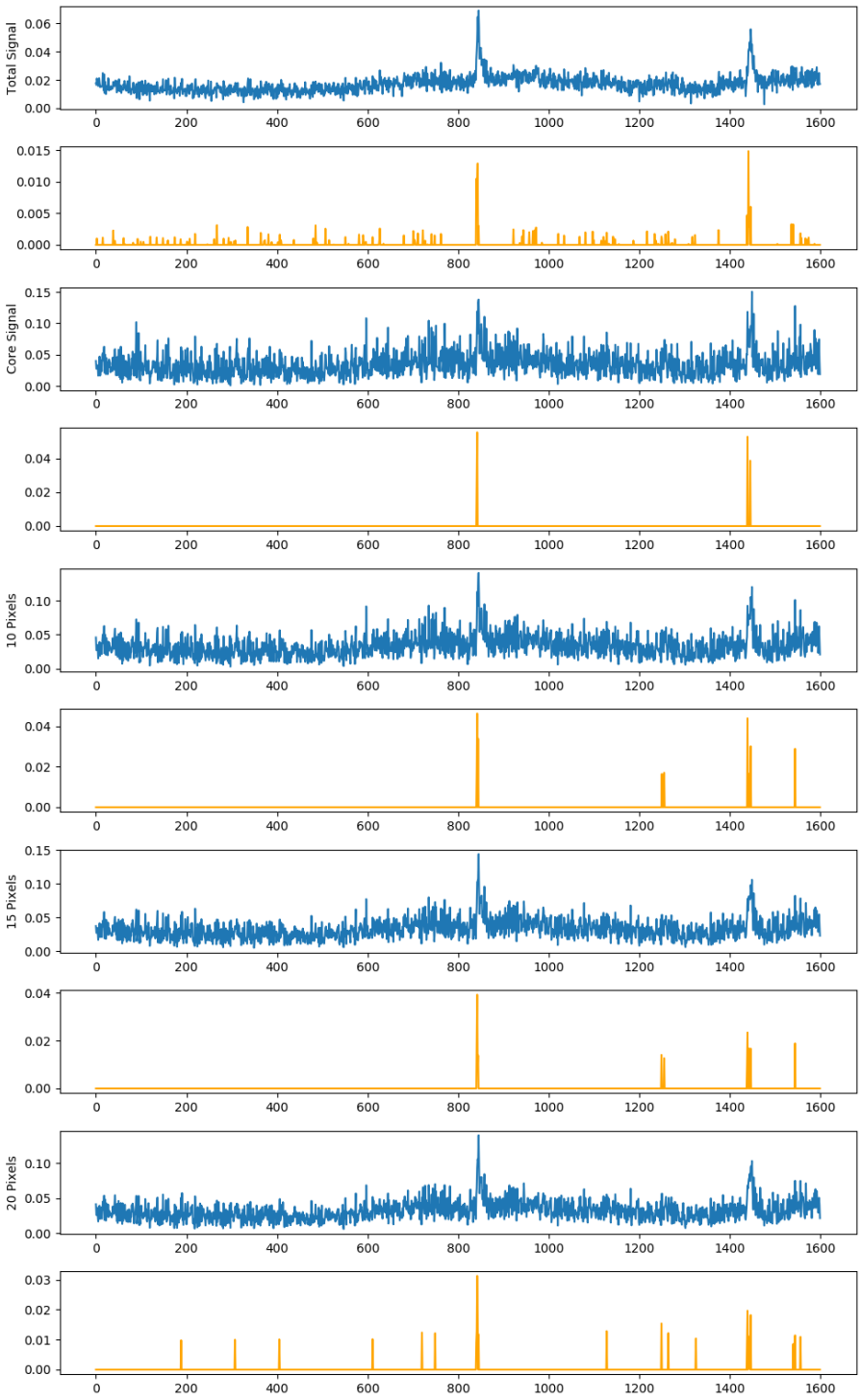

Figure 4-3: **Signals Extracted using Variance and Cosine Correlation and the Corresponding Burst Trains.** The topmost signal was extracted from all pixels in the contour. The second from the top is the core cluster signal consisting of 5 pixels. The next signal is the final cluster signal capped at 10 pixels. The following pixel is the final cluster signal capped at 15 pixels. The last signal is the final cluster signal capped at 20 pixels. Each signal is shown in blue followed by its burst train in yellow.

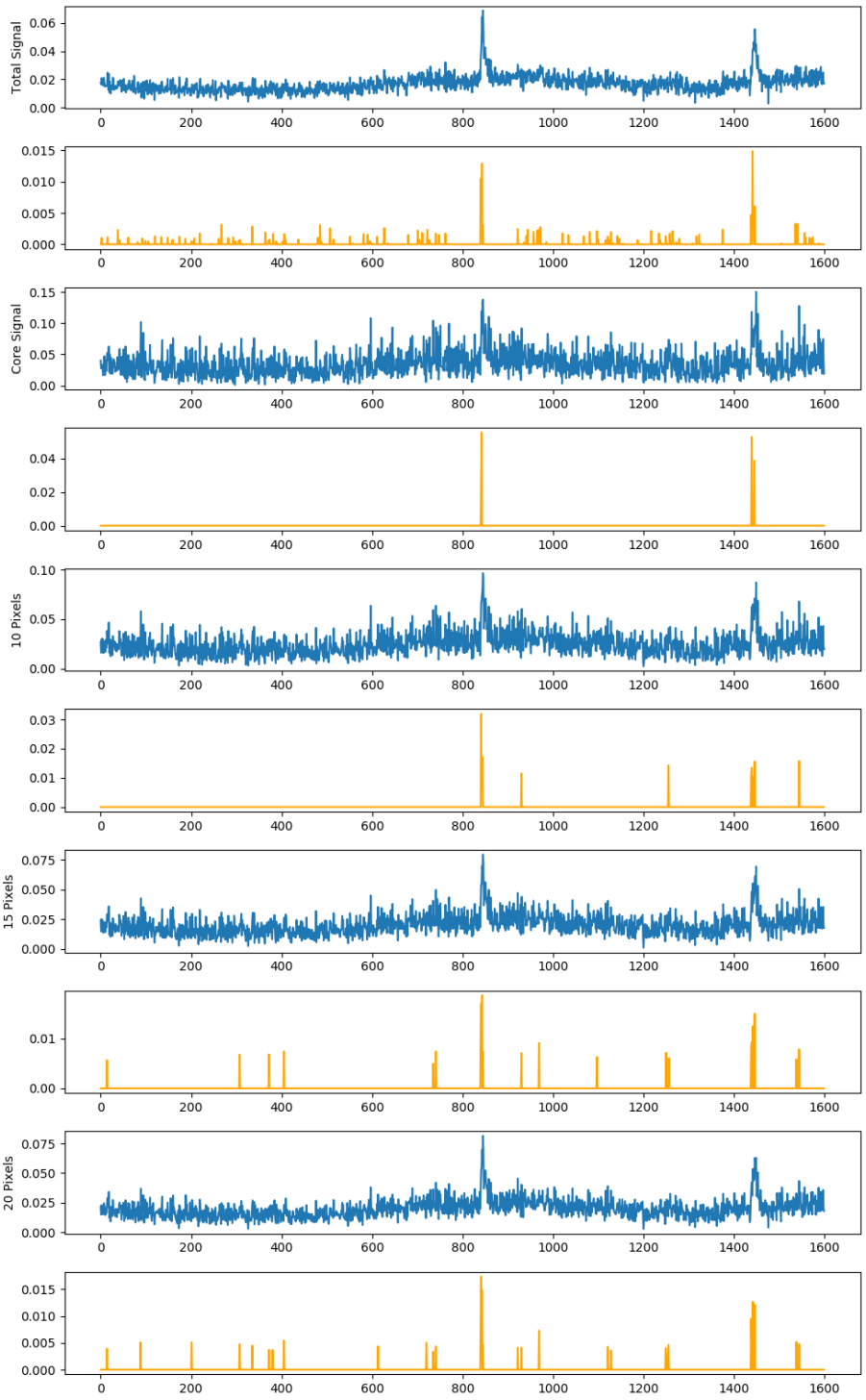

Figure 4-4: **Signals Extracted using Variance and OASIS Correlation and the Corresponding Burst Trains.** The topmost signal was extracted from all pixels in the contour. The second from the top is the core cluster signal consisting of five pixels. The next signal is the final cluster signal capped at 10 pixels. The following pixel is the final cluster signal capped at 15 pixels. The last signal is the final cluster signal capped at 20 pixels. Each signal is shown in blue followed by its burst train in yellow.

4.3 MMA-based Signal Extraction

This method follows a similar procedure as the variance-based extraction. Instead of calculating the variance of each pixel signal, we find the maximum, minimum, and average values of the pixel signal over the 1600 images in the sequence. We then calculate $\frac{maximum-minimum}{average}$ (MMA) for each pixel. We now use these values to find the core and final clusters. The rest of the procedure follows from the previous section.

In the variance-based extraction method, we found that the central pixels of a contour tend to show the highest values. However, with the MMA method, we find that the boundaries of the contour tend to show the highest values. This can be seen in the heat maps in Figure 4-5 and 4-6 (we ommited contours that have extremely high values that overshadowed the remaining values in Figure 4-5). Neuroscientists found the MMA values to align with their experiments as the fluorescent dye is applied to the outside of the cells. In the example in Figure 4-6, we see that the highest-value pixels according to variance are the lowest-value pixels according to MMA (and vice versa). Examples of signals extracted using MMA values and the two different correlation metrics can be seen in Figure 4-7 and 4-8. Figure 4-9 shows masks of the different pixels used for each signal extraction.

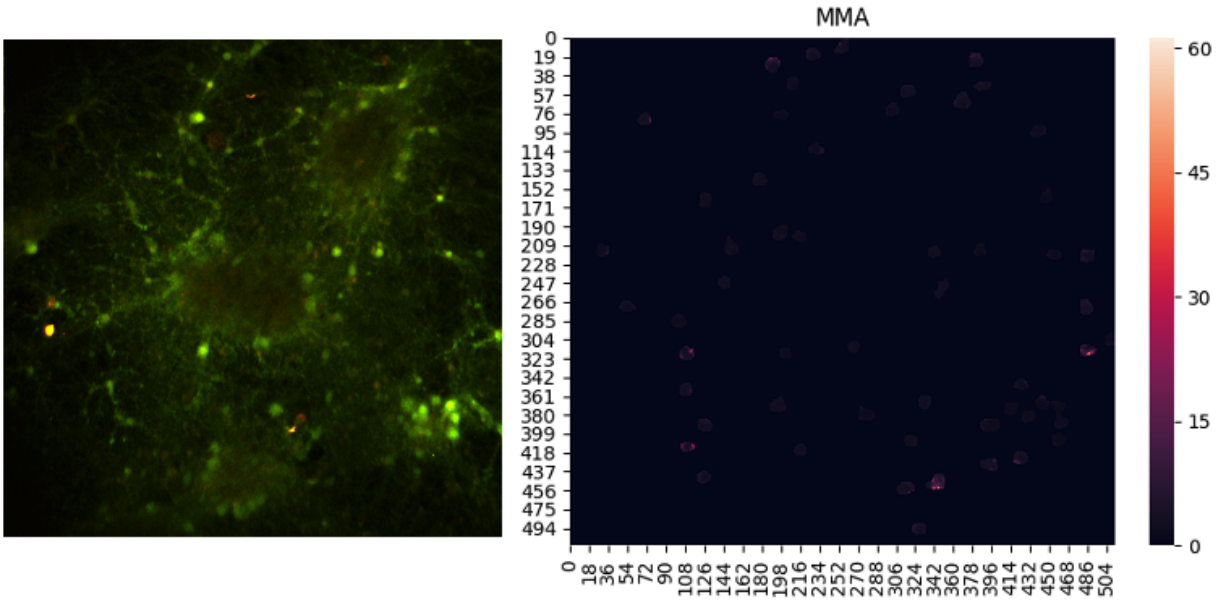

Figure 4-5: **Heatmap of Pixelwise Signal MMA-values for Detected Contours.** 3D image on right. Heatmap of detected contours on left.

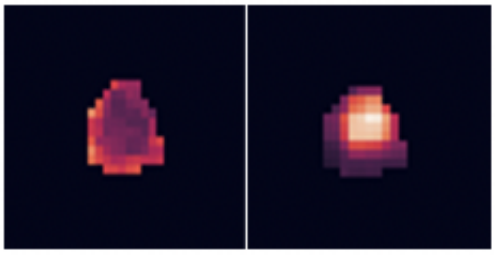

Figure 4-6: **Heatmaps for Single Contour.**. MMA-based heatmap on left. Variance-based heatmap on right.

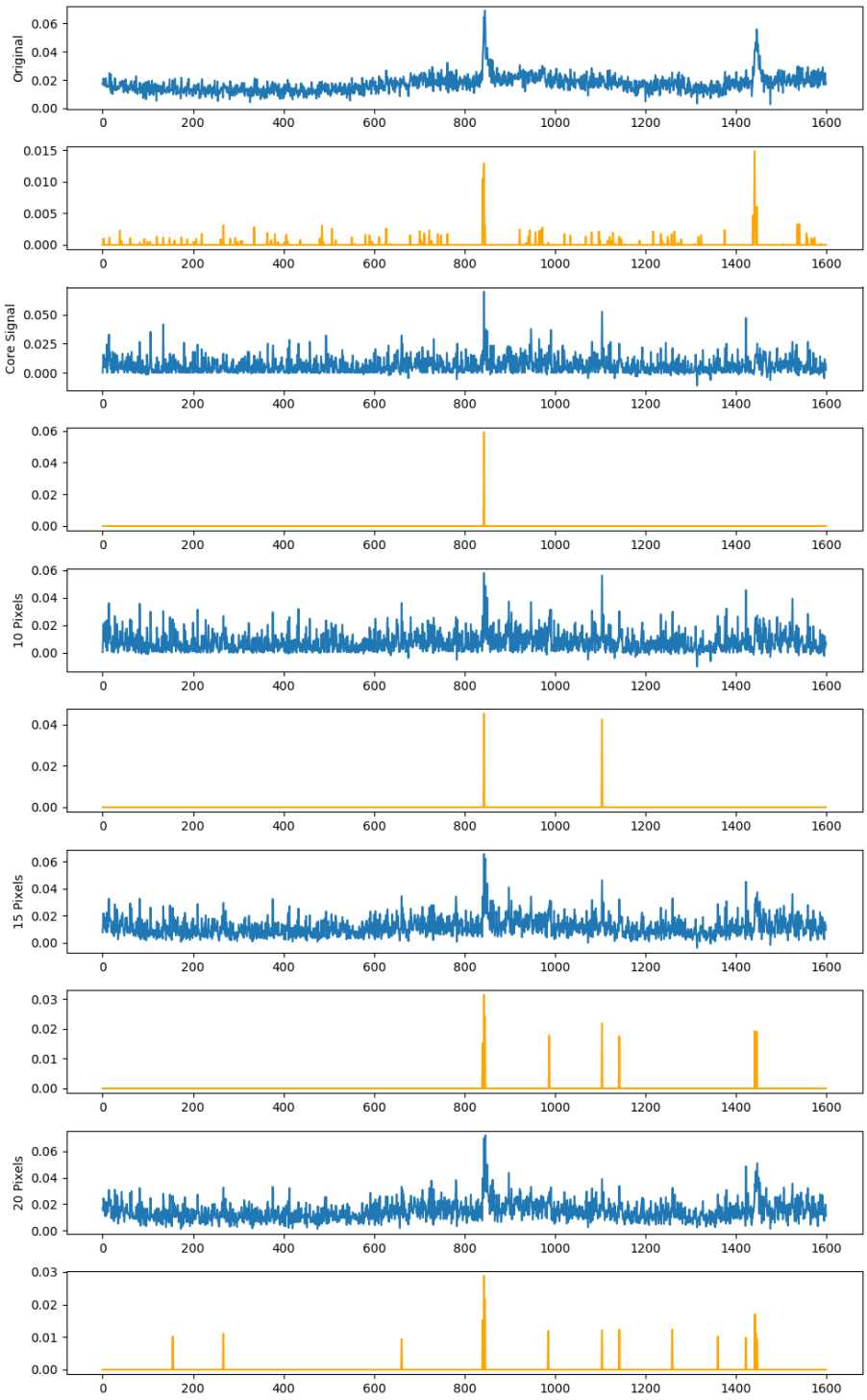

Figure 4-7: **Signals Extracted using MMA and Cosine Correlation and the Corresponding Burst Trains.** The topmost signal was extracted from all pixels in the contour. The second from the top is the core cluster signal consisting of five pixels. The next signal is the final cluster signal capped at 10 pixels. The following pixel is the final cluster signal capped at 15 pixels. The last signal is the final cluster signal capped at 20 pixels. Each signal is shown in blue followed by its burst train in yellow.

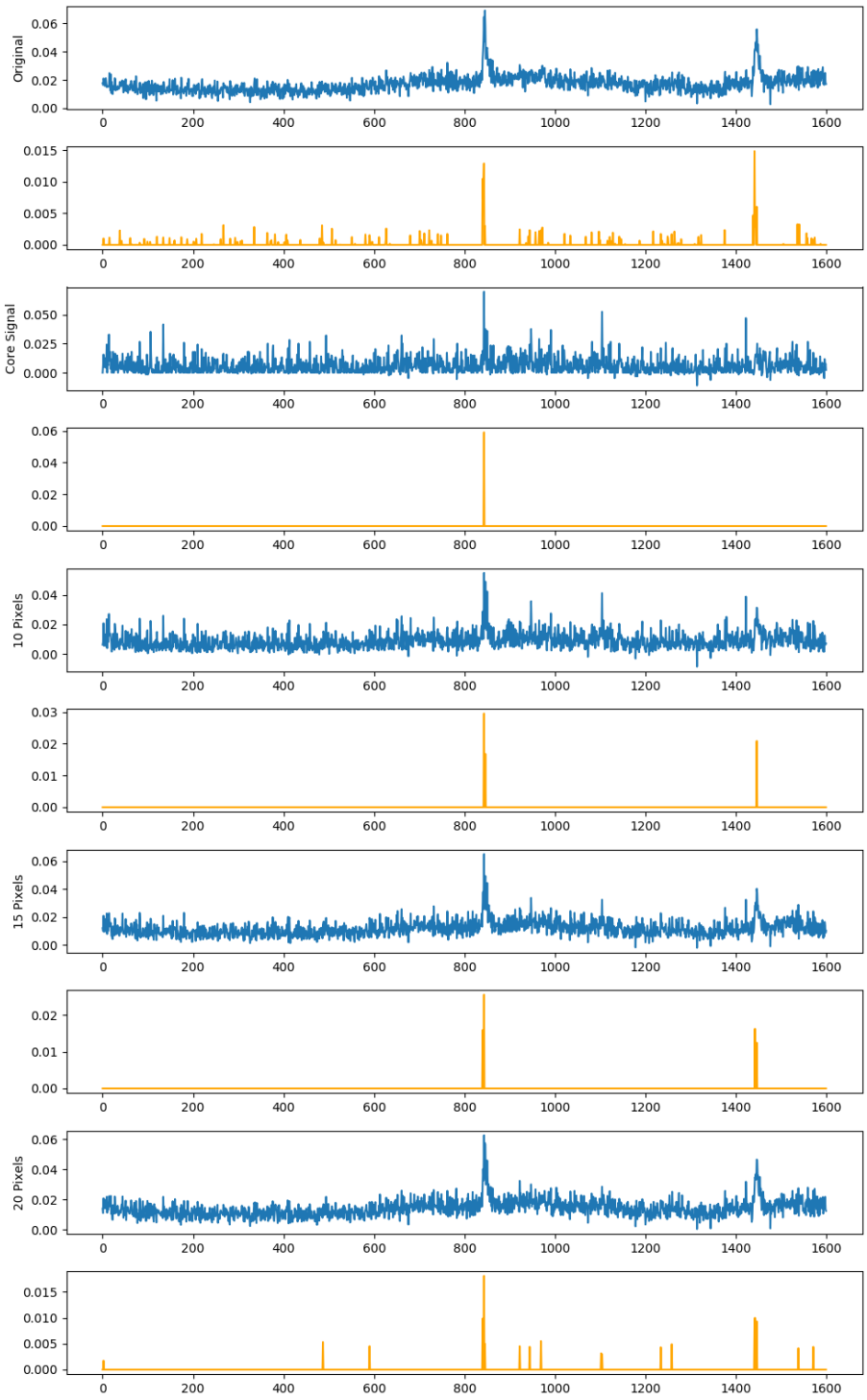

Figure 4-8: **Signals Extracted using MMA and OASIS Correlation and the Corresponding Burst Trains.** The topmost signal was extracted from all pixels in the contour. The second from the top is the core cluster signal consisting of five pixels. The next signal is the final cluster signal capped at 10 pixels. The following pixel is the final cluster signal capped at 15 pixels. The last signal is the final cluster signal capped at 20 pixels. Each signal is shown in blue is followed by its burst train in yellow.

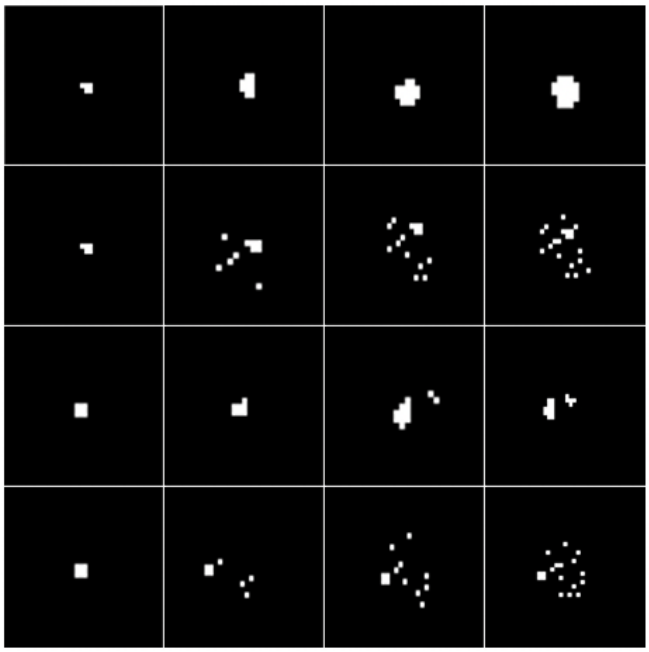

Figure 4-9: **Example of Final Cluster Pixels from Different Extraction Methods.** Left to right, the columns show increasing number of final cluster pixels: 5 pixels, 10 pixels, 15 pixels, 20 pixels. From top to bottom, the rows show different signal extraction methods used: variance with cosine correlation, variance with OASIS correlation, MMA with cosine correlation, MMA with OASIS correlation.

4.4 Slow Oscillation and Calcium-Drift Removal

Once we extract the final signal from each contour, we remove both the calcium drift and any slow oscillations from the signal. Calcium drift is the gradual directional shift of the signal. Slow oscillations may correspond to coordinated activity of neurons which alternate between active (up) states and silent (down) states [27]. Subtracting slow oscillations allows for signals to generally fluctuate about a fixed value. We believe that removal of slow oscillations allows for better burst activity detection in the next step of the pipeline. The methods for both calcium drift removal and slow oscillation removal were provided by Dr. Mierau's group. To find calcium drift, a Savitzky-Golay (savgol) filter with window length set to one less than the number of frames in the signal is fit to the raw signal to. This drift is then subtracted from the raw signal. Next, a Fourier transform is used to find the frequency of the slow oscillation. From this, we can determine whether slow oscillations exist in the signal. If it does, the frequency of oscillation is found using the Fourier transform and calculating the spectrum. A savgol filter of proper window-length based on this frequency is fit to the raw signal with calcium drift removed. An example of this calcium drift and slow oscillation removal can be seen in Figure 4-10. We evaluate burst activity detection on signals extracted using the methods in the next chapter.

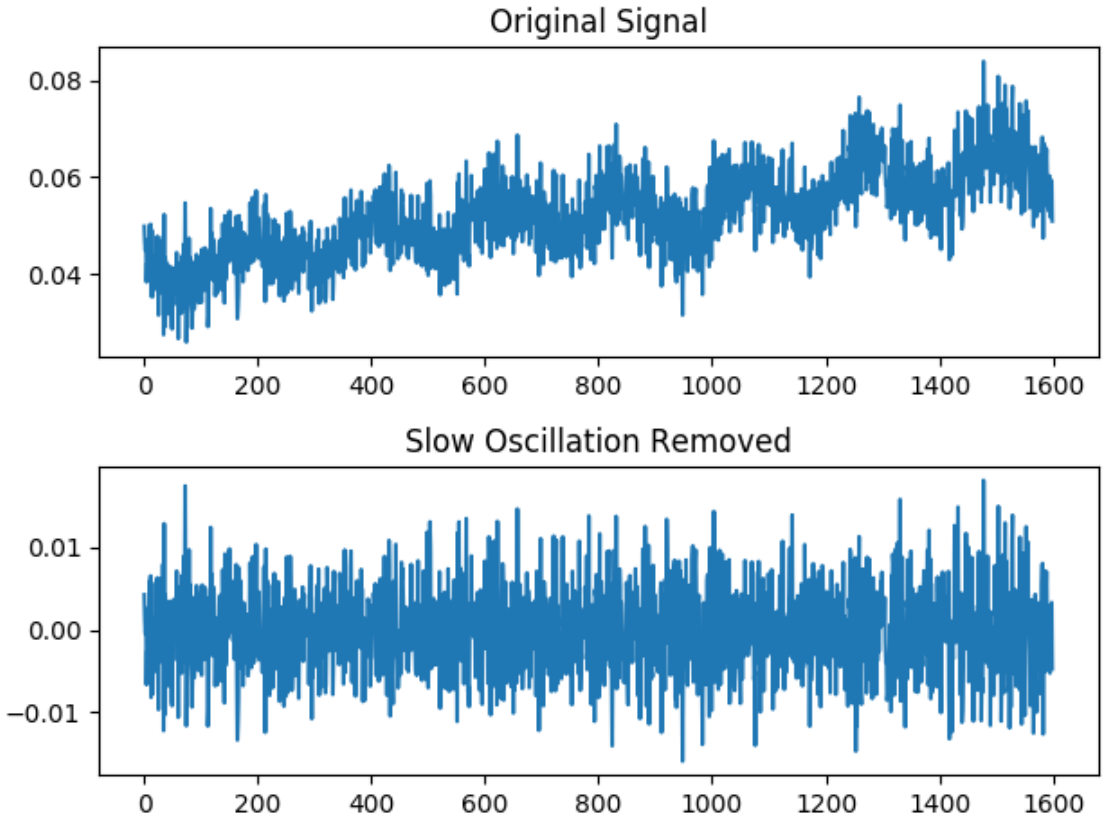

Figure 4-10: **Example of Slow Oscillation and Calcium Drift Removal.** The original signal is shown on top and the signal with calcium drift removed is shown below.

Chapter 5

Burst Activity Detection

5.1 OASIS Method for Deconvolution

While changes in fluorescence expression over time can be used to observe burst activity in a neuronal network, extracting the exact times of burst activity in the signals is a nontrivial problem. Luckily, [28] reports the top-performing algorithms in the *spikefinder* challenge. The authors of [28, p.3] state that they organized the *spikefinder* challenge to "crowd-source the problem of developing algorithms from inferring spike rates from calcium signals" found from two-photon calcium imaging. [29] notes calcium deconvolution using "Online Active Set method to Infer Spikes (OASIS)" as a top-performing method [17]. Therefore, we detect burst activity in extracted signals using the method presented in [17], which we refer to as OASIS. The OASIS method is a "fast online active set method to solve [the] problem of sparse non-negative deconvolution" for "estimation of neural activity" [17, p.1]. The algorithm is "a generalization of pool adjacent violators algorithm (PAVA) for isotonic regression" [17, p.1].

The extracted signals are raw signals that represent a time series of measured fluorescence values. However, it is common practice to baseline signals and consider the relative fluorescence at each time point, as this is often considered biologically more relevant [2]. While we explored different baselines for creating relative fluorescence signals, we ultimately thought burst activity appeared more obscured in the relative

fluorescence signals. Therefore, we decided to use the raw signals with calcium drift and slow oscillation removed as the final signals passed to OASIS. We thought that this would remove bias of the OASIS algorithm towards rising phases of oscillation and eliminate noise.

An example of burst activity detection using OASIS can be seen in Figure 5-1.

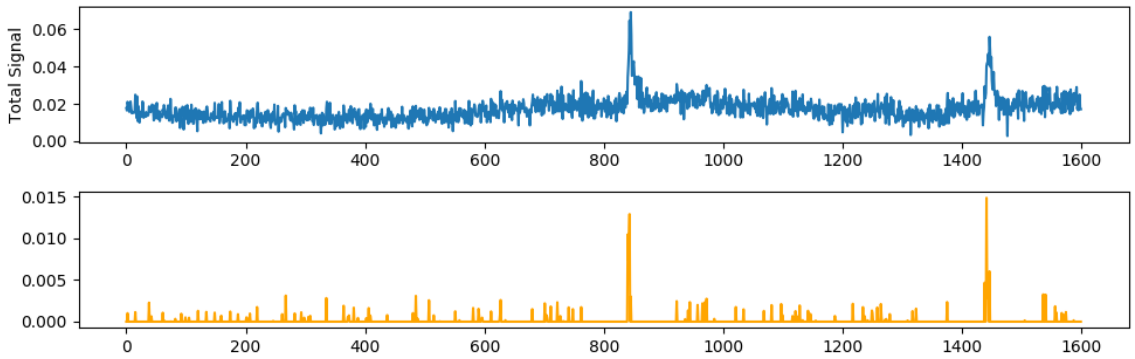

Figure 5-1: **Example of Burst Activity Detection using OASIS.** The signal are shown on top in blue and the associated burst trains found using OASIS is shown below in yellow.

5.2 Results and Discussion

The annotated burst activity data set provides start and end frames for identified bursts. Bursts from different neurons are annotated separately. The neurons in this data set are labeled with numbers corresponding to those in the annotated RoI data set. Because we extract signals from neurons detected using our neuron detection methods, we do not know for sure which neuron labels in the burst activity data set align with the automatically detected neurons. However, we manually matched detected neurons to labeled neurons for a single 3D image. We used a Mask R-CNN model for contour detection taken from the leave-one-out cross-validation, where the test image for the model corresponds to the imaging sequence chosen for evaluating burst detection. We extract signals from each detected contour using the different methods discussed in Chapter 4. We also extract a signal from all pixels in the contour, or the "total signal". We then detect bursts for these signals using OASIS.

OASIS gives a list of nonnegative values for each point in the signal as a burst train. However, many of these values can be very small and it is not clear what value to use as a threshold for considering burst activity events in our data. Therefore, we consider two different thresholds. We evaluated our methods using two different thresholds: 0.03 and 0.005. For every annotated burst interval, we check whether a detected burst index falls within the interval. We do not repeat detected burst indices for different annotated burst intervals. We calculate the proportion of annotated burst intervals for which a detected burst falls within the interval. We calculate this value for every neuron possible in the sequence. For the sequence we evaluated, seven neurons had associated annotations from all of the neuron cell bodies that we manually matched. The average values of this evaluation are shown in Table 5.1 and 5.2. For all evaluations using a heuristic filter, we use a final cluster capped at 15 pixels and a connected core cluster.

The method in [4] returns detected bursts activity events as time intervals. When evaluating this method, we considered only the start values of each interval for evaluation. The rest of the evaluation is the same as above. Figures 5-2 through 5-7 show the results of bursts detected from signals found through the different methods described previously.

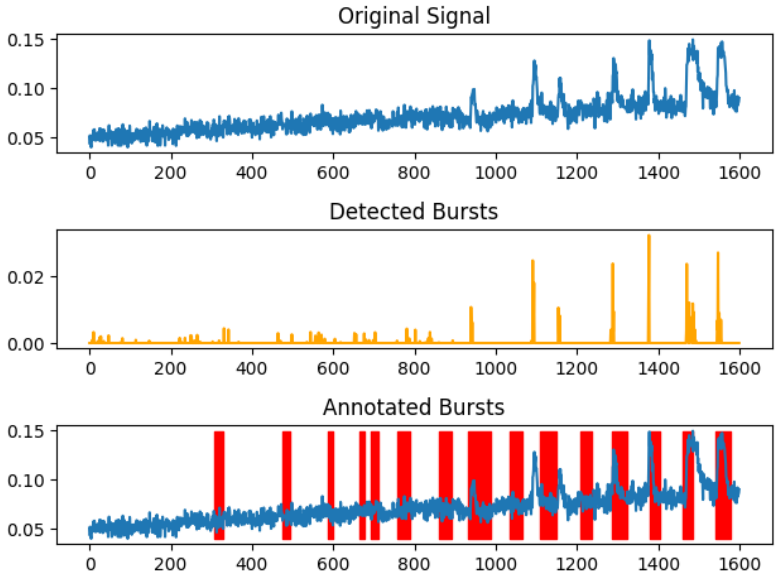

Figure 5-2: **Detected Bursts for Total Signal.** The total signal is shown on top and the corresponding burst train is shown below. The red regions in the bottom plot show the annotated burst intervals.

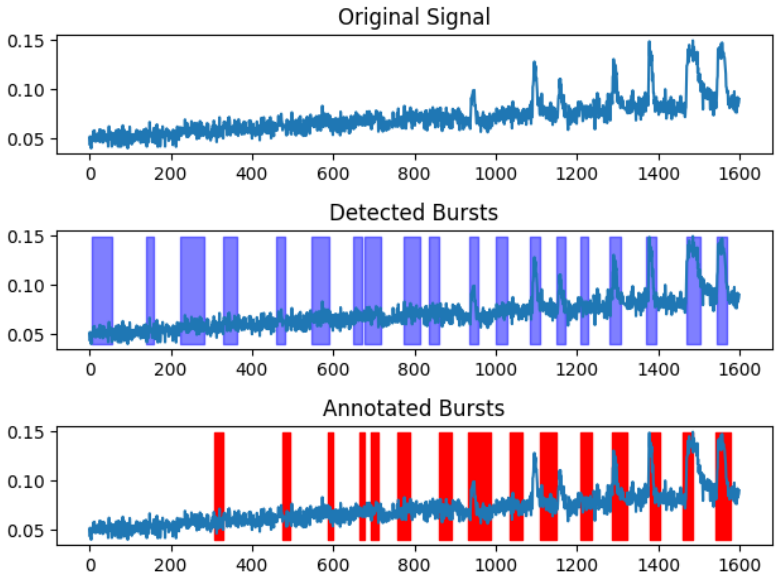

Figure 5-3: **Detected Bursts using Method from Previous Pipeline.** The total signal is shown in top the topmost plot. The purple regions in the middle plot depict the detected bursts. The red regions in the bottom plot show the annotated burst intervals.

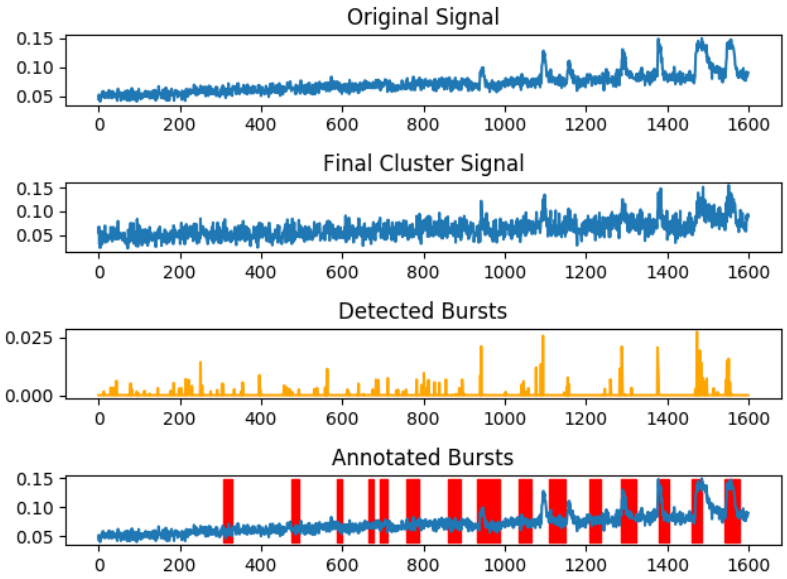

Figure 5-4: **Detected Bursts for MMA and Cosine Correlation Signal Extraction.** The total signal is shown on top. The final cluster signal is shown in the middle and the associated burst train is shown right below in yellow. The red regions in the bottom plot show the annotated burst intervals.

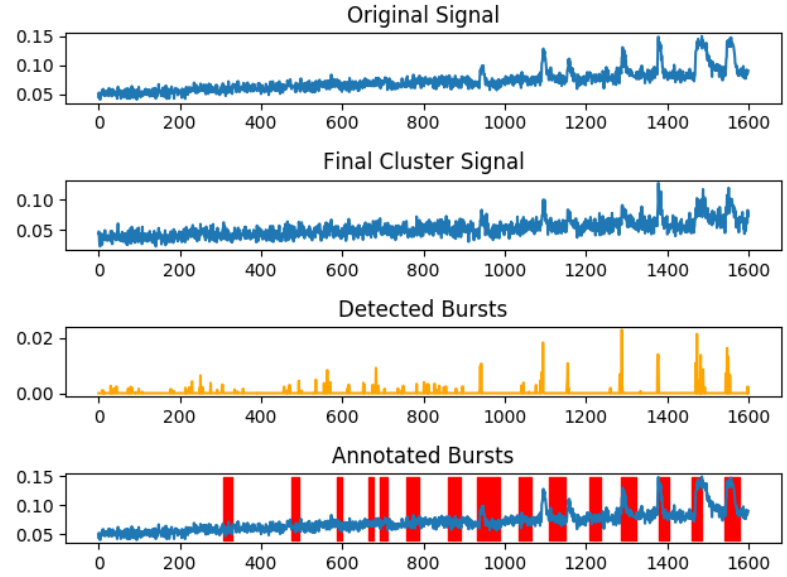

Figure 5-5: **Detected Bursts for MMA and OASIS Correlation Signal Extraction.** The original signal is shown on top. The final cluster signal is shown in the middle and the associated burst train is shown below. The red regions in the bottom plot show the annotated burst intervals.

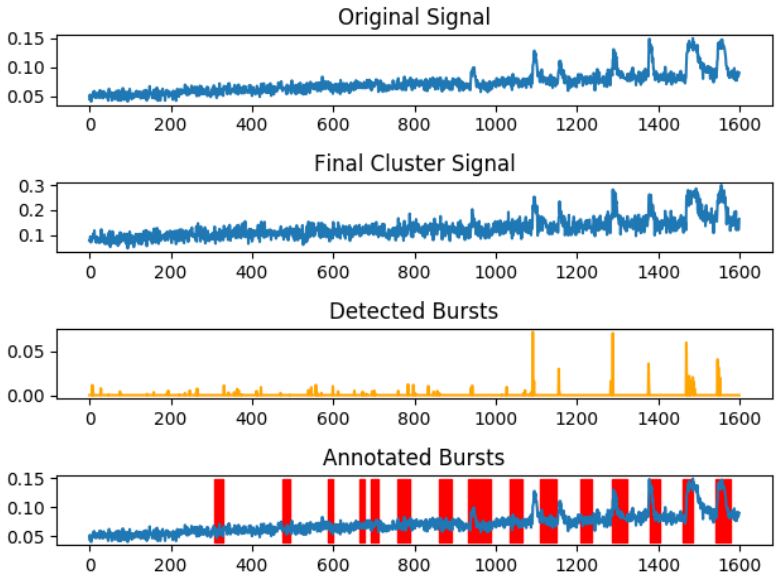

Figure 5-6: **Detected Bursts for Variance and Cosine Correlation Signal Extraction.** The total signal is shown on top. The final cluster signal is shown in the middle and the associated burst train is shown right below in yellow. The red regions in the bottom plot show the annotated burst intervals.

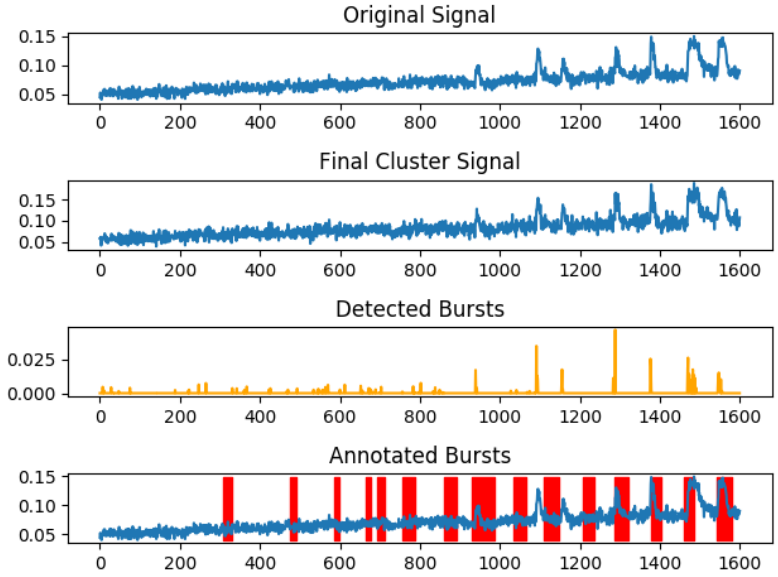

Figure 5-7: **Detected Bursts for Variance and OASIS Correlation Signal Extraction.** The original signal is shown on top. The final cluster signal is shown in the middle and the associated burst train is shown below. The red regions in the bottom plot show the annotated burst intervals.

Previous Method	Total Signal	Variance Cosine	Variance OASIS	MMA Cosine	MMA OASIS
$\mu = 33.8\%$	$\mu = 97.1\%$	$\mu = 100.0\%$	$\mu = 100.0\%$	$\mu = 99.0\%$	$\mu = 83.8\%$
$\sigma = 11.5\%$	$\sigma = 7.0\%$	$\sigma = 0.0\%$	$\sigma = 0.0\%$	$\sigma = 2.3\%$	$\sigma = 34.5\%$

Table 5.1: **Evaluation of Burst Detecting Over Different Signal Extraction Methods (Threshold = 0.03)**. The table shows the results of burst activity detection using signals extracted from the different methods and the total signal using 0.03 as the burst train threshold. The results of burst activity detection using the method from the previous pipeline are also shown.

Previous Method	Total Signal	Variance Cosine	Variance OASIS	MMA Cosine	MMA OASIS
$\mu = 33.8\%$	$\mu = 100.0\%$	$\mu = 100.0\%$	$\mu = 100.0\%$	$\mu = 100.0\%$	$\mu = 100.0\%$
$\sigma = 11.5\%$	$\sigma = 0.0\%$	$\sigma = 0.0\%$	$\sigma = 0.0\%$	$\sigma = 0.0\%$	$\sigma = 0.0\%$

Table 5.2: **Evaluation of Burst Detecting Over Different Signal Extraction Methods (Threshold = 0.005)**. The table shows the results of burst activity detection using signals extracted from the different methods and the total signal using 0.005 as the burst train threshold. The results burst activity detection using the method from the previous pipeline are also shown.

The results in Table 5.1 and 5.2 indicate using OASIS deconvolution to detect bursts outperforms the previous burst detection method. Furthermore, these results indicate that the signal extraction methods using variance outperform the remaining methods when with a burst train threshold of 0.03. However, they also show that changing the burst train threshold affects the performance of the methods. Therefore, we think that fine-tuning the threshold value used for burst evaluation may result in better evaluation of our methods. Additionally, the results shown are calculated for only seven neuronal signals in a single sequence. Evaluating more neuronal signals across more sequences may also give a better understanding of the performance of these methods. However, given that there seemed to be better signal-to-noise ratio in many examples using these methods, we think that these methods are promising.

Furthermore, because we do not know exactly which annotated bursts are associated with a particular signal from an automatically detected neuron without manual

matching, we provide the ability to iterate through and view each annotation option. Figure 5-8 shows an example of this.

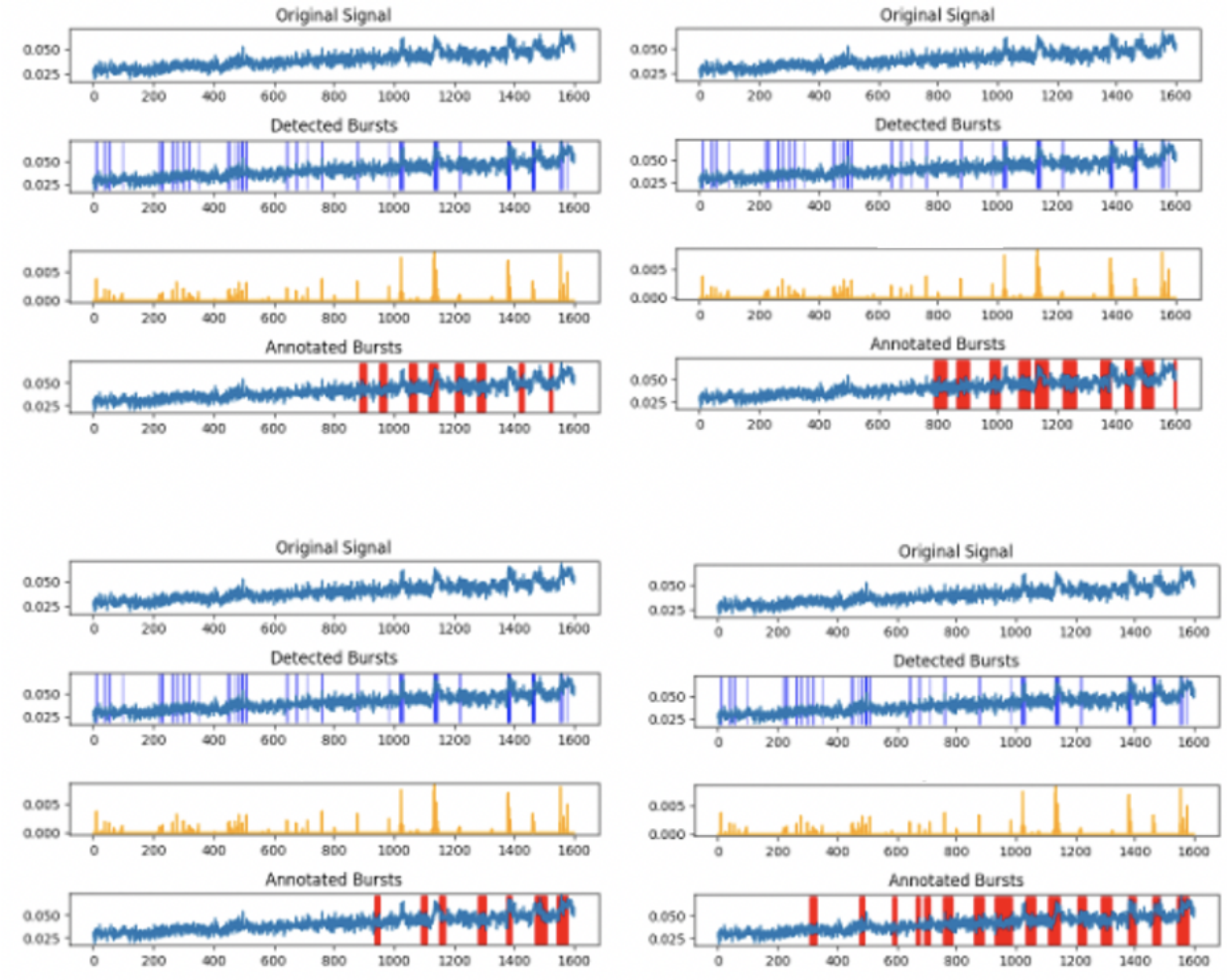

Figure 5-8: **Annotated Burst Intervals for Different Neurons of Same Sequence.** Each plot shows a different burst interval annotation found from separate neurons in the same sequence (annotated intervals shown in red).

Chapter 6

Conclusion and Future Work

In this work we have improved the existing pipeline for analyzing two-photon calcium imaging sequences. Specifically, we introduced new methods for neuron detection, signal extraction, and burst activity detection. The incorporation of these methods in the pipeline may increase researchers' ability to efficiently analyze two-photon calcium imaging sequences.

6.1 Future Work

6.1.1 Neuron Detection Improvements

Future work can improve the neuron detection methods by training with more images and creating masks that are truer to the neuron cell body regions. We think that better masks can significantly improve the performance of the methods using Mask R-CNN.

Given the training overhead for the Mask R-CNN model and the performance of the unsupervised models, fine-tuning the parameters of the unsupervised models may also result in better neuron detection.

6.1.2 Signal Extraction and Burst Activity Detection Improvements

As noted in the paper, we currently do not have a set threshold for adding pixels to the main contour. Additionally, it is unclear which threshold value to use when considering bursts in the burst train found by OASIS. Evaluating burst activity detection using more signals may help determine good thresholds both for the number of pixels in the final cluster and considering burst activity events. Further evaluation may also help determine which extraction method, if any, has the best performance.

Furthermore, it may be that using different parameters for the OASIS deconvolution algorithm would be better suited for this data set. Further investigation of these parameter values may produce better burst activity detection results.

Additionally, it may be possible to forgo the neuron detection step entirely by calculating variance and/or MMA values for every pixel, rather than just the ones assigned to detected neuron cell bodies. While we found this to take too long in terms of run time, future work can explore faster alternatives. Signals can then be directly extracted from the pixels whose pixel signals have the greatest values.

Future work can also investigate ways to combine information from the variance and MMA values assigned to pixels to create a final cluster instead of being restricted to a single method.

6.2 Clinical Implications

By detecting the number and timing of burst activity events from each neuron in a time-lapse sequence, the neurons which are more active can be found. Groups of neurons which are synchronized can be identified. From this information, neuroscientists can create graphical representations of the neuronal networks for analysis. These plots display information such as the number of active cells in the network and the number and strength of connections per cell. From these representations, they are able to find differences in development between the wild-type and Mecp2-deficient

mice. Identifying these differences can help us better understand neuronal network development and accelerate novel treatments to regulate network development.

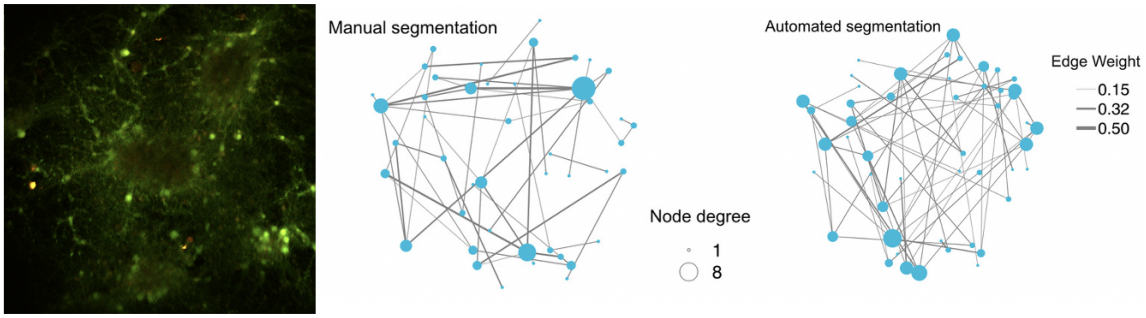

Figure 6-1: **Example of Network Analysis Plot.**

Bibliography

[1] S. Mierau, A. Patrizi, T. Hensch, and M. Fagiolini, "Cell-specific regulation of n-methyl-d-asparate receptor maturation by mecp2 in cortical circuits," *Biological Psychiatry*, 2016.

[2] J. P., "Network development and its dysfunction in a genetic model of autism," Master's thesis, University of Cambridge, July 2017.

[3] R. Amir, I. Van den Veyver, M. Wan, C. Tran, U. Francke, and H. Zoghbi, "Rett syndrome is caused by a mutation in x-linked mecp2, encoding methyl-cpg-binding protein 2," *Nature Genetics*, vol. 23, pp. 185–8, 1999.

[4] R.-E. R. Khouri, "Two-photon calcium imaging sequence analysis: A method for analyzing neuronal network activity," Master's thesis, Massachusetts Institute of Technology, June 2018.

[5] K. Svoboda and R. Yasuda, "Principles of two-photon excitation microscopy and its applications to neuroscience," *Neuron*, vol. 50, p. 823–839, Jun 2006.

[6] J. Lichtman and J. Conchello, "Fluorescence microscopy," *Nature methods*, vol. 2, pp. 910–9, 01 2006.

[7] R. Haugland, *Handbook of Biological Fluorescent Probes and Research Chemicals*. 6 ed., 1996.

[8] W. Denk, K. Delaney, A. Gelperin, D. Kleinfeld, B. Strowbridge, D. Tank, and R. Yuste, "Anatomical and functional imaging of neurons using 2-photon laser scanning microscopy," *Journal of neuroscience methods*, vol. 54, pp. 151–62, 11 1994.

[9] C. Grienberger and A. Konnerth, "Imaging calcium in neurons," *Neuron*, vol. 73, no. 5, p. 862–885, 2012.

[10] J.-A. Conchello and J. W. Lichtman, "Optical sectioning microscopy," *Nature Methods*, vol. 2, no. 12, p. 920–931, 2005.

[11] S. Cohen, H. Gabel, M. Hemberg, A. Hutchinson, L. Sadacca, D. Ebert, D. Harmin, R. Greenberg, V. Verdine, Z. Zhou, W. Wetsel, A. West, and M. Greenberg, "Genome-wide activity-dependent mecp2 phosphorylation regulates nervous system development and function," *Neuron*, pp. 72–85, 2011.

[12] J. Guy, B. Hendrich, M. Holmes, J. Martin, and A. Bird, "A mouse mecp2-null mutation causes neurological symptoms that mimic rett syndrome," *Nature Genetics*, pp. 322–6, 04 2001.

[13] S. Durand, A. Patrizi, L. Quast, K.B. Hachigan, R. Pavlyuk, A. Saxena, P. Carninci, T. K. Hensch, and M. Fagiolini, "Nmda receptor regulation prevents regression of visual cortical function in the abscence of mecp2," *Neuron*, vol. 76, pp. 1078–1090, 2012.

[14] V. Dani, Q. Chang, A. Maffei, G. Turrigiano, R. Jaenisch, and S. Nelson, "Reduced cortical activity due to a shift in the balance between excitation and inhibition in a mouse model of rett syndrome," *Proceedings of the National Academy of Sciences*, vol. 102, pp. 12560–12565, 2005.

[15] H.-T. Chao, H. Zoghbi, and C. Rosenmund, "Mecp2 controls excitatory synaptic strength by regulating glutamatergic synapse number," *Neuron*, vol. 56, pp. 58–65, 2007.

[16] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *CoRR*, vol. abs/1703.06870, 2017. `http://arxiv.org/abs/1703.06870`.

[17] F. J, Z. P, and P. L, "Fast online deconvolution of calcium imaging data," *PLoS Comput Biol*, vol. 13, 2017. `https://doi.org/10.1371/journal.pcbi.1005423`.

[18] `http://neurofinder.codeneuro.org/` (accessed June 2020).

[19] J. W. Johnson, "Adapting mask-rcnn for automatic nucleus segmentation," *CoRR*, 2018. `http://arxiv.org/abs/1805.00500`.

[20] "Opencv: Opencv-python tutorials." `OpenCV:OpenCV-PythonTutorials`.

[21] "sklearn.cluster.birch." `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html`.

[22] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," 2017. `https://github.com/matterport/Mask_RCNN`.

[23] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," *CoRR*, 2014. `https://arxiv.org/abs/1405.0312`.

[24] "R: Wilcoxon rank sum and signed rank tests." `https://stat.ethz.ch/R-manual/R-devel/library/stats/html/wilcox.test.html`.

[25] "Mann–whitney u test - wikipedia." `https://en.wikipedia.org/wiki/Mann-Whitney_U_test` (accessed June 2020).

[26] T.-W. Chen and *et al.*, "Ultrasensitive fluorescent proteins for imaging neuronal activity," *Nature*, vol. 499, 2013.

[27] Sanchez-Vives and M. Victoria, "Slow oscillations: Physiology," *Encyclopedia of Computational Neuroscience*, pp. 1–7, 2013. https://doi.org/10.1007/978-1-4614-7320-6_308-1.

[28] P. Berens, J. Freeman, T. Deneux, N. Chenkov, T. McColgan, A. Speiser, and et al., "Community-based benchmarking improves spike rate inference from two-photon calcium imaging data," *PLOS Computational Biology*, vol. 14. https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006157.

[29] P. Berens, J. Freeman, T. Deneux, N. Chenkov, T. McColgan, A. Speiser, J. H. Macke, S. C. Turaga, P. Mineault, P. Rupprecht, S. Gerhard, R. W. Friedrich, J. Friedrich, L. Paninski, M. Pachitariu, K. D. Harris, B. Bolte, T. A. Machado, D. Ringach, J. Stone, L. E. Rogerson, N. J. Sofroniew, J. Reimer, E. Froudarakis, T. Euler, M. Román Rosón, L. Theis, A. S. Tolias, and M. Bethge, "Community-based benchmarking improves spike rate inference from two-photon calcium imaging data," *bioRxiv*, 2018. https://www.biorxiv.org/content/early/2018/02/26/177956.