Imperial College London

MSc Computing

(Artificial Intelligence and Machine Learning)

Department of Computing

MSc Thesis

DEEP LEARNING FOR IMAGE-BASED PROFILING IN DRUG DISCOVERY

AUTHOR: ACER BLAKE

SUPERVISORS: PROFESSOR BEN GLOCKER,
DR BEN JONES

SECOND MARKER: PROFESSOR ABHIJEET GHOSH

September 2023

Abstract

Image-based profiling, the task of extracting rich biological data from detailed microscopic imaging acquisitions, has emerged in recent years as a powerful tool for drug discovery. In particular, localising and segmenting out cells from fluorescent imaging experiments allows researchers to compute the drug response profiles of individual cells over time. At present, the biomedical imaging software community has yet to leverage state-of-the-art foundation models for cellular detection and segmentation. To this end, the aims of the following thesis are tripartite. The first is the development of FRETLab, a machine-learning enabled image analysis platform for fluorescence microscopy. The second is an exploration of classical and deep unsupervised clustering for drug response time series analysis. Lastly, the third and final part consists of an investigation into the potential of few-shot dataset labelling to enable generalisable and low-resource cellular detection and segmentation for biologists without expertise in machine learning.

Acknowledgements

To my primary supervisor, Professor Ben Glocker, I would like to thank you for agreeing to my self-proposal and allowing me to undertake a project in drug discovery, an area I am deeply passionate about, and intend to pursue for my future career. I would also like to thank you for your advice and supervision over the course of the project. To my secondary supervisor, Dr Ben Jones, I would like to thank you for helping a non-biologist understand the nuances of fluorescence microscopy, drug discovery, and biology more generally. Developing domain knowledge in drug discovery, fluorescent imaging, and the neurobiology of metabolism has helped the project along greatly. Finally, to my second marker, Professor Abhijeet Ghosh, I would like to extend my thanks for your feedback on my initial report, and for collaborating in marking my thesis.

Contents

Chapter 1

Introduction

The development of machine learning models for the task of drug discovery has been identified by the World Health Organisation (WHO) as one of the most promising applications of artificial intelligence to modern medicine. In their first global report on artificial intelligence for healthcare [1] the WHO note that over the next two decades 'the role of AI could evolve,' and that by 2040, 'testing of medicines might be virtual – without animals or humans – based on computer models of the human body'. Chief among the many subtasks identified by the report as being critical to the automatic drug discovery pipeline is image analysis, particularly 'structuring unorganized data from medical imaging'. In the context of drug discovery, unlabelled data abounds in the microscopic imaging domain, spanning time-series acquisitions of cell responses to single-cell image tracking. This abundance of data coupled with the complex dynamics of cellular interactions makes microscopic imaging a key candidate for deep learning.

The recent successes of deep learning models at executing imaging tasks such as cell-counting, tracking, and dynamical modelling has significantly accelerated the pace of drug discovery already [2]. However, there remain three primary unsolved problems. The first is the practical issue of democratizing deep learning tools, namely that few open source software tools for deep-learning based microscopic analysis exist, and those that do are often inaccessible to biologists. While recent efforts have attempted to address this [3], barriers still remain. Another more theoretical problem is that of determining precisely how much information can be gleaned about a cell's reaction to a drug from images of its morphology. Of particular importance is the question as to whether models trained on imaging data alone are viable for making sufficiently accurate predictions, or if only rich, multi-modal genomic, chemical, and bioinformatic amalgams are capable of such a feat. At present, a standard approach to categorizing cellular responses entails imaging the fluorescent activity within each cell at regular intervals and extracting a time series from this [4]. Here there is a clear opportunity for unsupervised clustering on drug reactivity time series, which could prove useful in identifying meaningful response profiles. The third challenge is that of generalisation in low resource settings; where accessible methods exist for achieving state-of-art performance on cellular imaging tasks, these often require large datasets of images labelled down to the individual pixel level.

With the preceding challenges in view, the aims of the present project are three-fold: to automate the fluorescence microscopy pipeline currently in use by the Jones lab within Imperial's Department of Metabolism [5]; to determine whether classical and deep unsupervised time series clustering can be used to separate out cellular responses to one of a number of drugs; and finally to investigate the potential of an emerging foundation model as an aid for few-shot automatic dataset labelling.

Concerning the first of these tasks, automating the Jones lab's image analysis pipeline, the primary motivation is to accelerate the turnaround between fluorescent imaging acquisitions and the analysis of the data obtained. Further, pipeline automation has the effect of standardising the workflow by removing artefacts introduced by the imaging modality, as well obviating human biases which often contribute to errors or inconsistencies in quantitative measurements and datasets [6]. Removing fluorescence imaging artefacts such as chromatic aberration, drift, and vignetting is a well-studied area with a vast body of literature and open-source software behind it. Therefore, the focus here will be the task of eliminating human involvement and biases in image analysis, which remains an open problem. A common example is cellular segmentation, a preliminary step for computing responses to drugs in which the precise pixels belonging to each cell in an image of a given culture are annotated. This is often achieved by thresholds that are manually tuned by users. Segmentation will therefore be at the core of this part of the present report. In particular, herein will be developed an accessible, Python-based tool for fluorescence microscopy experiments based upon Meta AI's Segment Anything Model (SAM) [7]. Emphasis will be placed on the development of CellSAM, a model fine-tuned to segment out cells from imaging acquisitions and facilitate the determination of cellular reactivity using a fluorescence energy transfer protocol.

The second project objective, drug response analysis, could potentially aid in discovering biological insights as to why certain cells respond to drugs more than others, and serve as a principled method by which biologists can begin their investigations. Here, using the automated pipeline developed as part of the preceding objective, a dataset of univariate drug response time series will be built. Thereafter, classical and deep unsupervised clustering will then be used to group cells into reactivity clusters with a view to being able to determine whether meaningful patterns emerge in how neurons respond over time to various drugs.

Finally, as a tertiary goal, the report will explore the potential of the new paradigm of foundational segmentation models for generalisable, low-resource, and automatic labelling of cell biology datasets. The emphasis here will be on developing a pipeline to fine-tune the output of the Segment Anything Model, and then testing this method on a cellular detection dataset with the aim of being able to perform few-shot labelling.

The three aforementioned objectives will be addressed in Chapters 4, 5, and 6 respectively. Before this, however, it will be instructive to first detail the preliminaries for understanding the project – the subject of Chapter 2, and secondly to survey the related work in these areas, as conducted in Chapter 3. Finally, Chapter 7 will present closing conclusions and suggestions for future work, as well as an assessment of the legal, ethical and social ramifications of the present project.

Chapter 2

Preliminaries

This chapter introduces the necessary preliminaries for understanding the broader context in which the present work is set, the objectives the project will aim to meet, and the work that has been done in relevant areas thus far. This begins with a primer on the neurobiology of metabolism, followed by an introduction to fluorescence microscopy as an imaging medium and its applications to drug discovery. Following this, details of the image analysis pipeline currently in use by the Jones lab will motivate the software developed in Chapter 4. Finally, as machine-learning will be the focus of the methodology used throughout the project, an overview of the current state-of-the-art methods in deep learning for medical imaging will lay the groundwork for future chapters.

2.1 Drug Discovery and the Neurobiology of Metabolism

As noted in Chapter 1, the present project is being undertaken in conjunction with Dr Ben Jones of Imperial College London's Department of Metabolism. The Jones research group is focused primarily on the development of novel drugs to treat diabetes by inducing satiety, and, by extension, weight loss. This is being facilitated through the application of new molecular tools to understand and exploit the biology of G protein-coupled receptors (GPCRs) for therapeutic benefit.

Proteins are complex molecules made of amino acids that are responsible for many functions within cells, from providing immunity against foreign particles to transporting small molecules throughout the body [8]. Membrane receptors, or simply 'receptors', are a class of protein that reside on the outside of a cell's surface and respond to signals by binding to ligands – chemical messengers released by a cell to signal either itself or a different cell [9]. These cell surface receptors act as relays for molecular messages in the form of light energy, peptides [1], lipids [2], sugars, and proteins. Such signals inform cells about the presence or absence of sustenance in the form of light or nutrients within their environment, or they convey information sent by other cells.

[1]Peptides are typically defined as molecules that consist of between 2 and 50 amino acids, whereas proteins are made up of 50 or more amino acids [10].

[2]Organic compounds including fats, oils, hormones, and certain components of membranes [11].

According to the journal Nature, G-protein-coupled receptors (GPCRs) are the 'largest and most diverse group of membrane receptors in eukaryotes' [3] [12]. GPCRs play a role in an vast array of functions in the human body, and increased understanding of these receptors has greatly affected modern medicine, with recent research estimating that 'between one-third and one-half of all marketed drugs act by binding to GPCRs'.

In a recent publication [13] Dr Jones described a new mechanism, termed 'biased signalling', which maximises the action of the glucagon-like peptide 1 receptor (GLP-1R), a particular GPCR that could potentially improve treatment for diabetes. The term 'biased signalling' refers to the fact that different ligands stabilise distinct receptor conformations, leading to engagement with different cell signalling networks. The receptor system can thereby be tuned to increase beneficial effects and minimise side-effects. Following this work, Dr Jones has been funded to uncover the molecular mechanisms behind biased GLP-1R activation, particularly through the investigation of agonists – compounds that bind to a cellular receptor and produce the same action as a naturally produced molecule that ordinarily binds to the receptor [14], in this case GLP-1R.

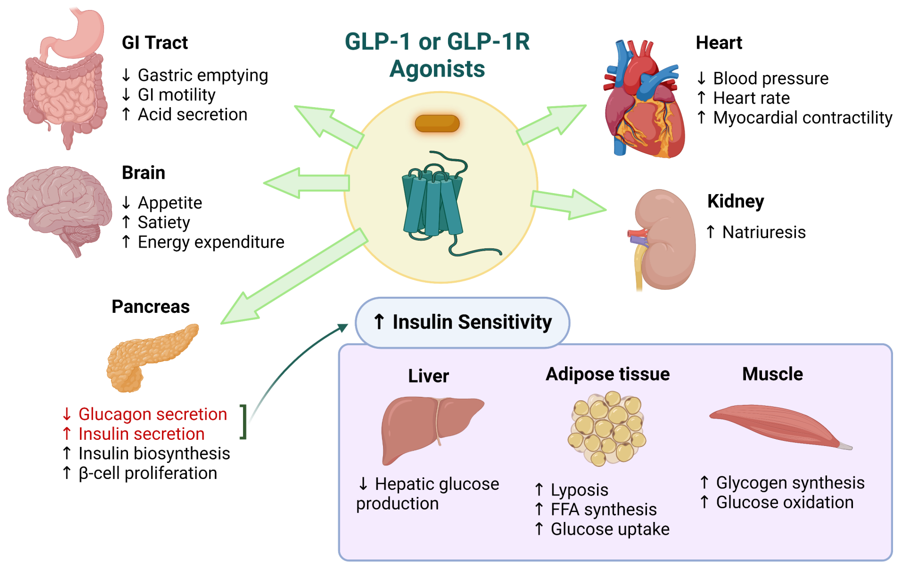


Figure 2.1: A graphic illustrating the effects of GLP-1 or GLP-1R agonists [15] [16].

Glucagon-like peptide-1 (GLP-1) is a hormone that is secreted from the intestine in response to nutrient ingestion. In addition to its well-known effects on insulin secretion and glucose-level maintenance, GLP-1 also plays an important role in the neurobiology of metabolism. GLP-1 receptors are expressed in several regions of the brain, including the hypothalamus, brainstem, and limbic system. Activation of these receptors by GLP-1 can lead to a range of metabolic effects, including increased energy expenditure, decreased food intake, and improved glucose metabolism [17].

[3]Eukaryotes are organisms with well-defined nuclei, often multi-cellular and complex, as opposed to prokaryotes, which have no distinct nucleus and are single-celled.

One of the central mechanisms by which GLP-1 affects metabolism is through its effects on the central nervous system's regulation of food intake. GLP-1 has been shown to act on the hypothalamus, a key brain region involved in the regulation of energy balance, to reduce appetite and increase satiety [18]. In addition to its effects on food intake, GLP-1 also has direct effects on energy expenditure. Studies in rodents have shown that GLP-1 can increase heat production in brown adipose tissue [4], leading to increased energy expenditure [20]. Similar studies in humans have also demonstrated that GLP-1 increases glucose uptake and utilization in skeletal muscle and adipose tissue, leading to improved glucose metabolism [21].

The Jones research group are currently using mice as a model organism for the development of diabetes alleviating drugs in humans. In particular, two molecules are being tested on mouse neurons in vitro: exendin-phe1 (ExF1) and exendin-asp3 (ExD3). Both molecules are modifications of exendin-4, a GLP-1R agonist currently used to treat type II diabetes [22]. Exendin-4 was discovered in 1990 by the endocrinologist Dr. John Eng [23]. His assays [5] revealed that that venom from certain snakes and lizards, including the Gila monster, caused enlargement of the pancreas, where insulin is synthesized. Exendin-4 was discovered to be similar in both structure and function to GLP-1, but only when glucose production is high. While GLP-1 is active in the body for only a few minutes, exendin-4 may remain active for hours, suggesting that it could be a long-lasting and effective diabetes treatment [23].

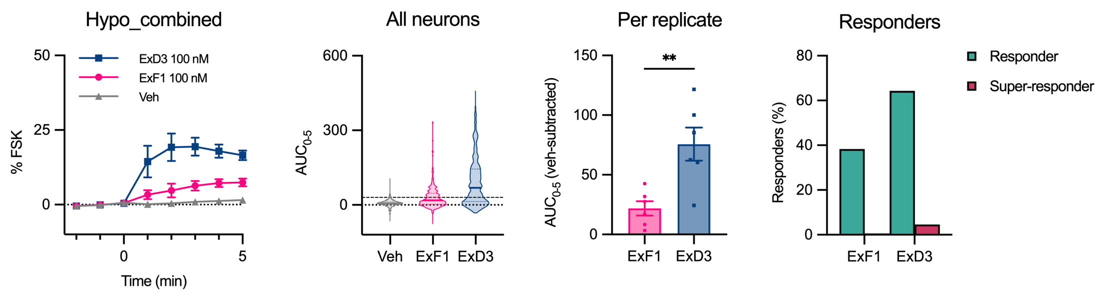


Figure 2.2: Graphs illustrating the reactivity profiles of a culture of neurons. From left to right: (a) Collective culture reactivity as measured by the amount of positive control, forskolin (FSK) present. (b) Area under the curve (AUC) for neuronal response. (c) Box and whisker plot for drug response with the control solution subtracted. (d) Profiles of responders and super-responders. All data is supplied by the Jones lab.

Key to understanding the efficacy and biological underpinnings of ExF1 and ExD3 is determining why some neurons respond more to the molecules than others. As pictured in Figure 2.2, neurons may differ widely in terms of both their peak reactivity and the length of time for which they display a response. To address this question, we must first understand the tools currently being employed by the Jones lab to measure cellular responses, namely fluorescence microscopy and fluorescence resonance energy transfer.

[4] Adipose tissue is connective fatty tissue that extends throughout the human body [19].
[5] An assay is a quantitative test to determine the quantity of a given substance in a particular sample.

2.2 Fluorescence Microscopy

Cellular imaging plays a central role in characterising biased signalling and related phenomena. For this reason, the Jones research group have established several fluorescence and fluorescence resonance energy transfer (FRET) – based assays in order to measure biased signalling. The focus of this project will be on a two-channel ratiometric FRET method in particular. However, before exploring this technique, it will prove instructive to first understand fluorescence microscopy and its applications to drug discovery.

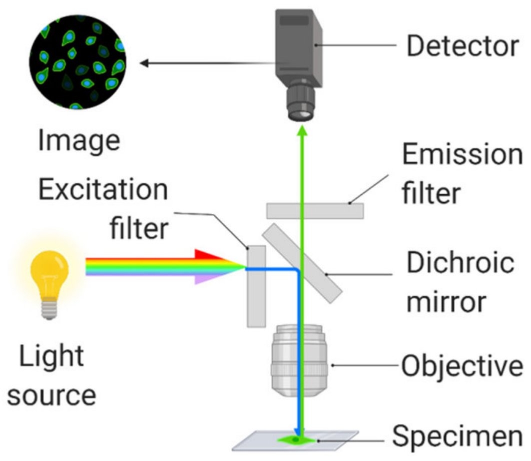


Figure 2.3: A general hardware schematic for fluorescence microscopes [24].

Fluorescence microscopy is a powerful imaging technique that enables the visualization and analysis of biological processes with high spatial and temporal resolution [25]. The technique utilises fluorescent dyes or proteins called fluorophores that emit light when excited by a specific wavelength of light. These fluorescent molecules can be introduced into living cells or tissues, allowing the observation of processes such as protein interactions, signalling events, and cellular transport [25].

As pictured in Figure 2.3, a fluorescent microscope has six major components: a light source, an excitation filter, an emission filter, a dichroic mirror, an objective lens, and a detector. The process of registering a fluorescent image involves several steps: initially, a high-intensity light source, such as a mercury or LED lamp, emits light of a specific wavelength, usually ultraviolet or blue, which serves as the excitation light. An excitation mirror ensures that only the desired excitation wavelength reaches the sample. This excites the fluorophores in the sample, causing them to absorb this energy and emit light at a longer wavelength. To separate the excitation and emission light, specialized optical elements are employed. A dichroic mirror reflects the excitation light towards the sample while allowing the emitted fluorescent light to pass through it. Additionally, an emission filter selectively allows the emitted fluorescent light to pass through, while blocking any residual excitation light. The emitted light then reaches a detector, such as a camera or photomultiplier tube, which captures the image.

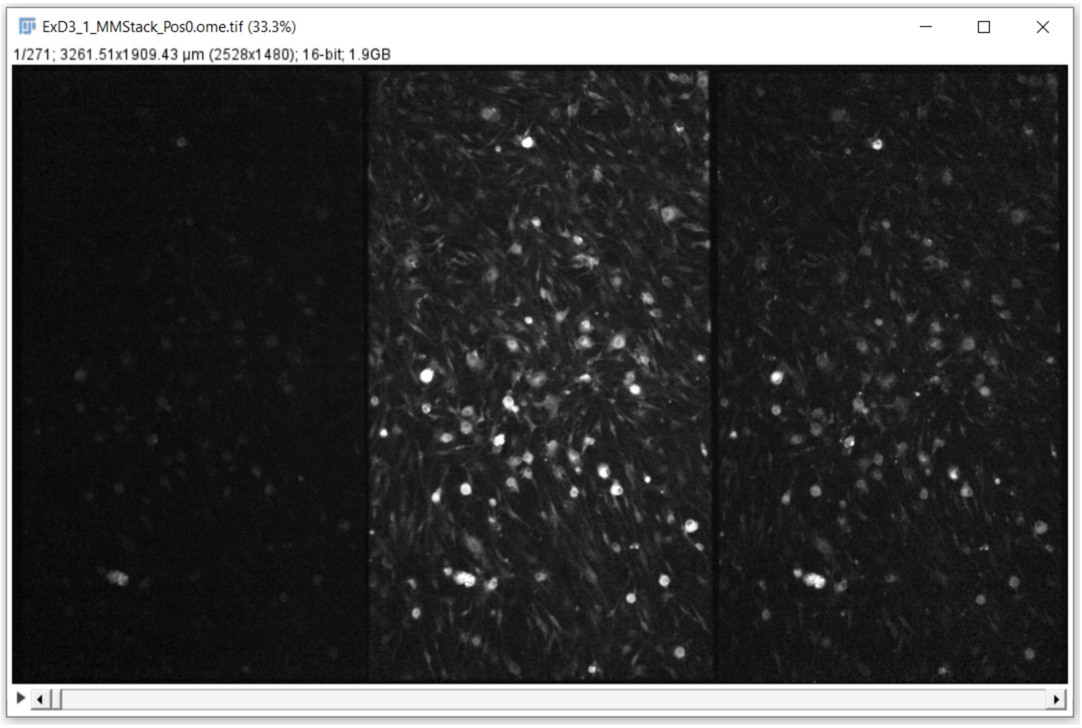

ExD3_1_MMStack_Pos0.ome.tif (33.3%)
1/271; 3261.51x1909.43 µm (2528x1480); 16-bit; 1.9GB

Figure 2.4: An example of an unprocessed, three-channel image of mouse neurons in vitro, taken by the Jones lab. Note that while there are three panes, each corresponding to the same field of view through three separate filters – red, yellow, and cyan, the first channel (leftmost) is unused. Observe also that acquisitions are rendered in grayscale by default. The fluorescent imaging acquisitions taken by the Jones lab take the form of time lapses with 16-bit pixel intensities. The file extension is .tif, representing the Tagged Image File format (TIF or TIFF), a common image format for microscopy.

In the context of drug discovery, fluorescence microscopy has become an indispensable tool for understanding the mechanisms underlying the action of drugs and identifying new drug targets. One of the main advantages of fluorescence microscopy is its ability to provide real-time visualization of cellular processes, enabling the observation of drug effects on living cells. This enables researchers to study drug interactions with specific targets, such as proteins or receptors, and investigate their effects on cellular functions.

Fluorescence microscopy is also well-suited to high-throughput drug screening, where large numbers of compounds can be rapidly screened for their effects on cellular processes. This is achieved using automated microscopes and imaging software that can capture and analyze large amounts of data quickly and efficiently. In addition, as we will shortly see, fluorescence microscopy can be combined with other imaging techniques, such as confocal microscopy or FRET imaging, to provide more detailed information about drug-target interactions and cellular processes. For example, FRET imaging can be used to monitor protein-protein interactions, providing insights into the biochemical states of the cells under study. It is this technique we will explore next.

2.3 Fluorescence Resonance Energy Transfer (FRET)

Fluorescence resonance energy transfer (FRET) is a widely used technique in fluorescence microscopy that allows the study of protein-protein interactions and other biochemical processes in living cells. The process of FRET entails measuring the transfer of energy between two fluorescent molecules, a donor and an acceptor, when they are in close proximity to each other. In the context of the present project, the donor is a yellow fluorescent protein (YFP) and the acceptor is a cyan fluorescent protein (CFP). For the sensitized emission FRET procedure used by the Jones lab, the energy stored in either protein is indicated by the pixel intensities for each cell, viewed through their respective channels. When ExF1 or ExD3 are introduced into the culture, a reaction is indicated by a transfer of brightness from one channel to the other. The ratio of pixel intensities is then used to measure reactivity. Formally, for the set of pixels belonging to any given cell, $Cell_i^{(t)}$, at a given timestep t, the FRET response of the cell is given by:

$$FRET(Cell_i^{(t)}) = \frac{1}{|Cell_i|} \sum_{p_j \in Cell_i} \frac{YFP^{(t)}(p_j)}{CFP^{(t)}(p_j)} \tag{2.1}$$

where $\frac{YFP^{(t)}(p_j)}{CFP^{(t)}(p_j)}$ is the ratio of pixel intensities between the YFP and CFP channels at time t of pixel p_j, with p_j being an element of the set $Cell_i$. Figure 2.5 provides an example of the initial energy disparity between the YFP and CFP channels before the introduction of a drug. Observe that neurons in the yellow channel image display higher pixel intensities, indicating that a drug induced energy transfer has yet to occur.

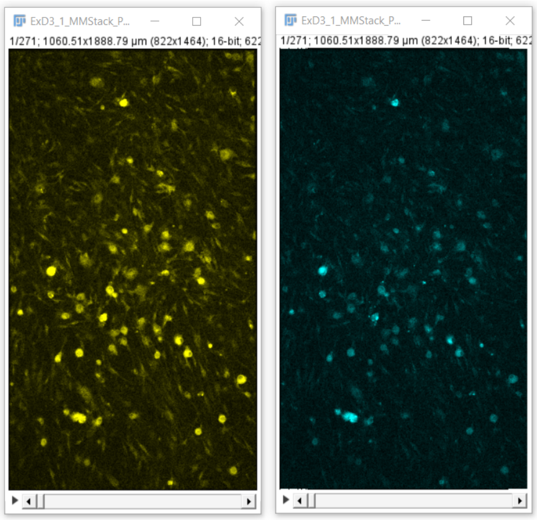

ExD3_1_MMStack_P...
1/271; 1060.51x1888.79 µm (822x1464); 16-bit; 622
ExD3_1_MMStack_P...
1/271; 1060.51x1888.79 µm (822x1464); 16-bit; 622

Figure 2.5: Visualising the energy disparity between the YFP and CFP channels in a FRET experiment. Prior to drug introduction the YFP channel is always the more energetic. Both images are of the same cells at identical time-steps. Note that these two images are the central and rightmost panes of Figure 2.4 with filters applied.

2.4 Image Analysis Pipeline

Equipped with an understanding of the neurobiogical questions being pursued by the present project, fluorescence microscopy as an imaging medium, and finally FRET imaging as applied to drug discovery, we now examine the current image analysis pipeline within the Jones lab. First, Section 2.4.1 will provide an overview of the pipeline, which will be followed by a detailed walkthrough for a particular acquisition in Section 2.4.2.

2.4.1 Pipeline Overview

At a high level, the image analysis task is to calculate cellular responses to two drugs: exendin-phe1 and exendin-asp3, hereafter referred to as ExF1 and ExD3 respectively. The determination of a given cell's response to these two molecules is accomplished through a ratiometric comparison of its pixel intensities under two fluorescent filters. This is presently achieved by acquiring time lapses of mouse neurons being stimulated by ExF1 and ExD3, and then comparing the response to these drugs against a control, or 'vehicle' (VeH). The bioimaging software Fiji [26] is used to perform this analysis.

Two channels are recorded to measure changes in one of two fluorescent proteins: yellow fluorescent protein and cyan fluorescent protein. The response of these channels indicates the amount of excitation of cylic adenosene monophosphate (cAMP), a neuronal receptor responsible for appetite. The ratio of yellow fluorescent protein to cyan fluorescent protein is then taken as the measure of activity a given neuron exhibits in response to the trialled drug.

Ten steps are required to prepare the time lapse acquisitions before they are ready to be used to collate the data graphed in Figure 2.2. The first of these is flat-field correction, a process in which patterns of non-uniform illumination produced by the imaging process are corrected for algorithmically. Following this, the two YFP and CFP channels are separated, registered, and then merged once again into a single time lapse. This is to correct for misregistration between images captured through the yellow and cyan filters, and will be discussed in greater detail in Section 2.4.2. Then, as the medium in which the neuron culture is suspended is fluid, we correct for cellular drifting over time. With the images across the two channels now formatted appropriately, we add a uniform constant to all pixels and separate the channels. This allows for the YFP channel to be divided by the CFP channel without division errors and computes one element of the ratiometric FRET metric outlined in the preceding section. A maximum pixel intensity projection is then used to create boundaries around cells in the z-dimension. This enables manual user thresholding of pixel intensities so as to produce a binary image from which cells or 'regions of interest' (ROI) can be drawn. These ROIs are then projected onto the FRET image, each cell's FRET score is calculated, and finally the results are analysed.

2.4.2 Pipeline Walkthrough

As previously described, the existing image analysis pipeline is a ten-step process that combines both automated plug-ins provided within Fiji and manual manipulation of the time lapse images. The full pipeline and the rationale for each step are detailed below.

1. **Apply BASiC flat-field correction algorithm**.

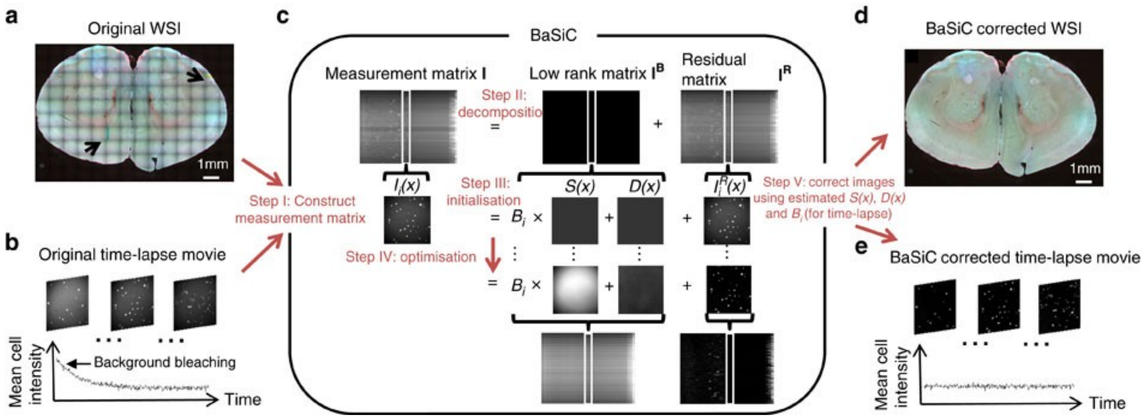


Figure 2.6: A schematic for the BASiC flat field correction algorithm [27].

Flat-field correction refers to the process of rectifying non-uniform patterns of illumination in digital images. In the case of microscopy, this often takes the form of vignetting – a radial pattern of decreasing pixel intensities, as pictured in Figure 2.7. Flat-field correction is necessary as the computation of FRET is sensitive to differences in pixel intensities within a given time-slice. Adjusting for this is therefore a necessary pre-procesisng step. The flat-field correction algorithm currently in use is BASiC, presented by Peng et al in [27]. The algorithm operates on the basis of the shading model, which approximates the process of image formation as a linear function that relates a measured image, $I^{meas}(x)$ at location x, to its uncorrupted true correspondence, $I^{true}(x)$, as:

$$I^{meas}(x) = I^{true}(x) \cdot S(x) + D(x) \tag{2.2}$$

where $S(x)$, the flat-field, represents the change in effective illumination across an image, and $D(x)$, the dark-field, represents camera offset and thermal noise, which are present even if no light is being received. The BASiC algorithm attempts to reconstruct I^{true} from I^{meas} by estimating $S(x)$ and $D(x)$. BaSiC first constructs a measurement matrix I, which is then decomposed into a low-rank matrix IB and a sparse residual matrix IR. The low-rank matrix is optimized by promoting the sparsity of the residual matrix with a reweighted L1-norm. Finally, the optimization problem is solved using a linearized augmented Lagrangian method.

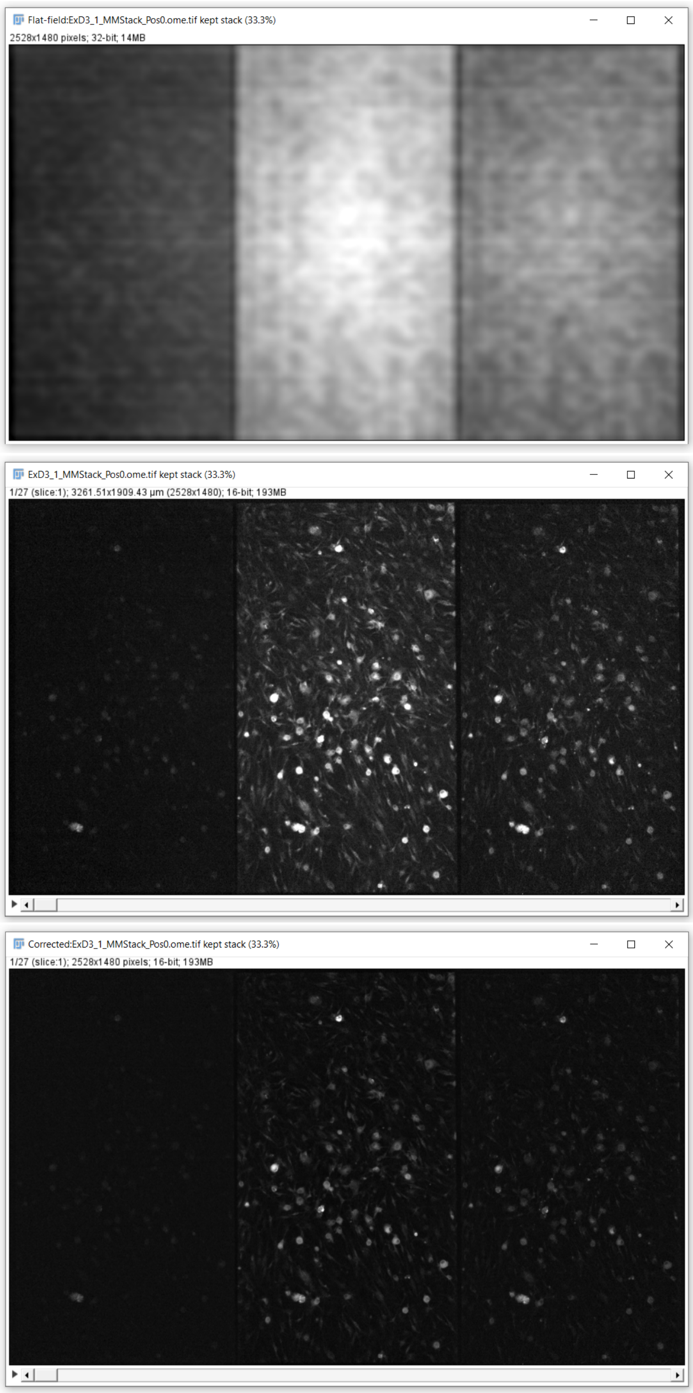

Flat-field:ExD3_1_MMStack_Pos0.ome.tif kept stack (33.3%)
2528x1480 pixels; 32-bit; 14MB
ExD3_1_MMStack_Pos0.ome.tif kept stack (33.3%)
1/27 (slice:1); 3261.51x1909.43 μm (2528x1480); 16-bit; 193MB
Corrected:ExD3_1_MMStack_Pos0.ome.tif kept stack (33.3%)
1/27 (slice:1); 2528x1480 pixels; 16-bit; 193MB

Figure 2.7: Demonstrating the effect of the BASiC flat-field correction algorithm. At top, the illumination pattern, or 'flat-field' model. Note the radial pattern of illumination emanating from the centre, with a darker pattern of illumination in the corners. At middle, the original image from which the flat-field model was calculated. At bottom, the corrected image produced by dividing the original image by the flat-field model.

2. **Channel separation and registration**. Having corrected for region-wise disparities in illumination, the two-channel time lapse is separated into individual z-stacks, one for the YFP and CFP channels respectively. This step is to allow for registration between the two channels. Two sources of misregistration contribute here: user error and chromatic aberration. The latter refers to a property of microscopic imaging whereby a lens may fail to focus all colours to the same point due to distinct refractive indices between wavelengths. Human error may also contribute to misalignment as at present the user is required to highlight regions of interest using a rectangular lasso tool. Consequently, there is no guarantee of complete consistency between the cropped regions. There are four subtasks involved in executing this step:

    (a) *Use SIFT registration algorithm.* The scale invariant feature transform (SIFT) algorithm [28] is used to register the two channels.

    (b) *Record the transformation matrix.* The output of the SIFT algorithm is a transformation matrix that defines a mapping between the YFP and CFP channels. This must be recorded for later use.

    (c) *Apply transformation.* Armed with the transformation matrix of the preceding step, the user manually applies the transformation specified to the template image.

    (d) *Merge YFP and CFP channels.* Once the time lapses have been registered, they are recombined. While they will later be split once again, another preprocessing step in the form of drift correction is required. Since this needs to be done for both channels, it is simplest to recombine them prior to this step.

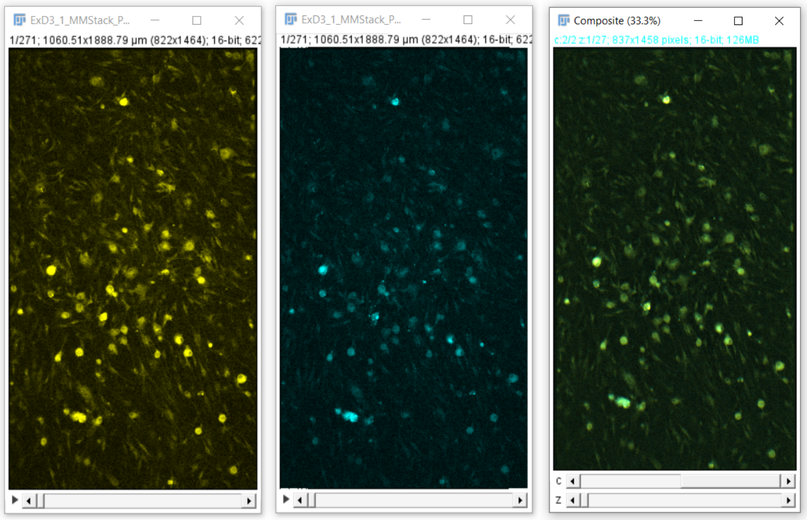

ExD3_1_MMStack_P...
1/271; 1060.51x1888.79 µm (822x1464); 16-bit; 622
ExD3_1_MMStack_P...
1/271; 1060.51x1888.79 µm (822x1464); 16-bit; 622
Composite (33.3%)
c:2/2 z:1/27; 837x1458 pixels; 16-bit; 126MB
c
z

Figure 2.8: YFP channel (left), CFP channel (middle), and their registered overlap image (right). Note that the registration is not perfect as only a translation may be applied, while misregistrations due to aberration are often rigid or affine transformations.

3. **Drift correction**. While the neurons being imaged are not motile, cells are liable to drift over the course of the time lapses due to collisions with debris floating in the medium in which they are suspended or the introduction of the aqueous drugs. This motivates drift correction, which accounts for temporal displacements of individual cells. This is simply another form of image registration, except that rather than being performed between channels, the registration is performed over the z-dimension within each channel.

4. **Standardisation and channel splitting**

   (a) *Add a uniform constant to pixel intensities.* Adding a small constant to all pixels in the merged image stack avoids division by 0 when calculating FRET. Recall that the FRET for a particular cell C_i is calculated as a sum involving the term $\frac{YFP(p_j)}{CFP(p_j)}$, where $YFP(p_j)$ is the amount of yellow fluorescent protein exhibited by a pixel $p_j \in C_i$ and $CFP(p_j)$ is the amount of cyan fluorescent protein exhibited by $p_j \in C_i$. We thus compute a FRET time series by dividing the YFP channel by the CFP channel. This step ensures that no illegal divisions by 0 occur, while also avoiding altering the final result.

   (b) *Split back into individual channels (two separate z-stacks).* Here we facilitate FRET computation in the next step by separating the two channels so the quotient of YFP over CFP can be calculated.

5. **Compute FRET image.** The FRET time lapse is produced by calculating the ratio between the registered YFP and CFP channels. The result is a time lapse which highlights the reactivity of each cell to the drug being tested or the control.

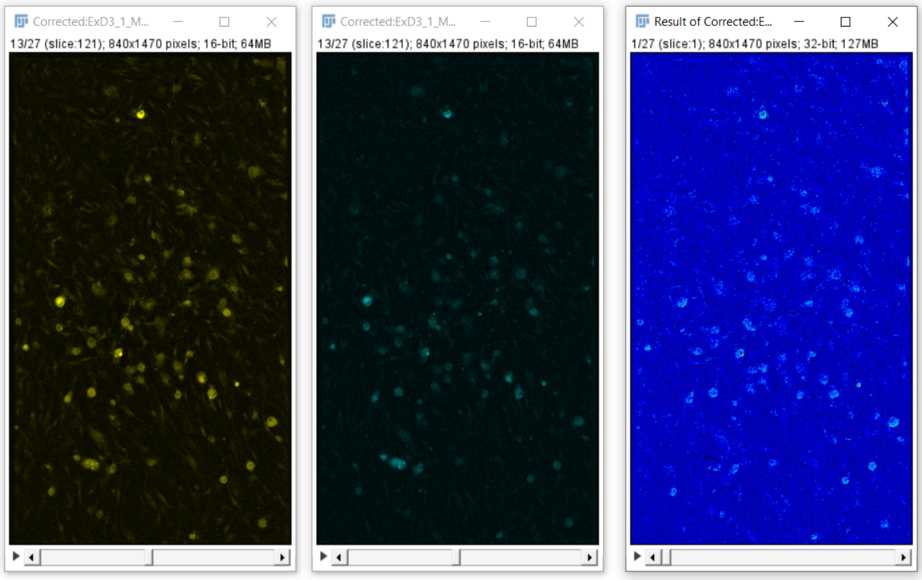

Corrected:ExD3_1_M...
13/27 (slice:121); 840x1470 pixels; 16-bit; 64MB
Corrected:ExD3_1_M...
13/27 (slice:121); 840x1470 pixels; 16-bit; 64MB
Result of Corrected:E...
1/27 (slice:1); 840x1470 pixels; 32-bit; 127MB

Figure 2.9: YFP (left), CFP (middle), and ratiometic FRET time lapse (right).

6. **Maximum intensity projection**. In calculating FRET we are interested in the mean ratio of YFP to CFP for each cell throughout the time lapse. Since we are now dealing with the FRET time lapse, this simply requires localising the exact pixel set of each neuron and taking the average over these values. To do this, we need to segment out each neuron. A first preprocessing step towards achieving this is to compute the maximum pixel intensity over the whole image along the z-dimension to emphasise the boundaries of each neuron.

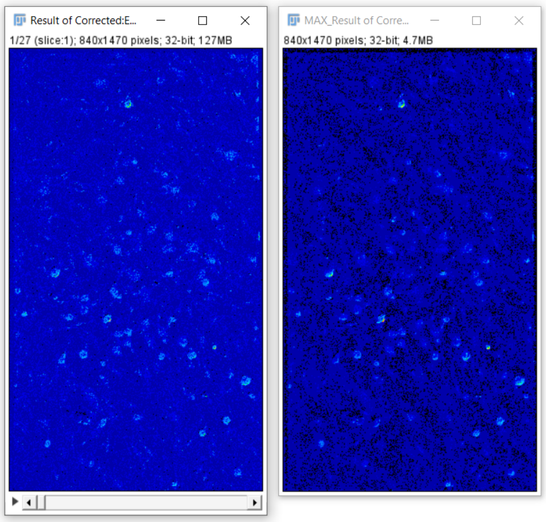
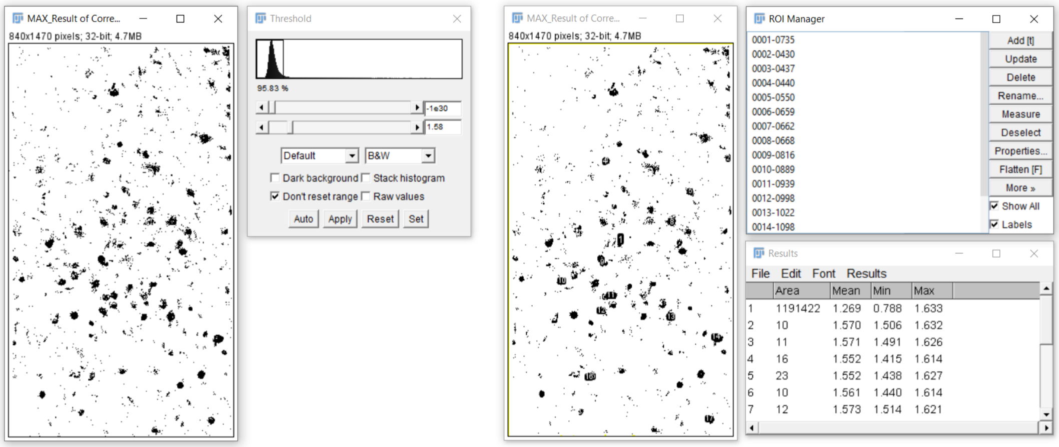

Result of Corrected:E...
1/27 (slice:1); 840x1470 pixels; 32-bit; 127MB
MAX_Result of Corre...
840x1470 pixels; 32-bit; 4.7MB

Figure 2.10: A FRET time lapse (left) and its maximum intensity projection (right).

7. **Cell segmentation**. Using the maximum intensity projection image, cells are segmented so that boundaries may later be drawn around them in the FRET time lapse, thereby allowing only the activation of individual neurons as measured by changes in their YFP-CFP ratio to be taken into account and not the image background. This is currently a two-step manual process:

   (a) *Thresholding*. Segment the cells and produce a binarized image according to a single intensity threshold. The threshold is set by a combination of intuition and experimentation.

   (b) *Analyse particles*. The resulting segmentation will be imperfect, and often includes segmentations representing non-cellular debris. To remedy this, Fiji's particle analysis tool is used to impose constraints on the minimum area an image region must have in order to constitute a valid segmentation.



840x1470 pixels; 32-bit; 4.7MB

95.83 %

-1e30

1.58

Default B&W

☐ Dark background ☐ Stack histogram
☑ Don't reset range ☐ Raw values

840x1470 pixels; 32-bit; 4.7MB

0001-0735
0002-0430
0003-0437
0004-0440
0005-0550
0006-0659
0007-0662
0008-0668
0009-0816
0010-0889
0011-0939
0012-0998
0013-1022
0014-1098

☑ Show All
☑ Labels

	Area	Mean	Min	Max
1	1191422	1.269	0.788	1.633
2	10	1.570	1.506	1.632
3	11	1.571	1.491	1.626
4	16	1.552	1.415	1.614
5	23	1.552	1.438	1.627
6	10	1.561	1.440	1.614
7	12	1.573	1.514	1.621

Figure 2.11: Thresholding (left) and particle analysis (right).

8. **Region of interest (ROI) projection**. Having established a reasonable segmentation, Fiji's region of interest manager is used to project regions of interest on to the FRET image. This maps borders from each segmented region in the binarized image to the FRET time lapse. Since we have performed drift correction, the expectation is that these borders will be faithful outlines of each cell over time.

9. **Compute FRET**. Now that we have the FRET time lapse and the regions of interest representing each suspected cell, we can compute the FRET value within every ROI for each z-slice.

10. **Plot time series**. Finally, with a time lapse of ratiometric measurements for each cell, the time series data may be plotted and further analysis may now be done.

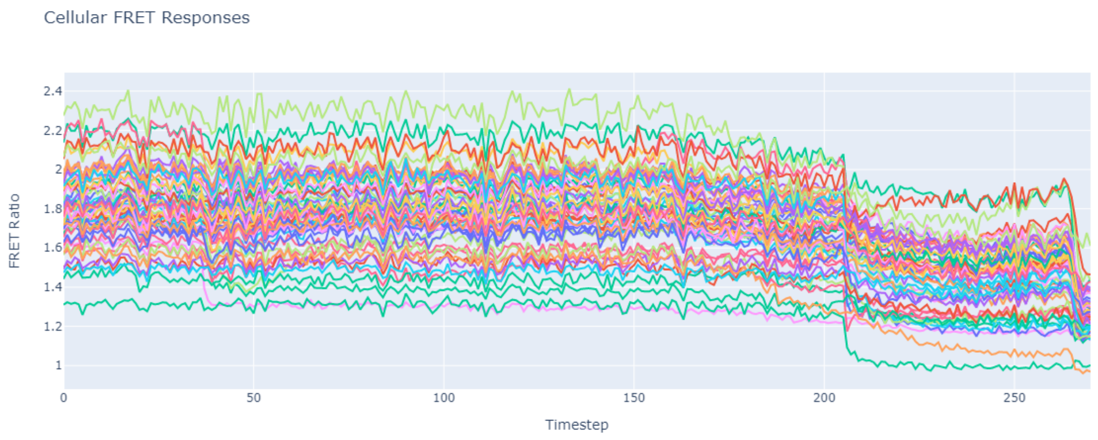


Figure 2.12: Visualising the per cell FRET reponses over time to the ExD3 drug.

Several stages in the preceding pipeline may be automated using modern techniques in deep learning. We turn to the prerequisites for understanding these next.

2.5 Deep Learning

Deep learning is a branch of machine learning which is itself a sub-discipline within the broader field of artificial intelligence. Here the emphasis is on deep neural networks, particularly those with three or more layers. As much of the focus of the present work will be on deep-learning based approaches to object detection, segmentation, and classification, we will review the common deep learning architectures for these next.

2.5.1 Neural Networks

A neural network is comprised of interconnected computational units called neurons, which are organized into layers. The fundamental building block of a neural network is the artificial neuron, or perceptron, which loosely emulates the behaviour of biological neurons. Each perceptron receives inputs, applies a transformation, and produces an output. The inputs are weighted by adjustable parameters, referred to as weights, and combined with a bias term. This weighted sum is then passed through an activation function, introducing non-linearity into the network. The output of the activation function represents the output of the perceptron. Mathematically, the operation of a perceptron can be formalized as follows:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.3}$$

where x_i is a scalar input, w_i represents the corresponding weight, b is the bias term, and f signifies the activation function. Commonly used activation functions include the sigmoid function, hyperbolic tangent function, and rectified linear unit (ReLU) function. These non-linearities play a vital role in enabling neural networks to capture intricate relationships and make predictions on nonlinear data.

Multiple perceptrons are combined to form a layer, and multiple layers are stacked together to create a neural network. Within a layer, each neuron is interconnected with all neurons in the subsequent layer, resulting in a dense pattern referred to as a fully connected layer. The first layer is the input layer, which receives the raw input data. The layers between the input and output layers are known as hidden layers. Lastly, the final layer is responsible for producing the network's predictions or outputs.

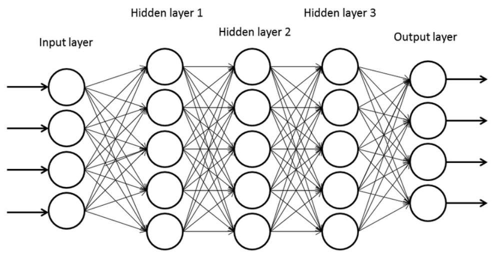


Figure 2.13: A visualisation of a five-layer neural network.

2.5.2 Convolutional Neural Networks and Fine-Tuning

Convolutional neural networks (CNNs) are an extension of traditional neural networks specialised for processing image data that have emerged as the dominant architecture for computer vision tasks [29]. The core operation in a CNN is the convolution, which involves sliding a set of filters or kernels over the input image and computing the dot product between the filter and local patches of the input. Mathematically, the convolution operation between an input image X and a filter W is represented as:

$$(X * W)(i,j) = \sum_m \sum_n X(i+m, j+n) \cdot W(m,n) \tag{2.4}$$

where i and j are coordinates of the output feature map. This process enables the network to learn local features and capture spatial relationships effectively.

The architecture of a typical CNN comprises multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers are responsible for learning various image features by applying convolution operations. Pooling layers downsample the spatial dimensions of the feature maps, reducing computational complexity and providing translational invariance. Fully connected layers serve as the final layers of the network, enabling high-level reasoning based on the learned features.

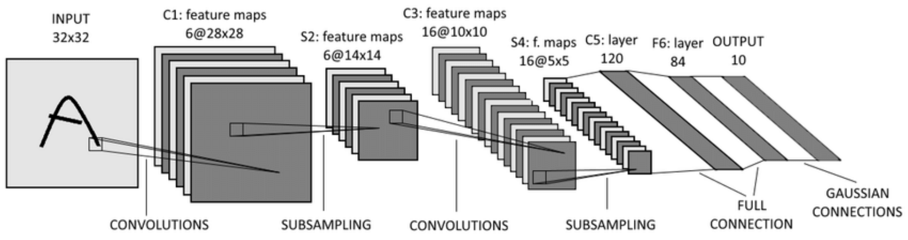


Figure 2.14: An example of LeNet, an early CNN architecture for digit recognition [30].

Training a neural network involves adjusting the weights and biases to minimize a suitable objective function, often referred to as the loss function. The most prevalent technique for training neural networks is backpropagation, which leverages the gradient of the loss function with respect to the network parameters to update them iteratively. This iterative process of forward propagation (computing the network's output) and backward propagation (updating the parameters based on computed gradients) enables the network to learn from labelled training examples and improve over time.

Pre-trained CNNs, trained on large-scale datasets like ImageNet [31] or COCO [32], offer valuable features and learned representations that can be utilized as backbones for other imaging models. We will see in later chapters that by leveraging pre-trained CNNs, it is possible to avoid the need for training models from scratch, which is computationally expensive and time-consuming. Additionally, pre-trained CNNs capture general image features in their early layers, making them effective feature extractors. By removing the task-specific layers and using the intermediate feature maps, pre-trained CNNs can serve as powerful feature generators for object detection – a topic to which we will turn next.

2.5.3 Faster R-CNN Object Detection

As will later be discussed in Chapter 4, object detection will play a crucial role in the development of a system that can localise and segment out cells in preparation for FRET analysis. Owing to their robustness and usability, emphasis will be placed on the Faster R-CNN [33], a two-stage object detection architecture.

The Faster R-CNN consists of three main components: a shared convolutional backbone network, a region proposal network (RPN), and a region-based detection network. The backbone network processes the input image and generates a feature map. The RPN takes the feature map as input and produces region proposals by sliding a set of predefined anchor boxes across the feature map and predicting their 'objectness' scores and refined bounding box coordinates. The proposals are then used to extract fixed-size feature maps from the backbone network. Finally, the region-based detection network classifies each proposed region into different object classes and further refines their bounding box coordinates. The network is trained in a multi-task manner using a combination of losses, including the RPN loss and the detection loss, which consist of classification and regression components. During inference, the network outputs the final detections by combining the predicted bounding boxes and class probabilities from the RPN and the detection network. The Faster R-CNN architecture achieves accurate and efficient object detection by leveraging region proposals and a shared backbone network to localize and classify objects in images.

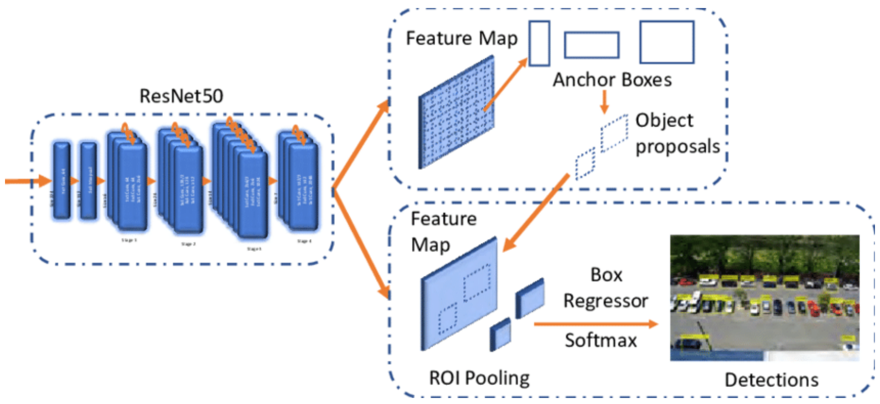


Figure 2.15: Faster R-CNN object detector architecture with a ResNet backbone [33].

The Faster R-CNN compares favourably to other state-of-the-art object detection methods [34]. As a two-stage object detector, the Faster R-CNN has slower inference times than YOLO [35] and other popular one-stage object detectors. However, the Faster R-CNN's superior performance in challenging object detection scenarios makes it a popular choice in many applications. Moreover, while one-stage detectors achieve impressive real-time performance, they may struggle with accurate localization of smaller objects due to their coarse grid-based predictions. This makes the Faster R-CNN the most appropriate object detector for the task of locating cells in images.

2.5.4 Vision Transformers

While convolutional neural networks are still a foundational model for computer vision tasks and play a significant role in the present project, the recently developed 'vision transformer' is the base architecture behind the Segment Anything Model, a tool that will form the core of the cell segmentation pipeline to be discussed in Section 3.2.

Vision transformers represent a novel approach to processing visual data by adopting the transformer architecture, originally proposed by Vaswani et al [36] for sequence modelling tasks in natural language processing (NLP). Unlike traditional CNN-based models that operate on local image patches, vision transformers treat the entire image as a sequence of patches, enabling the model to capture global dependencies and context. The vision transformer (ViT) model introduced by Dosovitskiy et al in [37] achieved competitive results on image classification tasks, showcasing the effectiveness of vision transformers in capturing global dependencies.

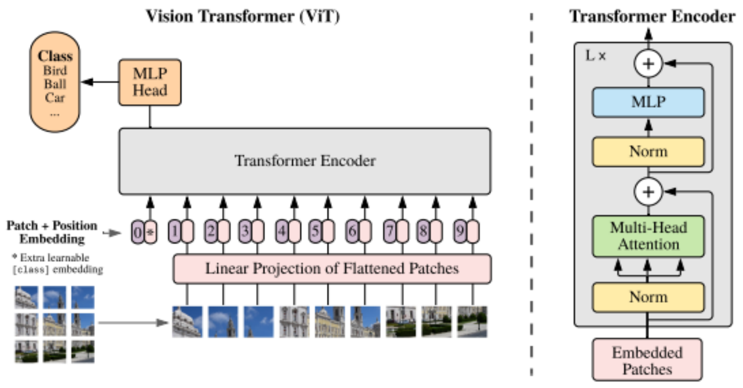


Figure 2.16: A schematic of a vision transformer from the Dosovitskiy et al paper [37].

At the core of vision transformers are self-attention mechanisms, which allow the model to attend to different parts of the input image when making predictions. The self-attention mechanism calculates the importance of each patch by considering the relationships between all patches in the image. This mechanism enables the model to capture long-range dependencies, making it effective in handling spatial relationships.

The vision transformer architecture consists of an embedding layer that linearly maps the input image patches to a high-dimensional feature space. These embedded patches are then processed through a series of transformer blocks. Each transformer block consists of multiple self-attention layers, followed by feed-forward neural networks, layer normalization, and residual connections. The self-attention layers enable the model to capture spatial dependencies, while the feed-forward networks help in modelling non-linear relationships within the patches. The output of the vision transformer is typically fed into a classification head, usually a single or multi-layer perceptron, to perform downstream tasks. For object detection and segmentation tasks, additional components such as positional encodings are often incorporated to enable spatial awareness.

Chapter 3

Related Work

This chapter details relevant existing work in the space of machine learning for fluorescence microscopy and microscopic imaging tasks more generally. From the walkthrough of the existing ten-step pipeline conducted in Section 2.4 we may identify a number of subfields within deep learning with potential to enhance the existing workflow, each of which has its own vast body of literature. The discussion that follows will therefore focus on the core concerns of the project, namely existing FRET imaging software in Section 3.1, and deep learning for image segmentation in Section 3.2.

3.1 FRET Imaging Software

A number of free and open-source software tools exist for automating the FRET analysis pipeline, with many of these being available as Fiji plug-ins. One such is PixFRET, developed by Feige et al and evaluated in their accompanying paper [38]. Being one of the earliest plug-ins for FRET imaging, the tool predates modern deep learning and is therefore largely reliant on traditional computer vision algorithms and user intervention for several stages of the image analysis process, namely cell segmentation and denoising. For example, the task of cell segmentation requires the user to draw boundaries separating cells from the image background manually. This is both time consuming and error prone, with inter-annotator and intra-annotator effects being liable to influence the final segmentation. In particular, given that cells tend only to have diameters on the order of tens or hundreds of pixels, inconsistencies in the segmentation process can lead to material changes in the final outcome of the FRET calculation.

PixFRET offers no tools for drift correction or inter-channel registration, meaning the task of FRET analysis for time-lapse acquisitions cannot be fully managed within the plug-in. The limitations of the software notwithstanding, a key merit of the plug-in is its usability and ability to accommodate variations in spectral bleed-throughs – a source of noise in FRET analysis in which fluorescence from a neighbouring channel appears in the channel of interest [39].

Another open-source Fiji plug-in for FRET analysis is the FRET and Colocalization Analyzer [40], a tool for sensitized emission FRET which computes FRET indices using a ‘pixel-by-pixel’ method. The software provides bleed-through control to control for artefacts that could lead to false FRET, as well as confidence intervals for FRET index calculation. Again, the FRET and Colocalization Analyzer predates the advent of modern deep learning, and therefore does not leverage modern cell-detection or segmentation models, instead using regression to ‘colocalize’ images on the basis of their pixel intensities and displaying to the user a regression graph that informs further processing.

More modern FRET analysis software has since been developed, with RiFRET being one such example. Originally released alongisde a paper evaluating intensity-based ratiometric FRET by Roszik et al in 2009 [41], RiFRET has been actively maintained since. A second version of the software was released in 2023, alongside a paper assessing software solutions to pixel-by-pixel autofluorescence corrected FRET [42].

The RiFRET software is considerably more sophisticated than PixFRET, taking spectral crosstalk, instrument sensitivity, and varying donor-acceptor ratio into account. RiFRET also provides facilities for pixel-by-pixel autofluorescence correction and batch mode analysis of large datasets. The same authors of the RiFRET software and accompanying papers also provide AccPbFRET, a plug-in for acceptor photobleaching FRET [43]. However, RiFRET skirts the issue of cellular segmentation by computing ratiometric FRET over all pixels and then performing background correction. Background correction means that an average of intensities in a given area without fluorophores or cellular materials is subtracted from the image. Autofluorescence can be corrected by subtracting average background corrected intensities of an unlabelled sample in the corresponding channel. Another modern open-source analysis tool for cell-level FRET analysis is the FLIM-FRET Analyzer [44], released in 2017. Again, despite having been released as recently as 2017, this FRET analysis software is still reliant on conventional intensity based image segmentation.

In [45], Thomson et al present DeepFRET, an automated and open-source deep learning pipeline for single-molecule FRET analysis. DeepFRET is the first, and at present only, open-source FRET software package that leverages deep learning to obviate the need for human annotation and intervention, requiring user input only in setting a quality threshold. Here the authors focus less on using deep learning for preprocessing tasks such as localization and segmentation, and instead emphasise the use of a corrective method called trace selection. In FRET analysis, trace selection is the process of filtering out noisy FRET computations. For example, these may arise due to image artefacts, non-uniform illumination, false segmentations, or misregistration across channels. Here the authors use simulated data to train a deep neural network to retroactively filter out corrupted FRET traces and remove them from the downstream analysis.

The primary limitation of the DeepFRET methodology for trace selection is that it is reliant on high volumes of simulated data. The network is pre-trained on 150,000 simulated traces that uniformly sample all possible FRET states, their respective lifetimes, transition pathways, as well as all possible noise levels, ensuring that the data represents all theoretically possible configurations. This is a computationally expensive procedure, and is not a method that would generalise well to cell-level FRET analysis.

3.2 Segmentation

Section 2.4.2 introduced the task of segmentation – assigning labels to each pixel in an image such that each connected component of uniform labels represents a semantically meaningful object. Further, in the preceding discussion of Section 3.1, we saw that a number of techniques for carrying out this task exist, with thresholding based on cell pixel intensities being among the most popular. In this section we explore the potential of machine learning alternatives for accelerating cell segmentation.

3.2.1 Pre-Foundation Model Approaches

At present, the leading cellular segmentation tool is CellProfiler [46], a general platform for cellular profiling which integrates Ilastik [47] into its image analysis toolkit. Ilastik uses a random forest classifier which learns from user provided annotations and outputs a probability map of class labels for each cell in an image. This probability map can then be postprocessed by the user to generate segmentation masks for cells in an acquisition.

Another commonly used tool among cell biologists is the trainable WEKA segmentation (TWS) toolkit [48]. This tool enables the training of segmentation pipelines by using generic hand-tailored image features. TWS operates by transforming the segmentation problem into a pixel classification problem in which each pixel can be classified as belonging to a specific segment or class. A set of input pixels that has been labelled is represented in the feature space and then used as the training set for a selected classifier.

More recently, attention has shifted toward deep learning. The first breakthrough in deep learning for cellular segmentation was the development of the U-Net architecture [49]. The U-Net architecture consists of an encoder that down-samples the input image to extract high-level features and a decoder that up-samples to reconstruct the segmentation mask. Skip connections connect corresponding layers between the encoder and decoder, facilitating multi-scale feature fusion and precise localisation of cell boundaries. The U-Net architecture excels at cellular segmentation due to its ability to accurately segment cells with varying shapes, sizes, and complex morphologies.

A more recent deep learning architecture for segmentation is the Mask R-CNN [50], an extension of the R-CNN framework for object detection outlined in Section 2.5.3. The Mask R-CNN employs a backbone network such as ResNet [51] to extract features from input images, a region proposal network to generate potential object-containing bounding boxes and refine their coordinates, and a region-based CNN for feature extraction from these proposed regions. These extracted features then contribute to both object classification and bounding box regression. Finally, the Mask R-CNN generates segmentations using the spatial distribution of objects and their pixel-wise locations.

While the U-net and Mask R-CNN architectures were developed for instance and semantic segmentation of any objects, a number of models developed explicitly for cell segmentation have been developed in recent years. One such is CellPose [52], a U-net based model which is the current state-of the art in generalised cell segmentation. Several similar tools have also been developed with a focus on specific cell types and shapes, such as Omnipose [53] and Stardist [54], but at present CellPose remains the most general.

3.2.2 Segment Anything Foundation Model

Taking inspiration from the recent successes of large scale promptable natural language processing models, recent research by Meta AI has produced the Segment Anything Model (SAM) [7] a foundation model for segmentation and object detection. The release of SAM consisted of three parts: a reframing of segmentation as a promptable task, the SAM model itself, and finally the publication of SA-1B, a dataset of over 1 billion masks on which SAM was trained.

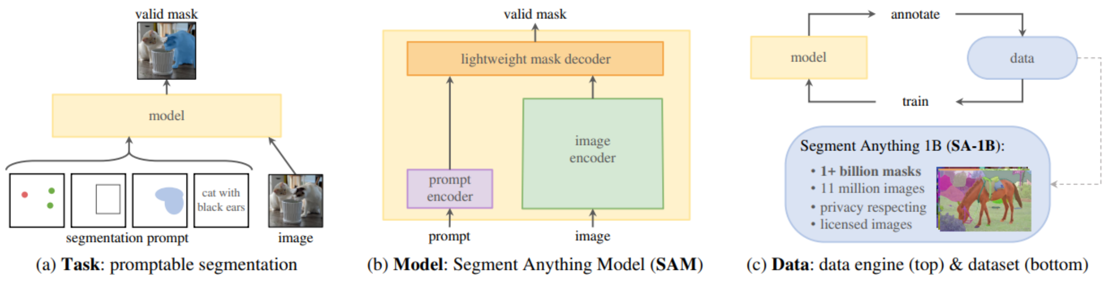


Figure 3.1: The tripartite structure of the SAM foundation model release [7].

Concerning the core model, the architecture of SAM consists of three sequentially connected networks: an image encoder, a prompt encoder, and a mask decoder. The image encoder is a masked autoencoder vision transformer (MAE). Having been pretrained on self-supervised masked image prediction tasks, the purpose of the image encoder is to compute an image embedding which encodes a useful representation for a given target image. Once an image embedding has been generated, the prompt encoder receives as input dense and sparse prompts in the form of bounding boxes, points, or provisional masks. These sparse and dense prompts are then mapped down into a 256-dimensional vector using learned positional encodings and convolutions. Lastly, the mask decoder efficiently maps the image embedding, prompt embeddings, and an output token to a mask. This model employs a modification of a transformer decoder block followed by a mask prediction head. The modified decoder block uses prompt self-attention and cross-attention in two directions to update all embeddings. After running two blocks, the embeddings are then up-sampled and an MLP maps the output tokens to a linear classifier, which then computes the mask probabilities at each image location. To rank masks, the model predicts estimated intersection over union (IoU) scores for each mask.

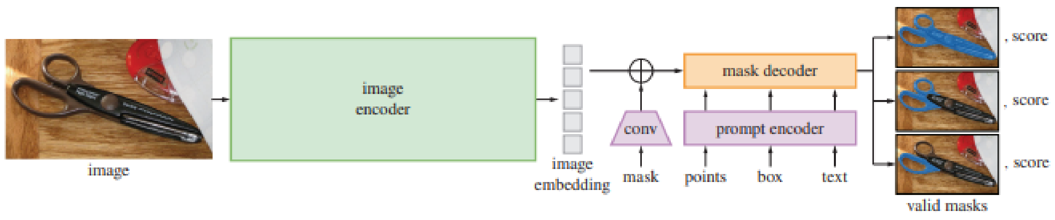


Figure 3.2: The multiple encoder-decoder architecture of SAM [7].

By separating SAM into an image encoder, a fast prompt encoder, and a mask decoder, the same image embedding can be reused with different prompts. The authors note that given a precomputed image embedding, the prompt encoder and mask decoder can predict a mask from a prompt in $\sim$ 50ms in a web browser, a demonstration of which is available online [55] and is shown in Figure 3.3.

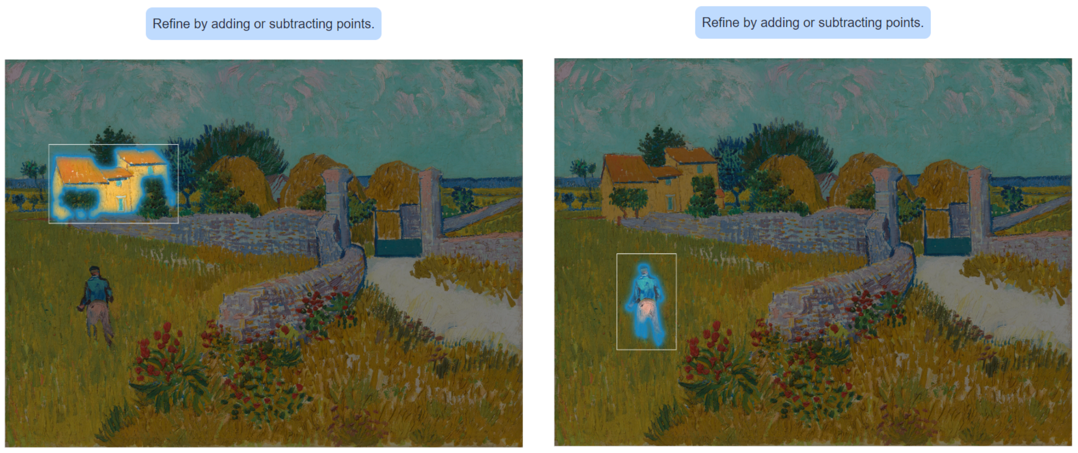

Refine by adding or subtracting points.
Refine by adding or subtracting points.

Figure 3.3: Demonstrating the performance of SAM on a difficult segmentation task in the form of a painted image. Box prompts have been provided to indicate an object of interest. Observe in the left hand image that even when significant occlusions are present in the prompt, the model is able to resolve the primary subject of the prompt and produce a segmentation mask exclusively for this.

Figure 3.3 demonstrates the reframing of segmentation as a promptable task. However, an alternative automatic, or ‘everything mode’, is also available. Here, dense point prompts are sampled from the given image in a regular grid, thresholded by the quality of the masks they predict, de-duplicated using non-maximal suppression, and finally the remaining masks are used as the final segmentation. This is shown in Figure 3.4.

With the majority of images in the SA-1B dataset being of natural scenes with macroscopic subjects, a question naturally arises as to whether SAM is capable of dealing with small objects. However, as visualised in Figure 3.5, SAM has been trained to produce high-quality masks even for dense segmentation tasks. Moreover, the authors of the SAM paper fine-tuned the model on 10,506 masks from the BBBC038v1 dataset [56] from the from Broad Bioimage Benchmark Collection. This dataset consists of a diverse collection of biological images collectively containing a broad range of cell types. The variety within the dataset is reflective of the types of images collected by research biologists at universities, biotech companies, and hospitals. In addition, the cells have been treated and imaged under a number of conditions, including fluorescent and histology stains, multiple magnification scales, and varying qualities of illumination. The preceding evidence is suggestive that cell segmentation is a promising candidate application for the Segment Anything Model.

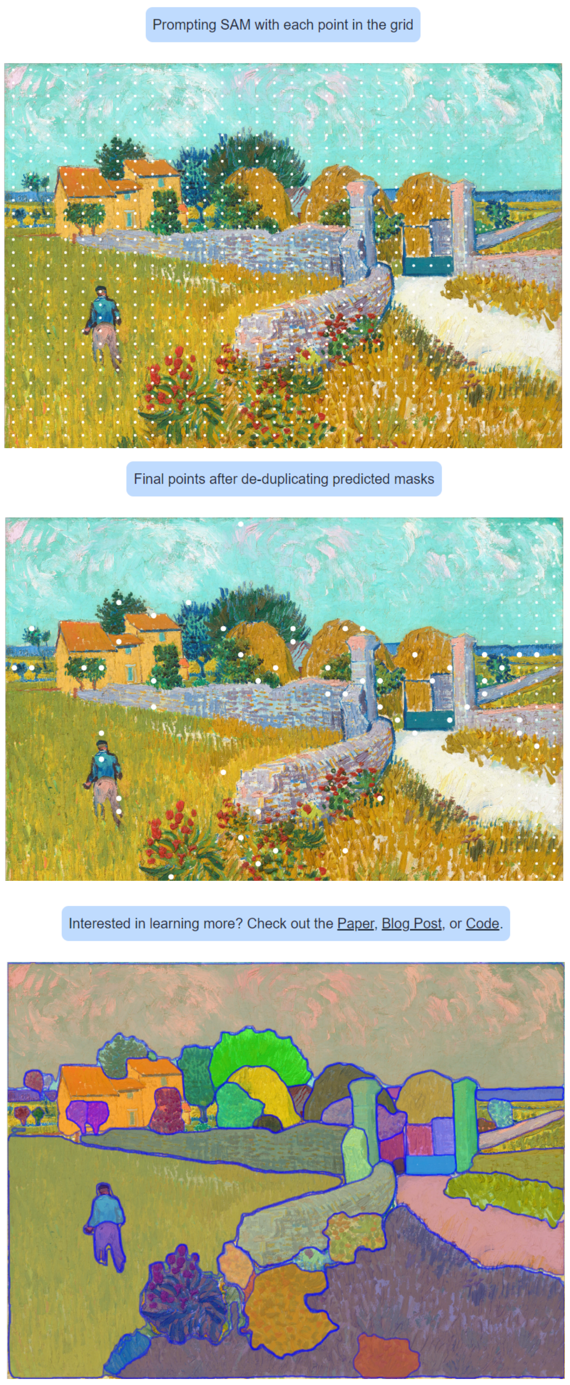

Prompting SAM with each point in the grid
Final points after de-duplicating predicted masks
Interested in learning more? Check out the Paper, Blog Post, or Code.

Figure 3.4: Demonstrating the ‘everything mode’ of SAM on the same segmentation task of Figure 3.3. Point prompts are first sampled from a regular grid (top). The resulting masks are then thresholded by quality and filtered using non-maximal suppression (middle), and finally the remaining masks are superimposed over the original image and displayed (bottom). The images of Figure 3.5 were produced using this same method.

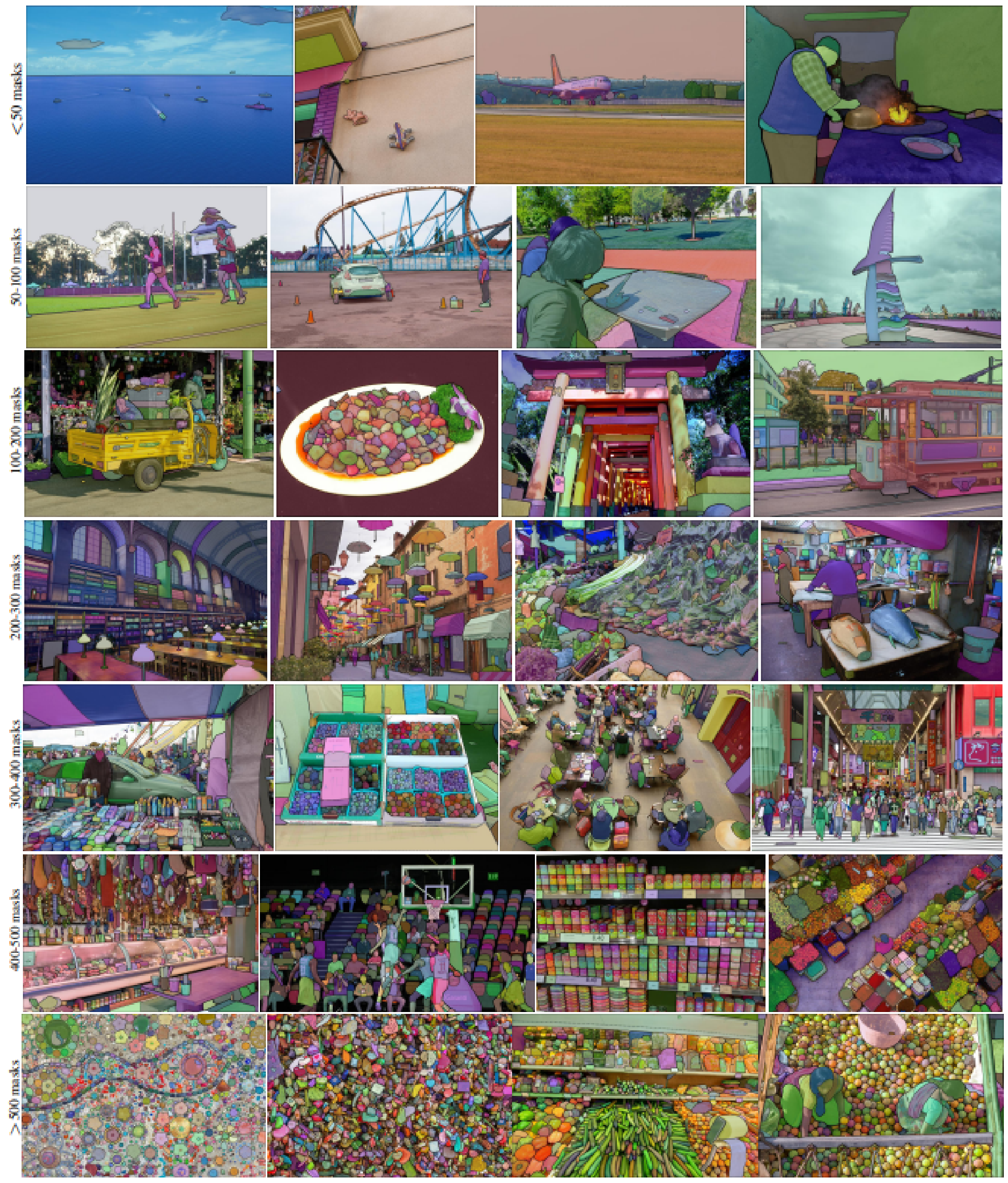


Figure 3.5: The segmentation performance of SAM across a wide variety of scenes from the SA-1B dataset [7]. These masks were annotated fully automatically by SAM, and have been verified by the authors using human ratings and numerous experiments to be of high quality and diversity. Images are grouped in increasing order of masks per image for visualization. Observe the performance of the model on dense scenes.

3.3 Summary

From the preceding subsections it will be clear to the reader that while a number of open-source software tools exist for accelerating FRET imaging, either by providing a self-contained package for FRET analysis as a whole or solving one of the individual subtasks in the FRET imaging pipeline, there are many desiderata that have yet to be unified within a single plug-in. Most notably, manual annotation and human supervision are still key parts of most FRET analysis software. Further, where deep learning solutions for various subtasks exist, such as for the task of segmentation, these are not available in commercial software packages. Even for those software packages which incorporate machine learning into their pipelines, such as CellProfiler and Fiji, these do not leverage deep learning, and are reliant on classical feature extraction. The demerit of this approach is that hand-crafted feature sets are rarely robust enough to capture the full complexity of cellular textures and morphologies, necessarily limiting the generalisability of the methods.

Concerning the task of cell segmentation specifically, this is a well-explored image analysis task with a vast body of literature behind it. However, despite the primacy of cell segmentation for other downstream applications in medical imaging, no satisfactorily universal method for cell segmentation yet exists. While a number of specialised and generalised deep learning algorithms for cell segmentation have been proposed, being supervised approaches such tools all require large labelled datasets and a significant amount of postprocessing to fine-tune their underlying models.

Transitioning to the Segment Anything Model as a foundational base for segmentation, three observations are suggestive that the method employed by the model are suitable for cellular segmentation. The first is the fact that the authors of the paper explicitly trained the model on extremely dense samples, as visualised in Figure 3.5, as well as a modest bank of cellular data. The second is that the model is promptable, allowing for human-in-the-loop interaction, which is criticial for biologists to be able to specify the areas of cells they are interested in, for example nuclei. Lastly, the final observation is the fact that the SAM prompt encoder and mask decoder are able to run in 50 milliseconds on a web browser, which hints at the suitability of the model for real-time interactive segmentation for bioimaging applications. It is this idea that we will explore in the following chapter. In particular, the next chapter will explore the development of a tool which incorporates the lessons learned from these three observations to perform interactive FRET analysis in fluorescence microscopy.

Chapter 4

FRETLab

This chapter details the development of FRETLab, a machine-learning enabled software solution which addresses each of the core processes of the Jones lab's image analysis pipeline, as detailed in Section 2.4. The chapter begins with Section 4.1, a high-level overview of the software and its applications in image analysis and drug discovery. This is followed by a description of the overall system architecture in Section 4.2, in which is detailed the core machine learning components, namely in the context of cell detection, segmentation, and clustering. Finally, the evaluation in Section 4.3 compares FRETLab to the existing pipeline through the analysis of user stories and assessments of reproducibility, data fidelity, and time efficiency. A user guide is available in Appendix A.

4.1 Software Overview

Available as a standalone executable for Linux OS, FRETLab is a Python-based image analysis tool with an embedded machine learning pipeline developed using PyTorch. The graphical user interface was developed using Python's inbuilt GUI toolkit, Tkinter [57]. Tkinter allows for the development of intuitive and interactive GUIs that can be easily integrated with native Python and various image analysis libraries, making it a natural choice for the present project. To facilitate usability, a wrapper around Tkinter was used, CustomTkinter [58], which provides added functionality not available in the base implementation of Tkinter, as well as a modernised UI overlay. The primary novelty of the software is in combining promptable segmentation with fine-tuned cellular object detection. While specialised for the image analysis task outlined in Section 2.4, FRETLab serves as a general demonstration that promptable segmentation models such as SAM coupled with fine-tuned cellular object detectors can accelerate drug discovery.

The construction of FRETLab is tripartite, with three tabs each providing their own functionality and utilities. Figure 4.1 highlights the first of these, which is presented on application launch. Titled *'Cell view'*, the initial tab allows the user to load a three-channel TIFF stack into the central pane, while the left pane provides utilities for preprocessing prior to FRET computation. On loading an image stack, a progress bar in the bottom left corner maintains track of the current frame being processed.

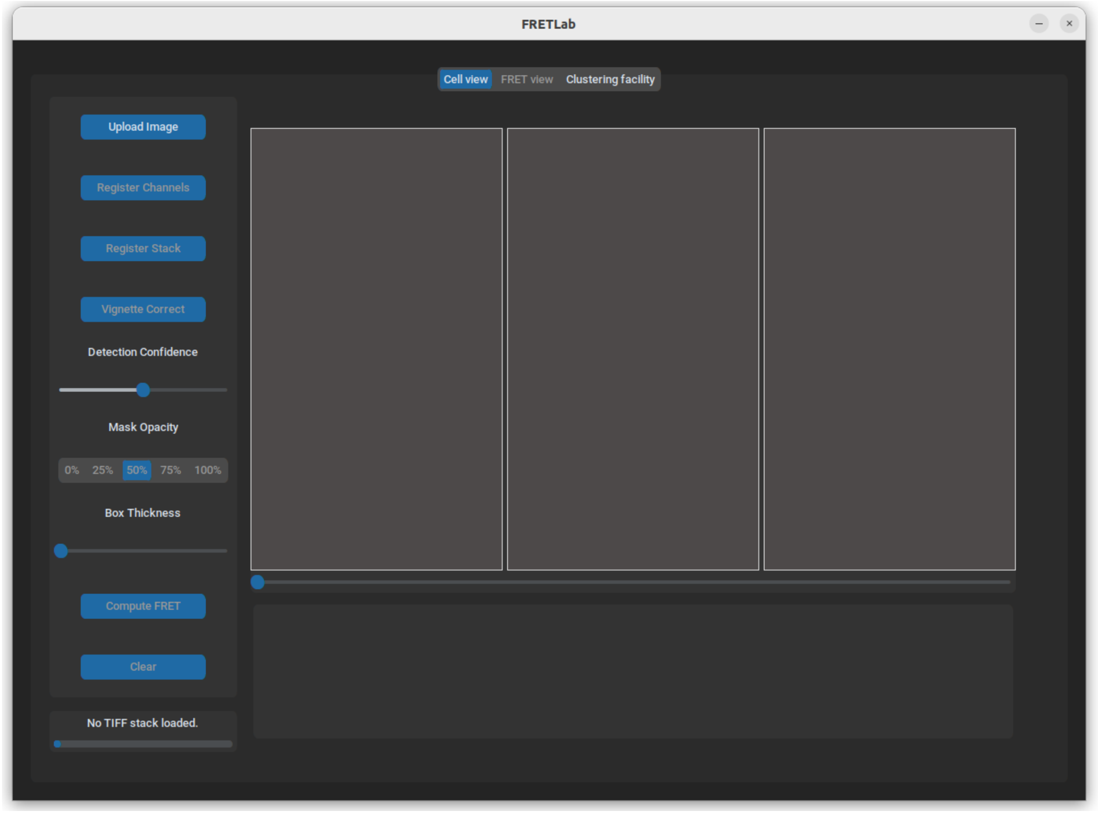
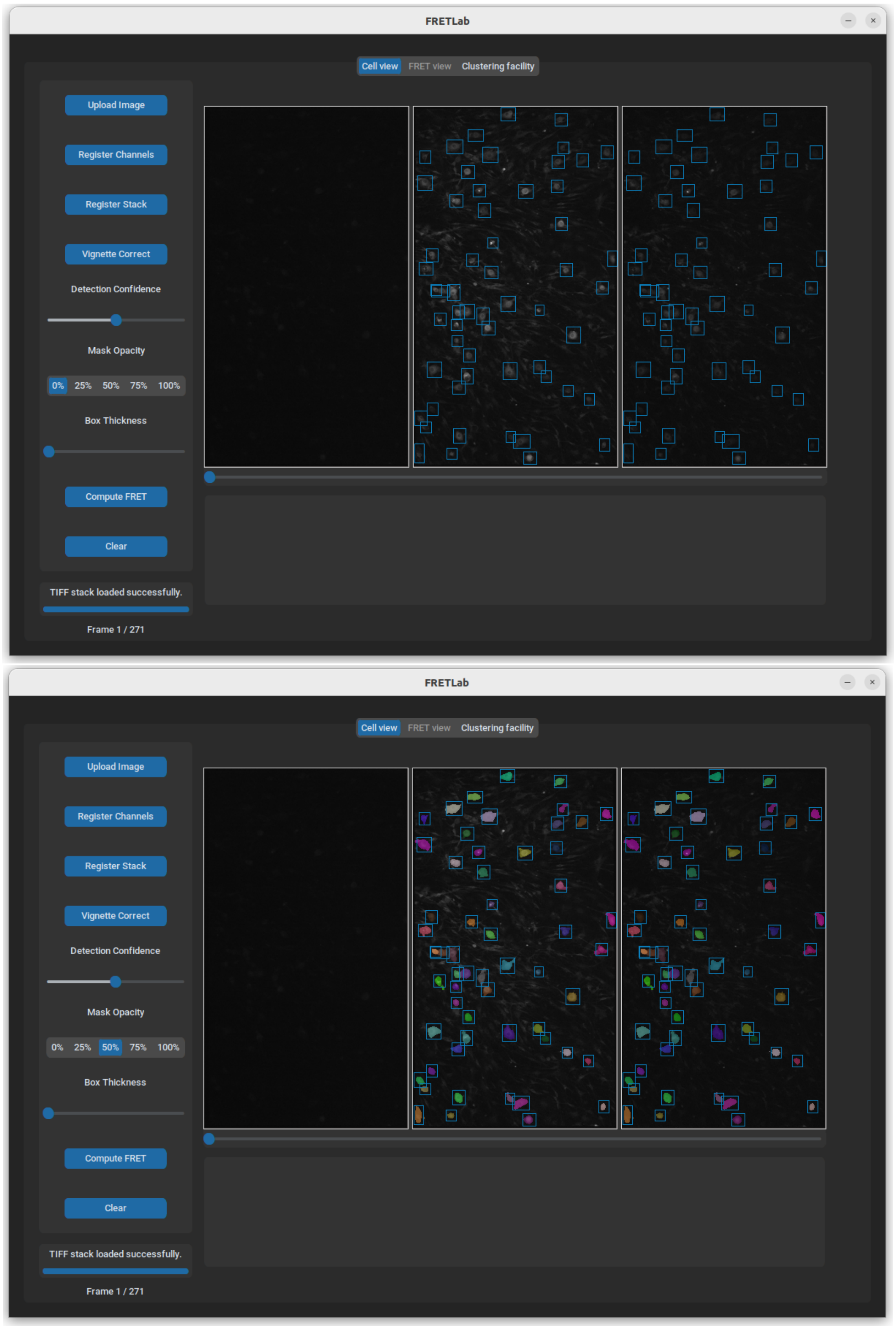


Figure 4.1: The initial display of the FRETLab software. On application launch, the user is presented with two panels, with the leftmost providing image processing tools, and the center displaying the area in which three-channel TIFF images are to be loaded.

At a high level, on loading an image stack using the *'Upload Image'* button, a Faster R-CNN object detector is run over the central panel of the topmost frame. As will be discussed at greater length in Section 4.2.2, the Faster R-CNN has been fine-tuned on a bounding box dataset hand-annotated by the author using 132 imaging acquisitions from the Jones lab and containing more than four-thousand neurons. Once the detections have been generated, these are transferred on to two translucent, interactive Tkinter canvases, which are superimposed on top of the central and rightmost image panes.

Prior to being displayed, the cellular detections are fed through to CellSAM, a fine-tuned Segment Anything Model, whereupon segmentation masks are generated for each. Two copies of each segmentation mask are maintained: a two-dimensional binary segmentation mask and a four-dimensional RGBA mask. The binary masks mark the precise image pixels belonging to each cell and are reserved for the computation of drug response, while the RGBA masks are coloured versions of their associated binary masks with an added alpha channel that allows the opacity of the masks to be modulated in real time to suit the user's needs. Figure 4.2 provides some example detections and segmentations for a previously unseen image stack in the cellular detection test set.



Mask Opacity: [✓ 0%] 25% 50% 75% 100%

TIFF stack loaded successfully.

Frame 1 / 271

Mask Opacity: 0% 25% [✓ 50%] 75% 100%

TIFF stack loaded successfully.

Frame 1 / 271

Figure 4.2: Cellular detections and segmentations produced by FRETLab once an image stack has been loaded. Bounding boxes indicate the presence of a cell with the confidence threshold set by the slider. At top, the cellular detections with the segmentation mask opacity set to 0% for clear visualisation. At bottom, the same detections, but the with the mask opacity set to 50% to allow the fidelity of the masks to be inspected.

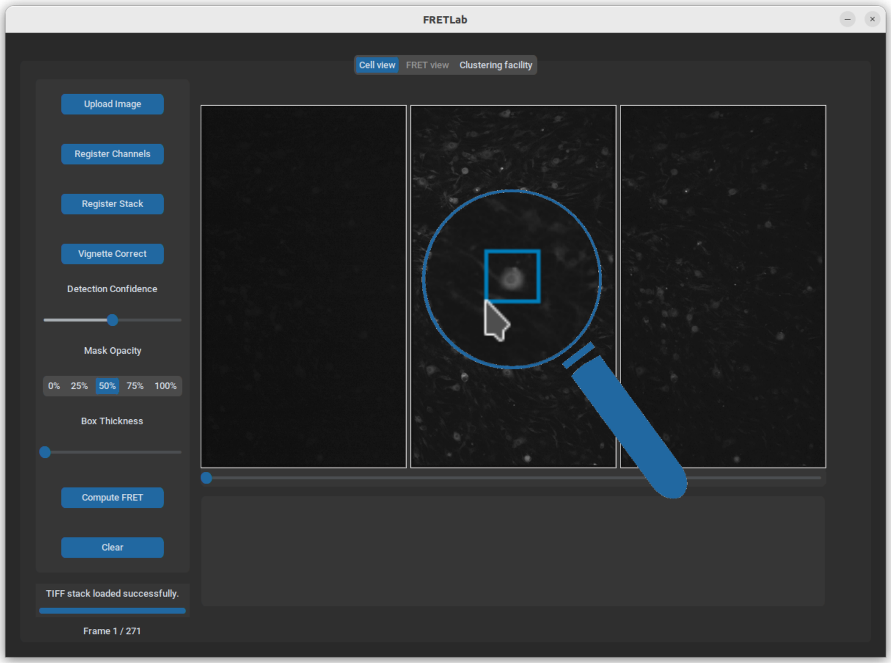
The demonstrations of the user-guided zero-shot prompt performance of SAM in Figure 3.3 were suggestive of a similar approach for CellSAM in FRETLab. Here the user may draw bounding boxes around individual cells should they have been missed by the object detection pipeline. As the image embedding for the top image frame is computed on loading the TIFF stack, user prompts may be passed through to the prompt encoder and mask decoder to produce segmentations in real time. These are then converted to translucent RGBA masks and overlayed over the current canvas frame. Each mask is bound to the box prompt that produced it via a dictionary, allowing the user to delete masks by clicking on the associated bounding boxes. An example is provided in Figure 4.3. As we will see in Section 4.2.3, CellSAM has been fine-tuned on a cellular segmentation dataset such that for reasonable perturbations to any given bounding box prompt the segmentation masks produced are invariant. However, this perturbation invariance was enforced only within a narrow pixel window. Thus, with some prompt engineering, the user may 'guide' SAM to produce a more faithful segmentation.

Once the image stacks, detections, and segmentations have been displayed, several utilities become available for use: channel registration, stack registration, vignette correction, detection thresholding, mask opacity modulation, and the setting of the bounding box thickness. These utilities range from aesthetic user preferences that aim to make the tool more usable to preprocessing steps that can be used to standardise the computation of cellular drug response.

Recall from Section 2.4 that channel registration, stack registration, and vignette correction play a critical role in the original image analysis pipeline. Channel registration is used to correct for chromatic aberration that occurs between the yellow and cyan filters, while stack registration is used to correct for temporal drift of the microscope due to experimental conditions. Finally, vignette correction is used to correct radial patterns of decreasing pixel intensities emanating from the image centers. To ensure standardisation between the existing image analysis pipeline and FRETLab, the same algorithms have been used for channel registration, stack registration, and vignette correction. The registration utilities are both enabled by the package *pystackreg* [59], while vignette correction is implemented with the aid of *pybasic* [60]. Both packages are wrappers around the exact algorithms employed within the Fiji plug-ins used in the original pipeline. For *pystackreg* this is the SIFT-based registration algorithm described in Section 2.4.2, and for *pybasic* this is the BASiC algorithm, also described in Section 2.4.2.

Confidence thresholding allows the user to change the detections present on the screen on the basis of the confidence score computed by the Faster R-CNN. These are cached at loading time, and thereafter the detections remain on the canvas at all times, either displayed or rendered transparent according to the current setting. This design allows the threshold slider to update the detections in real time, as opposed to repeatedly rerunning the model. An example is provided in Figure 4.4. Further, the detections may be enlarged by increasing the thickness of the bounding boxes using the provided slider.

The final feature of the *'Cell view'* tab is a video slider which enables the user to play through the image stack. Coupled with the bounding boxes superimposed over the cells, the user may use this feature to determine whether a cell has drifted over time, and thereby exclude it from the FRET analysis. An example is given in Appendix A.



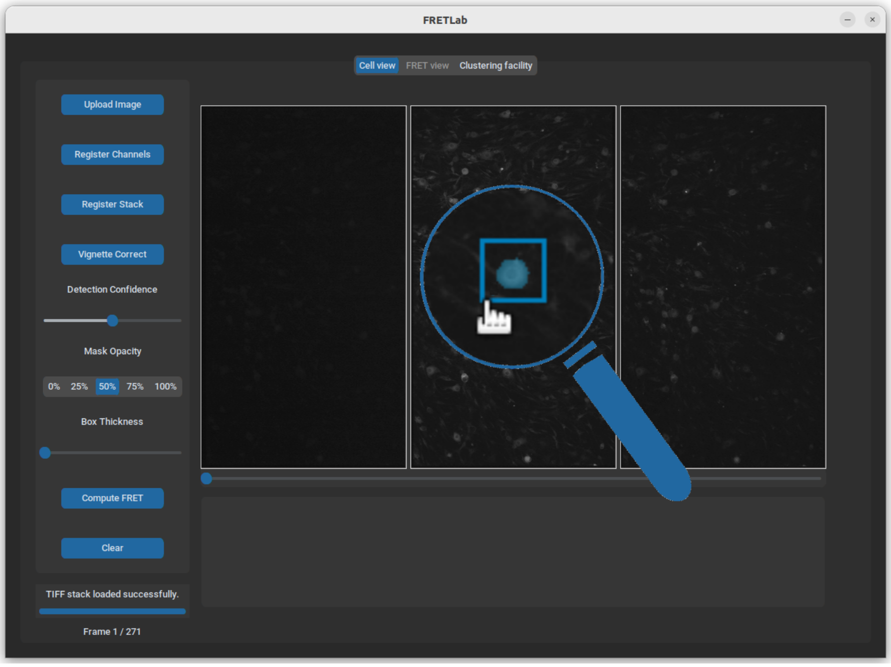
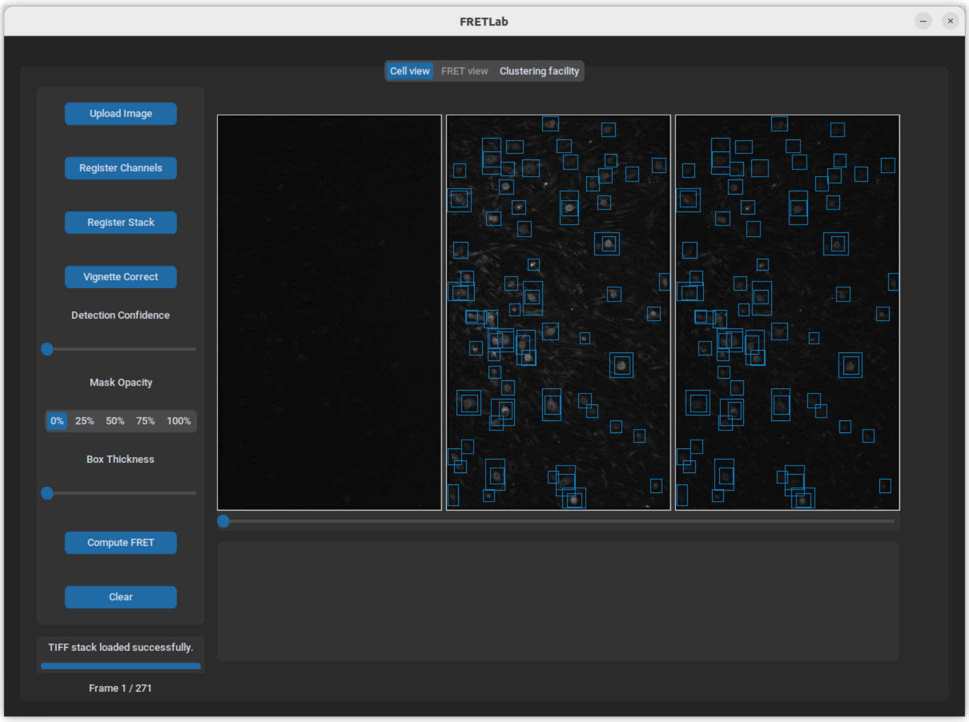


Figure 4.3: Drawing segmentation boundaries free-hand with the mouse. At top, the user's bounding box coordinates are transfered from the interactive Tkinter canvas through to the prompt encoder of a fine-tuned version of the Segment Anything Model. Once the prompt embedding has been computed, this is then run through the mask decoder, along with the precomputed image embedding, to produce a segmentation in real time. At bottom, the resulting segmentation mask.



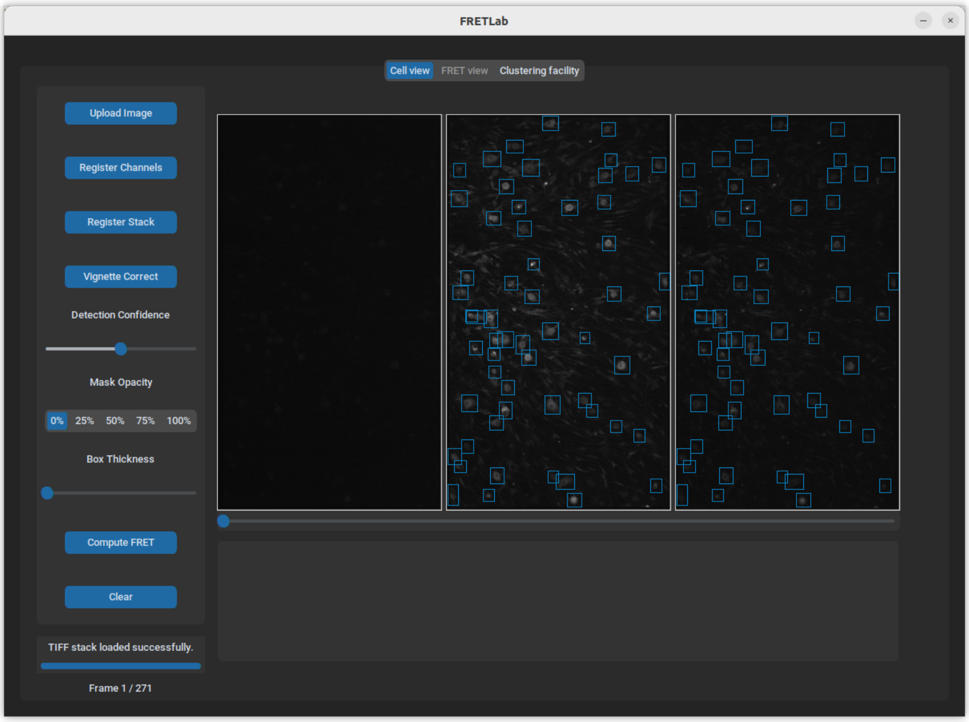


Figure 4.4: Demonstrating the effects of confidence thresholding. Both the cellular detections and segmentations can be thresholded in real time utilising the detection confidence slider. At top, a detection confidence of $\geq 0\%$. At bottom, a detection confidence of $\geq 50\%$.

Once the user has processed the image stacks and verified the segmentation masks, they may compute the FRET response. Once the *'Compute FRET'* button has been clicked, the tool will process all image stacks in order. At each time-step, the Hadamard product is taken between each binary segmentation mask and both the YFP and CFP channels. Then, the pixel ratios between the extracted neuronal crops are summed and averaged by the total number of pixels contained in the crops. The scalar value produced by this process is the drug response of each cell C_i at time-step t.

Once the FRET score has been computed for each cell across all time-steps, the interface is set to the second tab of the application, titled *'FRET view'*. As shown in Figure 4.5, the *'FRET view'* tab plots the data, and provides utilities for exporting both the raw FRET output and graphs. Four views are provided to aid the analysis of the data: the raw time series data for each cell across each frame in the acquisition, a smoothed version of the raw time series data, a heatmap, and finally a histogram which charts the average FRET ratio per cell over time.

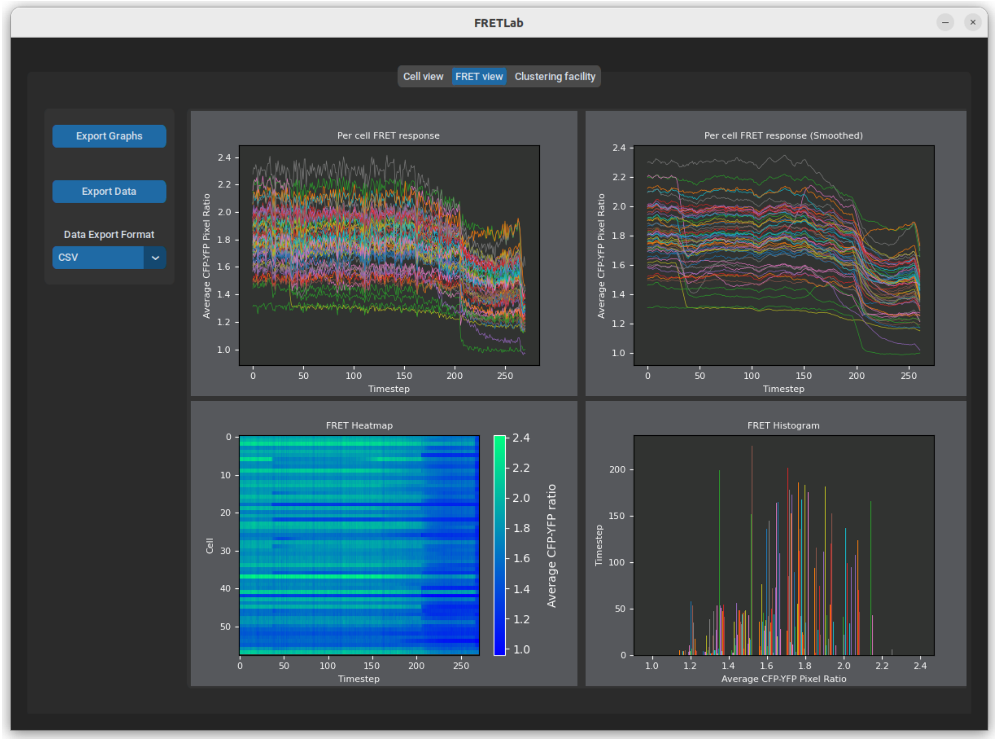
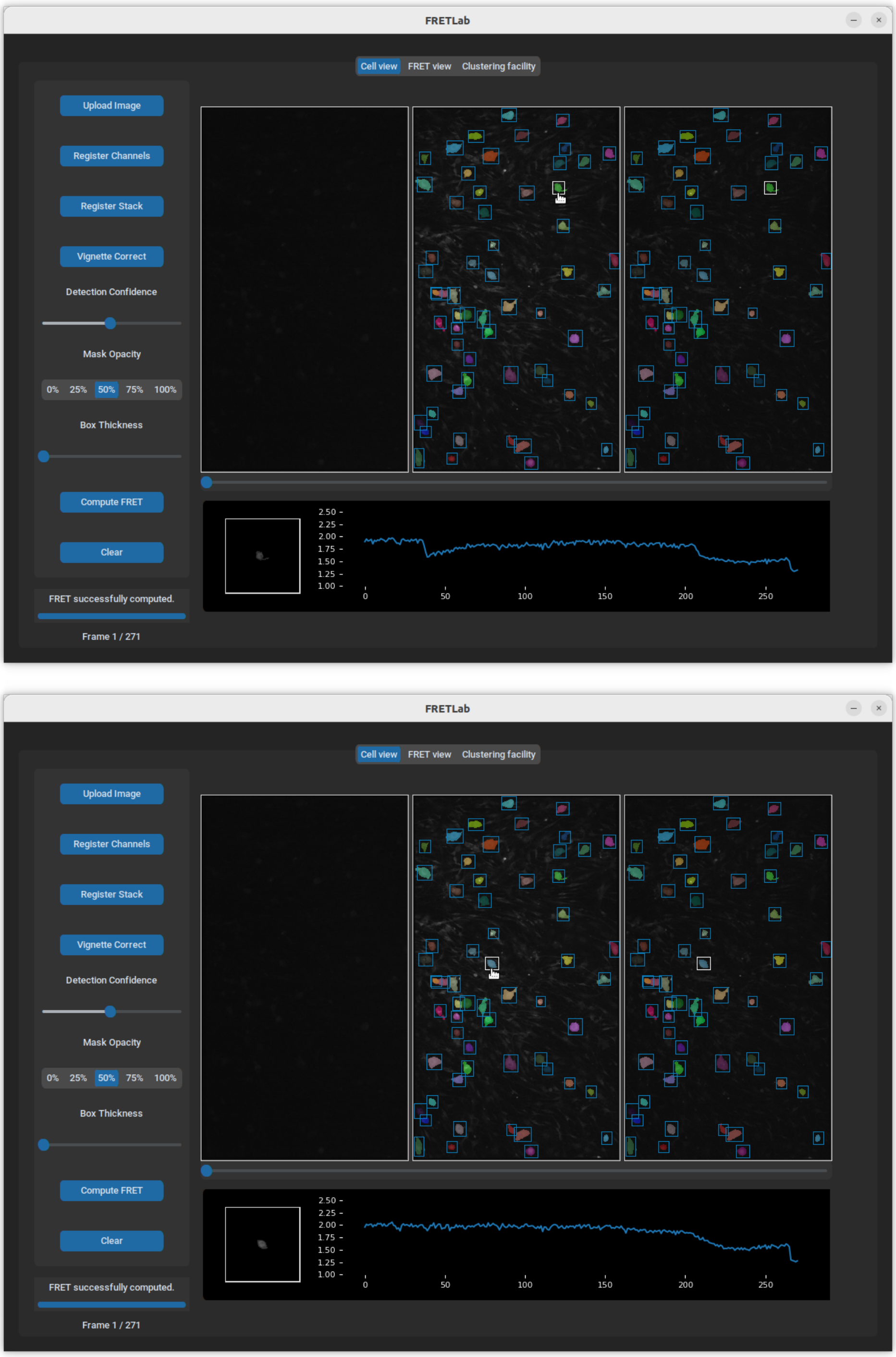


Figure 4.5: The *'FRET view'* tab for displaying and exporting FRET data.

Of particular interest when inspecting FRET data are cells which have atypical response profiles. Thus, as per Figure 4.6, on returning to the *'Cell view'* tab after FRET computation, the drug response of an individual cell can be visualised by hovering over it with the cursor. Paired with the ability to set the mask opacity to 0 and play the time lapse, it is possible to quickly discern whether an atypical response by any given cell is the consequence of noise, such as drift or occlusions, or a genuine readout.



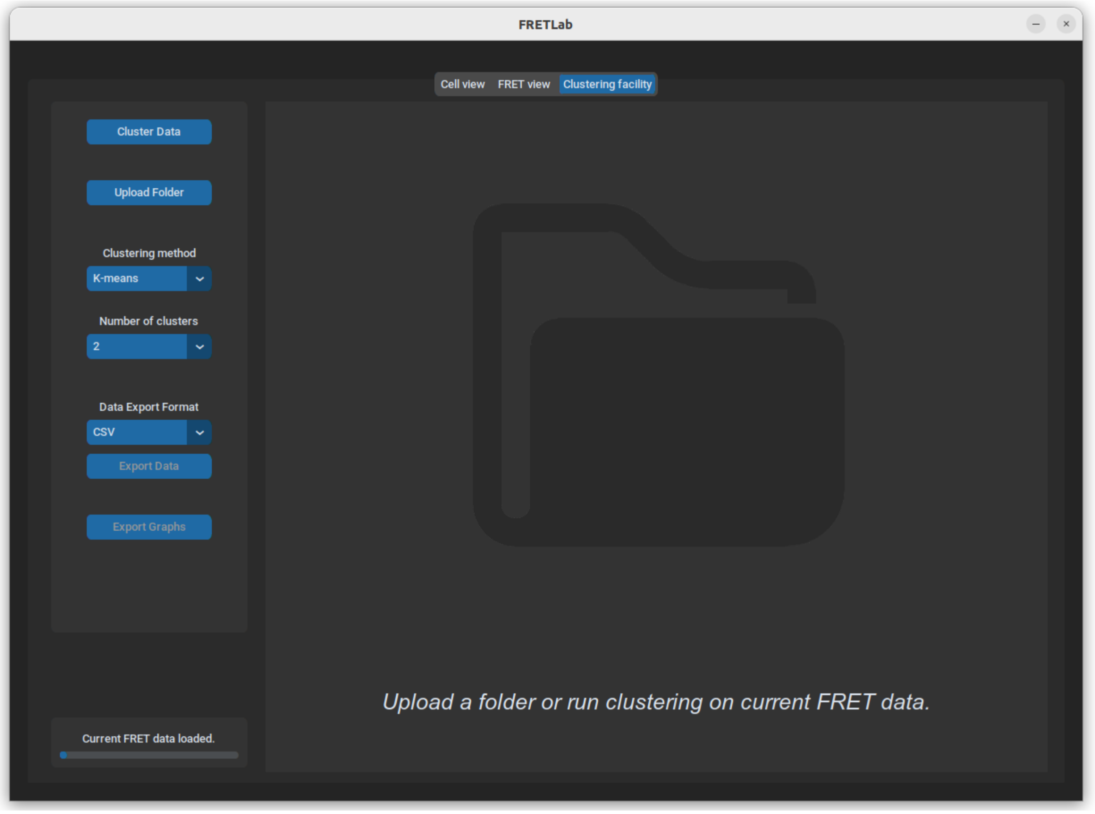
Mask Opacity: [✓ 50%]

FRET successfully computed.

Frame 1 / 271

Figure 4.6: Inspecting the reactivity of two different neurons using the *'Cell view'* tab. After the FRET score has been computed for each neuron, users may hover over each cell and visualise drug response in real time. A magnified neuronal crop accompanies the time series and the bounding box of the active neuron is highlighted.

The final tab is the clustering facility, which provides methods for clustering the data generated by the *'Cell view'* tab. Users may cluster time series data generated within the current session, or upload a folder of files with a uniform format [1] and cluster them as a whole [2]. Files may be those previously generated by FRETLab or another tool.

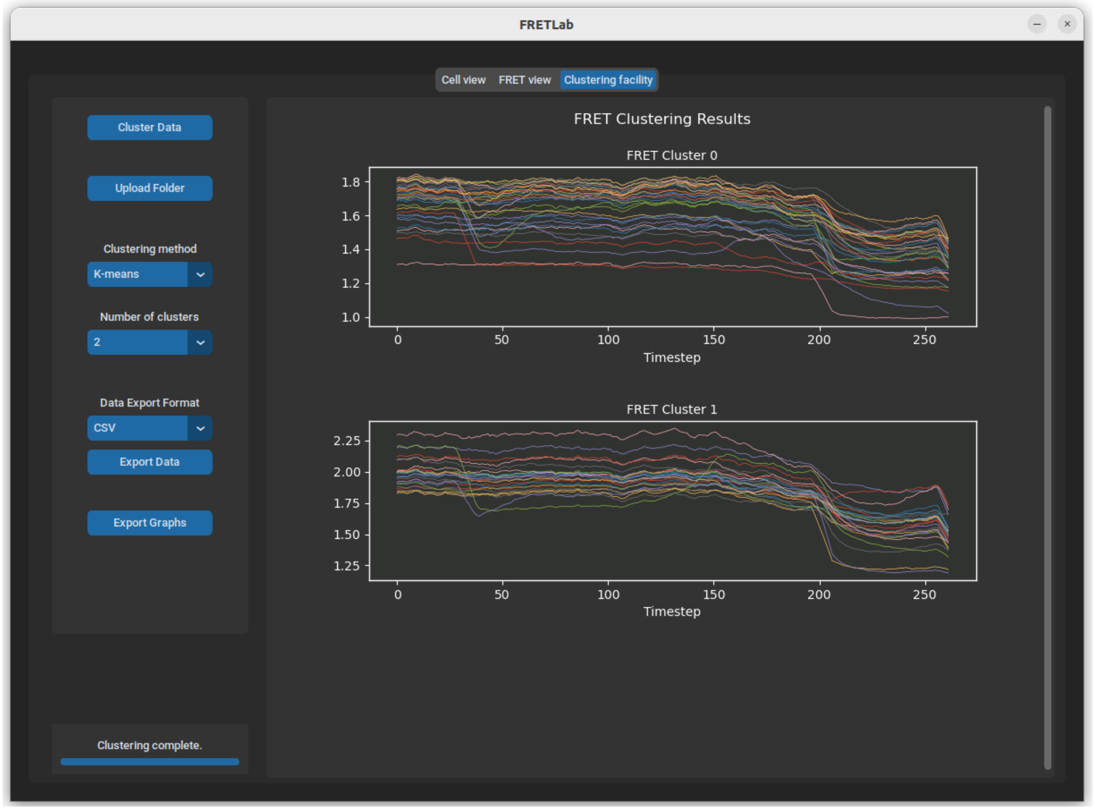


Upload a folder or run clustering on current FRET data.

Figure 4.7: The *'Clustering facility'* tab. On launching the tab for the first time, users are prompted to either cluster the data produced during the current session, if available, or upload a folder of previously generated FRET data as exported from the *'FRET view'* tab. If the latter option is selected, all files will be treated as a singular dataset and the clustering will be applied to this unified dataset. Options are provided for exporting the produced graphs and clustering data.

The clustering facility was used to provide an initial insight into the nature of the data and identify whether it would be feasible to further investigate the possibility of identifying meaningful response clusters in the FRET curves. Three clustering algorithms are currently available from the drop-down menu: k-means, k-medioids, and kernel k-means. Each method employs a shift invariant dynamic time warping metric. Further details on clustering are given in Section 4.2.4 and Chapter 5.

[1]Either .csv or .npy, being the options provided by the *'FRET view'* tab.
[2]This requires that the acquisitions be of the same length.



Clustering method: K-means

Number of clusters: 2

Data Export Format: CSV

Clustering complete.

Figure 4.8: Results from the clustering facility for the FRET computation performed in Figure 4.5. A drop-down menu is provided to enable the user to cluster the data up to a maximum of 10 clusters. This limit was imposed on the basis of biological domain knowledge of the maximum plausible number of groupings expected from the neurons.

4.2 System Architecture and Design

The FRETLab software is comprised of six core classes: *UI*, *Segmentor*, *Faster R-CNN*, *BASiC*, *FRET*, and *Cluster*, each contained within their own respective *.py* files. Figure 4.9 presents a class diagram highlighting the system architecture.

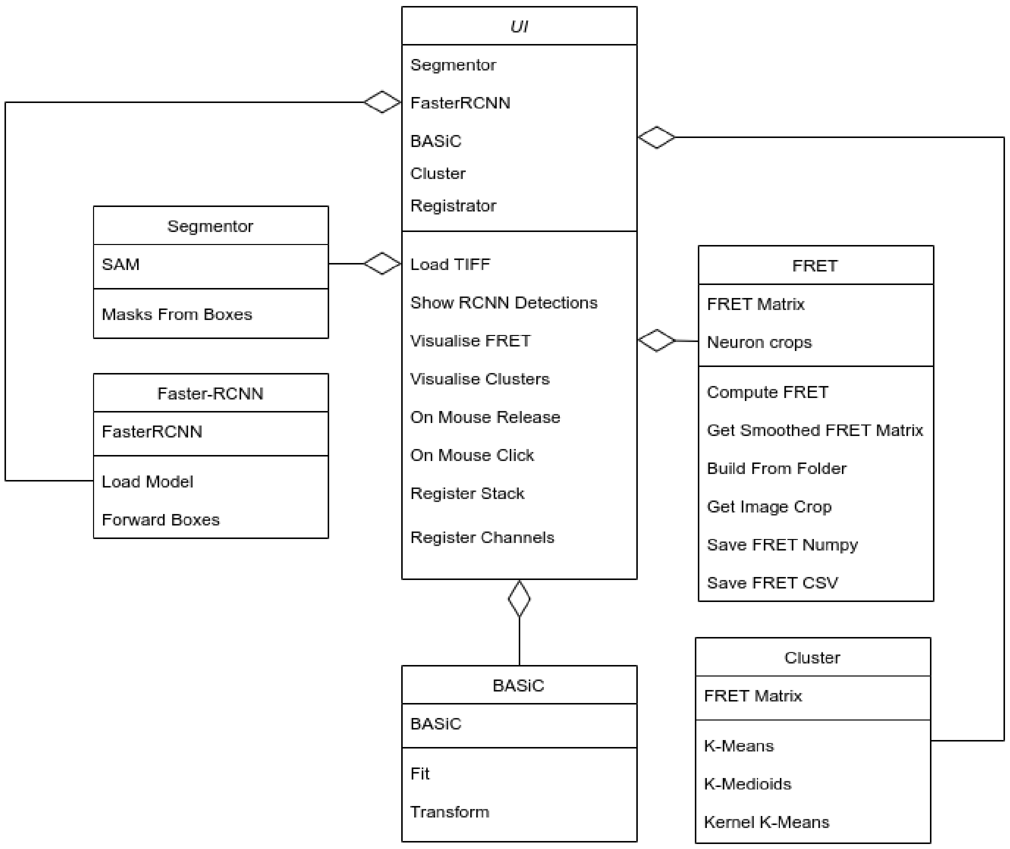


Figure 4.9: A high-level class diagram for the FRETLab software. For brevity and simplicity, only the core public methods for each class are displayed.

The central controller is the *UI* class, which invokes the remaining classes in response to user input. The core deep learning components are the *Faster R-CNN* and *Segmentor* classes, which are wrapper classes around the PyTorch Faster R-CNN and SAM models respectively. The *FRET* class is responsible for computing the raw and smoothed FRET matrices for each acquisition, which are then formatted and displayed by the *UI* class. Finally, the *Cluster* class receives FRET matrices and performs clustering, the results of which are relayed to the user through the UI class. In the following sections we will examine how the fine-tuned cellular object detector and segmentation models were developed. However, before delving more deeply into the machine learning components of FRETLab, we first turn to the datasets used to train the models.

4.2.1 Datasets

Before examining the deep learning components of FRETLab, it will be instructive to first critically examine the in-house and external validation datasets used to train and evaluate the two core models. Both datasets will be used in Sections 4.2.2, and 4.2.3 for detection and segmentation respectively. Further, Chapter 5 on time series analysis using the FRETLab tool will analyse the data generated from the in-house dataset.

In-House Neuron Dataset

The in-house dataset consists of 132 unlabelled TIFF image stacks available via the Imperial RDS. Each image contains approximately twenty to sixty neurons, as well as cellular debris and glial cells [3]. All image stacks were acquired by the Jones lab using the same microscope and fluorescent imaging procedure. The experiments are grouped under ExD3, ExF1, and the positive control drug, VeH. Each experiment consists of several fields of view (FOV) for each drug, with the typical number of FOVs being four or five. While procedurally consistent across experiments, given that the dataset was not originally acquired with machine learning in mind, there are several limitations:

- **Low image quality:** Some imaging acquisitions are of low quality due to equipment errors or changes in lighting conditions between experiments. While some imaging artefacts such as misregistration and vignetting are expected and may be addressed by a standard preprocessing pipeline, others are less readily corrected.
- **Non-standard series length:** The majority of the image stacks have been standardised to 271 time steps in length. However, many acquisitions, particularly those for the ExF1 drug, are much shorter in length. This issue will become particularly relevant to the matter of time series analysis in Chapter 5.
- **Inconsistency in positive controls:** As noted in Section 2.4, the experimental procedure for a FRET drug testing experiment within the Jones lab is to image neurons in a culture in four stages: a baseline stage, the drug introdcution stage (ExF1 or ExD3), followed by two periods in which positive controls are added. While the baseline and drug introduction stages were tightly regulated, the introduction of the controls occasionally varies significantly between experiments. Once again, this issue will inform the time series analysis experiments in Chapter 5.
- **Low quantity:** Perhaps the primary limitation of the dataset is the low total cell count. The total cell count is $\sim 4,000$, considerably lower than for analogous datasets, for which examples number in the tens or hundreds of thousands. However, this problem is partially offset by the highly homogeneous appearance of the neurons, and can be further mitigated by generous use of data augmentation.

The preceding observations were factored in when building the detection and segmentation pipelines, and will be alluded to throughout the remainder of the report.

[3] A glial cell is any non-neuronal cell in the brain that supports neuronal functions [61].

CellPose Dataset

CellPose [52] is a dataset of 70,000 segmentation masks over 608 images, and was released alongside the general purpose U-net based model for cellular segmentation described in Section 3.2.1. The authors collected the images from a variety of sources, with the majority being sourced from internet searches for keywords such as ‘cytoplasm’, ‘cellular microscopy’, or ‘fluorescent cells’. The dataset consists primarily of fluorescently labelled proteins, but also includes images of cells from brightfield microscopy and images of membrane-labelled cells. The authors further included a small set of images from other types of microscopy, as well as a small set of nonmicroscopy images that contained large numbers of repeated objects such as fruits, rocks, and jellyfish. The authors justify the inclusion of non-cellular objects by hypothesizing that the inclusion of such images in the training set would allow their U-net based network to generalize more robustly.

The dataset was chosen because of the diversity it offers. Further, the model released alongside the dataset is considered the state-of-the-art in general cell segmentation. This provides a strong benchmark against which to compare CellSAM, the fine-tuned Segment Anything Model which is discussed in Section 4.2.3.

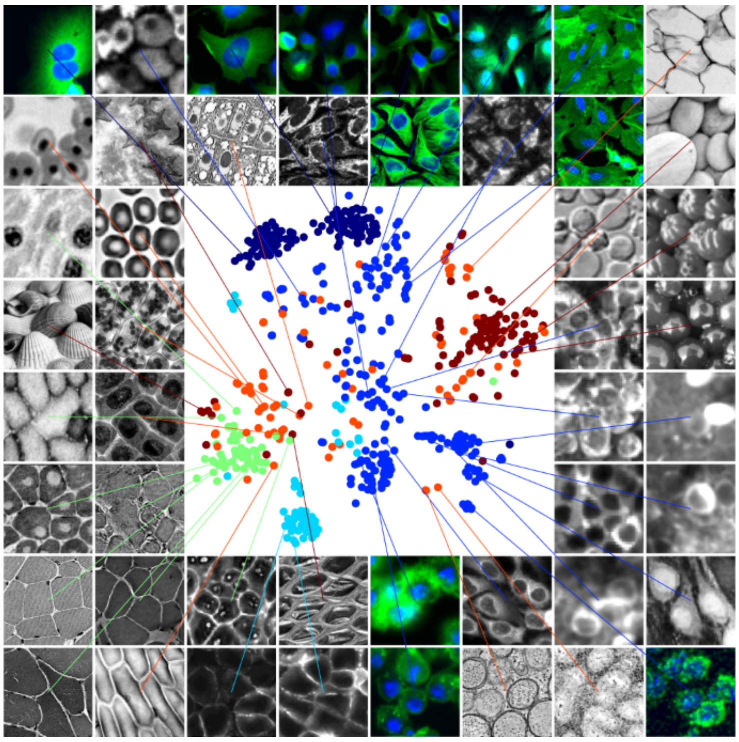


Figure 4.10: An example of the diversity of imaging protocols, culture densities, and cell morphologies present in the CellPose dataset [52]. Here the styles from all of the cells in the dataset were embedded using t-SNE. Each point represents a different image. The legend is as follows: dark blue, Cell Image Library; blue, cytoplasm; cyan, membrane; green, non-fluorescent cells; orange, microscopy other and red, non-microscopy.

Despite being one of the more general and accessible cell segmentation datasets available, the CellPose dataset suffers from several limitations which are addressed below:

- **Non-cellular images:** As previously mentioned, the CellPose dataset contains a number of non-cellular images. While the inclusion of such images is justified in the initial CellPose paper by the argument that they contain the same fundamental properties as cells, namely being self-contained, overlapping, and dense, this creates a clear bias in how suitable the dataset is for methods which look expressly for cells and cells alone. This point will be addressed further in Section 4.2.3.

- **Low image quality:** As the majority of the images acquired for the CellPose dataset were sourced from the internet via an image search, many of them contain artefacts. Examples include watermarks, filters, and noise introduced by up and downsampling many of the images to a standard resolution.

- **Unrepresentative images:** Many of the CellPose images are unrepresentative of real-world microscopy acquisitions. For instance, comparing the debris laden and imperfect images of Figure 2.4 from the in-house dataset to those of Figure 4.10, one may quickly identify that the CellPose images are those which are intended to accompany publications, and are therefore not entirely representative.

- **Degenerate masks:** As will be discussed in Section 4.2.3, the CellPose dataset was repurposed for promtable segmentation by converting all 70,000 segmentation masks into encompassing bounding boxes. These bounding boxes are of the form $(x_{min}(o_i), y_{min}(o_i), x_{max}(o_i), y_{max}(o_i))$ where o_i is the ith cell in an image and each function returns either the maximum or minimum x and y co-ordinate of the given cellular object. Thus, each box is the minimal area rectangle that encompasses all pixels belonging to the mask associated with the corresponding cell. A requirement of using bounding boxes to train object detection models is that $0 \leq x_{min} < x_{max}$, and $0 \leq y_{min} < y_{max}$, otherwise the area of the bounding box will be undefined. The conditions for both the x and y functions are violated however, as CellPose contains erroneous masks consisting of single pixels or straight lines – likely resulting from errors in the annotation process. Therefore, a secondary post-processing stage was required in which all bounding boxes violating the constraints above were removed.

- **Low cell count:** While substantially larger than the in-house neuron dataset, the number of data points offered by CellPose is still modest by modern deep learning standards.

While several other datasets exist for cellular detection and segmentation, with the largest being EVICAN [62] and LIVECell [63], these lack the diversity required to prove the generality of the method being developed. For example, while the LIVECell dataset contains a range of cell morphologies, all images were acquired using the same label-free, non-fluorescent imaging technique. Similarly, the EVICAN dataset contains only bacterial cells with simple morphologies, and is therefore not suitable for assessing the generalisability of a cell segmentation model.

4.2.2 Detection

The first step in constructing an automatic cellular segmentation pipeline using SAM was to establish the out-of-the-box performance of SAM on the in-house dataset. For macroscopic images of natural scenes, the impressive out-of-the-box performance of SAM in everything mode, as demonstrated in Figure 3.4, was suggestive that such a sampling based approach would be appropriate for the present task of segmenting cells too. However, as can be seen from Figure 4.11, the performance of the model on the in-house fluorescence microscopy dataset is respectable, but far from the human-assisted benchmarks. We may attribute this to the previously highlighted fact that SAM was trained on a dataset consisting primarily of photographs of real-world scenes. As such, the model excels at generating masks for macroscopic objects, and is not optimized for dealing with noisy microscopic images containing debris and poorly illuminated cells with ill-defined boundaries. We now turn to the three key failure modes of SAM for cell segmentation.

SAM Failure Modes

- **Overdetection $\implies$ Oversegmentation:** The first is overdetection leading to oversegmentation. Particularly evident in the first two columns of Figure 4.11 is the tendency of SAM to misinterpret the entire image background as an object and segment it out. A first remedy would be to mitigate this issue using heuristics such as size-based thresholding. However, as may be seen from Figure 4.11, partial background oversegmentation may also occur, making this a difficult task.
- **Overdetection $\implies$ Undersegmentation:** The second failure mode is overdetection resulting in undersegmentation. Observing the bottom entry of the second column of Figure 4.11, we observe that SAM occasionally generates masks corresponding to part of an cell of interest, or for small particulate debris.
- **Underdetection $\implies$ Undersegmentation**: The final failure mode is failing to detect a cell at all, resulting in no mask being produced for it; underdetection leading to undersegmentation. Using everything mode this may occur for two reasons: either no point in the sample grid fell over the surface of the cell, or one did and the mask quality threshold was too low and was subsequently removed.

Three potential approaches suggest themselves when observing the out-of-the-box performance of SAM on the in-house fluorescence microscopy data. The first approach would be to increase the sample density in everything mode. Indeed SAM's *'predictor'* class provides just such an option. However, as might be anticipated from Figure 4.11, using denser sampling is not effective for noisy images in which debris is present. An alternative method would be to create an interactive tool to allow users to provide bounding boxes for each cell in an image, and feed threse through to SAM to produce an output. The limitation of this approach is time, as biologists would be required to label tens to hundreds of cells per image. The final approach is to utilise the zero-shot prompt capabilities of SAM by training an object detector to produce bounding boxes for each cell and using these predictions as prompts for SAM. This is the approach taken by FRETLab, and is the subject to which we turn next.

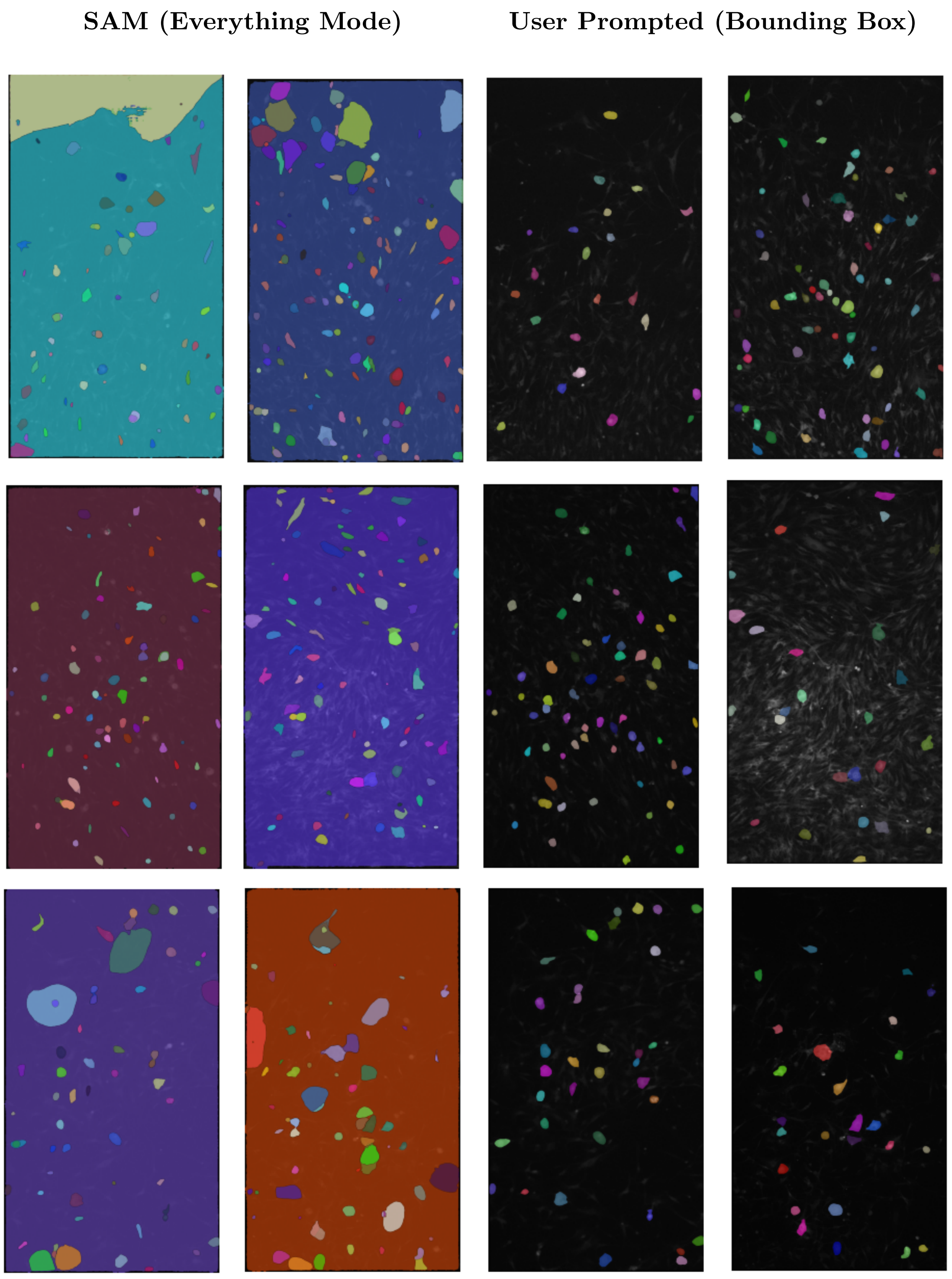


Figure 4.11: Performance of SAM on a sample of neuronal image acquisitions from the RDS (first two columns) against the same images with user provided bounding box prompts (third and fourth columns). All images were generated using the ViT-H SAM checkpoint, the most powerful version of the model.

Dataset Labelling and Preprocessing

Having established that neither the default version of SAM alone nor SAM aided by human provided prompts were adequate for the needs of the project, the next task was to investigate the application of machine learning for localising and bounding cells. Producing a model that is capable of reliably localising cells with tight bounding boxes was the first step in constructing a machine learning pipeline for cellular segmentation. A prerequisite for developing such a model was to create a dataset to enable the testing of various methods for cellular detection.

The available data for the present project takes the form of ∼300GB of raw and unprocessed TIFF image stacks acquired by the Jones lab. These were made available via the Imperial Research Data Store (RDS). The first frame of every acquisition was extracted from each image stack and converted to a JPEG format to ensure compatability with online dataset labelling tools. The Roboflow software [64] was used to create the dataset. In total, 132 images were annotated, with ∼4000 cells circumscribed with bounding boxes. Of these annotated images, 88 were reserved for the training set, 25 were reserved for the validation set, and the remaining 19 constituted the test set. This split was determined on the basis of the number of cells contained in the images, with the divison being an 80:10:10 assignment for the number of annotations in the training, validation, and test sets respectively. Once constructed, the dataset was exported into a Pascal VOC xml format [65]. Pictured in Figure 4.13, Pascal VOC is a standard format commonly used to train object detection models.

As we will see in future sections, though modest in size, the in-house neuron dataset was effective in demonstrating the efficacy of using an object detector to localise cells and forwarding these predictions as prompts to SAM. We turn next to the method by which the data was annotated.

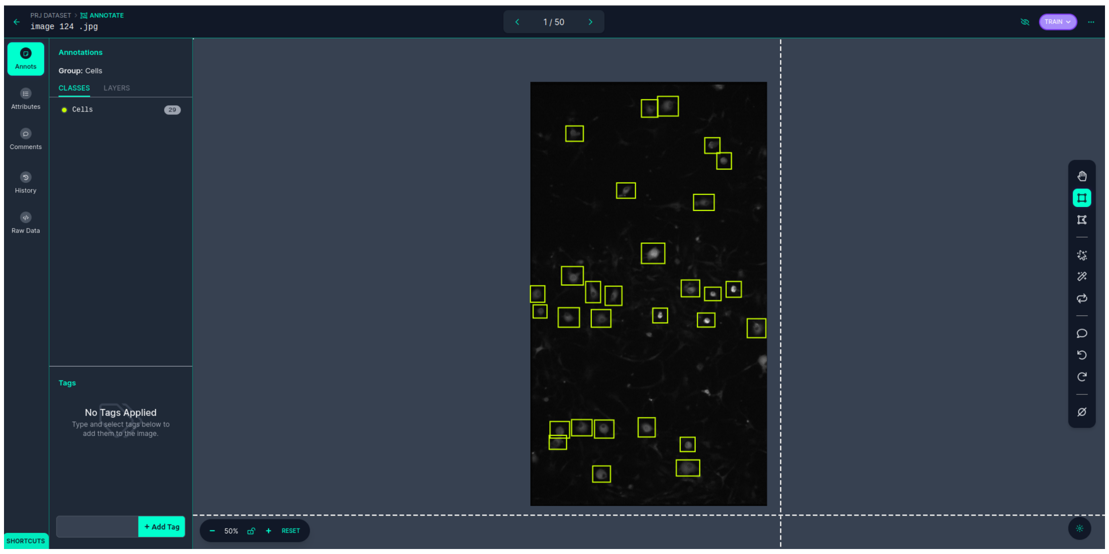


Figure 4.12: Illustrating the annotation process using the Roboflow platform.

To ensure the development of a high quality dataset, a protocol was developed by which the raw microscopy images were processed and annotated. The protocol for the construction of the dataset was as follows:

- All non-overlapping cells with clearly visible nuclei and membranes were circumscribed by a bounding box.
- In such cases in which the boundaries of a cell's extent were not clear, leading to material ambiguity as to where the cell membrane terminates, these cells were not annotated.
- Overlapping cells were not annotated unless their overlap concerned only their dendrites. This is because cells which overlap in their central regions are not of use in calculating the FRET metric, as overlapping clusters of two or more cells will represent the reactivity of multiple cells as that of a single cell.
- All bounding boxes were drawn so as to encompass one single cell, with a margin of a few pixels width on all sides. It was permitted for two bounding boxes to overlap, and possibly even to contain the partial extent of another cell, so long as it was apparent which cell was the subject of the annotation.
- All annotated images were checked in a second pass over the dataset by the author for quality assurance.

```
<annotation>
        <folder></folder>
        <filename>image−0.jpg</filename>
        <path>image−0.jpg</path>
        <source>
                <database>roboflow.ai</database>
        </source>
        <size>
                <width>825</width>
                <height>1478</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>Cells</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <occluded>0</occluded>
                <bndbox>
                        <xmin>407</xmin>
                        <xmax>467</xmax>
                        <ymin>369</ymin>
                        <ymax>419</ymax>
                </bndbox>
        </object>
```

Figure 4.13: A sample of the Pascal VOC xml format in which the bounding boxes for the neuron dataset are stored. The dataset can be found in the *Neuron Dataset* folder.

Object Detection Metrics

Before quantitatively evaluating various methods for cellular object detection against the in-house dataset, it will be necessary to first briefly review object detection metrics. The object detection literature contains a number of evaluation metrics, with the most commonly used being variants of accuracy, precision, and recall. All three metrics are defined in terms of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Accuracy, given by Equation 4.1, tells us the overall correct detection rate of the model. However, while appropriate for single class problems such as the present task, accuracy alone is often an unreliable metric on account of being susceptible to the total number of predictions made. For this reason, we also consult two further metrics, precision and recall.

$$Accuracy = \frac{TP + TN}{TP + FP + FP + FN} \tag{4.1}$$

Precision, defined as the ratio of the true positive predictions to the total number of positive predictions per Equation 4.2, is a representation of the confidence we may have in positive predictions produced by a model. In the context of object detection, precision is the answer to the question as to how much trust we should have that a bounding box represents an object when a model predicts one.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Relatedly, but in contrast to precision, recall is defined as the ratio of the total true positive detections to the total number of ground truth bounding boxes. Recall answers the question as to how many objects that are actually present in an image are detected by the model.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

The precision-recall (PR) tradeoff can be visualised as a curve which shows the precision of a model at various recall intervals. We can think of this as a summary of how precise the model is when we demand a certain level of correct detections. A method we can consult to summarise this tradeoff is average precision (AP). Average precision summarizes the precision-recall curve to a single scalar value. Per Equation 4.4, the average precision is calculated as the area under the precision-recall curve and reflects how well a model maintains a high precision while detecting objects at different levels of recall. The metric is bounded between 0 and 1 and is high when both precision and recall are high, and low when either of them is low across a range of confidence threshold values. Average precision considers both false positives and false negatives and provides a more comprehensive view of a model's performance compared to accuracy alone.

$$AP = \int_0^1 p(\mathrm{r})\ dr \tag{4.4}$$

Average recall, similar to average precision, is calculated by averaging recall values at different precision thresholds. Given in Equation 4.5, average recall represents how well a model can identify objects across varying levels of confidence, helping to understand how reliably the model detects objects at different precision levels.

$$AR = \int_0^1 p(\mathrm{p})\ dp \tag{4.5}$$

Maximum average precision (mAP) and maximum average recall (mAR) are another pair of commonly used metrics. Defined as the average AP and AR across all classes, mAP and mAR are given by:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \qquad mAR = \frac{1}{N}\sum_{i=1}^{N} AR_i \tag{4.6}$$

where i represents one of N possible classes. However, as we are presently in a binary classification setting with only one object class, mAP and mAR reduce to AP and AR respectively. Therefore we will not be consulting these metrics in the following analysis. Similarly, other popular multi-class metrics will not be used.

When we consider that a model's predicted bounding box for an object may not be perfectly aligned with a ground truth label but nevertheless be an accurate detection in a meaningful sense, the question as to how a bounding box can be classified as a true positive, true negative, false positive or false negative detection naturally arises. For this we use the intersection over union (IoU) metric, also known as the Jacquard index, defined for any two bounding boxes A and B as:

$$\mathrm{IoU} = \frac{A \cap B}{A \cup B} \tag{4.7}$$

where $\mathrm{IoU} \in [0, 1]$. Intuitively, the IoU of two bounding boxes tells us their degree of overlap, where 1 represents two identical detections and 0 represents two entirely disjoint bounding boxes. Equipped with this metric, we may use thresholds to stablish what IoU a detection is required to have with another ground truth bounding box in order to be considered a true positive detection. Many such thresholds are in use [66], with 0.5 and 0.75 being common benchmarks.

In evaluating cell detection baselines and the performance of deep learning models, thresholds will be denoted using subscripts. For example, $AP_{0.5}$ represents the average precision of a model with an IoU threshold of 0.5 consistuting a true positive. Colon notation may also be used to represent the average of a metric over a range of thresholds. For instance, $AP_{0.5:0.95}$ represents the mean of the average precision in the interval $[0.5, 0.95]$ using thresholds with a step size of 0.05 between each.

Cell Detection Baselines

Before establishing the need for a deep learning approach, several classical computer vision baselines were first trialled. Table 4.1 summarises the results of each method.

The first method trialled was thresholding, the technique used to localise cells in the current image analysis pipeline and described in Section 2.4. Thresholding with a fixed intensity parameter established by the author through empirical experimentation proved to be reasonably successful. However, as there are frequent non-uniform changes in intensities due to changing experimental conditions, the method was inconsistent between images. In an attempt to guard against this, several other variants of thresholding were tested, including adaptive Gaussian thresholding [67] and adaptive mean thresholding [68], both of which estimate intensity thresholds on local image patches.

A more advanced baseline in the form of the circular Hough transform was used next. The circular Hough transform is a feature extraction technique used in digital image processing for detecting approximately circular objects in images [69]. Since the neurons being tested by the Jones lab are highly circular, per Figure 2.4, this method was thought to be appropriate for the task at hand.

Finally, blob detection was tested. Blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or colour, compared to surrounding regions [70]. This method suffered from overdetection, as there are many small particles in the in-house dataset images which were inadvertently detected.

Method	$\mathrm{Pr}_{0.5}$	$\mathrm{Pr}_{0.75}$	$\mathrm{Re}_{0.5}$	$\mathrm{Re}_{0.75}$
Circular Hough Transform	**0.63**	**0.58**	0.60	0.58
Thresholding	0.57	0.54	**0.78**	**0.70**
Adaptive Gaussian Thresholding	0.41	0.36	0.58	0.52
Adaptive Mean Thresholding	0.44	0.39	0.46	0.40
Blob Detection	0.27	0.18	0.30	0.27

Method	$\mathrm{Acc}_{0.5}$	$\mathrm{Acc}_{0.65}$	$\mathrm{Acc}_{0.75}$	$\mathrm{Acc}_{0.85}$
Circular Hough Transform	0.55	0.49	0.37	0.18
Thresholding	**0.59**	**0.52**	**0.41**	**0.29**
Adaptive Gaussian Thresholding	0.50	0.44	0.41	0.21
Adaptive Mean Thresholding	0.49	0.42	0.24	0.19
Blob Detection	0.33	0.29	0.21	0.21

Table 4.1: Object detection results for various cell detection methods.

As can be seen from Table 4.1, the circular Hough transform and regular thresholding significantly outperform the remaining baselines. However, with neither method achieving an overall accuracy of $\geq 60\%$ nor a precision above 63%, none of the baseline methods perform nearly well enough for a producion platform. Further, while suitable for the neurons under study, the circular prior the Hough transform imposes is not suitable for generalising to other cell types. Similarly, the detections of the thresholding baseline required filtering using bounds for the minimum and maximum area.

Cellular Detection with Deep Learning

Once it had been established that classical algorithms were unsuitable for the present task, deep learning methods were tested. Per Section 2.5.3, a common method of approaching tasks such as object detection is to fine-tune exisiting convolutional architectures for domain specific applications. Table 4.2 summarises the performance of the models tested, namely Faster R-CNN, RetinaNet, FCOS, SSD, and SSDLite. Further, given that the FRETLab application is intended for low-resource settings, Table 4.2 additionally summarises the model sizes, inference speeds, and resource constraints.

Method	Backbone	$AP_{0.5}$	$\mathrm{AP}_{0.75}$	$\mathrm{AP}_{0.95}$	$\mathrm{AP}_{0.5:0.95}$
Faster R-CNN	ResNet50	**0.903**	**0.461**	**0.214**	**0.415**
Faster R-CNN	MobileNet V3 Large	0.875	0.376	0.205	0.391
RetinaNet	ResNet50	0.831	0.310	0.198	0.357
FCOS	ResNet50	0.811	0.255	0.201	0.334
SSD	VGG16	0.766	0.276	0.276	0.318
SSDLite	MobileNet V3 Large	0.728	0.221	0.180	0.301

Method	Backbone	$Acc_{0.5}$	$\mathrm{Acc}_{0.75}$	$\mathrm{Acc}_{0.95}$	$\mathrm{Acc}_{0.5:0.95}$
Faster R-CNN	ResNet50	**0.917**	**0.855**	**0.765**	**0.815**
Faster R-CNN	MobileNet V3 Large	0.882	0.823	0.741	0.801
RetinaNet	ResNet50	0.831	0.795	0.732	0.778
FCOS	VGG16	0.847	0.805	0.729	0.746
SSD	VGG16	0.810	0.798	0.766	0.732
SSDLite	MobileNet V3 Large	0.740	0.718	0.670	0.699

Method	Backbone	Parameters	Memory	Inf Speed (s)
Faster R-CNN	ResNet50	41,299,161	180MB	1.2
Faster R-CNN	MobileNet V3	18,930,229	85MB	0.8
RetinaNet	ResNet50	32,168,694	160MB	0.6
FCOS	ResNet50	32,064,455	160MB	0.6
SSD	VGG16	23,745,908	80MB	0.4
SSDLite	MobileNet V3 Large	**3,708,680**	7MB	**0.3**

Table 4.2: Object detection results for various cell detection methods. Inference speed was computed as the average time for a forward pass on a CPU.

Per Table 4.2, a Faster R-CNN with a ResNet50 backbone outperforms all other object detectors tested in every category except inference speed and model size. However, as the FRETLab pipeline caches detections, the object detector needs to run only once, and the marginal difference in inference speed can be discounted against the loading time of the TIFF stacks. As such, the Faster R-CNN with a ResNet50 backbone was selected as the primary cell detector. Note the steep decline in average precision performance between $IoU = 0.5$ and $IoU = 0.75$ in each case. We may attribute this to the small size of the ground truth bounding boxes, as each cell has a roughly 15 pixel radius.

4.2.3 Segmentation

Ground truth segmentations are not available for the in-house neuron dataset. Thus, to quantitatively evaluate the performance of CellSAM for the task of cellular segmentation, the CellPose dataset was used. However, before discussing how CellSAM was trained and evaluated, it will be instructive to first examine segmentation metrics.

Segmentation Metrics

As for the task of detection in Section 4.2.2, evaluating cell segmentation performance leans upon accuracy, precision, and recall. Here again a threshold is required using the intersection over union measure. In the context of segmentation masks, IoU remains defined per Equation 4.7, with IoU $\in [0, 1]$. Now, however, the sets A and B contain coordinates for each pixel belonging to the predicted and ground truth masks respectively.

Similarly to IoU, the Dice similarity coefficient is another measure of overlap between the predicted and ground truth regions. Per Equation 4.8, it is calculated as twice the intersection divided by the sum of the sizes of the predicted and ground truth regions:

$$DSC(G, S) = \frac{2|G \cap S|}{|G| + |S|} \tag{4.8}$$

where G and S are ground truth and predicted masks respectively. The main difference between IoU and Dice is in the way they handle the denominator. In IoU, the denominator includes the union of the predicted mask and ground truth mask areas, whereas in Dice, the denominator is the sum of the areas of the predicted mask and the ground truth mask. As a consequence, IoU tends to be more sensitive and penalizes under and oversegmentation more [71], whereas the Dice score is more robust to false positives and false negatives. This is particularly important in cell segmentation tasks, since it is necessary for metrics to be reliable for minute cells with low mask volume.

Turning to metrics which assess the fidelity of segmentation boundaries, the average symmetric surface distance (ASSD), given by Equation 4.9 [72], is particularly useful when accurate delineation of object boundaries is critical. Defined as the average of the closest distances from points on the boundary of a predicted mask to the boundary of the ground truth mask and vice versa, this metric provides insight into how well the predicted boundary matches the ground truth boundary. Low ASSD values indicate a strong match between the predicted and ground truth boundaries, while higher values suggest that there are significant discrepancies between the two. Here $d(g, S)$ denotes the distance between pixel $g \in G$ and its closest point on the surace of S.

$$ASSD = \frac{1}{|S| + |G|} \times \left(\sum_{s \in S} d(s, G) + \sum_{g \in G} d(g, S) \right) \tag{4.9}$$

When referring to ASSD in the following discussion, we will be using the Chebyshev distance, defined for two points (x_1, y_1) and (x_2, y_2) as $\max(|x_1 - x_2|, |y_1 - y_2|)$. This can be thought of as the length of the shortest path from one pixel location to another using an eight-way neighbourhood.

CellSAM: Fine-Tuning SAM for Cellular Segmentation

The process of fine-tuning SAM for cellular image data was inspired by MedSAM [73], a variant of the Segment Anything Model which has been adapted and fine-tuned for biomedical image segmentation. Here, the authors simplify the task of fine-tuning SAM for various anatomical segmentation tasks by focusing only on the mask decoder.

As discussed in Section 3.2, SAM is composed of three components: an image encoder, a prompt encoder, and a mask decoder. Taking the form of a large vision transformer [4], the image encoder is the computational bottleneck, taking several seconds to compute an image embedding. Both the prompt encoder and mask decoder, however, are lightweight MLPs. The mask decoder is reponsible for 'decoding' the combination of the image embedding and prompt embedding into a plausible segmentation. The moderate size of the mask decoder coupled with its responsibility for producing the final masks suggests a training approach in which the remainder of the model is frozen.

An alternative method to fine-tune SAM is to freeze only the prompt encoder while training the image encoder and mask decoder simultaneously. This is the approach taken in the second version of MedSAM, and is justified as an extension over the previous approach by the development of higher-quality embeddings for domain specific tasks. Both approaches were tested as part of the development of the cellular segmentation model we will hereafter refer to as CellSAM. However, due to GPU resource restraints, only the former method was able to be fully developed upon and evaluated, as training even the ViT-B image encoder quickly becomes infeasible at modest batch sizes. Figure 4.14 provides a schematic for fine-tuning SAM by training only the mask decoder.

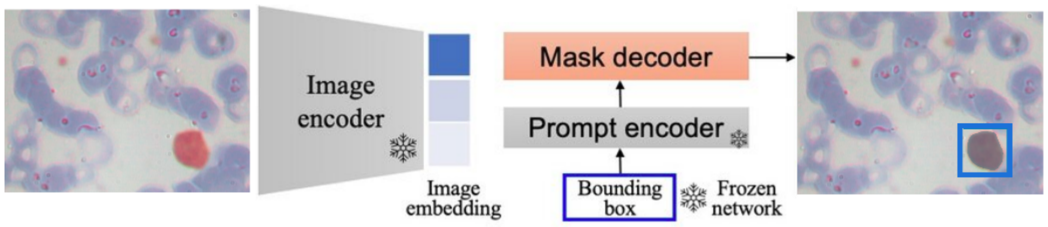


Figure 4.14: A schematic for fine-tuning CellSAM by freezing the image encoder and prompt encoder. Only the mask decoder is trained. Inspired by and adapted from [73].

The training procedure was as follows. First, the minimum area bounding box circumscribing each ground truth segmentation was created for every cell in the training set. Then, for each image in a batch, a single cell's bounding box would be sampled uniformly at random. This was used as the prompt for CellSAM, which would then generate a predicted mask. These ground truth and prediction pairs were then used to fine-tune the mask decoder. When testing the method, two approaches were trialled: an oracular method, in which the minimum area bounding box encompassing each ground truth mask was provided as a prompt, and a detection method, in which prompts were provided by a Faster R-CNN trained on the training set. The purpose of performing both experiments was to separate out segmentation performance from prompt quality.

[4]91M, 308M, 636M parameters for ViT-B, ViT-L, and ViT-H respectively.

To achieve efficient fine-tuning, two other techniques are adopted from the MedSAM paper. The first is pre-computing the image embeddings. Given the cost of repeatedly generating these embeddings while training, the procedure was to instead pre-compute all of the embeddings in advance and load them into memory for efficient access and batching. The second technique is to encourage small-scale translation invariance by randomly perturbing the four corners of each ground truth bounding box. By perturbing each corner of the ground truth bounding boxes the model becomes robust against prompts which do not perfectly encompass the cell of interest. This was done using a perturbation factor of five pixels, a setting which was found to work well empirically through experiments with this parameter. Larger perturbations can result in the model being prompted for the wrong cell, as the dataset contains cells which are on the order of only tens of pixels wide.

Concerning preprocessing, as the images in the CellPose dataset are not of a standard size, all ground truth training and test masks were resized to shape 256×256 using nearest neighbour interpolation. Further, as noted in Section 4.2.1, degenerate masks which do not correspond to accurate segmentations were removed from the training and test sets. Non-cellular images were retained to avoid altering the test set and to facilitate a fairer comparison between the CellPose model and CellSAM.

The training hyperparameters are given in Table 4.3 and were modified on the basis of the method being tested. Due to the highly variegated morphologies in the dataset, higher batch sizes proved to be the most decisive parameter adjustments in reducing the loss during training. Further necessitating larger batch sizes was the presence of noisy annotations, generated either as part of the original labelling process or as an inevitable consequence of rescaling the segmentation masks to a standard size.

Batch Size	Training Epochs	Learning Rate	Optimizer	Weight Decay
64	50	0.0001	Adam	0.0001

Table 4.3: Training hyperparameters for the final version of CellSAM.

Following the recommendation of a recent survey [74] and the MedSAM paper, the loss function used is the Dice cross entropy loss, an unweighted sum of the conventional Dice loss and cross entropy loss. This hybrid loss function was found to to work well empirically by the author. The Dice and cross entropy losses are given by:

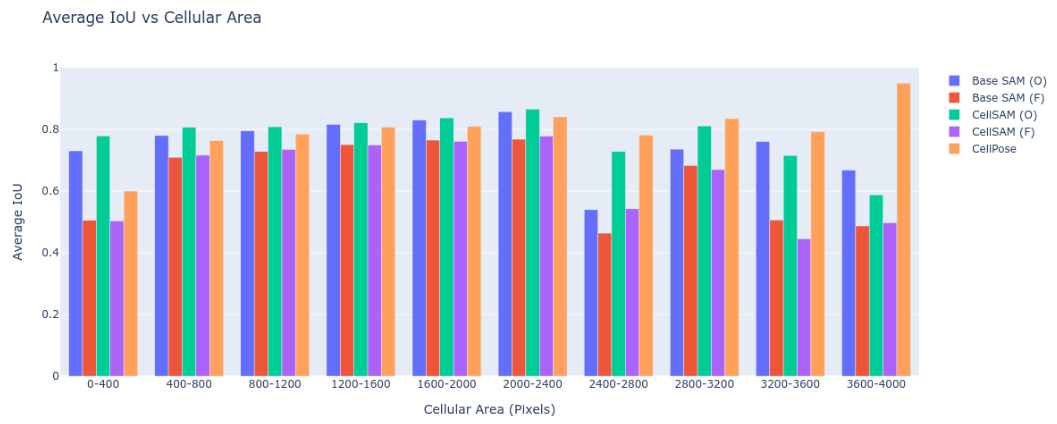
$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} g_i \log s_i \qquad L_{DICE} = 1 - \frac{2\sum_{i=1}^{N} g_i s_i}{\sum_{i=1}^{N}(g_i)^2 + \sum_{i=1}^{N}(s_i)^2} \tag{4.10}$$

where N is the number of pixels in the image I, and g_i, s_i are the predicted segmentation and ground truth of pixel i, respectively. Thus, the final loss L is simply the unweighted combination of L_{CE} and L_{DICE} as:

$$L = L_{CE} + L_{DICE} \tag{4.11}$$

Evaluation

Here the focus is on benchmarking CellSAM against the base implementation of SAM and the CellPose model on the CellPose dataset. Note that the CellPose authors use several test time enhancements to further increase the predictive power of their model in their original paper, including 'test time resizing, ROI quality estimation, model ensembling, and image tiling with overlaps'. These methods are not replicable, thus all models were assessed using unprocessed test set images resized to 256×256. The following results are for the oracular (O) and detection (F) methods previously described.



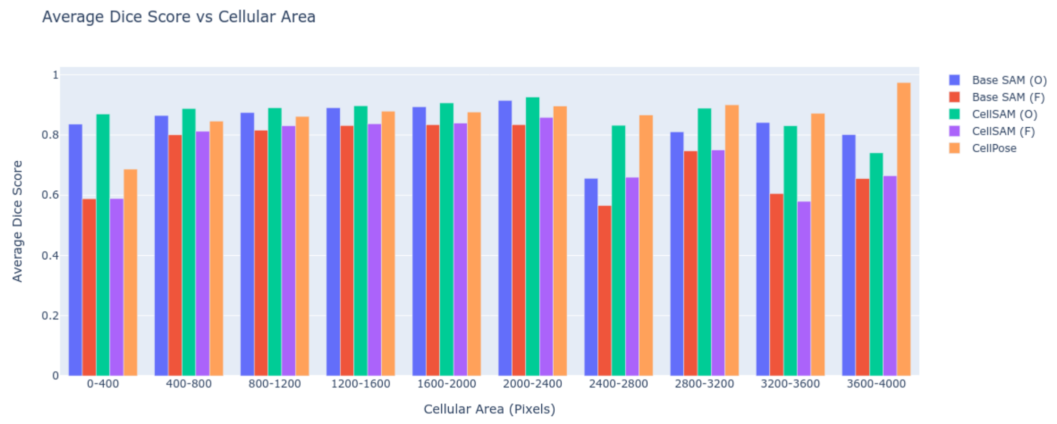
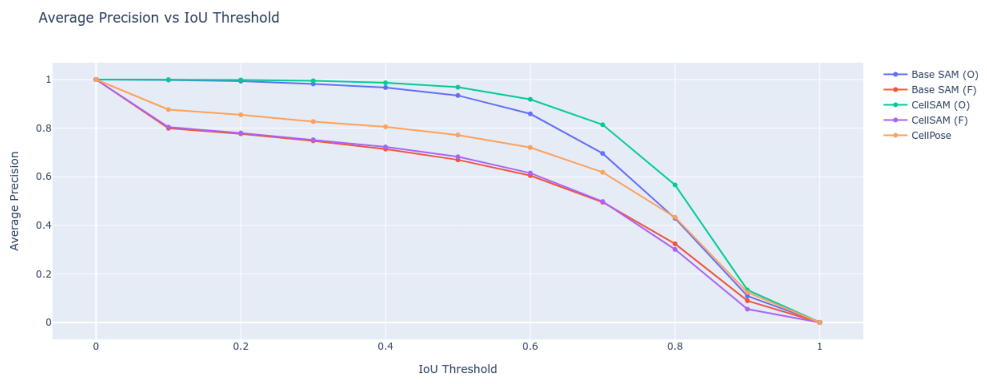


Figure 4.15: Visualising the average intersection over union and Dice similarity score as a function of cellular area for the base implementation of SAM, CellSAM, and CellPose.

As can be seen from Figures 4.15 and 4.16, in the oracular setting, denoted (O), both the base implementation of SAM and CellSAM are competitive with and in many cases outperform the current state-of-the-art in general cell segmentation, CellPose. Charting the average IoU and Dice scores against cellular area in Figure 4.15, we can see that CellSAM outperforms both the base implementation of SAM and CellPose consistently for cells with pixel areas in the interval $[0, \sim 2400]$. Similarly, looking at the average precision against IoU threshold at the top of Figure 4.16, CellSAM continues to be more performant than both the base implementation of SAM and CellPose at all values ≥ 0.3.



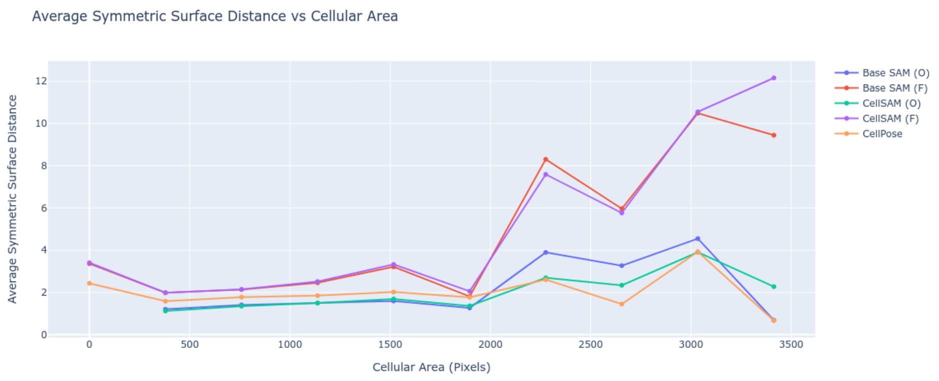


Figure 4.16: At top, the average precision of each model as a function of the IoU threshold. At bottom, the average symmetric surface distance as a function of cellular area.

Next, turning to the quality of the segmentation boundaries, we consult the bottom plot of Figure 4.16, where we can see the ASSD against cellular area. In the oracular setting, both SAM and CellSAM perform well, with an average ASSD of never more than 5 pixels. Through the author's experimentation, a key failure mode of SAM is the model's tendency to only segment out cell nuclei when provided with a prompt that encompasses a whole cell. This tends to occur for larger cell types, as SAM was trained on a segmentation dataset which contained this task. This failure mode is not present in CellSAM, as the model has been tuned to segment whole cells.

While expectedly worse, the performance of the models when provided with detections from a Faster R-CNN, represented by (F), remain competitive with CellPose for the IoU, Dice, and AP metrics, but are considerably poorer for ASSD. However, per Section 4.2.1, we may attribute this to the fact that the dataset contains non-cellular objects and class imbalances between cell types. Consequently, a detector simply cannot localise many of the objects in the test set, making an equal comparison difficult. The true performance of the full pipeline on an unbiased dataset is likely to fall between the two approaches, with detection being the bottleneck to performance. In summary, while the base performance of SAM is impressive, with changes only to the mask decoder performance can be improved appreciably. Perfectly prompted, both SAM and CellSAM outperform CellPose and achieve exceptional average precision even at high IoU thresholds. Training the image encoder would likely enhance performance further.

4.2.4 Clustering

Several clustering algorithms are available from within the clustering facility tab, namely k-means, k-medioids, and kernel k-means. These were implemented using the *sktime* [75] package, an extension of *sklearn* [76] providing utilities for time-series clustering.

Key to performing time-series clustering is the selection of an appropriate distance metric. Conventional distance measures over vectors, such as Euclidean distance, are inappropriate in the case of time-series clustering. This is because Euclidean distance is not shift invariant, and will therefore produce large values for identical time series which are shifted in time. For the task of comparing drug responses, the shape of a time series is of primary concern. However, slight changes in experimental conditions, such as non-uniform diffusion of a drug, may cause shifts in the time at which a cell's curve changes shape. This is a confounding variable, and requires a distance metric invariant to such experimental variability. Dynamic time warping has been implemented as a solution to this problem. The method operates by comparing the Euclidean distance between X and Y across all possible discrete time shifts between them and taking the minimum.

```
Dynamic-Time-Warping($X, Y$)
 1  $n \leftarrow$ length of $X$
 2  $m \leftarrow$ length of $Y$
 3  $\triangleright$ Initialize the cost matrix
 4  for $i \leftarrow 0$ to $n$
 5       do for $j \leftarrow 0$ to $m$
 6              do $D[i][j] \leftarrow \infty$
 7  $D[0][0] \leftarrow 0$
 8  $\triangleright$ Fill in the cost matrix
 9  for $i \leftarrow 1$ to $n$
10       do for $j \leftarrow 1$ to $m$
11              do $cost \leftarrow \text{distance}(X[i], Y[j])$
12                 $D[i][j] \leftarrow cost + \min(D[i-1][j], D[i][j-1], D[i-1][j-1])$
13  return $D[n][m]$
```

Figure 4.17: Pseudocode for dynamic time warping between two vectors X and Y.

Two key preprocessing steps are performed prior to clustering the time series. The first is smoothing, as shown in the top right image of Figure 4.5. Smoothing is performed using 1D convolutions along the length of each time series, with a kernel size of $k = 10$. The kernel weights are set to $\frac{1}{k}$, and thus, for each neuron, the FRET score at each time step is the average of the scores within the fixed window. This eliminates noise caused by fluctuating experimental conditions over time, such as lighting and cellular drift.

The other preprocessing step is filtering. Section 3.1 discussed DeepFRET, a method of training a deep neural network for false FRET detection. Recall that using this method, valid and false FRET traces are simulated, and these simulated time series are used to train a classifier. The method employed by FRETLab is simpler. Given that the Jones lab have a biological prior on the feasible range of FRET ratios for each drug, we simply omit those traces which are outside of this range.

4.3 Evaluation

The FRETLab software was provided to the Jones lab for a period of two weeks prior to the conclusion of the project so as to enable users to test the software, provide feedback, and have critical requests addressed. The author installed the software as a standalone executable in person and provided a walkthrough on its use. The following is verbatim feedback from Dr Ben Jones, an expert cell biologist and the primary user of the software:

We have been analysing our FRET imaging experiments using ImageJ. For every experiment or series of experiments this involves constructing a macro to perform the required analyses, taking ∼15-20 minutes, and then applying this to other image stacks from the same series (∼5 minutes per stack). FRETLab, just using the default settings, was able to analyse an image stack and produce high quality data in a fraction of this time, increasing the time we have available to design and perform experiments, and interpret our results. I found the user interface very intuitive and easy to use. The cell segmentation was highly accurate – the default setting was almost always in agreement with what I would have selected. The outputs generated were very useful for including in presentations – obviously having the raw data means we can prepare our own figures where different appearance is needed (for a publication for example). The clustering facility was a great addition and I believe will generate interesting hypotheses once we have more data to analyse.

There are three ways FRETLab could be improved to make it more flexible for our workflows and for other researchers:

1. *Batch processing option – our live cell imaging experiments typically include several fields-of-view of the same sample, and being able to analyse these as a batch without needing to interact with FRETLab for each stack would be very helpful, with even larger benefits for high content imaging applications e.g. in multi-well plates.*
2. *Being able to re-train the detection AI for different cell types would be a huge advantage. I realise that implementing this would be a significant challenge in its own right.*
3. *More minor suggestions could include allowing the software to cope with different image sizes and making channel images on the camera chip user-selectable, as other users may run FRET analysis using different wavelengths.*

In its present form FRETLab will make a significant difference to our image analysis workflow, and if in the future some of the suggestions above are possible to implement, this will make it even more useful.

From the preceding user story, we can see that speed, accuracy, and usability are the key merits of the software. Concerning speed, as noted by Dr Jones, the existing pipeline takes approximately 15-20 minutes to process a single image stack for the first time. Then, once recorded, further analyses take 5 minutes. Comparably, FRETLab takes approximately 4 minutes for any acquisition, with the bottleneck in processing time being loading the 2GB image files and computing the vignette correction. Thus, FRETLab represents a four to five factor speedup over the existing pipeline. On this note it should also be observed that the time estimate of 15-20 minutes using the original pipeline is for an expert Fiji user such as Dr Jones with high familiarity with the image analysis pipeline. The author's replications of the pipline took $\sim$ 45 minutes on average.

Concerning accuracy of both the detections and segmentations, Section 4.2.2 demonstrated the high hit rate of the Faster R-CNN used to perform the cellular detections. Further, Section 4.2.3 established that SAM can be made to be competitive with and even outperform the current state-of-the-art in cell segmentation, CellPose. Since no segmentation masks were available for the in-house neuron dataset, the segmentation model was not trained on any of the images produced by the Jones lab. Therefore, the validation of an expert cell biologist that the masks produced by FRETLab on the in-house data are of high-quality indicates the generalisability of the proposed method.

Finally, regarding usability, FRETLab was developed with the intention of abstracting away the complexities of the machine learning pipeline underlying it. To this end, Dr Jones' feedback indicates this aim was met. With drag and click tools for interactive segmentation, detection thresholding, and clustering, the barrier to use is low.

Moving on to sources of improvement, Dr Jones' feedback highlights three sources of potential improvement for future work: generalisation to multiple cell types, support for batch computation of FRET data, and generalisation to other FRET modalities. Regarding the first of these, zero-shot generalisation to multiple novel cell types is an exceedingly difficult task, as there are no large scale datasets for either cell segmentation or detection that cover a sufficiently wide number of cell types and imaging modalities. No model has yet been developed that does this satisfactorily in the literature. The author experimented with training a generalised cell detector by repurposing the CellPose dataset as a detection dataset. However, the performance was found to be inadequate when tested on the in-house dataset. In both Chapter 6 and the discussion of future work in Chapter 7, we will touch on this matter further.

Concerning batch processing, this feature was implemented without a user interface for the author's personal experiments with time series clustering, as discussed in Chapter 5. However, time and resource constraints prevented its inclusion in the final project release. The next iteration of the software will provide support for this feature.

Finally, regarding generalising to different image formats, sizes, and FRET styles, there is no data available for this for the present project, as all neuron FRET data produced by the Jones lab has a uniform file type, size, and FRET modality. Thus, had these features been implemented, there would have no means by which to test or use them immediately. A balance was therefore struck between producing generalisable software and the demands of the remainder of the project. However, in future releases, support for these features will be provided.

Chapter 5

Time Series Analysis

This chapter presents the results of the exploratory data analysis conducted using the FRETLab pipeline developed in Chapter 4 . In particular, Section 5.1 details the process of using the FRETLab software to construct and analyse a drug reactivity dataset using imaging acquisitons from the ExD3 drug trials. This begins with an explanation of the overall time series pipeline and clustering metrics in Sections 5.1.1 and 5.1.2. Then, using classical dimensionality reduction methods, Section 5.1.3 investigates whether meaningful clusters of response profiles can be found in the data. Next, in Section 5.1.4, we explore deep clustering through temporal convolutional autoencoders. A summary of the clustering results and a taxonomy of the clusters is given in Section 5.1.5. Lastly, in Section 5.2, we conclude with experiments in reactivity prediction.

5.1 Time Series Clustering

The primary objective of the present chapter is to determine whether a quantitative determination can be made as to whether the neurons under study can be said to form meaningful clusters with respect to their response profiles. Evidence collected using the FRETLab tool is suggestive that such is the case. For instance, Figure 4.8 highlights what looks to be a tight grouping of response profiles for the ExD3 drug. Further, recalling back to Figure 2.2, the Jones lab have already established FRET ranges, grouping neurons into responders and super responders. However, these ranges are arbitrarily set and do not have any cluster evaluation metrics underlying them.

The following sections explore the results of classical and deep-learning based unsupervised time series clustering. Of the two drugs being tested, ExD3 and ExF1, the data for the ExD3 drug is the most amenable to analysis. As mentioned in Section 4.2.1, one issue with the in-house dataset is non-standard series length. In particular, for many of the ExF1 acquisitions, the experiments vary in length by up to 171 timesteps. Thus it was decided to focus exclusively on the data for the ExD3 drug. The ExD3 drug data is partitioned into eight experiments, each of which consists of four to five imaging acquisitions with a unique field of view, combining to make 37 independent trials in total. The full experimental set-up is explored next.

5.1.1 Experimental Details, Procedure, and Preprocessing

The first preprocessing step was the removal of errant FRET traces using a biological prior on expected ranges. For the purpose of the following experiments, the FRET traces were required to be within the interval [0, 2.5], otherwise they were excluded. The second preprocessing step was truncation. As noted in Section 4.2.1, one of the limitations of the in-house neuron dataset is that there are occasionally considerable differences in the introduction of the control drugs. This is a confounding variable, as it effects the drug response curves in a way that is independent of the inherent biochemistry of the cells being studied. Thus, all time series were truncated at time point 156.

Three sets of experiments were performed for classical unsupervised time series clustering, and two for deep unsupervised clustering. In the first experiment, the clustering method under investigation was applied at the individual FOV level. This was to examine the behaviour of neurons under the exact same experimental conditions. The second trial was to examine the trends at the experiment level, encompassing up to four of five FOVs, each taken at different times. Finally, all of the data from each experiment across all fields of view were amalgamated and clustered. The aim of this multilevel analysis was to determine whether there are any global properties of the time series that persist even across different experimental conditions. The experimental workflow along with summary statistics are given in Figure 5.1 and Table 5.1 respectively.

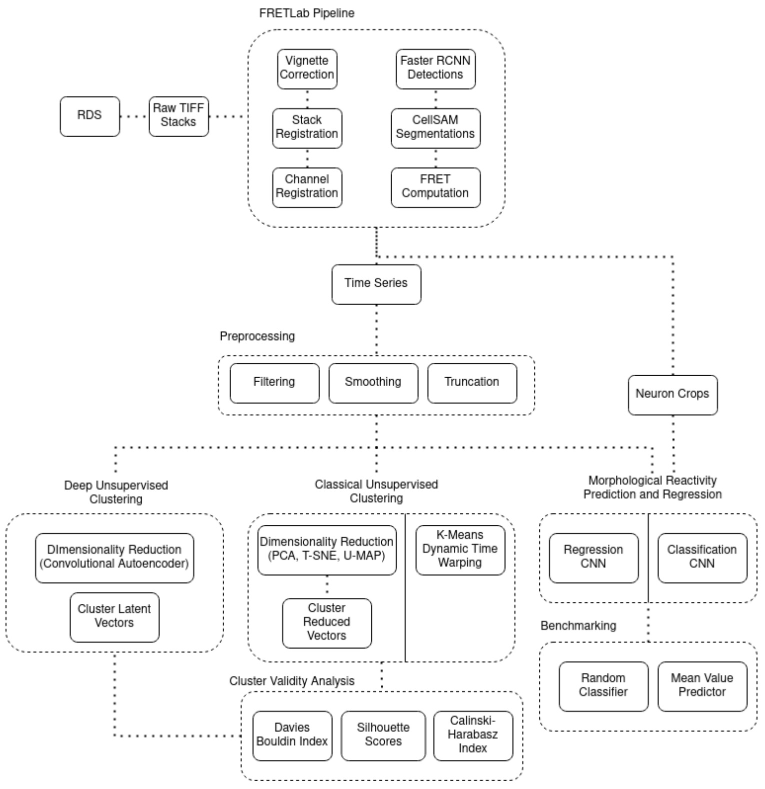


Figure 5.1: Workflow for time series analysis, classification and regression tasks.

Experiment	FOV	Cell Count	Mean	Median	Range	Stdv	Var
410A	Pos 0	67	1.37	1.36	0.33	0.0007	0.014
	Pos 1	58	1.36	1.35	0.33	0.0006	0.014
	Pos 2	44	1.42	1.42	0.37	0.0012	0.018
	Pos 3	41	1.40	1.40	0.30	0.0006	0.014
	Total	**210**	**1.38**	**1.38**	**0.37**	**0.0008**	**0.015**
410B	Pos 0	74	1.36	1.35	0.40	0.0014	0.023
	Pos 1	41	1.36	1.36	0.28	0.0007	0.017
	Pos 2	44	1.32	1.31	0.49	0.0029	0.032
	Pos 3	25	1.36	1.35	0.56	0.0020	0.024
	Total	**184**	**1.35**	**1.34**	**0.56**	**0.0017**	**0.024**
411A	Pos 0	58	1.58	1.57	0.48	0.0030	0.029
	Pos 1	49	1.54	1.56	0.51	0.0039	0.031
	Pos 2	61	1.64	1.65	0.40	0.0027	0.027
	Pos 3	48	1.60	1.60	0.46	0.0022	0.022
	Total	**216**	**1.59**	**1.60**	**0.51**	**0.0029**	**0.027**
414A	Pos 0	34	1.50	1.50	0.49	0.0028	0.029
	Pos 1	37	1.43	1.43	0.38	0.0008	0.016
	Pos 2	38	1.41	1.41	0.40	0.0017	0.025
	Pos 3	28	1.48	1.48	0.24	0.0002	0.012
	Pos 4	36	1.50	1.49	0.48	0.0024	0.022
	Total	**173**	**1.46**	**1.46**	**0.49**	**0.0016**	**0.021**
414B	Pos 0	40	1.57	1.56	0.40	0.0025	0.029
	Pos 1	35	1.56	1.55	0.37	0.0020	0.027
	Pos 2	20	1.48	1.48	0.53	0.0022	0.021
	Pos 3	29	1.43	1.43	0.28	0.0005	0.016
	Pos 4	22	1.38	1.38	0.24	0.0007	0.017
	Total	**146**	**1.50**	**1.49**	**0.53**	**0.0016**	**0.023**
415A	Pos 0	24	1.55	1.55	0.38	0.0025	0.033
	Pos 1	30	1.55	1.55	0.35	0.0023	0.032
	Pos 2	53	1.42	1.42	0.58	0.0022	0.026
	Pos 3	23	1.50	1.48	0.61	0.0066	0.053
	Pos 4	30	1.47	1.46	0.40	0.0014	0.023
	Total	**160**	**1.49**	**1.48**	**0.61**	**0.0028**	**0.032**
415B	Pos 0	27	1.54	1.54	0.24	0.0003	0.013
	Pos 1	37	1.50	1.50	0.29	0.0009	0.020
	Pos 2	35	1.47	1.46	0.57	0.0025	0.027
	Pos 3	39	1.46	1.46	0.43	0.0018	0.023
	Pos 4	25	1.56	1.56	0.28	0.0009	0.020
	Total	**163**	**1.50**	**1.49**	**0.57**	**0.0014**	**0.021**
ACTB	Pos 0	38	1.48	1.48	0.43	0.0013	0.017
	Pos 1	51	1.55	1.54	0.61	0.0028	0.025
	Pos 2	42	1.43	1.43	0.39	0.0022	0.027
	Pos 3	26	1.50	1.49	0.39	0.0022	0.023
	Pos 4	48	1.53	1.52	0.59	0.0037	0.031
	Total	**205**	**1.50**	**1.50**	**0.61**	**0.0025**	**0.025**
Summary		**1457**	**1.47**	**1.47**	**0.61**	**0.0019**	**0.023**

Table 5.1: Experimental details and FRET statistics for the eight ExD3 drug trials.

5.1.2 Inter-Cluster and Intra-Cluster Evaluation Metrics

In order to quantitatively evaluate the results of cluster analysis experiments, we will consult several metrics: silhouette score, Calinski-Harabasz index, and the Davies-Bouldin index. Taken in combination, these will allow us to gauge inter and intra-cluster validity.

Silhouette Score

The silhouette score [77] ranges from -1 to $+1$, where a high value indicates that a data point is well matched with its own cluster and poorly matched to neighbouring clusters. To understand the silhouette score, we must derive two terms. The first is $a(i)$, a measure of how well suited a given data point $i \in C_I$ is to its own cluster C_I.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_I, i \neq j} d(i,j) \qquad b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i,j) \tag{5.1}$$

That is, $a(i)$ is the average distance of data point i from every other data point in the same cluster. The next term is $b(i)$, the minimum mean distance between a given data point i and all data points belonging to a different cluster, over all clusters.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{5.2}$$

The final score, $s(i)$, is given by Equation 5.2. By the definition of $s(i)$, $-1 \leq s(i) \leq 1$.

Calinski-Harabasz Index

The Calinski-Harabasz index [78] is a score in the interval $[0, \infty)$ which measures the ratio of the between-cluster variance to the within-cluster variance. Defined per Equation 5.3, a higher Calinski-Harabasz value indicates better separation between clusters.

$$CH = \frac{B}{W} \times \frac{N - k}{k - 1} \tag{5.3}$$

Where B is the between cluster variance, W is the within cluster variance, N is the total number of data points, and k is the number of clusters.

Davies-Bouldin Index

The Davies-Bouldin index [79] is a score in the interval $[0, \infty)$ which measures the average similarity between each cluster and its most similar cluster. The lower the Davies-Bouldin index, the better the clustering quality. The formula for calculating the Davies-Bouldin index for a set of clusters is as follows:

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{j \neq i} \left(\frac{S_i + S_j}{M_{ij}} \right) \tag{5.4}$$

Where N is the number of clusters, S_i and S_j are the average distances from the samples in clusters i and j to the centroids of clusters i and j respectively, and M_{ij} is the distance between the centroids of clusters i and j.

5.1.3 Classical Unsupervised Time Series Clustering

The classical clustering results are suggestive of $k = 2$ clusters, which aligns well with the intuition provided by the clustering facility of the FRETLab tool. While the Calinski-Harabasz score is sporadic, the other metrics are consistent in this regard. As shown by the silhouette plot of Figure 5.2 and Table 5.2, this is borne out clearly when clustering all 1457 data points as a whole. While there is more variability in the FRET curves at the indivdual FOV and experiment level per Table 5.1, $k = 2$ clusters continues to dominate when the leading cluster is tallied across the three metrics for each of the 37 trials. Silhouette plots for the eight experiments are given in Appendix B.

Three features are relevant to the interpretation of a silhouette plot: the average silhouette score, the relative areas of the clusters, and whether all clusters exceed the average silhouette score line. Concerning the average silhouette score, an industry standard threshold for a good clustering value is a silhouette score ≥ 0.5 [77]. Further, we take an interest in the relative areas of the clusters, as this indicates how many data points are assigned to each. Finally, it is desirable that the average silhouette score for each cluster be above the mean trendline, given in red. One or more clusters below the the trend line indicates that the average score is being raised by the other clusters.

Method	Silhouette	Davies-Bouldin	Calinski-Harabasz	Leader
K-means$_{DTW}$	**0.53**, 0.46, 0.45, 0.44	**0.64**, 0.69, 0.73, 0.75	2297, 2353, 2379, **2416**	$k = 2$
K-medioids$_{DTW}$	**0.53**, 0.46, 0.45, 0.43	**0.64**, 0.70, 0.74, 0.76	2296, 2347, **2357**, 2343	$k = 2$
Kernel KM$_{DTW}$	**0.53**, 0.47, 0.44, 0.44	**0.65**, 0.71, 0.74, 0.78	2300, 2342, **2371**, 2340	$k = 2$
PCA: K-means	**0.52**, 0.47, 0.44, 0.44	**0.64**, 0.70, 0.71, 0.74	2327, 2406, 2451, **2514**	$k = 2$
PCA: Spectral	**0.52**, 0.43, 0.28, 0.30	0.63, 0.87, 0.50, **0.47**	**2228**, 1057, 335, 336	$k = 2$
Agglomerative	**0.53**, 0.43, 0.43, 0.41	**0.65**, 0.78, 0.72, 0.73	**2273**, 1916, 2231, 2052	$k = 2$

Table 5.2: Silhouette, DB, and CH scores for clusters k $\in \{2, 3, 4, 5\}$, in left to right order. The leader column indicates the value of k with the majority metric leads.

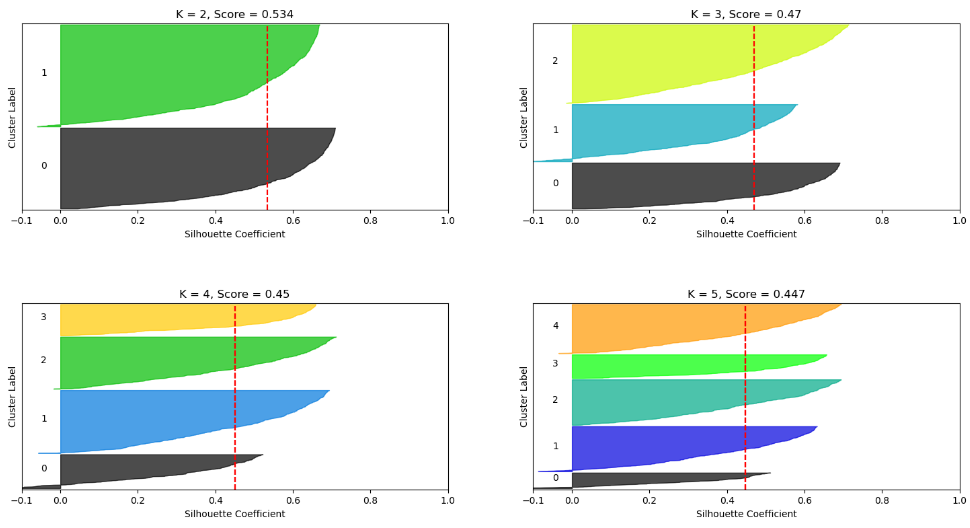


Figure 5.2: Silhouette plots for $k \in \{2, 3, 4, 5\}$ on the aggregated data using K-means.

5.1.4 Deep Unsupervised Time Series Clustering

Deep unsupervised clustering was carried out by performing dimensionality reduction using a temporal convolutional autoencoder (TCN-AE) as opposed to classical techniques such as PCA. A temporal convolutional autoencoder is a convolutional autoencoder with 1D convolutional layers and has been shown to demonstrate robustness to noise [80]. The motivation here was to determine whether non-linear dimensionality reduction would change the optimal number of clusters per the metrics used in the previous subsection.

The procedure was to first perform dimensionality reduction on the aggregated time series data by passing it through the TCN-AE. The resulting latent vectors were then clustered using the same clustering algorithms as in the preceding section. The key hyperparameter here is the size of the latent dimension in the autoencoder, which we may think of as being equivalent to the number of components used when performing PCA. The results of deep unsupervised clustering are volatile for low dimensional latent spaces, but remain consistent as the size of the latent space is scaled beyond five variables. The author's conjecture is that beyond this number of latent variables, most of the variability between each time series has already been captured, and there is little additional information captured in the added dimensions. Again, as shown by Table 5.1.4, the indication is that $k = 2$ is the optimal number of clusters.

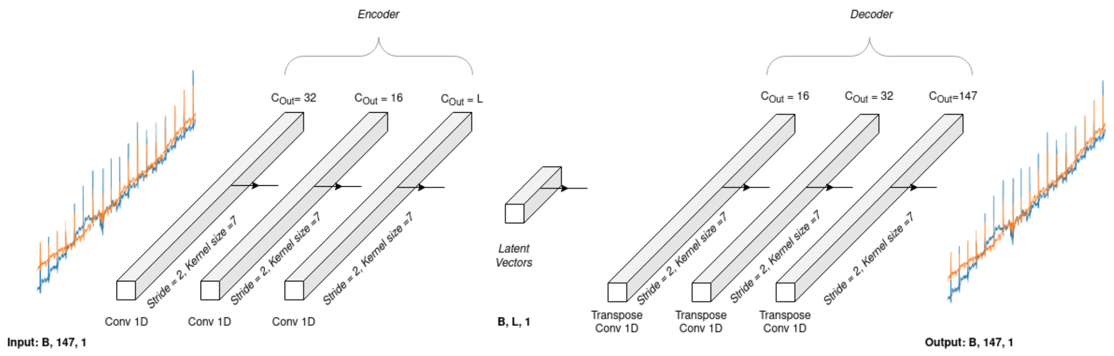


Figure 5.3: Architecture diagram for the temporal convolutional autoencoder. After truncation to time point 156 and performing running average smoothing with a 1D convolutional kernel of length 10, the input time series curves are of length 147. ReLU activations were used in between convolutional and transpose convolutional layers. Here B is the batch size, C_{out} represents the number of output channels for a particular convolutional layer, and L is the dimensionality of the latent space.

Method	Silhouette	Davies-Bouldin	Calinski-Harabasz	Leader
K-means	**0.52**, 0.40, 0.41, 0.36	**0.51**, 0.79, 0.74, 0.82	1694, 1688, 1860, **1866**	$k = 2$
Spectral	**0.46**, 0.40, 0.41, 0.20	**0.50**, 0.56, 0.55, 1.01	**1671**, 765, 921, 416	$k = 2$
Agglomerative	**0.53**, 0.43, 0.43, 0.41	**0.67**, 0.78, 0.72, 0.73	**2273**, 1916, 2231, 2052	$k = 2$

Table 5.3: Evaluation results for deep unsupervised clustering, with clustering performed on the latent vectors produced by a TCN-AE. Formatted as described in Table 5.2.

5.1.5 Cluster Taxonomy

Having arrived at a suspected number of clusters at $k = 2$, we now develop a taxonomy of the clusters and their characteristics. We do this by exporting the cluster data using the FRETLab tool and then by computing statistics for each cluster.

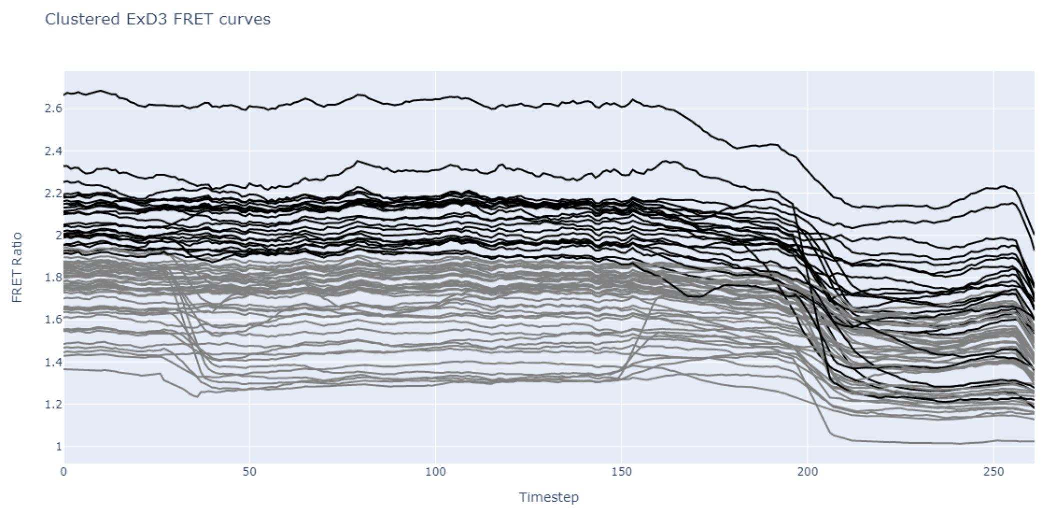


Figure 5.4: Visualising the cluster taxonomy using a single field of view sample.

	Cluster 1				Cluster 2			
Experiment	Mean	Median	Range	Stdv	Mean	Median	Range	Stdv
410A	1.63	1.62	0.34	0.0021	1.55	1.54	0.32	0.0023
410B	1.66	1.65	0.52	0.0019	1.54	1.54	0.39	0.0017
411A	1.60	1.59	0.49	0.0017	1.53	1.52	0.37	0.0015
414A	1.60	1.61	0.47	0.0018	1.53	1.51	0.35	0.0021
414B	1.64	1.64	0.49	0.0018	1.56	1.55	0.31	0.0015
415A	1.62	1.63	0.57	0.0029	1.51	1.51	0.32	0.0024
415B	1.65	1.64	0.54	0.0016	1.55	1.51	0.32	0.0016
ACTB	1.62	1.62	0.51	0.0025	1.53	1.51	0.38	0.0012

Table 5.4: Statistics for each cluster using K-Means with dynamic time warping.

As can be seen from Figure 5.4, the clusters separate out relatively neatly when visualised. In particular, we observe two key features distinguishing the clusters. The first is their height, with the first cluster starting with an elevated FRET response. The second is range, with the second cluster having a narrower range of FRET values. These insights are supported by the data in Table 5.4, which shows that the average range of FRET is narrower for cluster two, as well as the mean and median FRET response being lower. In conclusion, the preceding analysis suggests that there may indeed be two classes of neurons, those which have an elevated initial response and a steeper decline, and those with a narrower reactivity range. This may suggest a point of biological difference between the neurons that could inform future research.

5.2 Morphological Reactivity Prediction and Regression

Once it had been established through the preceding experiments in classical and deep unsupervised clustering that there may indeed be biologically meaningful response profiles, the question arose as to whether these profiles could be predicted from image crops of the neurons alone. This task, morphological reactivity prediction, would provide meaningful biological insights into the neurobiology of metabolism and aid biologists in directing their experiments. Similarly, morphological reactivity regression, the task of predicting numerical features such as minimum FRET response, maximum FRET response, and approximate entropy, would help direct biochemical investigations into what distinguishes the neurons under study. While the prediction results were ultimately on par with chance, and the regression results were within close range of a mean-predictor baseline, it is nevertheless instructive to examine the prediction and regression pipelines so as to guide future work.

Both methods utilised a convolutional neural network. For the prediction task, a crop of a neuron was fed into the network, processed by several convolutional layers, and finally fed through an MLP with $k = 2$ outputs, per the preceding analysis of the number of clusters in the data. Cross entropy was used as the loss function. The regression task utilised a similar architecture, but with the mean squared error loss function.

Based on similar research in image-based profiling [81], the literature suggests that ultra high-resolution is a requirment for achieving statistically significant results, and this may be aided further by the addition of other biochemical data sources such as genomic data in order to create multimodal inputs. Per Figure 5.5, high-resolution data was not available for the present project, as the imaging acquisitions were intended for high-volume drug screening, and therefore compromise resolution for a wide field of view.

As will be discussed in Chapter 7 in the discussion of future work, here would be an opportunity to acquire a large scale dataset of single cell resolution images and apply the same pipeline once more to determine conclusively whether such a task is possible for the neurons and drugs under study.

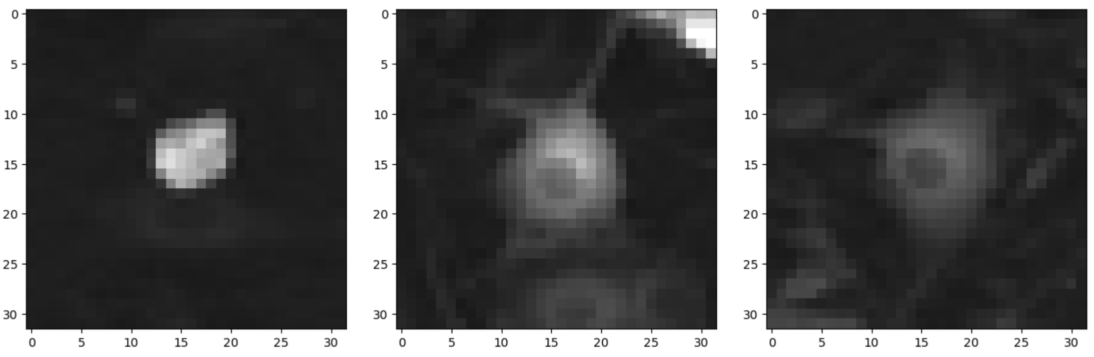


Figure 5.5: Examples of the neuronal crop images used for both the cluster membership prediction and regression tasks. Observe the low resolution of the neuron crops.

Chapter 6

Few-Shot Dataset Labelling

This chapter introduces the Segment Anything Model With Selective Extraction (SAM-WISE), an experimental first step towards a generalisable pipeline for few-shot segmentation and bounding box labelling of cell-biology datasets. We begin with the motivation for the problem in Section 6.1, describe the method and its preliminaries in Section 6.2, and finally conclude with an evaluation in Section 6.3.

6.1 Motivation

The evaluation of CellSAM in Section 4.2.3 demonstrated the ability of the Segment Anything Model to produce faithful and high-quality masks that are competitive with current state-of-the-art methods for cell segmentation. Further, the FRETLab software demonstrates the practicality of a fine-tuned SAM paired with a cellular object detector. The bottleneck here, however, is that thousands of hand-annotated labels were required to produce the dataset to train the object detector to accurately localise the cells. This motivated an exploration into the use of SAM for few-shot dataset labelling.

The possiblity of using SAM to automatically label datasets is suggested by the fact that the model already has inherent object detection capabilities, and in many cases is capable of segmenting all objects in an image. The authors of the Segment Anything [7] paper suggest this potential for the model, noting that in constructing SAM they iterated between using their 'efficient model to assist in data collection and using the newly collected data to improve the model'. Their approach was to construct what they termed a 'data engine'. The data engine had three stages: assisted-manual, semi-automatic, and fully automatic. In the first stage, SAM assisted annotators in annotating masks, similar to the classical interactive segmentation setup. In the second stage, SAM was used to automatically generate masks for a subset of objects by prompting it with likely object locations and the annotators then focused on annotating the remaining objects. In the final stage, the authors prompted SAM with a regular grid of points, per Figure 3.4, and had it generate masks automatically. The task of cell segmentation has several properties which make it ideal for this type of method, namely low ambiguity and high object homogeneity. We turn to an experimental method for this next.

6.2 Segment Anything Model With Selective Extraction

The Segment Anything Model with Selective Extraction (SAMWISE) is predicated on a simple idea. When generating masks in everything mode, SAM often correctly segments out cells perfectly but includes erroneous masks in its final output. The idea is to extend this idea, deliberately increasing the sample density of the prompt grid and aggressively oversegmenting the image. This produces a mask set which contains high quality masks for all of the cells in the image, in addition to many low quality errant masks for debris or the image background. The task then becomes one of filtering out only false detections.

6.2.1 Pipeline

The pseudocode for SAMWISE is given in Figure 6.1. The first step is to have the user label a small set of exemplar cells using SAM and then perform feature extraction on the area under the segmentation masks. These are then used to fit a novelty detection classifier. In the second stage, the dataset is deliberately oversegmented by SAM by prompting it with a dense grid of foreground points. Finally, a second pass is made over the automatically segmented dataset, and the areas under all masks are filtered or retained on the basis of their features. Abstracted away is the matter of how feature extraction and novelty detection are performed. This is because the method is agnostic about these implementation choices. Section 6.2.2 will explore several possible methods.

```
SAMWISE($D, M_k$)
1  ▷ Input: Dataset $D$ and user-provided masks $M_k$
2  ▷ Output: Filtered bounding boxes $B_{\text{filtered}}$
3  Perform feature extraction on areas under $M_k$ to obtain feature matrix $F_k$
4  Fit a novelty detection model on $F_k$ ▷ e.g., One-Class SVM
5  Automatically label the dataset $D$ using the model
6  Initialize an empty set $B_{\text{filtered}}$
7  for image $I$ in $D$
8      do
9          for mask $M_i$ in $I$
10             do
11                 Extract features from $M_i \odot I$ to obtain feature vector $F_i$
12                 if Novelty Detection Model predicts $F_i$ as non-anomalous
13                     then Add BOUNDING-BOX($M_i$) to $B_{\text{filtered}}$
14 return NON-MAXIMAL-SUPPRESSION($B_{\text{filtered}}$)
```

Figure 6.1: High-level pseudocode for the SAMWISE pipeline.

The dataset used to evaluate the method is the Blood Cell Count and Detection dataset (BCCD) produced by MIT [82]. Consisting of 364 images across three classes: white blood cells, red blood cells, and platelets, the dataset contains 4888 labels total. The BCCD dataset has several properties which make it suitable for evaluating the method, namely that the classes are clearly visibly distinct, the task of distinguishing each class is of clear practical utility, and the dataset is a commonly used benchmark for biomedical imaging. Examples images are given in Figure 6.2.

6.2.2 Novelty Detection Methods

Several classifiers were trialled to test the SAMWISE method, namely support vector machines, local outlier factor, and random forests. These are briefly summarised below.

One-Class Support Vector Machine

The assumption of a one-class support vector machine (SVM) [83] is that the majority of the training data is normal, and the goal is to create a model that captures the characteristics of this majority class. The training data consists of feature vectors representing instances from the normal class. The one-class SVM seeks to find a hypersphere that best encapsulates the training data. This hypersphere should have a maximum margin from the closest data points of the normal class. The decision boundary of the SVM is determined by finding the hypersphere that minimizes the following objective function:

$$\min_{\mathbf{w},\xi,\rho} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho \right) \quad \textit{s.t:}\;\; \mathbf{w}\cdot\Phi(\mathbf{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{6.1}$$

Here, $\mathbf{w}$ represents the weight vector, ξ_i are slack variables, ρ is the radius of the hypersphere, ν is the user-defined parameter controlling the trade-off, $\Phi(\mathbf{x}_i)$ represents the feature mapping of input data $\mathbf{x}_i$, and n is the number of training examples.

Random Forest

A random forest consists of an ensemble of fixed-depth decision tree classifiers, each trained on a subset of both the training examples and the feature set. When a new data point is presented for prediction, each decision tree in the random forest independently classifies or scores the data point. In the context of outlier detection, the output from each tree can be considered as a measure of how outlying the data point is [84]. The predictions from individual trees are then aggregated to produce a single outlier score for the data point. As discussed in Section 3.2, this is the method currently employed for trainable segmentation in popular bioimaging libraries.

Local Outlier Factor

The Local Outlier Factor (LOF) [85] aims to identify outliers or anomalies in a dataset by measuring the local deviation of a data point with respect to its neighbours, making it particularly effective for detecting anomalies in complex, non-uniformly distributed datasets. The LOF of a point A measures how much the local density of A differs from the local densities of its neighbours. It is calculated as the ratio of the average local reachability density of A's neighbours to the local reachability density of A itself. A high LOF value for a point indicates that it has a significantly lower density than its neighbours, marking it as a likely outlier or anomaly.

6.3 Evaluation

The results of the SAMWISE experiments are summarised below in Table 6.3. To develop and test the method, a simple utility was developed based on the techniques employed by FRETLab. This enabled the author to segment exemplars for each class with the SAM ViT-H checkpoint. Exemplars were kept consistent across experiments.

Classifier	Features	$P_{0.5}$		$P_{0.75}$	
		RBC	WBC	RBC	WBC
One-class SVM	RGB Histogram	0.60	0.59	0.57	0.54
Random Forest	Multi-scale local	**0.67**	**0.65**	**0.63**	**0.62**
Local Outlier Factor	RGB Histogram	0.62	0.58	0.57	0.55

Classifier	Features	$R_{0.5}$		$R_{0.75}$	
		RBC	WBC	RBC	WBC
One-class SVM	RGB Histogram	0.66	0.65	0.62	0.60
Random Forest	Multi-scale local	**0.84**	**0.87**	**0.79**	**0.76**
Local Outlier Factor	RGB Histogram	0.64	0.62	0.60	0.59

Classifier	Features	$Acc_{0.5}$		$Acc_{0.75}$	
		RBC	WBC	RBC	WBC
One-class SVM	Histogram	0.60	0.51	0.55	0.49
Random Forest	Multi-scale local	**0.70**	**0.77**	**0.67**	**0.71**
Local Outlier Factor	Histogram	0.62	0.63	0.56	0.59

Table 6.1: Per class precision, recall, and accuracy metrics for various outlier detection methods. The titles RBC and WBC indicate white and red blood cells respectively.

As can be seen from Table 6.3, the random forest method trained with multiscale local features was the most successful across all categories. This may be attributed to the fact that this method uses pixel-level features to train an ensemble of decision trees which subsample both the features and the exemplar pixels. A majority class vote within the boundaries of each mask produced by SAM are then taken. This provides robustness, as each pixel is treated independently, as opposed to evaluating the features of the mask as a whole against the positive exemplars, as is done by the remaining methods.

A noticeable trend among all classifiers is a shared bias with respect to the precision-recall tradeoff. Every method displays higher recall than precision, indicating that while the vast majority of cells are detected for each class, a significant number of false positives also occur. We may attribute this to the oversegmentations produced by SAM. This effect is visualised in the bottom row of Figure 6.2, which shows the detections produced by the random forest filtering method for the white blood cell class using a single exemplar.

While more work remains to be done to refine the method, the preceding evaluation suggests that human-in-the-loop few-shot prompting could potentially enable dataset labelling in low-resource settings. As discussed in Section 7, an extension to this method would be to use deep learning to extract features from the masks. Experiments with convolutional autoencoders were conducted in this vein, but none yielded results.

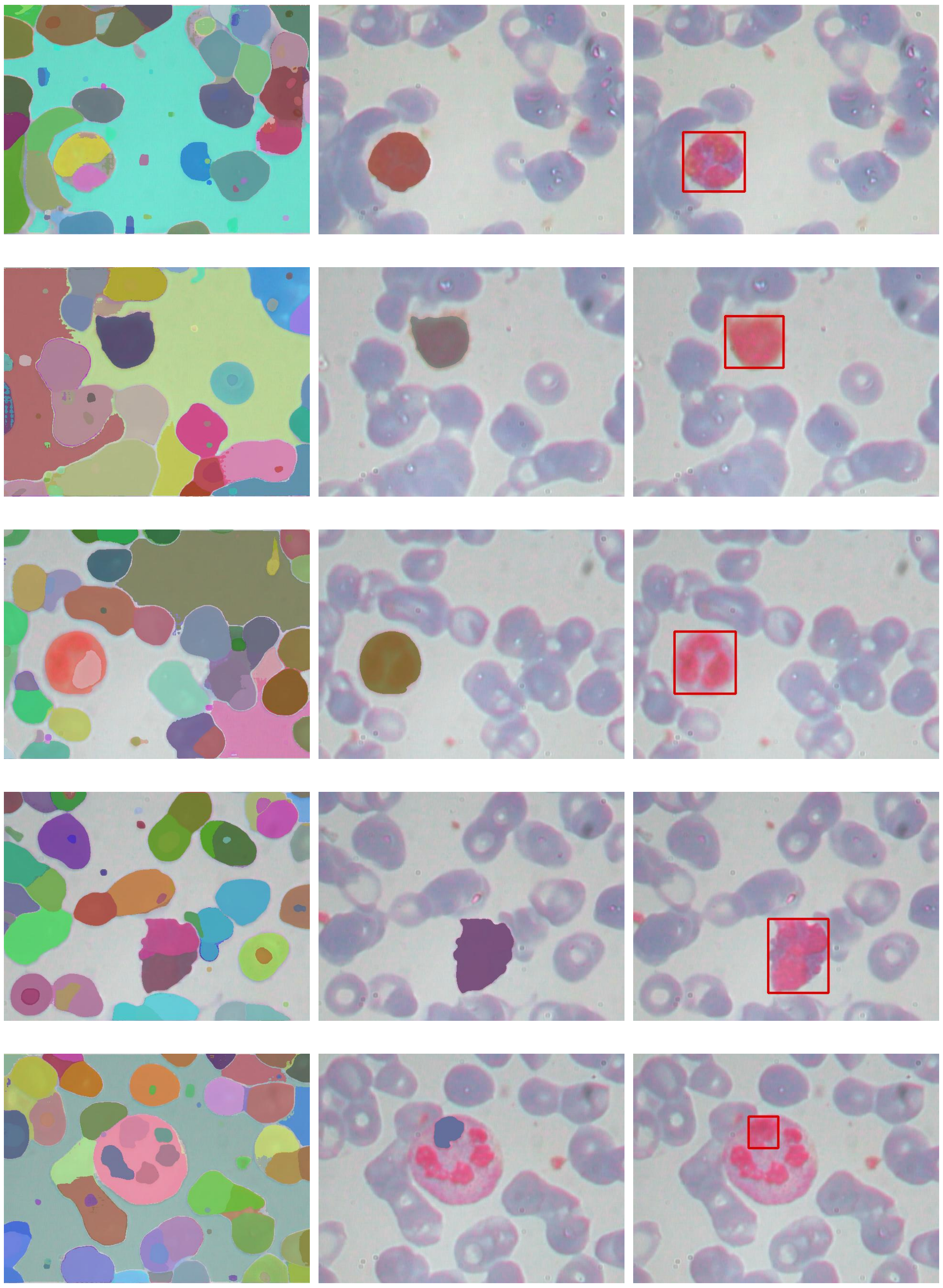


Figure 6.2: Demonstrating the performance of SAMWISE after using 1-shot exemplar prompting on the white blood cell class (stained pink) with a random forest filterer. The first column is the original segmentation produced by SAM. The second column shows the same segmentation masks post filtering and non-maximal suppression. The final column shows the bounding box detections. Per the last row, a current failure mode is underdetection of part of a cell's anatomy, as opposed to the whole cell.

Chapter 7

Conclusions and Future Work

This final chapter presents the author's concluding remarks and evaluation of the project as a whole. Alongside this is an examination of future work that could stand to compliment the project and extend the methods developed. Here we focus on each of the main three project components: the FRETLab software, time series analysis, and few-shot dataset labelling. Legal, ethical, and social considerations are given in Section 7.1.

The primary objective of the work outlined over the preceding six chapters was to explore the foundational Segment Anything Model (SAM) for cellular segmentation, and to develop an application which facilitates this task in the context of drug discovery. To this end, the FRETLab application was developed. The development of the application spanned the full machine-learning pipeline: preprocessing an in-house dataset of raw images, labelling thousands of cells by hand to create a detection dataset, training and fine-tuning existing models, developing a new method, benchmarking performance against a widely available dataset, building a user-friendly front end, and finally collaborating with an end-user to create a usable product. The user story of Section 4.3 confirms the tool's usefulness in accelerating the image analysis process for drug discovery. Further, quantitative analyses show that CellSAM, the specialised version of SAM the present project developed for cellular segmentation, is competitive with the current state-of-the-art model in general cell segmentation, CellPose.

In future work, the FRETLab software could be extended by generalising to multiple cell types and other variants of FRET analysis. Generalising the tool would enable other labs with different experimental procedures to make use of the software. One manner in which this could be achieved is by exploring the possibility of generating a large scale dataset for general cell segmentation. A plausible avenue of research here could be simulating a broad range of cell morphologies using generative methods such as generative adversarial networks [86] or diffusion [87]. Alternatively, since these methods themselves require large volumes of data, classical simulations could be performed and then techniques such as neural style transfer [88] could be used to imbue the cells with realistic features. The merit of such an approach is that simulated cells would be accompanied by ground truth segmentations and detections, which could then be used to train a highly generalised model which extends the CellSAM pipeline discussed in Section 4.2.3.

A further avenue of exploration would be to expand upon CellSAM. While experiments with CellSAM proved to be very promising, due to constraints on both time and compute resources, only ViT-B and not the larger variants of the SAM backbone were able to be used. ViT-B, the smallest version of the SAM image encoders, with 91 million parameters, was found by the authors of the original SAM paper to be signficantly worse than the ViT-L (308M) and ViT-H (636M) models. Future work could explore either securing greater compute resources and further developing CellSAM with a more powerful image encoder, or distilling SAM and enabling it to be competitive with these larger image encoders. Such work has been explored by [89] and would be a promising line of research.

The second project objective was to determine whether meaningful groupings arise in the response profiles of the neurons being tested. Indeed, the time series clustering results of Chapter 5 strongly suggest that neuronal responses can be meaningfully grouped and categorized into two distinct clusters. Following this, the project pushed further, and through the use of convolutional neural networks, experiments were conducted with a view to being able to predict this cluster membership from images of individual neurons alone. While the predictive accuracies of the models tested were on par with chance, the question as to whether cellular morphology alone is an accurate predictor remains. In future work, images taken at single cell resolution could be used to construct a new dataset, one which would consist of cell-time series pairs. Armed with higher resolution data, rich biochemical information could be gleaned from the images, allowing a convolutional neural network to learn features associated with drug response. Were such an approach to prove successful, visualisation techniques such as guided backpropagation [90] could be used to determine which cellular features in particular are being used to determine which cluster a given cell belongs to. This would provide actionable biological insights that biologists could use to inform future experiments.

Finally, the experiments in few-shot dataset labelling arose due to the author's experience hand-annotating a cellular detection dataset over thousands of cells. However, given that the Segment Anything Model has been trained on a dataset of 1 billion masks – the largest publicly available dataset yet published – the model already has impressive powers of generalised object detection. This led to the development of SAMWISE, an exploratory and general method for few-shot automatic dataset labelling. Due to time and resource constraints, the method was only able to be developed and tested on a small scale dataset. In future work, SAMWISE could be extended with deep-learning based filtering techniques and validated against a larger scale dataset such as LIVECell.

7.1 Legal, Ethical, and Social Considerations

The neuronal imaging data produced by the Jones lab and used throughout this project is sourced from mouse dissections. All such dissections are performed in a humane way and with the sole aim of advancing biomedical discovery, namely for the development of diabetes alleviating drugs. Further, the Jones lab operate with a license. All data used was pre-existing, and no dissections were commissioned for the present project.

Bibliography

[1] World Health Organization. *Ethics and governance of artificial intelligence for health.* World Health Organization, 2021.

[2] Debleena Paul et al. “Artificial intelligence in drug discovery and development”. In: *Drug discovery today* (2021).

[3] Chamier et al. “Democratising deep learning for microscopy with ZeroCostDL4Mic”. In: *Nature communications* (2021).

[4] Daniel R. Stroik et al. “Targeting protein-protein interactions for therapeutic discovery via FRET-based high-throughput screening in living cells”. In: *Scientific reports* (2018).

[5] *Dr Ben Jones: Research Projects.* URL: `https://www.imperial.ac.uk/people/ben.jones/research.html`.

[6] Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. “Pervasive Label Errors in Machine Learning Datasets Destabilize Benchmarks”. In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021)* (2021).

[7] Alexander Kirillov et al. “Segment Anything”. In: *arXiv:2304.02643* (2023).

[8] *What are proteins and what do they do?* URL: `https://medlineplus.gov/genetics/understanding/howgeneswork/protein/`.

[9] *Physiology, Cellular Receptor.* URL: `https://www.ncbi.nlm.nih.gov/books/NBK554403/#:~:text=These%20receptors%20are%20also%20known%20as%20transmembrane%20receptors.,large%20to%20make%20it%20through.`.

[10] *Peptides.* URL: `https://www.nature.com/scitable/definition/peptide-317/#:~:text=A%20peptide%20is%20a%20short,and%20protein%20can%20be%20arbitrary.`.

[11] *Biochemistry, lipids.* URL: `https://www.ncbi.nlm.nih.gov/books/NBK525952/#:~:text=Lipids%20are%20fatty%2C%20waxy%2C%20or,Waxes`.

[12] *GPCR.* URL: `https://www.nature.com/scitable/topicpage/gpcr-14047471/#:~:text=G%2Dprotein%2Dcoupled%20receptors%20(,lipids%2C%20sugars%2C%20and%20proteins.`.

[13] Ben Jones et al. “Targeting GLP-1 receptor trafficking to improve agonist efficacy”. In: *Nature communications* (2018).

[14] *Agonist.* URL: `https://www.cancer.gov/publications/dictionaries/cancer-terms/def/agonist`.

[15] José Francisco Kerr Saraiva and Andrei C Sposito. “Cardiovascular effects of Glucagon-like peptide 1 (GLP-1) receptor agonists”. In: *Cardiovascular Diabetology* (2014).

[16] *GLP-1R and Diabetes.* URL: `https://bpsbioscience.com/glp-1r-diabetes`.

[17] Michel Garcia Maciel et al. “The effect of glucagon-like peptide 1 and glucagon-like peptide 1 receptor agonists on energy expenditure: A systematic review and meta-analysis”. In: *Diabetes Research and Clinical Practice* (2018).

[18] Meera Shah and Adrian Vella. “Effects of GLP-1 on appetite and weight”. In: *Reviews in Endocrine and Metabolic Disorders* (2015).

[19] *What is adipose tissue?* URL: `https://my.clevelandclinic.org/health/body/24052-adipose-tissue-body-fat#:~:text=Adipose%20tissue%2C%20otherwise%20known%20as,(bone%20marrow%20adipose%20tissue).`.

[20] Ismael Gonzalez-Garcia et al. “Glucagon, GLP-1 and Thermogenesis”. In: *International Journal of Molecular Sciences* (2019).

[21] Haitham Abdulla et al. “Effects of GLP-1 Infusion Upon Whole-body Glucose Uptake and Skeletal Muscle Perfusion During Fed-state in Older Men”. In: *The Journal of Clinical Endocrinology and Metabolism* (2023).

[22] Shayan Fakhraei Lahiji et al. “Exendin-4–encapsulated dissolving microneedle arrays for efficient treatment of type 2 diabetes”. In: *Scientific reports* (2018).

[23] *Exendin-4: From lizard to laboratory...and beyond.* URL: `https://www.nia.nih.gov/news/exendin-4-lizard-laboratory-and-beyond`.

[24] Xinyi Wang and Yunyan Lai. “Three basic types of fluorescence microscopy and recent improvement”. In: *E3S Web of Conferences* (2021).

[25] Michael J. Sanderson et al. “Fluorescence microscopy”. In: *Cold spring harbor protocols* (2014).

[26] *Fiji.* URL: `https://imagej.net/software/fiji/`.

[27] Tingying Peng et al. "A BaSiC tool for background and shading correction of optical microscopy images". In: *nature communications* (2017).

[28] David G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *International Conference on Computer Vision* (1999).

[29] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *Computing Research Repository (CoRR)* (2015).

[30] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In: *IEEE* (1998).

[31] *ImageNet.* URL: `https://www.image-net.org/`.

[32] *COCO.* URL: `https://cocodataset.org/#home`.

[33] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Pattern Analysis and Machine Intelligence* (2015).

[34] Shrey Srivastava et al. "Comparative analysis of deep learning image detection algorithms". In: *Journal of Big Data* (2021).

[35] Joseph Redmon, Santosh Divvala, and Ross Girshick. "You Only Look Once: Unified, Real-Time Object Detection". In: *CVPR* (2016).

[36] Ashish Vaswani et al. "Attention Is All You Need". In: *NeurIPS* (2017).

[37] Alexander Kolesnikov Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR* (2020).

[38] Jerome N Feige et al. "PixFRET, an ImageJPlug-in for FRET Calculation That Can Accommodate Variations in Spectral Bleed-throughs". In: *Microscopy Research and Technique* (2005).

[39] S Bolte and F P Cordelières. "A guided tour into subcellular colocalization analysis in light microscopy". In: *Journal of Microscopy* (2006).

[40] Muriel Haas et al. "FRET and colocalization analyzer – A method to validate measurements of sensitized emission FRET acquired by confocal microscopy and available as an ImageJ Plug-in". In: *Microscopy Research and Technique* (2006).

[41] János Roszik et al. "Evaluation of intensity-based ratiometric FRET in image cytometry-Approaches and a software solution". In: *Cytometry Part A* (2009).

[42] István Rebenku et al. "Pixel-by-pixel autofluorescence corrected FRET in fluorescence microscopy improves accuracy for samples with spatially varied autofluorescence to signal ratio". In: *Scientific Reports* (2023).

[43] János Roszik, János Szöllősi, and György Vereb. "AccPbFRET: An ImageJ plugin for semi-automatic, fully corrected analysis of acceptor photobleaching FRET images". In: *Scientific Reports* (2008).

[44] Jiho Kim et al. "FLIM-FRET analyzer: open source software for automation of lifetime-based FRET analysis". In: *Source Code for Biology and Medicine* (2017).

[45] Johannes Thomsen et al. "DeepFRET, a software for rapid and automated single-molecule FRET data classification using deep learning". In: *eLife* (2020).

[46] Anne E Carpenter et al. "CellProfiler: image analysis software for identifying and quantifying cell phenotypes". In: *Genome Biology* (2006).

[47] Stuart Berg et al. "Ilastik: interactive machine learning for (bio)image analysis". In: *Nature methods* (2019).

[48] Ignacio Arganda-Carreras et al. "Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification". In: *Bioinformatics* (2017).

[49] Thorsten Falk et al. "U-Net: deep learning for cell counting, detection, and morphometry". In: *Nature methods* (2018).

[50] Kaiming He et al. "Mask R-CNN". In: *IEEE* (2017).

[51] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *IEEE CVPR* (2016).

[52] Carsen Stringer, Michalis Michaelos Tim Wang, and Marius Pachitariu. "Cellpose: a generalist algorithm for cellular segmentation". In: *Nature methods* (2020).

[53] Carsen Stringer Kevin J. Cutler et al. "Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation". In: *Nature methods* (2022).

[54] Uwe Schmidt et al. "Cell Detection with Star-convex Polygons". In: *MICCAI* (2018).

[55] *Segment Anything.* URL: `https://segment-anything.com/demo`.

[56] *Kaggle 2018 Data Science Bowl.* 2018. URL: `https://bbbc.broadinstitute.org/BBBC038`.

[57] *Tkinter.* URL: `https://docs.python.org/3/library/tkinter.html`.

[58] *CustomTkinter.* URL: `https://customtkinter.tomschimansky.com/`.

[59] *Pystackreg.* 2023. URL: `https://pystackreg.readthedocs.io/en/latest/`.

[60] *Pybasic.* 2023. URL: `https://github.com/linum-uqam/PyBaSiC`.

[61] *Neuroglial cells.* 2001. URL: `https://www.ncbi.nlm.nih.gov/books/NBK10869/`.

[62] Mischa Schwendy, Ronald E Unger, and Sapun H Parekh. "EVICAN—a balanced dataset for algorithm development in cell and nucleus segmentation". In: *Bioinformatics* (2020).

[63] Christoffer Edlund et al. "LIVECell—A large-scale dataset for label-free live cell segmentation". In: *Nature methods* (2021).

[64] *Roboflow.* 2023. URL: `https://roboflow.com/`.

[65] *Pascal VOC.* URL: `http://host.robots.ox.ac.uk/pascal/VOC/`.

[66] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *IEEE* (2020).

[67] Nehal Abdul Rehman and Farah Haroon. "Adaptive Gaussian and Double Thresholding for Contour Detection and Character Recognition of Two-Dimensional Area Using Computer Vision". In: *Engineering Proceedings* (2023).

[68] Payel Roy et al. "Adaptive thresholding: A comparative study". In: *IEEE* (2014).

[69] Mohamed Rizon et al. "Object Detection using Circular Hough Transform". In: *American Journal of Applied Sciences* (2005).

[70] Kay Thwe Min Han; Bunyarit Uyyanonvara. "A Survey of Blob Detection Algorithms for Biomedical Images". In: *IEEE* (2016).

[71] Dominik Müller, Iñaki Soto-Rey, and Frank Kramer. "Towards a guideline for evaluation metrics in medical image segmentation". In: *BMC research notes* (2022).

[72] Varduhi Yeghiazaryan and Irina Voiculescu. *An Overview of Current Evaluation Methods Used in Medical Image Segmentation.* Tech. rep. RR-15-08. Oxford, UK: Department of Computer Science, 2015, p. 22.

[73] Jun Ma et al. "Segment Anything in Medical Images". In: *arXiv preprint arXiv:2304.12306* (2023).

[74] Lena Maier-Hein et al. "Metrics reloaded: Recommendations for image analysis validation". In: *arXiv preprint arXiv:2206.01653* (2022).

[75] *Sktime.* URL: `https://www.sktime.net/en/stable/`.

[76] *Sklearn.* URL: `https://scikit-learn.org/stable/`.

[77] Edwin S. Dalmaijer, Camilla L. Nord, and Duncan E. Astle. "Statistical power for cluster analysis". In: *BMC Bioinformatics* (2022).

[78] Tadeusz Calinski and Harabasz Ja. "A Dendrite Method for Cluster Analysis". In: *Communications In Statistics* (1974).

[79] David L. Davies and Don Bouldin. "A Cluster Separation Measure". In: *IEEE* (1979).

[80] Markus Thill et al. "Temporal convolutional autoencoder for unsupervised anomaly detection in time series". In: (2021).

[81] Srinivas Niranj Chandrasekaran et al. "Image-based profiling for drug discovery: due for a machine-learning upgrade?" In: *Nature Reviews Drug Discovery* (2020).

[82] *BCCD Dataset.* 2018. URL: `https://github.com/Shenggan/BCCD_Dataset`.

[83] Bernhard Scholkopf et al. "Support Vector Method For Novelty Detection". In: *NeurIPS* (1999).

[84] Qi-Feng Zhou et al. "Two approaches for novelty detection using random forest". In: *Expert Systems With Applications* (2015).

[85] Markus M. Breunig et al. "LOF: Identifying Density-Based Local Outliers". In: *ACM SIGMOD* (2000).

[86] Ian Goodfellow et al. "Generative Adversarial Networks". In: *NeurIPS* (2014).

[87] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *arXiv preprint arXiv:2112.10752* (2022).

[88] Alexander S. Ecker Leon A. Gatys and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *Journal of Vision* (2015).

[89] Chaoning Zhang et al. "Towards Lightweight SAM for Mobile Applications". In: *arXiv:2306.14289* (2023).

[90] Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *ICLR* (2014).

Appendices

Appendix A

FRETLab User Guide

Appendix A provides instructions on using the FRETLab software. FRETLab was compiled into a standalone distributable folder which launches from a single exe file using PyInstaller. The software was developed for Ubuntu 22.04.03 but will be compatible with later releases of Ubuntu.

To launch the application, navigate to the folder in which the software has been stored. Run the FRETLab.exe file using a double left-click or by expanding the file-browser menu with a right-click and selecting 'Run'. A custom splash screen will be displayed while the software is loading, per Figure A.1.

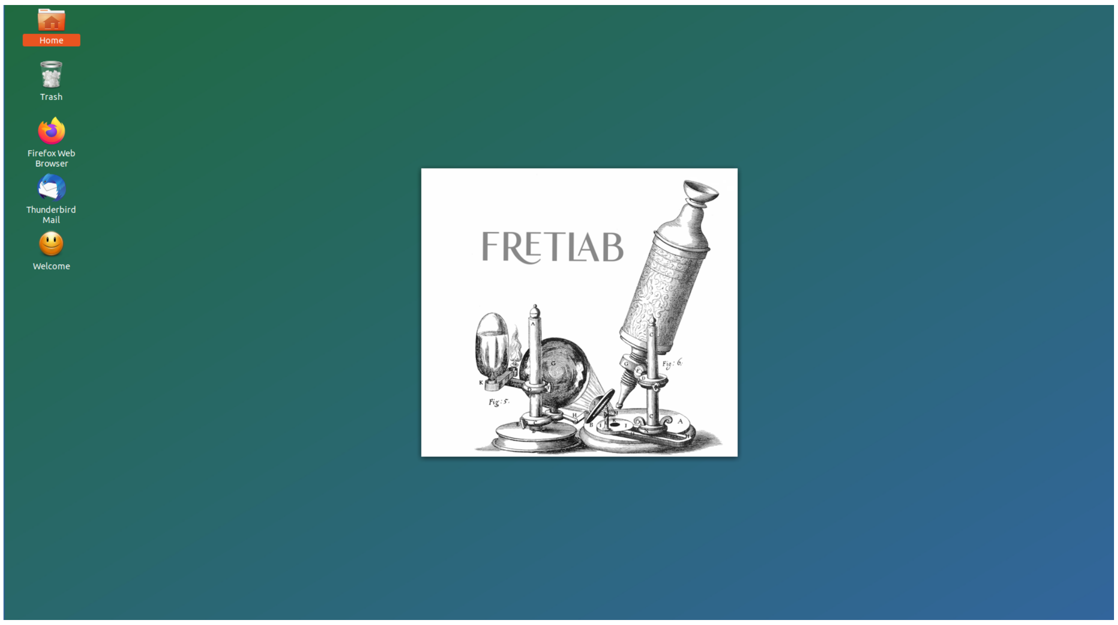


Figure A.1: Launching the FRETLab software.

Once the application has launched, the splash screen will be replaced by the main application window. By default, the main application window will be set to the *'Cell view'* tab, in which image processing of three-channel TIFF files may be executed. Only two options are available to the user at this point: uploading a three-channel TIFF stack or switching tabs to the clustering facility by clicking the tab with the mouse. The clustering facility may be used as a standalone component. This use of this tab will be discussed in the following pages.

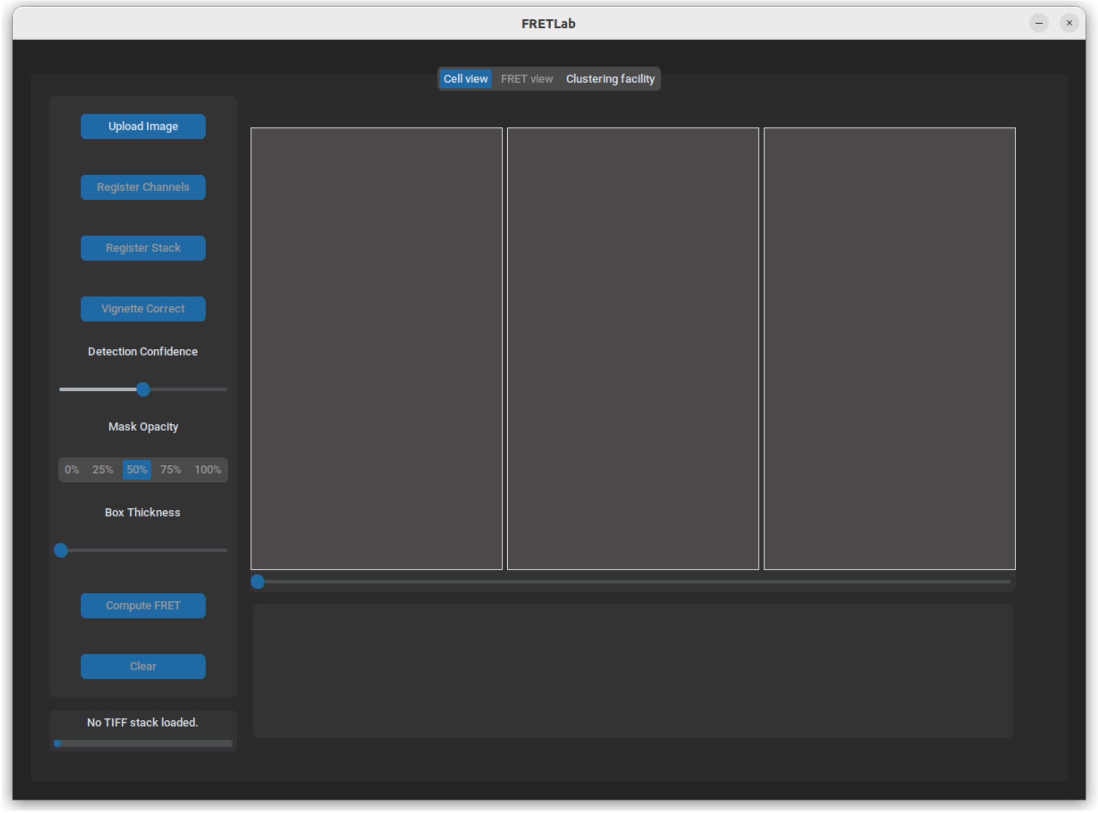
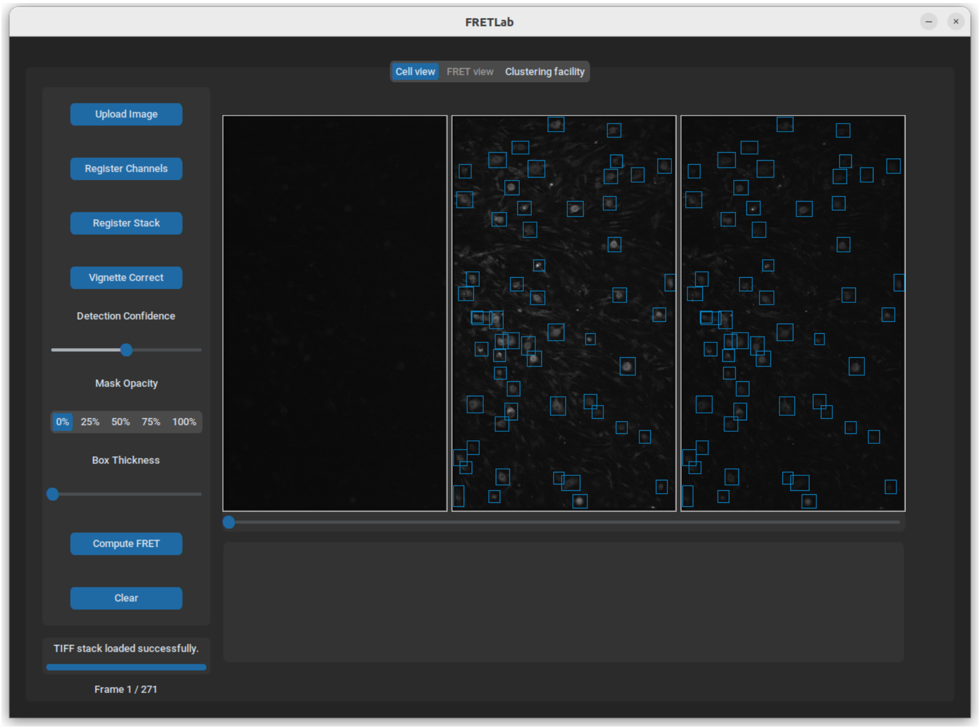


Figure A.2: The initial display of the FRETLab software. On application launch, the user is presented with two panels, with the leftmost providing image processing tools, and the right displaying the area in which three-channel TIFF images are to be loaded.

To upload an image stack, click the *'Upload Image'* button. This will open a dialog box linked to the directory in which the software is saved. Navigate to the desired folder and select a three-channel TIFF image. Once the file has been selected, the loading bar in the bottom left corner of the application will inform the user of the current stack frame being processed. As shown in Figure A.3, when the TIFF stack has been loaded, the top image frame will be displayed along with detections and segmentations.



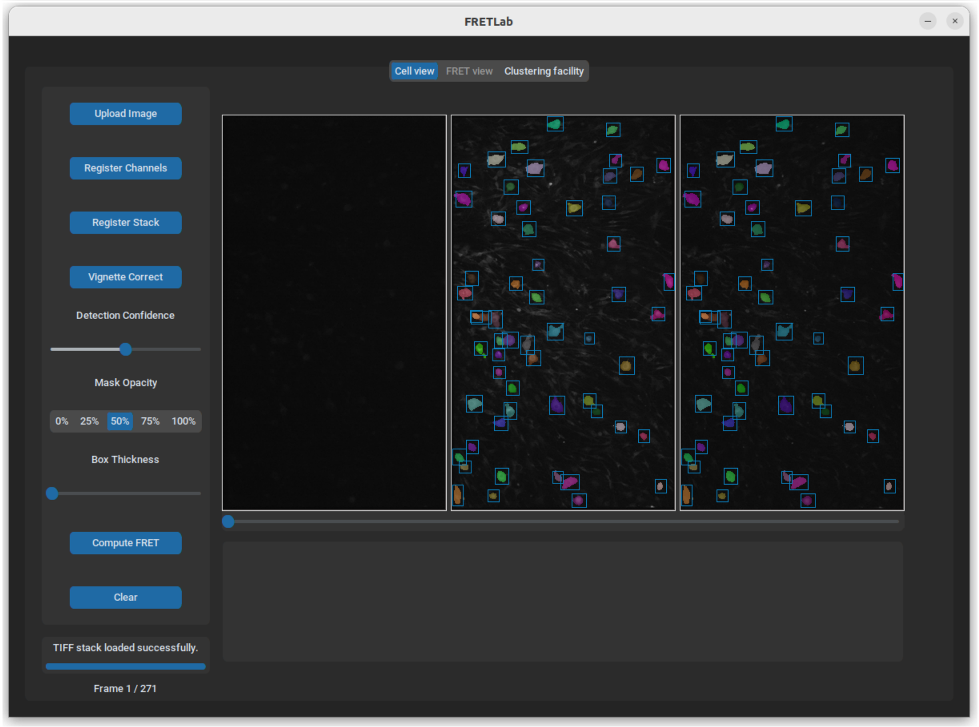


Figure A.3: Cellular detections and segmentations produced by FRETLab once an image stack has been loaded. Bounding boxes indicate the presence of a cell with the confidence threshold set by the slider. At top, the cellular detections with the segmentation mask opacity set to 0% for clear visualisation. At bottom, the same detections, but the with the mask opacity set to 50% to allow the fidelity of the masks to be inspected.

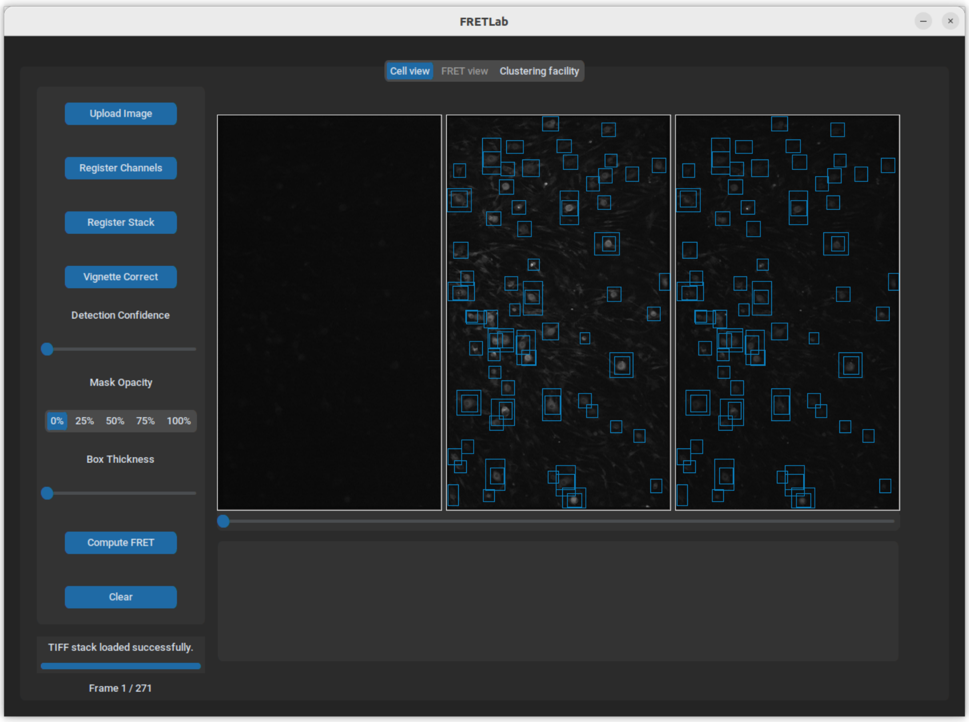
Several utilities become available for use once the TIFF stack has been loaded, namely channel registration, stack registration, vignette correction, detection thresholding, mask opacity modulation, box thickness adjustment, and FRET computation. These utilities range from aesthetic user preferences that aim to make the tool more usable to preprocessing steps that can be used to standardise the computation of cellular drug response.

Channel registration is used to correct for chromatic aberration that occurs between the yellow and cyan filters, while stack registration is used to correct for temporal drift of the microscope due to experimental conditions. Finally, vignette correction is used to correct radial patterns of decreasing pixel intensities emanating from the image centers. To ensure standardisation between the existing image analysis pipeline and FRETLab, the same algorithms have been used for channel registration, stack registration, and vignette correction. The registration utilities are both enabled by the package *pystackreg*, while vignette correction is implemented with the aid of *pybasic*. Both packages are wrappers around the exact algorithms employed within the Fiji plug-ins used in the original pipeline. For *pystackreg* this is the SIFT-based registration algorithm, and for *pybasic* this is the BASiC algorithm.

Confidence thresholding allows the user to change the detections present on the screen on the basis of the confidence score computed by the detection model. These are cached at loading time, and thereafter the detections remain on the canvas at all times, either displayed or rendered transparent according to the current setting. This design allows the threshold slider to update the detections in real time, as opposed to repeatedly rerunning the model. An example is provided in Figure A.4. Further, the detections may be enlarged by increasing the thickness of the bounding boxes using the provided slider.

Should the output of the object detection and segmentation pipeline require refinement, the user may draw bounding boxes around individual cells and the software will produce a new segmentation in real time. An example is given in Figure A.5. The segmentation model has been fine-tuned on a cellular segmentation dataset such that for reasonable perturbations to any given bounding box prompt the segmentation masks produced are invariant. However, this perturbation invariance was enforced only within a narrow pixel window. Thus, with some prompt engineering, the user may 'guide' SAM to produce a more faithful segmentation. Both the automatic and user drawn segmentations may be deleted by left clicking the encompassing bounding boxes with the mouse. The automatic segmentations can be made to reappear by clicking on the threshold slider.

The final feature of the *'Cell view'* tab is a video slider which enables the user to play through the image stack. An example is given in Figure A.6. Coupled with the bounding boxes superimposed over the cells, the user may use this feature to determine whether a cell has drifted over time, and thereby exclude it from the FRET analysis.



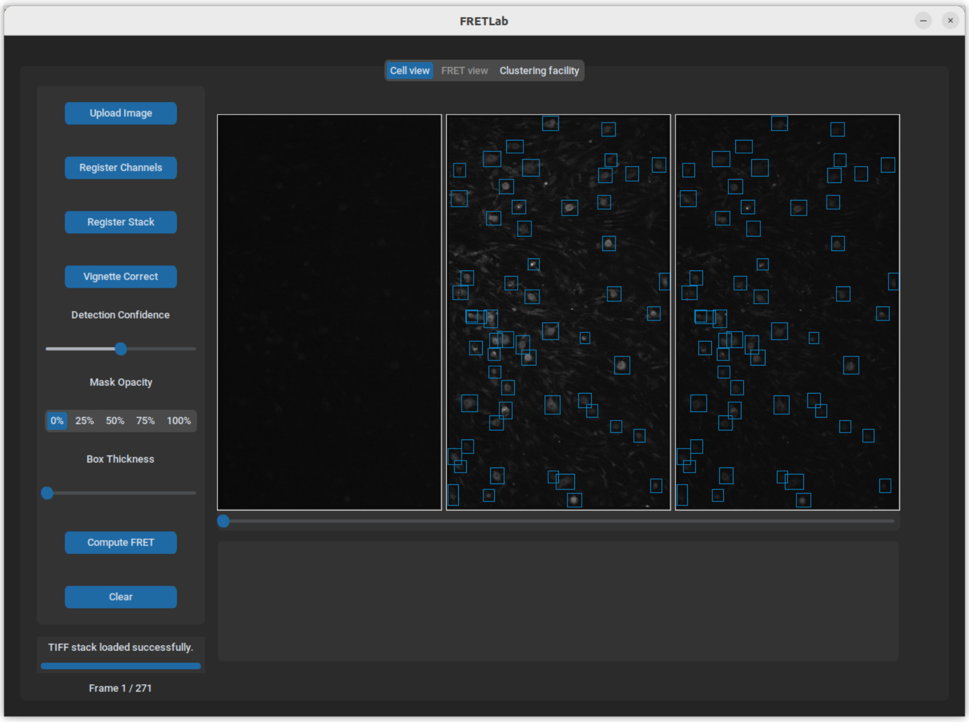
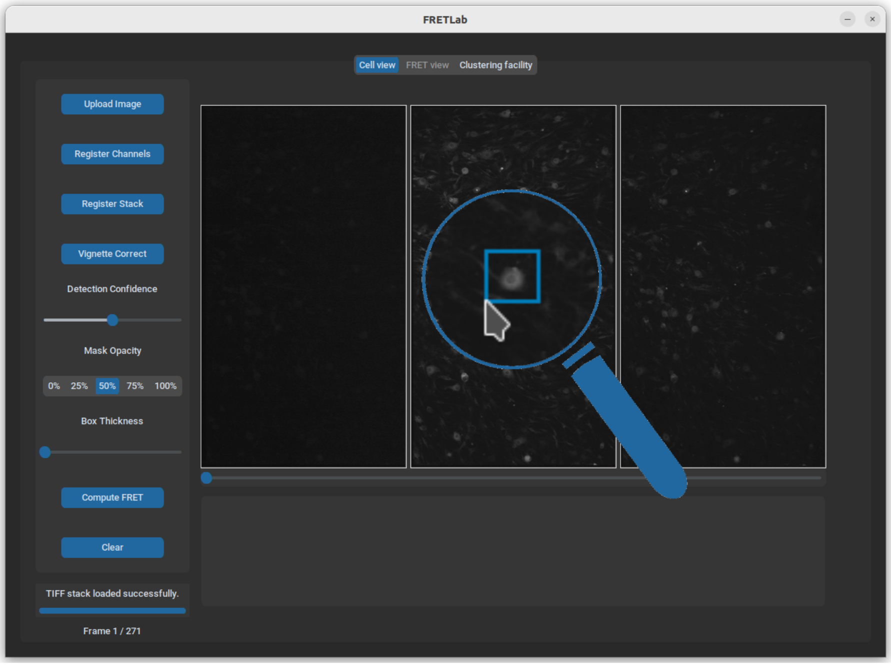


Figure A.4: Demonstrating the effects of confidence thresholding. Both the cellular detections and segmentations can be thresholded in real time utilising the detection confidence slider. At top, a detection confidence of $\geq 0\%$. At bottom, a detection confidence of $\geq 50\%$.



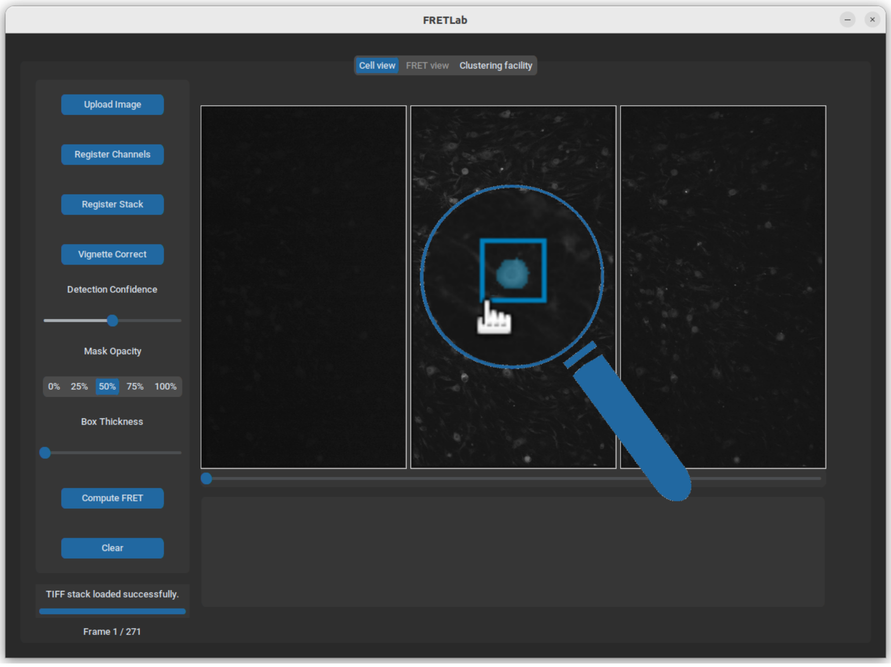
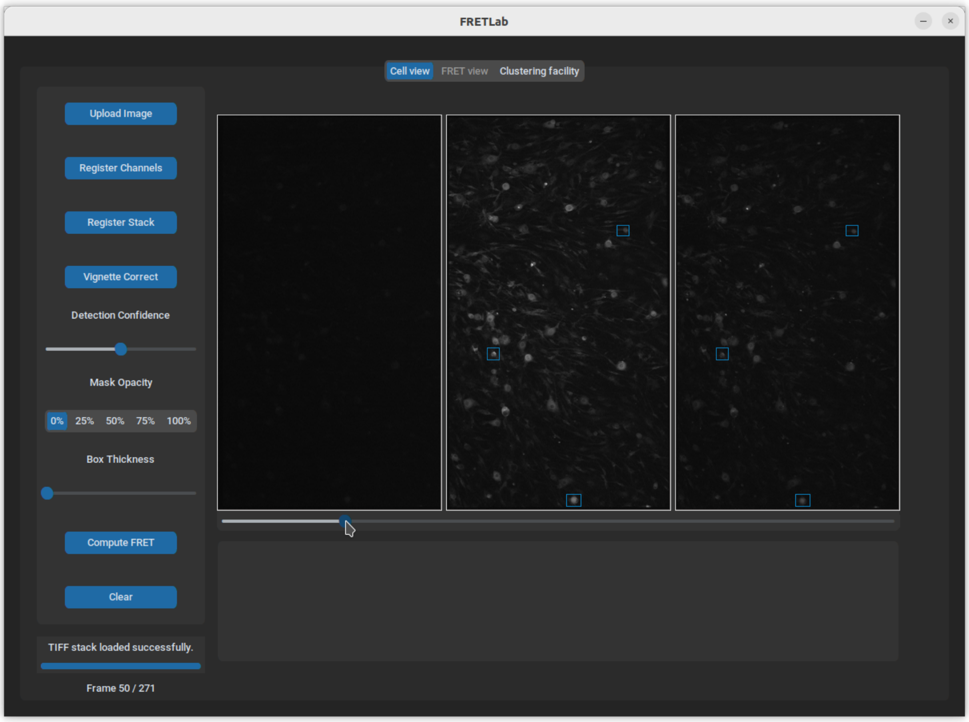


Figure A.5: Drawing segmentation boundaries free-hand with the mouse. At top, the user's bounding box coordinates are transfered from the interactive canvas through to a specialised cellular segmentation model. At bottom, the resulting segmentation.



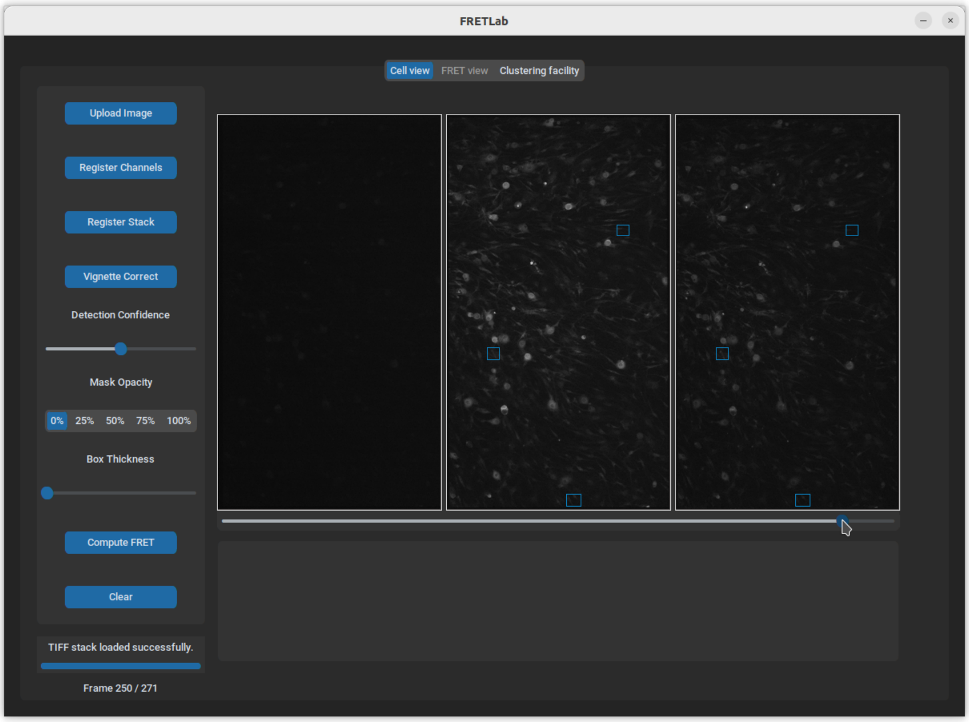


Figure A.6: Viewing the image stack with the mouse using the video slider. Several examples of cellular drift over time have been highlighted with bounding boxes.

Once the user has processed the image stacks and verified the segmentation masks, they may compute the FRET response. Once the *'Compute FRET'* button has been clicked, the tool will process all image stacks in order. At each time-step, the Hadamard product is taken between each binary segmentation mask and both the YFP and CFP channels. Then, the pixel ratios between the two extracted neuronal crops are summed and averaged by the total number of pixels contained in the crop. The scalar value produced by this process is the drug response of each cell C_i at time-step t.

Once the FRET score has been computed for each cell across all time-steps, the interface is set to the second tab of the application, titled *'FRET view'*. As shown in Figure A.7, the *'FRET view'* tab plots the data, and provides utilities for exporting both the raw FRET output and graphs. Four views are provided to aid the analysis of the data: the raw time series data for each cell across each frame in the acquisition, a smoothed version of the raw time series data, a heatmap, and finally a histogram which charts the average FRET ratio per cell over time.

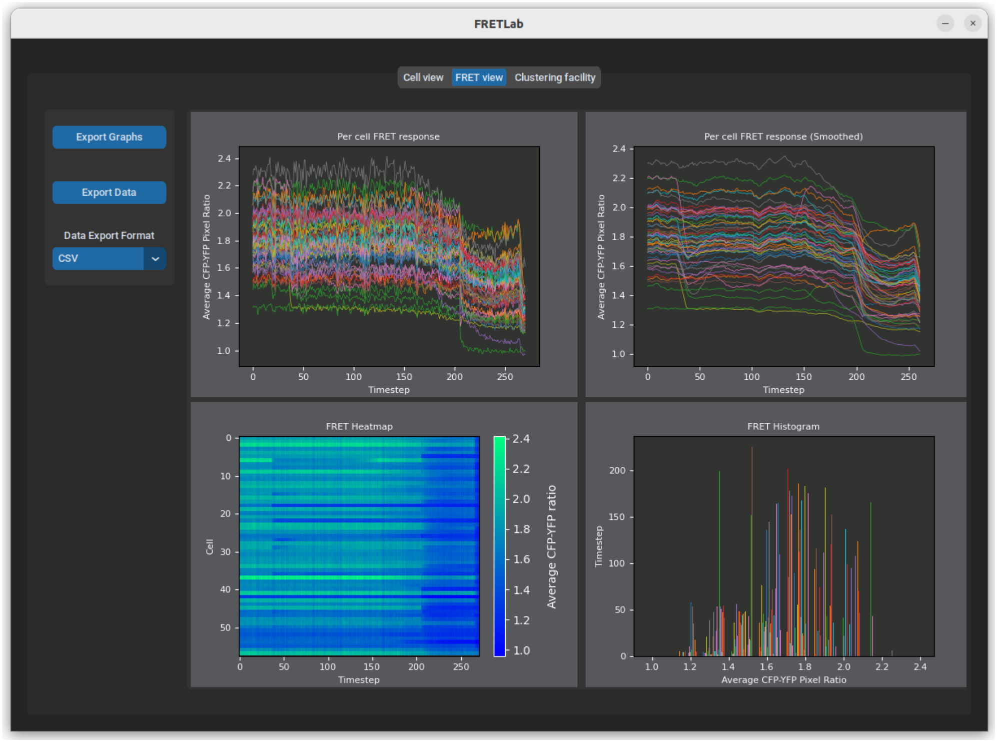
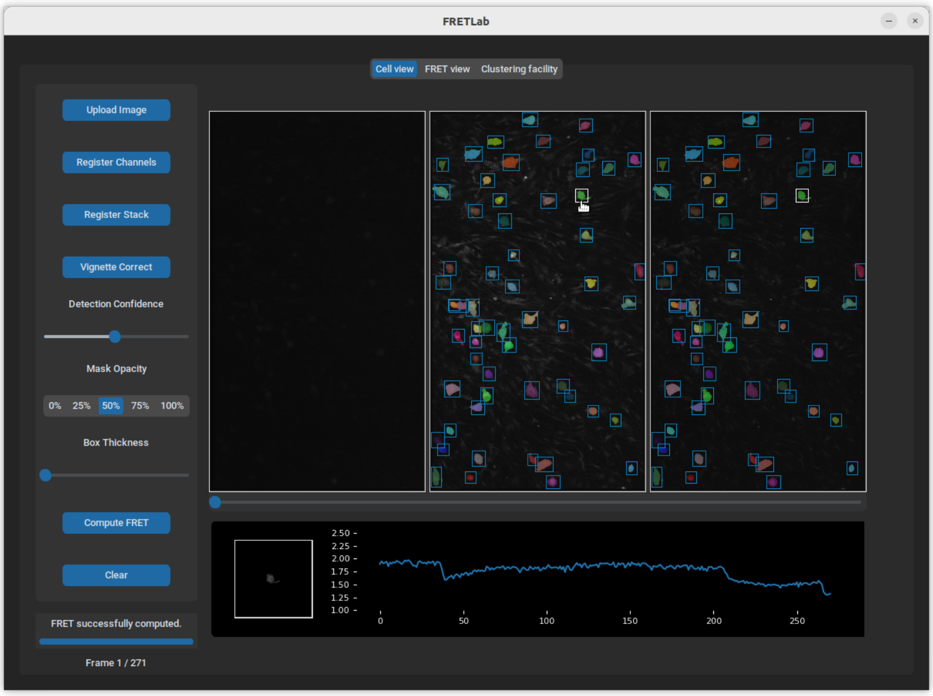


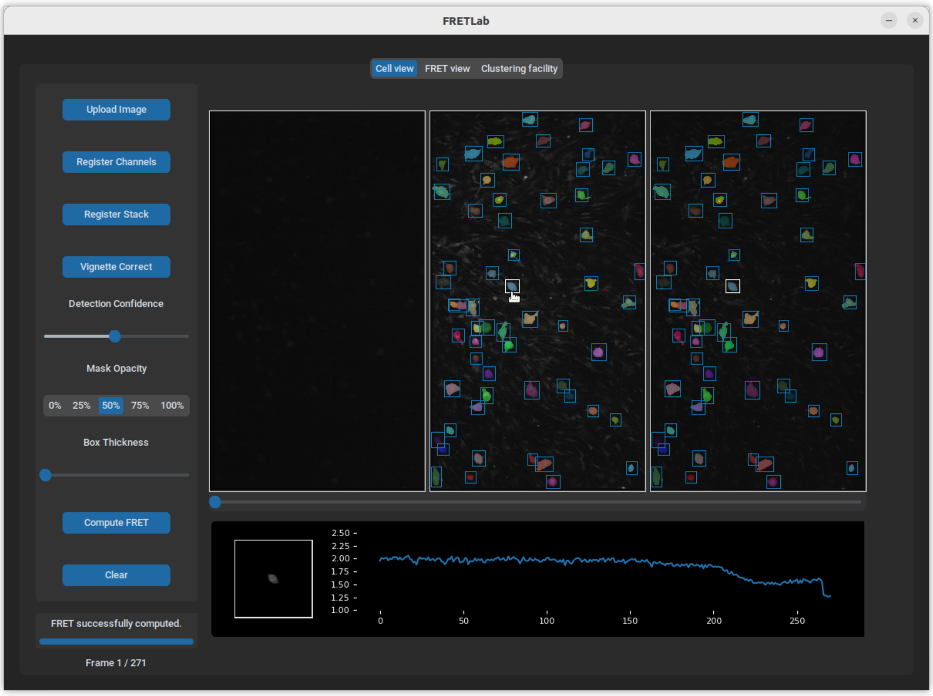
Figure A.7: The *'FRET view'* tab for displaying and exporting FRET data.

Of particular interest when inspecting FRET data are cells which have atypical response profiles. Thus, as per Figure A.8, on returning to the *'Cell view'* tab after FRET computation, the drug response of an individual cell can be visualised by hovering over it with the cursor. Paired with the ability to set the mask opacity to 0 and play the time lapse, it is possible to quickly discern whether an atypical response by any given cell is the consequence of noise, such as drift or occlusions, or a genuine readout.



FRET successfully computed.

Frame 1 / 271



FRET successfully computed.

Frame 1 / 271

Figure A.8: Inspecting the reactivity of two different neurons using the *'Cell view'* tab. After the FRET score has been computed for each neuron, users may hover over each cell and visualise drug response in real time. A magnified neuronal crop accompanies the time series and the bounding box of the active neuron is highlighted.

The final tab is the clustering facility, which provides methods for clustering the data generated by the *'Cell view'* tab. Users may cluster time series data generated within the current session, or upload a folder of files with a uniform format [1] and cluster them as a whole [2]. Files may be those previously generated by FRETLab or another tool.

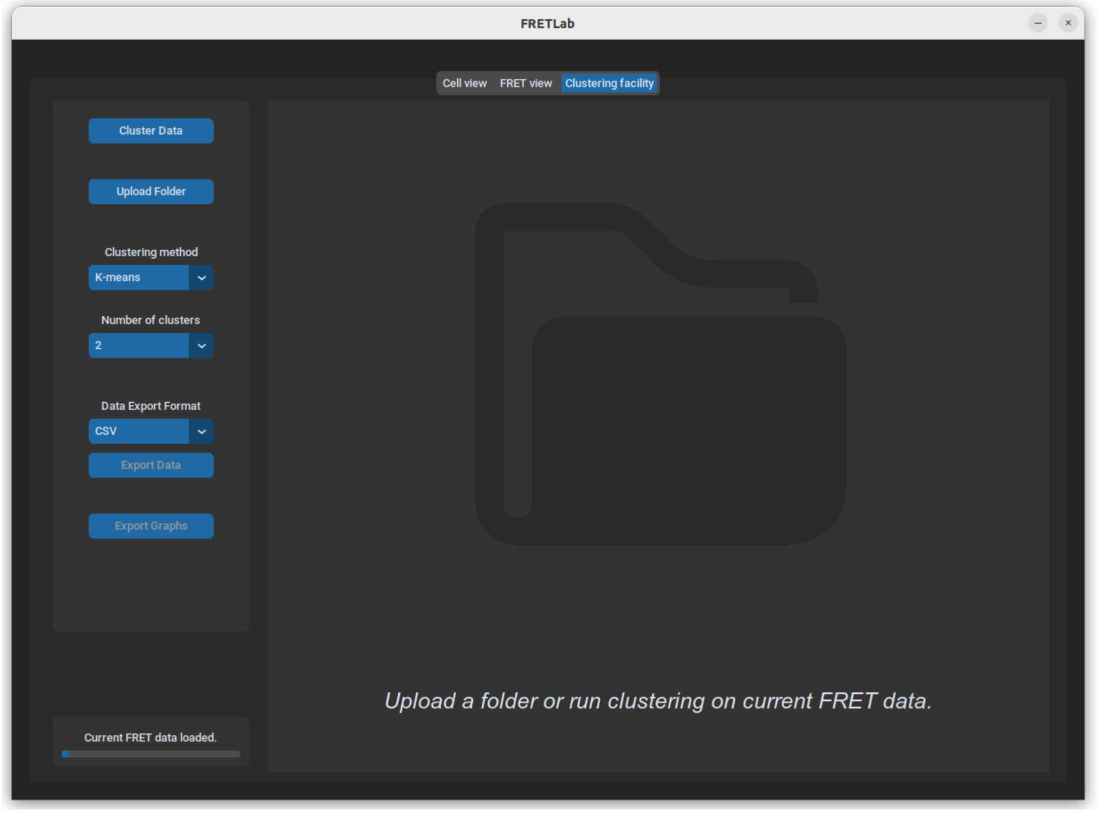


Figure A.9: The *'Clustering facility'* tab. On launching the tab for the first time, users are prompted to either cluster the data produced during the current session, if available, or upload a folder of previously generated FRET data as exported from the *'FRET view'* tab. If the latter option is selected, all files will be treated as a singular dataset and the clustering will be applied to this unified dataset. Options are provided for exporting the produced graphs and clustering data.

The clustering facility may be used to provide an initial insight into the nature of the data and identify whether any trends occur in the FRET curves. Three clustering algorithms are currently available from the drop-down menu: k-means, k-medioids, and kernel k-means. Each method employs a shift invariant dynamic time warping metric.

[1]Either .csv or .npy, being the options provided by the FRET view tab.
[2]This requires that the acquisitions be of the same length.

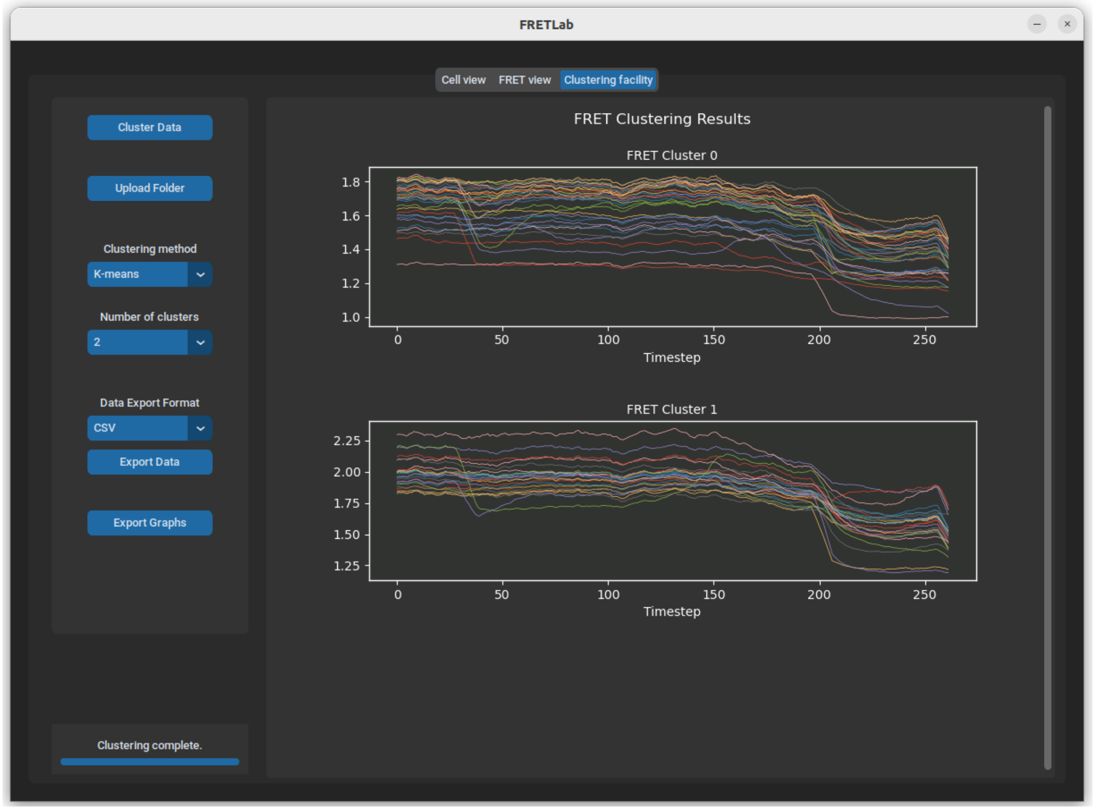


Figure A.10: Results from the clustering facility for the FRET computation performed in Figure 4.5. A drop-down menu is provided to enable the user to cluster the data up to a maximum of 10 clusters. This limit was imposed on the basis of biological domain knowledge of the maximum plausible number of groupings expected from the neurons.

The exported cluster data will be organised into c rows, where c is the number of segmented cells in the acquisition for which the FRET score was computed. Further, there will be $t + 1$ columns, where t is the number of timesteps in the acquisition. The first column will contain the cluster labels, which are integers from 0 to $n - 1$, where n is the number of clusters selected. All columns from the second column rightwards contain the raw FRET data at each time point for every segmented cell.

Appendix B

Time Series Analysis

Appendix B presents the silhouette plots that support the time series analysis of Chapter 5. In particular, presented below are the silhouette plots for the eight independent ExD3 drug trials. Observe the dominance of the average silhouette score for $k = 2$ clusters across all experiments with the exception of the 410A experiment.

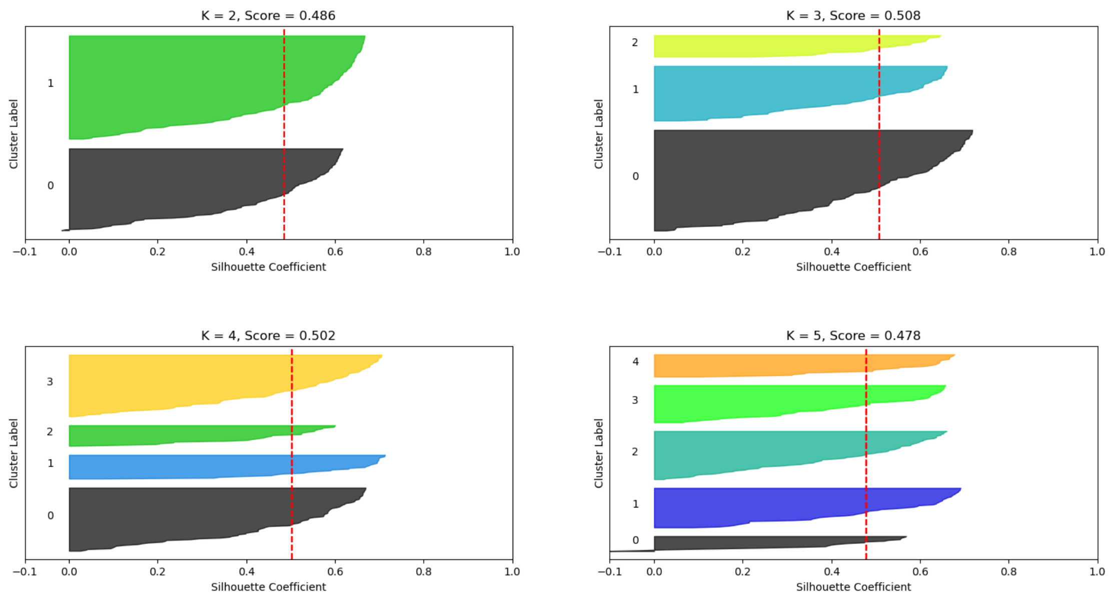


Figure B.1: Experiment 410A

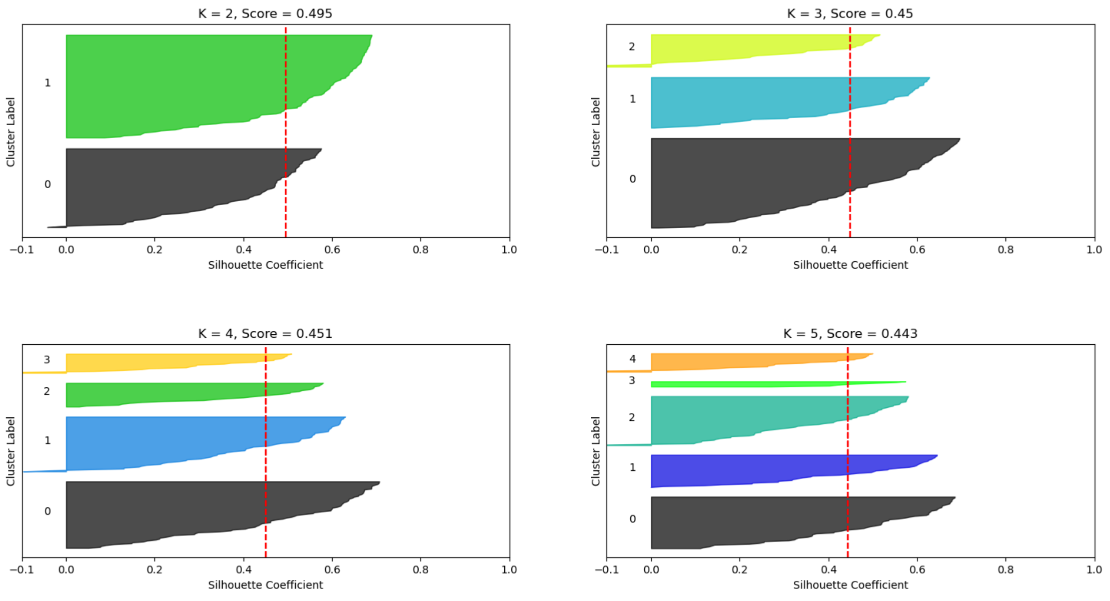


Figure B.2: Experiment 410B

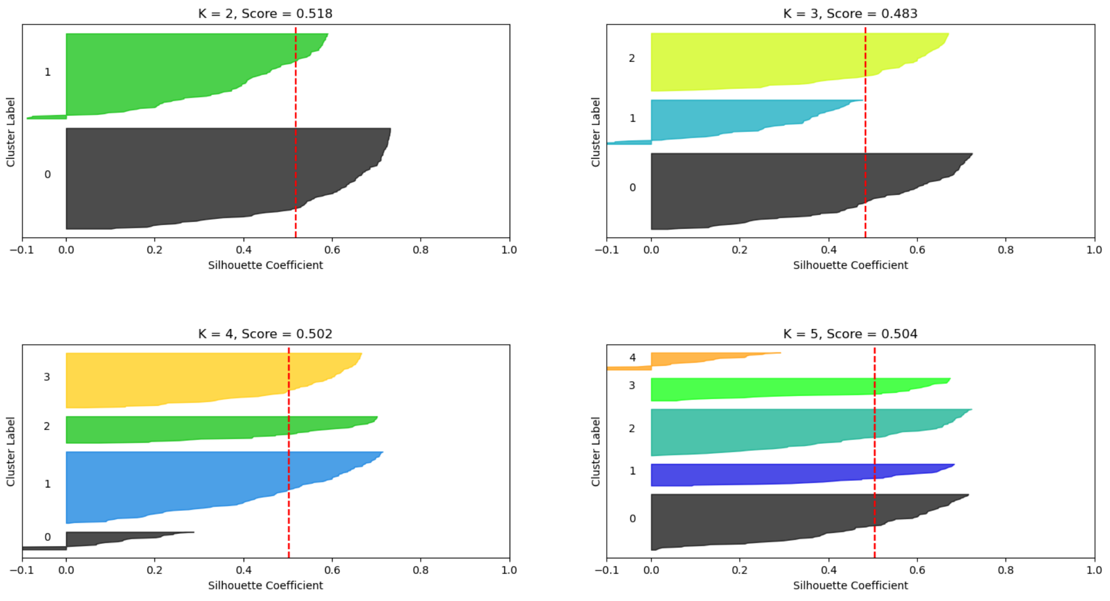


Figure B.3: Experiment 411A

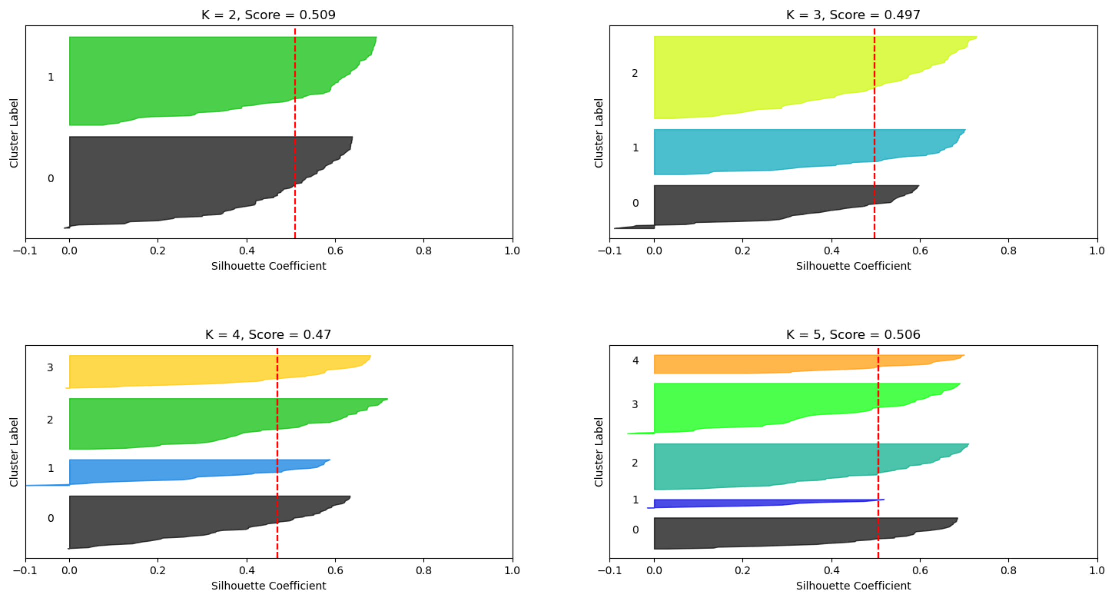


Figure B.4: Experiment 414A

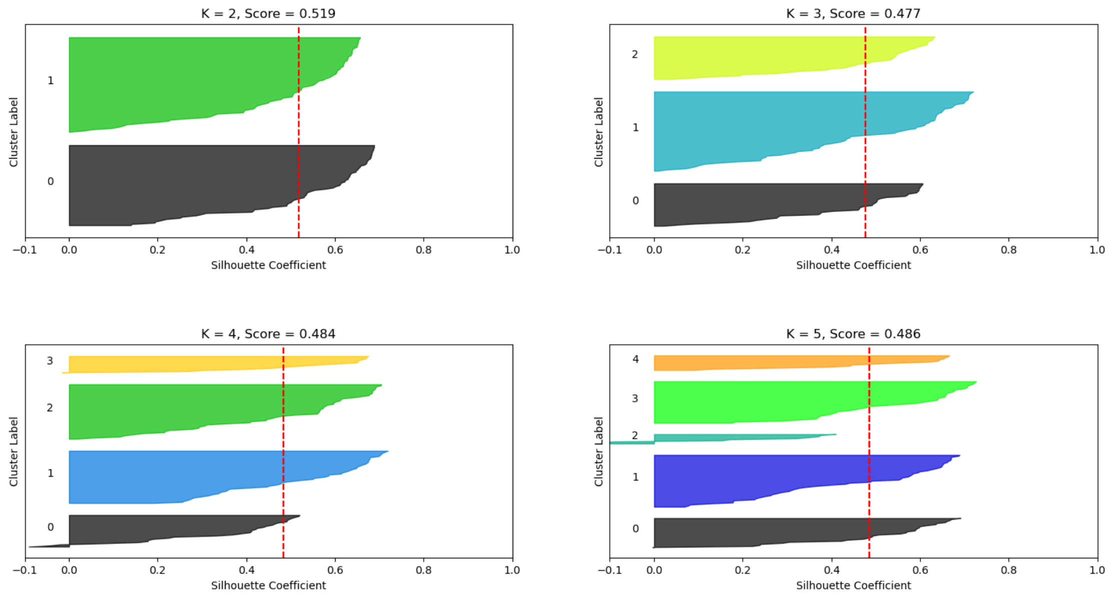


Figure B.5: Experiment 414B

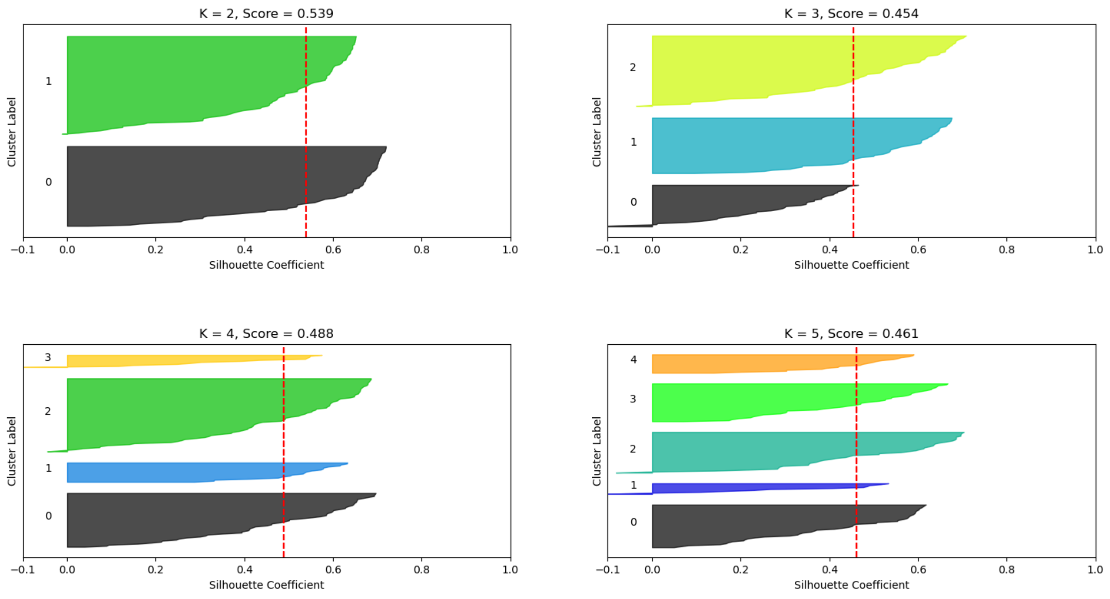


Figure B.6: Experiment 415A

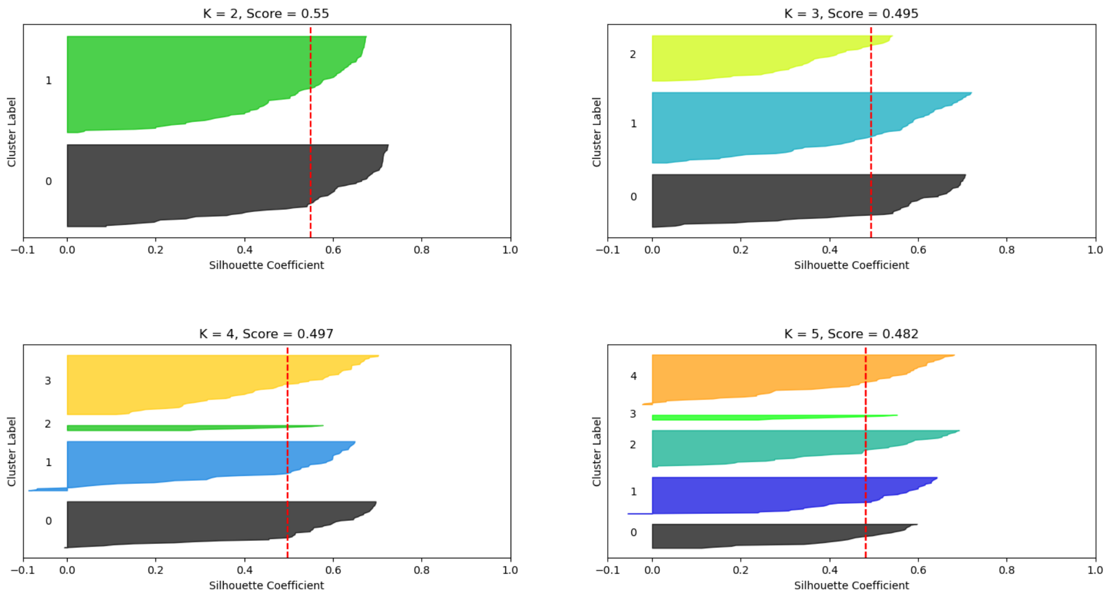


Figure B.7: Experiment 415B

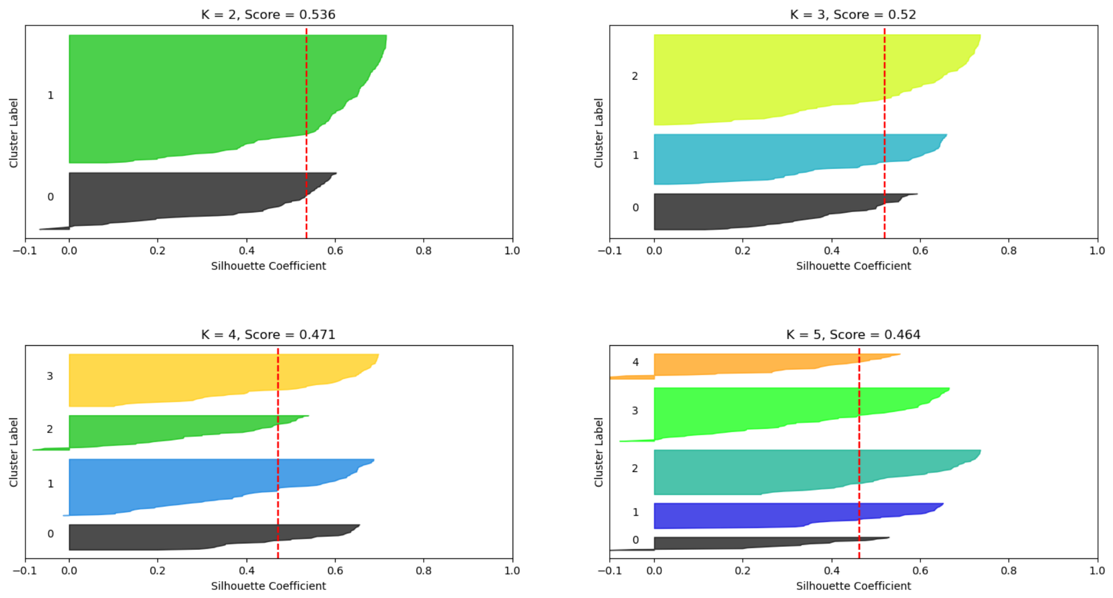


Figure B.8: Experiment ACTB