National University of Computer and Emerging Sciences

Weakly-Supervised Cell Instance-Segmentation in Multi-Modality Microscopy Images

FYP Team

Ayesha Kanwal……19L0902
Najia Ikhlaq……..…19L1169
Saqib Ali……19L0939

Supervised by

Dr Hammad Naveed

FAST School of Computing

National University of Computer and Emerging Sciences

Lahore, Pakistan

May 2023

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

Title: Weakly-Supervised Cell Instance-Segmentation in Multi-Modality Microscopy Images

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as **Plagiarism**. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: 12th October 2022

Student 1

Name: Ayesha Kanwal

Signature: _____

Student 2

Name: Najia Ikhlaq

Signature: _____

Student 3

Name: Saqib Ali

Signature: _____

Authors' Declaration

This states Authors' declaration that the work presented in the report is their own, and has not been submitted/presented previously to any other institution or organization.

Abstract

Biomedical segmentation is an important precursor to medical diagnosis, the localization of pathology, study and analysis of the anatomical structures, planning of treatment for a specific illness, computer-aided surgery, and the quantification of tissue volumes. It also includes cell segmentation, which is an essential part of the aforementioned fields, especially for assistance in diagnostic routines. Moreover, it is an essential step for conducting downstream single-cell analysis in biomedical research. Various deep learning models have shown promising results in terms of cellular segmentation, but each model has certain limitations. We find that a significant limitation of most of these models is that they require many annotated images with ample variety. Additionally, most datasets are limited to one modality and trained models perform poorly on cross-modality images. Our project makes use of largely unlabelled samples to improve the model's performance on very little labelled data. This project pursues cell segmentation with a weakly-supervised learning approach and produces a general-purpose segmentation framework that allows you to segment cells from diverse modalities. Along with this, we provide an accessible and easy-to-use web-based tool to run the proposed algorithm.

Executive Summary

Cell segmentation is the process of highlighting and separating cellular bodies from images. It allows the study and analysis of biological features such as cell count, type, division that assist in downstream tasks of diagnosis and disease localization. Furthermore, this also helps in the advancement of biology, medicine, drug discovery and personalized medicine. The project finds its use cases in pathology. Training cell segmentation models that work effectively is a challenging task due to a lack of a large number of annotated images. Even state-of-the-art deep learning models are unable to perform well with this deficiency. The limiting factor to obtaining these annotations is the fact that annotations need to be done by specialists who can better understand and interpret the images. Additionally, there can be many impurities in whole-slide images making it challenging for pathologists to label these cells. Working with these images is a tedious task. As [44] states, it took three experts 2.5 hours to manually segment five whole-slide images on average with above-average accuracies. Since whole-slide images can contain anywhere from a single cell to millions, it becomes a tedious task to manually annotate these instances.

In this report, we start with a review of traditional cell segmentation methods such as region-growing, thresholding, edge-detection, and clustering. Such methods are not effective in a biomedical setting due to flat contrast levels in most images. Furthermore, these algorithms are tailored to the dataset at hand and cannot generalize well on a variety of cell shapes and sizes. Another reason to look beyond these segmentation algorithms is the prohibitive computational cost of the inferences made through these operations. Ultimately, it is found traditional segmentation techniques are non-ideal for the project.

Consequently, we look into deep learning-based architectures that handle the majority of the feature engineering, i.e. what features to select, patterns to identify, and thresholding to apply. In this regard, many state-of-the-art deep learning architectures exist such as Fast R-CNN, Faster R-CNN, and Mask R-CNN, with each of them having their strong points. However, it is observed that these architectures are highly reliant on good-quality pixel-level annotations. Moreover, their evaluation is primarily conducted on real-life images where objects tend to have a high difference in contrast and colours.

While these architectures are excellent for performing inferences on real-life scenarios, they tend to underperform when it comes to biomedical images. This leads us to our next research avenue, biomedical image segmentation architectures. One key attribute of such architectures is that they all are some variants of U-Net with minor improvements or backbone alterations. The basic architecture of U-Net consists of a series of downsampling layers with convolutions, max pooling, and a non-linear activation function. This so-called contracting path connects with a series of upsampling layers which U-Net terms the expansion path. Using this architecture, U-Net variations achieve state-of-the-art scores on challenges with slight modifications.

A highlight of these biomedical image segmentation architectures is the presence of skip connections. We deeply explore the effect of skip connections within a network and how they can help with the flow of gradients while giving a boost in model convergence. These skip connections also help in propagating higher-level details to the expansive path of the U-Net-based architectures which they generally lose while performing downsampling. The U-Net requires fewer annotated samples of pixel-level annotations, but the requirement is still vastly more than the available datasets.

To remedy this, weak supervision comes into play. We explore a collection of weak-supervision techniques such as weighted loss training, pixel-level mask synthesis, utilizing point annotations, automatic mask generation, and transfer learning to apply concurrently with

U-Net-based architectures. Furthermore, we have come up with our novel weak-supervision framework to assist in improving the performance of the network. Another avenue for improvement in this problem was through revisiting cell representation. In this regard, we explored cell-representation-based algorithms such as Cellpose and Omnipose that use the same architecture as the U-Net but instead resort to handling cells in form of mathematical diffusion-based gradients. We have made use of these diffusion-based algorithms in our project as well. Cell segmentation in a weakly-supervised context is an exciting area for research and has many tangential applications.

Our goal in this project was to create a utility that can implement state-of-the-art algorithms and improve on them using a combination of weak supervision, post-processing, and modern deep-learning methodologies.

Our system will be used by pathologists who want to get segmented cell images. Users will be able to run inferences on cell images and get segmentation results. The application will use Mongo Express React Node stack for development. Regarding the machine learning models, python and flask will be used for the backend operations. For the frontend methods, the user interface will be designed according to the constraints of Material UI design and validated through Apple's Human Interface Guidelines. The version control system for this project is GitHub. In terms of our design, we have a local client and a web client that can communicate with the tool through an API. The client and backend modules cater to the major needs of the user.

To counter the problem of less datasets with annotated cell images, we have devised a methodology that revolves around weak supervision. The proposed techniques include pre-processing and normalization, localization, post-processing, and label correction details.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Cell segmentation is an essential step for conducting downstream cell analysis in biomedical research. It is a rudimentary step in biomedical studies and is considered the foundation of image-based cellular groundwork. Practical usage of cell segmentation allows the analysis of features like the cell shape, size, count, type etc. This cell tracking allows the quick evaluation of how these features progress over time, their migration, the morphological changes they encounter and how they respond to different conditions. This discernment aids in diagnostics, the discovery of drugs and relevant branches of biology, medicine and pharmacology. Despite the progress in various cell segmentation techniques, there are certain impediments. These limitations include the acquisition of quality datasets, low variety in cell shapes in structure, modality-specific architectures, and high cost of cell annotations [1] - which are usually done by qualified medical professionals.

Pathologists are medical professionals who conduct tissue and cell analysis through the use of pattern recognition, deduction, and reasoning while employing necessary testing techniques. A pathologist's path to diagnosis is composed of multiple evidence-based inferences which can be costly in terms of time as well as money [2]. Our project aims to help pathologists and related medical professionals by providing them with an efficient tool to run cell segmentation procedures on major modalities of microscopic images and assisting them in their standard decision-making and diagnostic routines. These computer-generated inferences can significantly reduce the time taken for standard tests and allow pathologists to conduct evaluations with higher efficiency. This potential synergy of pathology and artificial intelligence can further be augmented through the utilization of faster networks and cheaper storage options to collect whole-slide images [3]. Additionally, recent advances in cloud computing and neural network optimisation have allowed the deployment of such computation-intensive systems without a need for the end-users to have powerful machines. Our project will utilize a combination of state-of-the-art algorithms and techniques to implement and hopefully improve cell segmentation algorithms with a weak-supervision constraint. Moreover, we aim to pair this tool with a user-friendly public-facing interface to boost the diagnosis capacity across the labs.

The first section provides a comprehensive introduction to the focus of our project and clearly outlines the specific objectives to be achieved during the course of our research. In the second chapter, we elaborate further on our problem with deeper insights into the challenges and potential research gaps that can be addressed in the near future. Moving to the third chapter, we conduct a thorough literature review relevant to our problem which includes a background of the problem along with analysis of the state-of-the-art techniques that can be applied for the solution of the given problem.

1.1 Purpose of this Document

The purpose of this document is to give a detailed elaboration of the research problem while exploring state-of-the-art solutions, highlighting previous research, identifying research gaps, and analysing techniques to enhance and improve existing solutions. Additionally, it also describes the work that has been conducted so far by our group.

1.2 Intended Audience

The intended audience for this document is biomedical researchers, academics, students, and learners with above-average technical knowledge of artificial intelligence-related terminologies and techniques.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Definitions

Bright-field microscopy: Bright-field microscopy is a simple microscopy technique that employs bottom-lit imaging and captures the contrast between high-passing and dense areas of the sample. Due to its simplicity, it's one of the most widely used microscopic modalities.

Phase-contrast microscopy: Phase-contrast microscopy maps the phase shifts in lights passing through the sample. It is usually performed on transparent or semi-transparent specimens. It reveals many cellular structures that are generally unable to be seen through simple bright-field microscopy.

Differential-interference contrast microscopy: It is interferometry-based microscopy which is used in unstained or transparent samples to highlight their contrast and differences which allows the person to view features that would be otherwise invisible in other microscopies.

Fluorescent microscopy: It is a type of microscopy that maps the behaviour of a specimen under exposure to electromagnetic radiation. The microscope captures the emitted light from the sample after exposure.

Neural Network: These are computing systems that emulate mathematical models and are loosely modelled on the biological structure of the brain, i.e. neurons connected to other neurons and firing upon certain activation thresholds.

Convolutional Neural Network: This is a neural network driven by operations known as convolutions. Convolutions are an important part of image processing in which certain kernels/filters are applied to images to extract features. Convolutional neural networks exploit this feature extraction to learn patterns from images and update filters for precise results.

Instance Segmentation: This refers to the process of highlighting instances of a class within an image. In this segmentation, we are only concerned with the cases of the object rather than its type.

Semantic Segmentation: This refers to the process of highlighting instances of certain classes within an image and then classifying them into their specific classes. This segmentation is simply instance segmentation paired with a classifier.

Weakly-Supervised Learning: This type of learning uses a set of crude or low-cost labels to emulate high-cost annotations using several augmentation and refinement techniques. The model utilizes these techniques to compensate for the lack of high-quality annotations.

Semi-Supervised Learning: This learning pairs unlabelled data with labelled data to train the model using several "loss" weighting and pseudo-labelling techniques.

1.3.2 Acronyms

TIFF: Tag Image File Format

1.3.3 Abbreviations

SDG: Sustainable Development Goal
WSI: Whole Slide Images
JPEG: Joint Photographic Experts Group
PNG: Portable Networks Graphic
GPU: Graphics Processing Unit

Chapter 2: Project Vision

The following section elaborates on the domain of our problem, the problem statement and its elaboration. Moreover, it also discusses the goals and objectives that we aim to achieve, the scope of our project and the SGD that this project aims for.

2.1 Problem Domain Overview

The general domain for this problem would be biomedical computing. This domain utilizes the problem-solving capabilities of computer science along with the state-of-the-art techniques required for the improvement of healthcare. More specifically, this problem would aim at providing an efficient tool that assists in medical diagnosis on the basis of computational mechanism i.e., cell segmentation. Computational techniques like weak supervision and deep learning will be put into practice to carry out multi-microscopy cell segmentation. Moreover, a surface knowledge of the different modalities of cells is also required to identify them based on their associated characteristics.

2.2 Problem Statement

A generalized model for cell segmentation with the constraint of weak supervision is an area that has little work done on it. Upon analysis of the present research, it can be concluded that the majority of the work conducted on the segmentation of cells uses techniques other than weak supervision or the target is a specific modality rather than multiple modalities. Addressing this problem will allow medical professionals, researchers and pathologists to annotate the images at a lower cost and in general medical diagnosis as well. Our objective is to implement state-of-the-art models and refine them for better accuracy. This addresses the aforementioned research gap.

2.3 Problem Elaboration

Segmentation of cells is a significant challenge in biomedical computing with multiple avenues for research and improvement. Everything from cell representation, deep learning architectures, image processing techniques and learning routines poses an opportunity to improve on already existing models. Cell segmentation is an important problem because it helps pathologists, researchers, and medical professionals. It is a prerequisite to many pre-diagnosis routines. Like many biomedical problems, cell segmentation algorithms exist and have given state-of-the-art performance on a variety of datasets. However, most of these architectures are data-driven and require large amounts of annotated data. Such annotated data can drive up the cost to develop tools. Moreover, the annotations need to be done by qualified medical professionals. In addition to the high cost of data acquisition, the model's ability to generalize on a variety of modalities is another aspect of this problem. Most state-of-the-art models are tailored and perform well on similar cell structures or single modalities. This is limiting and can pose an issue in cases where the model needs to infer a completely different cell shape or modality.

2.4 Goals and Objectives

Cell segmentation plays a crucial role in assisting pathologists while conducting cell analysis, drug discovery, and performing diagnostic operations. Despite a significantly important role, weakly-supervised cell segmentation is largely an unaddressed challenge in modern research. Our project develops ease and convenience for medical professionals and has the following objectives:

- Utilize a few labelled samples and generalize the model to multiple modalities.

- Apply state-of-the-art models and possibly improve them to accurately segment cells.
- Lessen the costs associated with data annotation and medical diagnosis.

To learn about the dataset source and general statistics, see Appendix A.

2.5 Project Scope

We will provide a tool for pathologists and medical professionals to accurately perform cell segmentation on whole-slide images taken from various microscopic modalities. Additionally, we will build on top of state-of-the-art algorithms to improve their performance either in resource consumption or accuracy. We will leverage techniques like weak supervision, deep learning, cell representation methodologies, transfer learning, and morphological post-processing to find a balance between utility and performance and allow pathologists to make better diagnostic decisions in their routine work.

2.6 Sustainable Development Goal

Our project focuses on the SDGs that relate directly to the healthcare and social improvement domains. We aim to target the following areas in the project:

2.6.1 Good Health and Well-Being

The UN states multiple targets following this SDG. One of the relevant targets for this goal (3.d), highlights the need to strengthen the diagnostic and testing capacity of countries across the world, especially in developing countries. Needless to say, robust diagnostic infrastructure can help in the timely interception, reduction of risk, and national health risk management. Additionally, target (3.8) highlights the need to provide universal coverage in terms of quality healthcare services and affordable healthcare. Our project works towards this SDG by focusing mainly on the targets (3.d & 3.8). Since our tool assists in improving diagnostic capabilities, pathologists can make faster and more accurate inferences which ultimately leads to the early interception of diseases. Furthermore, as diagnosis takes much less time than standard methods, pathologists can cater to a lot more cases with overall affordable costs.

Chapter 3: Literature Review / Related Work

The following section covers a comprehensive literature review of previous research done concerning our project while also providing detailed insights and knowledge into how the concepts can be applied in our context.

3.1 Definitions, Acronyms, and Abbreviations

3.1.1 Definitions

Histogram: A histogram represents the distribution of colours within an image.

Fuzzy Logic: In contrast to boolean logic where a state can be either 0 or 1, fuzzy logic advocates partial truth. Fuzzy logic advocates the presence of partial truth, i.e. states can be between 0 and 1.

Hue: The attribute, shade, or property of a colour that allows it to be classified into three primary colours, namely red, green and blue.

Saturation: The concentration of a certain colour within an image.

Luminance: The lightness or darkness of an image.

3.1.2 Acronyms

PET: Positron Emission Tomography

3.1.3 Abbreviations

MRF: Markov Random Field
CT: Computed Tomography
MRI: Magnetic Resonance Imaging
US: Ultrasound
ANN: Artificial Neural Network
CNN: Convolution Neural Networks
ROI: Region of Interest
FCN: Fully Convolutional Neural Networks
FCRN: Fully Convolutional Residual Networks
DCNN: Deep Convolutional Neural Networks
RNN: Recurrent Neural Networks
GPU: Graphics Processing Units
FCM: Fuzzy C-Means
PCA: Principal Component Analysis
BCFCM: Bias-Corrected Fuzzy C-Means
IFCM: Improved Fuzzy C-Means
RCNN: Region-Based Convolutional Neural Networks
DIC: Differential Interference Contrast
CAM: Class Activation Map
PRM: Peak-Response Map
RPN: Region Proposal Network

3.2 Detailed Literature Review

This section covers notable literature published in the domains of weak supervision, instance segmentation, biomedical image segmentation, cell representation, and morphological post-processing while giving detailed knowledge of every technique with its contribution to the overall improvement of the system.

3.2.1 Image Segmentation

To approach cell segmentation, it is necessary to have an underlying knowledge of the image segmentation problem and associated solutions. This section covers literature oriented towards image segmentation and various techniques developed throughout the years to assist in this regard. Our selected papers explore various segmentation techniques, challenges, and types of segmentation.

3.2.1.1 Image Segmentation Techniques and Color Models

Image segmentation refers to the partitioning of an image into segments based on certain conditions. These conditions may be based on the colour differences, contrast, edges, or any other feature present in the image itself. This paper by Savita Agrawal et al. [5], discusses various image segmentation techniques developed throughout the years. They classify algorithms into two categories.

- Tracking differences in intensity.
- Tracking similarity in regions.

Within these categories, they divide segmentation techniques into various classes depending on their core approach or algorithm to partitioning images.

3.2.1.1.1 Edge-Detection Based Segmentation

In this form of segmentation, the algorithm attempts to detect the edges present within the image and conducts a close object boundary analysis. By itself, edge detection is an actively researched field in image processing. These methods rely on the precondition that there is always a sharp difference in intensity between partitions of an image. Consequently, edge detection becomes a basis to identify those differences and ultimately carve out the specific regions in the image. An example of segmentation through edge-detection is proposed in [6], which uses already traced edges and segments it into straight or curved edges for part-based object recognition. In a nutshell, edge-detection-based algorithms are based on the inherent difference between the intensity of certain regions in images. In images where the contents are largely homogeneous, it would have a difficult time performing segmentation. Furthermore, it is a tedious task to generate boundaries through any edge-detection algorithm.

3.2.1.1.2 Thresholding Based Segmentation

As implied from the name, this method relies on some form of threshold mechanism to identify regions of interest within the image. Prior knowledge of the image itself is helpful to apply these algorithms but is not necessary. Usually, the thresholds are applied to the colour distributions or more commonly known as image histograms. Thresholding can be very effective in some cases, especially when the region of interest has a clear distinction from its background, however, this distinction is hard to come by in real-life biomedical use-cases. Notably, it is highly prone to image noise which is an inherent part of microscopic images.

3.2.1.1.3 Region Based Segmentation

The motivation behind region-based segmentation derives from the fact that edge-detection and thresholding methods fail to show credible results when images exhibit less or no distinction within their composition. Additionally, their inherent proneness to artifacts and noise can affect the overall performance of segmentation. In this case, region-based segmentation algorithms provide a method to accommodate noise and perform segmentation on images with relatively flat appearance. Region Growing and Region Splitting & Merging are two major types of region-based segmentation methods. The primary idea of region-based segmentation is to have initial starting points for segmentation, commonly termed seeds. From that point onwards, the algorithm attempts to cluster homogeneous regions by either growing the seeds or by using a combination of splitting and merging. The algorithm works significantly better in images with noise as compared to previously discussed algorithms. It is to be noted that these algorithms are sequential and can take extensive computation time. Furthermore, the choice of the initial seed has a strong impact on the final result. To remedy this issue, some techniques use pre-processing to find the best seeding spots. Another drawback of region-based segmentation is that in the splitting and merging procedures, round segments can be quite hard to trace and can come out as square or jagged.

3.2.1.1.4 Clustering Based Segmentation

One drawback observed in region-based segmentation was that the initial selection of seeds had a strong impact on the final results of the segmentation. In contrast, clustering-based segmentation methods take away that initial choice and instead approach segmentation with an unsupervised approach. The algorithm attempts to define a bunch of pixels and classifies them into regions of interest. This type of segmentation is usually done when the class labels are known before the segmentation, i.e. cell and background. Through the use of an iterative process that applies some form of inter-class similarity metric, clustering groups pixels belonging to a certain class. There are usually three types of clustering depending on similarity and thresholding. Hard, K-Means, and Fuzzy Clustering. In general, Fuzzy Clustering is the standard for images. In such segmentation, fuzzy operations and inference rules are applied while ambiguities are handled through a well-defined set of rules rather than randomness. A drawback of fuzzy clustering is that its implementation is not easy. Everything from algorithm development to application is challenging. Furthermore, clustering is iterative and can be computationally quite expensive.

3.2.1.1.5 Neural Network Based Segmentation

ANN is a representation consisting of neurons that emulates mathematical models or mappings with high precision by learning through various strategies. Consequently, they have also been used to model the relationships between every pixel and its classes in image segmentation problems. The main advantage of neural networks is that they do not rely on a certain probability density function and instead learn the mappings through a learning process. Furthermore, it can provide results if there is some variety in data without the need for expert intervention. There are primarily two steps in the neural network-based segmentation pipeline: feature extraction and image segmentation. While neural networks show great performance in segmenting images, their training times can be prohibitively long. Additionally, the initialization and dataset acquisition can be limiting factors as well. Lastly, even though neural networks can generalize on a variety of data types, there must be expert intervention to avoid overfitting.

3.2.1.1.6 Image Color Models

Every image is an array of pixels with certain values assigned to them. While most of the work done in segmentation is usually performed on RGB images, the segmentation algorithms are

not necessarily limited to the colour spaces. In this section, multiple color models are discussed along with their representation.

3.2.1.1.6.1 Image Representation using RGB

This image representation model uses three primary colours (red, green, and blue) to represent any image. A special property of RGB is that it allows the representation of any visible colour on the light spectrum through a combination of simply red, green, and blue channels. With different weights of R, G, and B, the model can represent various colours in an image.

3.2.1.1.6.2 Image Representation using CMY

The CMY image representation model uses complementary colours instead of primary colours to represent images. The colours used in this representation are cyan, magenta, and yellow. As the system uses complementary colours, the maximum values in the spectrum eventually result in black colour while the minimum values result in white color. In terms of mathematical notation, the relationship between this model and the RGB model can be stated in (1):

$$(C, M, Y) = (1, 1, 1) - (R, G, B) \tag{1}$$

3.2.1.1.6.3 Image Representation using YIQ

The motivation behind YIQ is to represent the perception used by our visual system. representation models the lightness and hue of an image by mapping the RGB model through this relationship:

$$\begin{aligned} Y &= (0.299, 0.587, 0.114) * (R, G, B) \\ I &= (0.596, -0.275, -0.321) * (R, G, B) \\ Q &= (0.212, -0.523, 0.311) * (R, G, B) \end{aligned} \tag{2}$$

In (2), the Y represents the luminance of the image while I and Q are the indicators of hue. The hue of the image is expressed with two variables to denote its weights. One of the advantages of using YIQ over other systems is that it deals with the luminance component independent of every other attribute. If your segmentation relies on luminance, it is a good idea to lean towards the YIQ model.

3.2.1.1.6.4 Image Representation using HIS

Much like the YIQ model, the HSI model is also motivated from the idea of representing human perception. Similarly, it maps light intensity and color. In the case of HSI, the light intensity I gives a general idea of the "lightness" of an image while hue H gives an idea of the color purity. An additional attribute, saturation S denotes the amount of color spread throughout the image composition. The RGB model can be mapped to HSI using (3) as such:

$$\begin{aligned} I &= (R + G + B)/3 \\ S &= 1 - \min(R, G, B)/I \\ H &= arctan'(\sqrt{3}(G, B), 2R - G - B) \end{aligned} \tag{3}$$

3.2.1.1.6.5 Image Representation using I1_I2_I3

Similar to YIQ and HSI, this model also has a focus on mapping human perception. Like YIQ, it has two variables to represent the colour information. While in YIQ, the two variables represent hue, in the case of I1_I2_I3, the variables indicate colour information. The relationship between RGB representation and I1_I2_I3 representation can be mapped by (4) as such:

$$I1 = \frac{1}{3}(R + G + B)$$
$$I2 = (R - G)$$
$$I3 = \frac{1}{2}(R + G) - B$$

(4)

3.2.1.1.6.6 Image Representation using L*a*b

This model was proposed by CIE as the standard for international colour surveys in 1931. Later on, it was rebranded as CIE L*a*b. Similar to the YIQ model, there are two focuses of this model, lightness and colour. (L) in the model denotes the lightness component while (a and b) denote the colour component. Unlike previous models, the mapping from RGB to L*a*b is not as simple. The (a) in the equation quantifies the degree from green to red while (b) highlights the degree from blue to yellow. The mapping of RGB to XYZ and then L*a*b is given as such in the following equation:

$$L* = 25100\frac{Y^{\frac{1}{3}}}{Y_0} - 16$$
$$a* = 500[\left(\frac{X}{X_0}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_0}\right)^{\frac{1}{3}}]$$
$$b* = 200[\left(\frac{Y}{Y_0}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_0}\right)^{\frac{1}{3}}]$$

(5)

However, to compute XYZ, the following relation can be applied:

$$X = (0.607, 0.174, 0.200) * (R, G, B)$$
$$Y = (0.299, 0.587, 0.114) * (R, G, B)$$
$$Z = (0.000, 0.066, 1.116) * (R, G, B)$$

(6)

3.2.1.2 Discussion: Image Segmentation Techniques and Color Models

The paper by Savita Agrawal et al. [5] aims to dive into currently present and previously used image segmentation techniques. The goal of this paper is to give a broad overview of the image processing landscape while highlighting specific advantages and disadvantages of each technique. Furthermore, their work also highlights different imaging and colour representation models that can be exploited for segmentation depending on your needs. The methods discussed within the paper cover traditional image processing techniques along with some intelligence-driven methods. While the techniques are effective in certain types of images, they fail to show effective results in a biomedical setting. The primary reason such methods have low efficacy in the biomedical context is due to mostly flat contrasts and prohibitive

computational costs. Most whole-slide images are 10,000 pixels by 10,000 pixels in resolution with sizes varying from 600 megabytes to 1 gigabyte. Running segmentation algorithms on such large whole-slide images can be exhaustive and inefficient.

Additionally, the colour models are classified into two major categories. One based on generic perception and another based on modelling human perception. While models that base images on human perception are expected to perform well, in a biomedical context, human perception is of little importance. Instead, microscopic perception is more meaningful and it can be easily mapped in simple RGB models. It would be interesting to see what other motivations there could be to develop colour models other than just to map human perception. Lastly, there is a discussion about segmentation through ANNs, however, there is no discussion about specific architectures that are used for segmentation. Considering modern neural network operations such as Convolution, Max-Pooling, and Dropout [7], there is a significant lack of information being provided in the literature. Overall, the paper is a comprehensive insight into image segmentation methods, but it lacks modern segmentation methods and constraints itself to much more archaic but classic image processing techniques.

3.2.1.3 Relevance: Image Segmentation Techniques and Color Models

Multiple techniques and algorithms have dominated the domain of image segmentation in recent years. To build our segmentation algorithms and possibly improve these techniques, it is important to have the fundamental knowledge and broad understanding of these existing solutions. This paper helps us understand the ontology of segmentation and identify how these techniques work with their placement in the overall image segmentation ecosystem. The reason for selecting a paper that focuses on archaic image segmentation techniques is to pair them with modern architectures and achieve improvements outside of the model development cycle. It has been observed that there is in fact an upper-bound to the segmentation quality that can be achieved through state-of-the-art models.

3.2.1.4 A survey on instance segmentation: State-of-the-Art

Instance segmentation refers to the process of classifying specific pixels in an image as an instance of a specific class. It is usually done as a single-class operation, meaning that if an image is given, the task is usually to highlight instances of a single class, i.e. cells from an image, houses in satellite images, etc. However, it can also be extended to multi-class instance segmentation as well.

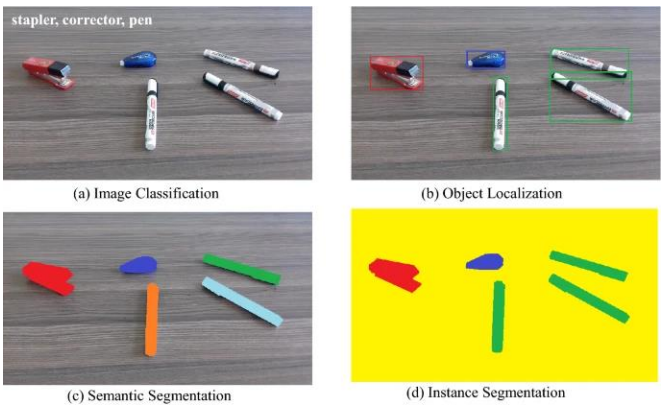

Figure 1: Evolution of Detection

This figure represents the evolution of detection where (a) represents classification, (b) represents object-detection/localization, (c)

represents segmentation, and (d) represents instance segmentation.
[8, Fig.1]

This paper by Abdul Mueed Hafiz et al. [8], is the second one in our exploration of image segmentation problem and discusses state-of-the-art techniques developed in the domain of instance segmentation. It provides a baseline for researchers to learn about instance segmentation and work things up from fundamental algorithms to state-of-the-art architectures.

Segmentation, by itself, is not an isolated research area but a continuation of pursuing fine-grain inferences on images. The progression of inference, as displayed in Fig. 1, highlights how research has continued to refine the predictions from simply detecting presence of an object to now denoting specific pixels containing the object.

3.2.1.4.1 What makes a good segmentation architecture?

In terms of issues related to segmentation, taking a look at semantic segmentation - there are two. The requirements of a "good" semantic segmentation algorithm are that it works efficiently and accurately. With efficient segmentation, low-resource machines can run inferences without too much time trade-off, i.e. real-time; with accuracy, the algorithm ensures that it can localize objects easily, i.e. better distinctiveness. To incorporate both of these attributes into an algorithm, many older strategies require fine-tuned feature engineering and extraction. However, modern deep-learning methods that use CNN have been powerful enough to learn variable-level feature maps from images. Since feature engineering is now out of the question, the problem has become more about designing efficient and high-quality feature extractors using CNN.

3.2.1.4.2 "Deep" Learning

There have been various deep learning architectures for feature extraction over the years with a variety of sizes and layers. For instance, AlexNet consisted of eight layers while recent architectures such as DenseNet and VGGNet have more than 100 layers, hence the term deep learning. Architectures such as RCNN, Fast RCNN, Faster RCNN, and YOLO have achieved high accuracy in detection, however, they suffer from a major memory bottleneck flaw due to their pyramid inference schemes. In environments where resources are constrained, it's not ideal to apply these models. To remedy this problem, various techniques such as dilated convolution and super-resolution have been proposed.

3.2.1.4.3 Handling Occlusions in Segmentation

Occlusion happens when an object overlaps with another object in the image. This results in a loss of details from object instances. Consequently, it creates issues during the training process.

Figure 2: Examples of occluded face images from MAFA dataset.
This figure represents the faces being blocked/occluded due to hands, masks, and other objects. [51]

To resolve this problem, the following techniques are applied to achieve better results.

- Deformable ROI Pooling
- Deformable Convolution

Both of these techniques primarily provide flexibility in fixed geometric structures. With the advent of GANs, there are further possibilities to interpolate or remove these occlusions from the image.

3.2.1.4.4 Handling Image Degradation

Image degradation usually occurs from lighting, quality, and noise issues. In a biomedical context, the degradation may occur due to microscopic quality or impurities. It is important to handle such degradation because ultimately it's all about data. However, it needs to be considered that contemporary techniques are evaluated in a degradation-free environment. All major datasets such as MS-COCO and ImageNet have near-perfect high-quality images. In terms of research, there is little work done on handling image degradation.

3.2.1.4.5 Instance Segmentation

Instance segmentation is a challenging problem in computer vision and image processing with applications in robotics, surveillance, and biomedical computing. The primary goal in instance segmentation tasks is to highlight all instances of certain classes. Many instance segmentation architectures have been developed with each of them having specific applications, i.e. UNet for biomedical segmentation, and Mask R-CNN for object localization. Each architecture has led to an increase in the overall accuracy of the segmentation problem. In addition to fully-convolutional neural networks, architectures such as feature pyramid networks have exploited variable levels of image features to achieve excellent results on a variety of datasets.

3.2.1.4.5.1 Classification of Mask Proposals

This technique was popularized by Hariharan et al. [9] in which they used mask proposals generated through image processing and then performed segmentation.

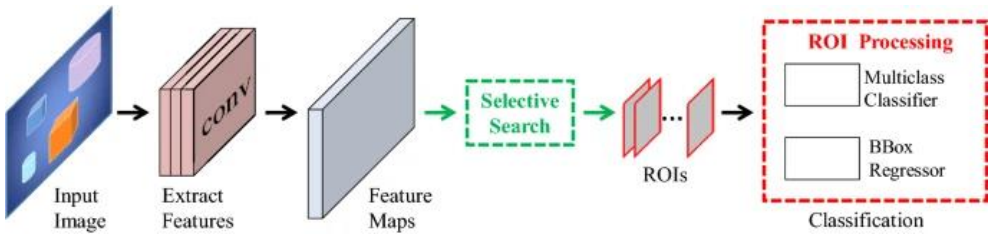

Figure 3: Classification of Mask Proposals Framework.
This figure illustrates the generic framework of traditional mask proposal frameworks. [8, Fig. 2]

However, in earlier methods, for example, by [10], the mask proposals were used to extract ROIs through an exhaustive search pipeline. This classification pipeline consisted of a classification algorithm such as. SVM, which produced considerably better results in instance segmentation.

After the introduction of deep learning, prior techniques took over the mask proposal generation pipeline. In this regard, networks like R-CNN were developed. Since these networks

had memory-hogging pipelines, later iterations, namely Fast R-CNN and Faster R-CNN were introduced that allowed for faster and memory-efficient inferences.

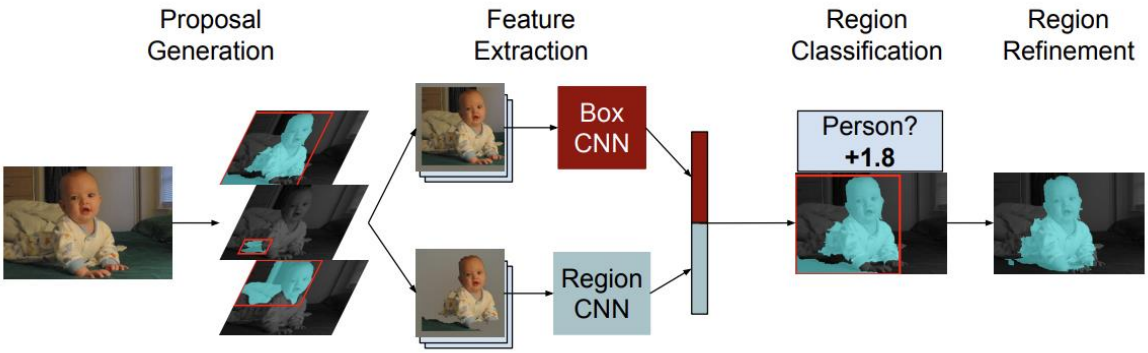

Figure 4: New CNN-based feature extraction pipeline.
This figure illustrates the newer workflow that uses CNN instead of probabilistic mask extraction through image processing. [9, Fig. 1]

In a nutshell, the idea behind the classification of mask proposals technique is that the model extracts ROIs before feeding them into your feature extraction pipeline for better results. The feature extraction pipeline learns only the most important features without only looking at the unimportant features.

3.2.1.4.6 Detection-Driven Segmentation

In the current instance segmentation landscape, a popular approach is to perform object detection on an image and feed the weighted bounded boxes into a segmentation head along with feature maps.

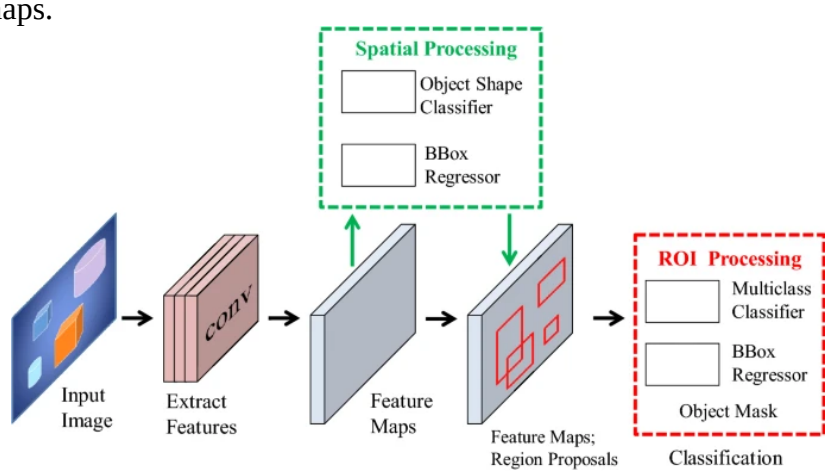

Figure 5: New CNN-based feature extraction pipeline.
This figure illustrates the detection-driven segmentation pipeline [8, Fig. 3]

The primary idea behind detection-driven segmentation is that rather than using the network simply for segmentation, this framework emphasizes first finding out the region of interest through object-detection. To assist in this regard, this framework simply adds a spatial processing and object-detection layer within the pipeline. At the end of the pipeline, all the

extracted features along with the weighted bounding boxes are forwarded to the classification and segmentation layer.

Mask R-CNN is one of the most successful techniques in detection-driven segmentation. It extends Faster R-CNN and simply swaps the ROIs module with a custom object detection configuration.

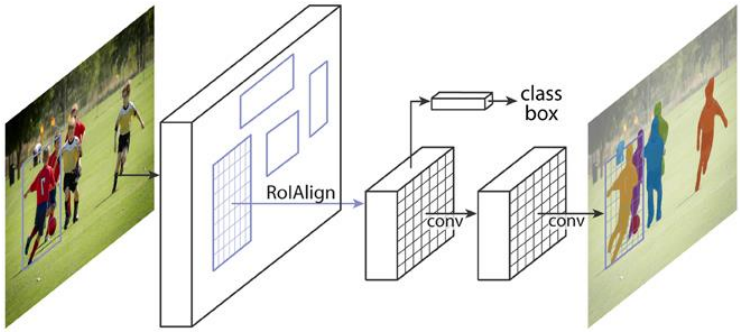

Figure 6: Mask R-CNN.
This figure illustrates detection driven segmentation in Mask R-CNN.
[52, Fig. 1]

Additionally, Mask R-CNN only imposes a small overhead over Faster R-CNN by using a simplistic mask detector while giving significantly better semantic and instance segmentation results. One thing to note in Mask R-CNN architecture is that instead of simple ROI pooling, the ROIAlign operation is being performed. ROIAlign addresses the data loss that occurs while performing ROI pooling in Faster RCNN. Through these improvements, Mask R-CNN beat the state-of-the-art by outperforming all COCO 2016 winning models.

3.2.1.4.7 Labelling Pixels and Clustering

This approach uses a combination of a fully-capable semantic segmentation network and a feature extraction backbone to harness the developments made in semantic segmentation frameworks. By using semantic segmentation information and pairing it with the feature maps, it fuses the information from both branches to produce instance segmentation masks. The accuracy is lower than detection-driven segmentation. Moreover, the computation required by pixel labelling and clustering requires more computation power.

3.2.1.4.8 Dense Sliding Window Methods

Rather than doing a single pass on the input image, dense sliding window methods perform a sliding window operation over the image and create multi-resolution inferences. These multiple masks are then evaluated using a network-specific scoring mechanism which is fed into the segmentation head. These algorithms show decent results on various datasets, however, the algorithmic complexity of these algorithms makes them prohibitively costly.

3.2.1.4.9 Evolution of Instance Segmentation

Previously, broad classification of instance segmentation networks was discussed. In this section, our focus will be more on how these networks have evolved over time with the specific improvements introduced in each state-of-the-art solution.

Table 1: Instance Segmentation Architectures Timeline
This table depicts the timeline of instance segmentation architectures. Models developed from 2014-2019 are presented here.

Year	Architecture
2014	R-CNN
2015	Fast R-CNN
2016	Multipath Network
2017	Faster R-CNN
2017	Mask R-CNN
2017	Non-local Neural Network
2018	PANet
2019	YOLOACT
2019	Mask Scoring R-CNN

3.2.1.4.9.1 RCNN

At the time of its publication, RCNN started a movement of development in the domain of computer vision which had been somewhat slow-progressing and stagnant. While the RCNN family is primarily tailored to object detection, they can easily be adapted to cater to segmentation needs as was observed earlier in detection-driven segmentation models as well. After observing the breakthroughs in classification using CNNs, Girshick et al. [54] attempted to utilize exhaustive search along with classification to solve the instance segmentation problem using RCNN. While the approach had its specific architecture head, it is important to understand the fundamental architecture of RCNN itself in a detection context. A forward pass of RCNN consists of the following steps:

- Generate ~2000 ROIs from the image provided. These ROIs can be extracted through an exhaustive search algorithm or some combination of the heuristic of randomness. In standard RCNN, an exhaustive search-driven ROI extraction is implemented.
- These extracted ROIs are warped and then forwarded to a CNN-based classifier. This classifier is responsible for categorizing the ROI into the programmed classes.

While RCNN proved to be a great starting point for detection in images, it had drawbacks when it came to performance and accuracy. One of the major drawbacks comes from the fundamental steps of RCNN – the proposals / ROI extraction.

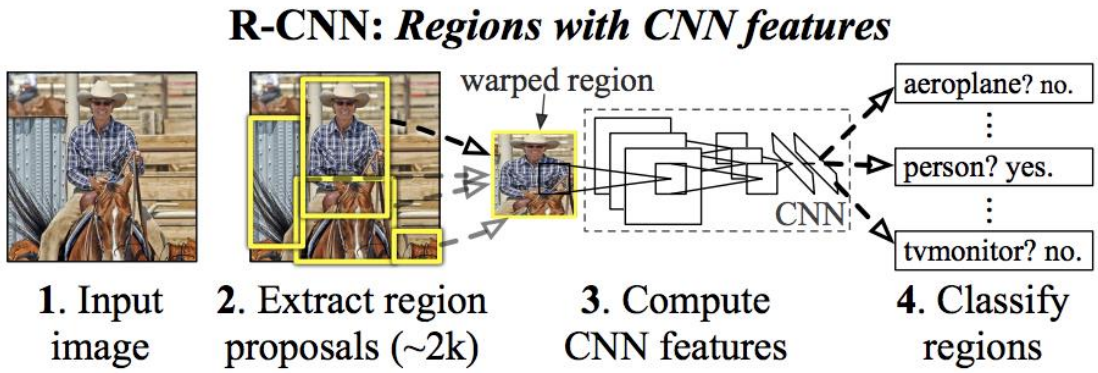

Figure 7: R-CNN Pipeline.
This figure illustrates the R-CNN pipeline. [53]

As the model extracts around 2000 patches through an exhaustive search, it can be costly to run this extraction. Furthermore, during the training phase, the patch extraction search is not trained or updated. This means that while the classification accuracy may improve through an iterative training process, the ROI extraction module has to be improved manually. The second drawback, of the 2000 patches extraction, is the very slow inference and training time of the network itself.

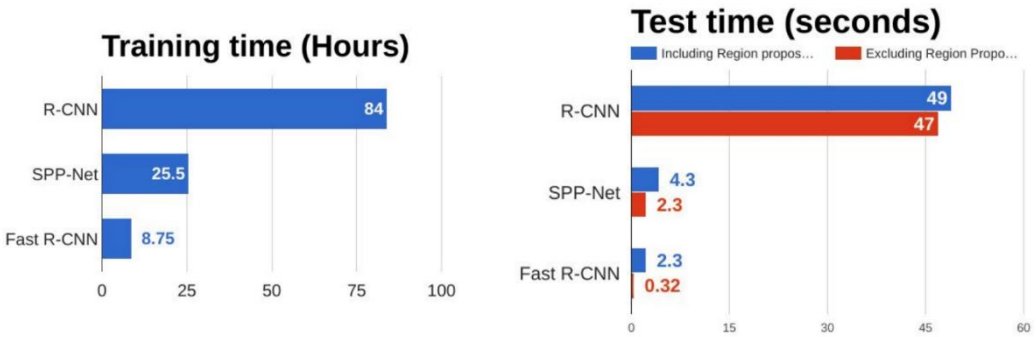

Figure 8: Time Comparison for R-CNN and Fast R-CNN.
This figure illustrates the time comparison between R-CNN and Fast R-CNN. [53]

As shown in the figure above, the average training time can be 10x more than Fast R-CNN. While the inference time can be more than 12x more than Fast R-CNN.

3.2.1.4.9.2 Fast RCNN

While R-CNN proved to be a leap forward in computer vision and detection tasks, it had obvious drawbacks in terms of performance and memory consumption. More specifically, the ROI extraction module was a substantial roadblock in making the network more efficient. In the following year after the publication of R-CNN, Fast R-CNN was introduced with a much better ROI extraction mechanism. One distinction between R-CNN and Fast R-CNN is that the latter took images as a whole rather than decomposing them into patches. This mechanism was replaced with a CNN-driven feature extraction backbone. This backbone extracts feature maps that are converted into proposals to be forwarded into the ROI pooling layer which brings all the ROIs into the same sizes. The reason for this resizing is to avoid the warping that generally

occurs in R-CNN. In the downstream layers, these feature vectors directly connect to the output where the model gives a bounding box regressor. While Fast R-CNN was vastly faster and better than R-CNN, it still faced some issues with performance and training times.

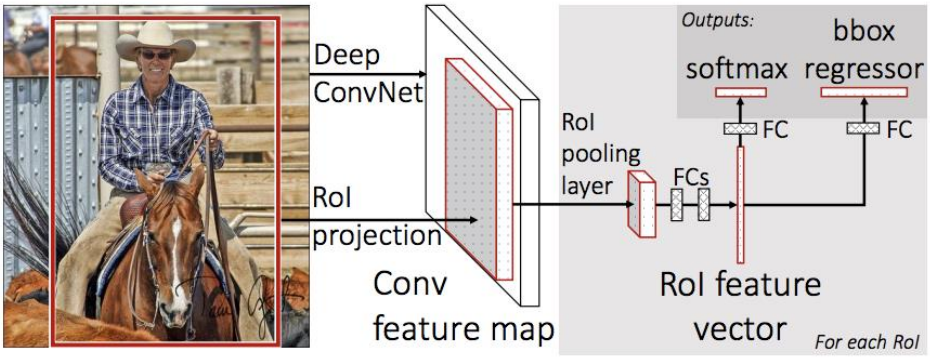

Figure 9: Fast R-CNN Pipeline.
This figure illustrates the general pipeline of the Fast R-CNN. [53]

3.2.1.4.9.3 Faster RCNN

Faster R-CNN addresses the bottlenecking operations in Fast R-CNN by introducing a new network between the ROI extraction and proposals forwarding. It terms this network the Region Proposal Network (RPN) whose primary goal is to create bounding boxes or highlight regions of interest which get passed to the classifier. The reason this is more efficient is that unlike Fast R-CNN or R-CNN, the RPN learns during the training process. It significantly improves the inference times while providing robust feature extraction that generalizes well to a variety of computer vision tasks.

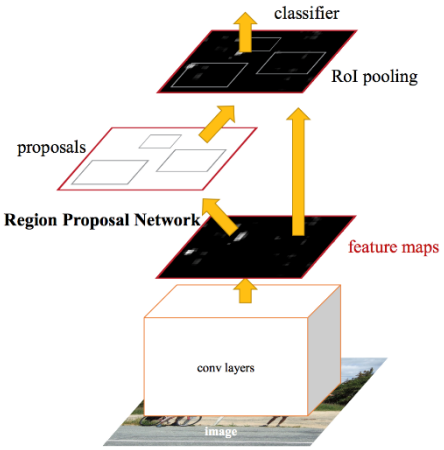

Figure 10: Faster R-CNN Pipeline.
This figure illustrates the general pipeline of the Faster R-CNN. [53]

3.2.1.4.9.4 Mask RCNN

Mask R-CNN is similar to Faster R-CNN and improves the performance of the network while focusing on the instance segmentation task. Mask R-CNN specifically addresses the loss of data while performing ROI pooling in Faster R-CNN, whether the stride is quantized or not. Instead of ROI pooling, the Mask R-CNN applies ROIAlign which outputs a pixel-to-pixel

alignment of image and segmentation masks. Through this change, the network preserves the spatial information and orientation of features without losing any data.

3.2.1.4.9.5 Single-Stage Detectors vs. Two-Stage Detector

It's important to understand the distinction between the detectors available to be plugged behind different segmentation heads. Generally, there are two types of detectors in this regard:

- Single-Stage Detectors
- Two-Stage Detectors

It all comes down to the accuracy vs. time trade-off. If the application is time critical, i.e. near real-time detection needed, then single-stage detection mechanisms such as YOLO are the ideal choice. These networks have a lighter backbone and extract a lesser number of candidate regions from the image. A drawback of single-stage detectors is the often struggle to detect smaller objects.

Meanwhile, if the application has high computational resources and a requirement for higher accuracy, then two-stage detectors provide flexible and winning models. In this regard, models like Mask-RCNN provide a strong baseline to work on.

3.2.1.5 Discussion: A survey on instance segmentation

This paper [8] provides a comprehensive insight into instance segmentation in a modern research context. Their work gives a deeper background of the instance segmentation problem itself while providing precursor knowledge to various issues faced in implementing such models. One thing that stands out about this paper is its focus on explaining the evolution of segmentation architectures and what improvements have been made in every iteration to assist in enhancing the segmentation performance. Their work starts with an introduction to the overall taxonomy of the instance segmentation ecosystem with a transition into the RCNN family of models, i.e. R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN. The discussion in the paper is restricted to the R-CNN family of models only even though segmentation architectures such as UNet have shown much higher performance against Mask R-CNN or comparable networks. The architectures proposed in this paper are outdated in a biomedical context. Furthermore, the variety in these architectures makes it difficult to evaluate them in a consistent format. It's a good paper for researchers looking to get a deeper understanding of instance segmentation and general segmentation architectures. However, strong precursor knowledge and sifting are required for some parts of the paper.

3.2.1.6 Relevance: A survey on instance segmentation

Instance segmentation is the process of highlighting all instances of a class from the image. In our project, we aim to perform cell instance segmentation on biomedical images. Before being able to dive into developing instance segmentation architectures, it is necessary to attain knowledge of issues related to instance segmentation along with the architectures that help solve them. Additionally, it is also necessary to visualize how segmentation architectures have evolved in the previous decades with insight into their reasoning and implementation. Consequently, we select this paper because it provides an amalgamation of all these characteristics in a single resource.

3.2.2 Biomedical Image Segmentation

Biomedical Image Segmentation is defined as the process used for the segmentation and detection of the boundaries of anatomical structures majorly. This segmentation can be applied to different types of two-dimensional and three-dimensional images. Three-dimensional images have various modalities such as CT, MRI, X-Rays, US, and PET. This technique is a fundamental part of diagnostics, localization of pathology, the study of anatomical structures, planning of treatment, computer-based surgery, and the quantification of tissue volumes.

3.2.2.1 Biomedical Segmentation Techniques

There are numerous segmentation techniques that have been used in the past. Due to the significance of segmentation in medical applications, different methods have been reviewed by Alzahraniet al. and Boufama et al. [11], in this paper. Different variables are considered during the analysis of these methods such as colours, patterns, orientations, and textures. This is due to the different modalities or the type of organs under examination.

Following is a discussion of the popular methods discussed in this paper.

3.2.2.1.1 Edge Methods

This method of segmentation is also known as the "Edge-Based Method". In this method, the focus is on edge detection between objects. The edges are denoted as the boundaries between the objects and the regions. Another indicator of edges would be the discontinuity in colours. This method faces challenges such as false edges, over-segmentation, and under-segmentation. Edge detection-based methods aid in identifying the border by the addition of edges into the already existing edge chain. Such methods include border detection, edge relaxation and the Hough transform method [12]. Other edge detection operators work on gradient functions. These include Canny, Laplacian, and Sobel edge detectors. The Canny edge detector is an optimization method that consists of edge training and enhancement.

There are edge detection methodologies like the one proposed by Lee et al. [13] that do not require parameter tuning. Instead, a lookup table along with a binary pattern is used to locate two groups of pixels and their threshold value, gradient magnitude and direction are obtained as a part of their average intensity. The lookup table is then searched for the ideal magnitude/direction and edge confidence measure and linking are used to detect the correct edges. Furthermore, there are multi-stage methods like the one proposed by Su et al. [14] that use histograms for segmentation. Despite the simplicity and easy implementation of this method, this does not work efficiently when encountered with noise.

3.2.2.1.2 Markov Random Field (MRF) Models

This method makes use of Undirected graphical models called MRFs, in which edges are preserved by parameter estimation. An MRF model can be created based on a probability distribution of nearby pixels which are related in some way. Two statistical physics models; Potts and Ising serve as the basis of MRFs. A Hyper-graph is used consisting of vertices and edges. In binary segmentation, Ising MRF is used where background and foreground are denoted by two values {0,1} [15]**.** The graph represents a family of distributions and the probability model relies on the pixels and their labels.

To obtain the labels, bias field, and the parameters, a three-step expectation maximization algorithm was used. MRF is fundamentally based on an image intensity density function but some findings showed it was based on the addition of structural features to the Region-based

Hidden Markov Model [16]. Hidden Markov Models are defined as an optimization issue, where the estimation methods frequently use the Maximum Posteriori and Maximum Likelihood concepts. MRF gives successful performance when used for brain MRI segmentation. This method is complex and relatively slow, and it works well for homogeneous tissue.

3.2.2.1.3 Watershed Transform Methods

This method assumes a grayscale image to be a topological one. When a water droplet touches the surface, it would form a path towards the local minima of the terrain. The catchment basin comprises the water droplets that come into contact with the local minima path [17]. Highlighted pixels of the grayscale image are following the water droplets and the catchment basin forming the image's subregion.

Watershed based on morphological and gradient operations has a chance of over-segmentation. This problem can be mitigated by using pre-processing and post-processing techniques along with filtration methods. This method is robust for segmentation due to its speed, simplicity and the capability of total subdivision of the image. Moreover, it also works for parts of the image where there is low contrast or a weak boundary.

Watershed has been used to resolve the problem of over-segmentation and is relatively resistant to noise. An algorithm based on this was used to segment MRI images of the brain and cardiac ventricle [18]. Watershed is usually implemented on gradient images. Sobel operator [19] along with automated histogram thresholding was applied for this purpose. This path was time-consuming as the post segmentation step is required.

Despite this method being effective and simple, over-segmentation and seed point selection are its drawbacks. This method works well for images that have a high contrast between the target region pixels and neighbouring pixels.

3.2.2.1.4 Graph Search-Based Methods

In this method, an image is seen as a graph with notes denoting pixels and links between nodes called n-edges. It seeks to make the best cut possible which separates the objects and the background [20]. Normalized cuts and Minimum cuts are two of the methods that are performed. The marching cubes method was used to build the graph [21]. The aim was to transform the issue of optimal surface detection into a search for a minimum s/t cut, shown in a weighted graph G(N, E), which is constructed based on mesh M. The associated cost c(Vi, K) for each node n(Vi, K) represents the possibility for a node to exist within such a surface. The number of nodes in each column m is determined by the image resolution. The total sum of the cost of all surface nodes is used to represent the overall cost.

Graph-cut methods perform well on medical images and are suitable for all kinds of image modalities. There are multiple graph-based developments in research. It is not considered complex, mostly moderate in speed and gives accurate results when shape information is available. However, a limitation is that it is computationally expensive.

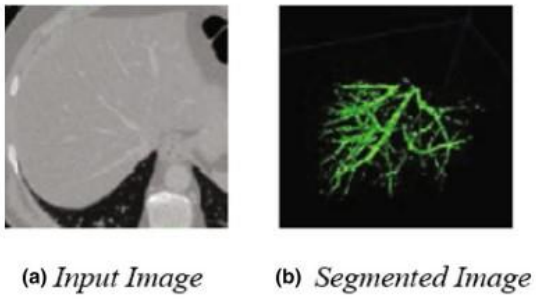

Figure 11: Graph Cuts
This figure illustrated image segmentation using Graph Cuts [11, Fig. 8]

3.2.2.1.5 Parametric Active Contour

This is one of the two types of deformable models. These models, also known as snakes, are a collection of discrete points connected by straight lines used to track moving objects [22]. However, it is unable to track several objects or objects with concavities. Snakes also need a strong initialization and cannot change topology. To improve the model, a dynamic programming algorithm was used [23]. This resulted in improved stability but with more processing time. Another improvement was suggested to the original model, which made use of accelerated convergence. In this method, the curve was depicted by cubic B-splines (B-snake). However, if the estimate is not close to the local maxima, it may continue to decrease until it is too small to be useful in areas with several edges [24].

Furthermore, many more works have been improved on this model. One of those works enhanced it by using piecewise polynomial functions. This method did lessen the complexity of the underlying algorithm, but it also increased the cost of processing [25]. Another work fixes the problem of initialization previously mentioned, through the use of splitting of the B-snake and automatic initialization using Djistkra's algorithm and the divergence of the vector field convolution. In the image, the presence of an object's contour is shown by the negative divergence values, and the separation is indicated by the positive values.

This method has some benefits such as a decrease in the iterations required for the B-snake to converge and the ability to change topology with aid of the proposed splitting method. However, this approach fails when the segmentation targets are not close to one another [26]. Overall, this method proves to be complex and slow. It is initialization dependent and it works for all modalities. It works best for three dimensional MRI.

3.2.2.1.6 Level-Set Methods

The Level Set Function is a curve reorientation used in energy minimization methods to locate curves or interfaces in an image. In this method, the zero-level set of the function indicates the contour that forms the boundary of the ROI, or the foreground and background. The evolution of the curve is controlled by a speed function. Based on the image information and curve properties, the motion can be defined (front curvature and gradient). In level-set methods, speed and high resolution of the imaging modalities is necessary.

There are many improvements made to this method in several publications. Casseles et al. created a model which could describe topology changes and used fluid dynamics for capturing front curvature. However, there are many irregularities that could occur and re-initialization is needed, causing high computational cost [50]. In this work [27], a mathematical formula has been proposed which involves producing a signed distance function as a level-set initialization

function. The re-initialization issue mentioned previously, is resolved with this technique and the computational cost is decreased. The advantage of the level-set method is that it can handle topological changes and it can work in various image modalities. It works well for three dimensional MRI images.

3.2.2.1.7 Supervised Segmentation Methods

All major classifiers belong to the category of supervised segmentation. These consist of a number of layers of artificial neurons that are linked with each other [28]. These methods include ANN, Support Vector Machine, K-Nearest Neighbor, Naive-Bayes, Random-Forest, Logistic Regression, Decision Trees, and discriminant analysis. The advantages of this method include: adaptive learning, self organization and ability to test in real time due to the parallel linkages. The challenge faced in this method is the selection of the architecture, the number of layers and the topology being used, and the size of the network.

3.2.2.1.8 Unsupervised Segmentation Methods

Methods involving unsupervised segmentation do not require training and are usually used with such algorithms that are based on clustering. The main objective of clustering is to put data into various groups based on their similarity. These algorithms include K-means, FCM, and PCA. K-means yields hard segmentation findings which means only one cluster is assigned to each instance. FCM allows soft segmentation through an instance to be associated with multiple clusters. The computation of centroids and the subclass is done adaptively and it lowers the cost function.

Another clustering method is BCFCM aims to overcome noise and bias fields by modelling them through segmentation. Chunyuan et al. [29] gave the idea of gaussian filtering in addition to improving the FCM clustering algorithm for MRI images of the brain. The filtering algorithm helped in eradicating the noise whereas the initial cluster centre was judged based on histograms. A recently improved methodology called the FCM clustering technique was proposed for segmenting image slices. It was used to enhance the clustering accuracy despite the noise.

Overall, this method may not produce efficient results on a few biomedical image segmentation tasks like CT and US due to noise. These can be used efficiently with 2D MRI images. There is no specific clustering algorithm that fits all the segmentation problems.

3.2.2.1.9 Deep Learning Methods

Deep learning models have the capability of feature extraction and segmentation. This implies that only one prediction model needs to be implemented for performing segmentation. CNN is the most used due to its ability to overcome conventional machine-learning approaches. CNN have two major drawbacks which are; the scarcity of labelled data and the problems faced during the domain adaptation.

A general CNN architecture consists of many convolution layers. Each layer; input, hidden, and output has fully weighted connected nodes that seek definite attributes from the prior layer feature map based on the kernel size. The dimensions of the feature map increase as the network delves deeper into the layers. This method is incorporated with the U-Net architecture in a Recurrent CNN for different applications in fields like medical imaging.

CNN is made adaptive to different testing images by using image-specific fine-tuning [30]. A combination of two CNNs for the detection and segmentation of ROI has proved to overcome the overfitting problem.

An advanced form of CNN called FCNs is used for image normalization whereas FCRNs require pre-processing because the input forwarded to it would impact the medical imaging data. FCRN prepares appropriate generation maps whereas FCNs yield an input proposal. These methods work well for two-dimensional images but are not adequate for three dimensional images.

Three-dimensional CNNs called DCNNs were introduced by Rongzho et al. [31] to overcome this problem specifically for MRI images. A DenseNet is divided into dense blocks and batch normalization is applied. CNN are applied to two approaches for image segmentation, these are: semantic and instance segmentation approach [32]. Deep learning provides solutions not only for computer vision problems but object detection and classification problems as well.

Deep learning offers exciting approaches to the analysis of medical images and their prospects. Different modalities like MRI, CT, US, X-Ray and microscopy images can be supported by these techniques. These include auto-encoders, multilayer perceptrons, CNNs, reinforcement learning, and RNNs. These methods are the most efficient when the datasets are large. However, the images available for training are limited for some organs and have privacy issues as well. These methods are slow although this can be overcome by parallel processing on network layers and the use of GPUs. These models are less sensitive to noise and are based on factors such as fixed threshold to generate an ROI. These methods are suitable for almost all image modalities.

3.2.2.1.10 Atlas-Guided Methods

Image segmentation is a complex task and prior knowledge is helpful in this regard. This segmentation methodology uses the previously built atlases which have reference information extracted from reference images. Anatomical details from previous segmentation of a reference image are incorporated in these atlases. Though there is a variation in performance as atlases vary depending on the problem. Atlas is a mapping technique and is also known as label propagation for the sole purpose of accurate image segmentation [33]. Atlas segmentation is based on several parameters to minimize the impact of bias in the construction of the atlas. The following formula is usually used in atlas-based segmentation:

$$\epsilon = E_i = (I_i, S_i), i = 1, \dots n \tag{7}$$

Where Ii is an image and Si is the related segmentation for each instance of i. Moreover, the segmentation function is defined as

$$S = f(I, E_1, \dots E_n) \tag{8}$$

Where S is the segmented image, and I is the input image.

Among the work done, multi-atlas segmentation is used for its ability to provide accurate structural segmentation of the brain. This method helps in adding flexibility that helps in coping with anatomical variations but has a higher computational cost.

All in all, this method is one of the highly regarded segmentation methods for medical images. These are efficient for homogenous tissues and work well for all modalities. The only limitation would be that these models require manually labelled datasets.

3.2.2.2 Discussion: Biomedical Segmentation Techniques

This paper by Alzahrani and Boufama [11] successfully explains the previous and present biomedical image segmentation techniques. This paper gives us a general overview of the status of biomedical image segmentation techniques and how they work on different modalities along with the challenges each technique faces. It is evident that while these techniques may be mentioned in a biomedical context, there is still a need to conduct a series of experiments to determine the ideal architecture for the task at hand. Moreover, many of the architectures mentioned lack generalizability. In a project like this where you are dealing with images from various modalities, you need to have architectures that can adapt to the problem.

The techniques mentioned in the paper are a fine starting point for researchers looking to develop biomedical segmentation solutions of their own. Not only are they getting a rough overview of the segmentation domain but also familiarizing themselves with the challenges that come with pathological data. Overall, the paper is a good start for someone who wants an idea about basic biomedical segmentation techniques along with the work done on various modalities.

3.2.2.3 Relevance: Biomedical Segmentation Techniques

Since the focus of this paper is on the currently known biomedical segmentation techniques, the detailed discussion about these techniques goes to tell that there is no ideal method that works for all kinds of modalities and different kinds of organs. Some methods work well for one set of modalities and organs but have poor results for another set. So to come up with our algorithm for the segmentation of cells, we need a basic idea of the segmentation techniques that can be applied to medical images. These existing solutions will help us choose a segmentation method that works well for our problem.

3.2.2.4 U-Net: Convolutional Networks for Biomedical Image Segmentation

It is an assumption that the successful training of deep networks requires a huge number of annotated samples for training. U-Net is a network architecture that relies on the use of data augmentation of the existing annotated samples. This architecture has a contracting path that captures context and a symmetric expanding path that allows precise localization. This paper outperforms the sliding window convolution network and helped the authors win the ISBI challenge.

Typical usage of convolutional networks is on classification tasks where the output is a single label. But in many visual, biomedical image processing tasks, the desired output is localization. This means a class label is assigned to each pixel. Moreover, in most biomedical tasks obtaining large datasets is beyond reach. This architecture aims at the building of a “fully convolutional network” [34] that trains on few images and has precise segmentations by successive layers and where pooling operators are replaced by unsampling operators. The resolution of the output is increased due to the additional layers. For localization, unsampled output is combined with high-resolution features from the contracting path.

Moreover, a large number of feature channels are incorporated in the unsampling. This allows the network to propagate the contextual information to the high-resolution layers. This forms a U-shaped structure where the expansive path is almost symmetric to the contracting path.

This architecture does not have fully connected layers and uses the valid part of each of the convolutions, that is, the segmentation map only contains the pixels, and has a full context available in the input image. This allows accurate segmentation by the overlap tile strategy. The missing context in the borders is extrapolated by mirroring input images.

For overlapping issues of objects of the same class, this method proposes the weighted loss. This method assigns the separating background labels between the touching cells, a large weight in the loss function.

3.2.2.4.1 Architecture of U-Net

The architecture of U-Net is illustrated in Fig. 12. It has an expansive side towards the right and a contracting side on the left. The contracting side has a typical convolutional network architecture. It has repeated applications of two 3x3 convolutions (unpadded), each of them is followed by Rectified Linear Unit and 2x2 max pooling operation with stride 2 for downsampling. At each downsampling part, the number of feature channels is doubled. On the other hand, the expansive side consists of upsampling of the feature map, followed by a 2x2 convolution which halves the number of feature channels, addition with the corresponding cropped feature map from the contracting path and two 3x3 convolutions each of which is followed by a Rectified Linear Unit. The cropped part is necessary due to the loss of the border pixels in each convolution. The last, final layer has a 1x1 convolution to map each 64-component feature vector to the desired number of classes. All in all, there are 23 convolution layers.

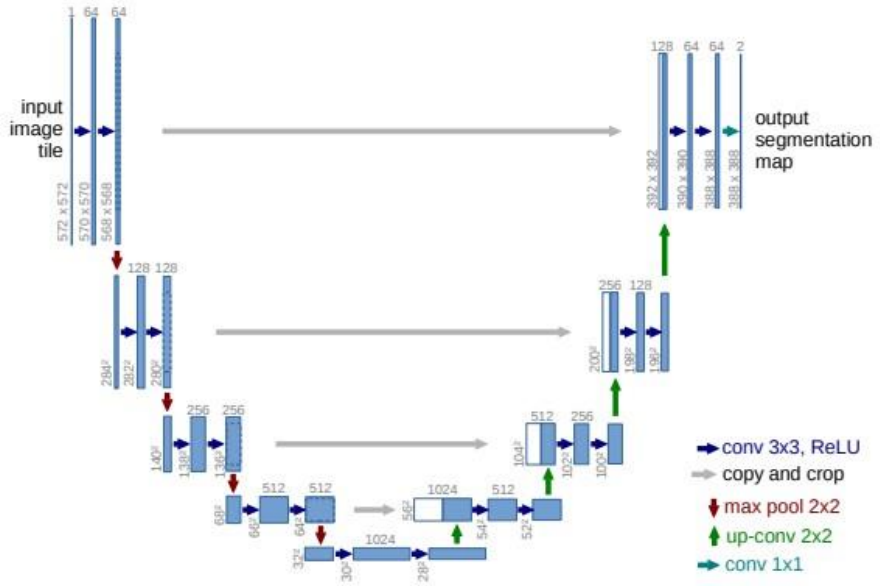

Figure 12: U-Net Architecture
[35, Fig. 1]

3.2.2.4.2 Training Overview

Stochastic Gradient Descent is used for training the network through the input images and their corresponding segmentation maps. Since unpadded convolutions are used the output image is smaller in size than the input by a constant border width. The energy function is computed by pixel-wise-soft-max (activation function) on the final feature map along with the cross-entropy loss function. Initial weights should be assigned to get uniform activations. Ideally, such

weights should be assigned so that each feature map has unit variance. In this architecture, this is achieved by Gaussian distribution.

3.2.2.5 Discussion: U-Net: Convolutional Networks for Biomedical Image Segmentation

This paper by Olaf Ronneberger et al. [35] is a classic in biomedical image segmentation. The model was the first to utilize skip connections in segmentation problems and addresses the fact that biomedical data is scarce. It relies on deep augmentations to fulfill its generalization needs. The U-Net is a benchmark in biomedical image segmentation and modern research more or less derives from this work. While the U-Net has excellent performance on segmentation tasks where objects may not have much occlusion or overlap, it struggles to find contours and edges when objects have strong occlusions or overlap. In the case of instance segmentation, the model has a fair number of issues in determining the edges of the object. Consequently, better versions of the U-Net have been introduced since then and continue to come out as research progresses. Regardless, it still forms the foundation of numerous segmentation networks in existence today.

3.2.2.6 Relevance: U-Net: Convolutional Networks for Biomedical Image Segmentation

Since the focus of U-Net is on working with less training data, it is an appropriate architecture for our problem. Since our dataset is based on 1000 labeled and 1500 unlabeled images, U-Net provides a solution for the training of this limited dataset. Moreover, U-Net provides us with an effective solution for applying data augmentation to the existing images. Multiple variations of the existing images will help in more precise training of the data. This model also provides a solution for the overlapping problem by the separation of touching objects. This is a complication that needs to be eliminated, U-Net provides a solution for this.

There are certain limitations for U-Net that can be overcome by architectures like U-Net++. These include the addition of convolution layers on the skip pathways that are used for bridging the semantic gap between the encoder and decoder feature maps. Furthermore, having dense skip connections on these pathways, helps in improved gradient flows.

3.2.2.7 Importance of Skip Connections in Biomedical Image Segmentation

Skip connections have been the backbone of recent developments in biomedical imaging and segmentation. They form the fundamental operating structures of networks such as U-Net. This paper by Michal Drozdzal et al. [36] explores the effects of skip connections in model architectures. Furthermore, they conduct deep experiments into model weight updates and changes throughout the training using a variety of skip connection configurations.

3.2.2.7.1 Long Skip-Connections vs. Short Skip-Connections

When skip connections are being used in neural networks, there are multiple ways of conquering them. In architectures where downsampling and upsampling are employed, every connection made from a downsampling layer to an upsampling layer is considered to be a long skip connection since it generally sends features from the contraction path to the expansion path. On the contrary, deep architectures where we employ skip connections to create alternate paths to downstream layers are called short skip connections. There are advantages to both approaches as will be discussed below.

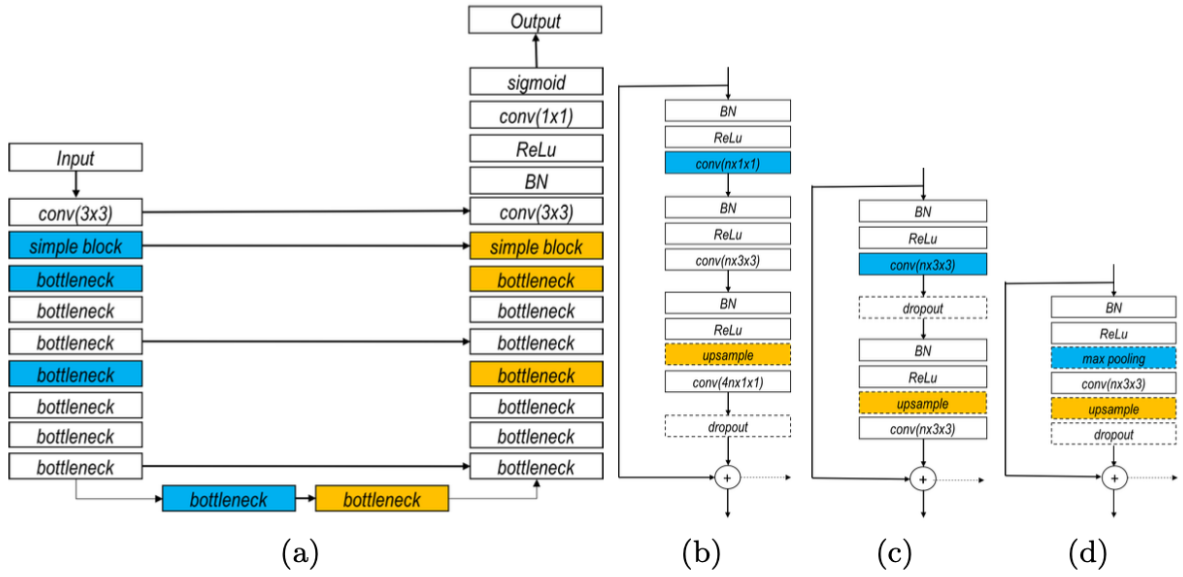

Figure 13: Comparison of Long Skip Connections vs Short Skip Connections.
This figure illustrates long skip connections in (a) and short skip connections being used in different scenarios in (b), (c), and (d). The connection is highlighted by an arrow going from one layer to another. [36, Fig. 1]

3.2.2.7.2 Feature Forwarding and Preservation

In literature, skip connections have been cited as a crucial element to preserve higher-level details when the network goes from serial downsampling to upsampling. Most modern architectures follow this encoder-decoder approach which applies a series of convolutions to an image until it reaches a specified bottleneck. As the network up-samples the image, the features are forwarded to the expanding path through the use of skip connections. There are two major strategies for handling the merging of skip connections.

- Addition
- Concatenation with Non-Linearity

Simply put, in the addition schemes the features are added to the output layer while in a concatenation scheme, some form of non-linearly is introduced to the features and they are concatenated with the corresponding output layer. Due to their feature preservation property, these skip connections are a major part of semantic segmentation and instance segmentation architectures.

3.2.2.7.3 Faster Convergence

Skip connections generally lead to faster convergence during training due to their expedition of the learning process. Deeper architectures have been shown to improve performance in segmentation tasks. However, as the architectures get deeper, there are issues of vanishing and exploding gradients associated as well. This can lead to the model being completely untrainable in some cases. To avoid this situation, skip connections can be utilized. This comes with its caveats as well. As observed through experimentation, shallow networks with skip connections

tend to perform worse than shallow networks without skip connections. It is important to determine whether your network needs to skip connections or not.

3.2.2.7.4 Long vs. Short

Unsurprisingly, using both long and short skip connections within a network improves its performance in terms of learning and convergence.

Table 2: Model Loss
This table shows the best loss in networks while utilizing all techniques and training them for 600 epochs.

Method	Training Loss	Validation Loss
Long & Short	0.163	0.162
Short	0.188	0.202
Long	0.205	0.188

3.2.2.7.5 Training Stabilization

It is evident from experimentation and monitoring neuron weights that parameter updates are much more stable and well-distributed when short skip connections are present in the network. While without short skip connections, the network weights tend to update only at the initial and final end of layers.

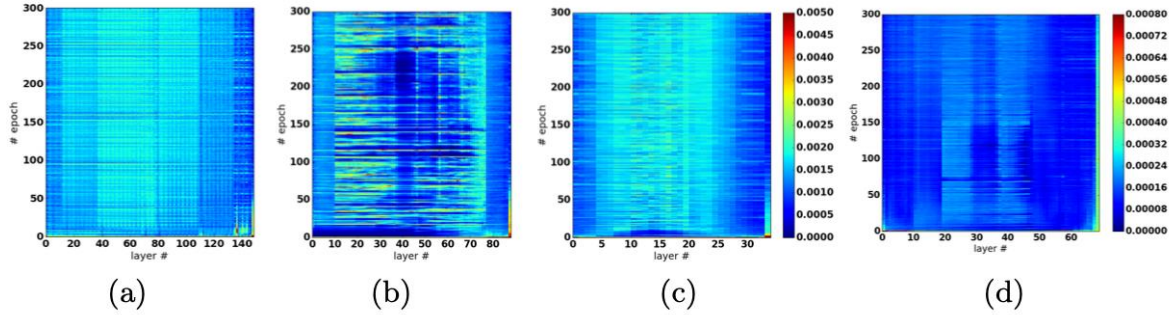

Figure 14: Neuron Weight Visualization.
This figure illustrates the network weight updates. [36, Fig. 4]

As shown in the figure above (a) which has both long and short skip connections experiences uniform updating of weights. (c) with only long skip connections and shorter repetition counts also obtains similar results. However (b) and (d) experience uneven weight updating and the learning is not completely propagated throughout the network. The reason behind this uneven weight updating is the due to higher repetitions in long skip connections (b) and failure to utilize batch normalization (d). While long and short skip connections are great techniques to enhance your network's learning, it is also important to involve low repetitions and classic techniques such as batch normalization to aid in the learning process further.

3.2.2.8 Discussion: Importance of Skip Connections in Biomedical Image Segmentation

This paper by Michal Drozdzal et al. [36] provides a thorough insight into the importance of skip connections along with their classification, i.e. long skip connections and short skip

connections. They discuss the applications of specific connections along with their need in modern convolutional neural networks. Furthermore, the arguments are experimentally backed with an emphasis on practicality rather than theory.

Although the paper solely focuses on skip connections, there are still doubts about the inner detail of the architectures used in the paper. It could very well be possible that the architectures implemented by the reader may not be able to take full advantage of this technique. Experimentation has been performed as a comparison to other models. However, none of the experiments has been done on standard biomedical or non-biomedical datasets such as PASCAL VOC or MSCOCO to benchmark the performance of the skip connections. Additionally, there is a mention of post-processing to improve their predictions but no specific details or references have been provided to the post-processing itself.

While this paper is a good resource for intermediate researchers looking to improve their knowledge about the inner workings of the skip connections, it is not an ideal starting point for someone looking to improve their models immediately. A significant amount of precursor knowledge is necessary for reading and major extrapolation is needed to identify the meaning behind the author's decisions.

3.2.2.9 Relevance: Importance of Skip Connections in Biomedical Image Segmentation

In recent research, skip connections have been a core part of biomedical image segmentation architectures. While many papers do go over the importance of skip connections briefly, it is important for us to understand the deeper meaning behind the technique and how it can be used effectively while building our own segmentation architectures. Consequently, this paper is an essential component of our project’s literature review.

3.2.3 Weakly-Supervised Learning

Weak-supervision or weakly-supervised learning is a collection of techniques that are utilized to accommodate for costlier and much finer-grain annotations. An example of weak-supervision in cell segmentation would be to use object-detection dataset for cell localization while using limited pixel-level annotations to train a segmentation head. Weakly-supervised learning techniques allow for modeling highly accurate models without having access to costly datasets or annotations. In this section, various weakly-supervised learning methods and techniques are explored that shall assist in development of the project.

3.2.3.1 Instance Segmentation with Image-Level Supervision

Instance Segmentation in medical imaging has advanced a lot recently. For this method of segmentation, a large amount of training data with per-pixel labels or labels which differentiate between object categories and instances in the image is needed. However, obtaining this amount of data can be very expensive and time-consuming so there is only a small collection of datasets that can be used for segmentation purposes. To solve the problem of requiring per-pixel labels, some weakly supervised methods are proposed. Instead of per-pixel labels, these methods would make use of weaker labels such as bounding boxes, scribbles, and image-level annotations [37]. This makes the process relatively scalable as obtaining these datasets is easier.

This publication presented a method of Weakly-supervised Instance Segmentation (WISE) which was based upon Peak Response Maps and using Mask R-CNN. They achieve effective

supervision using Mask R-CNN as a fully supervised method, trained on pseudo masks which are relatively easily obtained as compared to per-pixel labels.

3.2.3.1.1 Image-level labels as weak supervision

As compared to per-pixel labels, it is cheaper to annotate images using image-level labels. In this method, only whether a specific object class appears or not is shown. This method is mostly used for semantic segmentation. Instance segmentation is a more difficult process so there has not been much research done for it using this annotation method. Though, recent publications worked on this problem, and what they did was make use of Class Activation Map. A heatmap is identified which depicts the areas where the locations of various objects are located. Moreover, it distinguishes peaks of that heatmap that have shown the locations of objects. Using an object proposal method which generates a proposal and a post processing step, each located object is matched with that generated proposal which is considered as the instance segmentation output [38].

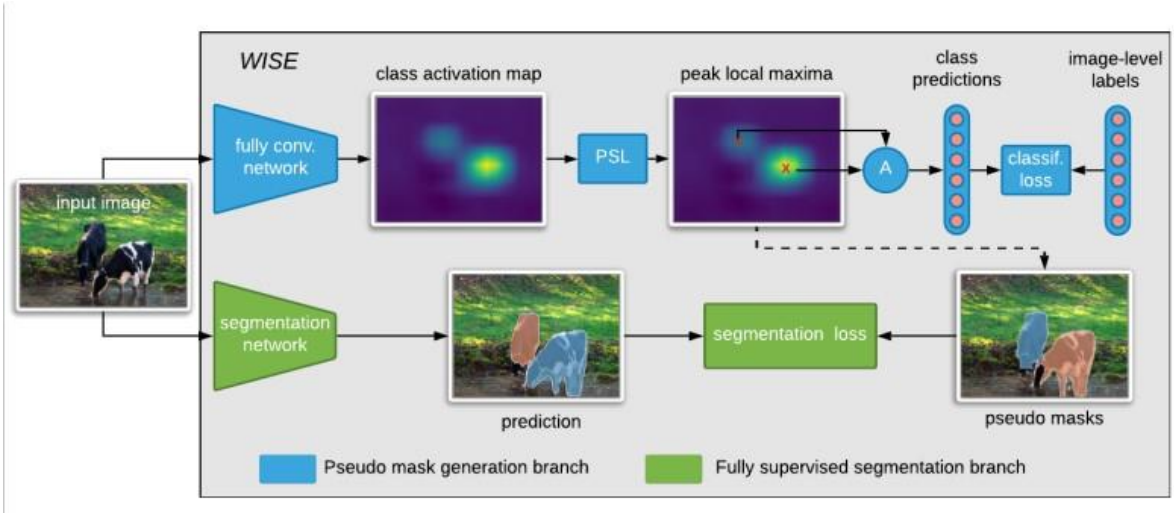

Figure 15: WISE Training
This figure illustrates the process of WISE [39, Fig. 2]

3.2.3.1.2 Learning with pseudo labels

In their method, they generated pseudo labels and trained on them in a fully supervised manner. However, this method is unable to differentiate between object instances so they cannot be directly applied to instance segmentation. Object proposals, which are class-agnostic methods outputting object candidates per image, are used to help with segmentation. The pseudo masks are generated using object proposals, similar to methods PRM and PRM+Density.

3.2.3.1.3 Pseudo Mask Generation Branch

In this process, PRM identifies the important parts of the objects in the image by creating segmentation seeds, which also create the pseudo masks to be used as supervision for Mask R-CNN. The authors trained a CAM-based classifier with the components; fully convolutional network (FCN) and a peak stimulation layer. The FCN results in a class activation map and the output for PSL indicates the locations in the map for the object class as shown in Fig. 15. [55]

Next, pseudo masks are generated for the images through the use of trained classifiers and the object proposal method. The proposal masks are put in place of the peaks which were obtained through PSL. A proposal is then randomly selected based on the objectness score. A proposal

with higher objectness has more probability of being selected. A set of proposals whose masks interact with each other are generated to get a mask for an object located at a specific peak. The reason we do this method is that there are common pixels of the important parts of the located object. We obtain object class label information from CAM and specify it for proposals which can then be trained as pseudo masks for a fully supervised instance segmentation method.

3.2.3.1.4 Fully Supervised Segmentation Branch

Pseudo masks can be used to form segmentation labels. These are then trained in the Mask R-CNN. According to the type of segmentation that needs to be done, different methods are used. For instance segmentation, YOLACT can be used and for semantic segmentation, DeepLab segmentation network can be used. Other than this, Mask R-CNN is simply used to predict the object masks for a test image. A refinement process; the object proposal method, used to create better object masks.

Figure 16: Inference
This figure illustrates the refinement of masks [39, Fig. 3]

3.2.3.2 Discussion: Instance Segmentation with Image-Level Supervision

This paper by Issam H. Laradji [39] presents a weakly supervised instance segmentation method. Their aim was to overcome the issue of the expensive cost of obtaining a large number of labeled images. They presented the problem in a structured way and explained the algorithm they used. Although their method "WISE" seems to be up to the mark to compete against other stronger supervised methods or methods with better labeling, it does not perform well in some cases.

Comparison with Mask R-CNN trained on pixel-level labels showed that their method still had many improvements that could be made. Their method did not perform well with images which consisted of small objects and a large number of those objects. In our project, we aim to segment cells which are in many different shapes and sizes. For the small sized cells and a large number of those cells in an image, WISE would not be an effective solution for our problem. Mask R-CNN, on the other hand, when trained on per-pixel labels, shows a good performance with larger numbers of small objects.

Overall, the paper made a good attempt at solving the issue of the high cost of obtaining large datasets of labeled images. Using the image-level labeled images which are less costly to obtain, they implemented a method making use of Mask R-CNN which worked better compared to most methods.

3.2.3.3 Relevance: Instance Segmentation with Image-Level Supervision

There has been a lack of research in weak supervision in instance segmentation which has proved to be a very challenging task. The main part of our project is to perform segmentation on cells through the use of weakly supervised methods. A limited dataset is provided to us where there are about 1000 per-pixel labeled images and 1500 unlabeled images. It is necessary

for us to go through different methods in weak supervision so we can apply those methods to improve our segmentation model. Therefore, this paper helps us understand some ways we can do that.

3.2.4 Cell Representation

Biological images of cells are diverse in nature due to the various microscopy techniques available, the different kinds of cells, the tissue types and the types of labelling. This diversity is the greatest challenge for segmentation methods as these are not made for generalized datasets but apply to specific applications. This problem requires an adaptable and accurate segmentation model that works for all types of microscopy images. Some models allow users to annotate their data and train their models on the same annotations [40] and other models use the "human-in-the-loop" approach to label a small amount of data initially to train an imperfect model. Then this imperfect model is applied to other images and the user then corrects the results [41].

3.2.4.1 Cellpose -- deep learning based, generic cell segmentation

Cellpose is defined as a deep-learning network that is used for instance segmentation of cells. This cellular segmentation method provides good results for images of different modalities. Many segmentation models do not allow the adaptation of segmentation styles to specific needs thus producing substandard results. However, Cellpose is a generalized, pre-trained model that offers segmentation for a broad category of cells, nuclei and tissue with little to no preprocessing requirements or additional training.

Other than discussing the working of Cellpose, this paper also discusses the open-source image analysis program of Cellpose that can be incorporated into the segmentation workflow.

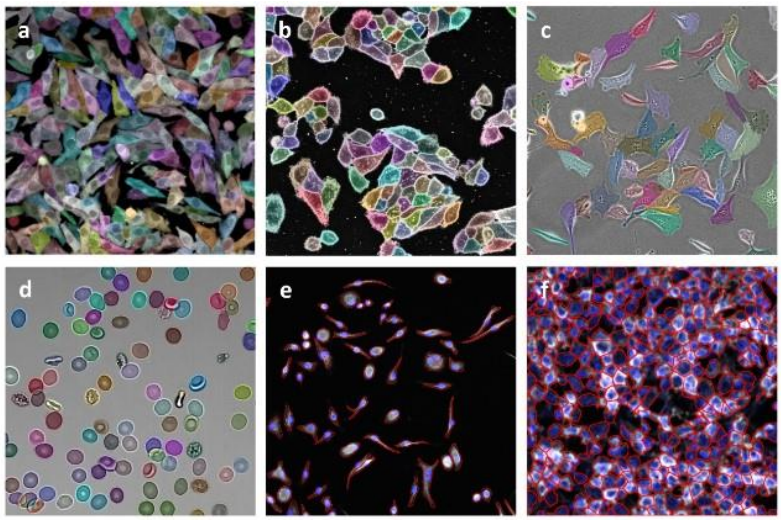

Figure 17: Cellpose
Cellpose performs robust segmentation on different microscopy images of cells [43, Fig. 1]

Cellpose attains effective results by training the images with their ground-truth labels. These labels are divided into three categorical images: the vertical and horizontal flow fields, and the binary cell mask. The Cellpose network, which is a modified U-Net, is then trained to predict the images. Post-processing the network output images into a combined gradient vector field assists in achieving instance segmentation. After the vector field is obtained, gradient tracking

calculations are performed and a label map is obtained. This method can predict complicated shapes of the cells as seen in Fig. 17, whereas methods like StarDist [42] are limited to star-convex objects that have minimal protrusions.

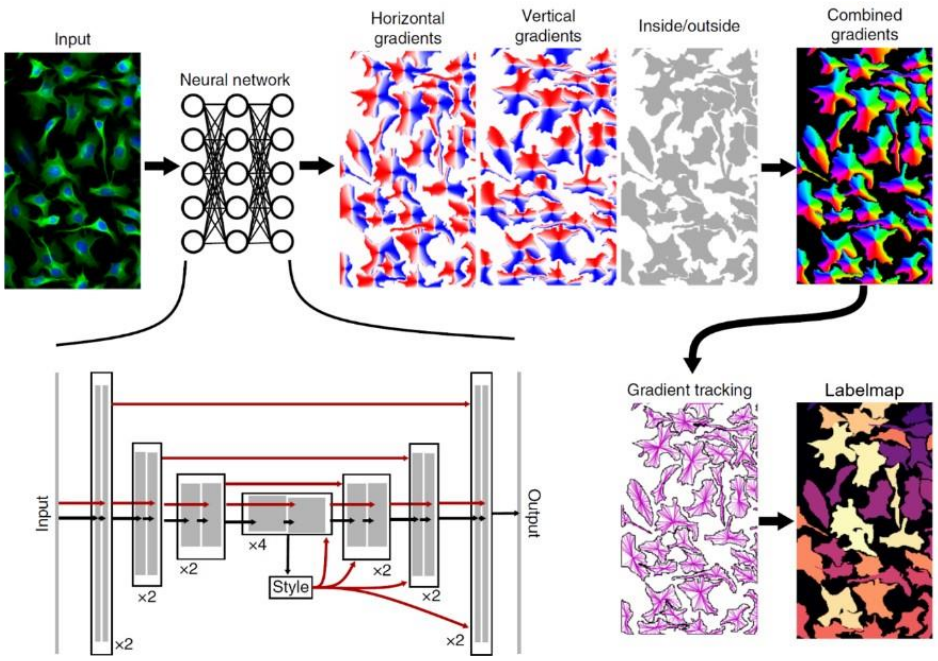

Figure 18: Cellpose network and post-processing steps
[43, Fig. 2]

Cellpose comes with pre-trained models that have been applied to a diverse set of images of cells with different modalities, morphologies, and staining. There are other variations of images in the training dataset like seashells and rocks, but this does not harm the model predicting capacity. This model works well for cells with only a few parameters to tweak. Cellpose works for both two-dimensional and three-dimensional images.

There are certain limitations to this method. Two-dimensional images containing cells with low convexity along with images with densely packed, overlapping cells are difficult to segment. DIC images are not accurately segmented due to the low contrast whereas phase contrast provides better results. Lastly, since Cellpose can segment both the nuclei and the cell bodies independently, the label values in their respective label maps do not correspond.

3.2.4.2 Discussion: Cellpose – deep learning based, generic cell segmentation

This work by Bram van den Broek [43] effectively explains Cellpose. The paper also provides an overview of the Cellpose network, the installation and running of the Cellpose software, its ability to segment two and three-dimensional images as well as limitations. The document is written in a systemized manner and proves to be highly useful for experts in this field or people who want a straightforward explanation of how the software works while also being beginner-friendly.

A weakness of this technique is that much of the internal mechanism of Cellpose is rather obscure with little to no explainability. Rather it only provides a sole image for the purpose of showing the network and the postprocessing steps but no description of it. More information is provided for the working of the software rather than the understanding of the model. For

someone who has little knowledge of U-Net, gradient mapping, and labelmaps, understanding the working of this model would be a difficult task.

Overall, this paper is a great intermediate paper for biomedical researchers or medical professionals to get a gist of what batteries are present in the ecosystem in terms of cell segmentation.

3.2.4.3 Relevance: Cellpose – deep learning based, generic cell segmentation

Since a major part of our problem is the segmentation of various modalities of cells, cell representation methods like Cellpose are necessary. Our dataset consists of phase contrast, DIC, fluorescent, and bright-field microscopy images. Cellpose's training data consists of the aforementioned images as well. This makes the cell segmentation process relatively robust. Albeit, there are certain limitations to this method. First is the problem of overlapping cells. The images present in our datasets have multiple instances of overlapping cells which is a problem Cellpose can not deal with effectively. Moreover, the presence of a significant number of DIC images may result in inaccurate segmentations due to contrast limitations. However, this method will work well for the remaining modalities. Keeping these restraints in mind, we look for other cell representation methods.

3.2.4.4 Omnipose – a high-precision morphology-independent solution for bacterial cell segmentation

Omnipose is an algorithm that precisely segments samples of cells of diverse morphology, antibiotic-treated cells, and bacterial cultures. Omnipose is an algorithm that is based on its predecessor, Cellpose. This method also uses distinct neural network outputs such as the gradients of the distance fields. Omnipose is an algorithm that aims to perform unbiased identification of cells despite their optical characteristics and morphology.

Defining cell boundaries within micrographs is known as cell segmentation and is an important part of the image analysis pipelines [44]. There is a diverse morphology that can be seen through these images. Shapes of the bacterial cells include rods or spheres [45], long filamentous and branched hyphal structures, and other variations.

Existing research on cell segmentation has a scope further than microbiological research and finds many solutions in image analysis programs. The majority of these solutions are image processing techniques such as using image intensity threshold. This method is only effective for isolated cells and does not perform well for densely packed cells. SupperSegger [46] was developed to address the issues specifically for phase contrast images that used filtering techniques to correct errors caused by watershed and threshold. Like Cellpose, StarDist, and MiSiC, Omnipose is also based on U-Net architecture with two residual blocks per scale each having two convolution layers [44]. Furthermore, Omnipose has a dropout layer before the densely connected layer. Like Cellpose outperforms Mask R-CNN and StarDist, Omnipose significantly outperforms all prior algorithms.

Since there are no existing means of defining cell centres for irregular cells, Omnipose is a technique that works on distance fields. This helps in faster convergence and better stability. Moreover, distance fields are independent of topology and morphology and can be applied to all cells. The resulting flow field points move from cell boundaries towards the local centre, coinciding with the skeleton that is defined by the stationary point of the distance file. This helps in overcoming the over-segmentation problem.

A limitation of the aforementioned method would be that distance fields are sensitive to boundary pixelations causing them to extend deep into the cell, changing the overall parameter of the cells.

3.2.4.5 Discussion: Omnipose – a high-precision morphology-independent solution for bacterial cell segmentation

This paper by [44] provides a comprehensive overview of the cell segmentation algorithms and explains how Omnipose is built from the improvements on Cellpose. It explains how Omnipose is more robust than Cellpose, MiSic, SupperSegger, StarDist and Mask R-CNN. There is an in-depth discussion about the results of all these algorithms. Omnipose is oddly high performant in generalized scenarios while under-performing, but still state-of-the-art, on specific scenarios. The model beats any other cell-representation technique published at the moment and provides a general approach towards cell segmentation without the need to engineer specific post-processing routines. While there is little evidence of omnipose working on images associated with our modality, it is a strong and robust model that can be used as a baseline for all general purpose or biomedical segmentation tasks.

3.2.4.6 Relevance: Omnipose – a high-precision morphology-independent solution for bacterial cell segmentation

To date, there is no perfect solution for the segmentation of cells of different shapes and sizes in a generalizable manner. But Omnipose provides us with an almost-general solution that is better than all previous methods. This is essential for our problem since we have different morphologies of cells that include round, elliptical, irregular, spherical, rod-shaped, twisted etc. Omnipose proposes to be a promising solution for this.

A limitation of this method is that, on the low end, cells need to be at least three pixels wide in all dimensions. On the high end, 60 pixels appear to work well while 150 pixels is too large. The solution to this is to downscale the images so that the cells are in the acceptable size range [47]. Despite this solution, our dataset comprises cell sizes that are larger than the proposed size range.

Moreover, since we have multimodality images, a generic set of post-processing techniques will not work effectively for each type of image. Omnipose does not raise this problem. All in all, Omnipose gives us a strong baseline to work on and scale in the future.

3.2.4.7 A General Deep Learning Framework for Neuron Instance Segmentation Based on Efficient U-Net and Morphological Post-Processing

Deep learning in medical imaging has shown superior performance in biomedical applications, especially cell segmentation which is one of the fundamental steps of biological studies. However, training such networks requires vast amounts of unbiased data with high-quality annotations which can be expensive to procure. This paper by Huaqian Wu et al. [48] attempts to utilize a combination of weak supervision along with morphological post-processing techniques to segment NeuN stained neuronal cells on histological images from WSI. It uses an EfficientNet encoder and achieves state-of-the-art results in the segmentation problem. This research is important not just for the robustness of segmentation algorithms but also in the context of neurodegenerative diseases.

Neuronal segmentation is challenging due to the lack of high-quality annotations as well as the high variance between the size and shapes of the neurons themselves. Automatic segmentation

algorithms such as graph-cut, water-shedding, and thresholding can be utilized to help with annotation deficits, but still, they are vulnerable to noise and variety. Furthermore, their algorithms have to be adapted to specific modalities and shapes to avoid under-segmentation or over-segmentation. Additionally, these algorithms do not accommodate occlusions and overlapping objects well enough to be considered feasible.

Although deep learning architectures can accommodate for variance in input data, they are limited by scarce datasets and few general post-processing techniques available in previous research. This paper approaches the neuron instance segmentation problem with a weak-supervision constraint and a focus on post-processing to refine results.

3.2.4.7.1 Pixel-Level Mask Synthesis Pipeline

As discussed earlier, the prohibitive cost of manual annotations makes it difficult to create and train general models for segmentation. To address this issue, first we approach this as a semantic segmentation task, i.e. 1 = neuron interior, 2 = neuron contours, and 0 = background. However, to train such a model, pixel-level annotations are needed. A pipeline is created that utilizes point annotations manually provided in neuron images. The point is a disk shape that is 5 pixels in diameter. Moreover, the contour width is kept at 4 pixels to avoid colliding objects. From this point onwards, a combination of region-growing segmentation algorithms and random forests classifier is applied to develop synthetic pixel-level annotations. These annotations can then be used to further train the models and improve performance. This pipeline is generic and can be adapted to other segmentation tasks easily.

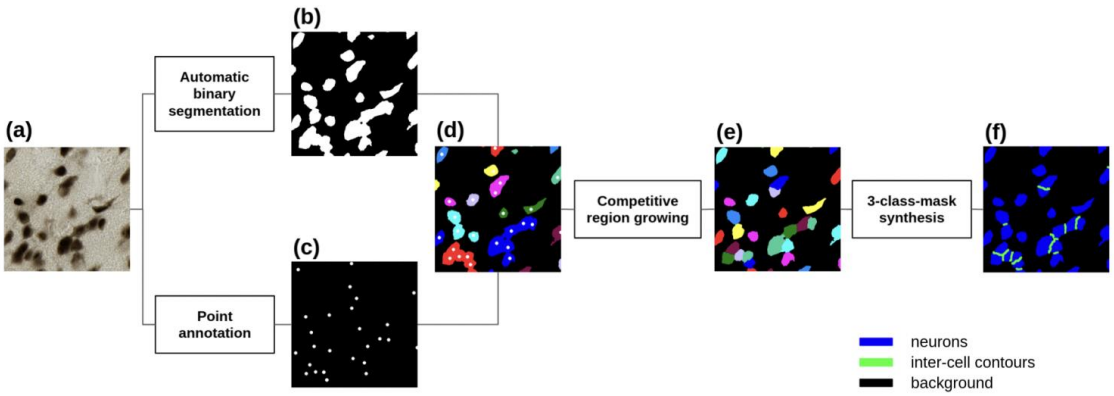

Figure 19: Pixel-Level Mask Synthesis Pipeline
This figure represents the general pipeline of creating pseudo pixel-level masks for neuronal segmentation. [48, Fig. 2]

As noted in the figure above, the point annotations, which are much less costly than pixel-level annotations are provided in (c). A random forest classifier is used side-by-side to generate binary masks (b) for the neuronal slide image. Using a competitive region-growing algorithm for clustering and segmentation, the model generates a pseudo-pixel-level mask that is ready to be fed into training.

3.2.4.7.2 Model Selection

When it comes to biomedical image segmentation, UNet-based variants have dominated the domain since their inception. Consequently, we choose a UNet-like architecture for neuron instance segmentation albeit with an EfficientNet backbone. To accommodate for the larger

parameter space of the problem, EfficientNet-B5 was chosen. Fig. (21) shows the structure of the proposed encoder while Fig, 22 shows the structure of our complete network.

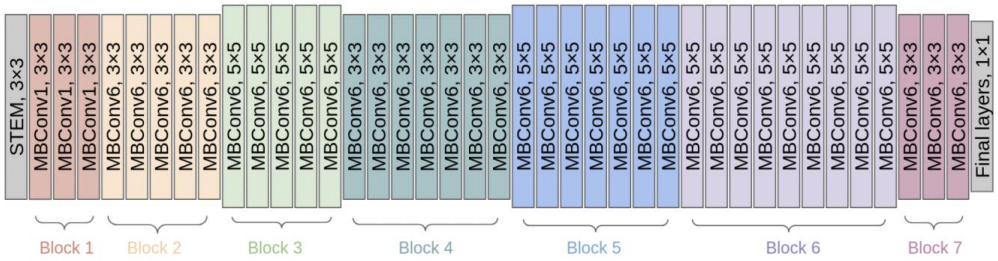

Figure 20: EfficientNet-B5 Structure
This figure represents the layer by layer structure of EfficientNet-B5. [48, Fig. 3A]

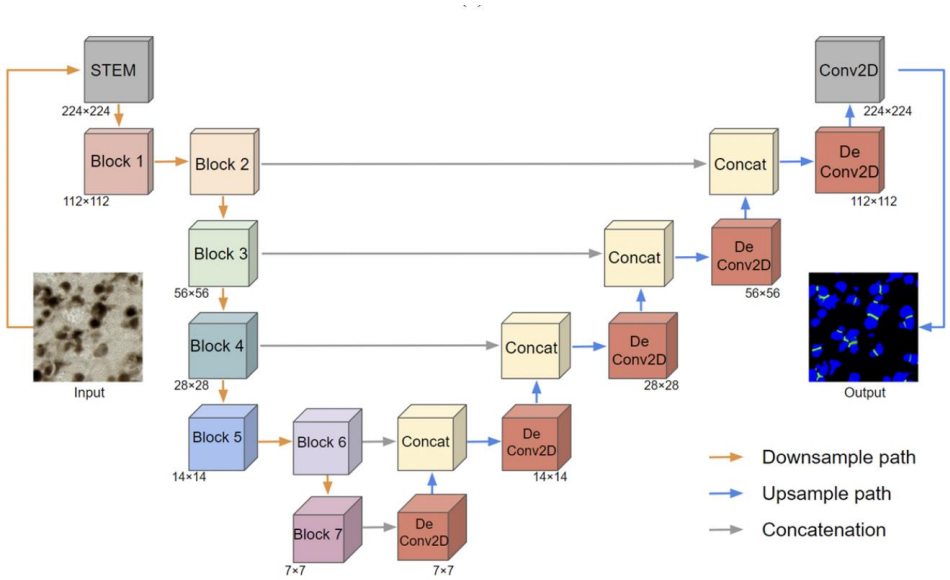

Figure 21: Pixel-Level Mask Synthesis Pipeline
This figure uses the blocks formed through EfficientNet-B5 layers and shows their placement in an encoder-decoder architecture with skip-links. [48, Fig. 3B]

Skip connections are necessary for this context because as the model downsamples the image from 224x224 to a 7x7 image, the higher-level details are lost within the process. To preserve these higher-level details, we utilize skip connections that propagate the feature maps from the encoder block to the corresponding decoder block. In this figure, concatenation simply means an addition operation.

3.2.4.7.3 Loss and Loss Weightages

As stated earlier in the summary, the neuron segmentation is being approached as a semantic segmentation problem, i.e., 0 = background, 1 = neuron interior, and 2 = neuron contour. To handle the loss during training, a combination of categorical cross entropy and soft dice losses is being used.

3.2.4.7.3.1 Categorical Cross Entropy Loss

This loss function is generally used in multi-class classification tasks. In the context of neuron instance segmentation, we classify each pixel as being a part of either 0, 1, or 2. This function assists in quantifying the difference between the probability distribution of all classes associated with the model. The categorical cross-entropy loss CE is defined using (...):

$$CE = -\frac{1}{nc}\sum_{i,j}\sum_{k}^{c} t_{i,j,k}\log\left(p(i,j,k)\right) \tag{9}$$

In this equation, c is the total number of classes within the model. In our case, it is 3. $t_{i,j,k}$ is a predicate that is 1 when the pixel at position (i,j) is from class k. The function p is the probability function. Consequently, the term p(i,j,k) is the probability of (i,j) being from class k.

3.2.4.7.3.2 Soft Dice Loss

This loss is an adaptation from the Dice coefficient which helps us in gauging the similarity between two images. In this case, we attempt to compute the similarity between the predicted label and the synthesized label we propagated through the network. In this loss, we have added 1 to keep the probability p(i,j,k) from being 0. We define this new soft dice loss using (10) as such:

$$D_{class} = 1 - \frac{2\sum_{i,j} t_{i,j,k}p(i,j,k) + 1}{\sum_{i,j} t_{i,j,j} + \sum_{i,j} p(i,j,k) + 1} \tag{10}$$

Now before combining these losses, we have to assign them specific weights to aid in the learning process. The intuition behind assigning weightage is to penalize the network on certain predictions while encouraging it on others. Consequently, we assign 0.3 to D_{neuron} to to emphasize its higher preference while giving 0.2 to $D_{contour}$ to indicate its lower preference. Lastly, an aggregate of both dice losses is assigned to CE, i.e. 0.5 Assigning relevant weights to these losses, we get the final result:

$$L = 0.5CE + 0.3D_{neuron} + 0.2D_{contour} \tag{11}$$

3.2.4.7.4 Post-Processing

Post-processing is a significant part of this research because it is by far the most lucrative domain to improve segmentation results. Although many techniques are available in post-processing, it is still an underdeveloped field when it comes to segmentation for various shapes and sizes. Simple techniques such as thresholding and probability maps fail to generalize to various shapes and sizes of objects in the image.

Methods such as graph partitioning or distance transform have given better results, however, they are computationally expensive and rely on a collection of hyperparameters specific to the dataset itself.

3.2.4.7.4.1 Post-Processing Using Mathematical Morphology

This paper proposes a generic morphological approach that aims to work on all sorts of shapes and sizes of segmentation. Furthermore, the technique must be computationally efficient to allow for an overall faster inference pipeline. The morphological post-processing pipeline consists of the following steps:

Firstly, all the pixels belonging to the neuron class are extracted and translated into a binary mask. Secondly, an erosion filter is applied to this binary mask with a disk-shaped element. This shrinks down the objects present in the binary mask by the diameter of the disk itself. It must be kept in consideration that while performing erosion, none of the neurons must get completely removed from the binary mask itself. While it can be hypothesized that the contours class kept in the target labels will be sufficient for separating the neurons, it's not always possible in dense regions. However, it does create a segment or some form of concavity within the cells which is enough for the post-processing to perform effectively. After erosion, the binary mask consists of residues that are reconstructed through the use of a morphological dilation. The principle is that if the image is eroded by x pixels, then it must be dilated x pixels as well. Using the residues and the dilated masks, water-shedding segmentation technique can be applied to get reconstructed cells with little or no overlap. Furthermore, this technique restores the cell with minimal computation using a single parameter of x, which is the diameter of the structuring element while applying the erosion/dilation.

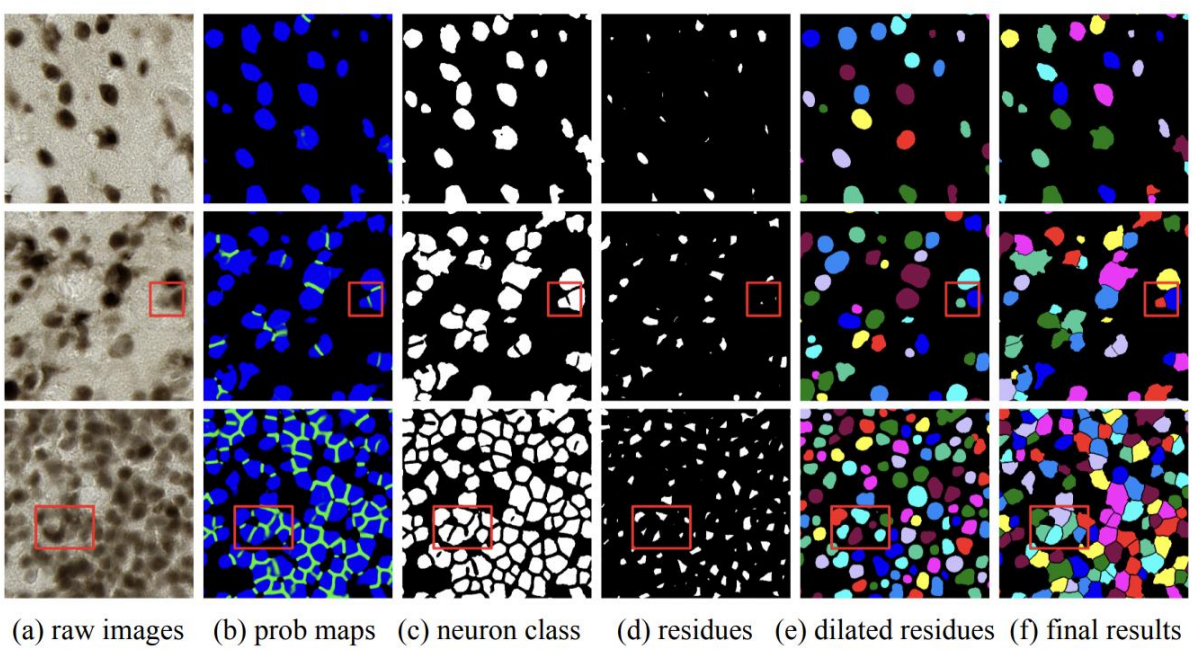

Figure 22: Proposed Pre-Processing Pipeline
This figure depicts the pre-processing pipeline proposed in the paper. (a) is the raw input, (b) probability maps predicted by the model, (c) binary masks after applying argmax to the neurons, (d) residues after erosion, (e) residues after dilation, and (f) results after applying watershed segmentation. [48, Fig. 8]

3.2.4.8 Discussion: A General Deep Learning Framework for Neuron Instance Segmentation Based on Efficient U-Net and Morphological Post-Processing

The paper by Huaqian Wu et al. [48] highlights a general-purpose and effective method of instance segmentation on neurons. Even though the paper only focuses on neuronal segmentation, the authors have done a commendable job of explaining the techniques in a

general segmentation context without any specificities and emphasis on modalities. One thing that stands out in this paper from the others is a special focus on generic morphological post-processing to achieve much better results in segmentation.

More than often, it is stated that segmentation architectures have achieved an upper bound in segmentation accuracies. Consequently, as a researcher, one has to look into things that do not quite relate to model building or training. Instead, improvements can even happen outside of the model itself. The authors make a strong point of using limited data within the experiment. In the context of weak supervision, it is arguable that the techniques used by the authors may have given them an unfair advantage, i.e. manually annotating neurons in the image to assist the synthetic pixel-level mask generation pipeline. It would have been more appropriate if the authors covered techniques related to using either transfer learning or generating automatic annotations. It is not possible, especially in a biomedical context, to manually annotate data since it requires expertise as well as time.

Overall, this paper is a strong baseline to work on, especially, from the morphological post-processing perspective since our dataset includes many overlapping cells. The paper covers general techniques that are helpful in not just biomedical, but any context. Moreover, there is a well-rounded discussion on everything from modelling to loss computations and post-processing. It's a decent read for anyone aiming to get knowledge about weakly supervised instance segmentation.

3.2.4.9 Relevance: A General Deep Learning Framework for Neuron Instance Segmentation Based on Efficient U-Net and Morphological Post-Processing

Our project primarily focuses on two aspects, weak-supervision and instance segmentation. The paper covers both bases quite comprehensively. Although the discussion is exclusively related to neuron segmentation, the applications of techniques mentioned are generalizable to all forms of biomedical segmentation. This paper is relevant because it covers the end-to-end model development flow with a comprehensive discussion on experimentation and implementation. Consequently, it's an essential read for our project.

3.3 Literature Review Summary Table

This table has a general overview of the papers that have been studied so for and their results.

Table 3: Image Segmentation Literature Summary

The summary of various image segmentation research papers.

No.	Name, reference	Authors	Year	Techniques	Results	Description
1.	Survey on Image Segmen-tation Techniques and Color Models [5]	Savita Agrawal, Deepak Kumar Xaxa	2014	Edge-Detection, Thres-holding, Region-Based, Clustering, ANN	ANN based segmentation performs vastly better than other techniques. It should be paired with other techniques to avoid training overheads.	A descriptive analysis of image processing techniques relating to image segmentation. Additional discussion on colour models with emphasis on human-

						perception-modelling.
2.	Survey on Instance Segmen-tation [8]	Abdul Mueed Hafiz, Ghulam Mohiuddin Bhat	2020	RCNN, Fast RCNN, Faster RCNN, Mask RCNN, YOLOACT	Mask RCNN is the most effective instance segmentation architecture, however, its inference times are slower.	A descriptive analysis of the taxonomy of segmentation architectures and RCNN family along with discussion about their distinction with each other.
3.	Biomedical Image Segmen-tation: A Survey [11]	Yahya Alzahrani, Boubakeur Boufama	2021	Edge-Detection, Markov Random Field, Watershed Transform Methods, Graph Search Methods, Parametric Active Contour, Level-Set Methods, Supervised Segmen-tation, Unsuper-vised Segmen-tation, Deep Learning, Atlas-Guided Methods	Deep learning methods outperform any existing image processing by a great margin due to their flexibility on data. Primary driver of these methods are convolutional neural networks which allow feature engineering without explicitly encoding anything.	An analysis of current segmentation methods and their accuracies in a biomedical context.
4.	UNet: CNN for Biomedical Image Segmen-tation [35]	Olaf Ronneberger, Philipp Fischer, Thomas Brox	2015	Skip Connection, CNN, Encoder-Decoder, Architecture	0.000353 warping error in EM segmentation challenge. 0.9203 (IoU) on PhC-U373 dataset. 0.7756 (IoU) on DIC-HeLa dataset.	CNN and Skip Connections are utilized to create an Encoder-Decoder driven architecture for biomedical segmentation.

					Very good performance for biomedical images with little data usage and heavy reliance on augmentations.	
5.	Importance of Skip Connections in Biomedical Image Segmen-tation [36]	Michal Drozdal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal	2016	Skip Connection, Batch Norm	Training and Validation Loss reduce significantly when using Skip Connections in the network.	An overview of why skip connections are important in biomedical image segmentation networks and how they can improve the model performance. An experiment-based walkthrough of its uses and disadvantages.
6.	Instance Segmen-tation with Image Level Supervision [39]	Issam Laradji, David Vazquez, Mark Schmidt	2019	Psuedo Mask Generation, Class Activation Maps	Image-level annotations fed into this model give comparable results to a pixel-level annotated Mask R-CNN. The model gives an mAP25=49.2 on instance segmentation tasks.	Image-level supervision model WISE for instance segmentation is discussed along with synthetic label generation.
7.	Cellpose: Deep learning based generic cell segmen-tation [43]	Carsen Stringer, Michalis Michaelos, Marius Pachitariu	2021	Combined Gradients for Cell Represent-ation	In comparison to Mask R-CNN, the model performed greatly better. Specialist Model ap@0.5 = 0.89 Generalist Model ap@0.5 = 0.93	Discussion of a new cell-representation technique with a generalist approach that uses heat diffusion principles for mapping cell annotations to reversible gradients.
8.	Omnipose: a high-	Kevin Cutler, Carsen Stringer,	2022	Cellpose, Non-linear activation	Better performance than Cellpose and	Discussion on a new state-of-the-art cell

	precision morphology independent solution for bacterial cell segmen-tation [44]	Paul Wiggins, Joseph Mougous		while merging skip connections	subsequently Mask R-CNN. 83% predictions are above 0.8 IoU.	segmentation and representation method that outperforms Cellpose by making improvements in merging skip connections.
9.	A General Deep Learning framework for Neuron Instance Segmen-tation based on Efficient UNet and Morph-ological Post-processing [48]	Huaqian Wu, Nicolas Souedet, Caroline Jan, Cedric Clouchou, Thierry Delzescaux	2022	Pixel-level mask synthesis, EfficientNet-B5 based UNet, Generic Post-Processing Framework	Beats previous state-of-the-art in neuron instance segmentation at 0.931 F1 Score.	Describes a better end-to-end segmentation pipeline consisting of synthetic mask generation, training, and general post-processing. Successfully achieves state-of-the-art results.

3.4 Conclusion

The literature review discussed in this report primarily focuses on four areas: weak-supervision, post-processing, cell-representation, and deep-learning. The reason for choosing these areas is to assist us in covering all bases related to the project itself. The review starts with a primer of segmentation techniques available within general images and then biomedical images. We reviewed survey papers that thoroughly discuss and perform a comparative analysis amongst several techniques for image segmentation. In this regard, [5] provides a baseline in traditional image processing and segmentation through techniques such as region-growing, thresholding, clustering, and water-shedding. Additionally, it provides and overview of practical cases where such algorithms prove to be useful. The analysis presented in [5] led to the affirmation that traditional image processing techniques are fairly outdated and cannot encompass the variety and flexibility exhibited in biomedical images.

In a shift to diving deeper into deep learning based approaches, [8] provides a close look into the instance segmentation problem itself along with its challenges. They shed light on the taxonomy of instance segmentation along with the R-CNN family of architectures. While the discussion is solely constrained to the R-CNN family, it highlights a general limitation present across all deep learning architectures; lack of data. These deep learning architectures need vast amounts of high-quality annotations which can help the model generalize to a number of images. Moreover, some architectures such as R-CNN or Fast R-CNN face issues when dealing with object bodies of variable sizes. This paper highlights the need to discover newer and more biomedical-oriented architectures. While we did review certain image processing techniques pertaining to biomedical imaging [11] such as edge-detection and Markov Random Fields, our focus was on deep learning because of the limitation faced by almost all image processing techniques, i.e. lack of generalization.

In this pursuit, the U-Net [35] architecture stood out with its state-of-the-art architecture and lesser reliance on large samples of annotated data. U-Net variants have dominated the area of biomedical image segmentation for many years. In an effort to understand U-Net deeply, it was only rational to explore the only thing common to all modern image-segmentation architectures – skip connections. This paper [36] approaches skip connections in an exclusively biomedical image segmentation context and through various experiments proves their positive contribution to the overall performance of the network. However, even after these contributions, the lack of data overpowers the positive attributes of the U-Net. In the final category of our literature review, we wished to acquire knowledge about techniques that could help us make use of very little data to achieve excellent results comparable to that of having high quality annotations. In a nutshell, we concluded that the upper bound on image segmentation models has been achieved and the improvements must be made out of the box.

This compelled us to implore weakly-supervised learning techniques in [39, 48]. While both of these papers cover comprehensive techniques in weak-supervision, only [48] applies them in a biomedical setting. In general, we observed that most weak-supervision techniques involve synthesizing labels for images and training the network with a certain confidence. Using such pseudo-labels helps network learn (possibly) good features and eradicate the bad ones using strongly labeled data. This in turn improves the overall feature extraction of the network. Our literature review choices revolved around the major challenges of our project: few labelled data items, variation in cells, and accurate segmentation. This helped us get a detailed insight into our problem while also giving a direction for direction. We are positive that these techniques will help us in implementing state-of-the-art models ourselves and hopefully improve on them in the future.

Chapter 4: Software Requirement Specification

This section provides an in-depth analysis of all associated software requirements of our project. It includes the list of features, the functional requirements, the quality attributes, the non-functional requirements, related assumption, the hardware and software requirements, the use cases, graphical user interface, and the risk analysis of our project.

4.1 List of features

The following features will be available in the system.

- Run pre-trained segmentation models on whole-slide images.
- Modify inference, pre-processing, and post-processing pipelines.
- Save inferences in common formats, i.e. TIFF masks & COCO annotations.

4.2 Functional Requirements

The following functional requirements shall be available in the system.

- The user shall be able to read whole-slide images in specified formats.
- The user shall be able to run the pre-trained segmentation model on whole-slide images.
- The user shall allow users to save segmentation results.
- The user shall allow users to alter post-processing parameters.
- The user shall be able to toggle the RGB channels of the image.
- The user shall be able to zoom in and zoom out of the image.
- The user shall be able to change the contrast of the image.
- The user shall be able to change the brightness of the image.
- The user shall be able to change the sharpness of the image.
- The user shall be able to normalize the image.
- The user shall be able to choose their required model.
- The user shall be able to access the GPU.
- The user shall be able to choose the overlay results.
- The user shall be able to change the overlay opacity.
- The user shall be able to change the prediction confidence.
- The user shall be able to adjust the minimum cell size.
- The user shall be able to view the insights like the total cells detected, the mean cell diameter, and the cell size variation.

4.3 Quality Attributes

The project aims to incorporate the following fundamental quality attributes in order to ensure the prevention of application defects. Each attribute shall be used to evaluate the application quality and performance during operation.

4.3.1 Reliability

Since the project is tailored for biomedical usage, it is critical that the application produces consistent results across a variety of inferences. Regardless of the environment or the operating conditions, the product shall produce correct and reliable results.

4.3.2 Maintainability

The project comes with a battery of pre-trained segmentation models that can be applied to whole-slide images. While the implemented algorithms shall be the current state-of-the-art, it is reasonable to assume that more algorithms shall be proposed in the future as well.

Considering this need, the application shall have enough maintainability to update the algorithms or switch the underlying technologies for efficient inferences later on.

4.3.3 Usability

The project aims to serve as a utility for biomedical professionals to assist in their diagnostic workflow. Since the audience is non-technical, the project must have high usability and accessibility. The usability of the project shall be evaluated with the following performance indicators.

- Consistent user interface across the application to avoid any confusion.
- Easy for new or infrequent users to learn to use the system.
- Easy for users to input data and interpret the output.

4.3.4 Correctness

Considering a biomedical context, it is critical that the application produces functionally correct results and avoids false positives. Given such constraints, the project aims to strictly comply with the functional requirements and incorporate empirical metrics for further improvement.

4.3.5 Efficiency

Training a model for cell segmentation can be costly in terms of time and computational resources. The project aims to minimize the barrier to entry by running the inferences over the cloud. Additionally, the project also provides local inference options in cases where the user has sufficient computational resources.

4.3.6 Flexibility

This attribute closely relates to the maintainability aspect of the application. The project aims to provide flexibility in terms of code and model training for researchers and users to adapt the application to their needs. While the project comes with several pre-trained models for use, it also comes with options for users to train the models on their own biomedical data. Additionally, in terms of code, the project shall give enough room to add and implement further custom models.

4.4 Non-Functional Requirements

The following non-functional requirements will be incorporated into the system for efficient functioning of the system. Each requirement has been described in the context of the software quality attributes discussed earlier.

4.4.1 Reusability

The system shall be modular and have reusable components that can be incorporated into other research projects with ease. Additionally, it will produce correct results regardless of the working structure.

4.4.2 Reliability

The system shall be reliable and produce valid inferences for whole-slide images regardless of the operating environment and hardware constraints. In addition to the inferences, the system shall allow users to evaluate the results through state-of-the-art metrics such as F1 and IoU.

4.4.3 Performance

The system shall allow users to take advantage of cloud computing to accelerate performance and produce correct results at high speed. In addition to utilizing cloud computing, the system shall also give users the ability to use their offline resources if available.

4.4.4 Robustness

The system shall be able to process and deal with extensive amounts of data without loss of critical information and operational capability. In case of connection dropouts or hardware failures, the system shall preserve the progress made so far.

4.4.5 Extensibility

The system shall use standard development standards to provide extensibility for future researchers. This shall allow potential contributors to extend the project through new algorithms or better architectures.

4.5 Assumptions

The specification makes use of the following assumptions:

- The reader has sound knowledge of the project's purpose and utility.
- The reader is familiar with basic machine learning and AI algorithm training protocols.
- The reader has sufficient technical knowledge about Python development.

4.6 Hardware and Software Requirements

The project employs a multitude of hardware and software services for its operations. For the purpose of developing this project, the following hardware and software requirements need to be fulfilled.

4.6.1 Hardware Requirements

The following are the hardware requirements required for the operation of the system:

- Stable wired or wireless internet connection.
- Sufficient secondary memory (256 GB) to store whole-slide images and inference results.
- Sufficient primary memory (16 GB) to run the inference algorithms.
- Sufficient GPU memory (16 GB) to perform inferences locally.

4.6.2 Software Requirements

The following are the software requirements required for the operation of the system:

- Modern web browser, i.e. Google Chrome or Firefox.
- Python, 3.8 or above.
- Anaconda.
- Flask / Django.
- OpenCV / SKImage.
- PyTorch.
- Google Colab.
- MONAI.
- Albumentations.
- TQDM.
- Operating System: Ubuntu or macOS.

- React
- JavaScript

4.7 Use Cases

Following are the use cases identified in our web application.

4.7.1 Select Post Processing Methods

Name	Select Post-Processing Algorithm		
Actors	User		
Summary	The user shall be able to select from various post-processing techniques and algorithms to apply to their inferences.		
Pre-Conditions	The user is connected to the internet. The images have been pre-processed and the segmentation model has been applied.		
Post-Conditions	The post-processing algorithms have been selected and will be applied on inferences.		
Special Requirements	None		
Basic Flow			
Actor Action		**System Response**	
1	The user selects the post-processing operations to apply.	2	The system marks those post-processing operations as selected and displays a preview.
		3	The system stores the post-processing pipeline and applies it after inference.
Alternative Flow			
No alternative flow.			

4.7.2 Run Inferences on Model

Name	Run Inferences on Model		
Actors	User		
Summary	The user can run predictions on whole-slide images using the pre-trained models.		
Pre-Conditions	The user is connected to the internet. The pre-trained models are downloaded.		
Post-Conditions	The segmentation results will be displayed.		
Special Requirements	None		
Basic Flow			
Actor Action		**System Response**	
1	The user uploads an image.	2	The system displays the image.

3	The user toggles the image view parameters to analyze the image.	4	The system displays the selected channels.
5	The user enters the zoom percentage.	6	The system displays the selected zoom percentage.
7	The user selects the contrast percentage.	8	The system displays the selected contrast percentage
9	The user selects the brightness percentage.	10	The system displays the selected contrast percentage.
11	The user selects the sharpness percentage.	12	The system displays the selected sharpness percentage
13	The user selects normalize.	14	The system displays the normalized image.
15	The user selects the segmentation model level.	16	The system displays the selected segmentation model level.
17	The user selects GPU.	18	The system displays the choice.
19	The user clicks the “Segment” button.	20	The system slices the whole-slide image into patches and starts batch requests to the inference API.
		21	The system displays the segmentation results as they are fetched from the API.
		22	The system displays the insights of the results fetched from the API.
Alternative Flow			
19	There is an error with segmenting the image.	20-A	The system will display the error message: “Segmentation failed.”

4.7.3 Save Segmentation Results

Name	Save Segmentation Results
Actors	User
Summary	The user shall be able to save the segmentation results performed by the predefined model.
Pre-Conditions	The user is connected to the internet. The inference has been performed on the image.
Post-Conditions	The results shall be saved on the user disk.
Special Requirements	None

Basic Flow			
Actor Action		**System Response**	
1	The user clicks “Save”	2	The system converts inference results into a TIFF image.
		3	The system starts the download process.
Alternative Flow			
1	The system fails to convert the image into a TIFF format.	1-A	The system will display the error message: “Image conversion failed.”

1	The download fails.	1-A	The system will display the error message: “Download failed.”

4.8 Graphical User Interface

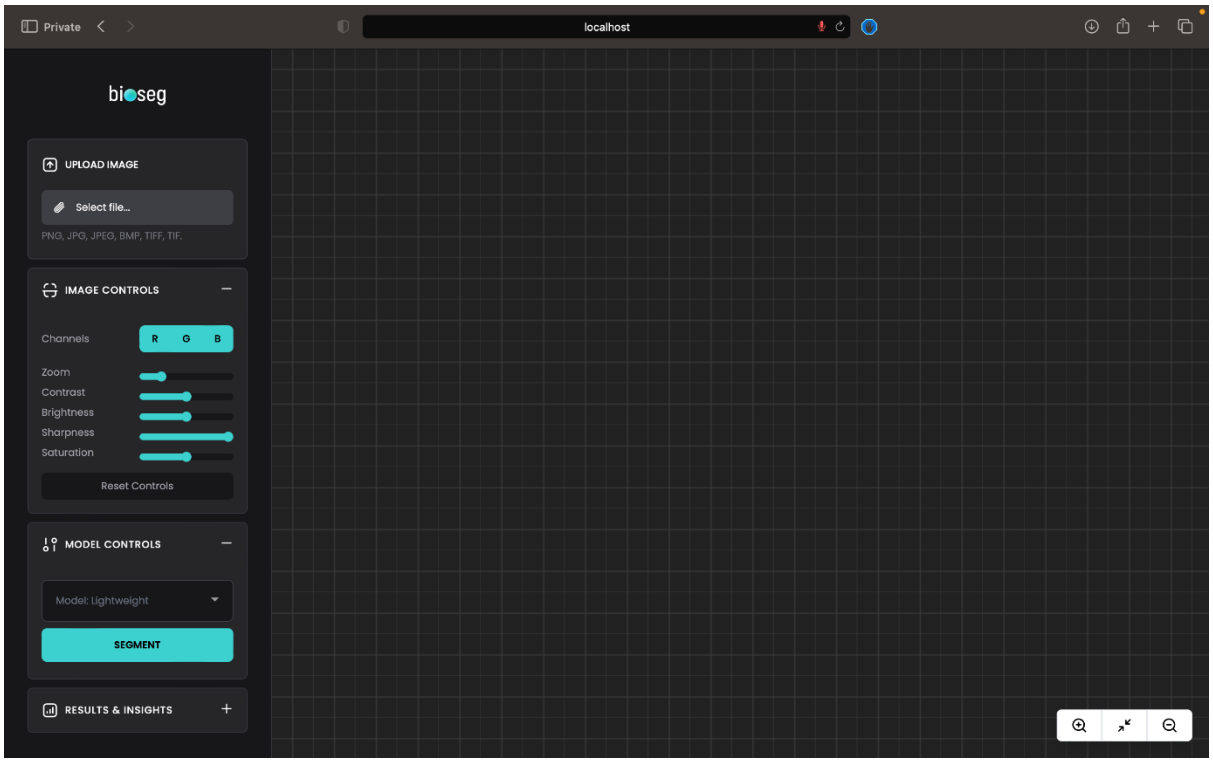

Figure 23: Inference page before segmenting

This figure depicts how display would be before segmentation occurs.

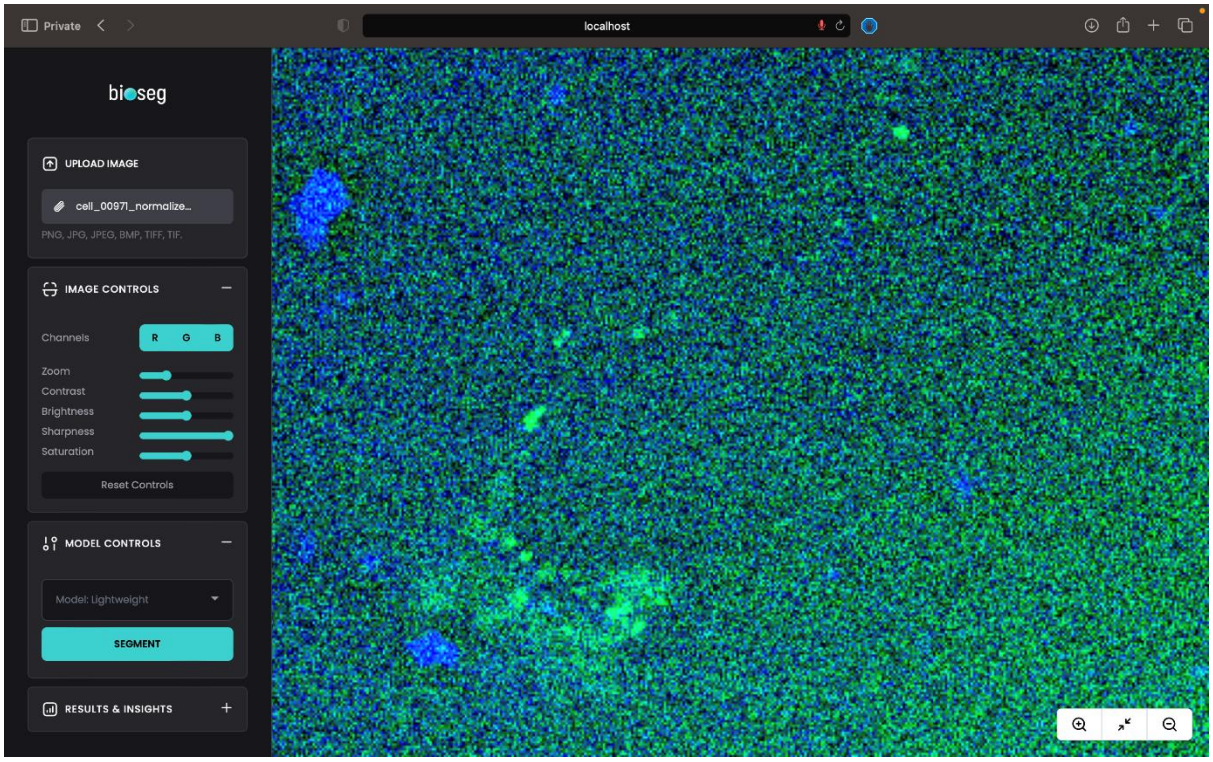

Figure 24: Inference page after uploading the image.
This figure depicts how display would be after uploading the image.

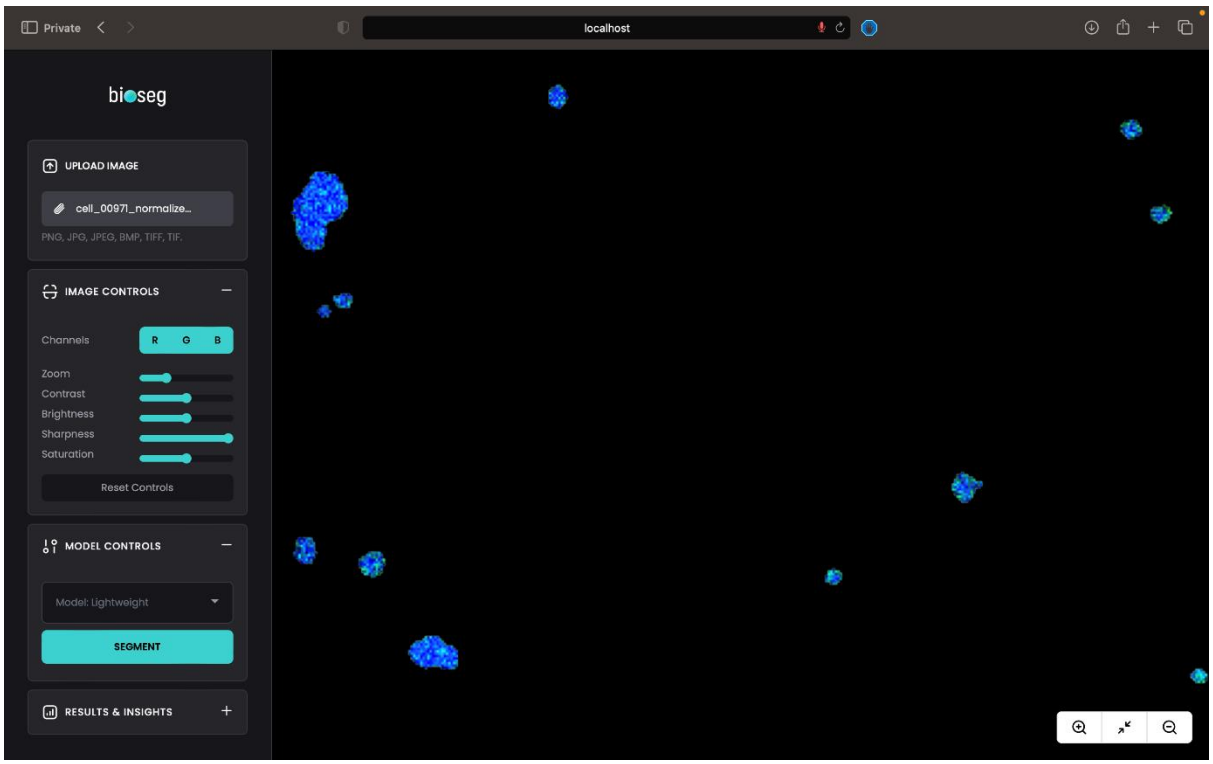

Figure 25: Inference page after segmenting
This figure depicts how display would be after segmentation occurs. Image can be saved after segmentation.

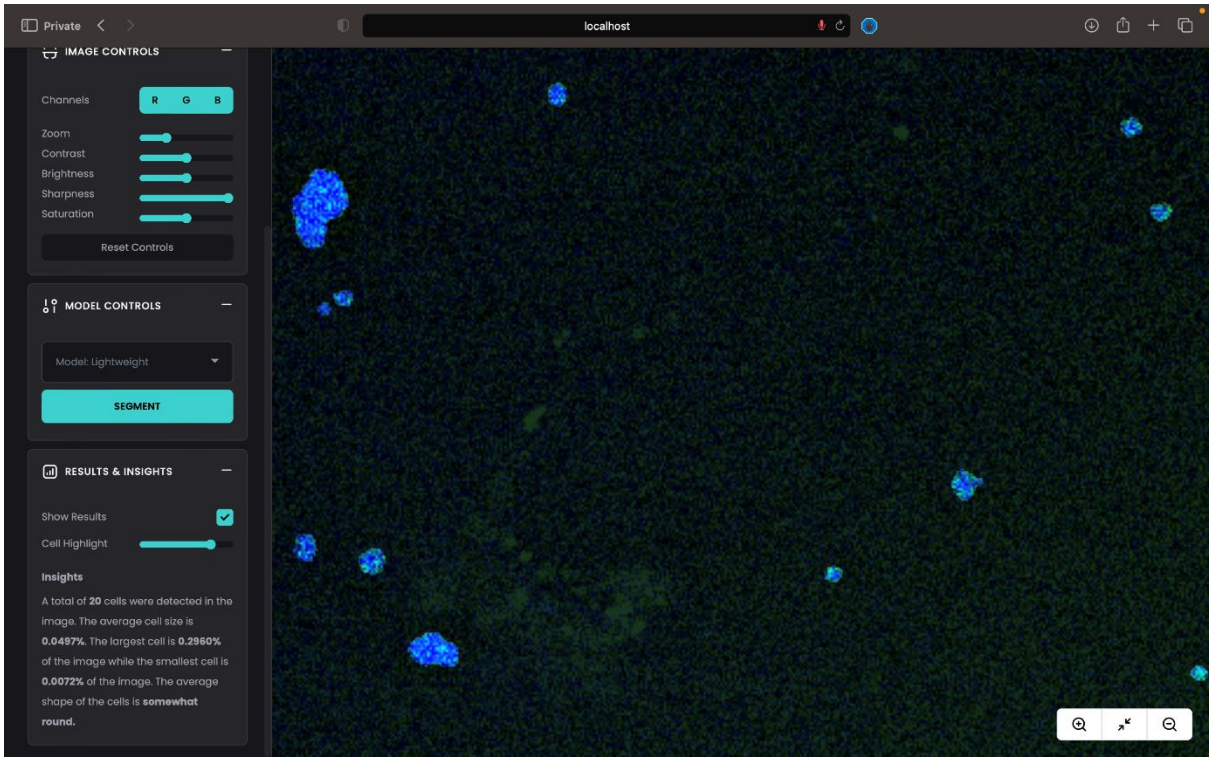

Figure 26: Inference page after segmenting and adjusting overlay.
This figure depicts how display would be after segmentation occurs and adjusting the overlay.

4.9 Risk Analysis

In the conducted analysis, the project may be exposed to certain risks during development. The risks have been identified and elaborated further below.

4.9.1 Technical Analysis

As biomedical segmentation using artificial intelligence is a continuously evolving field, new training protocols and libraries are constantly being published to aid in one or the other part of the process. As the research and development for this project will be conducted throughout the specified duration, it may be necessary to evolve the project requirements or constraints as new developments come forward. In addition to the learning time required for these technologies, the project may also suffer from extended deployment time due to the risks that come along with adopting new technologies. The overall impact of this risk is low.

4.9.2 Performance Analysis

The project shall be deployed on a major cloud platform such as Amazon Web Services or Google Cloud. However, these cloud service providers tend to utilize cold starts and lazy execution to save on resources. Even with paid plans, most cloud providers give a first-come-first-serve solution for model deployment. In such a case, project performance may be at risk. The overall impact of this risk is low.

Chapter 5: Proposed Approach and Methodology

This chapter discusses the proposed approach and methodology to implement accurate cell segmentation in a weakly-supervised learning constraint. It includes the pre-processing and normalization, localization, segmentation, post-processing, weak supervision, and label correction details.

5.1 Pre-Processing and Normalization

Image pre-processing and normalization are critical steps prior to training as well as inference. The provided training dataset consists of images with the following inconsistencies that need to be addressed:

- Missing channels.
- More than 4 channels.
- Colors and contrast issues.

In terms of normalization, as there are various modalities present in the dataset with variable levels of contrast and colouring, there is little intuition in selecting a reference image for normalization. Consequently, normalization needs to be done on an image basis, i.e. image distribution is used to normalize the pixel colour ranges.

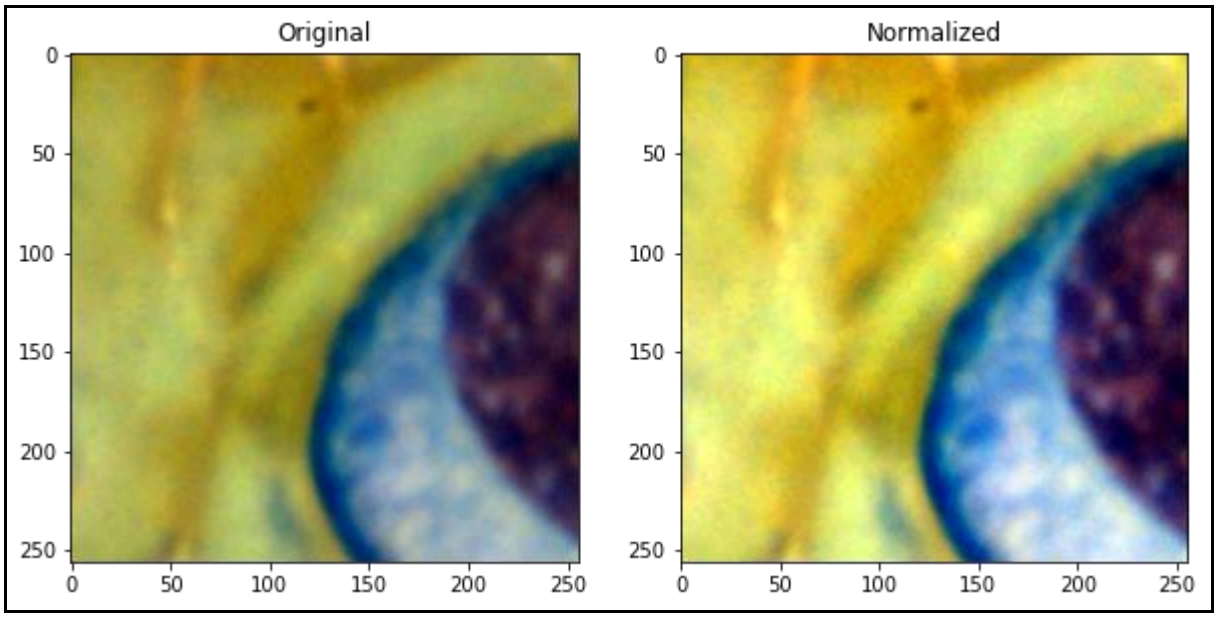

Figure 27: Percentile Normalization

This figure illustrates the normalization on a random patch.

In the example above, normalization highlights the regions of interest while providing more clarity into the structural composition of the cell itself. It is evident that normalization can assist the model in the learning process as well as in inference. The missing channels in the images shall be dealt with using the channel repetition technique. If an image is missing channels, the first channel will be assumed as the R channel and duplicated twice to form G and B channels. Besides numerical normalization techniques, stain networks can also be employed to perform contextual normalization.

An additional operation of pre-processing needs to be performed on the label as well. As the provided labels are usually given in a TIFF format in which every individual cellular body is

labelled with a unique integer, an extra step of converting the cellular bodies into interiors and contours must be performed. This step computes the contours of the cells and labels them as a boundary class while labelling the enclosed regions as the interior class.

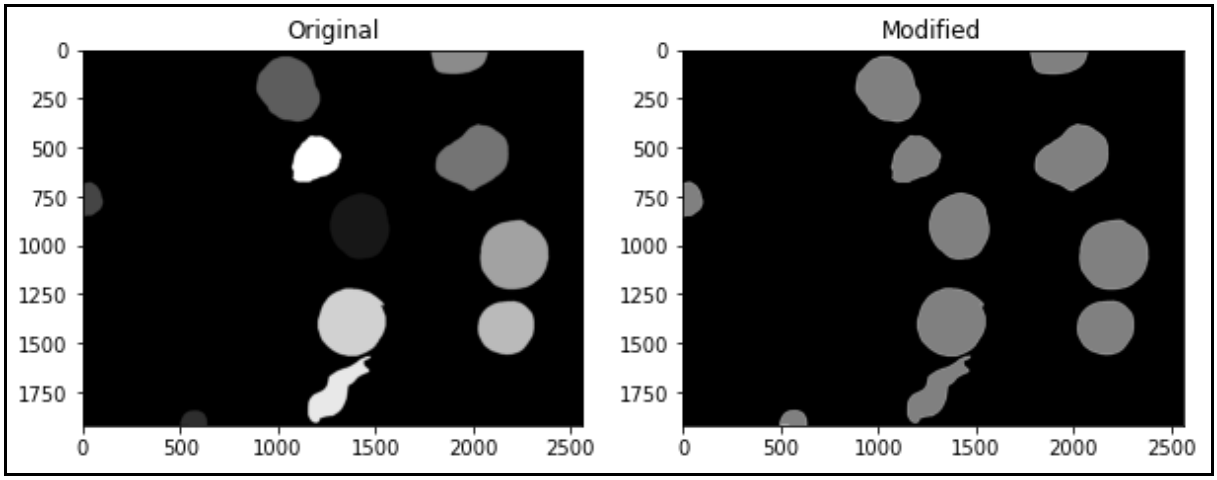

Figure 28: Label Pre-processing
This figure illustrates the label before and after normalization.

Label normalization is also necessary due to the fact that the model must compute only a handful of classes to perform well. If the data is given as is, it will hinder the learning process of the model and require a vast number of trainable parameters.

5.2 Localization

During cell segmentation, there is a high concentration of false positives and artifacts scattered across the predicted label. This happens due to the model trying to identify everything from small specs to large cellular bodies. In this approach, a lot of misidentified cells also make the cut to the final label. The proposed method applies localization to focus on regions where the cells may be present. Moreover, to avoid complete data loss, it assigns a low-confidence score to regions excluding the area of interest. This is achieved primarily through the object detection backbone of the network. The method uses a collection of single-stage and multi-stage detectors to identify regions of interest within the image itself.

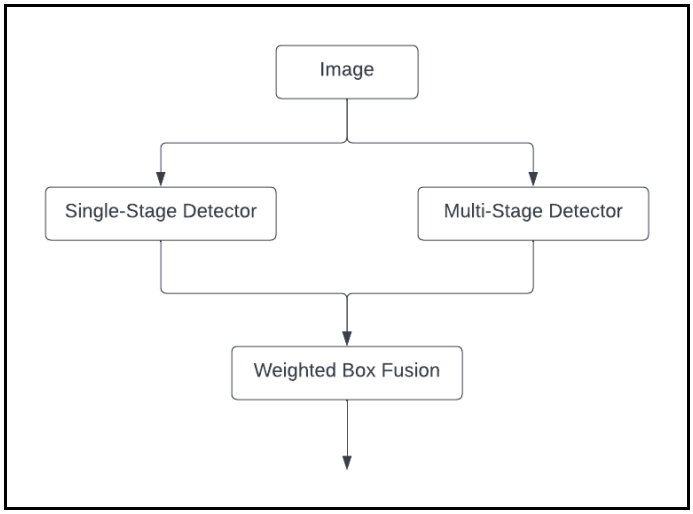

Figure 29: Model Detection Head

This figure illustrates the model detection head that uses multiple detectors and combines their bounding boxes with weighted box fusion.

Due to the nature of object detection, it can work sufficiently well on the provided data unlike segmentation which requires numerous annotated samples. While any single-stage and multi-stage detectors can be used for filtering area of interest, YOLO variants for single-stage detection and Mask R-CNN for multi-stage detection have shown promising results.

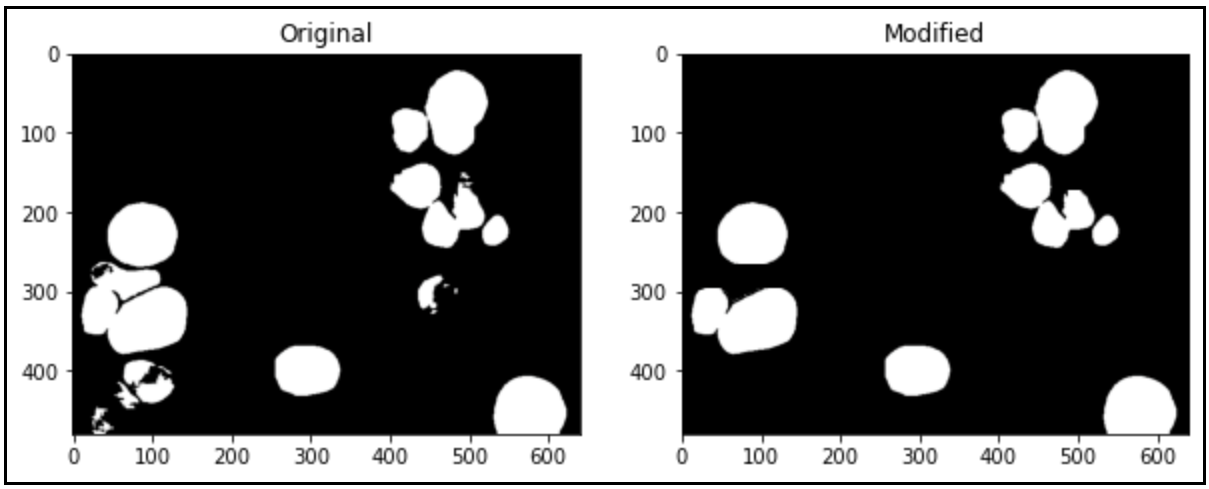

Figure 30: Segmentation Results after Detection
This figure illustrates the inferences after excluding false positive regions.

To merge the bounding boxes from these multiple detectors, a weighted box fusion method is applied which ensembles and combines bounding boxes in the most beneficial way. It provides significantly better results than non-max suppression in biomedical images.

5.3 Augmentation

As the problem deals with a limited number of data samples, it is important to explore techniques that can potentially help fill that gap and increase the generalizability of the model itself. Augmentations offer a way to bridge this gap by generating samples through morphological operations, transformations, and visual manipulations. When it comes to biomedical data, not every augmentation is a winner. The nature of augmentations depends directly on the nature of the testing data. Consequently, the augmentations must try to be as close to the real-life data as possible to avoid potential overfitting. Through experimentation and keeping in view the nature of testing data, the following augmentations show promising results:

- For bringing out edges and details: CLAHE, Sharpen, and Emboss.
- For emulating noise that happens in real-life microscopy: Gaussian Noise.
- For learning structural information rather than colour: Color to Grayscale.
- For learning cells rather than modality: Grayscale to Colors.
- For dealing with poor quality images: Blur, Median Blur, and Motion Blur.
- For generalization on poorly processed images: Contrast and Brightness.
- For learning structural inconsistencies: Elastic Transform, and Perspective.
- For generalization on positioning: Random Scaling, Rotation, and Flipping.
- For adapting to testing data: Channel Shuffling.
- For accurate segmentation regardless of modality: Cut and Transplant.

In addition to image manipulation, external datasets are also being employed to perform transfer learning and pick the features. These datasets do not have to be explicitly segmentation datasets. They can also be detection datasets to perform training of the detection heads.

5.4 Segmentation

As discussed in the previous section, the model relies extensively on the accurate regions selected by the detector backbone. When it comes to biomedical image segmentation, the contrast and color discrepancy can make it a challenging task. There are two considerations for the segmentation architecture for this problem:

- Large parameter capacity to accomodate multiple modalities.
- Ability to segment high and low-level features.

Through experimentation, it is evident that UNet-based architectures have an exceptional supremacy over any instance segmentation architecture. Moreover, their performance in biomedical segmentation outperforms any other network at hand. It is only reasonable that the proposed method also employs a UNet-based network along with a Mask-RCNN head. The Mask R-CNN gives an affirmation that all smaller cellular bodies will not get skipped past by the UNet-based network. The general architecture of such a segmentation network is shown in the figure below.

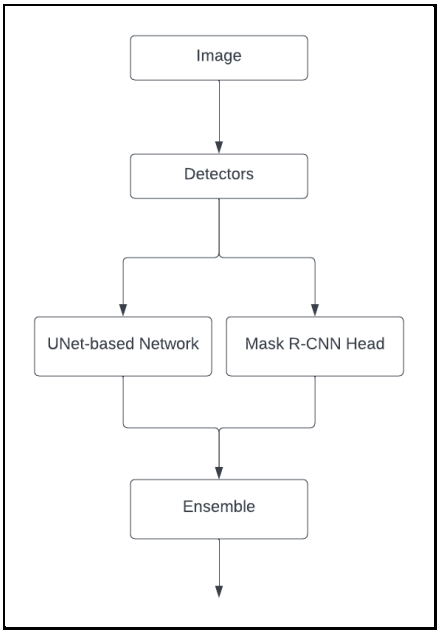

Figure 31: Segmentation Pipeline
This figure illustrates the segmentation pipeline with ensemble.

5.5 Weakly Supervised and Semi Supervised Learning

To maximize the utilization of the dataset and generalize the model to segment various images, certain techniques and training protocols need to be employed for effectiveness. One of the major avenues to improve the performance of a model with limited data is through weakly-

supervised and semi-supervised learning. The proposed method uses a combination of these two to gain performance gains in terms of segmentation accuracy. It is to be noted that there is a large amount of unlabelled data has been provided as well. This unlabelled data can be fed through the existing model to generate labels. While the labels will not be accurate due to the initial lack of training, they will be a guiding heuristic for our model to learn features. After a specified number of epochs, the model shall be trained on the strongly labelled data to fine-tune. The goal of this pipeline is that in the process of learning from weakly-labelled data and fine-tuning on strongly-labelled data, the model will learn the good features of cells while discarding the bad features. To reflect the nature of weakly-labelled data, the loss function will be weighted and shall penalize the model accordingly. In a nutshell, the penalty for strongly-labelled data will outweigh the weakly-labelled data.

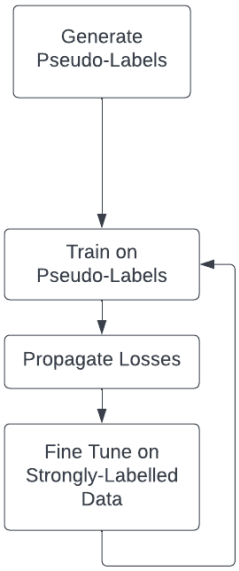

Figure 32: Weakly-Supervised and Semi-Supervised Learning Pipeline
This figure illustrates the weakly-supervised and semi-supervised learning pipeline.

5.6 Post Processing and Label Correction

A significant issue in weakly-supervised and semi-supervised learning was the presence of bad features in pseudo-labels. While a fine-tuning step was incorporated in the pipeline to unlearn the bad features, there is still a possibility that such features may still stay in the model after the training. This can happen especially when the unlabelled data has a particular set of features that the strongly-labelled features may not be able to outweigh. Consequently, a concept of label correction networks is proposed that use a network designated to computing and correcting label errors.

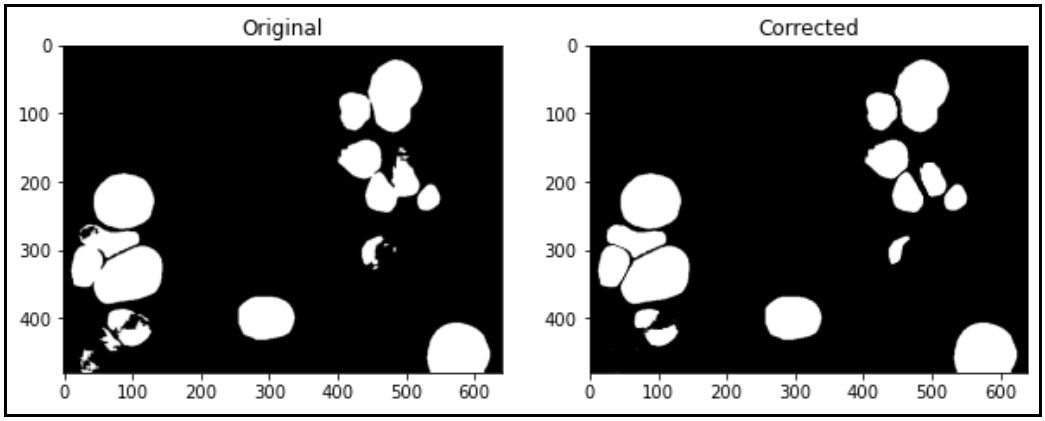

Figure 33: Label Correction Network Outputs
This figure illustrates the label correction output that can refine pseudo-labels.

Such a network can also effectively work as a variable post-processing pipeline for cell separation and refinement. As discussed in the earlier sections of the report, one of the major issues faced during the cell segmentation is the separation of cells in the mask due to their variable nature. A general purpose post-processing and label correction pipeline would follow the structure below.

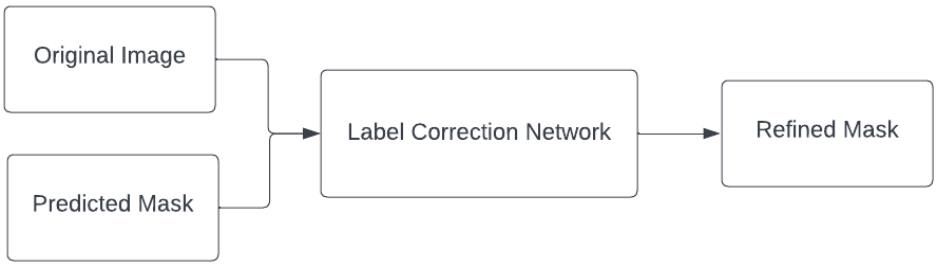

Figure 34: Label Correction Network Overview
This figure illustrates the label correction network.

Chapter 6: High-Level and Low-Level Designs

This chapter discusses the high-level and low-level designs of the project. It includes the system overview, the design considerations, system architecture, architectural strategies, class diagram, sequence diagrams and the policies and tactics related to the project.

6.1 System Overview

The system can be split into two core areas. The front end and the back end. In the proposed system architecture, the backend works in a completely independent runtime. Additionally, it exposes communication endpoints through a public-facing API. Through this monolithic architecture, the system caters to various clients, whether desktop or web. A detailed description of the system architecture is provided in Section 6.3.

6.2 Design Consideration

This section describes the design considerations for the system. It includes the assumptions and dependencies of the system, the general constraints, the goals and guidelines, and the development methods.

6.2.1 Assumptions and Dependencies

The project makes use of the following assumptions:

- The user shall have basic knowledge of computer software installation and usage.
- The user shall have basic knowledge of operating systems and their usage.
- The user shall have basic knowledge of web browsing and web surfing.
- The operating system shall be GNU-based for convenience.

The project makes use of the following dependency:

- A stable and constant internet connection is required for this tool as it is web based.

6.2.2 General Constraints

Following are the general constraints related to the web application.

6.2.2.1 Hardware or Software Environment

As it is a web application, a stable wired or wireless internet connection would be required. A sufficient secondary memory (256 GB) would be required to store whole-slide images and the inference results. A sufficient primary memory (16 GB) would be needed to run the inference algorithms and a GPU memory (16 GB) to perform inferences locally. Moreover, a GNU-based Operating System and an up-to-date browser would be required to run the web-based application.

6.2.2.2 End-User Environment

The end-user will require a web browser in their system as they will have to run this web application.

6.2.2.3 Availability or Volatility of Requirements

A high-speed internet connection will be always required for the smooth running of the application. Furthermore, sufficient storage capacity would be required as well.

6.2.2.4 Standards Compliance

The system will be in accordance with the policies and standards described by the World Wide Web Consortium (W3C).

6.2.2.5 Interoperability Requirements

The backend of the application will be hosted on a Linux based system while the frontend will be platform-independent.

6.2.2.6 Interface/Protocol Requirements

The HTTPS protocol will be used for communication between the end-to-end users while running the application.

6.2.2.7 Data Repository and Distribution Requirements

MongoDB shall be used for the data storage and retrieval implementation of the system. The database shall be accessed through a standard ORM such as Mongoose or Prisma.

6.2.2.8 Security Requirements

The application will use the HTTPS protocol for over-the-internet communication.

6.2.2.9 Memory and Other Capacity Limitations

The system can have an alternating choice between a CPU and a GPU. It should at least have the memory limitations mentioned in 6.2.2.1.

6.2.2.10 Performance Requirements

A high-speed internet connection and GPU will add to the performance of the application.

6.2.2.11 Network Communication

Communication will be done through HTTPS.

6.2.2.12 Validation and Verification Requirements

No validation and verification requirements are present as the system shall be accessible to everyone.

6.2.2.13 Language Constraint

This web application will be available in English language.

6.2.3 Goals and Guidelines

Following goals, guidelines, principles, and priorities are a part of this project:

- User friendliness
- Emphasis on accessibility
- Keep it simple stupid! or the KISS principle
- Emphasis on accuracy rather than resource consumption

6.2.4 Development Methods

The development method used for this project is Scrum which is an incremental approach that allows collaboration and quick development of projects with fluid requirements. Scrum was chosen as the development method as it allows the team members to work on small pieces of

the task at a given time. These smaller chunks of work, more commonly called “sprints”, allowed our team to put forward significant improvements in the model over the course of the development of the project. Furthermore, the progress for each sprint was put in a retrospective manner in front of the scrum master to gain constructive feedback.

6.3 System Architecture

The system uses a monolithic architecture with various services running on the same instance. It is important to have the services closely knit together to avoid communication overheads since the application deals with several MBs of images and their processing. Additionally, this approach gives the app flexibility to be incorporated with any front-end client.

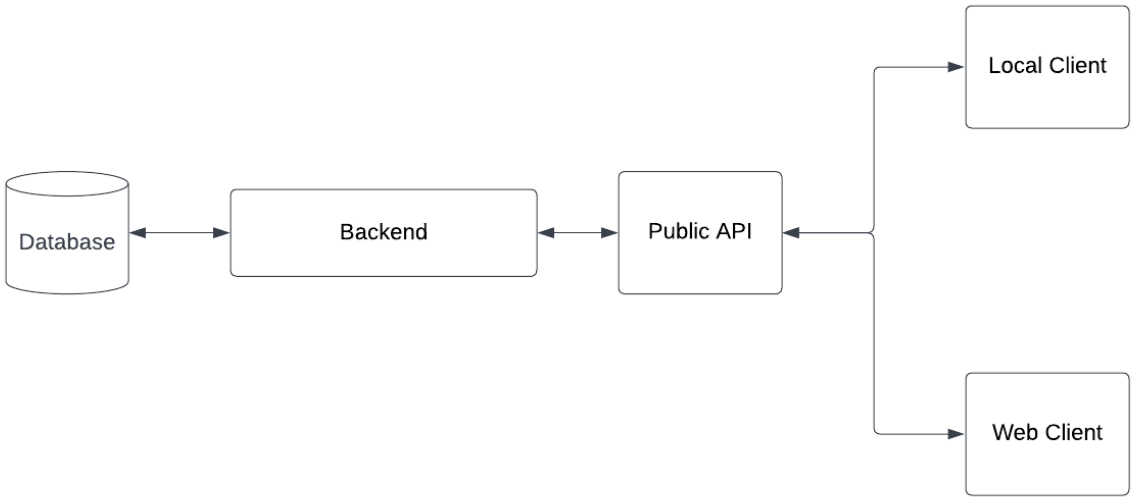

Figure 35: System Architecture Diagram
Diagram for high-level system architecture.

The application provides a public-facing API for all authorized clients to make requests for image segmentation. The backend module provides an abstraction to perform all common functions stipulated in the requirements, i.e. run inference on a whole-slide image, run pre-processing on a whole-slide image, or train model on custom biomedical data.

6.3.1 Client Component

The web client and local client follow the same architecture. However, deployment operations differ as needed. A major highlight in both clients is the batch processing gateway that handles large amounts of data efficiently.

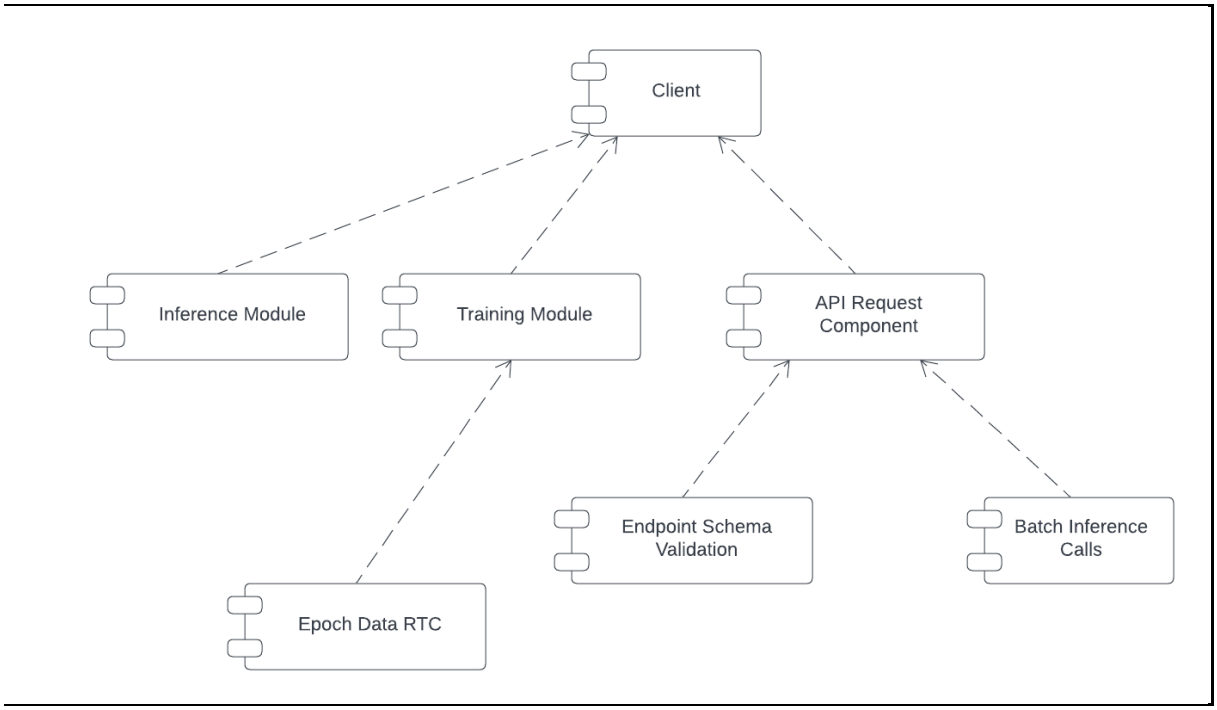

Figure 36: Client Component Diagram
Diagram that illustrates major components within the application client.

Common whole-slide images can range from 500 MBs to several GBs. In such a scenario, it is not a robust approach to send the image content in a single request. Consequently, the batch processing gateway handles the splitting of this image into several API requests that are catered to in a queue-like fashion. This approach has been discussed in detail in the architectural strategies.

6.3.2 Backend Component

The backend component primarily handles computationally intensive tasks at runtime. The intuition behind offloading such tasks to the backend server is to allow users to fully leverage the power of cloud computing. Additionally, it makes the tool more accessible to professionals with minimal computational resources.

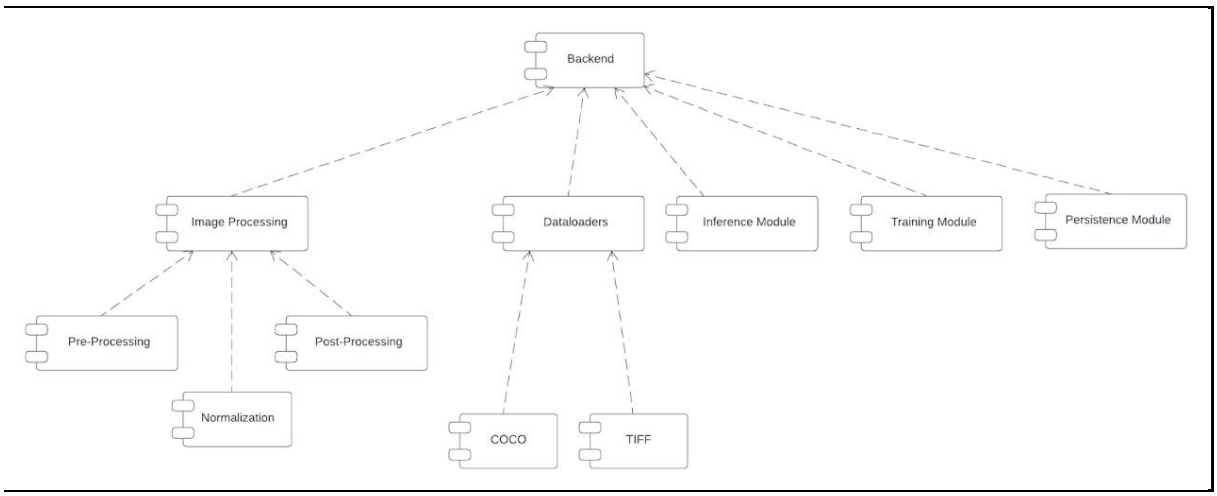

Figure 37: Backend Component Diagram
Diagram that illustrates major components within the application backend.

As observed in the component diagram, the backend employs a host of utilities ranging from image manipulation, processing, and normalization, to training and inference. Furthermore, the component structure is defined in a way that more dataset formats or models can be accommodated in the future with ease.

6.4 Architectural Strategies

The architectural strategies for the given system derive from industry standards and prioritize system robustness, flexibility, and interoperability. To avoid excessive communication overheads within the services, monolithic architecture with a public-facing API is the most appropriate solution.

6.4.1 Technology Stack

As the project primarily deals with machine learning models and research, Python and Flask are the most suitable choices for backend operations. As for the front end, the application will use the Mongo Express React Node (MERN) stack for development. To compile native executables, the web app will be paired with ElectronJS if needed. This technology stack has been chosen due to the minimal learning curve and widespread support. Additionally, the technology stack has plenty of online resources in case of issues.

6.4.2 Database Management

As the selected technology stack is MERN, the database of choice shall be MongoDB. The application uses MongoDB due to its simplicity and ease of integration using Mongoose with the backend.

6.4.3 Frontend Paradigms

The application UI shall be designed within the constraints of the Material UI Design system and shall be validated with Apple’s Human Interface Guidelines.

6.4.4 Version Control System

The application shall be hosted in the form of a collection of repositories on GitHub. The development team shall make pull requests and the changes will be incorporated upon approval of two other team members.

6.5 Class Diagram

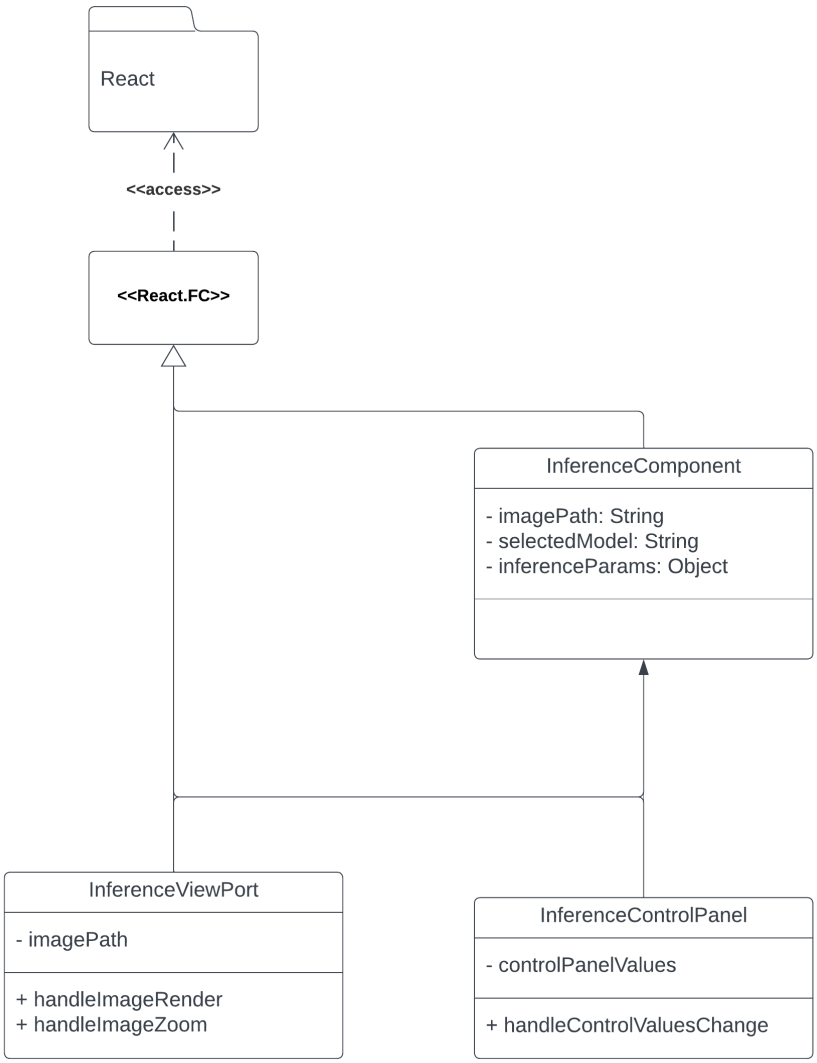

Figure 38: Class Diagram for Frontend
Class diagram for the frontend of the project.

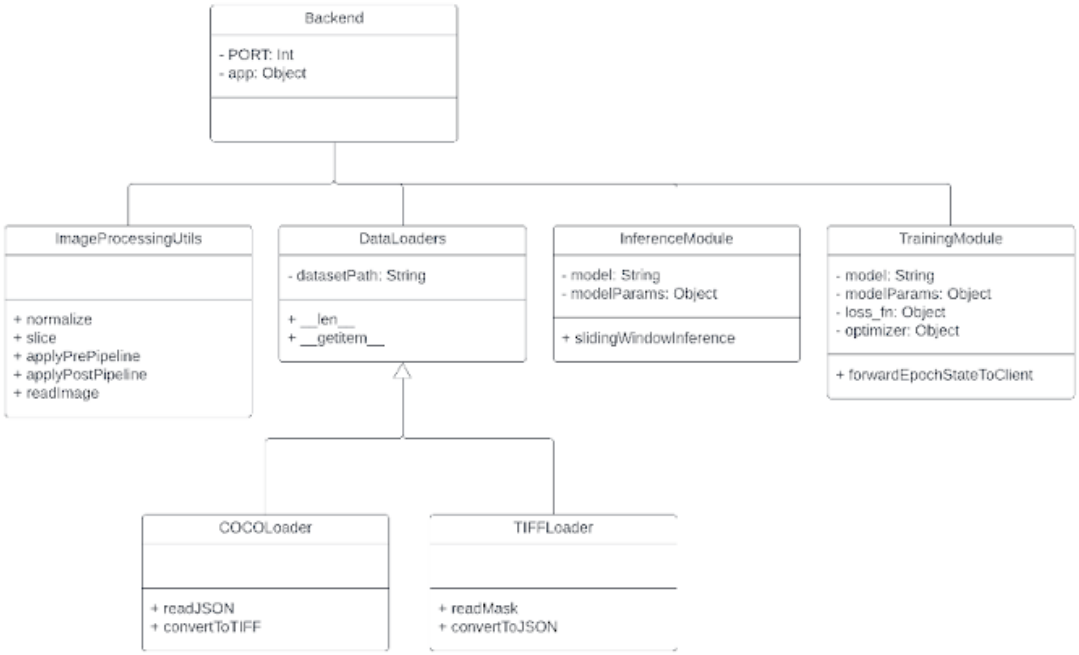

Figure 39: Class Diagram for Backend
Class diagram for the backend of the project.

The class diagram for the application correlates to every major functionality stated for the system. It comprises two individual parts that signify the front-end and the back-end. As the front-end is being developed in React, all the UI components shall inherit from React.FC interface. This interface also accesses some functionality such as state management, state updating, and component lifecycle management from the React package.

6.6 Sequence Diagrams

Following is a list of the sequence diagrams that are present in our web application.

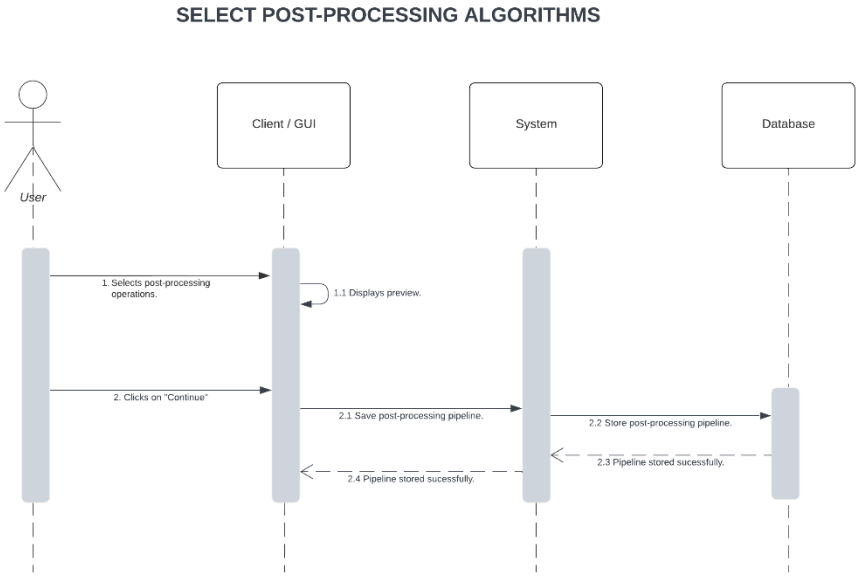

Figure 40: Post-Processing Techniques Selection Sequence Diagram
Diagram that illustrates how the post-processing techniques will be selected.

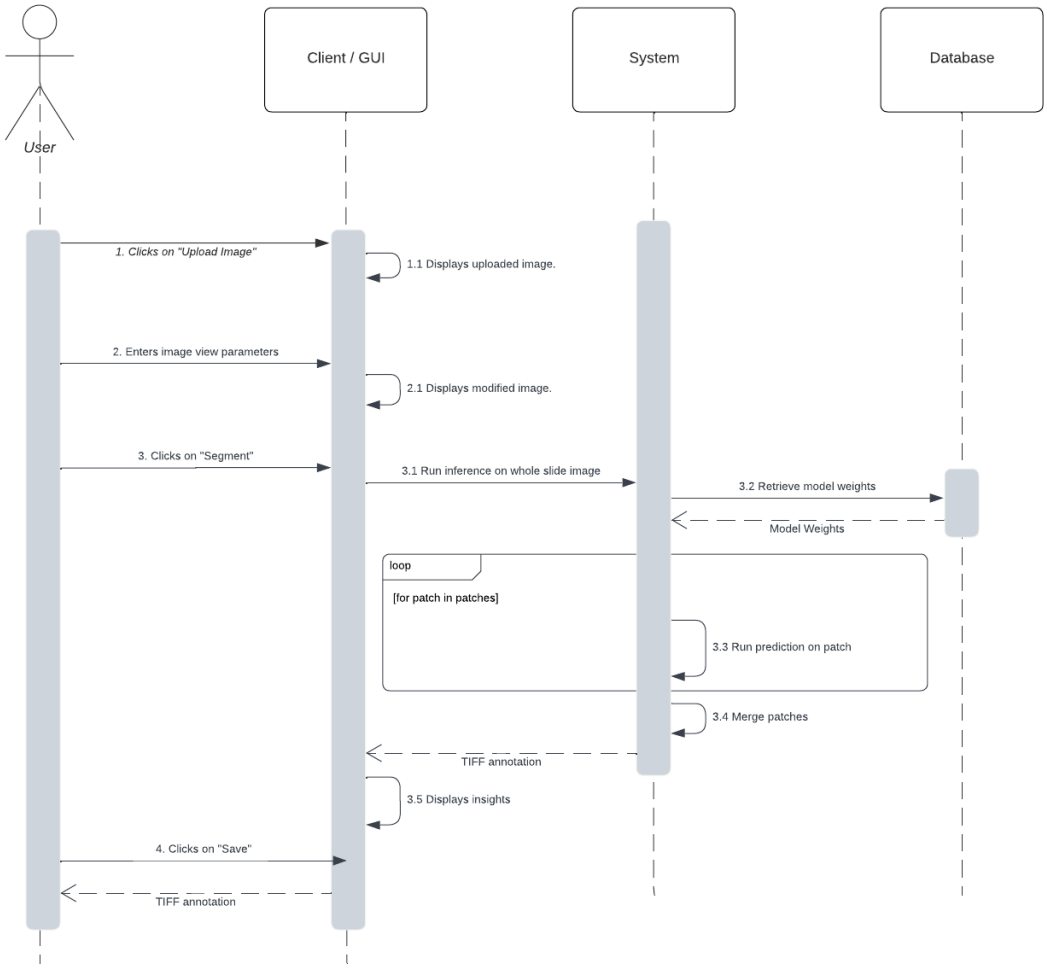

Figure 41: Run Inference Sequence Diagram
Diagram that illustrates how the inference will be run.

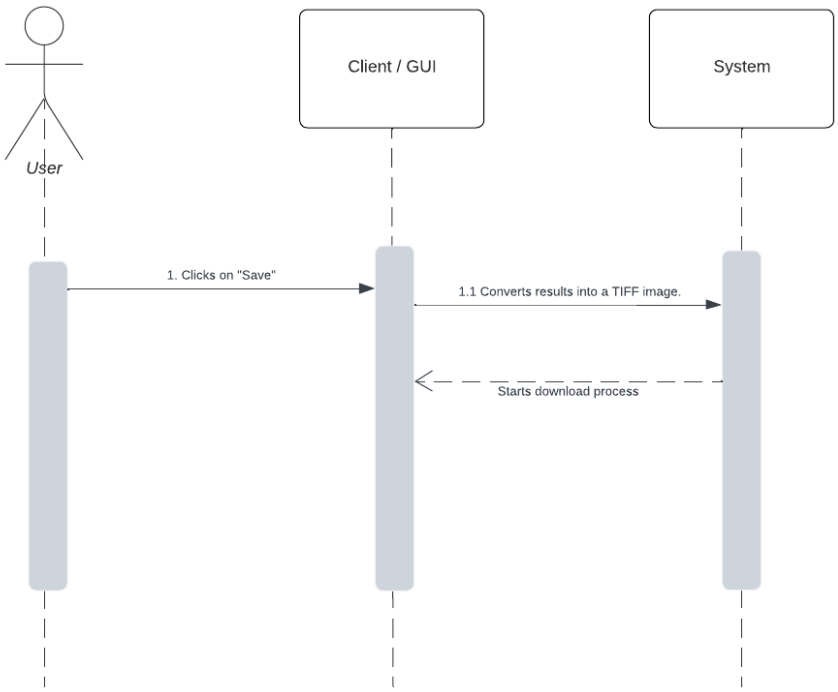

Figure 42: Save Result Sequence Diagram
Diagram that illustrates how the results will be saved.

6.7 Policies and Tactics

To ensure the robustness and efficiency of the final application, the project employs various industry standards and policies. Sufficient research has been conducted to identify applicable policies and tactics for the project. The following policies and tactics shall be used during the course of project development:

6.7.1 Conventions

Since the project deals with two separate components, i.e. backend and frontend. The programming languages shall differ depending on the requirements. In terms of backend conventions, the application shall use PEP-8 standards for software development. For frontend development, a standard functional programming paradigm assisted through JavaScript shall be used to reduce code bloatware. The IDE choice shall be VSCode due to the cross-compatibility and lightweight runtime.

6.7.2 Testing

The application codebase is relatively small and consists of minimal use cases. Consequently, white box testing shall be employed for testing the application functionality. The core focus of performing this testing shall be to evaluate the performance and flexibility of the application.

6.7.3 End-User Interface

As demonstrated in the system architecture diagram, the end-user interface shall be hosted on a client. The client can be a web-based client (which connects with an online API) or a local client that runs an instance on a web browser, however, fetches data from a local API instance.

In such a case, the end-user interface policy shall be to make the website sufficiently responsive to cater for a wide variety of devices and screens.

Chapter 7: Implementation and Test Cases

The following section encompasses the discussion related to the implementation of the project so far along with technical details excluded from the overall high-level design. The content is subject to change as the project continues to evolve.

7.1 Implementation

The team has primarily dedicated time to developing tools and utilities that can assist in conducting experiments to improve existing segmentation algorithms. Additionally, significant work has been done to develop modules that form the crux of the final product such as image processing, normalization, slicing, and inferencing. These modules shall form the backend component which has been explained earlier in the high-level design of the system. Following are the individual modules and the work done in their development:

7.1.1 Image Pre-processing

As described in the previous chapters, image pre-processing is a crucial component in biomedical image processing. Our tool provides multiple standard pre-processing routines that can be employed on the basis of data. Additionally, there are recommended routines that have been shown to improve model performance in a biomedical context. The image pre-processing modules come equipped with an ample battery of pre-processing routines.

As the provided images are in RGB format, they follow a specific pixel value storage scheme, i.e. (R, G, B, A) where R, G, B ∈ {0, 255} and A, which is the alpha value denoting transparency, ∈ [0, 1]. These values need to be normalized within the range of [0, 1] to improve the model's learning process. This is performed through the pixel value scaling functions. It does not change the overall look of the image but helps in the learning process.

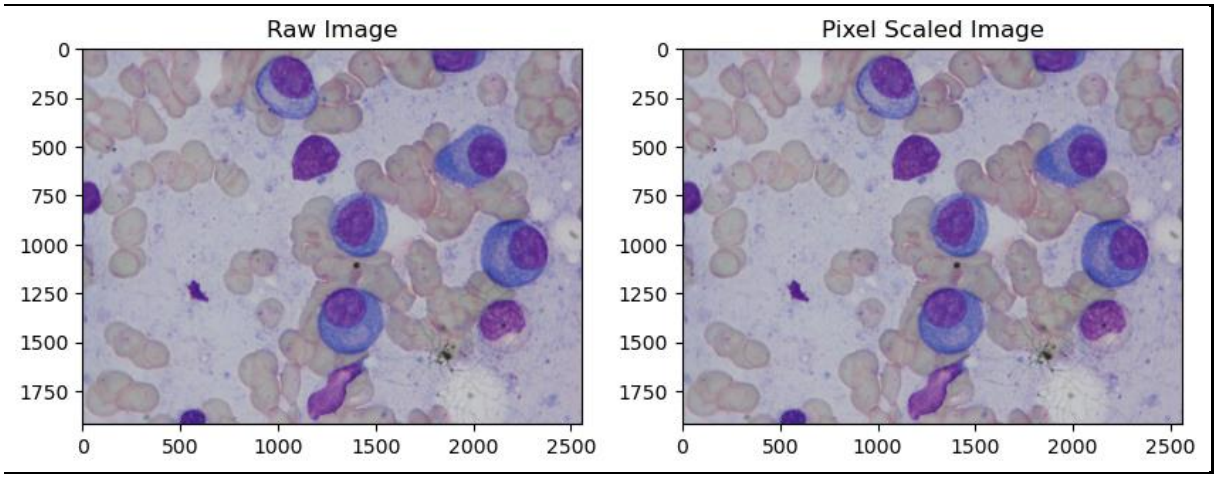

Figure 43: Pixel Scaling

This figure illustrates the microscopic image after pixel scaling. No visual difference.

Once the pixel values are aligned within [0, 1], the next step is to handle the regions with high intensity in the image. These high-intensity regions may cause the model to utilize parameters for accommodating rare artifacts. This is something that can hinder convergence in the long run.

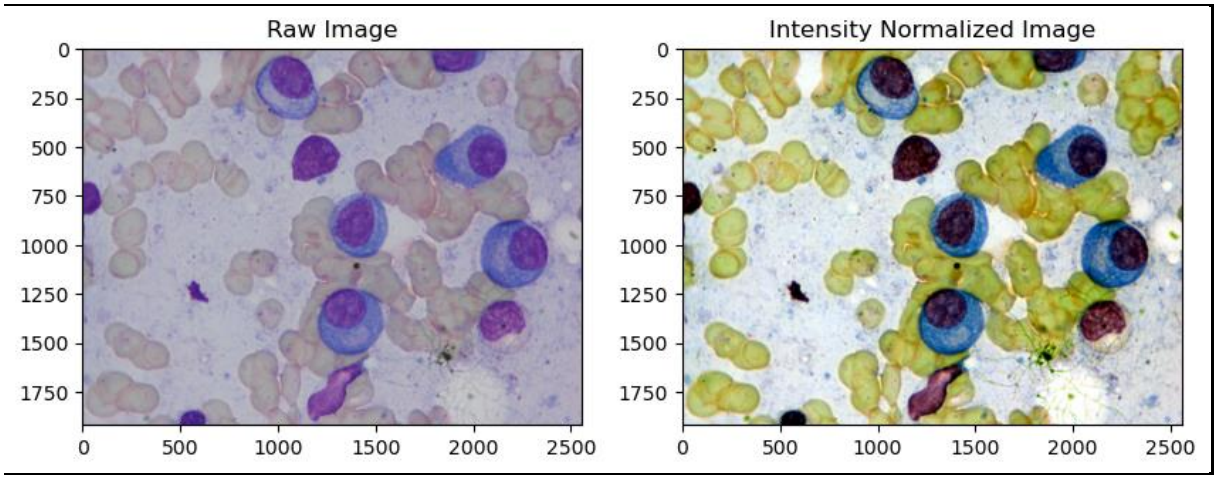

Figure 44: Intensity Scaling

This figure illustrates the microscopic image after intensity scaling.

At runtime, these routines shall be available to the user to toggle and perform a comparative analysis among various options. Moreover, sensible defaults shall also be available for non-technical users to perform inferences without hassle.

```
def normalize_image(self, img_data):
        if len(img_data.shape) == 2:
            img_data  =  np.repeat(np.expand_dims(img_data,  axis=-1),  3,
axis=-1)
        elif len(img_data.shape) == 3 and img_data.shape[-1] > 3:
            img_data = img_data[:,:, :3]
        else:
            pass
        pre_img_data = np.zeros(img_data.shape, dtype=np.uint8)
        for i in range(3):
            img_channel_i = img_data[:,:,i]
            if len(img_channel_i[np.nonzero(img_channel_i)])>0:
                pre_img_data[:,:,i] = self.normalize_channel(
                                        img_channel_i,
                                        lower=1,
                                        upper=99)
        return pre_img_data.astype(np.uint8)
```

7.1.2 Sliding Window Inference

As the project deals with large-scale whole-slide images that can be 10000x10000 pixels in dimension, a module has been implemented to handle the slicing and merging of these whole-slide images. All the data is kept within NumPy arrays for efficient memory management and faster computations. Additional options have been provided to retrieve these patches in the form of torch tensors. These patches can be sent to the model for inferences in a sliding window fashion. Furthermore, options are available for overlap and merging strategies. The functions are exposed through a module-level export in Python.

```
def infer(self, image):
        return sliding_window_inference (
            image,
            roi_size=512,
            sw_batch_size=4,
            predictor=self.model,
```

```
            padding_mode='constant',
            mode='gaussian',
            overlap=0.5
        )
```

7.1.3 Segmentation Module

The application comes with our open-source SOTA segmentation model with multiple options. Currently, we support the following models.

- Light (MANet with SegFormer-0)

All associated model weights shall be stored within the specified data storage. The implementation for MANet with SegFormaer have been sourced from Segmentation-Models-PyTorch [58] and MONAI [59] respectively. All model class interfaces shall be exposed through module-level exports in Python.

7.1.4 Public-facing Inference Endpoint

Given the user has uploaded a whole-slide image to the server, they can access and perform inference on the image through the public inference endpoint. The route is modular and delegates the responsibility to several classes to assist in robust inference.

```
@app.route('/')
def Index():
    labelURL = PredictRoute().handle_route(request)
    return send_file(labelURL, mimetype='image/png')
```

7.1.5 Public-facing Uploading Endpoint

Given the user has uploaded a whole-slide image to the server, they can access and perform inference on the image through the public inference endpoint. The route is modular and delegates the responsibility to several classes to assist in robust inference.

```
@app.route('/upload', methods=['POST'])
def Upload():
    return UploadRoute().handle_route(request)
```

7.1.6 Rule-Based Division

In our exploratory data analysis, we discovered that almost half of the images are single channel images. This presents and opportunity to incorporate simple rule-based learning in which a different model is trained to cater single channel images. Not only would that make inferences faster but also more accurate as the model pipeline does not have to cater to multi-channel images.

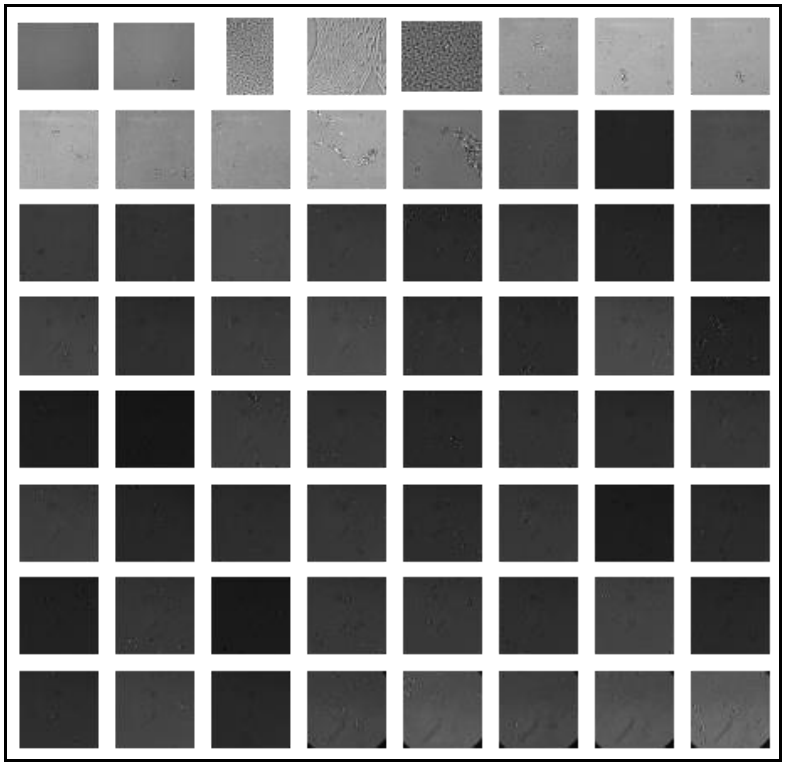

Figure 45: A Sample of Single Channel Images
This figure illustrates a sample of single channel image plotted with cmap='gray' property.

7.1.7 Modality Discovery & Clustering

The model fails to generalize on multiple modality data. To remedy this, we cluster the images into several modalities based on their GLCM features. The features include:

- Contrast
- Dissimilarity
- Homogenity
- Energy
- Correlation
- ASM
- Eccentricity

For clustering, a simple implementation of KMeans is utilized as the data is normalized. However, it was a challenge to determine which features need to be considered for clustering. We applied n-feature brute force for determining ideal subset of features. For now, homogeneity seems to be the most important feature amongst the available ones. While there are some modality mismatches, we discover the elbow point to be at 10 clusters in single channel images.

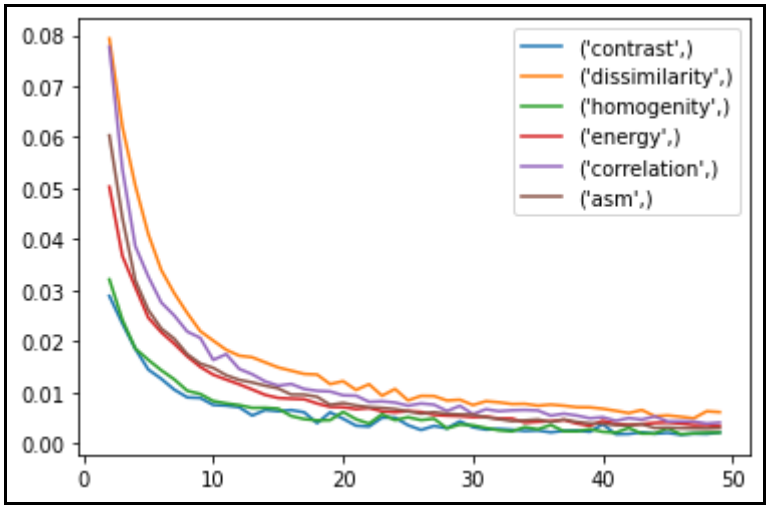

Figure 46: Plot of inertia vs. no. of clusters (Single Feature)
This figure illustrates the overall inertia by selecting a feature and clustering a certain number.

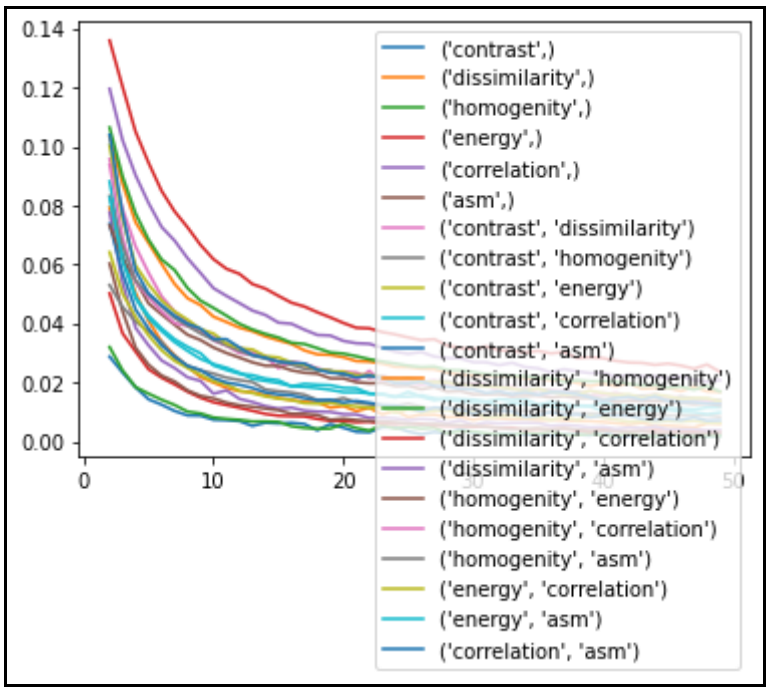

Figure 47: Plot of inertia vs. no. of clusters (Pair-Wise Features)
This figure illustrates the overall inertia by selecting a pair of features and clustering a certain number.

We observed that there are mismatches in certain clusters, however, the color profile seems to be homogeneous with the other images in the cluster. A neural network based feature extractor can be incorporated as well to make better clustering.

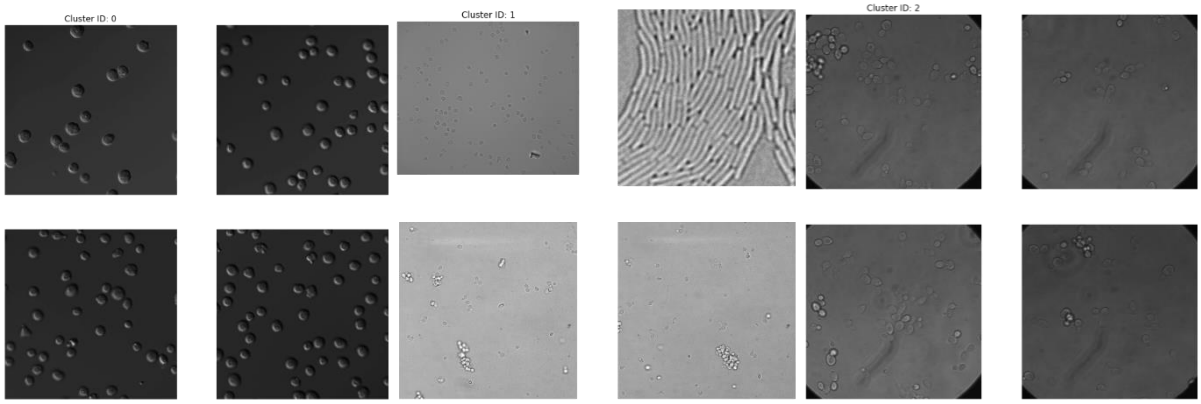

Figure 48: An example of clusters from our dataset
This figure illustrates a sample of 3 clusters from our dataset.

7.1.8 BioSeg

We have researched and implemented various deep learning models for cell segmentation, however, BioSeg serves as a user-friendly tool to run inferences. This tool has been designed to be extremely simple to use and contains only the most important features for the functionality. The frontend has been implemented in React and Tailwind while the backend has been discussed previously.

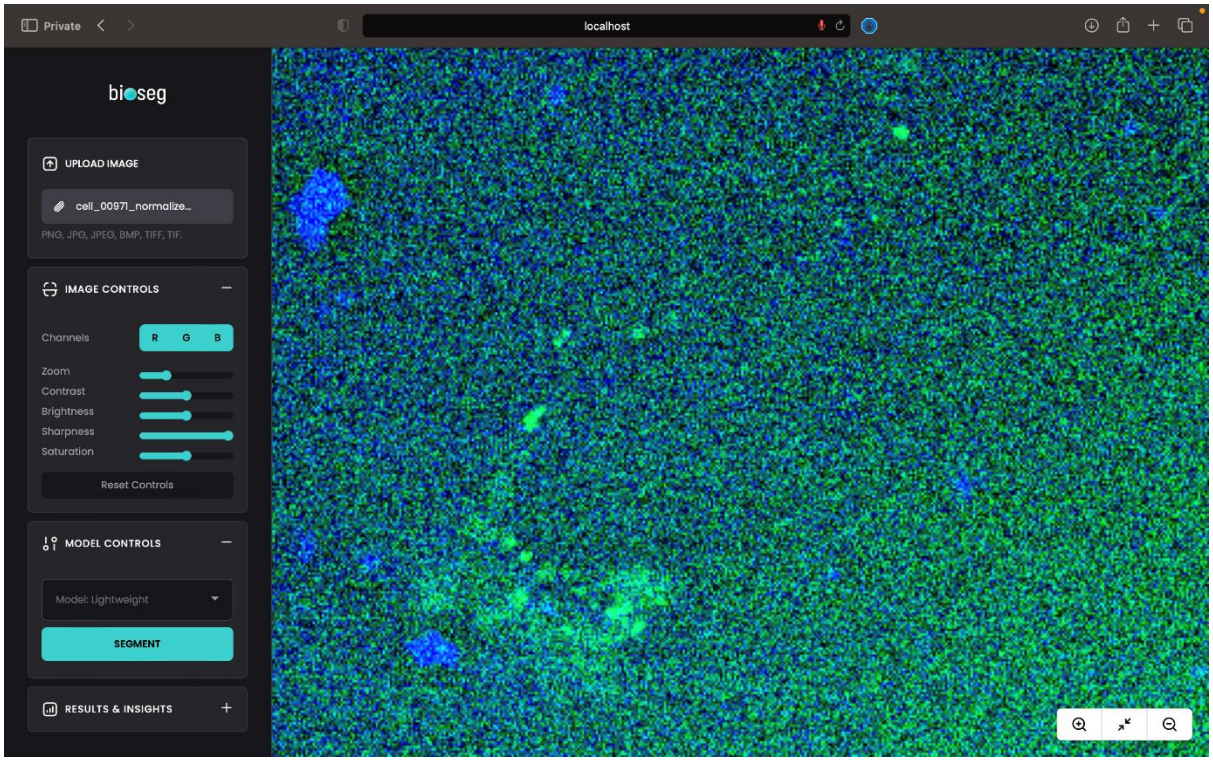

Figure 49: Screenshot of BioSeg Application
This figure illustrates a screenshot of the BioSeg application.

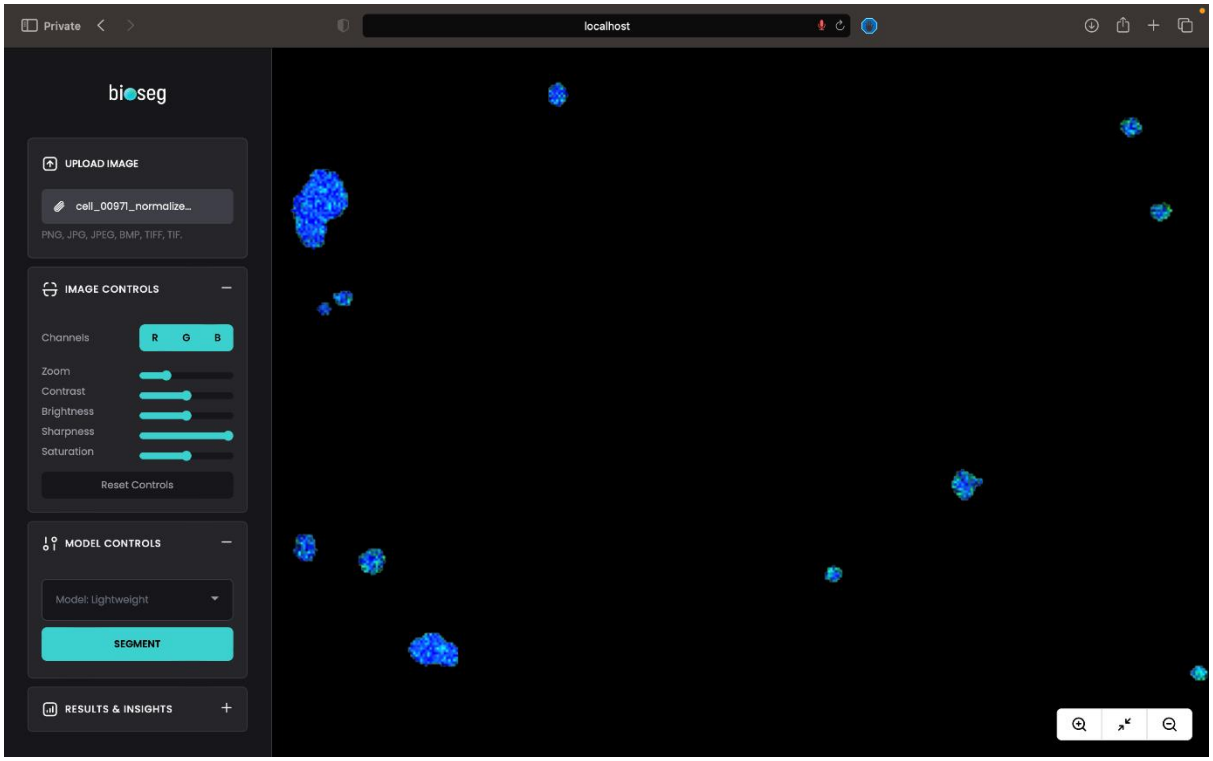

Figure 50: Screenshot of Inference Results BioSeg Application
This figure illustrates a screenshot of the BioSeg application after performing inferences.

7.2 Test Cases

Following are the test cases identified in our project.

7.2.1 Upload Image Test Case No. 1

Run Inferences on Model			
Inference Module			
Test Case ID:	*1*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if an image is uploaded correctly*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*Necessary condition that needs to be fulfilled prior to the test case.*		
Step No.	**Execution description**	**Procedure result**	
1	*The user clicks the 'Upload Image' button and selects an image.*	*The image is shown on the 'Inference Image Preview' panel.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.2 Toggle Channels Test Case No. 2

Run Inferences on Model			
Inference Module			
Test Case ID:	*2*	**QA Test Engineer:**	*Najia Ikhlaq*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if toggling view parameters gives correct results*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be uploaded*		
Step No.	**Execution description**	**Procedure result**	
1	*The user toggles the image view parameters to analyze the image.*	*The selected channels for the image are shown.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.3 Zoom Slider Test Case No. 3

Run Inferences on Model			
Inference Module			
Test Case ID:	*3*	**QA Test Engineer:**	*Najia Ikhlaq*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if an image is displaying the correct zoom percentage.*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be uploaded*		
Step No.	**Execution description**	**Procedure result**	
1	*The user selects the zoom percentage.*	*The system displays the selected zoom percentage.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.4 Contrast Slider Test Case No. 4

Run Inferences on Model			
Inference Module			
Test Case ID:	*4*	**QA Test Engineer:**	*Najia Ikhlaq*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if an image is displaying the correct contrast percentage*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be uploaded*		
Step No.	**Execution description**	**Procedure result**	
1	*The user selects the contrast percentage.*	*The system displays the selected contrast percentage.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.5 Brightness Slider Test Case No. 5

Run Inferences on Model			
Inference Module			
Test Case ID:	*5*	**QA Test Engineer:**	*Najia Ikhlaq*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if an image is displaying the correct brightness percentage*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be uploaded*		
Step No.	**Execution description**	**Procedure result**	
1	*The user selects the brightness percentage.*	*The system displays the selected brightness percentage.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.6 Sharpness Slider Test Case No. 6

Run Inferences on Model			
Inference Module			
Test Case ID:	*6*	**QA Test Engineer:**	*Najia Ikhlaq*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if an image is displaying the correct sharpness percentage*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be uploaded*		
Step No.	**Execution description**	**Procedure result**	
1	*The user selects the sharpness percentage.*	*The system displays the selected sharpness percentage.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.7 Choose Segmentation Model Test Case No. 7

Run Inferences on Model			
Inference Module			
Test Case ID:	*7*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if the correct model is being used.*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*The image has been uploaded.*		
Step No.	**Execution description**	**Procedure result**	
1	*The user selects the segmentation model level.*	*Segmentation will occur using the selected model.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.8 Reset Button Test Case No. 8

Run Inferences on Model			
Inference Module			
Test Case ID:	*8*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if all controls are being reset.*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*The controls are changed from default.*		
Step No.	**Execution description**	**Procedure result**	
1	*The user clicks "Reset Controls".*	*The image controls are reset.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.9 Segment Test Case No. 9

Run Inferences on Model			
Inference Module			
Test Case ID:	*9*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Run Inferences on Model*
Revision History:	-		
Objective	*To test if segmentation results are displayed.*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be uploaded*		
Step No.	**Execution description**	**Procedure result**	
1	*The user clicks the 'Segment' button.*	*The segmentation results are displayed.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.10 Overlay Results Test Case No. 10

Select Post Processing Methods			
Inference Module			
Test Case ID:	*10*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Select Post Processing Methods*
Revision History:	-		
Objective	*To test if insights are displayed*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be segmented*		
Step No.	**Execution description**	**Procedure result**	
1	*The user checks the checkbox for "Overlay Results".*	*The insights are displayed.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.11 Overlay Opacity Test Case No. 11

Select Post Processing Methods			
Inference Module			
Test Case ID:	*11*	**QA Test Engineer:**	*Saqib Ali*
Test case Version:	*1.0*	**Reviewed By:**	*Najia Ikhlaq*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Select Post Processing Methods*
Revision History:	-		
Objective	*To test if an image is displaying the correct overlay opacity percentage*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be segmented*		
Step No.	**Execution description**	**Procedure result**	
1	*The user selects the overlay opacity percentage.*	*The system displays the selected overlay opacity percentage.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.12 Save Results Test Case No. 12

Save Segmentation Results			
Inference Module			
Test Case ID:	*12*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	*Save Segmentation Results*
Revision History:	-		
Objective	*To test if segmentation results are saved*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be segmented.*		
Step No.	**Execution description**	**Procedure result**	
1	*The user clicks the 'Save' button.*	*The inference results are downloaded.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.13 Reliability Test Case No. 13

Reliability Test Case			
Inference Module			
Test Case ID:	*13*	**QA Test Engineer:**	*Saqib Ali*
Test case Version:	*1.0*	**Reviewed By:**	*Najia Ikhlaq*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	-
Revision History:	-		
Objective	*To test if the system gives reliable and valid inferences.*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be segmented.*		
Step No.	**Execution description**	**Procedure result**	
1	*The user segments images.*	*The inference results are consistent and valid across all images.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.2.14 Correctness Test Case No. 14

Correctness Test Case			
Inference Module			
Test Case ID:	*14*	**QA Test Engineer:**	*Ayesha Kanwal*
Test case Version:	*1.0*	**Reviewed By:**	*Saqib Ali*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	-
Revision History:	-		
Objective	*To test if inference results are functionally correct and avoid false positives*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	*An image must be segmented.*		
Step No.	**Execution description**	**Procedure result**	
1	*The user segments an image.*	*The inference results are accurate.*	
Comments: System works properly according to the requirements.			
■*Passed* ☐*Failed* ☐*Not Executed*			

7.2.15 Usability Test Case No. 15

Usability Test Case			
Inference Module			
Test Case ID:	*15*	**QA Test Engineer:**	*Najia Ikhlaq*
Test case Version:	*1.0*	**Reviewed By:**	*Ayesha Kanwal*
Test Date:	*26/04/2023*	**Use Case Reference(s):**	-
Revision History:	-		
Objective	*To test if system is easy to use*		
Product/Ver/Module:	*BioSeg/1.0/Control Panel*		
Environment:	*Web Browser* *Internet connection*		
Assumptions:	-		
Pre-Requisite:	-		
Step No.	**Execution description**	**Procedure result**	
1	*The user opens the application.*	*The interface is user friendly and the user is able to interact with ease.*	

Comments: System works properly according to the requirements.
■*Passed* ☐*Failed* ☐*Not Executed*

7.3 Test Metrics

Following are the test case metrics.

7.3.1 Functional Test Metrics

Metric	Value
Number of Test Cases	12
Number of Test Cases Passed	12
Number of Test Cases Failed	0
Test Case Defect Density	0
Test Case Effectiveness	100
Traceability Matrix	The file is enclosed separately.

7.3.2 Non-Functional Test Metrics

Metric	Value
Number of Test Cases	3
Number of Test Cases Passed	3
Number of Test Cases Failed	0
Test Case Defect Density	0
Test Case Effectiveness	100
Traceability Matrix	The file is enclosed separately.

Chapter 8: User Manual

This chapter includes the installation and setup details, the major features of the application and the application workflow for the users.

8.1 Installation and Setup

The user will have to download and install the application (.exe file). Refer to “Features” and “Application Workflow” for further details.

8.2 Features

The following are the major features of the application.

8.2.1.1 Sliders

A slider is a GUI element that is used to select a value from a range by sliding the thumb along a track. The slider values in our case are from -1 to 1.

8.2.1.1.1 Zoom Slider

Zoom slider is used to increase or decrease the magnification of the uploaded image.

8.2.1.1.2 Contrast Slider

Contrast slider is used to adjust the difference in color, brightness or tone between the lightest and darkest regions of the uploaded image.

8.2.1.1.3 Brightness Slider

Brightness slider is used to adjust the amount of light emitted or reflected by an object which determines its perceived visibility.

8.2.1.1.4 Sharpness Slider

Sharpness slider is used to adjust the clarity and level of detail in the uploaded image.

8.2.1.1.5 Overlay Opacity Slider

This slider is used to determine how transparent, translucent or opaque the produced label would be with respect to the uploaded image.

8.2.1.2 Checkboxes

A checkbox is a GUI element that allows a user to choose one or more options from multiple choices by marking the square box with a tick.

8.2.1.2.1 Overlay Results

This checkbox allows the user to see the Insights.

8.2.1.3 Insights

Insights provide a summary of the label that is produced. It includes the total cells detected, mean cell diameter and the cell size variation.

8.2.1.4 Buttons

A button is a GUI element that the user can click to trigger an action.

8.2.1.4.1 Select Image

Allows the user to choose an image and view it in the Inference Image Preview panel.

8.2.1.4.2 Channels

Allows the user to choose which channel they want to run the inference on. They can choose one or more than one channel.

8.2.1.4.3 Segment Button

Allows the user to run the segmentation on the uploaded image.

8.2.1.4.4 Save Button

Allows the user to save the produced label after segmentation.

8.2.1.4.5 Reset Button

Allows the user to reset all image controls.

8.2.1.5 Dropdown

A dropdown is a GUI element that displays a list of options in a menu that drops down from a button, allowing the user to select an option from it.

8.2.1.5.1 Model Selection

Allows the user to select which model to use from three options.

8.3 Application Workflow

1. User selects an image.
2. User selects the channel(s) they want the segmentation to run on (optional).
3. User adjusts the zoom (optional).
4. User adjusts the contrast (optional).
5. User adjusts the brightness (optional).
6. User adjusts the sharpness (optional).
7. User can reset all image controls (optional).
8. User selects the model they want to work with. Default model is light.
9. User clicks on the segment button to generate the label.
10. User clicks on the save button to save the generated label.
11. User selects if they want to see the overlay results.
    a. User can view the Insights.
12. User can adjust the opacity of the label (optional).

Chapter 9: Experimental Results and Discussion

This chapter highlights the experiments conducted so far along with the empirical and quantitative evaluation of their results. Moreover, a discussion on their overall contribution to the improvement of the previous models has been highlighted as well.

9.1 Experimental Results

The following table shows the results of the experiments that have been carried out so far.

Table 4: Experiments and their F1-Score

This table depicts the F1-Scores obtained through experimentation.

Experiment Name	**Backbone**	**Epochs**	**Test-f1**
Bottom-liner Training	SwinUNETR	10	0.5518
Variable-context Training	SwinUNETR	10	0.5635
Inference-time Augmentation	SwinUNETR	10	0.5686
Single Channel Images	efficientnet-b0	10	0.5049
Single Channel Images	efficientnet-b0	25	0.6633
Single Channel Images	efficientnet-b0	40	0.8505
Single Channel Images	efficientnet-b0	50	0.8720
Single Channel Images with 25% memory replay for Cluster 0	efficientnet-b0	20	0.9097
Single Channel Images with 25% memory replay for Cluster 1	efficientnet-b0	20	0.8078
Single Channel Images with 25% memory replay for Cluster 0	**mit-b0**	**20**	**0.9235**
Single Channel Images with 25% memory replay for Cluster 1	**mit-b0**	**30**	**0.8223**

Table 5: Experiments and their F1-Score

This table depicts the F1-Scores obtained through experimentation.

Experiment Name	**Backbone**	**Epochs**	**Test-f1**
Multiple Channel Images with 25% mix with other data for large-purple cluster	mit-b0	20	0.7212
Multiple Channel Images with 25% mix with other data for large-purple cluster	**mit-b0**	**30**	**0.815**
Multiple Channel Images with 25% mix with other data for large-purple cluster	mit-b1	20	0.6652

Multiple Channel Images with 25% mix with other data for large-purple cluster	mit-b1	30	0.7155
Multiple Channel Images with 25% mix with other data for white-spots and normal-emboss	mit-b0	10	0.8919
Multiple Channel Images with 25% mix with other data for white-spots and normal-emboss	**mit-b0**	**20**	**0.9131**
Multiple Channel Images with 25% mix with other data for white-blotches, white-balls, and white-emboss	mit-b1	20	0.6523
Multiple Channel Images with 25% mix with other data for white-blotches, white-balls, and white-emboss	mit-b1	30	0.7591
Multiple Channel Images with 25% mix with other data for white-blotches, white-balls, and white-emboss	mit-b1	40	0.8601
Multiple Channel Images with 25% mix with other data for white-blotches, white-balls, and white-emboss	**mit-b1**	**50**	**0.8801**
Multiple Channel Images with 25% mix with other data for white-blotches, white-balls, and white-emboss + white-spots and normal-emboss	**mit-b1**	**50**	**0.8716**

9.2 Bottom Liner Training

By running a standard inference on training data, it is evident that some images are hard to segment due to the following attributes:

- Dense presence of cells.
- Hard to distinguish boundaries.
- Contrast issues.

In this experiment, the worst-performing images from the training dataset were filtered with a threshold of F1 $\leq$ 0.3. Then, the standard SwinUNET model was trained on these specific images for 10 epochs. Through this targeted training, the model's segmentation performance increased from 0.5482 (baseline) to 0.5518.

9.3 Memory Replay

We hypothesized that when finetuning a backbone on a single cluster, the model tends to lose its initial feature recognition capabilities and perform quite badly on misclassified images. To remedy this issue, we use memory replay. The idea behind memory replay is to fine-tune the backbone, but also keep a portion (~25%) from other cluster within the batch as well. While the model orients itself to fine-tuned data, it also retains previous important features.

9.4 Inference-Time Augmentations

These augmentations are slightly different from training-time augmentations. In inference-time augmentations, some transformations are applied to the input image. The model runs segmentation on these transformed images. After inference, the inverse of these augmentations is applied to the inferences themselves. A weighted average strategy is applied to untransformed inferences to merge and improve results. The figure below helps visualize the flow:

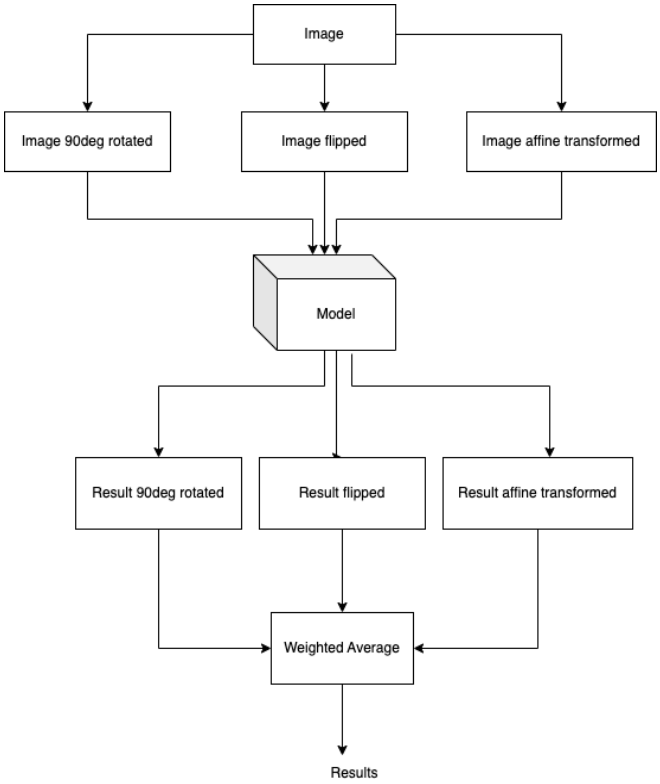

Figure 51: Test-Time Augmentation Pipeline
This figure illustrates a large cell within a 512x512 patch.

This pipeline helped in improved the model's robustness and generalizability. Moreover, it improved the F1 score marginally to 0.5686.

Chapter 10: Conclusion and Future Work

Cell segmentation is an important problem in biomedical research due to its potential in amplifying the diagnostic abilities of pathologists, being foundational to biological studies, and providing an understanding of cell behaviors to refine antibiotics, vaccines, and important drugs. It is an exciting area in biomedical research with real-life applications that impact hundreds of millions of lives. While there are various deep learning solutions available to cater to this problem, all of them fall short due to the scarcity of high-quality annotated data. Previous work has shown excellent results in utilizing limited amounts of data for a variety of tasks through weak-supervision techniques. Moreover, many researchers have also tried to exploit mathematical distributions to map cells into reversible gradient maps which are efficient to compute and highly generalizable. Despite concurrent development in both of these domains, there has been little progress in their amalgamation, i.e. weakly-supervised cell instance segmentation.

Our work focuses on combining, developing, and improving state-of-the-art techniques and solutions available in both domains. In the report, we have successfully covered major literature highlighting the fundamentals of segmentation to modern weak supervision, cell representation, and morphological post-processing techniques. Classic image processing techniques such as water-shedding, region-growing, clustering, and edge-detection-based segmentation were analyzed with their applications in our project. Specifically, their application was considered in a biomedical context to refine cell boundaries and reduce overlap. In later parts of our initial research, deep learning architectures and their specific advantages over other ones are explored to filter out the best attributes. A challenge we faced in this regard was finding literature that sits at the crossroads of both domains, i.e. weak supervision and cell segmentation. Additionally, there is little literature published in terms of general post-processing in variable-sized images. To address these issues, we resorted to utilizing specific literature in each domain and relating their attributes to each other. Several papers have used various methods and collections of operations to address the cell segmentation problem.

Our proposed methodology comprises of the pre-processing and normalization techniques that can be applied to the dataset. It goes over the impact of localization and augmentation on the dataset. It encompasses the basic components of the project i.e., segmentation and weak supervision and lastly, it overgoes the post-processing pipeline for label refinement.

In addition to the research, a GUI to cater to the needs of the pathologist has been devised in an easy-to-use manner. All design considerations, goals, and guidelines that are a part of this web application have been while keeping the basic functionality in mind.

We incorporated various techniques into our dataset and analyzed the impact experienced in terms of overall efficiency and accuracy. We utilized weak-supervision techniques in sync with better cell representation and post-processing to achieve state-of-the-art results in cell segmentation. While our work improved the existing solutions in various ways, there are still many ways forward and improving it even further.

References

[1]Z. Jia, X. Huang, E. Chang and Y. Xu, "Constrained Deep Weak Supervision for Histopathology Image Segmentation", *IEEE Transactions on Medical Imaging*, vol. 36, no. 11, pp. 2376-2388, 2017. Available: 10.1109/tmi.2017.2724070 [Accessed 12 October 2022].

[2]G. Pena and J. Andrade-Filho, "How Does a Pathologist Make a Diagnosis?", *Archives of Pathology & Laboratory Medicine*, vol. 133, no. 1, pp. 124-132, 2009. Available: 10.5858/133.1.124 [Accessed 12 October 2022].

[3]M. Niazi, A. Parwani and M. Gurcan, "Digital pathology and artificial intelligence", *The Lancet Oncology*, vol. 20, no. 5, pp. e253-e261, 2019. Available: 10.1016/s1470-2045(19)30154-8 [Accessed 12 October 2022].

[4]"Cell Segmentation in Multi-modality Microscopy Images - Grand Challenge", *grand-challenge.org*, 2022. [Online]. Available: https://neurips22-cellseg.grand-challenge.org. [Accessed: 12- Oct- 2022].

[5]S. Agrawal and D. Xaxa, "Survey on Image Segmentation Techniques and Color Models", *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 3025-3030, 2014. [Accessed 12 October 2022].

[6]T. Lindeberg and M. Li, "Segmentation and Classification of Edges Using Minimum Description Length Approximation and Complementary Junction Cues", *Computer Vision and Image Understanding*, vol. 67, no. 1, pp. 88-98, 1997. Available: 10.1006/cviu.1996.0510 [Accessed 12 October 2022].

[7]H. Wu and X. Gu, "Max-Pooling Dropout for Regularization of Convolutional Neural Networks", *Neural Information Processing*, pp. 46-54, 2015. Available: 10.1007/978-3-319-26532-2_6 [Accessed 12 October 2022].

[8]A. Hafiz and G. Bhat, "A survey on instance segmentation: state of the art", *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, pp. 171-189, 2020. Available: 10.1007/s13735-020-00195-x [Accessed 12 October 2022].

[9]B. Hariharan, P. Arbeláez, R. Girshick and J. Malik, "Simultaneous Detection and Segmentation", *Computer Vision – ECCV 2014*, pp. 297-312, 2014. Available: 10.1007/978-3-319-10584-0_20 [Accessed 12 October 2022].

[10]K. van de Sande, J. Uijlings, T. Gevers and A. Smeulders, "Segmentation as selective search for object recognition", *2011 International Conference on Computer Vision*, 2011. Available: 10.1109/iccv.2011.6126456 [Accessed 12 October 2022].

[11]Y. Alzahrani and B. Boufama, "Biomedical Image Segmentation: A Survey", *SN Computer Science*, vol. 2, no. 4, 2021. Available: 10.1007/s42979-021-00704-7 [Accessed 12 October 2022].

[12]M. Shah, *Fundamentals of Computer Vision*. Orlando: University of Central Florida, 1997.

[13]F. Yang, S. Wan and Y. Chang, "Improved Method for Gradient-Threshold Edge Detector Based on HVS", *Computational Intelligence and Security*, pp. 1051-1056, 2005. Available: 10.1007/11596448_157 [Accessed 12 October 2022].

[14]L. Su et al., "Delineation of Carpal Bones From Hand X-Ray Images Through Prior Model, and Integration of Region-Based and Boundary-Based Segmentations", *IEEE Access*, vol. 6, pp. 19993-20008, 2018. Available: 10.1109/access.2018.2815031 [Accessed 12 October 2022].

[15]K. Held, E. Kops, B. Krause, W. Wells, R. Kikinis and H. Muller-Gartner, "Markov random field segmentation of brain MR images", *IEEE Transactions on Medical Imaging*, vol. 16, no. 6, pp. 878-886, 1997. Available: 10.1109/42.650883 [Accessed 12 October 2022].

[16]T. Chen and T. Huang, "Region Based Hidden Markov Random Field Model for Brain MR Image Segmentation", *International Journal of Computer and Information Engineering*, vol. 1, no. 4, pp. 1129 - 1132, 2007. Available: doi.org/10.5281/zenodo.1063070 [Accessed 12 October 2022].

[17]A. Hanbury, "Image Segmentation by Region Based and Watershed Algorithms", *Wiley Encyclopedia of Computer Science and Engineering*, 2009. Available: 10.1002/9780470050118.ecse614 [Accessed 12 October 2022].

[18]K. Parvati, B. Prakasa Rao and M. Mariya Das, "Image Segmentation Using Gray-Scale Morphology and Marker-Controlled Watershed Transformation", *Discrete Dynamics in Nature and Society*, vol. 2008, pp. 1-8, 2008. Available: 10.1155/2008/384346 [Accessed 12 October 2022].

[19]H. Ng, S. Ong, K. Foong, P. Goh and W. Nowinski, "Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm", *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*. Available: 10.1109/ssiai.2006.1633722 [Accessed 12 October 2022].

[20]Y. Boykov and M. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images", *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Available: 10.1109/iccv.2001.937505 [Accessed 12 October 2022].

[21]P. Hua, Q. Song, M. Sonka, E. Hoffman and J. Reinhardt, "Segmentation of pathological and diseased lung tissue in CT images using a graph-search algorithm", *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2011. Available: 10.1109/isbi.2011.5872820 [Accessed 12 October 2022].

[22]M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active contour models", *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321-331, 1988. Available: 10.1007/bf00133570 [Accessed 12 October 2022].

[23]A. Amini, T. Weymouth and R. Jain, "Using dynamic programming for solving variational problems in vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855-867, 1990. Available: 10.1109/34.57681 [Accessed 12 October 2022].

[24]S. Minut, G. Stockman and D. Segev, "Interpolation Snakes with Shape Prior for Border Detection in Ultrasound Images", 2008. [Accessed 12 October 2022].

[25]M. Charfi and J. Zrida, "Speed Improvement of B-Snake Algorithm Using Dynamic Programming Optimization", *IEEE Transactions on Image Processing*, vol. 20, no. 10, pp. 2848-2855, 2011. Available: 10.1109/tip.2011.2134857 [Accessed 12 October 2022].

[26]"Correction to Lancet Respir Med 2021; published online April 9. https://doi. org/10.1016/S2213-2600(21)00171-5", *The Lancet Respiratory Medicine*, vol. 9, no. 6, p. e55, 2021. Available: 10.1016/s2213-2600(21)00186-7.

[27]Chunming Li, Chenyang Xu, Changfeng Gui and M. Fox, "Distance Regularized Level Set Evolution and Its Application to Image Segmentation", *IEEE Transactions on Image Processing*, vol. 19, no. 12, pp. 3243-3254, 2010. Available: 10.1109/tip.2010.2069690 [Accessed 12 October 2022].

[28]T. Cootes, C. Taylor, D. Cooper and J. Graham, "Active Shape Models-Their Training and Application", *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38-59, 1995. Available: 10.1006/cviu.1995.1004 [Accessed 12 October 2022].

[29]K. Van Leemput, F. Maes, D. Vandermeulen and P. Suetens, "Automated model-based tissue classification of MR images of the brain", *IEEE Transactions on Medical Imaging*, vol. 18, no. 10, pp. 897-908, 1999. Available: 10.1109/42.811270 [Accessed 12 October 2022].

[30]G. Wang et al., "Interactive Medical Image Segmentation Using Deep Learning With Image-Specific Fine Tuning", *IEEE Transactions on Medical Imaging*, vol. 37, no. 7, pp. 1562-1573, 2018. Available: 10.1109/tmi.2018.2791721 [Accessed 12 October 2022].

[31]R. Zhang et al., "Automatic Segmentation of Acute Ischemic Stroke From DWI Using 3-D Fully Convolutional DenseNets", *IEEE Transactions on Medical Imaging*, vol. 37, no. 9, pp. 2149-2160, 2018. Available: 10.1109/tmi.2018.2821244 [Accessed 12 October 2022].

[32]Z. Akkus, A. Galimzianova, A. Hoogi, D. Rubin and B. Erickson, "Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions", *Journal of Digital Imaging*, vol. 30, no. 4, pp. 449-459, 2017. Available: 10.1007/s10278-017-9983-4 [Accessed 12 October 2022].

[33]Hyunjin Park, P. Bland and C. Meyer, "Construction of an abdominal probabilistic atlas and its application in segmentation", *IEEE Transactions on Medical Imaging*, vol. 22, no. 4, pp. 483-492, 2003. Available: 10.1109/tmi.2003.809139 [Accessed 12 October 2022].

[34]J. Long, T. Darrell and E. Shelhamer, "Fully Convolutional Networks for Semantic Segmentation", 2015. Available: https://arxiv.org/abs/1411.4038. [Accessed 12 October 2022].

[35]O. Ronneberger, T. Brox and P. Fischer, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015. Available: https://arxiv.org/abs/1505.04597. [Accessed 12 October 2022].

[36]M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury and C. Pal, "The Importance of Skip Connections in Biomedical Image Segmentation", *Deep Learning and Data Labeling for Medical Applications*, pp. 179-187, 2016. Available: 10.1007/978-3-319-46976-8_19 [Accessed 12 October 2022].

[37]H. Cholakkal, G. Sun, F. Khan and L. Shao, "Object Counting and Instance Segmentation with Image-level Supervision", 2019. Available: https://arxiv.org/abs/1903.02494. [Accessed 12 October 2022].

[38]H. Cholakkal, G. Sun, F. Khan and L. Shao, "Object Counting and Instance Segmentation with Image-level Supervision", 2019. Available: https://arxiv.org/abs/1903.02494. [Accessed 12 October 2022].

[39]I. Laradji, D. Vazquez and M. Schmidt, "Where are the Masks: Instance Segmentation with Image-level Supervision", 2019. Available: https://arxiv.org/pdf/1907.01430.pdf. [Accessed 12 October 2022].

[40]S. Berg et al., "ilastik: interactive machine learning for (bio)image analysis", *Nature Methods*, vol. 16, no. 12, pp. 1226-1232, 2019. Available: 10.1038/s41592-019-0582-9 [Accessed 12 October 2022].

[41]N. Greenwald et al., "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning", *Nature Biotechnology*, vol. 40, no. 4, pp. 555-565, 2021. Available: 10.1038/s41587-021-01094-0 [Accessed 12 October 2022].

[42]M. Weigert, U. Schmidt, R. Haase, K. Sugawara and G. Myers, "Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy", *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. Available: 10.1109/wacv45572.2020.9093435 [Accessed 12 October 2022].

[43]"Cellpose: deep learning-based, generic cell segmentation", *Wiley Analytical Science*, 2021.

[44]K. Cutler, C. Stringer, P. Wiggins and J. Mougous, "Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation", 2021. Available: 10.1101/2021.11.03.467199 [Accessed 12 October 2022].

[45]D. Kysela, A. Randich, P. Caccamo and Y. Brun, "Diversity Takes Shape: Understanding the Mechanistic and Adaptive Basis of Bacterial Morphology", *PLOS Biology*, vol. 14, no. 10, p. e1002565, 2016. Available: 10.1371/journal.pbio.1002565 [Accessed 12 October 2022].

[46]S. Stylianidou, C. Brennan, S. Nissen, N. Kuwada and P. Wiggins, "SuperSegger: robust image segmentation, analysis and lineage tracking of bacterial cells", *Molecular Microbiology*, vol. 102, no. 4, pp. 690-700, 2016. Available: 10.1111/mmi.13486 [Accessed 12 October 2022].

[47]K. Cutler, "GitHub - kevinjohncutler/omnipose: Omnipose: a high-precision solution for morphology-independent cell segmentation", *GitHub*, 2022. [Online]. Available: https://github.com/kevinjohncutler/omnipose. [Accessed: 12- Oct- 2022].

[48]H. Wu, N. Souedet, C. Jan, C. Clouchoux and T. Delzescaux, "A general deep learning framework for neuron instance segmentation based on Efficient UNet and morphological post-processing", *Computers in Biology and Medicine*, p. 106180, 2022. Available: 10.1016/j.compbiomed.2022.106180 [Accessed 12 October 2022].

[49]C. Stringer, T. Wang, M. Michaelos and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation", *Nature Methods*, vol. 18, no. 1, pp. 100-106, 2020. Available: 10.1038/s41592-020-01018-x [Accessed 12 October 2022].

[50]V. Caselles, F. Catte, T. Coll and F. Dibos, "A geometric model for active contours in image processing", *Numerische Mathematik*, vol. 66, no. 1, pp. 1-31, 1993. Available: 10.1007/bf01385685 [Accessed 12 October 2022].

[51]"mafa-dataset", *Kaggle.com*, 2022. [Online]. Available: https://www.kaggle.com/datasets/revanthrex/mafadataset. [Accessed: 12- Oct- 2022]

[52]"Mask R-CNN", *Ieeexplore.ieee.org*, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/8372616. [Accessed: 12- Oct- 2022]

[53]"R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms", *Medium*, 2022. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed: 12- Oct- 2022]

[54]R. Girshick, J. Malik, J. Donahue and T. Darrell, "Rich feature hierarchies for accurate object detection and semantic segmentation", 2014. Available: https://arxiv.org/abs/1311.2524. [Accessed 12 October 2022].

[55]Y. Zhou, J. Jiao, Y. Zhu, Q. Ye and Q. Qiu, "Weakly Supervised Instance Segmentation using Class Peak Response", 2018. Available: https://arxiv.org/abs/1804.00880. [Accessed 12 October 2022].

[56]E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 4, pp. 34–41, 2001.

[57] H. Kang, D. Luo, W. Feng, S. Zeng, T. Quan, J. Hu, and X. Liu, "Stainnet: A fast and robust stain normalization network," *Frontiers in Medicine*, vol. 8, 2021.

[58] Qubvel, "Qubvel/segmentation_models.Pytorch: Segmentation models with pretrained backbones. pytorch.," *GitHub*. [Online]. Available: https://github.com/qubvel/segmentation_models.pytorch. [Accessed: 05-Dec-2022].

[59] "Monai," *MONAI*. [Online]. Available: https://monai.io/. [Accessed: 05-Dec-2022].

Appendix

10.1 Appendix A: Dataset

The data for this project is a set of 1000 labelled and 1500 unlabelled WSI sourced from the NeurIPS Grand Challenge “Weakly Supervised Cell Segmentation in Multi-modality High-Resolution Microscopy Images” [4]. The dataset is divided into four microscopic modalities with variable cell shapes and sizes. The image formats include TIFF, JPG, and PNG.

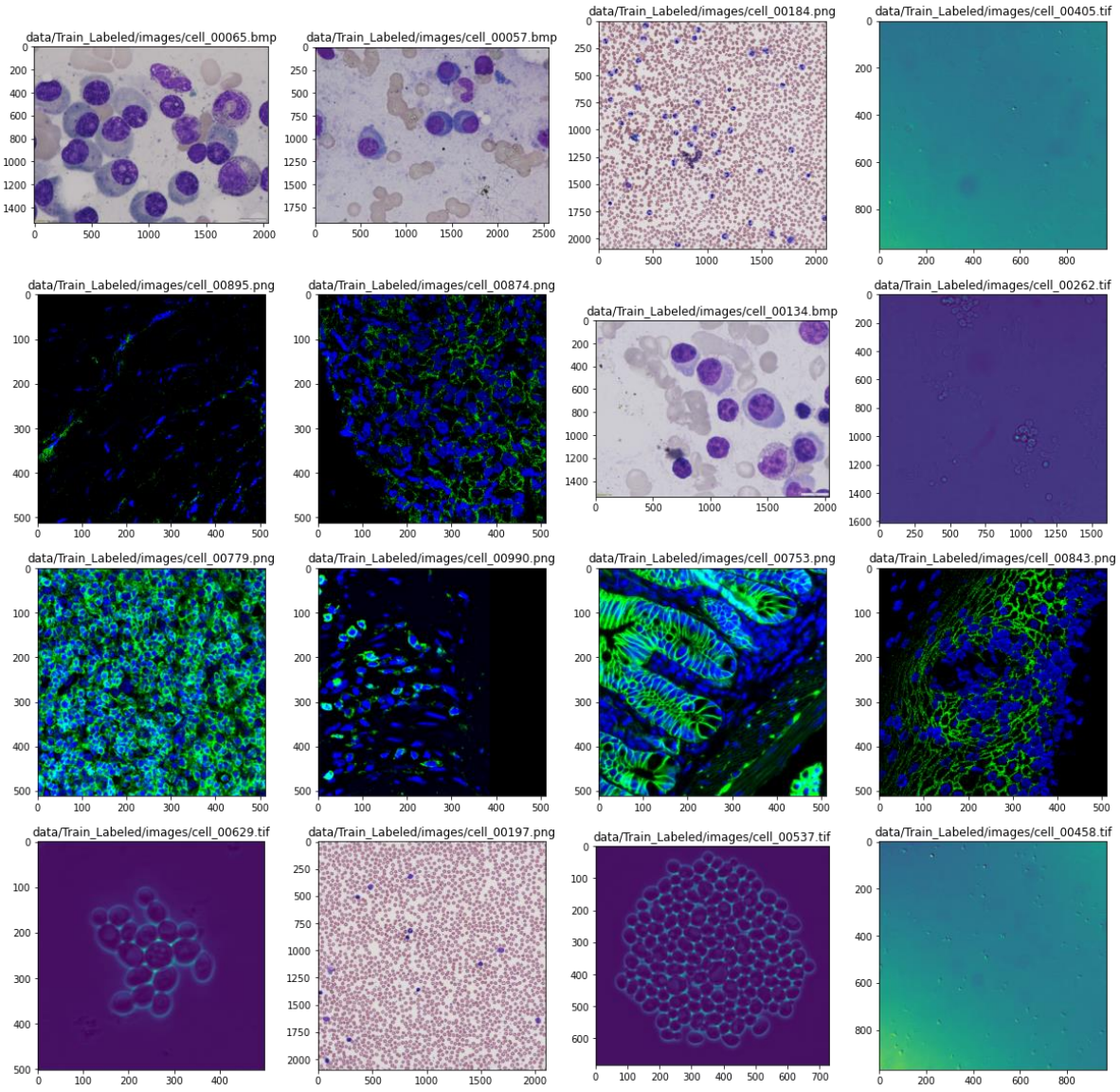

Figure 52: Dataset Sample

This figure represents a random sample from the dataset.

While the modalities are mentioned along with their image count, the dataset has no modality-specific information associated with the images themselves.

Table 5: Dataset Composition
This table depicts the dataset composition and how many patches have been released for each modality.

Modality	Patches
Brightfield	300
Fluorescent	300
Phase-Contrast	200
Differential Interference Contrast (DIC)	200

The dataset contains images with variable resolutions and sizes. Furthermore, the aspect ratios are different as well. In our observations, we found that the mode image dimensions are roughly 512x512, which we show as a red point in the figure below. As observed in the plot, we cannot rely on mean because it is skewed due to some outliers.

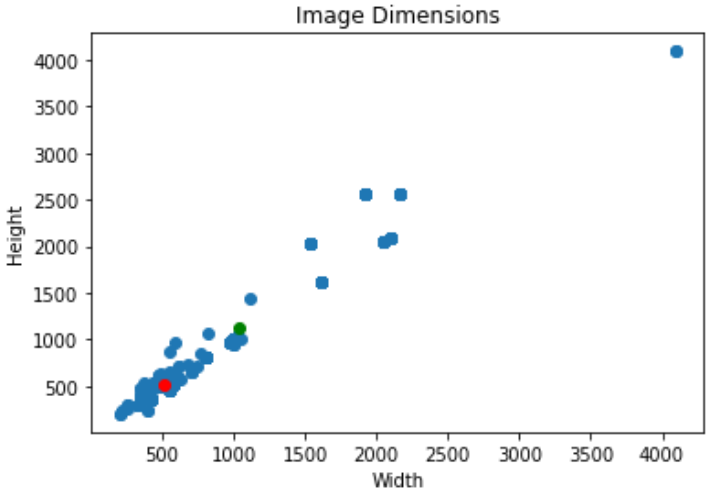

Figure 53: Dataset Dimensions
This figure represents the dataset dimensions. R = Mode. G = Mean.

The challenge also provides 101 WSI in the form of a validation and tuning set. This set is used for inferences and benchmarking user models. Additionally, the competition will also release a test set containing more than 200 images for the final benchmarking of competition models.