University of California, Merced

A High-throughput Pipeline for the Analysis and Annotation of Sectored Yeast Colonies

A Capstone submitted in partial satisfaction of the requirements for the degree of Master of Science

in

Applied Mathematics

by

JORDAN COLLIGNON

Committee in charge:
Professor Suzanne Sindi, Chair
Professor Erica Rutter

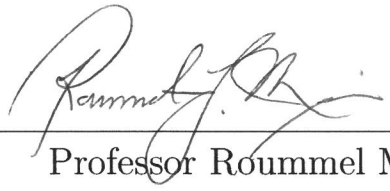
2022

This is to certify that I have examined a copy of a technical report by

Jordan Collignon

and found it satisfactory in all respects, and that any and all revisions required by the examining committee have been made.

Applied Mathematics
Graduate Studies Chair:

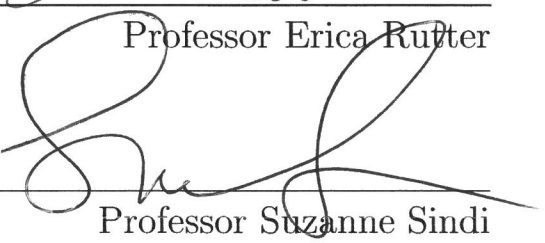


Professor Roummel Marcia

Reading Committee:

Professor Erica Rutter

Committee Chair / Research
Advisor:



Professor Suzanne Sindi

12/5/2022

Date

Contents

List of Symbols

p	p-value for hypothesis testing
α	significance level for hypothesis testing
$[R, G, B]$	vector of intensities for the red, green, and blue channels respectively
R_i	the i^{th} red region of a colony
W_i	the i^{th} white region of a colony
a	the total number of red regions for a given colony
b	the total number of white regions for a given colony
$N(R_i, red), N(R_i, white)$	the number of red or white pixels respectively inside the i^{th} red region of a colony.
$N(W_i, red), N(W_i, white)$	the number of red or white pixels respectively inside the i^{th} white region of a colony.
$p(R_i, red), p(R_i, white)$	function defining the purity of region R_i in a given colony with respect to the red or white pixels respectively in that region.
$p(W_i, red), p(W_i, white)$	function defining the purity of region W_i in a given colony with respect to the red or white pixels respectively in that region.
$\mu(R_i), \mu(W_j)$	weight function denoting the proportion of pixels in the colony occupied by region R_i and region W_j respectively.
p_w	function denoting the weighted purity of a colony with respect to both red and white regions.

List of Figures

Thank you to Dr. Tricia Serio and Dr. Wesley Naeimi at the University of Massachusetts, Amherst for providing the experimental data necessary for the work detailed in this capstone. I also want to thank the National Science Foundation and the National Institutes of Health for the grant funding that supported this work, as well as the UC Merced Graduate Division for the Fletcher Jones Fellowship which funded me during the period of this capstone's completion.

A High-throughput Pipeline for the Analysis and Annotation of Sectored Yeast Colonies

by

Jordan Collignon

Master of Science in Applied Mathematics

Suzanne Sindi, Committee Chair

University of California, Merced

2022

Abstract

Prion proteins are commonly associated with fatal neurodegenerative diseases in mammals, but are also responsible for a number of harmless heritable phenotypes in the yeast *Saccharomyces cerevisiae*. In normal conditions, circular yeast colonies exhibit a prion phenotype, displaying a white, pink, or red color related to the fraction of normal (non-prion) protein. While in mammals prion phenotypes are irreversible, in yeast mild experimental manipulations destabilize prion phenotypes, introduce changes in the intracellular prion aggregation dynamics, and cause colonies to exhibit sectors showing both prion (white or pink) and non-prion (red) phenotypes. The precise mechanism of this destabilization and forces influencing the emergence of mixed colony phenotypes are unknown.

Images of experimental colonies provide a rich dataset for characterizing the unknown molecular mechanisms influencing destabilization of prion phenotypes and uncovering relationships between colony-level phenotypic transitions, molecular processes, and individual cell behaviors. However, this rich diversity of data is often ignored in practice in traditional biological pipelines, both because colony counting is labor intensive and procedures for characterizing sectored colonies are scarce. A computational pipeline for studying large quantities of experimental yeast colony phenotypes would facilitate the use of existing unmined data to explore these relationships.

In this capstone, I build the first computational pipeline designed for providing a deep analysis of sectored yeast colonies in experimental image data. I employ a deep learning strategy that includes synthetic images of yeast colonies during the training process to aid in the extraction of yeast colonies from experimental data, then develop a post-processing procedure to annotate and classify each colony extracted. At present, this pipeline correctly predicts the frequency of sectors in approximately 91.4% of colonies detected in hand annotated experimental images. Furthermore, this pipeline is able to categorize plates containing more sectored colonies than uniform (fully white or red) colonies, allowing a way to distinguish between plates based on the outcome of experiments on yeast. This approach will streamline quantification and annotation of yeast colonies grown under experimental conditions and offer additional insights into mechanisms driving colony-level phenotypic transitions.

Chapter 1

Introduction and Background

1.1 Prions and Yeast

Prion diseases are a class of fatal and incurable neurodegenerative diseases in mammals. About one in every one million humans are afflicted by a type of prion disease which includes Creutzfeldt-Jacob disease, fatal familial insomnia, Gerstmann-Straussler-Scheinker syndrome, and Kuru [46]. One of the first prion diseases studied in great detail was scrapie in sheep [21]. Early research by Stanley Prusiner [45, 46] revealed that a protein–not a virus–coined as a proteinacious infectious particle–or prion–was the key infectious agent causing scrapie. This finding established what is known as the prion hypothesis, which suggests misfolded proteins are associated with all types of prion disease [42, 62], regardless of the mammalian host. These misfolded proteins act as templates that can induce normally folded proteins of the same type to misfold [25, 26, 47, 57] (see Figure 1.1). Furthermore, these misfolded proteins can merge to form aggregates [42] which can grow in size or be fragmented into smaller aggregates to induce further misfolding, thus leading to a self-replicating infection process [14, 25]. Since the formalization of the prion protein [47], the study of biological processes behind prion disease and the search for appropriate solutions to eradicate them remains an active area of research.

Prion proteins are not exclusive to mammals. The yeast *Saccharomyces cerevisiae* has served as a model system to understand more about the spread of human diseases, including prion-like diseases such as Alzheimers [3, 6, 56]. There are at least eight naturally occurring prion proteins [11, 36, 63], setting the stage for yeast to help screen potential anti-prion drug candidates [27]. One of the most widely studied prion forming proteins in yeast is Sup35 which is essential release factor in translation-termination [37,59]. However, toxicity can occur when Sup35 is overexpressed, causing the proteins to misfold and merge with other misfolded proteins to form aggregates [27, 37, 59]. Furthermore, similar to mammalian prions, Sup35 aggregates have the ability to self-propagate within yeast cells [42], resulting in cells exhibiting the white ($[PSI^+]$) phenotype. In contrast, cells that only contain the non-prion form exhibit the red ($[psi^-]$) phenotype.

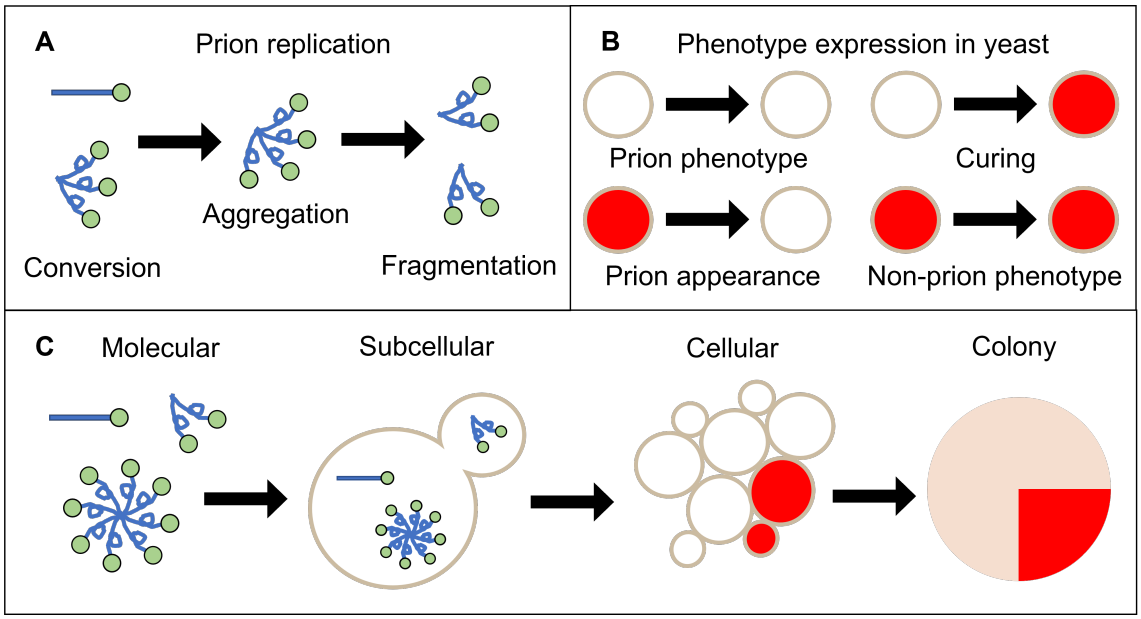


Figure 1.1: **Yeast prion phenotypes are the result of multiscale processes.** A: At the molecular scale, misfolded proteins (twisted) act as templates that convert normally folded proteins (straight) into the misfolded form which can merge to form aggregates. The aggregates can then split into smaller segments (fragmentation) which increases the number of aggregates.
B: At the cellular scale, the presense of prion aggregates inside individual cells (represented as circles) is responsible for their phenotype (color). The presence of prion aggregates inside a cell are responsible for its white color, while the absence of prions gives it its red color. These phenotypes are commonly inherited from the mother to the daughter cell, but that is not always the case. The prion phenotype can be lost sporadically, resulting in cured cells. In rare instances the prion phenotype can be acquired via a spontaneous misfolding event occurring with a protein within a cell, which in turn converts the phenotype of that cell from red to white.
C: Phenotype expression in yeast phenotype involves multiscale processes. The dynamics inherent in protein misfolding are found at the molecular level (A). At the subcellular level, since prions are also found in yeast which undergo their own process of reproduction, cells can also transmit prions between them. At the cellular level, the presense of prions within a cell in turn determines their phenotype (B). At the colony level, the collection of intercellular interactions that occur on the scale of a cell results in structured regions of one phenotype within the colony.

Remarkably, unlike their human counterparts, the $[PSI^+]$ phenotype in yeast is reversible [22, 35], and such switches have been observed in experiments [31, 52]. The event that a cell originally exhibiting the $[psi^-]$ phenotype spontaneously switches to the $[PSI^+]$ phenotype is rare, occurring in approximately one in every 10^6 cell divisions [22]. The switch from the $[PSI^+]$ to the $[psi^-]$ phenotype is known to occur sporadically [22] as well as occur under heat shock as demonstrated in [31], giving rise to colonies with both phenotypes resembling sectors. Figure 1.1 B and C summarizes the possible events that can determine the phenotype of newly born cells, and how the collective expression of all cells in a colony give rise to sectored phenotypes at the colony level. One interesting observation about sectored colonies is that the $[PSI^+]$ and $[psi^-]$ regions independently exhibit uniformity, suggesting this partial $[PSI^+]$ curing results in well-defined spatial structures in yeast colonies. This type of behavior is also observed as a result of natural biological mechanisms in other yeasts such as *Candida albicans* which exhibit sectors related to the white-opaque phenotypes [1, 7, 18]. Given that we hypothesize that the $[PSI^+]$ phenotype is reversible, it is not clear why $[PSI^+]$ and $[psi^-]$ colonies appear spatially organized rather than well-mixed. Furthermore, while there exist models that are capable of tracking phenotypes in microbial colonies [40], to our knowledge models which specifically assign colony phenotypes under the same conditions in experiments are scarce. In addition, we lack an understanding of how the cumulative effect of intercellular interactions impact sectoring behavior in the context of protein misfolding in yeast. As such, it is important to design a framework for understanding the interplay between prion dynamics and sectoring behavior observed in experiments in order to uncover information about the underlying mechanisms involved in protein misfolding.

To understand the dynamics of prions in yeast, we need to consider what is happening at different spatial scales (Figure 1.1 C). The dynamics of prions occur at the molecular scale where conversion and aggregation of existing proteins can be observed [30, 41]. Mathematical models have been proposed that explore these dynamics [15, 38, 54] with one recently proposed to explore how multiple prion strains interact [34]. However, this model would be taking place inside individual cells of yeast, each with their own biological properties such as division which can result in the transfer of molecular material between attached mother-daughter cell pairs. At the scale of a cell, experiments can be used to track the presence and the abundance of prion proteins by looking at a cell's phenotype. As cells continue to divide over time to form a colony of thousands to millions of individual cells, phenotypic segregation can be observed [31] indicating where subsequent daughter cells did not inherit misfolded proteins. Thorough understanding of these processes may require large samples of yeast colonies under different experimental settings. This leads to two potential problems. First, the same experimental settings can still produce lots of variation in the observed output of the experiment. Second, experimentalists may be required to analyze each colony under experimental procedures manually, which is an extremely tedious process which grows more intense as the number of colonies increases. Furthermore, since these colonies are often too small for performing de-

tailed phenotypic analyses, use of suitable instruments may be necessary to acquire the desired information at the colony scale.

1.2 Advances in Image Processing

As computational tools advance over time, it is possible to use efficient approaches involving large amounts of data that are either very complex or too large for conventional storage. This data can take the form of digital images which can be described as structured representations of visual perspectives. When performing image analysis tasks, this data is often represented in two-dimensional spatial structures consisting of pixels that store information such as color or intensity of each location in the image. In computer vision, one of the common objectives is to efficiently leverage this information in order to recognize and isolate regions of interest in an image. This is a process known as image segmentation, which is to partition an image into a set number of groups such that each group meets specific criteria. More specifically, it is a process of partitioning an image into regions that each have common properties [28].

Since images can be large, complex, and problem-specific, different segmentation methods are required for locating objects of interest. One method–namely instance segmentation–is the problem of partitioning an image by locating and quantifying the number of desired individual objects present. An example of this is first performing edge detection [9] to obtain boundary information, then feature or pattern extraction to locate groups of edges which exhibit a desired structure such as circles [2]. Another method–namely semantic segmentation–is the problem of assigning a label to every pixel in an image [19], where the number of labels considered are the number of components that partition the entire image. This is used to find which pixels belong to objects of interest and those that do not. One of the simplest tools for semantic segmentation is thresholding, where an image is partitioned into distinct components usually based on a range of brightness or intensity. Different thresholding and region-based algorithms have been proposed [55] such as Otsu's method [43] to place objects of interest into the foreground. However, many of these algorithms are performed on single-channel or grayscale images, so any color images must first be transformed to the grayscale representation to be applicable. This results in significant loss of information and previously distinguishable objects by eye in color images may become indistinguishable in grayscale images. To avoid this issue, one can consider color-based image segmentation schemes [12] or turn to more data-driven approaches using neural networks where predictions can be made on images.

Neural networks have been developed over the years to not only address the generalizability issue that many traditional methods possess, but also to significantly improve efficiency of segmentation tasks. For image data, one of the most common approaches is to build and train a convolutional neural network (CNN) that could successfully detect and segment objects of interest that would otherwise be tedious or impossible to detect using traditional methods. A CNN is a type of neural network that contains at least one convolutional layer which applies a convolution operation

to an input image with a filter. As more convolutional layers are applied, a CNN can learn how to segment images using higher-level features. Like most neural networks, a CNN predicts labels with the help of a series of weights inherent in the network which are tuned through a process of "training". Ghosh et al [20] mentioned that two benefits of using a CNN are that each weight works with every pixel in an image, rather than having one weight per pixel, and that weights are shared between two adjacent convolutional layers. This significantly reduces the number of weights to train on compared to non convolutional network. CNNs often have a large number of weights regardless, but they can be tuned to be applicable to a wide variety of image data with the disadvantage of requiring significant computational resources such as a Graphics Processing Unit (GPU). Examples of well-known CNNs for image segmentation include the U-Net architecture [50], ResNet [58], and VGGNet [39, 53], all of which can be re-trained on various datasets–such as ImageNet [16]–to be applicable to specific settings. For yeast cells specifically, YeastNet [51] was developed to segment individual cells from bright-field microscopy images where many colonies lie on different focal planes.

Computational pipelines with more than one neural network can be developed where one neural network can be used as pre-processing for a subsequent one. For instance, the study by Carl et al [10] details the construction of a high-throughput deep learning pipeline which takes in images of plated *S. pombe* yeast colonies and assigns each colony detected a label based on their color. Their method can be partitioned into three steps: semantic segmentation, extraction, and classification. The study uses one neural network to segment colony pixels from their images. Individual clusters of colony pixels are then extracted as separate images for further analysis. These images were used as input to a second neural network which performs classification on a colony by assigning the colony one of five distinct color labels (red, white, pink, variegating, bad segmentation). This method has been shown to outperform the popular tool CellProfiler [33] which was reported to classify many red colonies as white in their images.

The method by Carl et al [10] serve as a good baseline for learning about phenotypic transitions. However, while this method is able to place colonies into one of five classes based on their phenotype, it does not do enough to perform a more detailed analysis of segmented colonies found in the variegating class. Moreover, we do not yet have a related analysis performed on sectored *S. cerevisiae* colonies, nor do we have a dedicated toolset geared for quantifying individual sectored colonies from image data. The work presented in this capstone seeks to create a toolset for such image data and to learn more about the mechanisms behind prion protein dynamics which produce different phenotypes in yeast colonies.

1.3 Overview of Capstone

In this capstone, I discuss work on the creation of a computational pipeline which aims to both provide a deeper understanding of prion protein dynamics through the

colony scale and address the scarcity of computational tools for performing a phenotypic analysis on full plates of yeast colonies. In Section 2, I describe the primary components of this computational pipeline suited for images of plated yeast colonies. The methodology discussed in the capstone consists of two tasks performed in series to extract plate-level, followed by colony-level information from experimental yeast colony images. First, I incorporate a popular CNN known as U-Net to perform semantic segmentation on images of sectored yeast colonies. I show how we incorporate synthetic image in the training process and how this is a powerful aid in segmenting the experimental images. Second, I develop a post-processing procedure within the pipeline that uses the output of U-Net for specifically segmenting and analyzing sectored phenotypes in yeast colonies detected, employing the framework of Carl et al [10] as a baseline. I show that using the two tasks in series simplifies the process of analyzing individual yeast colony phenotypes. In Section 3, I discuss the pipeline's performance on real images and what insight these results provide us about the nature of phenotypic transitions in yeast. Finally, in Sections 4 and 5, I provide a brief discussion on the limitations this pipeline experiences as well as how my pipeline can be improved in the future. This method allows for a more convenient, accessible approach for facilitating further understanding of phenotypic transitions in yeast due to the loss of the prion protein.

Chapter 2

The Pipeline

In this section, we detail the components of our proposed high-throughput pipeline for analyzing sectored yeast colony phenotypes (see Figure 2.1). This pipeline comprises two primary components. The first component involves constructing a neural network to perform image segmentation on plates containing hundreds of yeast colonies and using output of the network to locate and extract individual colonies. In Section 2.1 I discuss how we train the network used in this component to recognize colonies that can be extracted from our images. In particular, I detail how to incorporate synthetic training data of yeast colonies into the training process and show its effectiveness in quantifying real colonies. The images we use to test our pipeline are discussed in Appendix A, while the construction of the images for training our network to segment colonies is discussed in Appendix B. Our methodology for extracting colonies following image segmentation is discussed in Appendix C. The second component involves taking each colony extracted previously and uses tools from image processing and graph theory to estimate the frequency and shape of sectors present in each colony. In Section 2.2, I discuss how I annotate and partition colonies in the output segmentations to locate and quantify sector-like regions of red and white phenotypes. Metrics to aid in quantifying detected colonies are discussed in Appendix D.

2.1 Colony Detection

Here I discuss the procedure for segmenting images of plates containing many yeast colonies and the procedure for locating and extracting the segmented colonies. The segmentation procedure is performed using a slightly modified version of the U-Net deep-learning architecture [50] whose implementation is discussed in Section 2.1.1 and 2.1.2. Colony extraction is done via post-processing of the segmented images by locating circular regions corresponding to the location of colonies in the original images. The methodology for this procedure is discussed in Section 2.1.3.

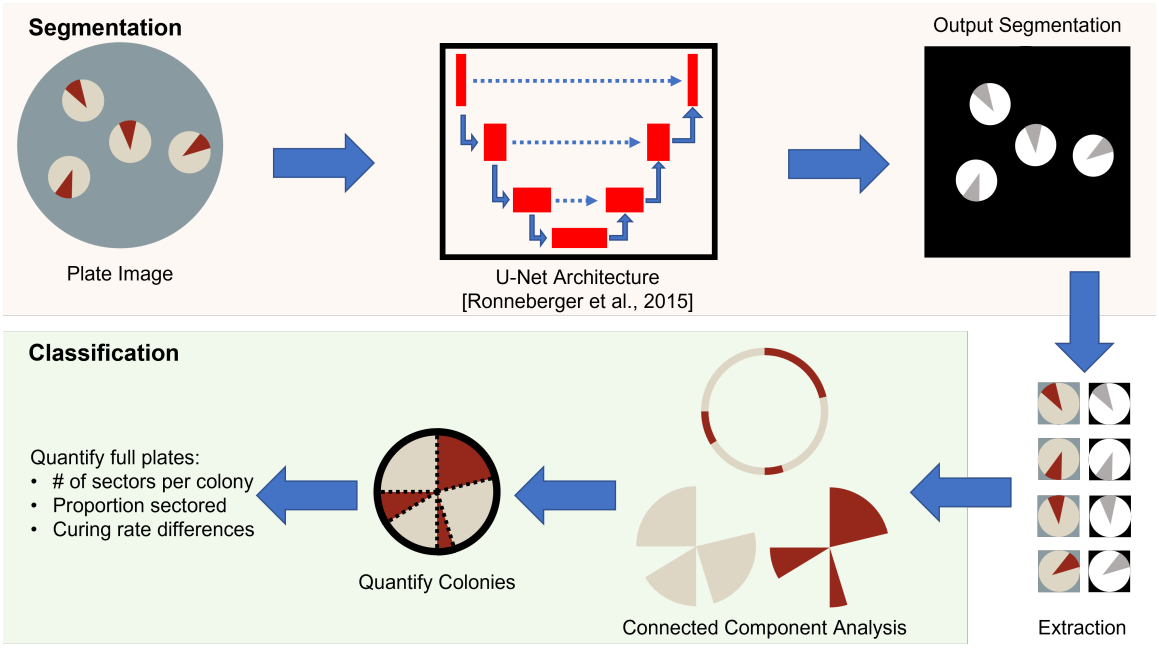


Figure 2.1: **Illustration of the full computational pipeline.** The pipeline is divided into two primary components. the first component of the pipeline is to use a modified U-Net architecture to perform image segmentation of full plates to locate colonies. The second component of the pipeline is to classify and annotate individual colonies by leveraging the spatial information in their corresponding segmentations.

2.1.1 Deep-learning Architecture

Based on a similar structure to what was done by Carl et al. [10], I utilize the U-Net architecture for perform semantic segmentation on images of plates to assign a label to every pixel in the images. U-Net is a type of supervised convolutional neural network originally designed for biomedical image segmentation [10, 44], but is widely generalized to other segmentation tasks. The architecture can be divided into contracting, bottleneck, and expansion paths. The contracting path consists of multiple blocks having two convolutional layers that increase the number of channels in the image followed by a max-pooling layer to cut the spatial dimensions of the image in half. The bottleneck path serves as the last of these blocks, at which the number of channels in the image is greatest. The expansion path does the opposite of the contracting path: within each block, upsampling is done to double the image resolution followed by two convolutional layers that decrease the number of channels. However, skip connections are applied which merge the resulting image with the image of similar size from the contracting path. The last layer of U-Net is a 1×1 convolutional layer where the user can specify number of channels (or labels) in the output image.

For my implementation of U-Net, I use a few deviations from the original architecture described in [50]. First, while the original architecture used images of size 572x572, I use images of size 1024x1024 as input. Next, I apply padding to the image before each convolutional layer to preserve the spatial dimensions to the image. This is reasonable since each image almost exclusively has background pixels on their borders. I then modify the output layer such that the final segmentation is of the same spatial dimensions as the input image and whose pixels are one of three classes: background, white colony, or red colony. To assign labels to each pixel, I use the softmax function at the last layer of the network to obtain the probability of each class per pixel, then take the maximum probability across the three classes at each pixel respectively. All other components of the architecture–the max-pooling layers, upsampling, and concatenation at each depth of the architecture–are the same as in the original implementation.

2.1.2 Training Process of U-Net

To train my implementation of U-Net, I wrote a script in Matlab to generate synthetic images of yeast colonies with features similar to those found in the experimental images. Training masks corresponding to these images are created via thresholding at intermediate steps throughout the generation process. The process for generating those images is defined in Appendix B. A total of 200 synthetic images and corresponding ground-truth masks were created for the training process, where 150 of these images are used directly in training and 50 are used for validation.

The creation of the U-Net architecture as described in Section 2.1.1 was implemented in Google Colaboratory's interactive python notebook with GPU access, where the compilation of the architecture is done using the Keras packages in Ten-

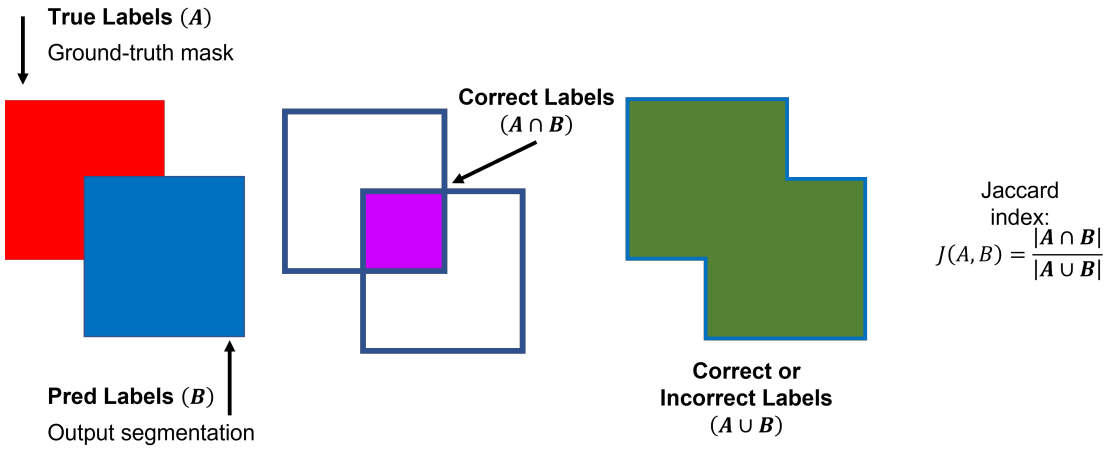


Figure 2.2: **Visualizing the Jaccard index for measuring accuracy in image segmentation tasks.** Each pixel with the ground-truth mask has exactly one label assigned to it. Similarly, following the result of a segmentation algorithm, the corresponding segmentation mask will also have exactly one label per pixels assigned. The primary objective of a segmentation scheme is to maximize the number of pixels in the segmentation mask whose labels match the corresponding pixels within the ground-truth mask. The Jaccard index is a metric to evaluate the accuracy of a segmentation scheme by taking the ratio of the total number of pixels whose labels in the ground-truth mask and segmentation mask match over the total number of distinct labels assigned between the two masks.

sorflow. The network is trained with a batch size of 1 due to the size of the images used and the amount of computational memory available. We use Tensorflow's categorical cross-entropy loss function and Adam optimizer. The number of epochs was not predetermined; instead, training stopped when the validation loss decreased by at most 0.001 over a period of 5 epochs. This is a helpful check to prevent the model from overfitting to the image set. The learning rate is initially set to 10^{-4}, but as the validation loss decreases and reaches a local minimum, the learning rate decreases by a factor of 10, with the lowest learning rate possible being 10^{-6}. The Jaccard index is used to compute the segmentation accuracy of U-Net (see Figure 2.2). This is done by first looking for the number of pixels that share the same colony labels between the ground-truth masks and prediction, then dividing this by the number of pixels which have either this exact criteria in addition to the pixels where they are labelled as colony in either the ground-truth or predicted mask but do not share the same label between the masks. After each epoch, a check is performed on the validation images to determine if the segmentation accuracy is higher than in the previous epoch; if the accuracy is higher, the new parameters are saved, which can be used as a checkpoint for future training of U-Net. When U-Net is sufficiently trained to segment red and white colony pixels, I use this to produce output segmentations of the colonies for each image whose pixels are assigned one of the three labels mentioned in Section 2.1.1.

2.1.3 Colony Extraction

Using the output segmentation, I can find colonies by looking for clusters of colony pixels (both red and white). However, since colonies can intersect in both the synthetic and experimental images, additional steps are needed to disambiguate clusters of multiple colonies and to account for variation in colony sizes. Colonies in both the real and synthetic images appear circular, and as such we use the circle Hough transform implementation in Octave to detect circular objects in the output segmentation. The details of the circle Hough transform in terms of this work are described in Appendix C.

To perform the above, I first find all the connected components of non-background pixels. Next, to get an estimate of the sizes of each colony present, I look for the connected components that closely resemble isolated colonies. Here, I make the assumption that a connected component in the segmentation corresponds to an isolated colony if it meets the following conditions:

1. The connected component must have a number of pixels between a minimum and maximum value. In my case, I required all connected components to have between 100 and 2000 pixels. This is a way to filter colonies that are too big or too small.

2. The bounding box of the connected component must have an aspect ratio close to 1. In my case, I required the greater ratio between length and width of the bounding box to be less than 1.2 to account for image compression and imperfections in the circularity of colonies in the output segmentation. This is also a filter for removing most clusters of colonies from consideration, especially those whose colonies appear to be co-linear.

3. The proportion of pixels within the bounding box consisting of either red or white colony pixels must be between a minimum and maximum value. In my case, I required that the proportion of pixels inside the bounding box to be between 0.7 and 0.9 which contains $\pi/4$, the ratio between the area of a circle and smallest enclosing square respectively. This helps remove colonies whose circularity is very insignificant or are too close to the border of the plate.

For each connected component that meet all the criteria above, we record half the length and width of its corresponding bounding box which will serve as estimates for the radius of the connected component.

For each connected component that meets the above criteria, we then apply the circle Hough transform (see Appendix C) using Octave's function `imfindcircles` to locate the circular objects in the output segmentation. We set the sensitivity parameter to 0.9 to allow for imperfect circles to be detected, and the radii range the minimum and maximum width of the boxes found in the previous step. Since this function strongly recommends that circular objects have a radius of at least 5 pixels, we also set an arbitrary minimum of 7 pixels for the radius of circular objects. If the

minimum dimension of any bounding box is less than 7, we temporarily rescale the entire segmentation so that the smallest dimension of any bounding box is 7, before using `imfindcircles`. For each circle detected, the radii, center coordinates, and the coordinates of the bounding boxes are recorded. If rescaling was done prior to recording this data, the data is rescaled so that it corresponds to the original sized segmentation. For each circle detected, a bounding box was drawn around the colony, and the region of the bounding box was cropped from the image. These images will be used in the next subsection.

2.2 Colony Classification

In this section I discuss the second major component of this pipeline designed for quantifying the colonies detected using the procedure in the previous section. Section 2.2.1 discusses how I use the segmentation of each colony for quantifying regions of red and white phenotypes. Section 2.2.2 discusses how we aggregate this information and transform it in a way where we can infer results from the experimental procedures carried out on the plates within the images.

2.2.1 Annotation

Figure 2.3 shows the annotation procedure for counting and quantifying sectors each detected colony. Since the red and white regions of a colony may cluster as irregular to complex shapes, it is less likely to detect "ideal" sectors in colonies. This proposed procedure is well-suited for locating sectored regions as long as those regions do not appear too irregular initially.

Given a colony segmentation, we first decompose it into its interior and boundary components. A pixel in the colony segmentation is considered a boundary pixel if it is a colony pixel that is also adjacent to a background pixel. Otherwise, that pixel is considered to be an interior pixel. For simplicity, I skeletonize the boundary of the colony so that it has pixel width 1.

Next, I further decompose both the interior and boundary components of the colony respectively into their red and white regions, and them find the connected components of red and white pixels separately on the boundary. For each component, we construct an "idealized" sector where the connected components serves as one of the sector boundaries. To estimate the other two boundaries of the idealized sector, I proceed to find the endpoints of each component on the boundary. I use two methods to find the endpoints for each component. This relies on there being no more than 2 endpoints for each skeletonized boundary. I first use the hit-miss algorithm within the SciPy package [61] to find the endpoints. Next, I use a strategy very similar to what I did previously for finding the colony boundary. A padded version of the colony skeleton is created, each pixel is initialized to 8, and subtracted by the number of non-zero pixels nearby. Any pixel that has a value of 7 is an endpoint. Note that this method can find endpoints on a corner of a skeleton, while the hit-miss algorithm can

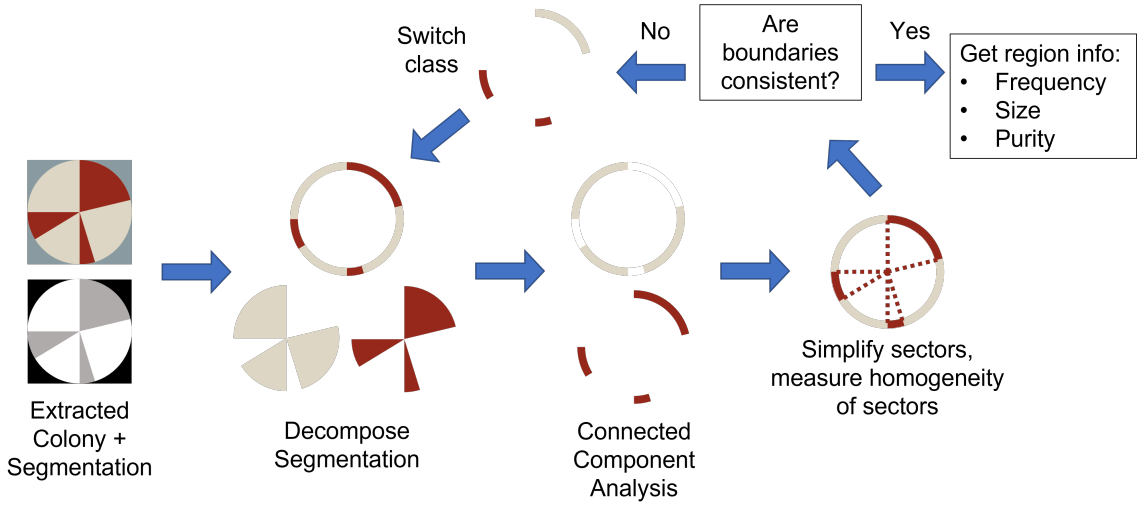


Figure 2.3: **Novel sector counting procedure.** Flowchart of the connected component analysis technique I propose to count the number of sectors and estimate their properties. Using the output segmentation, I proceed to decompose the classes into the interior and boundary components. The second method involves taking only the pixels found on the boundary of the colony before decomposing the classes and finding the number of connected components. For each component found, I perform a check to see if the corresponding sector has completely segregated from other sectors present, and determine whether the resulting component is part of a cohesive sector by checking whether its interior also contains a significant number of red colony pixels. Should a significant number of pixels in the interior of the sector contain pixels of a different class, the class of the boundary component will be switched. The process repeats until all boundary components are consistent with their corresponding interior regions. The number of consistent red regions remaining is used as the prediction for the number of sectors present in the image.

find endpoints near a corner. I then take the union of endpoints located from both methods, because initial observations suggest they correct each other's shortcomings. The remaining two boundaries are then drawn using Bresenham's line algorithm [8] to connect the endpoints with the colony center via lines in pixel space. This results in a closed shape representing the idealized sector boundary, while the collection of pixels within represents the interior of the sector. This process is repeated for all red and white regions.in the colony.

Since I am working with croppings from the full plate images, each colony will have fairly low resolution (around 20 to 40 pixels in any dimension). At this scale, it is difficult to accurately capture the interfaces between the colony background and different colored regions of the colonies. The process described above allows us to estimate these interfaces; what remains is to examine the accuracy of the regions. Here I develop a procedure for annotating and analyzing features in these low resolution images and to determine whether the corresponding output adequately captures the red and white regions.

Figure 2.3 shows the overview of how my proposed pipeline estimates the location and shape of red and white regions in a colony. I define a metric known as colony purity (see Appendix D) to measure how close sectors in the segmentation are to their corresponding idealized sectors constructed previously. We compute the purity for each red and white region located, then perform a check for consistency to ensure the regions are of the "correct" colors (see Figure D.1 B). If the purity of any region is at least 0.5 (at least 50% of the region's pixels is of the same class as the region itself), this suggests that the segmentation has adequately captured the location of a sector-like region in the real image. If otherwise, then we change the color of the region by swapping the labels of the pixels on the region's boundary from red to white, assuming that the region was initially red. If any changes to the boundary pixels are made, we recompute the purity for all regions in the colony segmentation. As a consequence, this also ensures that the color whose purity is greater corresponds to the color of the region. Following this procedure, regions are merged if their corresponding boundary pixels are of the same color (see Figure D.1 B (right)). The number of red regions that remain is used as the predicted number of sectors in the colony. I then use the number of sectors on a colony to place a qualitative label on the colony that signifies whether the colony is sectored (has at least 1 red and 1 white region), "cured" (1 red region and 0 white regions), or "stable" (1 white region and 0 red regions).

2.2.2 Characterizing Full Plates

Data collected from each colony as described in Section 2.2.1 is aggregated to characterize each plate independently, generating a breakdown of the frequency and proportion of sectored, cured (fully red), and stable (fully white) colonies detected in each image. We use Fisher's exact test to quantify whether a significant difference exists in the rates of curing between any pair of plates. This test is commonly applied to data in 2x2 contingency tables [29, 60], so we adapt this test to compare the

proportion of cured, stable, and sectored colonies between a pair of plates. Here we assume that the null hypothesis is that the proportion of cured colonies in each pair of plates are equal, where the alternative hypothesis is that the proportions are not equal. We use $\alpha = 0.05$ as the significance level for each test. In my case, I use this test to compare the proportion of cured vs non-cured colonies, the proportion of stable vs cured and sectored colonies, and the proportion of sectored vs homogeneous colonies between two plates. This is a useful test for comparing two plates each containing colonies grown under different experimental conditions.

Chapter 3

Results

In this Section, I provide the results for the output of the pipeline in its entirety when applied on the image sets described in Appendix A. Section 3.1 details the results of the segmentation process, with emphasis on the colonies extracted from both the experimental and synthetic images. Section 3.2 details the results of quantifying sectors from the colony segmentations, with comparison to manually counted colonies. I show that my pipeline is very useful for quantifying sectoring activity in images of yeast colonies.

3.1 Segmentation

Figure 3.1 shows an example of one synthetic image and its corresponding segmentation with isolated colonies clearly distinguishable. From the 150 images used to train U-Net, we obtained a cross-entropy loss of 0.0022 for the training and validation images independently after 24 epochs, achieving sufficient performance on the synthetic images in preparation for testing U-Net on the experimental images.

Using image set 1 in Appendix A, we obtained segmentations suitable enough for colony detection without pre-processing. An example of the output segmentation on one of the images in this set is shown in Figure 3.2. Following the execution of the circle Hough transform on these images, we obtained a total of 1,266 colonies with corresponding segmentations which were extracted for classification. We note that many of the colonies near the edge of each plate were difficult to discern structurally, so nearly all of the colonies found there were ignored.

Using image set 2 in Appendix A and using the color transfer methods as pre-processing the quality of the output segmentations has significantly improved to the point where clearly distinguishable colonies can be extracted (see Figure A.3). We obtained a total of 715 colonies with corresponding segmentations which were extracted for classification. Similarly, nearly all colonies near the edge of each plate were ignored.

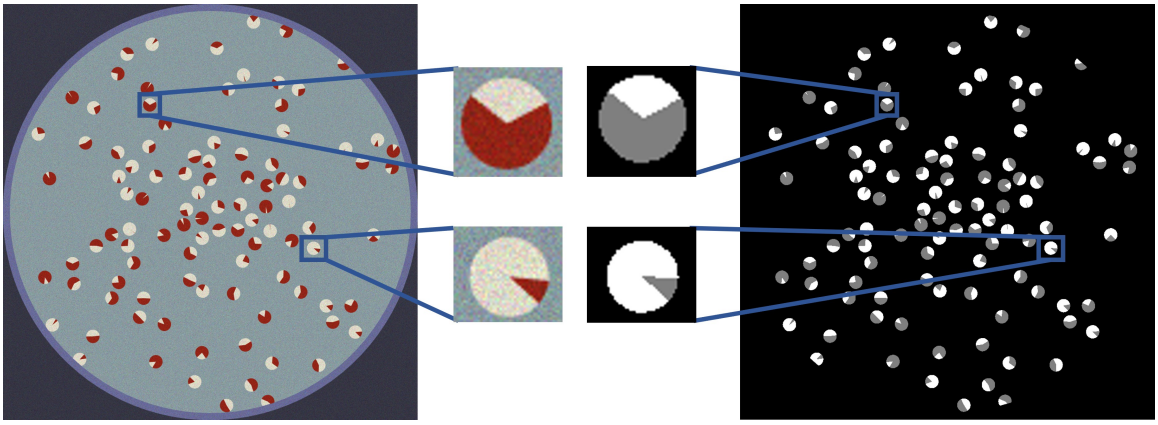


Figure 3.1: **Example of synthetic image segmentation.** Example of a synthetic image (left) and its corresponding output segmentation from the trained U-Net (right). Two isolated colonies are shown with their segmentations (middle). The U-Net segmentations have the following color code: Background pixels are black, red colony pixels are gray, and white colony pixels are white.

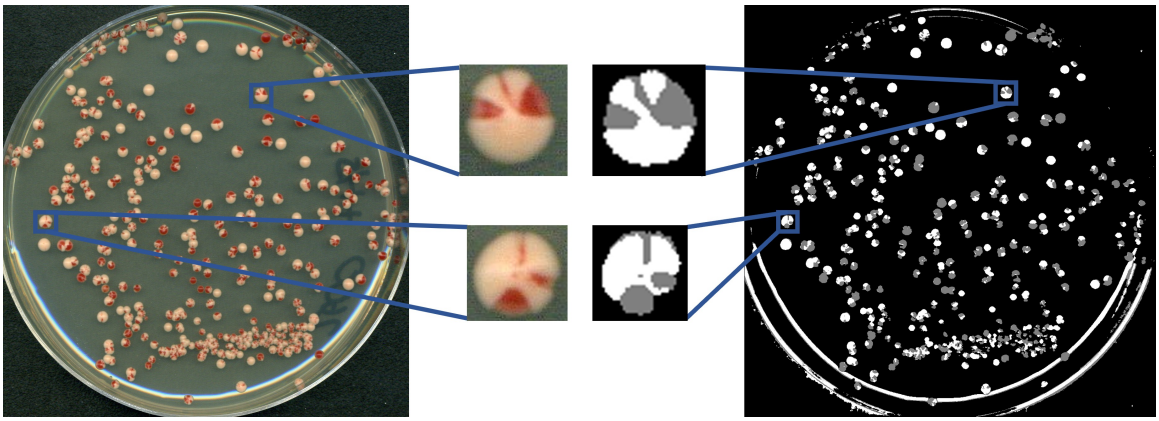
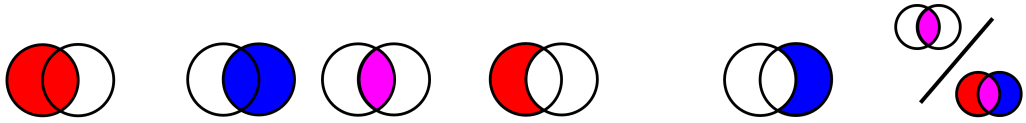
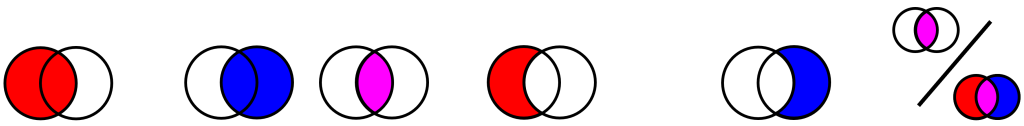


Figure 3.2: **Example of experimental image segmentation.** Output for U-Net using one of the experimental images as input. In the middle are the original representations and corresponding output segmentations from U-Net for two colonies.



	Quantifiable Colonies Counted Manually	Colonies Detected in Pipeline	Colonies in Common	Quantifiable Colonies not Detected in Pipeline	Non-quantifiable Colonies Detected in Pipeline	Jaccard Index
Plate 1	355	322	243	112	79	56%
Plate 2	190	156	122	68	34	54.46%
Plate 3	269	318	186	83	132	46.38%
Plate 4	236	283	192	44	91	58.72%
Plate 5	177	187	151	26	36	70.89%

Figure 3.3: **Breakdown of the number of colonies found on each plate from image set 1.** The first column shows the number of colonies in the image that a biologist performing manual counting would be considered quantifiable, or colonies that can be analyzed with simplicity. The second column is the number of colonies found using the circle Hough transform after segmentation is performed. The third column is the set of colonies that were considered quantifiable and were detected in my pipeline. The fourth column is the set of quantifiable colonies that the circle Hough transform did not find. The fifth column is the set of colonies found with the circle Hough transform but were not considered easy to analyze. The Jaccard index in the last column was computed to be the ratio of quantifiable colonies detected in my pipeline over the union of colonies detected by either the pipeline or are deemed quantifiable by experimentalists (the sun of columns 3-5).



	Quantifiable Colonies Counted Manually	Colonies Detected in Pipeline	Colonies in Common	Quantifiable Colonies not Detected in Pipeline	Non-quantifiable Colonies Detected in Pipeline	Jaccard Index
Plate 1	127	139	121	6	17	84%
Plate 2	106	122	106	0	16	86.9%
Plate 3	92	98	81	11	17	75%
Plate 4	127	101	91	36	10	66.4%
Plate 5	112	119	105	7	14	83.3%
Plate 6	131	136	120	11	16	81.6%

Figure 3.4: **Breakdown of the number of colonies found on each plate from image set 2.** The first column shows the number of colonies in the image that a biologist performing manual counting would be considered quantifiable, or colonies that can be analyzed with simplicity. The second column is the number of colonies found using the circle Hough transform after segmentation is performed. The third column is the set of colonies that were considered quantifiable and were detected in my pipeline. The fourth column is the set of quantifiable colonies that the circle Hough transform did not find. The fifth column is the set of colonies found with the circle Hough transform but were not considered easy to analyze. The Jaccard index in the last column was computed to be the ratio of quantifiable colonies detected in my pipeline over the union of colonies detected by either the pipeline or are deemed quantifiable by experimentalists (the sun of columns 3-5).

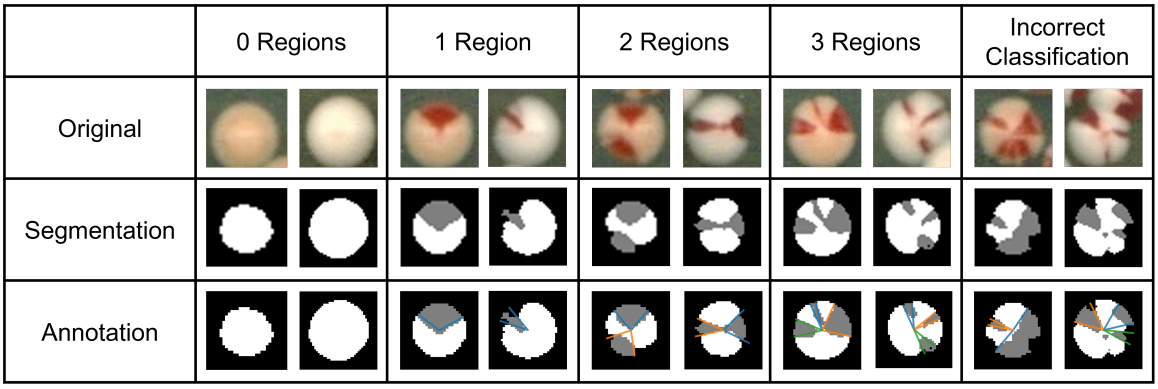


Figure 3.5: **Example colony annotations.** Output segmentations of colonies extracted from the image sets with their regional annotations such that white and red regions are separated. Each set of colored lines in each annotation represents one region, where the lines estimate the ideal interfaces between red and white regions. Each column shows two examples of colonies with correctly predicted number of red regions. the last column shows two examples of predictions that did not match the true number of sectors present.

3.2 Classification

To determine how accurate my proposed pipeline is for quantifying sectors, we compare our predictions with manually acquired data on "quantifiable" colonies (colonies easy enough to analyze manually) which consist of the true number of sectors per colony and, for single-region colonies, whether they are truly sectored or cured (fully red).

In the synthetic images, each colony has at most one sector (many colonies have sectors so small that they could not be rendered properly in the image). The error rate for counting sectors within all 200 images is substantially low, with $\approx 99.4\%$ of isolated colonies ($\approx 19,426$ colonies in total) extracted also predicted to have one sector. When the proposed pipeline is applied to all experimental images in sets 1 and 2, we obtain approximately 1,981 colonies. Example segmentations and regional annotations of colonies are shown in Figure 3.5.

Out of the 1,266 colonies segmented and extracted from image set 1, colonies were predicted to have anywhere between 0-3 sectors (Figure 3.6 (top left)). From these, 640 colonies were predicted to have only one red region, where 462 were cured colonies, and 178 were sectored colonies (Figure 3.6 (top right))). Out of the 715 colonies segmented and extracted from the images in set 2, colonies were predicted to have anywhere between 0-4 sectors (Figure 3.6 (bottom left)), with 279 predicted to have one red region. From these 279 colonies, 57 were predicted cured, and 222 sectored (Figure 3.6 (bottom right)).

Our results show that the accuracy of sector counting significantly improves when we use our purity metric in this pipeline to correct the estimated regions in the

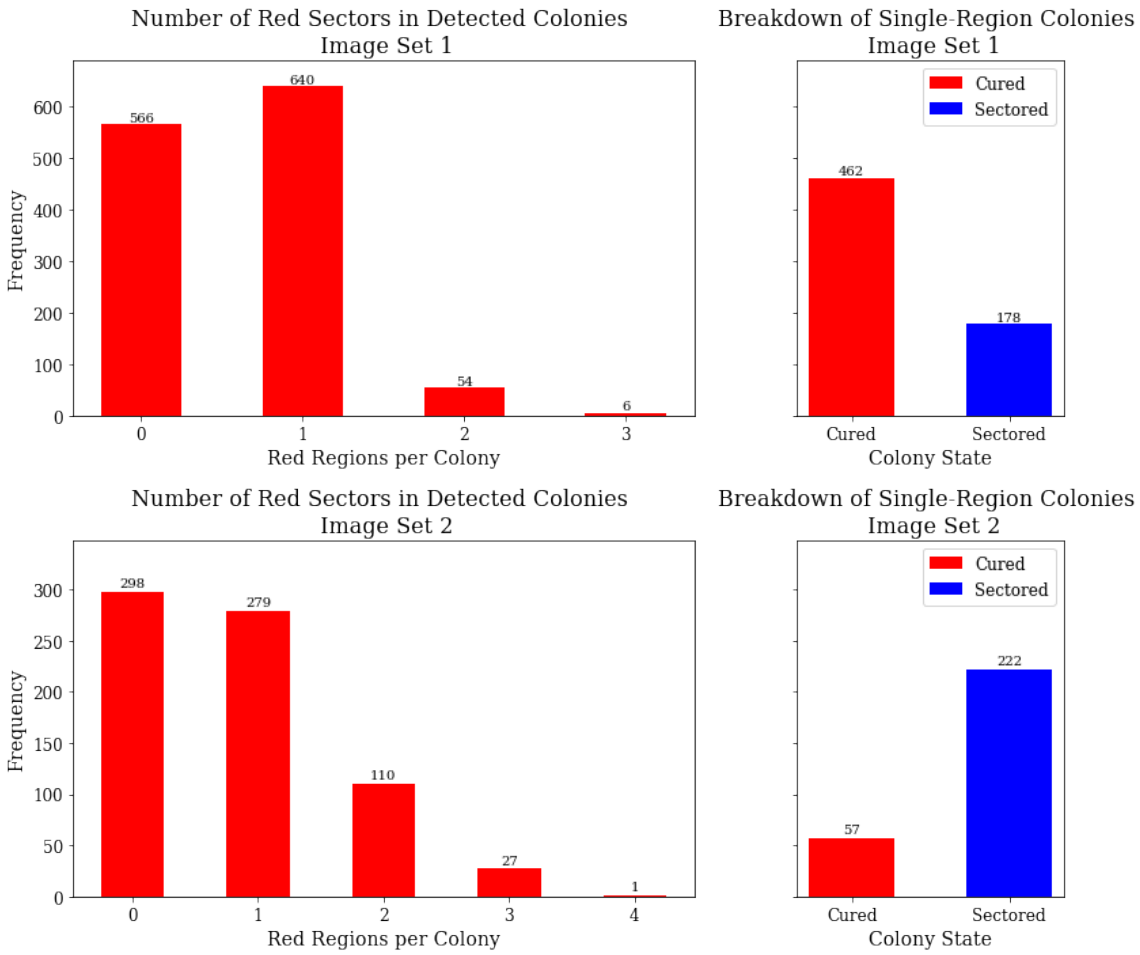


Figure 3.6: **Aggregated sector frequency and colony state predictions.** Predictions of the proposed pipeline on all detected colonies from image set 1 (top) and image set 2 (bottom) (Left): Total number of colonies that had "x" number of sectors predicted through our proposed pipeline. (Right): Out of the colonies predicted to have one red region, how many of them were actually sectored (have a white region) and those that are cured (have no white region).

colony segmentations, rather than only relying on the number of connected components of pixels on the colony boundary. Figure 3.7 provides a comparison between the estimated number of colonies predicted to have specific number of sectors for two scenarios: when only the connected components on the colony boundary are quantified, and when the purity metric is used to correct for bad regions in the segmentation. As a consequence, the accuracy for the number of sectors present in each colony showed a significant improvement. Our proposed pipeline correctly predicted the frequency of sectors in approximately 91.4% of all quantifiable colonies detected across all image sets used in this work. In image set 1, about 660 out of the 961 quantifiable colonies ($\approx 68.9\%$) extracted had correctly predicted sector counts when the purity correction step was not applied. When the correction step was applied, 914 out of these 961 colonies ($\approx 95.1\%$) extracted had correctly predicted sector counts (Figure 3.7 (top left)). For single region colonies, 320 of colonies correctly predicted to have one red region were also correctly labeled as sectored/cured (accuracy of $\approx 73.2\%$) when only the number of connected components per colony was considered. This quantity increased to 432 (accuracy of $\approx 98.9\%$) when purity was used to correct for bad regions in colony segmentations (Figure 3.7 (top right)).

For image set 2, we detected 624 quantifiable colonies over the 715 total detections. About 352 out of the 624 quantifiable colonies ($\approx 56.4\%$) extracted from this set had correctly predicted sector counts when the purity correction step was not applied. When the correction step was applied, 535 out of these 624 colonies ($\approx 85.7\%$) extracted had correctly predicted sector counts (Figure 3.7 (bottom left)). Similarly, the accuracy for labeling colonies by state is greater when the purity correction procedure is used (accuracy $\approx 87.1\%$) than without it (accuracy $\approx 67.6\%$) (Figure 3.7 (bottom right)).

We also use confusion matrices to see how both sector counting schemes place colonies into the correct groups in more detail across both image sets (Figure 3.8). We clearly see that including our purity correction scheme places more colonies on the main diagonal of the matrix. Furthermore, many of the outlier colonies (those initially predicted to have four or more regions) moved to regions with less sectors present as a consequence of the purity correction scheme. This suggests our purity correction scheme is sufficiently preventing overcounting of the number of regions per colony in our dataset.

To look deeper at how our purity correction scheme influenced the predicted frequency of sectors in quantifiable colonies as well as the structure of each of their regions, we grouped colonies with the same number of sectors as determined by the manual annotations and then aggregated the colonies whose predictions changed or remained the same as a result of the purity correction scheme (see Figure 3.9 (left)). For 860 colonies whose initial sector frequency predictions were correct, the purity correction scheme did not alter these frequencies. However, we find that 589 colonies, which initially had incorrect sector frequency predictions, became correct after our purity correction scheme was applied, with more than half of these colonies manually annotated as fully white. This suggests that most of the red regions in these colony

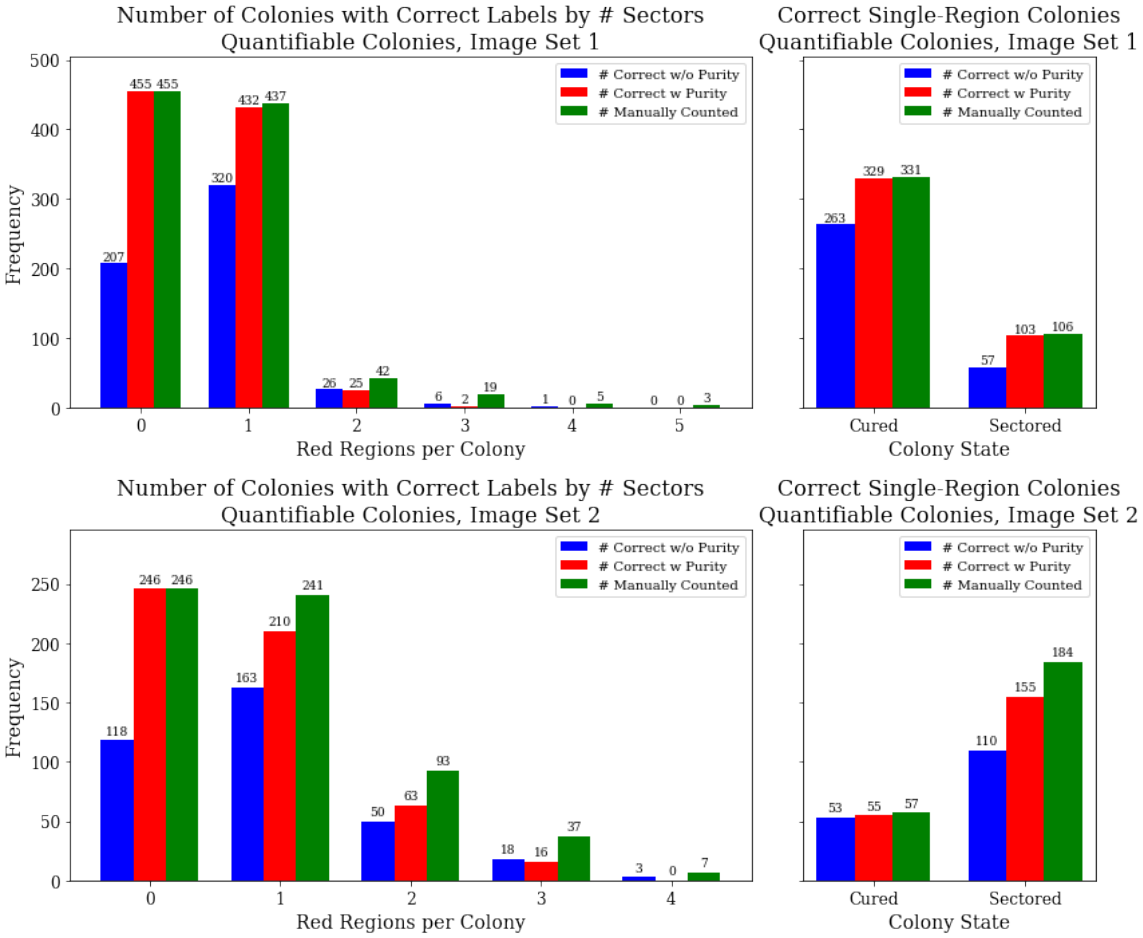


Figure 3.7: **Aggregated counts of sector frequencies and sector states matching manual annotations.** Total # of quantifiable colonies whose predicted labels match the manually collected labels for number of sectors (left) and states of single-region colonies (right) for image set 1 (top) and image set 2 (bottom). The blue bars represent the predictions made using only information about the number of connected components on the boundary of each colony. The red bars represent the predictions made using the purity metric to isolate and correct unfavorable regions as described in Appendix D. The green bars represent the number of colonies in the manually counted data which meet the characteristics and is considered the ideal scenario.

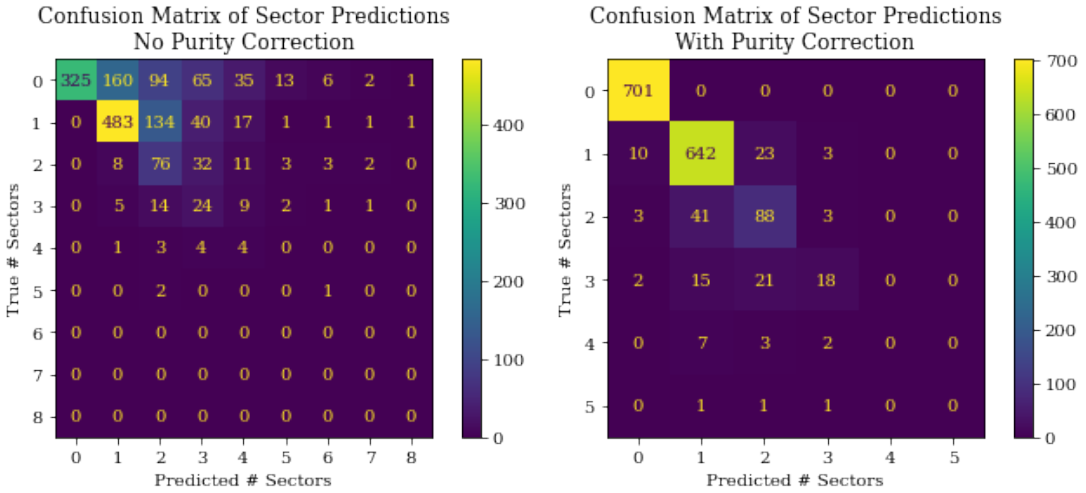


Figure 3.8: **Using the purity metric to correct for bad colony segmentations improves accuracy of sector frequencies.** Confusion matrices showing, across all quantifiable colonies from both image sets ($N = 1585$), the number of colonies which actually have "x" sectors (rows) that were predicted to have "y" sectors (columns). The quantity of colonies whose predicted counts match the true data is the sum of the diagonal of the matrices. (Left): Confusion matrix showing the number of sectors predicted without including the purity correction procedure in the pipeline. (Right): Confusion matrix showing the number of sectors predicted when we do include the purity correction procedure. Many more colonies lie on the main diagonal, indicating greater accuracy in sector counting.

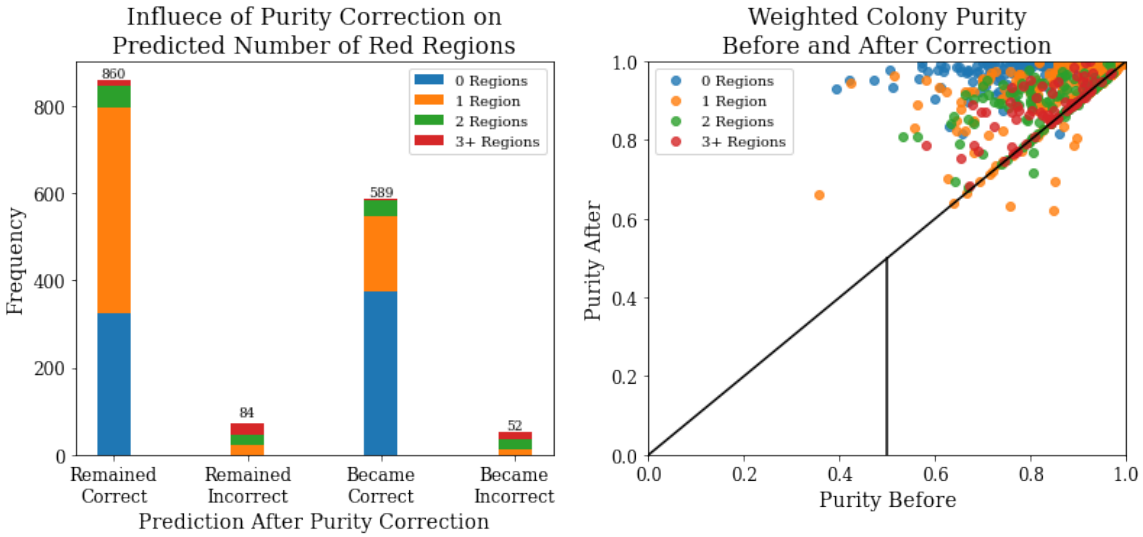


Figure 3.9: **Our purity correction scheme significantly increases the accuracy of sector counting.** (Left): Breakdown of the number of accurate and inaccurate predictions of each colony based on the accuracy of the initial region estimation compared to when our purity correction scheme is applied. Each color represents the group of quantifiable colonies whose true number of sectors is known through manual annotations. (Right): Weighted purity scores of each quantifiable colony between the initial annotation (x-axis) and the annotation following purity correction (y-axis). Any point above the line $y = x$ indicates that the weighted purity of a colony increased after the correction step was applied. The vertical line at 0.5 on the x-axis signifies the minimum threshold for purity at any region of a colony.

segmentations did not have a sufficient number of red pixels, thus they were eliminated in the purity correction scheme. Colonies where the purity correction scheme did poorly on were colonies that had multiple red regions, suggesting that the sectors in those colonies may be too small to adequately segment. We then looked at how the weighted purity (see Equation D.6 in Appendix D) of each colony changed after applying our purity correction scheme to their initial estimated regions (see Figure 3.9 (right)). We find that for most colonies, the weighted purity increased when our correction scheme was applied, suggesting that our estimated regions better capture the sector-like structure in these colonies.

3.3 Test for Differences in Colony States

Using per-plate aggregated data, we use Fisher's exact test to look for differences in proportion of colonies in each plate that are cured, stable, and sectored independently. This test uses data stores in a 2×2 contingency table containing frequency distributions of elements across two datasets that satisfy a set of conditions.

Figure 3.10 shows matrices of p-values computed using Fisher's exact test to look for significant differences in the proportion of cured, stable, and sectored colonies for each pair of plates compared. In image set 1, we found a statistically significant difference in the rate of curing between all pairs of plates except for plates 4 and 5 ($p \approx 0.742$) (Figure 3.10 (top left)). More specifically, the test suggests two plates appear to have the same proportion of cured colonies that are cured, despite the different densities of colonies on both plates. Similarly, the rate of stability between plates 4 and 5 does not appear to have a statistically significant difference ($p \approx 0.433$) (Figure 3.10 (top middle)), and the rate of uniformity between plates 4 and 5 ($p \approx 0.383$)–as well as plates 3 and 5 ($p \approx 0.297$)–also does not appear to be significantly different (Figure 3.10 (top right)).

In image set 2, since plates 1 and 2 both only contain stable colonies, no significant differences in curing, stability, or sectoring were found between them ($p = 1$), but when both of these plates are compared with the others in this set, there are obvious differences in the rates of curing, stability, and sectoring at the 0.05 significance level (Figure 3.10 (bottom)). Rates of stability do not appear to be statistically significant between any pairs of plates 3-6 ($p \geq 0.237$). Rates of sectoring also do not appear to be significantly different between any pair of plates 3-6 ($p \geq 0.054$) at the 0.05 significance level.

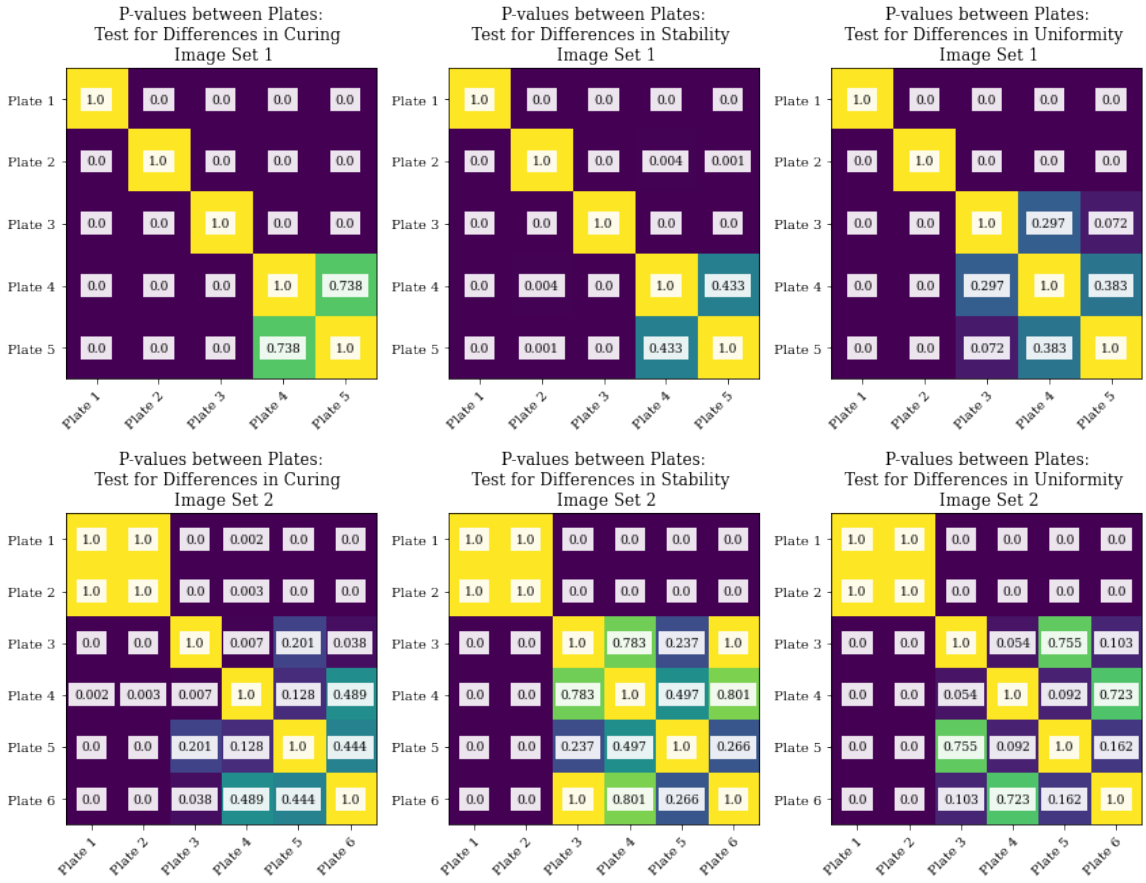


Figure 3.10: **Testing pairs of plates for differences in curing, stability, and sectoring.** P-values for Fisher's exact test when comparing two plates to test between proportions of cured versus non-cured colonies. (Left): P-values from Fisher's exact test for colonies in image set 2. In this set, plates 4 and 5 have the highest p-value off the main diagonal, indicating that the rate of curing of colonies between these two plates are the most similar out of all possible pairs from this image set. (Right): P-values from Fisher's exact test for colonies from image set 3. In this set, plates 1 and 2 consist of only stable (white colonies), indicated by their p-value of 1 off the main diagonal. Plates 4 and 6 also have similar rates of curing.

Chapter 4

Discussion

4.1 A Note on Training Images

Two of the practical reasons for using synthetic images to train U-Net for segmentation is to circumvent the difficulty and tediousness of manual annotation as well as acquisition of specific data. The ideal objective for creating the synthetic images is to ensure they they appear very similar to the images that the pipeline is geared to analyze. However, it is also important to note that a subset of content found in real images–such as shot noise from an imaging device [32]–cannot be completely controlled, thus creating variation in what we expect to see in an image at the pixel level. Two sources of variation not currently present in the synthetic images can be addressed for improving colony segmentation. First, while the synthetic images account for most of the color variation present in the second class of colony images as described in Section A, they are not good enough for training U-Net to segment colonies within a different image set which appear to have darker colonies. Arbitrary colors for the colonies and background regions were chosen as baselines before the entire image was given Poisson noise. Other experimental images however appear to have different baseline colors from the naked eye. Second, all the colonies in the synthetic images are of the same size, while in the experiential images different colonies have different sizes. The synthetic images make geometric use of primary features from these experimental images such as colony circularity and sector-like regions. However, there are features in the synthetic images that do not account for enough variation across multiple image sets. To account for these sources of variation in the experimental images, more structured diversity in the synthetic training data should be introduced. I am refining the script generating the synthetic images to select colony and background colors as well as circle sizes from specified distributions. A diverse training set is imperative to ensure robust performance of my pipeline across all experimental conditions.

4.2 Quality Control

Initial findings suggest that my proposed pipeline will greatly benefit from a robust pre-processing procedure, which motivated my use of the methods in Section A. As mentioned in Section A, image set 2 had the color profile I used for generating the synthetic images. What prompted the use of pre-processing methods on all other image sets was the inadequacy of the output segmentations of these images. For example, the segmentation of one of the images (Figure A.3) we used in this work without any pre-processing was insufficient enough such that the circle Hough transform could not detect any circular objects. After applying the color transfer method as described in Section A, the circle Hough transform detected 122 circular objects–nearly all of them colonies–in this image following pre-processsing. This example demonstrates two things with my pipeline. First, images need to be similar to those originally trained on U-net to produce good segmentations. Second, specialized pre-processing methods are important tools to adapt diverse datasets for training and testing neural network-based models.

An additional issue I have noticed in the experimental image segmentation was that a significant proportion of misclassified pixels appeared to be at the interfaces between regions of different classes. For example, as seen in Figure A.3, many colony pixels were often misclassified as background pixels at the interface between red and white colony regions. This issue is possibly tied not only to differences between the true and estimated radii of colonies in the pixel space, but also due to low resolution of the colony following extraction from the original image (many colonies are between 20-30 pixels wide in both dimensions). Furthermore, many pixels at the white region of the colony boundary are being misclassified as red colony pixels. This is likely due to the colors of the red and background pixels having similar probabilities after the softmax function is applied at the last layer of U-Net. However, since this algorithm is primarily focused on predicting frequency of sectors rather than highly accurate segmentation of entire sectored regions, the issue between the red and white regions of a colony was not a high priority. My classification method helps correct the predicted sector counts by checking for consistency of the pixel classes between the boundary and the interior of each possible sector, even when the interface between the red and white regions is somewhat inaccurate.

I believe future research can address these segmentation issues by placing greater emphasis on two procedures. The first procedure is to standardize image acquisition by using equipment capable of taking images that can capture the interfaces between different classes more easily. This should also include a way of standardizing a method of post-processing the test images after acquisition so that U-Net is more capable of producing good quality segmentations. The second procedure is to use a very specific training regimen which is easy for others to reproduce results. This should include detail on the image data used for training, how all hyperparameters are initialized and updated, and what technical architecture (i.e. Tensorflow or Pytorch in Python) is used to implement the training process.

4.3 A Note on Colony Quantifiablity

Part of quality control regarding the colones found in the real images may also include the placement of colony forming units at the time of plating in a way where colonies do not overlap nor cluster as a single cohesive unit. Accounting for these features would make all colonies isolated, which would be the most ideal colonies to quantify depending on the experiment. However, most colonies in the images we used in this pipeline are not isolated. Colonies on the border of the plate present another challenge with resolving visual defects and aberrations. As such, we need more specific criteria for which colonies are non-quantifiable for my purpose. Experimentalists suggest that criteria for colonies that are best suited for my pipeline are the following:

- Colonies should not be too close to the edge of the plate as there may be aberrations present.
- Colonies should not be too small, as they may not be trustworthy even if sectors are present.
- Colonies should be at least close to a perfect circle. If a colony does not fall under this category, it may be because it is composed of two very close colonies that appear as one at first glance.
- Colonies should not significantly intersect. This will make it difficult to determine which sector belongs to which colony.

As a consequence, quality control in the lab where the yeast in the images are produced is one of the most important factors to ensure ideal images for the entire pipeline. In our results, most of the colonies our proposed pipeline extracted which were not manually annotated appeared to be clustered with at least one other colony. While it is not necessary for colonies to be isolated for experimentalists to quantify, they need to be sufficiently apart to minimize bias in quantification. We believe our proposed pipeline performs well on these colonies and can still be applied to a small set of colonies that do not fit the criteria above, assuming that it is clear that the information in each colony to be examined does not reside in regions where two or more colonies overlap.

4.4 Can a Fully Data-Driven Approach be Implemented?

I argue that the pipeline presented in this capstone provides more detailed output than the method of of Carl et al [10] by not only computing a proportion of the colony that is red or white, but also giving a frequency for the number of regions of different phenotype per colony. There are two shortcomings however. First, my pipeline relies on the output of U-Net in order to predict the number of sectors

present in a colony. Second, all other components of my pipeline that deal with colony detection and sector quantification does not fall under the category of deep-learning. I would like to suggest that it is possible to address both of these issues with other deep-learning frameworks. I would argue that the method of Carl et al [10] can be modified to quantify sectors as it already provides a prediction for the percentage of phenotype per cell. More specifically, their methods already has a procedure to semantically segment colonies, but this could potentially be expanded by providing a thorough spatial analysis of each segmented colony to isolate distinct regions corresponding to sectors of different phenotypes. In addition, I also argue the method by Ferrari et al [17]–while designed to disambiguate bacterial colony clusters–can be modified to detect sectored yeast colonies by retraining the CNN used in this study. Image data on yeast colonies can have colonies that cluster, especially for very dense plates. As such, a CNN that can analyze colonies found within larger clusters can reliably mine additional data from the same images, including on colonies too small or ambiguous to analyze manually. By using deep-learning based methods in conjunction for colony detection and quantification, every major component of my pipeline can be consistently data-driven, enabling analysis of sectored yeast colonies that relies solely on training data.

4.5 From Colony Scale To Multiscale

My computational pipeline is primarily designed to quantify sectored yeast colonies at the scale of the naked eye. However, this pipeline also presents an opportunity to understand how different experimental procedures (involving smaller-scale dynamics) can affect the outcome of sectoring in yeast. This model can potentially be applied to colonies in the study by Klaips et al [31] to quickly aid in the quantification of yeast colonies with the $[PSI^+]$ and $[psi^-]$ prion phenotypes by comparing plates with colonies at different temperature controls.

Future research should attempt to bridge the gaps between the output of my pipeline and known mathematical models of prion aggregation. In a simulation study I worked on with Banwarth-Kuhn and Sindi [4], we constructed an agent-based model for quantifying colony structure in a growing, budding yeast colony. This paper shows that the cumulative effects of cell-cell budding throughout the colony coupled with a nutrient limited environment give rise to the formation of independent, well-defined sector-like structures of colonies, each of which whose cells are descendent of a "founder" cell. Such model can be extended to include intracellular activity of prion-forming proteins being transferred between mother-daughter cell pairs. Very few models even consider the dynamics of prions in yeast cells [23, 35], while no model at this time explicitly models the dynamics of prions in growing yeast colonies. Constructing a model that can provide insight into the dynamics behind the loss of the prion phenotype in yeast is ideal for future researchers to explain, at multiple spatial scales, the dynamics of prions in yeast cells and the phenotype structures that form at the colony level.

Chapter 5

Conclusion

In this capstone, I discussed development of a new computational pipeline designed for quantifying sectored yeast colonies found in images of experimental plates. This pipeline is designed for high-throughput segmentation and quantification of sectored yeast colonies from these images. Results show that we are able to obtain acceptable colony counts from plated colony images, given that the images have decent segmentations following the necessary pre-processing. Furthermore, my pipeline also demonstrates that we can obtain sector frequencies comparable to manual annotations from experimentalists. This the first model designed specifically for quantifying sectors stemming from prion dynamics in yeast colonies. The work discussed in this document is a big step for providing researchers a computational framework to gain novel insights into the mechanisms driving prion loss in yeast colonies.

Appendix A

Image Acquisition

Images of plates were obtained from the Serio Lab at the University of Massachusetts, Amherst. All images consist of either one plate or a set of plates, all of which are viewed from the top-down. In all plates, each colony consists of a visible, round cluster containing millions of cells that are growing on an agar medium. In this paper, we use two sets of images–henceforth called image set 1 and image set 2 respectively–containing plates housing anywhere between approximately 80-400 colonies each.

Image set 1 contains five images with one plate per image each containing between 80 to 400 colonies (example in Figure A.1 (middle)). All the colonies in these images are either white ($[PSI^+]$) , red ($[psi^-]$) , or sectored phenotype (a mix of both $[PSI^+]$ and $[psi^-]$). One of these five plates contains a large number of colonies with sectored phenotypes. Colonies analyzed within the entire pipeline as described in Section 2 were manaully assigned a label signifying whether they are quantifiable by experimentalists. Image set 2 contains six images which are similar to those in image set 1 (example in Figure A.1 (right)). The primary differences here are that the colonies are less saturated than in the second class, and that four of these plates contain a significant number of sectored colonies present. Another label was assigned to each colony in the image signifying the colony is quantifiable by experimentalists.

Each of the image sets described here were used for different experiments at different times. One important feature to note across image sets 1 and 2 is variation of color and lighting conditions. Since U-Net is trained on synthetic images whose color is based off the images in set 1, U-Net may not accurately segment colonies from image set 2 because by eye the color profiles are different from what U-Net was trained with. Instead of retraining U-Net to address this issue, we opted toward pre-processing the real images until they appear close to a "standardized" image (see Figure A.2). We implement the color profile transfer scheme written by chia56028 on Github (https://github.com/chia56028/Color-Transfer-between-Images) and adapted it for execution on Google Colab. This code is an implementation of the work by Reinhard et al [48] which transforms a source image by applying onto it the color characteristics of a desired "target" image. The objective for this pre-processing

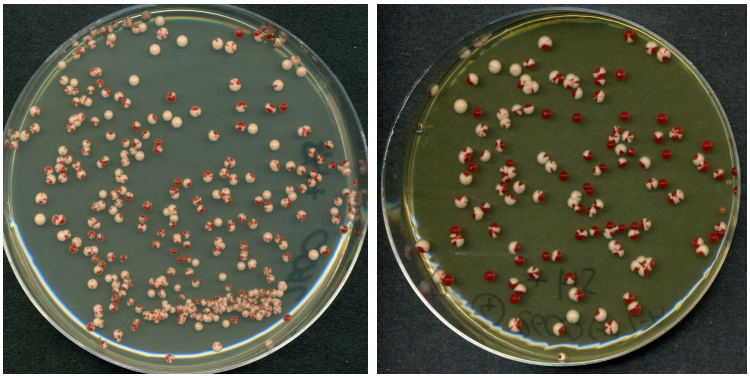


Figure A.1: **Yeast colony images.** Example images of the yeast colony plates in image set 1 (left) and image set 2 (right) taken by the Serio Lab at the University of Massachusetts, Amherst.

step is to ensure that the images in set 2 have similar color features as the images in set 1 so that U-Net will produce similar quality output segmentations. What this method does is produce an image whose color profile is more similar to that of the target image (see Figure A.2).

For the purpose of this work, we chose the target image to be the image of the one sectored plate in image set 1 (also shown in Figure A.1 (left)). All six images in set 2 were used as source images for the color transfer scheme before input to U-Net. No pre-processing was done on image set 1 because these images have the color profiles that U-Net was originally trained on. No adjustments in brightness and contrast were applied to these images before or after the color transfer scheme was applied. While the difference between the color profiles in the original and pre-processed images in set 2 are, admittedly, quite subtle visually, their output segmentations are significantly different. In particular, the segmentation of the preprocessed images display obvious quality improvements such that many more colonies can be discerned (see Figure A.3). Most of the colonies present in the output segmentations were sufficient enough for the classification scheme as described in Section 2.2.

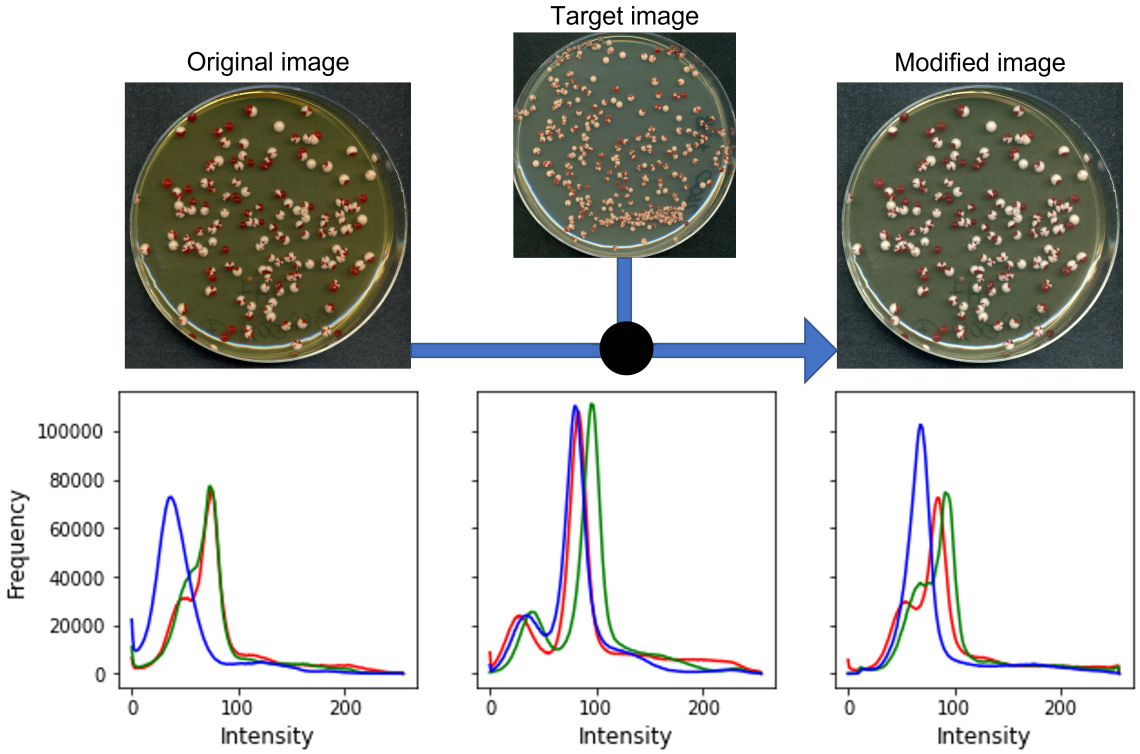


Figure A.2: **Example output of the color transfer method.** Top: The original image is modified so that the color profile of the original image is closer to the color profile of the target image. Bottom: The RGB color distributions of the red, green, and blue channels of each image.

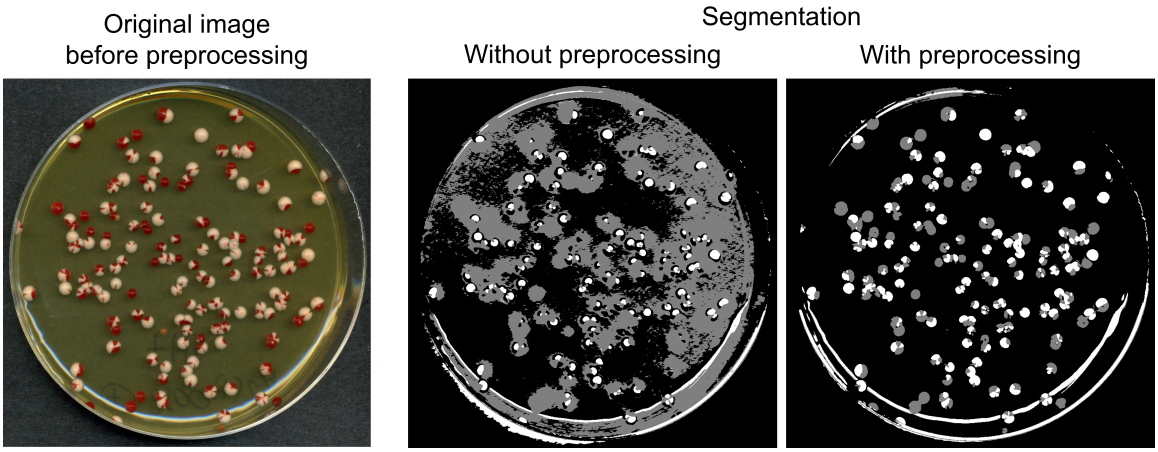


Figure A.3: **Using color transfer as preprocessing aids in improving quality of the output segmentation.** (Left): An example image from set 2 before any pre-processing was done. (Middle): The output segmentation using the original image as input. (Right): The output segmentation of the pre-processed image by first applying the color transfer method to the original image as described in Appendix A before subsequently being used as input to U-Net.

Appendix B

Synthetic Image Generation

In this section, I discuss the procedure for creating the images used for training U-Net to perform image segmentation.

Due to the lack of hand annotated colony images, I turn to training a neural network with synthetic images where I can easily create ground-truth masks labeling each pixel. In Figure B.1, I show an example of a synthetic image generated with its corresponding ground-truth mask. My approach involves generating sets of synthetic images of yeast colonies which display the key features of the yeast colonies found in the experimental images. Each colony in one experimental image has sectored red and white regions comprising each colony, with slight variation within the color gamut. I use two representative colors (1 red and 1 white) to fill each circle representing the colony, where the circle is filled with the white color and the red sector overlayed. Since the colonies rest on a plate in each image, I also select three representative colors for the background corresponding to the interior of the plate, the border of the plate, and the table on which the plate rests respectively. Each color selected corresponds to an RGB vector $[R, G, B]$ such that $R, G, B \in [0, 255]$.

For each synthetic image, two representations as well as four masks are generated, each with size 1024x1024. The two representations of each image include one with Poisson noise and one without. The images containing Poisson noise are used for training U-Net in Section 2.1, while the images without Poisson noise are to simplify the process for creating the associated ground-truth masks. The four masks created label 1) the colony pixels, 2) the white colony pixels, 3) the red colony pixels, 4) the red and white colony pixels merged, and 5) the number of sectors in each colony. The first three masks are created through a series of grayscale conversions and binary thresholding operations on the image at intermediate steps of the process. the The fourth mask is used as the ground-truth mask for training U-Net, while the fifth mask is used to assess the accuracy of my pipeline in quantifying the frequency of sectors in each colony (see Section 2.2).

For each synthetic image, the process for creating the noisy/noiseless representations and ground-truth masks is as follows:

1. We first initialize the image by changing the color of the background represented

by the RGB vector $[54, 54, 68]$. This element represents the tabletop at which the plate rests.

2. A circle of radius 30 whose center coincides with the image center is generated above the background and filled with the color represented by the RGB vector $[137, 155, 160]$. This element represents the body of the plate.

3. 100 points are uniformly sampled inside the circle generated in step 2 such that the minimum distance between any two points is at least 2. Then, circles of radius 1 are generated whose centers coincide with the sampled points. Each circle is then filled with the color represented by the RGB vector $[221, 217, 199]$. These elements represent the colonies on the plate.

4. Two circles of radius 29 and 31, each with the same center as the circle generated in step 2 are generated. The space in between the circles is filled with the color represented by the RGB vector $[105, 107, 152]$. This element represents the part of the background corresponding to the border of the plate.

5. An image of size 1024x1024 is saved. Then, binary thresholding is performed on the resulting image following a grayscale transformation. The result is the final ground-truth mask representing colony pixels.

6. For each circle generated in step 3, two points are uniformly selected on the circle, and lines connect those two points independently with the center of the circle. The space in between is filled with the color represented by the RGB vector $[148, 36, 23]$. This element represents the red region of a colony. For circles where n sectors will be generated, $2n$ points are uniformly selected, and the process described here is performed for each pair of points along the length of the circle.

7. Step 4 is repeated to regenerate the border above the colonies.

8. An image of size 1024x1024 is saved. The result is the noiseless representation of the synthetic image.

9. Binary thresholding is performed following a grayscale transformation on the image from step 8. The result is the final ground-truth mask representing white colony pixels only.

10. The white colony mask form step 9 is subtracted from the full colony mask in step 5. The result is the final ground-truth mask representing red colony pixels only.

11. Since the red colony pixel and white colony pixel masks are fully disjoint, we merge the two masks, assigning the label 1 to white colony pixels and 2 to red colony pixels while all background pixels are labeled 0. The result is the final

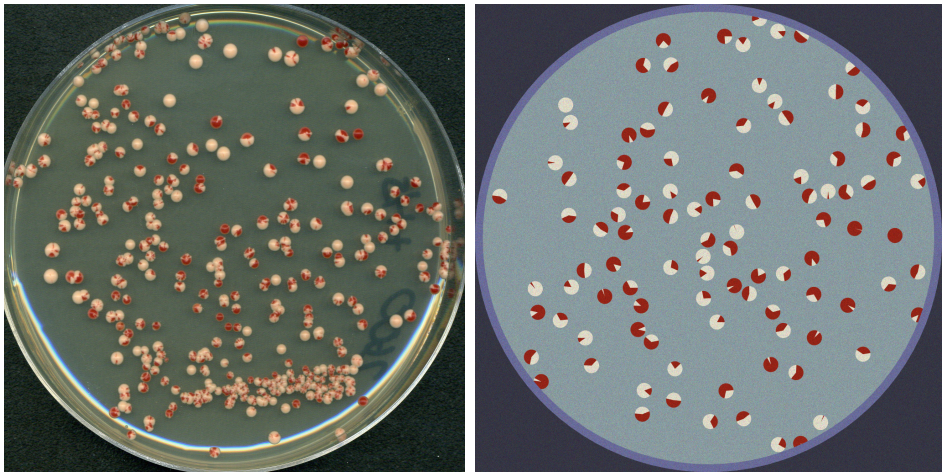


Figure B.1: **Synthetic Representation of Experimental Images.** (Left): Experimental image of yeast colonies with both red and white phenotype obtained with permission from the Serio lab. (Right): Synthetic image of yeast colonies with both red and white phenotypes generated using Matlab.

ground-truth mask showing the locations of red and white colony pixels and is used for training U-Net (Section 2.1).

12. An additional mask is created at the center of each colony which shows a small square whose label is the number of sectors generated plus 1. The result is saved as an image of size 1024x1024. This represents the true labels for the frequency of sectors in each colony within the synthetic image and is used to assess the performance of my pipeline (Section 2.2).

13. Finally, from step 8, the image is given Poisson noise, then saved with size 1024x1024. The result is the noisy representation of the synthetic images that is used for training U-Net (Section 2.1).

Appendix C

Circle Detection using Circle Hough Transform

Colony counting is usually a component of an experiment done to obtain a measure for the viability of microbial samples [5, 13]. However, manual counting of colonies is often time-consuming, non-reproducible, and at risk of inaccuracy for large samples [5, 10, 13]. This challenge, coupled with the advancement of modern technologies, has motivated the development of techniques in computer vision designed to automate colony counting. In order to detect and count colonies in our images, we use the circle Hough transform [49, 64], an adaptation of the original Hough transform [24] which was introduced for detecting circular objects of pre-defined radii in images. I also explain the limitations of this method on images that contain both red and sectored colonies and how our pipeline in this work addresses this insufficiency.

Figure C.1 shows how the circle Hough transform is used to find circles of pre-determined radii. I implemented the circle Hough transform using Octave's function `imfindcircles` within the `image` package. As pre-processing, this function converts color images to grayscale using the standard luminosity method, then uses a Sobel edge detector on the grayscale image, resulting in a binary image whose pixels are white if there is high contrast in either the vertical or horizontal directions, and black otherwise (Figure C.1 B). The white pixels (edges) are used to cast votes for where to find circles of a fixed size. These locations are then stored in a new array called the accumulator. The accumulator is generated using the Atherton-Kerbyson method [2] and the location of local maxima and circle radii are approximated as post-processing to obtain detected circles in the original image (Figure C.1 C). This function also includes a parameter between 0 and 1 for sensitivity to adjust the threshold at which any local maxima in the accumulator have to exceed in order to be considered a center of a circle, where 0 means only perfect circles can be detected, and higher values allow for more imperfect circular objects to be detected.

For our images, we set the sensitivity parameter to 0.95 to allow `imfindcircles` to detect adequately imperfect circular objects in the output segmentations that U-Net produces in Section 2.1. We do not use this function on entire images however

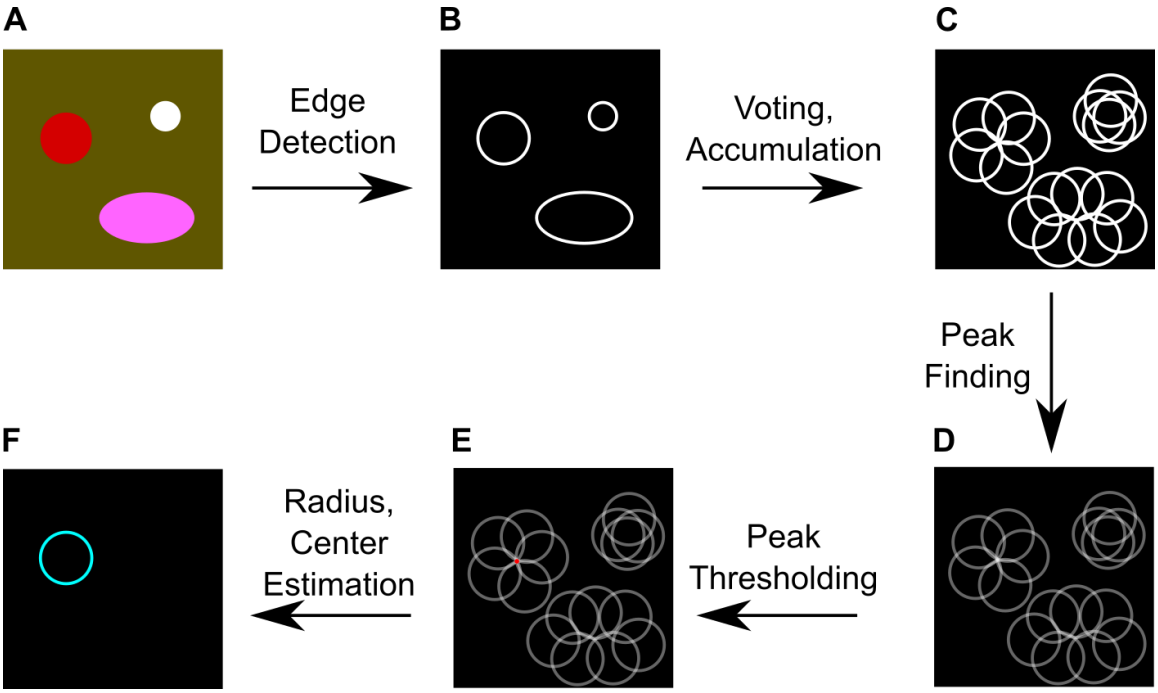


Figure C.1: **Application of the "classical" circle Hough transform to find circles of a pre-defined radius.**
A: An example image in which we want to find circles of a certain radius.
B: Edges of objects in the original image are found using an edge detection method such as Canny [9] or Sobel.
C: Applying the circle Hough transform to the edge-only image. Each edge pixel in B becomes the center of a circle with a pre-defined radius.
D: At each pixel in C, the number of circles passing through are recorded in an intensity map such that pixels with higher intensity (white) have more circles passing through them and pixels with lower intensity (gray to black) have less or no circles passing through them.
E: Local maxima are filtered out based on their value in the intensity map. The desired peaks found in D are kept only if the number of intersecting circles in C exceed a threshold. The locations of pixels where the number of circles passing through exceeds this threshold are shown as red dots.
F: At each local maximum found in E, post-processing is performed to estimate the center of the circle. Depending on the method used, an additional step to estimate the radius is also performed. The location of the detected circle is shown in cyan and corresponds to the location of the original circular object in A.

since all the images are are using contain mostly background pixels. To decrease computational time, we only use `imfindcircles` on the pixels inside the bounding boxes whose conditions in Section 2.1.3 are met.

The circle Hough transform has two significant drawbacks. First, implementation of this method requires conversion of the original image to a single-channel image (i.e. grayscale), which is standard to both simplify the result and reduce computational time. However, it is important to note that a reduction in the number of channels in an image can eliminate a significant amount of information. Colonies which visually appear distinct from the background by eye in colored images may not be distinct enough when the image is reduced to a single channel. The second drawback is a consequence of the first, in that the circle Hough transform relies on the result of an edge detection algorithm to build the accumulator. This imposes a requirement for colonies in the single-channel image to have an adequate amount of contrast with the background on which they rest in order for an edge detector to correctly segment colonies. If the circular object of interest does not contrast well with the background, the circle Hough transform will fail to detect that object. Moreover, since the edge detector is applied to an entire image, edges can exist in areas other than on the objects of interest. As a result, this can lead to false detections of other "circle-like" objects not intended for detection, thus affecting the accuracy of the circle Hough transform.

While the circle Hough transform is effective at detecting and counting white colonies on the plate background, exhaustive testing suggests that this method is less effective at detecting red and sectored colonies (see Figure C.2). The primary issue occurs through the RGB to grayscale transform required in the algorithm where red colony pixels and background do not contrast well enough. As a result, there are typically not enough edge pixels for the circle Hough transform to detect these colonies. In the case of sectored colonies, edge pixels can be found at the interface between red and white regions, as both red pixels and background pixels on the plate contrast well against white colony pixels. The consequence is that the circle Hough transform may detect small regions in the colony that contrast well enough, resulting in a "partial" detection that excludes most the red regions of the colony.

Due to these limitations, it is necessary to simplify the image by transforming it in a way where all colony pixels contrast well with the background. To do this, we first perform semantic segmentation on the original images using U-Net as described in Section 2.1 to label the colony pixels corresponding to red and white regions of colonies as well as the background. We then associate these labels with arbitrary color such that the the background contrasts significantly with the color assigned to the colony pixels. We then use the circle Hough transform on this image to obtain the locations of colonies to use for the remainder of our pipeline.

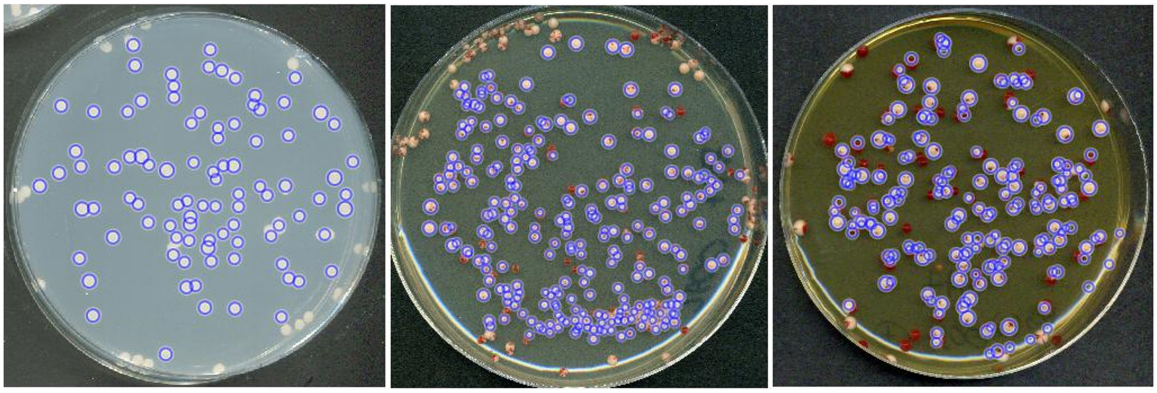


Figure C.2: **Colonies detected using the circles Hough transform.** All circle detections obtained are shown in blue. Testing suggests that the circle Hough transform performs well at detecting fully-white colonies in the left image. However, the same method does not perform well as well at detecting red and sectored colonies in the middle and right images. Many red colonies do not contrast significantly well against the background, and many sectored colonies appear to be either missed completely or were partially detected in that a circle encompasses the white region of a colony but not the red region.
Left: Colonies detected in image containing only white colonies. For `imfindcircles`, sensitivity was set to 0.9, and the radius range was set to be between 7-10 pixels.
Middle: Colonies detected in image from image set 1 containing both red and sectored colonies. For `imfindcircles`, sensitivity was set to 0.95, and the radius range was set to be between 15-35 pixels.
Right: Colonies detected in image from image set 2 containing both red and sectored colonies. For `imfindcircles`, sensitivity was set to 0.95, and the radius range was set to be between 15-35 pixels.

Appendix D

Purity of a Colony and its Regions

This section explains how I define purity in terms of both the colony and each of its red and white regions. I then explain how this metric is used in the annotation procedure (Section 2.2) to look for probable sectors and to aid in improving the accuracy of quantifying sectors.

It is important to note that the pipeline requires the output segmentation. As a consequence, it is highly possible that the segmentation will contain inconsistencies between itself and the true image by eye. In particular, the interface between the red and white regions of the colony may not be well reflected in the segmentation, nor the interface between the colony and background. To measure how well-defined the region is, we need to analyze the physical structure of the region itself and look for inconsistencies in the segmentation that can be addressed using simple methods. To that end, we define a metric we call "purity" to denote the proportion of pixels in each red/white region that are of the same class. Furthermore, since by eye each region appears sector-like, this purity metric can give us information about the "sectoriness" of colony segmentations.

We first define purity in terms of a single region of a colony. After creating the regions as described in Section 2.2.1, the color of the region (red or white) is assigned to be the same color as the pixels in the region along the boundary of the colony. Assume we have a sectored colony that appears sectored with a red regions and b white regions (see Figure D.1 A). We denote red regions as $R_1, \cdots, R_a$ and white regions as $W_1, \cdots, W_b$. We denote the function $N(R_i, red)$ to be the number of red pixels in region R_i (similarly, $N(R_i, white)$ is the number of white pixels in region R_i). The total number of colony pixels in the region is therefore the sum: $N(R_i, red) + N(R_i, white)$. We then define the purity of region R_i with respect to the red pixels as

$$p(R_i, red) = \frac{N(R_i, red)}{N(R_i, red) + N(R_i, white)}. \tag{D.1}$$

$$p(W_i, white) = \frac{N(W_i, white)}{N(W_i, red) + N(W_i, white)}. \tag{D.2}$$

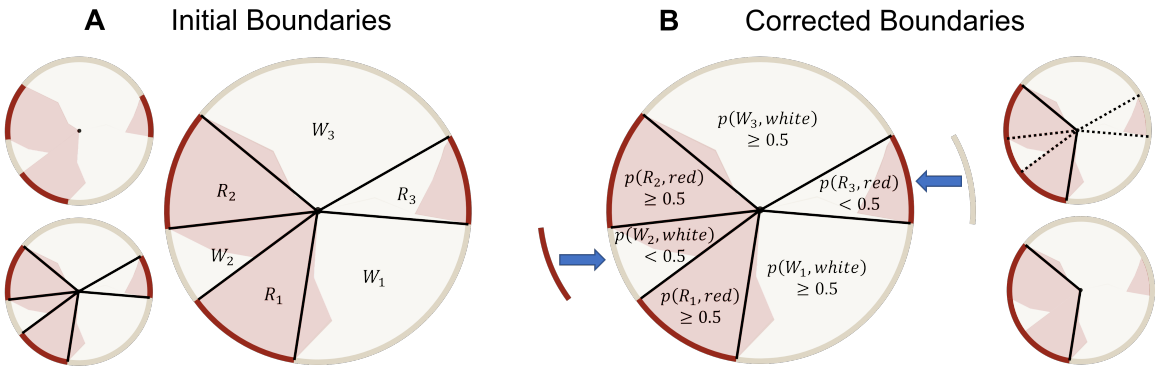


Figure D.1: **Procedure for locating inconsistent boundaries in the segmentation.**
A: Assuming that the colony segmentation is split into red and white pixels (left top), we take the boundary of the colony and find the connected components of the red and white colony pixels respectively. We locate the endpoints of each component which correspond to the interfaces between red and white components, and for each point construct a line segment from that point to the center of the colony (left bottom). The line segments partition the entire colony into idealized regions whose color is defined by the boundary in each region (i.e. R_1 for red and W_1 for white) (right).
B: To find which regions of the colony do not correspond to the interior, we use the purity metric to find the proportion of pixels inside the region that have the same color as the pixels on the boundary in that region. Any regions whose purity metric is less than 0.5 will have the outer boundary change color (left). After the change, adjacent components that have the same color will be merged (right top), and the lines representing the interfaces between the newly merged regions are removed (right bottom).

This metric assumes that the red and white regions estimated are idealized sectors. As such, this metric computes the proportion of pixels inside the region that are labeled with the same color as the region itself. However, depending on the shape of the colony segmentation, regions may not appear sector-like at first, so we include a procedure to partition good and bad idealized regions with respect to the output segmentation.

The condition is as follows: Assume that the red and white regions have been estimated and the purity for each has been obtained using Equations D.1 and D.2. Without loss of generality, if region R_i had a purity of less than 0.5 (i.e. $p(R_i, red) < 0.5$), this suggests that more pixels from the segmentation that belong inside this region region are white. As described in Section 2.2.1, the labels of the pixels along the colony boundary in this region change from red to white. As a consequence, this also changes the assigned color of the region from red to white. Computing the purity of this region after making the change will always result in the purity being at least 0.5.

If sectors are present in the segmentation, by changing the boundary of the region whose purity is less than 0.5, then it will be of the same color as the boundary in any region adjacent to it. Whenever this happens, we merge the associated regions into one region. Using the mediant inequality, it can be shown that if the purity of each of these regions is at least 0.5, then the resulting merged region will also have purity greater than 0.5. For example, if there are n red regions adjacent to each other following the correction, In other words,

$$\begin{aligned} 0.5 \leq \min_{1\leq i\leq n} p(R_i, red) = \min_{1\leq i\leq n} \frac{N(R_i, red)}{N(R_i, red) + N(R_i, white)} \\ \leq \frac{\sum_{i=1}^{n} N(R_i, red)}{\sum_{i=1}^{n} [N(R_i, red) + N(R_i, white)]} \leq \max_{1\leq i\leq n} \frac{N(R_i, red)}{N(R_i, red) + N(R_i, white)} \\ \leq \max_{1\leq i\leq n} p(R_i, red). \end{aligned} \tag{D.3}$$

To define purity for an entire colony, we apply weights to each region to account for size difference between the regions. Without loss of generality for each region R_i and W_j, we assign a weight, $\mu(R_i)$ and $\mu(W_j)$, where

$$\begin{aligned} \mu(R_i) &= \frac{N(R_i, red) + N(R_i, white)}{\sum_{k=1}^{a} [N(R_k, red) + N(R_k, white)] + \sum_{k=1}^{b} [N(W_k, red) + N(W_k, white)]}. \\ \mu(W_j) &= \frac{N(W_j, red) + N(W_j, white)}{\sum_{k=1}^{a} [N(R_k, red) + N(R_k, white)] + \sum_{k=1}^{b} [N(W_k, red) + N(W_k, white)]}. \end{aligned} \tag{D.4}$$

We then define colony purity as the weighted average over all regional purities, i.e.

$$p_w = \sum_{j=1}^{a} p(R_j, red)\mu(R_j) + \sum_{j=1}^{b} p(W_j, white)\mu(W_j) \tag{D.5}$$

or equivalently,

$$p_w = \frac{\sum_{k=1}^{a} N(R_k, red) + \sum_{k=1}^{b} N(W_k, white)}{\sum_{k=1}^{a} \left[N(R_k, red) + N(R_k, white)\right] + \sum_{k=1}^{b} \left[N(W_k, red) + N(W_k, white)\right]}. \tag{D.6}$$

The equation above takes a value between 0 and 1, where values closer to 1 indicate the estimated regions in the colony are collectively more sector-like with respect to the output segmentation.

References

[1] Julia M Anderson and David R Soll. Unique phenotype of opaque cells in the white-opaque transition of candida albicans. Journal of bacteriology, 169(12):5579–5588, 1987. 3

[2] Tim J Atherton and Darren J Kerbyson. Size invariant circle detection. Image and Vision computing, 17(11):795–803, 1999. 4, 40

[3] Sviatoslav Bagriantsev and Susan Liebman. Modulation of aβ 42 low-n oligomerization using a novel yeast reporter system. BMC biology, 4(1):1–12, 2006. 1

[4] Mikahl Banwarth-Kuhn, Jordan Collignon, and Suzanne Sindi. Quantifying the biophysical impact of budding cell division on the spatial organization of growing yeast colonies. Applied Sciences, 10(17):5780, 2020. 31

[5] HR Barbosa, MFA Rodrigues, CC Campos, ME Chaves, I Nunes, Y Juliano, and NF Novo. Counting of viable cluster-forming and non cluster-forming bacteria: a comparison between the drop and the spread methods. Journal of Microbiological Methods, 22(1):39–50, 1995. 40

[6] Prashant Bharadwaj, Ralph Martins, and Ian Macreadie. Yeast as a model for studying alzheimer's disease. FEMS yeast research, 10(8):961–969, 2010. 1

[7] Sushma Basavaraj Bommanavar, Sachin Gugwad, and Neelima Malik. Phenotypic switch: The enigmatic white-gray-opaque transition system of candida albicans. Journal of oral and maxillofacial pathology: JOMFP, 21(1):82, 2017. 3

[8] J. E. Bresenham. Algorithm for computer control of a digital plotter. IBM Systems Journal, 4(1):25–30, 1965. 14

[9] John Canny. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, 8(6):679–698, 1986. 4, 41

[10] Sarah H Carl, Lea Duempelmann, Yukiko Shimada, and Marc Bühler. A fully automated deep learning pipeline for high-throughput colony segmentation and classification. Biology Open, 9(6), 2020. 5, 6, 9, 30, 31, 40

[11] Sean M Cascarina and Eric D Ross. Yeast prions and human prion-like proteins: sequence features and prediction methods. Cellular and Molecular Life Sciences, 71(11):2047–2063, 2014. 1

[12] Heng-Da Cheng, X_ H_ Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. Pattern recognition, 34(12):2259–2281, 2001. 4

[13] Pei-Ju Chiang, Min-Jen Tseng, Zong-Sian He, and Chia-Hsun Li. Automated counting of bacterial colonies by image analysis. Journal of microbiological methods, 108:74–82, 2015. 40

[14] John Collinge. Molecular neurology of prion disease. Journal of Neurology, Neurosurgery & Psychiatry, 76(7):906–919, 2005. 1

[15] Jason K Davis and Suzanne S Sindi. A mathematical model of the dynamics of prion aggregates with chaperone-mediated fragmentation. Journal of mathematical biology, 72(6):1555–1578, 2016. 3

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 5

[17] Alessandro Ferrari, Stefano Lombardi, and Alberto Signoroni. Bacterial colony counting with convolutional neural networks in digital microbiology imaging. Pattern Recognition, 61:629–640, 2017. 31

[18] Corey Frazer, Aaron D Hernday, and Richard J Bennett. Monitoring phenotypic switching in candida albicans and the use of next-gen fluorescence reporters. Current protocols in microbiology, 53(1):e76, 2019. 3

[19] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv:1704.06857, 2017. 4

[20] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. Fundamental concepts of convolutional neural network. In Recent trends and advances in artificial intelligence and Internet of Things, pages 519–567. Springer, 2020. 5

[21] John S Griffith. Nature of the scrapie agent: Self-replication and scrapie. Nature, 215(5105):1043–1044, 1967. 1

[22] Randal Halfmann, Daniel F Jarosz, Sandra K Jones, Amelia Chang, Alex K Lancaster, and Susan Lindquist. Prions are a common mechanism for phenotypic inheritance in wild yeasts. Nature, 482(7385):363–368, 2012. 3

[23] A Ali Heydari, Suzanne S Sindi, and Maxime Theillard. Conservative finite volume method on deforming geometries: the case of protein aggregation in dividing yeast cells. Journal of Computational Physics, page 110755, 2021. 31

[24] Paul Hough. Method and means for recognizing complex patterns, December 1962. US Patent 3,069,654. 40

[25] Charles R Hutti, Kevin A Welle, Jennifer R Hryhorenko, and Sina Ghaemmaghami. Global analysis of protein degradation in prion infected cells. Scientific reports, 10(1):1–13, 2020. 1

[26] Daehee Hwang, Inyoul Y Lee, Hyuntae Yoo, Nils Gehlenborg, Ji-Hoon Cho, Brianne Petritis, David Baxter, Rose Pitstick, Rebecca Young, Doug Spicer, et al. A systems approach to prion disease. Molecular systems biology, 5(1):252, 2009. 1

[27] Takao Ishikawa et al. Saccharomyces cerevisiae in neuroscience: how unicellular organism helps to better understand prion protein? Neural Regeneration Research, 16(3):489, 2021. 1

[28] Ramesh Jain, Rangachar Kasturi, Brian G Schunck, et al. Machine vision, volume 5. McGraw-hill New York, 1995. 4

[29] Sin-Ho Jung. Stratified fisher's exact test and its sample size calculation. Biometrical Journal, 56(1):129–140, 2014. 14

[30] Mehdi Kabani and Ronald Melki. Yeast prions assembly and propagation: contributions of the prion and non-prion moieties and the nature of assemblies. Prion, 5(4):277–284, 2011. 3

[31] Courtney L Klaips, Megan L Hochstrasser, Christine R Langlois, and Tricia R Serio. Spatial quality control bypasses cell-based limitations on proteostasis to promote prion curing. eLife, 3:e04288, 2014. 3, 31

[32] Hideyasu Kuniba and Roy S Berns. Spectral sensitivity optimization of color image sensors considering photon shot noise. Journal of Electronic Imaging, 18(2):023002, 2009. 28

[33] Michael R Lamprecht, David M Sabatini, and Anne E Carpenter. Cellprofiler™: free, versatile software for automated biological image analysis. Biotechniques, 42(1):71–75, 2007. 5

[34] Paul Lemarre, Laurent Pujo-Menjouet, and Suzanne S Sindi. Generalizing a mathematical model of prion aggregation allows strain coexistence and co-stability by including a novel misfolded species. Journal of mathematical biology, 78(1):465–495, 2019. 3

[35] Paul Lemarre, Laurent Pujo-Menjouet, and Suzanne S Sindi. A unifying model for the propagation of prion proteins in yeast brings insight into the [psi+] prion. PLoS computational biology, 16(5):e1007647, 2020. 3, 31

[36] Liming Li and Anthony S Kowal. Environmental regulation of prions in yeast. PLoS pathogens, 8(11):e1002973, 2012. 1

[37] Douglas R Lyke, Jane E Dorweiler, and Anita L Manogaran. The three faces of sup35. Yeast, 36(8):465–472, 2019. 1

[38] Joanna Masel, Vincent AA Jansen, and Martin A Nowak. Quantifying the kinetic parameters of prion replication. Biophysical chemistry, 77(2):139–152, 1999. 3

[39] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence, 2021. 5

[40] Carey D Nadell, Kevin R Foster, and Joao B Xavier. Emergence of spatial structure in cell groups and the evolution of cooperation. PLoS Comput Biol, 6(3):e1000716, 2010. 3

[41] Yoshiko Nakagawa, Howard C-H Shen, Yusuke Komi, Shinju Sugiyama, Takaaki Kurinomaru, Yuri Tomabechi, Elena Krayukhina, Kenji Okamoto, Takeshi Yokoyama, Mikako Shirouzu, et al. Amyloid conformation-dependent disaggregation in a reconstituted yeast prion system. Nature Chemical Biology, 18(3):321–331, 2022. 3

[42] Martin A Nowak, David C Krakauer, Aron Klug, and Robert M May. Prion infection dynamics. Integrative Biology: Issues, News, and Reviews: Published in Association with The Society for Integrative and Comparative Biology, 1(1):3–15, 1998. 1

[43] Nobuyuki Otsu. A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1):62–66, 1979. 4

[44] Toyah Overton and Allan Tucker. Do-u-net for segmentation and counting. In International Symposium on Intelligent Data Analysis, pages 391–403. Springer, 2020. 9

[45] Stanley B Prusiner. Novel proteinaceous infectious particles cause scrapie. Science, 216(4542):136–144, 1982. 1

[46] Stanley B Prusiner. Molecular biology and pathogenesis of prion diseases. Trends in biochemical sciences, 21(12):482–487, 1996. 1

[47] Stanley B Prusiner. Prions. Proceedings of the National Academy of Sciences, 95(23):13363–13383, 1998. 1

[48] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. IEEE Computer graphics and applications, 21(5):34–41, 2001. 33

[49] Mohamed Rizon, Haniza Yazid, Puteh Saad, Ali Yeon Md Shakaff, Abdul Rahman Saad, Masanori Sugisaka, Sazali Yaacob, M Rozailan Mamat, and M Karthigayan. Object detection using circular hough transform. 2005. 40

[50] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015. 5, 7, 9

[51] Danny Salem, Yifeng Li, Pengcheng Xi, Hilary Phenix, Miroslava Cuperlovic-Culf, and Mads Kaern. Yeastnet: Deep-learning-enabled accurate segmentation of budding yeast cells in bright-field microscopy. Applied Sciences, 11(6):2692, 2021. 5

[52] Prasanna Satpute-Krishnan and Tricia R Serio. Prion protein remodelling confers an immediate phenotypic switch. Nature, 437(7056):262–265, 2005. 3

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 5

[54] Suzanne S Sindi. Mathematical modeling of prion disease. Prion-an overview, InTech, pages 207–227, 2017. 3

[55] Simrandeep Singh, Nitin Mittal, Diksha Thakur, Harbinder Singh, Diego Oliva, and Anton Demin. Nature and biologically inspired image segmentation techniques. Archives of Computational Methods in Engineering, pages 1–28, 2021. 4

[56] Michael G Smith and Michael Snyder. Yeast as a model for human disease. Current protocols in human genetics, 48(1), 2006. 1

[57] Kinshuk Raj Srivastava and Lisa J Lapidus. Prion protein dynamics before aggregation. Proceedings of the National Academy of Sciences, 114(14):3572–3577, 2017. 1

[58] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3147–3155, 2017. 5

[59] Mick F Tuite and Brian S Cox. The genetic control of the formation and propagation of the [psi+] prion of yeast. Prion, 1(2):101–109, 2007. 1

[60] Graham JG Upton. Fisher's exact test. Journal of the Royal Statistical Society: Series A (Statistics in Society), 155(3):395–402, 1992. 14

[61] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17:261–272, 2020. 12

[62] Joel C Watts, Aru Balachandran, and David Westaway. The expanding universe of prion diseases. PLoS pathogens, 2(3):e26, 2006. 1

[63] Reed B Wickner and Amy C Kelly. Prions are affected by evolution at two levels. Cellular and molecular life sciences, 73(6):1131–1144, 2016. 1

[64] HK Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. Image and vision computing, 8(1):71–77, 1990. 40