

DSE-260

Alzheimer's Research

June 5, 2020

Ashok Mondal
asmondal@ucsd.edu

Daniel Silva
dasilva@ucsd.edu

Doreh Kazemisefat
dkazemis@ucsd.edu

Matthew Bond
m1bond@ucsd.edu

Peishan Sha
psha@ucsd.edu

Steve Peltier
peltier@ncmir.ucsd.edu

Matthew Madany
madany@ucsd.edu

Mark Ellisman
mellisman@ucsd.edu

Abstract

The microarchitecture of the brain during Alzheimer's and other neurodegenerative diseases can provide novel insights into underlying disease mechanisms and pathogenesis. The ability to accurately analyze and draw insights from existing large volumes of 2-dimensional electron microscopy or electron tomography images is frequently limited by inconsistent image quality. Current open source and proprietary tools to convert 2 dimensional images from stacks or voxels into 3-dimensional reconstructions are bottlenecked by the quality of the input data. In cases where image qualities are low, or the structure of the brain is abnormally altered due to disease or tissue processing, images must be pre-processed before rendering can take place. This can result in sub-optimal reconstructions.

Here we present a stand-alone tool based on mouse brain samples that, when combined with our image processing pipeline, model retraining, and 3-dimensional image rendering, improves image segmentation and component predictions. Membrane prediction performance is increased by 13.59% (F-Beta) and 20.23% (F-1) and mitochondria prediction performance is increased by 22.82% (F-Beta) and 22.22% (F-1). Rendered images are manipulatable in real time, with component selection, zoom pan and tilt manipulation as well luminosity and opacity controls. To ensure biological relevance of the pre-processing application we provide an option for subject matter expertise input allowing for the model to be retrained in real time.

We validate the performance of this pipeline utilizing an example mouse model of Alzheimer's disease data set and demonstrate that it can be efficiently scaled via a remote computer cluster environment to eliminate individual computer resource artificial thresholds. This will drastically improve the speed at which conclusions that can be drawn regarding the microarchitecture of the brain.

ACKNOWLEDGMENT

The authors wish to thank the UCSD NCMIR personnel and especially and specifically Matthew Madany for unparalleled assistance, guidance and patience. Our work efforts would have been without direction or understanding without their help.

Additional thanks are extended to the UCSD MAS program office for their support and work, Dr. Ilkay Altinas for direction and feedback and of course our families for patience, understanding and backing throughout our time.

I. INTRODUCTION

ALZHEIMER'S disease (ALZ) is a type of brain disease, just as coronary artery disease is a type of heart disease. As a degenerative disease the underlying brain pathologies develop in a time dependent manner. Alzheimer's disease is thought to begin 20 years or more before symptoms arise, with small changes in the brain that are unnoticeable to the person affected. Only after years of brain changes do individuals experience noticeable symptoms, such as memory loss and language problems. Symptoms occur because nerve cells (neurons) in parts of the brain involved in thinking, learning and

memory (cognitive function) have been damaged or destroyed [1]. Additionally, connections between parts of the brain (i.e. neurons, synapses, etc.) can be altered or lost.

Data imaging of Alzheimer's affected brain cells started in the 1960's. Dr. Robert Terry (January 14, 1924 - May 20, 2017) is remembered as an early pioneer in imaging of brain cells, specifically targeted at classifying, identifying with the eventual hope of understanding Alzheimer's Disease. Thanks to Dr. Terry and others, brain samples (taken from living patients) acquired in the 1960s are still available and usable now. In the 1980's advancements in technology and renewed interest led to pioneering research into 3-dimensional (3D) modeling of brain cells.

II. CHALLENGES, OPPORTUNITIES AND QUESTION FORMULATION

A. Challenge

Early 3D renderings of cells were groundbreaking but extremely computationally expensive. Early 3D rendering of cells was groundbreaking but extremely computationally expensive. Persons with high levels of understanding were required to label every feature of each cell by hand, often forced to make best approximations due to the thickness of each sample and how cell structure could move/change across the Z dimension when samples were taken. Typical samples have a thickness in the Z-direction of 100 nm, allowing for cell structures to change in non-intuitive manners between samples, especially in compromised cells.

Differences in sample preparation, labeling, and data acquisition techniques as evidenced by a multitude of non-standardized procedures/SOPs can result in diverging resolutions, clarity and contrast of collected images. Without a standardized sample preparation protocol there will be variability in sample handling including: perfusion techniques, the chemical composition of the fixative utilized, duration of sample fixation and post-fixation storage. These fixation differences impact cell membrane and organelle membrane quality. Labeling of samples for Electron Tomography and Scanning Electron Microscopy can be accomplished using heavy-metal staining and labeling techniques. Labels can be included pre or post fixation. As a result of sample preparation differences, there can be drastic variations in labeling quality and label density in regions of interest. In addition to potential loss of resolution and increased variability introduced by non-standardized sample handling methods the equipment selected for data acquisition is critical. For example, two common data acquisition techniques (Electron Tomography and Scanning Electron Microscopy) result in a significantly different resolution of collected images. Electron tomography has high resolution, but due to its slow processing speed, the field of view is severely diminished. In comparison the use of a Scanning Electron Microscope results in much faster processing, allowing for a large field of view. This increase, however, comes at the cost of resolution (810x worse than Tomography). In addition, the sample is destroyed during evaluation by the Scanning Electron Microscope. As a result, this makes cross-comparison of samples from different institutions difficult. Image translation networks have the power to represent the features and resolutions of different imaging modalities to fill in information gaps with regards to the cost of extent vs detail.

Scale of data makes analysis of any individual data set extremely difficult without dedicated supercomputer resources. Data can be managed to allow for small subsections to be run locally, allowing for verification of first order models. In communication with the end user, analysis that results in a usable output needs to be fully automated and a good target is to leverage resource parallelization (whether nodes, GPUs, CPUs etc.) to complete whole computational workflows within a week's time.

Setting up the containerized virtual environments that can be staged on high performance resources and installing required libraries is one of the challenges faced by this project. Requirements for

different analysis functions can change between analysis steps. This requires the pipelining of analysis techniques, as well as containerizing the environments with sufficient documentation and integration into the pipeline. Ideally the pipeline will work regardless of operating systems, meaning that containers must be multi-platform. We intend to use Docker for containerization since it is multi-platform.

B. Initial State of the Project

This project is a continuation of an established project that includes multiple Jupyter notebooks. The main notebook takes as input a dataset of images that make up a single sample. The images are stored in active memory while a stored image segmentation model based on CDeep3M runs on the input data.

The CDeep3M image segmentation model [2] is a cloud based pre-trained neural network model designed to take 2-dimensional images from brain scans as input and return predictions on cellular structures such as mitochondria and cytoplasmic membranes. The CDeep3M model is a pre-trained model stored in a model zoo. Due to the pre-training and offsite housing, the model can be directly applied to a set of input data, or the model can be retrained for additional performance. CDeep3M has been developed to run on cloud computing environments, further increasing performance. As we can upload the input data in full into the same environment as the model, the model can run and predict on all input variables consecutively, resulting in more impactful predictions. Maintaining the same environment that houses and analyses the data is also advantageous due to the lack of movement of data between states or resources. Combined with image scaling, overfitting can be avoided when pipelining the CDeep3M model.

The predictions output from CDeep3M are then combined with the output of a Neuron Instance Segmentation. The combination allows for the dataset to be transformed to encapsulate the probabilistic measurement along with the region of the cell. Once a watershed algorithm assigns initial labelling, a cost function is applied and progressively minimized, resulting in a convergence of the labels within the boundary map predicted by the CDeep3M/Instance segmentation combination. It is important to note that the structure of the data storage, in hdf5, allows for chunk tensors to be called, and parallelization to occur when minimizing the cost function. The final output of this is a compressed file containing the dataset that includes labelling for structures bound by the boundary map.

This set of pixels in 3-dimensions (voxels), that can be plotted using Matplotlib. These plots show a flat representation of a three-dimensional structure that can be exported in typical file types such as png, jpg, etc.

C. Questions to be Investigated and Answered

Utilizing the previous segmentation model one can develop critical questions to guide improvements in image processing for these ALZ samples.

- 1) Can the images be modified to meet a minimum benchmark across all data sources?
- 2) Can a pipeline be built for resolution analysis?
- 3) Can this pipeline be run on parallel tracks to fully exploit available resources?
- 4) Can a feedback system be implemented allowing for correcting or retraining of existing models?
- 5) Can we implement methodologies for quantitative model analysis?
- 6) Can we expand or better existing visualization methods?

D. Related Works

There are other approaches to applying various machine learning algorithms and approaches in both Alzheimer’s research and medical imaging generally. T. Liebmann et al [3] made a concerted effort to combine various scanning, staining and imaging techniques with python-based visualizations. We differ our procedure by the use of the Cycle-Consistent Adversarial Networks (CycleGANs) for image pre-processing. Advantages of implementing CycleGAN models include the ability to train on unmatched pairs of images, not requiring labeled ground truths for training and the model being trained in a fully unsupervised manner.

Due to these advantages, CycleGAN image modification has been involved in multiple human data research. P. Welander et al [4] used multiple CycleGAN based models to create synthetic Magnetic Resonance (MR) images from Computer Tomography (CT) scans of brain tissue. The model performance is analyzed and the results presented. A. S. Becker et al [5] applied CycleGAN image modification to breast cancer object and anomaly detection. CycleGAN models are used to artificially inject or remove suspicious features into existing images in an effort to create test data for radiologists rather than to derive deeper understanding of the input data.

We differentiate ourselves in this project by approaching the implementation of CycleGAN models in an attempt to further enhance the performance of public models designed to label cellular structures, as well as to demonstrate the ability of applying CycleGAN models to distorted brain tissue such as in compromised samples.

III. DEFINING OUR HYPOTHESES

Our hypotheses can be broken down into short term and long-term hypothesis, testing and evaluation. Short term hypothesis testing will allow us to evaluate ongoing progress and ensure that our contributions make a tangible difference.

1) Effect of Reinforced Learning on Model Predictive Behavior (Short Term): Studying differing brain images, the current predictive modelling algorithms are much more effective on certain datasets (Cerebellum images) as compared with Hypothalamus or Cortex images. This is due to many factors. Some factors are completely out of our control (differences in laboratory SOPs, equipment and handling of preserved samples), while some factors are due to decisions made in how predictive models are created, trained and implemented. In certain cases, such as when a model is trained in a domain, the model will perform better when presented with input data of that same domain or similar. This opens an opportunity for retraining to alter the model’s optimal domain, or modification of input data to better reflect the input domain.

To address these differences, we have decided to advance with human in the loop based transfer learning. For subsets of the data, we will allow a human expert in the field to simplistically interact with the output of the algorithm and either confirm the prediction or reject and amend the prediction. The output of the human’s interaction with the prediction will then be fed back into the model for retraining.

We define the following criteria to measure the effectiveness of reinforced learning.

- a. Measure current rate of incorrect predictions on a randomized sample of algorithm outputs (baseline) for each brain image type (cerebellum, hypothalamus, etc).
- b. Introduce reinforced learning tool, allowing expert in the field to correct predictions.

- c. After a set number of reinforcements, measure new rate of incorrect predictions on a randomized set of algorithm outputs.

We expect there to be variability within effectiveness based on brain image type.

2) *Effect of Improved Visualization of End-User Experience (Short Term)*: During interviews with end-users, we have found overall unhappiness with the current visualization methodology. End-users have reported frustration with limitations based on current 3D rendering, specifically with the trouble interacting with rendered data. Users are unable to change perspective, select features, or vary opacity without time-consuming, and often single-use code generation. This results in decreases in efficiency, paired with inability for the end user to fully customize visualizations for output to sponsors/journals/etc. In many biological labs there are few with high mastery of computer science. This results in the necessity of tools to be intuitive, user-friendly, and exist in a GUI interface so that the end users are not interacting with the program via code manipulation. Just the 3D renderings already performed are some of the first 3D renderings in the field.

To address this issue, visualizations with an inbuilt tool for image To address this issue, visualizations with an inbuilt tool for image selection/creation/rendering will be utilized. A tool will be created that allows for selection of features through radio buttons, real-time perspective changing with mouse-based pan, click and rotate tools, and layering effects allowing for changes in opacity, or slicing of 3D renders to allow for better output visuals. Through execution of biomedical computer vision methods, exploration of the reduction of terascale 3D image volumes to robust, repeatable, and generalizable quantitative analysis will occur. Additionally, rather than moving the data between servers/supercomputers, the Nautilus platform is a single platform that can host, process and allow the user to interact with data. This reduces time and cost associated with work.

Figure 1 shows a partial flow chart. These images are from instance segmentation and are 3D image blocks. What we're seeing are the region labels being extracted from this series of images, which were labeled using automated methods.

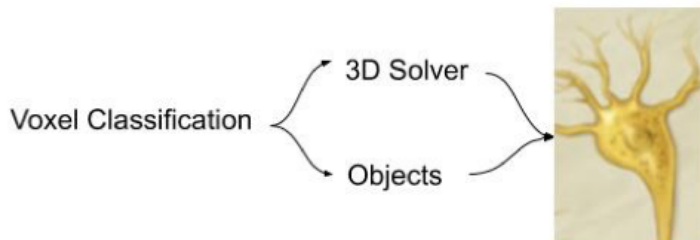


Fig. 1: Partial flow chart of object detection and classification.

The following criteria are defined to measure the effectiveness of our visualization tool.

- a. Complete surveys with current end-users to find desired features and current limitations.
- b. Introduce visualization tools on subset of data, allowing expert in the field to work with tool over a pre-determined timeline.
- c. Repeat survey to find end-user satisfaction/ability with tool.

Effectiveness of our tool will be based on the final end-user survey. Expansion of the tool to more datasets will determine the tool to be effective.

3) *Effect of Image Normalization on Model Predictive Behavior (Long Term)*: Depending on the technique and technology used to create our images, some are of a lower quality than others. In order for the pipeline to work, images must meet a threshold for quality so that predictions can be accurate.

To address this issue, we are implementing a neural-network based image transformation to reconstruct the features of an image into input data that is closer to that of which the classification models are trained on. This results in better predictions and requires less retraining on the models. Scenarios where image quality or resolution is below a desired level can be optimized, as effects will be normalized at the voxel-feature level. Output images will then be fed into our pipeline for analysis.

The following criteria are defined to measure the effectiveness of our visualization tool.

- 1) Measure current rate of incorrect predictions on a randomized sample of algorithm outputs (baseline) for each brain image type (cerebellum, hypothalamus, etc.).
- 2) Introduce reinforced learning tool, allowing expert in the field to correct predictions.
- 3) After a set number of reinforcements, measure new rate of incorrect predictions on a randomized set of algorithm outputs.

Variability within effectiveness based on brain image type are expected.

IV. TEAM ROLES AND RESPONSIBILITIES

Each team member on the project needs to optimize project outcomes using their expertise and interests. While roles will be fluid, especially regarding allocation of computing resources, team members are expected to lead certain aspects of the group efforts.

- Ashok Mondal - Team lead. Ashok will work to aggregate work efforts related to training, retraining, image visualization. Ashok will be primary on model performance analysis and primary on data pipeline construction.
- Daniel Silva - Daniel will be the primary POC in conversations with the advising team, while relaying tasking, observations and concerns from advising/leadership to the team or individual team members as they appear. Daniel will lead the image segmentation tool development and model retraining.
- Doreh Kazemisefat - Project and task management. Doreh will manage the project goals, delegate responsibilities to each team member and oversee the timeliness and tasking through Gitlab's tasking tool. Doreh will be primary for model training efforts and secondary for model performance analysis.
- Matthew Bond - Matt was the primary technical writer including research reports, taking minutes during all external encounters and providing background information and updates on all action item statuses. Matt will lead the volume rendering effort.
- Peishan (Sophia) Sha - Team cohesion and dispute management. Peishan will be integral to keeping people on message, managing internal disputes and addressing staffing/role management. Peishan will act as secondary for image segmentation tool development and primary for outward communication from group to general audience.

In addition to these specific roles and responsibilities, as new challenges or positions appear or are needed, people will have their responsibilities changed, amended and updated to meet both new and existing challenges.

V. DATA SOURCES

Datasets used in this project are primarily from previously acquired mouse brain images. Human brain images are in processing by the National Center for Microscopy and Imaging Research (NCMIR). Due to the scale of the datasets, work will be performed exclusively on cloud computing resources capable of providing sufficient memory and processing capabilities.

- Cerebellum: Image database consisting of 2512 png files. Each file represents one slice of a sample of a cerebellum.
- Cortex_1: Image database consisting of 298 png files. Each file represents one slice of a sample of a cortex.
- Cortex_2: Image database consisting of 140 png files. Each file represents one slice of a sample of a cortex.
- Hypothalamus: Image database consisting of 600 png files. Each file represents one slice of a sample of a hypothalamus.
- dat1: Image database for prediction of specific structures within brain cells. Based on 242 png images, ML algorithms are set to predict the likelihood of structures falling into certain categories.
- SBEM: A typical SBEM dataset consists of individual image slices collected in increments of a slice or section thickness, with x and y resolution is in the range 3-20 nanometers.
- TEM: Segmentation Dataset for Transmission Electron Microscopic Cell Recording.

These datasets are well-representative of the diversity and normal variance associated with data collection. Access to 20 TB of data currently stored in the Cell Image Library (CIL), hosted by UCSD. To date the CIL is expanding with human biopsy from ALZ patients adding additional images. In many cases these new images are of low quality, necessitating image normalization to allow for them to be funneled into the pipeline. Additionally, access has been granted to information such as past Machine Learning algorithms that have been developed, such as supervised learning algorithms intended for small scale classification and identification. These algorithms are stored in the “Model Zoo”, and are available to the public.

Much of the data has been previously cleaned, and should contain relevant and dense datasets that can be routed to the pipeline immediately. Extensive pre-processing will occur, as modification of images ensures compatibility both within datasets and across datasets.

VI. DATA EXPLORATION FINDINGS AND SUMMARY

Exploration of the datasets leads to identification of performance improvement in the image segmentation model. Of the datasets, the Cerebellum dataset presents images of high quality with a good definition of cellular structures which directly leads to good image segmentation predictions. The high quality of the Cerebellum dataset is not inherently due to any one factor, but to the combination

TABLE I: Raw Data Sources

Database name	Source location	Destination in your Data Pipeline	Data Movement and Processing Scripts and Notebooks	Data Size
Cell Image Library (Public) -SBEM -TEM	www.cellimagelibrary.org/cdeep3m		Jupyter Notebook	100 GB
Cerebellum	Google Drive	As Target data in CycleGAN Process	Jupyter Notebook	2.92 GB
Cortex_1	Google Drive	Original images as Source in CycleGAN Process -Generated Images as CDeep3M input for image segmentation	Jupyter Notebook	3.5 GB
Cortex_2	Google Drive	Original images as Source in CycleGAN Process -Generated Images as CDeep3M input for image segmentation	Jupyter Notebook	7.1 GB
Hypothalamus	Google Drive	Original images as Source in CycleGAN Process -Generated Images as CDeep3M input for image segmentation	Jupyter Notebook	1.3 GB

of sample preparation, extraction, imaging and storage, combined with the raw images being of the same domain as the images used to initially train the image segmentation model. Expected variances in dataset quality due to differences in dataset quality due to alterations in methodologies or technique can be seen in Cortex 1, Cortex 2 and Hypothalamus datasets.

The Cerebellum dataset can be utilized as a target for enhancement of other images, training models that will modify other dataset images to meet the quality of the Cerebellum. Initial work will be focused on improvements in the Cortex_1 dataset. This includes the training, testing and implementation of a CycleGAN model with the Cortex_1 data as source and Cerebellum data as target. Hyperparameters to tune include scaling, learning rate and epoch among others.

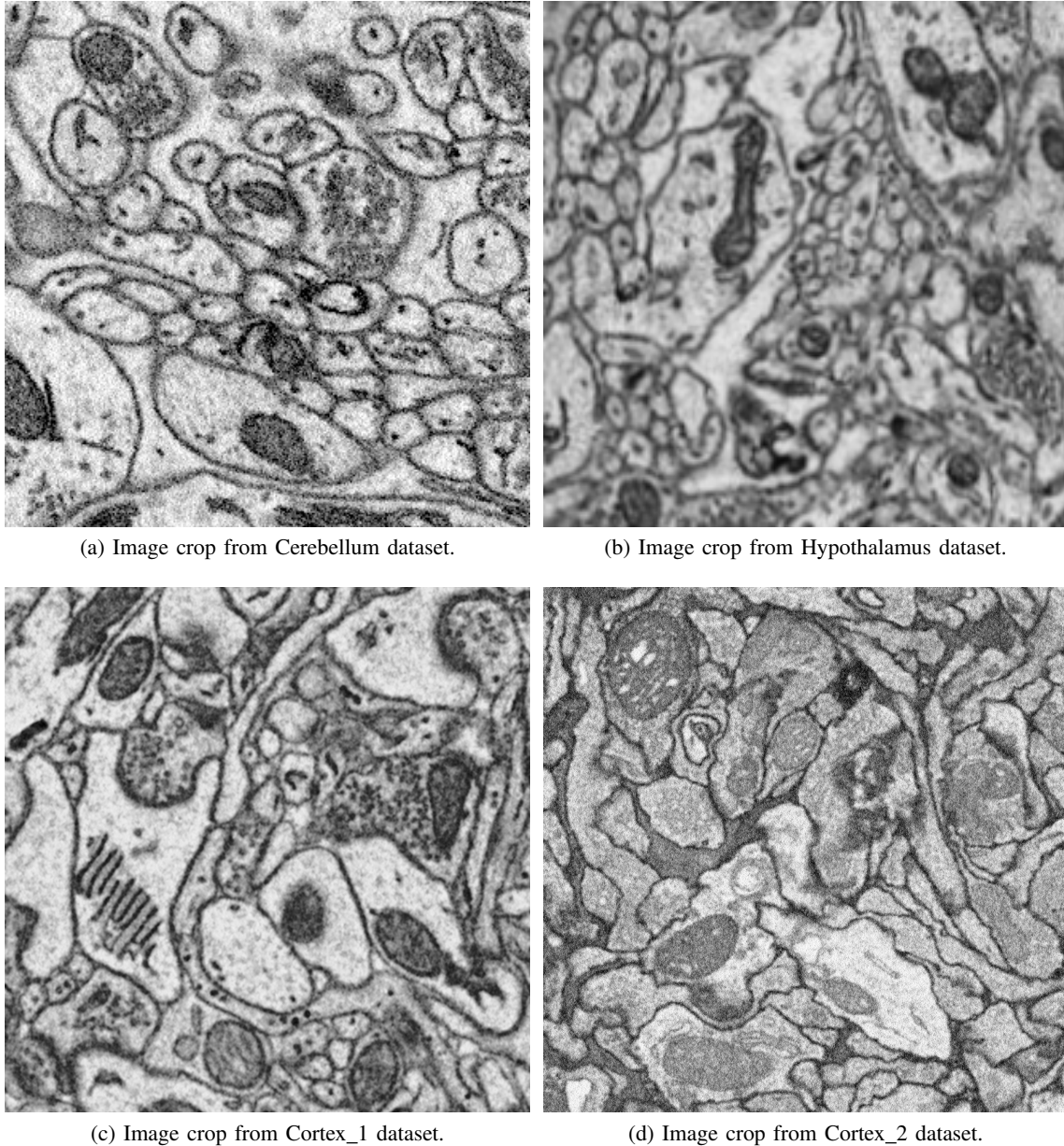


Fig. 2: Image crops from datasets. As seen quality of images differs between datasets. We use the Cerebellum dataset as a target for CycleGAN model.

Once a set of models are trained and implemented, evaluation of model performance can be accomplished. Specifying boundaries through model prediction or human interaction is dependent on the quality of the image. As models are trained and predictions are created, a quantitative analysis technique must be developed. A direct method of comparisons between predicted pixels against the user defined pixels will be utilized. This requires the creation of ground truth data that can be compared against model predictions. Objective measurements such as precision, recall, specificity, etc. can be calculated, and a performance metric must be used to select final CycleGAN model.

Supervised learning for classification of areas or boundaries benefit from human generated feedback. A goal of this project is to enable the end user to choose to interact with the output classifications and

verify or correct the ML algorithm's prediction.

Testing of the outcomes of transfer learning within our models must be evaluated. Due to the human in the loop retraining, samples that the model has not previously seen will be used to enhance the stock segmentation and prediction models. Evaluation of the efficacy of this must be tested and cataloged. As retraining of the CDeep3M model modifies the domain of the predictions, high quality ground truth images can affect model output regarding boundary thickness. Ground truth images may need to be dilated to achieve ideal retraining.

Initial work was with local png files or directories with image files stored uncompressed. These files have been imported into Jupyter IDE's for analysis and processing. Team efforts have been split between improved 3D rendering and user defined bounding. All data and processing will occur on the Kubernetes cluster, negating the need for a separate supercomputer or for implementation on AWS.

The 3D rendering team investigated some common visualization packages, and ipyvolume was selected to be integrated into the pipeline. While the inbuilt mechanisms in ipyvolume allow for simple scatter plots to volumetric rendering, additional work allowing multi volume rendering to be implemented.

VII. APPROACH

Initial exploration has identified the image quality as a key driver of performance. Increasing the image quality directly results in better outcomes from the segmentation model. The first step in increasing output performance is therefore to modify the input images to gain quality. This image normalization must be performed prior to any work on the CDeep3M model or visualizations such as the volume rendering.

Once the input image quality has been increased, the image segmentation model can be modified to better comport with the specific domain in which our datasets exist. Since the CDeep3M model has been trained and hosted off site, performance gains can be made in the retraining of the model specifically with our domain. This retraining will customize the model further and will lead to more accurate segmentations and identifications.

Lastly the output of the models will be visualized for comprehension and analysis. Much has been focused on the 3-dimensional rendering, but in addition visualizations showing the CycleGAN model effect on image inputs, the retrained CDeep3M predictions and any performance metrics will be displayed for consumption and use by the end-user.

VIII. SET UP OF DATA ENVIRONMENT

Due to the scale of the datasets, cloud computing environments are especially key. For example, Cortex_1 dataset contains 298 png images, while each png contains 3500x3500 pixels. This is a total of 3.65 million pixels that must be preprocessed, segmented, watershed, grouped, presented for structure selection and rendered in 3 dimensions.

Working on datasets and voxels of this size is not possible on most personal computers. NCMIR therefore employs an alternative to a localized supercomputer, a distributed cluster known as Nautilus. Nautilus is currently hosted on the Pacific Research Platform kubernetes portal, which we are able to utilize to expand our computing ability. Nautilus is a HyperCluster for running containerized Big Data Applications. It is a large-scale Kubernetes cluster that orchestrates containerized applications amongst

a non-uniform shared pool of resources. This non-uniformity allows for additional compute nodes to be added without there being a hard requirement on specific hardware. In its current state, Nautilus consists of GPU nodes that amount to a total of 524 GPU's. Figure 3 shows a snapshot of the GPU allocation and usage on nautilus.

We take advantage of the scalable nature of Kubernetes through the use of JupyterHub. JupyterHub allows the user to access these computational resources through a spawner that instantiates a Jupyter Lab environment with user defined minimum requirements. The user inputs the minimum GPU allocation, having access to various types of GPUs with differing levels of memory, differing chipsets and differing performance capabilities. These resources are not locked to any specific request or project, and therefore any amount of resources can be allocated automatically or dynamically at runtime. This means automated scaling of the resources (e.g. number of GPU's, CPU's, RAM, etc.), allowing for the user experience to be tailored for each interaction. Additionally the user has access to over a petabyte of storage on the same platform where processes will be run. This makes the Kubernetes cluster a single entry point for data storage, processing and interaction, negating the divide between off-site data storage and supercomputing level resources.

With this flexibility, users are able to complete computational tasks in an efficient manner and a familiar environment that allows them to write portable code with interactive features. Unfortunately, Jupyter has its shortcomings. Since it isn't necessarily an integrated development environment (IDE), it is difficult to debug and create large scale applications.



Fig. 3: Nautilus GPU utilization example.

In order to work in this environment, our code must run in JupyterLab, and the environments must either be pre-loaded, or easily installable either through a docker image or a short YAML or requirements.txt file. JupyterLab is a web-based interactive development environment (IDE) for Jupyter notebooks and code. We take advantage of this JupyterLab environment through the use of JupyterHub. JupyterHub allows the user to access these computational resources through a spawner that instantiates a JupyterLab environment with a user defined set of resources (e.g. number of GPU's, CPU's, RAM, etc.). With this flexibility, users are able to complete computational tasks in an efficient manner and a familiar environment that allows them to write portable code with interactive features. Since these spawned environments come with prepackaged libraries, it enables users to easily share their code and provide reproducible results. Unfortunately, Jupyter has its shortcomings. JupyterLab lacks advanced debugging features that other IDE's provide, which makes debugging code a difficult task.

We take advantage of JupyterLab’s extensions to be able to generate complex visualizations with the ipyvolumes library. By taking advantage of the computational resources on Nautilus, we were able to render Figure 15, which was overly complex for our local machines. To put it in perspective, we requested a JupyterLab instance with multi-GPU powered jupyter interfaces streamed from a nationwide distributed computing grid.

Additionally, since we can perform modelling and rendering at a higher level with multiple GPU’s, we have performed model training and image normalization on the cluster through the use of GPU batch processing. Figure 6 was generated on the cluster while Figure 5 could be generated locally.

IX. DATA PREPARATION

A. Image Normalization

As previously stated, sampling differences across labs, institutions and medical practitioners results in differing qualities and resolutions. Procedures optimized for speed and volume of processed samples incur a reduction in resolution, while procedures with high resolution results are costly with respect to time, limiting the total number of samples.

In our data sets there are also differences in resolution dependent on the type of sample. The dataset Cortex is of an incredibly high quality. We will use this dataset as an ideal example, and will serve as a template for what we want our other datasets to emulate. Increasing the quality of the other datasets will result in better predictive outcomes by the various ML algorithms.

To address and correct this imbalance, we have implemented a cycle-consistent generative adversarial network (CycleGAN) with a deep learning framework, PyTorch, to re-render source datasets, such as cortex and hypothalamus, to look like a target dataset, cerebellum [6], [7]. This technique allows us to alter a supplied image and effectively re-render the image based on a pixel-wise transformation of the feature representations. Examples provided show the recreation of horses as zebras, or the transformation of a photograph into various artists’ styles [8].

CycleGAN solves image to image translation problems even with unpaired training data which are common tasks in many scenarios. The way we do unpaired image to image is we crop the images randomly to create a set of training data and train the data from X to Y with translator G and train the data from Y to X using translator F, where G and F are inverses of each other and both mappings are bijections. This is what we called cycle consistent. Generative Adversarial Networks (GANs) [9], [10] has achieved impressive results in image generation and image editing, and representation learning.

In our approach to Alzheimer images, we use CycleGANs to learn a mapping $G : X \rightarrow Y$ and couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a cycle consistency loss to push $F(G(X)) \approx X$ (and vice versa) as seen in figure 4.

As part of the evaluation of this new algorithm, we need to identify and implement best practices in our model to generate the most expressive data, therefore allowing us to predict boundaries and elements of a cell with higher accuracy. In our first attempt in implementing this methodology, we will train our model on un-paired images. Target images will be of the resolution that we want, but will not be directly mappable to the training images that are fed to the algorithm. To our understanding this methodology is still relatively novel, requiring time and effort to be spent on modifying the hyperparameters of training

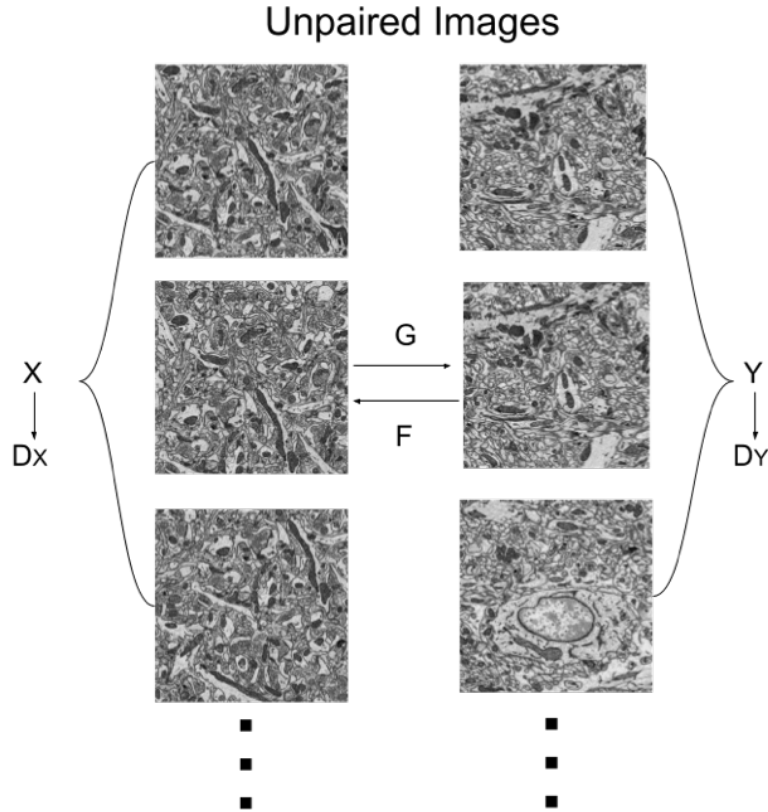


Fig. 4: Two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$ and the associated adversarial discriminators D_X and D_Y

in order to optimize the balance between object detection and false positives.

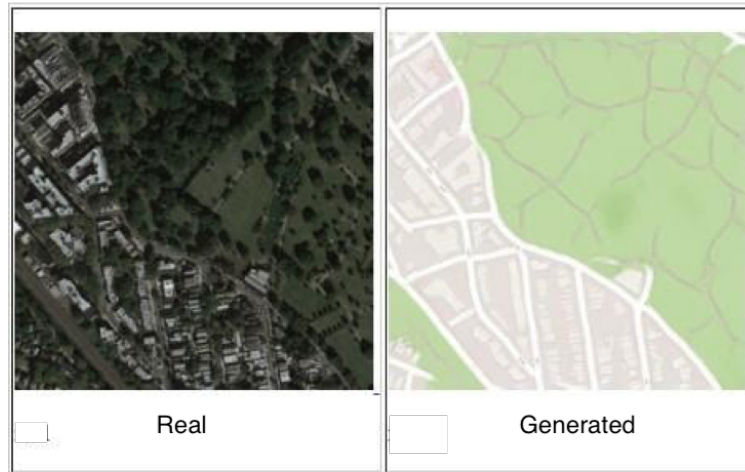
Figure 5 shows some of our initial training, attempting to recreate google satellite views into the style of google maps, allowing us to compare the effect of different numbers of epochs on the output result. Through our investigation of map images, we identified 200 epochs as an optimal balance of resolution and predicted features. It was determined that the 200 epochs was optimal by comparing the images for consistency and limitation of erroneous features. While the image looks different, key markers (roads, bodies of water/tree lines) are in the same places with the same physical features. In the cases where the number of epochs was applied incorrectly, additional features are either not transcribed during the image-to-image transition, or extra features were added that are not correct. These would encapsulate false negatives or false positives respectively.

Moving from our map transformations, we used actual images from cortex and cerebellum to create and train models on various numbers of epochs. Figure 6 shows the outputs of CycleGAN models trained on various numbers of epochs at a scaling factor of 0.6. That is to say the initial image is of size 3500x3500 and the scaled version is of size 2100x2100.

B. Model Description

The following four data sets of the brain are analyzed: Cerebellum, Cortex-1, Cortex-2 and Hypothalamus. As the quality of the Cerebellum data set is relatively higher than other parts of the brain

1021_A



(a) Output of model trained on 175 epochs

1021_A



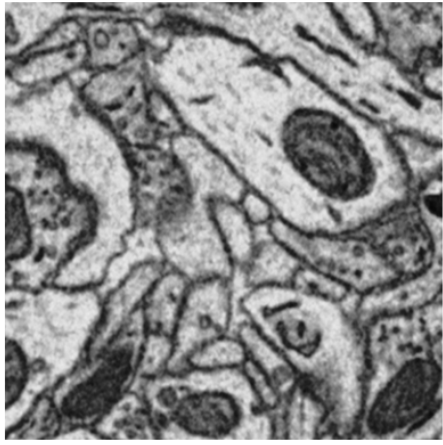
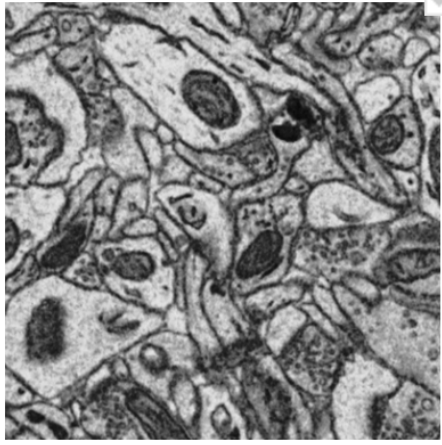
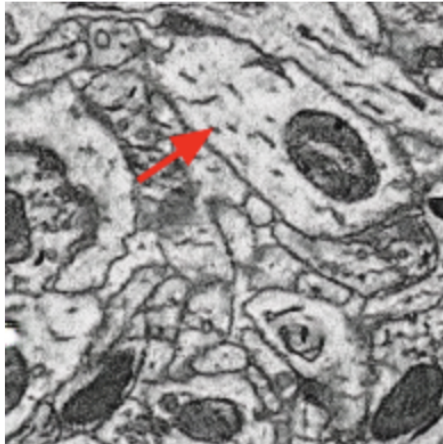
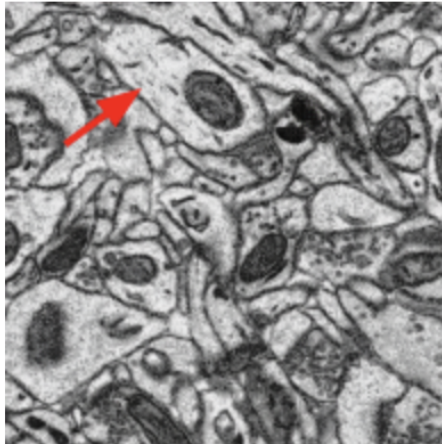
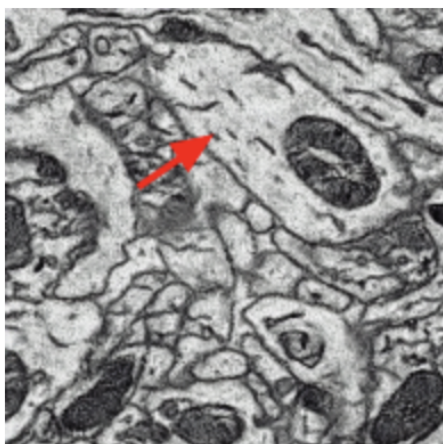
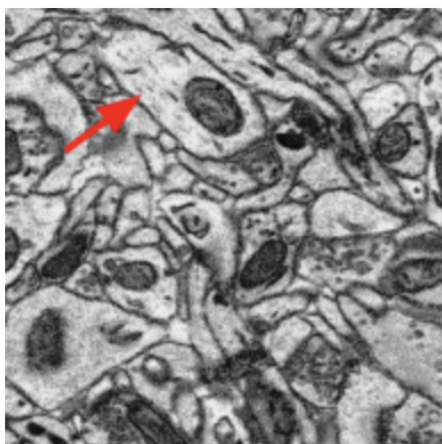
(b) Output of model trained on 200 epochs

Fig. 5: (a), (b) Input (Real) and output (Generated) of CycleGAN models trained on 175 and 200 epochs respectively.

images, Cerebellum images are used as the target domain objects. Against this target domain, other three parts (Cortex-1, Cortex-2 and Hypothalamus) of the images are used individually as a source to get better quality images. The quality of augmented images are evaluated in different approaches as:

- (a) Splitting data in different train/test ratios like 70/30 and 80/20 with original number of images.
- (b) Simulated more images applying random image croppings from the original full sized images and train/test split ratio of 80/20.
- (c) Generating heterogeneous mix of the source images by mixing original source images and degraded images from target domain. Original and degraded images are mixed in different ratios as 85/15, 75/25 and 50/50.

To optimize the model, we are investigating optimal learning rates, optimal number of epochs, and training sizes. Optimizing these hyperparameters will be done over the cortex data set as well as the

	Non-scaled Image	Scaled Image
Original Sample		
100 epochs		
125 epochs		

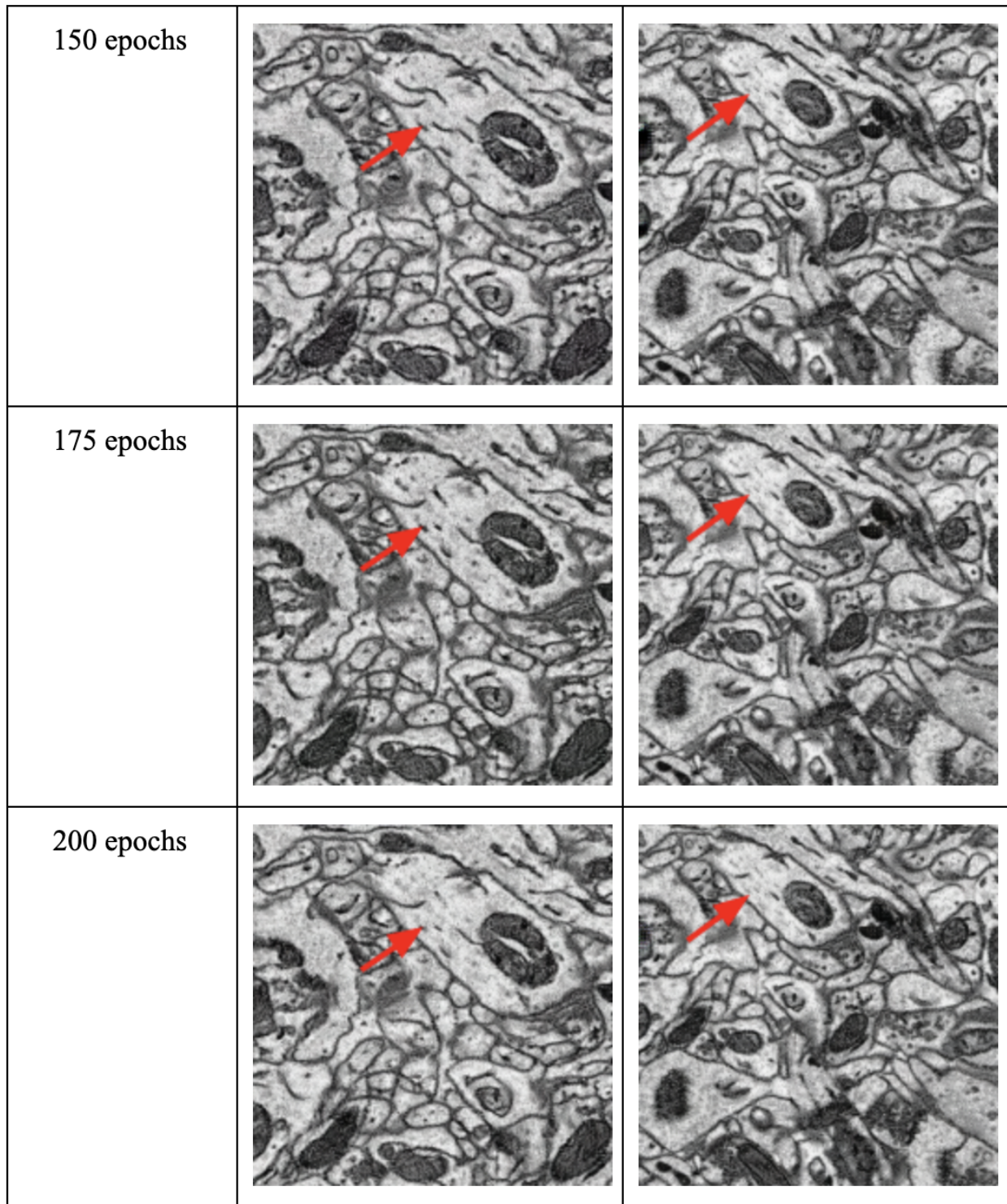


Fig. 6: Training outputs by Epoch. Arrows indicate areas of interest.

hypothalamus data set.

In order to evaluate the effect of learning rate and epoch size on model performance, 7 models with 7 different learning rates have been trained:

- Learning rates from 0.00000002 to 0.02 at the step of 10 times growth have been compared.
- Learning rate 0.02 does not have coverage at all.
- The quality of generated images for epoch 250 is the best for 0.002, 0.0002 and 0.00002.
- For learning rate 0.000002, 0.0000002 and 0.00000002, the quality of generated images for epoch 400 is the best.
- For 0.00000002 with 7200 epochs, the quality of generated images for epoch 7200 is the best, but it does not have coverage on high resolution.
- Learning rate 0.000002 has the best result with 400 epochs among all scenarios. Computational cost/time vs model accuracy should be calculated.

C. Model Performance

Evaluation of resultant images show that 0.00000002 is too small. With this learning rate the training model requires more training epochs due to smaller changes made to the weights on each update. Experiments at this learning rate used epoch values of 800, 1600, 3200, 6400, and 7200. Despite significant improvements at each increase, the output images were insufficiently clear for use in our pipeline. These tests had a run time of over 70 hours. Figure 7 shows the relative gains in clarity across epochs.

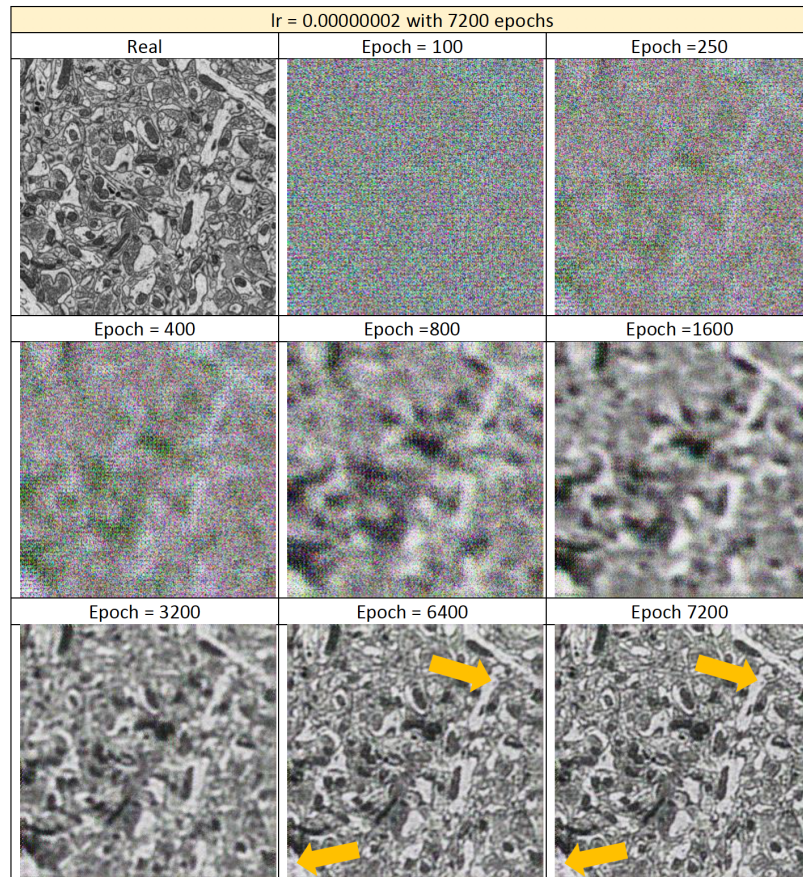


Fig. 7: Improved CycleGAN Performance across Epochs

Another measure of performance can be found in cycle consistent losses. By limiting the losses encountered in each test, we can gain confidence in the model and the reproducibility of our generated images. Comparing early epoch losses against late epoch losses should give us an understanding of the performance of the model. Figure II shows cycle losses for a select number of runs.

TABLE II: Cycle Consistent Losses

Run	Epochs	Iters	D_A	G_A	cycle_A	idt_A	D_B	G_B	cycle_B	idt_B
400	20	200	0.254	0.289	0.500	0.530	0.240	0.278	1.185	0.220
	200	200	0.225	0.298	0.188	0.104	0.222	0.311	0.375	0.059
200_het50	20	81	0.252	0.269	0.383	0.705	0.209	0.378	1.523	0.147
	200	201	0.224	0.293	0.196	0.172	0.222	0.329	0.514	0.065
200_het25	20	81	0.244	0.475	0.472	0.687	0.258	0.240	1.566	0.197
	200	201	0.138	0.410	0.183	0.179	0.028	0.624	0.575	0.063
200_het15	20	81	0.251	0.265	0.252	0.654	0.248	0.274	1.454	0.112
	200	201	0.055	0.649	0.157	0.189	0.273	0.469	0.609	0.077
200	20	200	0.258	0.258	0.281	0.608	0.153	0.296	1.458	0.156
	200	200	0.237	0.306	0.200	0.120	0.025	0.666	0.359	0.066

D. Pre and Post Processing Images

To enhance the CycleGAN training, testing and output capabilities, the images are resized, and tiled. A pre-processing script has been implemented, taking portions of the input image, tiling them and thereby ensuring that boundary conditions do not interfere with CycleGAN image modification. Generally the CycleGAN model experiences issues at boundaries of the images, since it can not “see” past the boundary. To enhance model operation, at the boundary of the image, we will buffer the image with pixels that represent the reflection of the boundary, similar to having a mirror on a wall. This allows the CycleGAN model to “see” past the boundary and output a better modified image. After CycleGAN output, the extra regions will be cropped out, so that the output image is the same size as the input.

These modified tiles are then run through a post-processing script to recombine them into the full, modified image. Through this procedure we can apply the CycleGAN model to larger, non-square images, and images that would exceed our machine or environment’s capabilities. This pre and post processing also allows us to better compare model performance between multiple CycleGAN models subjectively.

Figure 8 shows the original image along with the outputs of a few pre-trained models, with pre and post processing scripts applied. In this figure we are comparing the output of models trained on 400, 600 and 800 images, as well as 200 image training using epoch 800. As seen with the green, yellow and red arrows, boundary lines have been modified from the original input image. To objectively compare the models we can visualize the confidence of the boundaries at those locations, giving us a numerical measure of the model outputs.

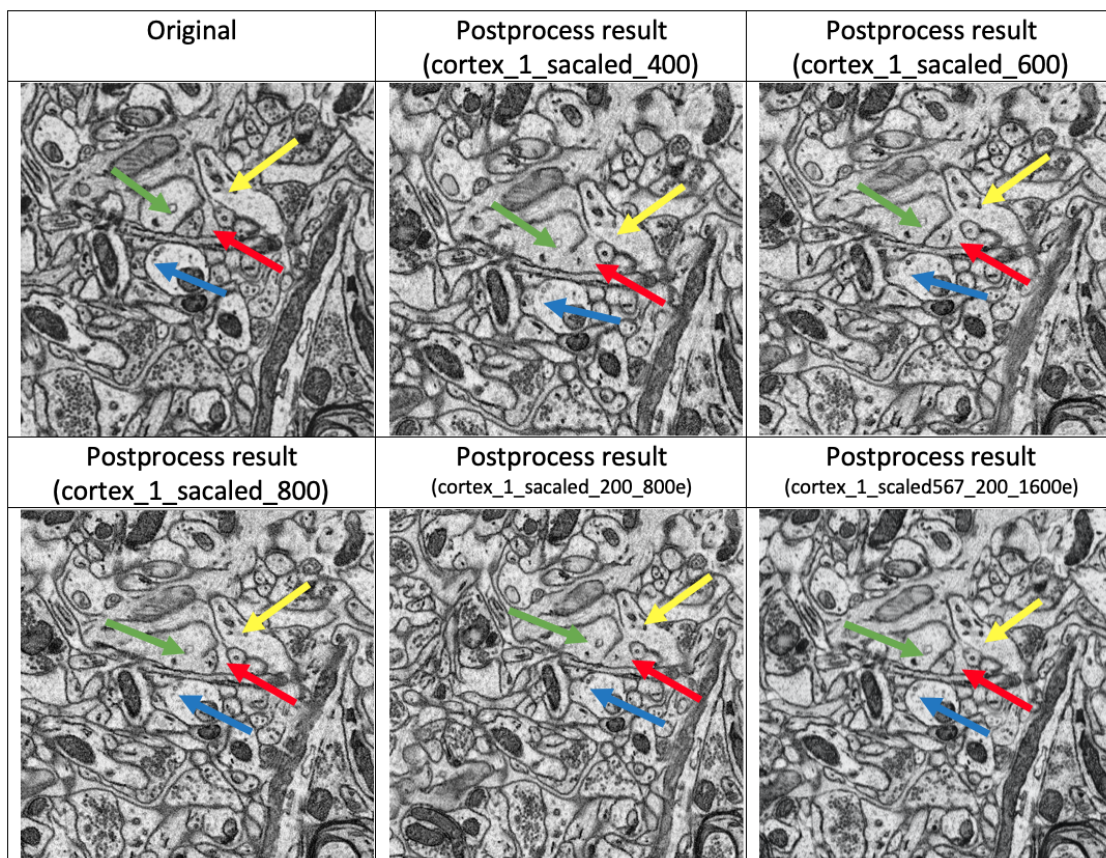


Fig. 8: Model Output Comparisons.
Arrows direct attention to differences
in model output factoring into final selection.

E. Training Size

To evaluate the effect of the size of the training set on the CycleGAN model, initial data was randomly cropped to create more images to use for testing and training. The model was then trained on various training sizes and epochs, with their results compared.

After analysis, some combinations of epoch and training size resulted in an output that included phantom structures that did not exist in the original. For example in figure 9, the lower left corner of training size 400 for epoch 175 includes boundary walls that should not exist. These structures appear in many of the training sizes for this epoch, leading to the determination that we should avoid this epoch.

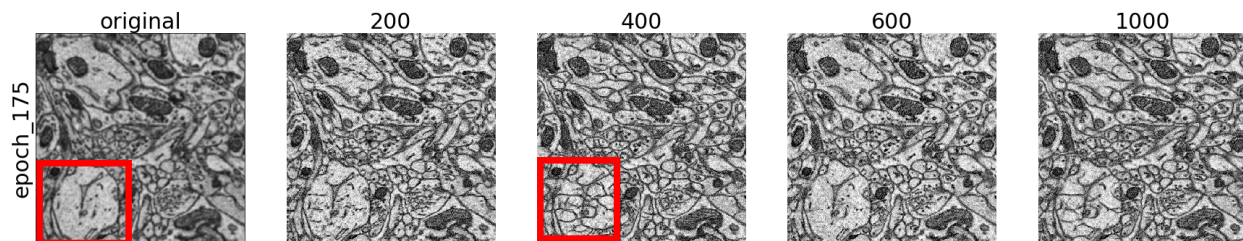


Fig. 9: CycleGAN Trained on 175 Epochs with non-heterogeneous training data.

As an example of an output that improved quality, we can see figure 10. The boxed regions show improved clarity, resulting in a higher confidence for the structures within. This would lead us to believe that epoch 100 should be investigated further.

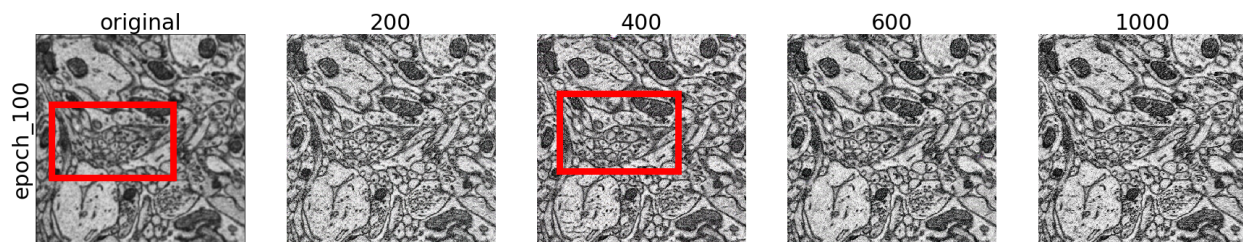


Fig. 10: CycleGAN Trained on 100 Epochs with non-heterogeneous training data.

F. Intermediate Visualization

Visualization for this project is broken into three distinct parts at differing points in the pipeline. The first visualization is the output of the CycleGAN model, allowing us to process lower quality images with the same image segmentation model.

Input into the CycleGAN model are the raw images, scaled and cropped accordingly. The outputs of the CycleGAN model are collected and grouped by voxel according to their original sample. For testing and verification we can intercede at this juncture and visualize the CycleGAN outputs. Typically these images, out of context, mean very little, so it is imperative that the visualization that is presented to the end user is easily comprehensible. To ensure readability, we visualize the inputs and outputs of the CycleGAN model as coupled sets, showing the input and the output images (sometimes referred to as “Real” and “Generated”). Since this visualization is primarily used for debugging and testing/tuning, the visualization is typically not rendered in the foreground but available. An example of this visualization is seen in figure 11.

X. SEGMENTATION TOOL IMPLEMENTATION

All of the data used in this project raw, unlabeled data. This creates an immediate roadblock in terms of objective measures of model performance. To address these challenges, ground truth images must be created to compare against the output of the CDeep3M image segmentation model. Once ground truth images are created, performance metrics can be calculated for measurement of performance improvement.

A secondary effect of the ability to create ground truth images for performance analysis is the ability to create paired images (input and ground truth) to be used for CDeep3M model retraining. These paired images are created in a specific domain and therefore the retrained model will be customized for that domain. The tool must therefore be independent and easily useable by the end-user regardless of technical capability.

The image segmentation tool is created in house based on the Pygame engine. The tool has been extensively tested and evaluated. Multiple ground truths were created by multiple persons on the segmentation tool. Edge use cases that exceeded the ability of the tool were found, the root of the issue was identified and the tool was updated/enhanced to compensate.

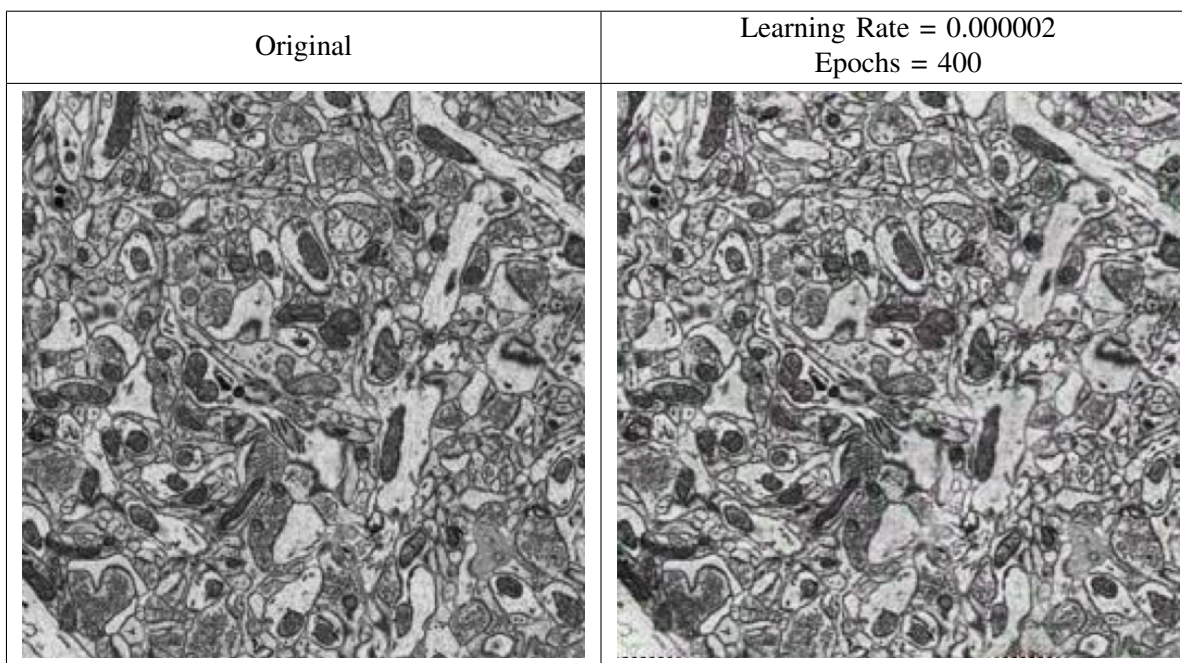


Fig. 11: Input and Output Images for CycleGAN Model

The tool can be loaded on a personal machine. The tool has a graphical user interface (GUI) that allows the user to select and input one or multiple images from the local file tree. If a region of interest (ROI) is desired, the x,y coordinates along with the size of the crop to be worked on can be input and previewed prior to selection.

Once the user has input their images, coordinates and sized, layers can be added to independently mark structures such as membranes, mitochondria, etc. Each layer acts independently, allowing the user to mark structures, erase, save current state, and export final product. The export feature exports a file tree that includes the roi cropped from the input image, along with each layer worked on by the end user. The file has its metadata with more information on the image, cropped size and x,y location of the crop was embedded in the output's metadata. This ensures that the end user can easily locate desired files, and immediately have paired images for training or validation.

This tool was used directly by most of the team to create ground truth images for use in both evaluating the various CycleGAN models, and retraining the Segmentation model, both of which are explained in Fig. 12.

XI. CDEEP3M MODEL RETRAINING

To aid in retaining the CDeep3M model, we created a 10 image stack of membrane predictions, used as optimal desired outputs. This was achieved by manually labelling membrane and mitochondria from 800x800 crops of various images within the Cortex_1 dataset. This includes 10 consecutive images, as well as some random images/crops from the dataset. This stack of 10 images, paired with the CycleGAN enhanced input images, can be used to retrain the model to fit better with our sample data. Fig. 13 shows an overview of the retraining process.

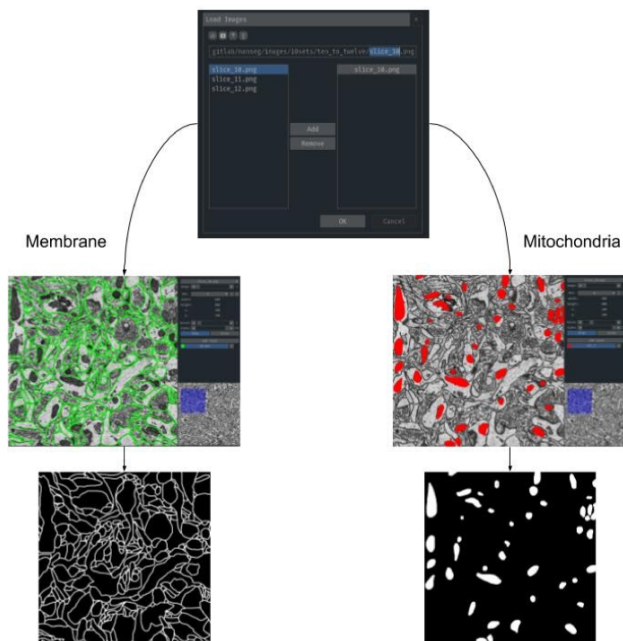


Fig. 12: Image Segmentation Pipeline.

User can select multiple layers to mark cell structures.
Output of pipeline are paired with initial input image.

A last step that can be performed is dilation of the ground truth images. In many cases the ground truth images represent a thin membrane, which does not always match with the actual image or the assumed membrane thickness. Rather than recreating the ground truth images, we instead use the Scikit Morphology module, specifically binary dilation to grow the cell membrane thickness on the pre-existing ground truth images. This can ensure that during testing and retraining, the chosen model is expected to create thicker boundaries, further optimizing the resulting visualizations.

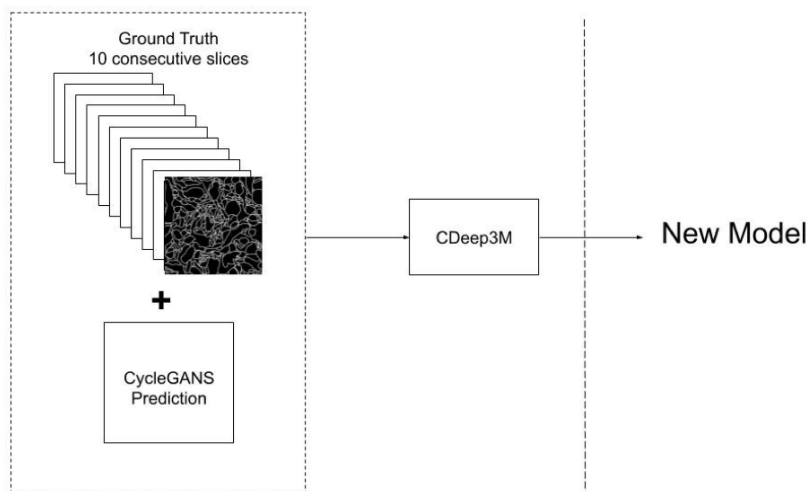


Fig. 13: Retraining of CDeep3M model. 10 image stack of ground truth with associated CycleGAN outputs are used to retrain the CDeep3M model.

XII. OUTPUT VISUALIZATIONS

Our final visualization is built on the outputs of the 3D rendering as in Fig. 15. While we have some visualizations that can be investigated during processing and analysis, our main focus is the end product rendering. For use in modifying, retraining and explaining the model performance, we have shown a number of overlays and side by side comparisons. These comparisons of predictions, or input image quality are most useful for optimizing the model for the specific dataset. As we have been informed by subject matter experts, the main visualizations that they are interested in are firstly side by sides of the original model as in Fig. 14, predictions overlaying the original model and predictions against a blank background. This allows the end user to ensure that coverage of the model prediction is complete.

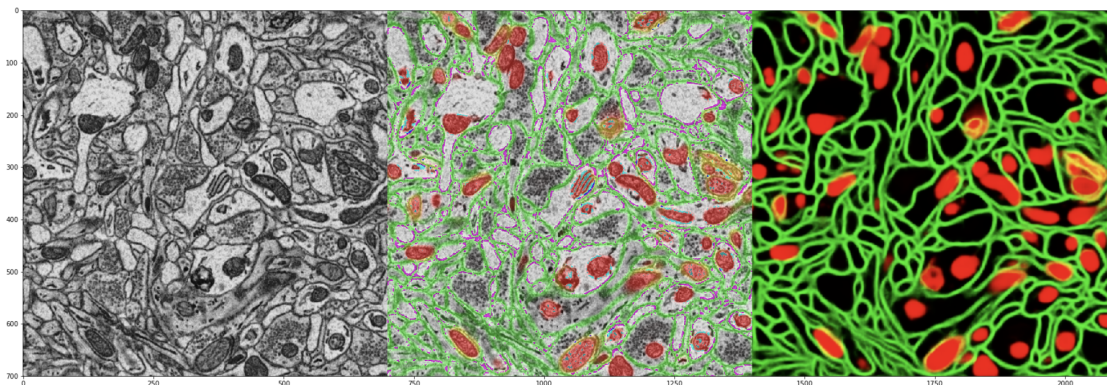


Fig. 14: Original Image vs Predictions with an Overlay

Secondly the 3D rendering is most requested. This is primarily focused on presentations to their colleagues and interested 3rd parties. These renders are more easily digestible by non-subject matter experts as seen in Fig. 15.

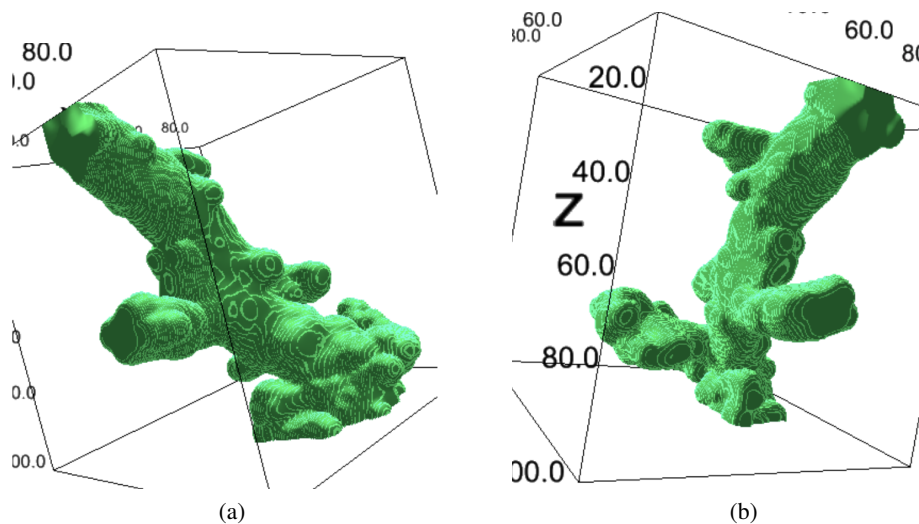


Fig. 15: 3D reconstruction of a neurite projecting from a neuronal axon or dendrite.

XIII. COMPLETE DATA PIPELINE

Currently we are working on installing and understanding the previously written and executed python-based files. These have primarily been written in Jupyter Notebooks, with JavaScript based add ins for 3-Dimensional rendering and add-ins. As we gain understanding and expertise with these analysis tools, we can start optimizing them for cascading processes. Stated goals of the project include a total time of roughly 1 week from beginning of automated processing to output rendering and analysis of the samples.

The pipeline shown in Figure 16 is a basic overview of what we are working to achieve. We seek to integrate multiple datasets, each representing many slices of a multiple sample. After resolution boosting and integration, we will use human in the loop verification on border assignment. After borders are defined, predictive algorithms will identify structures, assign probabilities of sub-cell structures, and visualizations in 2 and 3D will be created.

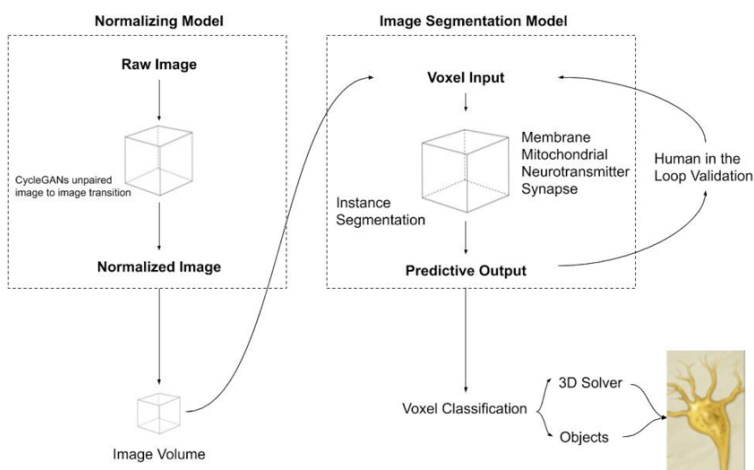


Fig. 16: Data Pipeline Overview

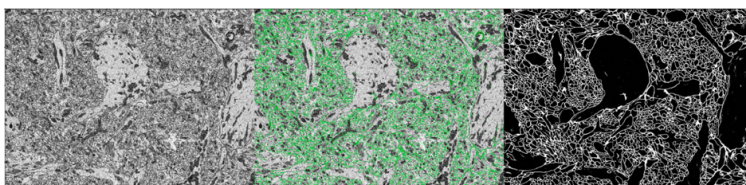


Fig. 17: CDeep3M Model Output

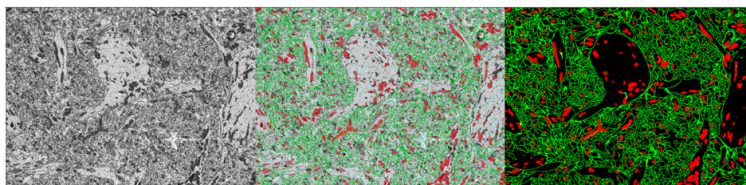


Fig. 18: Input image, output predictions and overlay of both shown. Mitochondria in the Sample shown in red. Membrane prediction shown in green.

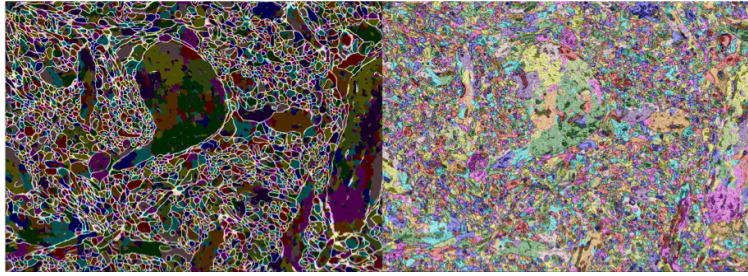


Fig. 19: Instance Segmentation with watershed filling in cellular features in the 2D slice.

Figures 17 to 19 show some of the progression of raw input data into bounded, coded output data. Figure 17 shows the cleaning and absolute bounding of cell structures using ML algorithms, based on datasets that were verified by humans in the loop. Figure 18 shows an output of prediction values for mitochondria within the cell, highlighted as an overlay of the original detail, with increased contrast for easy comprehension. Figure 19 shows the result from instance segmentation. Instance segmentation diverges from object detection by labelling an image in its entirety. The object detection performed so far outputs probabilities that a pixel either does or doesn't belong to the object of interest. In instance segmentation, every pixel is assigned a label id that corresponds to its respective region in the image.

XIV. PERFORMANCE

A. Ground Truths for Model Evaluation

800x800 pixel crops were taken from sampled images and membranes and mitochondria were labeled. These samples were used to evaluate the accuracy of the segmentation model, both before the implementation of CycleGAN image enhancement, and for the various CycleGAN models that have been developed, built and implemented. Results from this side to side evaluation can be found in Section XVI.

B. Understanding of Performance

As we define performance, we are not concerned with processing speed, but rather with predictive output. Now that we have constructed ground truth data for both membranes and mitochondria, we focus on the quantitative measures, combined with subjective analysis of the predictive output, to measure performance.

C. Performance Metric Selection

For quantitative measurements, an obvious choice is F-1 score. For this application of model prediction, recall is considered to be more important than precision. The justification being that a false positive resulting in more numerous membrane predictions is more desirable than a false negative, which would ignore the membrane segment and cause more merging of cellular structures. F-Beta will therefore be used to measure model performance.

XV. MODEL RESULTS EVALUATION METRICS

During testing and evaluation more than 30 independent models were trained and compared. The results of many of the hyperparameter testing can be found in Appendix III.

The selection of the threshold is the most subjective portion of the evaluation process. The outputs from the image segmentation model rank every pixel in terms of probability that the pixel is of a known

cellular structure. For example the output of the membrane segmentation model will predict 0 for all those pixels where the probability of that pixel being part of a membrane is 0%. Since the images are imported and evaluated in 8-bit data, a pixel with 100% confidence of being a membrane would be labeled with a rank of 255. Therefore the confidence of each pixel will fall onto the continuum from 0-255. By selecting a threshold value, all pixels with a rank below the threshold will be ignored and rewritten as 0 with all ranks above the threshold being rewritten as 255. As seen in Appendix III, attempting to balance the quality of the output prediction against the performance of the model is model specific. Generally a threshold value of 125, which roughly relates to a confidence of 49%, balances the resulting predictions creating a fully formed image, and the scoring metrics.

By selecting model I200_S567_LR000002_D_E1600_BL_O9G1_R800 performance increases of 13.59% (F-Beta) and 20.23% (F-1) were recorded against membrane ground truth while performance increases of 22.82% (F-Beta) and 22.22% (F-1) were recorded against mitochondria ground truth. This results in a better visualization, and more detail in visualizing areas between and around membranes.

A. Membrane Predictions

As previously noted for membrane predictions created ground truth images are used to compare the F-Beta score and to select the top performing model. The results of the objective measurement are seen in Fig. 20.

Model	Threshold	Images	Scale	Epoch	Denoise	Learning Rate	Retrained	Blended	Original(%)	Generated(%)	Precision	Recall	Specificity	Accuracy	F1	F_Beta
I200_S567_LR000002_D_E1600_BL_O1G9_R800: Re-trained Model (+800 iterations) with generated & groundtruth data, prediction on blended(10% Original 90% Generated) images, and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0.6823	0.8716	0.8355	0.8459	0.7654	0.8258
I200_S567_LR000002_D_E1600_BL_O9G1_R800: Re-trained Model (+800 iterations) with generated & groundtruth data, prediction on blended(90% Original 10% Generated) images, and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.6907	0.867	0.8426	0.8497	0.7688	0.8249
I200_S567_LR000002_D_E1600_BL_O6G4_R800: Re-trained Model (+800 iterations) with generated & groundtruth data, prediction on blended(60% Original 40% Generated) images, and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	Yes	Yes	60	40	0.7074	0.8394	0.8593	0.8536	0.7678	0.8092
I200_S567_LR000002_D_E1600_BL_O6G4: Base Model, prediction on blended(60% Original 40% Generated) images and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.7229	0.7363	0.8856	0.8426	0.7295	0.7336
Base Model, prediction on original images	125		Single	200	No	0.0002	No	No	100	0	0.5332	0.7985	0.7167	0.7403	0.6394	0.7262

Fig. 20: Top results from membrane model testing, sorted descending by F-Beta score. Dark colors indicate better score.

These top results can also be shown in Fig. 21, where each model's F-Beta score for each model is compared against the threshold values. As shown the F-Beta score falls off dramatically for all models at a threshold of 200, and at the selected threshold of 125 there is clustering of our models' performances, all well above the score for the base model.

To select the optimal threshold value we must view the predictions for each model at each threshold. Working just on the F-Beta score would lead to select a threshold value of 200. When viewing the predictions, at a threshold of 200 the predictions are disjointed and will not create good volume renders. To combine the objective and subjective measures, refer to Fig 22. In these overlays, the green represents ground truth data, and the red is the model predictions. The overlapping predictions turn yellow, so we can easily see models that predict accurately or inaccurately.

Though it is a subjective measure, the model predictions differ highly despite similar F-Beta scores. Using the results shown in Fig 22, a subjective analysis of the model performance can isolate the best performing model to incorporate into the pipeline. The end result of combining objective performance in Figs. 20 and 21 and subjective measurements from Fig. 22 leads us to choose the

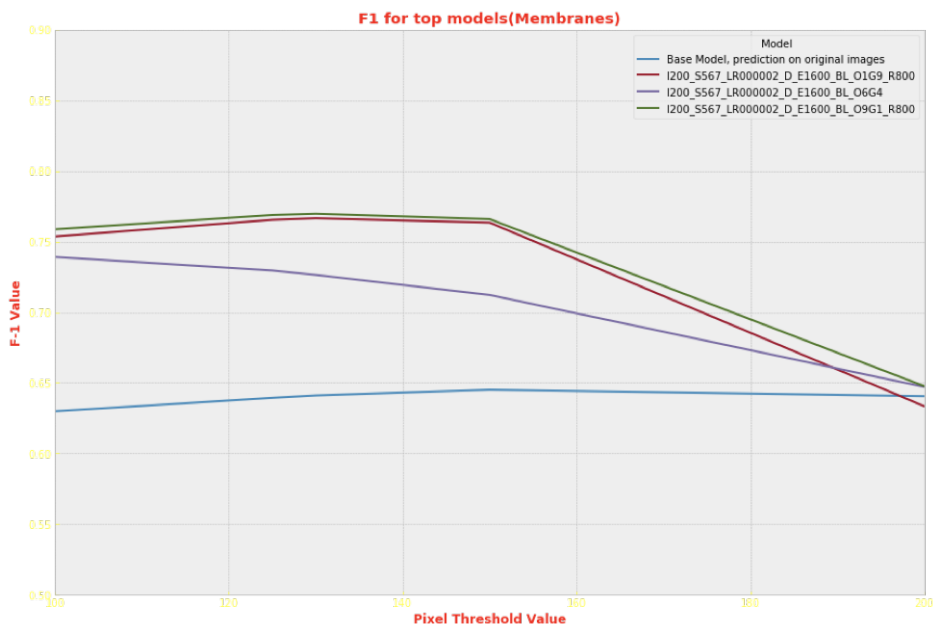
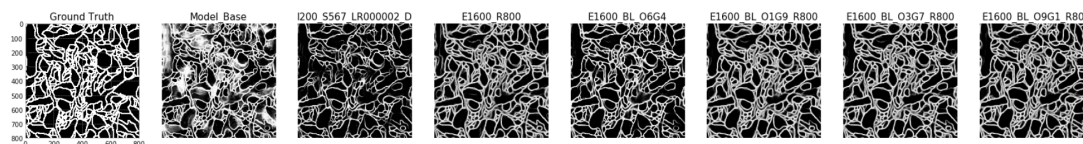


Fig. 21: Line graph of Top results from Membrane Testing. Y-axis is F-Beta value. X-axis is pixel threshold value. Final selection of threshold is 125.

Ground Image vs Predicted Images from different model:



Images with Threshold value 125, Overlay Presentation:- GREEN:Original, RED:Predicted:

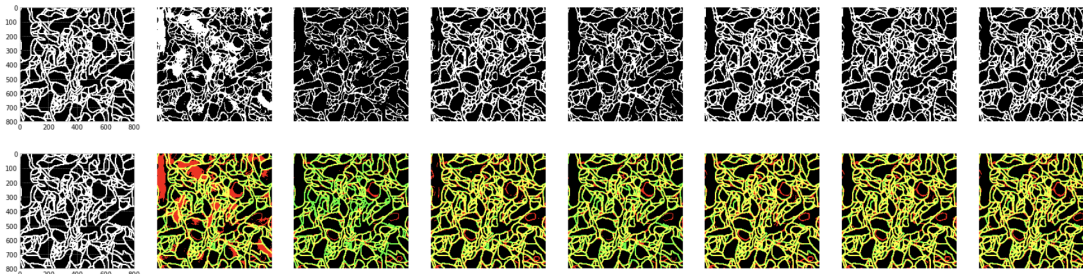


Fig. 22: Model Membrane Predictions

Green color indicates ground truth. Red color indicates model predictions.
Yellow indicates overlap of ground truth and model prediction.

I200_S567_LR000002_D_E1600_BL_O9G1_R800 model. This is the 1600th epoch of a model trained on 200 images, a learning rate of 0.000002, retrained on 10 ground truth images, run with the CycleGAN option of denoise turned off, using a multi-factor scaling and a blending of 90% original image and 10% CycleGAN enhanced image.

1) *Mitochondria Predictions:* In addition to the membrane segmentation analysis and selection, ground truth images were generated for use in retraining the model to predict mitochondria within the image volumes. Performance measurements such as F-1 and F-Beta can be seen in 23 and 24.

Model	Threshold	Images	Scale	Epoch	Denoise	Learning Rate	Retrained	Blended	Original(%)	Generated(%)	Precision	Recall	Specificity	Accuracy	F1	F_Beta
I200_S567_LR000002_D_E1600_BL_O9G1_R800: Re-trained Model (+800 Iterations) with generated & groundtruth data, prediction on blended(90% Original 10% Generated) images, and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.6413	0.929	0.9487	0.9469	0.7588	0.8525
I200_S567_LR000002_D_E1600_BL_O1G9: Base Model, prediction on blended(10% Original 90% Generated) images and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.5935	0.8117	0.9451	0.9331	0.6857	0.7561
I200_S567_LR000002_D_E1600_BL_O3G7: Base Model, prediction on blended(30% Original 70% Generated) images and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.5404	0.8211	0.931	0.9211	0.6518	0.7438
Base Model, prediction on original images	125		Single	200	No	0.0002	No	No	100	0	0.5279	0.7534	0.9334	0.9172	0.6208	0.6941
I200_S567_LR000002_D_E1600_BL_O9G1: Base Model, prediction on blended(90% Original 10% Generated) images and tuned hyperparameters	125	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.3297	0.8618	0.8269	0.83	0.4769	0.6515

Fig. 23: Top results from mitochondria model testing, sorted descending by F-Beta score. Light colors indicate better score.

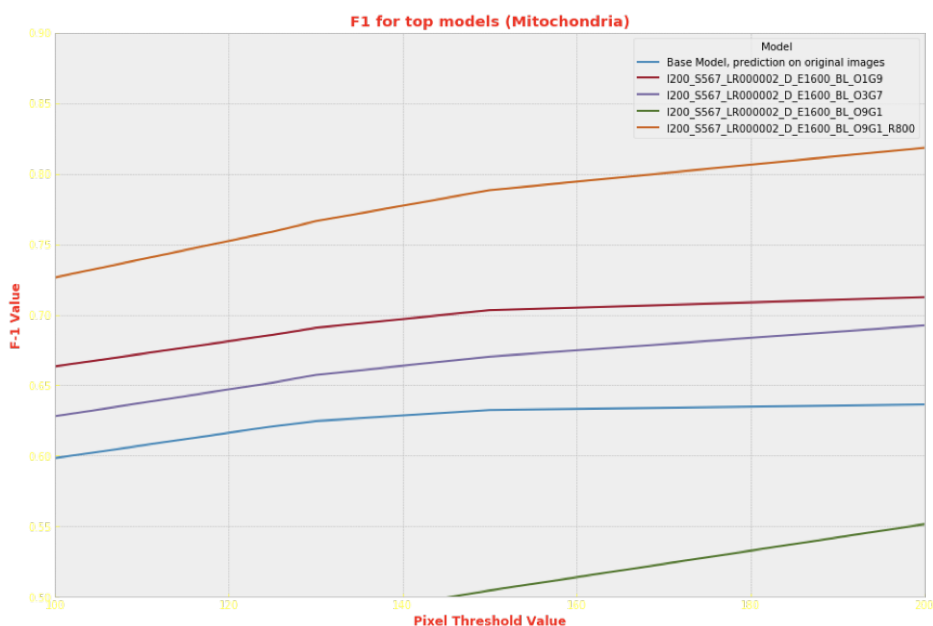


Fig. 24: Line graph of Top results from Mitochondria Testing. Y-axis is F-Beta value. X-axis is pixel threshold value. Final selection of threshold is 125.

In addition to the objective measurement of F-1 and F-Beta, subjective comparisons were generated as in Fig. 25, where the yellow is where the model predictions match with the ground truth data, and the red is where the model predictions differ from the ground truth.

Using the results shown in figures 23 and 25, an analysis of the model performance can result in the best performing model. Due to the inherent folds within the mitochondria substructure this ground truth can be subjective as a result of inequitable stain absorption. Unlike membrane labelling, it is unlikely that observed models label only parts of mitochondria. Either the entire mitochondria would be marked or not, so there is less impetus to find a model that will "fill in" missing segments.

Lastly it is advisable to not over-weigh the impact of the size of certain mitochondria relative to others. Since mitochondria will be more continuously marked to begin with, one large cross section would disproportionately affect the measure of the performance of the model. Taking these factors into consideration, subjective measurements were weighted higher than objective measurements of performance in selecting the final model. As with the membrane prediction analysis, the

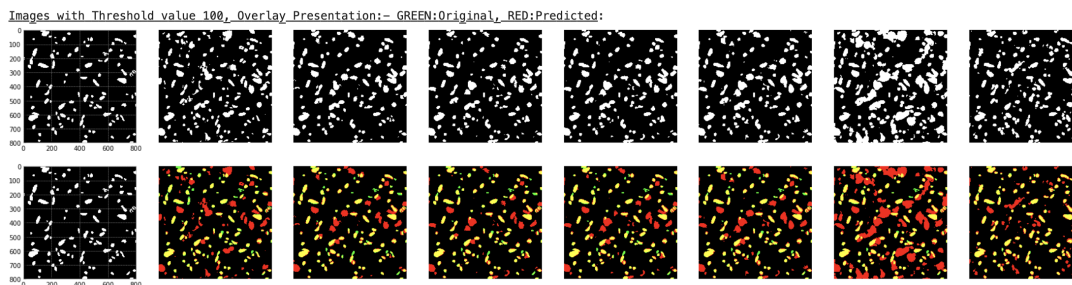


Fig. 25: Model Mitochondria Predictions

Green color indicates ground truth. Red color indicates model predictions.
Yellow indicates overlap of ground truth and model prediction.

I200_S567_LR000002_D_E1600_BL_O9G1_R800 model performed best, confirming the selection for use in the pipeline.

XVI. FUTURE WORK

The NCMIR group has begun to use optimization, evaluation, and performance data from these experiments to process human image datasets for persons with ALZ. During initial collection, aggregation and pre-processing of the human images, information gathered by this project impacted the results seen in Fig. 26.

While only initial results, this pipeline shows promise. Models can be enhanced with further retraining within the specific human dataset domain.

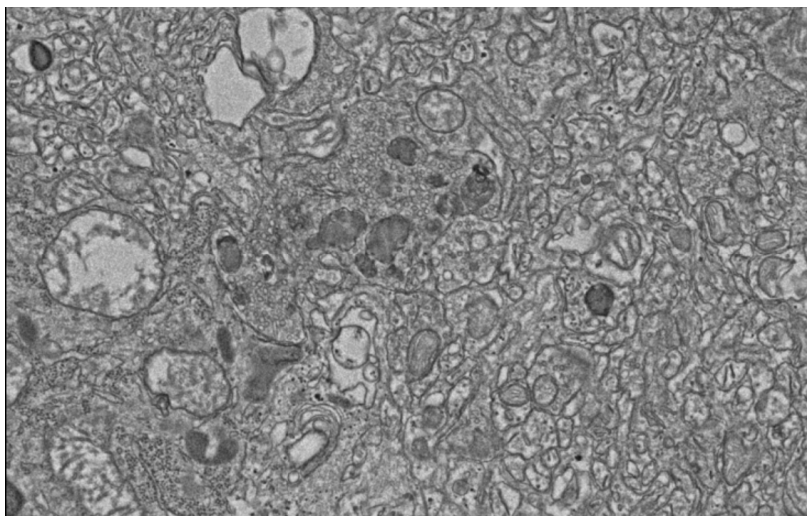
XVII. CONCLUSION

In this project datasets made of images from samples of mouse brain were investigated and analyzed. Differences in image quality were noted, and areas for possible advancement were identified. Project pipeline, data processing and visualizations were performed in python and output visualizations with quantitative measurements were presented.

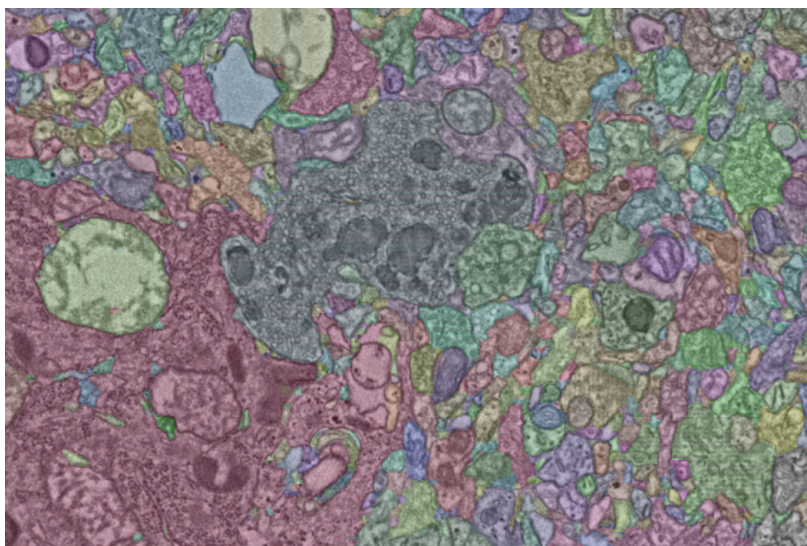
By training and implementing a CycleGAN neural network, image qualities were increased. Training was aided by image tiling and output blending. The resulting improved images were fed to a CDeep3M segmentation model. This segmentation model was additionally enhanced through retraining with paired images within the dataset domain. All ground truths used for retraining and performance measurements were created using a custom made, standalone image segmentation tool. This tool works outside of the main pipeline, allowing for user interaction with the models encapsulated within the pipeline. Output data is visualized with a combination of 2 dimensional and 3 dimensional renderings. Overall image segmentation can be viewed as results of differing steps of the pipeline, and the end-user can select specific cellular structures on the fly to render in an interactive 3D environment.

Final results show an increase in membrane prediction performance of 13.59% (F-Beta) and 20.23% (F-1) and an increase in mitochondria prediction performance of 22.82% (F-Beta) and 22.22% (F-1) as compared against the output of the CDeep3M model. This increase directly leads to improved segmentation results and improved visualizations for the end user.

This pipeline is built and implementable on the Nautilus cloud computing resource, allowing for scalability. Users are not confined by either the scale of the input data or the computational resources of



(a) Original Human ALZ Image



(b) Human ALZ Image after Segmentation

Fig. 26: Pipeline input and output on human ALZ dataset performed by NCMIR research group.

local machines. The pipeline is inherently scalable to a level that is only contained by the availability of cloud computing resources. This pipeline has additionally been integrated into initial human data by collaborators in the NCMIR. Initial outputs from our pipeline on human data has been presented.

REFERENCES

- [1] A. Association, “2019 alzheimer’s disease facts and figures,” *Alzheimer’s & Dementia*, vol. 15, no. 3, pp. 321–387, 2019.
- [2] M. G. Haberl, C. Churas, L. Tindall, D. Boassa, S. Phan, E. A. Bushong, M. Madany, R. Akay, T. J. Deerinck, S. T. Peltier *et al.*, “Cdeep3m—plug-and-play cloud-based deep learning for image segmentation,” *Nature methods*, vol. 15, no. 9, pp. 677–680, 2018.
- [3] T. Liebmann, N. Renier, K. Bettayeb, P. Greengard, M. Tessier-Lavigne, and M. Flajolet, “Three-dimensional study of alzheimer’s disease hallmarks using the idisco clearing method,” *Cell reports*, vol. 16, no. 4, pp. 1138–1152, 2016.
- [4] P. Welander, S. Karlsson, and A. Eklund, “Generative adversarial networks for image-to-image translation on multi-contrast mr images—a comparison of cyclegan and unit,” *arXiv preprint arXiv:1806.07777*, 2018.
- [5] A. S. Becker, L. Jendele, O. Skopek, N. Berger, S. Ghaffoor, M. Marcon, and E. Konukoglu, “Injecting and removing suspicious features in breast imaging with cyclegan: A pilot study of automated adversarial attacks using neural networks on small images,” *European journal of radiology*, vol. 120, p. 108649, 2019.
- [6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.
- [7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [8] —, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [10] J. Zhao, M. Mathieu, and Y. LeCun, “Energy-based generative adversarial network,” *arXiv preprint arXiv:1609.03126*, 2016.
- [11] H. Liang, L. Yang, H. Cheng, W. Tu, and M. Xu, “Human-in-the-loop reinforcement learning,” in *2017 Chinese Automation Congress (CAC)*, Oct 2017, pp. 4511–4518.
- [12] D. Purves, *Neuroscience*. Oxford University Press, 2012.
- [13] H. H. Denk W, *Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure*.
- [14] E. B. Bloss, M. S. Cembrowski, B. Karsh, J. Colonell, R. D. Fetter, and N. Spruston, “Single excitatory axons form clustered synapses onto ca1 pyramidal cell dendrites,” *Nature neuroscience*, vol. 21, no. 3, pp. 353–363, 2018.
- [15] C. Bosch, A. Martínez, N. Masachs, C. M. Teixeira, I. Fernaud, F. Ulloa, E. Pérez-Martínez, C. Lois, J. X. Comella, J. DeFelipe *et al.*, “Fib/sem technology and high-throughput 3d reconstruction of dendritic spines and synapses in gfp-labeled adult-generated neurons,” *Frontiers in neuroanatomy*, vol. 9, p. 60, 2015.
- [16] B.-C. Chen, W. R. Legant, K. Wang, L. Shao, D. E. Milkie, M. W. Davidson, C. Janetopoulos, X. S. Wu, J. A. Hammer, Z. Liu *et al.*, “Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution,” *Science*, vol. 346, no. 6208, p. 1257998, 2014.
- [17] D. D. Bock, W.-C. A. Lee, A. M. Kerlin, M. L. Andermann, G. Hood, A. W. Wetzel, S. Yurgenson, E. R. Soucy, H. S. Kim, and R. C. Reid, “Network anatomy and in vivo physiology of visual cortical neurons,” *Nature*, vol. 471, no. 7337, pp. 177–182, 2011.
- [18] M. Claesen and B. De Moor, “Hyperparameter search in machine learning,” *arXiv preprint arXiv:1502.02127*, 2015.
- [19] X. Ren and J. Malik, “Learning a classification model for segmentation,” in *null*. IEEE, 2003, p. 10.
- [20] G. Mori, X. Ren, A. A. Efros, and J. Malik, “Recovering human body configurations: Combining segmentation and recognition,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. IEEE, 2004, pp. II–II.
- [21] D. Kang, O. Rübél, S. Byna, and S. Blanas, “Comparison of array management library performance—a neuroscience use case,” 2019.
- [22] C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk, “Solving large multicut problems for connectomics via domain decomposition,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1–10.
- [23] L. Derczynski, “Complementarity, f-score, and NLP evaluation,” in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 261–266. [Online]. Available: <https://www.aclweb.org/anthology/L16-1040>

APPENDIX A
LINK TO THE LIBRARY ARCHIVE FOR REPRODUCIBILITY

Bond, Matthew L.; Kazemisefat, Doreh; Mondal, Ashok; Silva, Daniel; Sha, Peishan; Ellisman, Mark; Madany, Matthew; Peltier, Steve (2020). Alzheimer's Disease Data Analysis. In Data Science & Engineering Master of Advanced Study (DSE MAS) Capstone Projects. UC San Diego Library Digital Collections. doi.org/10.6075/J0B856N6

APPENDIX B
MAS DSE KNOWLEDGE APPLIED TO THE PROJECT

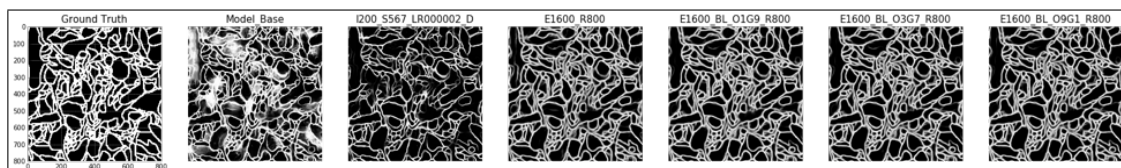
This project pulls knowledge and ability built across the MAS DSE program. Included are:

Visualizations rendered here and overall presentation of results is directly influenced by DSE 241 (Data Visualization). This includes research into isosurfaces, proper etiquette for displaying results from numerous model evaluations, and most effective use of visualizations where small changes are highlighted and isolated.

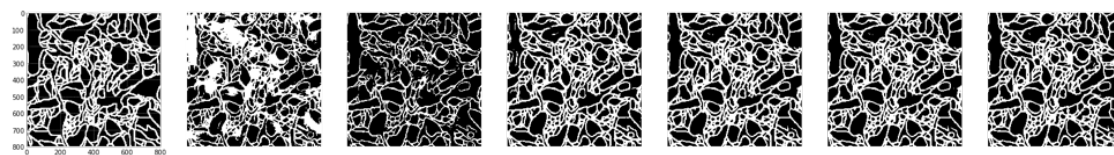
The largest contributing coursework to this project stemmed from DSE 220 (Machine Learning) and DSE 210 (Probability and Statistics Using Python). In this project the principles used for optimizing hyperparameters, model training and model evaluation techniques are directly and indirectly derived from DSE 220. Additionally the model evaluation, specifically objective measurements of performance were introduced in DSE 210.

The project foundation in python guarantees the derivation of techniques, environments and package utilization presented in DSE 200 (Python for Data Analysis). Optimization of pipelines and techniques for enhanced performance were introduced in DSE 203 (Data Integration & ETL).

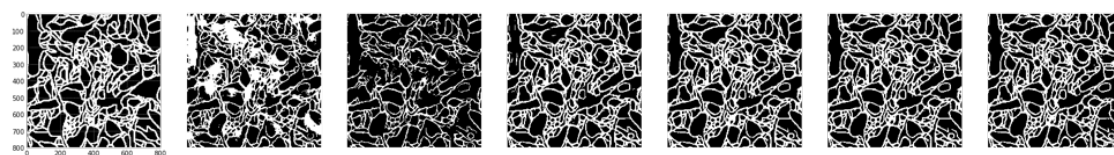
APPENDIX C
MODEL PREDICTIVE OUTPUTS



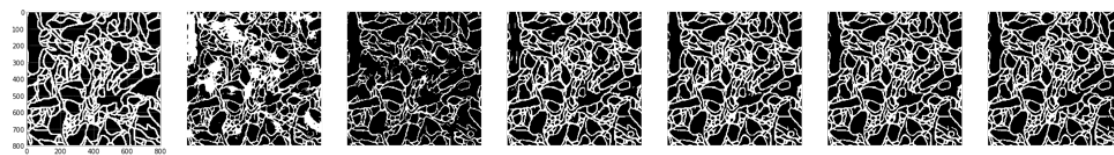
(a) Ground Image vs Predicted Image



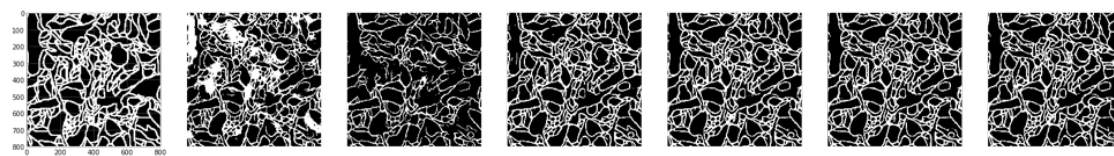
(b) Image with Threshold Value 100



(c) Image with Threshold Value 125



(d) Image with Threshold Value 130



(e) Image with Threshold Value 150

APPENDIX D
MEMBRANE PERFORMANCE MEASUREMENTS FOR ALL MODELS

Model	Threshold	Images	Scale	Epoch	Denoise	Learning Rate	Retrained	Blended	Original (%)	Generated (%)	Precision	Recall	Specificity	F1	F-Beta
I200_S567_LR000002_D_R800	100	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.65	0.9062	0.8023	0.757	0.84
I200_S567_LR000002_D_R800	125	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.6912	0.8573	0.8448	0.7653	0.818
I200_S567_LR000002_D_R800	130	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.7016	0.8427	0.8548	0.7657	0.8101
I200_S567_LR000002_D_R800	150	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.7373	0.7826	0.887	0.7593	0.7731
I200_S567_LR000002_D_R800	200	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.8388	0.5015	0.9609	0.6277	0.5453
I200_S567_LR000002_D_R800	250	200	Multi	200	Yes	0.000002	Yes	No	100	0	1	0	1	0	0
I200_S567_LR000002_D_R400	100	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.656	0.8931	0.8102	0.7564	0.8329
I200_S567_LR000002_D_R400	125	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.6952	0.8443	0.85	0.7625	0.8096
I200_S567_LR000002_D_R400	130	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.7047	0.8301	0.859	0.7623	0.8016
I200_S567_LR000002_D_R400	150	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.7379	0.7725	0.8888	0.7548	0.7653
I200_S567_LR000002_D_R400	200	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.8329	0.518	0.9579	0.6387	0.5603
I200_S567_LR000002_D_R400	250	200	Multi	200	Yes	0.000002	Yes	No	100	0	0.968	0.0018	1	0.0036	0.0023
I200_S567_LR000002_D_E1600_R800	100	200	Multi	1600	Yes	0.000002	Yes	No	100	0	0.642	0.911	0.7942	0.7532	0.8405
I200_S567_LR000002_D_E1600_R800	125	200	Multi	1600	Yes	0.000002	Yes	No	100	0	0.6851	0.8601	0.8398	0.7627	0.8183
I200_S567_LR000002_D_E1600_R800	130	200	Multi	1600	Yes	0.000002	Yes	No	100	0	0.6941	0.8477	0.8486	0.7633	0.8118
I200_S567_LR000002_D_E1600_R800	150	200	Multi	1600	Yes	0.000002	Yes	No	100	0	0.7318	0.7867	0.8832	0.7582	0.7751
I200_S567_LR000002_D_E1600_R800	200	200	Multi	1600	Yes	0.000002	Yes	No	100	0	0.838	0.4942	0.9613	0.6217	0.5383
I200_S567_LR000002_D_E1600_R800	250	200	Multi	1600	Yes	0.000002	Yes	No	100	0	0	0	1	0	0
I200_S567_LR000002_D_E1600_BL_O9G1_R800	100	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.6492	0.9127	0.8002	0.7587	0.8442
I200_S567_LR000002_D_E1600_BL_O9G1_R800	125	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.6907	0.867	0.8426	0.7688	0.8249
I200_S567_LR000002_D_E1600_BL_O9G1_R800	130	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.701	0.8534	0.8525	0.7697	0.8178
I200_S567_LR000002_D_E1600_BL_O9G1_R800	150	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.737	0.7971	0.8848	0.7659	0.7843
I200_S567_LR000002_D_E1600_BL_O9G1_R800	200	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	0.8404	0.5268	0.9595	0.6476	0.5693
I200_S567_LR000002_D_E1600_BL_O9G1_R800	250	200	Multi	1600	Yes	0.000002	Yes	Yes	90	10	1	0.0001	1	0.0002	0.0001
I200_S567_LR000002_D_E1600_BL_O9G1	100	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.6905	0.8022	0.8543	0.7422	0.7771
I200_S567_LR000002_D_E1600_BL_O9G1	125	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.7121	0.7606	0.8754	0.7356	0.7504
I200_S567_LR000002_D_E1600_BL_O9G1	130	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.7169	0.7496	0.88	0.7329	0.7428
I200_S567_LR000002_D_E1600_BL_O9G1	150	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.7332	0.7115	0.8951	0.7222	0.7157
I200_S567_LR000002_D_E1600_BL_O9G1	200	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.7786	0.5809	0.9331	0.6654	0.612
I200_S567_LR000002_D_E1600_BL_O9G1	250	200	Multi	1600	Yes	0.000002	No	Yes	90	10	0.9028	0.1421	0.9938	0.2456	0.1709
I200_S567_LR000002_D_E1600_BL_O8G2_R800	100	200	Multi	1600	Yes	0.000002	Yes	Yes	80	20	0.6494	0.912	0.8005	0.7586	0.8438
I200_S567_LR000002_D_E1600_BL_O8G2_R800	125	200	Multi	1600	Yes	0.000002	Yes	Yes	80	20	0.6904	0.8667	0.8425	0.7686	0.8246
I200_S567_LR000002_D_E1600_BL_O8G2_R800	130	200	Multi	1600	Yes	0.000002	Yes	Yes	80	20	0.7009	0.853	0.8525	0.7695	0.8175
I200_S567_LR000002_D_E1600_BL_O8G2_R800	150	200	Multi	1600	Yes	0.000002	Yes	Yes	80	20	0.7361	0.7966	0.8843	0.7652	0.7837
I200_S567_LR000002_D_E1600_BL_O8G2_R800	200	200	Multi	1600	Yes	0.000002	Yes	Yes	80	20	0.8392	0.5233	0.9594	0.6446	0.5659
I200_S567_LR000002_D_E1600_BL_O8G2_R800	250	200	Multi	1600	Yes	0.000002	Yes	Yes	80	20	1	0.0001	1	0.0002	0.0001

I200_S567_LR000002_D_E1600_BL_O8G2	100	200	Multi	1600	Yes	0.000002	No	Yes	80	20	0.6938	0.7974	0.8574	0.742	0.7743
I200_S567_LR000002_D_E1600_BL_O8G2	125	200	Multi	1600	Yes	0.000002	No	Yes	80	20	0.7144	0.7548	0.8777	0.734	0.7464
I200_S567_LR000002_D_E1600_BL_O8G2	130	200	Multi	1600	Yes	0.000002	No	Yes	80	20	0.7193	0.744	0.8824	0.7315	0.7389
I200_S567_LR000002_D_E1600_BL_O8G2	150	200	Multi	1600	Yes	0.000002	No	Yes	80	20	0.7352	0.7062	0.8969	0.7204	0.7118
I200_S567_LR000002_D_E1600_BL_O8G2	200	200	Multi	1600	Yes	0.000002	No	Yes	80	20	0.7803	0.5744	0.9345	0.6617	0.6064
I200_S567_LR000002_D_E1600_BL_O8G2	250	200	Multi	1600	Yes	0.000002	No	Yes	80	20	0.9059	0.1385	0.9942	0.2402	0.1667
I200_S567_LR000002_D_E1600_BL_O7G3	100	200	Multi	1600	Yes	0.000002	No	Yes	70	30	0.6972	0.7895	0.8611	0.7405	0.7691
I200_S567_LR000002_D_E1600_BL_O7G3	125	200	Multi	1600	Yes	0.000002	No	Yes	70	30	0.7175	0.7473	0.8808	0.7321	0.7412
I200_S567_LR000002_D_E1600_BL_O7G3	130	200	Multi	1600	Yes	0.000002	No	Yes	70	30	0.7224	0.7364	0.8853	0.7293	0.7336
I200_S567_LR000002_D_E1600_BL_O7G3	150	200	Multi	1600	Yes	0.000002	No	Yes	70	30	0.7382	0.6974	0.8998	0.7172	0.7052
I200_S567_LR000002_D_E1600_BL_O7G3	200	200	Multi	1600	Yes	0.000002	No	Yes	70	30	0.7825	0.5646	0.9364	0.6559	0.5979
I200_S567_LR000002_D_E1600_BL_O7G3	250	200	Multi	1600	Yes	0.000002	No	Yes	70	30	0.9064	0.1334	0.9944	0.2326	0.1609
I200_S567_LR000002_D_E1600_BL_O6G4	100	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.703	0.7791	0.8666	0.7391	0.7626
I200_S567_LR000002_D_E1600_BL_O6G4	125	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.7229	0.7363	0.8856	0.7295	0.7336
I200_S567_LR000002_D_E1600_BL_O6G4	130	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.7274	0.7254	0.8898	0.7264	0.7258
I200_S567_LR000002_D_E1600_BL_O6G4	150	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.7421	0.6846	0.9036	0.7122	0.6954
I200_S567_LR000002_D_E1600_BL_O6G4	200	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.7857	0.55	0.9392	0.6471	0.5851
I200_S567_LR000002_D_E1600_BL_O6G4	250	200	Multi	1600	Yes	0.000002	No	Yes	60	40	0.9068	0.1267	0.9947	0.2224	0.1531
I200_S567_LR000002_D_E1600_BL_O5G5	100	200	Multi	1600	Yes	0.000002	No	Yes	50	50	0.7086	0.765	0.8725	0.7357	0.753
I200_S567_LR000002_D_E1600_BL_O5G5	125	200	Multi	1600	Yes	0.000002	No	Yes	50	50	0.7278	0.7192	0.891	0.7235	0.7209
I200_S567_LR000002_D_E1600_BL_O5G5	130	200	Multi	1600	Yes	0.000002	No	Yes	50	50	0.7322	0.7073	0.8952	0.7195	0.7121
I200_S567_LR000002_D_E1600_BL_O5G5	150	200	Multi	1600	Yes	0.000002	No	Yes	50	50	0.7472	0.6655	0.9088	0.704	0.6804
I200_S567_LR000002_D_E1600_BL_O5G5	200	200	Multi	1600	Yes	0.000002	No	Yes	50	50	0.7903	0.5287	0.9432	0.6336	0.5662
I200_S567_LR000002_D_E1600_BL_O5G5	250	200	Multi	1600	Yes	0.000002	No	Yes	50	50	0.9051	0.118	0.995	0.2087	0.1428
I200_S567_LR000002_D_E1600_BL_O4G6	100	200	Multi	1600	Yes	0.000002	No	Yes	40	60	0.7139	0.7442	0.8792	0.7288	0.738
I200_S567_LR000002_D_E1600_BL_O4G6	125	200	Multi	1600	Yes	0.000002	No	Yes	40	60	0.7331	0.696	0.8973	0.7141	0.7031
I200_S567_LR000002_D_E1600_BL_O4G6	130	200	Multi	1600	Yes	0.000002	No	Yes	40	60	0.7376	0.6837	0.9015	0.7097	0.6939
I200_S567_LR000002_D_E1600_BL_O4G6	150	200	Multi	1600	Yes	0.000002	No	Yes	40	60	0.7523	0.6403	0.9146	0.6918	0.66
I200_S567_LR000002_D_E1600_BL_O4G6	200	200	Multi	1600	Yes	0.000002	No	Yes	40	60	0.7944	0.5034	0.9472	0.6163	0.5432
I200_S567_LR000002_D_E1600_BL_O4G6	250	200	Multi	1600	Yes	0.000002	No	Yes	40	60	0.9054	0.1091	0.9954	0.1947	0.1323
I200_S567_LR000002_D_E1600_BL_O3G7_R800	100	200	Multi	1600	Yes	0.000002	Yes	Yes	30	70	0.6463	0.9103	0.7981	0.7559	0.8416
I200_S567_LR000002_D_E1600_BL_O3G7_R800	125	200	Multi	1600	Yes	0.000002	Yes	Yes	30	70	0.6876	0.8639	0.841	0.7657	0.8218
I200_S567_LR000002_D_E1600_BL_O3G7_R800	130	200	Multi	1600	Yes	0.000002	Yes	Yes	30	70	0.6981	0.85	0.851	0.7666	0.8145
I200_S567_LR000002_D_E1600_BL_O3G7_R800	150	200	Multi	1600	Yes	0.000002	Yes	Yes	30	70	0.7334	0.7912	0.8835	0.7612	0.779
I200_S567_LR000002_D_E1600_BL_O3G7_R800	200	200	Multi	1600	Yes	0.000002	Yes	Yes	30	70	0.8365	0.5133	0.9594	0.6362	0.5563
I200_S567_LR000002_D_E1600_BL_O3G7_R800	250	200	Multi	1600	Yes	0.000002	Yes	Yes	30	70	1	0	1	0	0
I200_S567_LR000002_D_E1600_BL_O3G7	100	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.7196	0.7176	0.8867	0.7186	0.718
I200_S567_LR000002_D_E1600_BL_O3G7	125	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.7383	0.6668	0.9042	0.7007	0.6799
I200_S567_LR000002_D_E1600_BL_O3G7	130	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.7427	0.6546	0.9081	0.6959	0.6705
I200_S567_LR000002_D_E1600_BL_O3G7	150	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.7569	0.6099	0.9206	0.6755	0.6346
I200_S567_LR000002_D_E1600_BL_O3G7	200	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.7997	0.4731	0.952	0.5945	0.5152

I200_S567_LR000002_D_E1600_BL_O3G7	250	200	Multi	1600	Yes	0.000002	No	Yes	30	70	0.9039	0.099	0.9957	0.1784	0.1204
I200_S567_LR000002_D_E1600_BL_O2G8	100	200	Multi	1600	Yes	0.000002	No	Yes	20	80	0.7247	0.6863	0.8944	0.705	0.6937
I200_S567_LR000002_D_E1600_BL_O2G8	125	200	Multi	1600	Yes	0.000002	No	Yes	20	80	0.7439	0.6337	0.9116	0.6844	0.653
I200_S567_LR000002_D_E1600_BL_O2G8	130	200	Multi	1600	Yes	0.000002	No	Yes	20	80	0.7479	0.6203	0.9153	0.6782	0.6422
I200_S567_LR000002_D_E1600_BL_O2G8	150	200	Multi	1600	Yes	0.000002	No	Yes	20	80	0.7622	0.5754	0.9273	0.6558	0.6051
I200_S567_LR000002_D_E1600_BL_O2G8	200	200	Multi	1600	Yes	0.000002	No	Yes	20	80	0.8055	0.4388	0.9571	0.5681	0.4828
I200_S567_LR000002_D_E1600_BL_O2G8	250	200	Multi	1600	Yes	0.000002	No	Yes	20	80	0.9017	0.0889	0.9961	0.1619	0.1085
I200_S567_LR000002_D_E1600_BL_O1G9_R800	100	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0.6395	0.9172	0.7905	0.7536	0.8439
I200_S567_LR000002_D_E1600_BL_O1G9_R800	125	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0.6823	0.8716	0.8355	0.7654	0.8258
I200_S567_LR000002_D_E1600_BL_O1G9_R800	130	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0.6928	0.8578	0.8458	0.7665	0.8188
I200_S567_LR000002_D_E1600_BL_O1G9_R800	150	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0.7299	0.8001	0.88	0.7634	0.785
I200_S567_LR000002_D_E1600_BL_O1G9_R800	200	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0.8343	0.5104	0.9589	0.6334	0.5534
I200_S567_LR000002_D_E1600_BL_O1G9_R800	250	200	Multi	1600	Yes	0.000002	Yes	Yes	10	90	0	0	1	0	0
I200_S567_LR000002_D_E1600_BL_O1G9	100	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.7308	0.6493	0.9031	0.6876	0.6641
I200_S567_LR000002_D_E1600_BL_O1G9	125	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.7489	0.5952	0.9191	0.6633	0.6207
I200_S567_LR000002_D_E1600_BL_O1G9	130	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.7529	0.5824	0.9225	0.6568	0.61
I200_S567_LR000002_D_E1600_BL_O1G9	150	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.7667	0.5372	0.9338	0.6318	0.5714
I200_S567_LR000002_D_E1600_BL_O1G9	200	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.809	0.4049	0.9613	0.5397	0.4498
I200_S567_LR000002_D_E1600_BL_O1G9	250	200	Multi	1600	Yes	0.000002	No	Yes	10	90	0.8993	0.0787	0.9964	0.1448	0.0963
I200_S567_LR000002_D_E1600	100	200	Multi	1600	Yes	0.000002	No	No	100	0	0.7347	0.6122	0.9104	0.6679	0.6333
I200_S567_LR000002_D_E1600	125	200	Multi	1600	Yes	0.000002	No	No	100	0	0.7526	0.5586	0.9256	0.6413	0.589
I200_S567_LR000002_D_E1600	130	200	Multi	1600	Yes	0.000002	No	No	100	0	0.7567	0.5452	0.929	0.6338	0.5775
I200_S567_LR000002_D_E1600	150	200	Multi	1600	Yes	0.000002	No	No	100	0	0.7711	0.5011	0.9397	0.6075	0.5388
I200_S567_LR000002_D_E1600	200	200	Multi	1600	Yes	0.000002	No	No	100	0	0.8125	0.3735	0.9651	0.5118	0.4188
I200_S567_LR000002_D_E1600	250	200	Multi	1600	Yes	0.000002	No	No	100	0	0.8989	0.0692	0.9968	0.1285	0.0849
I200_S567_LR000002_D	100	200	Multi	200	Yes	0.000002	No	No	100	0	0.7321	0.655	0.9029	0.6914	0.6691
I200_S567_LR000002_D	125	200	Multi	200	Yes	0.000002	No	No	100	0	0.7514	0.5964	0.9201	0.665	0.6221
I200_S567_LR000002_D	130	200	Multi	200	Yes	0.000002	No	No	100	0	0.7556	0.5819	0.9237	0.6574	0.6099
I200_S567_LR000002_D	150	200	Multi	200	Yes	0.000002	No	No	100	0	0.7718	0.534	0.936	0.6313	0.5691
I200_S567_LR000002_D	200	200	Multi	200	Yes	0.000002	No	No	100	0	0.8143	0.3879	0.9641	0.5254	0.4332
I200_S567_LR000002_D	250	200	Multi	200	Yes	0.000002	No	Nop							
Base Model, prediction on original images	100		Single	200	No	0.0002	No	No	100	0	0.5059	0.8345	0.6698	0.6299	0.7386
Base Model, prediction on original images	125		Single	200	No	0.0002	No	No	100	0	0.5332	0.7985	0.7167	0.6394	0.7262
Base Model, prediction on original images	130		Single	200	No	0.0002	No	No	100	0	0.5399	0.789	0.7276	0.6411	0.7224
Base Model, prediction on original images	150		Single	200	No	0.0002	No	No	100	0	0.5622	0.7567	0.7613	0.6451	0.7077
Base Model, prediction on original images	200		Single	200	No	0.0002	No	No	100	0	0.6334	0.648	0.848	0.6406	0.645
Base Model, prediction on original images	250		Single	200	No	0.0002	No	No	100	0	0.792	0.2758	0.9707	0.4091	0.3171