Neuronal Circuit and Synapse Analysis with Deep Neural Networks

Advisors: Dr. Daniela Boassa, Dr. Matthias Haberl
Team Members: Faezeh Ghazi, Jason Kha

Abstract

Biomedical image segmentation has led to numerous breakthroughs in neuroscience research by helping scientists map the brain. Brain mapping could be the key to understanding the progression of neurodegenerative diseases. Currently, the most accurate neural segmentations and synapse detections are annotated manually. However, manual annotation of synapses is unsustainable given the very large size of electron microscopy datasets of the brain. Deep learning tools such as CDeep3M can produce automated synapse segmentations on serial block-face scanning electron microscopy (SBEM), but the segmentations are relatively inaccurate. This is due to the numerous barriers including the inherent heterogeneity of neurons and their synapses and the limited tools for assessing and analyzing ultrastructure in molecularly defined synapses. Our developed solution for improving the automated segmentations of synapses, known as CDense3M, could help neuroscientists discover new insights in neurodegenerative diseases such as Alzheimer's and Parkinson's disease. We implement and apply two different modeling components to perform 3D segmentations and labeling to improve the accuracy of synaptic density segmentations. The first modeling component creates different algorithms for 3D concatenation, labeling, and filtering. As for 3D filtering, we also implement multiple models that use erosion and determine the statistical thresholds for 3D vesicle filtering on 3D objects. In each step of our 3D modeling, the new data is based off of previous modeling and as a result the data in each step of modeling becomes the new data for the next step. The evaluation for our image processing including 3D labeling and 2D and 3D vesicle filtering models is through the visualization and using the biological behavior of known neuronal structures. This improves the accuracy of defining the presynaptic areas based on the vesicle functions. The second modeling component applies flood-filling networks, a class of 3D convolutional neural networks, to perform cell-body segmentation on a 3D image volume. We train a custom flood-filling network model on membrane segmentations produced by the CDeep3M automated segmentation tool. The membrane model is more generalizable at segmenting cell bodies than the standard FIB-25 model trained on raw electron microscopy data from fruit flies. The improvement in the accuracy of presynaptic area recognition from applying both modeling components results in a 10% increase in precision and recall for the voxel-wise classification of synapses in a 3D image volume of the *Nucleus accumbens* from lab mice.

Introduction and Question Formulation

Challenge

Electron microscopy datasets of the brain are very large, on the scale of 75 million neurons for a mouse brain and 86 billion neurons for the human brain. Even the brain of a fruit fly, with just 100,000 neurons, requires 100 TB of image data. Identifying organelles from these 3D image

volumes is quite labor-intensive. For example, the manual segmentation of all mitochondria in one mouse brain would require 24/7 work for one year by 2.7 million scientists.

Manual segmentation of 3D image volumes of the brain is not a viable long-term approach. CDeep3M is a deep learning software toolkit that automates the image segmentation of a variety of organelles in 3D light, electron, and X-ray microscopy. Currently, the segmentations for membranes, mitochondria, and synaptic vesicles are fairly accurate, but the segmentations for synapses still require considerable improvement.

There are hundreds to thousands of small objects in each presynaptic terminal, also known as vesicle clouds, some with sharply delineated membranes and others with much blurrier edges, spanning between one to a few sections. This combination makes it a particularly difficult task for human and computer segmentation to identify individual synapses.

Ingredients of Data Science Problem

Our data science problem is that the current CDeep3M neural network model trained on synapses from serial block-face scanning electron microscopy (SBEM) predicts too many false positives.

We identify these synapses as false positives based on biological knowledge. For most of these false positives, we can categorize them into the following groups based on the biological principle that is violated:

- Synapses erroneously overlapping with mitochondria
- Synapses that do not overlap enough with a membrane
- Synapses that do not have vesicles on one side

The goal of this capstone project, Neuronal Circuit and Synapse Analysis with Deep Neural Networks, is to increase the accuracy of the automated segmentation of synapses. We extend the existing CDeep3M framework by adding two new modeling approaches that leverage 3D image analyses and deep learning to achieve higher accuracy on the automated dense segmentation of synapses. Hence, we have given our developed solution the name CDense3M.

Given the relationship between synapses and these three other organelles, we can use knowledge from the segmentations of these three organelles to improve the segmentation of the synapses via image processing. However, thresholding alone will not increase the accuracy of the synapse segmentation enough to be considered acceptable. In turn, we explore 3D vesicle filtering and 3D labeling of the presynaptic sites. In parallel, we use flood-filling networks (FFN), a 3D convolutional neural network, to perform automated neuronal tracing as a possible approach to improve the 3D labeling algorithm. Accurately labeling 3D presynaptic areas is the key to significantly improving the segmentation of the synapses.

The ingredients that form our data science problem are the raw SBEM data of brain tissue from lab mice, the resulting segmentations produced by CDeep3M, the image processing techniques used to slightly improve the synapse segmentation accuracy, and the more complex 3D instance label segmentation algorithms used to significantly improve the synapse segmentation accuracy.

Questions

1. How can we use biological knowledge to improve the segmentation accuracy of synapses?
2. Why is thresholding and filtering not enough to increase the accuracy of the synapse segmentation?
3. Why do we apply 3D image processing?
4. What are the fundamental differences of 2D and 3D algorithms and what techniques and tools are applied in 2D, but not 3D, and vice versa?
5. How is the data and model in the next step of image processing related to the previous step?
6. How do we evaluate our image processing models?
7. Which features should we use in 3D image analyses to more accurately label the 3D presynaptic areas?
8. What tuned statistical thresholds can be used to improve the accuracy of presynaptic recognition and in which model?
9. What morphological image processing operations (erosion, dilation) are needed to further improve 3D presynaptic area recognition?
10. Can transfer learning be applied to improve synapse segmentation accuracy by using a pre-trained FFN model on membrane segmentations to segment and trace neurons?
11. Which data preprocessing steps need to be taken to ensure optimal FFN performance?
12. How should we integrate 3D filtering and FFN into CDeep3M?
13. How do we improve the scalability and usability of our models?

Related Work

CDeep3M is the existing tool that our project is attempting to improve with our developed solution called CDense3M. CDeep3M—Plug-and-Play cloud-based deep learning for image segmentation by Haberl et al. is the original publication on CDeep3M. Our advisor Matthias is the lead author, and our advisor Daniela is a co-author. This paper describes the automated image segmentation workflow provided by CDeep3M.

There are three main reasons why we consider CDeep3M state-of-the-art. The first reason is the generalized applicability of CDeep3M on a variety of image segmentation tasks, such as cellular organelle segmentation and cell counting and classification. The second reason is that CDeep3M deep neural network models outperform existing models in the field such as three-class Conditional Random Field and Ilastik. For some datasets, the segmentation accuracy of CDeep3M is equal to the accuracy of human expert annotators, according to the consensus ground truth labels. The third reason is CDeep3M's application of transfer learning, specifically domain adaption, to speed up training. The paper describes how CDeep3M successfully retrained a convolutional neural network trained on scanning electron microscope (SEM) to fully adapt the network to an SBEM dataset. This shows that adaption of pre-trained models can reduce the effort and time required for training by up to 90%. Our capstone project intends to build on CDeep3M to improve its performance in the dense segmentation of synaptic densities.

High-precision automated reconstruction of neurons with flood-filling networks by Januszewski et al. is the original publication on flood-filling networks. FFN is a 3D convolutional neural network that contains in addition a recurrent pathway that allows the iterative optimization and extension of individual neuronal processes. It has outperformed other automated segmentation approaches on FIB-25, a public dataset of *Drosophila* optic lobe electron microscopy. This capstone project uses a different FFN model that Matthias trained on membrane segmentations instead of FIB-25 to increase the generalizability of FFN on different types of electron microscopy data besides FIB-25.

Team Roles and Responsibilities

Faezeh Ghazi is the project manager.

Jason Kha is the budget manager and record keeper.

Faezeh worked on the image processing. She used data modeling in 2D image analyses and their binaries to create the data for 3D image analyses. She wrote different algorithms to break down the 3D image processing into different steps of 3D labeling, 3D concatenation, and 3D vesicle filtering. She dedicated a large amount of time to the 3D color labeling visualizations of the presynaptic sites. Faezeh also implemented erosion models to further improve the accuracy of 3D presynaptic area recognition and ran several combinations of her models to test her hypotheses. She iterated through the scientific method several times to find that 3D presynaptic object erosion combined with a specific 3D vesicle filtering threshold. These strategies led to more accurate presynaptic area labeling, which improves the accuracy of synapse segmentations. Faezeh documented her findings very thoroughly in the step reports and presentations. In these documented reports, she explained her progression and thought process of how she refined the 3D filtering and labeling models to improve their performance.

Jason worked on the FFN approach to improve the accuracy of synapse segmentations. Jason got the FFN membrane model to segment well on its own training data by retraining the model for another two million epochs. Afterwards, he implemented image preprocessing techniques to help the FFN membrane model segment a test image volume more accurately than the baseline FIB-25 FFN model, proving that the FFN membrane model is more generalizable than the FIB-25 model. Jason fully integrated FFN into the original CDeep3M data pipeline and added FFN color overlays on the raw EM data. He also was responsible for designing the project's solution architecture. He successfully deployed his pipeline in both AWS inside a Docker container, and in Google Colaboratory (Colab) as a Jupyter notebook.

Data Acquisition

Data Sources

Our data consists entirely of electron microscopy images of brain tissue. The only training data we use is a set of 100 images of cleaned CDense3M membrane segmentations, along with their corresponding ground truth labels. This dataset is used for training our custom FFN membrane

model to perform cell-body segmentation. The original FFN training data is the only training data we need because we had to retrain the FFN membrane model to improve its performance. All of our other modeling work consists of performing image processing on segmentations outputted by CDense3M. The existing CDense3M models we use to segment membranes, mitochondria, and synaptic vesicles are already pre-trained, minimizing the need for additional training data. Accessing data from a variety of sources using different technologies helps us solve the questions we identified by adding another dataset that could potentially support our hypotheses. Data variety may also help us discover new findings related to the different sources and technologies used to collect the data, which will make our answers to the questions more generally applicable. The table for our training datasets is below.

Name	Volume	Variety	Velocity	Location
FFN Training Images (Membrane Segmentations)	100 1024x1024 pixel images (44.3 MB)	PNG, grayscale 8-bit image (uint8)	N/A	https://drive.google.com/drive/folders/1D-WzDtsh1uZwJ7sqL4_WwpU-iC1Mb6wk
FFN Ground Truth Cell Body Segmentation Labels	100 1024x1024 pixel images (4.1 MB)	PNG, color 64-bit image (int64)	N/A	https://drive.google.com/drive/folders/1EfFlA0qfTQqymTnLGOnMd8AyJRgXychJ

Table 1: Training Data

We used two validation datasets to tune our models. The first dataset is from the *Nucleus accumbens* region in the brain of C57BL/6NHsd mice, a common strain of lab mice. We use this dataset to tune both the 3D filtering and labeling algorithms and FFN. The second dataset is the FIB-25 dataset of optic lobe electron microscopy of *Drosophila*, more commonly known as fruit flies. FIB-25 is used to tune the FFN. Both datasets are quite large in size in their original form, so we cropped a subset of the 3D image volume when developing our models.

Name	Volume	Variety	Velocity	Location
Nucleus accumbens Validation Images	100 1024x1024 pixel images (105.3 MB)	TIFF, grayscale 8-bit image (uint8)	N/A	https://drive.google.com/drive/folders/1GmfrqFrKdTQZNqwC_y2Ua2d514LbpnAZ
Nucleus accumbens FFN Validation Ground Truth Cell Body Segmentation Labels	100 1024x1024 pixel images (4.8 MB)	PNG, color 64-bit image (int64)	N/A	https://drive.google.com/drive/folders/1rxxQxB6dClJauiOjzSYXAp0G9xHU2oif
Nucleus accumbens Validation Ground Truth Synapse Segmentation Labels	100 1024x1024 pixel images (3.3 MB)	PNG, grayscale 8-bit image (uint8)	N/A	https://drive.google.com/drive/folders/1vgUUgWiWGRnfNhpBm8XMgoMTKOtk6H4j
FIB-25 Validation Images	250	PNG, grayscale 8-	N/A	https://drive.google.com/drive/folders/1RtFD5R4e

	250x250 pixel images (18.5 MB)	bit image (uint8)		FyI3ijZoahx3dph2BaNxaKI3
FIB-25 FFN Ground Truth Cell Body Segmentation Labels	250 250x250 pixel images (0.9 MB)	PNG, color 64-bit image (int64)	N/A	https://drive.google.com/drive/folders/1gI1aX336kFDkP9V3ugcgaj-7ZFzhFfzF

Table 2: Validation Data

Our test data is a different cropped subset of the *Nucleus accumbens* image volume. This is used to evaluate the accuracy of FFN segmentation and 3D synaptic area labeling.

Name	Volume	Variety	Velocity	Location
Nucleus accumbens Test Images	100 1024x1024 pixel images (105.3 MB)	TIFF, grayscale 8-bit image (uint8)	N/A	https://drive.google.com/drive/folders/1pmr2WA4oIaVgkf27C9awRCnSpHf0k9Dd
Nucleus accumbens FFN Test Ground Truth Cell Body Segmentation Labels	100 1024x1024 pixel images (4.8 MB)	PNG, color 64-bit image (int64)	N/A	https://drive.google.com/drive/folders/1O0iu32wEjupx50Ep0N_Q3GftVL7lyN9s
Nucleus accumbens Test Ground Truth Synapse Segmentation Labels	100 1024x1024 pixel images (3.3 MB)	PNG, grayscale 8-bit image (uint8)	N/A	https://drive.google.com/drive/folders/1q6L7rU7yIPNKiGymSyn04LDtOUVGz5ej

Table 3: Test Data

Data Collection

The size of the full *Nucleus accumbens* image volume that we used is 121 GB. However, using the full *Nucleus accumbens* dataset would be infeasible for our computationally expensive deep learning models, so we took a 1024x1024x100 voxel slice that is 105.3 MB. The size of the full FIB-25 dataset is 100 GB and we use a 250x250x250 sample of that, which is 19 MB.

The raw training data comes from SBEM microscopy data of C57BL/6NHsd laboratory mice brain tissue. SBEM was performed using a Merlin SEM (Zeiss) with a Gatan 3View system at high vacuum. SBEM constructs high resolution 3D images of tissues by repeatedly using an ultramicrotome knife to cut a thin section from a block face and then imaging the next layer.

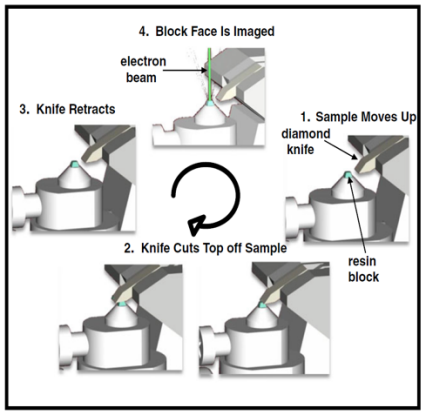

Figure 1: SBEM Data Collection Process

Data Pipelines

Our data pipeline, also known as CDense3M, can be divided into three sequential stages. The first and leftmost diagram is the original CDeep3M pipeline. The middle diagram is FFN inference. The rightmost diagram is 3D vesicle filtering and 3D presynaptic site labeling.

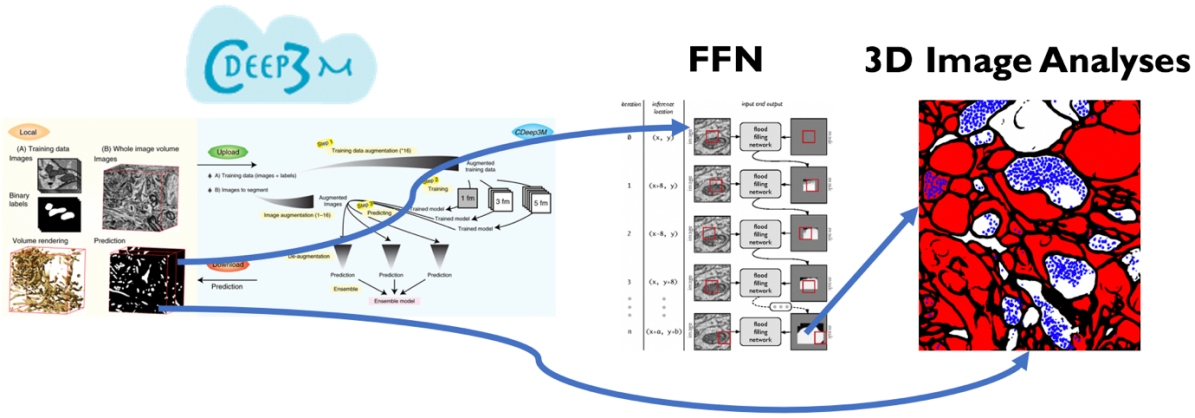

Figure 2: CDense3M Data Pipeline

The CDeep3M data pipeline takes two data sources as input: 2D images slices and image labels as training data, and a whole 3D image volume to predict. It performs data augmentation on both, and then segmentation on the test volume and training in parallel. Once both processes are complete, it uses the ensemble of up to three trained models to predict the segmented test volume. For the neural network models, 1fm or one frame is a 2D model, whereas the 3fm and 5fm are 3D models. The 3fm model uses 3 frames and the 5fm model uses 5 frames for each predicted image depth-wise by a constant z-step size. Finally, the slices are de-augmented and merged back together to form the segmentation layer predictions for the test volume. For our capstone project, we only used CDeep3M pre-trained models specific to membranes, mitochondria, vesicles, and synapses. Since there is no need for training data, we only need to provide the 3D image volume test dataset of the raw EM images as a folder of TIFF files.

The diagram in the middle represents the FFN inference pipeline, which can only take membrane segmentations as input, and outputs 3D cell labeled segmentations.

The diagram on the right is the 3D image analyses pipeline, which takes the FFN cell segmentation output and synaptic vesicle segmentations from CDeep3M as input, and outputs the presynaptic site and synapse labels.

Data Environment Setup

CDeep3M and FFN are typically run in the cloud because they consist of neural networks, which are so computationally intensive that they require graphics processing units (GPUs) to be usable for most deep learning tasks. However, the 3D image analysis was done locally. All our data is in flat files because the data only consists of image. It does not make sense to insert images into a database. 3D image volumes are either stored as a folder of its 2D image slices and labels in TIFF format, or an HDF5 file that contains the entire 3D image stack.

Data Preparation

Data Quality Issues

The quality issues with the datasets are relatively minor. Some image stacks have noise along the edges that confused CDense3M when it made its predicted segmentations. A voxel exclusion cropping zone around the edges is used in order to compensate for inaccuracies of human annotations along the borders of objects. This zone is usually one or two voxels wide. CDense3M membrane segmentations tend to include stray myelin sheaths and membrane holes. We use a membrane completion network, which is a CDense3M pre-trained model, to clean up stray myelin and fill in the membrane holes in the membrane segmentations before running FFN inference.

Data Transformation and Integration

We use the IMOD software to crop out a smaller portion of the 3D image volume to work with, while maintaining an accurate 3D reconstruction and modeling of microscopy images.

The data formats we work with are listed and described below:

- DM4
    - Data format for raw electron microscopy data
- MRC
    - Data stack
    - Can be read or opened partially
    - No compression
    - Can only be opened with molecular microscopy specific software like IMOD
- TIFF
    - Usually individual files, but can also store multiple images in one TIFF
    - Usually uncompressed

    - Slower data transfer
    - Quick to read, since no decompression required
- PNG
    - Strictly single-image format
    - Lossless compression
    - Fast data transfer
    - Slower to read for large files, since it requires decompression
- HDF5
    - Made for big data
    - Supports n-dimensional datasets
    - Each element in the dataset may itself be a complex object
    - Can have multiple containers (folder structure)

Raw electron microscopy data for the *Nucleus accumbens* are stored in DM4 files. These DM4 files are then transformed into an MRC stack with the IMOD dm2mrc command. The *Nucleus accumebens* bin-2 MRC aligned stack file is still 17 GB with 16606x10438x403 voxels. It would be impractical to run deep learning algorithms on a 17GB image volume, so we use the IMOD trimvol command to extracts a subset of the volume as a new MRC stack file.

The validation and test images for the *Nucleus accumbens* data were created using the trimvol command to extract two different 1024x1024x100 voxel stacks, which are a much more reasonable 40 MB. These smaller volumes are then converted into 100 separate TIFF files for each image slice depth-wise using the IMOD mrc2tif command. Although IMOD can also convert MRC files to PNG, we use TIFF files because they can be read faster than PNG files.

Next, the CDense3M segmentation prediction command segments a specified organelle, which is determined by the CDense3M model used, in all TIFF or PNG images of the raw data in a defined input folder. CDense3M then produces grayscale 8-bit organelle segmentations as PNG files.

FFN inference expects its input 3D image volume to be an HDF5 file. Thus, the PNG membrane segmentations outputted by CDense3M must be converted to a single HDF5 file. This is done automatically using a Python script. FFN inference saves its predicted 3D cell segmentations both as an HDF5 file and as separate PNGs. The 3D image analyses algorithms work directly on the PNG segmentations of cell bodies predicted by FFN and of vesicles predicted by CDense3M.

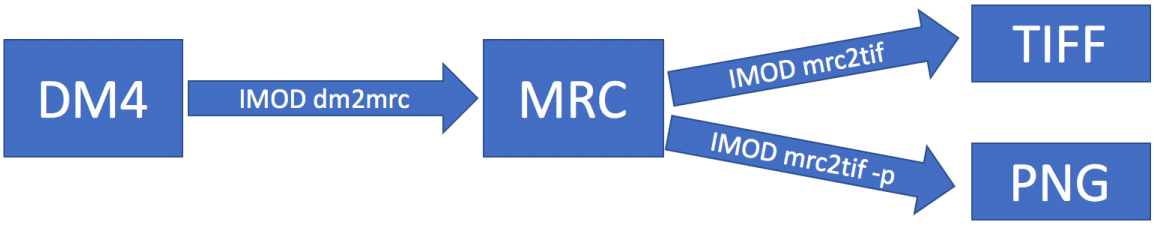

Figure 3: Data Transformation Process

Data Preprocessing

The significance of the pre-processing methods we used in our project is that simple image processing techniques can result in a nontrivial increase in segmentation accuracy. CDense3M includes a Python script that enhances the stack of raw EM data images before predicting the segmentations on it. The image enhancement data preprocessing script removes extreme outlier pixels and denoises each image using the scikit-image, OpenCV, SciPy, and NumPy.

Data preprocessing methods require to integrate both 3D image processing and FFN inference with the existing CDeep3M pipeline. For 3D image analyses, new data is based on previous data. In other words, the 3D image analysis is a sequential model, where the output of the previous model becomes the input of the next model. For example, the results of 2D image analyses with the positive and negative controls of mitochondria and membranes for synapses became the data for the analyses of the next step of 2D vesicle filtering. The data from 2D image analyses became the new data for 3D image processing. Also, in 3D image analyses, the data from each step of 3D modeling became the data foundation for the next step. For instance, the data from 3D modeling and concatenation became the data for the 3D vesicle filtering and erosion models to identify and recognize synapses based on the vesicle functions. Since each data output is the input of the next step, it is crucial to provide accurate data initially. This evaluation through visualization and biological interpretation supports the determination of any quality issues in our modeling in which its output of visualization becomes the new data. Every step of data modeling requires different Python scripts to do the model evaluation and provide the data for the next step.

Data preprocessing methods were also required to integrate FFN inference with the existing CDeep3M pipeline. We had to write a custom Python script to convert the CDeep3M membrane segmentations, which are separate PNG images for each image slice, into a single HDF5 stack in order to link the existing CDeep3M workflow with FFN. This required inserting our Python script into the pipeline right after CDeep3M produces PNGs containing the ensembled membrane segmentations, but before calling FFN inference. The NumPy and H5py packages were used to convert a series of PNG files into one HDF5 file.

Feature Selection

We selected and managed features based on the data type of the images being used. However, in this capstone project, we only work directly with the pixel values of any given image. As for the image analyses, feature selection depends on evaluation of the models derived from the visualization as well as neurological interpretation. For example, 3D labeling identified those labels under the same identifiers for the 3D concatenation. Also, the erosion models applied on 3D objects of labels and vesicles underline those labels that are presynaptic. FFN does also look at the contour lines as a feature in its watershed flood-filling algorithm, but we did not interact directly with the contour lines when running FFN inference.

The raw EM data TIFF files that are to be segmented by CDense3M are typically 8-bit unsigned integer grayscale images (CDense3M segmentations are always 8-bit unsigned integer (uint8) grayscale images, whose pixel values range from 0 to 255, inclusive. Out of the box, FFN inference outputs its cell-body segmentations as npz files, which are NumPy arrays compressed using gzip. After unzipping the npz file, we save the NumPy array with the float64 data type to

support segmentations with a large range of pixel values and avoid clipping at too low of a maximum pixel value, which distorts the segmentation. In addition, the Fiji ImageJ software does not support uint32 or uint64 images, so we had to use float64 to gain additional precision.

One could consider each individual pixel value in an image to be a separate feature, so a 1024x1024 pixel image would have 1,048,576 features, and a 1024x1024x100 voxel image volume would have 104,857,600 features. All three components of CDense3M consider each pixel value as a feature, which are used by the models to segment and label the input images.

Analysis Methods

Preliminary Analysis Method Identification, Significance, and Influence

For exploratory data analysis, we used CDeep3M to generate segmentations on the same *Nucleus accumbens* 3D image volume for four different organelles: membranes, mitochondria, synapses, and synaptic vesicles. Our first task was to overlay these four segmentations on top of an enhanced version of the raw EM data, with each organelle color-coded in a different color.

Predicted segmentations were generated by submitting 20 1024x1024 pixel images of the *Nucleus accumbens* data to the CDeep3M-Preview web interface, which runs on the Nautilus Pacific Research Platform (PRP) cluster. As CDeep3M can only recognize one organelle at a time, we had to repeat this process four times by selecting each organelle-specific model in order to generate each organelle-specific segmentation.

The stacked multi-color overlays with the four organelles distinguish each object with contrasting colors, which helps determine where the models are making conflicting predictions. For example, CDeep3M often predicts the presence of synapses when the structure is actually a mitochondrion. This preliminary analysis of visualizing multiple segmentations with colored overlays on a single image helps us identify current weaknesses in CDeep3M's algorithms. Our advisors identified this preliminary analysis method as a good introductory task to familiarize ourselves with the data and the core of our data science problem. By color coding each cell structure with a different color, it is be much easier to recognize errors visually. The significance of using this stacked colored overlay method is that it provides a clear picture of our starting point of synapse segmentation accuracy with CDeep3M alone, revealing both its strengths and weaknesses in its segmentations. Moreover, this analytical method is used at the end of the project to visually determine whether our models did result in an improvement for synapse segmentations, and where our models could still improve further.

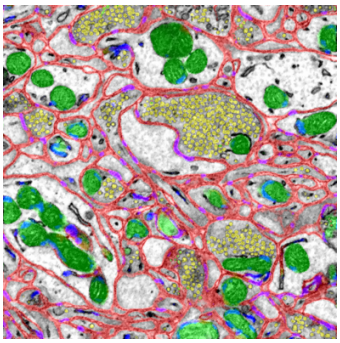

Figure 4: Stacked Overlay of Four Cell Structures in Mouse Brain Tissue

The structures outlined in red are membranes. The structures outlined in green are mitochondria. The structures outlined in blue are synapses. The structures outlined in yellow are vesicles.

Based on the multicolor stacked overlay of the test image above, the segmentations for mitochondria and membranes are excellent. The segmentations for synaptic vesicles are good. However, the segmentations for synapses are poor, with many false positives and erroneous overlap with other organelles, especially mitochondria. Exploratory data analysis shows there is a problem at the segmentation of synapses produced by CDeep3M's models erroneously overlapping with mitochondria and not overlapping with the membranes.

Our minimum viable modeling product (MVMP) attempted to correct problems identified in our EDA. It involves using the corresponding segmentations of mitochondria as a negative control filter and corresponding segmentations of membranes and vesicles as a positive control filter against the segmentations of the synapses, to ensure that the segmentations of the synapses do not appear where mitochondria are located, and only appear if the synapse is near a membrane.

This exploratory data analysis and the MVMP influenced the design of the next steps of the project in that based on the results of applying our MVMP to the data, we each focused on one modeling approach to further improve the segmentation of synapses. These analysis methods also define our data science questions further in that we address weaknesses in the current CDeep3M synapse model with more sophisticated techniques.

Involvement in Applying Analysis Techniques

The first analysis technique leverages the biological knowledge that synapses that have no vesicles on one side should be removed.

A presynaptic neuron communicates by releasing neurotransmitter chemicals, which then move across the synapse to be detected by and bind with receptors in the postsynaptic neuron. These neurotransmitters are present within synaptic vesicles clustered at presynaptic terminals. If the vesicles are not there, the signals cannot get through and the two neurons will not be able to communicate. Therefore, all synapses must be near synaptic vesicles.

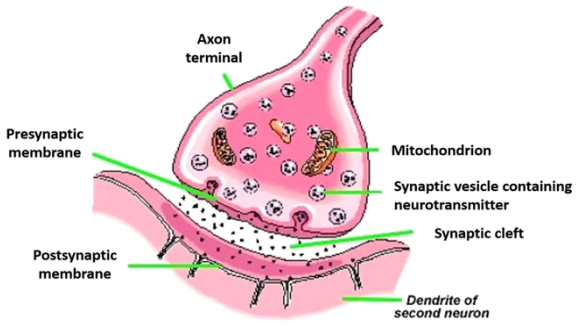

Figure 5: Synapse Structure

The 3D vesicle filtering algorithm uses presynaptic sites to filter synapses, as every synapse should be touching a presynaptic site. It assigns the membrane as the boundary of the presynaptic site and grows the vesicle area until it reaches the membrane using image dilation techniques. The algorithm removes synapses that have less than 80% overlap with the membranes. For synapses that are 100% within the membrane, they are assigned to the nearest labeled presynaptic sites. Then those synapses that are completely on the membrane will be in the same area as vesicle when we grow the vesicles.

The second analysis technique assigns the cell identities in 3D using flood-filling networks (FFN), which are 3D convolutional neural networks designed for the instance segmentation of 3D image volumes. FFN is implemented in TensorFlow and provides scripts for both training and inference. We use a custom FFN trained model trained on the membrane segmentations from CDense3M to perform cell-body segmentation. Initially, we believed that we would only need to run FFN inference, but it ended up that we also had to retrain the custom FFN membrane model for another two million epochs to obtain high-quality results, even on the training data.

In the flood-filling algorithm, the membranes serve as the watershed lines. Neighboring pixels of each flood marker are inserted into a priority queue based on the gradient magnitude of the pixel. Each pixel is assigned the label of its marked neighbors when removed from the queue, and all non-marked neighbors are placed on the queue. This process repeats until the queue is empty. After all pixels are labeled, FFN inference is done with its cell-body segmentations.

Using the cell boundaries predicted by FFN, we can identify the presynaptic neurons using 3D image analyses techniques described later on. We use the Fiji ImageJ tool to apply a 3-3-2 RGB image lookup table to add color to the FFN segmentations, which gives us a clear picture of how well FFN inference is performing. The only involvement in applying the FFN inference analysis technique to our data is configuring a Protobuf file with model parameters and data locations, and executing a Python script to run FFN inference and perform the cell-body segmentation.

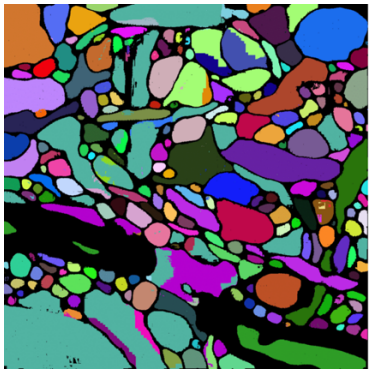

Figure 6: Cell Identities Assigned by Flood-Filling Network

Basic Analysis Techniques

Our first cut at a solution, or MVMP, involved erasing synapse segmentations that either overlap with mitochondria or do not overlap with membranes. This is a basic analysis technique because it can be implemented in a simple Python script, using standard data science libraries such as NumPy and Pillow. We used the MVMP as a basic analysis technique because our exploratory data analysis and biological principles both indicated that the MVMP would be a simple model that could eliminate the most conspicuous synapse segmentation errors.

Our MVMP led to a 2% increase in validation voxel-wise accuracy of synapse segmentations compared to the unfiltered model. However, even with our thresholding and filtering MVMP, synapse segmentations are still not accurate enough. There are too many false positives. Synapses should only be adjacent to presynaptic cells, which must contain synaptic vesicles in order to facilitate the transmission of neurotransmitters from the presynaptic neuron to the postsynaptic neuron. The results from applying our MVMP led us to implement and apply our two more advanced models – 3D image analyses and FFN.

After mitochondria and membrane filtering, the aim of reducing the presynaptic recognition and synapses errors influenced the design of the 2D vesicle filtering method. The data from the previous filtering step is our new data for this. We use the remaining synapses, derived from the negative control of mitochondria and positive control of vesicles. This 2D vesicle filtering method leverages biological knowledge and eliminate the remaining synapses that have less than a certain number of vesicles on one side defined by a threshold. In the 2D vesicle filtering, we set the threshold on the number and area of vesicles and membranes to determine the appropriate presynaptic areas. The threshold for membrane is 80% and the threshold for vesicles is 5 and 100 pixels.

The method for 2D vesicle filtering is to grow a region around the vesicles until it reaches the membranes, define this as the presynaptic site, and accept all synapses that are on a membrane, that is part of this presynaptic site. In order to do so, the kernel was used for the synapses and vesicles separately to reach the boundary of a presynaptic site. The results of the dilation of vesicles and synapses to find the presynaptic areas will be the same. Those presynaptic sites could then be colorized to visualize if our method is performing accurately. The vesicle filtering algorithm uses presynaptic sites to filter synapses, as every synapse should be touching a presynaptic site. We have assigned the membrane as the boundary of the presynaptic site, and the

code grows the vesicle area until it reaches the membrane using image dilation techniques. The algorithm removes synapses that have less than 80% overlap with the membranes. For synapses that are 100% within the membrane, they are assigned to the nearest labeled presynaptic sites. Then those synapses that are completely on the membrane will be in the same area as vesicle when we grow the vesicles. Then, the algorithm applies the vesicle filtering threshold of 5 vesicles and 100 pixels and removes those synapses that are below this threshold. The primary analyses of visualizing the presynaptic sites that will be colorized help us to evaluate our model performance. The significance of using 2D filtering is we applied the grown vesicles along with the statistical threshold for our primarily model performance analyses and was able to find the best models. As for the visualized primary analyses, we visualized the model based on the vesicle threshold of 5 objects on the left-side and threshold of 100 pixels on the right-side below.

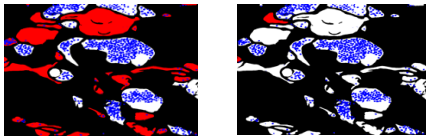

Figure 7: Different presynaptic and vesicles for 5 vesicles on the left and 100 pixels on the right

The primary analyses of our above visualization finding indicates that the model with 5 vesicle objects will give us more accurate results, as we can keep more presynaptic areas that pass the vesicle thresholds. As a result, the binary image of synapses and presynaptic areas before and after filtering for the threshold of 5 vesicles is shown in Figures 8 and 9.

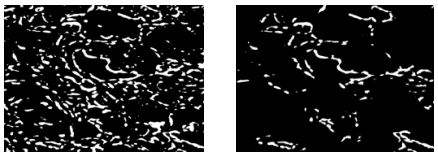

Figure 8: Synapses before filtering on the left and synapses after filtering on the right

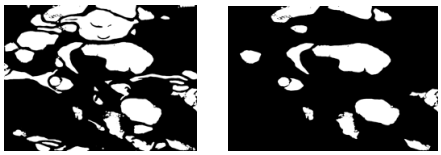

Figure 9: Presynaptic areas before filtering on the left and after filtering on the right

In Figure 10, we considered the positive vesicle control variables with the threshold of 5 to combine the segmentations of the 4 cell structures of mitochondria, synapses, vesicles, and membranes into a single combined segmentation image. We then overlay these combined segmentation images on top of an enhanced version of the original microscopy image with each organelle color-coded in a different color.

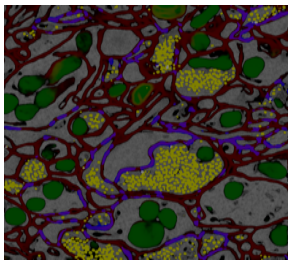

Figure 10: Stacked Overlay of Four Cell Structures after vesicle filtering

The results of visualization in terms of images obtained from the model are the predicted output. The predicted output has to be evaluated for its accuracy and significance. The findings of the above visualization demonstrate that the 5-vesicle threshold is better than 100 pixels vesicle threshold. This is because we can keep more accurate presynaptic areas, which leads to better synapse predictions. Figures 6 and 7 indicate that filtering is effective on reducing the synapse's errors. It can be seen that the number of synapses and 3D objects(labels) are reduced as the errors were eliminated. This threshold of 5 vesicles gives us the output of Figure 8 that has a higher prediction accuracy.

Next Steps for 2D Model Evaluation and the Reason for Applying 3D Image Processing

The problem of 2D image filtering is the loss of some presynaptic sites in our microscopy images that led to inaccurate presynaptic recognitions. The reason for the loss of presynaptic areas is 2D vesicle filtering on segmented images of 3D microscopy images. The 3D microscopy images are sliced into segmented images. These segmented images consist of 3D objects in which the membranes are their boundary. One can visualize each 3D object in each image as different pieces of a worm's body. As a result, these 3D objects in each image are in contact with 3D objects in other images as they contribute to each other to form the worm, for example. For instance, a worm's head is a 3D object in one image, and its body is composed of other 3D objects divided in other images. These 3D objects that are connected throughout the images may or may not have vesicles. The problem caused by 2D vesicle filtering on these images is that it considers these images individually without the connection of 3D objects of images. This affected the count of vesicles as the number of vesicles in each 3D object that should be summed together are counted separately. This 3D object's connection is one of the fundamental differences of 2D and 3D image processing that can make vesicle filtering and grown regions. As a result, some 3D objects of images that do not meet the vesicle threshold requirement are deleted by mistake at the time of 2D vesicle filtering. These 3D objects could be presynaptic sites and meet the requirement of vesicle filtering if their connection with other images' 3D objects are taken into account. Therefore, the preliminary analyses of these inaccurate presynaptic areas help us move toward the applicable solution of 3D image analyses. The significance of using this 3D analyses is this method considers the connection of 3D object's vesicles, so the vesicle counts in each 3D object will be summed together. 3D image processing can be dissected into different steps that are 3D concatenation, 3D labeling and 3D filtering. These techniques involved in 3D modeling are not applicable in 2D image processing in that its algorithm interacts with each image's 3D objects separately. The significance of our 3D image processing is this 3D technique can help improve the accuracy of presynaptic site recognition.

Creating the 3D Images

A Python program was written to create the 3D images stack using CDeep3M's pre-trained models for synapses, membrane, vesicles, and mitochondria. The technique for creating 3D images is to integrate the shape as a parameter into the 3D pixels in which the area is on the *xy* plane and depth along the *z*-axis. When this one dimension is added to all the segmented 2D images, the images are concatenated along the depth of the z-axis.

3D Concatenation

3D concatenation merges 3D objects of each image with each other. The problem is that the location of 3D objects that are in contact with each other are not the same throughout the images. We considered these 3D objects as the pieces of a worm's body that are divided into different pieces across the images. These different pieces as 3D objects have different areas in each image that caused the problem of not being exactly at the same location in each image. As a result of this issue, the irrelevant 3D objects that should not be linked together become one area and get one label. The primary analyses of this 3D concatenation issue help identify the 3D objects' labels in each image.

3D Labeling

3D labeling is the technique used to set the label for each 3D object throughout the images. The 3D images created in our previous method are the new data for 3D labeling. Since all the 3D objects in each image are related to 3D objects of other images, the kernel was used in our program to link the 3D objects from up and down, right and left. Also, the front and back (depth) of 3D objects of each image were taken into account. For each pixel, we looked at the cube around it. The 3D labeling model was calculated both through the similar areas and overlapping to evaluate the model performance through the primary analyses of visualization. In both methods, the threshold is based on the areas above 80%. The significance of 3D labeling is the labeling of the next image was based off the labeling of the previous image. With this, we labeled the first image and then labeled the second image based on the following algorithms.

The first algorithm for 3D labeling intended to label the 3D objects of new images that could be found in the old images. If the 3D object's areas of image 2 had above 80% similarity with the area of image 1's 3D objects, the area of image 2 was updated to get the same label as the area of image 1. As a result, the updated label of new images is based on the labeling of the last images was saved in the matrix. We first assigned the value of zero to all elements of our matrix, and then updated the labels according to the areas of previous image's 3D objects. As for the overlapping regions, if the area of the old image overlapped two areas of new image, the label of the area that had more overlapping regions would be the label for both areas.

The second algorithm of 3D labeling aimed to label the 3D objects of new images that could not be found in the old images. The biggest label method was used for this labeling. The purpose of the biggest label method was to assign the new label for those labels of the new images that were not in the labels of previous images. If there were some areas in image 2 that had either no overlap or overlapping areas less than 80%, those areas of image 2 were assigned the new labels. This new label of new images was based on the biggest label. For example, if the biggest label

was 117 in image 2 and 100 labels of image 2 were in common with image 1, then the labels from 100 until 117 (the rest of labels) got the new label. These new labels also got a new color.

Analytical Workflow

The analytical workflow for the two new modeling components of CDense3M are quite similar. We run through the algorithm with the default parameters to get a baseline level of performance. Then we make one change at a time and compare the results after making that change to the baseline. If the change consistently results in an improvement in accuracy, the model with the change becomes the new best model. We iteratively test different hyperparameter combinations and other preprocessing techniques in an attempt to outperform the best model so far. This corresponds to the analyze stage in the data science analytical workflow process, which involves model selection and analyzing the results. Once we are satisfied with the best model's performance on the validation data, we give it new test data and repeat the process. The process for acquiring and preparing the data is illustrated in the figure below.

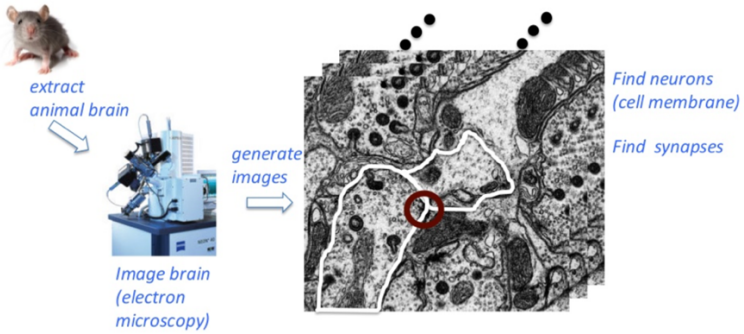

Figure 11: Acquire and Prepare Steps in Analytical Workflow

The analytical workflow is partially automated. CDense3M can run a parameter sweep for FFN and 3D filtering and labeling to obtain the results for each simulation run with a different parameter combination. All our image processing models are evaluated through visualizations using different algorithms. These image processing algorithms are run in Python, using the OpenCV, NumPy, and SciPy libraries. We aggregate relevant metrics and visualizations from our trial runs and save them in a Plotly Dash reporting dashboard. We present these findings to domain experts like Daniela and Matthias, and they act by figuring out the neuroscience principles that explain our findings.

Processing Environment Setup

Our exploratory data analysis and MVMP work was done locally in Jupyter notebooks and Python scripts. The 3D image analyses scripts were implemented locally in Python. We retrieved a cropped image volume of the *Nucleus accumbens* data from a National Center for Microscopy and Imaging Research (NCMIR) lab machine using IMOD and copied the images to our local and cloud machines. We ran CDense3M and FFN in AWS and Colab for most of this project. Both CDense3M and FFN include convolutional neural networks, and most deep learning models require a GPU for reasonable runtimes on large datasets.

Findings and Reporting

Findings

This section describes the findings related to our experiments to test hypotheses related to 3D image analyses and flood-filling networks. We will first discuss findings for 3D filtering and labeling using erosion models. Then we will report our findings and evaluation metrics from applying the FFN membrane model.

3D Image Analyses Findings

The preliminary analysis of the 3D labeling algorithms was to use the visualizations to evaluate the model's performance. The visualization was based on the logic of how much of the area of the new image (image 2 for example) was explained by last image (image 1).

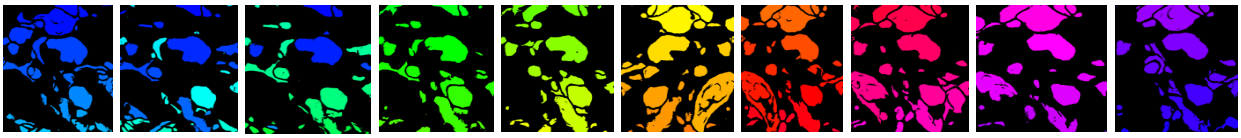

Figure 12: Labeling of 3D images based on the similar areas

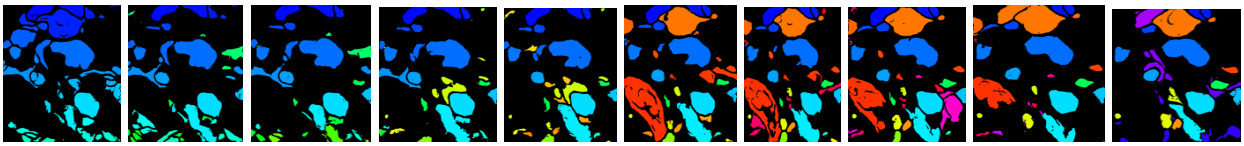

Figure 13: Labeling of 3D images based on the overlapping areas while there are grown vesicles

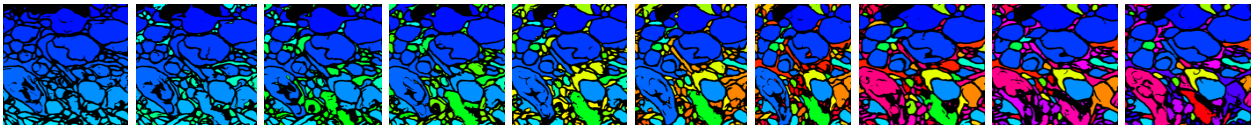

Figure 14: Labeling of 3D images based on overlapping areas

In the above figures, if the color of the areas of images stayed the same, that means those areas of images had identical labels. If the color of areas of images were changed, that means the label of the image was changed. Also, if the labels' areas overlapped over 80%, that means these labels are unique. If the areas' similarity and overlapping were under the threshold, it means that the labels had different identities and colors. The difference between Figures 13 and 14 is that in Figure 13, the vesicles were grown. But, in Figure 14, we considered all the areas inside the membrane. The preliminary analysis of evaluation was based on the consistency of labels that pass the 80% threshold, which Figure 14 performed the best in.

We could reach the conclusion that there were some algorithms that were assigned for 2D image analyses were not in 3D image analyses. The results of Figure 13 also indicated that the growth of vesicles caused the loss of some 3D presynaptic areas. The growth of vesicles was used to identify presynaptic regions used in 2D analyses. The reason was that the growth of vesicles was applied in each 3D object separately, while all these 3D objects in each image were actually related to 3D objects of corresponding images. This approach ended up losing some 3D objects

that had no vesicles, visualized in Figure 13. These 3D objects could be connected with 3D objects of other images that had met the threshold for vesicles. Therefore, some labels that could be true presynaptic sites are removed, which increased presynaptic recognition errors. In our overlapping Figure 14, there was still not very clear visualizations. That was the reason we applied 3D color labeling visualization to identify each label of 3D labeling more precisely. Our preliminary analyses indicated that each step of 3D modeling influences the next step to improve the accuracy. For example, the 3D color labeling visualization was influenced by the 3D labeling data to improve the model performance.

3D Color Labeling Visualization

The goal of the first category of this model was to provide more distinguishing colors for 3D labeling. The algorithm for this model creates random labels equal to the number of available labels in our 3D images, and then shuffle these random labels. Then, our images were saved in the zero-filled matrix with the same size as the image. This matrix is updated based on the random labels. For example, label 2 of the matrix was replaced with the second random label, such as 100 for instance. As a result, label 100 comes after label 1 instead of label 2. Therefore, based on the relocation of labels, we could assign different colors to each label.

The visualization of the 3D analysis algorithms performance is illustrated as follows.

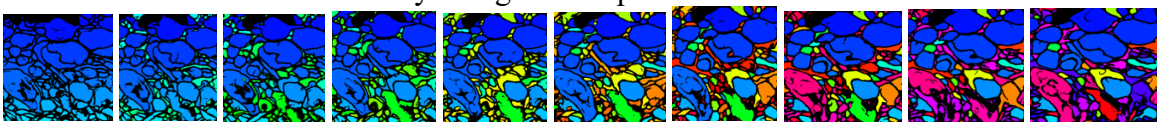

Figure 15: 3D labeling with the order of labels

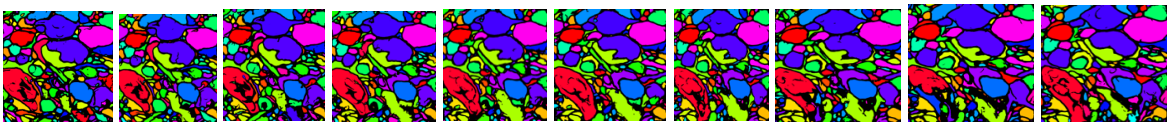

Figure 16: 3D Labeling based on random colors

In Figure 15, each label was in order, while in Figure 16, the labels were not in order. In Figure 16, the Python random library was used to generate the random labels.

3D Vesicle Filtering

As for the vesicles, we want the content of the vesicles within the 3D structure. The idea is that all of the vesicles are the individual parts of neurons, which forms a presynaptic site. In our visualizations above, each of the labels has a unique identifier, shown in the image with different colors. Each of those colors means 3D objects, and in 3D objects each color could have no vesicles or many vesicles. The threshold is our criteria to determine the presynaptic areas. If the number of vesicles is under the threshold, that means noise. If the number of vesicles exceed the threshold, that means presynaptic areas. This approach can be divided into two parts.

3D Vesicle Filtering – First Modeling Part

In this part, four models are created to improve the accuracy of presynaptic sites. The first model applies 3D vesicle filtering with the threshold of 5 vesicles on the 3D label objects. The basis of this algorithm is to count the number of vesicles for each label of each image individually. The preliminary visualization analysis of the model illustrates that there are some errors in our first model. These errors are either from removing some areas that have above 5 vesicles, or keeping some areas that have less than 5 vesicles. The significance about using this method is we can visualize the origins of the errors throughout the images. The reason for these errors is the dissimilar size of 3D objects throughout the images. This caused membranes that are the boundaries of 3D objects in some images to disappear, and thus 3D objects to be falsely connected to each other. In addition, the vesicles are overlapping on each other in some images that caused the algorithm to consider all the overlapping vesicles as one vesicle.

Our preliminary analyses indicate that the erosion technique is used to correct the above errors and generate more accurate presynaptic recognitions. The errors of 3D vesicle filtering in the first model influence the design of the second, third, and fourth models that erosion come into account with 3D vesicle filtering. In the erosion models, the kernel method is involved to erode vesicles and count the vesicles before and after the erosion. If both vesicles before and after the erosion are less than 5, that means the removal of that label from that image. The erosion models were applied on the vesicles and 3D objects. We also tuned the 3D vesicle filtering threshold to evaluate the best model. The tuned threshold involves 5 vesicles and 500 pixels. In the following Figures 17-20, the color blue illustrates the vesicles. The red/white regions illustrate those labels that are removed/stay after applying 3D vesicle filtering. The white labels represent the presynaptic areas and the red labels represent noise.

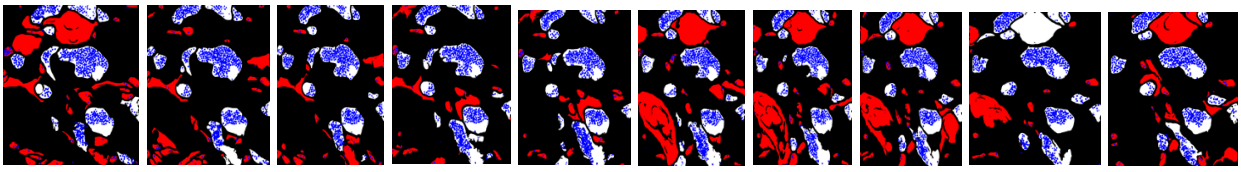

Figure 17: 3D vesicle filtering without erosion

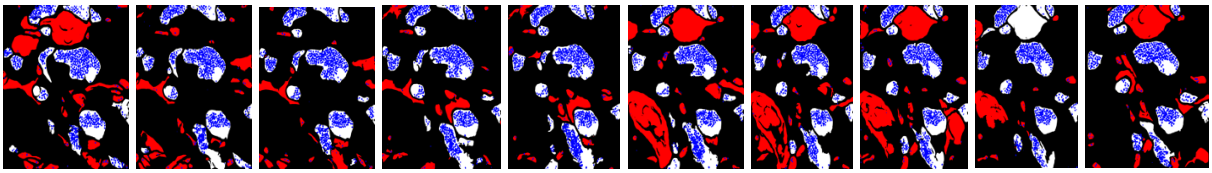

Figure 18: 3D vesicle filtering, applying the erosion of vesicles

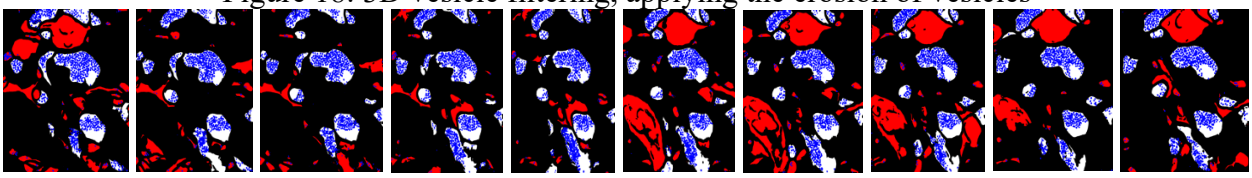

Figure 19: 3D vesicle filtering and applying the erosion of 3D objects (presynaptic)

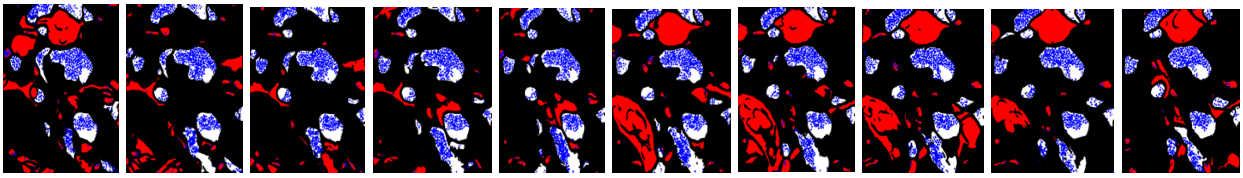

Figure 20: 3D vesicle filtering with two thresholds of 5 vesicles and 500 pixels and applying the erosion of 3D objects (presynaptic)

3D Vesicle Filtering – Second Modeling Part

The first modeling part influences the design of the second modeling part, which focuses on assigning the label for each presynaptic site that is identified in the first modeling part. That is the reason the labels of this part come from the remaining presynaptic sites left from first modeling part after applying filtering. With this, we need to match those remaining presynaptic sites with the unique identifiers in our 3D color map images. Each of these 3D objects has a unique ID. Then, we need to go through each of the remaining presynaptic areas after applying the 3D filtering to look for the labels that are occupied in 3D labeling. This would indicate our label group is under the same identifier. The performance of these models is shown in Figures 21-24. The preliminary visualization analyses of following figures demonstrate the labeling of the remaining presynaptic sites after vesicle filtering. The significance about the two modeling parts of vesicle filtering is the link between these two modeling parts. For example, in the first model, the presynaptic areas are defined, and in the second part, they received the labels. As a result, the presynaptic labels of Figures 17-20 are Figures 21-24.

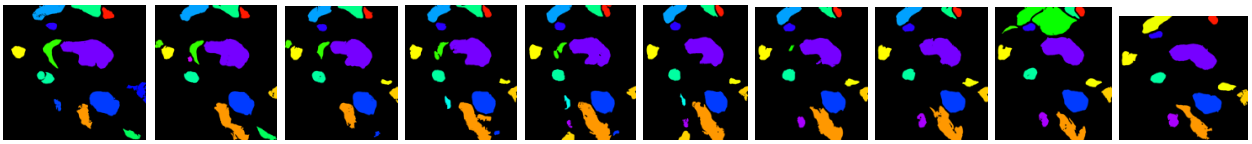

Figure 21: Determination of the labeling of 3D vesicle filtering (Figure 17 presynaptic label)

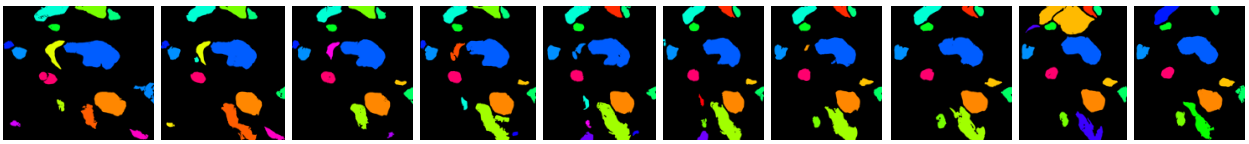

Figure 22: Determination of the labeling of 3D vesicle filtering, applying the vesicle erosion (Figure 18 presynaptic label)

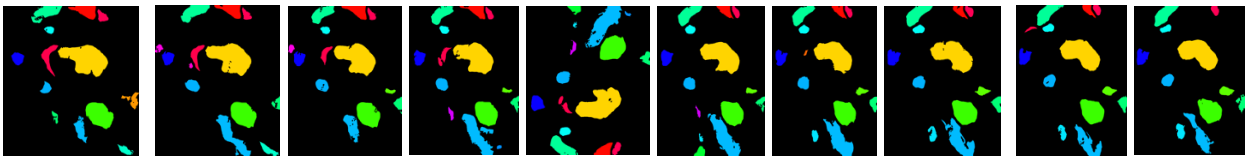

Figure 23: Determination of the labeling of 3D vesicle filtering, applying the 3D objects (presynaptic) erosion (Figure 19 presynaptic label)

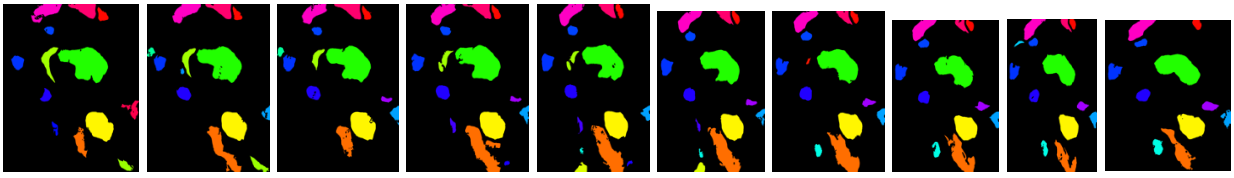

Figure 24: Determination of the labeling of 3D vesicle filtering with two thresholds that are 5 vesicles and 500 pixels, along with 3D objects (presynaptic) erosion. (Figure 20 presynaptic label)

In the following Figures 25 and 26, the performance of our second modeling part is evaluated. The preliminary analysis of this evaluation is based on the comparison between the presynaptic areas before applying the first model and after applying the first model. The first model is selected from modeling part 1, visualized in Figure 23.

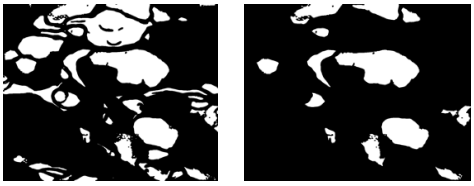

Figure 25: Binary images of presynaptic areas before filtering on the left side and after vesicle filtering of labeling with threshold 5 and 500 along with 3D objects erosion (best selected modeling part 1) on the right side

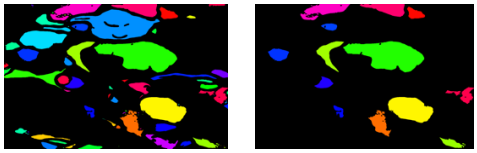

Figure 26: Labeling of binary images of presynaptic areas before filtering on the left side and after vesicle filtering of labeling with threshold 5 and 500 along with 3D objects erosion (best selected modeling part 1) on the right side

As for the findings of the first model, the criteria of 3D filtering visualized evaluation are based on keeping the presynaptic areas based on the vesicle functions that meet the threshold. In the visualizations above, Figure 20 is the best model, with the lowest error and the most accurate presynaptic site predictions. The significance of our finding is defining other criteria for the threshold of our 3D vesicle filtering that used to be 5 vesicles. The threshold of our 3D filtering for our best model is 5 vesicles and 500 pixels. The second modeling part demonstrates the labels of identified presynaptic sites in the first modeling part. To resolve the errors of Figure 17 and its label Figure 21, we applied erosion to separate the labels properly. As a result, the erosion models visualized in Figure 18 and its label Figure 22, Figure 19 and its label Figure 23, as well as Figure 20 and its label Figure 24. As for Figure 18 and its label Figure 22, it can be seen that the implementation of vesicle erosion increased the error. As for Figure 19 and its label Figure 23, it can be seen that the implementation of 3D objects (presynaptic) erosion caused the reduction of errors. There are more presynaptic sites that are recognized accurately. As for Figure 20 and its label Figure 24, it can be seen the implementation of applying 3D objects (presynaptic) erosion along with setting two thresholds for 3D vesicle filtering that are 5 vesicles and 500 pixels has the lowest error. This model demonstrates that those labels that have vesicles less/above 5 and 500 pixels should be removed/stay and therefore those 3D labels that stay are presynaptic sites. As a result, Figure 20 confirms our fourth hypothesis. The fourth model that is visualized in Figure 20 will give us the best results. Figure 20 and its label Figure 24 has the highest accuracy to define and recognize the 3D presynaptic areas. Figures 25 and 26 evaluated our second modeling part through the best-chosen model of modeling part 1, which was the combination of 3D objects erosion with the threshold tuning of 5 and 500. The presynaptic areas after filtering were less than before filtering, indicating the removal of irrelevant areas. The evaluation through these figures also indicated the improvement in the accuracy of presynaptic recognition and synapses segmentations.

The significance of all these image processing models is each new method is designed to improve on the previous step based on the errors, so we can achieve the goal of accurate presynaptic recognition. For example, our 2D image processing involves filtering mitochondria, membrane, and vesicles to achieve more accurate presynaptic areas and synapses. After the 2D vesicle filtering with the tuned threshold approach, we received errors in terms of losing some presynaptic areas. These errors were visualized by the stacked overlay figure. This 2D filtering influenced the design of 3D image processing to reduce the synapses' errors and retrieve more accurate presynaptic recognition. The 3D model is dissected into different algorithmic steps of 3D concatenation, 3D labeling, and finally 3D vesicle filtering. 3D labeling aimed to address the errors of the dissimilar label locations caused by the dissimilar size of 3D objects, through identifying the labels of 3D objects. Therefore, the 3D labeling was designed for more accurate 3D analyses of 3D vesicle filtering. There were still some errors after applying 3D vesicle filtering. These errors are derived from the erroneous connections or non-connections of some labels in images, or the overlapping of vesicles in one label that caused our model to not recognize the separation of these labels and vesicles. These errors influenced the erosion models we designed to improve the accuracy of presynaptic recognition. As a result, different erosion models were applied on vesicles, 3D objects, and the combination of vesicles and 3D objects. However, there were some errors on the erosion models that influenced our model building to tune the statistical thresholds to minimize the errors. After applying different thresholds, the erosion models on 3D objects with the tuned thresholds of 5 and 500 vesicles gave us the most accurate results.

However, we can still improve the accuracy of our best models so far to further improve the accuracy of 3D presynaptic recognition and synapses. In 3D analyses, the mathematical equation-based models that formulate the null hypotheses can calculate a more accurate statistical threshold based on the biological definition of neurons' functions. Our next approach is to bring mathematical and statistical tools to set more accurate thresholds for the connection and separation of labels and vesicles, therefore improving the accuracy.

Flood-Filling Networks Findings

Our hypothesis for FFN is that a custom FFN model trained on membrane segmentations is more generalizable at 3D cell segmentation than the provided FFN model trained on the standard FIB-25 *Drosophila* optic lobe raw electron microscopy (EM) data. The rationale is that an FFN model trained on membrane segmentations is more generic than an FFN model trained on raw EM data from fruit flies. Our custom FFN model is less likely to overfit on novel data because it is trained on 8-bit grayscale segmentation images of membranes produced by CDense3M's segmentation prediction script. The structure of grayscale membrane segmentations is consistent across all EM data. Thus, the FFN membrane model can handle any type of raw EM data that is segmented.

We had to train the provided FFN membrane model for another two million epochs in order to obtain a segmentation accuracy comparable to the FFN whitepaper. The provided 6 million epoch checkpoint produced poor segmentations, even when running FFN inference on the images the FFN was initially trained on. The dark spots indicate that FFN was struggling to find seed points to fill. Conflicting colors in a single cell indicate that there was a split error, while

cells that are different colors that should be together are considered merge errors. A sequential ID cell labeling scheme was used to keep track of merge and split errors across the *xyz* plane.

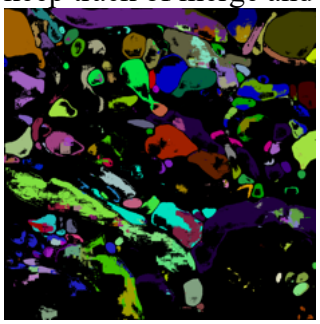

Figure 27: FFN inference result from using 6 million epoch checkpoint of FFN membrane model

Retraining the FFN for 2 million more epochs greatly improved the FFN dense segmentations on the original FFN training images. Matthias initially trained the model for 6 million epochs in UNI-EM, found at https://github.com/urakubo/UNI-EM. UNI-EM is a wrapper that provides an GUI for running FFN. But to eliminate a potential source of deviation, the FFN membrane model was retrained using the script from the original FFN source code. It took 88 hours of training on an AWS g4dn.xlarge EC2 instance to train the model 2 million more epochs. By 8 million epochs, the FFN achieves a much better dense segmentation on the training data, as seen below.

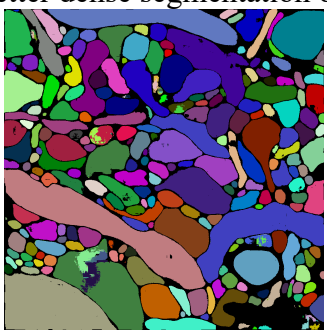

Figure 28: FFN inference result from using 8 million epoch checkpoint of FFN membrane model

We continued training the FFN membrane model for another two million epochs to reach ten million epochs. However, as seen in the line chart below, the performance of the FFN membrane model plateaus after eight million epochs. Since the training metrics for the FFN membrane model peak at and do not improve after eight million epochs, we chose the eight million epoch checkpoint as the final FFN membrane model to avoid overfitting on the training data.

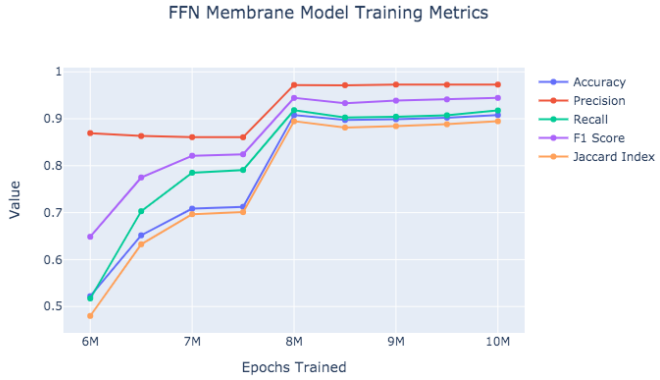

Figure 29: FFN Membrane Model Training Metrics

After confirming the viability of the FFN membrane model on the original training data, the next step was to apply the model to a different dataset to see how well it generalizes to new data. We used the *Nucleus accumbens* raw EM data since we already had ground truth labels for that dataset. The most important step to boost FFN's performance is to resize any input data to have the same *xy* dimensions as the training data. Our FFN model was trained on a 1024x1024x100 voxel image volume, so we took a 1024x1024x100 block of the raw EM data using IMOD.

The FFN membrane model expects membrane segmentations, not raw EM data, as input. Therefore, we first had to run the main CDeep3M pipeline to generate the membrane segmentations that would be then be used as the input to FFN inference. Fortunately, FFN inference is now fully integrated with CDeep3M in the form of the new CDense3M product, so CDeep3M and FFN segmentations can run consecutively. Running FFN inference with CDense3M is controlled by a boolean flag, since it only makes sense to run FFN when CDense3M is performing a membrane segmentation.

We use per-voxel accuracy, precision, recall, F1-score, and Jaccard index to evaluate FFN's performance on the original FFN training and raw EM test datasets. In a binary segmentation mask, a segmented voxel is positive and a non-segmented voxel is negative. We scaled the input images we used to match the *xy* dimensions of the training images for the two FFN models. This means the membrane model needs 1024x1024 images, and FIB-25 needs 250x250 images.

	Training Data with Membrane Model	Test Data with Membrane Model	Test Data with FIB-25 Model
Accuracy	0.9078	0.8237	0.6651
Precision	0.9720	0.9109	0.8737
Recall	0.9184	0.8262	0.7193
F1-score	0.9444	0.8665	0.7890
Jaccard index	0.8947	0.8041	0.6464

Table 4: FFN models segmentation evaluation metrics

Accuracy is the proportion of voxels that are classified correctly – segmented or not segmented.

Precision measures the fraction of true positives out of all the predicted segmented voxels. This is equivalent to the formula: Precision = TP / (TP + FP) = TP/Predicted Positive. We expect high precision out of the FFN model for both datasets because the watershed algorithm used by FFN will normally lead to an over-segmentation. This means the image is segmented into too many regions. FFN only starts flood-filling from seed points that are extremely positive - greater than 0.95 logits – and then only segments voxels that reach a high probability threshold. Thus, most of these segmented voxels predicted by FFN are true positives, leading to FFN's high precision.

Recall is equal to the true positive rate, or the proportion of true positives over all positives. We expect recall to be slightly lower than precision for FFN because FFN by default has a high threshold for accepting seed points. With fewer seed points and higher thresholds, positive voxels may be missed by FFN's segmentation. There are tunable parameters that lower these thresholds for FFN inference, but lowering these thresholds would also lower the precision.

To find a better balance between precision and recall, we calculate the F1-score, also known as the Dice coefficient, which is the harmonic mean of precision and recall.

Another value we measured is the Jaccard index, which is the proportion of overlap between the predicted and ground truth bounding boxes over their area of union.

As expected, our FFN membrane model is very accurate on its own training data, with most scores over 0.9 and a much higher precision than recall. The key finding, however, is that the FFN membrane model performs around 10% better than the FIB-25 FFN model on the raw EM test data in most metrics listed in Table 4. This supports our hypothesis that the FFN membrane model is more generalizable than the FIB-25 model on a variety of raw EM data.

Another finding is that the membrane model takes 16 times longer than the FIB-25 model to perform 3D segmentation, with an average duration of 64 minutes versus 4 minutes, respectively. One major caveat is that we scaled down our raw EM data image volume from 1024x1024x100 voxels when running the membrane model to 250x250x100 voxels when running the FIB-25 model. Resizing the images is required since FFN expects incoming data to have the same width and height as its training data. A 1024x1024x100 image stack has around 16.8 times more voxels than a 250x250x100 stack, which explains why the membrane model takes 16 times longer to finish. Two insights from this analysis are that the FFN membrane and FIB-25 models have a similar per-voxel runtime, and that FFN runs in linear time per voxel.

The last finding for FFN is that the quality of the incoming membrane segmentation is positively correlated with FFN segmentation accuracy and its other evaluation metrics. CDense3M's membrane model can sometimes generate membrane segmentations that include residual mitochondria. The most probable reason for this is that the membrane model is confusing the mitochondria with myelin sheaths, since the SBEM, TEM Membranes (50789) model was trained on EM data that contained myelin. A flawed membrane segmentation will spill over to the FFN, causing the FFN to mistakenly treat mitochondria as membranes and produce a less accurate 3D cell segmentation. This discrepancy partially explains how the segmentation evaluation metrics for the test data are on average 7% lower than the training data metrics.

Reportable Findings

The audience for these findings is the neuroscience research community. Neuroscientists like Daniela and Matthias, as well as the rest of the staff at NCMIR are especially interested in validating the findings related to erosion models for 3D labeling and the FFN membrane model.

Therefore, we determined that we would present a comparison between the accuracy CDense3M's and CDeep3M's synapse segmentations in reporting our findings. Our main hypothesis is that CDense3M is more accurate at segmenting synapses than CDeep3M. This includes individual hypotheses for each modeling component. The FFN hypothesis is that the FFN membrane model is more generalizable than the FIB-25 FFN model. The 3D image analyses visualizations prove the null hypothesis that erosion models improve the accuracy of the 3D labeling of the presynaptic sites.

We present reportable evaluation metrics and visualizations that support our findings to help the audience quantify and visualize the results. For each finding, we emphasize the potential business value in how each finding can positively contribute to the scientific process of automated brain segmentation to accelerate the study of neurodegenerative diseases. Determining the appropriate amount and level of information to present when reporting our findings is key to convincing a diverse audience that our CDense3M solution has satisfied its objective, which is improving the accuracy of existing synapse segmentations from CDeep3M.

Techniques and Tools Used to Communicate Results

We ran a controlled experiment to test our hypotheses that erosion models are better for 3D labeling and that the FFN membrane model is better than FIB-25. The 3D filtering and 3D presynaptic site recognition and visualization code used SciPy, NumPy, scikit-learn, and OpenCV. The FFN accuracy evaluation Python script used NumPy to quantify the results for each trial run. The FFN visualization Python script used NumPy, H5py, and OpenCV to produce an HDF5 stack file and separate flat PNG files of the FFN segmentation. The Fiji distribution of ImageJ is provided in the CDense3M Docker image to visualize the FFN segmentation HDF5 or PNG files in 3-3-2 RGB color and determine areas that need improvement.

We communicate our results in two primary mediums, a high-level reporting dashboard and more detailed log files generated by each stage in CDense3M. The techniques and tools we used to communicate our results were selected to increase the accessibility of CDense3M for new users, while communicating valuable details to more advanced users without overwhelming new users. The reporting dashboard summarizes the findings with charts and images, while the log files provide detailed statistics on the cells with the largest contribution to the error and the runtime for each step in the pipeline.

Visualization and other Reportable Products of Findings

Visualizations and other reportable products of our findings were added to the project as part of this step. Plotly's Dash framework was used to create a reporting dashboard to visualize the key metrics in our results in a single page.

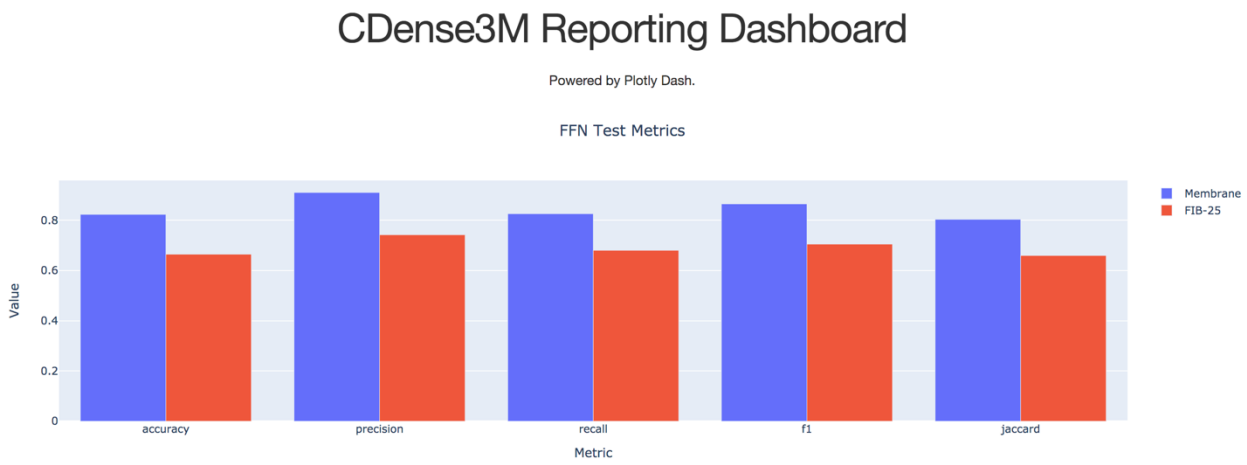

Figure 30: CDense3M Reporting Dashboard

For 3D labeling, Figures 12-26 are the visualizations for the findings. For FFN, Daniela requested a colored overlay of the FFN segmentations with the raw EM data. This is in addition to the HDF5 file and colorful segmentation PNGs produced by the CDense3M FFN inference script. We used the NumPy arange, vstack, meshgrid, reshape, and transpose functions to create a lookup table that colorized the grayscale FFN 3D cell segmentations with up to 65,536 colors. The opencv-python and scikit-image libraries were then used to overlay the FFN colored segmentations on top of the corresponding raw EM data images. An example is shown below.

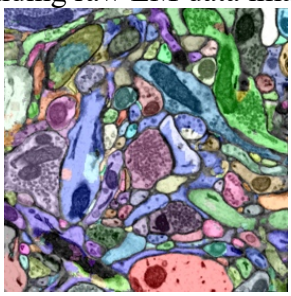

Figure 31: FFN colored segmentation overlay on top of raw EM data

Solution Architecture, Performance and Evaluation

Easy Deployment

CDense3M's portability makes it possible to deploy on various platform. AWS is the main deployment platform for CDense3M in production. Given our project's requirements of graphics processing units (GPUs) in deep learning for large-scale image segmentation, Amazon's g4dn.xlarge Elastic Compute Cloud (EC2) instance type was the lowest-cost platform available that met our needs, with an on-demand price of $0.526/hour.

For development, we use Google Colab for up to 12 hours of free access to GPUs. Colab even provides free tensor processing units (TPU), which are optimized for TensorFlow. We added TPU support for FFN inference in CDense3M, which was not in the originally upstream FFN source code. The Colab notebooks are in https://github.com/haberlmatt/cdeep3m-colab. However, Colab's GPU/TPU has usage limits, so critical CDense3M jobs should run in AWS.

We use an AWS CloudFormation template to automate the creation and provisioning of EC2 instances that are capable of running CDense3M. The CloudFormation stack is configured to launch an EC2 instance with the NVIDIA Deep Learning Amazon Machine Image (AMI). With CloudFormation, anyone with an AWS account can set up CDense3M in less than a minute.

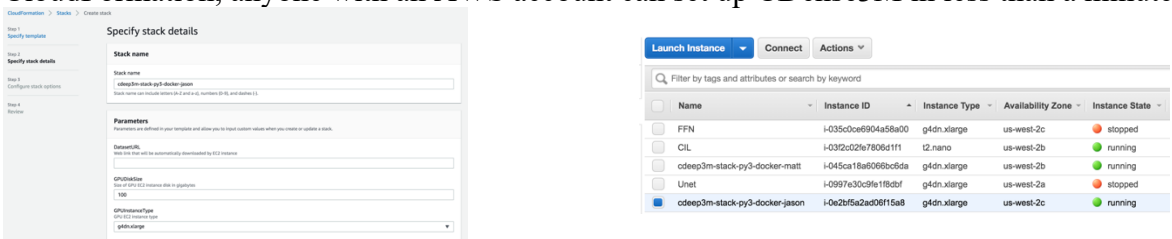

Figure 32: CDense3M CloudFormation Stack Creation Page and Resulting EC2 Instance Created

CDense3M itself can be run as a Docker container. It is available on Docker Hub at https://hub.docker.com/r/ncmir/cdeep3m. Docker containers isolate software applications from the host environment. By isolating processes from each other, similar to a sandbox, Docker containers avoid polluting the local machine with unwanted or possibly system-breaking changes. Docker also ensures developers are operating in a consistent environment. Deploying CDense3M with Docker also supports the multitenancy robustness requirement, allowing different users to run CDense3M on their own isolated sandbox in the same machine. We mount a Docker volume at the /data directory for persisting data generated and used by containers.

CloudFormation along with AMIs ensure users create machines that meet the specifications for running CDense3M. Docker completes the installation and configuration of CDense3M in a standardized way. AWS and Docker makes CDense3M a robust and scalable solution that satisfies the second and third case studies, as deploying CDense3M is consistent and repeatable for many machines, and users can run two CDense3M versions in isolation on the same machine.

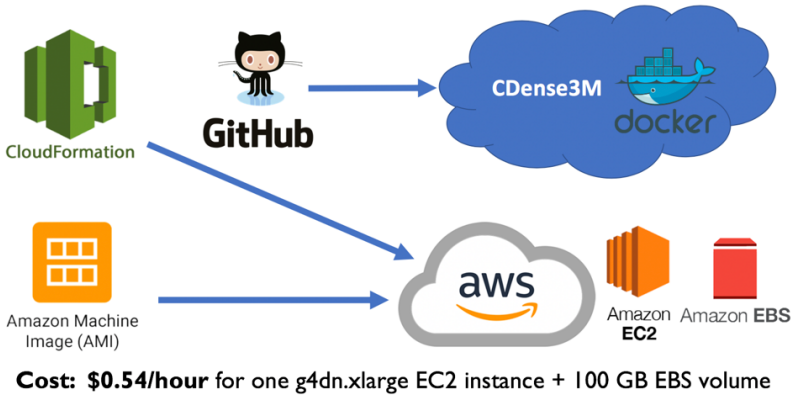

Figure 33: CDense3M Solution Architecture

Performance

Performance for CDense3M is primarily is measured by two metrics: voxel-wise segmentation quality and runtime. For FFN, we one-hot encode each cell ID into a separate 1024x1024x100 stack. This is equivalent to applying a binary segmentation mask for each cell ID, where 1 means the voxel is part of the cell, and 0 means the voxel is not part of the cell. We count the number of voxels that are true positives, true negatives, false positives, and false negatives for each cell ID using the predicted and ground truth segmentations. Then we take the average value for the five standard segmentation quality metrics among all cells. These segmentation quality metrics are the most important evaluation criteria for the performance of FFN, but speed is also very important. CDense3M must still remain usable even when processing a large volume of data.

CDense3M must also be robust when multiple users are running it on the same machine. In other words, CDense3M must support multitenancy. Two concurrently running CDense3M jobs that would in combination require more RAM than the machine can provide at one time should not cause the system to run out of memory. They also should not clobber each other's results regardless of the order in which the jobs finish. To ensure that CDense3M is fault-tolerant if the available RAM is not enough, we add retry logic to have it wait a certain amount of time for

RAM to become available before aborting. To guarantee the independence of the results, concurrent CDense3M jobs on the same machine must run in isolated environments.

Additionally, we envision multiple people working on different features of CDense3M at any given time. These branches of CDense3M, such as the one for flood-filling networks (FFN), require installing additional dependencies like TensorFlow 1.x. We want to be able to test different versions of CDense3M without interfering with other development efforts. Therefore, in order for CDense3M to be fully robust, different versions of CDense3M need to run in isolated sandboxes. Docker solves the multitenancy problem for keeping multiple CDense3M jobs running at the same time from clobbering each other, and keeping different version of CDense3M from clobbering each other. The accuracy of running two CDense3M jobs in Docker on the same machine was equal to running them on separate machines. As expected, it took twice as long to run the two CDense3M jobs in parallel, so CDense3M runs in linear time.

Scalability

Model scalability is how well CDense3M performs when given bigger or unseen datasets. If the available resources remain constant, CDense3M should run in linear time or faster as the input data size increases. If we provide CDense3M with additional resources like better or more GPUs, CDense3M should be able to process larger amounts of data just as quickly as before.

If we were to double the GPU RAM available by switching from a g4dn.xlarge to a g4dn.2xlarge EC2 instance, CDense3M should scale up and complete both jobs in the same amount of time as one job, or sooner. Similar logic applies when we double the size of the input images. For example, if we decide to use 1024x1024 pixel images instead of 750x750 pixels, each image will have almost double the number of pixels, so we would expect CDense3M to take twice as long.

When we doubled the GPU RAM to 32GB, CDense3M completed the two parallel tasks in 40 minutes, faster than the 1 hour it would take to complete a single task on the 16GB RAM GPU. FFN inference for the 1024x1024 images took 1.5 times longer compared to the 750x750 images. This means CDense3M's code is somewhat optimized to take advantage of additional GPU RAM, most likely via a combination of parallelism and increased batch sizes, so CDense3M is scalable. Benchmark testing was performed on AWS g4dn.xlarge or g4dn.2xlarge EC2 instances and the FFN model's scalability was evaluated using voxels segmented/minute.

Budget

We spent $238 in AWS cloud computing costs, which was paid for by NCMIR's operating budget. This cost is exclusively from testing FFN inference and integrating it into CDense3M, since the deep neural networks for these models require GPUs. About two-thirds of the total cost was from several attempts to retrain the FFN membrane model to obtain better performance on the FFN training images. The rest of the cost was performing hyperparameter tuning for FFN inference early on and testing its full integration with the original CDeep3M prediction script. It took approximately a week to train the original six million epoch FFN membrane model checkpoint for another four million epochs to reach ten million epochs.

We managed our budget by moving FFN inference tasks to Google Colab, starting in mid-April, to take advantage of the 12 hours of free GPU at a time. 12 hours is enough time for ten runs of FFN inference, so we significantly reduced our cloud costs by cutting our AWS usage in half. We also turned off our EC2 instances as soon as we completed our work for the day to eliminate unnecessary on-demand EC2 instance usage costs.

Conclusions

The developed solution of CDense3M, or CDeep3M extended with 3D image analyses and FFN, would align with the scientific process of improving the accuracy of the predicted segmentation of synapses. The purpose of CDense3M is to improve upon CDeep3M's performance on the dense segmentation of synaptic densities. We integrated our models into CDeep3M's automated workflow, and maintained its programmability, scalability, and ability to deploy to multiple platforms such as AWS and Colab. CDense3M will save scientists considerable amounts of time, since more accurate synapse segmentations are a step closer to the automated extraction of connectomes, which currently requires long hours of manual proofreading and editing.

We incorporate biological knowledge with all our models to improve the segmentation accuracy of synapses. We apply 3D image processing to prevent the loss of presynaptic areas during 2D vesicle filtering. The 2D vesicle filtering considers each 3D objects individually in each image, while the 3D processing links the 3D objects throughout the images. As a result, the 2D grown vesicles and the method of 2D vesicle filtering are not applicable in 3D image processing. Also, we are able to prove that each new step of image processing is relevant to the previous step. The significance of all these image processing models is each new method is designed to improve upon the errors from the previous step, which will increase the accuracy of presynaptic recognition. For example, our 2D image processing involves filtering mitochondria, membranes, and vesicles to achieve more accurate presynaptic areas and synapses. After the 2D vesicle filtering with the tuned threshold approach, we received errors in terms of losing some presynaptic areas, visualized in different figures. This 2D filtering influences the design of 3D image processing to reduce the synapses' errors and retrieve more accurate presynaptic recognition. The 3D model is dissected into the methods of 3D concatenation, 3D labeling and 3D vesicle filtering. These methods in 3D image processing are not applicable in 2D image processing. 3D labeling aims to address the errors of the dissimilar label locations caused by the dissimilar size of 3D objects, through identifying the labels of 3D objects. Therefore, the 3D labeling is designed for more accurate 3D analyses of 3D vesicle filtering. There are still some errors after applying 3D vesicle filtering. These errors are derived from the erroneous connections or non-connections of some labels in images, or the overlapping of vesicles in one label that caused our model to not recognize the separation of these labels and vesicles. These errors influence the erosion models we designed to improve the accuracy of presynaptic recognition. As a result, different erosion models are applied on vesicles, 3D objects, and the combination of vesicles and 3D objects. However, there are some errors on the erosion models that influence our model building to tune the statistical thresholds to minimize the errors. After applying different thresholds, the erosion models on 3D objects with the tuned thresholds of 5 and 500 vesicles give us the most accurate results.

Currently, automated synapse segmentations predictions on SBEM datasets are relatively inaccurate. CDense3M improves the voxel-wise classification accuracy of synapse segmentations by 10% compared to the baseline CDeep3M synapse segmentation. However, we can still improve the accuracy of our best models so far to further improve the accuracy of 3D presynaptic recognition and synapses. In 3D analyses, the mathematical equation-based models that formulate the null hypotheses can calculate a more accurate statistical threshold based on the biological definition of neurons' functions. Our next approach is to bring mathematical and statistical tools to set more accurate thresholds for the connection and separation of labels and vesicles, therefore improving the accuracy. For the FFN membrane model, we will optimize its performance on very large images by cropping the images to handle memory constraints. We will also develop data preprocessing methods to clean the input membrane segmentations to ensure cleaner cell-body segmentations. Implementing these improvements for both 3D analyses and FFN will lead to an even greater increase in the accuracy of synapse segmentations.

By providing accurate and automated segmentations of synapses, CDense3M will help researchers study neurodegenerative diseases such as Alzheimer's and Parkinson's disease more efficiently. Our hope is that CDense3M will help researchers make a breakthrough sooner in the search of better treatment options or even a cure for these neurodegenerative diseases.

References

1. Haberl, M.G., Churas, C., Tindall, L. et al. CDeep3M—Plug-and-Play cloud-based deep learning for image segmentation. Nat Methods 15, 677–680 (2018). https://doi.org/10.1038/s41592-018-0106-z

2. Januszewski, M., Kornfeld, J., Li, P.H. et al. High-precision automated reconstruction of neurons with flood-filling networks. Nat Methods 15, 605–610 (2018). https://doi.org/10.1038/s41592-018-0049-4

Appendices

DSE MAS Knowledge Applied to the Project

We made heavy use of NumPy, which was taught in DSE 200. The FFN 3D convolutional neural network was implemented in TensorFlow, which we learned in DSE 220 and DSE 230. The linear algebra we used in the image processing scripts for our exploratory data analysis and 3D image analyses was covered in DSE 210. We created our reporting dashboard in Plotly Dash and applied knowledge of effective and expressive visualization principles from DSE 241 to the project. Last but not least, the project management skills and data science process we learned in DSE 260A and DSE260B were essential to completing this capstone project.

Link to the Library Archive for Reproducibility

https://doi.org/10.6075/J0QJ7FT8