DSE-260
Alzheimer's Disease Microscopy Data Analysis
June 3, 2021

Team:

Shyam Banuprakash *sbanupra@eng.ucsd.edu*

Ellen Hein *ehein@eng.ucsd.edu*

Kim Nguyen *kin001@eng.ucsd.edu*

Justin Wahl *jwahl@eng.ucsd.edu*

Yuping Yu *y3yu@eng.ucsd.edu*

Advisors:

Matthew Madany *madany@ucsd.edu*

Mark Ellisman *mellisman@ucsd.edu*

Abstract

Advances in volume electron microscopy imaging have enabled the accumulation of large-scale, high-resolution biological data. In the neuroscience domain, the potential for these volumes to characterize the dense network of intertwining structures of the brain along with its subcellular components is within reach, but still limited in many ways by the size and complexity of the analysis. We have utilized the high-speed storage and GPU resources provided by the Pacific Research Platform (PRP) and CHASE-CI, managed by the Kubernetes engine, Nautilus, to implement and validate a 3D U-net for multi-class volumetric segmentation trained on a scarcely labeled mouse brain dataset. A deep learning internal zero-shot superresolution was evaluated on downsampled volumes to simulate its effect on pixel classification after x-y resolution boosting. In addition to traditional model performance metrics, a volume rendering tool was created to visualize voxel-level predictions to better understand problematic structures and subcellular features. These models and tools extend the neuroimaging reconstruction framework, NeuroKube.

I. INTRODUCTION

The human brain is an enormously dense network of over 86 billion neurons [1] with roughly 10^{15} connections [2]. Advances in electron microscopy technology have enabled the capture of high-resolution images of biological tissues that provide enough detail to resolve cellular ultrastructures previously unseen by optical light microscopy. Automated scanning and sectioning of tissue blocks has allowed for these images to be reconstructed into volumes. Until recently, these volumes were inevitably anisotropic due to early capture methods involving slicing tissue material, limiting z-axis resolutions to the thinness of the cuts. Focused Ion Beam Scanning Electron Microscopy (FIB-SEM) has helped mitigate this loss in z-axis resolution by ablating tissue material at much smaller increments, allowing for volumes $> 10^6$ μm^3 at <10 nm resolution in x, y, and z directions [3].

The data generated by these volume electron microscopy (VEM) methods can reach several terabytes in size, whereas a scan of the whole human brain would be truly astronomical. Speaking about a project involving mapping the mouse brain, our advisor and Director of the National Center for Microscopy and Imaging Research (NCMIR), Mark Ellisman, stated that "...this is roughly equivalent, in terms of the tyranny of scale, of making the mouse brain the size of the continent of North America...and being able to read, not just the numbers on the license plates of all of the cars...but down to the level of the bugs on the windshield" [4]. The potential that this technology brings to the study of connectomics, the mapping of neuronal connections, can help improve our understanding of human behavior, cognition, and neurodegenerative disease by shedding light on the dense network of cells and their intracellular components.

II. CHALLENGES, OPPORTUNITIES AND QUESTION FORMULATION

A. Challenges

Given the resulting size and underlying complexity of brain VEM data, the challenge of interpretation becomes as much a computer systems engineering problem as one in neuroscience. Traditionally, processing and storing data at this scale would require an institutional level of infrastructure, which often comes with limited access and finite resources built into on-site high-performance computer clusters. Today, these resources have been made much more available via various cloud platform services, and through cyberinfrastructure partnerships such as the Pacific Research Platform (PRP) and CHASE-CI, where shared cloud native deployments allow for dynamically scaled compute resources and unified access to valuable data [5].

However, despite the availability of computing resources, a limiting factor still remains in the expertise required to interpret the data. Without the assistance of chromogenic or fluorescent dyes, labeling EM data is a task that requires manual input from a subject matter expert trained to identify each and every structure of interest. The availability of such expertise is quite limited on its own, and considering the sheer magnitude of the task it would be nearly impossible to complete even a small volume in a reasonable amount of time without automated assistance.

With a limited amount of labeled data to work with but a need to automate segmentation across a vast tissue landscape subjected to variable preparation and handling, the dilemma then becomes how to train models such that they generalize well enough to handle unseen factors.

B. Initial State of the Project and Related Works

The state of this project as it was presented to us by our advisor Matthew Madany began with the question of whether a multi-class deep learning model would outperform several binary “one-vs-rest” models in a volume segmentation task. The initial assumption was that given the additional context provided by the other class labels, the multi-class model would be able to more readily identify underrepresented classes.

The code used to train the multi-class and binary voxel classifiers for this project is derived from work done by our advisor, Matthew Madany [6] . The *vclass* GitLab repository implements a 3D U-Net architecture [7] and consists of a training script that performs data augmentation and allows for setting model hyperparameters, such as the batch size and number of epochs, as well as an inference script that takes a trained model, reads raw images, and outputs voxel class predictions. Training and inference were designed to scale within the Nautilus infrastructure, however these features were outside the scope for this project.

In addition to the multi- vs. binary classification problem, Matthew also provided us with a number of publications with topics ranging from domain adaptation [8] to zero-shot learning [9], along with context to tie them to work being done at NCMIR. From these, we decided to pursue zero-shot superresolution [10] as it built upon our original pursuit.

C. Questions to be evaluated

1) Will a multiclass model outperform many binary “one-vs-rest” models especially in underrepresented structures?

2) How can we measure and visualize performance?

3) How can we improve segmentation in lower resolution captures?

4) How can we make these models and tools accessible to others?

III. TEAM ROLES AND RESPONSIBILITIES

Each team member utilized their own background and expertise throughout this project. Although team roles were adaptable and overlapped throughout the course of the project, the roles defined below were assigned with applicable responsibilities to be carried out.

- Justin Wahl - Project Manager
  Description: Responsible for managing the overall success of the project.
  Primary responsibilities:
    A. Facilitate actionable task creation,
    B. Oversee tasks to ensure completion on a timely basis,
    C. Periodically perform risk management on the project to assess any barriers to project completion,
- Kim Nguyen - Project Coordinator
  Description: Responsible for administrative tasks of the project to ensure specific tasks are running smoothly.
  Primary responsibilities:
    A. Communication liaison with our Project Advisor to facilitate weekly check-ins, team follow-up questions, and other logistical aspects to the project,

B. Create initial team reports to be review and finalized with the group,
C. Follow-up with team members to ensure the timely and accurate completion of project deliverables,

- Shyam Banuprakash - Web Design
  Description: Create and design our web application product.
  Primary responsibilities:
  A. Create and design foundational scripts for our web application product
- Ellen Hein & Yuping Yu - Data Scientist
  Description: Apply expertise and interdisciplinary knowledge to tackle data rich problems.
  Primary responsibilities:
  A. Implement and evaluate machine learning models,
  B. Deep learning subject matter expert.

IV. DATA SOURCES

Data used for this project was available to us within the ncmir-mm workspace in Nautilus. An advantage of the infrastructure provided by Nautilus is that data is readily available to users. User Pods mount to the ceph distributed file system, bringing tools and resources directly to the data versus users fetching it for themselves.

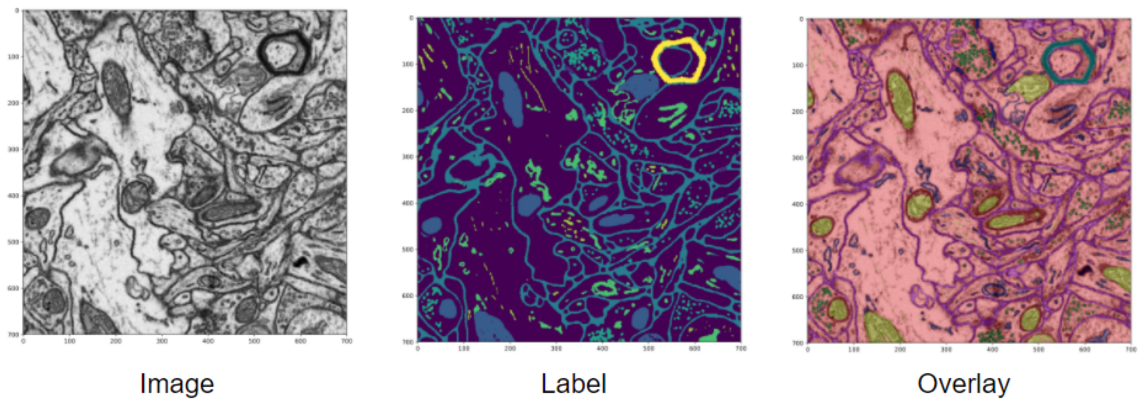
The dataset used in this project is from a single mouse cortex FIB-SEM volume consisting of 32 raw 700x700 TIF images. A labeled set of PNG images was also provided, with each pixel representing 1 of 8 cellular structures identified by a subject matter expert.

Table 1. Data Set

Description	Source	Type	Data Size
Mouse cortex FIB-SEM	Nautilus (ncmir-mm namespace)	TIF, PNG	700x700x32 (15.7Mb raw, 1.4Mb labeled)

V. DATA EXPLORATION AND FINDINGS

Initial data exploration consisted of viewing raw and label overlaid images in the x-y and y-z directions (Fig. 1). Visually, the resolution along all axes appeared to be equal, confirming the isotropicity provided by the FIB-SEM method.



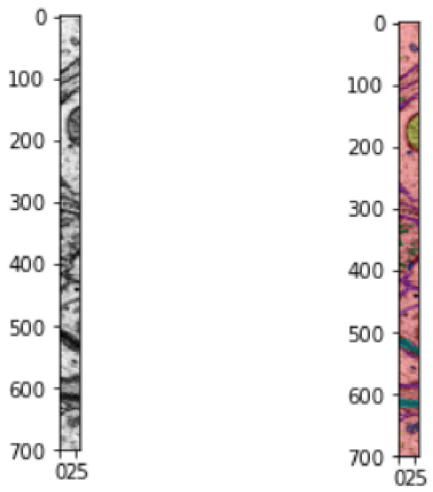


Fig. 1 A single FIB-SEM image along the x-y plane (top) and the z-stack along the y-axis (bottom)

We were uncertain of any preprocessing that may have been applied to the images, so the distribution of pixel intensities across all images in the volume were plotted. The resulting histogram shows two distinct peaks, suggesting that contrast may have been adjusted (Fig. 2). Considering that the images were already high in contrast no further preprocessing steps were applied. Unfortunately, metadata associated with this dataset was not available to confirm the source and methods of preparation, however no aberrations were observed that might affect our models.

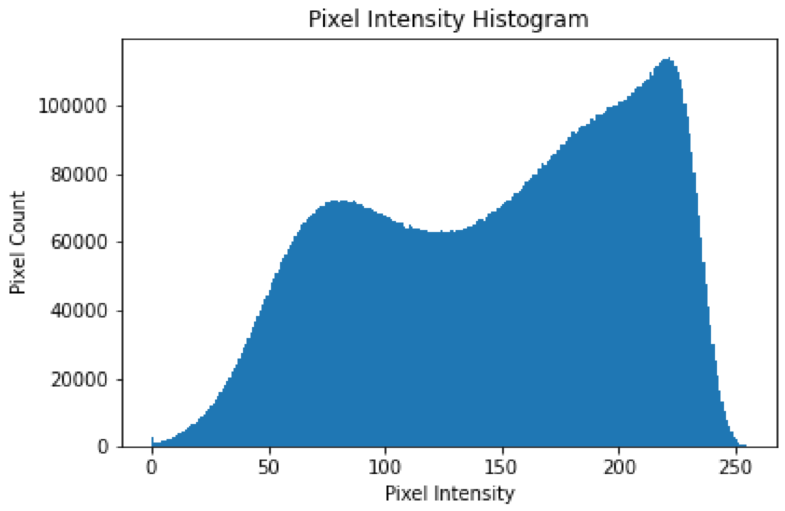
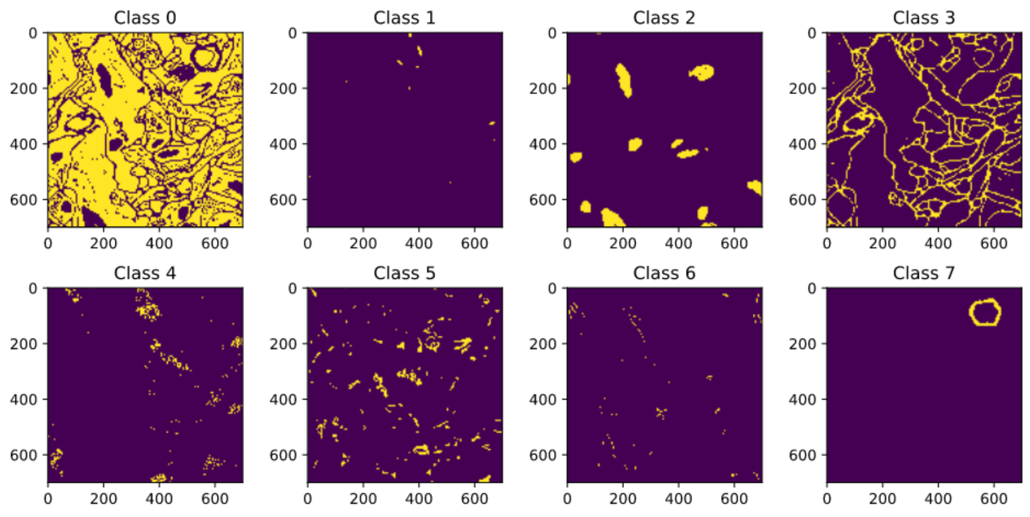


Fig. 2 Grayscale histogram for all images in the volume

To gain an understanding of the class distribution, an RGB labeled image was separated into binary images representing each class. It became immediately apparent that there was a class imbalance between larger cellular structures, such as cell membrane and mitochondria, and smaller structures such as microtubules and synapses. This was further confirmed by plotting the class distribution across all images in the volume (Fig. 3).



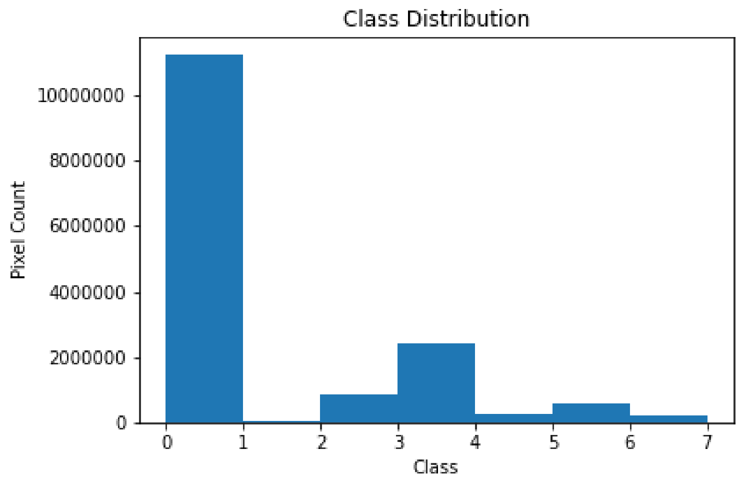


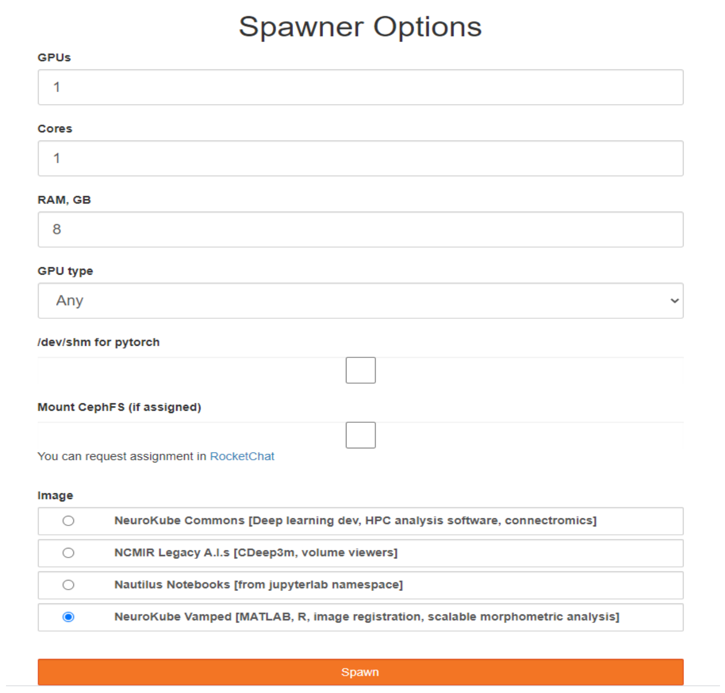
Fig. 3 Ground truth labeled images by class (top) and class distributions (bottom)
0-cytosol (background), 1-synapses, 2-mitochondria, 3-membrane, 4-neurotransmitter vesicles, 5-endoplasmic reticulum, 6-microtubules, 7-myelin

VI. SETTING UP OUR ENVIRONMENT

The cloud infrastructure used throughout this project is part of the PRP's Nautilus Kubernetes hypercluster which hosts over 700 CPU cores and over 500 GPUs distributed across 30 universities and national laboratories [11]. The cluster is designed to provide scalable big data services, using Kubernetes to manage containerized applications and Rook to bridge them to Ceph data storage.

The environment for this project was already well defined through use of the NeuroKube Gateway, which is a portal to access Nautilus resources. The gateway provides a user form to define compute resources such as the number of GPUs and cores, GPU type, amount of VRAM, an option to mount to shared cephs file storage, and Docker images containing neuroimaging models and toolkits (Fig. 4). Although resources were seemingly plentiful, we limited ourselves to the spawner options listed in Table 2, and at any given time used a total of 10 CPU cores and 5 GPUs across all users.

Submitting a request spawns a Kubernetes Pod hosting a JupyterLab environment with block storage allocated to the registered user which persists between sessions even after resources are released. The majority of the work for this project was performed in Jupyter Notebooks within these user JupyterLab spaces. Jupyter Notebooks provide a portable and reproducible document containing code, markdown text, and visualizations. All figures and interactive visualizations supporting this project were produced and rendered using Jupyter Notebooks.



Spawner Options

GPUs: 1

Cores: 1

RAM, GB: 8

GPU type: Any

/dev/shm for pytorch: ☐

Mount CephFS (if assigned): ☐

You can request assignment in RocketChat

Image:
- ○ NeuroKube Commons [Deep learning dev, HPC analysis software, connectromics]
- ○ NCMIR Legacy A.I.s [CDeep3m, volume viewers]
- ○ Nautilus Notebooks [from jupyterlab namespace]
- ● NeuroKube Vamped [MATLAB, R, image registration, scalable morphometric analysis]

Fig. 4 NeuroKube Gateway Spawner

Table 2. Project Spawner Options

GPUs	1
Cores	2
RAM	12
GPU Type	1080Ti or 2080Ti
/dev/shm for pytorch	Yes
Mount CephFS	Yes
Image	NeuroKube Commons

The NeuroKube Commons image provided us with the Python libraries required for this project. As the project progressed, additional libraries were added to the image, namely ipyvolume and pyvista for volume rendering, and streamlit for simple web development. Our code was maintained either in the Nautilus GitLab or a shared folder within the ncmir-mm namespace.

Using resources and code provided to us through the NeuroKube Gateway and GitLab was hugely beneficial in getting our environments up and running quickly, however, at times it did not provide the flexibility we would have liked to debug and move certain aspects of our project forward without adequate administrator expertise and access within the namespace, or a deep understanding of the repository code. Although this goes for any continuation project, it did require some time to onboard and become familiar with the tools.

VII. DATA PREPARATION

As determined in the data exploration findings, the raw images in our dataset did not require any preprocessing. However, to prepare the labeled images for each binary model training the class assignments for each pixel had to be converted to either a 1 to represent the class of interest, or 0 to represent another class. For the multiclass model the labeled images were used as is. The z-stacks were then split into training and test sets, consisting of 25 and 7 images, respectively.

Both *vclass* and the ZSSR scripts utilize data augmentation prior to training in order to generate additional training samples from the original images. Rotation, scaling, elastic deformation, contrast adjustment, the addition of noise, blurring, and sharpening, are randomly applied to the raw and labeled images to inject variation into the data set (Fig. 5). Considering the challenge in producing labeled VEM data, augmentation is necessary to enrich these data sets and make do with limited training examples. Also, due to intra- and inter-sample variations that may be introduced during tissue block preparation or during the course of a scan, augmentation can help the model generalize enough to handle unseen aberrations that were not present in the training set.

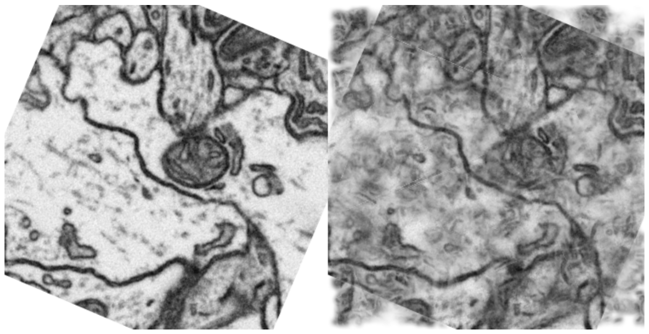


Fig. 5 Example of data augmentation applied during training script from *vclass*

VIII. MODEL DESCRIPTIONS

A. Multiclass and binary models

The architecture used in our models is a 3D U-Net described in “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation” [7] implemented in PyTorch [12]. The vclass repository also contained an implementation of a residual 3D U-Net, which we did not end up using.

The 3D U-Net is based on the 2D U-Net architecture, which is a fully convolutional neural network designed for 2D image semantic segmentation tasks where each pixel is assigned a class label. The architecture consists of a path of contracting encoding steps that capture context, followed by a path of symmetric decoding steps that enable precise localization. The contracting path extracts feature maps through performing repeated convolutions, each followed by rectified linear unit (ReLU) and max pooling operations. This downsampling reduces spatial information while increasing feature information. During the up-convolutional path, the extracted feature maps are concatenated to the high-resolution upsampled layers via skip connections. In the last layer, a final convolution reduces the number of channels to the number of prediction labels.

The 3D U-Net adapts the same conventions as its 2D counterpart, where the major difference is that the inputs, and the convolution, max pooling, and up-convolution processes are in 3D (Fig. 6). For example, in the original 2D architecture along the encoding path each layer contains two 3 x 3 convolutions, followed by a ReLU activation, followed by 2 x 2 max pooling with strides of 2 in both dimensions. In the 3D implementation, each layer contains two 3 x 3 x 3 convolutions, followed by ReLU activation, followed by 2 x 2 x 2 max pooling with strides of two in all three dimensions.

Due to the repetitive nature of cellular structures in biological tissues, especially at high resolution, this network is able to train on a very limited number of samples. Again, considering the challenge with producing labeled VEM images, this network architecture works well for the task by generalizing with only a sparsely annotated data set.

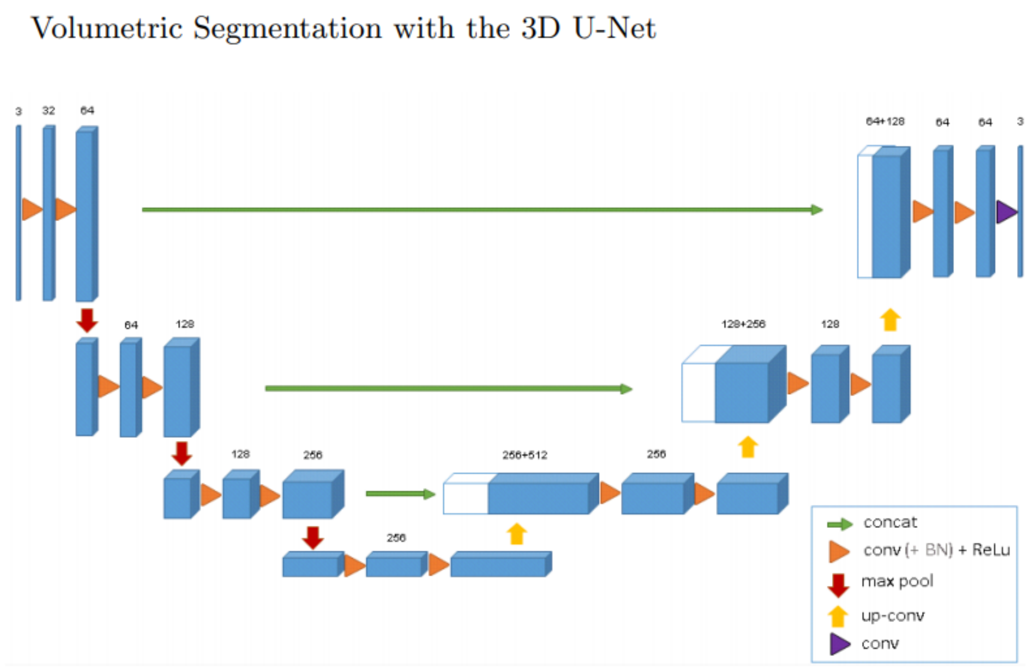


Fig. 6 3D U-Net architecture
[Adapted from “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation” by Ö. Çiçek, et al., 2016, arXiv:1606.06650.]

For each binary class model, a single U-Net was trained using raw and labeled images, where the pixel labels were either 1 or 0 representing membership in the class of interest. The outputs were a single stack of images with pixel grayscale values as predictions. The multiclass model was similarly trained as a single U-Net, but with the exception that each pixel in the labeled data represented one of eight classes. The outputs were several stacks of images, each representing predictions for a single class (Fig. 7).

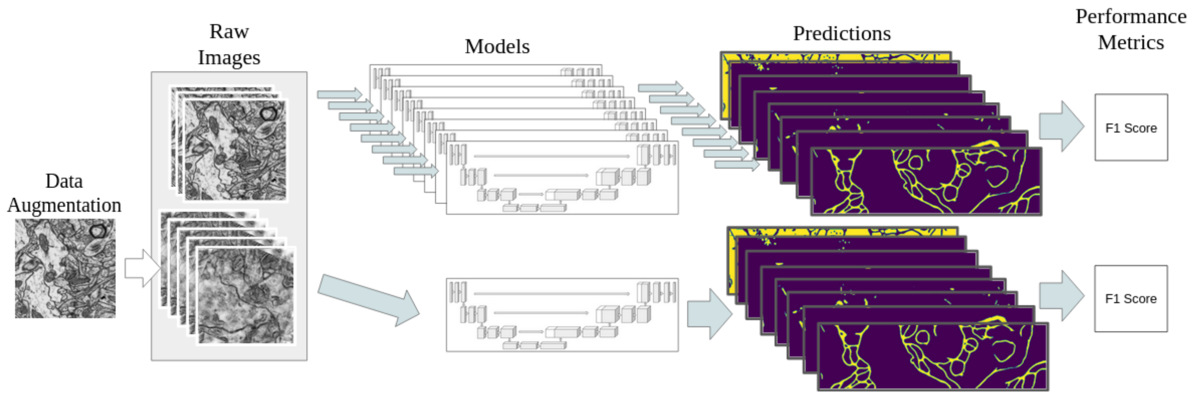
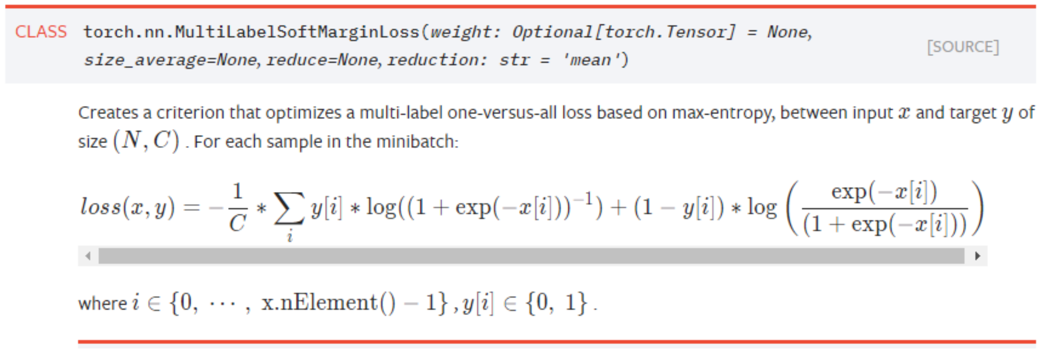


Fig. 7 Multiclass and binary model workflow

Of the hyperparameters that we were able to adjust using arguments passed to the *vclass* training script, batch size and the number of epochs were changed slightly from the default values. Due to our limited data set, the batch size was set to 1. We initially trained using the default number of epochs of 12000, but observed a checkerboard effect in the output images, which we were advised could be resolved by increasing the number of epochs to 20000. The optimizer and learning rates were hard coded, and therefore were not altered (Table 3).

Table 3. Final Unet Parameters

Parameter	Value
Optimizer	Adam
Learning rate	0.0004
Loss function	Multi Label Soft Margin Loss
Epochs	20000
Batch size	1



CLASS `torch.nn.MultiLabelSoftMarginLoss(weight: Optional[torch.Tensor] = None, size_average=None, reduce=None, reduction: str = 'mean')` [SOURCE]

Creates a criterion that optimizes a multi-label one-versus-all loss based on max-entropy, between input x and target y of size (N, C). For each sample in the minibatch:

$$loss(x, y) = -\frac{1}{C} * \sum_i y[i] * \log((1 + \exp(-x[i]))^{-1}) + (1 - y[i]) * \log\left(\frac{\exp(-x[i])}{(1 + \exp(-x[i]))}\right)$$

where $i \in \{0, \cdots, \text{x.nElement}() - 1\}$, $y[i] \in \{0, 1\}$.

Fig. 8 Multi-label soft margin loss

B. Zero-shot superresolution model

There are many neural network based superresolution methods, however the majority of them are supervised and are thus bound to the specific domain and conditions of the data they are trained with. In general, these methods are convolutional neural networks (CNN) trained using high-quality images degraded with a downscaling kernel to create a set of low-resolution (LR) and high-resolution (HR) pairs. These networks rely on small pieces of information that recur in an image, therefore one limitation is that when a patch does not exist in the target LR image, the trained model can't refer to the HR-LR relations resulting in reduced resolution boosting performance.

The superresolution method we are using is called zero-shot super resolution (ZSSR) [10] which does not require external training images as it uses internal learning on the target image itself. To achieve this, the target image is augmented and scaled down to create many LR versions of itself ("LR sons") which are then used as training examples for a CNN using its originally scaled pair from which it was derived as the HR test example ("HR fathers") (Fig. 9). In this way, the model learns the relationship between the LR-HR pair, which can then be applied to the target image to boost its resolution.

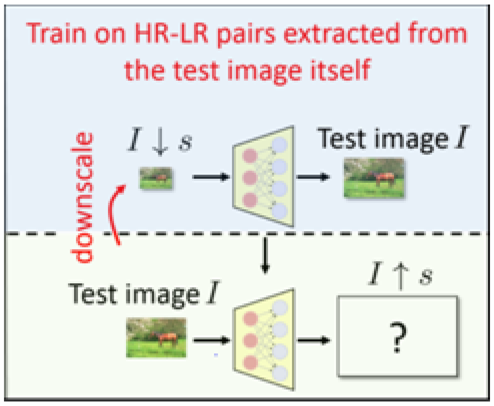


Fig. 9 Image specific CNN
[Adapted from "'Zero-Shot' Super-Resolution using Deep Internal Learning" by A. Shocher, et al., arXiv:1712.06087]

Again, considering differences in sample preparation that may produce intra- or inter-sample variations that might fool other superresolution methods, applying a method that adapts to each image in a volume could help normalize the resolution across the entire block. This ZSSR method also works especially well on images that have repeated structures, such as those seen in biological tissue samples (Fig. 10), which could help tease out fine processes and structures that older microscopy methods fall short in capturing, or that are still too small to resolve even with modern VEM technology.

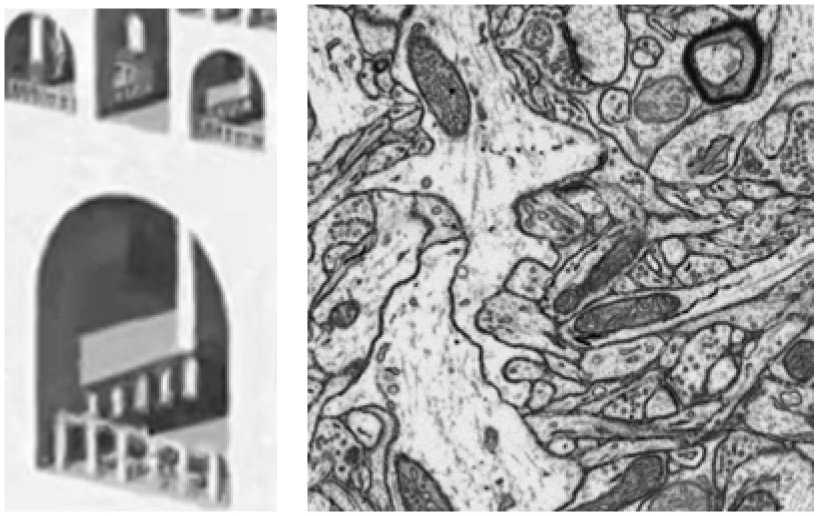


Fig. 10 Example of repeated structures from the "Zero-shot" paper (left) and our data set (right)

IX. ANALYTICAL METHODS

A. Comparing multiclass vs. binary segmentation

Each U-Net output consists of volume image stacks for each class, with grayscale values representing the prediction for each pixel. Due to the outputs from the models trained using *vclass* not yielding a binary class prediction, these grayscale values needed to be converted to either a 0 or 1 using thresholds that yielded the best F1-score from the training data. These calls were then used to calculate F1-scores as a measure for each model's prediction accuracy for each class.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

TP = number of true positives
FP = number of false positives
FN = number of false negatives

F1-score was selected as the performance metric due to the class imbalance observed during the exploratory analysis phase. Accuracy tends to be used when the true positives and true negatives are important in measuring a model's performance, however in our case, this metric would favor classes that are overrepresented. Our goal in comparing the multiclass model to several binary class models was to see an improvement in the underrepresented classes. F1-score puts more emphasis on the false positives and false negatives, highlighting each model's inability to classify certain classes. The F1-scores for each binary model were then compared to the F1-scores of each output class from the multiclass model.

B. Applying ZSSR to improve segmentation

We used two approaches to observe the effect ZSSR has on segmentation. The first was to apply superresolution (SR) boosting on the original HR images to evaluate its effect on the performance of our original multiclass model. The second approach was to evaluate the effect SR boosting has on the original multiclass model's performance in segmenting downsampled LR images (created using scikit-image's *rescale* function) compared to boosted LR images. New models were also trained using LR images and boosted LR images, and evaluated on the original HR images. Again, F1-score was the metric used to evaluate model performance on each class.

X. FINDINGS AND REPORTING

A. Multiclass vs. binary segmentation performance

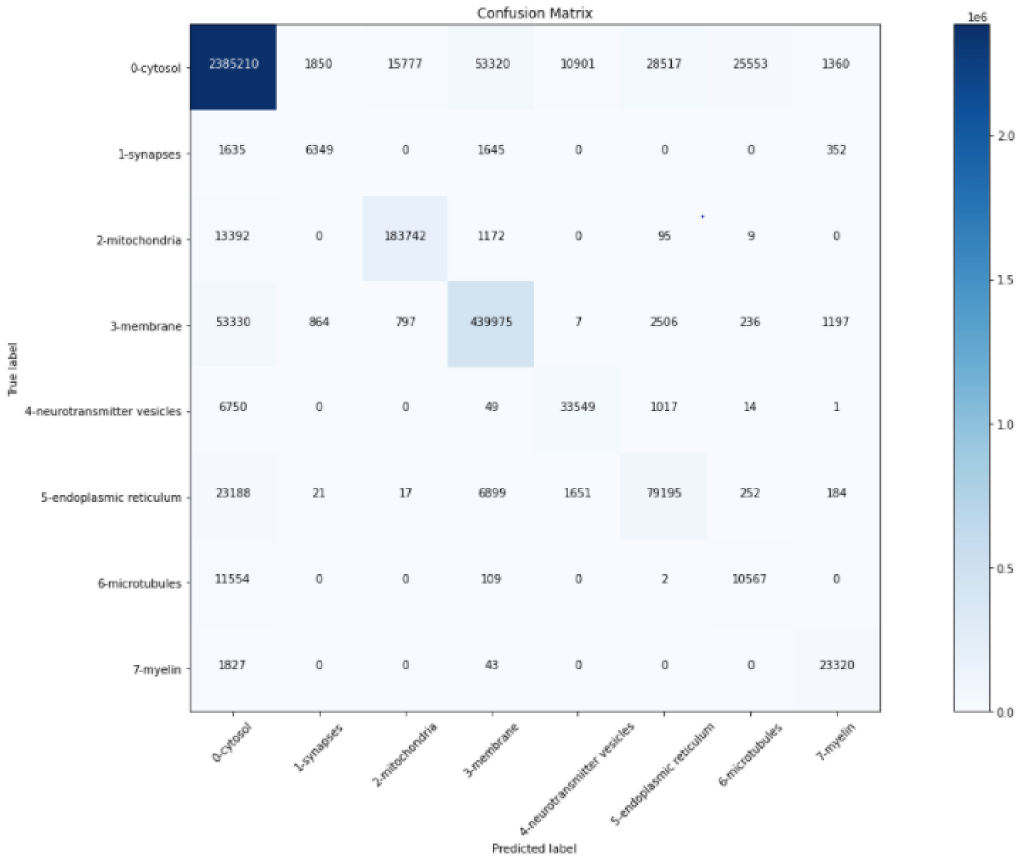
The performance of each binary model individually trained to predict pixels of a single class versus the predictions output by the multiclass model against the test data set are listed in Table 4. The value of a F1-score falls between 0 and 1, where a value closer to 1 represents better performance in terms of precision and recall. From the results we see that for the classes with higher prevalence, such as the cytosol (background), mitochondria, and membrane, there was not much improvement using the multiclass model versus the binary model, which was expected. However, there does seem to be some improvement in performance with the multiclass model in neurotransmitter vesicles, endoplasmic reticulum, microtubules, and myelin. Synapses, unfortunately, did not see an improvement, however, as the least represented class there may not have been enough instances for the model to learn from. Microtubules were slightly more prevalent, however neither the binary or multiclass model were able to correctly identify them.

Table 4. Binary "one-vs-rest" models vs. multiclass models

Class	**% of Pixels in Total**	**F1-Score Binary**	**F1-Score Multiclass**
0-cytosol	73.50%	0.954	0.955
1-synapses	0.29%	0.667	0.666
2-mitochondria	5.78%	0.921	0.922
3-membrane	14.50%	0.879	0.878
4-neurotransmitter vesicles	1.21%	0.752	0.766
5-endoplasmic reticulum	3.25%	0.689	0.711
6-microtubules	0.65%	0.346	0.359
7-myelin	0.73%	0.865	0.904

Fig. 11 is a confusion matrix for the multiclass predictions which gives some insight into which classes the model is misclassifying and which class is being called instead. In a confusion matrix, the diagonal is the true positives and true negatives, the squares below the diagonal are the false positives, the squares above the diagonal are the false negatives, and the numbers are the number of pixels.

Without considering the background class 0, the cytosol, some classes stand out as points of confusion for the model. Membranes, as the most prevalent class after the background, are the main source of confusion. This is not surprising as cell membranes will be present within the context of every organelle structure. Overall, the multiclass model tends towards classifying a negative pixel positively (false positives), rather than classifying a positive pixel negatively. Unfortunately, due to the binary models being one-vs-rest, a confusion matrix would not give any indication of which class the model is confusing for the class of interest, or vice versa.



	0-cytosol	1-synapses	2-mitochondria	3-membrane	4-neurotransmitter vesicles	5-endoplasmic reticulum	6-microtubules	7-myelin
0-cytosol	2385210	1850	15777	53320	10901	28517	25553	1360
1-synapses	1635	6349	0	1645	0	0	0	352
2-mitochondria	13392	0	183742	1172	0	95	9	0
3-membrane	53330	864	797	439975	7	2506	236	1197
4-neurotransmitter vesicles	6750	0	0	49	33549	1017	14	1
5-endoplasmic reticulum	23188	21	17	6899	1651	79195	252	184
6-microtubules	11554	0	0	109	0	2	10567	0
7-myelin	1827	0	0	43	0	0	0	23320

Figure 11: Confusion Matrix for multi-label image segmentation classification.

The U-Net architecture implemented in *vclass* does not use an activation function in the final layer, hence the need to apply thresholds on the grayscale pixel values for each class to generate a binary prediction. Plotting where these thresholds give the best F1-score for each class and comparing the peaks between the multiclass and binary models gives some insight into the nature of the false positives and false negatives generated by each (Fig. 12). When the threshold increases, if the F1-score increases as well this is an indication that false positives calls are being filtered out. As the threshold increases and the F1-score decreases, this is an indication that the number of false negatives is increasing. In synapses, it appears that there are fewer false positives generated by the multiclass model compared to the binary model for that class. Therefore, despite having similar F1-scores we can see that the multiclass model does a better job by not misclassifying other classes as synapses.

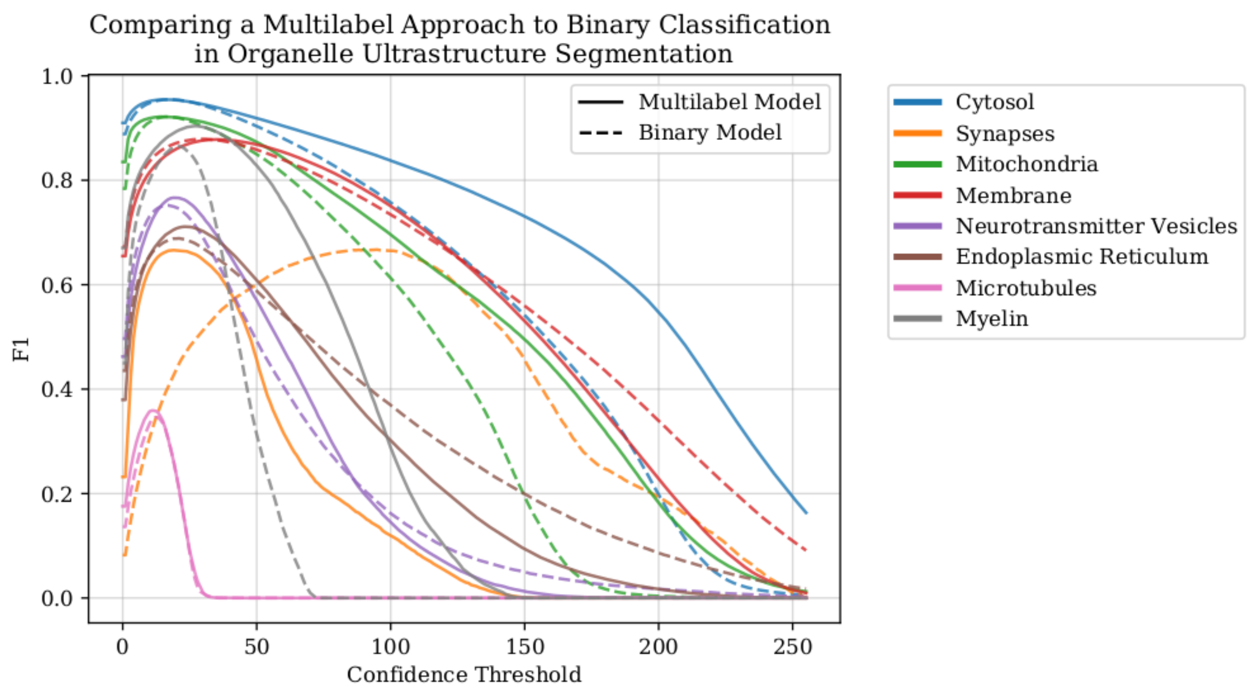


Fig. 12: F1-Scores across confidence thresholds comparing multi-label vs binary classification across different organelle classes.

Pixel intensity can be interpreted as a measure of confidence a model has in its prediction. For class 2 (mitochondria) we did not observe an increase in F1-score between the multiclass model, however when overlaid on the raw image, the multiclass model appears to be more confident in its predictions. The same can be observed for class 7 (myelin) in addition to the improvement to the F1-score.

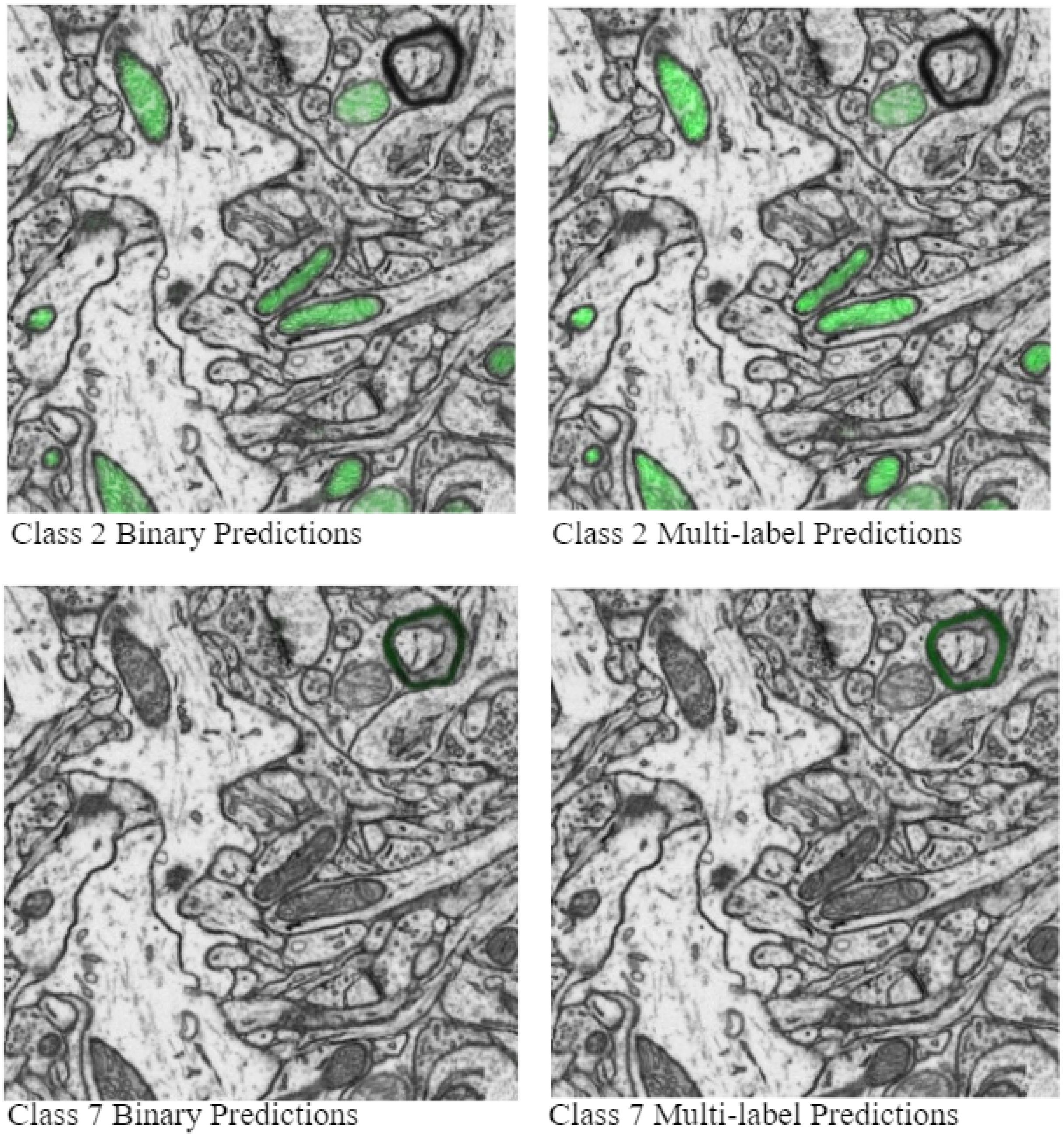

Class 2 Binary Predictions
Class 2 Multi-label Predictions
Class 7 Binary Predictions
Class 7 Multi-label Predictions

Fig. 13: Binary vs. Multi-class predicted pixel intensities for Class 2 and Class 7

To gain more perspective on how each structure was being classified in 3D we produced volumetric renderings of the true positive, false positive, and false negative voxels using ipyvolume (Fig. 14). Viewing the data in this way allowed us to see that the majority of the misclassification occurs on the edges of the objects. These volume renders were created in a Jupyter Notebook that allows for interactively adjusting the opacity of each pixel result type, and singling out certain classes for a more granular view of the prediction performance. In general, these renders proved to be particularly useful in identifying structures that were difficult to classify and could be adapted for use with any volume segmentation model.

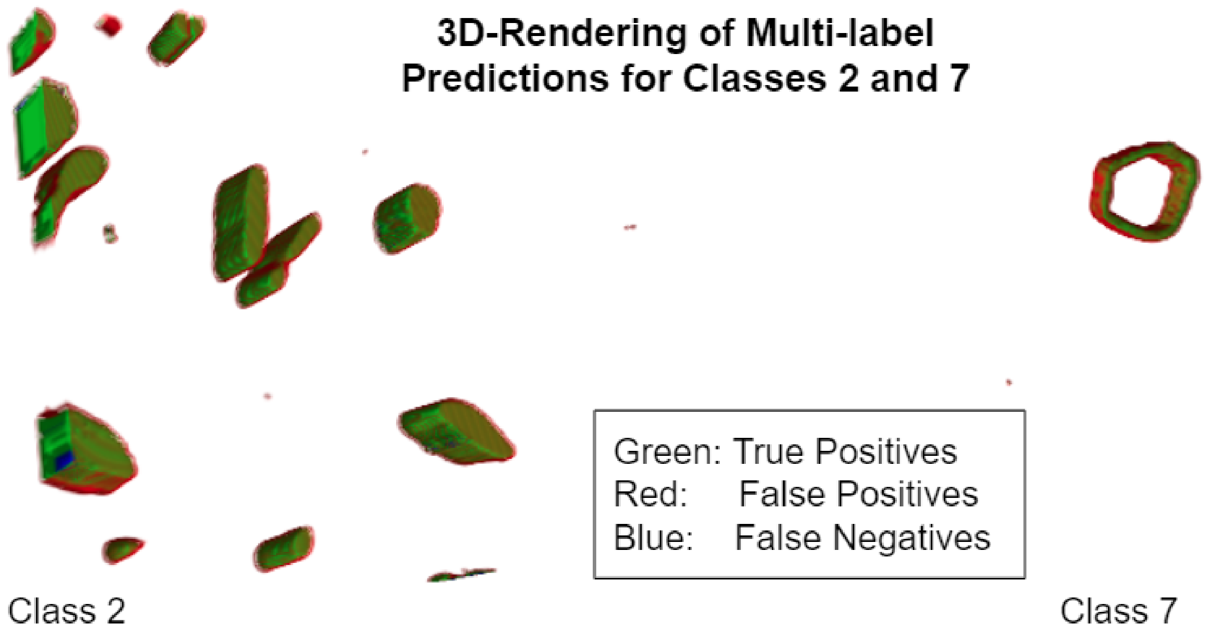


Fig. 14: ipyvolume volumetric rendering of prediction results for classes 2 (mitochondria) and 7 (myelin)

B. Measuring the effect of ZSSR

We applied ZSSR on 2D LR, HR images from our mouse cortex volume and on a single image we obtained from an Alzheimer's disease patient. Fig. 15 shows an example of one of our original images that was downscaled using scikit-image's *rescale* function, then upscaled using bicubic interpolation to simulate a low-resolution image. ZSSR boosting was then applied to produce a HR image.

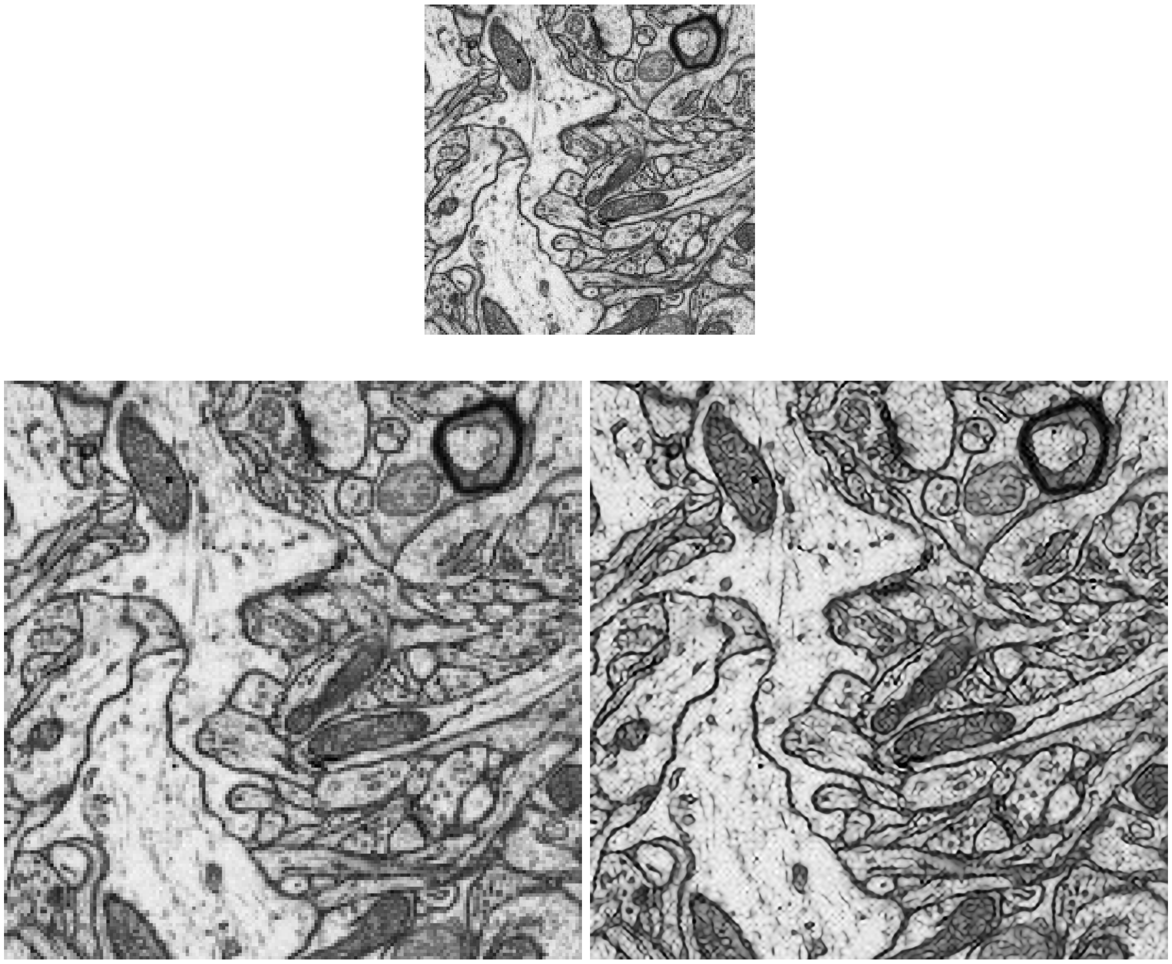


Fig. 15 Downscaled image (top), interpolated LR image (bottom left), ZSSR boosted image (bottom right)

We extended this usage to HR Alzheimer's disease images (Fig. 16) which could be more difficult to boost as they have more disordered structures. Unfortunately, we did not have label images of the Alzheimer’s data set, so could not measure ZSSR performance. In both cases however, visually we can see that ZSSR has the desired effect of increasing the resolution to reveal finer details.

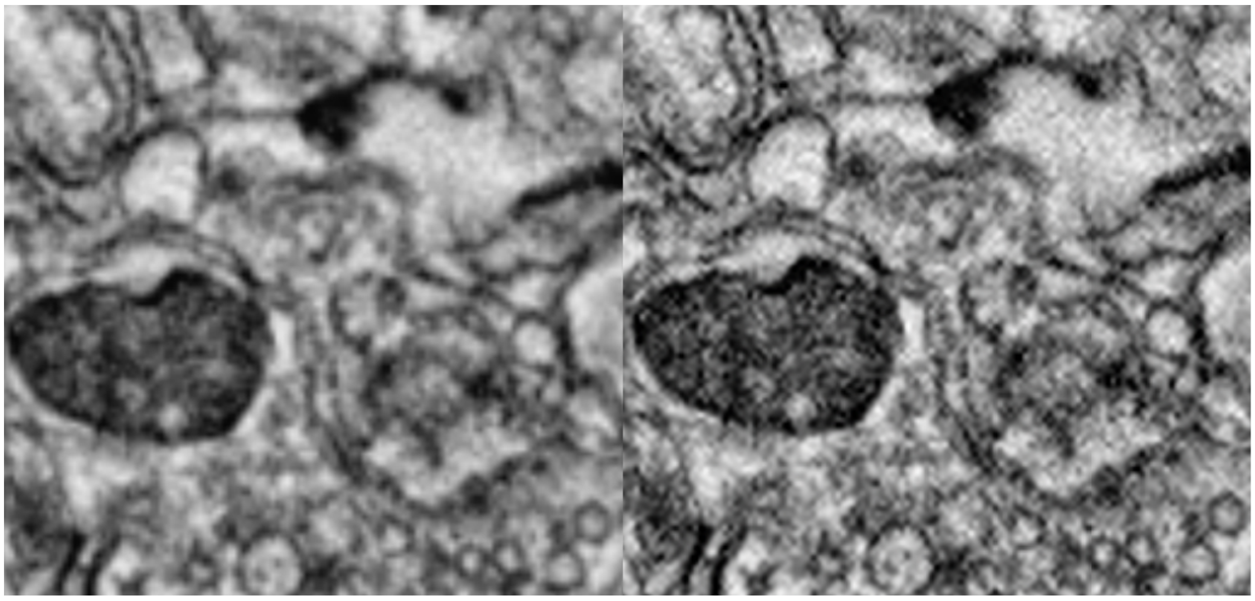


Fig. 16 Original HR image from Alzheimer's disease patient (left) and ZSSR boosted HR (left)

To evaluate the effect of ZSSR on volumetric segmentation we labeled LR and boosted images using our original multiclass segmentation model and the F1-scores for each class were calculated. Overall, the model did much worse in its performance, which we might expect from passing it artificially downsampled images. However, we do see that there is an improvement in performance in ZSSR boosted images compared to their non-boosted counterparts. This seems to be true for some underrepresented classes as well, such as the neurotransmitter vesicles.

Table 5. Original multiclass model performance on LR and LR-boosted volumes

Class	**F1-Score Low Res D**	**F1-Score Boost D**
0-cytosol	0.850	0.855
1-synapses	0.297	0.297
2-mitochondria	0.721	0.779
3-membrane	0.506	0.515
4-neurotransmitter vesicles	0.198	0.226
5-endoplasmic reticulum	0.234	0.289
6-microtubules	0.058	0.064
7-myelin	0.632	0.687

Fig. 17 shows an overlay of the ground truth for class 5 (endoplasmic reticulum) on an original HR raw image from our mouse cortex data set, as well as the predictions on the LR and LR-boosted image. In the downsampled image, we can see that the model has difficulty resolving the structure as the pattern seems to be lost. However, in the LR-boosted image, the pattern returns and the model is again able to make the correct prediction.

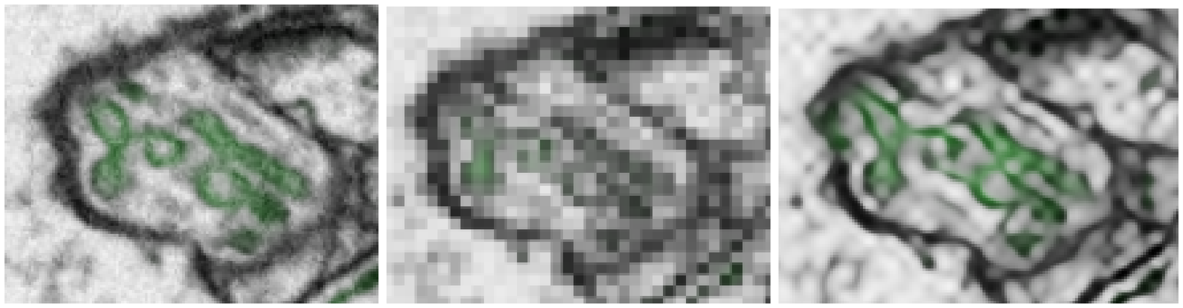


Fig. 17 Original mouse cortex image overlaid with ground truth (left), LR with overlaid class predictions (middle), LR-boosted with overlaid class predictions (right)

Another use case for ZSSR was to train multiclass models using boosted images and observe the effect on performance using our original test volumes. We trained two models, one with LR images as a control, and another with LR-boosted images. The LR-boosted model showed better F1-score results, especially on the minor classes (Table 6). Fig. 18 shows an overlay of the predictions from the LR model and LR-boosted model on class 4 (neurotransmitter vesicles).

Table 6. Model performance on HR images using LR and LR-boosted models

Class	**F1-Score LR model**	**F1-Score LR-boosted model**
0-cytosol	0.949	0.952
1-synapses	0.657	0.650
2-mitochondria	0.911	0.915
3-membrane	0.775	0.804
4-neurotransmitter vesicles	0.657	0.737
5-endoplasmic reticulum	0.698	0.699
6-microtubules	0.175	0.274
7-myelin	0.873	0.886

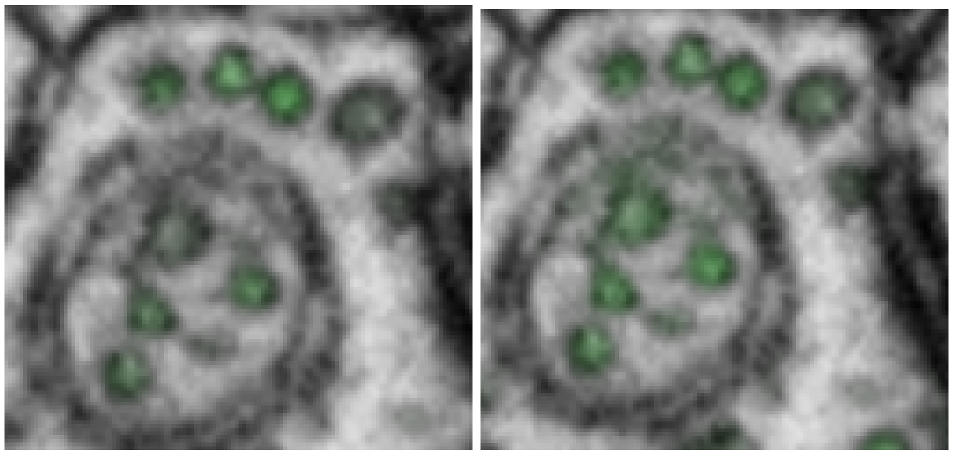


Fig. 18 Neurotransmitter vesicle predictions from LR model (left) and LR-boosted model (right)

XI. SOLUTION ARCHITECTURE

A. NeuroKube Microservice Extension

All tools and models used in this project are free to use and expand upon within the ncmir-mm namespace, which hosts NeuroKube. The trained models are stored in shared storage accessible to members of the namespace. Volume rendering tools are also available in shared Jupyter Notebooks.

The advantage of using Nautilus and the suite of neuroimaging tools provided by NeuroKube is that it removes the overall reliance on server based systems, replacing them with a serverless infrastructure that can dynamically re-allocates resources on-demand. Kubernetes allows for a multi-layered system that creates new nodes to scale an application when resource allocation is exceeded. To provide access to these resources for use with ad-hoc applications, we exposed a simple website using a Kubernetes Ingress API Service connected to a pod allocated with a GPU (Fig. 19). The website was created using Streamlit, an easy-to-use open-source Python library designed for machine learning and data science projects.

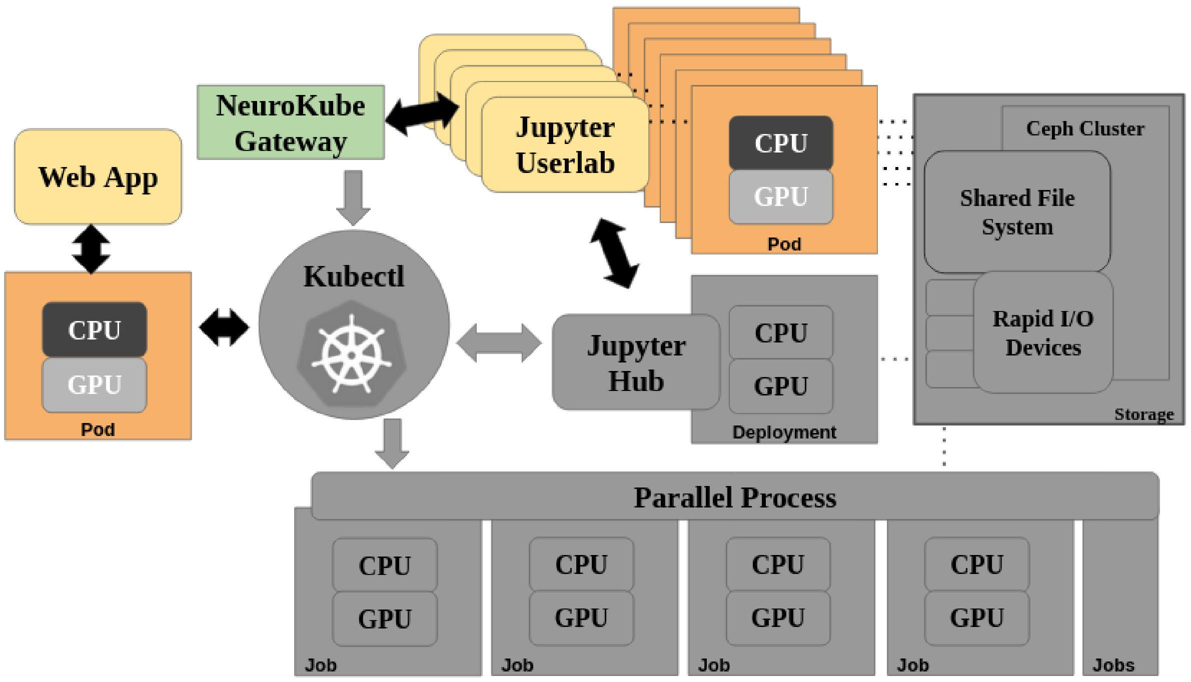
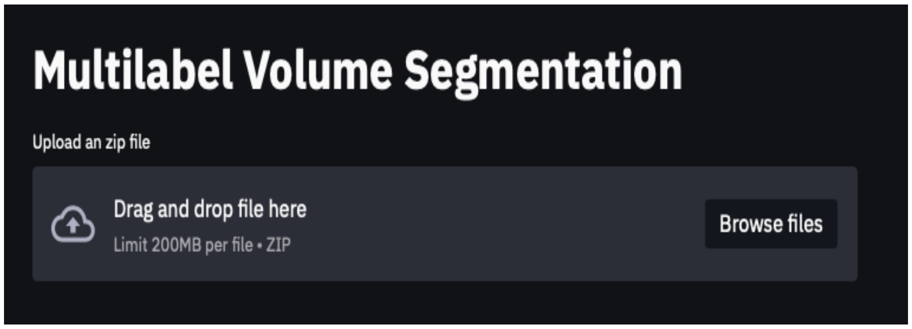


Fig. 19 NeuroKube with microservice framework

A simple use case was created using our multiclass model that takes a zipped folder of VEM images uploaded by a user (limited to 200MB) (Fig. 20), performs volumetric segmentation, and returns a volume render of the predicted classes using pyvista (Fig. 21). Ipyvolume was not used as it currently is not supported by streamlit.



Multilabel Volume Segmentation

Upload an zip file

Fig. 20 Streamlit file loader

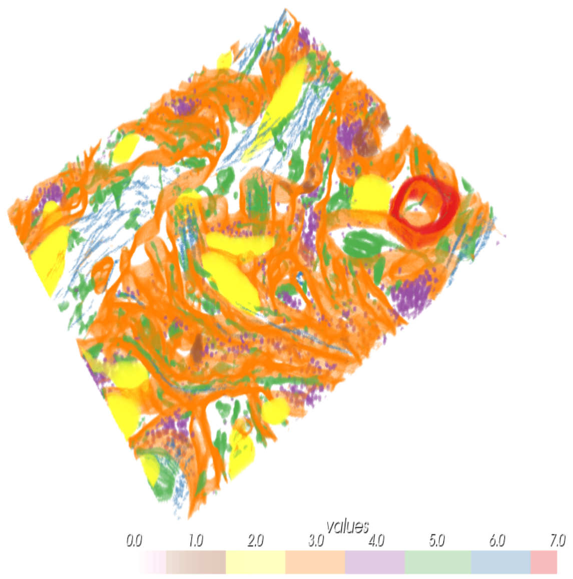


Fig. 21 Output pyvista volume render

XII. CONCLUSIONS

In conclusion, we were able to show that a multiclass model performed better than several binary “one-vs-rest” models in a volume segmentation task on brain VEM data. We demonstrated the effect on volumetric segmentation performance when applying zero-shot superresolution to boosted images. And finally, we added the models and visualization tools produced during this project to the neuroimaging toolbox provided through NeuroKube, and created a simple framework to expose them to researchers and the scientific community, powered by Nautilus.

ACKNOWLEDGEMENTS

We would like to thank our advisors Matthew Madany and Mark Ellisman for providing us with the context and direction for this project. We thank Dr. Ilkay Altintas for providing structure to the capstone process and keeping us on track. We thank the DSE MAS faculty and administrators for providing a seamless transition to remote learning and enabling our continued progress through the COVID-19 pandemic. Last but not least, we would like to thank our families and employers who provided their patience, flexibility, and encouragement during the duration of this two year program.

REFERENCES

[1] von Bartheld CS, Bahney J, Herculano-houzel S. “The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting”. *J Comp Neurol*. 2016;524(18):3865-3895. doi:10.1002/cne.24040

[2] S. DeWeerdt, “How to map the brain”. *Nature* 571, S6-S8 (2019). https://doi.org/10.1038/d41586-019-02208-0

[3] Narayan K, Subramaniam S. Focused ion beams in biology. *Nat Methods*. 2015;12(11):1021-1031. doi:10.1038/nmeth.3623

[4] M. Madany, M. Ellisman, S. Peltier (2020, December 4). NCMIR Data Science Capstone Zoom meeting.

[5] M. Madany, K. Marcus, S. Peltier, M. H. Ellisman and I. Altintas, "NeuroKube: An Automated and Autoscaling Neuroimaging Reconstruction Framework using Cloud Native Computing and A.I.," 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 320-330, doi: 10.1109/BigData50022.2020.9378053.

[6] M. Madany, vclass, (2020), GitLab repository, https://gitlab.nautilus.optiputer.net/madany/vclass

[7] Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, O. Ronneberger “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”, arXiv:1606.06650.

[8] J. Roels, J. Hennies, Y. Saeys, W. Philips, A. Kreshuk, “Domain Adaptive Segmentation in Volume Electron Microscopy Imaging”, arXiv:1810.09734

[9] S. Rahman, S. Khan, F. Porikli, “A Unified approach for Conventional Zero-shot, Generalized Zero-shot and Few-shot Learning”, arXiv:1706.08653.

[10] A. Shocher, N. Cohen, M. Irani, “‘Zero-Shot’ Super-Resolution using Deep Internal Learning”, arXiv:1712.06087.

[11] L. Smarr, “Draft Report to NSF on Pacific Research Platform Application Usage”, https://ucsd-prp.gitlab.io/images/reports/PRP_application_summaries--draft-12-10-20.pdf

[12] A. Wolny, “Accurate and versatile 3D segmentation of plant tissues at cellular resolution”, eLife 2020;9:e57613. Published 2020 Jul 29. doi:10.7554/eLife.57613