Semantic Segmentation of Ultra-Resolution 3D Microscopic Neural Imagery — CSCI599 Final Project Report

Hsiao-Lun Wang, Muye Zhu, Sourya Dey, Ting-Ru Lin

November 28, 2017

Abstract

Deep learning has emerged as the state of the art approach to semantic segmentation of biomedical images in fields such as neuroanatomy and oncology. Modern tissue clearing techniques and high throughput microscopy have made it possible to collect ultra-resolution biomedical imaging data on unprecedented scale, necessitating fully automatic image segmentation approaches. The characteristics of human fine tuning of tranditional approaches encourage the development of promising DNN-based techniques without the characteristics. Multiple renowned DNNs are modified and tested in this project to confirm the feasibility of automatic semantic segmentation of neuroanl cells. The evaluation results of proposed DNNs show more than 93% accuracy.

1 Introduction

Deep learning has emerged as the state of the art approach to numerous computer vision tasks in recent years, including image classification and segmentation. Segmentation of biological structures from microscopic data and other imaging modalities has long been a fundamental task in fields such as neuroanatomy and oncology. Satisfactory performance of traditional approaches of image segmentation are heavily dependent on parameter selection, which usually requires human fine tuning. Very recently, tissue clearing techniques [6] and high throughput microscopy [4] have made it possible to collect ultra-resolution biomedical imaging data on unprecedented scale, necessitating fully automatic image segmentation approaches. It is not straightforward applying deep learning to biomedical images due to the data hungry nature of neural networks and the difficulty in obtaining large quantities of ground truth annotations, as well as large file size as a result of high resolution, 3D volumetric image collection process. Several authors had success by taking advantage of fully convolutional architectures and image augmentation techniques that closely mimic the type of variation frequently observed in biological systems [12, 15, 3].

In this project, we tackle the problem of segmenting neuronal cell body (aka soma) and processes (aka neurites) from ultra-resolution terabyte range image dataset. The images are obtained by scanning Golgi stained mouse brain tissue under bright field at 30x magnification[8]. It is great interest to understand the organization of neural circuitries, and how the morphologies of each neuronal cell types contribute to their function within the circuitry. Segmentation of soma and neurites are a first step to obtain neuronal morphologies. The dataset is consisted of sequences of ultra-resolution microscopic image of the mouse brain, scanned at consecutive focal planes. Each individual image has a resolution of around 22,000 pixels by 26,000 pixels (Figure 1). Large scale microscopic imaging datasets often demonstrate varying levels of brightness, contrast and feature salience at different locations, making parameter sensitive classic segmentation approaches unsuitable for high-throughput semantic segmentations. Herein, the development of a robust, fully automatic semantic segmentation system based on deep neural networks (DNNs) appear an attractive alternative approach to the task.

The main goals and challenges of this work are the following:

- Enormous volumes of laboratory images are provided in primitive formats. Ground truth annotation needs to be derived by the team. The annotation process requires domain knowledge.

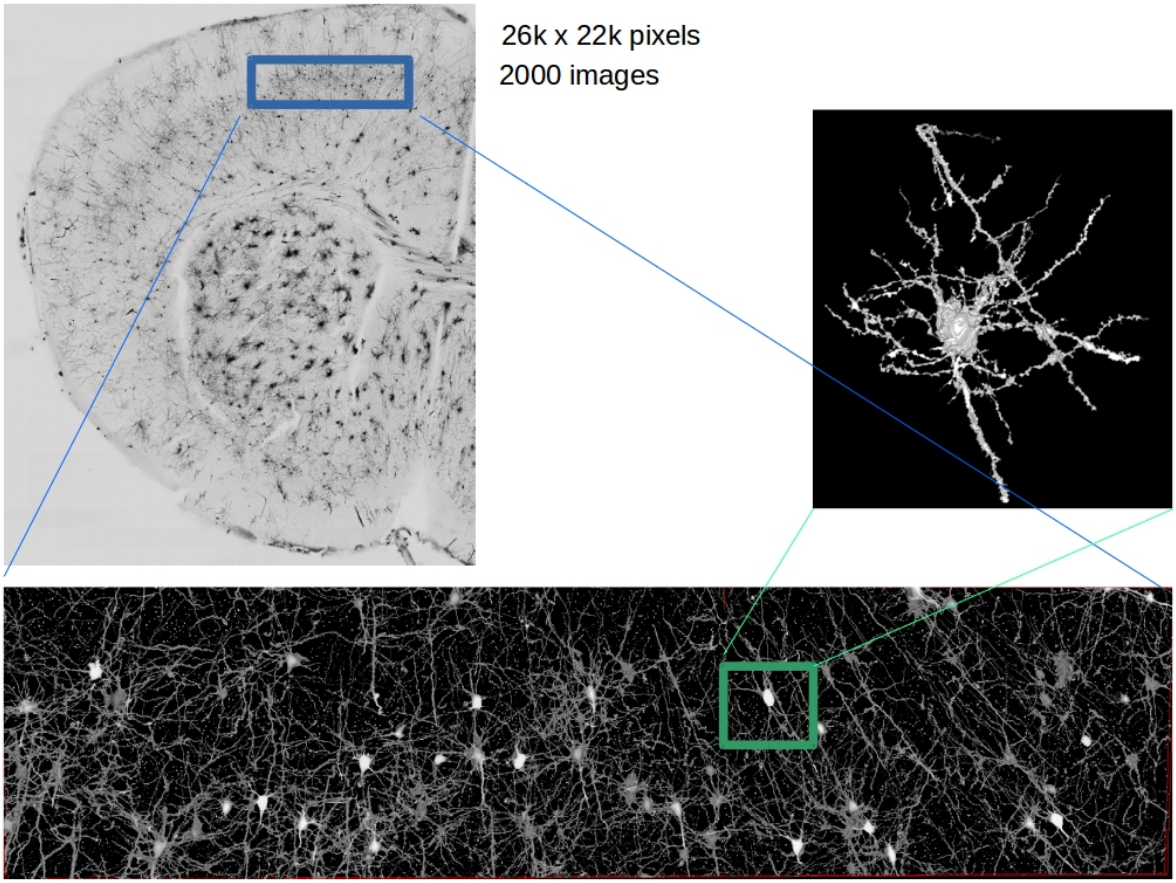


Figure 1: Overview of dataset

- Objects inside a biomedical image are very large and lossless preprocessing techniques should be conceived for following hierarchical feature extractions;
- Image annotation is slow and laborious. It is desirable to have a deep learning synthetic dataset generation approach.
- Appropriate deep learning architectures need to be selected and modified to accomplish semantic segmentation of soma and neurites.

The rest of this report is organized as follows. Section 2 gives an overview of pre-processing methods. Section 3 outlines our approach using generative-adversarial networks (GANs). Sections 4 and 5 discuss segmentation results obtained from several attempted network architectures, derived from existing popular network structures. Finally, section 7 concludes the report.

2 Pre-processing and Training data annotation

Since the dataset we are working with is collected in one team member's laboratory, there is no available data annotation initially. Our first challenge was to produce high quality image annotation with three class labels: (0: background, 1: neurite, 2: soma), or in some cases two class labels: (0: background, 1: foreground), to train the neural networks. Since majority of microscopy image is in grayscale, we transform he RGB raw images in the current dataset to grayscale, so that the trained network can be fine tuned for data collected under different configurations.

We experimented with two different image annotation strategies: manual annotation and automatic annotation by feature extraction, each with its pros and cons. Manual annotation is generated by labeling pixels according to their classifications with the image manipulation software GIMP on a small number of images. The manual annotation is reliable but suffers from low throughput. Thirty 2048x2048 image regions are manually annotated for training purposes. Half of these are deliberately chosen to contain only background or noise, so that the network can learn to produce robust segmentation results.

The automatic annotation is based on well established image processing techniques of multi-scale blob and curve linear structure detection[7, 13]. We have developed workflows to perform distributed processing on high performance computing clusters and are able to annotate a full 2.5 terabyte dataset in around 3 hrs. Annotation of neurites and somas are produced by thresholding their corresponding feature maps. However, the multi-scale feature detection algorithm has several disadvantages, as shown in Figure 2: (1) unaware of context – the golgi stain produces small particles of noise pixels throughout the image volume. The algorithm is unable to differentiate them from true foreground (signals arising from neurite or soma). (2) the optimal scales to use for each image is unknown – The algorithm is sensitive to the physical dimensions of curve lines in a given image, and require appropriate scale factors to be supplied in order to produce good results. The set of optimal scale factors is tedious to find, and often fails at very large or very fine structures, and does not have same magnitude of response across the dataset. The algorithm also is prone to losing the boundaries foreground pixels. (3) sensitive to noise – The algorithm performs reasonably well in crisp, well formed images, but produces grossly erroneous results in blurry or high background images.

A 4-class annotation is also produced from both automatic and manual annotation. The additional 4th class is the "border" class, defined as all pixels with a Manhattan distance less or equal to 2 to any foreground pixels. This annotation is generated to test whether explicitly including border pixels during training improves segmentation of structures in close proximity to each other. Data used in all training procedures are drawn from the annotation images described above. Input images of GANs are 256x256 pixels generated by manual and automatic annotation and are resized in 64x64 pixels to compressed features for hierarchical extraction. Input images of SegNet are 2048x2048 pixels in grayscale. Input images of UNet are 256x256 pixels in grayscale.

3 GANs

3.1 Overview

We also try to use Gan to generate the segmented images from the original images. The reason we try to generate segmented images from GAN is that we can use upaired images to train our Cycle Gan model to avoid collecting manual segmented images. We have first tried to generate segmented images from original images with three different GANs, DCGAN [11], WGAN [1], WGAN-improved [10] with paired data images. We have also use unpaired data to train cycle GAN [16]

3.2 GAN Architecture

The Gan architecture is composed of a Generator and a Discriminator. For the generator, the input layer size for our test case is 64x64x1, following by three convolution layers, and three deconvolution layers. The input image can be viewed as the prior condition. We do have other random prior distributions as inputs. The first convolution layers has size 32x32x16 with kernel size 6. The second convolution layer is 16x16x32 with kernel size 4, and the third convolution layer is 8x8x64. Each convolution layer is followed by a batch normalization layer, an average pooling layer and the lrelu activation functions. For up-sampling, we use three deconvolution layers. The first deconvolution layers is 16x16x32. The second layer is 32x32x16, and the third layer is 64x64x1. Except last layer, each deconvolution layer is followed by a batch normalization layer, and the lrelu activation functions. The last layer is has no normalization layer and the activations are hyperbolic tangent functions. The discriminator is composed of three convolution layers as generator, following by a fully connected layer with dimension 200x1.

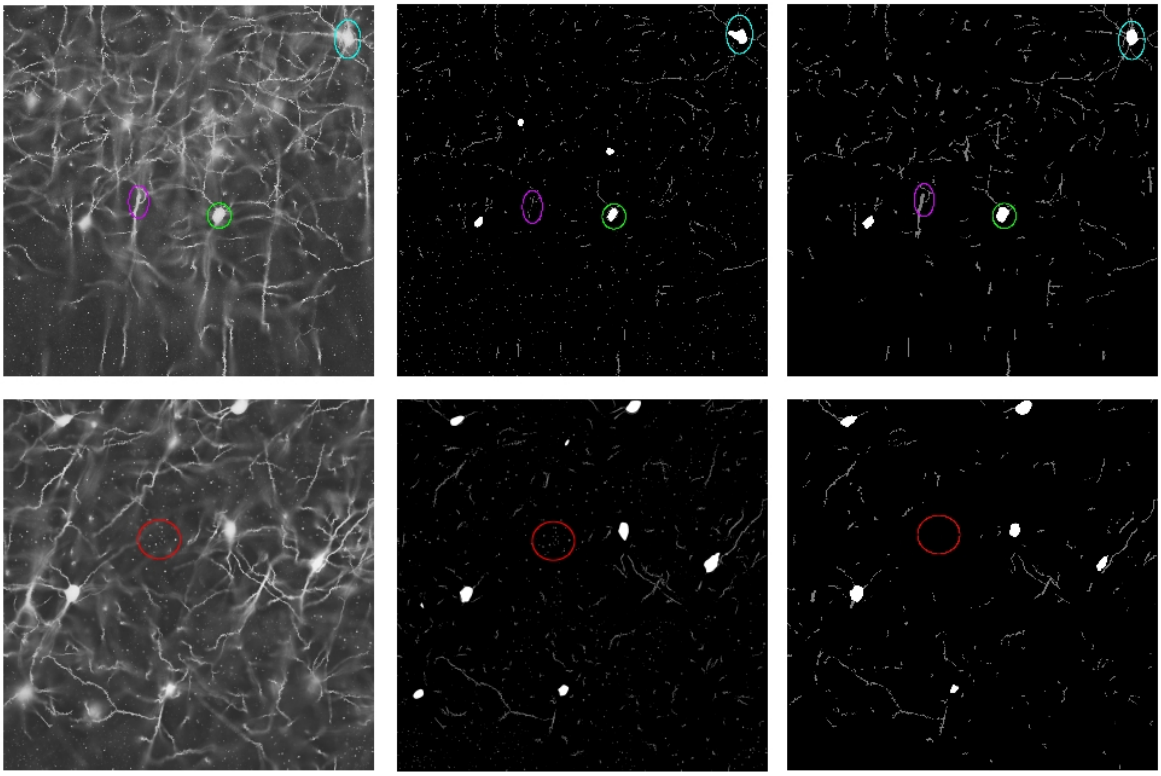


Figure 2: Issues with automatic image annotation using classic multi-scale feature detection algorithms. Left column: raw image. Mid column: automatic segmentation. Right column: manual segmentation. Pink circle: large tubular structures are often not extracted. Green circles: Foreground boundary pixels are often not extracted. Border of soma is often misclassified as neurite. Connectivity of neurite to soma is not well preserved. Red circle: high intensity background is retained in the segmentation.

3.3 Loss Functions and Gan Types

In this section x means the original image set, y means the segmented image set.

- DCGAN: The objective of G is $min\ \mathbb{E}_{x\sim pdata(x)}[1-\log D(G(x))]$. The objective of D is $max\ \mathbb{E}_{x\sim pdata(x)}[1-\log D(G(x))]+\mathbb{E}_{y\sim pdata(y)}[\log D(y)]$.
- WGAN: The objective of G is $max\ \mathbb{E}_{x\sim pdata(x)}[D(G(x))]$. The objective of D is $max\ -\mathbb{E}_{x\sim pdata(x)}[D(G(x))]+\mathbb{E}_{y\sim pdata(y)}[D(y)]$ with the constraints that $\forall\theta\in D,G,\ \theta<10$.
- WGAN-improved: The objective of G is $max\ \mathbb{E}_{x\sim pdata(x)}[D(G(x))]$. The objective of D is $max\ -\mathbb{E}_{x\sim pdata(x)}[D(G(x))]+\mathbb{E}_{y\sim pdata(y)}[D(y)]+\mathbb{E}_{x\sim pdata(x),y\sim pdata(y)}[\nabla(D(\alpha G(x)+(1-\alpha)y)-1)^2]$.
- Cycle-Gan: Let $G_{x\to y}, D_y$ be the generator and the discriminator of segmented images, and $G_{y\to x}, D_x$ be the generator and the original images. The objective of D is $max\ \mathbb{E}_{x\sim pdata(x)}[1-\log D_y(G_{x\to y}(x))]+\mathbb{E}_{y\sim pdata(y)}[\log D_y(y)]+\mathbb{E}_{y\sim pdata(y)}[1-\log D_x(G_{y\to x}(y))]+\mathbb{E}_{x\sim pdata(x)}[\log D_x(x)]$. The objective of G is $min\ \mathbb{E}_{x\sim pdata(x)}[1-\log D_y(G_{x\to y}(x))]+\mathbb{E}_{y\sim pdata(y)}[1-\log D_x(G_{y\to x}(y))]+L_{cycle}$, where L_{cycle} is the L2-norm of pixel-wise difference between $x, G_{y\to x}(G_{x\to y}(x))$, and $y, G_{x\to y}(G_{y\to x}(y))$.

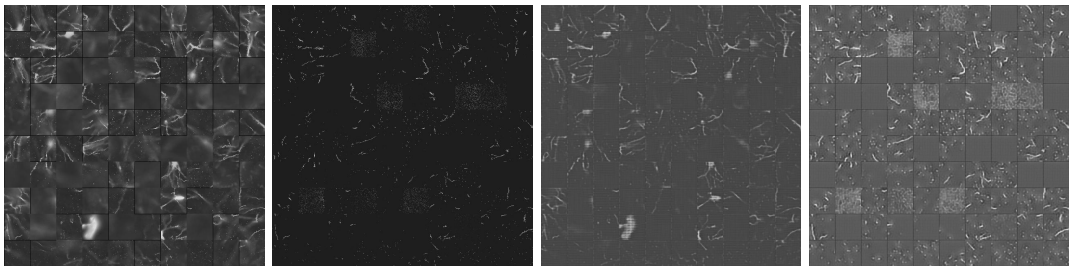


Figure 3: Some results of cycle-gan of 30th epoch. From left to right, real original images, real segmented images, generated original images, generated segmented images.

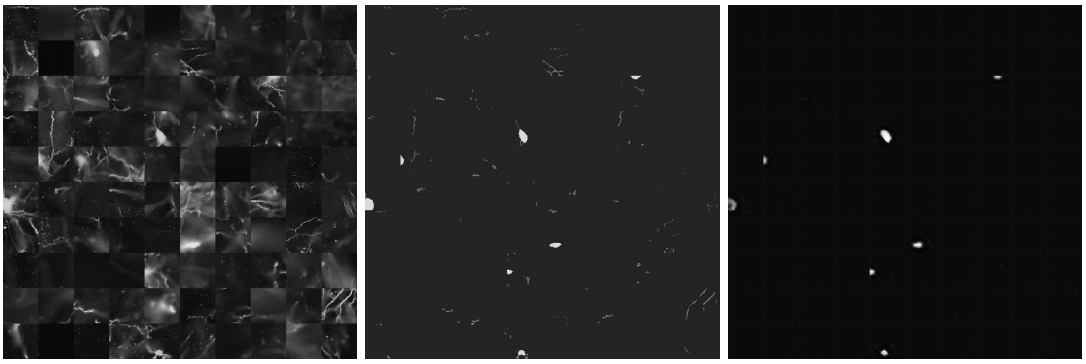


Figure 4: Some results of wgan. From left to right, real original images, real segmented images, generated segmented images.

4 SegNet

4.1 Overview

We used an architecture similar to SegNet[2], which has become a popular architecture for semantic segmentation. The deep learning library Keras[5] was used to train a model inspired from an imlab-uiip Github project[14], on a GPU running Tensorflow. The input images were pre-processed from the original images using filtering, which resulted in a resolution of 8192x2048 pixels with each having a value between 0 and 255, the former indicating complete black and the latter complete white. These were horizontally sliced into 4 separate images of size 2048x2048 pixels each. These images were then segmented by thresholding their values to result in similarly-sized images with pixel values either 0 or 1, i.e. 2 segmentation classes. Figure 7 shows these images side by side. These were used as the input and ground truth labeled images for our SegNet style architecture. The total number of images were 396, of which 296 were used for training and the remaining 100 for testing.

4.2 Training

One of the challenges we faced during training is the large size of the input images. The default float data type used in the Python Numpy library is 'float32', which needs 4 bytes to store a single value. This means that the size of the entire data was $396 \times 2048 \times 2048 \times 4 = 6.2$ GB, which is cumbersome for the machine memory to handle. To alleviate this problem, we split the data into sets of 80 images and loaded them one by one. The large size of data also meant that we had to put restrictions on the batch size and number of convolutional filters used during training. We ended up with a batch size of 2 and 4 filters for the initial convolutional layer. The final network configuration is given below:

Figure 5: Some results of wgan-imporved. From left to right, real original images, real segmented images, generated segmented images.

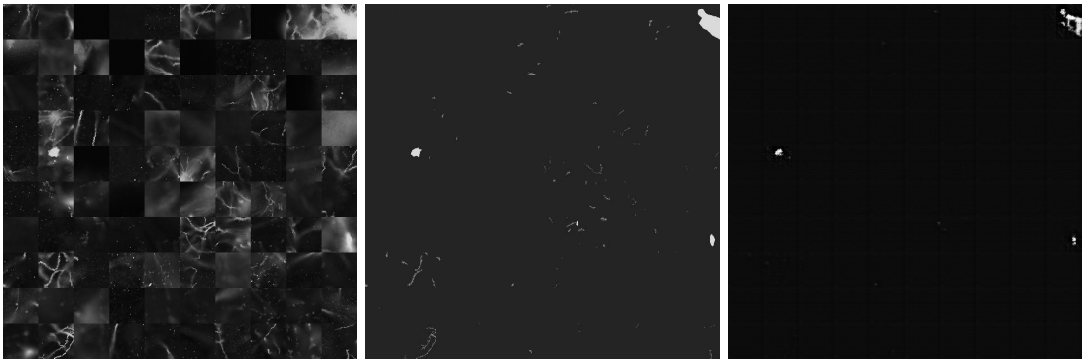
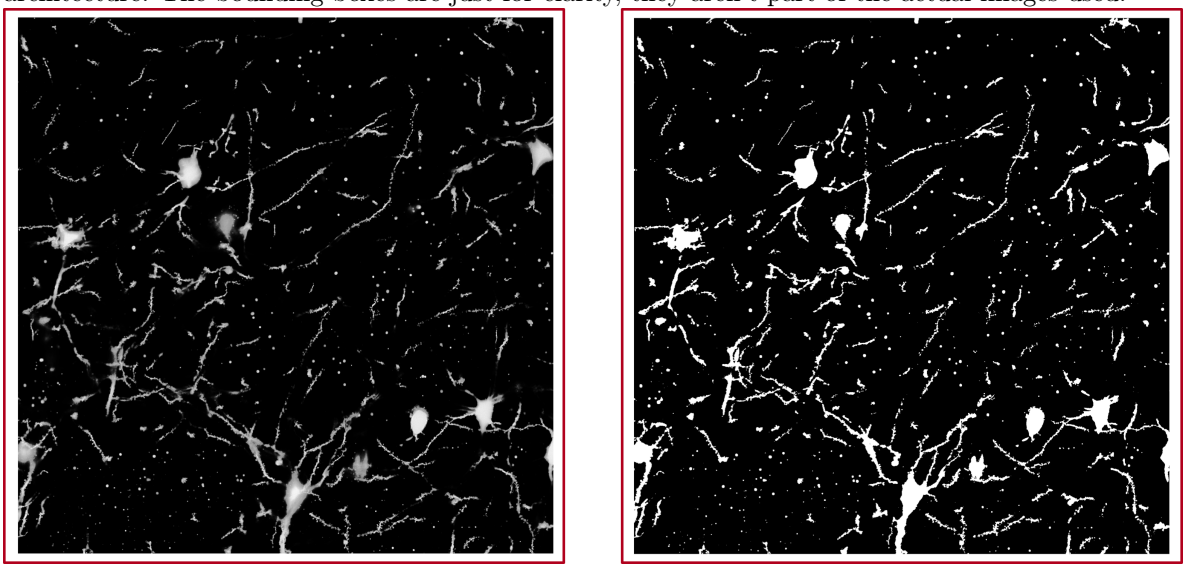


Figure 6: Some results of dcgan. From left to right, real original images, real segmented images, generated segmented images.

- Encoder:
  - 2 convolutional layers, each with 4 filters, window size 7x7, ReLU activation and succeeded by a batch normalization layer. This was followed by 4x4 non-overlapping max-pooling.
  - 2 convolutional layers, each with 16 filters, window size 5x5, ReLU activation and succeeded by a batch normalization layer. This was followed by 4x4 non-overlapping max-pooling.
  - 9 convolutional layers, each with window size 3x3, ReLU activation and succeeded by a batch normalization layer. The first 3 had 64 filters, and the last 6 had 128 filters. A 2x2 non-overlapping max-pooling layer succeeded every 3 convolutional layers.
- Decoder:
  - 9 convolutional layers, each with window size 3x3, ReLU activation and succeeded by a batch normalization layer. The first 6 had 128 filters, and the last 3 had 64 filters. A 2x2 upsampling layer preceded every 3 convolutional layers.
  - 2 convolutional layers, each with 16 filters, window size 5x5, ReLU activation and succeeded by a batch normalization layer. This was preceded by 4x4 upsampling.
  - A 4x4 upsampling layer followed by a convolutional layer with 4 filters, 7x7 window size, ReLU activation and succeeded by batch normalization.
  - A convolutional layer with 1x1 window size and 2 filters, each corresponding to an output class. This was followed by batch normalization.
  - A softmax output layer.

Figure 7: Examples of an input image (left) and an output image (right) used in our SegNet style architecture. The bounding boxes are just for clarity, they aren't part of the actual images used.



All weights were initialized using the Glorot Uniform technique[9], and all biases were initialized to 0. The entire network took about 13 minutes to train for 1 epoch on 296 training images in batches of 2. We used the stochastic gradient descent optimizer and experimented with different learning rates, decay coefficients, and momentum, as described in the next section.

4.3 Results and Lessons Learnt

We used 2 metrics for performance – cross-entropy loss and accuracy on the test set of 100 images. We found that Nesterov momentum performed better than ordinary momentum. For a learning rate of 0.001, the former gave a loss of 0.6108 and accuracy of 87.99% as compared to the latter which gave a slightly higher loss of 0.6171 and a significantly worse accuracy of 83.86% after training for 1 epoch. For the different learning rates we experimented with, the optimum value was 0.01 since it gave a loss of 0.3663 and accuracy of 93.82% after 1 epoch of training. However, this test accuracy did not improve after training for more epochs. We hypothesize that this is due to the fact that the decay coefficient might have been too small or regularization was insufficient, causing the network to overfit as a result. Unfortunately computational space and training time constraints led us to cut short our experiments and settle for the training parameters mentioned above.

5 U-Net

U-Net is a popular fully convolutional encoder decoder architecture with successful applications in biomedical image segmentation tasks [12, 15, 3]. Importantly, to obtain fine segmentation and preserve spatial localization, the decoder combines activations from both the convolutional and deconvolutional layers. Building on top of the original U-net publication, we trained an architecture end-to-end with 6 convolutional layers and 6 deconvolutional layers. We consider and experiment with several aspects of the architecture:

- Input image dimension. The network was trained with input image dimensions (256, 256, 1). The height and width are chosen to be sufficient to fully contain the largest known valid structure (which is about 120 pixels in height and width).

- Training dataset. Training was attempted with 3 class and 4 class images from both the automatic and manual annotations. 4 class annotation is not beneficial to the outcome. Network trained from automatic annotations is not able to perform well in difficult images with high background. Therefore the 3 class manual annotation dataset is selected for further training.
- Output dimension from the last convolutional layer. We choose (8, 8) to be the height and width of output from the final convolution operation to allow large enough receptive field and therefore context awareness in later layers. We observe height and width greater than (8, 8) result in noisy output in images where the whole frame is consisted of high intensity. background.
- Image augmentation. We use each image as well its 90, 180 and 270 degree rotated copies for training.
- Class weights. Since our dataset is dominated by background, we use a weighted cross entropy loss to avoid the network converging to all background outputs.
- Convolutional layer weights. Each deconvolutional layer combines the activation from deconvolution, as well as weighted activation from its corresponding convolutional layer. We find smaller weights in early convolutional layers and larger weights in later convolutional layers produce robust and accurate results.

5.1 Network architecture

- Convolutional layers
  - The 1st convolutional layer has 2D filter dimension (7, 7), the 2nd and 3rd convolutional layers have 2D filter dimensions (5, 5), remaining convolutional layers have filter dimensions (3, 3). The 1st convolutional layer employs 16 filters, with each succeeding layers doubling the filter number in the previous layer. The convolution maintains inputs' height and width.
  - Each convolutional filter operation is followed by batch normalization and leaky relu activation.
  - Each relu activation is followed by a max pooling with stride (2, 2)
- Deconvolutional layers
  - The dimensions and numbers of deconvolutional filter are identical to their respective convolutional layers. Strides of (2, 2) are used, mirroring the (2, 2) strides of max poolings. The final deconvolutional layer has 2D filter dimension (1, 1) and same number of filters as classes in order to produce pixel classification.
  - Each deconvolutional filter operation is summed with the weighted convolution operation result from the corresponding convolutional layeris, and batch normalized
  - Each batch normalization is followed by a leaky relu activation.
  - The 6th deconvolutional layer has 3 filters of dimension (1, 1). Segmentation is generated with softmax activation from this layer.

5.2 Network training

The 30 2048x2048 manual annotation images is equivalent to 1920 256x256 images. Including the image rotations, we have 7680 training images. The network is trained for 40 epochs with a batch size equal to 8. On a Nvidia GTX970 card, the training takes 2 hrs.

5.3 Results and future directions

Deep learning has not been as widely applied to biomedical images as natural images, due to difficulty in annotating large datasets. However, biomedical datasets often have characteristic structures of interest, for example, the tubular neurites in our image data. Recent research has demonstrated that even a small number of well annotated images will allow a neural network to

learn these structures and generalize to unseen data [3]. Our results support these previous findings. Qualitatively, the trained model is able to recognize somas and neurites of neurons in both training and unseen data (Figure 8). It also respects the topology of neuronal structures better than classic methods, retaining boundary pixels and capable of maintaining structures from a same neuron in an image connected component. It is also more robust to noisy image frames, a critical feature for deploying fully automated segmentation procedures on very large datasets (Figure 9). We also quantitatively compare segmentation training accuracies from the neural net and from classic methods, using 4 different metrics: pixel accuracy, mean accuracy, mean IU and frequency weighted IU, as described in [12] (Figure 10). Neural network outperforms classic methods in all metrics. We did not perform quantitative comparison on segmentation results from unseen images. The segmentation problem is difficult even for persons with significant domain knowledge (a PhD student in neuroscience produced the manual annotation). The per pixel classification is often ambiguous due to different degrees of blurring and background intensity. Holding the classification criteria consistent over time is also very challenging. Aliasing effects during image manipulation in GIMP can also lead to ambiguous classification of pixels. Therefore training the network towards 100% accuracy may not correlate with better outcome. The neural network's role in the image analysis workflow is a preprocessor that provides input to downstream algorithms which extract condensed topological structures from the neuronal structures. Per pixels accuracy is therefore less important than the similarities between skeletons of a ground truth structure and segmented structure. This comparison is an important future direction of the project. Additionally, we observe that in some images, the rotated copies generate slightly different segmentations from the original image when the rotation difference is accounted for, suggesting the network is not entirely invariant to rotation and could benefit from more training instances. We plan to use random elastic deformation, a common technique for biomedical images, to further augment the training dataset. Moreover, our image data are sequential in nature, where biological tissues are imaged consecutively at different focal planes. We have considered each image independently. However, sequential data will benefit from recurrent neural networks. Some of the pixel classification ambiguities we encounter in 2D images may be resolved when 3D informations are utilized. A natural and critical extension of the current project is then to apply a convolutional-LSTM network to the dataset and extract topologies of biological structures.

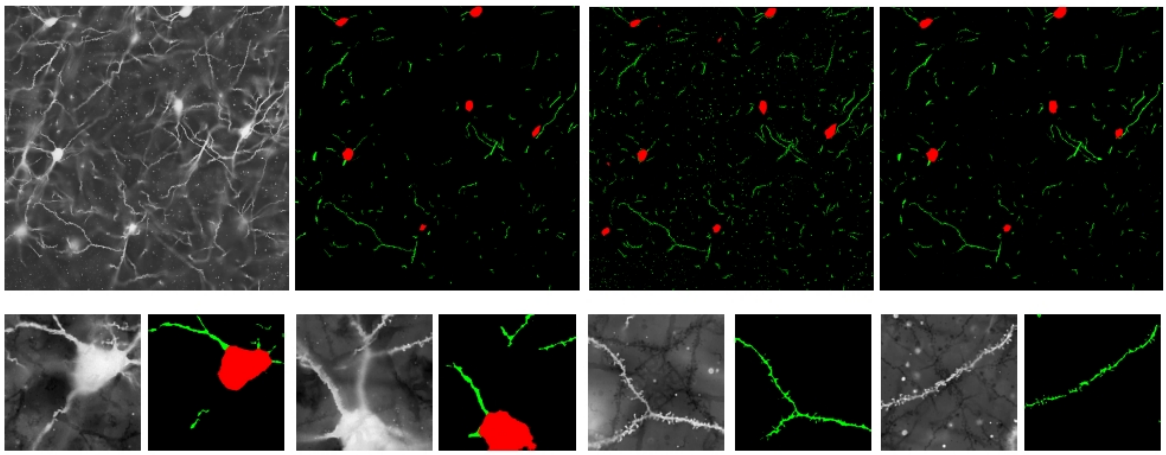


Figure 8: Representative segmentation results. Pixel class 1 (neurite) is colored green, and pixel class 2 (soma) is colored red. We aim to classify blurry / out of focus pixels as background. Top row: segmentation results during training. From left to right: original image, manual segmentation, classic method segmentation, neural network segmentation. Note the classic method segmentation is contaminated with many background stain artifacts. Bottom row: segmentation results paired with original, unseen images. The network has learned to identify biological structures from varying background intensities.

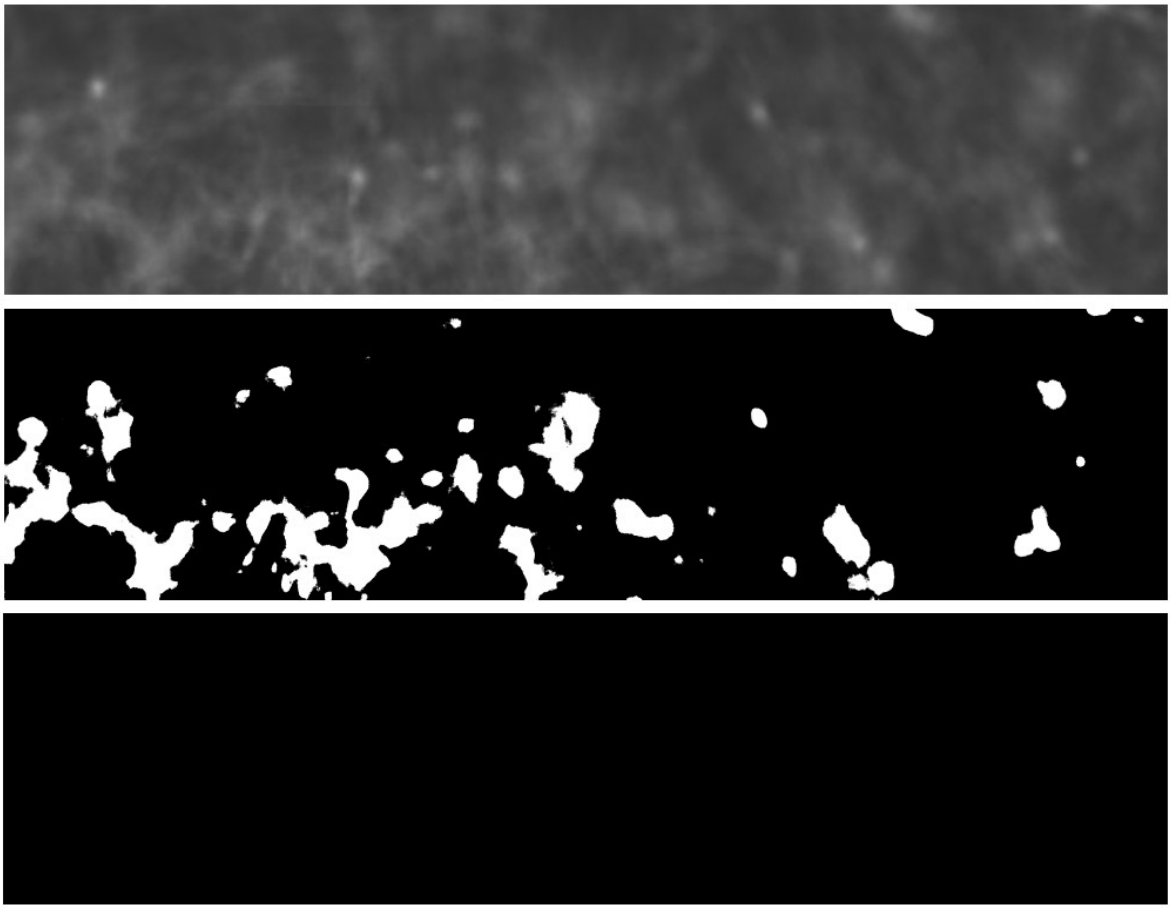


Figure 9: The neural network performs robustly in low quality image regions. Top: raw image. The image is blurry and has high intensity background. Because no reliable information can be derived, we would like for the segmentation result to be pure background. Middle: very noisy segmentation result obtained with classic method. Bottom: neurla network segmentation classifies all pixels as background.

6 Comparison and Discussion

To evaluate the performance of our model, we use the standard metrics from Cityscapes benchmark, including per-pixel accuracy, per-class accuracy, and mean class Intersection-Over Union (Class IOU).

Loss	Per-Pixel acc.	Per-Class acc.	ClassIOU
U-Net	0.979	0.3347	0.329
DCGan	0.993	0.444	0.442
WGAN	0.992	0.489	0.481
WGANGP	0.992	0.467	0.464
CycleGan	0.979	0.334	0.329

7 Conclusion

In this project, we tackle the problem of segmenting neuronal cells into three labels from ultra-resolution terabyte range image dataset. During preprocessing we realize manual annotation and

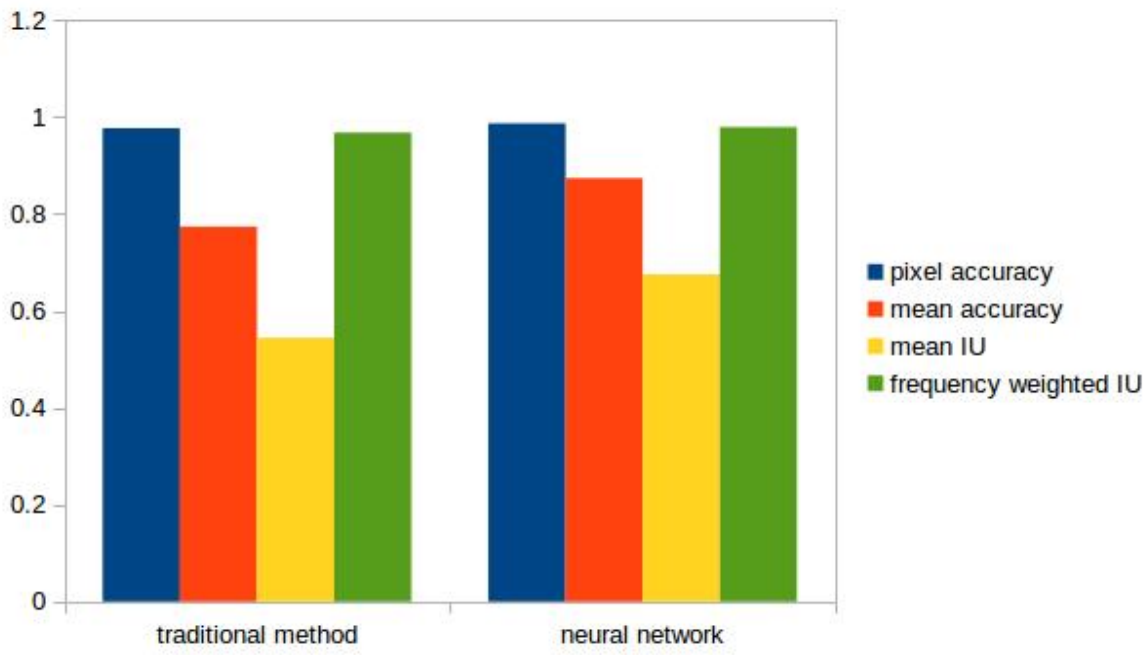


Figure 10: Training accuracies by classic method and by neural network, measured by pixel accuracy, mean accuracy, mean IU and frequency weighted IU.

automatic annotation of 30 images of 2048x2048 pixels. Later, we develop multiple DNN-based semantic segmentation systems in architectures referred to GAN, SegNet and U-Net. The standard metrics of per-pixel accuracy of U-Net and GAN can achieve more than 97%. The accuracy of SegNet after training is 93.82%. A natural and critical extension of this project is to combine 2D topologies of biological structures for precise 3D semantic segmentation.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[3] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. *ArXiv e-prints*, June 2016.

[4] Bi-Chang Chen, Wesley R. Legant, Kai Wang, Lin Shao, Daniel E. Milkie, Michael W. Davidson, Chris Janetopoulos, Xufeng S. Wu, John A. Hammer, Zhe Liu, Brian P. English, Yuko Mimori-Kiyosue, Daniel P. Romero, Alex T. Ritter, Jennifer Lippincott-Schwartz, Lillian Fritz-Laylin, R. Dyche Mullins, Diana M. Mitchell, Joshua N. Bembenek, Anne-Cecile Reymann, Ralph Böhme, Stephan W. Grill, Jennifer T. Wang, Geraldine Seydoux, U. Serdar Tulu, Daniel P. Kiehart, and Eric Betzig. Lattice light-sheet microscopy: Imaging molecules to embryos at high spatiotemporal resolution. *Science*, 346(6208), 2014.

[5] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[6] K Chung, J Wallace, S Kim, S Kalyanasundaram, A. S. Andalman, and T Davidson. Structural and molecular interrogation of intact biological systems. *Nature*.

[7] Alejandro F. Frangi, Wiro J. Niessen, Koen L. Vincken, and Max A. Viergever. *Multiscale vessel enhancement filtering*, pages 130–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[8] Mitch Glickstein. Golgi and cajal: The neuron doctrine and the 100th anniversary of the 1906 nobel prize. *Current Biology*, 16(5):R147 – R151, 2006.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, 2010.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[11] Stephan Halbritter. Generative adversarial networks. 2017.

[12] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *ArXiv e-prints*, November 2014.

[13] Rashindra Manniesing, Max A. Viergever, and Wiro J. Niessen. Vessel enhancing diffusion: A scale space representation of vessel structures. *Medical Image Analysis*, 10(6):815 – 825, 2006.

[14] Ivan Pazhitnykh. Segnet model implemented using keras framework. https://github.com/imlab-uiip/keras-segnet.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[16] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.