CS230

SomaNet: Segmenting Neurons in Forebrain Assembloid Microscopy

Julia Schaepe, Ashi Agrawal
Department of Computer Science
Stanford University
jschaepe@stanford.edu, ashi@cs.stanford.edu

Abstract

The segmentation of neuron soma in 2D microscopy images is a critical step for tracking neuron migration. There are many features, such as average velocity, direction, and saltation frequency, that are crucial to investigating and quantifying neuron migration. However, neuron tracking is still a challenging problem due to the complex and overlapping biological structure of neurons and the variability in microscopic imaging. To address these issues, we tackle the first step of tracking, segmentation, through the use of transfer learning on a UNet architecture trained on non-neuron nucleus segmentation data. We achieve an intersection over union between the segmented masks and the predicted masks of 50% and accuracy and precision of individual soma recognition of 74% and 94%, respectively. This network can ultimately be used as a basis for creating segmented videos of neuron somas that are much simpler to track than the original microscopy images.

1 Introduction

The Pasca Lab in Stanford's Department of Psychiatry and Behavioral Medicine recently developed a powerful human 3D brain organoid platform that involves directed differentiation, allowing for the creation of multi-region neural 3D cultures called forebrain assembloids. This process captures in vitro central nervous system developmental processes, including the saltatory migration of interneurons on their way to the cerebral cortex. These migrations have distinct phenotypes, measured by the saltation frequency and speed of the neuron cell body, the soma, in patients with psychiatric disorders, autism spectrum disorder, or learning or cognitive impairment. The current process to obtain these measurements is painstaking and time-consuming, presenting an opportunity for the use of deep learning to quickly assess and capture cell migration features in patients and controls.

By utilizing videos of interneuron migration provided by the Pasca Lab, we aim to ultimately track neuron soma movement. This paper details our use of transfer learning on a UNet architecture to segment neuron soma from 2D microscopy, the first step to tracking neuron soma in videos. We first train on non-neuron cell nucleus segmentation data from the Kaggle 2018 Data Science Bowl training set, which consists of 670 images and their corresponding ground truths [3]. After training the UNet, we implement transfer learning and retrain the final three layers of the convolutional neural network (CNN) on 96 256x256 RGB images of interneurons and their respective ground truths. The CNN outputs a 256x256 greyscale predicted mask of the soma of the neurons. The ultimate goal is that this network can provide a basis for the segmentation of interneuron videos that can then be easily tracked by existing software.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

Originally, we attempted to utilize a non-cell specific algorithm, idtrackerai, to track neuron migration. idtrackerai extracts the trajectories of individuals from videos of collectives of up to 100 individual animals [2]. The crucial advantage to their algorithm is that it is trained with a "protocol that adapts to video conditions and tracking difficulty" [2]. As a result, this platform held potential for handling the difficult nature of neuron migration videos. Unfortunately, it became apparent that there were not enough frames in which the neurons were not overlapping with each other for the algorithm to process correctly. Therefore, we chose to simplify the problem and instead tackle the first step of neuron nuclei tracking: segmentation. Within the last few years, neuron segmentation has been approached both with complex geometric methods along with deep learning.

Radojevíc and Meijering tackled automated neuron reconstruction from 3D fluorescence microscopy images, a problem very closely related to neuron segmentation [7]. They utilize sequential Monte Carlo estimation to perform probabilistic filtering and combine it with prediction and update models for a novel algorithm. They achieve performance under varying conditions similar to other state-of-the-art methods. This demonstrates the capacity of non-machine-learning models for neuron segmentation and reconstruction. In 2018, Zhang, Liu, Song, Feng, et. al. automated 3D soma segmentation by combining previous neuron reconstruction methods with complex geometry. This was a creative solution that successfully reduced the topological errors that previous methods present around the soma [10]. This work is more translatable to our problem because it directly focuses on soma segmentation, but critically is focused on 3D volumes and does not implement deep learning.

Aside from geometric or probabilistic models, Xu et. al implement a CNN with semi-supervised regularization to perform neuron segmentation in two-photon microscopy 3D volumes [9]. They generate putative labels for unlabeled samples so that the CNN can be trained on both labeled and unlabeled data, addressing the critical lack of labeled data that exists in this field. So, deep learning has been implemented to perform 3D neuron segmentation as well as to analyze neuron calcium imaging data, but this does not extend to 2D soma segmentation [5].

However, there are a multitude of publications focused on nuclei segmentation, a similar problem, the most renowned being UNet and DeepCell [6] [1]. We chose to utilize UNet architecture for our network because it has demonstrated high levels of performance on distinct biomedical image segmentations, and therefore could likely translate well to neuron microscopy. Additionally, the UNet is cleverly designed such that large datasets are not necessary to achieve high levels of performance. Overall, it is clear that 2D neuron tracking is a distinct unmet need in the biology research.

3 Dataset and Features

Our neuron-specific datasets and their corresponding ground truths are hand-labeled from the Pasca Lab videos of interneuron migration. We utilize ImageJ to individually crop 256 by 256 pixel images from the videos. After cropping these images, we then translated the images to png and RGB format to match the architecture.

To create the segmented images, we first set the threshold on the image to a maximum of 255 and a minimum of 35-135, depending on the noise present. Second, white pixels that do not represent nuclei (e.g. background noise or neuron axons) are manually set to black; this final ground truth image is saved as a png. In order to be given as ground truth to the CNN, the png has to be separated with only one soma segmented per image. This can be performed by duplicating the ground truth and painting over other nuclei in ImageJ to save pngs of individual soma masks.

Furthermore, Ronneberger et. al note that "data augmentation is essential to teach the network the desired invariance and robustness properties", and highly recommends mirror invariance alongside random elastic deformations [6]. Therefore, we augment our dataset by implementing elastic deformation (with an alpha of 34 and a sigma of 4) and mirroring. With augmentation, our dataset consists of 96 training examples.

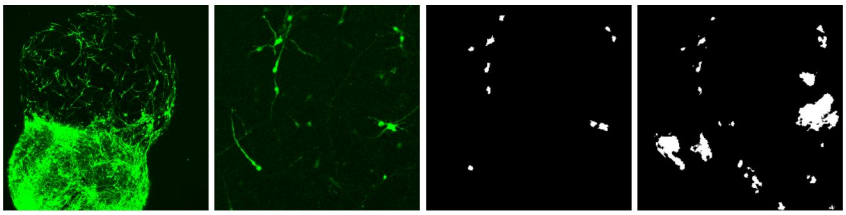

Figure 1: Interneuron migration in forebrain assembloid (left), cropped input image (middle left), ground truth mask (middle right), baseline predicted mask (right)

4 Methods

We utilize the source code from the UNet-pytorch project that was created as part of the Kaggle 2018 Data Science Bowl [4]. They implement the UNet architecture which serves as a basis for our cell segmentation problem. This architecture is a neural network which convolutes the input image and then deconvolutes into an image mask; it consists of combinations of 3x3 2D convolutions, batch normalization, ReLU layers, 2x2 max pooling layers, and transposed 2x2 2D convolutions (Figure 6). This allows end-to-end training, where the microscopy image is input and a segmentation map is output. We utilize binary cross entropy, defined below, as our loss function.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right]$$

We choose to train the network ourselves instead of downloading a pretrained network because it allows us more flexibility to manipulate both how we train the network and how we modify it for use on our own training set. This training was performed on an EC2 Deep Learning AMI (Ubuntu) 21.2 through Amazon Web Services and further training was performed locally on our devices.

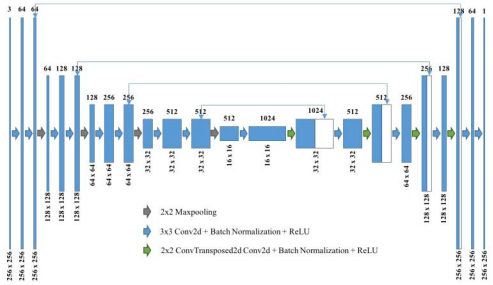

Figure 2: U-Net CNN Model Architecture [4]

We train on the 670 images from the Kaggle 2018 Data Science Bowl. We use the following parameters for training: batch size = 16, learning rate = 1e-3, weight decay = 1e-4. Using these parameters and 160 epochs, we obtain training loss of 0.01. We then test with a validation set of five images constructed by cropping images from the Pasca Lab videos. Resultant prediction masks show that this network is very sensitive to background noise and therefore does not perform well on our testing set (Figure 1).

Our next step was to perform transfer learning on the model using our hand-labeled neuron data. We choose to retrain the final three layers in order to retrain at least one deconvolutional layer. We use one image for the training set and the same one for the testing set to ensure that we could overfit on one image. We use these parameters: learning rate = 1e-4 and weight decay 1e-4 (batch size is irrelevant for a training sample set of size 1). Using these parameters and 150 epochs, we obtain training loss of 0.005. When predicting on this sample, we use intersection over union as the evaluation metric and obtain an IOU of 0.6. This clearly shows that we are capable of overfitting to one sample and that our model is functioning properly.

We then proceeded to retrain the final three layers of the network on our complete dataset, which consists of 32 hand-labeled images and masks, and evaluated on a validation set, which consists of 5

hand-labeled images and masks. This dataset consists of images from multiple types of microscopy: 20 images in the training set and the entire validation set are from the type of input we ultimately expect: RGB images with neurons labeled with green fluorescent protein (GFP). The remaining 12 images display interneurons from varying non-RGB microscopy images. We train with batch gradient descent and use a learning rate of 1e-4, weight decay 1e-4, and 100 epochs. This training achieves a final loss of 0.04.

5 Experiments/Results/Discussion

Once we exhibited that we were able to train our model, we focused on data augmentation and hyperparameter tuning. Our primary metric was IOU (intersection over union), which is defined as the ratio of the area of the intersection of the actual and predicted soma over the area of the union of these two somas. We also aimed to satisfy a precision goal of 80% and an accuracy goal of 90%, where precision is defined as the number of true predicted positives over true and false predicted positives and accuracy is defined as the number of true predicted positives over the number of predictions.

We introduced data augmentation as a key tool for reducing our training loss and improving these metrics. With data augmentation, we see our training loss decrease from 0.04 to 0.016, which shows significant improvement. All hyperparameter tuning is thus performed alongside data augmentation.

In hyperparameter tuning, we focused on changing the learning rate, weight decay, and batch size. While we did investigate the difference caused by using different numbers of epochs, we concluded that 50 epochs was enough to achieve a low training loss since the loss plots plateau early on. Furthermore, while we investigated retraining more layers, we found that it provided no significant value over retraining just three layers. Results are shown in the following table, and all loss plots are shown in the following graphs. We chose the hyperparameters for our final model by optimizing for validation IOU performance.

model	lr	batch size	epochs	decay	layers	loss	train IOU	val IOU
baseline	N/A	N/A	N/A	N/A	0	N/A	0.10	.07
1	1.00E-03	8	50	1.00E-04	3	0.016	0.39	0.50
2	1.00E-02	8	50	1.00E-04	3	0.017	0.17	0.25
3	1.00E-04	8	50	1.00E-04	3	0.018	0.25	0.34
4	1.00E-03	16	50	1.00E-04	3	0.027	0.22	0.32
5	1.00E-03	32	50	1.00E-04	3	0.028	0.24	0.37
6	1.00E-03	8	50	1.00E-03	3	0.028	0.22	0.34
7	1.00E-03	8	50	1.00E-05	3	0.027	0.25	0.34

Figure 3: Model performance comparison for hyperparameter tuning

We do not believe that we are overfitting to our training set since performance is higher on the validation set than the training set. This is not typical behaviour, but this is likely due to higher difficulty and variability in training set examples in comparison to validation set examples.

Overall, our results are satisfying since the final network has a high accuracy (94%), but we do hope to improve the precision to be above 80% and IOU above 60%. The majority of the error comes from small bits of noise that are segmented, rather than large masks. The input images themselves have high levels of noise, so we were anticipating this to translate to results as well. As for IOU, we believe that working to increase the size of segmented masks would be the most beneficial, since many of the predicted masks are smaller than their ground truth counterparts.

6 Conclusion/Future Work

This paper outlined the use of transfer learning on a UNet originally trained on non-neuron nucleus segmentation data. The highest performing algorithm involved transfer learning, data augmentation, and fine-tuning of hyperparameters. Transfer learning is necessary since the baseline model is not prepared to handle neuron microscopy. Data augmentation is also crucial since the training dataset is small and lacks robustness. Lastly, for the fine-tuning, a batch size of 8, learning rate of 1e-3, and

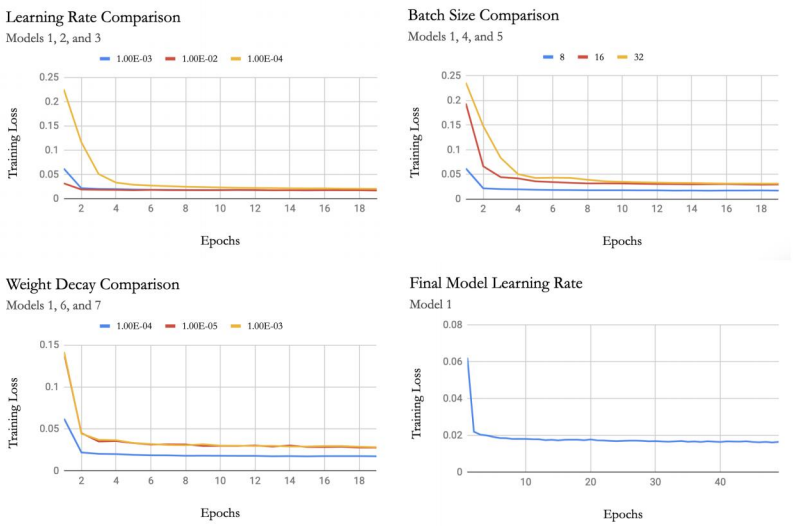

Figure 4: Training loss: learning rate comparison (top left), batch size comparison (top right), weight decay comparison (bottom left), final model (bottom right)

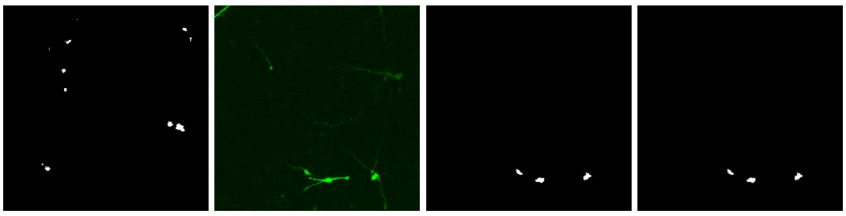

Figure 5: Validation results with model 1: predicted mask for input image in Figure 1 (left), input image (center left), ground truth mask (center right), predicted mask (right)

weight decay of 1e-4 resulted in the highest performing model. The smallest batch size performed best likely because the noise involved in batch gradient descent allows the training loss to escape saddle points. Ultimately, we achieve an IOU of 50%.

Additionally, it is important to acknowledge the inherent difficulties in this task. During interneuron migration, the soma and nucleus physically separate before the nucleus is pulled up to rejoin the soma, both of which appear similarly in microscopy and introduce a challenge to the network. The ground truth dataset is also subject to error since it is manually created and therefore there is variability between examples.

Moving forward, alongside improving the performance of this network, these are other important steps that we hope to implement.

1. Video Segmentation: Returning to the original motivation for this problem, we would like to extend our network to be able to track the neurons in a moving video. This involves segmenting videos of interneurons migrating in forebrain assembloids to visualize only the movement of the neuron somas. This step will simply involve adapting our current network to input and output a video instead of individual frames.
2. Tracking: By segmenting these videos, neurons no longer overlap with each other as frequently and are clearly defined. Therefore, we can return to our original approach for neuron tracking and utilize idtracker.ai to track these videos.
3. Data Processing: From the tracked videos, we then would like to analyze them to produce average velocity, direction, and saltation frequency.
4. Compare: We would like to compare the performance of this network to YOLO's performance for tracking [8].

7 Contributions

Julia works at the Pasca Lab and formulated the problem for this project. Both authors contributed to the code - Ashi adapted the original code repository to work with our inputs and outputs and implemented data augmentation, while Julia added IOU, added logging for comparisons of different implementations, and hand-labeled our data. Both authors experimented with hyperparameter tuning to adjust the model, and co-wrote the report and poster.

8 Acknowledgements

We would like to extend a special thanks to the Pasca Lab, and specifically Fikri Birey and Themasap Khan, for supplying neuron microscopy for us to use as training data, as well as the idtracker.ai authors who kindly responded to our questions early in the learning process.

9 Code

Our code may be found on GitHub at `https://github.com/ashi-agrawal/neuronal-migration`. We have used code from the following repository as a starting point: `https://github.com/limingwu8/UNet-pytorch`. Additionally, we adapted code for elastic deformation from the following repository: `https://github.com/developer0hye/Elastic-Distortion`.

References

[1] Keara M. Lane Derek N. Macklin Nicolas T. Quach Mialy M. DeFelice Inbal Maayan Yu Tanouchi Euan A. Ashley Markus W. Covert David A. Van Valen, Takamasa Kudo. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLOS Computational Biology*, 12:1–24, 11 2016.

[2] Robert C. Hinz1 Francisco J. H. Heras Gonzalo G. de Polavieja Francisco Romero-Ferrero, Mattia G. Bergomi. idtracker.ai: tracking all individuals in small of large collectives of unmarked animals. *Nature Methods*, 16(2):179–182, 2019.

[3] Booz Allen Hamilton. *Kaggle 2018 Data Science Bowl*. Available at `https://www.kaggle.com/c/data-science-bowl-2018/data`.

[4] Limingwu8. U-net pytorch github. Available at `https://github.com/limingwu8/UNet-pytorch`.

[5] Rob E. Aguilar1 Jan Homann Yi Gu David W. Tank H. Sebastian Seung Noah J. Apthorpe, Alexander J. Riordan. Automatic neuron detection in calcium imaging data using convolutional networks. *30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[6] Philipp Fischer Olaf Ronneberger and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Springer International Publishing Switzerland*, page 234–241, 2015.

[7] Miroslav Radojević and Erik Meijering. Automated neuron reconstruction from 3d fluorescence microscopy images using sequential monte carlo estimation. *Neuroinformatics*, pages 1–20, 2018.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] Kun Xu, Hang Su, Jun Zhu, Ji-Song Guan, and Bo Zhang. Neuron segmentation based on cnn with semi-supervised regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–28, 2016.

[10] Donghao Zhang, Siqi Liu, Yang Song, Dagan Feng, Hanchuan Peng, and Weidong Cai. Automated 3d soma segmentation with morphological surface evolution for neuron reconstruction. *Neuroinformatics*, pages 1–14, 2018.