Incorporate Deep-Transfer-Learning into Automatic 3D Neuron Tracing

Zhihao Zheng and Pengyu Hong
Computer Science, Brandeis University
Waltham, MA, U.S.A
Email: zhihaozh@brandeis.edu, hongpeng@brandeis.edu

Abstract — **Automated neuron tracing from microscopic images enables high-throughput quantitative analysis of neuronal morphology to elucidate functions of neural circuits. We have developed a transfer-learning approach that trains a deep convolutional neural network to trace neurons in 3D image stacks. Our neural network model consists of two major components. One is responsible for detecting foreground, the other takes the output of the first components and detect the central lines of neurites. They are trained sequentially, which is more efficient than training a whole deep neural network from scratch. The most spectacular aspect of our approach is that our training data is generated by synthesizing 2D simple lines in noisy backgrounds instead of consisting of manually labeled real neuron images which are labor intensive and time consuming to collect. Our method first processes each slices of 3D image, and then integrate them back to produce 3D tracing results. Preliminary test results show that the trained neuron tracer is capable of accurately tracing various types of neurons in noisy images.**

Keywords-Neuron Tracing; Convolutional Neuron Network; Neuron Tracing; Deep Learning, Transfer Learning.

I. INTRODUCTION

It is widely recognized that there is a strong connection between the morphological and functional properties of neurons. The analysis of neuron morphology can shed light on the functional bases of neural systems that consist of various types of neurons connecting with each other. With the rapid advances of imaging technologies, experimental neuroscientists are now able to quickly generate huge volume of 3D neuron images, which demands in-time analysis of neuron morphology. However, manual tracing of neurons in 3D images is time consuming, labor intensive, and often subjective. Hence, it is important to automate neuron tracing to generate accurate results. Digital reconstruction of neurons from microscopic images consists of several major tasks [1] , such as, soma segmentation, neurite tracing, spine segmentation, and so on. In this work, we focus on tracing neurites.

Many automated 3D neuron tracing algorithms have recently been developed [2]-[17] . They are in general capable of accurately tracing neurites. However, each of them relies on some pre-designed models to estimate certain parameters (e.g., foreground thresholds) from images, bridge gaps, or fit certain shape models (e.g., tubes, curves, etc.) to images. It is challenging to design a universal model for microscopic neuron images captured by different imaging instruments under a wide spectrum of setting. Hence, the designs of those models are often based on a small validation set, and can limit the generalization performance of the corresponding neuron tracing methods.

This work was motivated by the incredible capability of an average human being to trace the central lines of general curvature structures in various noisy backgrounds even though this individual never received any special training to perform such a task. We speculated that it might be possible to use simple synthetic stimuli to train a computational model to detect central lines, which can then be applied to trace complex neurite structures. The obvious advantage of this methodology is that we can avoid using intensive and subjective human labors to annotate training data. In this work, we explored the possibility of training the model by Deep Learning [18] and Transfer Learning [19] . We chose Convolutional Neural Network (CNN) [18] , which was biologically-inspired by the groundbreaking work of Hubel and Wiesel on visual cortex [20] , as the base of our Deep-Transfer-Learning network (DTL-NN) model. Our approach first trains a deep neural network to detect foregrounds and central lines in synthetic 2D images (ground truth is trivially known), and then refines the trained model by adding a hidden layer and using a small manually labeled real dataset to make it capable of accurately trace neurons imaged under the desired conditions. Given the 3D image stack of a neuron, we apply a trained DTL-NN to detect central lines of neurites in each 2D image slice. The detected 2D central lines are then pieced together to form the 3D structure of the neuron. Our methodology can not only be used to train neuron tracers, but also be used to build an expandable feature extractor for other complex computer vision problems.

The rest of the paper is organized as the following. In Section 2, we describe the structure of our neural network for foreground detection, centerline extraction and transfer learning. We also explain how to generate the synthetic training dataset, the training procedures, and the post-processing method. In Section 3, we show the experimental results of applying our approach to real datasets. Finally, the paper is concluded with discussions.

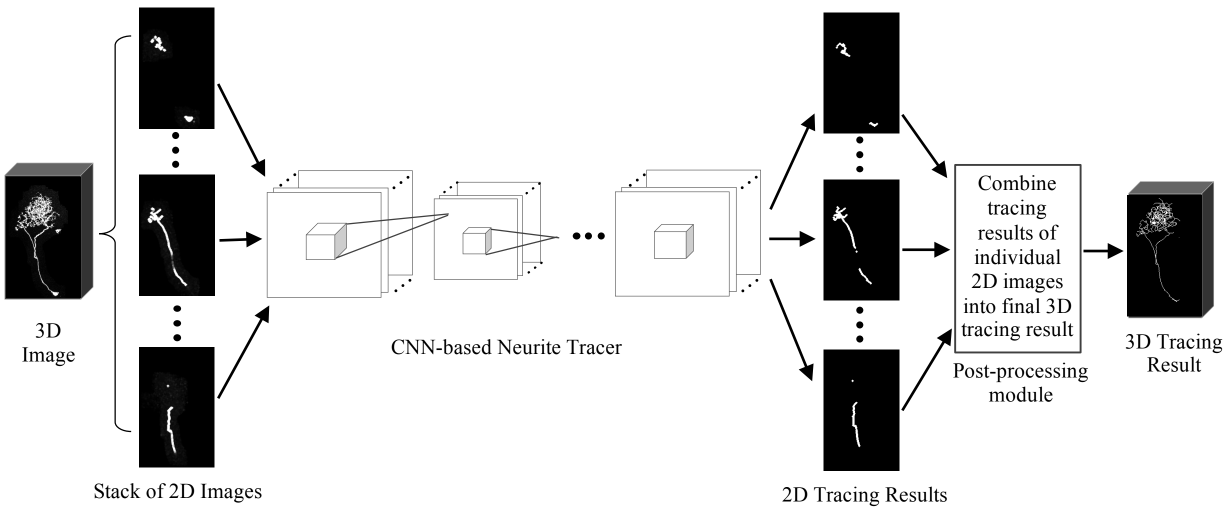

Figure 1. The pipeline of our neuron tracing approach.

II. Methods

This section describes the technical details of our Deep-Transfer-Learning neural network, the training data, and other key steps of our approach.

A. Overview

Our automated neuron tracing approach (Fig. 1) traces each individual image stack to obtain the intermediate tracing results, which are then combined together by a post-processing procedure to generate the final 3D tracing results. The tracing of individual image stacks is performed by our DTL-NN that consists of three main cascade components, which were trained to detect foreground, extract central lines, and adapt to real images, respectively. Below we explain in details how we effectively train the DTL-NN tracing model using synthetic data and transfer-learning, and how the intermediate tracing results are combined together to produce the final tracing results.

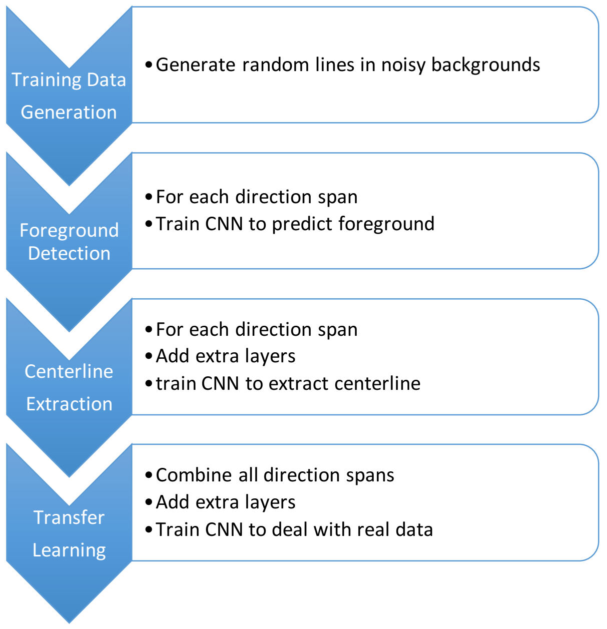
Training Data Generation
• Generate random lines in noisy backgrounds
Foreground Detection
• For each direction span
• Train CNN to predict foreground
Centerline Extraction
• For each direction span
• Add extra layers
• train CNN to extract centerline
Transfer Learning
• Combine all direction spans
• Add extra layers
• Train CNN to deal with real data

Figure 2. Training procedures of our neural network model.

B. Synthezie Training Data

It is well known that a large training data set of high quality is essential to obtain a superior machine-learning based model. For example, one of the driven forces behind recent striking advances in Computer Vision is high quality manually labelled training sets, such as, ImageNet [21] .

However, the amount of high-quality manually labeled neuron images is highly limited with respect to the almost infinite number of possible experimental conditions and subjects. To deal with this problem, we generated a large-scale training dataset by synthesizing a large number of lines in all directions with different widths and intensities in various noisy backgrounds. The ground truth of this dataset is obviously known. Some examples are shown in Fig. 3. Currently, we only consider lines as the basic structural elements of neurites. In the future, we can include more types of basic structural elements.

C. Train Foreground Detection Module

Foreground detection is a crucial step and can greatly affect downstream analysis. Most neuron tracing methods built their own model for detecting foregrounds manually

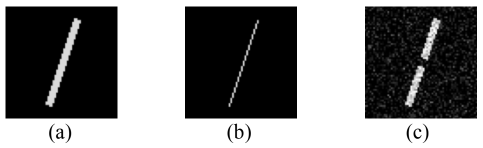

Figure 3. A synthetic training sample. (a) Foreground mask, (b) centerline mask, (c) synthesized image after adding noise to (a).

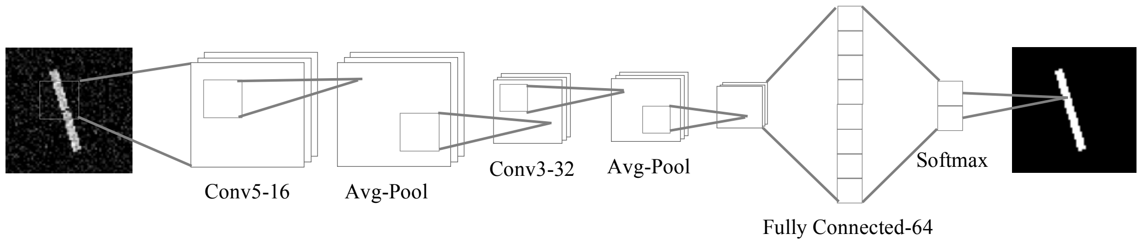

Figure 4. A trial CNN architecture for foreground detection.

(e.g., specify a relatively simple parametric form of the foreground detection models, and the model parameters are either fixed or can be adjusted based on local characteristics of images that can be calculated by some fixed rules). We would like to automatically learn a foreground detection model from data, which can learn to adjust itself to different imaging settings in the future. Initially, we built a large CNN (Fig. 4) for detecting foreground, which however worked relatively poor (Fig. 5b) especially in the areas around bright neurites. We hypothesized that the foreground detection results can be improved if the foreground detection model is able to take better advantage the local structural information, such as, directions. The model in Fig. 3 may be able to learn some local structural information, however, implicitly. In addition, mixing various structural information together makes learning more challenging (i.e., harder to converge to a better solution).

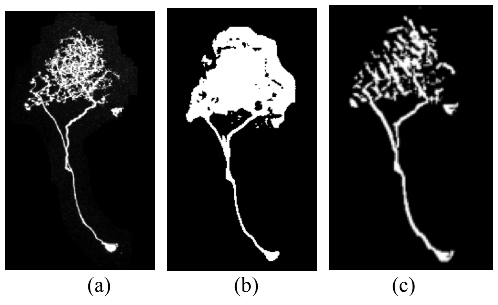

Figure 5. (a) Original image. (b) Foreground detected by the CNN designed in Fig. 4. (c) Foreground detected by the CNN design in Fig. 6.

Therefore, we redesigned our CNN-based foreground detection model (Fig. 6), and explicitly trained it to take advantage of local direction information. We divided all directions into six direction spans: [-22.5° 22.5°], [7.5° 52.5°], [37.5° 82.5°], [67.5° 112.5°], [97.5° 142.5°], and [127.5° 172.5°]. This design mimics the anatomy of the vision neural systems in carnivores and primates, in which neurons with similar direction preferences are clustered into radial columns and are organized in a systematic fashion across the V1 cortical surface [22] .

The synthetic training dataset was also divided into six subsets, one for each direction span. In addition, we design the CNN to consist of six columns, one for each direction span. Each column was pre-trained by using the training subset of the same direction span so that a trained column only responds to the directions within its chosen direction span. All neural network columns were then assembled into one CNN that was fine-tuned using all training data. This

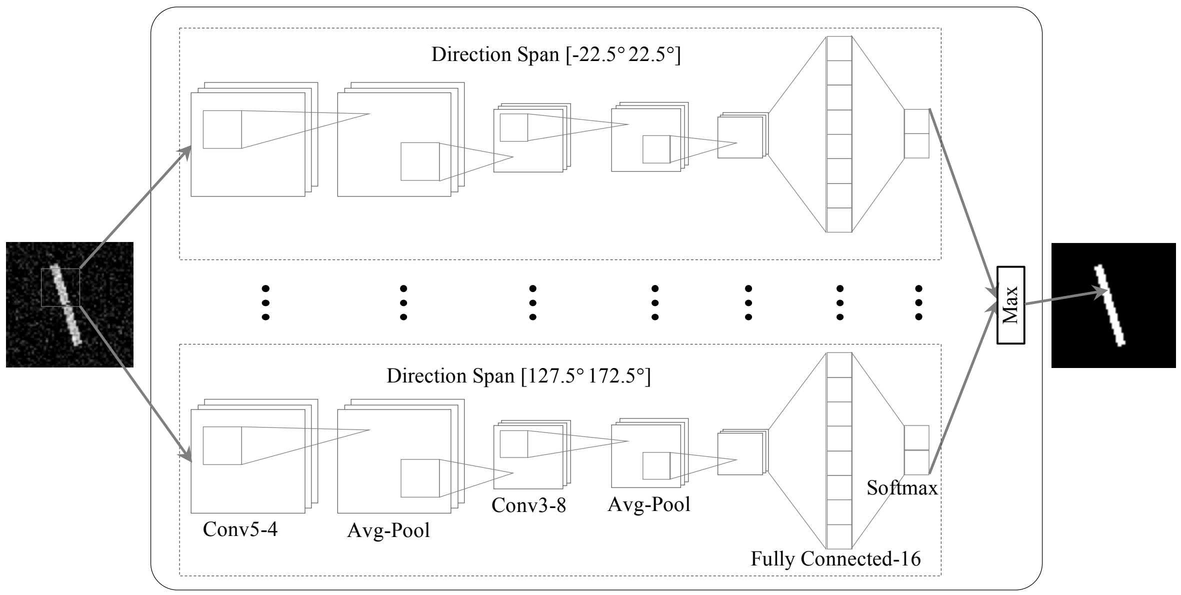

Figure 6. Improved CNN architecture of the foreground detection module.

arrangement sped up the training the whole CNN and produced significantly better results (Fig. 5c). The choice of the spans is based on the window size of our convolution. Since the window size of our first convolution layer is 5 x 5, the minimum angular change it can capture is around 27°. Thus we separated the direction spans by 30° apart. Each direction span covers 45° so that neighboring direction spans have 15° overlap. This non-exclusive design allows the model to better detect lines close to the boundaries of the direction spans.

D. *Train Centerline Extraction Module*

After obtaining a robust foreground detection module, we trained a centerline extraction module which takes the output of the second-to-the-last layer of the foreground detection module and outputs the corresponding centerline. Basically, we considered the foreground detection module as the feature extractor that learns the intermediate representations of line structures for the centerline extraction module. A fully connected layer was added between the FC-16 and Softmax layers of the foreground detection module (Fig. 7). This modified network was trained to output the centerlines of line structures. We found that it was easier to first train a foreground detection module and then insert a centerline extraction module than to train one big neural network to directly extract centerlines. Comparing the neural network weights of some neurons in the foreground detection module shows that their weights share similar patterns before and after being trained to extract centerlines (Fig. 8). This signals that the neurons in the foreground deteciton module have been appropriately trained during the training step of the foreground detection module.

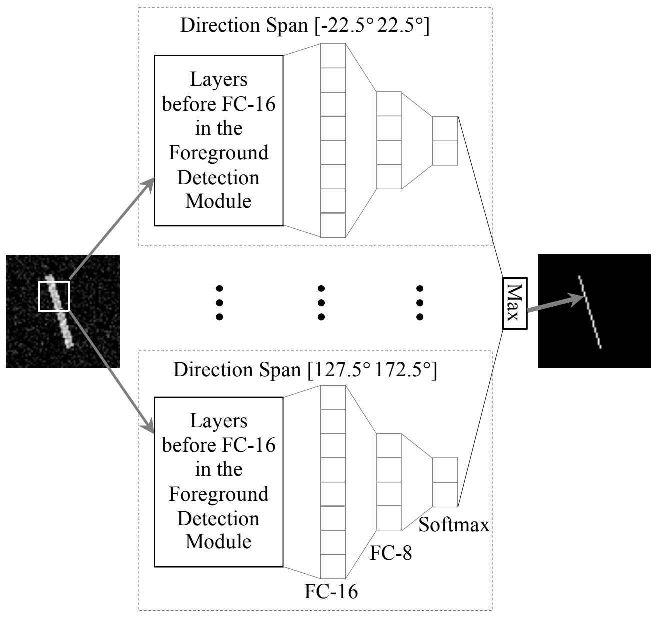
Direction Span [-22.5° 22.5°]
Layers before FC-16 in the Foreground Detection Module
Max
Direction Span [127.5° 172.5°]
Layers before FC-16 in the Foreground Detection Module
Softmax
FC-8
FC-16

Figure 7. Centerline extraction neural network module.

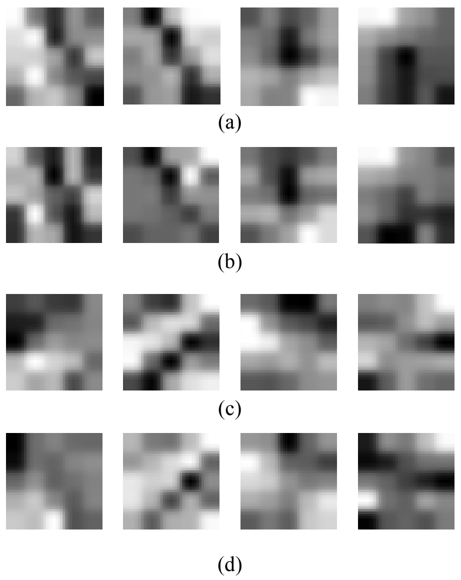

Figure 8. (a) & (c) Weights of a few neurons in the foreground detection CNN responsible for 30° and 150°, respectively. (b) & (d) Weights of the neurons in the centerline extraction CNN corresponding to the neurons in (a) & (c).

E. *Adaptation to Real Images and Post-Processing*

Previous steps give us a deep neural network that is able to extract central lines of synthetic line structures in noisy backgrounds. More importantly, the deep neural network has learned internal representations for describing various line structures and their centerlines, which can also be very useful, although not perfect, for representing curvature structures in real images. However, real images can have distributions quite different from those in our synthetic training data. To overcome the differences between synthetic images and real images, we apply transfer learning to adapt the trained network to real data by using a small amount of manually labelled real data that are much easier to obtain. In doing this, we use a hidden layer (FC-24) to connect the FC-8 layers of the Centerline Extraction CNN to a Softmax output layer (Fig. 9). This allows us to transfer the knowledge learnt from synthetic dataset to trace neurites in

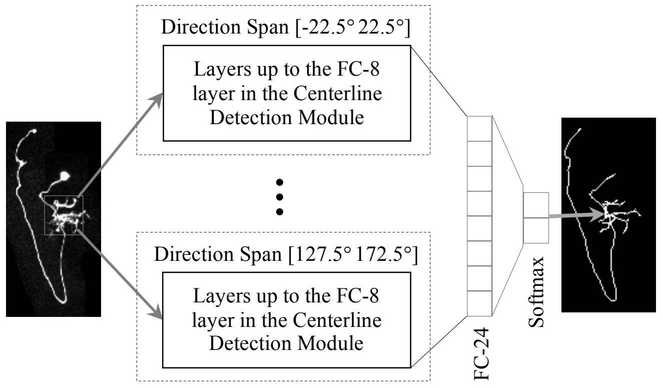
Direction Span [-22.5° 22.5°]
Layers up to the FC-8 layer in the Centerline Detection Module
Direction Span [127.5° 172.5°]
Layers up to the FC-8 layer in the Centerline Detection Module
FC-24
Softmax

Figure 9. Deep-Transfer-Learning neural network architecture.

real images. The new FC-24 and Softmax layers learns how to utilize the representations learned from synthetic lines and adapt them into new internal representations for processing real data.

Although the model is trained to output 2D centerlines, its outputs are more similar to a shrunk foreground because the model relies on information in local patches and the outputs are softmax results. There is no constraint to force the model to output centerlines with width of exact one. To obtain 3D tracing results, we developed a post-processing module to combine the 2D tracing results of the individual stacks of a 3D image into a 3D tracing result. The post-processing module mainly includes two steps: (a) link the 2D tracing results across stacks based on minimum spanning tree to obtain a 3D map; and (b) apply thinning to the binary 3D map to obtain the final 3D neurite tracing result.

III. Experimental Results

We tested our automated neuron tracing methods on a dataset containing 23 Drosophila neurons provided by the BigNeuron project [23][24] . Typical tracing results (Fig. 10 & 11) show that our model is able to accurately tracing neurons in real 3D microscopic images although it was trained primarily using synthetic data. Our DTL-NN is able to transfer knowledge from synthetic data into real images by adding only one extra hidden layer. The number of parameters added to perform the transfer learning is extremely small (only ~1200 parameters). Hence training of the DTL-NN can be done efficiently.

Table 1 shows the test results computed as the average and standard deviation of pairwise distance from the gold results to our detection results, and pairwise distance from our detection results to the gold results. The errors are mainly caused by the following reasons. First, the resolution sensitivity is reduced by the average pooling layers within the network. Second, some real data contain noise much stronger than what was used in training the network, or some neurites in real data are extremely faint. This led to false positives (i.e., falsely detected neurites) and false negatives (i.e., missed neurites), and hence dramatically increased the detection error. A large-scale experiment is being carried out to thoroughly test this approach.

TABLE I. Pairwise Distance with Gold Results

Pair	**Pairwise Distance**	
	Average	***Standard Deviation***
Gold to Predict	1.363	1.421
Predict to Gold	1.377	1.539

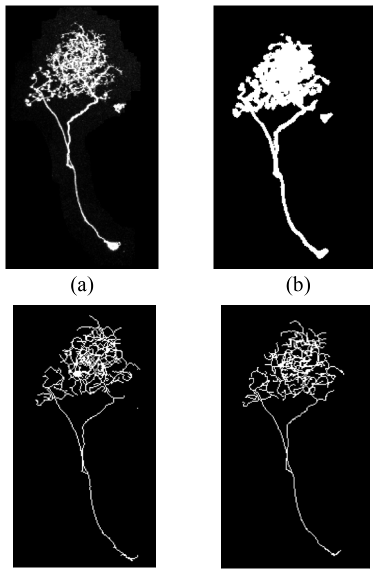
(a) (b) (c) (d)

Figure 10. A tracing example. (a) Original image. (b) Centerline extraction result. (c) Post-processed result. (d) Manually labeled result.

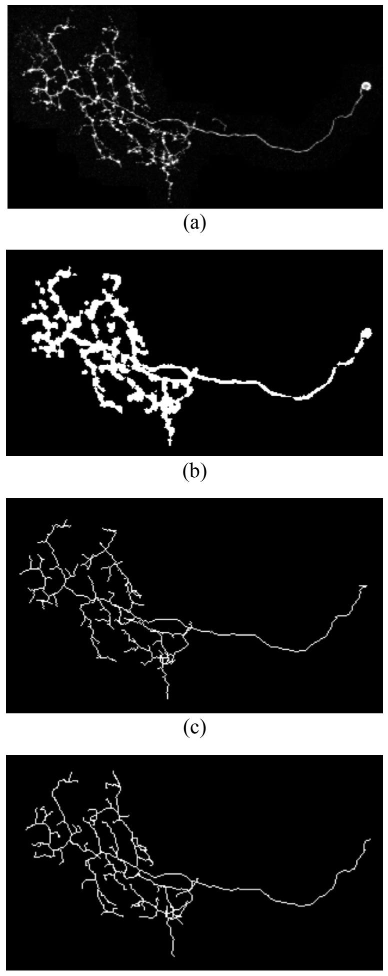
(a) (b) (c) (d)

Figure 11. Another tracing example. (a) Original image. (b) Centerline extraction result. (c) Post-processed result. (d) Manually labeled result.

IV. CONCLUSION AND FUTURE WORK

In this paper, we proposed a Deep-Transfer-Learning neural network which is able to learn essential features from synthetic lines and transfers the learnt knowledge to process real neuron images. One major advantage of our approach is that it does not requires a large amount of manually labeled training data. Currently, our approach trains the model to work on 2D images, and use a post-processing step to obtain the final 3D tracing results. We plan to design and train a DTL-NN to directly processes 3D images rather than process each slice, such that we may obtain more accurate 3D features from images than this network. We will also try to design a network, which can involve global information of the image, to further improve our results. More extensive validation tests of our approach will be carried out.

REFERENCES

[1] E. Meijering, "Neuron tracing in perspective". Cytometry A, 77(7): pp. 693-704. 2010.

[2] K. A. Al-Kofahi, et al., "Rapid automated three-dimensional tracing of neurons from confocal image stacks". IEEE Trans Inf Technol Biomed, 6(2): pp. 171-87. 2002.

[3] S. Schmitt, J. F. Evers, C. Duch, M. Scholz, and K. Obermayer, "New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks". Neuroimage, 23(4): pp. 1283-98. 2004.

[4] Y. Al-Kofahi, et al., "Improved detection of branching points in algorithms for automated neuron tracing from 3D confocal images". Cytometry A, 73(1): pp. 36-43. 2008.

[5] B. E. Losavio, et al., "Live neuron morphology automatically reconstructed from multiphoton and confocal imaging data". J Neurophysiol, 100(4): pp. 2422-9. 2008.

[6] H. Peng, Z. Ruan, D. Atasoy, and S. Sternson, "Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model". Bioinformatics, 26(12): pp. i38-46. 2010.

[7] J. Xie, T. Zhao, T. Lee, E. Myers, and H. Peng, "Anisotropic path searching for automatic neuron reconstruction". Med Image Anal, 15(5): pp. 680-9. 2011.

[8] H. Xiao, and H. Peng, "APP2: automatic tracing of 3D neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree". Bioinformatics, 29(11): pp. 1448-54. 2013.

[9] E. Turetken, G. Gonzalez, C. Blum, and P. Fua, "Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors". Neuroinformatics, 9(2-3): pp. 279-302. 2011.

[10] Y. Wang, A. Narayanaswamy, C. L. Tsai, and B. Roysam, "A broadly applicable 3-D neuron tracing method based on open-curve snake". Neuroinformatics, 9(2-3): pp. 193-217. 2011.

[11] A. Choromanska, S. F. Chang, and R. Yuste, "Automatic reconstruction of neural morphologies with multi-scale tracking". Front Neural Circuits, 6: pp. 25. 2012.

[12] X. Ming, et al., "Rapid reconstruction of 3D neuronal morphology from light microscopy images with augmented rayburst sampling". PLoS One, 8(12): pp. e84557. 2013.

[13] S. Basu, B. Condron, A. Aksel, and S. Acton, "Segmentation and tracing of single neurons from 3D confocal microscope images". IEEE J Biomed Health Inform, 17(2): pp. 319-35. 2013.

[14] J. Yang, P. T. Gonzalez-Bellido, and H. Peng, "A distance-field based automatic neuron tracing method". BMC Bioinformatics, 14: pp. 93. 2013.

[15] A. Rodriguez, D. B. Ehlenberger, P. R. Hof, and S. L. Wearne, "Three-dimensional neuron tracing by voxel scooping". J Neurosci Methods, 184(1): pp. 169-75. 2009.

[16] Z. Zhou, X. Liu, B. Long, and H. Peng, "TReMAP: Automatic 3D Neuron Reconstruction Based on Tracing, Reverse Mapping and Assembling of 2D Projections". Neuroinformatics, 14(1): pp. 41-50. 2016.

[17] S. Mukherjee, S. Basu, B. Condron, and S. T. Acton. "Tree2Tree2: Neuron tracing in 3D". in *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE. pp. 448-451. 2013.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning". Nature, 521(7553): pp. 436-44. 2015.

[19] L. Y. Pratt, S. Hanson, C. Giles, and J. Cowan, "Discriminability-based transfer between neural networks". Advances in neural information processing systems: pp. 204-204. 1993.

[20] D. H. Hubel, and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex". J Physiol, 195(1): pp. 215-43. 1968.

[21] J. Deng, et al. "Imagenet: A large-scale hierarchical image database". in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. pp. 248-255. 2009.

[22] D. H. Hubel, and T. N. Wiesel, "Brain and visual perception : the story of a 25-year collaboration". New York, NY: Oxford University Press, 2005.

[23] A. Jenett, et al., "A GAL4-driver line resource for Drosophila neurobiology". Cell Rep, 2(4): pp. 991-1001. 2012.

[24] H. Peng, et al., "BigNeuron: Large-Scale 3D Neuron Reconstruction from Optical Microscopy Images". Neuron, 87(2): pp. 252-6. 2015.