DeepNeuron: An Open Deep Learning Toolbox for Neuron Tracing

Zhi Zhou, Hsien-Chi Kuo, Hanchuan Peng*, Fuhui Long*

Allen Institute for Brain Science, Seattle, USA

* Corresponding authors: Hanchuan Peng <hanchuanp@alleninstitute.org>
Fuhui Long <fuhuil@alleninstitute.org>

Abstract

Reconstructing three-dimensional (3D) morphology of neurons is essential to understanding brain structures and functions. Over the past decades, a number of neuron tracing tools including manual, semi-automatic, and fully automatic approaches were developed to extract and analyze 3D neuronal structures. Nevertheless, most of them were developed with conventional techniques and yield limited performance given the complicated nature of neuronal structures. Recently, Deep Learning has been shown to outperform traditional methods in a wide range of image analysis and computer vison applications. Here we developed a new open source tracing toolbox, *DeepNeuron*, which utilizes Deep Learning networks to trace neurons in light microscopy images. DeepNeuron provides a family of modules to solve basic yet strenuous problems in neuron tracing, including detecting neurites under different image conditions, locally connecting continuous neuronal signals, pruning and refining automatic tracing results, quantitatively evaluating manual reconstruction quality, and real-time classification of dendrites and axons. We have tested DeepNeuron using challenging light microscopy images including bright-field and confocal images of human and whole mouse brain, on which DeepNeuron demonstrates very good robustness and accuracy in neuron tracing.

Introduction

Over the past few decades, researchers have developed algorithms and tools to extract 3D neuronal morphology. A number of manual/semi-automatic neuron tracing platforms including public-domain and commercial software packages have been developed[1-8]. These platforms allow users to reconstruct 3D structures of neurons but are usually time-consuming. To promote the development of efficient neuron tracing tools, DIADEM challenge [9] and BigNeuron project [10] have been launched to benchmark different automated algorithms. Whereas most algorithms could generate meaningful reconstructions on high quality neuron images in small or medium scales, UltraTracer [11] provides an efficient framework to trace large-scale images. Despite these efforts, how to precisely reconstruct neuronal morphology from challenging neuron images remains an open question.

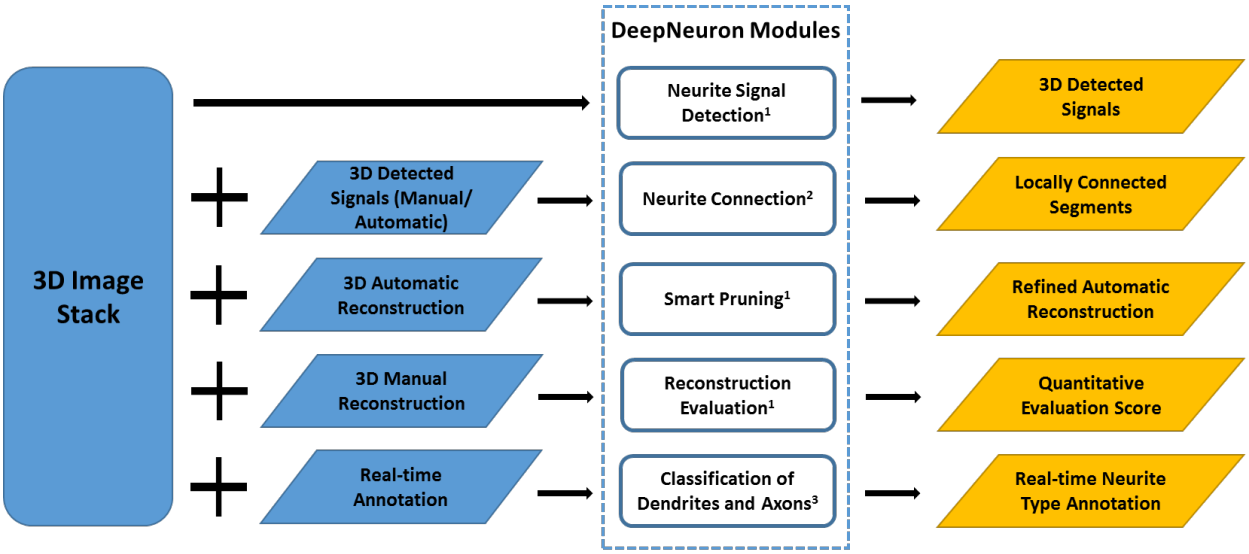
Starting from a cell body, a high-performance neuron tracing tool should follow all the dendrites and axons and eventually formulate a connected single tree that precisely represents the morphology of the cell. In neuronal images, dendrites are shown in thicker and more continuous signals in general, whereas axons are often accompanied with certain issues which make them more difficult to trace. First, due to dissection and slicing of brain specimen, axons often appear to be punctuated segments and not fully connected in a 3D image stack. This can be seen in bright-field image of biocytin-labeled neurons (Allen Cell Type Database [12]). Second, the image quality varies a lot depending on sample preparation, imaging process, cell types and the healthiness of neurons. As a result, axons could be continuous in one image, but dim and discrete in another. It is difficult to automatically extract them under all different conditions.

Third, following the complete trajectory of axons in an image stack with uneven background is also challenging. In a whole mouse brain image, thousands of smaller blocks need to be imaged first and then stitched together to produce a high resolution 3D image stack. Due to the different image time spent on each block, the background of the final stitched image stack can be highly inconsistent. To completely trace a neuron, annotators have to constantly adjust the contrast of every local region so that they can follow the axon across the entire brain. At this moment, this process cannot be fully automated since local nerite signals are hard to be adaptively enhanced and precisely extracted.

Taking neuron tracing as a classification problem, a very essential step is to detect neurite signals from background (foreground/background classification), and then reconstruct the topological structures by connecting neurite segments (connection/separation classification). Since Deep Learning methods have been widely used in pattern recognition and computer vision areas nowadays [13], a number of machine learning/Deep Learning based approaches trained with human-annotated dataset are able to generate more accurate neuronal morphology. For example, Gala et al. introduce an active learning model by combining different features to automatically trace neurites [14]. Chen et al. propose a self-learning based tracing approach, which does not require substantial human annotations [15]. Fakhry et al. [16] and Li et al. [17] use Deep Learning neural networks to segment electron and light microscopy neuron images. While the performance of these methods for neurite (e.g. axons) signal detection remains debatable, one important limitation is that none of these Deep Learning based methods has been publicly available for testing neuron image data.

We analyzed many commonly used neuron tracing and editing functionalities in different real applications and concluded that an open source Deep Learning toolbox could be an important contribution to this growing field. Using classification models trained by Deep Learning neural networks, we present DeepNeuron which provides essential functions to neuron tracing. In automated tracing, it can be used as a new tracing algorithm to reconstruct neurites from noisy neuron images, or an extra processing component to improve the performance of existing tracing algorithms. In manual tracing, it can be used as an assistant to help annotators efficiently and confidently trace neuronal structures. We discuss five distinct yet complementary modules that would be applicable to multiple tasks in computational neuromorphometry (Figure 1):

- **Neurite signal detection:** automatically identify 3D dendritic and axonal signals from background.
- **Neurite connection:** automatically connect local neurite signals.
- **Smart pruning:** filter false positive and provide refined automated reconstruction results.
- **Reconstruction evaluation:** evaluate and provide quality scores on neuron reconstructions.
- **Classification of dendrites and axons:** automatically assign neurite types during the real-time annotation.



[1]Convolutional neural network is used for foreground and background classification.
[2]Convolutional neural network is used to learn the similarity between two patches.
[3]Convolutional neural network is used for dendrite, axon, and background classification.

Figure 1. The overall workflow of the Open Source DeepNeuron toolbox. Here we list five deep learning based neuron tracing modules.

Results

Neurite Signal Detection

If neurites are not continuous and only shown in dots-like structures, it is hard to use any existing automated tracing algorithm to reconstruct. Even for human annotators, locating these isolated axon signals from the noisy background is a tedious work. We first introduce neurite signal detection module, which uses the convolutional neural network (CNN) based approach to detect neurite signals. Using a classification model, neurite signals can be precisely detected without any pre-processing step on the original image. To speed up the detection procedure, we employed the strategy of using 2D CNN model followed by 3D mapping, which showed satisfactory results on our testing data. This working scheme is not limited to 2D; 3D CNN model could as well be applied in more challenging scenarios.

Table 1 shows the five-fold cross-validation test of the model robustness. The training image sample (methods) is partitioned into five equal size subsamples. A single subsamples is used for testing, and the remaining four subsamples are used as training data. In our case, five single subsamples are neurons 1-24, 25-48, 49-72, 73-96, and 97-122. Both foreground and background can achieve > 98% accuracy for both training and testing images.

Table 1. Five-fold cross-validation on bright-field training sets

Trained Model	Foreground Accuracy		Background Accuracy	
	Training images	Testing images	Training images	Testing images
{1-122}\{1-24}	98.77%	97.78%	98.97%	96.87%
{1-122}\{25-48}	98.54%	99.07%	98.78%	98.34%
{1-122}\{49-72}	98.67%	98.28%	98.78%	99.13%
{1-122}\{73-96}	98.71%	96.64%	98.28%	99.23%
{1-122}\{97-122}	98.64%	99.02%	98.77%	98.41%
Average	**98.67%**	**98.08%**	**98.83%**	**98.44%**

We applied our module to two challenging datasets. The first set is a bright-field dataset. The second set is a whole mouse brain data imaged by fMOST imaging technology [18]. Figure 2 shows two examples of axon detection results. Using neurite signal detection module, most of axon signals have been precisely detected in both two datasets.

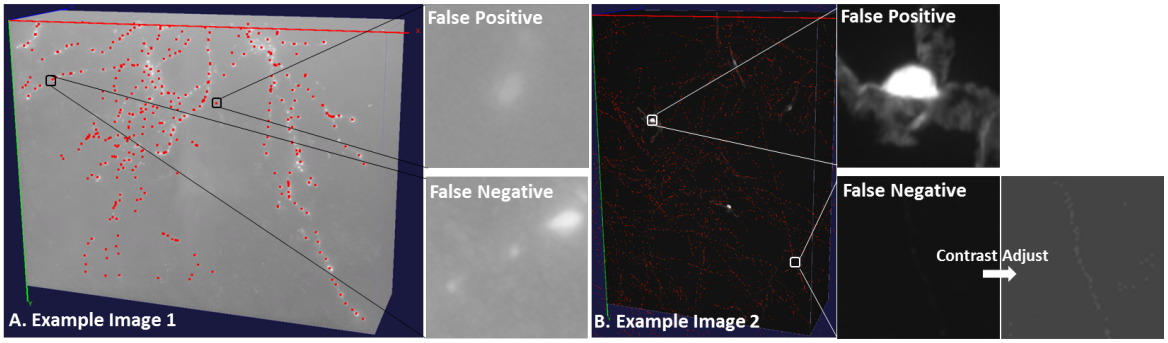


Figure 2. Axon detection results on two challenging datasets. A) The image stack is a cropped 3D bright-field image of a biocytin-labeled mouse neuron; the pixel resolution is 0.14 um x 0.14 um x 0.28 um. B) The image stack is a cropped 3D stack from a whole mouse brain data; the pixel resolution is 0.3 um x 0.3 um x 1 um. Red dots indicate 3D detected axonal signals. The two false positive example patches shown in A and B could be eliminated with more comprehensive background training samples. Those false negative patches (also as shown in A and B) with very weak signals at the center could be further identified by adjusting the probability threshold in the classification model or increasing the amount of these types of foreground in the training set.

Neurite Connection

Global, local, and topological features including total length, bifurcations, terminal tips are used to study the neuronal morphology. These features have to be extracted from neurite segments instead of dots. Therefore, finding the continuity of neurites and connecting them is a critical step in neuron tracing. Generally, automated tracing algorithms can achieve good performance on connecting neurites or filling small gaps based on the orientation of neurite segments. However, it is difficult to automatically connect dots-like neurite signals. Taking the spatial distance between these signals as the weight, Minimal Spanning Tree (MST) is an option for the connection. But without biological context, it could also introduce topological errors. Human beings are good at finding the continuity of isolated signals per their observation and domain knowledge. By learning the neurite connectivity from a large dataset annotated

by humans, we introduce DMST (Deep Learning based MST) to locally connect signals with the nature of neurite continuity considered (Figure 3).

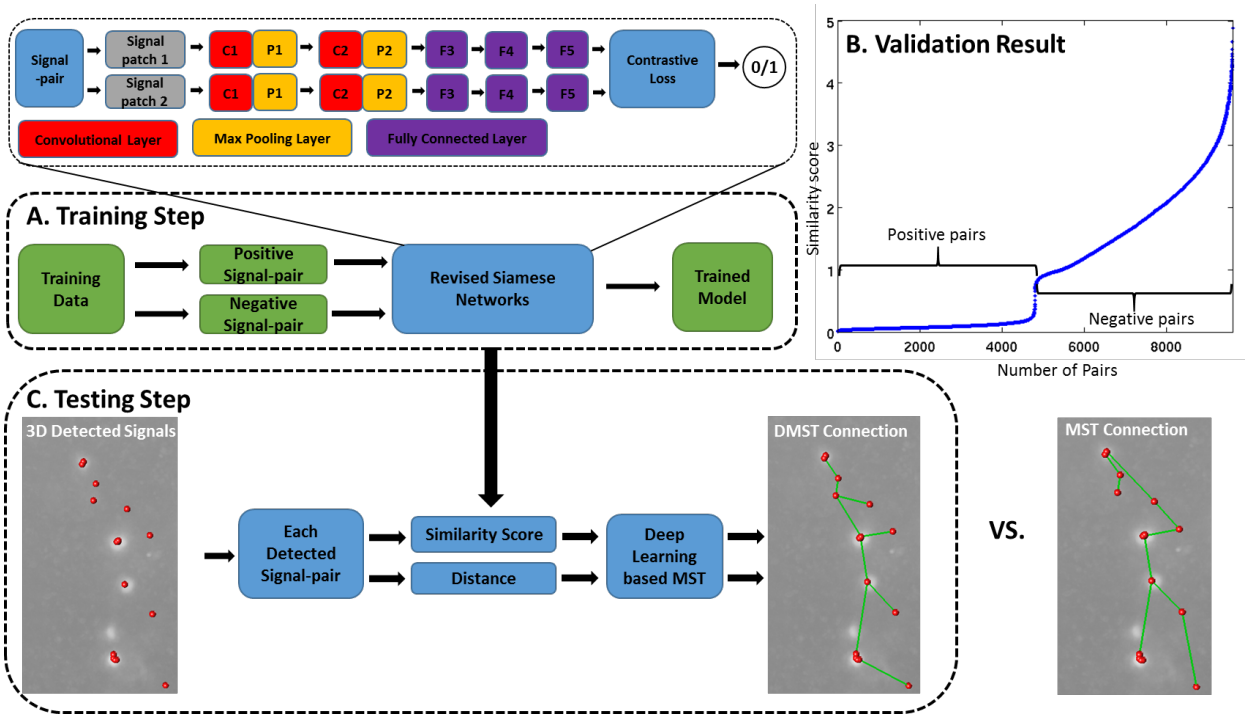


Figure 3. The workflow of neurite connection module. A) In the training step, the connectivity of a signal pair is learned using revised Siamese networks [19, 20]. In each pair, a 1x200 feature vector is extracted. B) The validated result with sorted similarity scores for all signal pairs show that the similarity scores of positive pairs are much lower than that of negative pairs. C) The trained model is applied in the testing step to calculate the similarity for each detected signal pair. The comparison result of our proposed DMST connection and the original MST connection (using distance as the weight only) is shown in the right-bottom corner.

We combined neurite signal detection and neurite connection modules to automatically reconstruct axon signals on bright-filed neuron images (Figure 4).

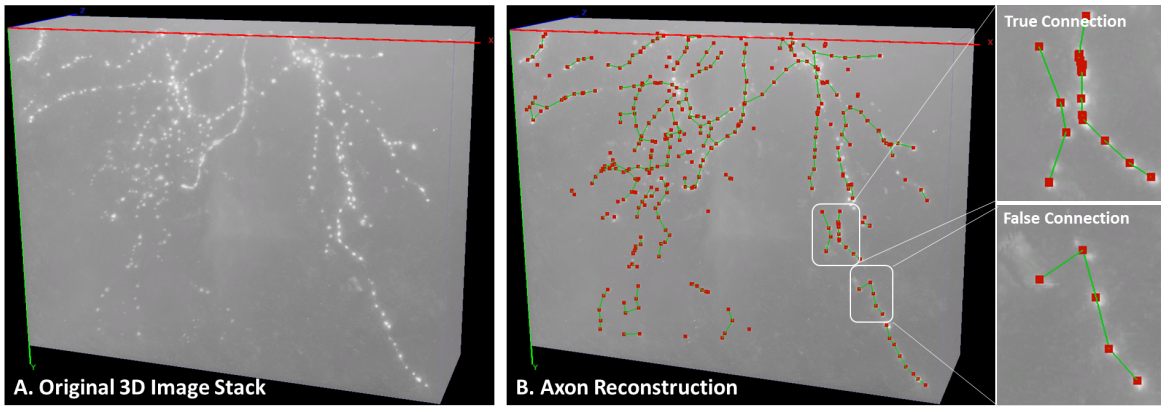


Figure 4. DeepNeuron on axon reconstruction. A) The same example image as shown in Figure 2A. B) 3D detected axon signals are shown in red dots by neurite signal detection module; green lines indicating local 3D connections by neurite connection module. As shown in B, the proposed DMST in the connection module can formulate the correct neurite structure even though the detected signals from different neurites are spatially close to one another. In a more extreme case, however, the absence of true signal in the local area can make the information captured by the network outweighed by the distance, resulting in false connection also as shown in B. This could be avoided by adequately enlarging the examined local area, or creating one more false connection class with signal-noise pairs to reinforce the network's knowledge. Note, unconnected fragments like those in the bottom-left would be further processed by our non-deep-learning functions, which are not in the scope of this article.

Smart Pruning

One of the key issues in automatic tracing algorithms is to determine the background threshold. To obtain better tracing results, this threshold input could be manually adjusted but with the efficiency being trade off. To automatically define a threshold, some algorithms use the weighted average intensity of the entire image or add a pre-processing step to enhance signals. However, choosing a global threshold or adding an enhancement step is not an optimal solution for neuron image with uneven background. DeepNeuron provides a smart pruning module, which works as an additional component to improve the tracing performance on existing automatic results.

For any individual tracing algorithm, a lower threshold could be conservatively set first to generate a over-traced result. For some algorithms which do not require the threshold input, the original image can be enhanced first to be able to produce similar over-traced results. Using our smart pruning model, all falsely detected annotation nodes (methods) in these results will be filtered out based on our classification model. Different tracing results from multiple algorithms could also be post-processed together to form a consensus using this module (Figure 5).

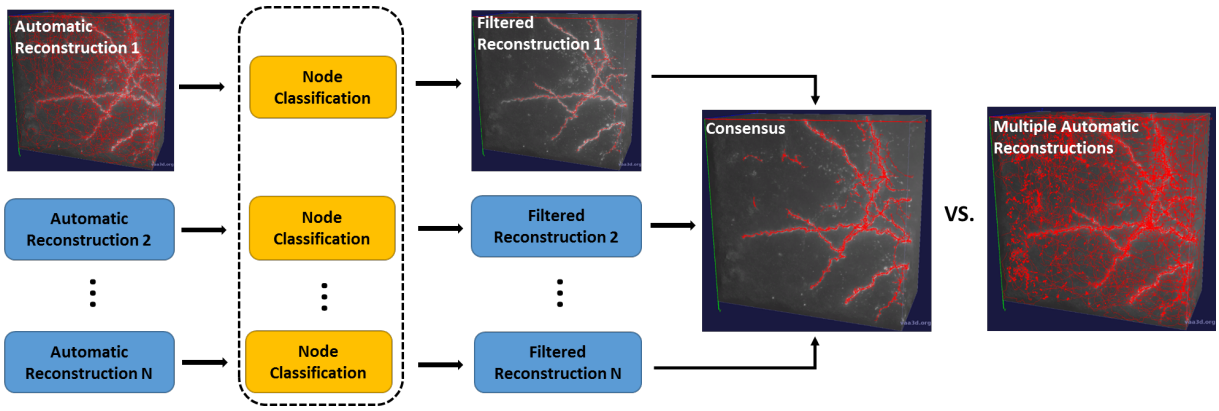


Figure 5. The workflow of consensus generation using the smart pruning module. Multiple automatic reconstructions are filtered by CNN based node classification (section 2.1) first. Then all filtered reconstructions are fused together to form a consensus. Reconstructions are shown in red lines on top of the original image stack.

Reconstruction Evaluation

Evaluating the manual reconstruction is important to validate the tracing performance. Cross-checking the result by multiple annotators is a standard but also time-consuming procedure. In order to efficiently evaluate the quality of reconstructions, DeepNeuron provides an evaluation module.

Table 2 shows the five-fold cross-validation results on 122 bright-field training images. From the table we can see, quality scores of thicker and more continuous dendrites (>99%) are higher than that of axons (>96%), which are dim and discontinuous.

Table 2. Five-fold cross-validation of quality evaluation scores

Trained Model	Reconstruction Quality Score					
	Training images			Testing images		
	Axon	Dendrite	Overall	Axon	Dendrite	Overall
{1-122}\{1-24}	97.02%	99.35%	98.45%	95.90%	99.74%	98.30%
{1-122}\{25-48}	96.69%	99.68%	98.45%	97.51%	98.07%	97.94%
{1-122}\{49-72}	97.14%	99.31%	98.52%	95.66%	99.79%	97.95%
{1-122}\{73-96}	96.67%	99.34%	98.41%	96.48%	99.58%	98.02%
{1-122}\{97-122}	96.82%	99.39%	98.36%	98.07%	99.61%	99.15%
Average	**96.87%**	**99.41%**	**98.45%**	**96.47%**	**99.34%**	**98.25%**

We applied our evaluation module to 31 bright-field testing image stacks including 10 human neurons and 21 mouse neurons in the Allen Cell Type Database. Table 3 shows comparison results, which are consistent with Table 2: dendrites have higher scores than axons. In addition, higher scores on human neurites (axons and dendrites) indicate that annotators have better tracing performance on physically larger human neurons.

Table 3. Comparison results of quality scores on human and mouse neuron images

	Number	Axon	Dendrite	Full
Human	10	98.23%	99.60%	98.93%
Mouse	21	94.28%	99.40%	95.38%

Depending on the stage of the input manual reconstruction, the evaluation score could be a different indicator. If the reconstruction is from an initial manual curated result, this score can be used to provide a confident level of the reconstruction. If the reconstruction is from a final cross-checked result, this score also can be used to assess the quality of input neuron image including the weakness and discontinuity of neurites.

Classification of Dendrites and Axons

Dendrites and axons have their own functions and play different roles in the nervous system. Due to their very similar appearance, it is difficult to automate the type assignment in most of tracing algorithms. Taking the advantage of Deep Learning neuronal networks in classification model, we present a module to automatically classify dendrites and axons on real-time annotation.

Figure 6 shows the classification performance on two testing examples. The neurite typing performance depends on how sophisticated the chosen network is. For instance, AlexNet [21] with both five and six convolutional layers can accurately classify continuous dendritic and discrete axonal signals in Figure 6A. However, when the signal of axons is similar to dendrites (Figure 6B), five-layer AlexNet mistakenly classified axons into dendrites, while six-layer AlexNet successfully distinguished axons from dendrites. We also compared the classification accuracy of these two AlexyNet models on training samples (Table 3). The model with five layers is more efficient (3.27s forward-backward time) than using six layers (335.91s forward-backward time), but using six layers yields much better performance of classifying axons and dendrites. On more challenging datasets, users could use trained models with even deeper layers for better classification results.

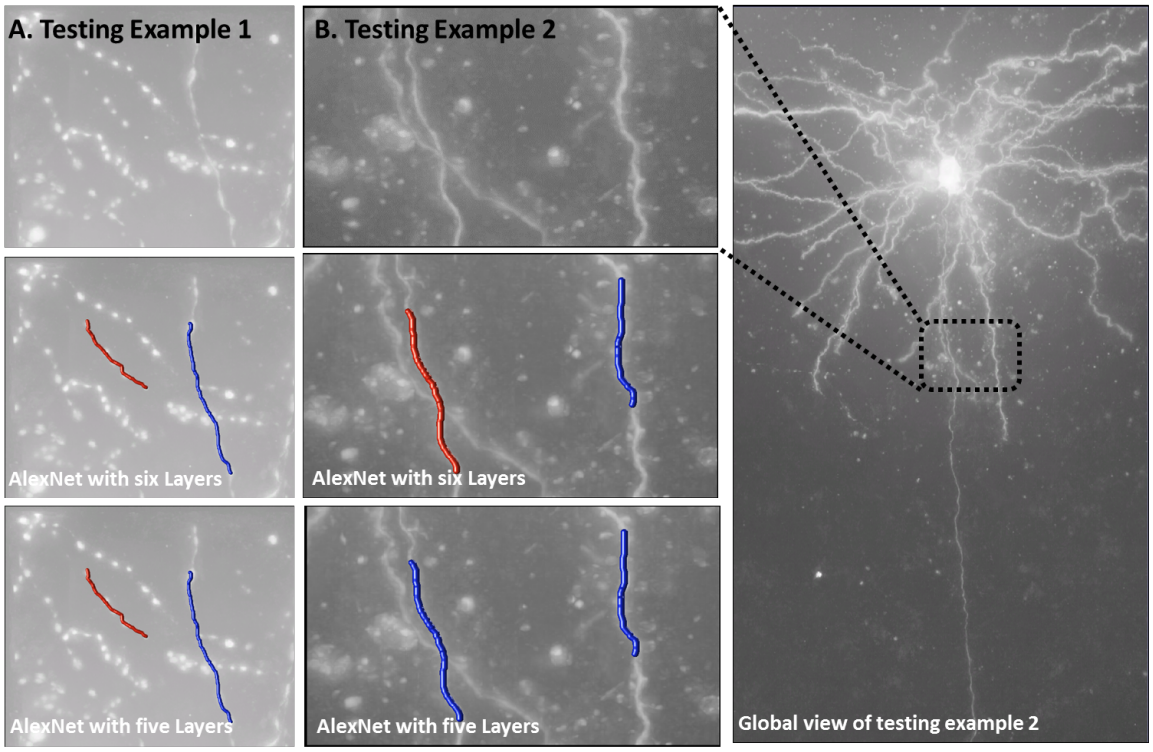


Figure 6. Comparison of dendrite and axon classification using AlexNet with five and six convolutional layers. A) Axons are discrete, and dendrites are continuous. B) Both axons and dendrites are continuous. The left segment is extracted from a long axon. The right segment is extracted from a local dendrite. Red color indicates the axon, and blue color indicates the dendrite in A and B. Note that all these 3D segments are manually annotated by humans using Virtual Finger technology [22]; neurite types are automatically annotated by the proposed module.

Table 4. Comparison results of AlexNet using two different convolutional layers

Deep learning network (AlexNet)	Averaged forward-backward time	Deemed axon	Deemed dendrite	Deemed background
Five convolutional layers	3.27 s	84.79%	93.86%	99.04%
Six convolutional layers	335.91 s	97.89%	98.55%	99.82%

Discussion

In this research, we propose a new Deep Learning based open source toolbox for neuron tracing: DeepNeuron. DeepNeuron provides different modules to comprehend the major tasks:

- For a neuron image stack, it can be used to automatically detect neurite structures.
- For a neuron image stack with detected 3D signals, it can automatically connect signals to generate local segments.
- For a neuron image stack with its associated manual reconstruction, it can provide an evaluation of the given reconstruction quality.
- For a neuron image stack with its associated automated reconstruction, it can be used as a filter to clean up all false tracing and generate a refined result.
- For a neuron image stack with interactive human annotation via the user interface, it can label neurite types in real-time depending on the chosen network model.

DeepNeuron has been implemented as an open source plugin in Vaa3D (http://vaa3d.org) [1, 2]. DeepNeuron toolbox is a highly flexible vehicle allowing investigators to introduce deep learning's capability that could facilitate and drive their research forward. As mentioned in this article, researchers can freely replace different network models that suit their needs. Combined with other related features in Vaa3D including 30+ automatic neuron tracing plugins, semi-automatic neuron annotation, annotation utilities, neuron image/reconstruction visualization, DeepNeuron works as a smart artificial intelligence engine which offers great help to biologists in exploring neuronal morphology.

Methods

Each DeepNeuron module is assembled with one or more processing components. Minor addition or modification on top of these components are added to serve each module's purpose. In module 1, 2-classes-labeled 2D CNN model is integrated with a signal localization process to enable signal detection. Module 2 utilizes a revised Siamese network with the proposed DMST method to formulate neurite structure from isolated dots. In module 3, 2-classes-labeled 2D CNN model serves as a filter to yield refined consensus result. The same 2-classes-labeled 2D CNN is also used in module 4 with the defined quality score to evaluate reconstructions. Finally, a 3-classes-labeled CNN is employed to perform neurite classification in module 5. Note that all networks in our five modules can be replaced with user's own design. The details of the processing components used by DeepNeuron modules are as follows:

2D CNN Training

We tested our module using AlexNet with five convolutional and three fully-connected layers. Human annotated reconstructions are used as the training sample. The 3D reconstruction result is represented using a tree graph, which contains a series of 3D X, Y, Z locations, radius, and topological "parent" of annotation nodes.

In training, a local 3D block is generated, where the center is an annotation node in a neurite segment. 2D MIPs of theses 3D blocks are then used as the positive training set. A similar number of 2D background MIPs set are randomly selected to produce the negative training set. We used 122 bright-field neuron image stacks and their associated manual reconstructions as the training set in the Allen Cell Type Database. In our experiment, there are total 813,012 training samples including 404,211 foreground patches, and 408,801 background patches. In the whole mouse brain dataset, we used a trained model with different training samples including 252,358 foreground patches, and 241,324 background patches from 22 whole mouse brain neuron reconstructions.

The output number in the final layer can be adjusted based on the class labels provided by the training and validation lists. In module 1, 3 and 4, we use two classes (foreground and background) while in module 5 three classes (dendrite, axon, and background) output is enabled.

Signal Localization

Signal localization process provides 2D input patches required by the 2D CNN, and then locates the signal patches recognized by the network back to 3D. We first generate a maximum intensity projection (MIP) image on the XY plane based on the whole 3D image stack. Starting from the origin of the MIP image, 2D patches are defined throughout the whole image domain in every *n*-pixels interval (*n* is the step size which can be defined by the user). These patches are then classified into foreground or background 2D CNN. In order to decrease the false positive rate, we applied a mean-shift approach [23] to these

detected 2D foreground signals to refine the detection result. Then all shifted 2D signals are mapped back to the actual 3D locations based on the local maximum intensity along Z-direction. In further exclude false positive patches caused by the noise in z direction, the MIPs of the local 3D area are extracted and classified using Deep Learning model once again. If the mapped 3D signal is still categorized as the foreground, it will be kept in the final detection result (Supplementary Figure 1).

Siamese Networks

Siamese networks are used among tasks that involve finding similarity or the relationship between two subjects being compared. Our revised Siamese model in this work includes two identical networks. Each one is a CNN consisting of two convolutional layers with max pooling, followed by three fully-connected layers. The two networks are then fed to a contrastive loss function to produce a binary decision.

In training (Figure 3A), we labeled a pair of patches generated from two consecutive annotation nodes as a positive pair, and a pair of patches generated from two separated annotation nodes as a negative pair. We used total 919,391 training pairs. 459,695 of them are positive pairs and 459,696 negative.

In testing (Figure 3C), a 1xM feature vector is extracted for individual input patch (M can be defined by the user. We used M = 200 in our experiment). The Euclidean distance between two feature vectors is calculated as the similarity score of a patch-pair, which is multiplied by the distance to form the weight in our proposed DMST graph.

Quality Score Generation

Supplementary Figure 2 shows the workflow to generate quality scores:

- First, all annotation nodes in a manual reconstruction are classified into two classes: foreground and background based on a trained Deep Learning classification model.
- All classified background nodes have been eliminated from the testing reconstruction.
- Based on the orientation, tip location, and distance, fragments in the initial prediction have been automatically connected to form a refined prediction. In our experiment, we only connect terminal tips between two segments whose orientation differs less than 30 degrees and distance is smaller than 30 voxels.
- Quality score $= \frac{Number\ of\ foreground\ nodes\ in\ the\ refined\ prediction}{Total\ number\ of\ nodes\ in\ the\ testing\ manual\ reconstruction} \times 100\%$

Toolbox and Software Availability

The DeepNeuron toolbox was written in C++ as a plugin to Vaa3D. DeepNeuron source is available at https://github.com/Vaa3D/vaa3d_tools/tree/master/hackathon/MK/Deep_Neuron. In addition to being available as source, the DeepNeuron plugin will be included as a plugin in future binary releases of Vaa3D.

Acknowledgement

We thank Allen Institute for Brain Science for providing neuron datasets and manual annotations. The authors wish to thank the Allen Institute founders, P. G. Allen and J. Allen, for their vision, encouragement and support.

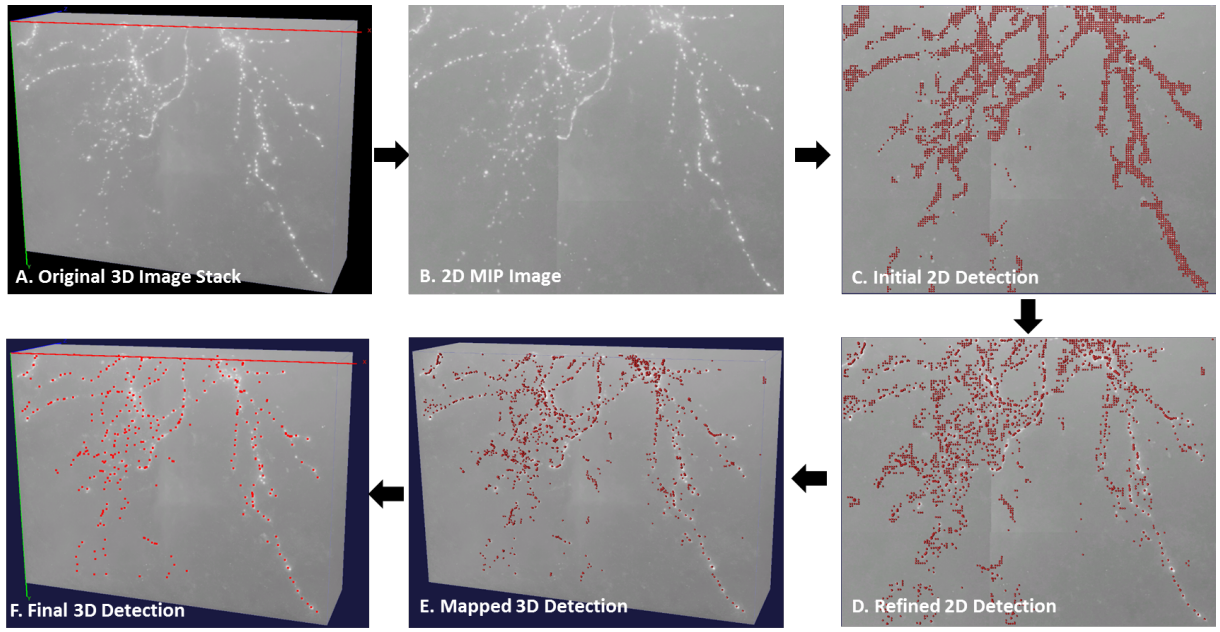
Author contributions

H.P. conceived and managed the project. F.L. proposed the overall technical framework. Z.Z. developed the toolbox and conducted the experiments. H.K. implemented a plugin for the dendrite and axon classification and assisted in several other experiments. All authors edited the manuscript.

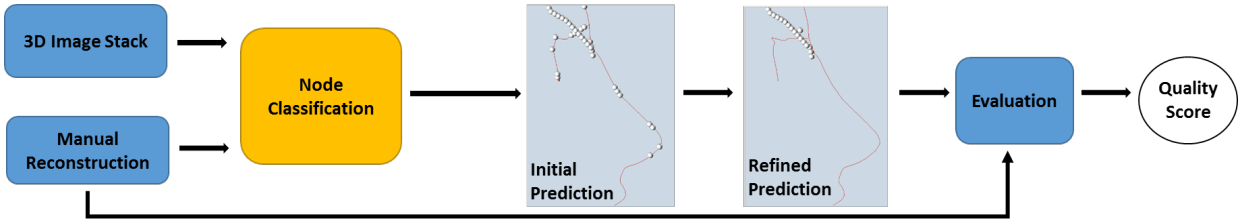
References

1. Peng, H., Ruan, Z., Long, F., Simpson, J.H. & Myers, E.W. V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nature biotechnology* **28**, 348-353 (2010).
2. Peng, H., Bria, A., Zhou, Z., Iannello, G. & Long, F. Extensible visualization and analysis for multidimensional images using Vaa3D. *Nature protocols* **9**, 193-208 (2014).
3. Luisi, J., Narayanaswamy, A., Galbreath, Z. & Roysam, B. (Springer, 2011).
4. Longair, M.H., Baker, D.A. & Armstrong, J.D. Simple Neurite Tracer: open source software for reconstruction, visualization and analysis of neuronal processes. *Bioinformatics* **27**, 2453-2454 (2011).
5. Meijering, E. et al. Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry Part A* **58**, 167-176 (2004).
6. Choromanska, A., Chang, S.-F. & Yuste, R. Automatic reconstruction of neural morphologies with multi-scale tracking. *Frontiers in neural circuits* **6** (2012).
7. Donohue, D.E. & Ascoli, G.A. Automated reconstruction of neuronal morphology: an overview. *Brain research reviews* **67**, 94-102 (2011).
8. Feng, L., Zhao, T. & Kim, J. neuTube 1.0: a new design for efficient neuron reconstruction software based on the SWC format. *eneuro* **2**, ENEURO. 0049-0014.2014 (2015).
9. Liu, Y. The DIADEM and Beyond. *Neuroinformatics* **9**, 99-102 (2011).
10. Peng, H. et al. BigNeuron: large-scale 3D neuron reconstruction from optical microscopy images. *Neuron* **87**, 252-256 (2015).
11. Peng, H. et al. Automatic tracing of ultra-volumes of neuronal images. *Nature Methods* **14**, 332 (2017).
12. 2015 Allen Institute for Brain Science. Allen Cell Types Database Available from: http://celltypes.brain-map.org/.
13. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436 (2015).
14. Gala, R., Chapeton, J., Jitesh, J., Bhavsar, C. & Stepanyants, A. Active learning of neuron morphology for accurate automated tracing of neurites. *Frontiers in neuroanatomy* **8** (2014).
15. Chen, H., Xiao, H., Liu, T. & Peng, H. SmartTracing: self-learning-based neuron reconstruction. *Brain informatics* **2**, 135-144 (2015).
16. Fakhry, A., Peng, H. & Ji, S. Deep models for brain EM image segmentation: novel insights and improved performance. *Bioinformatics* **32**, 2352-2358 (2016).
17. Li, R., Zeng, T. & Ji, S. Deep Learning Segmentation of Optical Microscopy Images Improves 3D Neuron Reconstruction. *IEEE Transactions on Medical Imaging* (2017).
18. Gong, H. et al. High-throughput dual-colour precision imaging for brain-wide connectome with cytoarchitectonic landmarks at the cellular level. *Nature communications* **7** (2016).
19. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. & Shah, R. in Advances in Neural Information Processing Systems 737-744 (1994).
20. Chopra, S., Hadsell, R. & LeCun, Y. in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1 539-546 (IEEE, 2005).
21. Krizhevsky, A., Sutskever, I. & Hinton, G.E. in Advances in neural information processing systems 1097-1105 (2012).
22. Peng, H. et al. Virtual finger boosts three-dimensional imaging and microsurgery as well as terabyte volume image visualization and analysis. *Nature Communications* **5**, 4342 (2014).

23. Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence* **17**, 790-799 (1995).



Supplementary Figure 1. The workflow of 3D neurite signal detection. A) The same example image as shown in Figure 2A. B) 2D MIP on the XY plane. C) Deep Learning based 2D detection result. D) Refined 2D detection result using a mean-shift. C and D are overlaid on top of B. E) Mapped 3D detection result based on local maximum intensity along Z-direction. F) Final 3D detection result after Deep Learning based elimination approach. E and F are overlaid on top of A. Red dots indicate 2D/3D detected signals.



Supplementary Figure 2. The workflow of reconstruction evaluation module. The CNN based trained model in section 2.1 is used in the node classification step. All annotation nodes are classified into foreground (red lines) and background (white dots). Gaps in the initial prediction have been automatically filled based on the orientation, tip location, and distance in the refined prediction.