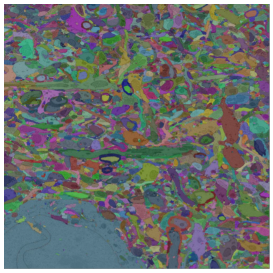
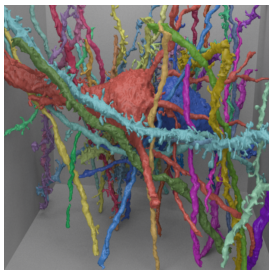
Robust Agglomeration of Labeled Neurons using 3D Skeletonization

Anonymous CVPR submission

Paper ID 446





Abstract

Advancements in electron microscopy image acquisition has created massive datasets petabytes in size that are too large for complete manual segmentation. Several previous strategies for automatic segmentation of these datasets rely work at the per-voxel level. Oftentimes, agglomeration strategies build on top of this layer to produce neuron reconstructions. Despite the success of these algorithms, there is significant room for improvement. Here, we propose a novel framework for correcting errors in existing agglomeration strategies. We create a level of abstraction on top of the existing methods by reformulating the segmentations as a graph and applying global graph-based optimization strategies. In the process we provide key insights into the use of machine-learned shape features for fixing errors.

1. Introduction

The field of connectomics concerns the wiring diagram of the brain at nanometer resolutions. Recent advancements in image acquisition using serial-section electron microscopy (ssEM) has allowed neuroscientists to produce a terabyte of electron micrscopy (EM) image data every hour [11]. It is not feasible for domain experts to manually segment the vast amount of 3-D image data to model an entire brain. Neuroscientists believe that reconstructing an entire mammilian brain at fine resolution will enable new insights into the workings of the brain [14]. These observations will allow for new advancements in neuromedicine and artificial intelligence (CITE).

A significant amount of research focuses on automatic reconstruction of the neurons in these EM images because of the scope and importance of the problem. Several of these algorithms attempt to extract complete neurons through the 3-D volumes using only the raw image data as input. Oftentimes, convolutional neural networks predict membrane probabilities or affinities between voxels and apply simple thresholds to agglomerate the voxels into clusters [17, 24]. These *per-voxel* algorithms produce useful outputs but currently fall short of complete reconstructions with sufficient accuracy.

Researches currently address the failures of the per-voxel algorithms by training random-forest classifiers to agglomerate an oversegmentation of voxels [20] (CITE NEUROPROOF). These classifiers take the output of the per-voxel algorithms as input and generate high-level statistics such as affinity distributions between pixel regions. Presently, these methods use hand-designed features despite the evidence that machine-learned features perform better [4]. These *per-supervoxel* algorithms outperform the per-voxel algorithms but still do not provide the accuracy needed for large scale reconstruction of the brain. Additionally, these methods do not fully leverage the wealth of shape information available.

Here we present a *shape-based* strategy that builds on top of the outputs of per-supervoxel methods. Similarly we take as input the output of the per-supervoxel methods. However, we focus on the overarching shapes traversing through the label volumes to identify potential merge candidates. From these locations we extract machine-learned features and generate a probability that two segments should merge. Lastly, we construct a graphical representation of the input label volume and apply a global segmentation algorithm to produce a better reconstruction. Using graph-based optimization strategies enables us to enforce global constraints that more closely match the underlying biology of the images.

2. Related Work

Significant research in connectomics has focused on extracting information from the pixel data of the raw electron microscopy (EM) images. Building off previous work in

computer vision that used normalized cuts and other graph partitioning algoirithms on images [13, 26, 27], some of the methods apply computationally expensive graph partitioning techniques on the per-voxel images [1]. Other approaches at the per-pixel level have focused on training 2-D convolutional networks to predict membrane probabilities [6]. More recent work has expanded on these early networks by adding the z-dimension to create 3-D networks [17]. Oftentimes these networks now predict affinities between voxels rather than membrane probabilities [24]. Extensive research has focused on enhancing the loss functions and optimizers for these networks [5, 18, 19].

ADD SECTION ON CORRECTING SPLIT ERRORS

These neural networks produce probabilities that neighboring voxels belong to the same neuron. Many algorithms build on top of these affinities by agglomerating voxels to produce reconstructed label volumes of the EM images [16, 20, 23] (CITE NEUROPROOF, ZWATERSHED). Despite the successes of these algorithms they often make early local mistakes leading to errors in the segmentation. Proofreading methods create a "human-in-the-loop" framework that allows users to find errors and correct them [8, 9, 10]. Flood-filling networks merge these two steps by training an end-to-end network that takes as input raw image data and produces a segmentation [12]. Although these networks have produced impressed segmentation accuracies, they are currently too slow for large scale connectomics reconstructions.

The fields of computer graphics, mathematics, and biomedical visualization have produced extensive research into the skeletonization of 3D binary volumes. A large amount of research has explored topologically consistent thinning algorithms and potential medical applications of these methods [21, 22]. In computer graphics, fast medial axis algorithms allow animators to quickly move characters through successive frames [2, 3]. The Tree-structure Extraction Algorithm for Accurate and Robust Skeletons (TEASER) has been used by connectomics researchers to parameterize 3-D shapes [25, 28].

ADD SECTION ON LEARNED SHAPE FEATURES

ADD SECTION ON GRAPH PARTITIONING

3. Method

Our method works on top of existing EM agglomeration algorithms such as *NeuroProof* or *GALA* (CITE BOTH). First we use a 3D U-net to generate voxel affinities predicting the probability that two voxels belong to the same neuron. The *zwatershed* algorithms takes these affinities and generates an oversegmentation of the neurons (CITE). *NeuroProof* agglomerates these supervoxels by training a multi-level random forest classifier. We use a low threshold for neuroproof to generate an oversegmentation.

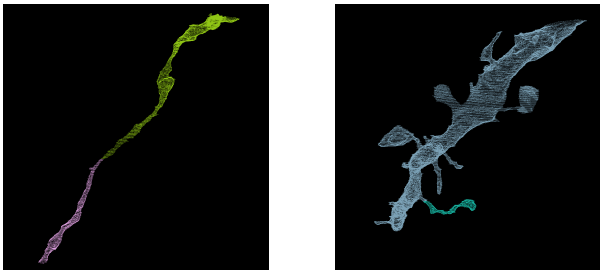


Figure 1: Two erroneously split segments that should merge together. Most segments that we want to merge have the same general structure.

In order to use graph-based optimization strategies we need to generate nodes N and edges E for the graph G.

3.1. Graph Creation

3.1.1 Node Generation

The simplest method of generating nodes is to create one node for every unique segment label in a volume. However, there are many small segments that remain after NeuroProof that only introduce noise when moving to a higher level of abstraction. For now, we decide to remove these segments from consideration and focus on the only the large segments that have expansive spans through the volume. Every segment with fewer than $20,000$ voxels is pruned from the set of considered nodes. Figure 2 shows two typical segments that each have one corresponding node in the graph G.

3.1.2 Edge Generation

The naïve edge generation approach is just to simply consider all nodes whose segmentations have a neighboring voxel. Current agglomeration approaches such as NeuroProof or GALA use this criteria for deciding which segments could potentially form one neuron. However, this produces too many edges in our graph, many of which are easily prunable. In fact, many of the segments that we need to merge have very similar structures. Consider Figure 1 which shows two erroneously split segments. In both instances the segments around the break follow the same general shape. The segment is more-or-less tubular in the close vicinity with an abrupt break at the end of segment where the next segment begins. The following algorithm identifies these locations without introducing as much "noise" as using the naïve strategy.

We begin by generating skeletons for every segment. The intuition behind this decision is that a skeleton is a simplified representation of the overarching shape of a segment. Figure 2 shows two example segments and their corresponding skeletons in white. These skeletons were generated using the Tree-structure Extraction Algorithm for

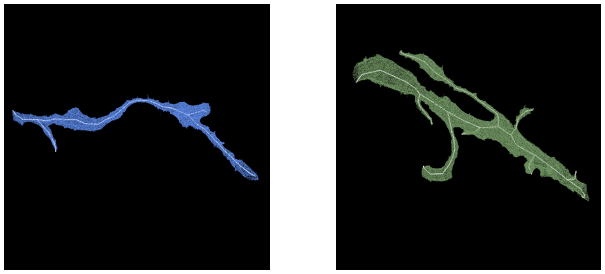


Figure 2: Two example outputs of the TEASER skeletonization algorithm.

Accurate and Robust Skeletons (TEASER) [25] which has been used in connectomics research previously [28]. The skeleton consists of a series of *joints* with edges between neighboring joints. Joints that have only one neighbor are also called *endpoints*. Joints that are within 50 voxels of each other are pruned to reduce unnecessary branching.

After skeletonization, we proceed to identify locations between segments that we should consider for merging. The algorithm proceeds in two passes. In the first pass we iterate over every endpoint, e, in all of the skeletons and create a set S_e where the set includes every segment that has a single voxel with t_{low}nm from the endpoint. This first pass often includes too many negative merge candidates leaving a very unbalanced set of merge/split candidates (FIX THIS). Therefore, in our second pass we iterate over all of the endpoints and their corresponding sets. We find the closest endpoint for a given segment $s \in S_e$ to the endpoint e. If the closest endpoint is less than t_{high}nm from the original endpoint e, we add an edge in the graph between these two segments. We also store the midpoint between the closest endpoint and the original endpoint e, which will be useful in the following section. Algorithm **??** provides pseudocode for this edge generation algorithm. The results in this paper follow from $t_{low} = 240nm$ and $t_{high} = 600nm$.

function GENERATEEDGES($skeletons$)
  for $skeleton$ in $skeletons$ **do**
    $candidates$ = set()
    for $endpoint$ in $skeleton$ **do**
    end for
  end for
end function

It is important to note that the above algorithm does not enforce neighboring voxel constraints on nearby segments. There are instances where two segments belong to the same neuron but the segments produced by the per-pixel and per-superpixel algorithms are non-adjacent. Figure (ADD FIGURE) shows two such examples that are not considered by previous research in this field. Finding and merge these examples is one of the benefits of retreating from per-pixel algorithms. Neither of these examples could merge under the GALA or NeuroProof pipelines.

3.2. Edge Probabilities

The previous section outlines how to generate edges between nodes in our graph structure. Here we will introduce a neural network architecture for generating probabilities that two nodes sharing an edge belong to the same neuron. The neural network takes as input only data from the input segmentation.

3.2.1 Network Architecture

The above skeletonization algorithm produces 3D locations that require further consideration for merging. To determine which of these segments should actually merge, we train a 3-D convolutional neural network. We extract a cubic region with length 1200nm centered at these locations. These cubes will provide the local information for the neural network to predict which neighboring segments belong to the same neuron. A cubic length of 1200nm provides enough local shape context for the network without introducing an excessive amount of noise. Figure 3 shows cubes of lengths 800nm, 1200nm, and 1600nm.

We transform the extracted cubes for input into the neural network. The network takes three input channels for each voxel in the 3-D volume corresponding to the following mapping. Consider a pair of segments with labels l_1 and l_2 for every voxel v in the cube of interest. The first channel is 0.5 if $v = l_1$ and -0.5 otherwise, the second channel is 0.5 if $v = l_2$ and -0.5 otherwise, and the third channel is 0.5 if $v = l_1$ or $v = l_2$ and -0.5 otherwise.

Each 3-channel volume is subsequently downsampled into an array of size $(3, 22, 68, 68)$ using nearest-neighbor interpolation. Our neural network trains on these 4-D arrays to generate probabilities that l_1 and l_2 should merge for every input.

Our network architecture has three layers of double convolutions followed by a max pooling step [5]. As described above, the input with into the network is $(22, 68, 68)$ with 3 channels. The filter size of the layers are 16, 32, and 64 with size doubling deeper into the network. The first two max pooling layers are anisotropic with pooling only in x and y to match the anisotropic nature of our EM datasets. All convolutions have kernel sizes of $(3, 3, 3)$. The output size of the final max pooling layer is $(5, 5, 5)$ with 64 channels. The output is flattened into a 1-D vector with 8000 entries which enters two fully connected layers with 512 and 1 dimensions in the output respectively. All activation functions are LeakyReLU with $\alpha = 0.001$ except for the final activation which uses a sigmoid function [7, 18]. There is a dropout of 0.2 after every pooling layer and the first dense layer. There is a dropout of 0.5 after the final dense layer. The above method uses stochastic gradient descent

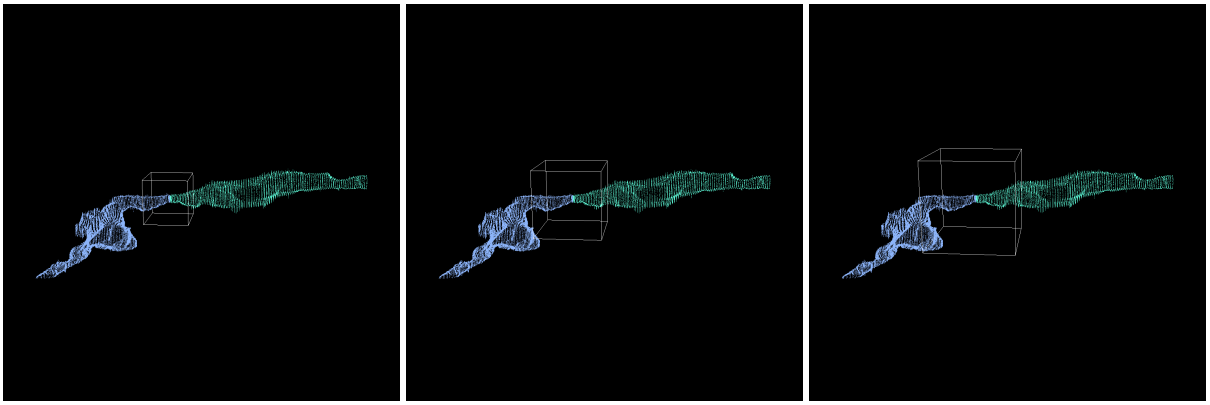


Figure 3: The white outlines show three different possible cubic sizes to input into the neural network. The left example (800nm) provides less local context than the middle example (1200nm). The right example (1600nm) extracts too large of a local region that produces noise as one of the segments leaves the bounding box only to reenter.

with Nesterov's Accelerated Gradient [19]. This optimizer has an initial learning rate of 0.01, momentum of 0.9, and a decay rate of $5 * 10^{-8}$. Figure 4 provides an overview of the proposed architecture.

3.2.2 Data Augmentation

Manual labeling of connectomics datasets is time intensive and expensive (CITE?). As we consider higher levels of abstractions of the data, the amount of useful labeled data decreases. This follows from the simple fact that manual annotation of the data happens at the per-pixel level, so starting with larger agglomerated segments leads to fewer unique human labels (THIS DOESN'T MAKE SENSE YET).

Therefore, we apply some data augmentation to the generated examples to increase the size of the training datasets. We consider all rotations of 90 degrees along the xy-plane in addition to mirrors along the x and z axes. This produces an additional 16 times more training data.

3.3. Agglomeration

After constructing the above graph structure we can apply a graph-based segmentation strategy. There are many formulations of graph-based optimization strategies that provide different guarantees on their output. Neurons in the brain should be acyclic, i.e. the output shape should have a genus of zero. Current connectomics agglomeration techniques do not leverage this additional information but rather consider neighboring regions in successive order without regard to created loops. In our graph formulation we can enforce this topological property by applying a multicut partition onto the graph which generates a forest on the nodes. There are several heuristics that solve the multicut problem. For our purposes we use the Kernighan-Lin algorithm [15].

4. Evaluation

4.1. Datasets

We evaluate our methods on three EM datasets.

4.1.1 NeuroProof Pipeline

Our methods build on top of existing agglomeration strategies. For the purpose of this paper we used the outputs from two established pipelines.

TODO: Toufiq description of neuroproof pipeline, description of FlyEM pipeline

4.2. Preprocessing

We perform significant pruning of potential merge candidates by considering only pairs of skeleton endpoints produced by Algorithm **??**. This strategy greatly reduced the number of false merge candidates that we considered. However, we evaluate the overall effectiveness of this strategy both in reducing trivial split candidates while maintaining a large majority of the candidates which should merge.

Our candidate generation strategy doesn't enforce an adjacency constraint which allows us to merge candidates that are not consider under existing frameworks. We examine the success of our pipeline in merging said candidates.

4.3. Classifier Training

DISCUSS PARAMETERS FOR CLASSIFICATION

4.4. Graph Optimization

We evaluate the improvement of using a graph optimization strategy over a naïve approach that performs hierarchical clustering for all segments above a certain threshold.

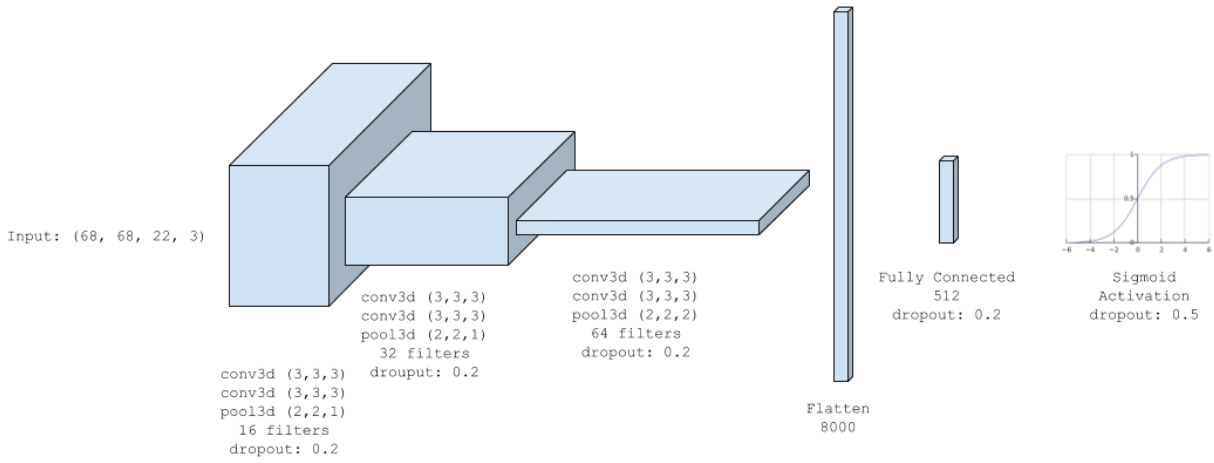


Figure 4: The architecture for the neural networks follows the *VGG* style of double convolutions followed by a max pooling operation. The number of filters doubles each layer leading to a fully connected layer and a sigmoid activation function.

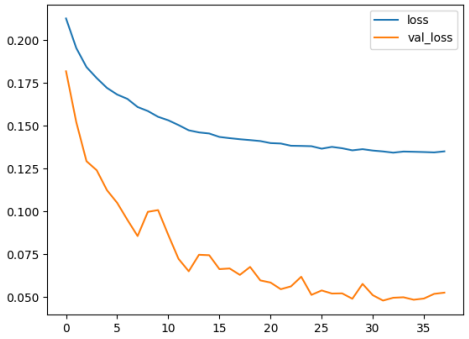


Figure 5: The training and validation loss when training the neural network on the L. Cylinder data set.

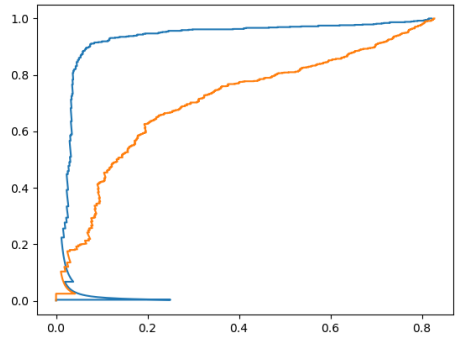


Figure 6: The ROC curve for the training and validation datasets.

5. Results

Here we show the results for the three main contributions for this paper: merge consideration using skeleton pruning, merge probabilities output from the trained convolutional neural network, and the final output of the pipeline.

5.1. Skeleton Pruning

5.1.1 Finding Merge Candidates

5.1.2 Finding Split Candidates

Effective pruning for merge candidates should have high precision and recall for segments that should be merged.

5.2. Neural Network Classification

5.2.1 Merge Network

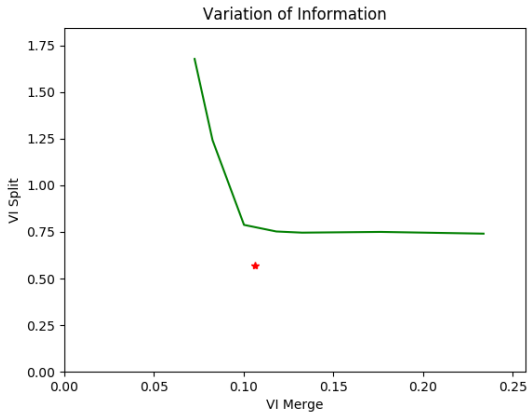
Figure 5 shows the training and validation loss during the training of the neural network.

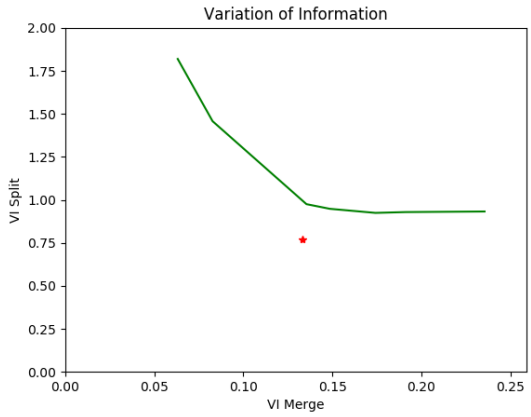
5.2.2 Split Network

5.3. Graph-based Error Correction

6. Conclusions

- Impact
- Future work





References

[1] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht. Globally optimal closed-surface segmentation for connectomics. In *European Conference on Computer Vision*, pages 778–791. Springer, 2012. 2

[2] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. In *ACM Transactions on Graphics (TOG)*, volume 26, page 72. ACM, 2007. 2

[3] G. Bharaj, T. Thormählen, H.-P. Seidel, and C. Theobalt. Automatically rigging multi-component characters. In *Computer Graphics Forum*, volume 31, pages 755–764. Wiley Online Library, 2012. 2

[4] J. A. Bogovic, G. B. Huang, and V. Jain. Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*, 2013. 1

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014. 2, 3

[6] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012. 2

[7] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989. 4

[8] D. Haehn, J. Hoffer, B. Matejek, A. Suissa-Peleg, A. K. Al-Awami, L. Kamentsky, F. Gonda, E. Meng, W. Zhang, R. Schalek, et al. Scalable interactive visualization for connectomics. In *Informatics*, volume 4, page 29. Multidisciplinary Digital Publishing Institute, 2017. 2

[9] D. Haehn, V. Kaynig, J. Tompkin, J. W. Lichtman, and H. Pfister. Guided proofreading of automatic segmentations for connectomics. *arXiv preprint arXiv:1704.00848*, 2017. 2

[10] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. W. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2466–2475, 2014. 2

[11] D. G. C. Hildebrand, M. Cicconet, R. M. Torres, W. Choi, T. M. Quan, J. Moon, A. W. Wetzel, A. S. Champion, B. J. Graham, O. Randlett, et al. Whole-brain serial-section electron microscopy in larval zebrafish. *Nature*, 545(7654):345–349, 2017. 1

[12] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain. Flood-filling networks. *arXiv preprint arXiv:1611.00421*, 2016. 2

[13] J. H. Kappes, M. Speth, G. Reinelt, and C. Schnörr. Higher-order segmentation via multicuts. *Computer Vision and Image Understanding*, 143:104–119, 2016. 2

[14] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015. 1

[15] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970. 4

[16] S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, J. Morgan, D. Lee, D. Berger, N. Kasthuri, J. W. Lichtman, and H. Pfister. Rhoananet pipeline: Dense automatic neural annotation. *arXiv preprint arXiv:1611.06973*, 2016. 2

[17] K. Lee, A. Zlateski, V. Ashwin, and H. S. Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary prediction. In *Advances in Neural Information Processing Systems*, pages 3573–3581, 2015. 1, 2

[18] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013. 2, 4

[19] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). 2, 4

[20] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics*, 8, 2014. 1, 2

[21] K. Palágyi. A 3d 3-subiteration thinning algorithm for medial surfaces. In *International Conference on Discrete Geometry for Computer Imagery*, pages 406–418. Springer, 2000. 2

[22] K. Palágyi, E. Balogh, A. Kuba, C. Halmai, B. Erdőhelyi, E. Sorantin, and K. Hausegger. A sequential 3d thinning

algorithm and its medical applications. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 409–415. Springer, 2001. 2

[23] T. Parag, F. Tschopp, W. Grisaitis, S. C. Turaga, X. Zhang, B. Matejek, L. Kamentsky, J. W. Lichtman, and H. Pfister. Anisotropic em segmentation by 3d affinity learning and agglomeration. *arXiv preprint arXiv:1707.08935*, 2017. 2

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 1, 2

[25] M. Sato, I. Bitter, M. A. Bender, A. E. Kaufman, and M. Nakajima. Teasar: Tree-structure extraction algorithm for accurate and robust skeletons. In *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, pages 281–449. IEEE, 2000. 2, 3

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 2

[27] S. Tatiraju and A. Mehta. Image segmentation using k-means clustering, em and normalized cuts. *Department of EECS*, 1:1–7, 2008. 2

[28] T. Zhao and S. M. Plaza. Automatic neuron type identification by neurite localization in the drosophila medulla. *arXiv preprint arXiv:1409.1892*, 2014. 2, 3