CVPR
#446

CVPR
#446

CVPR 2018 Submission #446. CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

Graph-based Neuron Agglomeration using 3D Skeletons

Anonymous CVPR submission

Paper ID 446

Abstract

Advancements in electron microscopy image acquisition have created massive connectomics datasets which are petabytes in size and make manual segmentation not feasible. Proposed methods for automatic segmentations generate super-pixel approximations of neural membranes followed by agglomeration strategies to create full neuron reconstructions. However, pixel-based segmentation results widely neglect global geometric properties. We generate skeletons of membrane labels to approximate neural pathways in 3D. This allows us to efficiently apply graph-based optimization strategies and to train automatic classifiers for shape description against manually labeled ground truth. Our classifiers detect feasible and impossible neural pathways by looking at geometric properties alone, and are able to improve the segmentation labels accordingly. We demonstrate the performance of our classifiers on multiple real-world connectomics datasets with average variation of information improvement of $X\times$. We discuss our findings and provide insights to learning shape features for neuron agglomeration.

1. Introduction

The field of connectomics is concerned with reconstructing the wiring diagram of the brain at nanometer resolutions. Recent advancements in image acquisition using multi-beam serial-section electron microscopy (sSEM) has allowed neuroscientists to produce terabytes of electron micrsocopy (EM) image data every hour [12]. Neuroscientists believe that reconstructing an entire mammilian brain at fine resolution will enable new insights into the workings of the brain [17]. These observations will allow for new advancements in neuromedicine and artificial intelligence (CITE). Segmentation is part of this reconstruction process, assigning a unique label for every neuron in the EM image. It is not feasible for domain experts to manually segment the vast amount of 3-D image data to model an entire brain.

A significant amount of research focuses on automatic reconstruction of the neurons in EM images because of
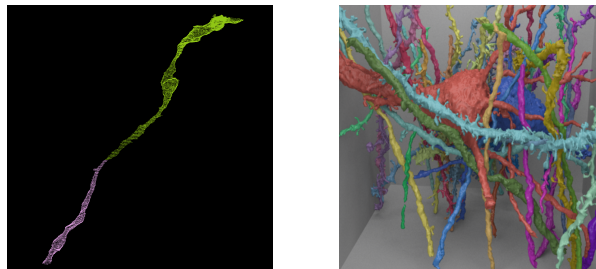

Figure 1: We use 3D skeletons to combine neuron labels to full reconstructions. (TODO: Add schematic figure showing the difference between voxel-based / super-voxel based / graph-based algorithms with connections and hierarchy)

the scope and importance of the problem (ADD REFERENCES).All of these algorithms extract neuronal processes through the 3-D volumes using only the raw image data as input. Oftentimes, convolutional neural networks (CNNs) predict membrane probabilities or affinities between voxels and apply simple thresholds to agglomerate the voxels into clusters [23, 32]. These *per-voxel* algorithms produce excellent results but currently fall short of complete reconstructions with error rates of approximately $X\%$.

Researches currently address the failures of the per-voxel algorithms by training random-forest classifiers to agglomerate an oversegmentation of voxels [26] (CITE NEURO-PROOF). 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 reconstructions of the brain. Additionally, these methods do not fully leverage the wealth of shape information available. E.g., it is well known that neurons angle at less than 30 degrees (CITE).

Here we present a *graph-based* strategy that builds on top of the outputs of per-supervoxel methods by taking neuronal shape properties into account. Similarly we take as

input automatic segmentations. However, we focus on general neuron 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 label segments should be joined. We construct a graph representation of the input label volume. Using graph-based optimization strategies enables us to enforce global constraints that more closely match the underlying biology of the images. Lastly, we apply a global segmentation algorithm to produce a better reconstruction.

2. Related Work

Voxel-based methods. A large amount of connectomics research considers the problem of extracting segmentation information at the voxel level from the raw EM images. Such works focus on training 2-D convolutional neural networks to predict membrane probabilities per image slice [6, 14, 37, 21, 18]. More recent strategies augment these neural networks with the z-dimension creating full 3-D convolutional networks [23]. Oftentimes these networks produce probabilities for the affinity between voxels rather than probabilities for a voxel belonging to a cell membrane [32]. A sizable amount of computer vision research analyzes various hand-designed shape features for determining similarity between objects [27] and for segmentation[7]. Currently, the random-forest classifiers used in connectomics rely on hand-designed features to make merge decisions. Some of these features encode the distribution of probabilities that voxels along a segment boundary belong to the same neuron as their neighbors [26, 31]. However, there is evidence that machine-learned features outperform hand-designed ones [4]. An extensive amount of research in computer vision, applied mathematics, and statistics evaluates the loss functions and optimizers for these networks [5, 24, 25].

Super-voxel methods. These approaches generate probabilities that neighboring voxels belong to the same neuron. Many algorithms work at the resulting super-voxel level and train random-forest classifiers to produce segmentations of the EM images where every unique neuron in the volume has a unique label [22, 26, 30, 39] (CITE NEUROPROOF). Despite the success of these classifiers, they often make mistakes based on the local information leading to errors in the segmentation. Proofreading methods create a "human-in-the-loop" framework that allows users to find errors and correct them [9, 10, 11, 40].

More recently, flood-filling networks merge the process of generating voxel affinities and then producing segmentations by training an end-to-end neural network that outputs segmentations [15]. These networks produce impressive segmentation accuracies of over $X\%$.

Graph-based methods. Many segmentation and clustering algorithms use graph partitioning techniques [1] or normalized cuts for image segmentation [16, 34, 36]. The multicut graph partitioning algorithm extracts segments the graph and remove edges such that there are no cycles. This closely resembles the biological constraint that neurons have a topological genus of zero. There are several useful multicut heuristics which provide good approximations with reasonable computational costs [13, 19, 20]

Skeletonization methods. We use a graph-based abstraction of segmentation data based on skeletonizations. This topic is the focus of extensive research in the fields of computer graphics, mathematics, and biomedical visualization. Some of the research explores topologically consistent thinning algorithms and potential medical applications of these methods [28, 29]. In computer graphics, fast medial axis algorithms allow animators to quickly move characters through successive frames [2, 3]. In these works, the skeleton acts as the simplest representation of a 3-D volume. The Tree-structure Extraction Algorithm for Accurate and Robust Skeletons (TEASER) has been used by connectomics researchers to parameterize 3-D shapes [33, 38].

3. Method

Our method takes as input an existing oversegmentation of an EM image volume. Section ?? discusses two pre-existing pipelines for generating these segmentations. We evaluate our proposed method on the outputs from both of these pipelines.

3.1. Graph Creation

We need to generate nodes N and edges E to apply a graph-based optimization strategy for segmentation. In addition, these edges need non-negative weights.

3.1.1 Node Generation

The simplest node generation strategy creates one node for every unique segment label in the input volume. However, there are often a small number of labels in the volume corresponding to very small regions. It is difficult to extract useful shape features from these segments because of their small, and often random, shape. These segments are pruned from the graph and do not have corresponding nodes. For the proposed framework, we remove segments with fewer than $20,000$ voxels (ADD PERCENT, current estimate less than 10%). Figure 2 shows two typical input segments that receive a corresponding node in the graph. .
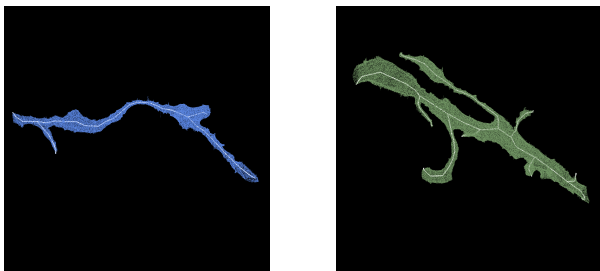
CVPR
#446

CVPR
#446

CVPR 2018 Submission #446. CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

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

Figure 3: Two erroneously split segments that should merge together. Most segments that we want to merge have the same general structure.

3.1.2 Edge Generation

A naïve approach to generating edges simply produces an edge between all segments that have a single pair of neighboring voxels. Current agglomeration strategies such as NeuroProof or GALA use this criteria when deciding pairs of segments to merge into one neuron. However, for us this method will produce too many edges in our graph, many of which are easily prunable. Many of the segments that are erroneously split have a similar structure. Consider Figure 3 which shows two such pairs. In both instances the segments around the break follow the same general shape. The segment is tubular in the close vicinity with an abrupt break perpendicular to the elongated direction. We present the following algorithm to identify these locations.

We generate a skeleton for every segment following the intuition that a skeleton represents a simplified representation of the overall shape of an object. We use the Tree-structure Extraction Algorithm for Accurate and Robust Skeletons (TEASER) [33, 38]. These skeletons consist of a sequence of *joints* with edges between successive joints. We prune the joints that are within 50 voxels of each other to reduce unnecessary branching. For the purposes of this algorithm, joints that have only one connected neighbor are referred to as *endpoints*. Figure 2 shows the skeletons in white of two typical segments from the label volume. After skeletonization, we begin to identify segments for merge consideration with the following two-pass heuristic. In the first pass, we iterate over all endpoints e belonging to a segment S and create a set of segments $\mathbb{S}'_e$ which includes all labels that have a single voxel within t_{low} nanometers from e. This first pass often includes too many candidates that should not merge so we apply an additional pass to prune these sets. In the second pass, we consider all of the segments $\mathbb{S}'_e$ for every endpoint e. If a segment $S' \in \mathbb{S}'_e$ has an endpoint within t_{high} nanometers of e, the segment S and S' are considered for merging. We store the midpoint between the two endpoints as the "center" of the potential merge. Algorithm (ADD REF) provides pseudocode for this edge generation algorithm. The results in this paper follow from $t_{low} = 240$ and $t_{high} = 600$.
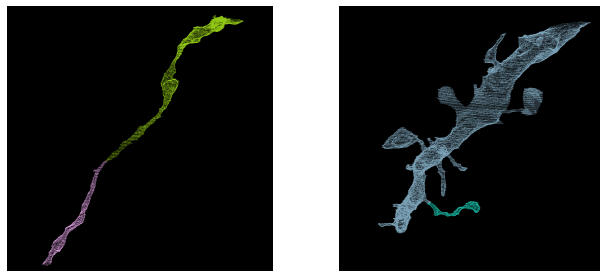
```
function GenerateEdges(skeletons)
    for skeleton in skeletons do
        candidates = set()
        for endpoint in skeleton do
            TODO ADD CODE
        end for
    end for
end function
```

Note that the above algorithm does not enforce any segment adjacency constraints. Figure (ADD FIGURE) shows two examples that would not be considered in previous research since they are not adjacent. Finding and merging these examples is one of the benefits of retreating from per-pixel algorithms.

3.2. Edge Probabilities

The previous section outlines how to generate edges between nodes in our graph structure. Here we 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 3-D 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 4 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 follow-

CVPR
#446

CVPR
#446

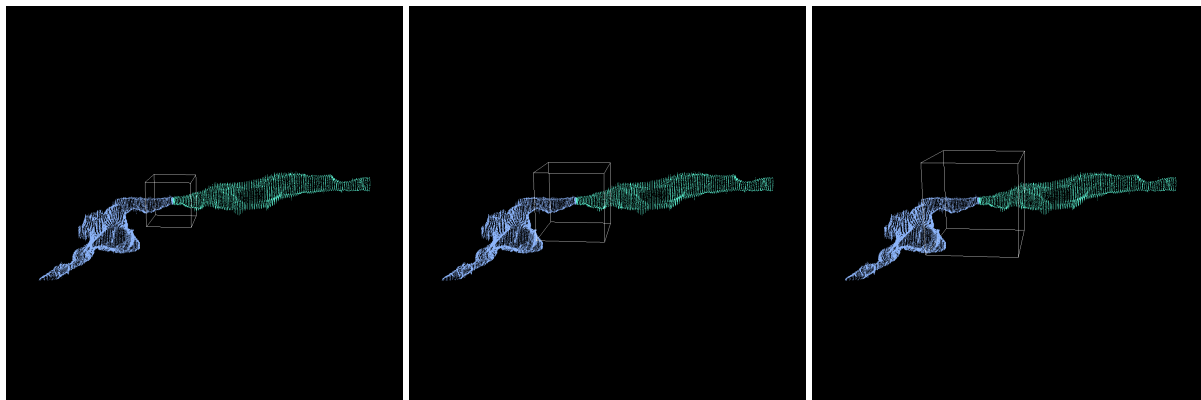CVPR 2018 Submission #446. CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

Figure 4: 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.

ing 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.

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 [8, 24]. 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 with Nesterov's accelerated gradient [25]. This optimizer has an initial learning rate of 0.01, momentum of 0.9, and a decay rate of $5 * 10^{-8}$. Figure 5 provides an overview of the proposed architecture.

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 loop creation. 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 [19].

4. Evaluation

We evaluate our proposed methods on two different connectomics datasets, one anisotropic and the other isotropic, from two different species.

4.1. Datasets

4.1.1 Kasthuri

The Kasthuri dataset images the neocortex of a mouse brain produced by a scanning electron microscope [17]. This dataset is $338 \times 3618 \times 5342$ voxels in size. The resolution of the dataset is $3 \times 3 \times 30 \text{nm}^3$. We evaluate our methods using the left cylinder of this 3-cylinder dataset.

4.1.2 FlyEM

The FlyEM dataset comes from the mushroom body of a 5-day old adult male *Drosophila* fly imaged by a focused ion-beam milling scanning electron microscopy. The mushroom body is this species is the major site of associative learning [35]. The original dataset contains a $40\mu\text{m} \times 50\mu\text{m} \times 120\mu\text{m}$ volume of which we use two subsections of size $1000 \times 1000 \times 1000$ voxels. The resolution
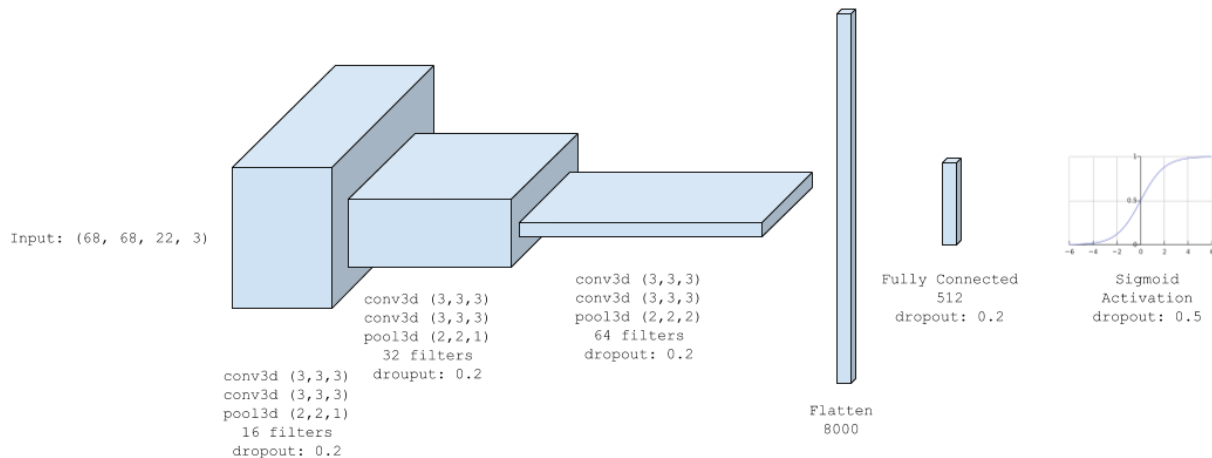
Figure 5: The architecture for the neural networks follows the *VGG* style of double convolutions followed by a max pooling operation. The number of filters doubles each layer leading to a fully connected layer and a sigmoid activation function.

of this dataset is $10 \times 10 \times 10 \mu m^3$ per voxel. The image data and the accompanying ground truth is available (CITE).

4.1.3 Segmentation Pipeline and Baseline

Our algorithm takes the output of a supervoxel based agglomeration strategy such as NeuroProof or GALA. For our experiments we use the output from NeuroProof (CITE). We generate voxel affinities from the raw image data using the U-Net architecture [32]. The Zwatershed algorithm (CITE) creates an oversegmentation of the neurons given these voxel affinities. We input the raw image data, the voxel affinities, and the oversegmentation into Neuro-Proof which performs a hierarchical agglomeration strategy to merge the supervoxels into larger and larger segments. Two neighboring segments receive a merge-likelihood score based on various extracted features such as the distribution of affinities along the segment boundary. A random forest classifier generates probabilities based on these features. (TODO: TOUFIQ ADD HERE)

4.2. Skeleton Pruning

Our method prunes potential merge candidates by considering the skeleton as a simplification of the segment shape. Segment skeletons that have nearby endpoints are merge candidates. The underlying assumption in this model is that the segments break in a particular fashion, primarily perpendicular to the direction of travel (CONFUSING). We perform a series of experiments to evaluate the effectiveness of this idea. First, we examine how many merge candidates are not considered using our method that would have been if we employed the strategy of considering all neighboring segments. These are locations that we cannot possibly correct and therefore we want this number to be as low

as possible. The second experiment considers how many locations are not considered that should not be merged. We want this number to be as large as possible since we want to avoid making mistakes by merging these segments. At best, these segments waste time and at worst they lead to incorrect merges.

4.3. Classifier Training

We divide both datasets into half to train the classifier. We generate all valid examples for merge consideration based on the skeletonization algorithm. In the training dataset, 80% of the examples are used for training and the remaining examples are used for validation. To start, we consider networks of varying sizes, optimization functions, and loss functions. The supplemental material has additional information concerning the tuning of the network. There are XXX,XXX learnable parameters in our final architecture. The training ran for XX epochs. Table 1 provides the final network parameters.

Parameters	Values
Loss Function	Mean Squared Error
Optimizer	SGD with Nesterov Momentum
Momentum	0.9
Initial Learning Rate	0.01
Decay Rate	$5 * 10^{-8}$
Activation	LeakyReLU ($\alpha = 0.001$)
Kernel Sizes	$3 \times 3 \times 3$
Filter Sizes	$16 \rightarrow 32 \rightarrow 64$

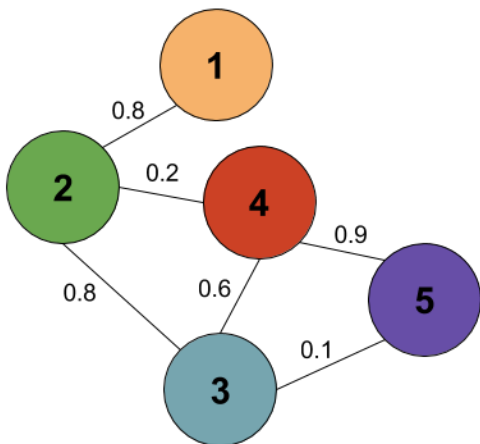Table 1: The parameters for the trained neural network.

5

Figure 6: An example of the benefits of using multicut over simpler agglomeration schemes. If we input this example into a naïve agglomeration strategy that collapses all edges with a threshold greater than 0.5, this entire segment will merge together despite the fact that nodes 2 and 4 and nodes 3 and 5 have low affinities.

Data Augmentation. 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.

4.4. Graph-based Strategies

Most current agglomeration strategies in connectomics consider all pairs of neighboring segments. A classifier produces a probability that each pair belongs to the same segment. All segment pairs with a probability above a certain threshold are merged together. Consider the simple example in Figure 6. Under the existing agglomeration schemes this entire graph would become one segment. However multicut eliminates all cycles within the graph creating two unique segments. We compare our results using both the traditional hierarchical agglomeration strategy with the graph-based approach we propose.

4.5. Variation of Information

We use the variation of information metric to determine the accuracy of our pipeline. The variation of information measures the distance between two segmentations and is similar to mutual information. A lower variation of information score corresponds to a segmentation closer to the ground truth. (TODO: TOUFIQ ADD HERE)
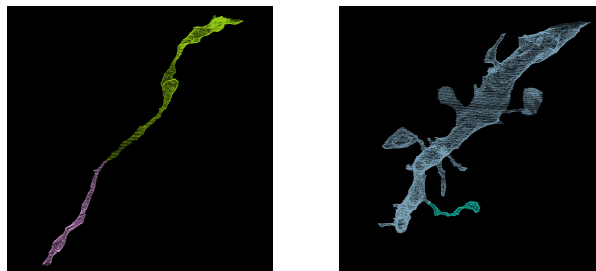

Figure 7: Two erroneously split segments that should merge together. Most segments that we want to merge have the same general structure.

5. Results

5.1. Skeleton Pruning

Finding Positive Examples. As a baseline, we consider all pairs of segments that have neighboring voxels and see what percent of these segments are considered for merging. In the L. Cylinder dataset, there are XXX neighboring segments which have at least adjacent voxels. Our method which uses skeletons to prune the considered candidates finds XXX of these segments, or XX% percent. On the FlyEM dataset, there are XXX adjacent segments with XXX found (XX% percent). Figure 7 shows some of the missed examples. The most common errors are the dendrite spines where the larger segment did not have an endpoint near the split. However, our algorithm does not require segments to be adjacent to merge. This offers a great benefit compared to traditional methods. In our datasets XXX and XXX segments are candidates for merging despite not having adjacent voxels. Figure 7 gives one such pair which is merged given the results of our algorithm.

Pruning Negative Examples. Traditional agglomeration strategies consider a large number of negative candidates that we hope to prune. The original approach of considering adjacent supervoxels has XX,XXX and XX,XXX candidates which do not belong to the same neuron. With our pruning algorithm, this reduces to X,XXX and X,XXX, a factor of over XX times. This allows a more reasonable training environment with greater balance between the positive and negative merge examples.

5.2. Classification Performance

Table **??** shows the precision and recall for all of the datasets. Our methods are more general than traditional connectomics examples allowing us to train the network on an anisotropic dataset and get impressive results on an isotropic dataset. Our initial network does not take in any image data, only the segment shapes, so we merely extract the 1200nm^3 region from any dataset and rescale as input

CVPR
#446

CVPR
#446

CVPR 2018 Submission #446. CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.
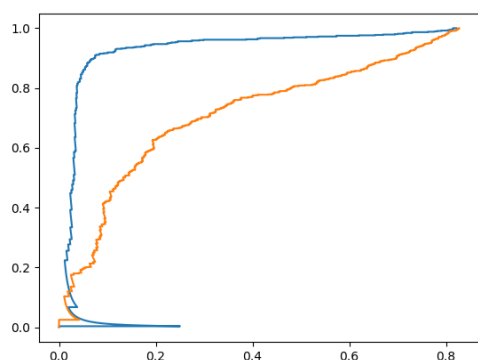

Figure 8: The receiver operating characteristic (ROC) curve for all four datasets.

into the neural network. For each of the tables, the top left square gives the number of segments which should merge that we predict merge. The bottom right square gives the number of segments which should not merge that we predict should not merge. The bottom left quadrant indicates the amount of false positives, candidates which we predict merge which do not. The top right quadrant gives the amount of false negatives where we predict split although the candidates should merge. Figure 8 shows the receiver operating characteristic (ROC) curve for all four datasets.

5.3. Graph Based Strategies

Using a graph-based optimization strategy prevents XX segments from merging compared to the simple hierarchical agglomeration strategy with an optimal threshold cut off. Since correcting merge errors is significantly more difficult than correcting split errors, it is desirable to limit the number of false merges. The graph-based strategies significantly improve this error type.

5.4. Variation of Information Improvements

The final metric for comparing two connectomics segmentations is computing the variation of information with respect to an expert-labeled ground truth dataset. Figure 9 shows the results on the datasets compared to the baseline (green) and an oracle (blue).

6. Conclusions

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, 2

[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, 4

[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] R. W. Conners, M. M. Trivedi, and C. A. Harlow. Segmentation of a high-resolution urban scene using texture operators. *Computer Vision, Graphics, and Image Processing*, 25(3):273–310, 1984. 2

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

[9] 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

[10] 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

[11] 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

[12] 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

[13] A. Horňáková, J.-H. Lange, and B. Andres. Analysis and optimization of graph decompositions by lifted multicuts. In *International Conference on Machine Learning*, pages 1539–1548, 2017. 2

[14] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstädter, K. Briggman, W. Denk, J. Bowden, J. Mendenhall, W. Abraham, K. Harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and S. Seung. Boundary learning by optimization with topological constraints. In *Proc. IEEE CVPR 2010*, pages 2488–2495, 2010. 2

[15] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain. Flood-filling networks. *arXiv preprint arXiv:1611.00421*, 2016. 2
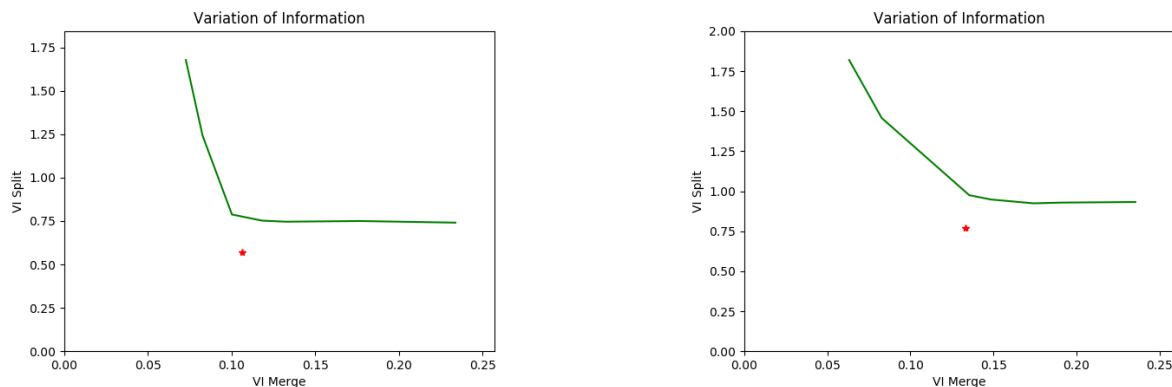
Figure 9: The improvement on variation of information from the baseline NeuroProof segmentation (green). Results closer to the origin are better.

[16] 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

[17] 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, 4

[18] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *Medical image analysis*, 22(1):77–88, 2015. 2

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

[20] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1751–1759, 2015. 2

[21] 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, 2016. (available on arXiv:1611.06973 [cs.CV]). 2

[22] 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

[23] 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

[24] 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

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

[26] 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

[27] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002. 2

[28] 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

[29] 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

[30] 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

[31] X. Ren and J. Malik. Learning a classification model for segmentation. In *null*, page 10. IEEE, 2003. 2

[32] 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, 5

[33] 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

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

[35] S.-y. Takemura, Y. Aso, T. Hige, A. Wong, Z. Lu, C. S. Xu, P. K. Rivlin, H. Hess, T. Zhao, T. Parag, et al. A connectome of a learning and memory center in the adult drosophila brain. *Elife*, 6, 2017. 4

CVPR
#446

CVPR
#446

CVPR 2018 Submission #446. CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

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

[37] A. Vázquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister. Segmentation fusion for connectomics. In *Proc. IEEE ICCV*, pages 177–184, Nov 2011. 2

[38] 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

[39] A. Zlateski and H. S. Seung. Image segmentation by size-dependent single linkage clustering of a watershed basin graph. *arXiv preprint arXiv:1505.00249*, 2015. 2

[40] J. Zung, I. Tartavull, and H. S. Seung. An error detection and correction framework for connectomics. *CoRR*, abs/1708.02599, 2017. 2