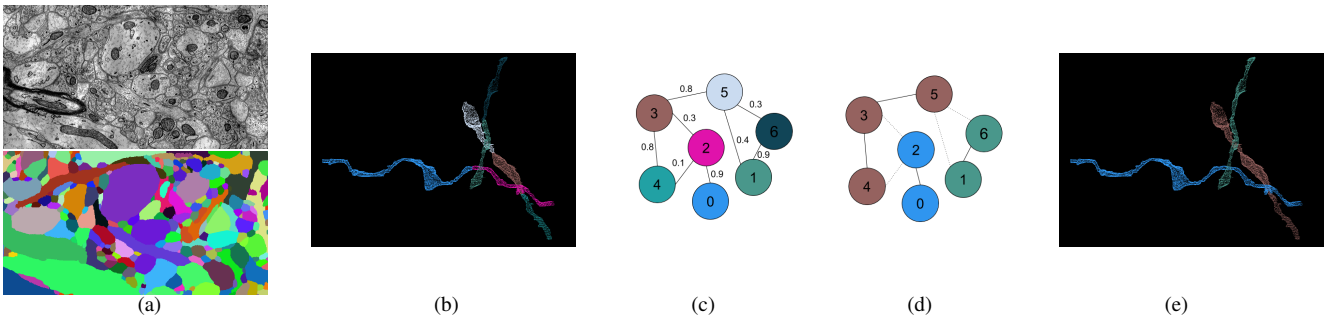
Top-Down Graph-Based Neuron Reconstruction

Anonymous CVPR submission

Paper ID 0446



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

Figure 1: Outline of our approach. (a) Raw EM image data (top) and results of pixel-based segmentation and agglomeration algorithms (bottom). (b) Extracted 3D skeleton from the segmentation. (c) Simplified 3D graph with error probabilities from trained classifier (schematic). (d) Result of graph partitioning algorithm with local and global geometric features (schematic). (e) Improved 3D reconstruction.

Abstract

Advancements in electron microscopy image acquisition have created massive connectomics datasets in the terabyte range that make manual reconstruction of neuronal structures infeasible. Current state-of-the-art automatic methods segment neural membranes at the pixel level followed by agglomeration methods to create full neuron reconstructions. However, these approaches widely neglect global geometric properties that are inherent in the graph structure of neural wiring diagrams. Instead, we follow bottom-up pixel-based reconstruction by a top-down graph-based method to more accurately approximate neural pathways. Using the membrane labels of the pixel-based segmentation we first generate skeletons in 3D. Using this 3D graph we train automatic classifiers for shape description to detect impossible neural pathways by looking at geometric properties. We then apply efficient graph-based optimization strategies to improve the segmentation labels. We demonstrate the performance of our approach on multiple real-world connectomics datasets with average variation of information improvement of $X\times$. The paper provides insights into learning shape features for top-down graph-based neuron reconstruction.

1. Introduction

The field of connectomics is concerned with reconstructing the wiring diagram of the brain at nanometer resolutions to enable new insights into the workings of the brain [16, 8]. Recent advancements in image acquisition using multi-beam serial-section electron microscopy (sSEM) have allowed researchers to produce terabytes of image data every hour [11]. It is not feasible for domain experts to manually reconstruct thousands or millions of neurons from this vast amount of image data [10]. State-of-the-art automatic reconstruction approaches use pixel-based segmentation with convolutional neural networks (CNNs) followed by agglomeration strategies [19, 24, 27, 40, 20, 29]. These *bottom-up pixel-based* algorithms produce excellent results but still fall short of acceptable error rates for large volumes.

We present a *top-down graph-based* method that builds on the outputs of bottom-up pixel-based segmentation approaches (Fig. 1a). We first extract 3D skeletons from the input segmentation (Fig. 1b) and generate a simplified 3D graph (Fig. 1c). We train a 3D CNN classifier on the agglomerated regions in the segmentation data to detect errors. During test time, we run the classifier to populate the graph edge weights with error probabilities. We then use a graph optimization algorithm to partition the graph into the final reconstruction by enforcing domain-specific global constraints from the underlying biology (Fig. 1d).

Our approach operates at a level of abstraction above existing pixel-based methods. This allows us to leverage

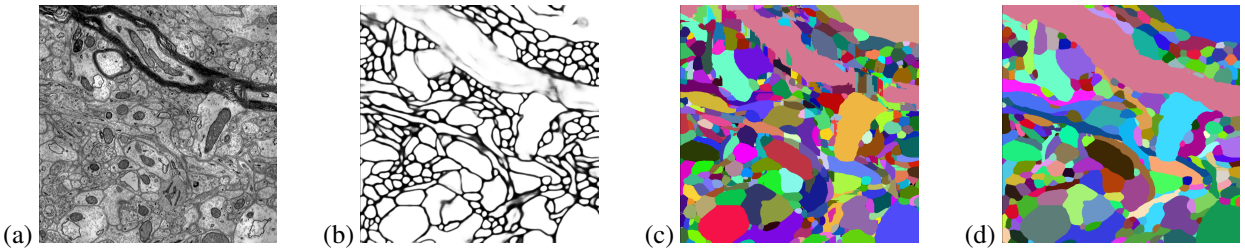


Figure 2: An overview of a pixel-based connectomics segmentation pipeline. (a) Original EM image data (b) Output of a CNN predicting voxel affinities (c) Clustering of the affinities using a watershed algorithm (d) Agglomeration of the supervoxels into larger segments.

both local and global information to produce more accurate reconstructions. Because it uses agglomerated regions our classifier is independent of image resolution and acquisition parameters, enabling its application to isotropic and anisotropic image data without retraining. Using the 3D graph of the segmentation allows us to enforce global constraints on the reconstruction that closely follow the underlying biology, such as limiting the angles of neuron branchings to the biologically plausible range. We can also use local information such as shape priors to weight the edges in our graph. This dual approach of assessing local decisions in a global context yields accuracy improvements over the existing reconstruction methods.

This work makes the following following contributions: (1) a novel top-down graph-based method for neural reconstruction of connectomics data; (2) a region-based CNN classifier to detect errors under global constraints; (3) an empirical evaluation of our method on several connectomics datasets; (4) our method yields improved performance over the state-of-the-art on X out of Y datasets, and competitive results on the remaining data. On average, we improve the Variation of Information (VI) metric across all datasets by X percent without drastically increasing the running time.

2. Related Work

A significant amount of research in computer vision focuses on the segmentation of images [38]. Here, we review some of the most successful methods that have been applied to large-scale EM images in connectomics. Fig. 2 shows the results of a typical connectomics segmentation pipeline.

A large amount of connectomics research considers the problem of extracting segmentation information at the pixel (i.e., voxel) level from the raw EM images. Some early techniques apply computationally expensive graph partitioning algorithms with a single node per pixel [1]. However these methods do not scale to terabyte datasets. More recent methods train classifiers to predict membrane probabilities per image slice either using 2D [4, 13, 17, 19, 37] or 3D CNNs [20, 29, 36].

Oftentimes these networks produce probabilities for the affinity between two voxels (i.e., the probability that adjacent voxels belong to the same neuron). The MALIS cost function is specifically designed for generating affinities that produce good segmentations [2]. HP: add seung ref More recently, flood-filling networks produce segmentations by training an end-to-end neural network that goes from EM images directly to label volumes [14]. These flood-filling networks produce impressive accuracies but at a high computational cost.

Several pixel-based approaches generate probabilities that neighboring pixels belong to the same neuron. Often a watershed algorithm will cluster pixels into small superpixels [40]. Many methods build on top of these strategies and train random-forest classifiers to produce region-based (i.e., super-pixel) segmentations [19, 24, 26, 27, 40].

Some recent research builds on top of these region-based methods to correct errors in the segmentation [28, 41, 9]. However, to our knowledge, our method is the first to extract a 3D graph from pixel-based agglomerated segmentations for a true top-down reconstruction approach. This allows us to enforce domain-specific constraints using graph-based partitioning algorithms. Many segmentation and clustering algorithms use graph partitioning techniques [1] or normalized cuts for traditional image segmentation [15, 32, 34]. Even though graph partitioning is an NP-Hard problem [5] there are several useful multicut heuristics that provide good approximations with reasonable computational costs [12]. We use the method of Keuper et al. to partition the extracted 3D graph into the final neuron reconstruction [18].

3. Method

There are two types of errors that can occur in connectomics segmentation. The first, called a split error, occurs when there are two segments that should have been merged. The second, called a merge error, happens when one segment should be split into two. Generally, it is much more difficult to correct merge errors than to correct split errors [25], which is why most reconstruction approaches are tuned towards over-segmentation with many

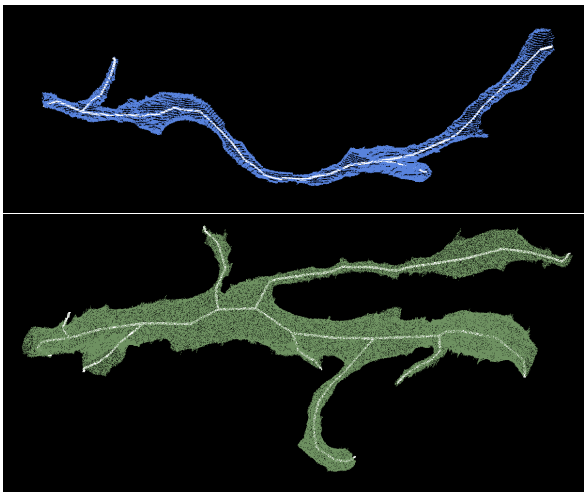


Figure 3: Skeletons using the TEASER algorithm.

more split than merge errors. Our method takes as input over-segmentations of EM image volumes generated by state-of-the-art connectomics reconstruction pieplines (Sec. 4.1). Our goal is to identify locations of split errors and merge the corresponding segments automatically.

From the input segmentation we generate a graph G with nodes N and edges E with weights w_e. The nodes correspond to label segments from the segmentation with edges between segments considered for merging. Ideally, our graph has edges corresponding to all of the segments which were erroneously split with few edges between correctly split segments. We generate a skeleton for every segment based on the intuition that a skeleton represents a simplified representation of the overall shape of the neuron. From this skeleton we quickly identify potential merge locations and produce the corresponding edges for the graph. We train a CNN to generate the edge weights by learning a merge function from ground truth data. A multicut heuristic generates a partition on the graph where nodes in the same partition are assigned the same output label. We will now discuss the three major components to our framework (skeletonization, CNN, and graph-partition) in more detail. HP: stopped here

3.1. Graph Creation

We generate nodes N and edges E to apply a graph-based optimization strategy for segmentation. In addition, these edges receive 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, some of the millions of labels in the volume correspond to very small volumes. Usually these locations have noisy image data so the pixel-based methods could not provide enough information to generate a larger segment. It is difficult to extract useful shape features from these segments because of their small, and often random, shape. We prune these nodes from the graph by removing all segments with fewer than $20,000$ voxels. This removes XX% of segments on a typical connectomics dataset.

3.1.2 Edge Generation

A naïve approach to generating edges produces an edge between all adjacent segments. Two segments l_1 and l_2 are considered adjacent if there is a pair of adjacent voxels where one has label l_1 and the other has label l_2. NeuroProof and GALA consider all pairs of adjacent segments for merging. For us, this method produces too many edges in the graph. Therefore we present the following algorithm to identify pairs of segments to consider for merging.

First, we extract a skeleton from each segment using the TEASER algorithm [30, 39]. Figure 3 shows the skeletons in white of two segments from the label volume. These skeletons consist of a sequence of *joints*, locations that are a local maximum distance from the segment boundary, with line segments connecting successive joints. We prune the joints that are within 50 voxels of each other to reduce unnecessary branching. For the purposes of our algorithm, joints that have only one connected neighbor are referred to as *endpoints*. Many of the segments that are erroneously split follow a similar pattern (Figure 4). In these split instances the two skeletons have nearby endpoints.

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 $\mathbb{S}'_e$ that includes all labels that have a single voxel within t_{low} voxels from e. Elements of these sets are candidates for merging. However the first pass often has too many candidates that should remain split so we apply an additional pass for further pruning. 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} voxels 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.

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.

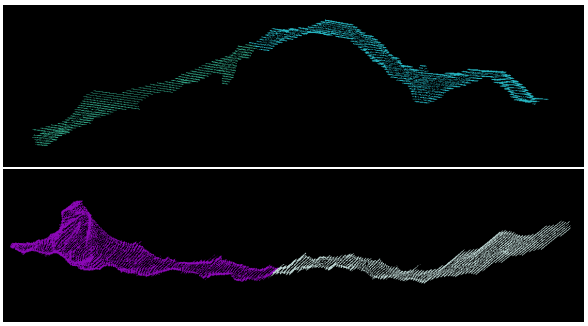


Figure 4: Two erroneously split segments.

3.2.1 Network Architecture

Our graph generation 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 CNN using the oversegmentation and the corresponding manually labeled ground truth data (Section 4.1). We extract a cubic region of interest around these locations as input to the CNN. These regions of interest provide the local information for the neural network to predict which neighboring segments belong to the same neuron.

The networks receives three input channels for every voxel in the region of interest around segments l_1 and l_2. The input to all of the channels is in the set $\{-0.5, 0.5\}$. The first channel is 0.5 only if the corresponding voxel has label l_1. The second channel is 0.5 only if the corresponding voxel has label l_2. The third channel is 0.5 if the corresponding voxel is either l_1 or l_2.

Figure 5 provides an overview of our architecture. Our network architecture has three layers of double convolutions followed by a max pooling step following the work of Chatfield et al. that found this technique improves on single convolution layers [3]. The first max pooling layer is anisotropic with pooling only in the x and y dimensions. The output after this final pooling step is flattened into a 1-D vector which is input into two fully connected layers. The final layer produces a probability with a sigmoid activation function [6]. All of the other activation functions are LeakyReLU [21]. We use a stochastic gradient descent optimizer with Nesterov's accelerated gradient [23]. There are dropouts of 0.2 after every pooling layer and the first dense layer, and a dropout of 0.5 after the final dense layer to prevent overfitting. We discuss all other network parameters in Section 4.3.

3.3. Graph Partitioning

After constructing the graph structure we apply a graph-based segmentation strategy. There are many graph partitioning minimization functions that provide different constraints on their output. Neurons in the brain should be acyclic (i.e. the output shape should have a genus of zero). We enforce this constraint by finding a multicut partition of the graph that generates a *forest* on the nodes. A forest is a partitioning of a graph into a set of trees (i.e. no segment has a cycle). There are several heuristics that solve the multicut problem (which is NP-Hard). We use the greedy-additive algorithm [18].

4. Evaluation

4.1. Datasets

Kasthuri. The Kasthuri dataset images the neocortex of a mouse brain produced by a scanning electron microscope [16]. This dataset is $5342 \times 3618 \times 338$ voxels in size. The resolution of the dataset is $3 \times 3 \times 30\,\mathrm{nm}^3$ per voxel. We evaluate our methods using the left cylinder of this 3-cylinder dataset. We downsample the dataset in the x and y dimensions to give a final resolution of $6 \times 6 \times 30\,\mathrm{nm}^3$ per voxel. We divide the dataset into two volumes along the x dimension. Thus, each volume is $8.0 \times 10.9 \times 10.1\,\mu\mathrm{m}^3$.

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 in this species is the major site of associative learning [33]. The original dataset contains a $40 \times 50 \times 120\,\mu\mathrm{m}^3$ volume of which we use two cubes of size $10 \times 10 \times 10\,\mu\mathrm{m}^3$. The resolution of this dataset is $10 \times 10 \times 10\,\mathrm{nm}^3$ so each volume has 1000 voxels in each dimension.

Segmentation Pipeline and Baseline. The segmentation on the Kasthuri dataset was computed by agglomerating 3-D supervoxels produced by zwatershed from 3-D affinity predictions [40]. A recent study by Funke et.al. demonstrated superior performance of such methods over existing ones on anisotropic data [31]. We learned 3-D affinities using MALIS loss with a U-net [35, 29]. We apply the zwatershed algorithm, with suitable parameters, to compute a 3-D oversegmentation of the volume. The resulting 3-D oversegmentation is then agglomerated using the technique of context-aware delayed agglomeration to generate the final segmentation [26]. The variation of information (Section 4.5) values at different stopping thresholds of agglomeration correspond to the curve plotted in Figure 8.

We have collected two $1000 \times 1000 \times 1000$ voxel ($10 \times 10 \times 10\,\mu\mathrm{m}^3$) volumes from the authors [33]. Based on the authors' suggestion, we applied the context-aware delayed agglomeration algorithm [26] that shows improved performance on this dataset over the pipeline used in the original publication. This segmentation framework learns voxel and supervoxel classifiers with an emphasis to minimize undersegmentation error. At the same time this frame-

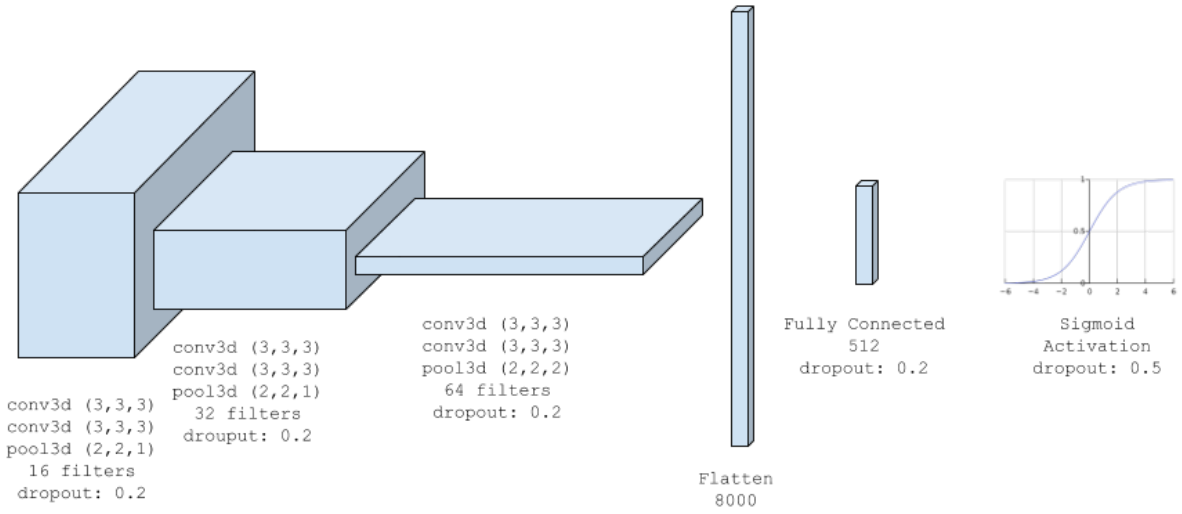


Figure 5: The architecture for the neural networks follows the *VGG* style of double convolutions followed by a max pooling operation. The number of filters doubles each layer leading to a fully connected layer and a sigmoid activation function.

work produces lower oversegmentation than standard algorithms. This algorithm first computes multi-channel 3-D predictions for membranes, cell interiors, and mitochondria, among other cell features. The membrane prediction channel is utilized to produce an oversegmented volume using 3-D watershed, which is then agglomerated hierarchically up to a certain confidence threshold. We used exactly the same parameters for the publicly available code for this algorithm and used different agglomerations thresholds to plot the variation of information error curve.

4.2. Pruning via Skeletonization

We prune potential merge candidates with the algorithm described in Section 3.1. There are two essential parameters to the algorithm: t_{low} and t_{high}. Ideally, the merge candidates output by this algorithm will contains all possible positive examples with a very limited number of negative examples. After considering various thresholds, we find that $t_{low} = 240\,\text{nm}$ and $t_{high} = 600\,\text{nm}$ produces the best results given this objective function. Note that these thresholds are in nanometers and not voxels. Connectomics datasets often have different sample resolutions and downsampling in z. Using nanometers allows us to have uniform units across all of these datasets and calculate the thresholds in voxels at runtime. The thresholds are $t_{low} = (40, 40, 8)$ voxels and $t_{high} = (100, 100, 20)$ voxels for the Kasthuri dataset and $t_{low} = (24, 24, 24)$ voxels and $t_{high} = (60, 60, 60)$ voxels for the FlyEM dataset. The thresholds are anisotropic or isotropic depending on the corresponding isotropy of the dataset. Table 2 shows the overall success of this method of candidate pruning.

4.3. Classifier Training

We use the Kasthuri Vol. 1 dataset for training and validation. We train on 80% of the potential merge candidates for this volume. We validate the neural network classifier on the remaining 20% of candidates. We consider networks with varying input sizes, optimizers, loss functions, filter sizes, learning rates, and activation functions. The supplemental material includes information on the experiments that determined these final parameters. Table 1 provides the parameters in the final network. There are 7,294,705 learnable parameters in our final architecture. All the parameters are randomly initialized following the Xavier uniform distribution [7]. Training concluded after 34 epochs.

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: Parameters for the CNN

Data Augmentation. We apply 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 mirroring along the x and z axes. This produces 16 times more training data.

Dataset	Baseline	After Pruning
Kasthuri Vol. 1	763 / 21242 (3.47%)	753 / 3459 (17.88%)
Kasthuri Vol. 2	1010 / 26073 (3.73%)	904 / 4327 (17.28%)
FlyEM Vol. 1	269 / 14875 (1.78%)	262 / 946 (21.69%)
FlyEM Vol. 2	270 / 16808 (1.58%)	285 / 768 (27.07%)

Table 2: The results of our pruning heuristic compared to the current baseline.

4.4. Graph-based Strategies

Applying a top-down globally optimal partitioning function allows us to enforce some domain-specific constraints on the reconstruction. In connectomics, we want to enforce the topological restrictions that each partition in the graph has a genus of zero. The greedy-additive heuristic solves the multicut graph partitioning problem by enforcing this constraint. Previous agglomeration strategies at the per-region level consider neighboring superpixels in order of merge probability. These superpixels are clustered hierarchically without concern for global constraints. We compare the benefits of a top-down partitioning function versus the traditional bottom-up clustering methods.

4.5. Variation of Information

We evaluate the performance of our methods using the split version of variance of information [22]. Consider a ground truth labeling GT and our automatically reconstructed segmentation SG. Over and undersegmentation are quanitified by the conditional entropy $H(GT|SG)$ and $H(SG|GT)$ respectively. Since we are measuring the entropy between two clusterings, better split variation of information scores are closer to the origin.

5. Results

5.1. Pruning via Skeletonization

Table 2 shows the results of pruning using the skeletonization heuristic. The baseline algorithm considers all adjacent regions for merging. Our method removes a significant portion of these candidates while maintaining a large number of the true merge locations. This edge pruning is essential for the graph partitioning algorithm which has a computational complexity dependence on the number of edges. Our pruning heuristic removes at least $6\times$ the number of edges between correctly split segments on all datasets, achieving a maximum removal ratio of $20\times$. Equally important is the number of split errors that remain after pruning. These are the locations that we want to merge to create a more accurate reconstruction. For every dataset, the number of positive candidates remains relatively even. However, since our heuristic does not enforce an adjacency constraint of two regions when constructing edges in the graph, the difference does not indicate the number of examples excluded by pruning. In fact, our method finds a number of examples which are non-adjacent. Figure 6 shows two example segments which are split errors. The top example our algorithm missed but the segments are adjacent. The bottom example our algorithm found despite the fact that they are not adjacent.

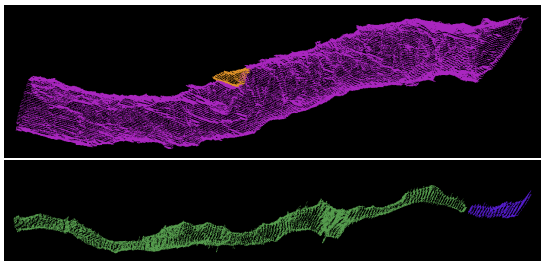


Figure 6: Example merge candidates.

Dataset	Precision	Recall	Accuracy
Kasthuri Training	0.919	0.936	0.974
Kasthuri Testing	0.737	0.717	0.907
FlyEM Vol. 1	0.796	0.478	0.862
FlyEM Vol. 2	0.762	0.422	0.810

Table 3: Precision and recall for the training and three test datasets.

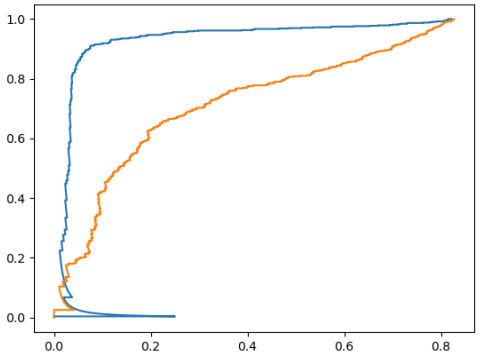


Figure 7: (NOT CORRECT) The receiver operating characteristic (ROC) curve for all four datasets.

5.2. Classification Performance

Table 3 shows the precision and recall for all of the datasets. Since, our method does not rely on the image data, we can train the network on an anisotropic dataset and get impressive results on an isotropic dataset. Figure 7 shows the receiver operating characteristic (ROC) curve for all four datasets.

Dataset	VI Split	VI Merge
Kasthuri Training		
Kasthuri Testing		
FlyEM Vol. 1		
FlyEM Vol. 2		

Table 4: Variation of Information Change

TODO - ADD QUALITATIVE RESULTS

5.3. Graph Based Strategies

TODO - QUANTIFY IMPROVEMENT BY MULTICUT - READERS CAN SKIP SECTION FOR NOW

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

We compute the variation of information for our segmentations against the expert-labeled ground truth datasets. The input segmentation to our network serves as a baseline. We evaluate NeuroProof (our input) at several different thresholds to create a variation of information curve. Similarly, we run the greedy-additive heuristic with varying parameters that increase the probability needed for merging nodes. Figure 8 shows the results on the datasets compared to the baseline (green) and an oracle (blue). The oracles sees the entire constructed graph from our algorithm and correctly partitions the graph based on the ground truth. Our methods decreases the split variation of information by a factor of XX% and only increases the merge variation of information by a factor of XX% on the test datasets. Table 4 provides a quantitative overview for the changes in variation of information.

TODO - ADD QUALITATIVE RESULTS

6. Conclusions

We present a novel method for reconstructing neuronal processes in connectomics image datasets. We extend on top of existing region-based agglomeration strategies and show significant accuracy improvement on two datasets from two different species. By extracting skeletons from every segment in the input we can quickly produce a graph G with N nodes and E weighted edges from the input. We populate the edge weights using a 3-D CNN that, unlike previous methods, does not use the raw image data. Separating the neural network from the images allows us to train on anisotropic data and test on isotropic data with different sample resolution. This is exceptionally important in connectomics because of the extreme labor-intensive cost of manually segmenting ground truth data. We use existing graph partitioning strategies on the G, enabling us to enforce domain-constraints on the resulting segmentation. For us, our constraints follow from the topology of neurons. However, this framework extends to other domains with varying constraints.

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] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga. Maximin affinity learning of image segmentation. In *Advances in Neural Information Processing Systems*, pages 1865–1873, 2009. 2

[3] 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. 4

[4] 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

[5] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006. 2

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

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 5

[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. 1

[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. 1

[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

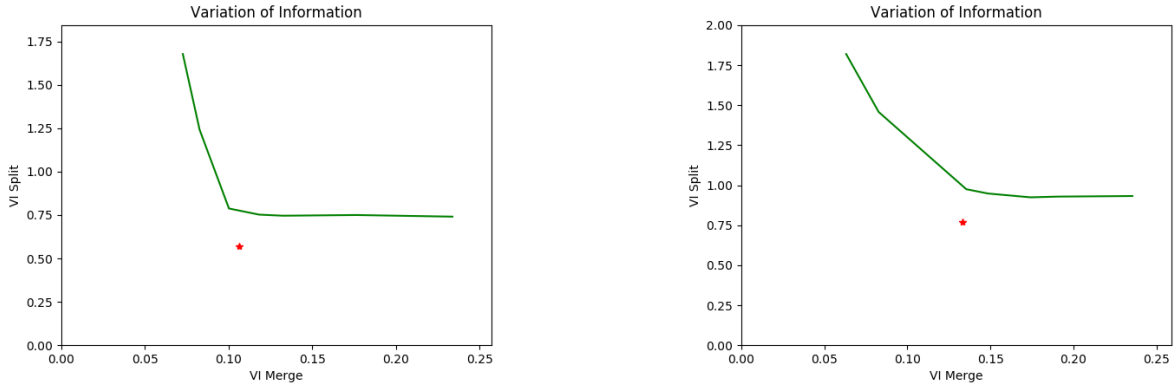


Figure 8: The improvement on variation of information from the baseline NeuroProof segmentation (green). Results closer to the origin are better.

[12] 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

[13] 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

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

[15] 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

[16] 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

[17] 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

[18] 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, 4

[19] 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. 1, 2

[20] 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

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

[22] M. Meila. Comparing clusterings by the variation of information. In *Colt*, volume 3, pages 173–187. Springer, 2003. 6

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

[24] 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

[25] T. Parag. What properties are desirable from an electron microscopy segmentation algorithm. *arXiv preprint arXiv:1503.05430*, 2015. 2

[26] T. Parag, A. Chakraborty, S. Plaza, and L. Scheffer. A context-aware delayed agglomeration framework for electron microscopy segmentation. *PLOS ONE*, 10(5):1–19, 05 2015. 2, 4

[27] 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. 1, 2

[28] D. Rolnick, Y. Meirovitch, T. Parag, H. Pfister, V. Jain, J. W. Lichtman, E. S. Boyden, and N. Shavit. Morphological error detection in 3d segmentations. *arXiv preprint arXiv:1705.10882*, 2017. 2

[29] 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, 4

[30] 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. 3

[31] P. Schlegel, M. Costa, and G. S. Jefferis. Learning from connectomics on the fly. *Current Opinion in Insect Science*, 2017. 4

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

[33] S.-y. Takemura, Y. Aso, T. Hige, A. M. Wong, Z. Lu, C. S. Xu, P. K. Rivlin, H. F. Hess, T. Zhao, T. Parag, S. Berg, G. Huang, W. T. Katz, D. J. Olbris, S. M. Plaza, L. A. Umayam, R. Aniceto, L.-A. Chang, S. Lauchie, and et al. A connectome of a learning and memory center in the adult drosophila brain. *eLife*, 6:e26975, 2017 Jul 18 2017. 4

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

[35] S. Turaga, K. Briggman, M. Helmstaedter, W. Denk, and S. Seung. Maximin affinity learning of image segmentation. In *Advances in Neural Information Processing Systems 22*. 2009. 4

[36] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural computation*, 22(2):511–538, 2010. 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] N. M. Zaitoun and M. J. Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015. 2

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

[40] 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. 1, 2, 4

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