Segmentation of Electron Microscopy Images for Connectomics

Brian Matejek
Advisor: Hanspeter Pfister
Harvard University
bmatejek@seas.harvard.edu

1. Introduction

One of the critical components of connectomics—the field concerned with reconstructing the wiring diagram of the brain at nanometer resolution—is automatic segmentation of electron microscopy (EM) images. With recent advancements in EM acquisition techniques, neuroscientists can now generate a terabyte of image data every hour [44]. These images are typically anisotropic with 4nm resolution in the xy plane and 30 to 40nm between slices. At this resolution, we can see the axon terminals and dendritic spines, and the synaptic connections between them. Neuroscientists hope to further the understanding of the brain's underlying circuitry with accurate reconstructions of every neuron and classification of every synapse.

An ambitious manual reconstruction effort in the 1980s resulted in the first complete connectome of any animal, the *Caenorhabditis elegans* worm [56]. This species has 302 neurons and manual labeling required several years. More recent studies focus on *Drosophila* flies [23, 50], rodents [44], and even humans [47]. With an EM throughput of one terabyte per hour, neuroscientists can image a cubic millimeter of data (two petabytes) in six months [48]. At this scale, the manual reconstruction techniques of the 80s are infeasible. Thus, researchers rely on machine learning methods to segment these massive datasets into label volumes (Fig. 1). These label volumes assign a 32- or 64-bit integer to every voxel where voxels have the same label only if they belong to the same neuron.

Uncompressed, these label volumes are larger in size than the already massive image datasets. In our paper published in *MICCAI 2017*, we explore the effectiveness of existing general-purpose, image, and video compression techniques on these segmentation datasets [32]. We find that these existing techniques fail to adequately exploit the typical properties of these label volumes. Thus, we propose *Compresso*, a new compression algorithm specifically tailored for large segmentation datasets, which achieves better compression ratios than all existing methods.

Automatic reconstruction techniques need to be fast, scalable, and accurate. Ideally, image acquisition is the bottleneck of the connectomics pipeline so reconstruction should occur at a rate of one terabyte per hour [16]. We can achieve this throughput with parallelization among several GPUs with existing techniques [13, 38]. However, these automatic methods typically rely on local context for decision making and are agnostic about the underlying biological systems. Thus, they often make errors at scale and currently require human proofreading or other error correction techniques [17, 61].

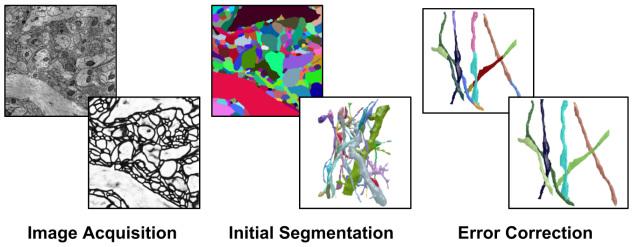

Figure 1: From an EM image stack, 3D convolutional neural networks generate affinities between voxels (left). A watershed algorithm agglomerates the voxels into supervoxels using these affinities, and these supervoxels are further merged to form a complete segmentation (center). These methods often produce errors and require error correction algorithms to improve the accuracy (right).

We propose a novel region merging framework that takes as input an oversegmentation of an EM image stack (under review, *ECCV 2018*). Our method imposes both local and global biological constraints onto the output segmentation to more closely match the underlying structure of neuronal processes. Additionally, our method is independent of image resolution and acquisition parameters, enabling its application to isotropic and anisotropic datasets without retraining. Images generated by electron microscopes often differ in appearance because of variations in staining techniques [7]. By removing the dependence of our algorithm on the input images, we greatly reduce the need for additional costly ground truth data for each new stack of images.

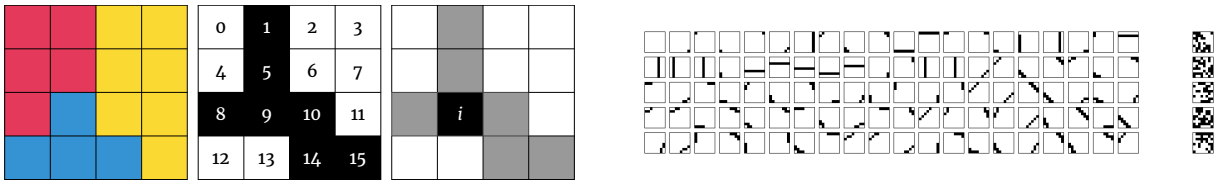

Figure 2: Compresso works by dividing up the segmentation data into small blocks. On the left we show the intersection of three segments in a $4 \times 4 \times 1$ window. We extract a boundary map from this segmentation to transform the window into an integral value between 0 and $2^{16} - 1$. This window has an encoded value of 50,978 $(2^1 + 2^5 + 2^8 + 2^9 + 2^{10} + 2^{14} + 2^{15})$. The location i requires some additional bookkeeping. On the right are the 100 most common $8 \times 8 \times 1$ windows accounting for $\sim 82\%$ of the volume on a representative dataset.

2. Compression of Label Volumes

An often overlooked aspect of these segmentation datasets is how to efficiently store the label volumes. These label volumes are often 32- or 64-bit to account for the massive number of neurons that interweave through a cubic millimeter. Uncompressed, a cubic millimeter of segmentation data is nearly 20 petabytes. Thus, compression techniques are needed for fast transmission and cost-efficient storage.

2.1. Related Works

General-purpose compression schemes [11, 12, 14, 31, 35, 39, 45, 53, 55, 59] are not optimized for this data. These methods do not exploit the typical characteristics of label volumes such as large invariant regions without natural relationship between label values. These properties render 2D image compression schemes inadequate since they rely on frequency reduction (using e.g., wavelet or discrete cosine transform) and value prediction of pixels based on local context (differential pulse-code modulation) [40, 46]. Color space optimization strategies in video codecs [1] also have no effect on label volumes, even though the spatial properties of a segmentation stack (z-axis) are similar to the temporal properties of video data (time-axis). A compression scheme designed specifically for label volumes is part of the visualization software Neuroglancer [15]. This method exploits segmentation homogeneity by creating small blocks with N labels and reducing local entropy to $\log_2 N$ per pixel. Lookup tables then decode the values $[0, N)$ to the original 64-bit labels.

2.2. Method

We propose Compresso, which is specifically designed for compression of EM label volumes [32]. Compresso works by decoupling the two important components across the image stack: per-segment shape and per-pixel label. To encode the segment shapes, we consider the boundary pixels between two segments. Removing the per-pixel labels, we produce a boundary map for each slice where a pixel (x, y, z) is 1 if either pixel at $(x + 1, y, z)$ or $(x, y + 1, z)$ belongs to a different segment (Fig. 2, left). The boundary map is divided into non-overlapping congruent 3D windows. If there are n pixels per window, each window w is assigned an integer $V_w \in [0, 2^n)$ where V_w is defined as:

$$V_w = \sum_{i=0}^{n-1} \mathbb{I}(i) 2^i, \tag{1}$$

and $\mathbb{I}(i)$ is 1 if pixel i is on a boundary and 0 otherwise. Figure 2, left, shows an example segmentation with a window size of $4 \times 4 \times 1$.

A priori, each window could take any of 2^n distinct values, and therefore require n bits to encode without further manipulation. However, boundaries in segmentation images are not random, and many of these values never appear. Indeed, we find that a small subset of high-frequency V_w values accounts for most windows, allowing for significant compression. Figure 2, right, shows the 100 most common windows for a representative connectomics dataset. These 100 frequently occurring windows account for approximately 82% of the over 1.2 million V_w values in this dataset. Nearly all of these windows correspond to simple lines traversing through the window. For contrast, we also provide 5 randomly generated windows that never occur in the dataset.

We define N as the number of distinct V_w representing all of the windows in an image stack. We construct an invertible function $f(V_w) \rightarrow [0, N)$ to transform the window values into a smaller set of integers. For all real-world segmentations $N \ll 2^n$; however, we assume no constraint on N in order to guarantee lossless compression. With this function, each V_w requires $\log_2 N$ bits of information to encode. This is fewer than the initial number of bits so long as $N \leq 2^{n-1}$. Therefore, when compressing the label volumes, we can store $f(V_w)$ for every window as well as a lookup function for the N unique $f(V_w)$ values to the original integral value.

So far we have focused exclusively on transforming the boundary map of an image segmentation. However, the per-pixel labels themselves are equally important. The bound-

ary map divides each image slice into different segments. By design, all pixels in the same segment have the same label so we store only one label per segment for each slice. We use a connected-component labeling algorithm to identify each component and store one label per segment [19].

Thus far, we have assumed the boundaries provide enough information to reconstruct the entire segmentation. Pixels not on a segment boundary are easily relabeled. However, more care is needed for pixels on the segment boundaries. Consider figure 2, left, which depicts a difficult boundary to decode. If a boundary pixel has a non-boundary neighbor to the left or above, then that pixel merely takes on the value of that neighbor. However, pixel i requires more care since its relevant neighbors are both boundary pixels. We simply just store the label values for indeterminate pixels like i.

To decompress the data, we first reconstruct the boundary map from the windows that we extracted. From there, we can run the same deterministic connected-components algorithm per slice. We can traverse through the saved labels to fill in all non-boundary labels. To determine the per-pixel labels for every boundary pixel, we iterate over the entire dataset in raster order. Any boundary pixel (x, y, z) with a non-boundary neighbor at $(x-1, y, z)$ or $(x, y-1, z)$ shares the same per-pixel label. If both relevant neighbors are boundaries (i.e., like pixel i in Fig. 2) we extract the value we previously stored.

2.3. Results

Our compression scheme outperforms all existing methods. We follow our scheme with LZMA which uses sophisticated probabilistic bit prediction strategies and together they achieve ratios of $920\times$ on average. This outperforms existing strategies by over 80%. The fundamental principles guiding Compresso are valid for a diverse set of segmentation datasets and we improve on existing methods for MRI and image segmentation datasets. We compressed an 18 terabyte label volume of 100 microns cubed to 26.2 gigabytes, a ratio of $\sim 687\times$.

3. Biologically-Constrained Region Merging

Automatic reconstruction methods need to be fast enough to scale to petabyte datasets while still maintaining a high-level of accuracy. The most accurate methods, like flood-filling networks [22], are currently far too slow. Unfortunately, methods that can scale to petabyte datasets typically rely on only local context and can produce errors at scale [16]. Therefore, it is necessary to employ error correction strategies on the segmentations from current reconstruction pipelines. We propose a biologically-constrained region merging algorithm to correct *split errors* in the initial segmentation.

3.1. Related Works

A significant amount of connectomics research considers the problem of extracting segmentation information at the voxel level of EM images. First, intermediate representations like boundary, affinity or binary segmentation maps are generated from the voxels. Random forests with hand-designed features [25], or 2D and 3D convolutional networks produce boundary probabilities [5, 10, 20, 27, 42, 54]. Often, the affinity between each voxel and its six neighbors are used [9, 29, 30, 38, 52]. The 3D U-Net architecture has become popular [9] and the MALIS cost function is specifically designed to re-weight affinity predictions by their contribution to the segmentation error [6]. More recently, flood-filling networks [22] produce binary segmentations from raw pixels with a recurrent convolutional network at a high computational cost. Orthogonal to the representation, model averaging [57] and data augmentation [30] methods can further improve the performance, where Lee et al. [30] surpass the estimated human accuracy on the SNEMI3D dataset.

Clustering techniques transform these intermediate representations into segmentations. Some early methods apply computationally expensive graph partitioning algorithms with a single node per superpixel [3]. Several pixel-based approaches generate probabilities that neighboring pixels belong to the same neuron. Often a watershed algorithm will then cluster these pixels into super-pixels [60].

Region merging methods can be categorized by the similarity metric between adjacent segments and the merging strategy. For the similarity metric, Lee et al. and Funke et al. rely solely on the accuracy of the predicted affinities and define the metric as the mean affinity between segments [13, 30]. Classification-based methods generate the probability to merge two segments from hand-crafted [21, 27, 34, 37, 38, 60] and learned features [5]. For the merging strategy, most methods use variants of hierarchical agglomeration [27, 34, 37, 38, 60] to greedily merge a pair of regions at a time. Jain et al. formulates the agglomeration problem as a reinforcement learning problem [21] and Pape et al. present a scalable multicut algorithm to partition superpixels with global optimization [4].

Additional research builds on top of these region-based methods to correct errors in the segmentation. These input segmentations can contain two types of errors. In a *split error*, one neuron contains multiple labels. In a *merge error*, two or more neurons receive the same label. This can be done either using human proofreading [17, 18, 28] or automatically [41, 61]. In both cases, available methods are pixel-based and do not include global biological constraints into the decision making process. Our method can take as input any existing segmentation pipeline.

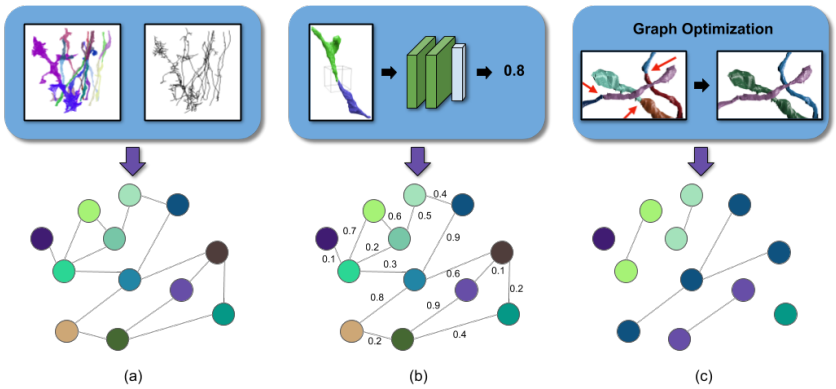

Figure 3: Our framework uses both geometric and topological biological constraints for region merging. We (a) generate a simplified graph from the skeleton representation of the input segmentation, (b) train a classifier to learn the edge weight based on the shape of the segment connection, and (c) perform graph partitioning with acyclic constraints.

3.2. Method

For our proposed method (Fig. 3), from the input segmentation we generate a graph G with nodes N and edges E with weights w_e. The nodes correspond to labeled segments from the input data with edges between merge candidates. We propose the following three steps to formulate and then partition the graph while obeying constraints from the underlying biology. First, we only consider merging segments based on a skeletonized representation of the input segmentation (Fig. 3a). This enables us to reduce the number of edges in the graph based on prior knowledge on the shape of neuronal processes. Second, we train a convolutional neural network that learns biological constraints based on the shapes of the input segmentation (Fig. 3b). This network generates probabilities that segments belong to the same neuron based only on the segmentations. An example of one such learned constraint is that neurons have small turning radii (Fig. 5). Third, we partition the graph using a lifted multicut formulation with additional acyclic constraints to enforce global biological constraints (Fig. 3c). The lifted multicut solution is globally consistent which we then augment to produce a tree-structured graph.

3.2.1 Skeleton-Based Graph Generation

Most region merging methods create a region graph by removing small-sized segments and linking each pair of adjacent segments, which can still lead to a large graph size due to the irregular shape of neural structures. We use a skeleton representation of the segmentation to reduce the graph size with geometric constraints on the connectivity of two adjacent segments to prune edges.

Our key observation is that if two segments belong to the same neuronal process, their skeleton end points should satisfy certain geometric constraints. To extract the skeleton from each segment, we use a variant [58] of the TEASER algorithm [43]. Fig. 4 shows four examples of extracted skeletons (in black). These skeletons consist of a sequence of *joints*, locations that are locally a maximum distance from the segment boundary, with line segments connecting successive joints. We refer to joints that have only one connected neighbor as *endpoints*. We find that approximately 70% of the segments that are erroneously split have nearby endpoints (Fig. 6).

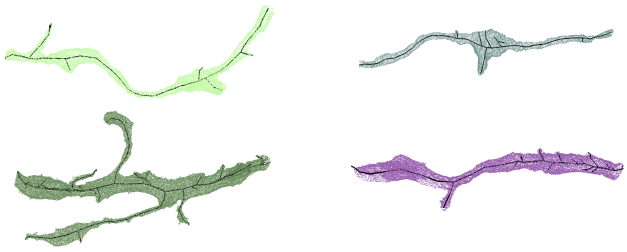

Figure 4: Example skeletons (in black) extracted from segments using a variant of the TEASER algorithm [43]. These skeletons not only capture the shape of the segmentation, but also provide endpoints useful for region merging proposals.

Two segments, s_1 and s_2, receive a corresponding edge if the two following conditions hold. First, endpoints in either s_1 or s_2 are within t_{low} nm of any voxel in the other segment. Second, there are endpoints in s_1 and in s_2 that are within t_{high} nm of each other. We store the midpoints between the two endpoints as the center of the potential merge in the set $\mathbb{S}_c$. This algorithm produces a set of segments to

Figure 5: Geometric constraints for region merging. We show two region merging proposals. On the left, the segments do not belong to the same neuron, as evidenced by the sharp turning radius indicated by the arrow; while on the right, the segments should be merged due to the continuity of the 3D shape. Instead of using handcrafted geometric features, we train a convolutional neural networks to automatically learn them from the ground truth labels.

consider for merging. Only these pairs have a corresponding edge in the constructed graph.

3.2.2 Learning-based Edge Weight

To predict the probabilities that two segments belong to the same neuron, we train a feed-forward convolutional network with three *VGG-style* convolution blocks [8] and two fully connected layers before the final sigmoid activation. For the input of the network, we extract a cubic region of interest (ROI) around each midpoint in $\mathbb{S}_c$ as input to the CNN. There are three channels corresponding to voxels belonging to label one, label two, or either label. We do not use the raw EM image information to avoid the need to retrain the network on datasets that have been stained differently or imaged at different resolution. This reduces the need for generating costly manually-labeled ground truth.

3.2.3 Optimization-Based Graph Partition

After constructing the graph we seek to partition it into labels where every label corresponds to a neuronal process. We formulate this graph partitioning problem as a multicut problem. There are two primary benefits to using a multicut formulation. First, the number of segments in the final graph is not predetermined but depends on the input. Second, this minimization produces globally consistent solutions (i.e., a boundary remains only if the two corresponding nodes belong to unique segments) [26].

We apply the algorithms of Keuper et al. [26] to produce a feasible solution to the multicut problem using greedy additive edge contraction. Following their example, we employ the generalized lifted multicut formulation. Traditional multicut solutions only consider the probabilities that two adjacent nodes belong to the same segment. In the lifted extension to the problem, we can penalize non-adjacent nodes that belong to different segments. These penalties between

Figure 6: Three erroneously split segments. In each instance, the skeletons for each segment have nearby endpoints indicating a split error.

non-adjacent nodes are called lifted edges. Ideally these lifted weights represent the probability that two nodes belong to the same neuron. However, determining such probabilities is computationally expensive. We approximate these probabilities by finding the maximal probable path between any two nodes using Dijkstra's algorithm [26]. This is an underestimate of the probability that two nodes belong to the same neuron since it does not consider all possible paths. Since our graphs are sufficiently small, we can generate lifted edges between all pairs of nodes.

We need to convert the probabilities into the following weighting scheme to solve the multicut problem with the greedy-edge heuristic [2, 26]. Given the probability that the nodes belong to the same neuron is p_e, the edge weight w_e is defined as

$$w_e = \log \frac{p_e}{1 - p_e} + \log \frac{1 - \beta}{\beta}, \tag{2}$$

where β is a tunable parameter that encourages over or undersegmentation. Since, there are many more lifted edges than adjacent edges, we scale down the lifted weights proportionally to their total number [4].

By reformulating the segmentation problem as a graph partitioning one, we can enforce some global constraints on our result based on the underlying biology. Traditional hierarchical clustering algorithms do not rely on such constraints but consider local decisions independently. We enforce a global constraint that neurons are tree-structured and should not contain cycles. The multicut problems returns a series of "collapsed" edges between nodes that belong to the same neuron. We iterate over these edges in order of the probability of merge generated by our CNN. We "collapse" an edge only if it does not create a cycle in the graph.

3.3. Experiments

We evaluate our method by comparing it to a state-of-the-art pixel-based reconstruction approach using datasets from mouse and fly brains.

3.3.1 Datasets

Our first dataset, which we call the Kasthuri dataset, consists of scanning electron microscope images of the neocortex of a mouse brain [24]. 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 divide the dataset into two

volumes (Vol. 1 and Vol. 2) along the x dimension, where each volume is $8.0 \times 10.9 \times 10.1\,\mu m^3$. We train and validate our methods on Vol. 1 and test on Vol. 2. Our second dataset, called FlyEM, comes from the mushroom body of a 5-day old adult male *Drosophila* fly imaged by a focused ion-beam milling scanning electron microscopy [49]. The original dataset contains a $40 \times 50 \times 120\,\mu m^3$ volume with a resolution of $10 \times 10 \times 10\,nm^3$ per voxel. We use two cubes (Vol. 1 and Vol. 2) of size $10 \times 10 \times 10\,\mu m^3$.

3.3.2 Method Configuration

Input Segmentation The segmentation of the Kasthuri dataset is computed by agglomerating 3D supervoxels produced by the z-watershed algorithm from 3D affinity predictions [60]. We learn 3D affinities using MALIS loss with a U-Net [42, 51]. We apply the z-watershed algorithm with suitable parameters to compute a 3D over-segmentation of the volume. The resulting 3D oversegmentation is then agglomerated using the technique of context-aware delayed agglomeration to generate the final segmentation [37].

For the FlyEM data, based on the authors' suggestion [49], we apply a context-aware delayed agglomeration algorithm [37] 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. The algorithm first computes multi-channel 3D predictions for membranes, cell interiors, and mitochondria, among other cell features. The membrane prediction channel is used to produce an oversegmented volume using 3D watershed, which is then agglomerated hierarchically up to a certain confidence threshold. We used exactly the same parameters as the publicly available code for this algorithm.

Graph Generation The two parameters for the graph pruning algorithm (Sec. 3.2.1) are t_{low} and t_{high}. Ideally, our graph will have an edge for every oversegmented pair of labels with few edges between correctly segmented pairs. After considering various thresholds, we find that $t_{low} = 210\,nm$ and $t_{high} = 300\,nm$ produce expressive graphs with a scalable number of nodes and edges. During implementation, we use nanometers instead of voxels to standardize units across all datasets.

Edge Weight Learning We use half of the Kasthuri dataset for training and validation. We train on 80% of the potential merge candidates for this volume. We validate the CNN classifier on the remaining 20% of the candidates. Since our input does not require the image data, we can train on the anisotropic Kasthuri data and test on the isotropic FlyEM data.

Training Augmentation Since our input is an existing segmentation of the EM images, there are very few training examples compared to per-pixel classifiers that train on a unique window for each voxel. The Kasthuri dataset represents a region of brain over $800\,\mu m^3$ in volume and only yields 640 positive merge examples and 4821 negative ones. To avoid overfitting our deep networks, we apply the following augmentations on the training examples. During a single batch, we randomly select ten positive and ten negative examples. With probability 0.5, an example is reflected across the xy-plane. We then rotate each example by a random angle between 0 and 360 degrees using nearest neighbor interpolation. We have 20,000 such examples per epoch.

3.3.3 Error Metrics

We evaluate the performance of the different methods using the split variation of information (VI) score [33]. Given a ground truth labeling GT and our automatically reconstructed segmentation SG, over and undersegmentation are quantified by the conditional entropies $H(GT|SG)$ and $H(SG|GT)$, respectively. Since we are measuring the entropy between two clusterings, lower VI scores are better. The sum of these conditional entropies gives the total variation of information.

We use precision and recall to evaluate the convolutional neural network and multicut outputs. Since our method only corrects *split errors*, we define a true positive as a pair of segments that are correctly merged together after our pipeline.

3.4. Results

In Fig. 7, we show the VI results of the pixel-based reconstructions of the Kasthuri and FlyEM data (Sec. 3.3.1) for varying thresholds of agglomeration (green). We show comparisons to an oracle (blue) that correctly partitions the graph from our method based on ground truth. Scores closer to the origin are better for this metric, and in every instance our results (red) are below the green curve. We see improvements on each data with a reduction in total VI score of 10.4% on the Kasthuri data and 8.9% and 5.4% on the FlyEM datasets.

Fig. 8 (left) shows successful merges on the Kasthuri dataset. Several of these examples combine multiple consecutive segments that span the volume. In the third example on the left we correct the over-segmentation of a dendrite and attached spine-necks. Fig. 8 (right) shows typical failure cases of our method (red circles). In two of these examples the algorithm correctly predicts several merges before a single error renders the segment as wrong. In the third example (blue circle) a merge error in the initial segmentation propagates to our output. We now analyze how

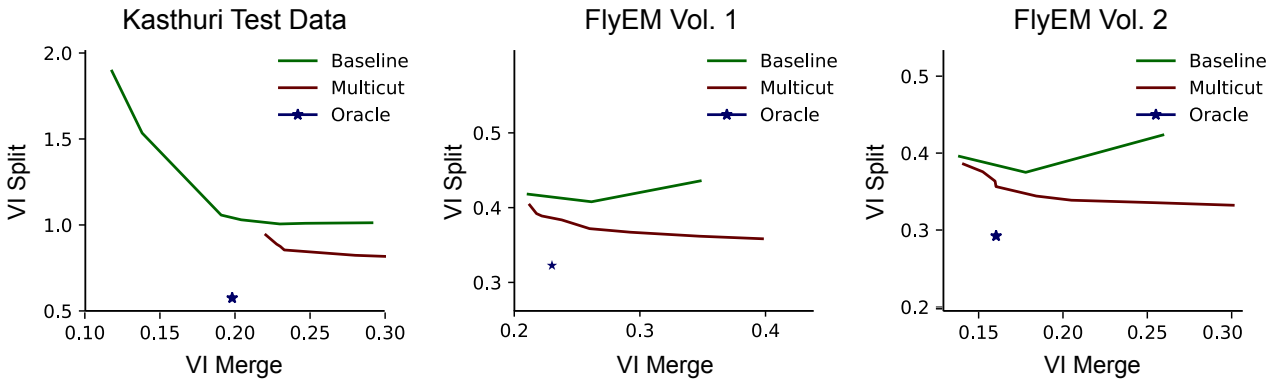

Figure 7: Segmentation benchmark results on three volumes. We compare our method (red) to the baseline segmentation (green) and an oracle (blue) that optimally partitions our constructed graph from our method. Lower VI scores are better. Our method improves the segmentation accuracy over the baseline in all cases. Note that our model is only trained on the Kasthuri training volume and it generalizes well to the FlyEM datasets.

each major component of our method contributes to this final result.

3.4.1 Empirical Ablation Studies

Graph Generation Table 1 shows the results of pruning the skeleton graph using the algorithm discussed in Sec. 3.2.1. This edge pruning is essential for the graph partitioning algorithm, which has a non-linear computational complexity dependence on the number of edges. 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 (e.g., 764 compared to 974). Our pruning heuristic removes at least $3.5\times$ the number of edges on all datasets, achieving a maximum removal rate of $4.15\times$. However, there are some adjacent oversegmented labels which are not considered.

We generate edges in our graph by using information from the skeletons. In particular, we do not enforce the constraint that edges in our graph correspond to adjacent segments. Although neurons are continuous, the EM images often have noisy spots which cause an interruption in the input segmentation. We still want to reconstruct these neurons despite the fact that the initial segmentation is non-continuous. The second and fourth examples in Fig. 8 show correctly reconstructed neurons where two of the segments are non-adjacent.

Failure Cases There are some pairs of segments which we do not consider for merging because of our reliance on the skeletons. Fig. 9, top, shows two such cases with the closest endpoints circled. In the right example the small segment is carved from the larger segment in a location where there are no skeleton endpoints. There are on average 177 such examples in our datasets.

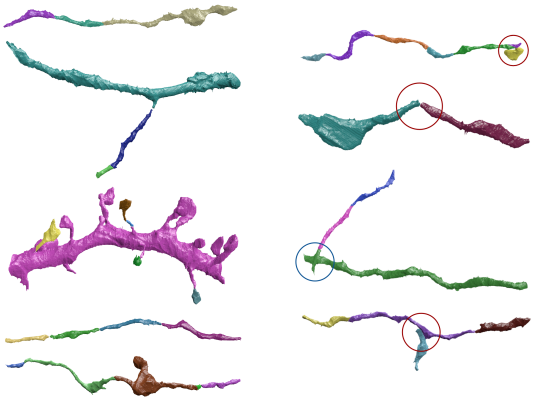

Figure 8: (left) Segments of neurons before they were correctly merged by our method. (right) Circles indicate areas of wrong merges by our method (red) or by the initial pixel-based segmentation (blue).

Edge Weight Learning Fig. 10 shows the receiver operating characteristic (ROC) curve of our CNN classifier for all test datasets. As shown by the ROC curve, the test results on the FlyEM data are better than the results for Kasthuri. In part this comes from the disparity in the number of positive to negative merge candidates in the two graphs. The network easily classifies most of the negative examples leaving only a few difficult examples to predict. Since there are more negative examples in relation to the positive examples in the FlyEM data, the ROC curve is greater.

Graph Partition The graph optimization strategy using

Table 1: The results of our graph pruning approach compared to the baseline graph with all adjacent regions. We show the number of true merge locations (e.g., 974) compared to total number of edges in the graph (e.g., 25,798) for each case. The number of missed splits corresponds to the number of split errors that our method misses compared to an adjacency matrix.

Dataset	Segment Adjacency	Skeleton Pruning	Missed Splits	Gained Edges
Kasthuri	974 / 25,798	764 / 6,218	307	97
FlyEM Vol. 1	304 / 15,949	212 / 4,578	105	13
FlyEM Vol. 2	298 / 17,614	197 / 4,366	120	19

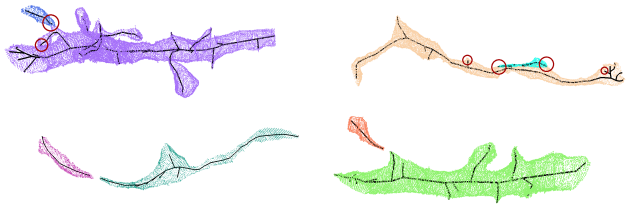

Figure 9: The top two examples correspond to segment pairs that we incorrectly prune from the graph. The distance between the circled endpoints is too great. The bottom two examples show pairs of segments that belong to the same neuron but are not adjacent in the input segmentation. However, we correctly merge these pairs.

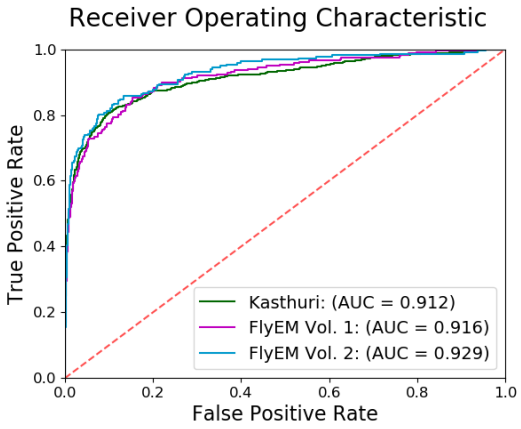

Figure 10: The receiver operating characteristic (ROC) curves of our classifier on three connectomics datasets. The classifier works best on previously unseen data of the Kasthuri volume.

multicut increases accuracy over using just the CNN. Table 2 shows the changes in precision, recall, and accuracy for all three datasets compared to the CNN with both the multicut and lifted multicut formulations. The precision increases on each dataset, although the recall decreases on each datasets. Since it is more difficult to correct merge errors than split errors, it is often desirable to sacrifice recall for precision. Over the three testing datasets, applying a graph-based partitioning strategy increased the precision by 31.9%, 40.9%, and 27.8% respectively.

4. Proposed Work

There is significant room for further research in the compression of label volumes. Compresso currently uses windows of size $8\times8\times1$ based on empirical evidence that these 2D shapes outperform 3D ones. However, there is certainly more information we can exploit across the stack. Window values in slice z are highly correlated with the corresponding windows in the slices above and below. Thus, we can reduce the number of bits stored for these windows. For example, consider a window w in slice z where its neighbors in slices $z-1$ and $z+1$ have window values of 0 (i.e., no boundary pixels in the window). The window in z will have value 0 with very high probability.

There are significantly more biological constraints that we plan to use in the future for region merging and error correction. Once we improve our synaptic classifiers, we can ensure that we do not merge dendritic spines with axons. By stipulating that a segment on one side of the cell body can only have either pre- or post-synaptic connections, we will prevent undersegmentation that merges multiple neurons together. Additionally, once we have a classifier to label each neuron as inhibitory or excitatory, we can prevent the merging of two different types of neurons. Both of these additional constraints will help with the current problem associated with large-scale reconstruction. That is, a small percentage of merge errors results in a tangled mess of multiple neurons per segment.

To further prevent this problem of undersegmentation, we will augment our current work to correct merge errors as well. Generally, correcting merge errors is more difficult than correcting split errors because the space of possible split candidates grows quickly [36]. However, we plan to prune these candidates more efficiently using a skeleton based method that considers the underlying biology. Skeletons corresponding to undersegmented labels will have junctions at the failed merge (Fig. 11). This greatly reduces the search space and will enable correcting split errors in a fraction of the current time.

After locating these potentially erroneous segments, we will run a watershed algorithm on the affinities that forces the voxels into two segments. This watershed algorithm will

Table 2: Precision, recall, and accuracy changes between CNN only and CNN paired with graph-optimized reconstructions for the training and three test datasets. The combined method results in better precision and accuracy. The lifted multicut extension provides very slight improvements in recall and accuracy over these three datasets.

	Multicut			Lifted Multicut		
Dataset	**Δ Precision**	**Δ Recall**	**Δ Accuracy**	**Δ Precision**	**Δ Recall**	**Δ Accuracy**
Kasthuri	31.94%	-36.24%	0.71%	-1.01%	0.60%	0.02%
FlyEM Vol. 1	40.87%	-42.37%	1.26%	0.35%	0.85%	0.04%
FlyEM Vol. 2	27.80%	-44.95%	0.33%	0.54%	0.92%	0.04%

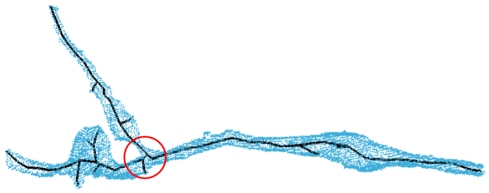

Figure 11: An example where the input segmentation incorrectly merged two neurons together. The red circle indicates the junction from the skeletonized representation.

provide two seeds on opposite sides of the discovered junction. The surface separating the two segments output by the watershed algorithm is our "split" candidate. We will extract a cubic region of interest around this candidate and input the region into our previously discussed merge classifier. If the classifier suggests that the segments belong to one neuron, we will ignore the split candidate. Otherwise we will divide this segment based on the watershed result. With a synaptic classifier, we can further constrain the watershed algorithm to only propose splits where pre- or post-synaptic connections occur on one side. This process will run recursively for a given segment in case more than two neurons are improperly merged.

5. Conclusions

Several challenges arise as the acquisition speeds for EM images improve and enable image stacks petabytes in size. Currently, automatic reconstruction techniques segment these image stacks into label volumes where each individual neuron receives a unique label. However, these label volumes are larger than the input image data because of the number of unique neurons in volumes of this size. Thus, one major challenge is how to efficiently store these large label volumes. We propose Compresso, which exploits properties unique to these datasets, and outperforms existing methods in terms of compression ratio.

Current state-of-the-art reconstruction algorithms that are fast enough to scale often rely on local context for merging and are agnostic to the underlying biology. We introduce a novel region merging algorithm that uses biological constraints at the local and global level to improve oversegmentations of these image stacks. We extract a graph from the oversegmentation and rely on a geometric prior on the shape of neuronal processes to prune the edges in the graph. A CNN learns edge weights based on the local region around two segments. We produce an improved segmentation from this graph by reformulating the partitioning problem as a multicut one and applying global constraints to the solution. We show significant accuracy improvements on datasets from two different species. The main benefits of our approach are that it enforces domain-specific constraints at the global graph level while incorporating pixel-based classification information.

In the future, these methods can be adjusted to apply additional domain constraints. We can augment the graph with more information from the image data, such as synaptic connections, cell morphology, and locations of mitochondria. This would allow us to match other biological constraints during graph partitioning. For example, we could then enforce the constraint that a given segment only has post- or pre-synaptic connections. An augmented graph would be helpful for splitting improperly merged segments by adding additional terms to the partitioning cost function. Using skeletons we can apply biological constraints on the topology for neurons that are improperly merged.

References

[1] L. Aimar, L. Merritt, E. Petit, et al. x264-a free h264/avc encoder, 2005. 2

[2] B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2611–2618. IEEE, 2011. 5

[3] 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. 3

[4] T. Beier, C. Pape, N. Rahaman, T. Prange, S. Berg, D. D. Bock, A. Cardona, G. W. Knott, S. M. Plaza, L. K. Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature methods*, 14(2):101, 2017. 3, 5

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

[6] 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. 3

[7] K. L. Briggman and D. D. Bock. Volume electron microscopy for neuronal circuit reconstruction. *Current opinion in neurobiology*, 22(1):154–161, 2012. 1

[8] 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. 5

[9] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016. 3

[10] 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. 3

[11] Collet and Turner. Smaller and faster data compression with zstandard, url: https://code.facebook.com/posts/1658392934479273/smaller-and-faster-data-compression-with-zstandard/, 2016. Accessed: 23-October-2016. 2

[12] P. Deutsch and J.-L. Gailly. Zlib compressed data format specification version 3.3. Technical report, 1996. 2

[13] J. Funke, F. D. Tschopp, W. Grisaitis, C. Singh, S. Saalfeld, and S. C. Turaga. A deep structured learning approach towards automating connectome reconstruction from 3d electron micrographs. *arXiv preprint arXiv:1709.02974*, 2017. 1, 3

[14] Google. Brotli compression format, url: https://github.com/google/brotli, 2016. Accessed: 11-October-2016. 2

[15] Google. Neuroglancer compression, url: https://github.com/google/neuroglancer/blob/master/src/neuroglancer/sliceview/compressed_segmentation/readme.md, 2016. Accessed: 21-October-2016. 2

[16] 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, 3

[17] 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. 1, 3

[18] 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. 3

[19] L. He, Y. Chao, K. Suzuki, and K. Wu. Fast connected-component labeling. *Pattern Recognition*, 42(9):1977–1987, 2009. 3

[20] 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. 3

[21] V. Jain, S. C. Turaga, K. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung. Learning to agglomerate superpixel hierarchies. In *Advances in Neural Information Processing Systems*, pages 648–656, 2011. 3

[22] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain. Flood-filling networks. *arXiv preprint arXiv:1611.00421*, 2016. 3

[23] T. Jovanic, C. M. Schneider-Mizell, M. Shao, J.-B. Masson, G. Denisov, R. D. Fetter, B. D. Mensh, J. W. Truman, A. Cardona, and M. Zlatic. Competitive disinhibition mediates behavioral choice and sequences in drosophila. *Cell*, 167(3):858–870, 2016. 1

[24] N. Kasthuri, K. J. Hayworth, D. R. Berger, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015. 5

[25] 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. 3

[26] 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. 5

[27] 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. 3

[28] S. Knowles-Barley, M. Roberts, N. Kasthuri, et al. Mojo 2.0: Connectome annotation tool. *Frontiers in Neuroinformatics*, (60), 2013. 3

[29] K. Lee, A. Zlateski, A. Vishwanathan, and H. S. Seung. Recursive training of 2d-3d convolutional networks for neuronal boundary detection. *arXiv preprint arXiv:1508.04843*, 2015. 3

[30] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017. 3

[31] M. Lehmann. Liblzf, url: http://oldhome.schmorp.de/marc/liblzf.html, 2016. accessed: 13-October-2016. 2

[32] B. Matejek, D. Haehn, F. Lekschas, M. Mitzenmacher, and H. Pfister. Compresso: Efficient compression of segmentation data for connectomics. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 781–788. Springer, 2017. 1, 2

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

[34] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, et al. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics*, 8, 2014. 3

[35] M. Oberhumer. Lzo real-time data compression library. *User manual for LZO version 0.28, url: http://www.infosys.tuwien.ac.at/Staff/lux/marco/lzo.html (February 1997)*, 2005. 2

[36] T. Parag. What properties are desirable from an electron microscopy segmentation algorithm. *arXiv preprint arXiv:1503.05430*, 2015. 8

[37] 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. 3, 6

[38] 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, 3

[39] I. Pavlov. Lzma sdk (software development kit), 2007. 2

[40] G. Roelofs and R. Koman. *PNG: the definitive guide*. O'Reilly, Inc., 1999. 2

[41] 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. 3

[42] 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. 3, 6

[43] 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. 4

[44] R. Schalek, D. Lee, N. Kasthuri, A. Suissa-Peleg, T. R. Jones, V. Kaynig, D. Haehn, H. Pfister, D. Cox, and J. W. Lichtman. Imaging a 1 mm3 volume of rat cortex using a multibeam sem. In *Microscopy and Microanalysis*, volume 22, pages 582–583. Cambridge Univ Press, 26 July 2016. 1

[45] J. Seward. bzip2, 1998. 2

[46] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001. 2

[47] O. Sporns, G. Tononi, and R. Kötter. The human connectome: a structural description of the human brain. *PLoS computational biology*, 1(4):e42, 2005. 1

[48] A. Suissa-Peleg, D. Haehn, S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, R. Schalek, J. W. Lichtman, and H. Pfister. Automatic neural reconstruction from petavoxel of electron microscopy data. *Microscopy and Microanalysis*, 22:536, 2016. 1

[49] 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. 6

[50] S.-y. Takemura, C. S. Xu, Z. Lu, P. K. Rivlin, T. Parag, D. J. Olbris, S. Plaza, T. Zhao, W. T. Katz, L. Umayam, et al. Synaptic circuits and their variations within different columns in the visual system of drosophila. *Proceedings of the National Academy of Sciences*, 112(44):13711–13716, 2015. 1

[51] 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. 6

[52] 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. 3

[53] Vandevenne and Alakuijala. Zopfli compression algorithm, url: https://github.com/google/zopfli, 2016. Accessed: 11-October-2016. 2

[54] 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. 3

[55] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984. 2

[56] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philos Trans R Soc Lond B Biol Sci*, 314(1165):1–340, 1986. 1

[57] T. Zeng, B. Wu, and S. Ji. Deepem3d: approaching human-level performance on 3d anisotropic em image segmentation. *Bioinformatics*, 33(16):2555–2562, 2017. 3

[58] T. Zhao and S. M. Plaza. Automatic neuron type identification by neurite localization in the drosophila medulla. *arXiv preprint arXiv:1409.1892*, 2014. 4

[59] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978. 2

[60] 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. 3, 6

[61] J. Zung, I. Tartavull, and H. S. Seung. An error detection and correction framework for connectomics. *CoRR*, abs/1708.02599, 2017. 1, 3