Learning Dense Voxel Embeddings for 3D Neuron Reconstruction

Kisuk Lee[1*], Ran Lu[2], Kyle Luther[2], and H. Sebastian Seung[2]

MIT, Cambridge MA 02139, USA
kisuklee@mit.edu
Princeton University, Princeton NJ 08544, USA
{ranl,kluther,sseung}@princeton.edu

Abstract. We show dense voxel embeddings learned via deep metric learning can be employed to produce a highly accurate segmentation of neurons from 3D electron microscopy images. A *metric graph* on an arbitrary set of short and long-range edges can be constructed from the dense embeddings generated by a convolutional network. Partitioning the metric graph with long-range affinities as repulsive constraints can produce an initial segmentation with high precision, with substantial improvements on very thin objects. The convolutional embedding net is reused without any modification to agglomerate the systematic splits caused by complex "self-touching" objects. Our proposed method achieves state-of-the-art accuracy on the challenging problem of 3D neuron reconstruction from the brain images acquired by serial section electron microscopy. Our alternative, object-centered representation could be more generally useful for other computational tasks in automated neural circuit reconstruction.

Keywords: Connectomics · Neuron reconstruction · Image segmentation · Deep metric learning · Dense embeddings.

1 Introduction

Neuronal connectivity can be reconstructed from a 3D electron microscopy (EM) image of a brain volume. A challenging and important subproblem is the segmentation of the image into neurons. In the SNEMI3D[1] and CREMI[2] segmentation challenges, almost all of the top submissions apply a convolutional network to detect neuronal boundaries [18,7], which are postprocessed to yield a segmentation. While impressive accuracy has been obtained via this approach, one could argue that the task of detecting boundaries does not force the net to learn about objects, and that such knowledge could be critical for further improvements in accuracy.

To go beyond simple boundary detection, one can add the auxiliary task of predicting affinities between nonneighboring voxels [18]. A more radical departure is to learn an object-centered representation by training a 3D convolutional

---

[*] Current affiliation: Princeton University (email: kisuk@princeton.edu)

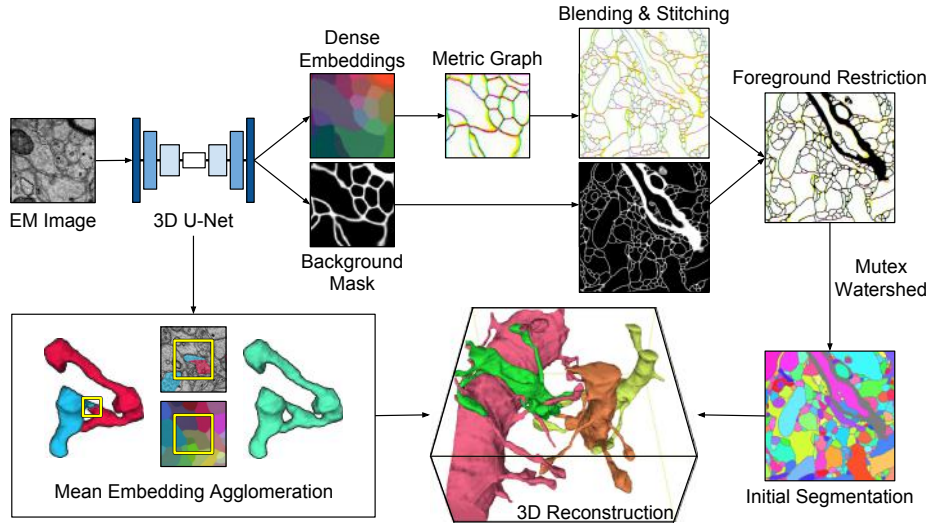[1] http://brainiac2.mit.edu/SNEMI3D/

[2] https://cremi.org/

Fig. 1. Overview of the proposed method. Dense voxel embeddings generated by a convolutional network (Sec. 2.1) are employed at two subtasks for 3D neuron reconstruction: neuron segmentation via *metric graph* [22] (Sec. 2.2) and agglomeration based on mean embeddings (Sec. 2.3). All of the graphics shown here (images and 3D renderings) are drawn from real data. For visualization, we used PCA to project the 24-dimensional embedding space onto the three dimensional RGB color space. For brevity, we only visualize nearest neighbor affinities on the metric graph by mapping x, y, and z-affinity to RGB, respectively.

net to iteratively extend one object at a time, as in flood-filling nets [11,12]. Here we explore an alternative approach to object-centered representation, which is to learn dense voxel embeddings via deep metric learning. Convolutional nets are trained to generate dense embeddings by assigning similar vectors to voxels within the same objects and well-separated vectors to voxels from different objects [9,6,3,26,15].

Luther & Seung [22] already applied this approach to 2D neuron segmentation. We extend their method to 3D neuron reconstruction, which requires fundamental changes in the way we generate and exploit the dense embeddings. While they trained a 2D convolutional net to generate dense embeddings on the entire 2D image slice, the extent of 3D dense embeddings is severely limited by the memory constraint of GPU during both training and inference.

Moreover, a 2D image slice contains neuronal cross sections of relatively simple shape, which are often well-isolated and locally confined within a small spatial extent. In contrast, a 3D image volume contains highly intertwined branches of neurons with complex morphology, which extend from one end of the volume to another most of the time. As a consequence, increasing the input size in 2D would not necessarily increase the size and extent of contained objects, whereas doing

so in 3D would obviously increase both. This poses a fundamental challenge to learning dense embeddings in 3D.

We tackle the challenging problem of 3D neuron reconstruction from dense embeddings by dividing and conquer. We train a 3D convolutional net to generate highly accurate dense embeddings in a local 3D patch of manageable size, which allows to limit the number and extent of contained objects. Stitching patch-wise embeddings together to produce consistent vector fields is hard though, because it requires *translational invariance* of the embeddings.

To circumvent this issue, we employ the "metric graph" approach proposed by Luther & Seung [22]. Within each patch, we can compute affinities between arbitrary voxel pairs as a function of the distance metric between the voxel embeddings. The resulting affinity graph is referred to as *metric graph* [22]. We produce a metric graph on the entire image volume by blending/stitching the patch-wise metric graphs on an overlapping grid spanning the entire volume (Fig. 1).

We demonstrate that our dense voxel embeddings can be successfully employed at two different subtasks for 3D neuron reconstruction: (1) neuron segmentation via metric graph, and (2) agglomeration based on mean embeddings (Fig. 1). As a whole system, our proposed method outperforms the strong baseline, with noticeable differences in both the absolute number and the qualitative types of remaining errors.

2 Methods

2.1 Means-based Loss Function

We use the discriminative loss function proposed by Brabandere *et al.* [3] to learn dense voxel embeddings. This loss function has recently been adopted by Luther & Seung [22] in the application of 2D neuron segmentation, and has also been applied to an increasingly wide variety of other problem domains including object discovery in videos [34], 3D point cloud embeddings [27,5,19], scene text detection [30], biomedical segmentation [8], driver information systems [21], 3D volumetric instance segmentation [16,17], single-image piece-wise planar 3D Reconstruction [35].

Luther & Seung [22] refer to the loss as "means-based loss," because the mean embeddings of distinct objects play a key role as cluster centers in the three main terms of the loss function: (1) the "internal" term $\mathcal{L}_{\text{int}}$ pulls embeddings towards the respective cluster centers, (2) the "external" term $\mathcal{L}_{\text{ext}}$ pushes distinct cluster centers apart from each other, and (3) the regularization term $\mathcal{L}_{\text{reg}}$ prevents all cluster centers from deviating too far from the origin. These terms can be written

down as follows:

$$\mathcal{L}_{\text{int}} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} \|\mu_c - x_i\|^2 \tag{1}$$

$$\mathcal{L}_{\text{ext}} = \frac{1}{C(C-1)} \sum_{\substack{c_A=1 \\ c_A \neq c_B}}^{C} \sum_{c_B=1}^{C} \max(2\delta_d - \|\mu_{c_A} - \mu_{c_B}\|, 0)^2 \tag{2}$$

$$\mathcal{L}_{\text{reg}} = \frac{1}{C} \sum_{c=1}^{C} \|\mu_c\| \tag{3}$$

where C is the number of ground truth objects, N_c is the number of voxels in object c, μ_c is the mean embedding for object c, x_i is an embedding for voxel i, $\|\cdot\|$ is the L1 norm, δ_d is the margin for the external loss term $\mathcal{L}_{\text{ext}}$. We choose $\delta_d = 1.5$ following [3,22]. Note that only those voxels that belong to the ground truth objects (i.e. foreground objects) are taken into account in Eq. 1–3. The embeddings for "background" voxels are not explicitly affected by these terms.

The final embedding loss is a weighted sum of the three terms:

$$\mathcal{L}_{\text{embedding}} = \alpha \mathcal{L}_{\text{int}} + \beta \mathcal{L}_{\text{ext}} + \gamma \mathcal{L}_{\text{reg}} \tag{4}$$

where we choose $\alpha = \beta = 1$ and $\gamma = 0.001$, same as in [22].

Besides the embedding loss, we additionally predict a voxel-wise background mask (Fig. 1 and Sec. 2.2) using the standard binary cross-entropy loss. The total loss is an equally-weighted sum of the embedding and background losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{embedding}} + \mathcal{L}_{\text{background}}. \tag{5}$$

Sometimes parts of the same object may seem to be distinct objects when they are restricted to a local patch of limited context (Fig. 2).[3] It is noteworthy that recomputing connected components of the ground truth objects in each training example, and then removing from $\mathcal{L}_{\text{ext}}$ the individual external terms between the locally split object parts, was found to be critical in achieving better generalization performance, confirming the finding of [22]. We additionally found that failing to properly separate such object parts in the training patch can hurt generalization performance, especially when there exists a systematic such failure mode (see "glial confusion" in Fig. S22).

2.2 Segmentation with Metric Graph

We define an affinity $a_{ij} \in [0, 1]$ between a pair of voxels i and j by

$$a_{ij} = \max\left(\frac{2\delta_d - \|x_i - x_j\|}{2\delta_d}, 0\right)^2 \tag{6}$$

---

[3] See Maitin-Shepard *et al.* (2016) [23] for the related discussion on local and global connectivity.

where x_i and x_j are embeddings for the voxel i and j, respectively. Note that this definition is directly derived from $\mathcal{L}_{\text{ext}}$ (Eq. 2). Given dense voxel embeddings within each local patch, we can construct a patch-wise *metric graph* whose nodes are voxels and edges are metric-derived affinities between voxel pairs [22].

Luther & Seung [22] explored only the simplest form of postprocessing, i.e., nearest neighbor metric graph partitioned by connected components clustering. However, we found in our 3D experiments that nearest neighbor affinities are occasionally noisy, especially around the background voxels that were excluded from $\mathcal{L}_{\text{embedding}}$ (Fig. S5).

To address this issue, and to push the accuracy further, we augmented our postprocessing by (1) removing noisy edges from the metric graph using a voxel-wise background mask predicted by the embedding net (Fig. 1), and (2) incorporating long-range affinities as repulsive constraints during clustering. For masking out noisy edges, the predicted background mask was binarized with a threshold θ_{mask} that had been chosen empirically on the validation set, and then every edge straddling background voxels were removed from the metric graph.

The resulting "foreground-restricted" metric graph (Fig. 1) was used as input to the *Mutex Watershed* [33], a recently proposed algorithm for partitioning a graph with both attractive/repulsive edges.[4] They trained a 2D convolutional net to directly predict affinities on a small predefined set of short and long-range edges, thus only a *fixed* graph can be generated. We have no such limitation – affinities on *any* edge can be computed from the dense embeddings.

2.3   Mean Embedding Agglomeration

We observe that both the Mutex Watershed [33] for the proposed method and mean affinity agglomeration [18] for the baseline (Sec. 2.4) make suboptimal decisions on "self-touching" objects, resulting in systematic split errors (Fig. 2). These greedy clustering/agglomeration algorithms are very effective at suppressing localized mistakes, whereas they often fail to reconcile the local evidence for disconnectivity at self-contacts with the evidence for connectivity elsewhere, if the former precedes the latter in the greedy decision making procedure.

To address this systematic failure mode, we detect the candidates for self-touching split errors based on a simple and intuitive heuristic. Specifically, we construct a region adjacency graph (RAG) from the initial segmentation. Foreground-restricted nearest neighbor metric graph is provided together to compute an agglomeration score S of each individual contact between adjacent segments. We select a pair of adjacent segments as candidate if (1) they share multiple contacts, and (2) the highest S among the contacts is above a certain threshold $\theta_{\text{self-contact}}$. Each candidate is a triplet $(s_1, s_2, (x, y, z))$, where s_1 and s_2 are the pair of candidate segments and (x, y, z) is the centroid coordinate of the contact with the highest S. We use mean affinity as agglomeration score, and choose $\theta_{\text{self-contact}}$ empirically on the training and validation sets.

---

[4] The Mutex Watershed [33] has more recently been integrated into GASP, a generalized framework for agglomerative clustering of signed graphs [2].

Given candidates detected by the heuristic, we reuse the embedding net without any modification to make an agglomeration decision (Fig. 2). Specifically, we generate dense voxel embeddings dynamically on each patch centered on (x, y, z), and then compute mean embeddings μ_{s_1} and μ_{s_2} of the restrictions of s_1 and s_2 to a central "focal" widow of size $p_x \times p_y \times p_z$ (yellow box in Fig. 2). The two candidate segments are agglomerated if the L1 distance $d = \|\mu_{s_1} - \mu_{s_2}\|$ is below a predetermined threshold θ_d. The parameters p_x, p_y, p_z and θ_d are determined empirically on the training and validation sets. Although determined empirically, the optimal θ_d turns out to be equivalent to the δ_d of $\mathcal{L}_{\text{ext}}$ (Eq. 2), being theoretically reasonable (Fig. S12–S14).

2.4   Establishing a Strong Baseline

As a strong baseline, we reproduced the state-of-the-art result of [18], which is currently leading the SNEMI3D benchmark challenge.[5] They trained a 3D variant of U-Net [29] to directly predict nearest neighbor affinities as a primary target [31], and long-range affinities as an auxiliary target. The predicted nearest neighbor affinities are then partitioned with a variant of watershed algorithm [36] to produce an initial oversegmentation. Although the costly $16\times$ test-time augmentation produced the best result, mean affinity agglomeration without test-time augmentation was shown to be competitive while being efficient [18]. Therefore, we compare both postprocessing methods in our baseline results.

Additionally, we performed a preliminary experiment with baseline models to test the following hypothesis: can we obtain segmentation with comparable
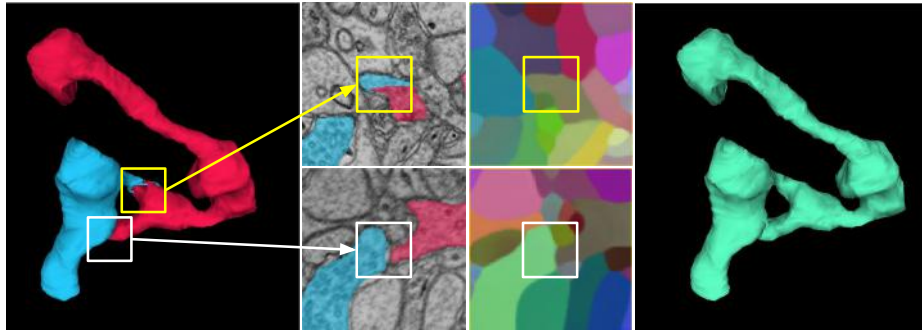
---

[5] http://brainiac2.mit.edu/SNEMI3D/leadersboard

Fig. 2. Mean embedding agglomeration (Sec. 2.3). A real example of self-touching axon from the validation set is shown here. Left: a false split (yellow box) in the Mutex Watershed [33] segmentation caused by self-touching (white box). Middle top: PCA visualization (right) reveals uniform embeddings across the false split. Middle bottom: distinct vectors were assigned to the parts of the same axon that make self-contact. Right: Mean embedding agglomeration correctly merges the two parts.

accuracy from images with lower resolution? This is an important question because connectomics is facing an imminent challenge of scaling up to petascale data [20]. We performed dual experiments with the original images ($6 \times 6 \times 29$ nm^3) and $2\times$ in-plane-downsampled images ($12 \times 12 \times 29$ nm^3), and found no significant loss in accuracy (Table 1 in Sec. 3). Therefore, we performed our main experiment on dense voxel embeddings (Sec. 2.1–2.3) with the $2\times$ downsampled images exclusively.

3   Experimental Setup

Dataset. We used the publicly available AC3/AC4 dataset,[6] which are small labeled subvolumes ($1024 \times 1024 \times 256$ and $1024 \times 1024 \times 100$ voxels at $6 \times 6 \times 29$ nm^3 resolution, respectively) from the mouse somatosensory cortex dataset of Kasthuri *et al.* [13]. Extra image padding was obtained from the full dataset,[7] and used only for inference to provide enough image context for preventing systematic drop in accuracy near the dataset edge. We used AC4 and the bottom 116 slices of AC3 for training, the middle 40 slices of AC3 for validation, and the top 100 slices of AC3 for testing. More details including data preprocessing can be found in Supplementary Material (Fig. S1–S4).

Network architecture. We adopted the "Residual Symmetric U-Net" architecture of [18], with several minor modifications. Specifically, we used (1) rectified linear units (ReLUs) instead of exponential linear units (ELUs, [4]), (2) all $3 \times 3 \times 3$ convolution filters instead of selectively using 2D filters at the finest scale, (3) Instance Normalization [32] instead of Batch Normalization [10], (4) added one more convolution layer to each "residual module" [18], and (5) used different numbers of feature maps at each scale. Architectural details are fully described in Supplementary Material (Fig. S6–S7).

Data augmentation. We used the same training data augmentation as in [18], with the slip-type misalignment simulation replaced by a novel "slip interpolation." Specifically, we simulated the slip-type misalignment only in the input, not in the target. The mismatch between the input and target forces the nets to ignore any slip misalignment in the input and produce smoothly interpolated prediction. The effect of slip interpolation is illustrated in Fig. S9.

Training details. We performed all experiments with PyTorch.[8] We trained our nets on four NVIDIA Titan X Pascal GPUs using synchronous gradient update. We used the AMSGrad variant [28] of the Adam optimizer [14], with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We used a single training patch

---

[6] `https://software.rc.fas.harvard.edu/lichtman/vast/AC3AC4Package.zip`
[7] `https://neurodata.io/data/kasthuri15/`
[8] `https://pytorch.org/`

(minibatch of size 1) for each model replica on GPUs at each gradient step. We trained the baseline and embedding nets for three and five days, respectively, and selected the checkpoints at the lowest validation error.

Inference. We used the overlap-blending inference of [18]. Both the baseline and embedding nets used the most conservative overlap of 50% in x, y, z-dimension. The embedding net's output cropping incurred nearly 2× overhead relative to the baseline due to the higher coverage factor for each output voxel. We used the same "bump" function as in [18] for weighting the voxels near the patch center more strongly than those near the borders.

Postprocessing. For the Mutex Watershed [33], we used three nearest neighbor attractive edges and nine longer-range repulsive edges. The exact edge neighborhood structure is shown in Fig. S11. We selected the hyperparameters for the foreground restriction (Sec. 2.2) and mean embedding agglomeration (Sec. 2.3) strictly on the training and validation sets. Selected hyperparameters were $\theta_{\mathrm{mask}} = 0.6$, $\theta_{\mathrm{self\text{-}touch}} = 0.25$, $\theta_d = 1.5$, and $p_x \times p_y \times p_z = 32 \times 32 \times 5$.

Evaluation. For quantitative evaluation, we adopted the *Variation of Information* (VI) metric [25,1], which is defined by

$$VI_{\mathrm{sum}} = VI_{\mathrm{split}} + VI_{\mathrm{merge}} = H(S|T) + H(T|S) \tag{7}$$

where $H(\cdot|\cdot)$ is the conditional entropy, S is the segmentation proposal, and T is the ground truth segmentation.

4   Results

Table. 1 shows the evaluation result of different methods on the test set. Metric graph followed by the Mutex Watershed produced an initial segmentation with high precision (indicated by low VI_{merge}), though suffered from the systematic self-touching split errors. Mean embedding agglomeration successfully healed most of these errors (see supplementary material for more details), resulting in a precipitous drop in VI_{split} and outperforming all baselines. The slight increase in VI_{merge} was caused by a correct agglomeration of self-touching glial fragments which already contained a small merge error.

To further test the efficacy of our dense voxel embeddings, we additionally applied mean embedding agglomeration to the baseline results that were post-processed with mean affinity agglomeration (MAA), which also suffered from the self-touching split errors. Remarkably, mean embedding agglomeration significantly reduced VI_{split} without any increase in VI_{merge}, outperforming the costly 16× test-time augmentation counterparts (TTA).

Next, we questioned whether the difference between our proposed method (MWS + MEA) and the strongest baseline (Baseline†‡) could still hold outside the dataset used for our experiments. To this end, three extra image volumes

of same size as the test set, named as E1-E3, were obtained from randomly chosen locations in the full Kasthuri *et al.* dataset [13] (see supplementary material for details). We applied the two competing methods on these volumes, and four expert brain image analysts exhaustively examined every single segment in automated segmentations to find out remaining errors.

Fig. 3 visualizes the spatial distribution of individual merge/split errors from each method in E1–E3. There were 17.0 ± 1.00 and 75.0 ± 7.23 remaining errors (mean $\pm$ SE, $N = 3$) from the proposed and baseline method, respectively. Qualitatively, our proposed method performs substantially better on very thin objects. Visualization of embeddings shows how the object-centered representation could outperform the boundary-centered representation on such thin objects (supplementary material). We have also curated the remaining failures of our proposed method in supplementary material to motivate future studies.

Table 1. Test set evaluation result. Abbreviations: watershed (WS, [36]), mean affinity agglomeration (MAA, [18]), test-time augmentation (TTA), Mutex Watershed (MWS, [33]), mean embedding agglomeration (MEA, Sec. 2.3). † Optimized postprocessing hyperparameters directly on the test set. ‡ We also include for transparency the strongest baseline that arose unexpectedly from a preliminary experiment without the "slip interpolation" data augmentation (Sec. 3).

Method	Postprocessing	Resolution	VI split	VI merge	VI sum
Baseline†	WS	6 nm	0.0545	0.0200	0.0745
Baseline†	WS + TTA (16×)	6 nm	0.0530	0.0218	0.0748
Baseline	WS + MAA	6 nm	0.0888	0.0187	0.1075
Baseline	WS + MAA + **MEA**	6 nm	0.0463	0.0187	0.0650
Baseline†	WS	12 nm	0.0574	0.0224	0.0798
Baseline†	WS + TTA (16×)	12 nm	0.0547	0.0150	0.0697
Baseline	WS + MAA	12 nm	0.0877	0.0172	0.1049
Baseline	WS + MAA + **MEA**	12 nm	0.0399	0.0173	0.0572
Baseline†‡	WS + TTA (16×)	12 nm	0.0475	**0.0135**	0.0610
Proposed	MWS	12 nm	0.1025	0.0163	0.1188
Proposed	MWS + **MEA**	12 nm	**0.0276**	0.0194	**0.0470**

5   Conclusion

We have presented a novel application of dense voxel embeddings for 3D neuron reconstruction. Future work includes scaling-up the proposed method by block merging [24,12] and large-scale evaluation on real world datasets. Our dense voxel embeddings could be more generally useful for other computational tasks in the connectomics pipeline. For instance, the error detection and correction system of [37] could be further enhanced with dense voxel embeddings.
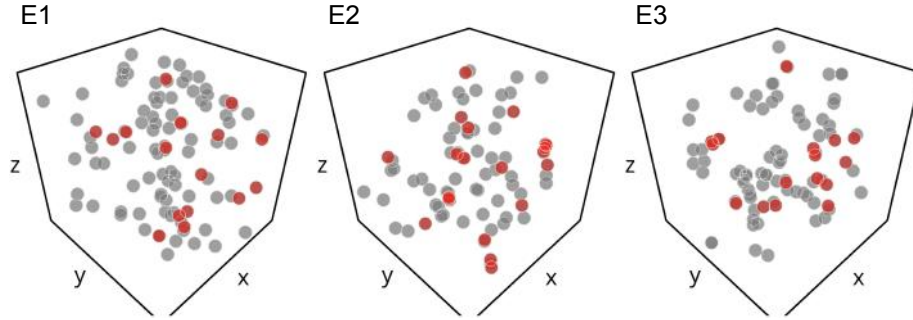
Fig. 3. Spatial distribution of the remaining errors in the three extra test volumes E1–E3. Gray circles: errors in the strongest baseline (Baseline†‡, Table. 1). Red circles: errors in the proposed method (MWS + MEA, Table. 1).

Author Contributions

K.Lee designed and performed all experiments and analyses. R.L. devised and implemented the region graph-based heuristic for self-touching candidate detection. K.Luther implemented the embedding loss function. K.Lee and H.S.S. wrote the paper.

Acknowledgement

References

1. Arganda-Carreras, I., et al.: Crowdsourcing the creation of image segmentation algorithms for connectomics. Frontiers in Neuroanatomy **9**, 142 (2015)
2. Bailoni, A., Pape, C., Wolf, S., Beier, T., Kreshuk, A., Hamprecht, F.A.: A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation. CoRR **abs/1906.11713** (2019), `http://arxiv.org/abs/1906.11713`
3. Brabandere, B.D., Neven, D., Gool, L.V.: Semantic instance segmentation with a discriminative loss function. arXiv preprint arXiv:1708.02551 (2017)
4. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). CoRR **abs/1511.07289** (2015)
5. Elich, C., Engelmann, F., Schult, J., Kontogianni, T., Leibe, B.: 3d-bevis: Birds-eye-view instance segmentation. CoRR **abs/1904.02199** (2019), `http://arxiv.org/abs/1904.02199`
6. Fathi, A., et al.: Semantic instance segmentation via deep metric learning. arXiv preprint arXiv:1703.10277 (2017)
7. Funke, J., Tschopp, F.D., Grisaitis, W., Sheridan, A., Singh, C., Saalfeld, S., Turaga, S.C.: Large scale image segmentation with structured loss based deep learning for connectome reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2018)
8. Halupka, K., Garnavi, R., Moore, S.: Deep semantic instance segmentation of tree-like structures using synthetic data. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1713–1722 (2019)
9. Harley, A.W., Derpanis, K.G., Kokkinos, I.: Learning dense convolutional embeddings for semantic segmentation. arXiv preprint arXiv:1511.04377 (2015)
10. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. pp. 448–456. ICML'15, JMLR.org (2015), `http://dl.acm.org/citation.cfm?id=3045118.3045167`
11. Januszewski, M., Maitin-Shepard, J., Li, P., Kornfeld, J., Denk, W., Jain, V.: Flood-filling networks. arXiv preprint arXiv:1611.00421 (2016)
12. Januszewski, M., et al.: High-precision automated reconstruction of neurons with flood-filling networks. Nature Methods **15**(8), 605–610 (2018)
13. Kasthuri, N., et al.: Saturated reconstruction of a volume of neocortex. Cell **162**(3), 648–661 (2015)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
15. Kong, S., Fowlkes, C.: Recurrent pixel embedding for instance grouping. In: 2018 Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
16. Konopczyński, T.K., Kröger, T., Zheng, L., Hesser, J.: Instance segmentation of fibers from low resolution CT scans via 3d deep embedding learning. In: British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018. p. 268 (2018)
17. Lahoud, J., Ghanem, B., Pollefeys, M., Oswald, M.R.: 3d instance segmentation via multi-task metric learning. CoRR **abs/1906.08650** (2019), `http://arxiv.org/abs/1906.08650`
18. Lee, K., Zung, J., Li, P., Jain, V., Seung, H.S.: Superhuman accuracy on the SNEMI3D connectomics challenge. arXiv preprint arXiv:1706.00120 (2017)

19. Liang, Z., Yang, M., Wang, C.: 3d graph embedding learning with a structure-aware loss function for point cloud semantic instance segmentation. CoRR `abs/1902.05247` (2019), `http://arxiv.org/abs/1902.05247`
20. Lichtman, J.W., Pfister, H., Shavit, N.: The big data challenges of connectomics. Nature Neuroscience **17**(11), 1448–54 (2014)
21. Liu, P., Yang, M., Wang, C., Wang, B.: Multi-lane detection via multi-task network in various road scenes. In: 2018 Chinese Automation Congress (CAC). pp. 2750–2755 (Nov 2018). https://doi.org/10.1109/CAC.2018.8623762
22. Luther, K., Seung, H.S.: Learning metric graphs for neuron segmentation in electron microscopy images. arXiv preprint arXiv:1902.00100 (2019)
23. Maitin-Shepard, J.B., Jain, V., Januszewski, M., Li, P., Abbeel, P.: Combinatorial energy learning for image segmentation. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 1966–1974. Curran Associates, Inc. (2016), `http://papers.nips.cc/paper/6595-combinatorial-energy-learning-for-image-segmentation.pdf`
24. Matveev, A., et al.: A multicore path to connectomics-on-demand. SIGPLAN Not. **52**(8), 267–281 (2017)
25. Nunez-Iglesias, J., Kennedy, R., Parag, T., Shi, J., Chklovskii, D.B.: Machine learning of hierarchical clustering to segment 2d and 3d images. PLoS ONE **8**(8), 1–11 (2013)
26. Payer, C., Štern, D., Neff, T., Bischof, H., Urschler, M.: Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks. In: Medical Image Computing and Computer Assisted Intervention (MICCAI) (2018)
27. Pham, Q.H., Thanh Nguyen, D., Hua, B.S., Roig, G., Yeung, S.K.: JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields. arXiv e-prints (2019)
28. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: International Conference on Learning Representations (2018)
29. Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI) (2015)
30. Tian, Z., Shu, M., Lyu, P., Li, R., Zhou, C., Shen, X., Jia, J.: Learning shape-aware embedding for scene text detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
31. Turaga, S.C., et al.: Convolutional networks can learn to generate affinity graphs for image segmentation. Neural Computation **22**(2), 511–538 (2010)
32. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. CoRR **abs/1607.08022** (2016)
33. Wolf, S., et al.: The mutex watershed: Efficient, parameter-free image partitioning. In: The European Conference on Computer Vision (ECCV) (2018)
34. Xie, C., Xiang, Y., Harchaoui, Z., Fox, D.: Object discovery in videos as foreground motion clustering. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
35. Yu, Z., Zheng, J., Lian, D., Zhou, Z., Gao, S.: Single-image piece-wise planar 3d reconstruction via associative embedding. CoRR **abs/1902.09777** (2019)
36. Zlateski, A., Seung, H.S.: Image segmentation by size-dependent single linkage clustering of a watershed basin graph. arXiv preprint arXiv:1505.00249 (2015)
37. Zung, J., Tartavull, I., Lee, K., Seung, H.S.: An error detection and correction framework for connectomics. In: Advances in Neural Information Processing Systems (NIPS) (2017)
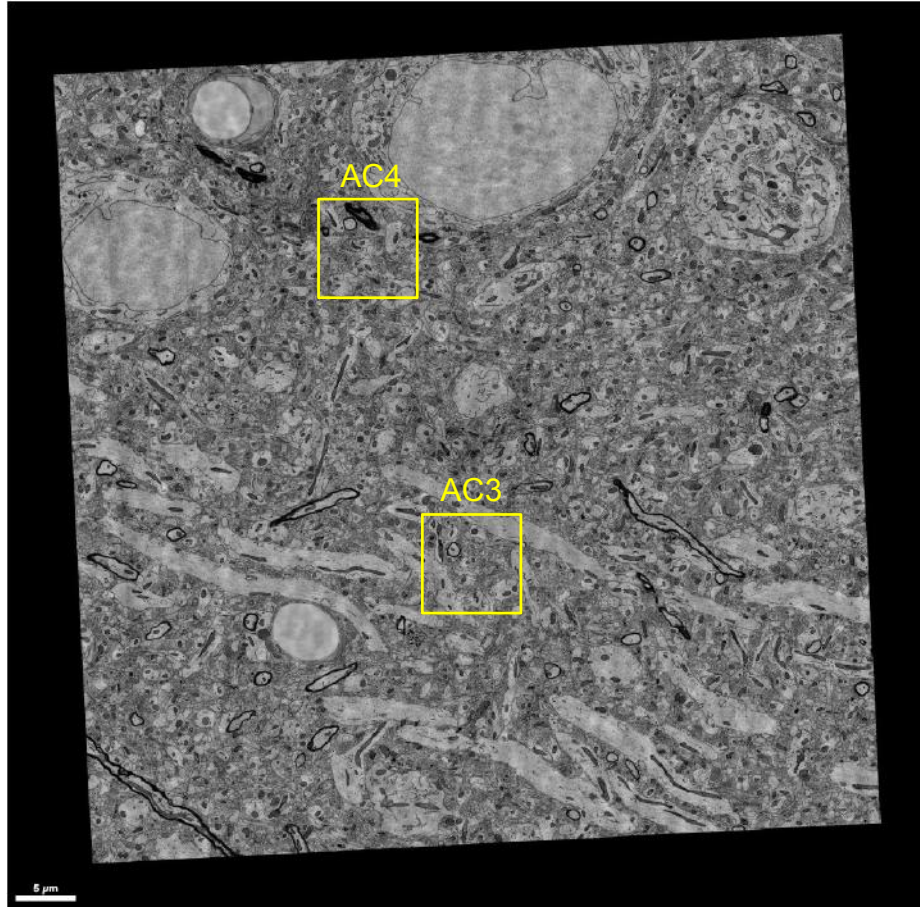
Fig. S1. Location of AC3/AC4 in the full Kasthuri *et al.* dataset. Shown is a section at $z = 1099$. The full dataset is accessible at `https://neurodata.io/data/kasthuri15/`. AC3's bounding box in the full dataset is (10944,17424,999)–(12992,19472,1255). AC4's bounding box is (8800,10880,1099)–(10848,12928,1199). Note that the original data was acquired at $3 \times 3 \times 29$ nm^3 resolution, making the bounding boxes $2\times$ larger in x and y dimension than the size of AC3 and AC4 used for experiments.
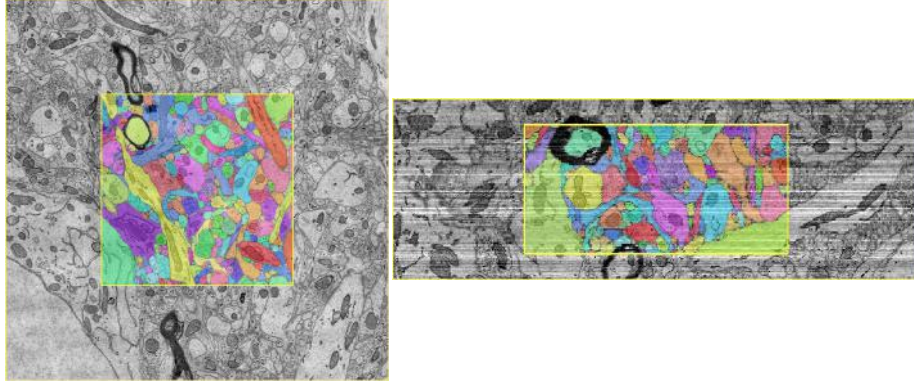
Fig. S2. Visualization of AC4 embedded in surrounding image context. Left: xy view. Right: xz reslice view. We used the entire AC4 for training. In the original AC4 annotation, myelin sheath is not labeled separately. We additionally annotated the myelin sheaths as separate objects.
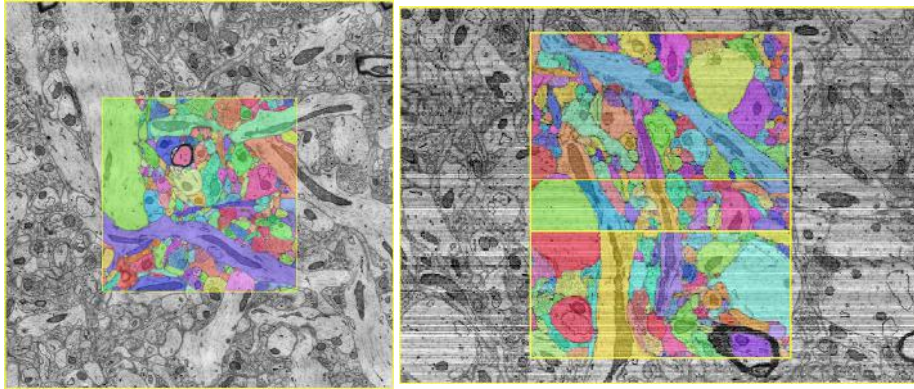
Fig. S3. Visualization of AC3 embedded in surrounding image context. Left: xy view. Right: xz reslice view. We used the top 116 slices for training, the middle 40 slices for validation, and the bottom 100 slices for testing. Note that the volume shown here is a flipped version of the downloadable AC3 volume, so the top/middle/bottom relationship of the training/validation/test split here is different from the description in the main text. In the original AC3 annotation, myelin sheath is not labeled separately. We additionally annotated the myelin sheaths as separate objects, and treated them differently in $\mathcal{L}_{\text{embedding}}$ and $\mathcal{L}_{\text{background}}$. Specifically, we treated myelin sheaths as background in $\mathcal{L}_{\text{background}}$, whereas they are treated as foreground objects in $\mathcal{L}_{\text{embedding}}$.
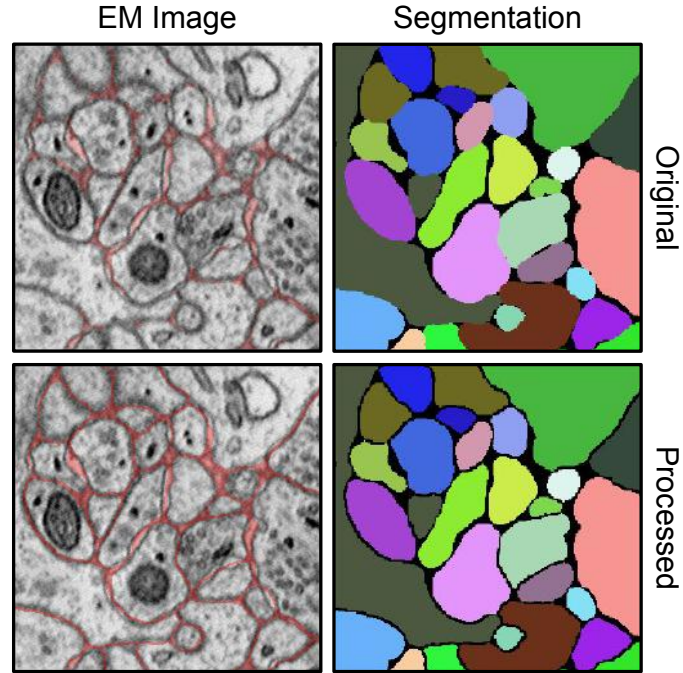
Fig. S4. Data preprocessing. For input image, we linearly rescale the original integer-valued intensity in $[0, 255]$ to the real-valued intensity in $[0, 1]$. For ground truth segmentation, we apply a $3 \times 3 \times 1$ background-augmenting kernel such that any voxel whose $3 \times 3 \times 1$ neighbors (including the voxel itself) contain more than one *positive* segment ID (*zero* is reserved for background) is additionally marked as background.
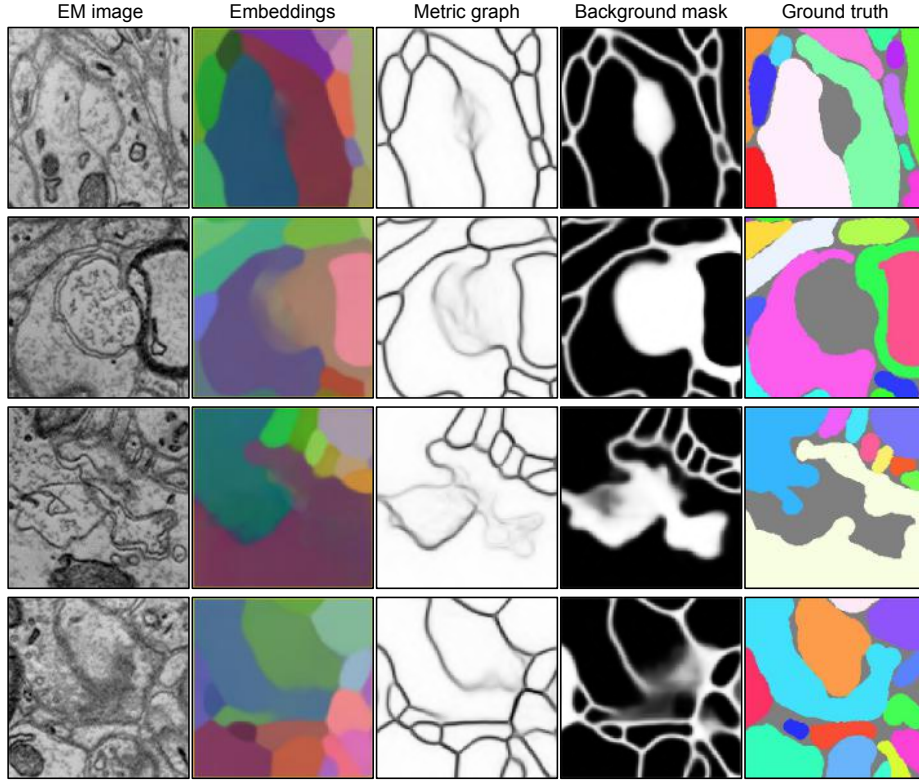
EM image	Embeddings	Metric graph	Background mask	Ground truth

Fig. S5. Noisy embeddings resulted from the exclusion of background voxels in $\mathcal{L}_{\text{embedding}}$. Top row: extracellular space between adjacent objects is enlarged, forming an object-like structure that is labeled as background. Second/third rows: endoplasmic reticulum from each side of adjacent objects form a complex structure that is labeled as background. Note that we treated the myelin sheath as background in $\mathcal{L}_{\text{background}}$ (second row, fourth column), whereas it is treated as a foreground object in $\mathcal{L}_{\text{embedding}}$ (second row, second column). Bottom row: diffuse boundaries at the synaptic interface that is parallel to the sectioning/imaging plane. Here the predicted background mask is also noisy (bottom row, fourth column), thus requiring long-range affinities to be included as repulsive constraints during clustering. For metric graph (third column), we show $\min(a_x, a_y)$ where a_x/a_y are x/y nearest neighbor metric-derived affinities. The embeddings (second column) and background mask (fourth column) are outputs of the embedding net. To visualize embeddings, we used PCA to project the 24-dimensional embedding space onto the three dimensional RGB color space.
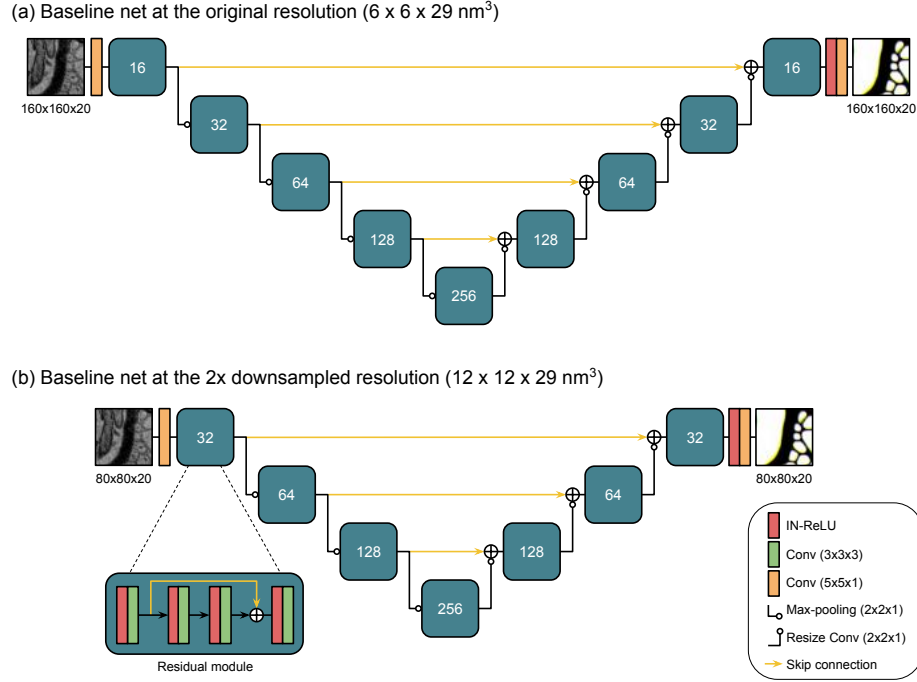
Fig. S6. The baseline net architecture. (a) We used a training patch of $160 \times 160 \times 20$ and an inference patch of $320 \times 320 \times 20$ voxels at the original resolution ($6 \times 6 \times 29$ nm^3). (b) For $2\times$ in-plane-downsampled images ($12 \times 12 \times 29$ nm^3), we halved the patch size in x and y dimension accordingly. Using larger inference patch was found to be effective at suppressing prediction noises in the large intracellular regions, although there was no significant difference in segmentation quality after postprocessing. Besides predicting nearest neighbor affinities as a primary target, we used long-range affinities as an auxiliary target during training. For long-range affinities, we used (x, y, z) offsets of $(-4, 0, 0)$, $(-8, 0, 0)$, $(-24, 0, 0)$, $(0, -4, 0)$, $(0, -8, 0)$, $(0, -24, 0)$, $(0, 0, -2)$, $(0, 0, -3)$, $(0, 0, -4)$ at the original resolution, and halved the x and y offsets for $2\times$ downsampled images. As a result, the baseline nets produce 12 output channels, three for nearest neighbor and nine for long-range affinities. The number inside the residual module represents the width (number of feature maps) of the module. For upsampling, we used the bilinear *resize convolution* (bilinear upsampling followed by $1 \times 1 \times 1$ convolution). Abbreviations: Instance Normalization (IN), rectified linear unit (ReLU), convolution (Conv).
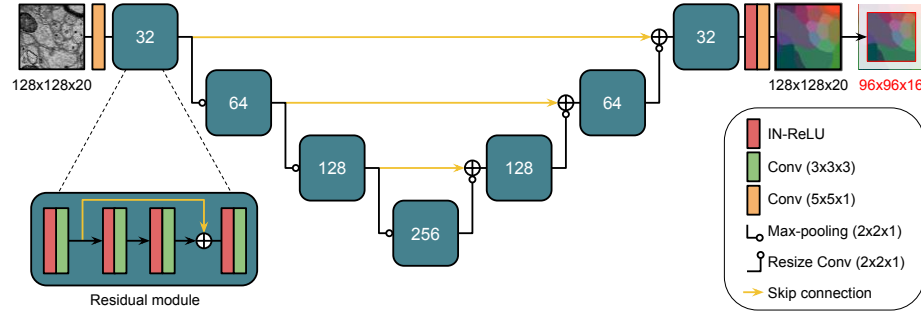
Fig. S7. The embedding net architecture. It was experimented exclusively with $2\times$ in-plane-downsampled images ($12\times12\times29\,\text{nm}^3$). We used a $128\times128\times20$ voxel input patch for both training and inference. Contrary to the baseline experiments, we found that using larger inference patch does not help but rather hurt generalization performance of the embedding net. It produces as output the same-sized patch, which is then cropped to $96\times96\times16$ in order to reduce uncertainty near the patch border. The dimensionality of embedding space is 24. We scale the embeddings linearly with a learnable scalar parameter, which was initialized with 0.1 in the beginning of training. The number inside the residual module represents the width (number of feature maps) of the module. For upsampling, we used the bilinear *resize convolution* (bilinear upsampling followed by $1\times1\times1$ convolution). Abbreviations: Instance Normalization (IN), rectified linear unit (ReLU), convolution (Conv).
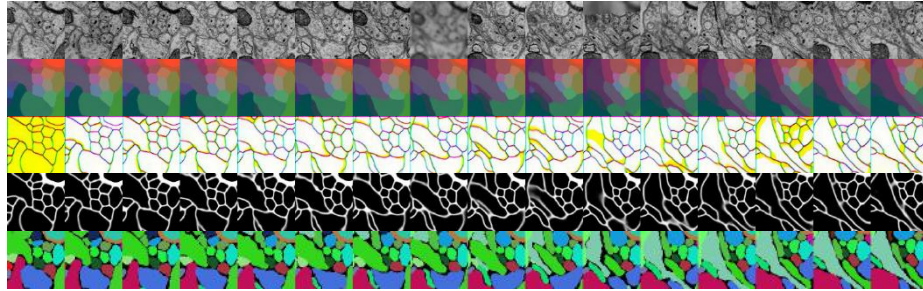
Fig. S8. Visualization of an example training patch. Each row is a flattened version of $96\times96\times16$ patch. From top to bottom, each row shows (1) input EM images, (2) embeddings, (3) nearest neighbor metric graph derived from the embeddings, (4) predicted background mask, and (5) ground truth segmentation. In this particular training example, two simulated out-of-focus sections (full: 8th column, partial: 11th column) and a simulated translation-type misalignment (third to last column) were injected.
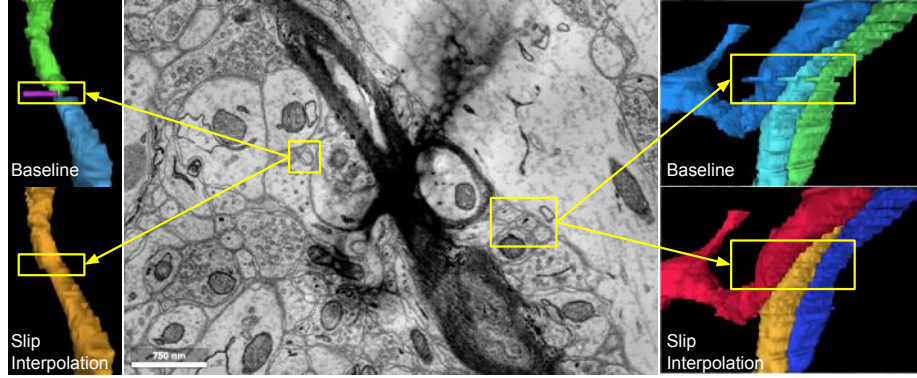
Fig. S9. Effect of slip interpolation. A mild fold (dark shade over the myelinated axon) in the test set causes a slip-type misalignment. Left: a very thin axon is broken at the slip misalignment (top), while the baseline net trained 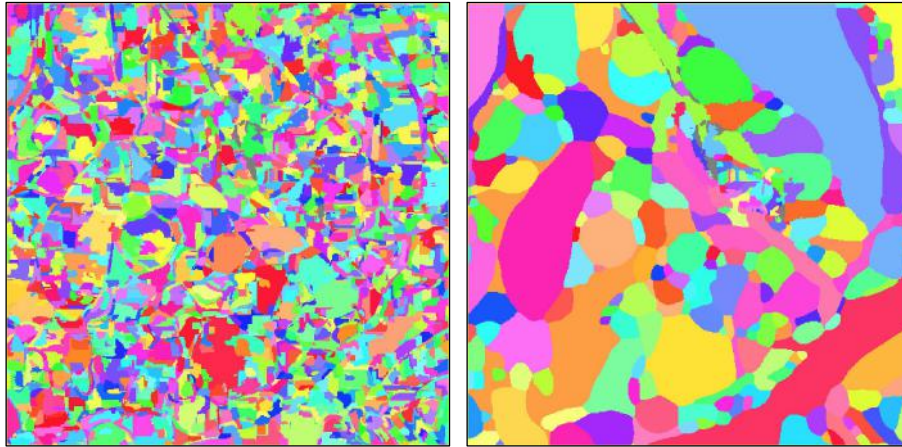with slip interpolation produces a smoothly interpolated output and heals the split error (bottom). Right: three abutting axons are not broken but affected by the slip misalignment (top), while the baseline net trained with slip interpolation produces smoothly interpolated segments (bottom).

Fig. S10. Watershed oversegmentation for the baseline. (a) To obtain initial oversegmentation, we used $T_{\min} = 1\%$, $T_{\max} = 99\%$, $T_{\text{size}} = (\theta_{\text{size}}, 1\%)$, $T_{\text{dust}} = \theta_{\text{size}}$, where $\theta_{\text{size}} = 600$ at the original resolution ($6 \times 6 \times 29$ nm^3) and $\theta_{\text{size}} = 150$ at the $2\times$ lower resolution ($12 \times 12 \times 29$ nm^3). (b) Final segmentation can be obtained by greedily agglomerating supervoxel pairs with the agglomeration score (max or mean affinity) higher than some threshold.
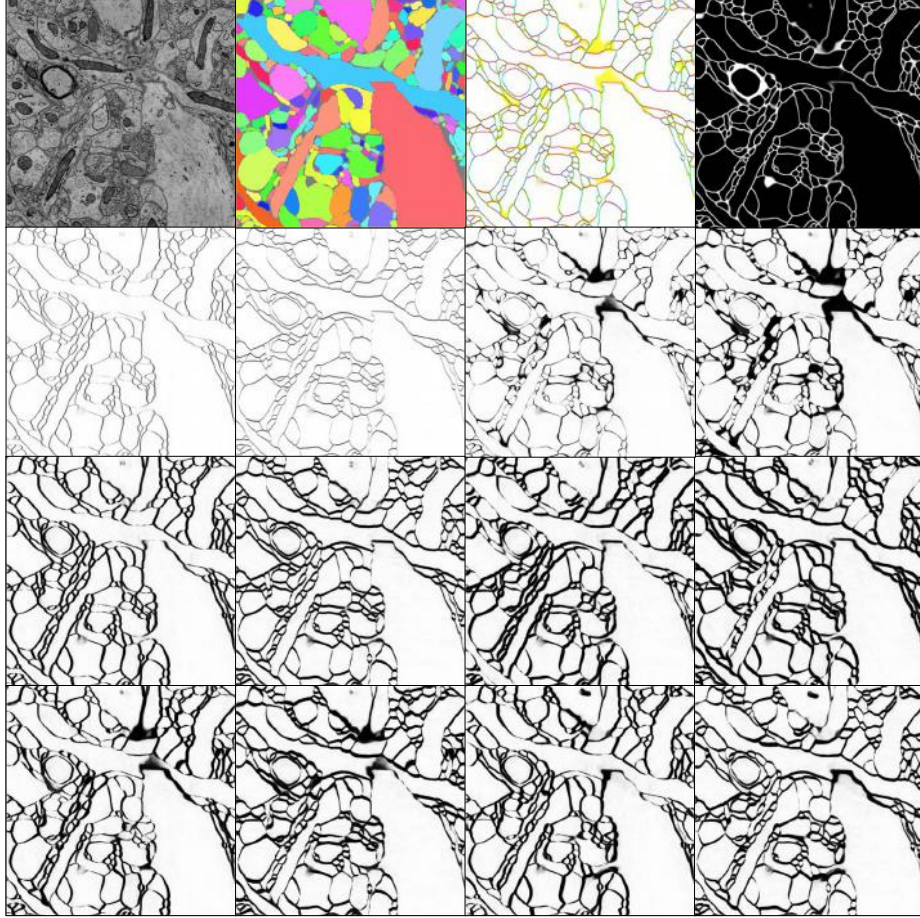
Fig. S11. Metric graph on short and long-range edges as input to the Mutex Watershed. Top row shows from left to right (1) input image from the validation set, (2) ground truth segmentation, (3) RGB visualization of the nearest neighbor metric graph, and (4) predicted background mask. For the Mutex Watershed, we compute, for every voxel, affinities on the three nearest neighbor *attractive* edges and nine longer-range *repulsive* edges. Each edge yields an affinity map with (x, y, z) offset of (-1,0,0), (0,-1,0), (0,0,-1), (0,0,-2) on the second row, (-5,0,0), (0,-5,0), (-5,-5,0), (-5,5,0) on the third row, and (-5,0,-1), (0,-5,-1), (-5,0,1), (0,-5,1) on the bottom row. We used sufficient image padding for computing the affinities near the dataset edge. Each affinity map is obtained by blending and stitching the patch-wise affinity maps derived from the patch-wise dense voxel embeddings.
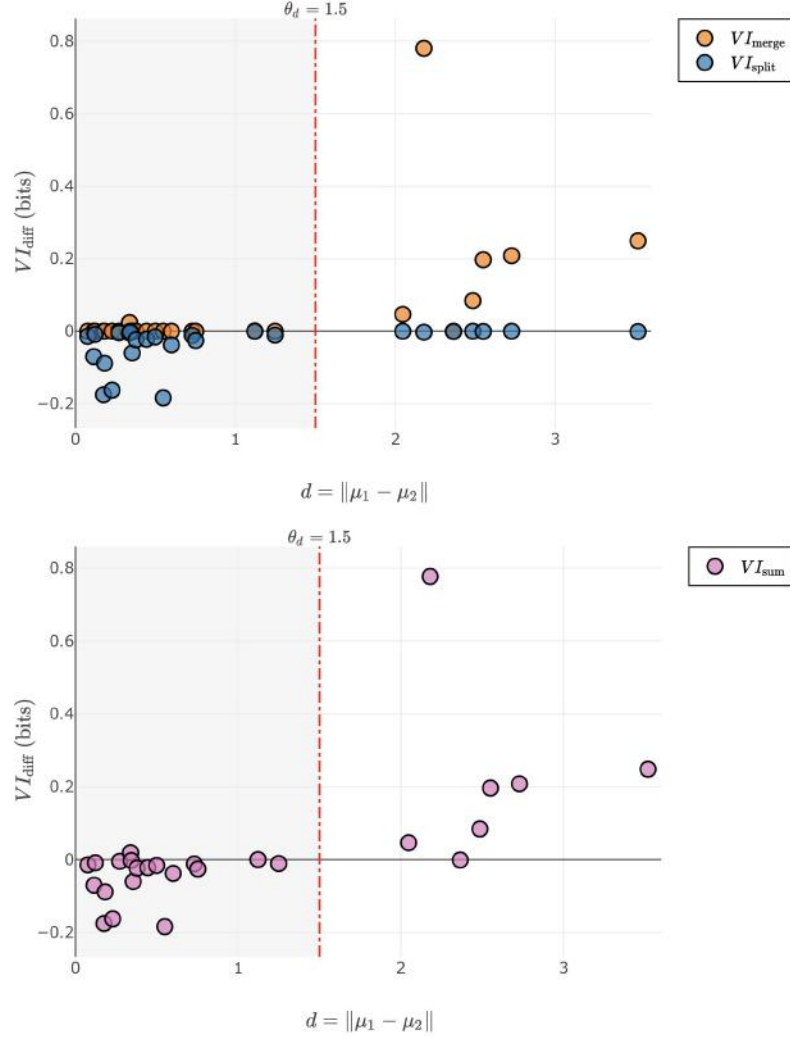
Fig. S12. Mean embedding agglomeration on the AC3 training + validation sets. We optimized the hyperparameters for the mean embedding agglomeration strictly on the combined AC3 training and validation sets. Each circle represents an individual agglomeration candidate captured by the heuristic for detecting self-touching split candidates. x-axis represents the L1 distance between two agglomeration candidate segments s_1 and s_2. y-axis represents the difference in the Variation of Information (VI) metric when the two candidate segments are *forced* to be agglomerated. Top shows the decomposition of the VI metric into VI_{merge} (orange) and VI_{split} (blue), and bottom shows the sum $VI_{\mathrm{sum}} = VI_{\mathrm{merge}} + VI_{\mathrm{split}}$. The chosen decision boundary $\theta_d = 1.5$ nicely separates the true positive (left to the decision boundary) and false positive (right to the decision boundary) incidents.
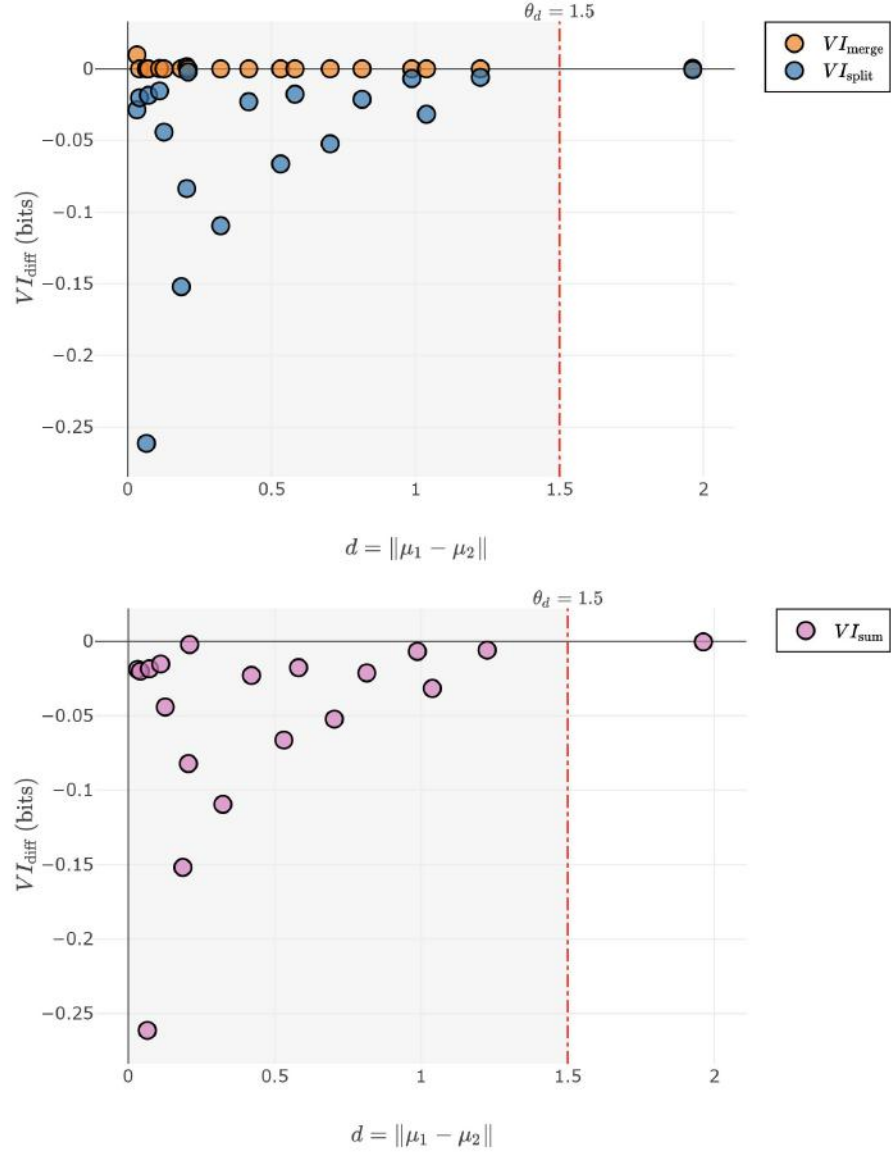
Fig. S13. Mean embedding agglomeration on the AC3 test set. This post-mortem analysis reveals that the proposed mean embedding agglomeration was very effective at agglomerating self-touching splits. Among 19 candidates detected by the heuristic, 18 were agglomerated correctly (true positive) and one was rejected incorrectly (true negative). The true negative rejection was caused by self-touching within the field of view of the embedding net.
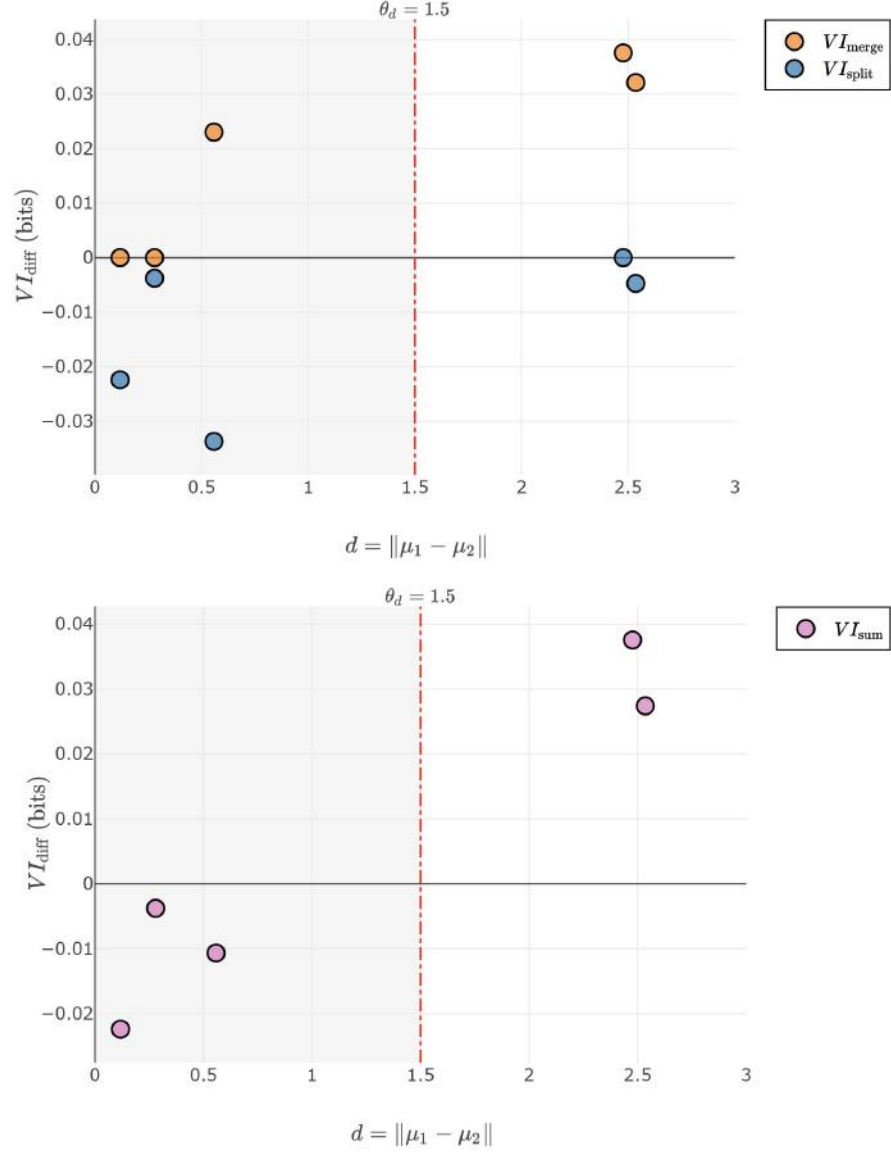
Fig. S14. Mean embedding agglomeration on the AC4 training set. AC4 does not contain many self-touching objects. The three true positive examples (left to the decision boundary) are all self-touching glial fragments.
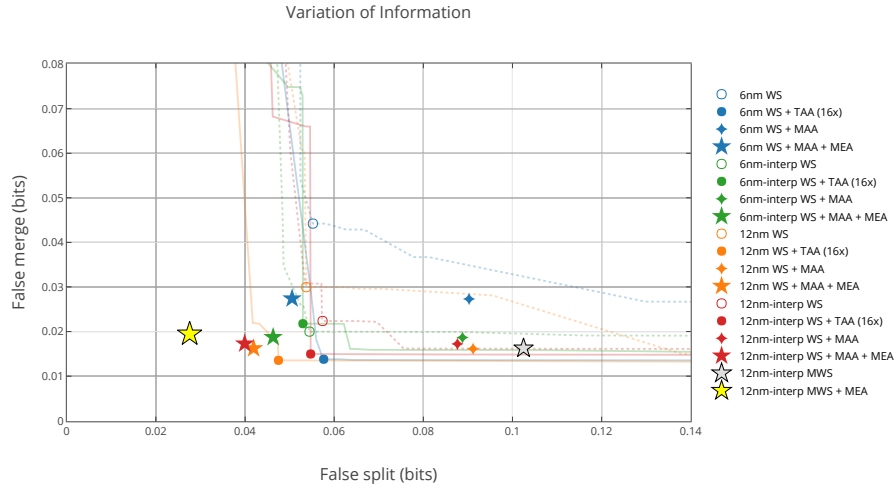
Fig. S15. Full merge-split plot on the AC3 test set. Abbreviations: watershed (WS), mean affinity agglomeration (MAA), test-time augmentation (TTA), mean embedding agglomeration (MEA), slip interpolation (interp), original resolution (6 nm), $2\times$ in-plane-downsampled resolution (12 nm). Note that the baseline models postprocessed with watershed only (WS, dashed lines) and $16\times$ test-time augmentation (TTA, solid lines) were directly optimized on the test set to find optimal operating points (empty and filled circles), taking full advantage over the proposed methods (MWS and/or MEA) that were tuned via formal model selection on the training and validation sets.
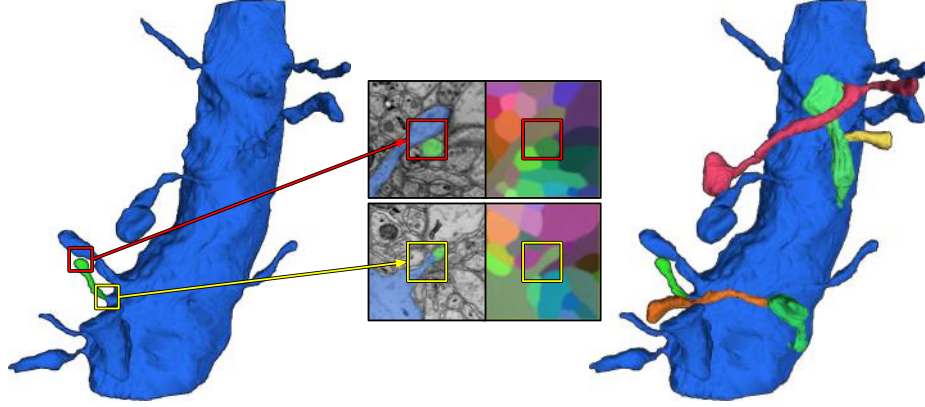
Fig. S16. Self-touching spines reattached by mean embedding agglomeration. Real examples of self-touching spines from the test set are shown here. Left: one of the six broken self-touching spines in the Mutex Watershed segmentation. Although there exists a few-voxel-thick gap formed by background voxels between the two spines, long-range repulsive edges are still able to put mutex constraints between them. Middle top: distinct vectors were assigned to the self-touching spines. Middle bottom: uniform embeddings across the false split. Right: Mean embedding agglomeration successfully reattaches all six self-touching spines to the dendritic shaft.
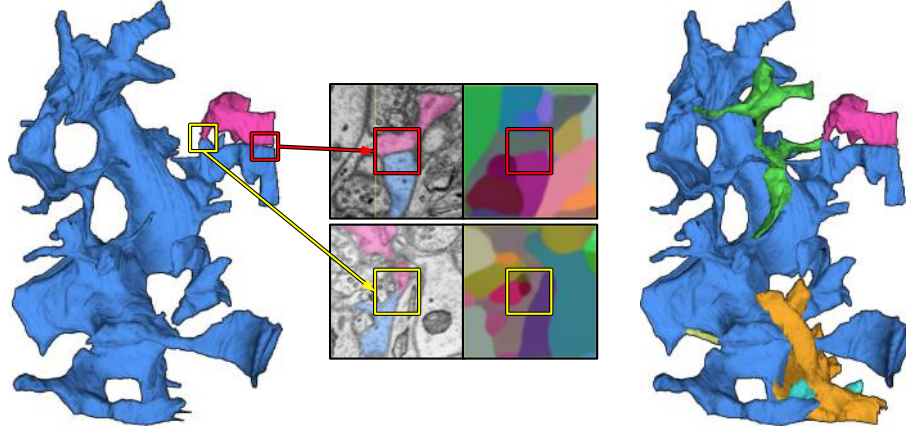

Fig. S17. Self-touching glial fragments reattached by mean embedding agglomeration. Real examples of self-touching glial fragments from the test set are shown here. Left: one of the five broken self-touching glial fragments in the Mutex Watershed segmentation. Middle top: distinct vectors were assigned to the self-contact. Middle bottom: uniform embeddings across the false split. Right: Mean embedding agglomeration successfully reattaches all five self-touching glial fragments to the main body.
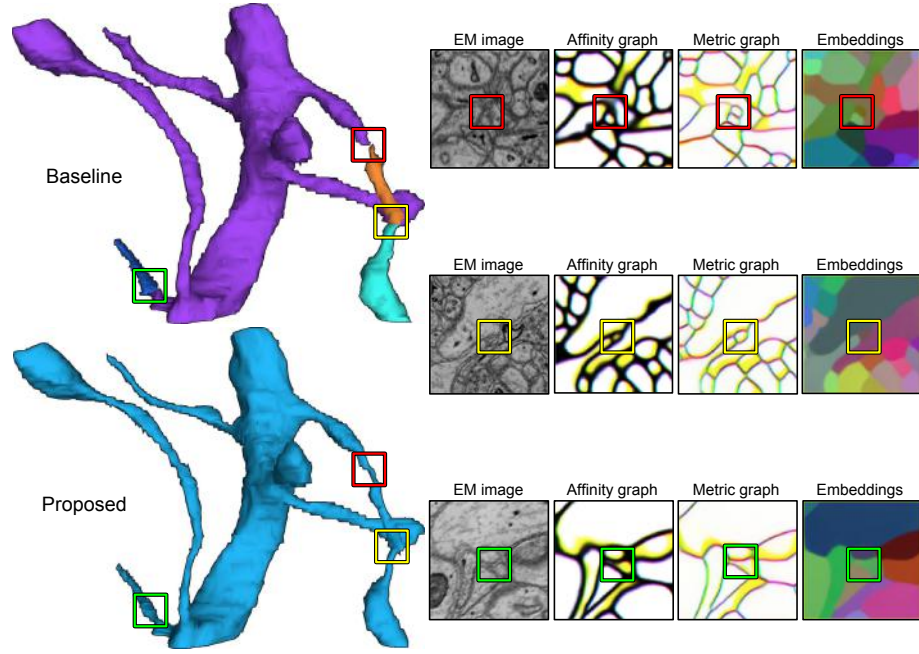
Fig. S18. Dense voxel embeddings bring substantial improvements on very thin dendritic spines. Shown here is a dendrite from the AC3 test set. The baseline segmentation (top left) has three split errors concentrated on the thin spines necks, while the proposed method makes no error (bottom left). Red box: the affinity graph directly predicted by the baseline net (second column) completely misses a very thin spine neck, whereas the nearest neighbor metric graph (third column) computed from the dense embeddings (last column) successfully detects it. Yellow box: a spine in the baseline segmentation is broken at a slight misalignment, whereas object-centered representation of the dense embeddings is very robust to such a slight misalignment, and successfully extends the spine across the misalignment. Green box: very thin constriction breaks a thin spine neck in the baseline segmentation, whereas the dense embeddings could resist it and successfully extend the spine across the constriction.
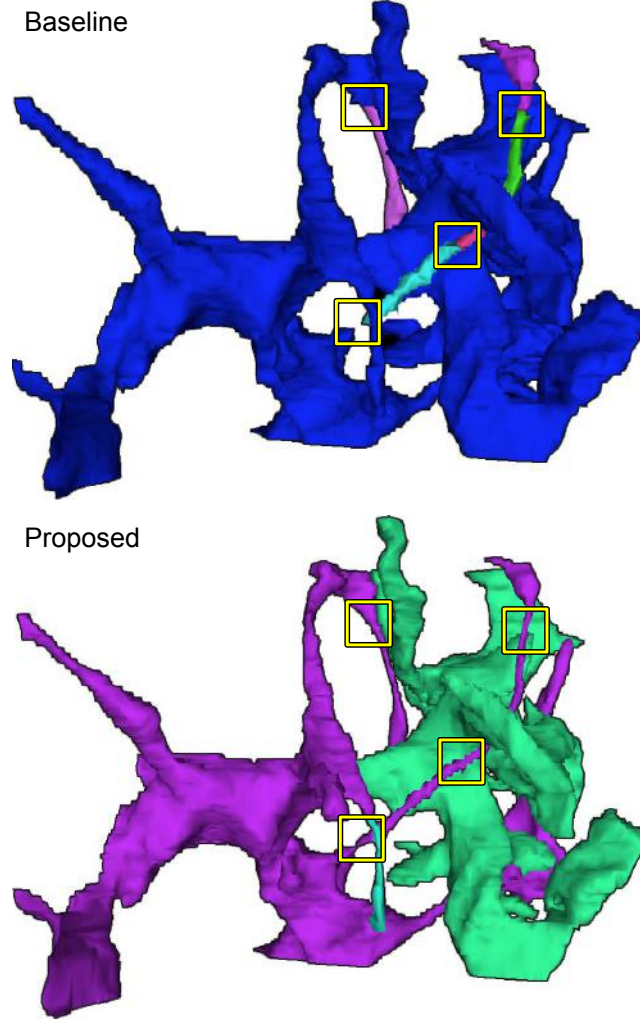
Fig. S19. Dense voxel embeddings bring substantial improvements on very thin glial processes. Shown here is a complex glia (putative astrocyte) from the extra test volume E1. The baseline segmentation (top) has systematic split errors concentrated on the very thin glial processes (yellow boxes), whereas object-centered representation of the dense embeddings successfully extends them. However, the proposed method makes a couple of split errors (green and cyan segments) due to the conservative repulsive constraints put by the Mutex Watershed on the self-touching glial fragments. Here mean embedding agglomeration was unable to perfectly heal the self-touching split errors, due to the failures of the heuristic for detecting self-touching split candidates.
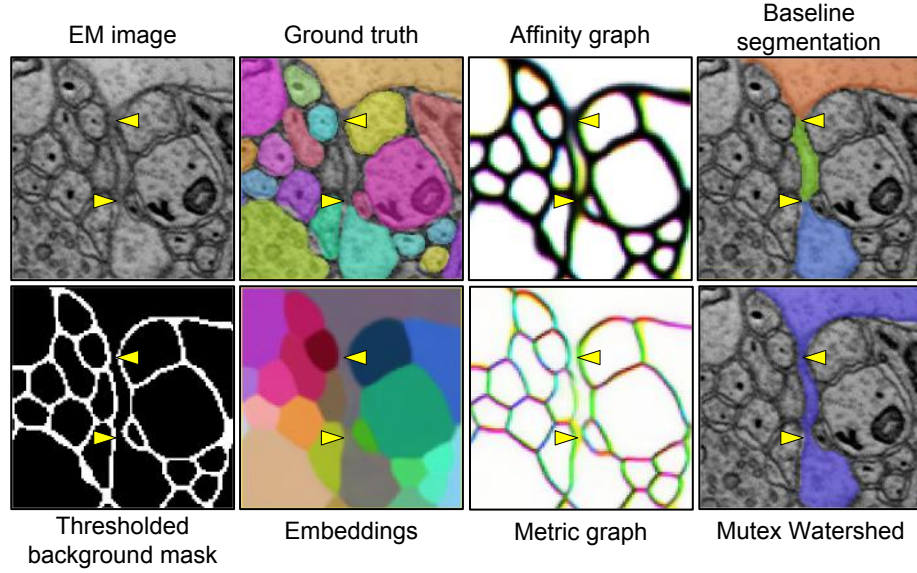
Fig. S20. A very thin spine neck that is parallel to the sectioning plane is missing in the ground truth annotation of the training set (AC4). Top from left to right: (1) EM image, (2) mistakes in the ground truth segmentation, (3) RGB visualization of the nearest neighbor affinity graph directly predicted by the baseline net, obtained with $16\times$ test-time augmentation, (4) split errors in the resulting baseline segmentation. Bottom: from left to right (1) thresholded background mask ($\theta_{\text{background}} = 0.6$) predicted by the embedding net, (2) dense embeddings, (3) RGB visualization of the nearest neighbor metric graph, (4) Mutex Watershed segmentation. Object-centered representation of the dense embeddings enables a correct segmentation of the very difficult thin spine neck that is parallel to the sectioning/imaging plane, despite of the wrong annotation in the ground truth.
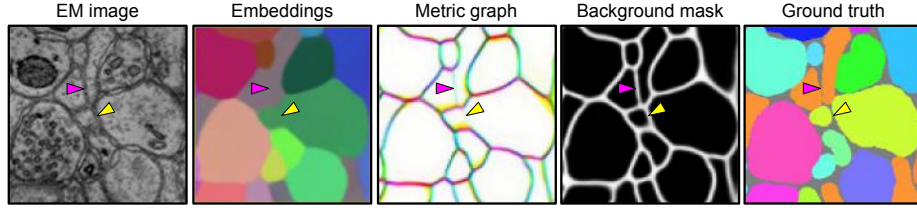
EM image	Embeddings	Metric graph	Background mask	Ground truth

Fig. S21. Self-touching "memorization" in the training set (AC3). Although the self-touching spines (yellow arrowheads) and glia (magenta arrowheads) should be treated as distinct objects in the local training patch of limited context, they are not properly separated by background voxels in the ground truth annotation. As a consequence, the embedding net "memorized" these examples and assigns uniform vectors across the self-contacts despite the obvious local evidence for disconnectivity. Note that the embedding net also memorized the background mask (fourth column) despite the self-contradiction with the embeddings. From left to right: (1) EM image, (2) dense embeddings, (3) RGB visualization of the nearest neighbor metric graph, (4) background mask predicted by the embedding net, (5) ground truth.
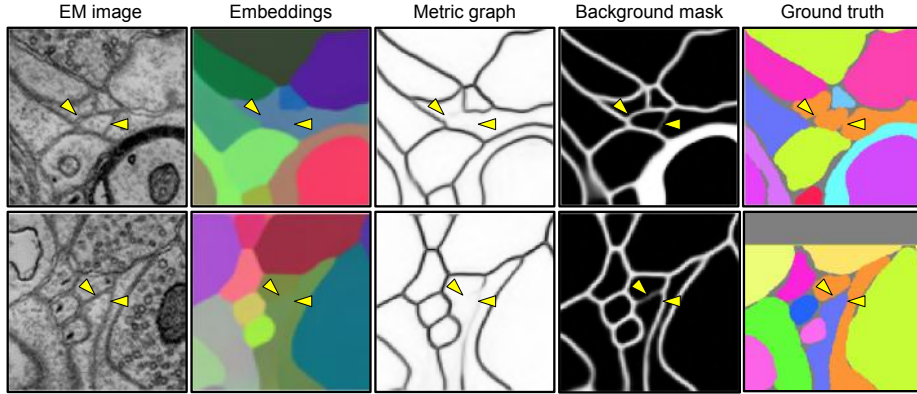
EM image	Embeddings	Metric graph	Background mask	Ground truth

Fig. S22. The embedding net is confused by complex glial self-contacts in the training set (AC3). Top: the embedding net "memorizes" the self-touching glia in the training set (orange object, last column) by assigning uniform embeddings across the self-contacts (yellow arrowheads, second column), effectively erasing boundaries in the nearest neighbor metric graph (third column). Bottom: the embedding net makes a mistake on a similar-looking location in the training set where the contacts are between two distinct glia this time (yellow arrowheads, last column). As can be seen here, glia with complex morphology (putative astrocytes) make numerous self-contacts that are not properly separated by background voxels in the ground truth annotation. This becomes a significant source of noise during training, systematically compromising the embedding net's performance around glia, even in the training set. For metric graph (third column), we show $\min(a_x, a_y)$ where a_x/a_y are x/y nearest neighbor metric-derived affinities. To visualize embeddings, we used PCA to project the 24-dimensional embedding space onto the three dimensional RGB color space.
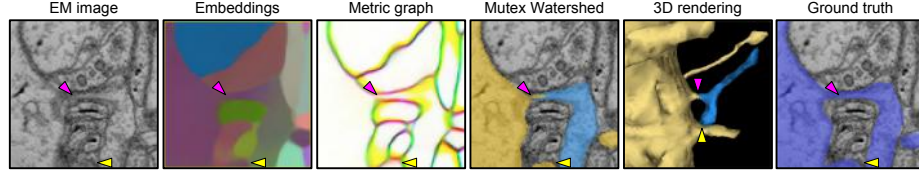
EM image	Embeddings	Metric graph	Mutex Watershed	3D rendering	Ground truth

Fig. S23. Self-touching within patch. An example dendritic spine that makes a self-contact (yellow arrowheads) within the field of view of the embedding net is taken from the training set (AC3). The embedding net fails to assign uniform vectors across the dendritic shaft and the self-touching spine (magenta arrowheads), despite that the net was trained on this example. From left to right: (1) EM image, (2) dense embeddings, (3) RGB visualization of the nearest neighbor metric graph, (4) Mutex Watershed segmentation, (5) 3D rendering, (6) ground truth.
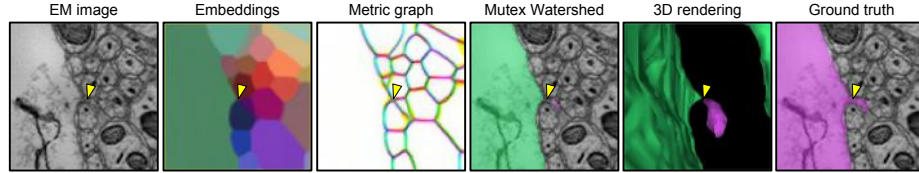
EM image	Embeddings	Metric graph	Mutex Watershed	3D rendering	Ground truth

Fig. S24. A tiny broken spine from the validation set (AC3). The embedding net assigns completely distinct vectors to the dendritic shaft and the tiny spine (yellow arrowheads). From left to right: (1) EM image, (2) dense embeddings, (3) RGB visualization of the nearest neighbor metric graph, (4) Mutex Watershed segmentation, (5) 3D rendering, (6) ground truth.
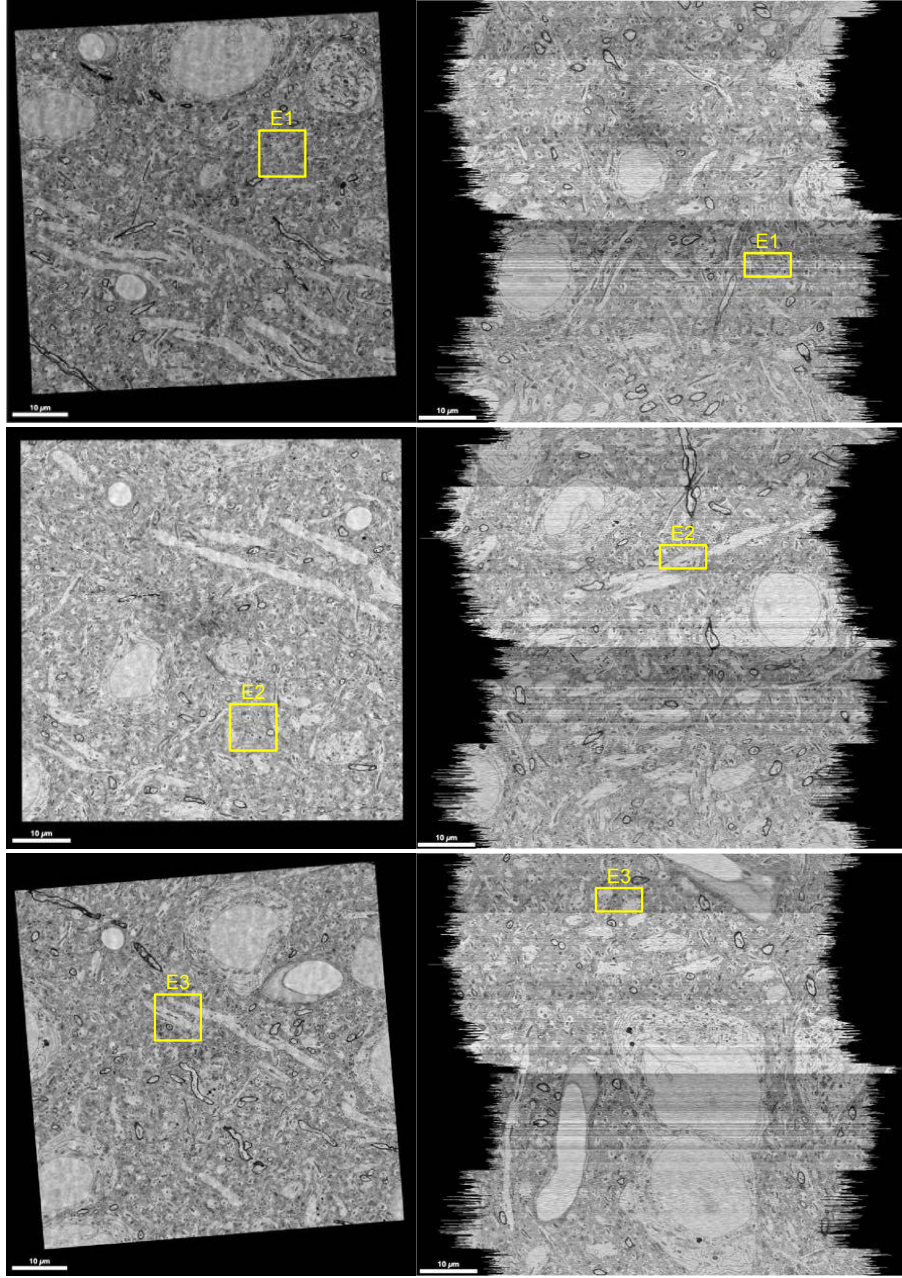
Fig. S25. Location of the extra test volumes E1–E3 in the full Kasthuri *et al.* dataset [13]. Left column: xy view. Right column: xz reslice view. E1's bounding box: (14300,12780,1099)–(16348,14828,1199). E2's bounding box: (10600,16200,516)–(12648,18248,616). E3's bounding box: (7800,9400,150)–(9848,11448,250).