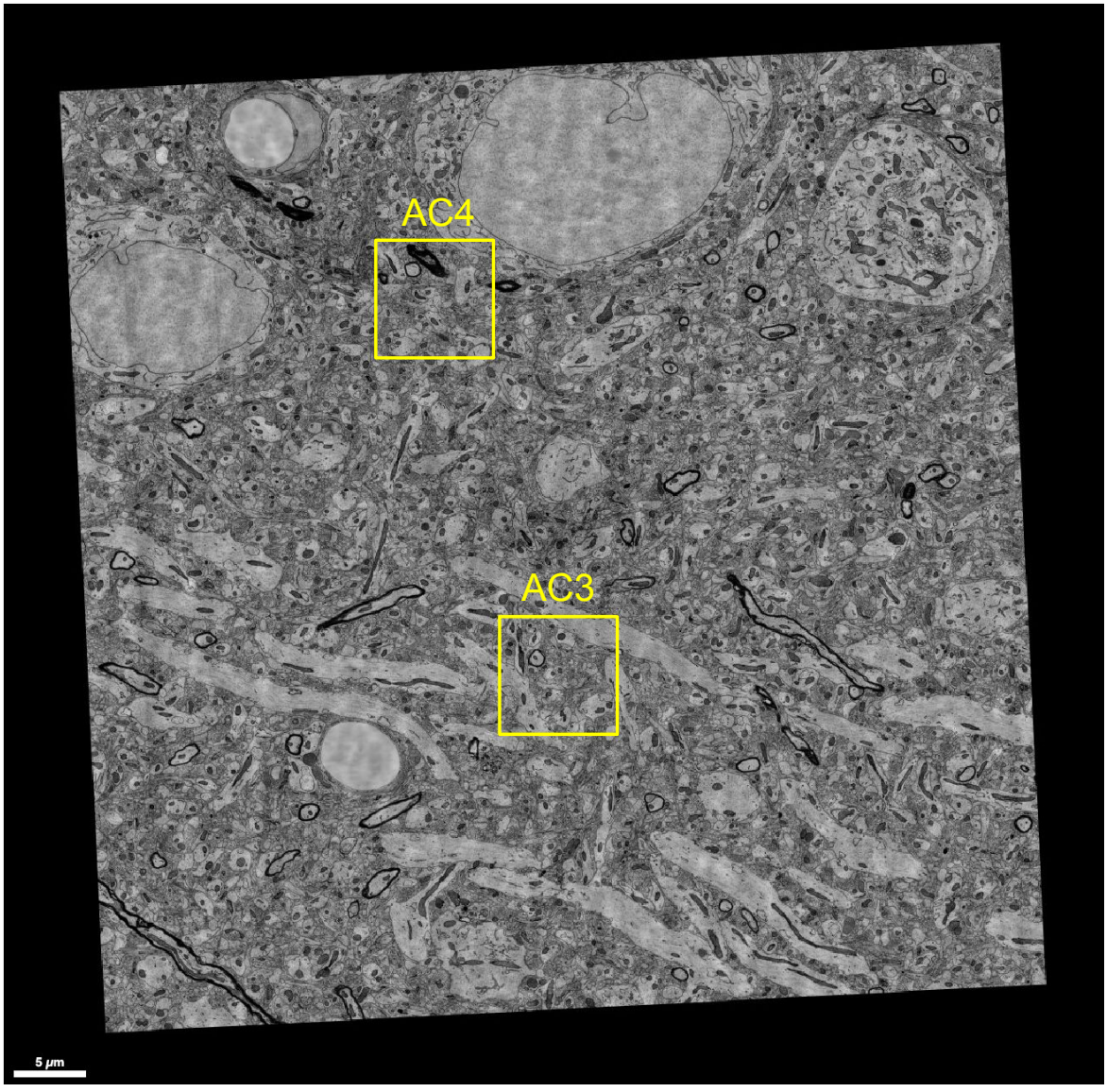

Fig. S1. Location of AC3/AC4 in the full dataset of Kasthuri *et al.* (2015). Shown here is the section at $z = 1099$. The full dataset is accessible via https://neurodata.io/data/kasthuri15/. AC3's bounding box in the full dataset is (10944,17424,999)–(12992,19472,1255), and AC4's bounding box is (8800,10880,1099)–(10848,12928,1199). Note that the original images were acquired at $3 \times 3 \times 29$ nm^3 voxel resolution. As a result, the bounding boxes are $2\times$ larger in x and y dimension than the AC3/AC4 dataset, which was annotated at $6 \times 6 \times 29$ nm^3 voxel resolution.

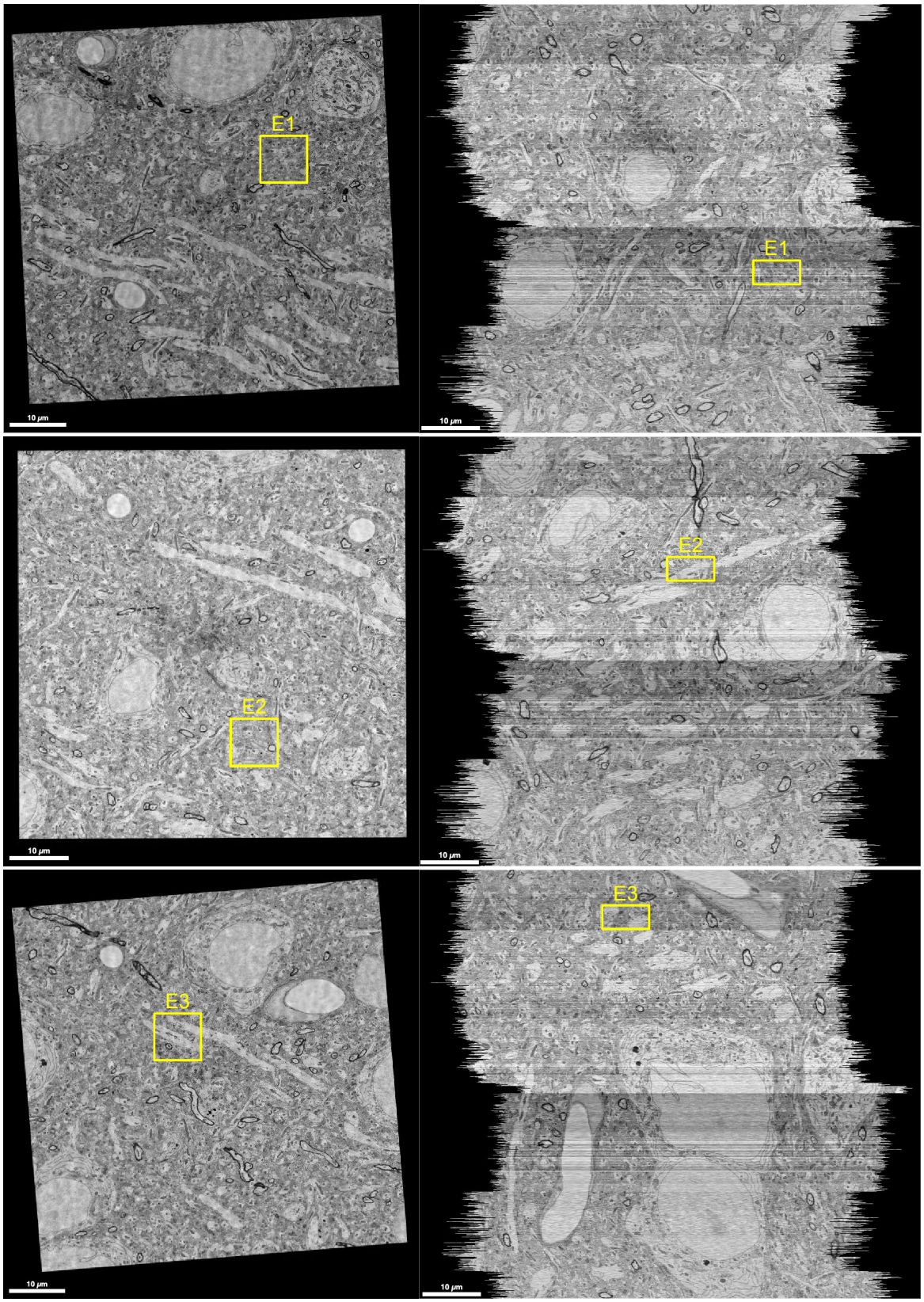

Fig. S2. Location of the extra test volumes E1–E3 in the full dataset of Kasthuri *et al.* (2015). **Left column:** xy view. **Right column:** xz reslice view. E1's bounding box: (14300,12780,1099)–(16348,14828,1199). E2's bounding box: (10600,16200,516)–(12648,18248,616). E3's bounding box: (7800,9400,150)–(9848,11448,250).

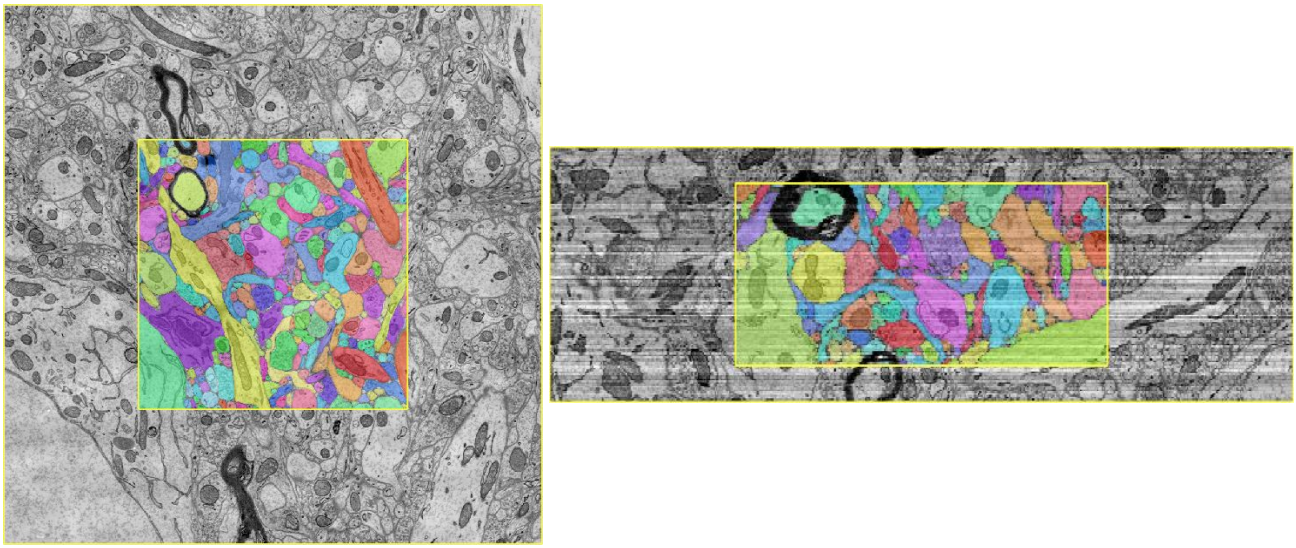

Fig. S3. Visualization of AC4 with surrounding image context. **Left:** xy view. **Right:** xz reslice view. We used the entire AC4 for training. In the original AC4 annotation, myelin sheath is not labeled separately. We additionally labeled myelin sheaths as separate objects.

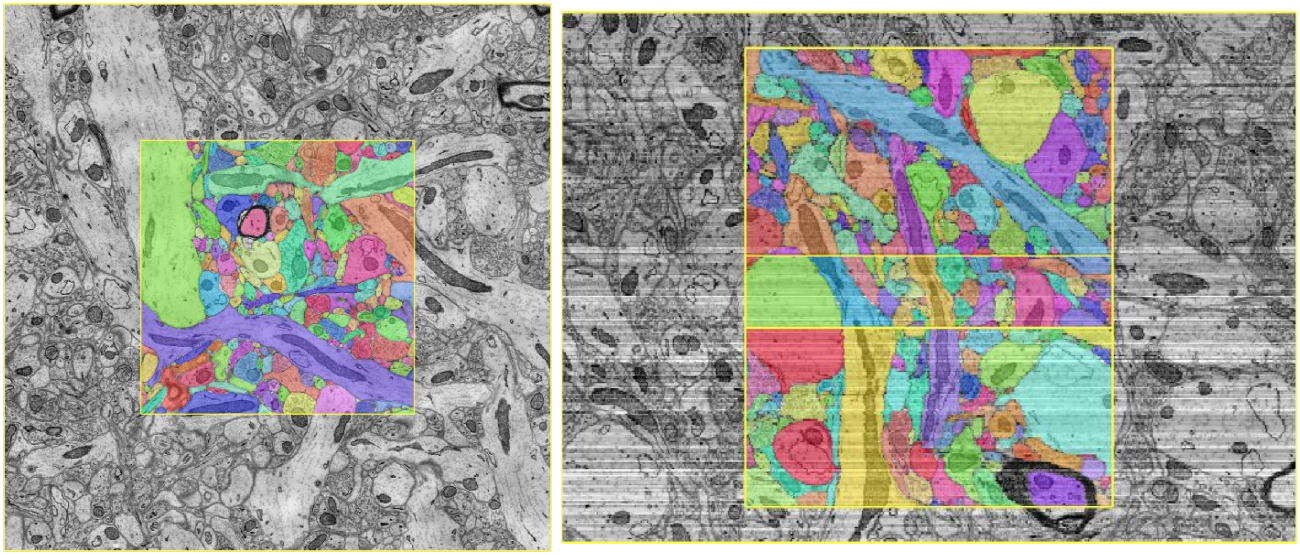

Fig. S4. Visualization of AC3 with surrounding image context. **Left:** xy view. **Right:** xz reslice view. We used the top 116 slices for training, the middle 40 slices for validation, and the bottom 100 slices for testing. Note that the volume shown here is a flipped version of the downloadable AC3 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 labeled 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 were treated as foreground objects in $\mathcal{L}_{\text{embedding}}$.

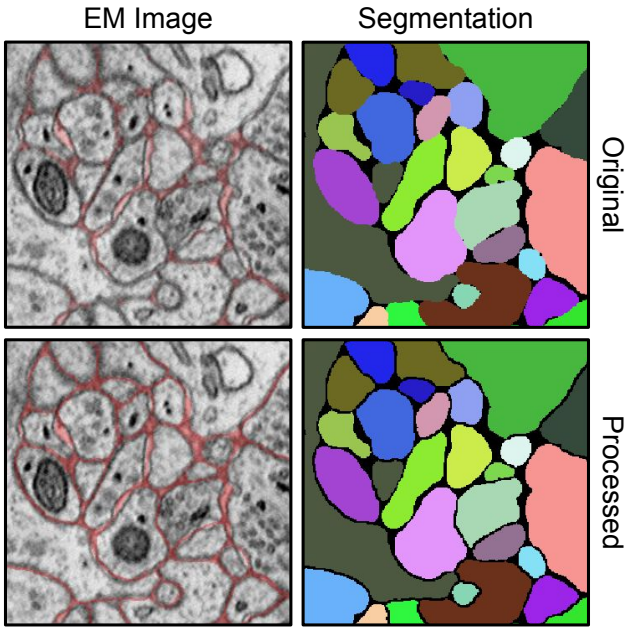

Fig. S5. Data preprocessing. For input image, we linearly rescaled the original integer-valued intensity in $[0, 255]$ to the real-valued intensity in $[0, 1]$. For background mask prediction, we applied a $3 \times 3 \times 1$ "background-augmenting" kernel to the ground truth segmentation such that 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) was additionally marked as background.

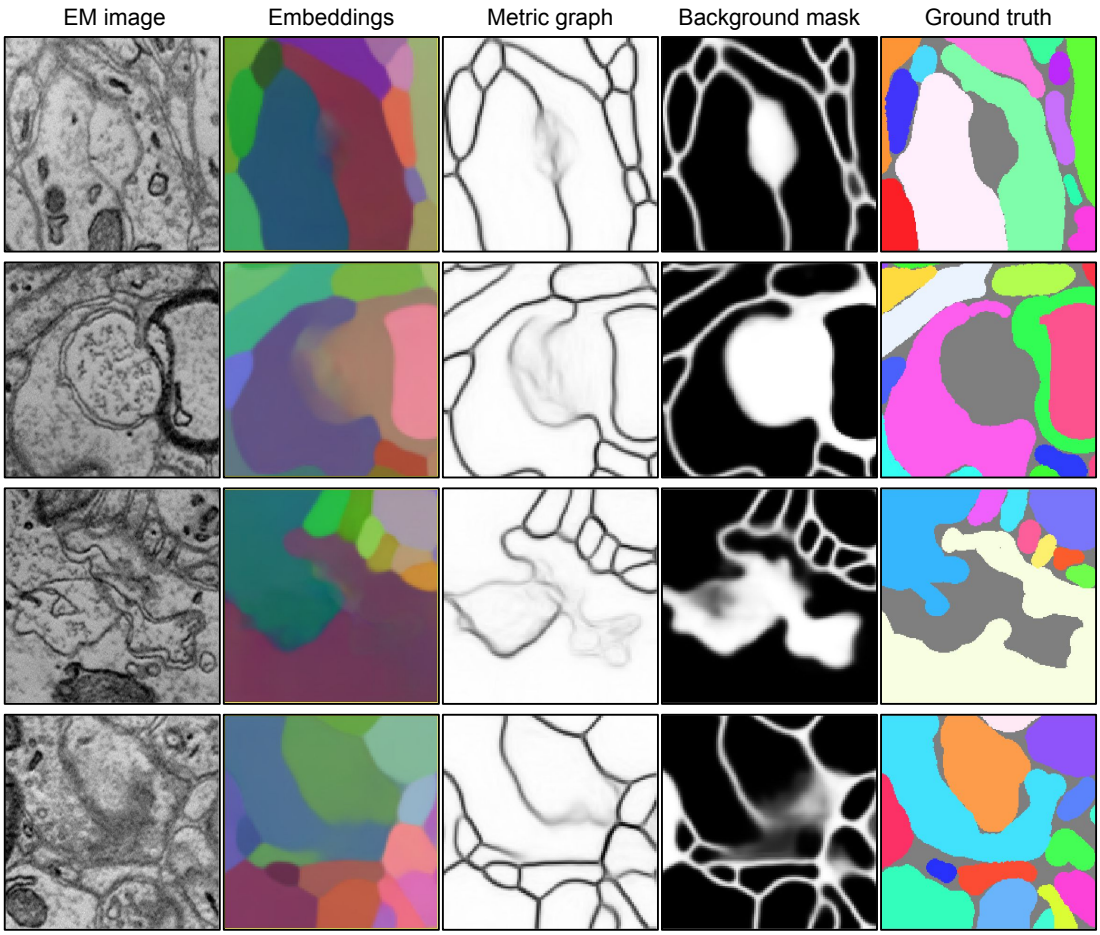

Fig. S6. Noisy embeddings resulted from the exclusion of background voxels from $\mathcal{L}_{\text{embedding}}$. **Top row:** extracellular space between adjacent objects is enlarged, forming an object-like structure that is labeled as background. **Second/third rows:** endoplasmic reticulum from each 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 was 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 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. To visualize metric graph (third column), we used $\min(a_x, a_y)$ where a_x/a_y are x/y nearest neighbor metric-derived affinities.

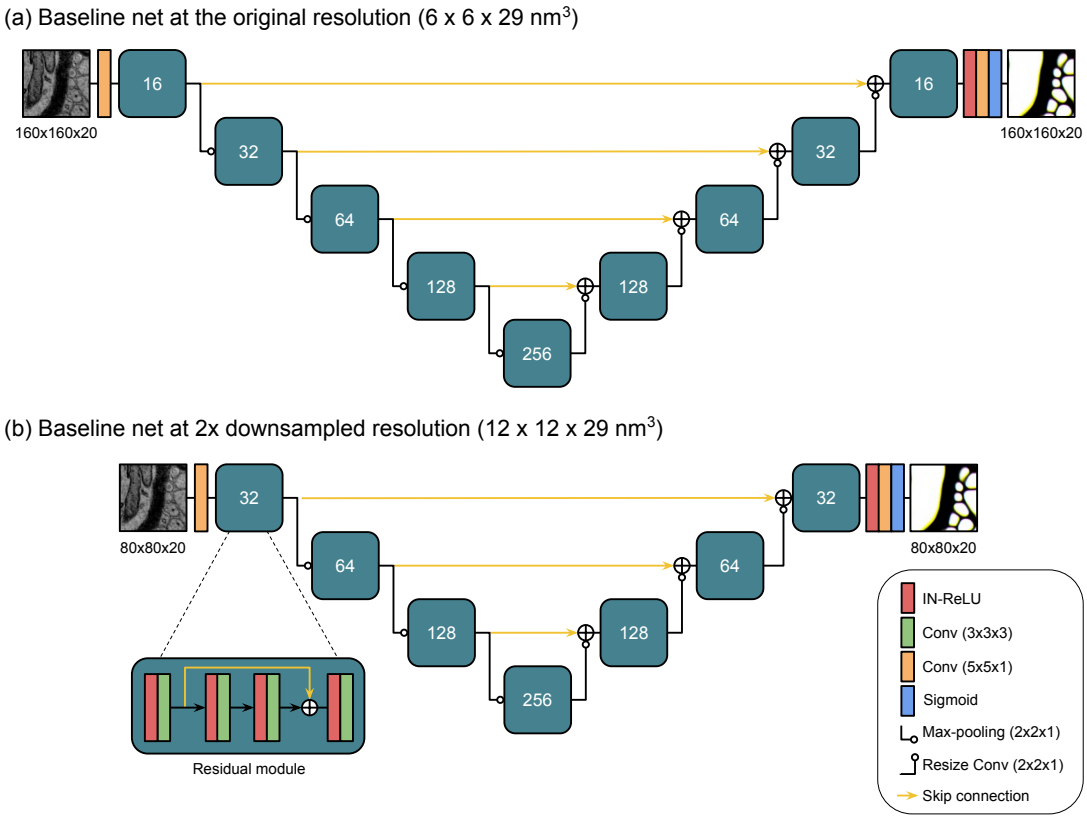

Fig. S7. Baseline net architecture. (a) We used a training patch of $160 \times 160 \times 20$ and an inference patch of $320 \times 320 \times 20$ voxels at the original resolution ($6 \times 6 \times 29$ nm^3). (b) For $2\times$ in-plane-downsampled images ($12 \times 12 \times 29$ nm^3), we accordingly halved the training and inference patch sizes in x and y dimension. 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 accuracy after postprocessing. Besides predicting nearest neighbor affinities as a primary target, we used long-range affinities as an auxiliary target during training. For generating long-range affinity maps, we used the following offset vectors: $(-4, 0, 0)$, $(-8, 0, 0)$, $(-24, 0, 0)$, $(0, -4, 0)$, $(0, -8, 0)$, $(0, -24, 0)$, $(0, 0, -2)$, $(0, 0, -3)$, and $(0, 0, -4)$ at the original resolution. We halved the x and y offsets for $2\times$ in-plane-downsampled images. As a result, the baseline nets produce 12 output channels, three for nearest neighbor and nine for long-range affinities. Here the dimensionality of output channels is omitted for brevity and only spatial dimensions are displayed. The number inside the residual module represents the width (number of feature maps) of the module. For upsampling, we used the bilinear *resize convolution*, i.e., bilinear upsampling followed by a pointwise $(1 \times 1 \times 1)$ convolution. Abbreviations: Instance Normalization (IN), rectified linear unit (ReLU), convolution (Conv).

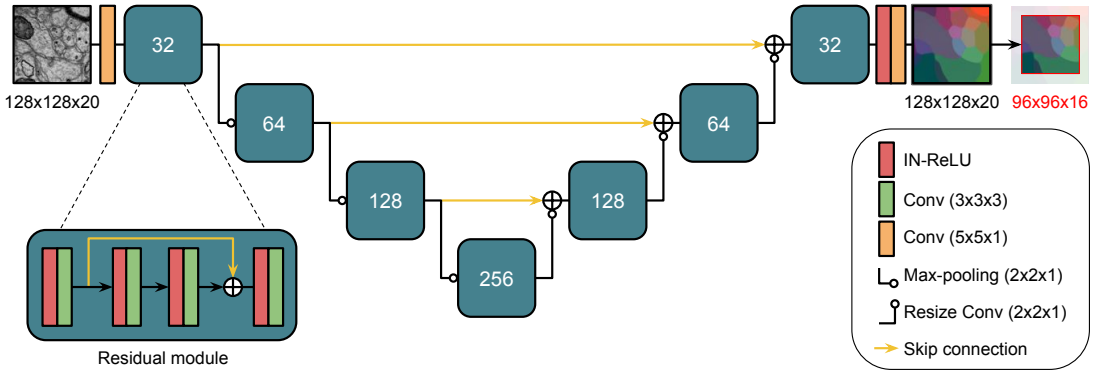

Fig. S8. Embedding net architecture. The number inside each of the residual modules represents the width (number of feature maps) of the module. Here the dimensionality of output embeddings is omitted for brevity and only spatial dimensions are displayed. Abbreviations: Instance Normalization (IN), rectified linear unit (ReLU), convolution (Conv).

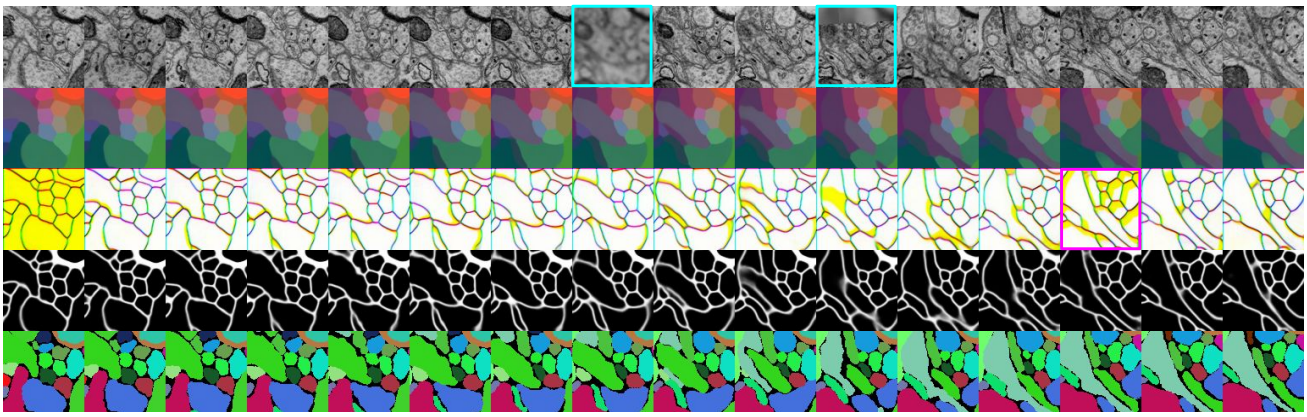

Fig. S9. Visualization of an example training patch. Each row is a flattened version of $96 \times 96 \times 16$ image patch. From top to bottom, each row shows (1) EM images, (2) dense voxel embeddings, (3) nearest neighbor metric graph derived from the embeddings, (4) predicted background mask, and (5) ground truth segmentation. Yellow-colored regions in the visualization of nearest neighbor metric graph (third row) indicate disconnectivity in z-direction. In this particular training example, one full and one partial simulated out-of-focus sections (cyan boxes) and one simulated misalignment (magenta box) were injected.

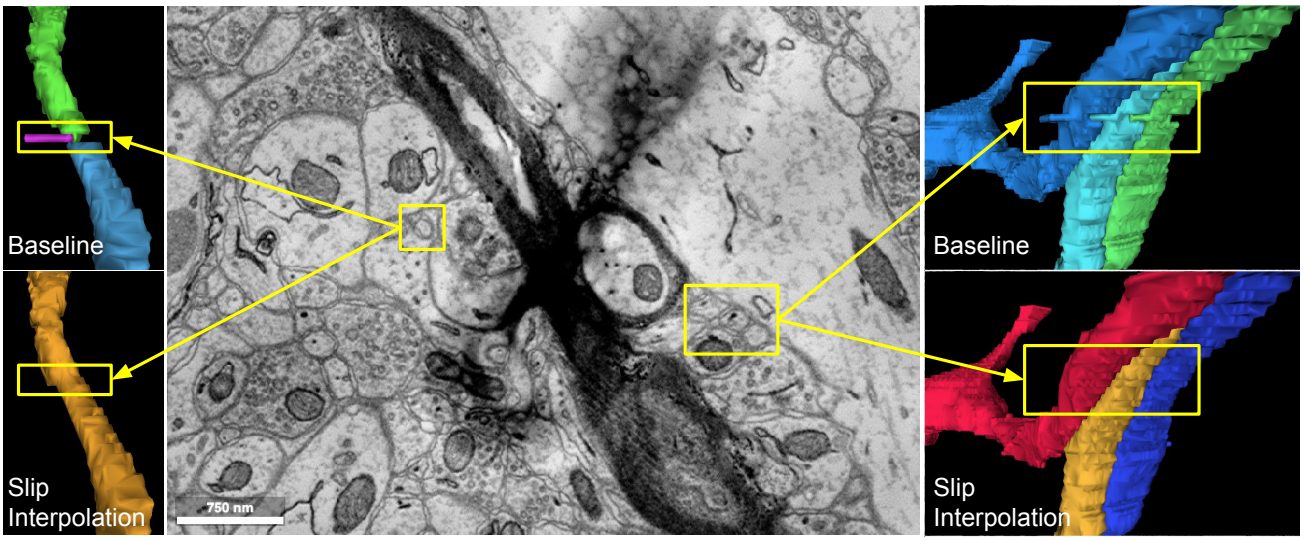

Fig. S10. Effect of slip interpolation. A mild fold (dark shade over the myelinated axon) in the test set causes a slip-type misalignment. **Left:** a thin axon is broken at the slip misalignment (top), whereas the baseline net trained with slip interpolation produces a smoothly interpolated output and heals the split error (bottom). **Right:** three abutting axons are not broken but affected by the slip misalignment (top), whereas the baseline net trained with slip interpolation produces smoothly interpolated segments (bottom).

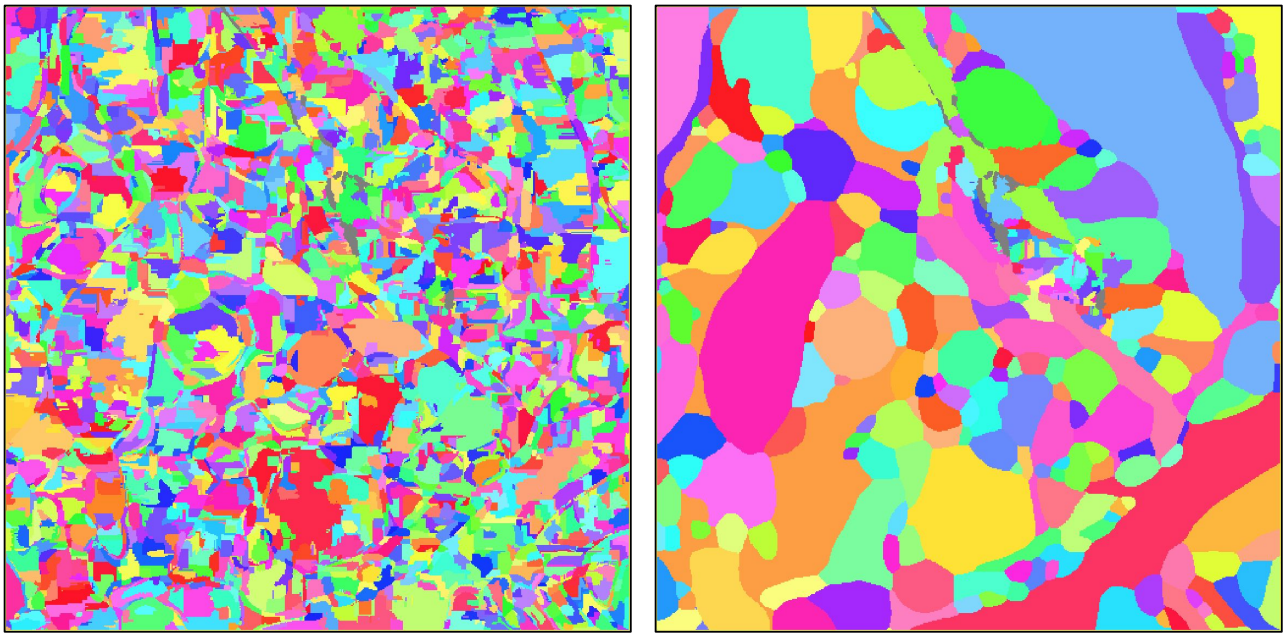

Fig. S11. Watershed oversegmentation for the baseline. (a) To obtain initial oversegmentation, we used $T_{\text{min}} = 1\%$, $T_{\text{max}} = 99\%$, $T_{\text{size}} = (\theta_{\text{size}}, 1\%)$, $T_{\text{dust}} = \theta_{\text{size}}$, where $\theta_{\text{size}} = 600$ at the original resolution ($6 \times 6 \times 29$ nm^3) and $\theta_{\text{size}} = 150$ at the $2\times$ in-plane-downsampled resolution ($12 \times 12 \times 29$ nm^3). (b) Final segmentation can be obtained by greedily agglomerating supervoxel pairs whose *agglomeration score* (max or mean affinity) are higher than a tunable threshold.

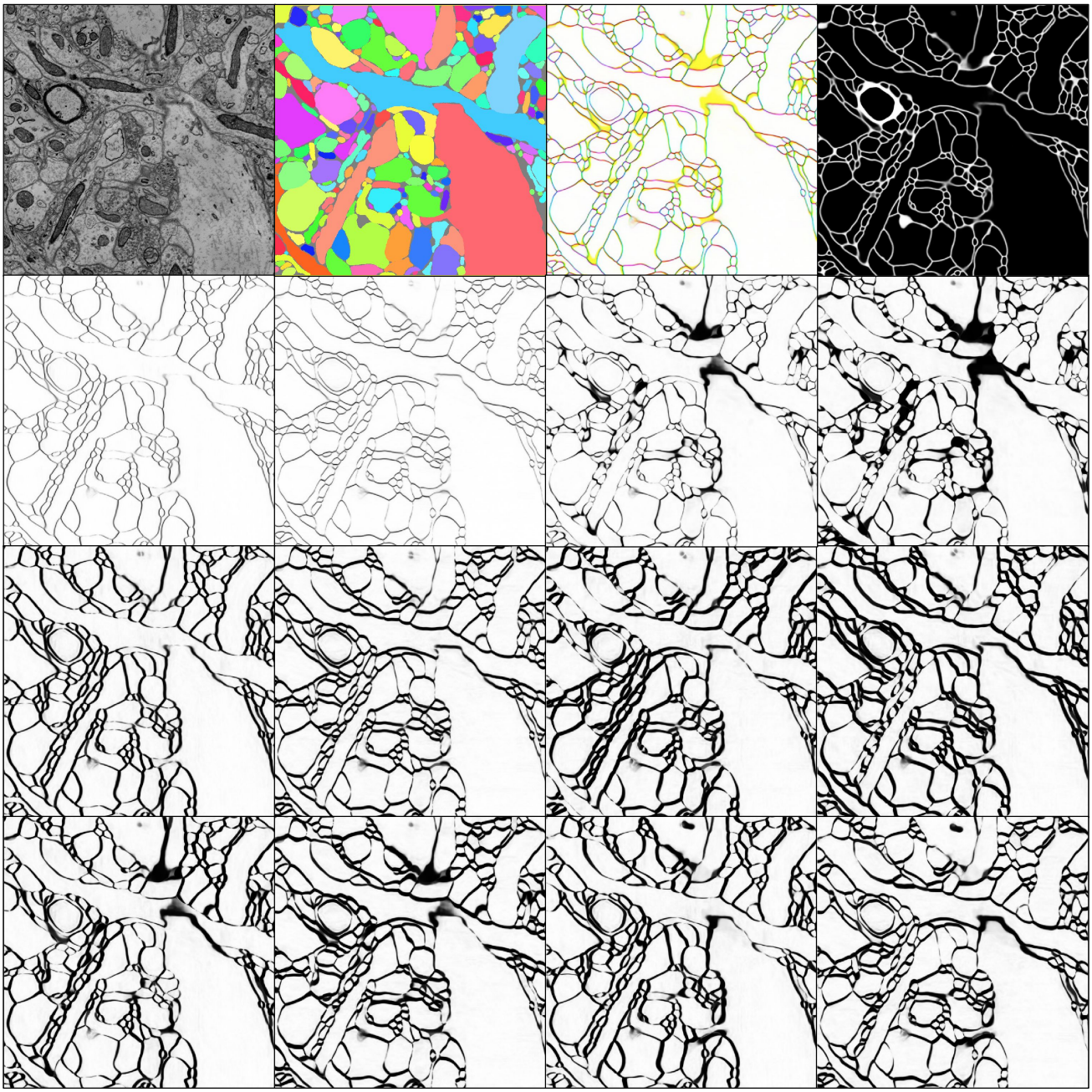

Fig. S12. Metric graph on short and long-range edges as input to the Mutex Watershed. Top row shows from left to right (1) input image from the validation set, (2) ground truth segmentation, (3) RGB visualization of the nearest neighbor metric graph, and (4) predicted background mask. For the Mutex Watershed, we used three nearest neighbor *attractive* edges and nine long-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 when computing the affinities near the dataset edge. Each affinity map is obtained by stitching and blending the patch-wise affinity maps derived from the patch-wise dense voxel embeddings.

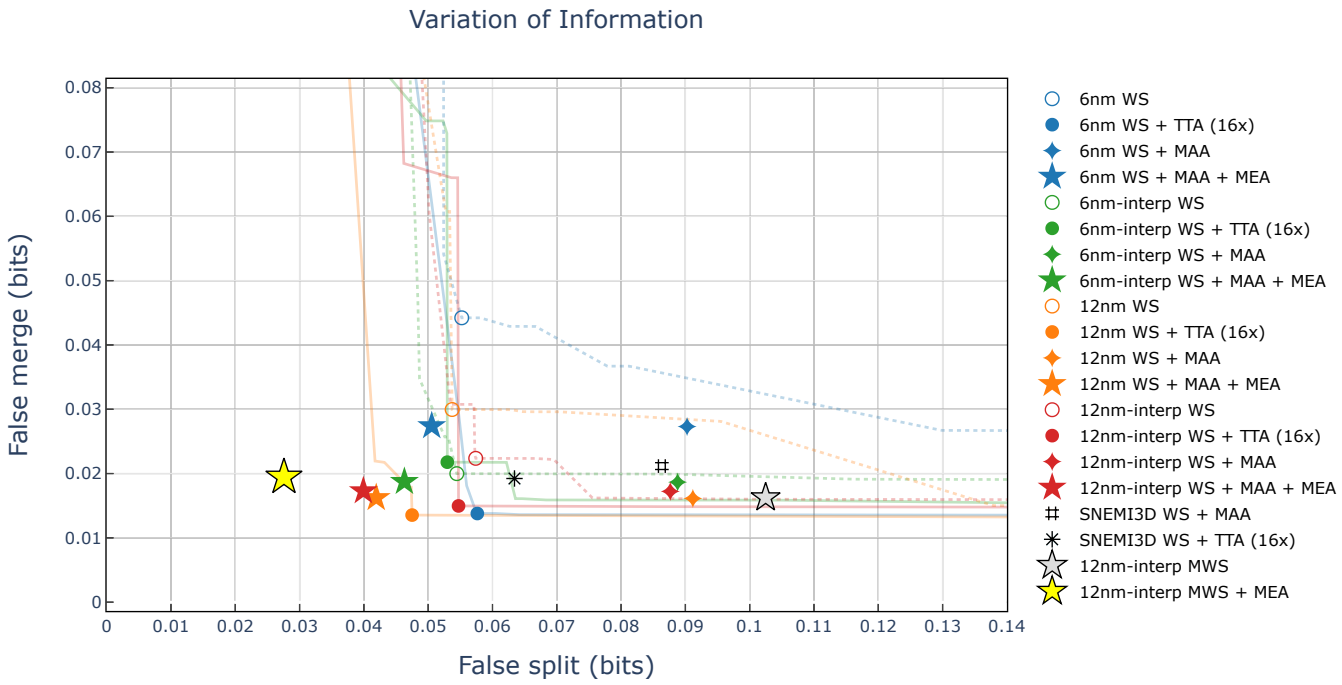

Fig. S13. Full merge-split plot on the test set. For comparison, we have included the SNEMI3D submission results of Lee *et al.* (2017). Asterisk (*) represents the top entry of the SNEMI3D leaderboard by Lee *et al.* (2017) based on 16× test-time augmentation (adapted Rand error of 0.0249 reported by the SNEMI3D leaderboard). Pound sign (#) represents the SNEMI3D submission result by Lee *et al.* (2017) based on mean affinity agglomeration (adapted Rand error of 0.0314 reported by the SNEMI3D leaderboard). Abbreviations: watershed (WS), mean affinity agglomeration (MAA), test-time augmentation (TTA), mean embedding agglomeration (MEA), slip interpolation (interp), original image resolution (6 nm), 2× in-plane-downsampled image resolution (12 nm).

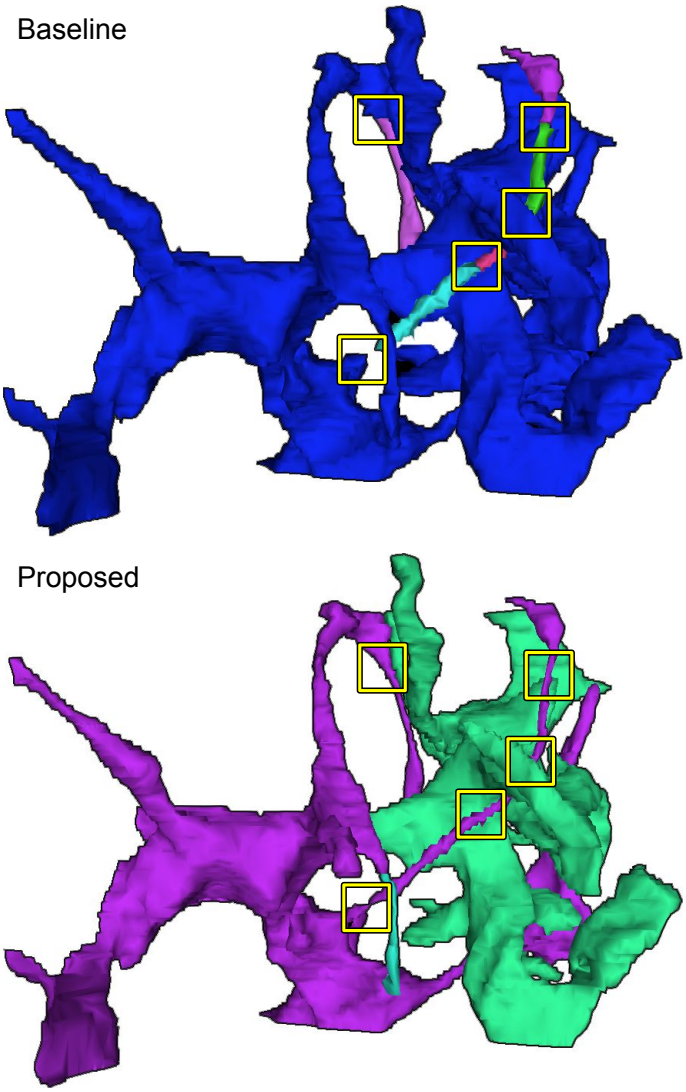

Fig. S14. Dense voxel embeddings bring substantial accuracy gain for very thin glia processes. Shown here is a complex glia (putative astrocyte) from the extra test volume E1. The baseline segmentation (top) has systematic split errors concentrated on the very thin glia processes (yellow boxes), whereas our proposed method successfully extends them. However, the proposed method made a couple of split errors (green and cyan segments) due to the conservative repulsive constraints put by the Mutex Watershed on self-contact. Here mean embedding agglomeration failed to resolve a couple of self-contact split errors, mainly due to the heuristic's failure in detecting them.

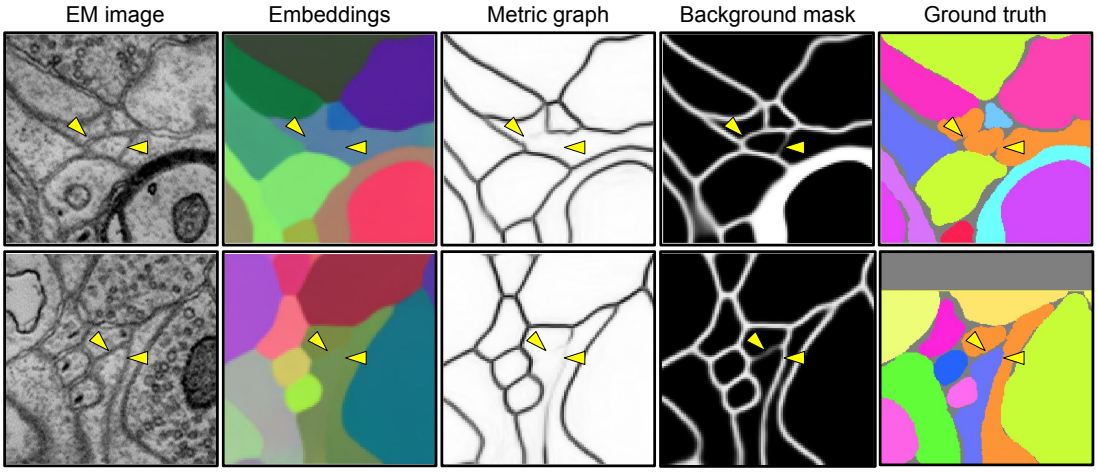

Fig. S15. The embedding net is confused by complex glia self-contacts in the training set (AC3). **Top:** the embedding net "memorizes" the glia self-contacts in the training set (orange object, last column) by assigning uniform embeddings across the self-contacts (yellow arrowheads, second column), 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. To visualize metric graph (third column), we used $\min(a_x, a_y)$ where a_x and a_y are nearest neighbor metric-derived affinities in x and y directions, respectively.

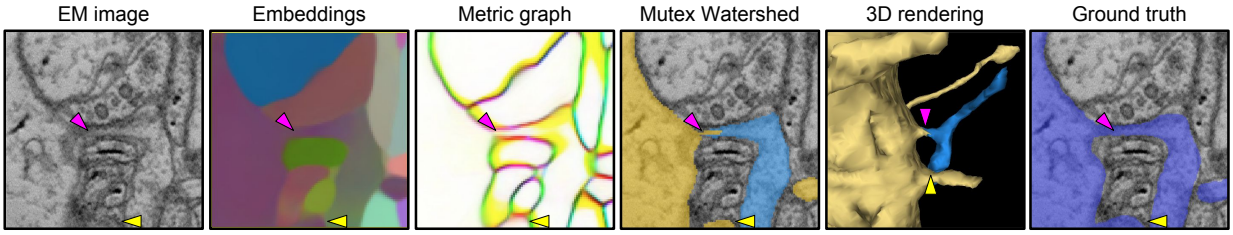

Fig. S16. Self-contact within local patch. A dendritic spine that bends back and makes a self-contact (yellow arrowheads) within the field of view of the embedding net, taken from the training set (AC3). The embedding net fails to assign uniform vectors across the dendritic shaft and spine (magenta arrowheads), despite that the net was trained on this example.

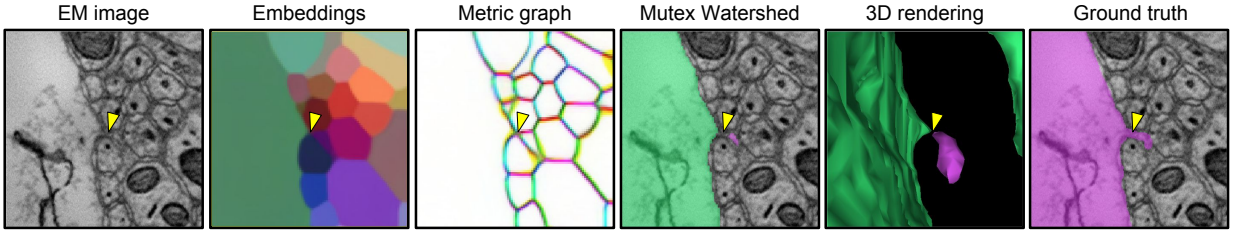

Fig. S17. A tiny broken spine in the validation set (AC3). The embedding net assigns completely distinct vectors to the dendritic shaft and the tiny spine (yellow arrowheads).