Supplementary Notes

S1. SNEMI3D Experiments

We trained the embedding net strictly under the SNEMI3D challenge setup, and segmented the test volume with our proposed method. We submitted our segmentation to the challenge leaderboard, allowing for objective comparison with other methods.

In all submissions, we used the same set of postprocessing parameters as in the main text, except for $\theta_{\text{mask}} = 0.5$. For both training and inference, we used $2\times$ in-plane downsampled images. We used the top 75 sections of the SNEMI3D training volume for training, and the bottom 25 sections for validation. Since the embedding net produces an output patch smaller than its input patch, extra context is required at the edge of the volume. We used mirror-padding to extend the test volume to provide extra context for inference. Prior to submission, the segmentation was dilated in 2D until no background voxels remain, and then upsampled with nearest neighbor upsampling to recover its original size.

We first started with the same network architecture as in the main text. The result was evaluated with an adapted Rand error of 0.0319, which outperformed the two recently published methods [1], [2], but fell behind the two topmost entries [3], [4] (see Table S1).

Qualitative inspection of our initial segmentation revealed two major problems. First, we found several merge errors near the edge of the volume, mainly caused by the systematic degradation of the embeddings and affinities near the volume edge. This is presumably due to the use of mirror-padding, which creates a biologically implausible scene that is never seen during training. Second, we found a different set of merge errors that seem to occur by generalization failure. This may have happened because the amount of training data (78.6 million voxels at the original resolution) is far less than that used in the main text (226.5 million voxels at the original resolution).

To address these issues, we explored two variations in the network architecture: the use of (1) *Group Normalization* (GN, [5]) instead of *Instance Normalization* (IN, [6]), and (2) smaller training/inference patches. GN is known to act as a strong regularizer and help achieving better generalization performance than IN [5]. We also hypothesized that using smaller *training* patches may benefit generalization, because it increases the effective size of the training set, decreases correlation between patches, and reduces the complexity of the input scene. Additionally, using smaller *inference* patches may make inference more robust to the biologically implausible scene created by mirror-padding, because smaller context may make local statistics more similar to that of the real image.

As can be seen in Table S1, using GN and smaller patch size ($64 \times 64 \times 20$) reduced the error from 0.0319 to 0.0265 and 0.0262 respectively, both outperforming the flood-filling net (FFN, [3]) and the mean affinity agglomeration (MAA) result of [4]. Combining both further reduced the error down to 0.0256, which is only marginally higher than the leading entry's error of 0.0249 [4].

Qualitative comparison of our best submission and the SNEMI3D's leading entry [4] revealed that ours has a substantially less number of split errors on thin neurites (it reduced 15 dendritic spine splits and 14 axon splits), but also has a couple of more merge errors. This suggests that such split errors may have been under reflected in the Rand error because of its overdependence on the size of erroneous objects. In other words, a small number of bigger merge errors could easily overwhelm the effect of a large number of smaller split errors.

S2. Notes on postprocessing parameters

A. θ_{mask}

The θ_{mask} parameter is a threshold for the real-valued background mask predicted by the embedding net to produce a binary background mask, which is used for excluding background voxels (nodes) and their incident affinities (edges) from the metric graph. θ_{mask} effectively controls the level of how conservatively we restrict the metric graph to foreground voxels by removing the potentially noisy embeddings and affinities near the background. When too liberal, mergers can occur through the uncertainty near the background. When too conservative, connectivity of the thin neurites may be altered.

To measure sensitivity of our system to varying θ_{mask}, we performed 3-fold cross-validation. To construct three folds, we split AC3/AC4 into three same-sized volumes: AC4, the top 100 slices of AC3, and the bottom 100 slices of AC3. Besides the three folds, we used the middle 56 slices of AC3 as an extra training set in all cases.

Table S2 summarizes the 3-fold cross-validation result for θ_{mask}. For each value of θ_{mask} on each fold, we optimized θ_d to obtain the best VI. Overall, θ_{mask} around 0.5 was found to be optimal, and segmentation accuracy measured by VI tends to decrease as θ_{mask} gets farther away from this optimal value.

B. θ_d

We performed a similar sensitivity analysis for θ_d with 3-fold cross-validation. Table S3 summarizes the 3-fold cross-validation result for θ_d, with θ_{mask} fixed to 0.5. Overall, the optimal range for θ_d was $\theta_d \in [1.0, 1.5)$. We found that the optimal value for θ_d on fold 2 is unusual and deviates from the overall optimal range. This was caused by a complication of merge errors. Although a *local* decision to agglomerate two segments is correct, *global* segmentation accuracy could get worsen if either of the segments involves a merge error. In other words, agglomerating two segments may correct a split error while amplifying an existing merge error. This property is generally relevant to any kind of error correction, and should

TABLE S1
SNEMI3D RESULTS

Group Name	Method	Group Norm [5]	Patch Size	Rand Error↓
PNI	Watershed [7] + TTA (16×) [4], [8]			0.0249
Ours	Metric Learning + MWS [9] + MEA	✓	$64 \times 64 \times 20$	**0.0256**
Ours	Metric Learning + MWS [9] + MEA		$64 \times 64 \times 20$	**0.0262**
Ours	Metric Learning + MWS [9] + MEA	✓	$128 \times 128 \times 20$	**0.0265**
GAIP	Flood-Filling Networks [3], [10]			0.0291
PNI	Watershed [7] + MAA [4]			0.0314
Ours	Metric Learning + MWS [9] + MEA		$128 \times 128 \times 20$	**0.0319**
S&T	STRU-Net [2]			0.0351
Ours	Metric Learning + MWS [9]		$64 \times 64 \times 20$	**0.0379**
Ours	Metric Learning + MWS [9]	✓	$64 \times 64 \times 20$	**0.0386**
CCG	Cross-Classification Clustering [1]			0.0410
Ours	Metric Learning + MWS [9]	✓	$128 \times 128 \times 20$	**0.0454**
Ours	Metric Learning + MWS [9]		$128 \times 128 \times 20$	**0.0456**
Human				0.0600
DIVE	DeepEM3D [8]			0.0602
IAL	Lifted Multicut [11]			0.0656

↓ The lower, the better.
Abbreviations: Mean Affinity Agglomeration (MAA), Test-Time Augmentation (TTA), Mutex Watershed (MWS), Mean Embedding agglomeration (MEA)

TABLE S2
3-FOLD CROSS-VALIDATION FOR θ_{MASK}

θ_{mask}	Fold 1	Fold 2	Fold3	Mean ± S.E.
0.1	0.1243	0.1338	0.1159	0.1245 ± 0.0052
0.2	0.0746	0.1094	0.0818	0.0886 ± 0.0106
0.3	0.0601	0.1625	0.0984	0.1070 ± 0.0299
0.4	0.0566	**0.0628**	0.0775	0.0656 ± 0.0062
0.5	0.0416	0.0681	**0.0763**	**0.0620** ± 0.0105
0.6	**0.0409**	0.0689	0.0781	0.0626 ± 0.0112
0.7	0.0469	0.1345	0.0877	0.0897 ± 0.0253
0.8	0.0465	0.0651	0.0882	0.0666 ± 0.0121
0.9	0.0680	0.0847	0.0907	0.0811 ± 0.0068

TABLE S3
3-FOLD CROSS-VALIDATION FOR θ_d WITH $\theta_{\text{MASK}} = 0.5$

θ_{mask}	Fold 1	Fold 2	Fold3	Mean ± S.E.
0.50	0.0586	**0.0681**	0.0885	0.0718 ± 0.0088
0.75	0.0558	**0.0681**	**0.0763**	0.0668 ± 0.0060
1.00	**0.0416**	0.0729	**0.0763**	**0.0636** ± 0.0110
1.25	**0.0416**	0.0729	**0.0763**	**0.0636** ± 0.0110
1.50	**0.0416**	0.0728	0.0778	0.0641 ± 0.0113
1.75	0.0465	0.0728	0.0778	0.0657 ± 0.0097
2.00	0.0465	0.0728	0.0778	0.0657 ± 0.0097
2.25	0.0502	0.0728	0.0778	0.0670 ± 0.0085
2.50	0.0502	0.0728	0.0778	0.0670 ± 0.0085

be considered carefully when designing a downstream system for human proofreading.

S3. INFERENCE BENCHMARK

Table S4 shows the decomposition of runtime for processing a $512 \times 512 \times 100$ volume at the voxel resolution of $12 \times 12 \times 29$ nm^3, averaged over six volumes. The benchmark was performed on a workstation equipped with Intel® Core™ i7-9700K CPU @ 3.60GHz and Nvidia GeForce RTX 2080 Ti. As can be seen in Table S4, the embedding net inference and the Mutex Watershed collectively dominate the total runtime (97.61% on average).

TABLE S4
INFERENCE TIME DECOMPOSITION

	Runtime (s)	Fraction (%)
Embedding net inference	189.84	53.09
Mutex Watershed	159.19	44.52
Self-contact split detection	6.47	1.81
Mean embedding agglomeration	2.05	0.57
Total	357.55	100.00

REFERENCES

[1] Y. Meirovitch, L. Mi, H. Saribekyan, A. Matveev, D. Rolnick, and N. Shavit, "Cross-classification clustering: An efficient multi-object tracking technique for 3-d instance segmentation in connectomics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 8425–8435.

[2] F. Gonda, D. Wei, and H. Pfister, "Consistent recurrent neural networks for 3d neuron segmentation," in *Proc. IEEE Int. Symp. Biomed. Imaging (ISBI)*, 2021.

[3] M. Januszewski *et al.*, "High-precision automated reconstruction of neurons with flood-filling networks," *Nat. Methods*, vol. 15, no. 8, pp. 605–610, 2018.

[4] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung, "Superhuman accuracy on the SNEMI3D connectomics challenge," *CoRR*, vol. abs/1706.00120, 2017.

[5] Y. Wu and K. He, "Group normalization," in *Comput. Vis. ECCV 2018*, 2018, pp. 3–19.

[6] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.

[7] A. Zlateski and H. S. Seung, "Image segmentation by size-dependent single linkage clustering of a watershed basin graph," *CoRR*, vol. abs/1505.00249, 2015.

[8] T. Zeng, B. Wu, and S. Ji, "DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation," *Bioinformatics*, vol. 33, no. 16, pp. 2555–2562, 2017.

[9] S. Wolf *et al.*, "The mutex watershed: Efficient, parameter-free image partitioning," in *Comput. Vis. ECCV 2018*, 2018, pp. 571–587.

[10] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain, "Flood-filling networks," *CoRR*, vol. abs/1611.00421, 2016.

[11] T. Beier *et al.*, "Multicut brings automated neurite segmentation closer to human performance," *Nat. Methods*, vol. 14, pp. 101–102, 2017.