Supplementary Information for "An Error Detection and Correction Framework for Connectomics" by Zung et al. (*NIPS 2017*)

A Baseline neuronal boundary detection

In this section, we describe our baseline segmentation pipeline, which is similar to what is described in [8]. The major difference is our novel densely multiscale 3D convolutional network architecture for neuronal boundary detection, which is described below. (The same *class* of architecture was employed in error detection and error correction. See main text.)

A.1 Network architecture

Our proposed densely multiscale 3D convolutional network for neuronal boundary detection is illustrated in Figure S1. Our model is built upon U-Net [12] with several interesting architectural augmentation. Our model can be viewed as a pyramidal stack of the basic computational module (diamond-shaped box in Figure S1). This diamond-shaped module can be interpreted as a residual building block (see Figure 2 in [5]) with two residual pathways, one top-down and the other bottom-up. Thus our model is *fully residual* in the sense that every computational pathway involving horizontal information flow is passing through the residual modules. Moreover, every residual module refines its input representation by integrating both top-down and bottom-up information, thus allowing for *dense* intermixing of multiscale features. Our model's *dense* and *fully residual* architecture allows an incremental and iterative top-down/bottom-up refinement of internal representation, which is in contrast to U-Net and variants' more restricted coarse-to-fine top-down refinement [12, 11, 9].

From a different point of view, Figure S2 illustrates another important motivation for our densely multiscale convolutional net architecture. Our model can be viewed as a feedback recurrent convolutional network unrolled in time (Figure S2). Weight sharing across time makes our model exactly equivalent to a convolutional net with recurrent feedback connections unfolded through time, and this novel perspective provides a better framework for understanding one of the unique characteristics

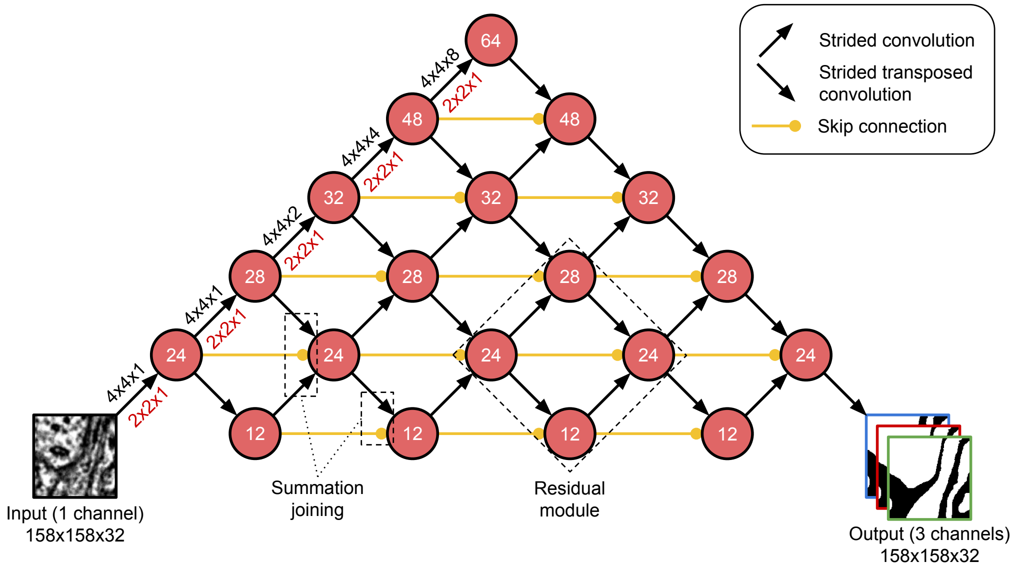

Figure S1: Architecture for the baseline neuronal boundary detection. Each node represents a layer and the number inside represents the number of feature maps. The layers closer to the top of the diagram have lower resolution than the layers near the bottom. The diagonal arrows represent strided convolutions, while the horizontal arrows represent skip connections. Associated with the diagonal arrows, the numbers indicate filter size (black) and strides (red) in x, y, and z-dimension. The target for our boundary detection network is a 3D *affinity graph* [13, 8, 3], thus outputting three channels corresponding to x (green), y (red), and z (blue) affinity map, respectively.

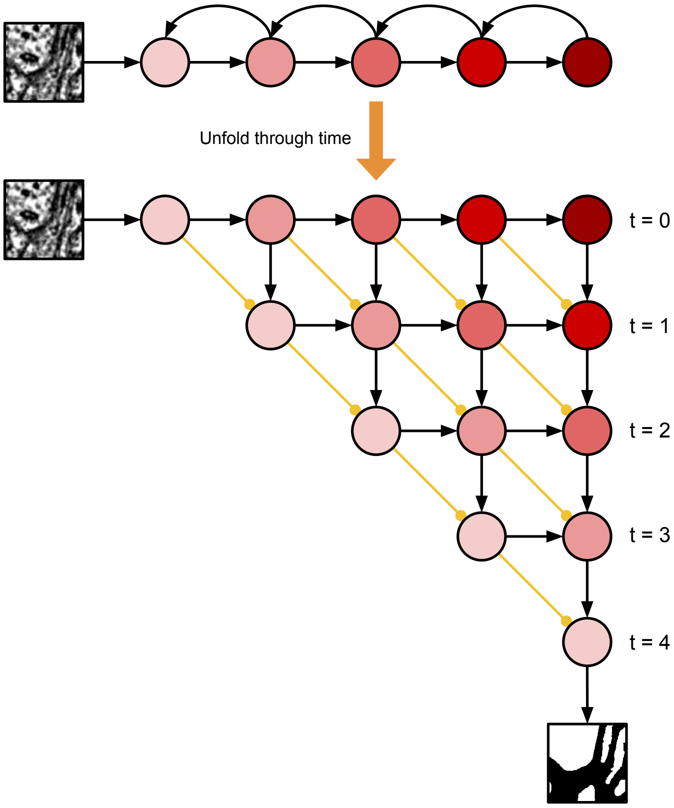

Figure S2: Feedback recurrent convolutional network unrolled in time (see Section A.1 for details).

of our model, i.e., the incremental refinement of internal representation by interative integration of top-down and bottom-up information. Our net's internal representation is incremetally and iteratively refined over time by integrating the top-down contextual information conveyed through the feedback recurrent connnections and the higer spatial-frequency information relayed through the bottom-up feedforward connections.

Note that we did not use weight sharing in the neuronal boundary detection net, whereas we used weight sharing between some convolutions at the same height in the error-detecting and error-correcting nets (Figure 2 and Figure S3).

Architectural details Due to the anisotropy of the resolution of the images in our dataset, we design our networks such that the first convolutions are exclusively 2D while later convolutions are 3D (Figure S1). The receptive field of a neuron in the higher layers is therefore nearly isotropic and roughly cubic in physical size. To limit the number of parameters in our model, we factorized all 3D convolutions into a 2D convolution followed by a 1D convolution in z-dimension. We employed exponential linear units (ELUs, [2]) as nonlinearity, except for the output layer with logistic activation functions. We trained our nets to generate a 3D affinity graph [13, 8, 3], thus outputting three channels corresponding to x, y, and z-affinity map, respectively.

A.2 Dataset

Our dataset is a sample of mouse primary visual cortex (V1) acquired using transmission electron microscopy at the Allen Institute for Brain Science. The voxel resolution is 3.6 nm $\times$ 3.6 nm $\times$ 40 nm.

Human experts produced multiple volumes of gold standard dense reconstruction, in total 20 volumes of size $512 \times 512 \times 100$. We trained our boundary detector using 19 volumes and used the last volume for training validation. We applied the trained boundary detector on a new image volume of size $2048 \times 2048 \times 100$ to obtain a preliminary segmentation, which was then proofread by the experts to generate a bootstrapped ground truth volume. This volume was used to optimize the parameters for watershed and mean affinity agglomeration [8]. Finally, the optimized segmentation pipeline was applied to generate further bootstrapped ground truth for the error detection and correction tasks.

A.3 Training procedures

Our boundary detection networks were implemented in the Caffe deep learning framework [6]. To train our nets, we minimized the pixelwise binomial cross-entropy loss with class-rebalancing using the Adam optimizer [7], initialized with $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 0.01$. The network weights were initialized following He et al. [4]. The learning rate (or step size parameter α in the Adam optimizer) was halved when validation loss plateaued out, five times in total at 35K, 175K, 250K, 300K, and 480K training iterations. We used a single patch of size $158 \times 158 \times 32$ (i.e. minibatch of size 1) to compute gradients at each training iteration. Each training sample was augmented with rotations, reflections, warping, brightness and contrast perturbations, and simulated misalignments [8]. The training lasted for 800K iterations until convergence, which took about five days on a single NVIDIA Titan X Pascal GPU.

A.4 Inference and postprocessing

We performed *overlap-blending* inference followed by watershed and mean affinity agglomeration [8]. We refer the interested readers to [8] for further details.

B Per-object VI score

Recall that the variation of information between two segmentations may be computed as

$$\begin{aligned}
VI_{\text{split}} &= -\frac{1}{\sum_{i,j} r_{ij}} \sum_{i,j} r_{ij} \log\left(r_{ij}/p_i\right), \\
VI_{\text{merge}} &= -\frac{1}{\sum_{i,j} r_{ij}} \sum_{i,j} r_{ij} \log\left(r_{ij}/q_j\right), \\
p_i &= \sum_j r_{ij}, \\
q_j &= \sum_i r_{ij},
\end{aligned}$$

where r_{ij} is the number of voxels in common between the i^{th} segment of the ground truth segmentation and the j^{th} segment of the proposed segmentation [10].

We define the split and merge scores for ground truth segment i as

$$\begin{aligned}
VI_{\text{split}}(i) &= -\sum_j r_{ij}/p_i \log(r_{ij}/p_i), \\
VI_{\text{merge}}(i) &= -\sum_j r_{ij}/p_i \log(r_{ij}/q_j).
\end{aligned}$$

Both quantities have units of nats. $VI_{\text{split}}(i)$ is zero if and only if ground truth segment i is contained within a segment in the proposed segmentation, while $VI_{\text{merge}}(i)$ is zero if and only if ground truth segment i is the union of one or more segments in the proposed segmentation. The total score VI_{split} or VI_{merge} is a weighted sum of the per-object scores $VI_{\text{split}}(i)$, $VI_{\text{merge}}(i)$ respectively.

C Training details

The error-detecting and error-correcting networks were implemented in TensorFlow [1] and trained using 4 TitanX Pascal GPUs with synchronous gradient descent. We used the Adam optimizer, initialized with $\alpha = 0.001$, $\beta_1 = 0.95$, $\beta_2 = 0.9995$, and $\epsilon = 0.1$ [7]. Both nets were trained until the loss on a validation set plateaued. The error-detecting net was trained for 700K iterations (approximately one week), while the error-correcting net was trained for 1.7M iterations (approximately three weeks).

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *CoRR*, abs/1511.07289, 2015. URL `http://arxiv.org/abs/1511.07289`.

[3] Jan Funke, Fabian David Tschopp, William Grisaitis, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. A deep structured learning approach towards automating connectome reconstruction from 3d electron micrographs. *arXiv preprint arXiv:1709.02974*, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR*, abs/1502.01852, 2015. URL `http://arxiv.org/abs/1502.01852`.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, Jan 2017. URL `https://arxiv.org/abs/1412.6980`.

[8] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H. Sebastian Seung. Superhuman accuracy on the SNEMI3D connectomics challenge. *CoRR*, abs/1706.00120, 2017. URL `http://arxiv.org/abs/1706.00120`.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[10] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B. Chklovskii. Machine Learning of Hierarchical Clustering to Segment 2D and 3D Images. *PLOS ONE*, 8 (8):1–11, 08 2013. doi: 10.1371/journal.pone.0071715. URL `https://doi.org/10.1371/journal.pone.0071715`.

[11] Pedro H. O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to Refine Object Segments. *CoRR*, abs/1603.08695, 2016. URL `http://arxiv.org/abs/1603.08695`.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, May 2015. URL `https://arxiv.org/abs/1505.04597`.

[13] 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., Feb 2010. URL `https://www.ncbi.nlm.nih.gov/pubmed/19922289`.

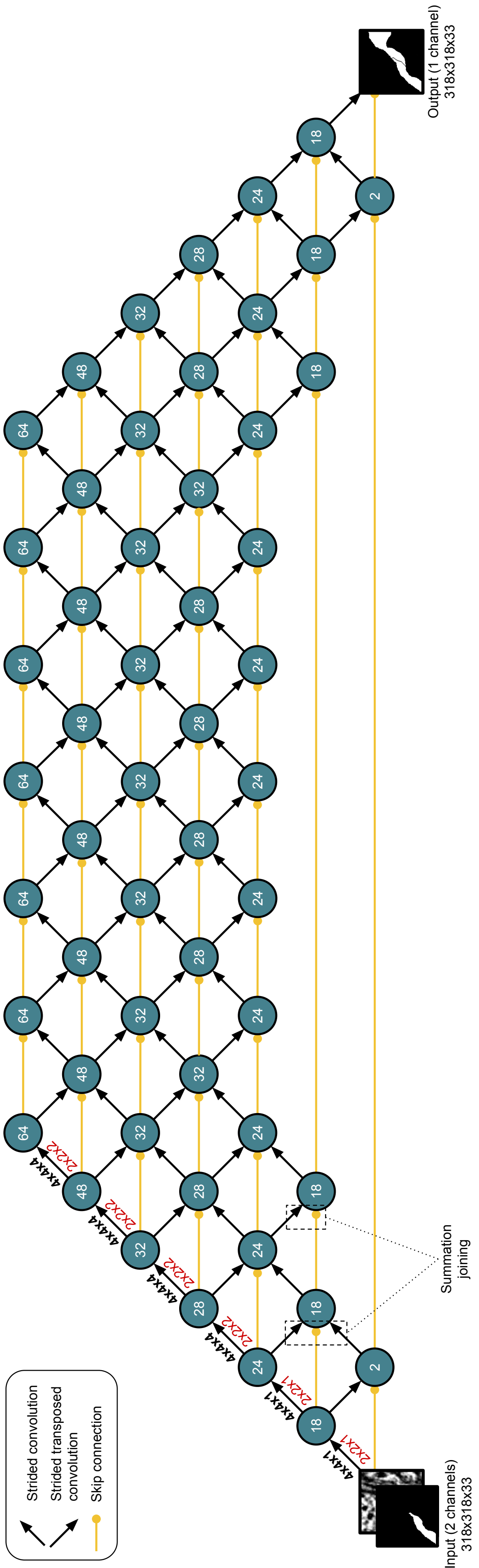

Figure S3: Error-correcting network architecture.