Guided Proofreading of Automatic Segmentations in Connectomics

Anonymous CVPR submission

Paper ID ****

Abstract

Automatic cell image segmentation methods in connectomics can lead to split *and* merge *errors, which require correction through proofreading. To aid error correction, we develop two classifiers that are able to recommend candidate errors and their corrections to the user. These classifiers are informed by training a convolutional neural network with known errors in automatic segmentations by comparison to expert-labeled ground truth. Our network architecture is able to detect potentially erroneous regions by considering a large context region around a segmentation boundary. With recommendations, proofreading of mouse cortex electron microscopy image segmentations can reduce VI scores on two different datasets from 0.4764 to 0.3996 and from XX to XX, which we find is an improvement on pure manual and pure automatic cases.*

1. Introduction

In connectomics, neuroanatomists build 3D reconstructions of neurons and their connectivity to gain insight into the functional structure of the brain. Rapid progress in automatic sample preparation and electron microscopy (EM) acquisition techniques has made it possible to image large volumes of brain tissue at $\approx 6\,nm$ per pixel to identify cells, synapses, and vesicles. For $25\,nm$ thick sections, a $1\,mm^3$ volume of brain contains 10^{15} voxels, or 1 petabyte of data: manual annotation is infeasible, and automatic methods are needed [14, 22, 25, 19].

Automatic segmentation and classification of brain tissue is challenging [1], so learning-based methods are common. The state of the art uses supervised learning with convolutional neural networks [11], or potentially even using unsupervised learning [10]. Typically, cell membranes are detected in 2D images and the resulting region segmentation is grouped into geometrically-consistent cells across registered sections, or cells are segmented across registered sections in 3D directly. Using dynamic programming techniques [23], and a GPU cluster, these classifiers can segment ≈ 1 terabyte of data per hour [17], which is sufficient to keep up with the 2D data capture process on state-of-the-art electron microscopes (though 3D registration is still an expensive offline operation).

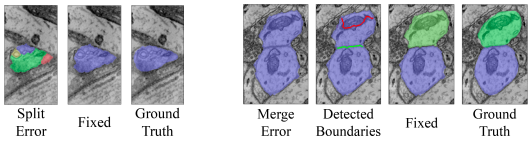


Figure 1: Split and merge error examples, their corrections, and their ground truths.

All automatic methods make errors, and we are left with large data which needs *proofreading* by humans. This crucial task serves two purposes: 1) to correct errors in the segmentation, and 2) to provide large corpora of labeled data to train better automatic segmentation methods. Recent proofreading tools provide intuitive user interfaces to browse segmentation data in 2D and 3D and to identify and manually correct errors [30, 15, 20, 13, 16, 31]. Many kinds of errors exist, such as inaccurate boundaries, but the most common are *split errors*, where a single cell is labeled as two, and *merge errors*, where two cells are labeled as one (Fig. 1). With user interaction, split errors can be joined, and the missing boundary in a merge error can be defined with manually-seeded watersheds [13]. However, even with semi-automatic correction tools, the visual inspection of the data to find the errors in the first place takes the majority of the time.

Our goal is to add automatic detection of split and merge errors to proofreading tools. Instead of the user visually inspecting the whole data volume carefully to spot errors, we design automatic classifiers that detect split and merge errors in 2D segmentations. Then, a proofreading tool can recommend regions with a high probability of an error to the user, and suggest corrections to accept or reject.

The initial automatic segmentation is constrained by the data rate of the microscope. Given a membrane segmentation from a fast automatic method, our classifiers operate on whole cell regions, which relaxes the constraint on speed: compared to techniques that must analyze every input pixel, this boundary assessment focus reduces the data

analysis to the boundaries only, and so allows us to employ wider convolutional neural networks that take regional context and multiple input channels into account. One reason to classify errors on 2D images lies with the cost of 3D registration. This is often slow as it requires non-linear image alignment [4, 29]. However, typically segmentation results are local decisions at the cell level. In this case, 3D reconstruction is unnecessary and, instead of waiting for the 3D output, proofreading can start immediately to maximize error correction before cell grouping occurs across sections.

We quantitatively validate our approach with variation of information (VI) versus ground truth expert segmentations. We compare our approach in an experiment against an existing proofreading tool that provides only semi-automatic merge error correction [13]. Here, with a simulated user, our automatic error suggestions and corrections decrease VI from 0.476 to 0.426, which is in contrast to pure manual or pure automatic methods that can both increase the VI. As a consequence, we are able to provide tools to proofread segmentations more efficiently, and so better tackle large volumes of connectomics imagery.

2. Related Work

Automatic Segmentation Multi-terabyte EM brain volumes require automatic segmentation [14, 18, 22, 24, 25, 32], but this data can be hard to classify in cases with ambiguous intercellular space.

The 2013 IEEE ISBI neurites 3D segmentation challenge [1] showed that existing algorithms which learn from expert-segmented training data still exhibit high error rates. NeuroProof [2] allows interactive learning of agglomeration of over-segmentations of images, based on a random forest classifier. Vazquez-Reina et al. [32] propose automatic 3D segmentation by taking whole EM volumes into account rather than a per section approach, then solving a fusion problem with a global context. Kaynig et al. [18] propose a random forest classifier coupled with an anisotropic smoothing prior in a conditional random field framework and 3D segment fusion.

It is also possible to learn segmentation classification features directly from images with CNNs. Ronneberger et al. [27] use a contracting/expanding CNN path architecture to enable precise boundary localization with small amounts of training data. Lee et al. [21] recursively train very deep networks with 2D and 3D filters to detect boundaries.

While these approaches make good progress, in general proofreading is required to improve them through generating more ground-truth segmentations.

Arganda-Carreras et al. [7] posed the ISBI 2D EM segmentation challenge in 2012, where a 30-image corpus of fly cell 'in/out' labels was used to train boundary detection. However, mouse brain EM data is more difficult than the ISBI 2012 challenge, as it contains large intercellular space which is hard to classify.

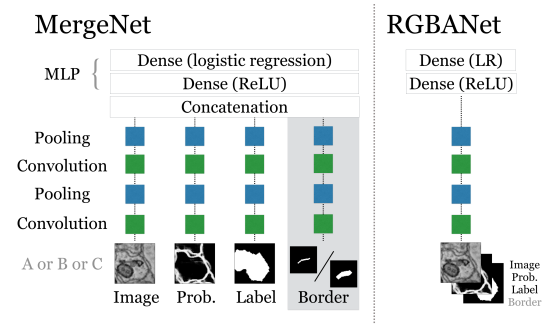
Bogovic et al. [10] learn 3D features, and show even that unsupervised learning can produce better features than hand-designs. In this paper, we extend the features reported by Bogovic et al. but limit the classification to 2D images rather than 3D. One major reason for attempting to classify errors on 2D images is with the scale of the task. 3D reconstruction pipelines are often slow as they require non-linear image alignment or registration [4, 9, 29]. However, typically segmentation results are local decisions at the cell level. In this case, reconstruction is unnecessary and, instead of waiting for the 3D output, proofreading can start immediately to maximize throughput.

Collaborative Interactive Segmentation Recent works attack the problem of massive volume segmentation through crowd-sourcing[28, 6]. EyeWire [3] asks novice users to participate in a segmentation game for segmenting neuronal structures using a semi-automatic algorithm. D2P [12] uses a micro-labor workforce approach where local boolean decisions are combined to produce a consensus segmentation. In general, our goal is to correct the output of a segmentation which is thought to be good; hence, our tool would be used after learning a segmentation model to direct user attention to correct likely erroneous areas.

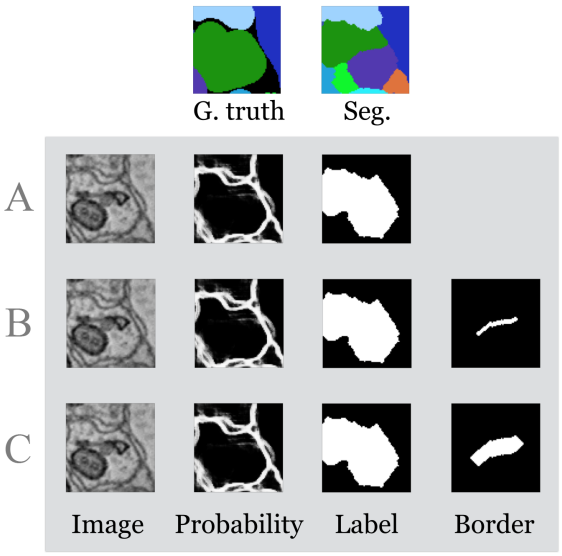
Proofreading Tools The bottleneck of editing an existing and imperfect automatic segmentation was identified by Peng et al. [26]. The authors developed software which supports three-dimensional proofreading. Also for 3D, Sicat et al. proposed a graph abstraction method to guide users to problematic regions indicating the need for corrections [30]. Janelia Farms then introduced Raveler [15], a software targeting expert users by offering many parameters for tweaking the proofreading process at the cost of a higher complexity. Raveler also includes a simple 3D renderer to validate corrections across slices.

In 2014, Haehn et al. developed two proofreading tools: Mojo and Dojo. Mojo is a stand-alone proofreading tool with a simple scribble interface for error correction. The need for distributed proofreading lead us to develop Dojo, a web-based proofreading framework with a minimalistic user interface. The authors defined requirements for proofreading tools in general and then evaluated the accuracy and speed of Raveler, Mojo and Dojo through a quantitative user study (Sec. 3 and 4) [13]. Al-Awami et al. then integrated Dojo into their proofreading management system Neuroblocks [5]. The system includes a scalable visualization for tracking the proofreading process among multiple users.

In 2015, Karimov et al. proposed a guided volume editing approach using histogram dissimilarity [16]. Measuring differences in histogram distributions helps their sys-



(a) CNN Layers



(b) Network configurations

Figure 2: (a) Our network architecture with up to four input channels, each with two convolutional and two pooling layers. MergeNet includes four separate input channels and RGBANet uses a 4-channel input. (b) We trained three different network configurations with three and four inputs: A) image, boundary map probability, and merged binary mask; B) A configuration extended with a small border mask, to focus on the specific boundary in question; C) A configuration extended with a large border mask.

tem to find potential errors and to suggest possible corrections. Their method shows promising results on Computed-Tomography Angiography datasets but is targeted towards expert users.

Recently, Uzunbas et al. showed that potential labeling errors can be found by taking the merge tree of an automatic segmentation method into account [31]. The authors keep track of uncertainties during the automatic labeling and then present them as regions for proofreading. This requires access to the merge tree of the segmentation stage which is not always available.

3. Method

We build a split error classifier using a convolutional neural network (CNN) to check the boundaries of an existing automatic segmentation. For each boundary, the CNN provides a probability that points sampled along the boundary caused a split error. For each boundary, we sample up to 10 decision points, where the decision points are spread evenly over the boundary length, so long as their context windows do not overlap. These probabilities are then weighted by the length of the boundary within the context over the total boundary length, and averaged. A greedy algorithm then merges neighboring regions sequentially, starting with the highest probability score. Following each merge, neighboring boundaries are re-evaluated for split errors. Correcting a split error is as simple as merging the two bordering labels.

Identification and correction of merge errors is more challenging, because we must look inside segmentation regions for missing or incomplete boundaries and then propose the correct boundary. However, we can reuse the same trained CNN for this task. For each segmentation label, we generate 30 potential boundaries through the region by placing watershed seed points at opposite sides of the label boundary and generating the corresponding split. Then, we check to see whether any potential edge is classified as a split error. If the CNN detects a boundary with a very low split error score, then the boundary should have been in the segmentation and the region is a candidate for a merge error.

3.1. Convolutional Neural Network Design

To train a CNN for split error detection we take multiple channels of context information of the boundary into consideration for the decision making process. We pass multiple inputs into the CNN windowed around a particular decision point or pixel: the input grayscale image patch, the corresponding boundary probability map patch, and two corresponding binary mask patches for the segmented regions at either side of the boundary. Following Bogovic et al. [10], these two masks can be combined into a single mask with comparable performance (configuration A, Fig. 5b). The network then leverages these multiple input patches to identify and correct errors made by the previous membrane detection network and automatic segmentation pipeline.

One way to combine these inputs is to treat them as a 4-channel input, so that alignment between the input image and the segmentation masks are not lost throughout the convolutions. We refer to this approach as *RGBANet*. However, training a boundary-classifying network can be difficult due to rigid ground-truth segmentations, which often differ substantially from automatic segmentation regions in ambiguous extra-cellular space. To cope with this variation, our network is based on multiple separate input channels

(Fig. 5a). Each of the input patches is connected individually to a 2-layer network, with each layer consisting of convolutional and pooling layers. The output of these networks is then combined by a fully connected multi-layer perceptron (MLP) with one hidden layer and a two class logistic regression output layer. The intuition for this multiple input channel approach is that we want to allow variation in the input and masks independently, to accommodate potential error, and then for the hidden layers to discover appropriate combinations of the relevant features learned separately for the different input channels. We refer to this approach as *MergeNet*.

To better direct both networks to train on the true boundary edge, which in many cases is missing from the boundary probability map and hence is the cause of merge errors, we additionally pass as input a second binary mask. This mask contains the true boundary edge (configuration B, Fig. 5b). To consider slight edge ambiguities, we also test a version of this network where the true boundary mask has been dilated by 5 pixels (configuration C).

3.2. Training

To train the network, we use the blue 3-cylinder mouse cortex volume of Kasthuri et al. [17] (2048 x 2048 x 300 voxels). The tissue is dense mammalian neuropil from layers 4 and 5 of the S1 primary somatosensory cortex of a healthy mouse. The resolution of our data set is $3\,nm$ per pixel, and the section thickness is $30\,nm$. A manually-labeled expert segmentation is available as a ground truth for the entire data set. We use the first 250 sections of the data for training and validation (split 0.25) and the last 50 for testing. To generate training data, we identify correct regions and split errors in the automatic segmentation by intersecting with the ground truth regions. From these regions, we sample 266,088 correct regions and 266,088 split error patches. As patch size, we defined $75x75$ pixels to cover $\approx 80\%$ of all boundaries in our segmentation output.

We train our networks using the following parameters: learning rate $lr = 0.03$ (iteratively decreasing until $lr = 0.00001$), momentum $m = 0.9$ (iteratively increasing until $m = 0.999$), filter size $fs = 13 \times 13$ and number of filters $fn = 16$ for MergeNet, and number of filters $fn_1 = 64, fn_2 = 48$ for RGBANet. This results in 1.5 million parameters for all network configurations. For regularization we use dropout layers after each pooling layer with $p = 0.2$. We assume that the training has converged if the validation loss does not decrease for 50 epochs. The network is specified using the deep learning libraries Lasagne and Theano [8], and trained on a Tesla K40m graphics card.

Figure 3 presents validation loss function scores (cross validation), test accuracy percent as well as precision/recall and F1 scores. Based on these performances, we select configuration MergeNet C to evaluate against human performance in a VI improvement experiment.

4. Evaluation

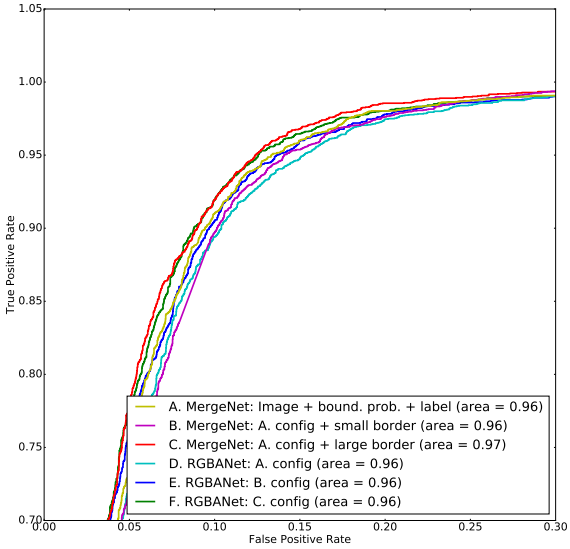
We evaluate our split and merge error detection and correction recommendation in the context of interactive proofreading tools: to direct users to regions with a high probability of error and to suggest corrections (Fig. 4). For comparison, we take publicly available mouse cortex data of the same kind as our training data. We perform two experiments: first, we compare our approach with previously reported proofreading results using real-world measurements and second, we evaluate guided proofreading in a simulated context with a much larger dataset.

4.1. Comparison Study

For our first experiment, our data is part of the ISBI 2013 challenge training dataset ($1024 \times 1024 \times 100$ voxels) which was acquired using a serial section scanning electron microscope (ssSEM) with a resolution of $6 \times 6 \times 30\,nm$ per voxel. We use the available manually-labeled ground truth to score our approach using the variation of information (VI) metric, which is closely related to mutual information. VI is a measure of the distance between two clusterings, where lower VI numbers are better. Since our classifiers are trained on 2D image slices, we perform all evaluations on slices rather than 3D volumes.

Guided proofreading. Recently, Haehn et al. discussed requirements for interactive proofreading and evaluated three different tools on connectomics data in a study with naive users [13]. This study asked users to spend 30 minutes proofreading with the different tools, to correct split and merge errors to improve the automatic segmentation. The best performing tool in their evaluation was Dojo. We use their findings and their user-generated proofreading result data, which they kindly provided, as a baseline for the evaluation of our method. Haehn et al. perform their user study on the most representative sub-volume ($400 \times 400 \times 10$ voxels) in terms of distribution of object size. For optimal comparison, we use exactly the same data and time constraints. We asked two experts to perform the proofreading task using our system (Fig. 6).

In addition, we simulate a user for proofreading correction. We assume that all classification has been computed ahead of time, and that the user is presented with a stream of error corrections to assess. The assessment is simulated by comparing the VI before and after each performed correction. Corrections are accepted only when VI reduces, and we test this across different user error rates (Fig. 4). In Haehn et al., the proofreading time was limited to 30 minutes, and human participants performed 59 corrections on average ($\approx$ 30 seconds per correction). In our scenario,



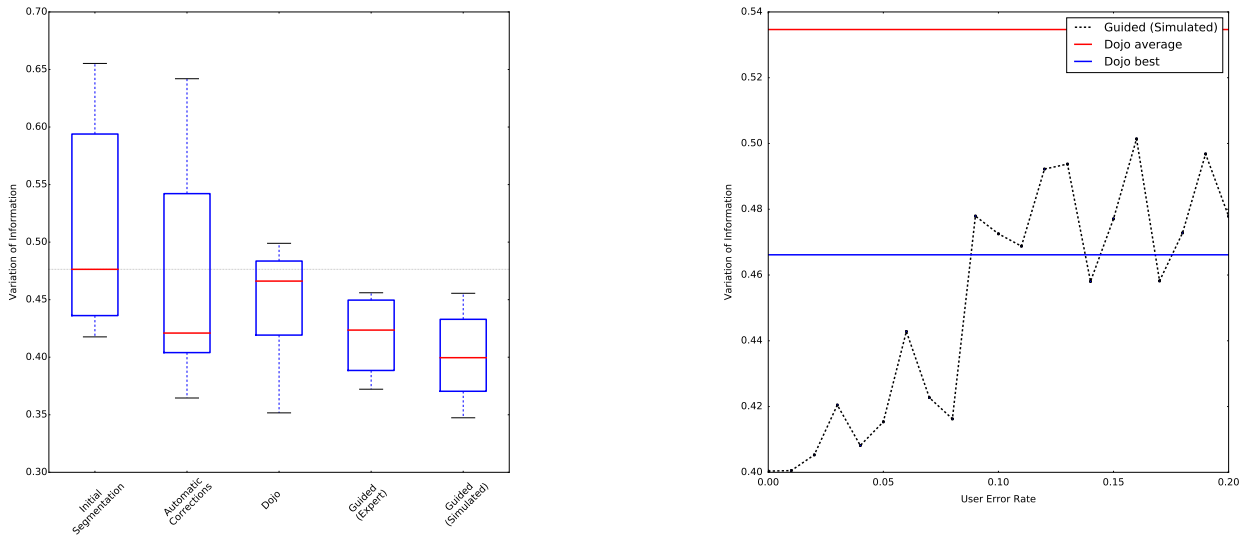
Network	Validation loss	Test acc. (%)	Prec./Recall	F1 Score
A. MergeNet: Image + boundary prob. + seg. label	0.073	0.906	0.906/0.906	0.907
B. MergeNet: A config. + small border overlap ($d = 1$)	0.076	0.905	0.907/0.905	0.908
C. MergeNet: A config. + large border overlap ($d = 5$)	0.07	0.908	0.908/0.908	0.909
D. RGBANet: A. config.	0.058	0.895	0.895/0.895	0.894
E. RGBANet: B. config.	0.054	0.907	0.907/0.907	0.908
F. RGBANet: C. config.	0.058	0.905	0.905/0.905	0.904

Figure 3: Network design training evaluation. Adding an extra channel containing a binary mask of just the border slightly increases performance in both network configurations. We take configuration MergeNet C to evaluate VI against human performance.

users do not need to visually find errors and manually correct them, and from real-world user performance we observed an average decision time of ≈ 3.2 seconds. For our simulated user, we assume each correction assessment takes 5 seconds (360 assessments in 30 minutes). Split errors are likely to take less time than this; however, merge errors are harder to assess, as the user must select between the top 5 candidate boundaries. Since the performance between human participants of Haehn et al.'s user study shows large variation within the average performance among all users (VI improvement: -0.0582), we present the best performing Dojo user (VI improvement: 0.0102) as our baseline. The average VI improvement of two experts using our system is 0.0528. For our simulated user, the VI improvement is 0.0768 (Fig. 4). In total, our classifier predicted 18 merge and 842 split errors in the data.

Random recommendations. We decided to test a classifier with random performance in comparison to our learned CNN. For split errors, the simulated user is presented with randomly picked boundaries, which they can accept or reject. For merge errors, the simulated user is presented with 5 randomly selected boundaries from the interior of the segmented region. Around 80% of all presented regions did not need to be corrected. Hence, the median VI did not decrease much (VI improvement: 0.0011). This significantly worse performance of this approach demonstrates that our network is informative to the user.

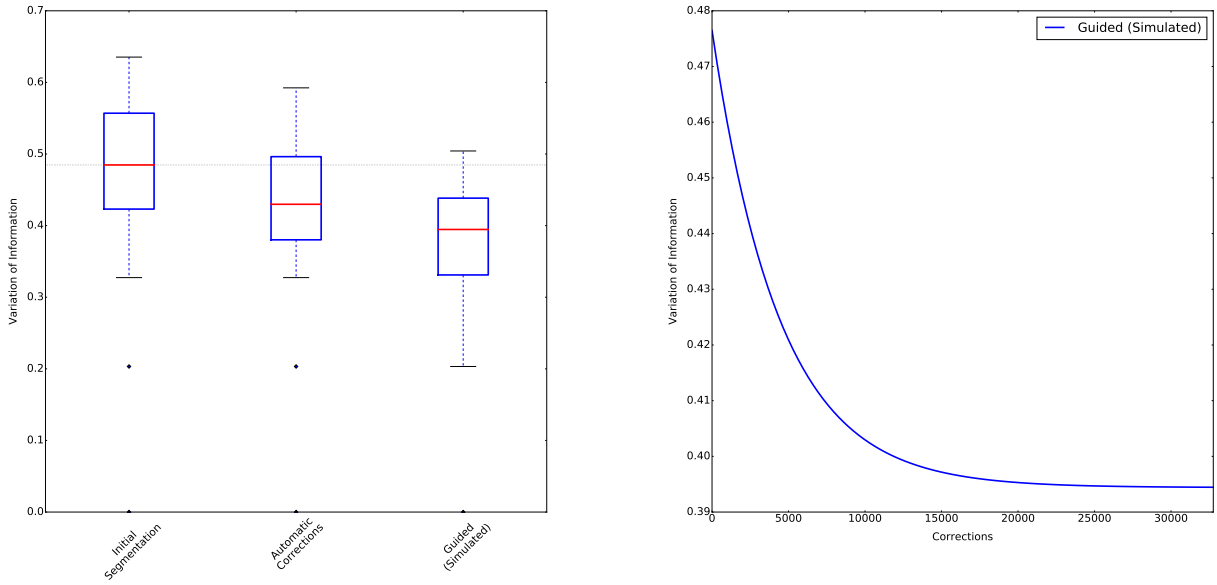
Automatic correction. As a comparison, we also perform automatic correction. During training, we define a probability threshold $p_t = 0.95$ for automatic split correction based on CNN probability from the test set. Then, for automatic correction, we apply both classifiers to produce lists of split and merge errors sorted by confidence. First, we correct merge errors with $\max(1 - p)$, followed by split error correction using p_t. The total time for correcting all errors was 17 minutes on a 3.2 GHz Quad-core Intel Xeon with an NVIDIA GeForce Titan (merge error correction 15min, split error correction 2min). The median VI improvement



(a) Interactive (Dojo) vs. Guided Proofreading

(b) Simulated User Error Rate

Figure 4: (a) We compare distributions of VI measures across 10 sections for the initial automatic segmentation, a fully-automatic correction of recommended errors based on a threshold of acceptance, the best user in the Haehn et al. experiment with Dojo, two experts using our system and our simulated user. Lower scores are better.



(a) Split error

(b) Merge error

Figure 5: Our web-based user interface includes a slice overview with the relevant area highlighted in yellow. The interface shows (a) a split error with a suggested correction as well as (b) a merge error with correction. The user selects whether to accept a correction or to skip it.

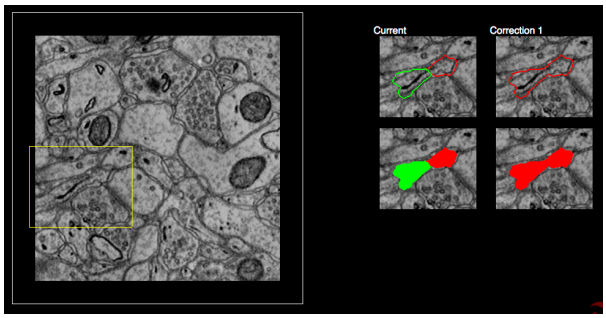
in comparison to the ground truth was 0.02 (Fig. 4). This is not surprising, as the problem is very challenging, and this motivates the need for human-in-the-loop proofreading tools.

4.2. Simulated Experiment

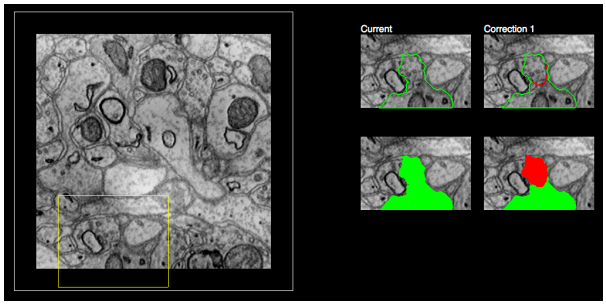
For our second experiment, we proofread 50 slices of the blue 3-cylinder cortex volume of Kasthuri et al. [17]. The data was not seen by the network before and includes 2048 x 2048 x 50 voxels with a total number of 33,076 labeled objects. Since interactive proofreading of such a large dataset would consume a significant amount of time, we restrict our experiment to a simulated user and to automatic corrections. Similar to our comparison study (see section 4.1), the simulated user assesses a stream of errors by comparing VI before and after each performed correction. As before, corrections are only accepted when VI reduces. In contrast to our time limit in the comparison study, the simulated user proofreads until all objects in the volume were assessed. For automatic corrections, we use our defined probability threshold $p_t = 0.95$. Based on a time budget of 5 seconds per correction, the proofreading process for a real user would in theory take over 45 hours for this data set. The total time for automatically correcting all errors was $\approx$ 6 hours on a 3.2 GHz Quad-core Intel Xeon with an NVIDIA Geforce Titan. The total time for the simulated user including the VI calculations after each assessment was $\approx$ 10 hours. Both approaches significantly reduce the VI in comparison to the initial segmentation by XX for the simulated user and by XX automatically. Results are shown in figure 5.

5. Application - Proof reading

In our experiments, we observed the best performance using a combination of user guidance and our trained network. In contrast to fully interactive proofreading tools like Dojo, Mojo and Raveler, our system requires only minimal user input. We distinguish between merge and split errors and provide a very simple user interface to correct these (see figure 6). The system shows only one potential error - either a false merge or a false split - in the interface. In the case of merge errors, the user is presented the highest scoring five possible boundaries as overlays on the corresponding grayscale image and also a possibility to draw a boundary interactively. The user then chooses one of the suggestions, draws a boundary or marks the cell as correct. For split errors, the system shows the grayscale image and a possible border and the user marks the cell as correct or indicates he wants to split. Our experiment baseline, the Dojo user study, was limited to 30 minutes and participants performed 59 corrections in average (30 seconds per correction). Our experiments have shown that even non-experts can perform a correction using our system in 5 seconds and thus, the system increases the proofreading performance.



(a) Split error



(b) Merge error

Figure 6: Our web-based user interface includes a slice overview with the relevant area highlighted in yellow. The interface shows (a) a split error with a suggested correction as well as (b) a merge error with correction. The user selects whether to accept a correction or to discard it.

6. Discussion and Conclusion

The task of automatic cell boundary segmentation is difficult, and trying to improve such segmentations automatically as a post-process through split and error correction is, in principle, no different than trying to improve the underlying cell boundary segmentation. This is shown by the approximately equivalent VI distributions of the initial segmentation and our automatic segmentation correction (Fig. 4). Due to the task difficulty, manual proofreading of connectomics segmentations is necessary, but it is a time consuming and error-prone task, as can be seen from the Dojo human trials: on average, participants actually made the segmentations worse. However, there is value in being able to recommend to users possible regions for correction, as the time cost of proofreading is dominated by the visual search for errors.

We have addressed this problem through training a CNN to detect ambiguous regions from labeled data—in effect, (re-)learning a confidence measure on boundaries. This allows us to recommend split and merge errors, and also to recommend their corrections, which is an improvement over

existing systems which just provide semi-automatic merge error correction. Through simulating users with different error rates, we have shown that, for an equivalent 30 minutes of work, correction recommendation has the potential to reduce VI over existing proofreading tools. This helps reduce the proofreading bottleneck to the analysis of large connectomics datasets. To encourage testing of our proposed architecture on more data, we provide the trained networks and classifier code as free and open source software at (link omitted for review).

References

[1] IEEE ISBI challenge: SNEMI3D - 3D segmentation of neurites in EM images. http://brainiac2.mit.edu/SNEMI3D, 2013. Accessed on 31/03/2016. 1, 2

[2] Neuroproof: Flyem tool, hhmi / janelia farm research campus. https://github.com/janelia-flyem/NeuroProof, 2013. Accessed on 03/15/2106. 2

[3] Eyewire. http://eyewire.org/, 2014. Accessed on 31/03/2014. 2

[4] A. Akselrod-Ballin, D. Bock, R. Reid, and S. Warfield. Improved registration for large electron microscopy images. In *IEEE Int. Symp. Biomedical Imaging (ISBI)*, 2009. 2

[5] A. K. Al-Awami, J. Beyer, D. Haehn, N. Kasthuri, J. W. Lichtman, H. Pfister, and M. Hadwiger. Neuroblocks - visual tracking of segmentation and proofreading for large connectomics projects. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):738–746, Jan 2016. 2

[6] J. Anderson, S. Mohammed, B. Grimm, B. Jones, P. Koshevoy, T. Tasdizen, R. Whitaker, and R. Marc. The Viking Viewer for connectomics: Scalable multi-user annotation and summarization of large volume data sets. *Journal of Microscopy*, 241(1):13–28, 2011. 2

[7] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. Pham, E. Bas, M. G. Uzunbas, A. Cardona, J. Schindelin, and H. S. Seung. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9(142), 2015. 2

[8] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 4

[9] J. Beyer, M. Hadwiger, A. Al-Awami, W.-K. Jeong, N. Kasthuri, J. W. Lichtman, and H. Pfister. Exploring the connectome: Petascale volume visualization of microscopy data streams. *IEEE Computer Graphics and Applications*, 33(4):50–61, 2013. 2

[10] J. A. Bogovic, G. B. Huang, and V. Jain. Learned versus hand-designed feature representations for 3d agglomeration. *CoRR*, abs/1312.6159, 2013. 1, 2, 3

[11] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, 2012. 1

[12] R. J. Giuly, K.-Y. Kim, and M. H. Ellisman. DP2: Distributed 3D image segmentation using micro-labor workforce. *Bioinformatics*, 29(10):1359–1360, 2013. 2

[13] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE SciVis 2014)*, 20(12):2466–2475, 2014. 1, 2, 4

[14] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstädter, K. Briggman, W. Denk, J. Bowden, J. Mendenhall, W. Abraham, K. Harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and S. Seung. Boundary learning by optimization with topological constraints. In *Proceedings of IEEE CVPR 2010*, pages 2488–2495, 2010. 1, 2

[15] Janelia Farm. Raveler. https://openwiki.janelia.org/wiki/display/flyem/Raveler, 2014. Accessed on 31/03/2016. 1, 2

[16] A. Karimov, G. Mistelbauer, T. Auzinger, and S. Bruckner. Guided volume editing based on histogram dissimilarity. *Computer Graphics Forum*, 34(3):91–100, May 2015. 1, 3

[17] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015. 1, 4, 7

[18] V. Kaynig, T. Fuchs, and J. Buhmann. Neuron geometry extraction by perceptual grouping in sstem images. In *Proceedings of IEEE CVPR*, pages 2902–2909, 2010. 2

[19] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *Medical image analysis*, 22(1):77–88, 2015. 1

[20] S. Knowles-Barley, M. Roberts, N. Kasthuri, D. Lee, H. Pfister, and J. W. Lichtman. Mojo 2.0: Connectome annotation tool. *Frontiers in Neuroinformatics*, (60), 2013. 1

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

[22] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen. A modular hierarchical approach to 3D electron microscopy image segmentation. *Journal of Neuroscience Methods*, 226(0):88 – 102, 2014. 1, 2

[23] J. Masci, A. Giusti, D. C. Ciresan, G. Fricout, and J. Schmidhuber. A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *ICIP*, 2013. 1

[24] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment 2D and 3D images. *PLoS ONE*, 8(8):e71715+, 2013. 2

[25] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz. Graph-based active learning of agglomeration (GALA): A python library to segment 2D and 3D neuroimages. *Frontiers in Neuroinformatics*, 8(34), 2014. 1, 2

[26] H. Peng, F. Long, T. Zhao, and E. Myers. Proof-editing is the bottleneck of 3D neuron reconstruction: The problem and solutions. *Neuroinformatics*, 9(2-3):103–105, 2011. 2

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 2

[28] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. CATMAID: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–1986, 2009. 2

[29] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets. *Bioinformatics*, 26(12):i57–i63, 2010. 2

[30] R. Sicat, M. Hadwiger, and N. J. Mitra. Graph abstraction for simplified proofreading of slice-based volume segmentation. In *EUROGRAPHICS Short Paper*, 2013. 1, 2

[31] M. G. Uzunbas, C. Chen, and D. Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical Image Analysis*, 27:31 – 44, 2016. Discrete Graphical Models in Biomedical Image Analysis. 1, 3

[32] A. Vázquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister. Segmentation fusion for connectomics. In *Proceedings of IEEE ICCV*, pages 177–184, Nov 2011. 2