

# ICON: AN INTERACTIVE APPROACH TO TRAIN DEEP NEURAL NETWORKS FOR SEGMENTATION OF NEURONAL STRUCTURES

F. Gonda\*, V. Kaynig\*, R. Thouis\*, D. Haehn\*, J. W. Lichtman†, T. Parag\*, H. Pfister\*

\* Harvard University, John A. Paulson School of Engineering and Applied Sciences, Cambridge, Massachusetts, USA. † Harvard University, Department of Molecular and Cellular Biology and Center for Brain Science, Cambridge, Massachusetts, USA.  
[fgonda, vkaynig, thouis, haehn, paragt, pfister]@g.harvard.edu, jeff@mcb.harvard.edu

## ABSTRACT

We present an interactive approach to train a deep neural network pixel classifier for the segmentation of neuronal structures. An interactive training scheme reduces the extremely tedious manual annotation task that is typically required for deep networks to perform well on image segmentation problems. Our proposed method employs a feedback loop that captures sparse annotations using a graphical user interface, trains a deep neural network based on recent and past annotations, and displays the prediction output to users in almost real-time. Our implementation of the algorithm also allows multiple users to provide annotations in parallel and receive feedback from the same classifier. Quick feedback on classifier performance in an interactive setting enables users to identify and label examples that are more important than others for segmentation purposes. Our experiments show that an interactively-trained pixel classifier produces better region segmentation results on Electron Microscopy (EM) images than those generated by a network of the same architecture trained offline on exhaustive ground-truth labels.

**Index Terms**— Segmentation, Interactive, Annotations, Neural Networks, Connectomics.

## 1. INTRODUCTION

Connectomics is an emerging discipline of neuroscience dedicated to the reconstruction of neural structures and connectivity from brain images. Electron Microscopy (EM) can provide extremely detailed images (at nanometer scale) of animal brain tissues for a comprehensive neural reconstruction. Recording at such high resolution generates a massive amount of data from a relatively small brain region; a  $1\text{mm}^3$  volume of rat cortex amounts to 33,333 images of size  $250,000 \times 25,000$  when imaged at  $4 \times 4 \times 30$  nm  $x, y, z$  resolution. Automated or semi-automated processing is the most viable strategy to process datasets at this scale.

Segmentation of neuron regions in EM volumes has become the central tool for many (semi-) automated neural reconstruction efforts [1, 2, 3]. Almost all these segmentation approaches utilize a pixel classifier to distinguish cell boundary (or membrane) pixels from cell interior (or other organelle) pixels. A set of annotated pixels is required to train such a classifier. Most of the existing segmentation algorithms [4, 5] require an exhaustive ground-truth volume, where each pixel (or voxel) within the volume is annotated by expert users for training. The type of annotation is generally a label for boundary pixels.

Exhaustive annotation of a sufficiently large volume demands significant time and effort from an expert in cell biology. Our experience suggests a  $1024 \times 1024 \times 250$  EM volume would require 6 ~ 8 weeks of dedicated labeling effort from a neurobiologist. Such a manual labeling step becomes a substantial bottleneck for the overall neural reconstruction process. This impediment is compounded for large datasets collected from multiple brain areas with biologically different cell characteristics and difference in tissue preparation techniques.

Several researchers in the EM segmentation community realized this issue and proposed different algorithms for training pixel classifiers from a relatively small (and often sparse) set of training examples. The interactive software Ilastik [6] for learning a Random Forest pixel detector has become very popular in the bioinformatics community. Rather than asking for dense pixelwise labels, Ilastik provides a user interface to identify training examples that could potentially improve the classifier performance given the current classifier output overlaid on the input image. The works of Kaynig et. al. [3] and Parag et.al. [7] also proposed methods for sparse selection of a subset of training examples and demonstrated their advantages for EM segmentation. However, the strong dependence of the random forest classifiers on hand-tuned features has the potential to limit their performances on images from different EM preparation/imaging techniques and, more generally, from other data modalities.

In this paper, we present a method for interactive training

of a Convolutional Neural Network (CNN) classifier [5] for EM image segmentation. The user paints on an input image to mark the pixels corresponding to a particular class. Presented with the pixel classification performance of the CNN, the user marks one or more areas that s/he thinks would improve the quality of the segmentation. We propose an efficient training algorithm to yield near real-time feedback to the user. Our training method (Section 2.1) is designed to emphasize the most recent user input while retraining the CNN. In addition, the learning algorithm also keeps track of past training examples that were adversely affected by this update and prioritizes them for learning in the next iteration. For segmentation purposes, our training strategy enables us to train a CNN with less than 2% of the total training examples, that is a few hundred thousands out of millions of total training examples. Compared to a CNN of the same architecture trained on all ground-truth images, our technique achieves slightly better segmentation accuracy. Our results corroborate well with past studies [3, 7] that found that interactive training tends to produce a classifier that performs better for segmentation.

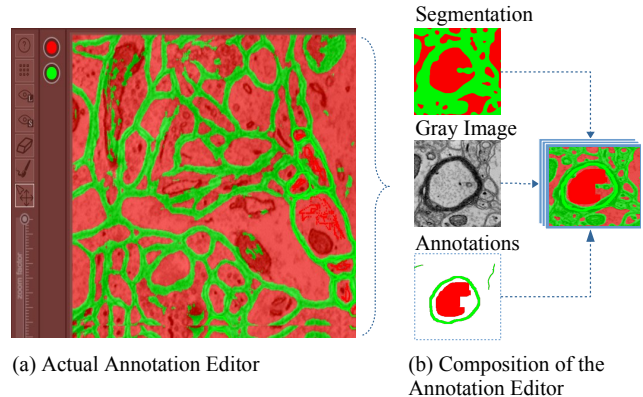
The primary objective of training the pixel detector using our method is region segmentation. Therefore, we use Variation of Information (VI), which has become a standard measure in connectomics [3, 8, 7], to quantitatively compare segmentation performance. To our knowledge, ours is the first effort for interactive training of deep networks for EM segmentation. Although the method is primarily targeted and tested on EM data, we believe several problems in biomedical image segmentation could benefit from our method, including mitosis detection [9], cell segmentation on histopathology images [10], and cell tracking [11].

## 2. METHOD

ICON, our tool for boundary detection, employs a CNN as a pixel detector to classify each pixel into membrane and non-membrane classes in a cell. The CNN is trained with sparse annotations collected interactively from users over a web-based graphical user interface (GUI) shown in Fig. 1.

The annotations are saved in a central database as training samples to be exploited for training by the CNN. The CNN confidences for pixel detection are overlaid on the grayscale input images and displayed on the GUI to guide the user during the annotation process. Based on the performance of the pixel detector and the user expertise, the user annotates locations where an improvement in pixel classification can lead to more accurate segmentation.

The CNN classifier runs on two parallel threads on a separate compute node than the GUI, as described in Section 2.3. The first thread is dedicated to training. It draws samples from the central database to train the classifier and saves to disk a model when better validation accuracy is achieved. The saved model is used by a second thread that produces segmentation outputs on demand for images that are currently being anno-



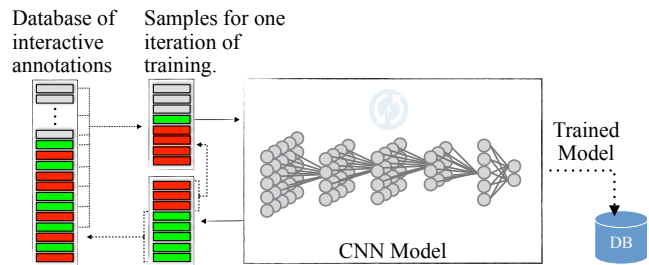
**Fig. 1.** (a) ICON’s main screen for capturing annotation and visualizing classifier output. It consists of tools for editing annotations, controlling visualizations, and radio buttons to indicate object classes. (b) The three layers that make up the visualization portion of the screen.

tated.

The classifier is trained on samples from multiple images collected from users via a web interface. Because we are training the classifier from scratch, a minimum of 100,000 samples are required before useful feedback is seen. Our training method is described next.

### 2.1. Classifier Training

One of the issues with conventional CNN training in an interactive environment is how to achieve real-time feedback to keep up with annotation input from users. In our training strategy, depicted in Fig. 2, we focus on accelerating the learning process so that the classifier is able to produce segmentation output in a reasonable amount of time. In addition, our learning scheme also employs a training sample selection technique in order to be responsive to the user input and maintain a level of accuracy over all the samples annotated so far.



**Fig. 2.** Iterative classifier training. New samples are shown in gray. Good performing samples are shown in green. Poorly-performing samples are shown in red.

At the beginning of a training iteration, the learning thread draws a large number of samples from the central

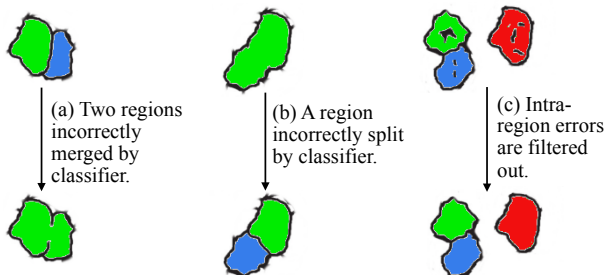
database of annotations. A *sample* is a patch of the image extracted from a square region centered around a pixel. The samples are drawn across all training images and labels, and we sub-sample classes equally to avoid imbalanced training data. During the sampling process, priority is given to new annotations. We also perform arbitrary rotation on the samples to ensure the network can recognize objects in different orientations.

After sampling, we combine the training samples with poorly-performing examples from the previous iteration of training. The performance of a sample is measured by the error between the network output and the target value. A large error indicates the sample has not been learned by the network, therefore the system needs to present this sample more frequently to the network. Given the set of all samples  $S$ , the set of poorly performing samples is described as:  $S_b = \{x_i \in S \mid \|y_i - f(x_i)\| > \delta\}$ , where  $y_i$  is the label of example  $x_i$ ,  $f_i$  refers to the network output, and  $\delta$  is a threshold that is set by application to 0.5 since the network is classifying membranes and non-membranes. Next, the learning thread iterates over the samples and perform Stochastic Gradient Descent (SGD) training using a small subset of the samples at a time.

After each training iteration, the samples are evaluated and a maximum of 50% of the poorly-performing samples is retained for the next iteration of training.

## 2.2. Classifier Refinement

Among the classification errors generated by the deep network, it is important to identify the misclassified pixels that causes false merge and splits in the segmentation. We display examples of these scenarios respectively in Fig. 3(a) and (b). Misclassification of pixels within segmented regions – as shown in Fig. 3(c) – often do not affect the quality of segmentation as a result of the region growing methods used in subsequent processing steps of the pixel predictions [3].

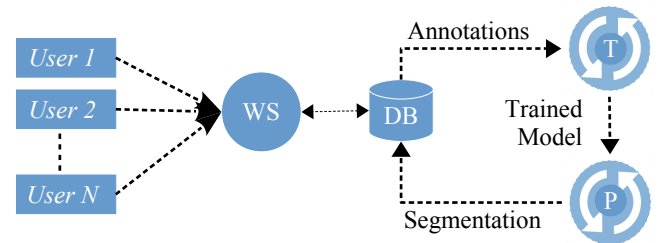


**Fig. 3.** Segmentation error induced by pixel classification: (a) A merge error, where two regions are incorrectly combined into one. (b) A split error, where one region is incorrectly divided into two. (c) Pixel classification error that does not affect segmentation.

Therefore, during the annotation process, we use the pixel confidences as an overlay to focus the manual effort on fixing classifier mistakes in order to guide the classifier to pay more attention to these cases. This refinement process is important because we are interested in region segmentation and we evaluate our system on this basis.

## 2.3. System Implementation

The architecture of the system is comprised of several parts, depicted in Fig. 4. On the front-end is a GUI that runs in a web browser and provides facilities for editing annotations. The GUI is connected to a web service that runs on a server. The web service facilitates the data exchange between the GUI and the CNN model by utilizing a database to store annotations and retrieve segmentation outputs. The classifier is trained on one thread and outputs a trained model that is used by a prediction thread. The prediction thread runs in parallel to the training thread and produces probability maps as described in section 2.1. These parts are integrated into a real-time feedback loop that synchronizes the server with user-provided annotations and the GUI with segmentation outputs from the classifier.



**Fig. 4.** The architecture of the system consists of a GUI, a web service (WS), a database (DB), a training thread (T), and a prediction thread (P).

## 3. RESULTS

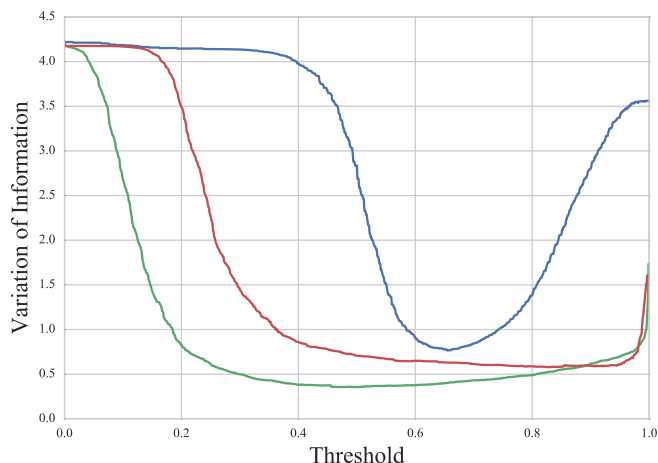
To evaluate our system, we use data from sections of a tissue of a dense mammalian neuropil from layers 4 and 5 of the S1 primary somatosensory cortex of a 5 month old healthy C45BL/6J mouse. 240 gray-value section images with  $1024 \times 1024$  pixels are used for training and validation. For testing, we use 120 images divided into two sets from which results were generated. Each image has a corresponding ground-truth membrane and background labels for every pixel.

The CNN architecture consists of two convolutional layers, each composed of 48 filters of size  $5 \times 5$ . The fully-connected layer of the network consists of 200 units and the output layer has two units, one for membrane and the other for non-membrane. We use a learning rate of 0.01 and a momentum of 0.9.

For comparison, we trained another CNN with the same architecture as the interactive setting offline with the full set of ground-truth labels. The interactive classifier was trained on 185,990 pixels sparsely annotated on ten training images, i.e., 1.7% of total pixels across the ten images. The interactive classifier typically converges after one hour of training time and the offline classifier after two hours.

We use VI to quantitatively measure segmentation performances produced on the test images by the different classifiers. We first generate probability maps for each image. Then we threshold each probability map at different intervals. The threshold is a decision boundary value that is applied to the probability map to separate membranes from non-membranes. We then compute region clusters from the probability maps using Mahotas [12] connected components routine and compare them against ground-truth region clusters to produce the VI measurements. The final VI for each threshold is averaged from all the probability maps. For comparison, we also compute segmentations by thresholding the gray value images following the same criteria as the probability maps to produce the VI measurements.

Following Kaynig et al. [3] and Nunez-Iglesias et al. [8], we plot the VI curves of segmentations produced by thresholding at different values of CNN pixel prediction in Fig. 5. These are results we generated from 100 test images. The blue graph is the results of thresholding the gray value images. The red graph, shown in the middle, represents the offline CNN classifier. The green graph is the interactive classifier, which achieved the lowest VI value of 0.36. Overall, the interactive classifier led to better results than the others consistently over a large range of thresholds.



**Fig. 5.** Segmentation error in VI on 100 test images. Blue, Red and Green curves show the VI errors for segmentations generated from thresholding gray value images, the output of offline classifier, and the prediction from interactively-trained classifier, respectively.

The results demonstrates the strength of our interactive training framework. With sparsely annotated pixels, we can label neuronal structures in EM images and achieve better results than the conventional method of manually labeling all pixels.

## 4. CONCLUSIONS

We presented an interactive approach for training a CNN for segmentation of EM images. We demonstrated that by training a classifier with sparse annotations using our technique we are able to produce better results than a classifier of the same network architecture trained offline on all ground-truth samples.

We primarily developed the system for applications in neuroscience to address the manual and tedious process of labeling neuronal structures in brain images. We believe that the resulting method is applicable to segmentation tasks beyond the field of neuroscience. We will make the software of this system freely available upon acceptance of this paper to a peer-reviewed publication.

Future work will concentrate on addressing the training start-up of the interactive system. Currently, the system requires 100,000 samples before meaningful results can be seen from the network. A possible remedy to this problem is to use a pre-trained offline network as a seed for the interactive system.

## 5. ACKNOWLEDGEMENT

This work is partially supported by NSF grants IIS-1447344 and IIS-1607800 and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00002.

## 6. REFERENCES

- [1] Moritz Helmstaedter, Kevin L. Briggman, Srinivas C. Turaga, Viren Jain, H. Sebastian Seung, and Winfried Denk, “Connectomic reconstruction of the inner plexiform layer in the mouse retina,” *Nature*, vol. 500, no. 7461, pp. 168–174, Aug. 2013.
- [2] Shin-ya Takemura, C. Shan Xu, Zhiyuan Lu, Patricia K. Rivlin, Toufiq Parag, et al., “Synaptic circuits and their variations within different columns in the visual system of drosophila,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 44, pp. 13711–13716, 2015.
- [3] Verena Kaynig, Amelio Vazquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R. Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister, “Large-scale automatic reconstruction of neuronal processes from electron microscopy

- images,” *Medical Image Analysis*, vol. 22, pp. 77–88, 2015.
- [4] Viren Jain, Benjamin Bollmann, Mark Richardson, Daniel R. Berger, Moritz Helmstaedter, Kevin L. Briggman, Winfried Denk, Jared B. Bowden, John M. Mendenhall, Wickliffe C. Abraham, Kristen M. Harris, Narayanan Kasthuri, Ken J. Hayworth, Richard Schalek, Juan Carlos Tapia, Jeff W. Lichtman, and H. Sebastian Seung, “Boundary learning by optimization with topological constraints,” in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, 2010, pp. 2488–2495.
- [5] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jurgen Schmidhuber, “Flexible, high performance convolutional neural networks for image classification,” *International Joint Conference on Artificial Intelligence, IJCAI*, vol. 2, pp. 12371242, 2011.
- [6] Christoph Sommer, Christoph N. Straehle, Ullrich Kthe, and Fred A. Hamprecht, “Ilastik: Interactive learning and segmentation toolkit,” in *IEEE International Symposium on Biomedical Imaging, ISBI*, 2011, pp. 230–233.
- [7] Toufiq Parag, Dan C. Ciresan, and Alessandro Giusti, “Efficient classifier training to minimize false merges in electron microscopy segmentation,” in *IEEE International Conference on Computer Vision, ICCV, Santiago, Chile*, December 2015, pp. 657–665.
- [8] 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*, vol. 8, no. 8, pp. 1–11, 2013.
- [9] Dan C. Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber, “Mitosis detection in breast cancer histology images with deep neural networks,” *Medical Image Computing and Computer-Assisted Intervention, MICCAI*, vol. 8150, pp. 411–418, 2013.
- [10] Hai Su, Fuyong Xing, Xiangfei Kong, Yuanpu Xie, Shaoting Zhang, and Lin Yang, “Robust cell detection and segmentation in histopathological images using sparse reconstruction and stacked denoising autoencoders,” *The 18th Annual International Conference on Medical Image Computing and Computer Assisted Intervention, MICCAI*, 2015.
- [11] Martin Maka, Vladimr Ulman, David Svoboda, Pavel Matula, Petr Matula, Cristina Ederra, Ainhoa Urbiola, et al., “A benchmark for comparison of cell tracking algorithms,” *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.
- [12] Luís Pedro Coelho, “Mahotas: Open source software for scriptable computer vision,” *Journal of Open Research Software*, vol. 1, 2013.