Automated Object Tracing for Biomedical Image Segmentation Using a Deep Convolutional Neural Network

Erica M. Rutter(✉), John H. Lagergren, and Kevin B. Flores

Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, Raleigh, USA
{erutter,jhlagerg,kbflores}@ncsu.edu

Abstract. Convolutional neural networks (CNNs) have been used for fast and accurate segmentation of medical images. In this paper, we present a novel methodology that uses CNNs for segmentation by mimicking the human task of tracing object boundaries. The architecture takes as input a patch of an image with an overlay of previously traced pixels and the output predicts the coordinates of the next m pixels to be traced. We also consider a CNN architecture that leverages the output from another semantic segmentation CNN, e.g., U-net, as an auxiliary image channel. To initialize the trace path in an image, we use either locations identified as object boundaries with high confidence from a semantic segmentation CNN or a short manually traced path. By iterating the CNN output, our method continues the trace until it intersects with the beginning of the path. We show that our network is more accurate than the state-of-the-art semantic segmentation CNN on microscopy images from the ISBI cell tracking challenge. Moreover, our methodology provides a natural platform for performing human-in-the-loop segmentation that is more accurate than CNNs alone and orders of magnitude faster than manual segmentation.

1 Introduction

Deep convolutional neural networks (CNNs) have recently attained state-of-the-art performance on many important biomedical imaging tasks [4,7–9]. CNNs have been increasingly applied to automated digital pathology and microscopy image analysis [4,9]. An advantage to using automated analysis is that the output can be less heterogeneous than manual annotation results from different observers, which could be variable due to differences in opinion [2,5].

Supported by the National Science Foundation under grant number DMS-1514929.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-00937-3_78) contains supplementary material, which is available to authorized users.

An application where machine learning models, and in particular CNNs, have seen wider recent adoption in biomedical image analysis is cell segmentation [4], i.e., delineating the boundary of each cell in an image [12]. Developing methods for implementing CNNs for segmentation is an active research area, since no one method currently outperforms all others on every data set [10]. One of the first methods using CNNs for segmentation was one in which the input to the network is a square patch from the image and the output is a prediction for the class of the center pixel, i.e., either inside, outside, or on the boundary of an object [1,11]. This method requires the user to input one square patch per pixel in order to classify all pixels in the image, making it computationally restrictive. Recent improvements to this method include fully convolutional neural network architectures [4,9]. These networks take as input a large tile of the image and use deconvolution layers to output another image tile containing classification probabilities for every pixel in the output tile. Since the output tile size can be large, this method is orders of magnitude more efficient than using a CNN architecture that outputs class predictions for a single pixel at a time.

A current challenge in using CNNs, including fully convolutional networks, for segmentation of cell microscopy images is that they do not ensure that one contiguous region is generated for each cell. For example, we have observed that CNNs may produce segmentation predictions with holes or several separate regions in each cell. Many possible factors could contribute to these "patchy" segmentation patterns, such as local differences in contrast or sharpness that make the cell boundaries appear more diffuse. In such regions, low segmentation accuracy may reflect that the CNN produces ambiguous class prediction probabilities, i.e., low certainty between whether a pixel is in the interior, exterior, or boundary of a cell (see Supplementary Figure S1 for such an example).

Here, we propose a novel method using CNNs to improve segmentation accuracy that specifically focuses on tracing the boundaries of the cell, mimicking the human task of tracing the boundary of an object. By formulating segmentation as a tracing task, our method constrains the segmentation problem to better ensure continuity of segmented regions. We found that our tracing method can outperform the state-of-the-art segmentation method on an ISBI cell microscopy tracking challenge data set. Since our methodology is formulated for the general task of segmentation, it can be applied to many different segmentation tasks. We exemplify the use of our automated tracing methodology for "human-in-the-loop" segmentation by simulating a scenario in which the user first defines a short 8-pixel long initial trace, and then our CNN completes the tracing path. Thereby, our method provides a platform for which minimal user supervision can enable increased accuracy over CNNs alone, while still leveraging CNNs to be orders of magnitude more efficient than completely manual segmentation.

2 Data and Methods

2.1 Data

We used benchmark data from the ISBI cell tracking challenge [6,10] to evaluate the performance of our methodology. We used a data set consisting of 34 fully annotated 2D grayscale light microscopy images of Glioblastoma-astrocytoma U373 cells on a polyacrylimide substrate. Each image is normalized by the pixel-wise mean and standard deviation across all 34 images. The normalized images are then randomly split into 20 training, 6 validation, and 8 testing images. The results we report here are averages from 10 random train/validation/test splits.

2.2 Network Architectures

A Convolutional Neural Network for Tracing Object Boundaries. Our tracing network takes as input a 64×64 image patch with an overlay of an 8-pixel long contour of the previously traced path ending at the center of the patch. The contour helped provide context for predicting the tracing direction. Ground truth masks were used to create the 8-pixel long contour in each training patch. An input patch generation example is shown in Fig. 1. We note that patch size needs to be tuned and we attempted smaller and larger patch sizes, but found that 64×64 provided adequate context for the prediction task on our considered data set. In general, we recommend using a patch size of the same order of magnitude as the mean object diameter, as previously suggested [11].

We also tested a tracing network whose input included an additional channel containing the output probabilities of a trained U-net, however, predictions from any segmentation method (e.g., Mask-RCNN [3]) could be used as additional data channels. A U-net prediction channel was concatenated with each full image and our $64 \times 64 \times 2$ patches were sampled from this 2 channel image.

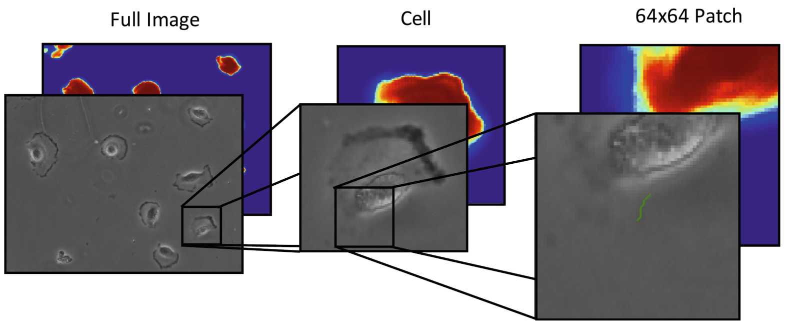

Fig. 1. Generating training data patches. Left: A full image from the microscopy data set. Middle: A cell within the image. Right: A 64×64 patch with the previous 8 pixels traced, shown in green. The U-net probability scores can be concatenated to the image to make a second channel, i.e., a $64 \times 64 \times 2$ patch.

The network architecture is comprised of 3 blocks of repeating layers followed by a final convolution layer (Supplementary Table S1). The repeating layers consist of 3 3×3 convolutional layers followed by a 2×2 max pooling layer with 32, 64, and 128 filters, respectively. The final layer is an 8×8 convolution with $2 \times m$ filters. The network output is a $2 \times m$ array consisting of predictions for the next m pixel locations along the cell border, i.e., the predicted horizontal and vertical pixel displacements of the next m steps along the tracing path. The training data for the output were extracted from the m pixels succeeding the center pixel in the input patch using the ground truth contours. We trained the network for 10 epochs using a mean squared error regression loss, the Adam optimizer, and a batch size of 32. Data augmentations were performed on full-size training images, including random rotations between $15\,^\circ$ and $75\,^\circ$, random shears between $10\,^\circ$ and $30\,^\circ$, vertical and horizontal flips, and Gaussian blur. Full-sized images were symmetrically-reflected with 32 pixels to ensure that pixels on the edge of the image could be used as the center of a 64×64 image patch.

U-net. We used a U-net architecture to create a second channel for the tracing network inputs, to propose candidate locations to initialize the tracing path, and as a baseline for comparing segmentation accuracy. We trained U-net on the same set of training images used to train the tracing network by following the overlap-tiling strategy for seamless segmentation outlined in [9]. The size of the input image tile for the network was 572×572 and the output is a 388×388 tile of probability scores for being in the interior of a cell. Input and output tiles were created from full-size training images and ground truth segmentation masks. Training data were augmented as described in the previous paragraph. For training, we used the pixel-wise soft-max cross entropy loss function described in [9] without the weighting scheme, since the data used in this work did not contain cells that touch.

2.3 Tracing Algorithm

We developed a tracing algorithm to create contours by iterating the output of our trained tracer network. The traced contours outline the boundary of cells and are used to define segmentation masks. The algorithm starts with an initial 64×64 patch with an 8-pixel long trace ending at the center pixel. Methods for generating the initial 8-pixel trace are discussed below. The patch is input to the tracer network, which then predicts the location of the next m pixel coordinates relative to the center pixel in the patch. We aggregate the information from the m pixel predictions by allowing each pixel to vote for one of the 8 directions to move the trace, corresponding to the 8 pixels adjacent to the center pixel of the patch. The vote for each pixel corresponds to the angle (θ) of the vector made by its coordinate relative to the center pixel. For example, if $\theta \in [\frac{-\pi}{8}, \frac{\pi}{8}]$, the vote is for the pixel adjacent to the right, and $\theta \in [\frac{\pi}{8}, \frac{3\pi}{8}]$ corresponds to the upper right adjacent pixel, etc. (Fig. 2). Pixel votes are weighted according to their distance (x) from the center pixel according to an exponential function $w(x) = e^{-\alpha x}$, where α is a hyper-parameter. The next pixel of the contour is drawn in the direction with the largest vote. The newly drawn pixel is used as the center of a new image patch and we iterate the process until the distance between the new pixel and the first 100 pixels in the trace is less than 5. Similar to the tracer network training, padded images were used for the tracing algorithm. The trace was not allowed to enter the padded region. If the trace was predicted to enter the padded region, it was instead automatically routed in a direction parallel to the boundary of the padded region that was most congruent to its previous path. Psuedo-code for our tracing algorithm is provided in the Supplementary Material.

To generate the initial 8-pixel trace, we used a segment of the contour for the ground truth mask to simulate a human-in-the-loop scenario. To implement a fully automated algorithm, the initial trace was generated by U-net. Because results may be sensitive to initial starting conditions, we want to ensure that the initial contour we choose from U-net is where U-net is most sure a boundary exists. Therefore, we determined the 8-pixel start path by calculating the laplacian of the U-net probability scores and the U-net generated masks. We calculated the 8 contiguous pixels that generated the highest U-net probability derivatives, within ± 4 pixels of contours found for U-net.

2.4 Training and Testing Procedure

U-net was used as a baseline for comparison, since it is currently the top performing segmentation method for the data set we use here. The augmented training and validation data sets were used for training U-net for 50 epochs, and the model from the epoch with the lowest validation loss was used for downstream procedures. The validation set was used to optimize the probability threshold for segmentation, where segmentation accuracy was quantified by the mean Jaccard score over the entire set of validation images.

We trained the tracer network with patches from the augmented training images, or 2-channel patches from the augmented training images concatenated with their corresponding U-net probability maps. Due to heavy data augmentation, 10 epochs were sufficient for training the network. The combined non-augmented set of training and validation data were used to optimize the hyper-parameter α with respect to the mean Jaccard score for the tracing algorithm.

We assessed the accuracy of each segmentation method, i.e., the fully automated or the ground-truth initialized tracing algorithm, each with one or two channel inputs, with the test set. To ensure robustness with respect to the randomly chosen training/validation/test split, we validated our results over 10 random data splits.

3 Results

We evaluated the segmentation accuracy of our tracer CNN method for several models corresponding to choices of whether we (1) used an 8-pixel initial trace generated by U-net gradients or the ground truth mask, (2) used the U-net probability scores as a second channel for patches input to the tracer CNN, and (3) set the number of pixels ahead predicted by the tracer CNN to $m = 10$ or 20. The Jaccard scores (intersection over union) over the full images in the testing set were used to quantify segmentation accuracy and U-net was used as a baseline for comparison. We note that, for the ISBI cell tracking challenge [10] and in the U-net paper [9], segmentation accuracy is averaged over all manually segmented objects and not on the full images as we calculate here. Importantly, this only quantifies accuracy for predicted segmentations that overlap with segmented objects in the ground truth, and hence ignores any false positives that do not overlap with ground truth objects. The Jaccard score calculation we perform here is over each entire test image, and thus better accounts for the

Table 1. Jaccard scores on the testing set averaged over ten train/val./test data splits for the current state-of-the-art segmentation algorithm (U-net) and our tracing network. Jaccard scores higher than state-of-the-art are in bold font.

Method	Ground truth Initialized	U-net channel	# pixels predicted	Jaccard score (mean ± std)	Jaccard score (median)
Tracer	No	No	10	0.8091 (0.04216)	0.8443
	No	No	20	0.8268 (0.03749)	0.8579
	No	Yes	10	**0.8407** (0.03873)	**0.8742**
	No	Yes	20	**0.8460** (0.04764)	**0.8841**
	Yes	No	10	**0.8479** (0.03485)	**0.8776**
	Yes	No	20	**0.8626** (0.01497)	**0.8942**
	Yes	Yes	10	**0.8621** (0.02785)	**0.9070**
	Yes	Yes	20	**0.8611** (0.02414)	**0.9044**
	Yes (retraced)	Yes	10	**0.8629** (0.0244)	**0.9074**
	Yes (retraced)	Yes	20	**0.8797** (0.0171)	**0.9054**
U-net [9]	-	-	-	0.8370 (0.03329)	0.8624

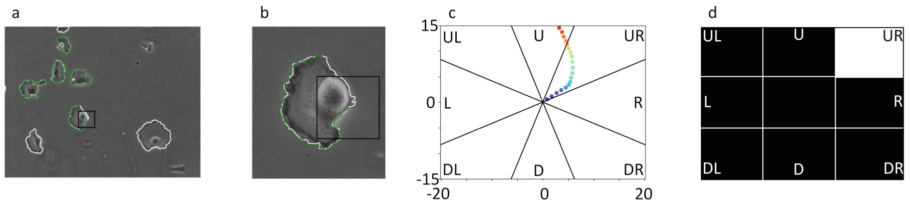

Fig. 2. (a) An example of an image in the process of being segmented. Ground truth contours in white, our traced contours in green. (b) A zoom of the current cell being traced. The black square in (a) and (b) is the current 64×64 patch input to the CNN to predict the location of the next pixel. (c) The predicted location of the next 20 pixels relative to the center pixel of the patch. (d) The score map used by the tracing algorithm to determine the location of the next pixel in the trace, e.g., the upper left (UR).

possibility of false positives. The results in Table 1 show the average scores over 10 train/validation/testing data splits. For a full table of results for each individual train/validation/testing data split, see Supplementary Table S2.

We found that our tracer method was more accurate than U-net when using either a ground truth initialization or the U-net prediction scores as a second channel to the tracer CNN. Using U-net for tracing initialization and to make a second data channel for the tracer CNN that predicts 20 pixels ahead resulted in higher mean (+0.9%) and median (+2.17%) Jaccard scores. When using ground truth initialization without a U-net channel, our method performed even better (+2.58% mean and +3.18% median jaccard score change for the 20-pixel predictions). The highest improvements in mean or median Jaccard score were achieved by the ground truth initialized tracer using a U-net channel (+2.51% mean and +4.46% median jaccard score change for the 10-pixel predictions). These findings suggest that a human-in-the-loop tracing system requiring minimal user input, combined with U-net to provide an additional data channel, could be more accurate than using U-net alone. Figure 2 shows the results of an example image from a testing set, with ground truth contours in white and our traced contours in green. An example video of our cell tracing algorithm is shown in Supplementary Video 1.

We observed that the margin for the median Jaccard scores between our tracer method and U-net were larger than the margin for the mean. For example, when using the 20-pixel tracer CNN with ground truth initialization and the U-net channel, the margin for the median was +4.20%, but the margin for the mean was +2.41%. This suggests that there may be outlier images for which the ground truth initialization performs poorly. Since the goal of testing the ground truth initialized method was to assess the potential increase in accuracy of a human-in-the-loop system, we hypothesized that the accuracy could be further increased by attempting several ground truth initializations for the 8-pixel trace if the Jaccard score was especially low for a given cell. This would emulate a system for which the user observes a poorly drawn trace, and then attempts to reinitialize the trace at a different location on the cell boundary. To investigate this, we implemented a check in the tracing algorithm to determine if the Jaccard score for a given cell was less than 0.8. We note that this condition was met on average <1 cell per image. If this condition was met, we reinitialized the 8-pixel trace at a random location in the ground truth cell boundary up to 10 times. The retraced contour with the maximum Jaccard score for that cell is then used when creating the segmentation mask for the image. We found that allowing for retracing of cell contours increased the margin for the mean to +4.27% when using the 20-pixel tracer CNN (Table 1).

4 Conclusions

We showed that the tracing CNN methodology presented here is able to increase segmentation accuracy over U-net when evaluating the Jaccard score over the entire image. Our tracing algorithm is fast, taking <1 s per cell to complete a trace initialized with an 8-pixel contour on an NVidia GTX 1080Ti (11 GB). This speed makes our methodology practical for implementation within a human-in-the-loop system that is orders of magnitude faster than completely manual segmentation. Moreover, although our fully automated results showed improvement over U-net, we found that the largest gains in accuracy came when we used ground truth initialization for the trace, which is representative of a human-in-the-loop system. For medical imaging applications of CNNs that currently require an expert to verify the CNN segmentation predictions, we envision that initial traces for our algorithm could be provided without much extra effort. We expect our method to benefit data sets with objects containing diffuse looking regions, which are common in biomedical data, e.g., brain tumor MRIs.

In future work we will test our methodology in settings such as brain tumor segmentation for which even small differences in accuracy could be detrimental and currently require human verification. We will also test whether our retracing strategy can be used with U-net initialized traces to handle inaccurate segmentations that arise from the rare occurrence if the trace deviates from the object boundary. For example, we can aggregate the results from 10 U-net initialized traces per object by majority vote.

References

1. Ciresan, D., et al.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) NIPS, pp. 2843–2851. Curran Associates, Inc. (2012)
2. Foran, D.J., et al.: Imageminer: a software system for comparative analysis of tissue microarrays using content-based image retrieval, high-performance computing, and grid technology. J. Am. Med. Inf. Assoc. **18**(4), 403–415 (2011). https://doi.org/10.1136/amiajnl-2011-000170
3. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: IEEE ICCV, pp. 2980–2988. IEEE (2017)
4. Litjens, G., et al.: A survey on deep learning in medical image analysis. Med. Image Anal. **42**, 60–88 (2017)
5. López, C., et al.: Digital image analysis in breast cancer: an example of an automated methodology and the effects of image compression. St. Heal. T. **179**, 155–71 (2012)
6. Maška, M., et al.: A benchmark for comparison of cell tracking algorithms. Bioinformatics **30**(11), 1609–1617 (2014)
7. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: fully convolutional neural networks for volumetric medical image segmentation. In: Fourth International Conference on 3D Vision (3DV), pp. 565–571 (2016). https://doi.org/10.1109/3DV.2016.79
8. Rodríguez Colmeiro, R.G., Verrastro, C.A., Grosges, T.: Multimodal brain tumor segmentation using 3D convolutional networks. In: Crimi, A., et al. (eds.) BrainLes 2017. LNCS, vol. 10670, pp. 226–240. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75238-9_20
9. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28

10. Ulman, V., et al.: An objective comparison of cell-tracking algorithms. Nat. Methods **14**, 1141 (2017). https://doi.org/10.1038/nmeth.4473
11. Valen, V., et al.: Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLoS Comput. Biol. **12**(11), 1–24 (2016). https://doi.org/10.1371/journal.pcbi.1005177
12. Xing, F., Yang, L.: Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: a comprehensive review. IEEE Rev. Biomed. Eng. **9**, 234–263 (2016)