Cell Nuclei Segmentation in 2D Fluorescence Microscopy Images

Onur Beker, Zhen Wei, Weitong Zhang. *Mentor*: Martin Weigert
Hosting Lab: Bioimage Analysis and Computational Microscopy Group, EPFL

***Abstract*—Cell nuclei segmentation is an essential step in the biological analysis of microscopy images, where such annotations are commonly obtained by manually labeling pixels by hand using an image processing package such as ImageJ/Fiji. Given the impracticality of such a manual labelling step, it is highly desirable to automate it. In this project, we address this problem and implement a convolutional neural network based segmentation model by extending the popular U-Net [1] architecture. We experiment with using different loss functions, transfer learning, and online hard example mining. As a baseline, we compare our model with *Stardist* [2], [3], a recent state-of-the-art method for cell nuclei segmentation.**

I. Introduction

Semantic segmentation is the task of assigning a category label to each pixel in an image (e.g., cell or background). Given its practical importance for many application domains, it is a well-studied problem where; as in almost every sub-field of computer vision; deep convolutional neural network based approaches are the common paradigm. In general, the majority of the recent deep-learning based segmentation methods roughly fall into the two following groups: *1)* methods that first predict regions of interest (in the form of axis-aligned bounding boxes), and then regress segmentation masks for every such region (e.g., the RCNN family of models); *2)* methods that use fully-convolutional models to directly regress pixel-wise class probabilities (e.g. U-Net based approaches). In our project, we adhere to the second approach and implement a U-Net based architecture that regresses a binary-class probability (i.e. cell/background) for each pixel.

II. Method

Our approach builds on top of the U-Net method by combining additional loss functions and regularization, as well as modifications to the architecture to allow for transfer learning. The following sections describe our approach in detail.

1) ***Training Data****:* For training, we use a recent dataset introduced in [4]. Given that the microscopy images have to be manually annotated by hand, the dataset is relatively small and consists of 79 fluorescence microscopy images, and corresponding ground truth instance segmentation masks. Since we are only doing a binary-segmentation, we aggregate all instance mask labels into cell/background categories. Out of the 79 images, we use 70 for training and 9 for testing. For data augmentation, we apply random crops, scaling, horizontal and vertical flips, and rotations.

2) ***Model Architecture****:* Our model combines a down-sampling feature extractor with an up-sampling segmentation head that regresses pixel-wise probabilities in the same resolution as the original input image. For the down-sampling path, we use a ResNet-18 backbone, which includes residual connections and batch normalization in each residual block, and down-samples its input by 16 times. The up-sampling segmentation head is a cascade of fractionally-strided convolutions (i.e., deconvolutions), where a skip connection is utilized after each up-sampling operation that concatenates features of the corresponding scale in the down-sampling path. Skip connections allow the up-sampling path to utilize high-resolution information, which would otherwise be lost in down-sampling due to pooling operations. As in the original U-Net paper, filter depth is doubled after each down-sampling step. All software implementations use the *Pytorch* framework. For optimization, we use the Adam optimizer and train our models for 100 epochs, with a learning rate of 0,0001.

3) ***Receptive Field****:* The essential parameter to consider when designing the architecture is the receptive field of each output pixel with respect to the input image. The residual summations in each ResNet block prevent the calculation of a theoretical receptive field (i.e. 2 to the power of pooling layers). This is because the non-residual path involves convolution operations that alter the receptive field of their input, whereas the residual connection doesn't. This means summing the residual and non-residual terms at the end of the ResNet block effectively blends pixels with different receptive fields, thus preventing a uniform notion of receptive field from being defined. As such, we compute an *effective receptive field* [5], which is essentially the derivative of the regression probability at the output of a single pixel with respect to all the input pixels (i.e., a matrix with the same dimensions as the input image, effectively a gradient image). The input pixels with very low gradient magnitude have no significant effect on the value of the output pixel. Therefore they are considered not to be in the receptive field. We validate that the effective receptive field of an output pixel is larger than the largest cell in the input images so that the regression probability of each output pixel depends on an area in the input image that is large enough to fit a cell.

4) ***Loss Functions****:* We experiment with two optimization metrics, Dice Loss and pixel-wise Binary Cross-Entropy (BCE). These loss functions are defined as follows:

1) Dice loss is defined as $(\frac{2|A \cap B|}{|A|+|B|})$, where A and B correspond to the ground truth and predicted masks. In

this context, intersection operation implies the multiplication of two masks, and cardinality operations imply the summation of all pixels in a mask, respectively.

2) For each individual pixel probability, the BCE loss is defined as $-((y \log(p) + (1-y) \log(1-p)))$, where y is the ground truth mask value (i.e. 1 for cells, 0 for background), and p is the predicted probability of a pixel belonging to a cell. The overall BCE loss is the average of all pixel-wise BCE losses.

The advantage of Dice loss over pixel-wise BCE is that it captures a rough representation of global statistics, due to the multiplicative term that mixes information from all pixels (in contrast, BCE is only an average of pixel-wise statistics). The biggest disadvantage of both methods is that they scale with area, meaning the loss function is essentially dominated by the pixels inside the mask and is much less sensitive to boundary pixels (i.e., since there are much fewer boundary pixels than there are interior pixels). The main effect of this relative insensitivity is that the resulting segmentation masks clump individual cells together, since the loss function provides no significant incentive for a boundary to appear in-between. Since cells often appear in clusters, and since the segmentation mask is ultimately intended for a down-stream analysis that requires well separated instances (e.g., counting the number of cells), this constitutes an important problem. To alleviate this effect, we tried utilizing online hard example mining to emphasize the boundary pixels.

*5) **Online Hard Example Mining**:* To emphasize the relative contribution of boundary pixels over the total loss function, we experimented with an online hard example mining scheme. In particular, we evaluate pixel-wise BCE at each training iteration to obtain a loss distribution over pixels. This step is intended to give a loss-map that identifies the pixels that have a high loss value (i.e. such pixels overlap with the boundary regions, and holes in the interior). After we get this loss-map over all pixels, we only propagate gradients from pixels that have the top K loss values. When the value of K is too small, we observed that the models do not converge. We hypothesize that this is because when only losses from a small subset of the pixels are considered, there is nothing constricting the consequent gradient updates to be consistent with each other. In particular, since the loss maps (i.e. and hence the top K pixels) change at each iteration and the final loss only depends on the values of the top K pixels, nothing is preventing a subsequent gradient update from disrupting those pixels that were the top K pixels (i.e. the corrected pixels) in the current gradient update. In other words, since the consequent gradient updates can re-introduce errors on the pixels that were corrected in the current update, training oscillates indefinitely when K is too small. To prevent this effect, we experiment with using a weighted sum of the losses of top-K pixels and a global loss term evaluated over all pixels (i.e. rather than just propagating the losses from top K pixels). We observed that while this approach helps diminishing holes in the resulting segmentation masks, it doesn't have any significant effect on the clumped boundaries.

*6) **Transfer Learning**:* Given that the dataset we use for training is quite small, we experiment with transfer learning to account for potential generalization issues. In particular, we initialize our models with a ResNet-18 backbone that is pre-trained on ImageNet, and modify the weights of this backbone in subsequent training iterations. Interestingly, we found that starting from an untrained ResNet-18 model instead results in identical performance. We hypothesize that this is due to the restricted nature of the potential scenes that appear in the microscopy imaging modality (e.g., compared to real world images obtained from consimer cameras). In particular, microscopy images are essentially a superposition of individual cells that are very similar to each other in appearance. The imaged scene is inherently 2D (i.e., cells between two thin glass slides), and the overall illumination setting is similar across images. In other words, there are very few variations in appearance (i.e., no lighting-transport based phenomena, and no complex occlusions or 3D variations in appearance). The relative simplicity of this setup, combined with the patchwise data-augmentations we employ and the convolutional nature of the models we train, imply that only a sub-patch of a single image is a good representative of the global statistics across all images, meaning it is far easier for models in this setup to generalize from small-datasets then it would be for models trained on datasets of real-world scenes (e.g. ImageNet, Microsoft COCO).

III. Results and Discussion

The following sections present results of the experiments we carried out to characterize our model.

*1) **Training Data**:* Figure 1 shows example images from the microscopy image dataset. We note that the dataset contains cells from patients obtained from 5 different pathological conditions, which is the main source of difference in the shapes of nuclei contained in the images. We train a single model that segments all cell types, rather than training individual models for each category (mainly because of the limited number of images)

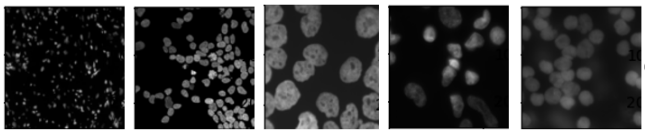

Fig. 1: Example images taken from the microscopy dataset

*2) **Receptive Field Computation**:* To compute the receptive fields for our models, we follow the procedure described in [5]. We start with a single pixel loss-image (i.e., the hypothetical Jacobian of the final loss with respect to the output image, a matrix where only a single entry is 1), and apply the chain rule to propagate gradients through the network to the features of every layer until the pixels of the input image at the end. Figure 2 shows the resulting receptive fields.

*3) **Dice Coefficient of the Baseline Model**:* Our baseline model is initialized with a ResNet-18 backbone that was

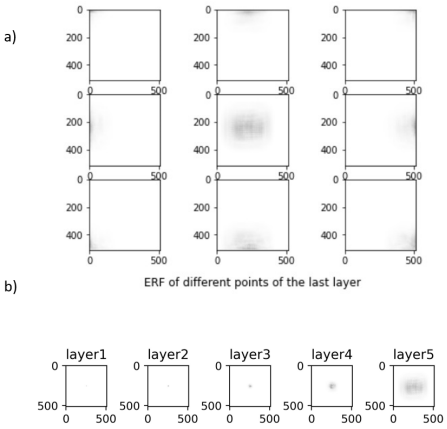

Fig. 2: a) Effective receptive field of an output pixel positioned at different locations, propagated back to the input image. b) Effective receptive field of a single pixel at the center of the output, propagated to different layers

pretrained on ImageNet. It achieves %98 accuracy, computed as the average Dice coefficient over test images. We note that we also initialized our model with a non-trained back-bone and observed an identical accuracy. Figure 3 shows example segmentation results for random images from the dataset. We observe that even though the Dice coefficient is high, segmentation results are clumped around cell boundaries, and there are holes present. This implies that the Dice Coefficient is not a perfectly ideal metric to capture the fidelity of segmentation masks.

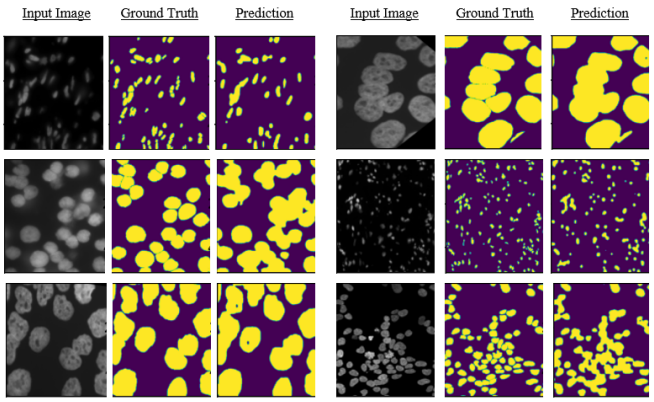

Fig. 3: Example images, ground-truth masks and segmentation results obtained using our model, side-by-side.

4) ***Evaluating the Performance on the Boundaries****:* To have a more fair evaluation of the segmentation fidelity of our model, we quantify its performance only around the boundaries. In particular, we extract cell boundaries using canny edge detection and then dilate the result using a 7x7 kernel. Figure 4 shows this processing step in detail. The resulting average dice coefficient on the test set is %56, which implies that even though mask interiors overlap relatively well, boundaries have poor alignment.

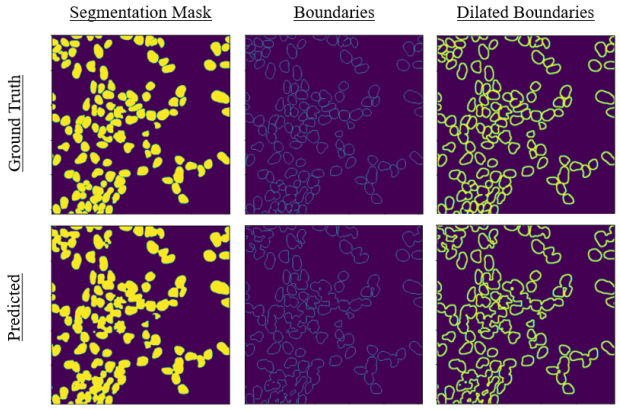

Fig. 4: Example images, ground-truth masks and segmentation results obtained using our model, side-by-side.

5) ***Online Hard Example Mining****:* To emphasize the contribution of the boundary pixels, we evaluate a loss map over all pixels using the BCE Loss, and propagate gradients from K pixels with the top loss values. Figure 5 shows these loss maps.

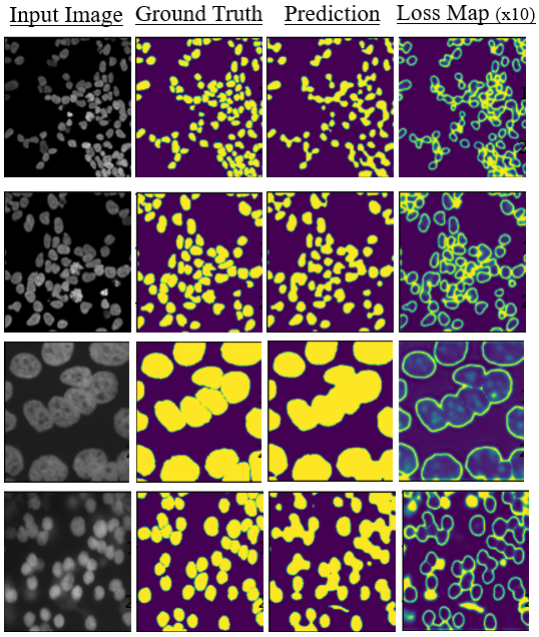

Fig. 5: Input images, ground-truth masks, predicted masks, and the resulting loss maps side-by-side.

We experimented with different values of K, and saw that when K is too small, the training did not converge. When this top K loss term is added as a regularization term to the dice loss (which captures global statistics and considers all pixels), we observe that the overall performance increases. Results of a rough grid search over different values of K are given in Figure 6.

6) ***Stardist Baseline****:* As an evaluation baseline, we trained the state-of-the-art Stardist instance segmentation model on our dataset. For each output pixel, Stardist model regresses

Model	Accuracy
K=6000 (%10)	% 0.2
K=14000 (%20)	% 95
K=33000 (%50)	% 96
Baseline Model (Dice Loss)	% 98
Dice Loss + K=6000	% 99

TABLE I: The accuracy for models trained with different top K values. Training is done on 256x256 patches.

a set of distances along a fixed number of predefined radial directions to the boundary of a cell, and an objectness score that is later used for non-maximum suppression. Given the particular shape of cell nuclei, star-convex polygons give a more suitable representation than general segmentation masks. By restricting the predicted shape to be a star-convex polygon, most segmentation artifacts (e.g., such as holes) are accounted for by the inductive bias that is introduced by the definition of this specific parametrization (i.e., the set of star-convex polygons can not include a shape with holes. Therefore it is impossible to predict a mask with holes). Example outputs from the Stardist model are given in Figure 6.

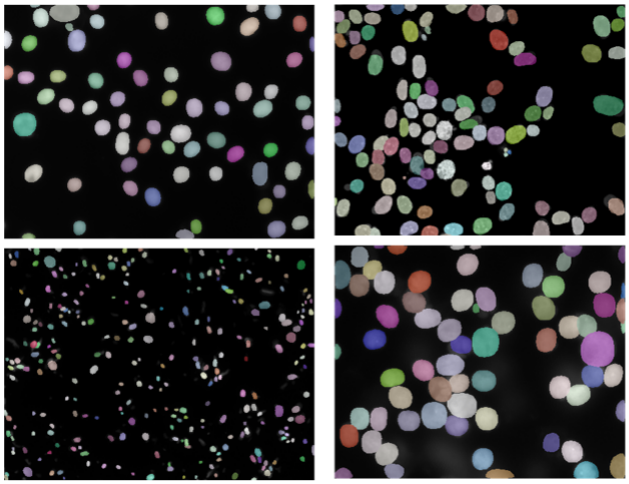

Fig. 6: Example segmentation results from the Stardist model.

Since Stardist is an instance segmentation model, we need to obtain instance segmentations from the binary masks of our model. To get a rough upper-bound on how our model would perform as an instance segmentation model, we employ a heuristic bipartite matching scheme between the foreground pixels of the predicted masks of our model and the instance labels in the corresponding ground-truth masks, effectively converting outputs of out model into instance segmentation masks. Specifically, we do the following:

1) Multiply our predicted binary masks with ground truth instance masks to assign instance labels to pixels that lie at the interior of both masks. For the sake of exposition, let us call the result as *mask1*.

2) We now want to propagate the labels of these interior pixels to all their connected components in our predicted masks, so that every foreground pixel in our predictions is assigned the label of the ground truth pixel that is closest to

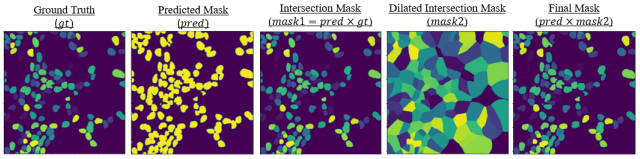

Fig. 7: Propagating instance labels from ground truth masks to predicted masks.

it. We dilate *mask1* with morphology operations (i.e., using the *skimage* module), to cover the entire image in a way that every pixel is assigned the instance label of the pixel in mask1 that is closest to it. Let us denote the result as *mask2*.

3) The final step is to filter out those pixels in *mask2* that are classified as back-ground by our model. To do this, we multiply *mask2* with our predicted mask. This way, every pixel on our predicted mask is assigned the instance label of the closest pixel in the ground truth mask. Figure 7 depicts the intermediate steps in this pipeline.

To evaluate instance segmentation performance, we compute the precision ($\frac{TP}{\#LabelsinPrediction}$), recall ($\frac{TP}{\#LabelsinGT}$), and F1 scores (i.e. harmonic mean of precision and recall) of our model and Stardist, using an IoU threshold of 0.5 (i.e. an instance prediction is a true positive if its IoU with the ground truth is larger than 0.5). We observe that our U-Net baseline achieves an F1 score of %78, whereas the Stardist model achieves an F1 score of %88. We note that as a result of the label propagation scheme we employed, the number of proposed labels always match the number of labels present in ground truth, meaning precision and recall will be equal.

IV. Conclusion

A dense segmentation model based on the U-Net architecture was implemented. To design the architecture in a principled way, the effective receptive field concept was used as a guideline. For regularization, a top-K loss term that emphasizes boundary pixels was implemented, and different K values were experimented with. As evaluation metrics, both standard Dice Loss, and Dice Loss on the boundary were used (i.e., the latter is a more plausible metric for measuring segmentation fidelity for this specific application domain). Finally, as a baseline comparison, instance labels were assigned to the binary segmentation outputs of the implemented model using a heuristic bipartite matching scheme, a state-of-the-art model was re-trained on our dataset, and the two were compared.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[2] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell detection with star-convex polygons," in *MICCAI*, 2018.

[3] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex polyhedra for 3d object detection and segmentation in microscopy," in *WACV*, 2020.

[4] F. Kromp, E. Bozsaky, and F. R. et al, "An annotated fluorescence image dataset for training nuclear segmentation methods," *Scientific Data*, 2020.

[5] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *arXiv:1701.04128*, 2016.