DeepCell: Automating cell nuclei detection with neural networks

Cristian Bartolome
Stanford University
cbartolm@stanford.edu

Yiguang Zhang
Stanford University
yiguang@stanford.edu

Ashwin Ramaswami
Stanford University
ashwin99@stanford.edu

Abstract

Automating cell nuclei detection is an important task in the entire process of drug testing and can help streamline the drug manufacturing process. We implemented a fully convolutional network (FCN) based on U-net, as a deep learning model to identify cell nuclei given a certain image. The network takes as input a grid of pixels with their RGB values from a microscope image, and gives as output a grid of binary labels. This approach performed significantly better than traditional, non-deep-learning methods for image segmentation. Refinement techniques such as dropout, batch normalization, and data augmentation helped to increase the performance of the original U-net architecture from 0.268 to 0.355 (+32%) in terms of the Mean IoU evaluation metric.

1 Introduction

Automating the detection of nuclei is a very important and crucial task during the process of drug testing. Identifying nuclei is often the starting point, but a large bottleneck, in the process of analyzing cells during drug tests. Currently, nuclei need to be individually identified and labeled by hand, a tedious and slow process when there are hundreds of thousands of data samples. An effective machine-learning solution will shorten drug tests, making it easier to find treatments for a wide variety of diseases, such as Alzheimer's, Huntington's, and Diabetes.

Owing to its relevance, this is the topic of the 2018 Kaggle Data Science Bowl. Sponsored by Booz Allen Hamilton and the Broad Institute, the competition aims to let competitors develop their own models to tackle this important task. Currently, there are about 2,800 teams from across the world developing models and sharing their knowledge on this task.

Traditional methods of image segmentation simply do not perform very well on this task because the images are simply too varied (see Methods, section 5.1). A challenge, however, for using deep learning with our task task was having to work with the relative paucity of data provided.

2 Related work

In image segmentation, thresholding is an important tool to separate objects from the background. Some classical thresholding algorithms include Otsu's method [1], quadtree method [2], and the relaxation method [3]. Among them, the Otsu's method is the most commonly used approach, and we implemented this approach as the first-step of our project. In the past decade, deep convolutional networks have outperformed the traditional methods in many image segmentation tasks. In 2012, Ciresan et al. [4] proposed a network in a sliding-window setup to predict the class label of each pixel. However, the model requires fixed input size, and there is a trade-off between the accuracy of localization and the use of context. In 2014, Long et al. [7] introduced a fully convolutional network (FCN) that solves the first problem by replacing the usual fully-connected layers in convolutional neural network (CNN) by upsampling layers. In 2015, Ronneberger [8] solves the second problem by modifying the upsampling part to contain a large number of feature channels, which creates a high-resolution segmentation map. In the second part of our project, we adapt the U-net architecture, and we add batch normalizations and regularizations (such as dropout, l2), in order to further improve the performance.

3 Dataset and Features

The data given consists of a set of 670 training images provided by the Broad Institute at Harvard and MIT [7]. These images have been obtained from microscopes in many different conditions. Some are black-and-white, while others are in color; they are in different shapes and sizes and contain pictures of varying types of cells. As a result, traditional image segmentation methods perform poorly on these images.

They came with human-labeled mask files, one for each nucleus detected. Each mask file can be represented as a matrix of 0's and 1's, with 1 representing a pixel that is part of a nucleus and 0 for everything else. For each microscopic image, we merged the individual mask files together to get a composite mask file.

We also resized all training images and masks to a 256x256 resolution so that they could properly be processed by the network. We used 10% of the training images for validation while training our set.

The dataset also contains 65 test images. However, the masks for these images have not been released yet, so we did not use these images in our training. However, once these images' masks are released later on in the competition, it will serve as a good benchmark to test our model closer to the competition due date.

Furthermore, we used techniques of data augmentation such as horizontal / vertical flipping. We also did random cropping of different-sized images. Finally, we did the elastic transformation, which involves local distortion and affine transformation. This allowed us to increase our training data tenfold. However, we made sure to continue to use only images sampled from the original training images for validation.

The only features we used were pixels; the network needs to achieve a mapping of RGB values of the colors of input pixels to a grid of 1's and 0's representing whether or noteach pixel is part of a nucleus.

4 Evaluation Metric

We adapt Kaggle 2018 Data Science Bowl's approach to evaluate our models on the mean average precision at different intersection over union (IoU) thresholds. Given a set of proposed object pixels (A) and a set of true object pixels (B), the IoU is calculated as

$$IoU(A,B) = \frac{A \cap B}{A \cup B}. \tag{1}$$

At each threshold point t, a predicted object is considered a "hit" if its IoU is greater than t. A true positive is counted when a single predicted object matches a ground truth object with an IoU above t, and we define $TP(t)$ as the sum of true positive counts. Similarly, we define $FP(t)$ and $FN(t)$ as the sum of false positive counts and false negative counts, respectively. A precision value is calculated as:

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)}. \tag{2}$$

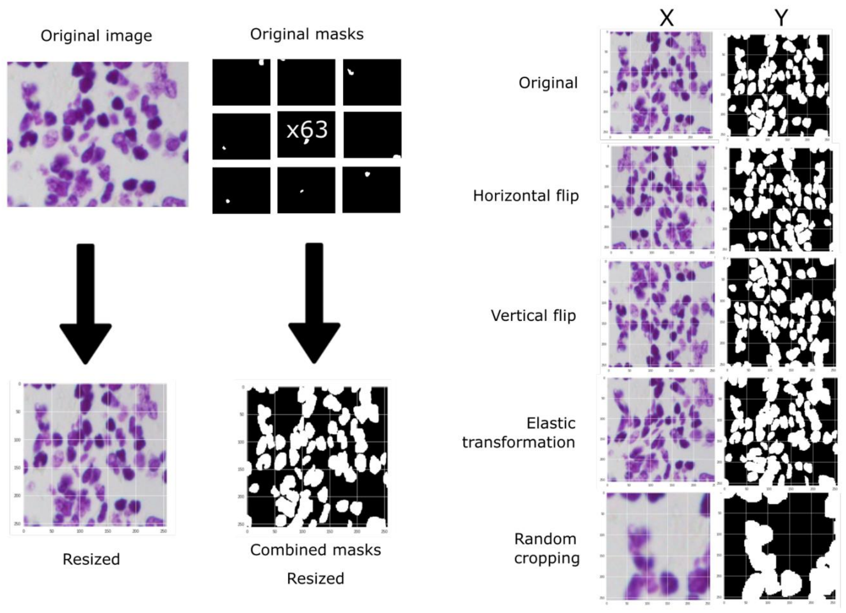

Figure 1: Data processing for DeepCell. Left: the process of mask combination and resizing. Right: examples of various data augmentation techniques used to increase the size of the training set.

With a finite set of thresholds T, the average precision of a single image is calculated as the mean of the precision values at each IoU threshold:

$$\frac{1}{|T|}\sum_{t\in T}\frac{TP(t)}{TP(t)+FP(t)+FN(t)}. \tag{3}$$

5 Methods

5.1 Otsu's Method

We first explore a non-neural network based method, in order to see the the applicability and limitations of a traditional approach. Otsu's method [1] assumes that the image contains two classes of pixels (foreground pixels and background pixels), and it computes a threshold that minimizes the intra-class variance for both classes. For our problem, this method works fine only for a very limited set of pictures (gray pictures mainly). Therefore, we didn't analyze it in depth.

5.2 Modified U-net

We then implement various modified U-net architectures. The U-net is a convolutional neural network architecture that learns segmentation in an end-to-end setting. It is built upon the so-called "fully convolutional network" (FCN) [7], which replaces the usual fully-connected layers in CNN by upsampling layers. This replacement eliminates the restriction of having input images to be of pre-specified sizes. U-net [8] modifies the upsampling part to contain a large number of feature channels, which creates a high-resolution segmentation map. We call the combination of these operators as the expansive path, contradicting with the contracting path. The contracting path follows the typical architecture of a CNN, and it steadily increases the "what" information, but decreases the "where" information. The expansive path is more or less symmetric to the contracting path, and enables the network to precisely locate the objects. As a consequence, the entire neural-network yields a u-shaped architecture (See Figure 2).

To further improve the performance of our neural network, we add batch normalizations and regularizations (See Figure 3 from [10]) to the network. The best model we trained achieved a Mean-IoU of 0.415 in the training set and a Mean-IoU of 0.339 in the public test set of Kaggle.

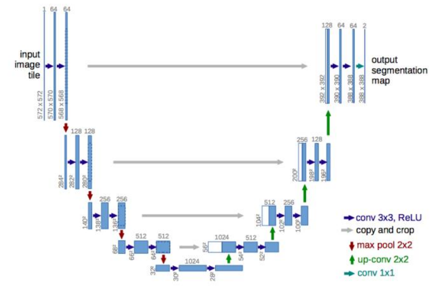

Figure 2: The contracting path of U-net follows the typical architecture of a convolutional network; the expansive path contains upsamplings of the feature maps, concatenations with the correspondingly cropped feature map from the contracting path, and convolutional layers.

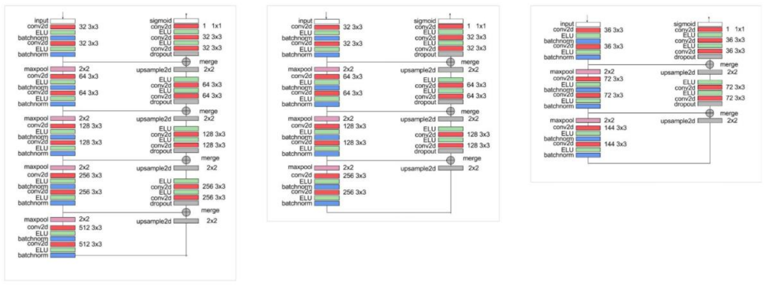

Figure 3: Modified U-net architectures.

6 Experiments/Results/Discussion

The following two graphs represent the improvement achieved by the modified U-Net with respect to the original architecture described in [8] during the training process:

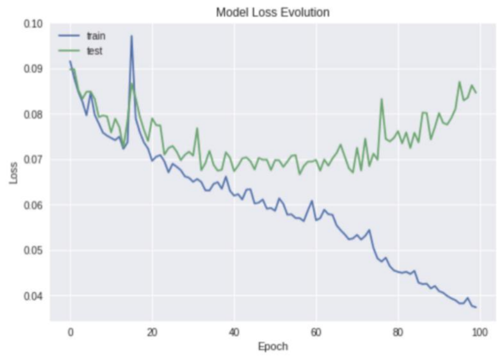

Figure 4: Original U-Net loss evolution

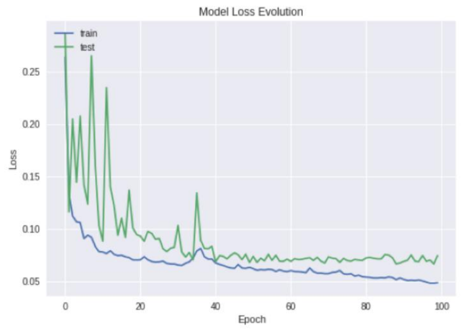

Figure 5: Modified U-net loss evolution

As it can be seen in Figures 4 and 5, the combined action of the dropout layers in the post-merge path and the extra L2 / weight decay regularization term added to all the filters of the convolution layers in the contracting part of the architecture vanishes the variance problem that the original U-Net architecture had. After 100 epochs, the right figure shows how the dev-set error follows closely the training error, which means that the modified architecture is able to generalize well.

A visual inspection of the predicted labels confirms that the modified U-net is capturing the most important features of the images as it is capable of locating successfully the majority of nuclei. Therefore, while not perfect, the modified architecture seems not to have a bias problem.

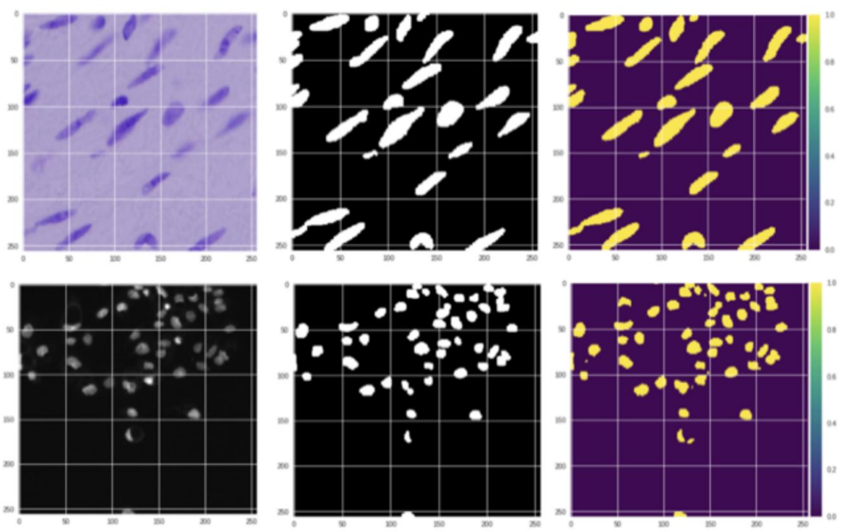
Figure 6: Modified U-net results. Left: Original image, Center: Ground truth mask, Right: Predictions (left, center and right, respectively)

A summary of the most relevant runs with their corresponding Mean IoU score is presented in Table 1.

Model architecture	# epochs	Data Augmentation	Mean IoU score on Test set
UNet (resized to 128x128)	50	None	0.268
UNet	50	None	0.310
UNet with BatchNorm	50	None	0.316
UNet with Dropout = 0.3	100	None	0.304
UNet with Dropout = A0.1	100	None	0.297
UNet with Dropout = A0.1	80	None	0.339
UNet with Dropout = A0.2	80	None	0.339
UNet with BN, Dropout = A0.1	100	Horizontal + Vertical flips	0.355
UNet with BN, Dropout = A0.1	100	H, V + random crops	0.346
UNet with BN, Dropout = A0.2	200	H, V	0.349

NOTE: "A0.1" and "A0.2" are specific configuration of dropout weights: 0.4, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.4, 0.4 and 0.2, 0.1, 0.2, 0.1, 0.2, 0.1, 0.2, 0.1, 0.2 respectively

Table 1: Selected modified U-net runs, with their results on the mean IoU for each. All the models resized the image to 256x256, except for one which is noted. The best run is colored in gray.

7 Conclusion/Future Work

In this project, we examined an end-to-end approach for image segmentation using few data, and with a relatively low resolution (500x500 max). We implemented a modified version of the U-net architecture developed by Ronneberger et al. in [8]. As seen through the results shown in the report, the best solution to the task of nuclei detection is a customized U-net architecture that receives an augmented dataset (with horizontal-vertical flips and elastic transformations [9]) and has the following added features: batch normalization layers after the non-linearities, weight decay in the filters of the contracting part of the architecture and dropout layers in the post-merge path.

Building upon this work and with the aim of improving the accuracy of the predicted nuclei boundaries, we plan to substitute the contracting part of our best architecture with a VGG11 Encoder, pre-trained on ImageNet as suggested by the authors of TernausNet [11]. Finally, we intend to use ensemble methods to improve the score by running multiple neural networks in parallel.

8 Contributions

Yiguang Zhang conducted research on articles which presented different architectures, explaining their pros and cons and helped the team come to the decision on which architectures to try out. She also searched for optimal hyper-parameters. Ashwin Ramaswami implemented data augmentation on the training set and created a reusable python library, which could then be imported by the scripts that actually trained the models. He also ran several model training runs on an AWS GPU. Cristian Bartolome implemented U-net through Google's Colaboratory. Furthermore, he assisted Yiguang in the literature review.

The model, with an explanation of all the steps of the process, can be found here: Google Colaboratory Notebook. Data download, processing, and the data augmentation Python library can be found here: Laughing-memory-util on Github. The final code we used to train models on AWS instances, as well as submission results, model files, copies of the code, and graphs for each model configuration we tried can be found on this repository: Laughing-memory on Github

References

[1] N. Otsu, *A threshold selection method from gray level histograms*, IEEE Trans. Systems Man Cybernet., 9 (1979), pp. 62-66.

[2] Finkel, Raphael A., and Jon Louis Bentley. "Quad trees a data structure for retrieval on composite keys." Acta informatica 4.1 (1974): 1-9.

[3] Rosenfeld, Azriel, and Russell C. Smith. "Thresholding using relaxation." IEEE Transactions on Pattern Analysis and Machine Intelligence 5 (1981): 598-606.

[4] Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural net- works segment neuronal membranes in electron microscopy images. In: NIPS. pp. 2852–2860 (2012)

[5] Kaggle, Inc. *2018 Data Science Bowl.* Retrieved from https://www.kaggle.com/c/data-science-bowl-2018.

[6] A. Rosenfeld, P. De la. Torre, *Histogram concavity analysis as an aid in threshold selection*, IEEE Trans. Systems Man Cybernet., 13 (1983), pp. 231-235.

[7] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation (2014), arXiv:1411.4038 [cs.CV]

[8] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. *"U-net: Convolutional networks for biomedical image segmentation."* Springer, Cham, 2015.

[9] Simard, Steinkraus and Platt, *"Best Practices for Convolutional Neural Networks applied to Visual Document Analysis"*, 2003.

[10] Kaggle, Inc. Retrieved from *Dstl Satellite Imagery Competition, 1st Place Winner's Interview: Kyle Lee.* http://blog.kaggle.com/2017/04/26/dstl-satellite-imagery-competition-1st-place-winners-interview-kyle-lee/.

[11] https://github.com/ternaus/TernausNet

Code / Libraries used:

- Keras (MIT License): https://github.com/keras-team/keras
- Matplotlib: https://github.com/matplotlib/matplotlib
- NumPy: https://github.com/numpy/numpy
- OpenCV (BSD License): https://github.com/opencv/opencv
- Scikit-Image: https://github.com/scikit-image/scikit-image
- SciPy: https://github.com/scipy/scipy
- Tensorflow (Apache License 2.0): https://github.com/tensorflow/tensorflow
- TQDM (Mozilla Public License): https://github.com/tqdm/tqdm