Instance Segmentation: Cell Nuclei

Shivam Kulkarni

skulkarni17@hawk.iit.edu

Illinois Institute of Technology

May 7, 2021

Abstract

The first phase in the quantitative analysis of imaging data for biological and biomedical applications is always segmenting the nuclei of cells in microscopy images. Many bioimage analysis tools can segment nuclei in images, but each experiment requires a different selection and configuration. The 2018 Data Science Bowl attracted 3,891 teams from all over the world to compete in the first attempt to create a segmentation method that could be applied to any two-dimensional light microscopy image of stained nuclei through experiments without requiring human intervention. The challenge's top participants were successful in this mission, creating deep-learning-based models that correctly defined cell nuclei across a wide range of image types and experimental conditions without the need for manual segmentation adjustments. This is a significant step forward in the development of configuration-free bioimage analysis software tools.

1 Introduction

Identifying a cell's nuclei is often the first step in nuclear/cellular research. Identifying nuclei allows researchers to identify individual cell in a sample. Consequently, the researchers can pinpoint the underlying biological process by measuring how cells react to various treatments. Manually detecting nuclei is an arduous effort. Researchers would rather spend time doing research that will lead to significant discoveries. Furthermore, manually detecting nuclei is a time-consuming and error-prone process especially because of the large amount of collected image data [1,2]. Hence, we need a machine

learning technique that can detect the nuclei accurately and expedite medical discoveries. An accurate machine learning technique has the potential to save researchers' time. Having said that, to identify a range of nuclei across varied conditions, we need a large labeled dataset of divergent images containing nuclei.

Choosing nuclei segmentation methods is not a simple task for non-expert users in daily biology labs. The majority of current user-friendly bioimage analysis tools [3, 4, 5] use traditional segmentation algorithms like thresholding [6], watershed [7], or active contours [8] to classify nuclei. These must be customized for each study to account for various microscopy modalities, scales, and experimental conditions, which sometimes necessitates a high level of expertise to choose the right algorithm and tweak its parameters. Advanced users can find the option overwhelming, as hundreds of papers are published each year proposing new methods for cell and nucleus segmentation. Even under carefully managed experimental conditions, no single parameter setting can correctly segment all images, since traditional algorithms can fail to adjust to the heterogeneity of biological samples or be sensitive to technical artifacts [9, 10, 11]. Overall, this situation slows the pace of research and makes it difficult for biological laboratories to implement imaging technology due to a lack of time and expertise required.

We investigate the possibility of developing a segmentation model that can automatically classify the nucleus of cells in a variety of stained two-dimensional (2D) light microscopy images without the need for human intervention. Future robotic microscopes may use this model to locate and count nuclei in images in real time, regardless of cell type, staining type, magnification, or experimental variations, facilitating a broad range of biological applications. Biologists' expertise and study would benefit from a single trained model that worked across instruments, stains, and cell types. Traditional algorithms for detecting nuclei in microscopy images use very similar computational methods, with a few differences in parameters or configurations. Our aim was to see if any modern solutions, such as large capacity deep-learning models, could provide a unified solution that didn't require manual configuration.

2 Problem statement

2.1 Instance Segmentation

Researchers in the field of computer vision use a variety of techniques. The best approach to use depends on the problem you're trying to solve. Here are some of the most common techniques used.

Object Detection: Object detection is a type of computer vision and image processing technology. Its goal is to find objects in a picture and draw boxes around them.

Semantic Segmentation: Semantic segmentation is a technique for detecting the object category that each pixel belongs to. The model must be aware of all object categories (labels).

Instance Segmentation: Instance segmentation is similar to semantic segmentation, but it goes a step further by identifying the object instance that each pixel belongs to. In contrast to semantic segmentation, the key difference being that it distinguishes between two objects with the same labels.

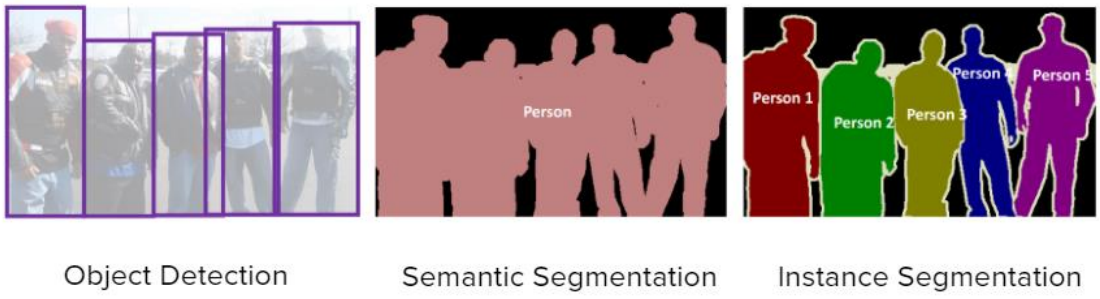

Figure 1: source: https://towardsdatascience.com

In figure 1, for object detection task we can see there are boxes drawn around the people. For the sematic segmentation task, all the pixels that belong to class person are colored pink and all the pixels that belong to class not-a-person are colored black. For the instance segmentation task, all the people in the image are colored with different showing that we know there are five people in the image.

We have an instance segmentation task in this project. We, however, use a technique that isn't commonly used for segmentation. As a consequence, we must control the input and output in order to achieve the desired effects.

2.2 Problem

The aim of this project is to automate the process of identifying nuclei from images of samples observed under a microscope. Multiple nuclei can be found in the majority of the images in the dataset. This is an instance segmentation task in which the model's input is a single image and the output is nuclei masks in the cell. Figure 2 depicts an example of input and output. On the left, we have an input image, and, on the right, we have the output which is created by stacking the masks onto another.

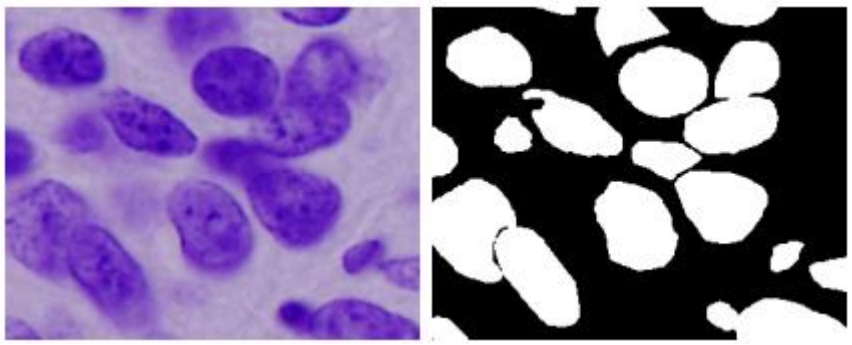

Figure 2: Example of an input image and an output image

2.3 Data

This dataset is provided by Booz Allen Hamilton [12]. This dataset contains a large number of segmented nuclei images. The images were acquired under a variety of conditions and vary in the cell type, magnification, and imaging modality (brightfield vs. fluorescence). The dataset is designed to challenge an algorithm's ability to generalize across these variations.

Each image is represented by an associated imageId. Files belonging to an image are contained in a folder with this imageId. Within this folder are two subfolders:

- images: contains the image file.
- masks: contains the segmented masks of each nucleus. Each mask contains one nucleus. Masks are not allowed to overlap (no pixel belongs to two masks).

An image from the images folder can have multiple nuclei. Hence, the masks folder may contain multiple corresponding masks for that nuclei. For example, figure 3 from the images folder has 9 nuclei. Hence, its 9 corresponding masks are present in the masks folder which are shown in figure 4.

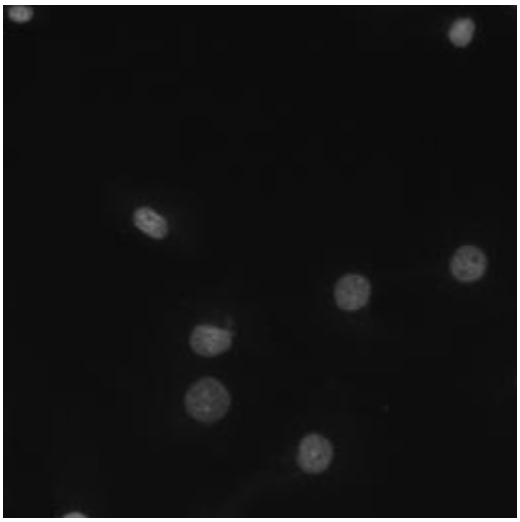

Figure 3: Image from images folder

Figure 4: Masks of the image in Figure 3

2.4 Performance Metric

The performance of the model was evaluated using dice coefficient. Dice coefficient is 2 times the Area of Overlap over the total number of pixels in both images.

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

For example, let's say that we have two images and we want to check how similar they are. They both have 100 pixels each. Hence, the total number of pixels for both images combined is 200. Now, let's consider a case where there is no overlap at all. In that case, dice coefficient is 0/200 that is 0. For another scenario, let's consider a case where the overlap between the two images is 95%. In that case, dice coefficient is 2 * 95 / 200 = 95. High dice coefficient indicates high performance of the model.

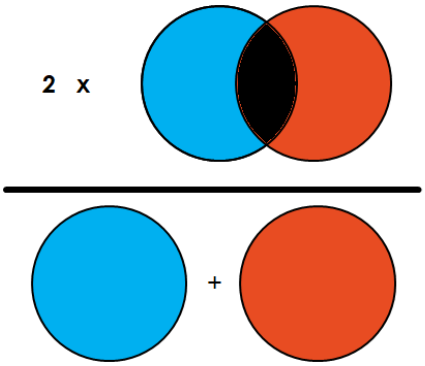

Figure 5: Dice coefficient

3 Model

U-net architecture is used for this problem. U-net architecture was developed specially for bio Medical Image Segmentation task. The architecture contains two paths. First path is the contraction path, also called as the encode, which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path, also called as the decoder, which is used to enable precise localization using transposed convolutions. Hence, it is an end-to-end fully convolutional network (FCN), i.e. it only contains convolutional layers. One of the advantages of using this model is that it is easy to implement using Keras. Also, it is tailor-made for problems such as the problem we are solving. Having said that this model has some disadvantages as well. U-net does not perform instance segmentation. Hence, In the last step, all the masks are combined to compare with the expected result. Furthermore, U-net requires square input images. Hence, all the images are resized to 256x256 images. The last disadvantage of U-net tends to overfit on training data. Dropouts are used at every stage of the model to combat this issue.

Initially, the input image size is 256x256. There are two convolutional layers of size 16 at the first stage. After these two convolutional operations, the matrix size is 256x256x16. We are at c1. The kernel used for convolution is 3x3 and the activation function is ReLU. Using maxpooling on c1, p1 is obtained. Maxpool kernel is 2x2 and the stride is 2 so 256x256x16 matrix becomes 128x128x16. Further, there are two convolution layers, this time of size 32. After the convolution operations, the matrix size is 128x128x32. This is repeated for three more stages until we reach the bottom of the image and then we start going up. Now, upsampling is used as we go from c5 to u6,

the size goes from 16x16x256 to 32x32x128. Now, we have u6 with dimensions 32x32x128 and on the left-hand side of the image, there is corresponding c4 with the same dimensions. u6 and c4 are concatenated to get matrix of dimensions 32x32x256. Two convolution layers are used here, and the process continues. The final convolution is 1x1 and the final output is of size 256x256x1. Dropouts are added between the two convolution layers at every stage. Padding = same is set meaning, means the input image ought to have zero padding so that the output in convolution doesn't differ in size as input. In other words, we add a n-pixel borders of zero to tell that don't reduce the dimension and have dimensions same as input.

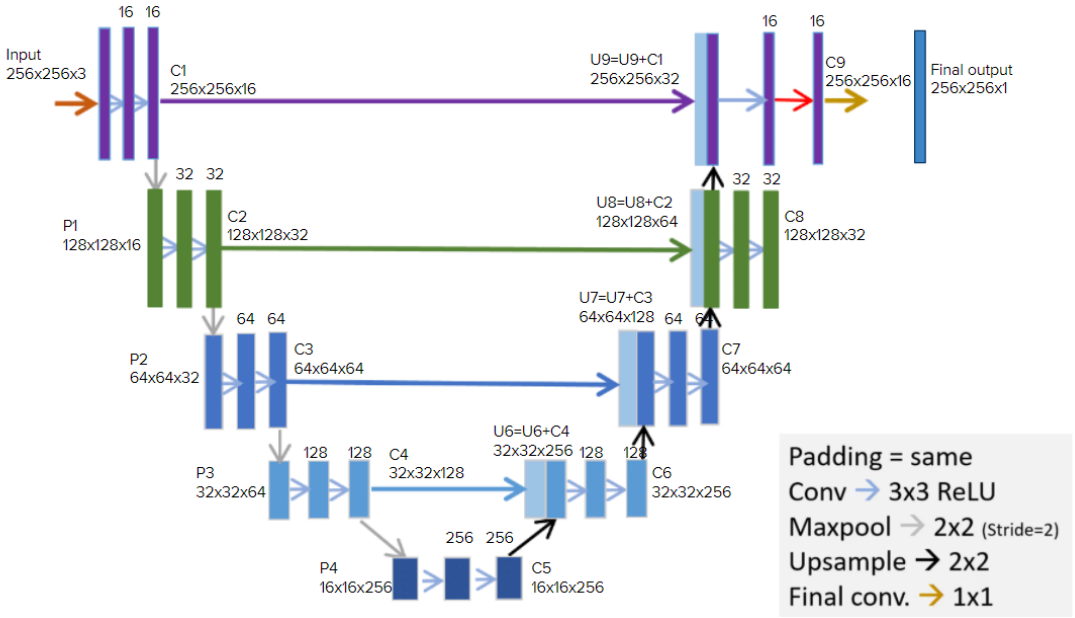

Figure 6: U-Net architecture used for this problem

4 Results

20% of the training data is used as validation data. The model is trained on the rest of the training data and validated on the validation set. The hope is the model produces equally good results on training as well as the validation data. The ideal model should perform well on the unseen test dataset as well. Even though this is an instance segmentation task, we use U-net model. Hence, we will get masks for every nucleus in the image as the output. We then, merge all the masks to create the final output. Pixel wise comparison for the actual and the predicted masks can give us the dice coefficient.

The U-net model produced 0.86 dice coefficient on both training and validation data. When the model is tested on unseen test dataset, it didn't

produce promising results. It was able to achieve less than 0.6 dice coefficient.

Figure 7 is an example of prediction. On the left-hand side, an input is given and on the right-hand side the predicted masks are shown.

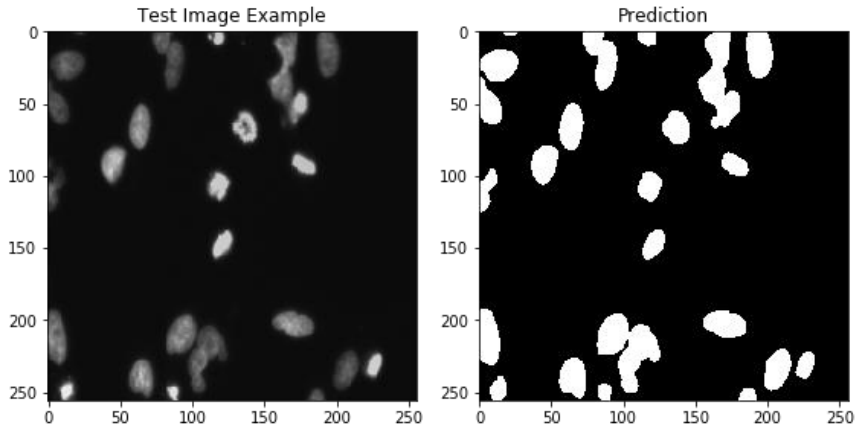

Figure 7: Prediction results example

5 Conclusion and Future Work

The model performing equally well on training dataset and validation is a sign that we are able to build a generalized model. Having said that, it's performance on test data may suggest otherwise. The reason for not producing comparable results could be that the test images are different than what we have in the training dataset. There are a few potential improvements to this model. Implementing mask R-CNN model could yield better results as the winning team of the competition used this model. Having said that, the winning team's model also performed well on training dataset but didn't perform particularly well on the test dataset. Furthermore, synthetic data can be generated to train the model as the training dataset consisted of only 670 observations. This approach doesn't guarantee improvement in performance, but it is worth trying this. Another approach to improve results could be adding multiple models and taking aggregated results. Ensemble methods can produce better results than a single model and could help generalize the overall model as well.

6 Discussion about Presentation

This project was presented in class on April 29, 2021 as part of CS 584 project of group 17.

One of the remarks from the professor was that this is not an instance segmentation task as we get masks of all the nuclei and then we stack them

one onto another. This is accurate but this was the only way this problem could be solved using U-net architecture. As discussed in the section 5, mask R-CNN could be a better idea for this problem statement. Because of the time constraint, further investigation was bot done about this.

This presentation is available at this link:

https://docs.google.com/presentation/d/1ifjx0XCfIFJwsuwGpMMqXN5zazbdQinM5pZgmifLoxw/edit?usp=sharing

Github repo for the code is following:

https://github.com/shivamkulkarni/CS584Project

7 References

[1] Sommer C, Gerlich DW. Machine learning in cell biology teaching computers to recognize phenotypes. J Cell Sci. 2013; 126(24):5529–39.

[2] Kothari S, Phan JH, Stokes TH, Wang MD. Pathology imaging informatics for quantitative analysis of whole-slide images. J Am Med Inform Assoc. 2013; 20(6):1099–108.

[3] Schindelin, J. et al. Fiji: an open-source platform for biological image analysis. Nat. Methods 9, 676–682 (2012)

[4] McQuin, C. et al. CellProfiler 3.0: next-generation image processing for biology. PLoS Biol. 16, e2005970 (2018)

[5] Wiesmann, V. et al. Review of free software tools for image analysis of fluorescence cell micrographs. J. Microsc. 257, 39–53 (2015)

[6] Otsu, N. A threshold selection method from Gray-level histograms. IEEE Trans. Syst. Man. Cybern. 9, 62–66 (1979)

[7] Malpica, N. et al. Applying watershed algorithms to the segmentation of clustered nuclei. Cytom. A 28, 289–297 (1998)

[8] Chan, T. F. & Vese, L. A. Active contours without edges. IEEE Trans. Image Process. 10, 266–277 (2001)

[9] Dima, A. A. et al. Comparison of segmentation algorithms for fluorescence microscopy images of cells. Cytom. A 79, 545–559 (2011)

[10] Meijering, E. Cell segmentation: 50 years down the road. IEEE Signal Process. Mag. 29, 140–145 (2012)

[11] Ulman, V. et al. An objective comparison of cell-tracking algorithms. Nat. Methods 14, 1141–1152 (2017)

[12] https://www.kaggle.com/c/data-science-bowl-2018