COMP 540 Spring 2018 Term Project Final Report

--- Automate Nuclei Detection

Rice University
Yunyi Lin (yl146) & Pei Zeng (pz14)

1 Introduction

1.1 Competition Goal

The goal of this competition is to create an algorithm to automate nuclei detection for a range of nuclei across varied conditions.

1.2 Platform and Python Packages

The system we use is macOS Version 10.12.6 and under Anaconda environment python 3.6 we installed the following Python Packages:

- Keras (version 2.1.4),
- matplotlib (version 2.1.2).
- OpenCV (version 3.3.1).
- pandas (version 0.22.0).
- seaborn (version 0.8.1).
- Sk-learn (version 0.19.1).
- skimage(version 0.13.0)
- TensorFlow (version 1.5.0).

We also use the online GPU to train our models, the platform we used is Google Colaboratory [4].

1.3 Workflow

At the beginning, we did exploration on the given data set. We tried to classify the data by some features like its size, color, shape and number of nuclei. By investigating on these features, we attempted to identify the potential noise. After exploration work on data, we used the Naïve model and U-Net models to solve the problem. Naïve model is designed to solve the problem by implementing the idea of traditional computer vision. U-NET model is a modified version of fully connected network, which is considered one of standard architectures for image classification tasks. Our U-Net model is based on Raoul's built software on Kaggle [7] and Naïve model is based on Bailey, S's work on Kaggle [3].

2 Exploratory Data Analysis

2.1 Various Image Shapes

By checking the stage1 data we found there are 65 images in the test set and 670 images in the training set [10]. We can see from *Figure1* that there are 16 different input image sizes in total, with 11 different types of height and 15 different types of width respectively. Moreover, there are 7 shapes only appeared in the test set and not in the training set.

Various image sizes and shapes made us considering resizing the image into a uniform size during the preprocessing stage. Since there are more than half of the images in the training set are of shape 256*256 according to *Figure1*, we first resizing all the images into this shape. However, we figured out later that larger sizes may lead to better results so we also tried resizing images into 384*384 and 512*512. In our final model, we choose to reshape all test/train images into 384*384.

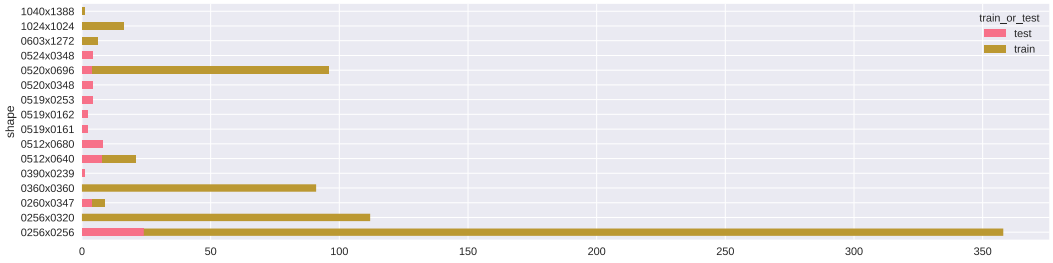


Figure1: Shape distribution of train/test (stage1) images [10]

2.2 Three Different Microscopy Types

According to Sharhan, M. [9], we find out there are 3 different modalities (microscopy types) in the dataset: Staining slides, fluorescent images as well as bright-filed images (as showed in *Figure2*)

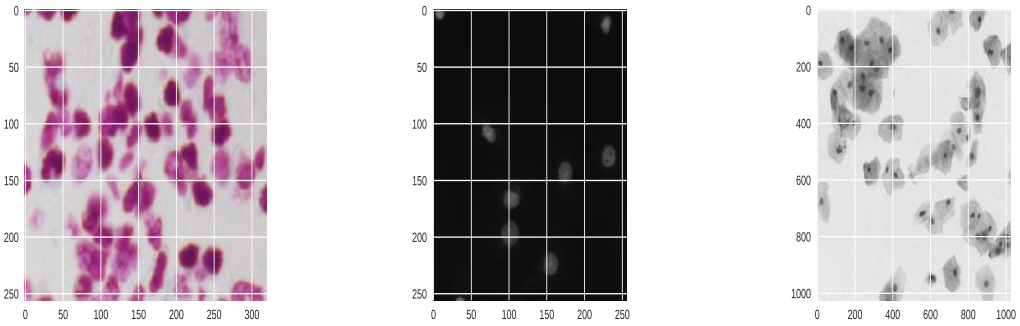


Figure2: Examples of three different microscopy types, Staining slides (left), Fluorescent images (middle) and Bright-filed images (right) [9]

According to *Figure2* we can see that the staining slides are colored images with light background and containing more violet pixels. While fluorescent images and bright-field images are both black and white images, with dark background for fluorescent images have and light back ground for bright-filed images.

We manually defined a function to classifying these three modalities [9]. First of all, we define the violet pixels simply as red channel & blue channel > green channel and if image contain violet pixel it will be classified as staining slide. Moreover, mean intensity is used to distinguish fluorescent images from bright-field images for that fluorescent images has dark background

thus would have low mean intensity. Table 1 showed the count as well as the percentage of these three different microscopy types in both stage1 test and training sets [6].

Table 1: Count and percentage of three different microscopy types in both stage1 test and training sets.

			Different Microscopy types		
			Staining Slides	Fluorescent Images	Bright-field Images
Data Source	Training Set	Count:	108	546	16
		Percentage:	16.12%	81.49%	2.39%
	Test Set	Count:	12	53	0
		Percentage:	18.46%	81.54%	0%

By analyzing different microscopy types, we know that staining slides might lead to potential noises such as blurring and low resolution, while bright-filed images might have low contract issue. We want to turn every image as close as to fluorescent images based on our analysis.

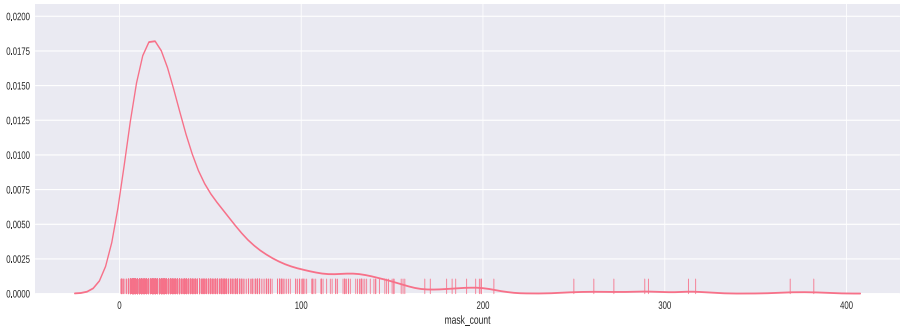
Therefore, it's necessary for us to normalize the images during preprocessing stage. We tried converting all images into grayscale images in naïve model to increase contrast for background distinguishing. While in our U-net model, we first normalizing the images and mask images by converting pixel values from [0,255] to [0,1] to eliminating violet pixels and inverting the images if the mean intensity is larger than a cutoff value to invert all images into dark backgrounds.

2.3 Nuclei: number and size

We can see from *Figure3*(a) that most images with nucleus counts around 10, and from *Figure 3*(b) we can see that the size of nucleus are not very large (mostly within the range of 10*10 to 80*60) and tend to be smaller in height.

According to these information, in post-processing stage, we would remove too small mask sizes as misdetection. Moreover, detection of too many nucleus in one image would also raise our attention.

a)



b)

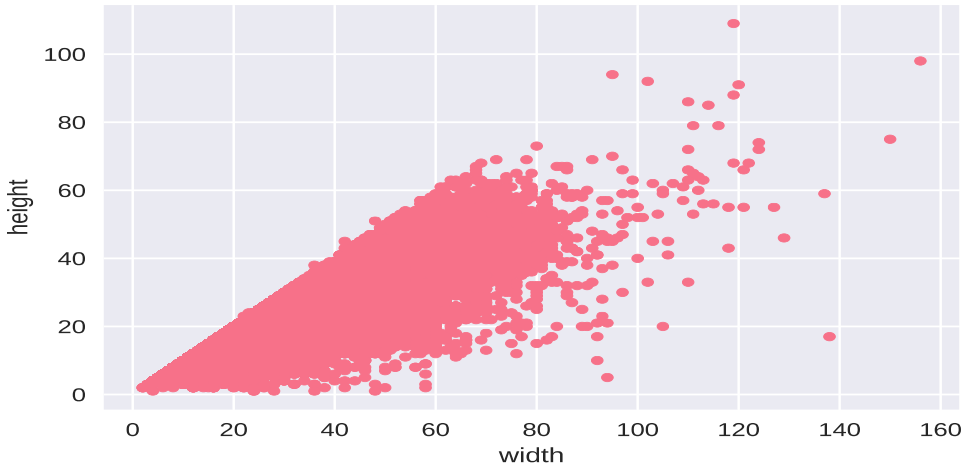


Figure3: a) Nucleus counts distribution in all training images (upper), b) Width and height distribution of all nuclei in training set(below)[10]

3 Models

3.1 Naïve Model

3.1.1 Introduction to the model

The underlying idea of the naïve model is to assume the data fall into two classes: nuclei and the background, then we can simply set up a reasonable threshold to mask out the background pixels [3].

The key idea of this model is how to select threshold wisely. We used the "Otsu" methods to set up the threshold here, which assuming the image histogram as two additive distributions: one distribution composed of background and the other thus represents for nuclei. Under this assumption, the "Otsu" method would set a threshold which distinguish two separate distributions. (See *Figure4* below for example).

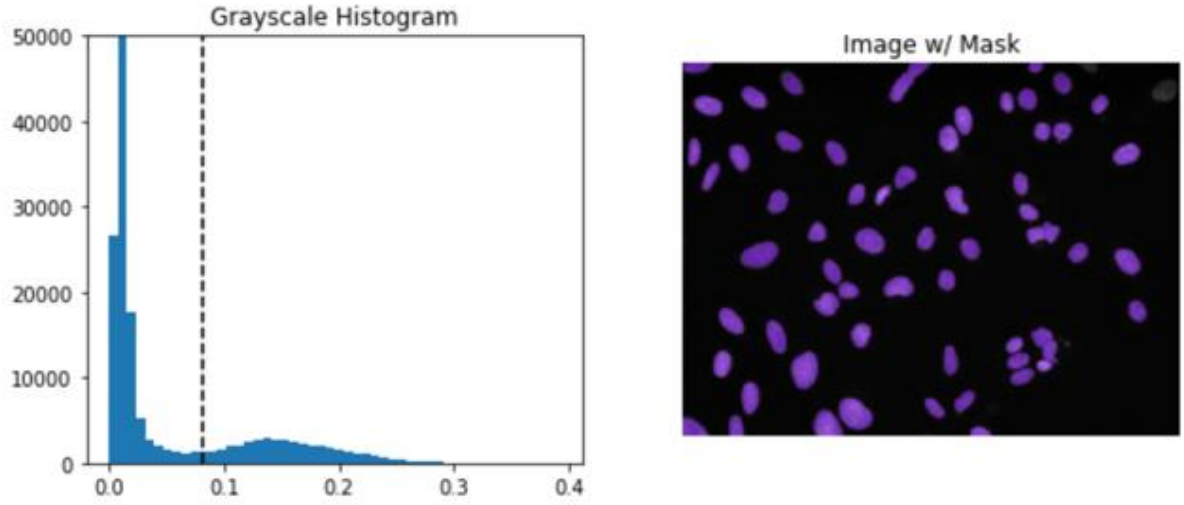


Figure4: An example of how the "Otsu" Method of choosing threshold works [3]

3.1.2 Workflow

Preprocessing: As discussed before, the preprocessing stage for this model is pretty easy just converting all images into grayscale even without resizing them into the same shape (which might affect the decision of threshold choice).

Figure5 below is an example training image (id = 193) before and after preprocessing, we can see that by converting original image into grayscale it solved the low contract issue.

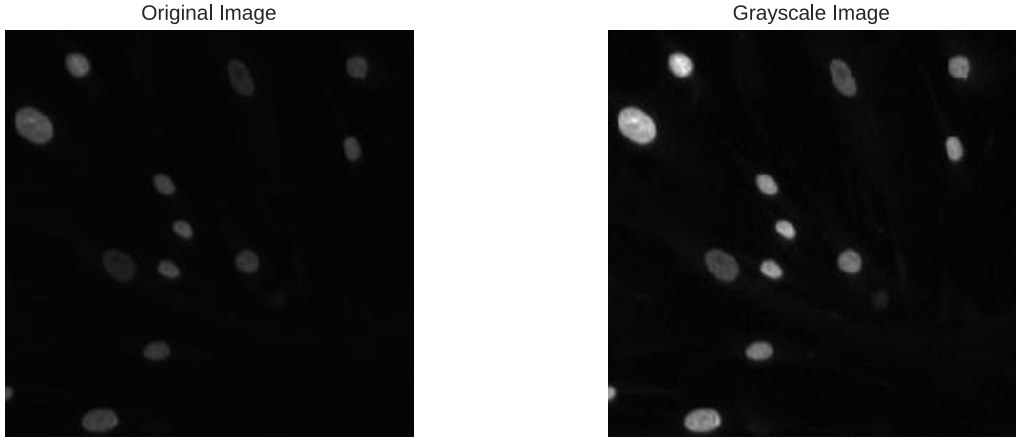


Figure5: One sample image from the training set (id=193) before and after Preprocessing

Applying model: Using "Otsu" method as discussed above to setup threshold and deriving individual masks for connected object after masking out the background. See *Figure6* below for sample image (id = 193).

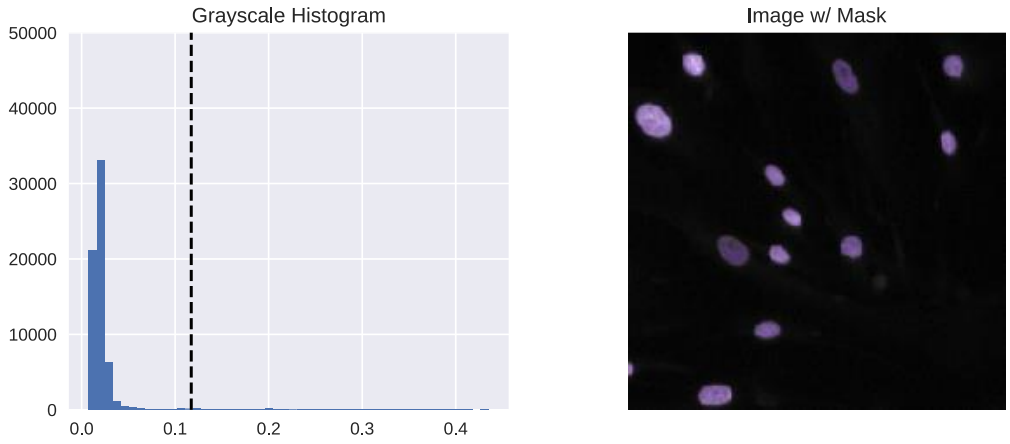


Figure6: Computing the grayscale histogram and masking out background

Post-processing: Setting labels with too small nucleus size (less then $\sqrt{20} * \sqrt{20}$) back to 0 and using mask erosion method to separate two nucleuses that was considered as one label. Converting each labeled object to run line encoding [5].

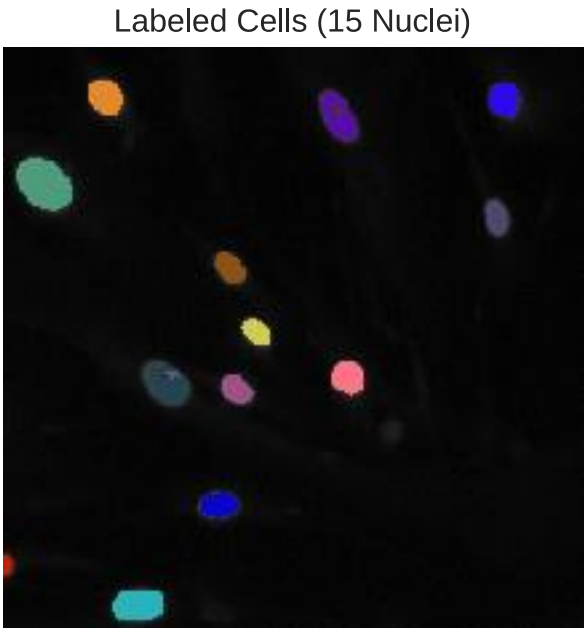


Figure7: Labeled nuclei using different colors, 15 nuclei detected without post-processing.

We can see from *Figure7* that without post-processing there are 15 nuclei detected without post-processing. However, after remove two labels with too small size and gave us exactly 13 nuclei left.

3.2 U-Net Model

3.2.1 Introduction to the model (A discussion of how your final model makes decisions. Why it works as well as it does/ why it doesn't)

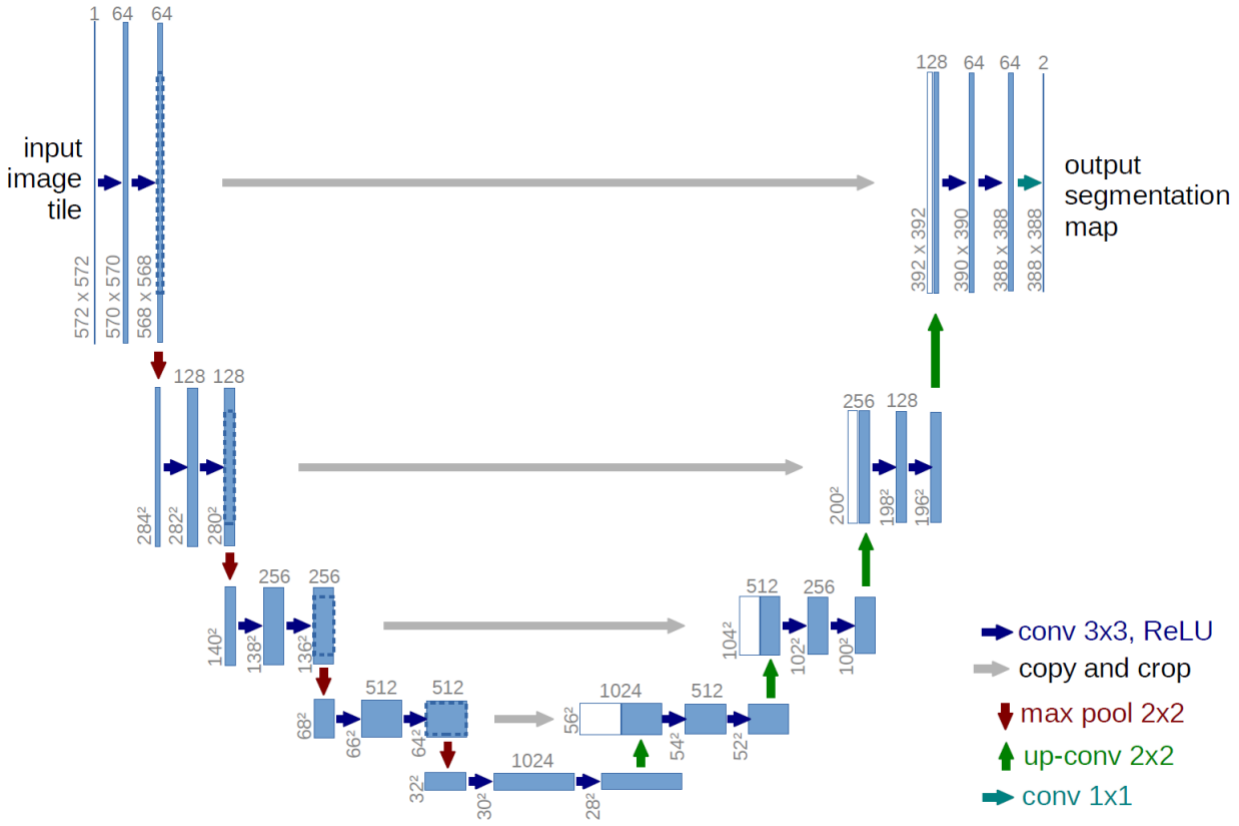


Figure8: U-net architecture [8]

The basic idea of U-Net [8] is to improve a common contracting network by successive layers, where pooling operators are replaced by up-sampling operators. Thus, the resolution of the output could be increased by layers. For the up-sampling part, the context information is transmitted by a large number of feature channels to higher resolution layers. Hence, a u-shaped architecture could be created, since the expansive path is approximately symmetric to the contracting path. As shown in *Figure8* above, each blue box corresponds to a multi-channel feature map with its channels denoted on top of the box. The x-y-size is provided at the left bottom of the box. White boxes represent copied feature maps. The arrows denote the different operations [8].

3.2.2 Workflow

Preprocessing:
Resizing training images to 512/384/256*512/384/256*3 and mask images to 512/384/256*512/384/256*1.

Normalizing the images and mask images by converting pixel values from [0,255] to [0,1] and inverting the images to solving for low contract issue [6].

Training the model: Calculating the score metric based on the method discussed by Bailey,S. [2] Setting Cross-validation Fold = 10 and feeding the data to U-net model.

Post-processing: Making test predictions, resizing the predicted masks to original size and converting each labeled object to run line encoding. [4]

3.3 U-net Model Improvement

3.3.1 Improvement to control for overfitting

Early Stop*:* We saved the best model after every epoch and stopped the model training if the accuracy of validation set doesn't decrease for more than 3 continuous epochs. The final model was trained for 30 epochs.

Dropout: Including the dropout technique in our model to regularize neural networks by randomly setting some features to zero during the forward pass. The dropout rate in our final model is set to be 0.15.

Update learning rate periodically: Instead of using a constant learning rate along all epochs, we uploaded the learning rate that decreasing by a certain percent after several epochs to ensure the model won't be overfitted. In our final model, the learning rate starts from 0.003, and after every 5 epochs, we reduce the learning rate of 35%.

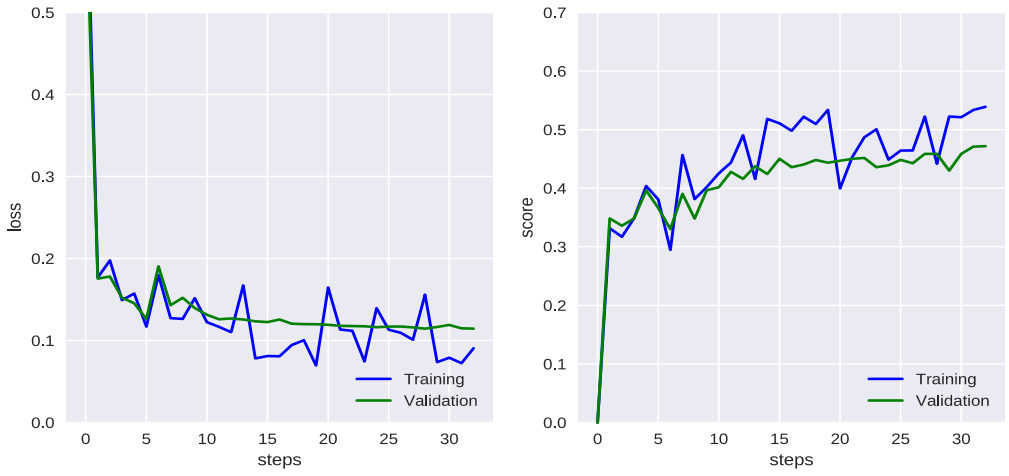


Figure9: Training loss and training score of our final model.

3.3.2 Other Improvements

We changed the way of calculating the IOU score metric, instead of just using k-means [1] we changed the calculation method as discussed by Bailey,S. [2]: We computed the score metric by segmenting true/predicted masks corresponding to the training data into different objects and then computing all intersections between the objects of true and predicted masks. Then assigned the true objects to predicted objects staring from the largest intersections, until there is no overlap between true objects and predicted objects. We then computed for each true/predicted mask pair their corresponding intersection over union ratio. Moreover, some threshold was set to count the true/predicted pairs and to compute the precision as showed in our final training/validation score.

Instead of using the keras package included in Tensorflow[1], we build our own U-net model under tensor flow which change the ReLU layer into a combination of leaky ReLU and ReLU layers.

4 Results

We submitted 13 times on Kaggle in stage1 and 4 times in stage2. Our highest score in stage 1 is 0.340 and our highest score in stage2 is 0.430. However, from the two score we chose for submission, our highest score submitted in stage2 is 0.418. We ranked 250/3634 which is top 7% in the stage 2 leaderboard.

From *Figure10*, we can see that we started to submit our results from 03/11/2018 and have improved from 0.22 to 0.430 for the overall accuracy on the leaderboard.

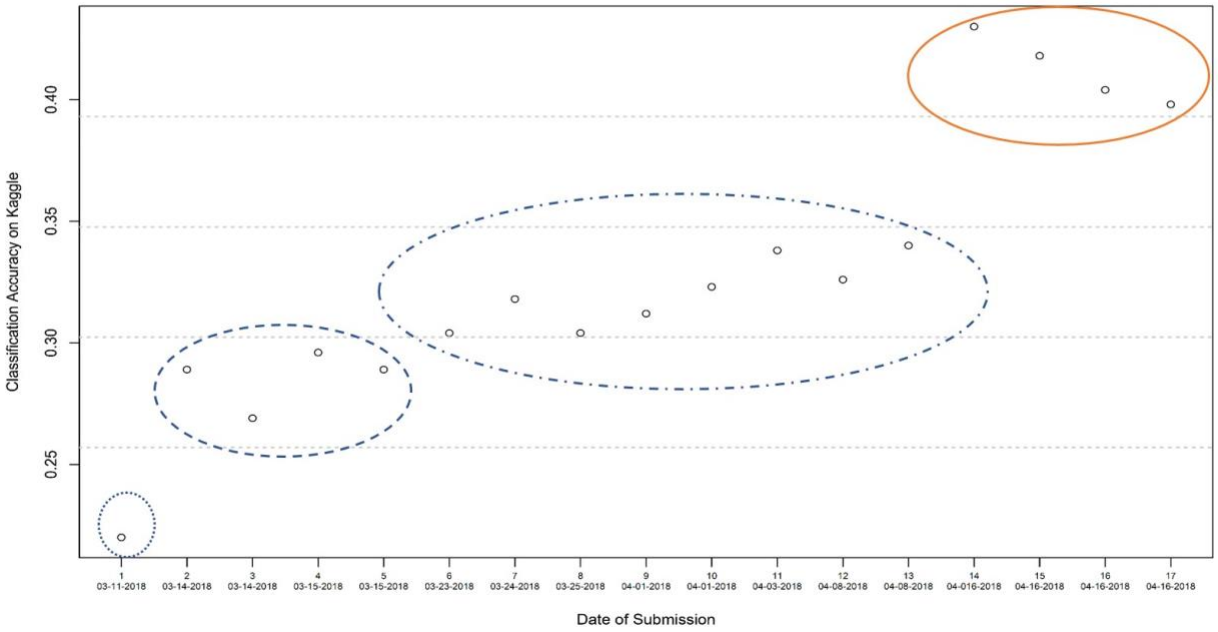


Figure10: Submission Timeline on Kaggle

- The single point circled by round dots is submission of the Naïve model (0.22) in stage1.
- The cluster of points circled in dash are submissions of the U-net model without improvements in stage1. The baseline score for that is 0.269 and the best score is 0.296.
- The cluster of points circled in dash dot are submissions of the U-net model with improvements in stage1. The baseline score for that is 0.304 and the best score is 0.340.
- The cluster of points circled in solid line are submissions of the U-net model with improvements in stage2 the baseline score for that is 0.398 and the best score is 0.430.

5 Discussion

We can see the comparison between stage1 test accuracy between different models from Table 2 below.

Table 2: Model Accuracy Comparison (A summary table of all algorithm/algorithm combinations tried, together with their performance.)

		Test Set Accuracy	
		Baseline Score	Best Score
Models	Naïve Model	0.22	0.22
	U-net Model w/o Improvements	0.269	0.296
	U-net Model w/ Improvements	0.304	0.340

The Naïve model failed in prediction accuracy for two reasons. First, sometimes background would have smaller percentage compared to nucleus. However, the threshold chosen by "Otsu" Method assume the distribution of more pixels are considered as background and may cause problems in particular case (See *Figure11*). Moreover, the Naïve model only masks the nucleus out from background and as discussed before, this method always leads to too small segments. Although we can check for some misclassification through post-processing, but when the nucleus are really large in size, this method would also failed (See *Figure11*). *Figure11* below is a perfect example for these two situations discussed below.

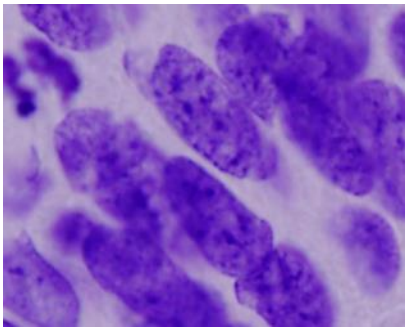


Figure11: An example when the Naïve Model would fail to correctly detect the nucleus [3]

On the other hand, U-net model is a powerful architecture for image classification when we need to segment areas of images by class. It has three significant advantages compared to the Naïve model based on past researches [8]: 1) It's an end to end processing, so it can be easily scaled to have multiple classes. 2) It can be applied to relatively small training sets. 3) It has relatively high predicting accuracy given proper training.

Moreover, we believe the increase in accuracy after improvements added to the U-net model is basically because we upgrade the method of calculating IOU score metrics. Moreover, by adding the control of overfitting methods, we can reach the sweet spot more easily compared to the unimproved model. *Figure12* showed a pretty good example of how the predicted test masks looks like for U-net Model.

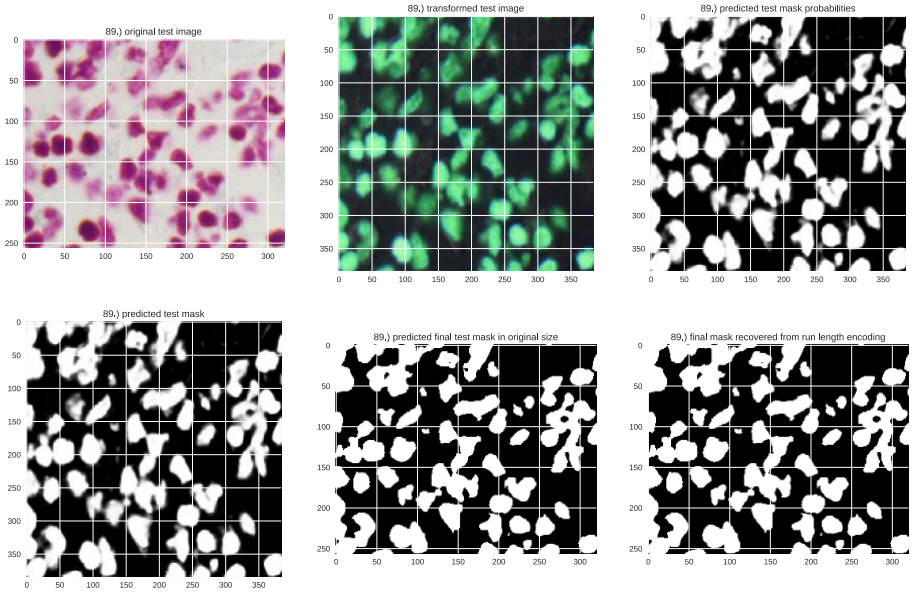


Figure12: Good Test Prediction example using U-net Model w/ Improvement on stage2 test set (id = 89)

However, when looking at the test predictions on stage2 test set, we found there would still be some potential problems for our final model. *Figure13* shows one of the examples. These kinds of images which neurons are linked together via long axons and only really small part of the detecting masks are cell bodies that contain nuclei are new in the stage2 data and by using our model, we cannot distinguish the differences between axons and nuclei. Future works still need to be done to improve the model for more global situations like this. We might have solve this problem by improve the score metrics again or do more data manipulating during the pre/post-processing stages.

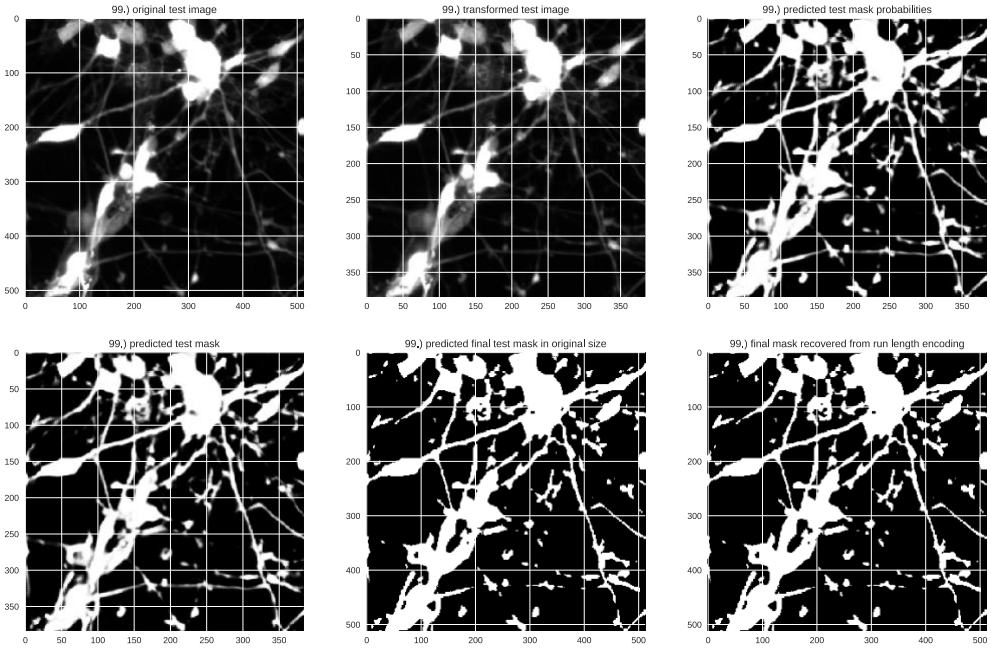


Figure13: Potential problem showed via Test Prediction example using U-net Model w/ Improvement on stage2 test set (id = 99)

6 Conclusion

First of all, the final model we chose is the U-net model with improved Relu layers and Score Metric. Table 3 below shows all the combinations we tried to achieve our final model while the **bold** ones are our final combination.

Table 3: Hyper-parameter Selection

Hyper-parameters	Combinations
Resizing Size	256*256, **384*384**, 512*512
Starting Learning Rate	0.001,**0.003**,0.01

Learning Rate Reduction Rate	15%, 20%, 25%, 30%, **35%**, 40%
Learning Rate Reduction Pace	Every 3 epochs & **5 epochs**
Dropout Rate	0.1,**0.15**,0.3

Moreover, although there are some potential problems as discussed before in stage2 predictions, we still improved our test accuracy from 0.340 to 0.418 in stage2 which might due to the larger test set size of stage2, 3019 test images in stage2 compared to only 65 test images in stage1.

Lastly, we learnt from this project that machine learning is not only about training a perfect model with choosing the best combinations of hyper-parameters. When dealing with new problems it's really important to look at the data first and preprocessing the data to make your model fits better. Proper post-processing is also necessary in increasing the test accuracy and there is still much more skills about that we need to learn about that.

7 Reference

1. Åmdal-SævikKeras, K. (2018, January). U-Net starter - LB 0.277. Retrieved from https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277?scriptVersionId=2164855
2. Bailey, S. (2018,). Step-by-step Explanation of Scoring Metric. Retrieved from https://www.kaggle.com/stkbailey/step-by-step-explanation-of-scoring-metric
3. **Bailey, S. (2018,). Teaching notebook for total imaging newbies. Retrieved from https://www.kaggle.com/stkbailey/teaching-notebook-for-total-imaging-newbies**
4. Google Colab: https://colab.research.google.com/
5. Rakhlin. (2016). Fast Run-length Encoding (Python). Retrieved from https://www.kaggle.com/rakhlin/fast-run-length-encoding-python
6. MPWARE Team. (2018, January). Stage1 EDA: Microscope image types clustering. Retreieved from https://www.kaggle.com/mpware/stage1-eda-microscope-image-types-clustering
7. **Raoul. (2018, February). Nuclei DSB 2018 TensorFlow U-Net Score 0.352.Retrieved from *https://www.kaggle.com/raoulma/nuclei-dsb-2018-tensorflow-u-net-score-0-352***
8. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.
9. Sharhan, M. (2018, January). Defining Microscopy Type. Retrieved from https://www.kaggle.com/nhargan/defining-microscopy-type
10. Thomas, J. (2018, March). Exploratory Analysis. Retrieved from https://www.kaggle.com/jerrythomas/exploratory-analysis

*Note the reference in **bold** are softwares built by others that we used for our experiments.