COMP 540, Statistical Machine Learning
Final Report

Instructor: Devika Subramanian

Group: Chengyin Liu (cl93), Ran Jin (Oliver) (rj23)

Motivation

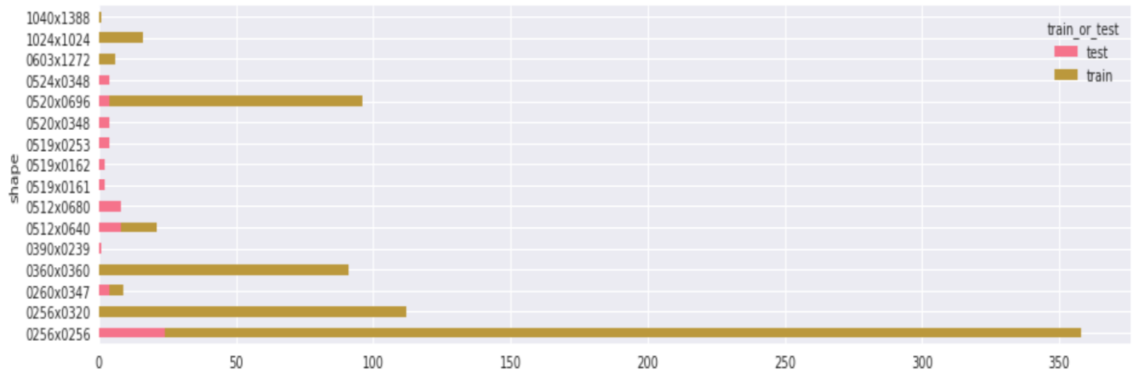
A lot of people are suffering from all various diseases like cancer, etc. Some of the diseases are still curable within a certain time constraint (i.e. if the nuclei can be precisely detected in the early stage). The purpose of this Data Science Bowl is to detect nucleus from cells faster in a more accurate manner. Specifically, we want to develop a generalized algorithm that does the dectection for us in automation. Thus, overall this competition is a great pracitce for us to apply what we have learned in class to a generalized segmentation problem.

Exploratory Data Analysis

Before developing models, we want to make sure that the data is executable. We are given two sets of datasets. One is testing dataset, and the other one is training dataset which includes images and masks (670 for train, and 65 for test in stage 1). By finding the number of image IDs and number of images, we see that they are equal. Thus, we say that they are uniquely corresponding to each other. We also looked at the masks. They are the ones that separately highlighted the nuclei in each image and are saved as multiple 'png' form images for each training image. Our task here is to find the nuclei within each image. To be more precise, we will be extracting the most shining part of each cell and that is the nucleus. However, after checking the image sizes, pixel intensities, and nuclei sizes, we do have several problems here:

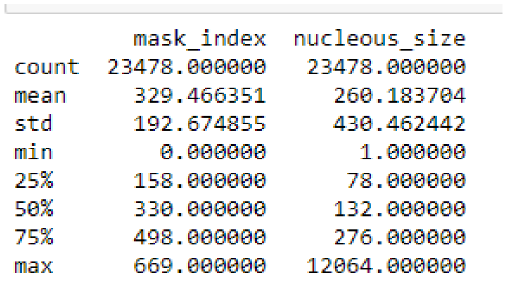
1. The background colors differ that some of them have a black background and some have a purple background. Thus, we need to align all colors to the same brightness level.
2. Since the luminance of images differs, we should adjust the luminance from the sample as well.
3. Images have different sizes. Thus, I ran a distribution of the images, and saw that

the majority of the images have a 256*256*3 for both test and train. Thus, we will convert the rest to the same standard and this will make our analysis easier as time proceeds. (Refer to Exploratory Analysis by Jerry Thomas on Kaggle [1]).



4. Some testing images are quite blurry that are hard to detect from the background. Therefore, we considered to randomly blur some of the images from the sample set.

5. The sizes of the nuclei are different and have a high variance. The smallest nucleus is only 1 pixel and the larger ones may have more than 10,000 pixels as showed below. This may lead to misclassification since it is not easy to tell the difference between the small ones and the noise pixels.



```
          mask_index  nucleous_size
count   23478.000000   23478.000000
mean      329.466351     260.183704
std       192.674855     430.462442
min         0.000000       1.000000
25%       158.000000      78.000000
50%       330.000000     132.000000
75%       498.000000     276.000000
max       669.000000   12064.000000
```

6. While detecting the nucleus from cells, we find that sometimes non-nucleus parts, which could be some other tissues of the cells, can be erroneously defined as nucleus because they look similar to the real nuclei.

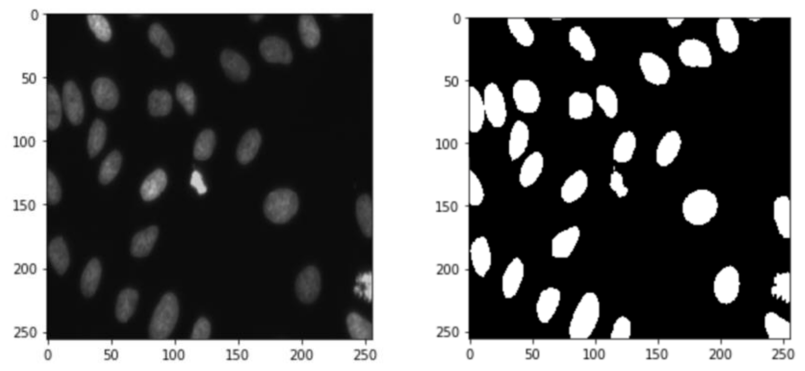
Data Preprocessing and Augmentation

1. Resize the images

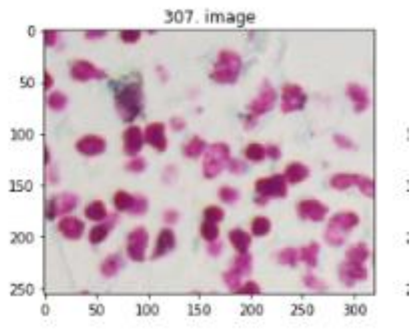
The original sizes of training images are showed below,

```
(256, 256)      334
(256, 320)      112
(520, 696)       92
(360, 360)       91
(1024, 1024)     16
(512, 640)       13
(603, 1272)       6
(260, 347)        5
(1040, 1388)      1
Name: 0, dtype: int64
```

To make them have the same size for the input of our network model, we resize both the training images and test images to the same size, for example, 256 * 256 or 512 * 512. After resizing, we randomly sampled images with their masks to check the resizing result and they looked pretty good.



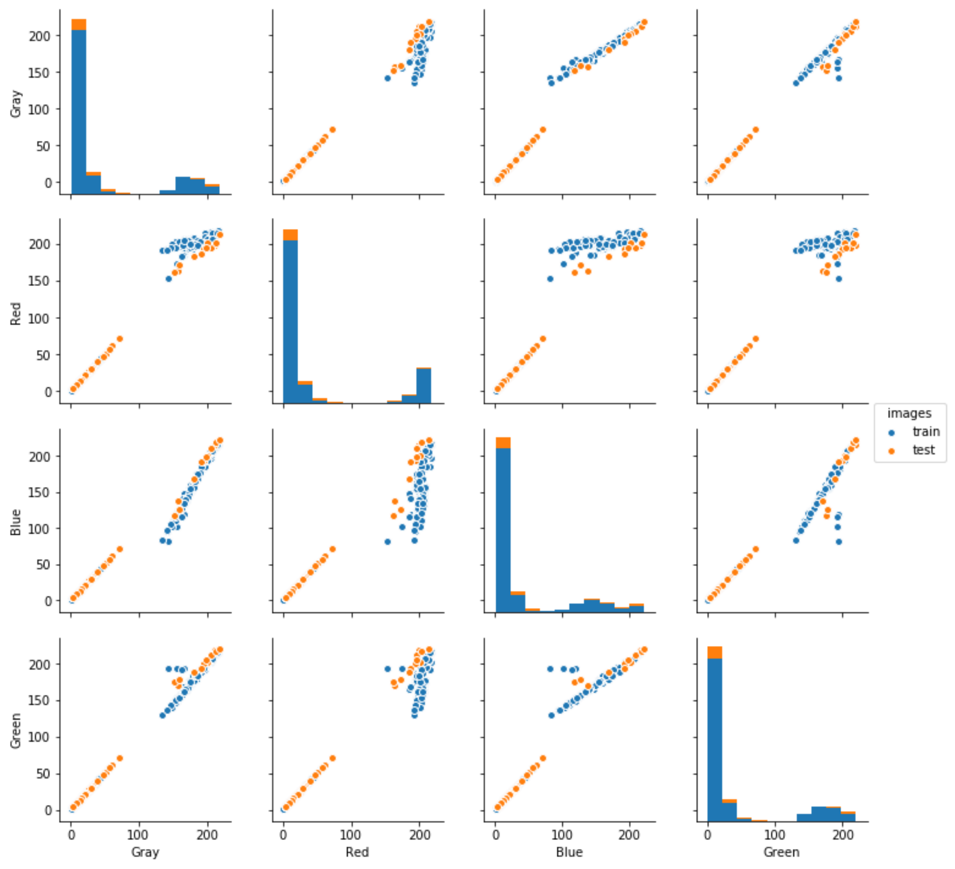
2. Invert bright images



For images with a very bright background, we invert their pixel intensity by thresholding. By doing this, most of our input images have the similar level of luminance values.

3. Normalize the pixel intensity

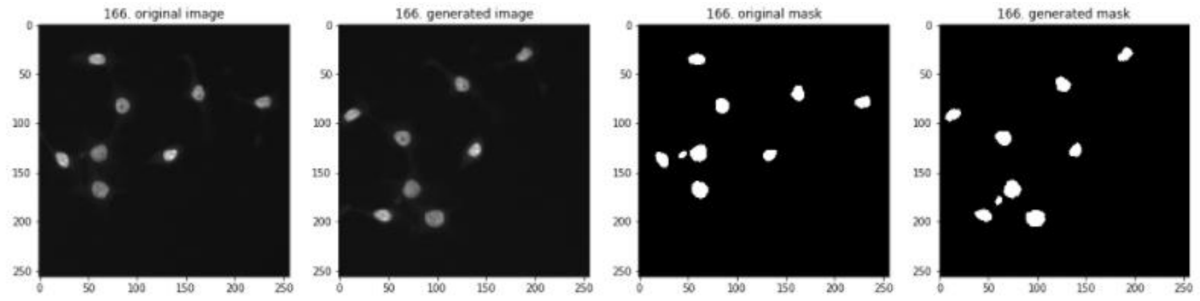
After studying the pixel intensity in the images, we found it is necessary to normalize the intensity so that the intensity values from different images are on the same level of scale.



Although the red, green, and blue channels have similar intensities on average, the intensities of different images have a large variation. So, we normalized the intensities of both the training images and the testing images. Otherwise, our model may focus only on the images with large pixel intensities and becomes unbalanced.

4. Image Augmentation

We do not have enough training data as always, especially for stage 2. Therefore, we generated some new images and masks based on our original training images and their masks. We reversed top and bottom, left and right of the images. And we also rotated the images by 90, 180, and 270 degrees respectively. For example, the result of the image and mask being rotated 90 degrees are showed below.

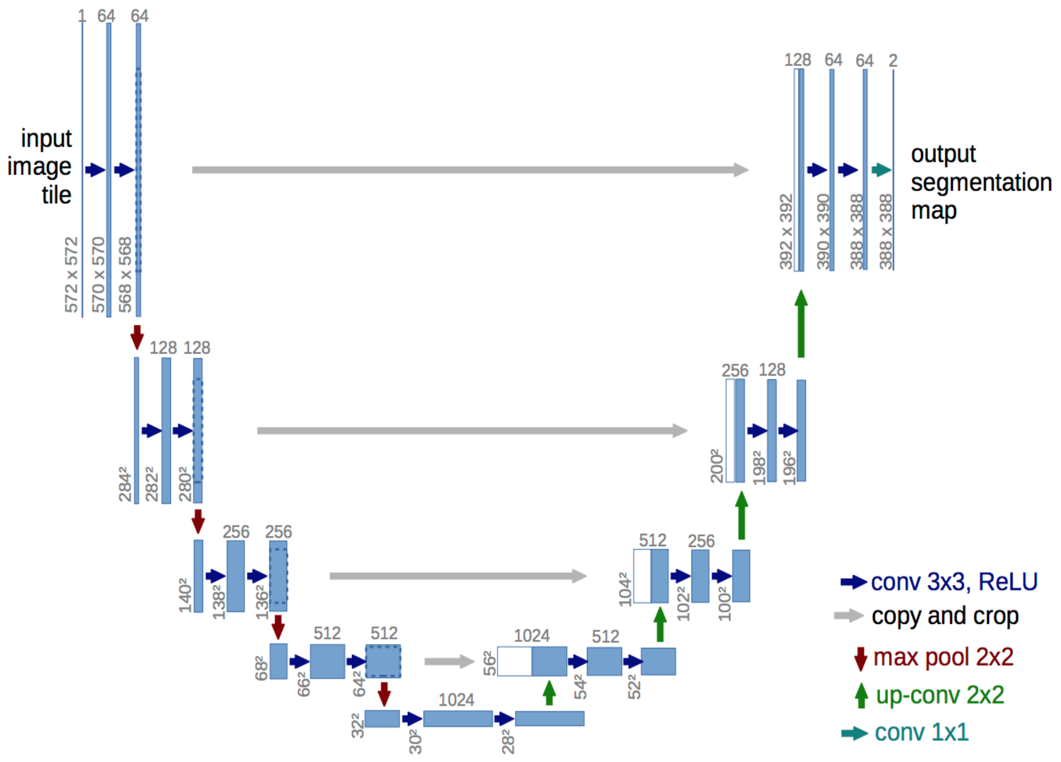


By doing Augmentation, we generated more training data with a wider cover of the nuclei shapes and orientations. This definitely made our model more robust.

Model selection

1. U-Net

We chose U-Net as our base model, and motivation of such model was from the paper '*U-Net: Convolutional Networks for Biomedical Image Segmentation'* by Olaf Ronneberger, Philipp Fischer, and Thomas Brox [2]. It is a convolutional network architecture for fast and precise segmentation of images. The architecture of this model is consisted of two paths as we see below. As we input the images, it captures the structure and contracts it down to the bottom and expands it back up which enables precise localization. The paper showed that such network can be well trained on an end to end segmentation model with few images which is our case here, recognizing all the masks in a single input image.

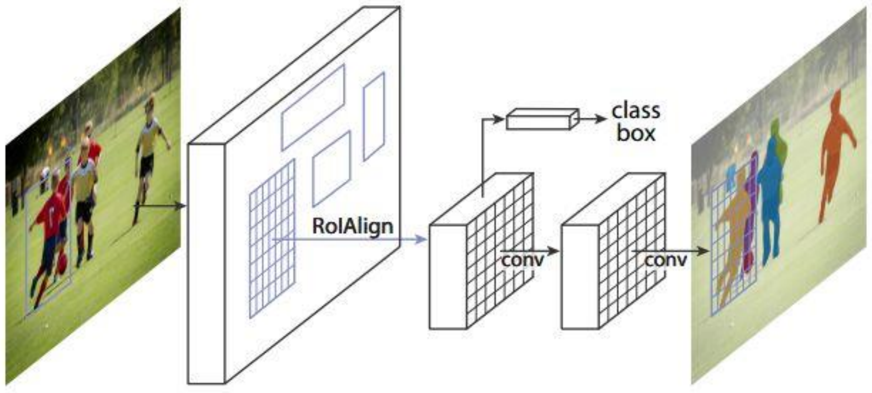


2. Mask R-CNN

As U-Net learns from the whole input image to segment the regions of nuclei, it failed to make correct detection for the overlapping nuclei in our experiments. That is to so, when nucleus overlaps with each other, our algorithm tends to treat them as single large nucleus. This will introduce double-error for both the nuclei being detected when calculating the precision score. So, we considered to use a pixel-level segmentation model as well.

Our second candidate model is Mask R-CNN [7]. Mask R-CNN improves the Fast R-CNN [5] and Faster R-CNN [6] by adding a third branch that gives the output of the object mask. It has two stages. The first stage scans the images to generate region proposals (Region Proposal Network, RPN), and the second stage classifies the proposals to generate bounding boxes and masks as we see in the image in the middle. We may look at it as it is consisted of three modules:

1. Backbone
    - Region Proposal Network (RPN)
    - Feature Pyramid Network (FPN)
2. RoI Classifier & Bounding Box Regressor
3. Segmentation Masks



Another important improvement is that Mask R-CNN uses RoIAlign layer instead of RoIPooling layer. This ensures more accurate alignment between the image and the feature map. As a result, the final detected masks are precisely located as we are required in our nuclei segmentation problem.

<u>Fitting the model</u>

1. U-Net

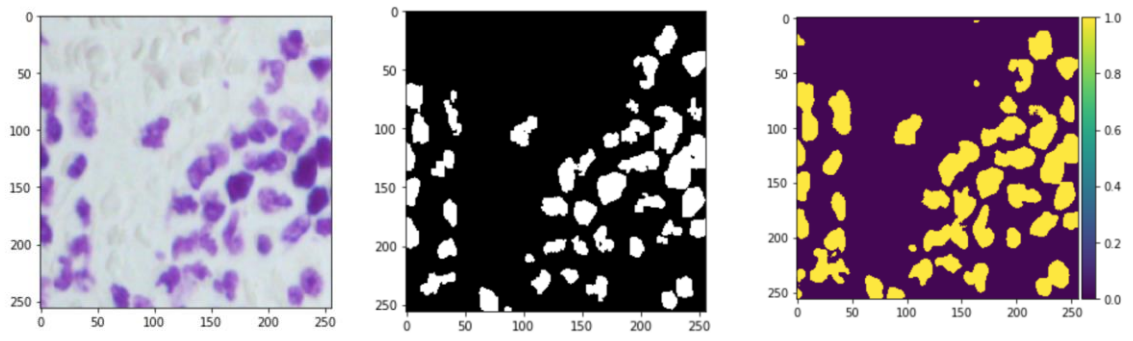
We developed our U-Net convolutional network model using Keras. Since it is a technique that is widely used with biomedical image segmentation. First, we had to define mean average precision at different intersection over union thresholds metric in Keras. The initial parameters we have determined were based on this paper [2].

We got our result using the resized 128*128 images as it took way too long for large image size. We had an accuracy of 0.283 on the leaderboard using U-Net with the initial parameters, and a single model. The initial depth of the model we used was 8 instead of 64 as indicated in the paper.

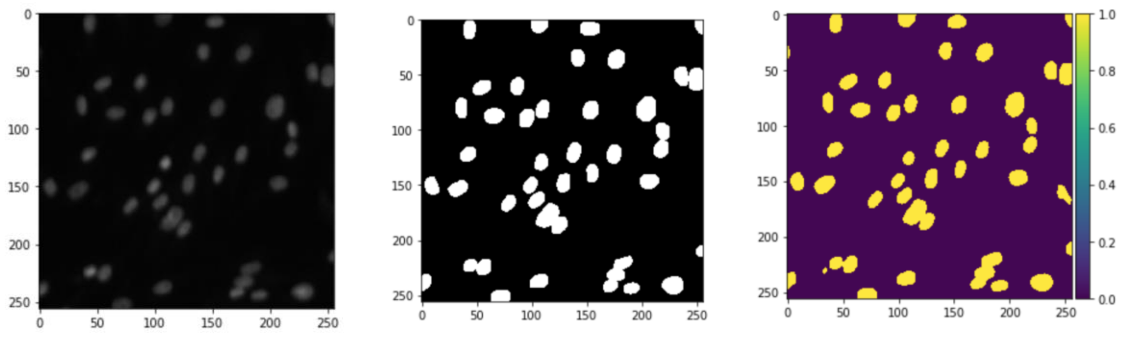
Here we tried to avoid overfitting by using K-fold cross validation. The losses on the validation sets were not much lower than the result of the training set in our experiments, indicating our model was not heavily overfitting yet at this stage.

As the time took too long running on our laptops, we used Google CoLab to speed up the process and thus, we changed the resizing to be 256*256. Then we changed the model to a different one (call it model 2) which was to change the depth of the convolutional layers from initial depth of 8 to 16 and the filter size from (7, 7) to (3, 3) for all. Initially, we used (7, 7) because it works well in our previous CNN implement, but as we have glanced through the U-net implementations for medical image segmentation problems, the majority of them are using (3, 3). For the depth, we tried to make it deeper to see if it is going to improve our result. We got a similar result of 0.285, which is slightly above our result from last time on the leaderboard. The epochs had an early stopping at around 31 with a validation loss of 0.0663.

Keep going, we used model 2 but with different parameters, changing the epochs to 30 from 50 and batch size from 8 to 16, and we got a 0.323 accuracy, which was our history high. Surprisingly, the epochs had an early stopping at around 11 with a validation loss of 0.0619. The following pictures are to show how model 2 fits the training data:



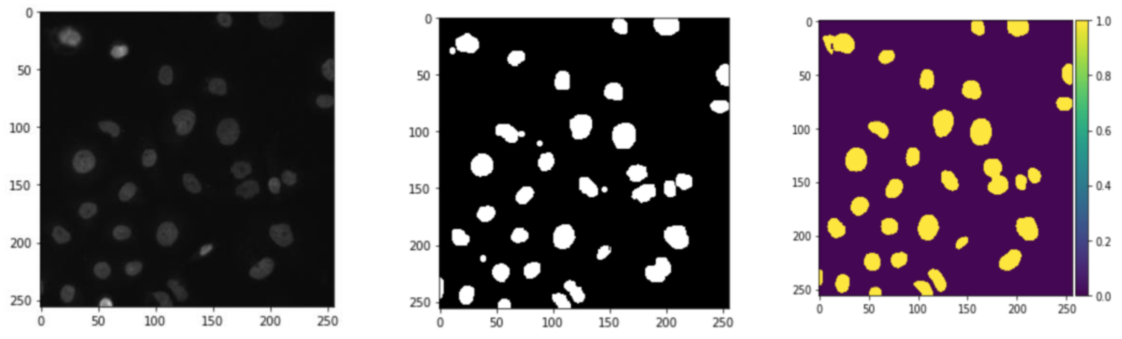
We see that it fits pretty well on most samples. And the Validation data also fits well as below.



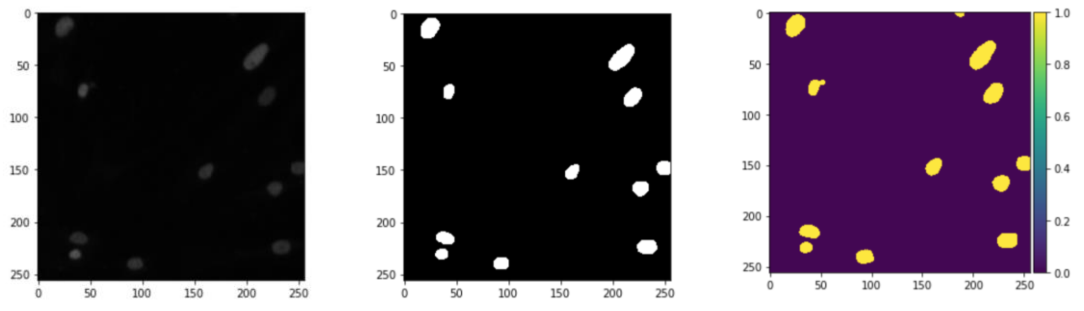
One thing that we did not notice was that the model 2 was not symmetrical in the sense that the depth of the second layer is 32 but the depth of the last layer was 64 since a theoretical U-net behaves in a symmetrical manner. we realized about this issue after the second run.

After that, we changed it to the model (model 3) with symmetry. Also, we changed the learning rate from 1e-3 to 1e-4 in order to better catch the minimum validation loss. The accuracy for this one is 0.236 which is actually lower than the previous two.

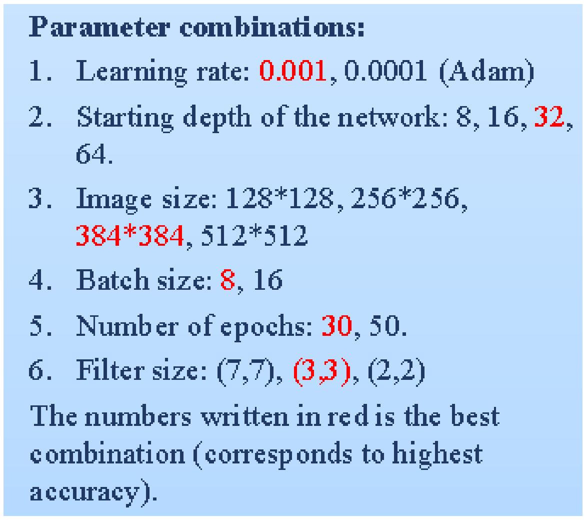
Samples of the training dataset:



And samples of the validation:



Aside from changing the depth, we also tried adding one more layer and we did some quite similar adjustments as above and got a history high of 0.338 with 30 epochs, 384*384 resized images, and the initial depth of this model is 32. The result shows that deeper network and large image size will help to extract more accurate information from the images. Below we list the parameter combinations for our U-Net model,



Parameter combinations:

1. Learning rate: 0.001, 0.0001 (Adam)
2. Starting depth of the network: 8, 16, 32, 64.
3. Image size: 128*128, 256*256, 384*384, 512*512
4. Batch size: 8, 16
5. Number of epochs: 30, 50.
6. Filter size: (7,7), (3,3), (2,2)

The numbers written in red is the best combination (corresponds to highest accuracy).

Also, we added dropout layer after every convolutional layer in order to avoid overfitting this time, which also contributes to the improvement of our score. This model finally got a score of 0.422 in stage 2 of the competition.

2. Mask R-CNN

Training Mask R-CNN is time-consuming. Therefore, we randomly picked 25 training samples as the validation set to reduce the validation time. Followed the guide of how to train the Mask R-CNN model by Waleed Abdulla [8], we started training our model with the weights of pre-trained models like ImageNet and COCO. For our nuclei segmentation problem, we chose the ResNet50 as our backbone network architecture because there is only one object type. The sizes of the RPN square anchor were set to (8, 16, 32, 64, 128) to satisfy the variance of the different sizes of our objects as showed in the exploratory data analysis.

An initial learning rate of 0.02 was used in the Mask R-CNN paper [7], however we found it is too large in our implementation after looking at several iterations. So, we changed it to 0.001 for stage 2. During training, 30 and 40 epochs are tried due to the limited time at stage 2. We believed that more iterations may improve the performance, but actually, the loss only changed a little after 30 epochs on our model, so we assumed its convergence.

For input image size, we resized images into 256 * 256, and 512 * 512 as well. The corresponding highest precision is 0.419 and 0.431 for stage 2 testing dataset. The larger size input images contain more information and lead to a better result as expected.

Problems encountered

Prediction problems:

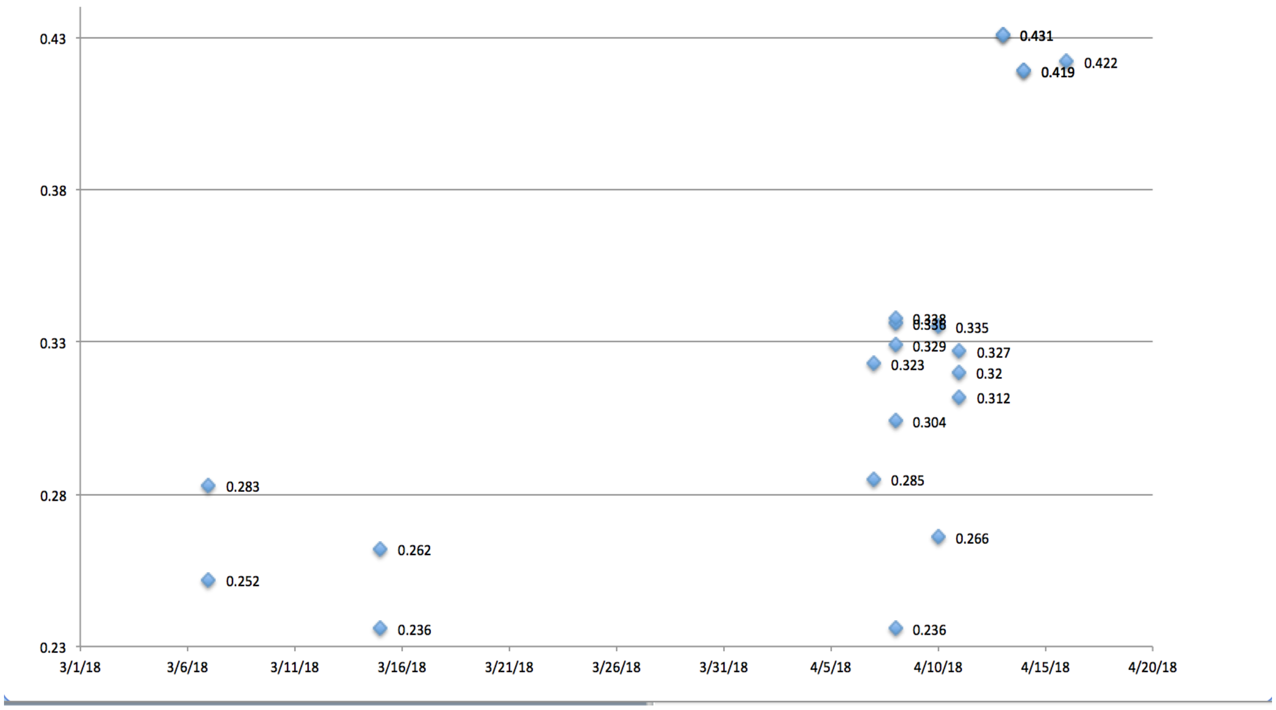
- In some images, the nuclei are very close to each other or even overlap. Thus, they may be detected as only one nucleus or the all clogged together nuclei do not get detected.
- Since some of the images are blurry, the non-nuclei tissues in those images could be mistakenly identified as nuclei.
- Overall, we think that we could improve our U-Net model by efficiently separating the clogged together nuclei into different images. However, due to the time constraint, we will have to implement it in the future exploration.

Executing code problems:

- When training with U-Net in stage 2, we had to download the new test data which is consisted of 3000+ images from the Kaggle. Since it was a relatively large dataset, we used Google Colab and the major problem we encountered was the unsuccessful data upload. Thus, we split over 3000 images into 20 zipped files and basically tested every 150 images each time and finally combine them into one submission file.

Leaderboard results

We started submitting our results from 1/25/2018 and we got some reasonable scores from 3/7/2018. We had a total of 27 submissions.



As you can see, we have improved from 0.252 to 0.431 for the overall precision score on the leaderboard. The highest score for U-Net was 0.422, and the highest score for Mask R-CNN was 0.431. Overall, we were ranked 236 / 3634 (Top 7%).

2018 Data Science Bowl
8 days ago · Top 7%

236th
of 3634

Libraries

The two main libraries we used were TensorFlow and Keras. TensorFlow is an essential library for deep learning and Keras [3] is a minimalist, highly modular neural networks library, written in Python and capable of running on top of TensorFlow. These greatly simplify the implementation when building the model since it is already implemented in the library.

Takeaways

1. EDA is extremely important for us to understand the data, also how to specifically approach the problem using the data available, and how to turn those data into an executable phase.
2. As validation loss goes down, it does not necessarily mean that the accuracy would go up.
3. For every run, even with the same parameters and model, the accuracy could vary probability within ± 0.05 accuracy range.
4. Resizing images is easier for training, but it does make some of the images lose information in the sense that if an image of 256*256 was downsized to 128*128, some information of the original image is lost. Thus, this is a trade off between complexity of training and information presented in the images.
5. Deeper Neural Networks capture more model structure than the shallower ones.
6. Algorithms like Mask R-CNN are complicated in terms of number of layers and depth that would cause really long training time if executed on our own laptops. Thus, a lot of times we would have to rely on other services like AWS or Google CoLab that significantly optimizes the running time.

References:

[1] https://www.kaggle.com/jerrythomas/exploratory-analysis

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[3] https://keras.io/

[4] R-CNN: https://arxiv.org/abs/1311.2524

[5] Fast R-CNN: https://arxiv.org/abs/1504.08083

[6] Faster R-CNN: https://arxiv.org/abs/1506.01497

[7] Mask R-CNN: https://arxiv.org/abs/1703.06870

[8] https://github.com/matterport/Mask_RCNN

[9] https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277?scriptVersionId=2164855

[10] https://www.yuthon.com/2017/04/27/Notes-From-Faster-R-CNN-to-Mask-R-CNN/

[11] https://www.kaggle.com/raoulma/nuclei-dsb-2018-tensorflow-u-net-score-0-352

[12] https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46