Capstone Project Report **Sandeep Anand**
Udacity MLND April 1^{st} 2018

I. Definition

Project Overview

- This project is a current open "Kaggle Competition" called **"2018 Data Science Bowl"**.
- The competition is to create an efficient Nuclei Detector Algorithm to detect disease and speed drug recovery. We ask if a deep learning model can be used to distinguish what a nuclei looks like regardless of the type of stain used
- Any Model or algorithm can be used and the purpose is to automate Nucleus detection
- Historically , scientists have done this process manually under the microscope till some time back. Identifying the cells nuclei is the starting point for most analyses because most of the human body contain a nucleus full of DNA, the genetic code that programs each cell. Identifying nuclei allows researchers to identify each individual cell in a sample, and by measuring how cells react to various treatments, the researcher can understand the underlying biological processes at work
- Our One general model that finds out Nuclei across different systems and experimental systems and data will be used to train and test over 25000 nuclei images for this challenge
- **Motivation**: This challenge is a medically inspired project and is a very important step towards finding a solution for difficult to identify diseases and reduce human suffering. It is also a fairly challenging project and is a current live project with a lot of interest in this area, which seems to a great way to learn more about legacy and new Deep learning approaches to solve image recognition problems in general
- Some other simpler problems that we see regularly that could be solved [7] via the techniques and algorithms that involve, Object detection, Object recognition, Object Segmentation, Image segmentation and localization and broadly use CNNs to solve these, and ImageNet is a benchmark for these kind of algorithms, some examples below
  - Object detection by a Vehicle for autonomous driving through image segmentation
    1. Pedestrian detection
    2. Brake light detection
    3. Face detection
  - Recognition Tasks
    1. Face recognition
    2. Fingerprint recognition
    3. Iris recognition
  - Motion Analysis
- Broadly, Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions.[11,12,13,14]

Problem Statement

- The submission main goal is to automate the process of Nuclei detection through an algorithm. The problem is that there are a high number of nuclei images that have been collected through different experimental processes and the job of the algorithm will be to identify the nuclei for every image in the test set across varied conditions and accurately predict nuclei in the given image
- The problem here can be one of the two types (I got this from some of the discussions)[1], object detection or image segmentation.I will go deeper on the evaluation metric later on the proposal
- **Possibilities and Potential Solutions: Considering cases and ideas on which I can build my project**
  - **Case 1:** Considering that this is a image detection problem, the natural Solution will be to solve this problem using the image classification algorithm that we learned in the Udacity Course-ware , such as a simple CNN/RNN architecture that we have used to classify the MNIST or SVHN dataset, though there are difficulties with this approach, but still my first try will be to use a simple CNN model and see if I can get a certain accuracy level on the test set. When I say difficulty I would mean over-fitting and under-fitting, considering the fact that MNIST and SVHN or even Dog breed classifier models have datasets/images which are more structured and have clear features, while Nuclei are far from that, they are very similar and even difficult to distinguish. I would guess a simple CNN model will under-fit, I can describe more on this with my first attempt on a CNN architecture
  - **Case 2:** As this is a Live competition, there are benchmark models available to start with . One of the models I see is the use of Unet architecture[2] for image segmentation, which has won several competitions and can be considered one of the finest benchmark models to start the image segmentation competition . The architecture involves standard Convolution Networks, followed by a Relu Activation and Maxpooling for downsizing the feature map and this is the contraction step. Similarly after the contraction step we do a Expansion or up-conversion where we need to finally output a high resolution segmentation map. It seems this achieves fast training and better accuracy . I plan to try an attempt on a Unet as my 2nd choice of the project
  - **Case 3:** Unet Model with image augmentation, this is a modification of the case2, where we use image augmentation to produce better accuracy besides using image just Unet
  - **Case 4:** There is another architecture altogether that is being discussed, Faster R-CNN[3]. This is considered as a State of the art Object detection algorithm and is also recent . This is a end-to-end Fully convolutional network that simultaneously predicts object bounds and objectness scores at each position[3] . For this project, I will take a look but currently this involved probably a lot of reading and higher Computation power
- The problem and the Case by case Solutions described above are some of the ways of approach to this problem , but as I progress I will add to my findings and probably experiment further with my approach

Evaluation Metrics[6]

This competition is evaluated on the **mean average precision at different intersection over union (IoU) thresholds**.
The IoU as the name suggests is Intersection over Union, This mean when we overlap the mask images over the real test image, how much overlap we are seeing between them, that matches , this is general legacy metric that is used for image detection.
The IoU of a proposed set of object pixels and a set of true object pixels is calculated as:

$$IoU(A, B) = (A \cap B) \div (A \cup B)$$

The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.5 to 0.95 with a step size of 0.05: (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95). In other words, at a threshold of 0.5, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than 0.5.

At each threshold value t, a precision value is calculated based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects:

$$TP(t) \div (TP(t) + FP(t) + FN(t))$$

A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold.
A false positive indicates a predicted object had no associated ground truth object.
A false negative indicates a ground truth object had no associated predicted object.
The average precision of a single image is then calculated as the mean of the above precision values at each IoU threshold :

$$(1 \div |thresholds|) * [\Sigma(TP(t)) \div (TP(t) + FP(t) + FN(t))]$$

Lastly, the score returned by the competition metric is the mean taken over the individual average precisions of each image in the test dataset.
After the calculation of this as a function, this metric is fed to the Model before training

II. Analysis

Data Exploration

The inputs/dataset is given, through the competition itself. The images were acquired under a variety of conditions and vary in cell type, magnification, and imaging modality (brightfield vs. fluorescence). This is designed to challenge an algorithm's ability across these variations when training and testing images
Each image is associated with the Image ID. Masks are not allowed to overlap (This is the criteria for the masks, for one image set , there will be multiple masks and each mask will have one Nuclei)

- Number of Training images in stage1_train : 670
- Number of Test images in stage1_test: 65

- Shape of Images are different, the below code snippet extracts the image sizes, hence I resize them into two different shapes based on performance using skimage.transform.resize
  The format below is (image height, image width, number of channels)
  The number of channels for masks is 1 always
  - (128, 128, 3)
  - (256, 256, 3)

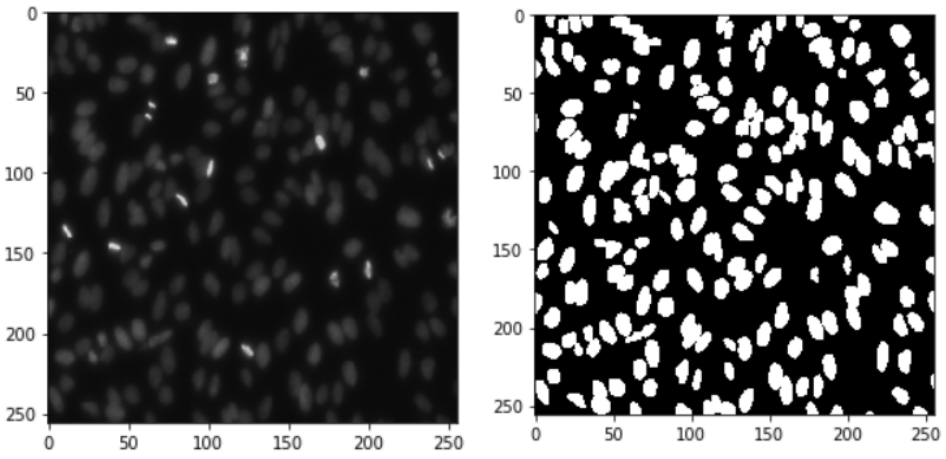
```
trainings = [train_ids[random.randint(0,len(train_ids))] for i in range(20)]
paths = [TRAIN_PATH+id_+'/images/'+id_+'.png' for id_ in trainings]
imsize = [imread(path).shape for path in paths]
for size in imsize:
    print(size,'\n')
```

```
(520, 696, 4)
(520, 696, 4)
(256, 256, 4)
(1024, 1024, 4)
(256, 320, 4)
```

```
testings = [test_ids[random.randint(0,len(test_ids))] for i in range(20)]
paths_t = [TEST_PATH+id_+'/images/'+id_+'.png' for id_ in testings]
imsize_t = [imread(path).shape for path in paths_t]
for size in imsize_t:
    print(size,'\n')
```

```
(256, 256, 4)
(512, 680, 3)
(520, 348, 3)
(256, 256, 3)
```

- Below is the example of a training image and the corresponding mask for it

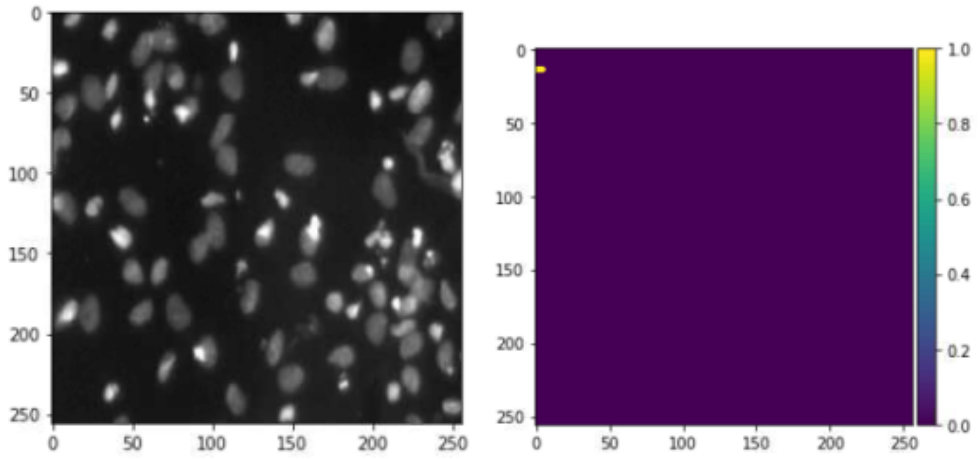


(a) Image. (b) Mask.

Figure 1: Example of Image and Mask.

- There are some abnormalities in the dataset as discussed below

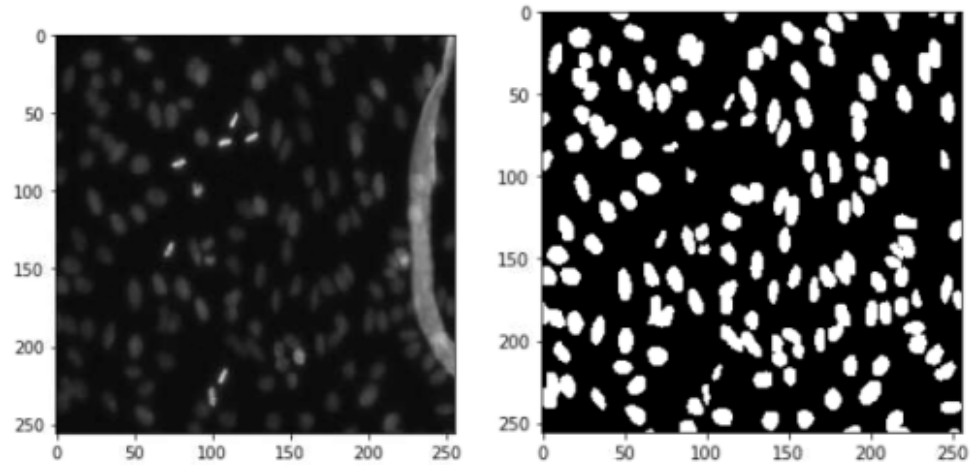
1. The Mask is just empty for this image



(a) Image. (b) Mask.

Figure 2: a.Abnormal data Image and Mask.

2. There are some figures not annotated in the mask



(a) Image. (b) Mask.

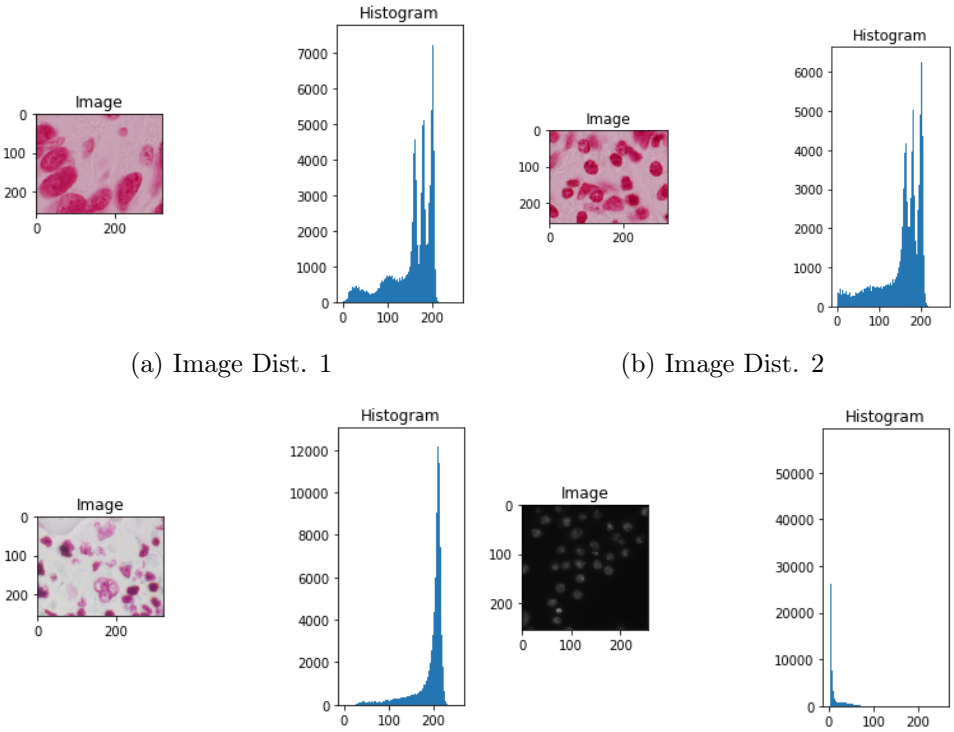
Figure 3: b.Abnormal data Image and Mask.

- I plan to use a validation set of 0.1/0.2 times of the training set

Exploratory Visualization

- Here I visualize the overall histogram distribution on some random nuclei images that are present to understand how they vary in pixels, ranged between [0,255].
  The reason is basically to know the variance in color as we analyze all the images present, The histogram corresponds to the image on the left, as we can see there are bigger spikes for darker images with color and very low concentration pixel spikes for dark images.

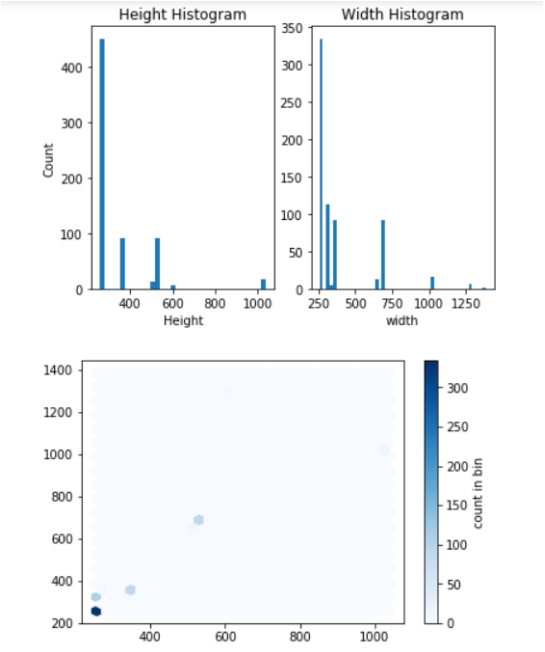
This image demonstration helps me to understand how the pixels are varying for different images and where the concentration lies



(a) Image Dist. 1 (b) Image Dist. 2

(c) Image Dist. 3 (d) Image Dist. 4

Figure 4: b.Image density Distributions



(a) Image Dist. 4

Figure 5: Density of Training Images.

- The above plots show that the best image size to fit all would be [256x256]

Algorithms and techniques

Here I discuss some of the Algorithms and techniques that are being used to solve this problem and what I have chosen to use for my LB score

- As the Starting point ,Let me discuss the easiest and Benchmark model that I will be using to solve this problem. Considering the fact that this is a image detection problem, just like the MNIST dataset, we can just treat it as simple images and solve this problem using CNN to see how a simple CNN does in predicting the masks.A multilayer Simple CNN architecture that I used first is seen in the below diagram

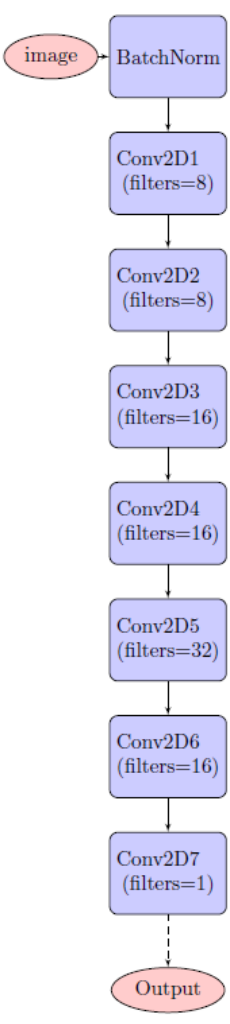


Figure 6: Simple CNN model used

- Next I will discuss the algorithm that I used to construct the base model that I will primarily use for this competition with possible modifications to make it better. This is called Unet[2]. I will be using Keras to implement the Unet model which seems to be a common architecture for biomedical image detection and segmentation problems as such.
  Parameters
  Number of conv 3x3, RELU = 19
  Number of MaxPool Layers/Downconversions = 4
  Number of Upconversions = 4

The Model is stated by the figure below

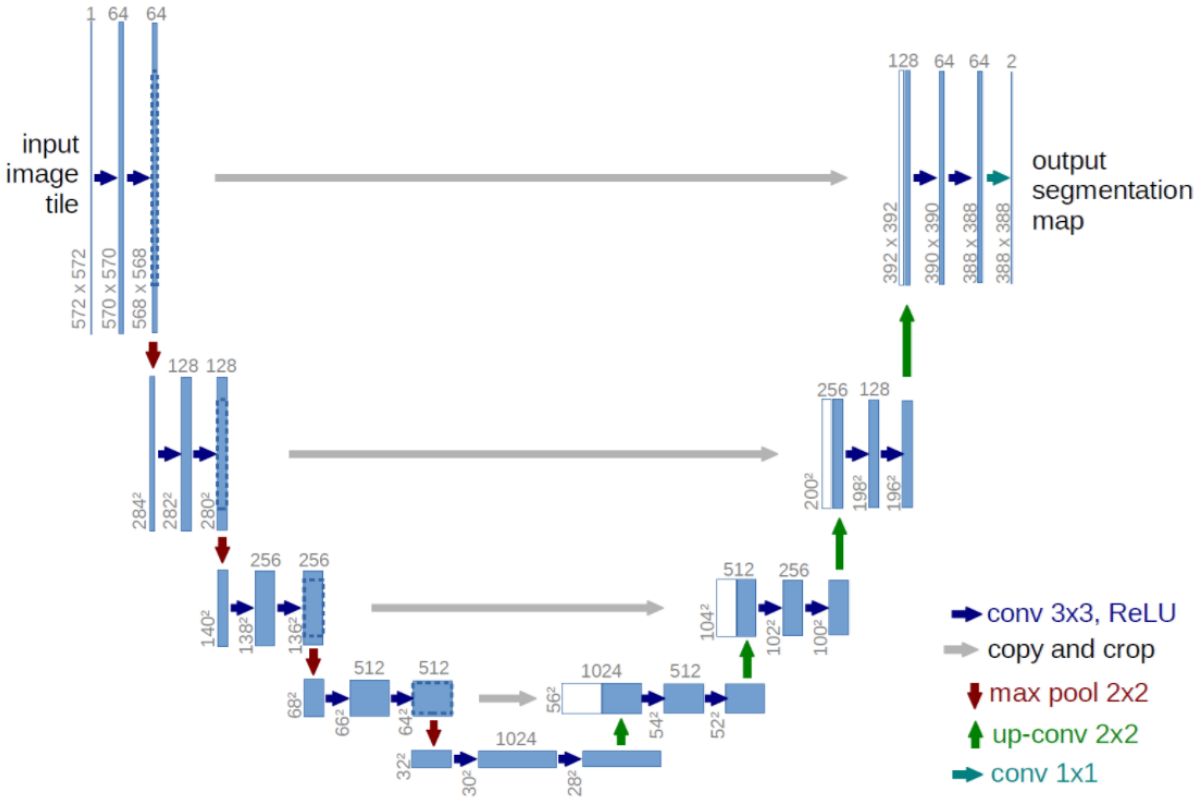


Figure 7: Unet Model Architecture[2]

This architecture is fairly new and came up in 2015-2016 time frame, so not a lot of changes required in respect to implement it in keras , but once the model in ready we can go ahead and train.This method can train end to end , and the network in fast to train. This model strongly uses data augmentation to use the available annotated images(training images and masks) samples.

Steps to handle raw images

- Raw Image In
- Each Blue Box is a Feature Map
- Each operation is a convolution followed by a Non Linear Activation function (RELU)
- The Input image is propagated through all possible paths
- Final segmentation map is achieved at the output
- The Output segmentation map has two classes one for background and one for the foreground (masks)

Benchmark Model

- I choose to benchmark this project by solving this project using a simple CNN model as below which scores much lower that the Unet model, **the LB score of this model comes**

up to be 0.156. My Benchmark was achieved through the CNN model seen below Please refer to my submissions at my Kaggle profile
https://www.kaggle.com/crafter707

Layer (**type**)	Output Shape	Param #
NormalizeInput (BatchNormali	(None, 128, 128, 3)	12
conv2d_856 (Conv2D)	(None, 128, 128, 8)	224
conv2d_857 (Conv2D)	(None, 128, 128, 8)	584
conv2d_858 (Conv2D)	(None, 128, 128, 16)	1168
conv2d_859 (Conv2D)	(None, 128, 128, 16)	2320
conv2d_860 (Conv2D)	(None, 128, 128, 32)	4640
conv2d_861 (Conv2D)	(None, 128, 128, 16)	528
conv2d_862 (Conv2D)	(None, 128, 128, 1)	17

Total params: 9,493
Trainable params: 9,487
Non-trainable params: 6

III. Methodology

Data Preprocessing

- The Data is directly in the form of raw images and has been already in a format that can be easily worked with as the Unet model takes input images in the raw format and gives out segmented masks in the raw format ie, .PNG, There are still some processing required and it seems the below preprocessing was able to boost my **LB score from 0.31 to 0.34** which is considerable
- **Normalization** , We see from the graphs already discussed "Exploratory Visualization" section that it is best to take the Image Height and the Image width as [256, 256], as most of the images are in that range., We convert the pixel values by a local factor, and hence make a pixels brighter or darker
- **Transformations**
  1. Conversions to gray scale from RGB, to make sure all images are like
  2. Inversions , if the background is lighter and hence difficult to recognize nuclei
- Through Analysis , I found some anomalies , but they are extremely few and will not hurt the training a lot, The reason for that is the data was meticulously collected and annotated .

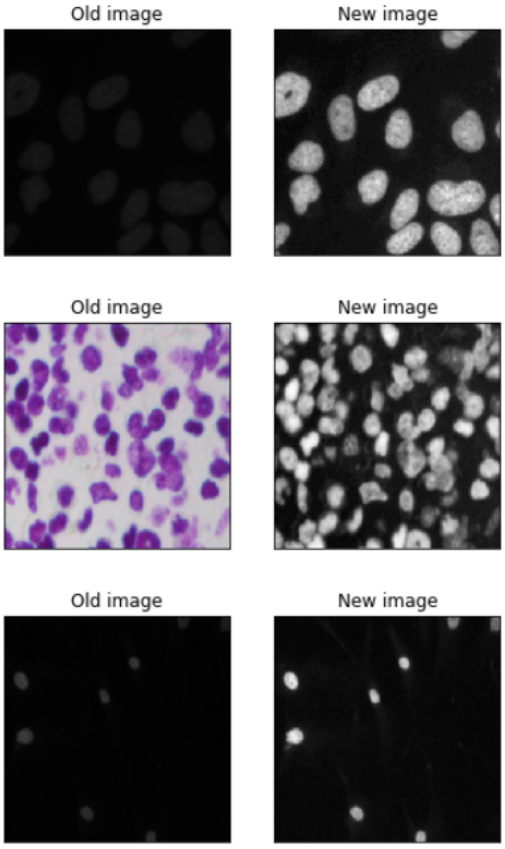


Figure 8: Image Transformations

Implementation

The Figure 9, gives an overview of the steps of implementation of the Process from start to finish

1. From the start of the project , I had an idea that to run any complex Model over a certain high number of epochs, I would certainly need the help of a GPU, which meant running the model through AWS Sagemaker or Floydhub. My latest High LB Score model is run on AWS ml.p2.xlarge EC2 Sagemaker Jupyter instance, on Python 3.x. For this, one of the challenges was to learn the use of s3 buckets, and the boto3 python package, which is required to access the data present in the boto3 package. Once that was set, I was able to access the train and the test images and masks.

2. I decided to normalize all the images to a fixed height and width of (256x256), as I found that most of the images lie in that pixel range. Resizing the images is important in providing an input to the Unet model[2] , as we provide raw input images and masks as training features and labels to the model

3. This competition is evaluated on the mean average precision at different intersection over union (IoU) thresholds. The IoU of a proposed set of object pixels and a set of true ground truth object pixels is calculated as a ratio

$$IoU = A \cap B / A \cup B \tag{1}$$

   The coding for this part was tricky and was done as below,

```
def mean_iou(y_true, y_pred):
    prec=[]
    for t in np.arange(0.5, 1.0, 0.05):
        y_pred_=tf.to_int32(y_pred>t)
        score, up_opt=tf.metrics.mean_iou(y_true, y_pred_,2)
        K.get_session().run(tf.local_variables_initializer())
        with tf.control_dependencies([up_opt]):
            score=tf.identity(score)
        prec.append(score)
    return K.mean(K.stack(prec))
```

4. The Unet Model that I have implemented using Keras is thoroughly used in the image segmentation domain and I have taken it from this paper[2], There are certain tweaks like converting into gray scale and inverting the images that helped me achieve better results than the plain Unet model.

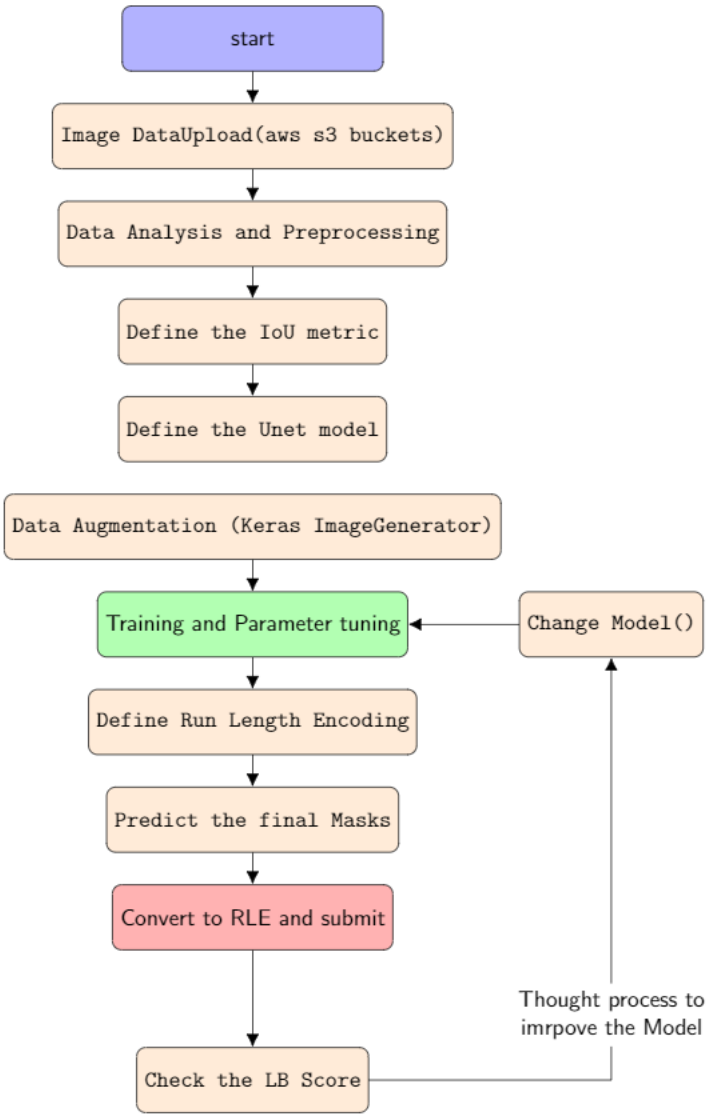


Figure 9: Implementation

IV. Results

Model Evaluation and Validation

- The final Model is one of the best models that I was able to achieve for this project try, currently my best submission is ranked at top 33% (1141/3380), with 10 days remaining for the project to end. The model hence is good to the expectations and I can do better which I will discuss in the later sections
- The final model is tested with test set as given per the competition and the masks that are predicted through the model is converted into a RLE format for submission. The submission is graded based on the Online Judge that is through kaggle itself.
- Yes the Model is robust , but may not be the best, There are other models that I am exploring and if possible will report the scores before the contest ends, The one is the mask-rnn model, which seems to outperform the U-net model by quite a bit.
- I have changed the model as below, into four sets , where I achieve respective scores that I mention below, The Model and the results are got using legacy proven techniques and they can be fully trusted as we use the Kaggle Online judge to measure the test predictions
- About the generalization of data, there are a stage-2 Test set that will be launched in a week time after kaggle code freeze which will be the final verification , I think that will be a very important criteria to estimate the accuracy of my best model that is the 4^{th} case

Method	**LB Score**	**Best Validation Loss**	**Best Mean IoU**	**Epochs**
Simple CNN	0.146	0.175	0.7826	150
Simple Unet	0.282	0.067	0.866	50
Simple Unet+Data Augmentation	0.316	0.067	0.866	50
Unet+Data Aug+Transformations	0.338	0.065	0.811	50

Justification

- The final result I am submitting now are better than the benchmark model by far, which is the Simple CNN model
- The final solution is pretty significant, lets analyze some the predicted images with the best solution so far for the test images that are provided by the dataset

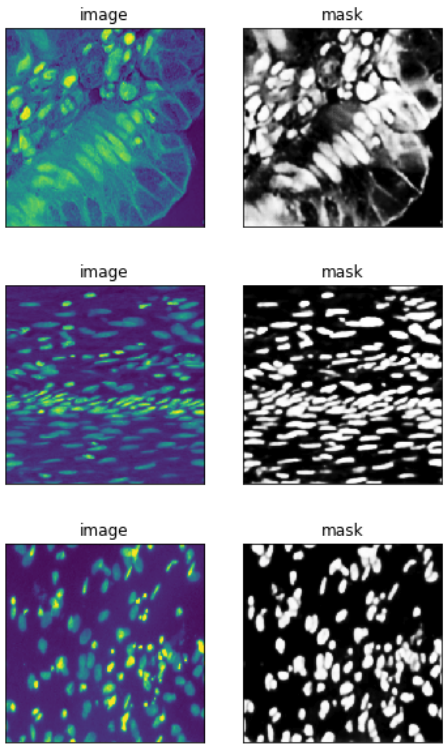


Figure 10: Test Predictions

V. Conclusion

Free-Form Visualization

There are a number of characteristics that I would like to discuss as the conclusion for the problem I am trying to solve. The results which calculates the accuracy in terms of mean IoU over a range of thresholds , saturates at a certain level, one of the characteristic which I found very amazing is the image preprocessing step which was able to boost the results got by my base model and that is the step of normalizing and inverting of the images
I went through some of the image transformation and normalization strategies that are present out there that really fascinates me , and I believe what I have done is similar to contrast stretching, which has helped me boost my scores. For analyzing I applied some of the strategies on the training images and plotted them below.

- **Histogram Equalization**[15] : This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This

allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values

- **Contrast stretching**[16] : Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the contrast in an image by 'stretching' the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows. It differs from the more sophisticated histogram equalization in that it can only apply a linear scaling function to the image pixel values. As a result the 'enhancement' is less harsh. (Most implementations accept a gray scale image as input and produce another gray scale image as output.)

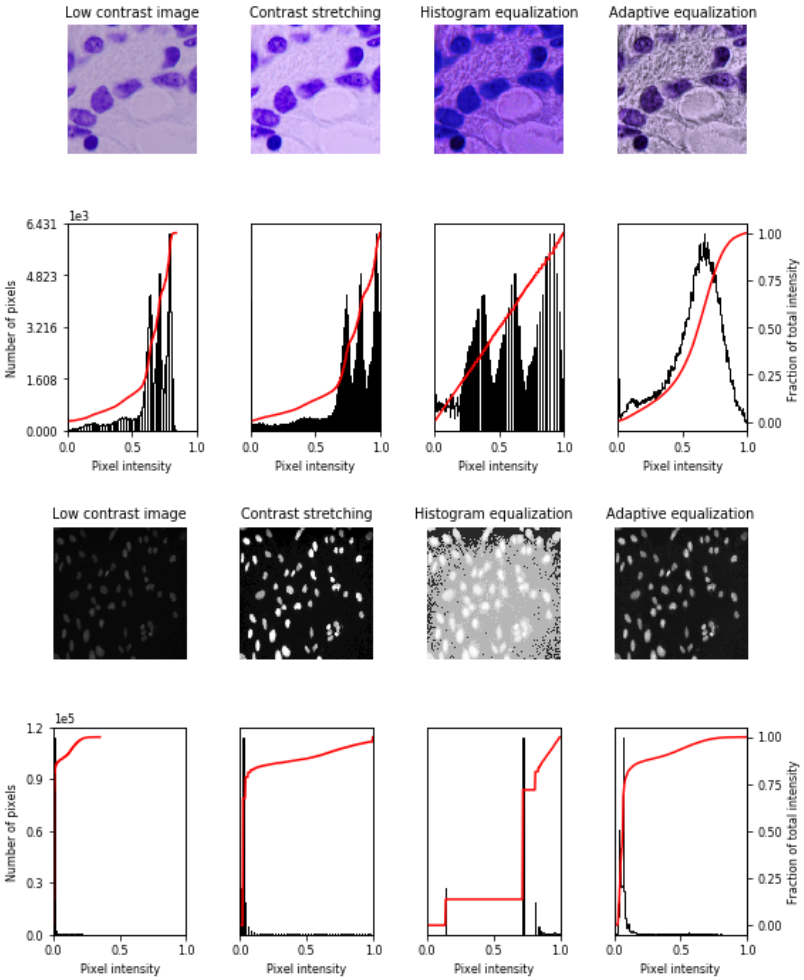


Figure 11: Image Normalizations

Improvements

I would like to put it that I have learned quite a bit about image processing and biomedical images mostly and the way to process them and train them to predict the masks that can detect the nuclei in them. There are a lot of improvements that I would put to propose as I keep digging for a better LB score as the Data Science Bowl comes to an end in 10 days. I would like to put down the most important ones below:

- **Implementing Mask R-CNN in particular**. What is Mask RCNN and how is it helpful? Mask RCNN is based on the principle of object Instance Segmentation, we not only detect the object that is present in the image, but also we detect the segmentation mask of the image. Just to let people know, of the challenges with Mask RNN, the COCO dataset and Cityscapes dataset have very few problems that are solved using the Instance segmentation approach. This is also a fairly new approach. A branch of **Fast/Faster R-CNN**, they are fast and have good accuracy and speed. It is a combination of R-CNN and FCN(Fully convolution Net)

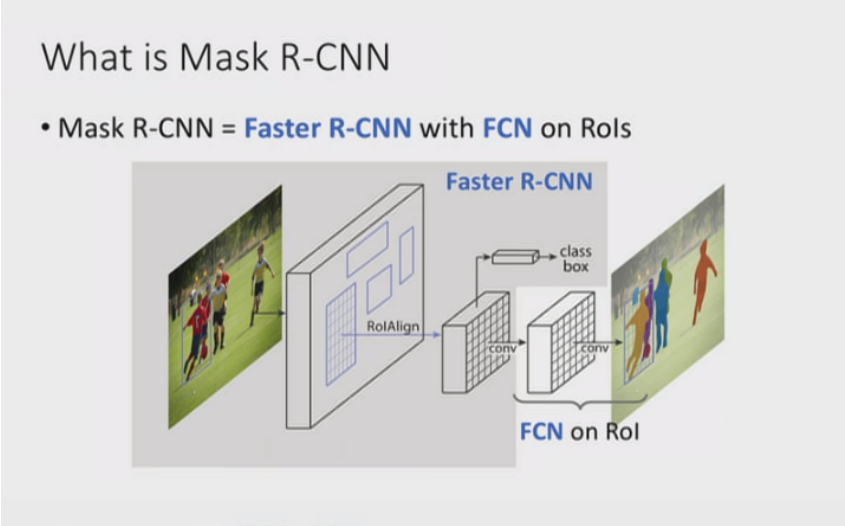


Figure 12: Image Normalizations

- Extensive use of data augmentation, I found a particular package, that will mostly help with this , imgaug[19]. I would like to try this out and check, if there is an improvement
- Morphological post processing by using marker based watershed techniques, to detach some of the nuclei and clean up the the binary masks.There is extensive information here[20]
- **Transfer learning**: I saw some of the kaggle discussions where people have discussed the use of pretrained models and use of resnet50 models .

References

[1] https://stackoverflow.com/questions/33947823/what-is-semantic-segmentation-compared-to-segmentation-and-scene-labeling.

[2] U-net: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/., https://arxiv.org/abs/1505.04597

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

[4] Faster R-CNN: (https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/)

[5] Starter U-net Architecture : https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277

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

[7] https://en.wikipedia.org/wiki/Image_sementation

[8] https://en.wikipedia.org/wiki/Computer_vision

[9] https://en.wikipedia.org/wiki/ImageNet#ImageNet_Challenge

[10] Milan Sonka; Vaclav Hlavac; Roger Boyle (2008).Image Processing, Analysis, and Machine Vision. Thomson.

[11] Reinhard Klette (2014).Concise Computer Vision. Springer. Linda G. Shapiro; George C. Stockman (2001). Computer Vision. Prentice Hall.

[12] Tim Morris (2004). Computer Vision and Image Processing. Palgrave Macmillan.

[13] Bernd JÃďhne; Horst HauÃ§ecker (2000). Computer Vision and Applications,A Guide for Students and Practitioners.

[14] David A. Forsyth; Jean Ponce (2003).Computer Vision, A Modern Approach. Prentice Hall.

[15] https://en.wikipedia.org/wiki/Histogram_equalization

[16] http://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm

[17] http://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_equalize.html

[18] https://arxiv.org/pdf/1703.06870.pdf

[19] https://github.com/aleju/imgaug

[20] http://cmm.ensmp.fr/ beucher/wtshed.html