Capstone Proposal **Sandeep Anand**
Udacity MLND Teammates: None

Domain Background

- This project is a current "Kaggle Competition" called "2018 Data Science Bowl".
- The competition is to create a Nuclei Detector 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**
  - **Case 1:** Considering that there are a large number of images, 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 competition 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
  - **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

Datasets and Inputs

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 to (128, 128, 3) for now using skimage.transform.resize

```
  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)
```

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

Note 1 As Put in the dataset, this dataset is human annotated and will have errors
Note 2 The submission requires that all the output are in Run length encoded Format

Runlength Encoding. Runlength encoding replaces a value that is repeated consecutively with a token that consists of the value and a count of the number of consecutive occurrences (the length of the run)

- Stage1 Train: Training set images (images and annotated masks)
- Stage1 Test: stage 1 test set images (images only, you are predicting the masks)
- Stage2 Test: **(Will be released Later)** stage 2 test set images (images only, you are predicting the masks)
- Submission .csv files as example during stage 1 and stage 2

Solution Statement

Briefly Describing an overview of my idea here, I will base my solution on the **U-net architecture**[2]. I will be tending to use a mixture or keras and Tensorflow for this project, depending on the complexity.
I will have to use a GPU architecture if possible for the training purpose
After I am able to get a benchmark result or more on my U-net architecture, I will use some post processing and image augmentation to see if I could improve my predictions on the test data

Note : This is the start of the project so I might be facing issues with Post processing as well as small change in my methodology while working with the U-net architecture, which I will describe in my final report .

Benchmark Model

- I would compare this to a Starter U-net Architecture which achieves a LB score of 0.277 and will improve on top of that
- 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, though I only trained this simple model on my CPU and with a GPU remotely , I am sure it will perform better Please refer to my submissions at my Kaggle profile **https://www.kaggle.com/crafter707**

```
from keras.preprocessing import image
from keras.models import Model, Sequential
from keras.layers import Dense, GlobalAveragePooling2D, BatchNormalization, Conv2D, Up
from keras import backend as K

simple_cnn= Sequential()
simple_cnn.add(BatchNormalization(input_shape = (IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS),
simple_cnn.add(Conv2D(8, kernel_size=(3,3), padding='same'))
simple_cnn.add(Conv2D(8, kernel_size=(3,3), padding='same'))

simple_cnn.add(Conv2D(16, kernel_size=(3,3), dilation_rate=2, padding='same'))
simple_cnn.add(Conv2D(16, kernel_size=(3,3), dilation_rate=2, padding='same'))
simple_cnn.add(Conv2D(32, kernel_size=(3,3), dilation_rate=2, padding='same'))

simple_cnn.add(Conv2D(16, kernel_size = (1,1), padding = 'same'))
simple_cnn.add(Conv2D(1, kernel_size = (1,1), padding = 'same', activation = 'sigmoid'
simple_cnn.summary()


inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs) #Normalization
encoded_images = simple_cnn(s)
outputs = Dense(1, activation='sigmoid') (encoded_images)

cnnmodel = Model(inputs=[inputs], outputs=[outputs])
cnnmodel.compile(optimizer='adam', loss='binary_crossentropy', metrics=[mean_iou])
```

```
_________________________________________________________________
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
```

Evaluation Metrics[6]

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 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.

Challenges

- Low number of training images present
- Touching cells in the images so have to consider those as two Nuclei instead of one
- Properly defining the IoU function to calculate the mean average precision

Project Design

- Please refer to the **Figure:1 Design Flow** , for the project design flowchart

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/.

[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.

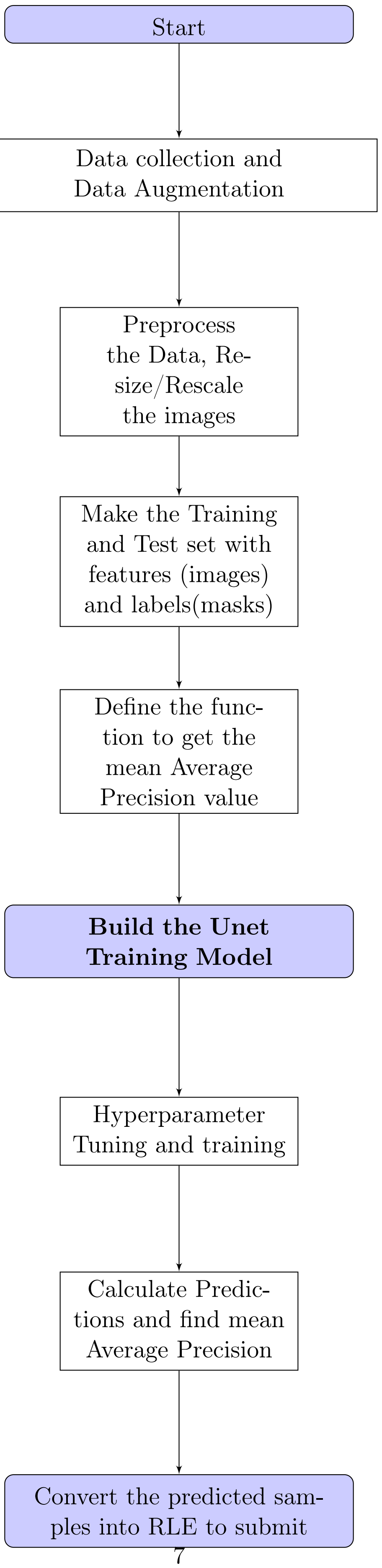
Start
Data collection and Data Augmentation
Preprocess the Data, Resize/Rescale the images
Make the Training and Test set with features (images) and labels(masks)
Define the function to get the mean Average Precision value
Build the Unet Training Model
Hyperparameter Tuning and training
Calculate Predictions and find mean Average Precision
Convert the predicted samples into RLE to submit

Figure 1: Design Flow