OBJECTIVE:-

To enable medical researchers to come with a cure for diseases caused due to bacteria or viruses like corona, faster.

ABSTRACT:-

PROBLEM STATEMENT:

The effects of any bacteria or virus are first observed in the nucleus of the cells hence our goal here is to detect the nucleus and their position in various microscopic images of the cells to aid the researchers to come with a solution faster as at present the solution takes thousands of hours as each image has to be checked manually.

APPROACH WITH SOLUTION:

Our approach to solving the problem of detecting nuclei consists of the following steps:
1-Image Classification – Checking whether the image is having a nuclei or not.

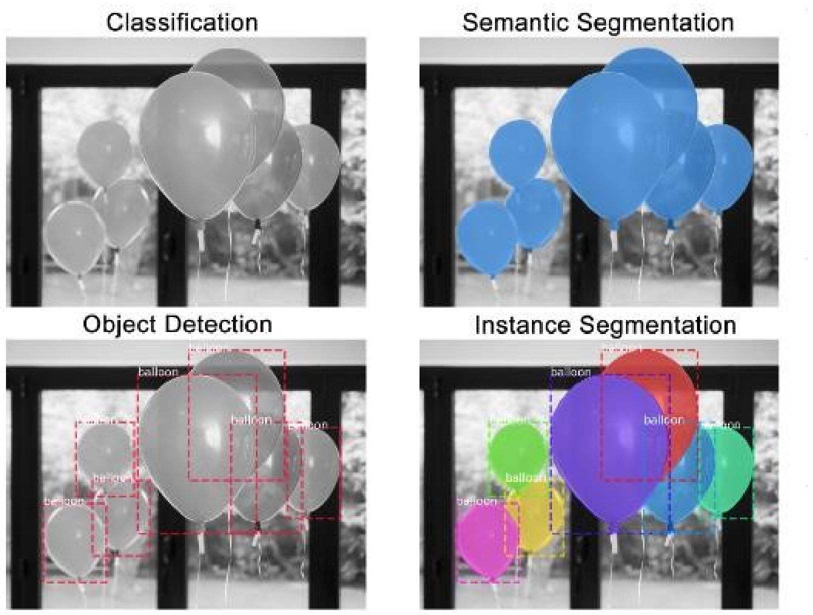
2- Object Detection – Creating the bounding boxes of nuclei in images.

3- Semantic Segmentation – Coloring all pixels of nuclei in a characteristic color.

4- Instance Segmentation(optional)– Coloring the pixel of two different nuclei differently so that we can color different Nucie differently.

We are going to solve this problem using the U-Net and Mask R-CNN Models. The one which will give us the better accuracy, will be kept while the other one will be discarded

Example of above approach -



We will be trying Mask RCNN and U-Net Model for the above segmentation and choose the better one.

Till now we are trying to do Semantic Segmentation, if we succeed and with a proper accuracy we will go to carry out Instance Segmentation.

PROGRESS:-

Until now we have implemented U-NET model for image segmentation. U-NET model consists of 3 phases: **1-Contraction, 2-Bottleneck and 3- Expansion** section as described in the figure

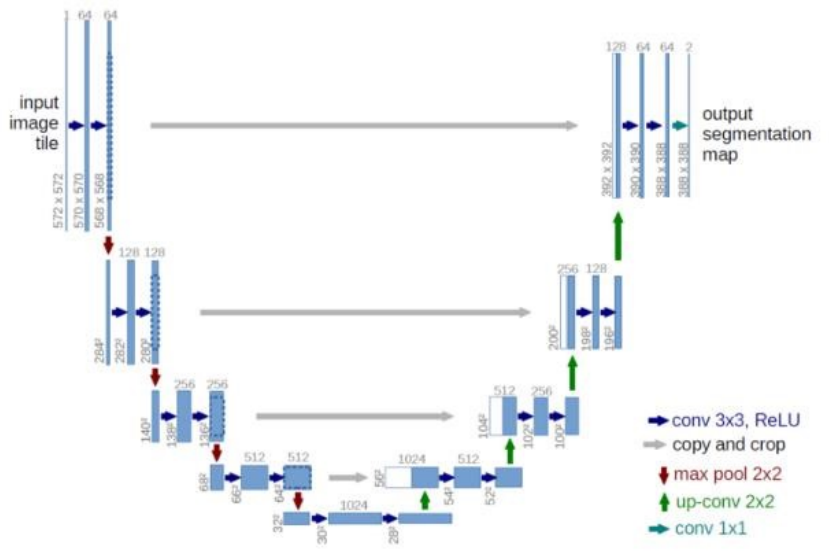


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

First Half i.e. **contraction** also called ***encoder*** is a combination of convolution layer and max pool layers used to extract features. is used to locate the **WHAT** section by telling us whether the nuclei is present or not and also the bounding boxes if present. But it loses a lot of information regarding where which is needed to know where the pixel is regarding that particular nuclei.

Second Half i.e. **expansion** also called **decoder** is a combination deconvolution(transpose convolution) layers that regain the dimension of image and help the computer to locate the **WHERE** information which was lost in the contraction section.

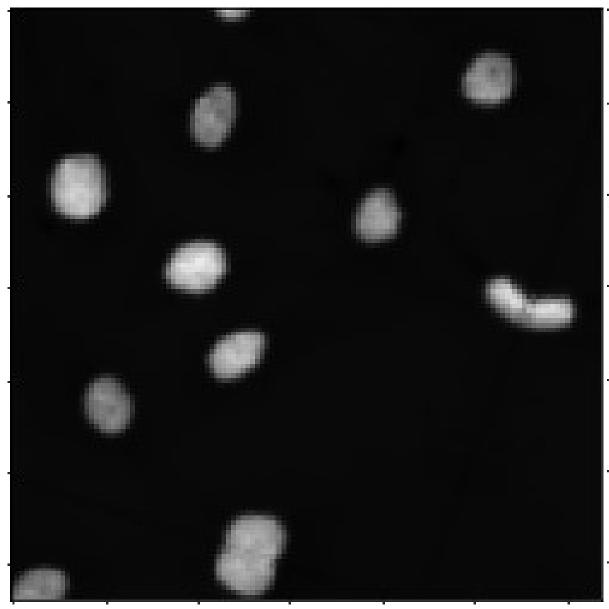
To get a more accurate result we will be using **skip connections** (which prevents the loss of information) between corresponding encoder and decoder stages. Deconvolution layer and Convolution layer are learned from the links provided in references.

<u>OUR WORKFLOW:-</u>

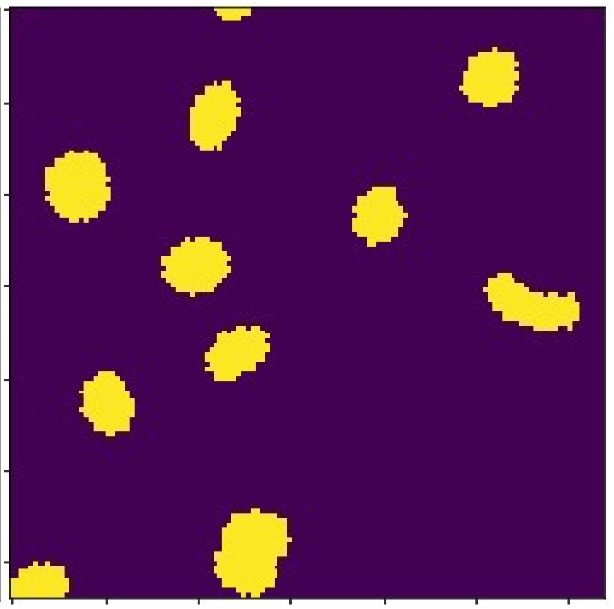
Our data-set includes images of cells and the corresponding mask images. This data set is divided into 3 parts **60% training, 30% validation and 10% test.**

First, we resize all our images towards a lower resolution image as it will help us in reducing the storage and processing time.

Then we performed **DATA AUGMENTATION** to solve the problem of fewer data. Augmentation operation includes mirroring, rotating and shering.



Original cell image



Target segmented image

Then we declared the IoU (intersection over union) metric which will be used for evaluation.

Note- IoU metric is an evaluation metric used measure the accuracy of an object detector on a particular data-set.

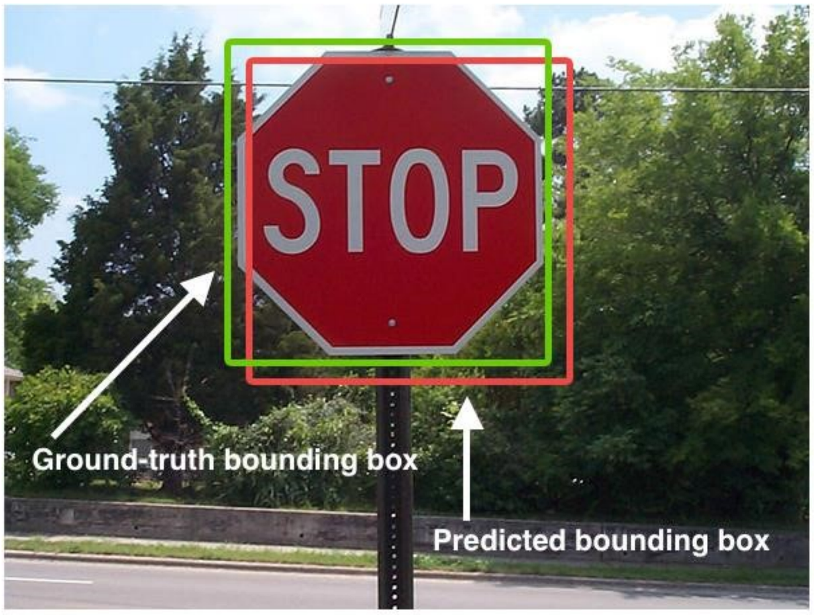
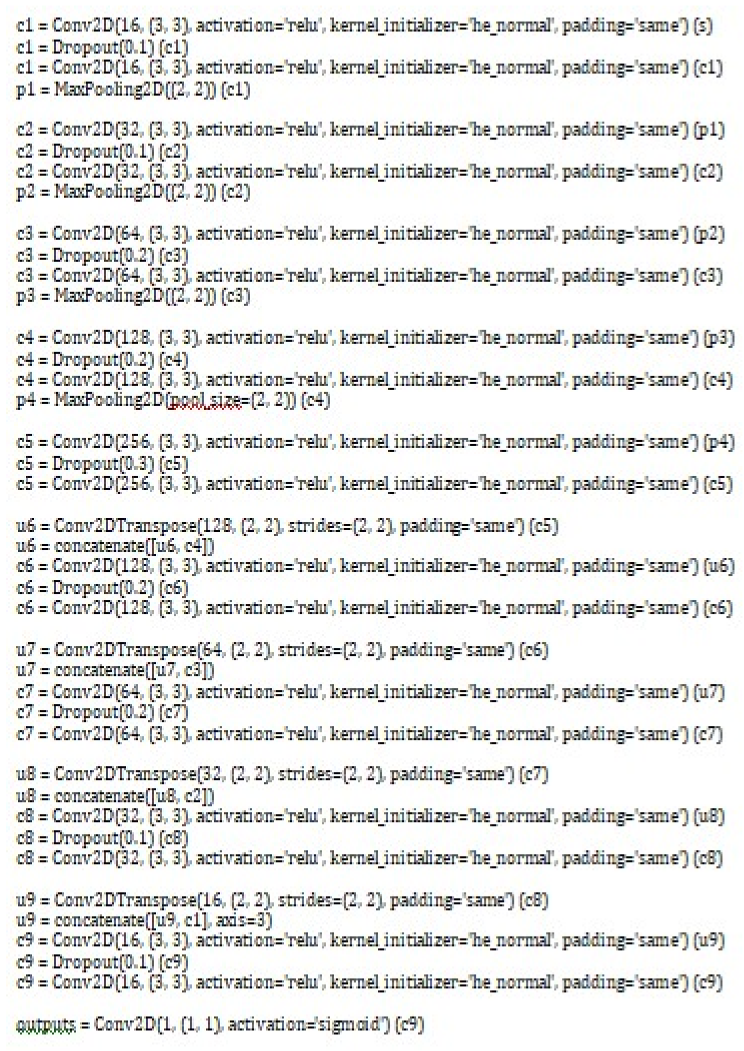


Figure 1: An example of detecting a stop sign in an image. The predicted bounding box is drawn in red while the ground-truth bounding box is drawn in green. Our goal is to compute the Intersection over Union between these bounding box.

IOU_FUNTION

```
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)
                        prec.append(score)
        return K.mean(K.stack(prec), axis=0)
```

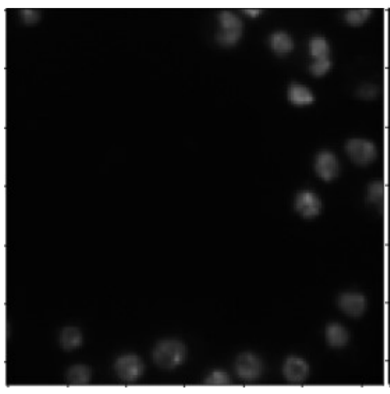
Than we created our U-Net model

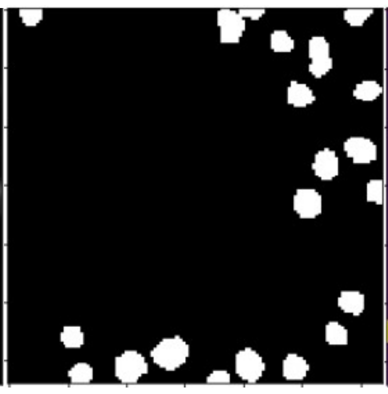


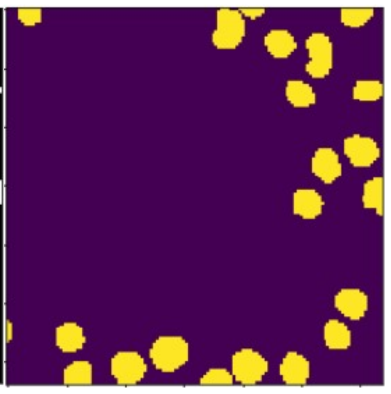
```
c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c5)

u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u6)
c6 = Dropout(0.2) (c6)
c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c6)

u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c6)
u7 = concatenate([u7, c3])
c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u7)
c7 = Dropout(0.2) (c7)
c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c7)

u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c7)
u8 = concatenate([u8, c2])
c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u8)
c8 = Dropout(0.1) (c8)
c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c8)

u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c8)
u9 = concatenate([u9, c1], axis=3)
c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u9)
c9 = Dropout(0.1) (c9)
c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c9)

outputs = Conv2D(1, (1, 1), activation='sigmoid') (c9)
```

Model is compiled with optimizer as “adam”, loss as “binary crossentropy”, metric as mean_iou(defined earlier) and activation function as “relu” and “sigmoid” only for the last layer.

This is the output of a random image:-


Original Image


Target Image


Predicted Image

Till now we have achieved a val_iou = 0.832.

FUTURE PLANS:-

We will be trying to increase val_iou by trying different Models, optimizer, batch size, etc. Then we will be doing this project using Mask R-CNN and compare the results of both.

If we complete these tasks by our second evaluation we will be work on Instances Segmentation.

REFERENCES:-

1-DATA - https://drive.google.com/open?id=1_L8uW-vUgNIZACyURCvbfAKrzSIRGP7p

2- https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

4- https://medium.com/@sh.tsang/review-unet-a-nested-u-net-architecture-biomedical-image-segmentation-57be56859b20

5-https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/

6- https://towardsdatascience.com/semantic-segmentation-popular-architectures-dff0a75f39d0

7- https://towardsdatascience.com/instance-segmentation-using-mask-r-cnn-7f77bdd46abd

8-https://www.coursera.org/specializations/deep-learning

9- https://datascience.stackexchange.com/questions/6107/what-are-deconvolutional-layers

10- https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

11- https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec

GROUP INFORMATION:-

Samyak Tanted	17JE002905
Abhishek Raj Permani	17JE003207
Suyash Srivastava(leader)	17JE003210
Swapnil Narayan	17JE003386
Sparsh Srivastava	17JE003481