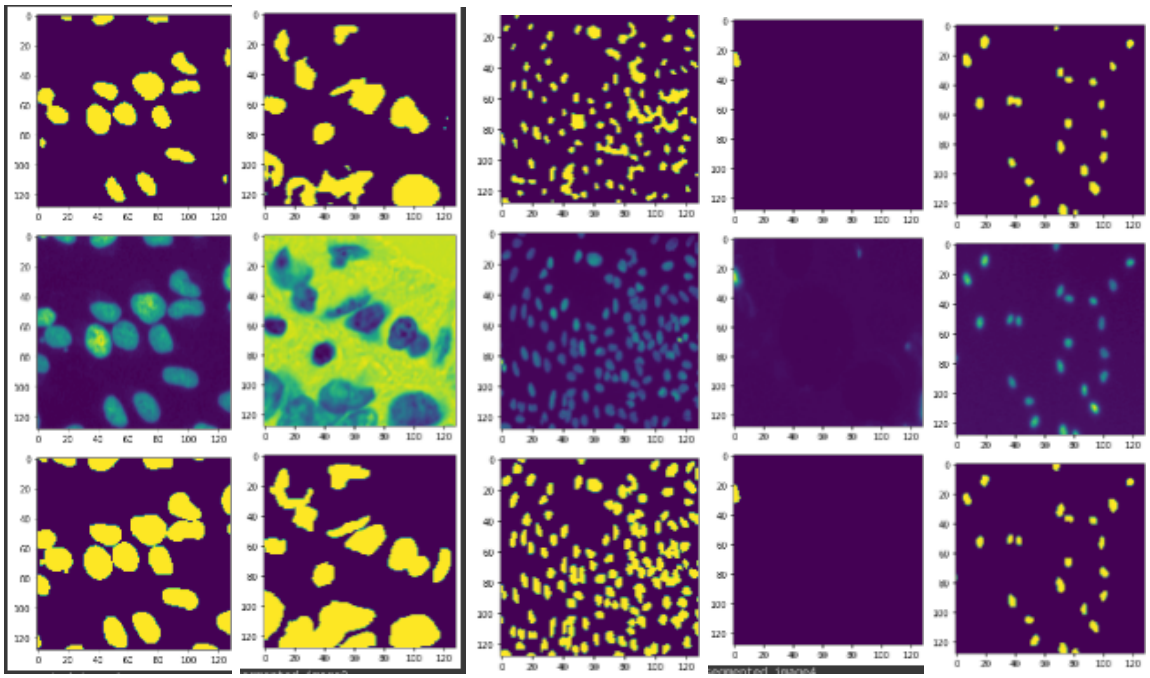
The five segmented testing images, with the top image being the result image, middle as the original image and bottom as the ground truth.

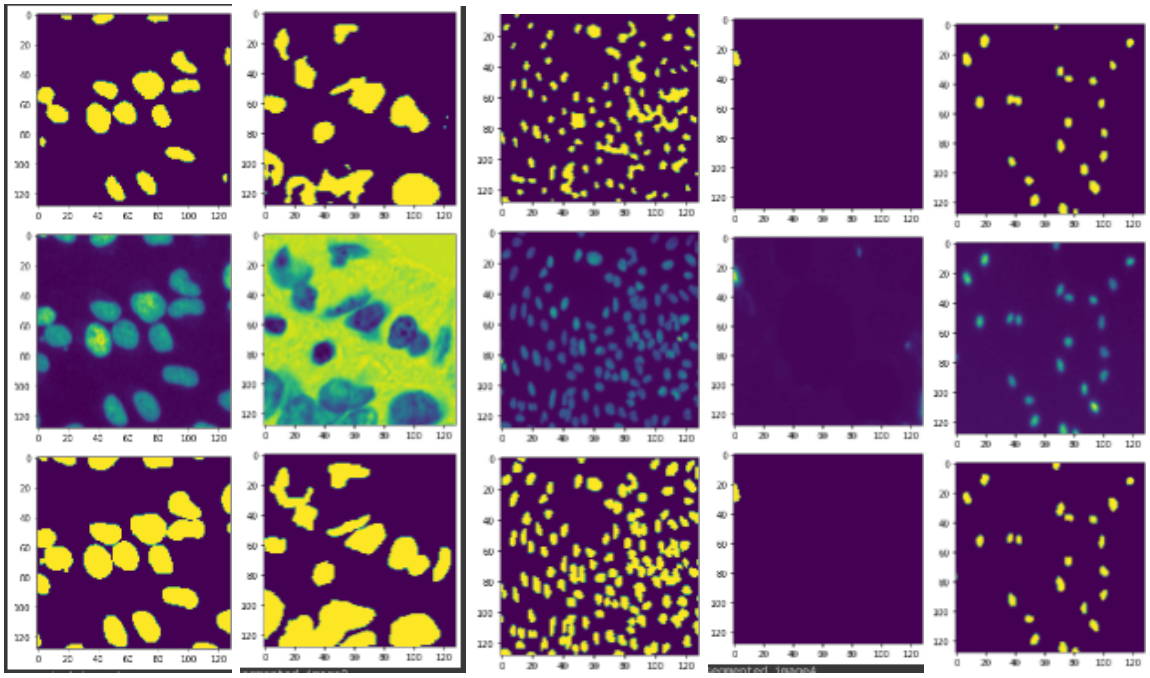


1. Upsampling does not have the keyword stride,which was solved by removing it.
2. Model.fit has the keyword epochs and not nb_epoch, so that was replaced.
3. The datatype being passed into the model needed to be changed, as it does not take in unit8, and so both x_train and y_train was changed to float32 dtype.
4. The conv2DTranspose value needed to have a 2, instead of 128, and so it was solved by fixing it.
5. The value in the model_and_write function had to be changed to pred[i] from seg0.
6. The inputs into model_and_write were changed to float32 dtype as well.
7. Had to round the prediction data values, to make it into 1 or 0.0. It was done so using the np.round operation.

The calculated values are:

```
originalpixelacc: 0.8882489013671875 iou: tf.Tensor(0.66937536, shape=(), dtype=float32) dicecoef: tf.Tensor(0.76482606, shape=(), dtype=float32)
```

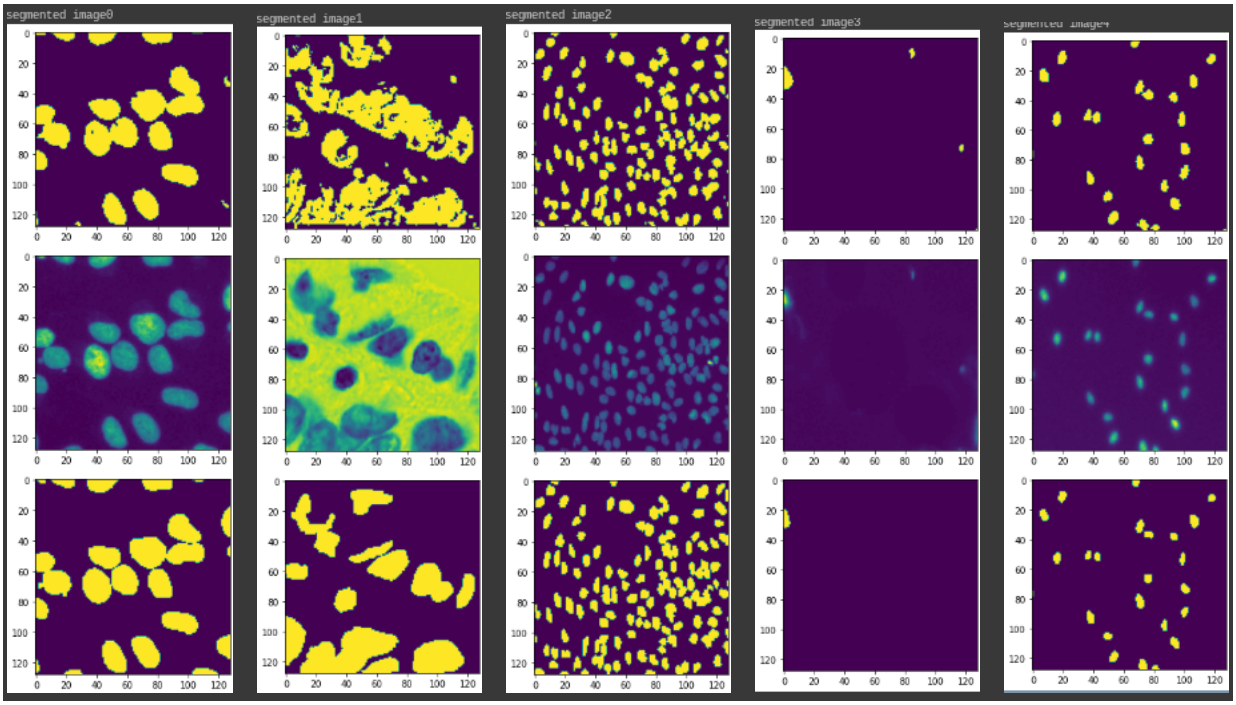
Top image is the result image, middle is the original image and bottom is the ground truth.



1c.
Reported test accuracy and score metrics for UNet architecture.

```
Epoch 20/20
15/15 [==============================] - 8s 564ms/step - loss: 0.1468 -
accuracy: 0.9393 - val_loss: 0.1648 - val_accuracy: 0.9341

DeepLearning_Unetpixelacc: 0.91484375436 iou: tf.Tensor(0.7896389,
shape=(), dtype=float32) dicecoef: tf.Tensor(0.86682993, shape=(),
dtype=float32)
```

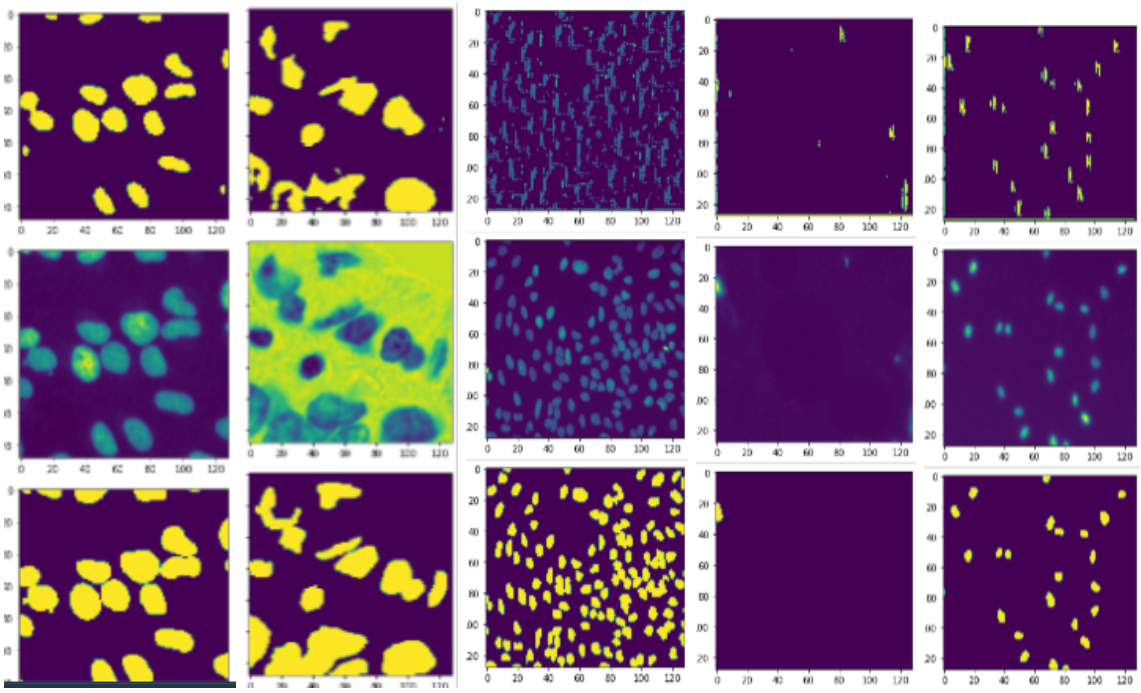


Description of Implementation:
The Deep Learning model architecture consists of an encoder and decoder. In the encoder part, downsampling is done that leads the neural network towards a bottle neck, from where the decoder starts upsampling, finally followed by a softmax output (which is usually a standard choice for the final output layer, for such deep learning networks, as it reacts to low stimulus and contributes to help reconstruct the image better). Each downsample block consists of two convolutional blocks, followed by a max pooling layer that holds a dropout function, and that helps to prevent the network from overfitting. The 2D convolution layer is accompanied by a Relu activation function. It was used as it does not activate all the neurons at the same time and helps the network learn better. The upsample block was composed of a conv2D transpose layer, as now the network was being expanded, and the shapes needed to be maintained, after being downsampled. It also consisted of two convoluted layers that helped to retain the dimensions the image started out with. The bottleneck of the layer was made of two convolutional blocks of size 1024. This whole architecture helped attain the Unet architecture core design, inorder for the image segmentation methods to perform well.

Adam optimizer was used, with keras default learning rate, ran for 20 epochs. Such was chosen, since the accuracy of the model seemed to increase optimally with the parameters, showing that the network was learning well. So, these parameters were maintained for the model. The architecture initially did not consist of two convolutional layers in the blocks, however increasing the conv layer significantly increased the network accuracy to the final output reported, and hence was kept and maintained. Finally, this architecture was therefore maintained as this ‘compressed’ standard Unet architecture did have a good test accuracy and image segmentation metric scores, for our dataset.

2.a

For Gaussian Mixture models(GMM), the images. In the GMM segmentationThe metric accuracy for the GMM was comparatively low (assuming that the images being tested were noisy). However, by altering some parameters, such as the cluster number could drastically change the metric accuracy. Scioy package was used and the cluster value was set to 2. Upgrading the parameter could’ve increased the scoring value.

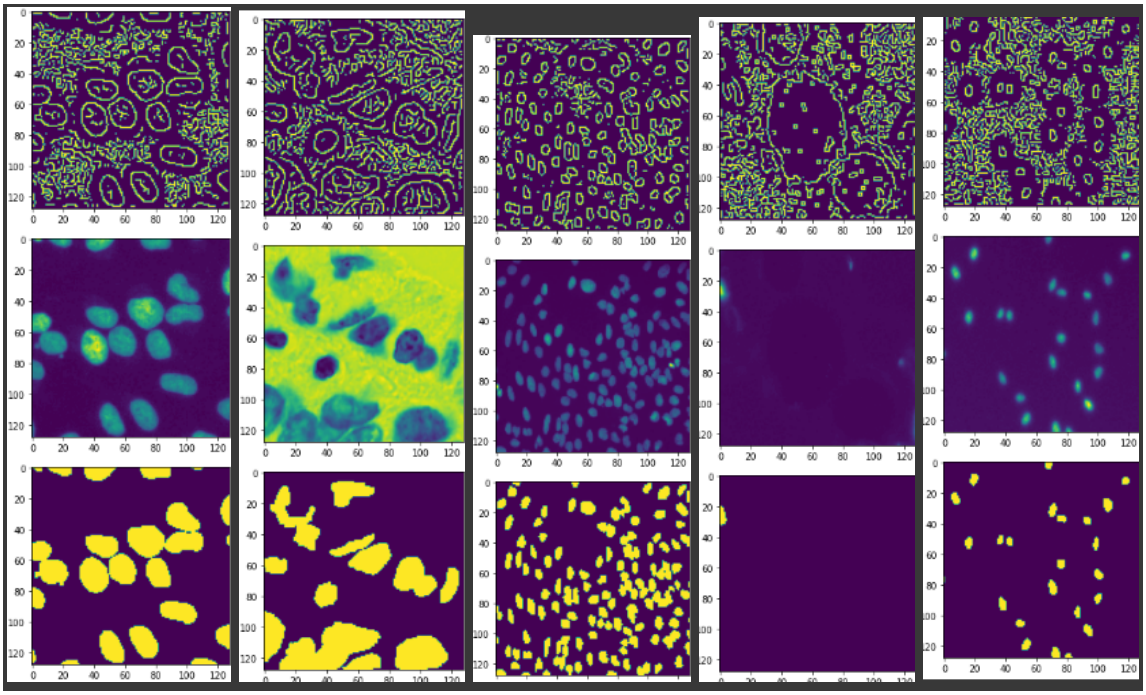


```
originalpixelacc: 0.652167871875 iou: tf.Tensor(0.46789061, shape=(), dtype=float32) dicecoef: tf.Tensor(0.663122954, shape=(), dtype=float32)
```

b.

Region Based Segmentation was performed utilizing both watershed and edge based detection segmentation. The Edge based segmentation held better performance (although extremely low), compared to watershed segmentation method, and hence the images shared below are of Region Based Segmentation utilizing canny edge detection method. This showed that the image had a lot of noise.

```
originalpixelacc: 0.314863321 iou: tf.Tensor(0.111572405, shape=(),
dtype=float32) dicecoef: tf.Tensor(0.2110810265, shape=(), dtype=float32)
```



2c.
The active contour model seemed not suited for the datatype, as it was hard to increase the metric scores. It was done using thresholding and chain approximation method, from cv2 package. Different thresholding values were tried, but the change was not significant. And although the predicted image and ground truth image had corresponding 1.0 and 0.0 values, the pixel accuracy was 0.0.

```
max and min val of img and gt: 1.0 0.0 1.0 0.0
shape of pred in loop: (128, 128, 1)
```

It gave blank images, with small dots, with very poor metric scores.
originalpixelacc: 0.0 iou: tf.Tensor(0.0002566453, shape=(), dtype=float32) dicecoef: tf.Tensor(0.0004919499, shape=(), dtype=float32).

2d. Normalized cut.
It was done using the standard skimage library, with certain changed tuned according to the dataset. However, poor metric scored were obtained.
originalpixelacc: 0.11030395507812506 with low dice score and iou.

2 Subpart
The implemented deep learning model performed the best compared to the other ones. The accuracy of the model, including the three other metrics, was also the highest, by approximately 10% more (as reported in the respective sections above).