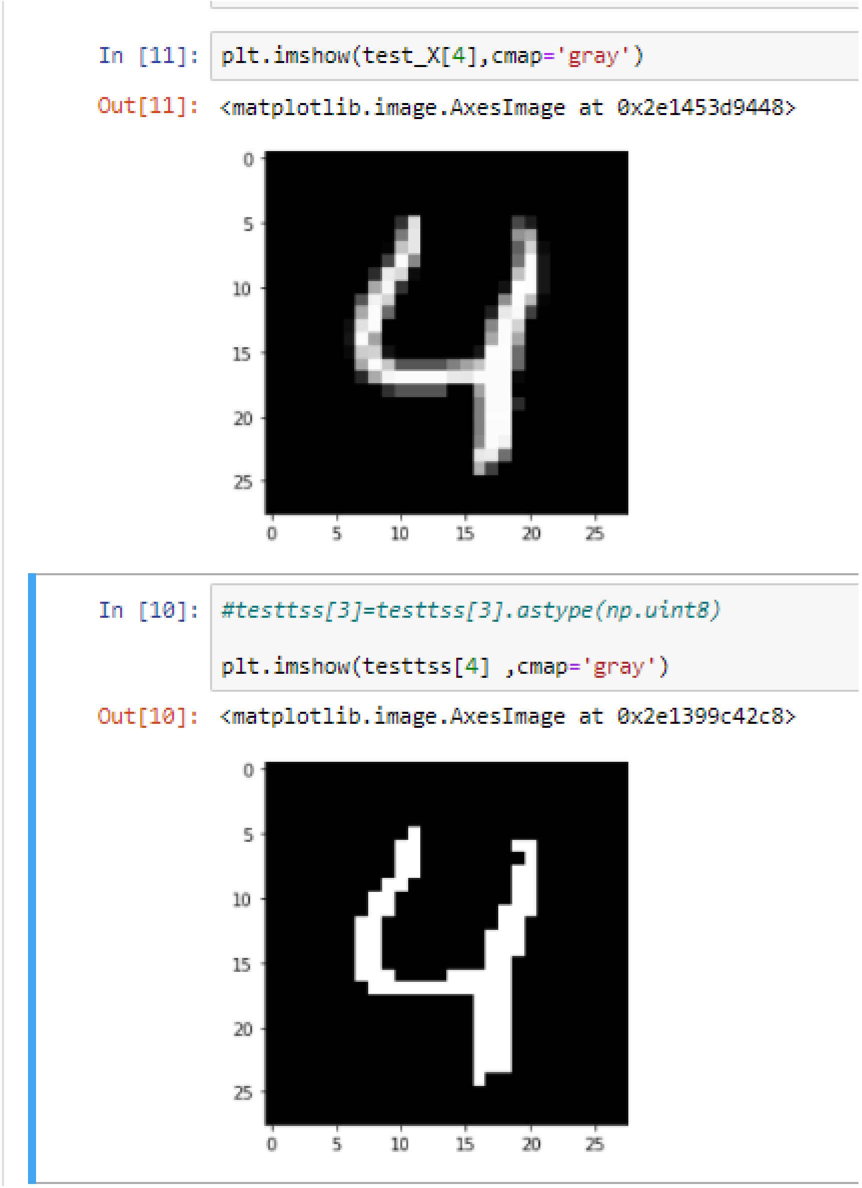
Q1.

1. Done -Straightforward



```
In [11]: plt.imshow(test_X[4],cmap='gray')

Out[11]: <matplotlib.image.AxesImage at 0x2e1453d9448>
```

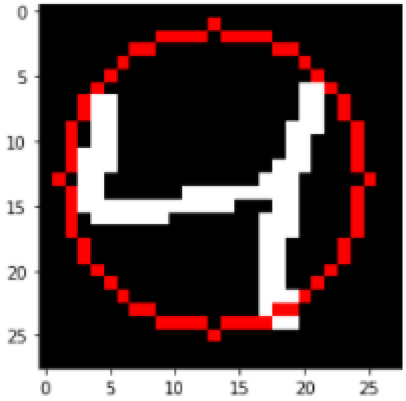
```
In [10]: #testtss[3]=testtss[3].astype(np.uint8)

         plt.imshow(testtss[4] ,cmap='gray')

Out[10]: <matplotlib.image.AxesImage at 0x2e1399c42c8>
```

Above-Orignal Image
Below- TSS thresholded image.
Done this for both training and test sets.

2

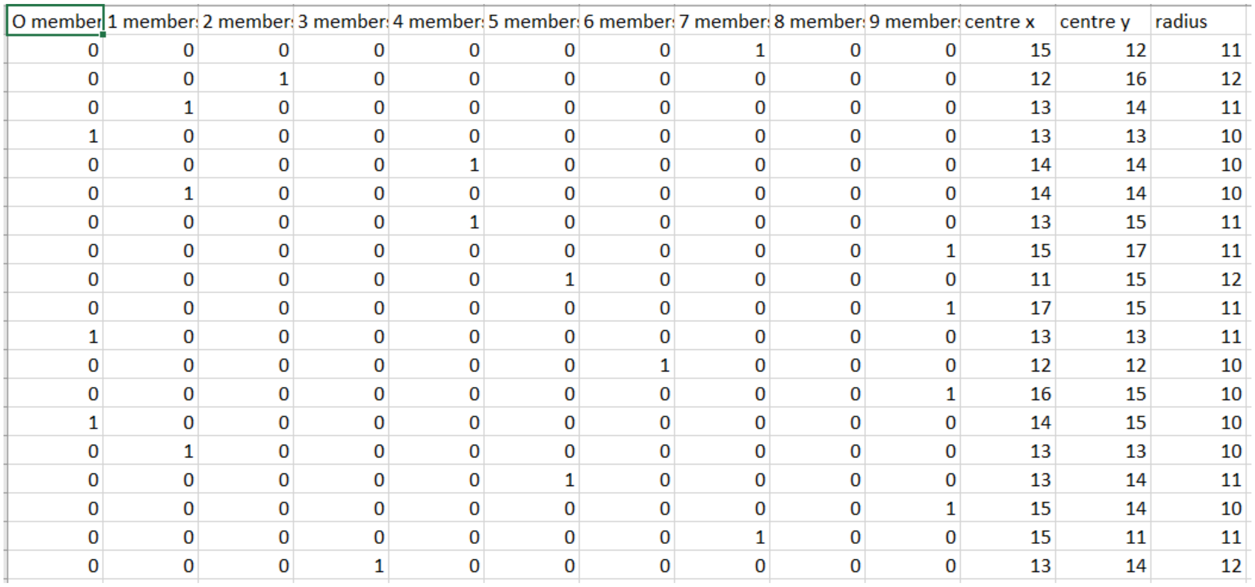
Done.

Used open cv’s minenclosingcircle() function for this.



The circles are not exact as it is not possible to draw a perfect appearing circle in 28x28 image. The function returns coordinates and radius in decimals so some pixels are left due to rounding off of coordinates and radius.

This was performed for both training and testing sets.



O member	1 member	2 member	3 member	4 member	5 member	6 member	7 member	8 member	9 member	centre x	centre y	radius
0	0	0	0	0	0	0	1	0	0	15	12	11
0	0	1	0	0	0	0	0	0	0	12	16	12
0	1	0	0	0	0	0	0	0	0	13	14	11
1	0	0	0	0	0	0	0	0	0	13	13	10
0	0	0	0	1	0	0	0	0	0	14	14	10
0	1	0	0	0	0	0	0	0	0	14	14	10
0	0	0	0	1	0	0	0	0	0	13	15	11
0	0	0	0	0	0	0	0	0	1	15	17	11
0	0	0	0	0	1	0	0	0	0	11	15	12
0	0	0	0	0	0	0	0	0	1	17	15	11
1	0	0	0	0	0	0	0	0	0	13	13	11
0	0	0	0	0	0	1	0	0	0	12	12	10
0	0	0	0	0	0	0	0	0	1	16	15	10
1	0	0	0	0	0	0	0	0	0	14	15	10
0	1	0	0	0	0	0	0	0	0	13	13	10
0	0	0	0	0	1	0	0	0	0	13	14	11
0	0	0	0	0	0	0	0	0	1	15	14	10
0	0	0	0	0	0	0	1	0	0	15	11	11
0	0	0	1	0	0	0	0	0	0	13	14	12

A snapshot of created csv file. The created dataset consists of 13 columns.

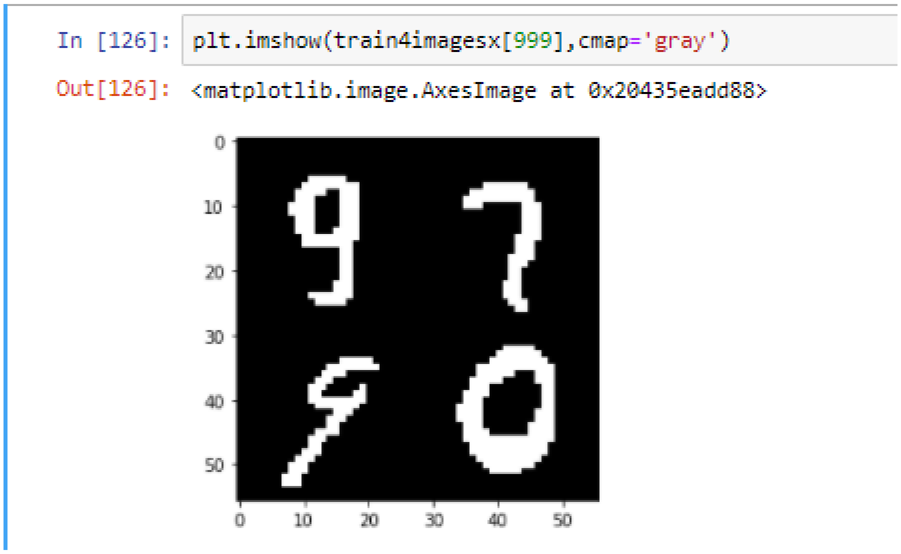
Column 1- O member - The label for image is 0 or not. 1 if image label is 0 else 0.

Similarly for columns 2 to 10

Columnn 11 is x coordinate of centre of calculated circle, column 12 is y coordinate of centre and column 13 is the radius.

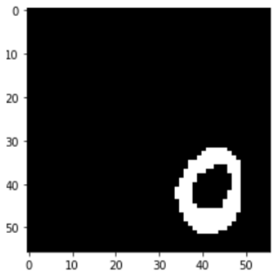
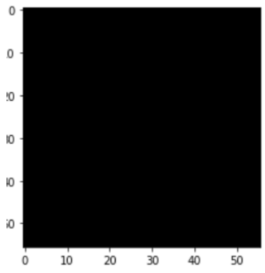
Q1 . 3)

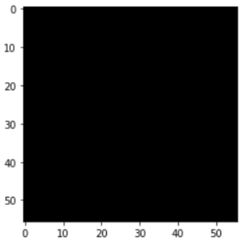
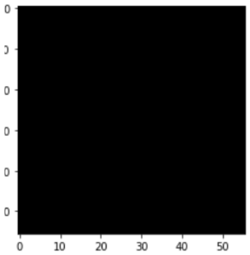
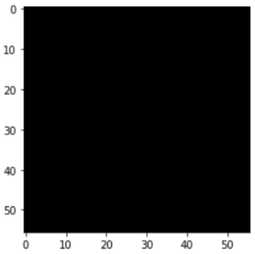
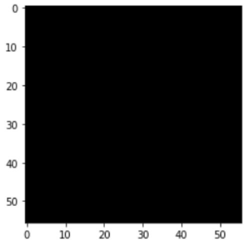
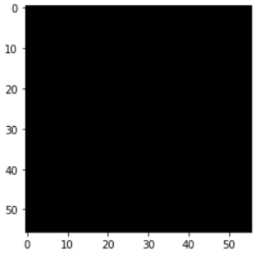
Generated 4 random numbers in each iterations and obtained corresponding images at these random indices from training set(same procedure for creating testing set)

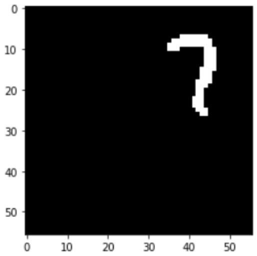
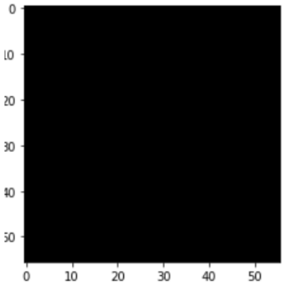
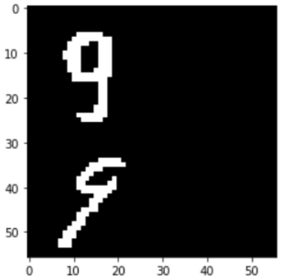
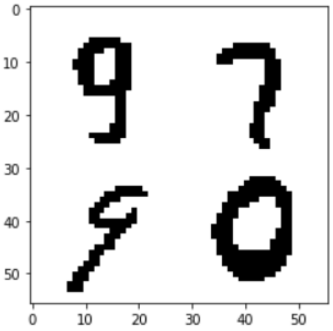


```
In [126]: plt.imshow(train4imagesx[999],cmap='gray')
Out[126]: <matplotlib.image.AxesImage at 0x20435eadd88>
```

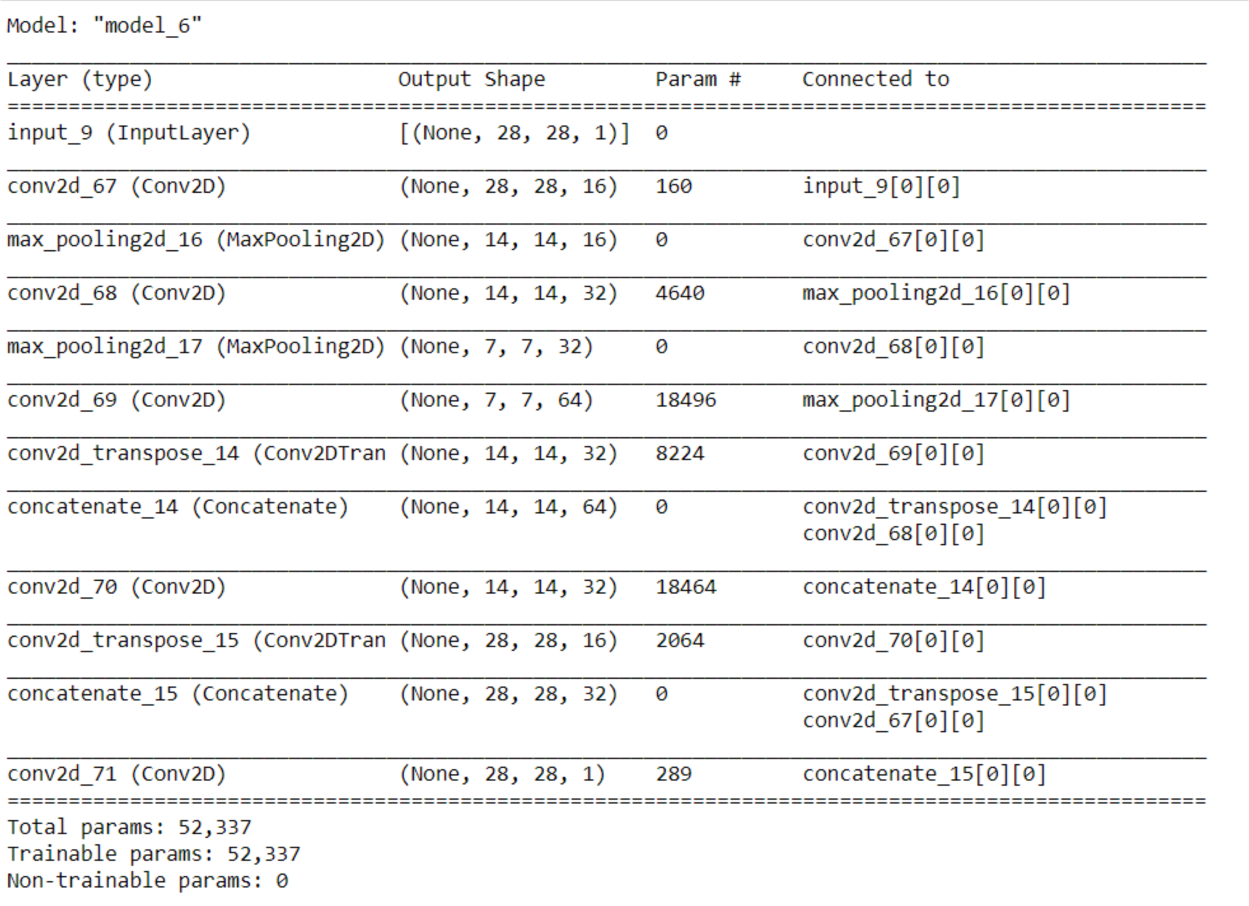
New obtained image after concatenation is 56x56.
Now 11 56x56 maps are obtained for each created image
Each map is for each label
Map 1 has pixels which have the label 0 as 1 else 0
Map 2 has pixels which have the label 1
Map 3 - Map 10 are similar to the above 56*56 maps
Map 11 highlights the background pixels as 1 else 0.

Map 1	
Map 2	

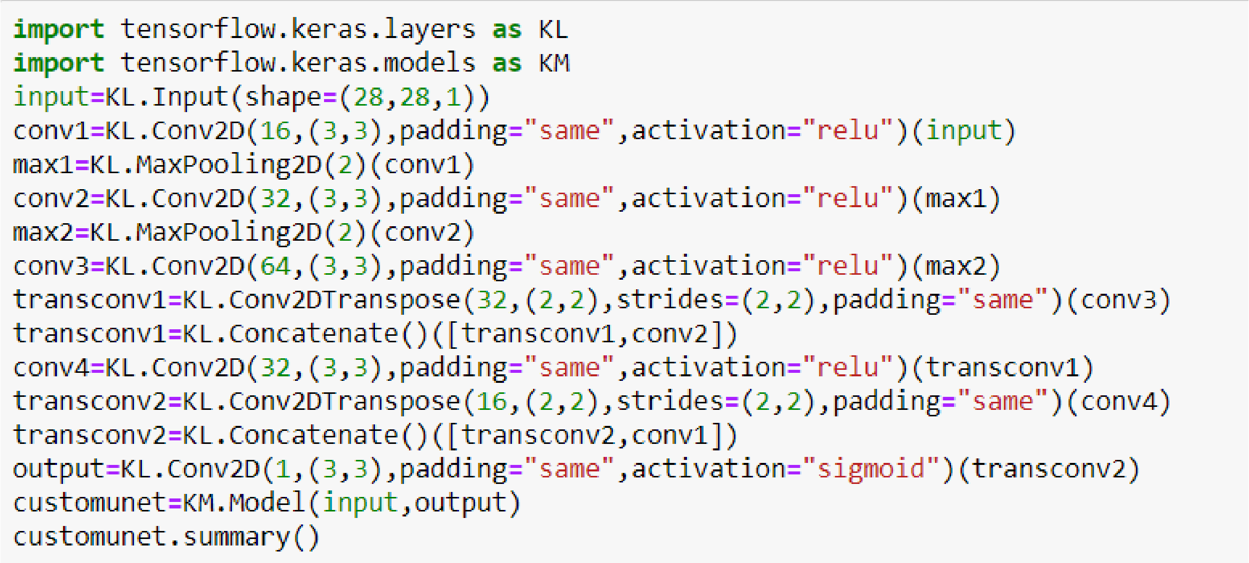
Map 3	
Map 4	
Map 5	
Map 6	
Map 7	

Map 8	
Map 9	
Map 10	
Map 11	

Q2. Designed a DL model inspired by unet which uses transpose convolution and concatenation after the convolutions. The DL model has a contraction part which reduces the dimension but increases the depth followed by a expansion part(achieved by transpose convolutions) which increases the dimensions but decreases the depth. Used Relu in all filters except the output one. Used sigmoid for the output layer as all data was between 0 and 1.



```
Model: "model_6"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_9 (InputLayer)            [(None, 28, 28, 1)]  0

__________________________________________________________________________________________________
conv2d_67 (Conv2D)              (None, 28, 28, 16)   160         input_9[0][0]
__________________________________________________________________________________________________
max_pooling2d_16 (MaxPooling2D) (None, 14, 14, 16)   0           conv2d_67[0][0]
__________________________________________________________________________________________________
conv2d_68 (Conv2D)              (None, 14, 14, 32)   4640        max_pooling2d_16[0][0]
__________________________________________________________________________________________________
max_pooling2d_17 (MaxPooling2D) (None, 7, 7, 32)     0           conv2d_68[0][0]
__________________________________________________________________________________________________
conv2d_69 (Conv2D)              (None, 7, 7, 64)     18496       max_pooling2d_17[0][0]
__________________________________________________________________________________________________
conv2d_transpose_14 (Conv2DTran (None, 14, 14, 32)   8224        conv2d_69[0][0]
__________________________________________________________________________________________________
concatenate_14 (Concatenate)    (None, 14, 14, 64)   0           conv2d_transpose_14[0][0]
                                                                 conv2d_68[0][0]
__________________________________________________________________________________________________
conv2d_70 (Conv2D)              (None, 14, 14, 32)   18464       concatenate_14[0][0]
__________________________________________________________________________________________________
conv2d_transpose_15 (Conv2DTran (None, 28, 28, 16)   2064        conv2d_70[0][0]
__________________________________________________________________________________________________
concatenate_15 (Concatenate)    (None, 28, 28, 32)   0           conv2d_transpose_15[0][0]
                                                                 conv2d_67[0][0]
__________________________________________________________________________________________________
conv2d_71 (Conv2D)              (None, 28, 28, 1)    289         concatenate_15[0][0]
==================================================================================================
Total params: 52,337
Trainable params: 52,337
Non-trainable params: 0
```

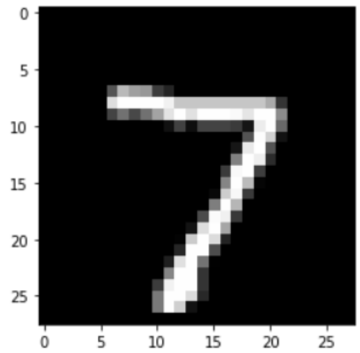


```python
import tensorflow.keras.layers as KL
import tensorflow.keras.models as KM
input=KL.Input(shape=(28,28,1))
conv1=KL.Conv2D(16,(3,3),padding="same",activation="relu")(input)
max1=KL.MaxPooling2D(2)(conv1)
conv2=KL.Conv2D(32,(3,3),padding="same",activation="relu")(max1)
max2=KL.MaxPooling2D(2)(conv2)
conv3=KL.Conv2D(64,(3,3),padding="same",activation="relu")(max2)
transconv1=KL.Conv2DTranspose(32,(2,2),strides=(2,2),padding="same")(conv3)
transconv1=KL.Concatenate()([transconv1,conv2])
conv4=KL.Conv2D(32,(3,3),padding="same",activation="relu")(transconv1)
transconv2=KL.Conv2DTranspose(16,(2,2),strides=(2,2),padding="same")(conv4)
transconv2=KL.Concatenate()([transconv2,conv1])
output=KL.Conv2D(1,(3,3),padding="same",activation="sigmoid")(transconv2)
customunet=KM.Model(input,output)
customunet.summary()
```

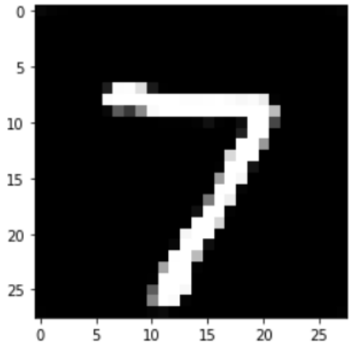
Input- image from mnist database

Training label- Images created in Q1 first part by tss bases thresholding.

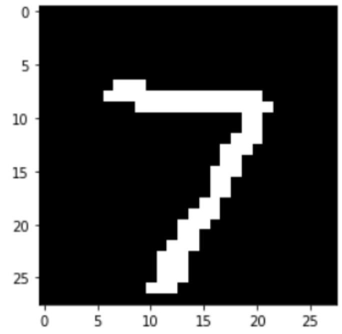
Input-



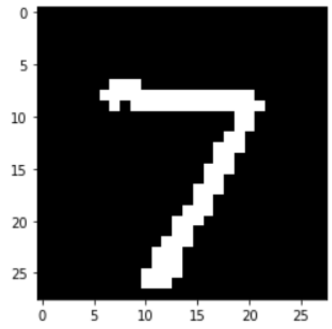
Output by model-

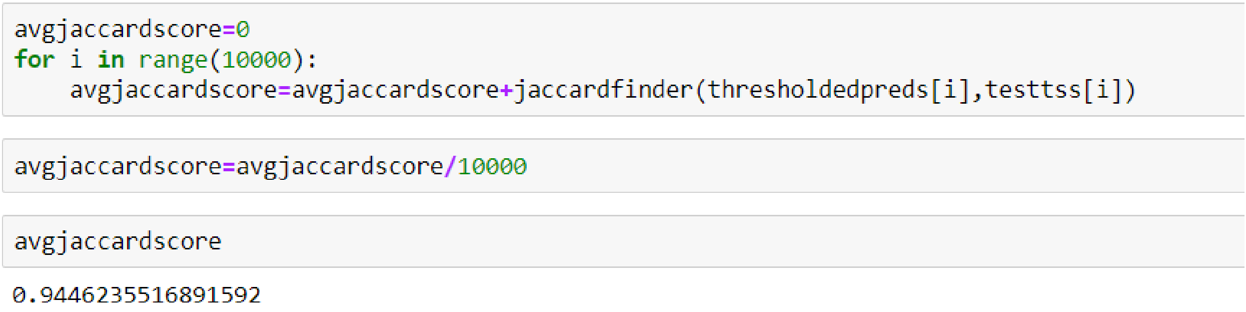


Thresholding the output(threshold=0.5)



Ground truth label-





```
avgjaccardscore=0
for i in range(10000):
    avgjaccardscore=avgjaccardscore+jaccardfinder(thresholdedpreds[i],testtss[i])
```

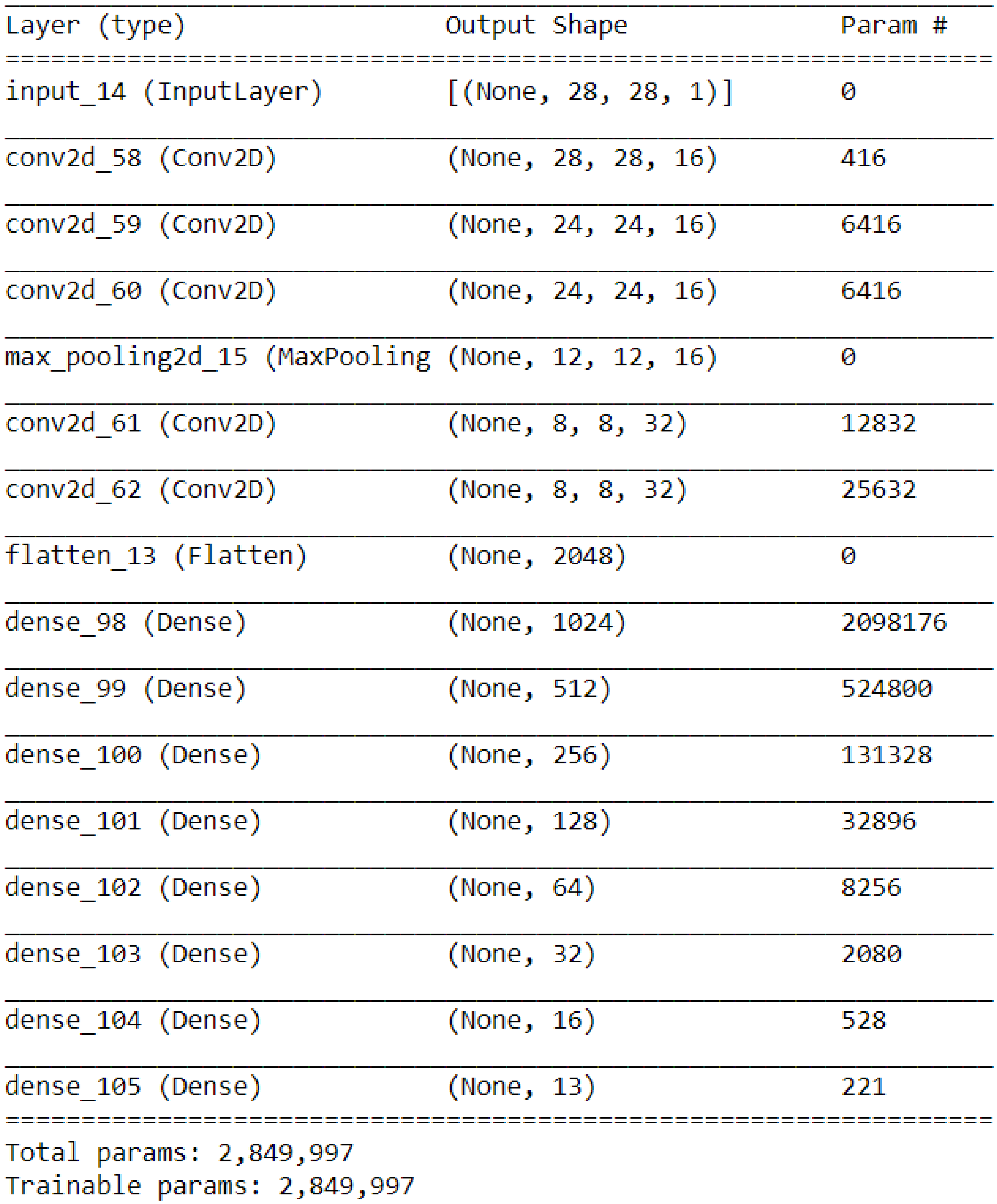
```
avgjaccardscore=avgjaccardscore/10000
```

```
avgjaccardscore
```

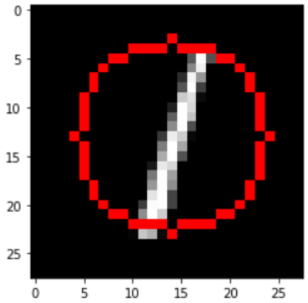
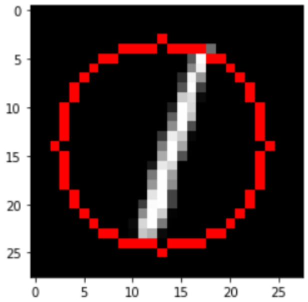
```
0.9446235516891592
```

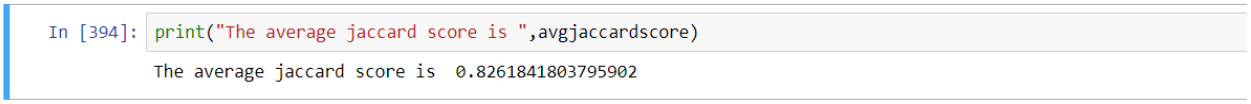
Model achieved a jaccard score of 0.9446 for the testing set.

Q3.
Some data preprocessing was need divided the x and y coordinated by 28 and the radius by 14. All data points were now between 0 and 1.
Designed a conventional CNN. With relu as activation in all layers except the output layer. Used Sigmoid in Output layer(as all output is between 0 and 1). Used custom loss function.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_14 (InputLayer)        [(None, 28, 28, 1)]       0

_________________________________________________________________
conv2d_58 (Conv2D)           (None, 28, 28, 16)        416

_________________________________________________________________
conv2d_59 (Conv2D)           (None, 24, 24, 16)        6416

_________________________________________________________________
conv2d_60 (Conv2D)           (None, 24, 24, 16)        6416

_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 12, 12, 16)        0

_________________________________________________________________
conv2d_61 (Conv2D)           (None, 8, 8, 32)          12832

_________________________________________________________________
conv2d_62 (Conv2D)           (None, 8, 8, 32)          25632

_________________________________________________________________
flatten_13 (Flatten)         (None, 2048)              0

_________________________________________________________________
dense_98 (Dense)             (None, 1024)              2098176

_________________________________________________________________
dense_99 (Dense)             (None, 512)               524800

_________________________________________________________________
dense_100 (Dense)            (None, 256)               131328

_________________________________________________________________
dense_101 (Dense)            (None, 128)               32896

_________________________________________________________________
dense_102 (Dense)            (None, 64)                8256

_________________________________________________________________
dense_103 (Dense)            (None, 32)                2080

_________________________________________________________________
dense_104 (Dense)            (None, 16)                528

_________________________________________________________________
dense_105 (Dense)            (None, 13)                221
=================================================================
Total params: 2,849,997
Trainable params: 2,849,997
```

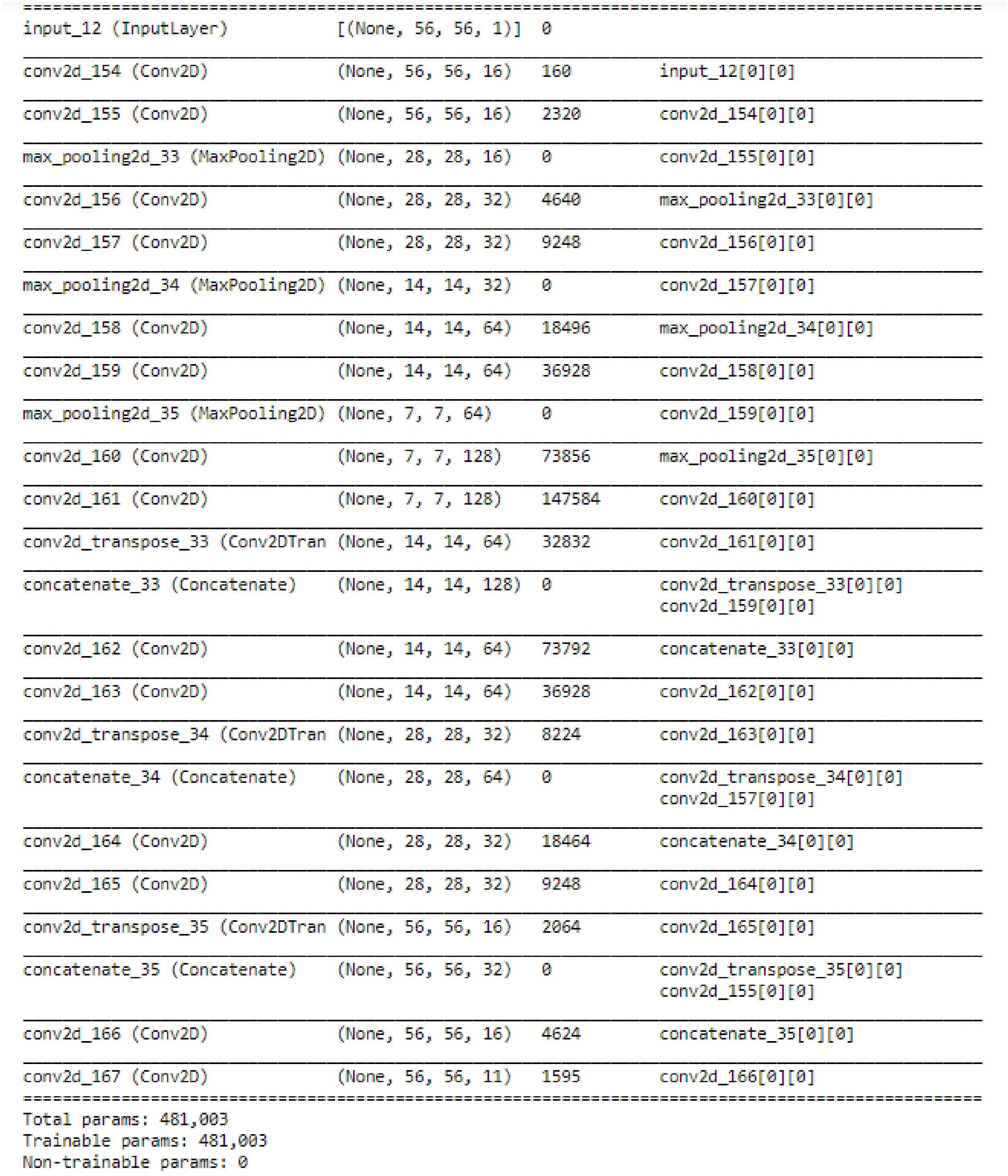
Prediction by Model	
Ground Truth	



```
In [394]: print("The average jaccard score is ",avgjaccardscore)

          The average jaccard score is  0.8261841803795902
```

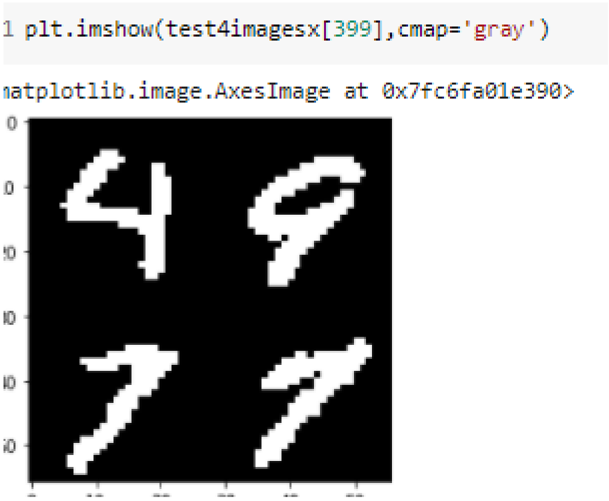
Designed a custom loss function used categorical_crossentropy for columns 1-10(responsible for classification) and Mean Squared Error loss for columns 11-13(circle centre coordinates and radius)
The average Jaccard Score came out to be 0.826 or about 0.83.(Jaccard Score calculated as 0 when model classification was wrong.)

Q4.
Model Inspired by U-Net. Input consists of 4 concatenated images and ground truth label consists of 11 maps(explained in Q1 part 3)



```
==================================================================================================
input_12 (InputLayer)           [(None, 56, 56, 1)]  0

__________________________________________________________________________________________________
conv2d_154 (Conv2D)             (None, 56, 56, 16)   160         input_12[0][0]
__________________________________________________________________________________________________
conv2d_155 (Conv2D)             (None, 56, 56, 16)   2320        conv2d_154[0][0]
__________________________________________________________________________________________________
max_pooling2d_33 (MaxPooling2D) (None, 28, 28, 16)   0           conv2d_155[0][0]
__________________________________________________________________________________________________
conv2d_156 (Conv2D)             (None, 28, 28, 32)   4640        max_pooling2d_33[0][0]
__________________________________________________________________________________________________
conv2d_157 (Conv2D)             (None, 28, 28, 32)   9248        conv2d_156[0][0]
__________________________________________________________________________________________________
max_pooling2d_34 (MaxPooling2D) (None, 14, 14, 32)   0           conv2d_157[0][0]
__________________________________________________________________________________________________
conv2d_158 (Conv2D)             (None, 14, 14, 64)   18496       max_pooling2d_34[0][0]
__________________________________________________________________________________________________
conv2d_159 (Conv2D)             (None, 14, 14, 64)   36928       conv2d_158[0][0]
__________________________________________________________________________________________________
max_pooling2d_35 (MaxPooling2D) (None, 7, 7, 64)     0           conv2d_159[0][0]
__________________________________________________________________________________________________
conv2d_160 (Conv2D)             (None, 7, 7, 128)    73856       max_pooling2d_35[0][0]
__________________________________________________________________________________________________
conv2d_161 (Conv2D)             (None, 7, 7, 128)    147584      conv2d_160[0][0]
__________________________________________________________________________________________________
conv2d_transpose_33 (Conv2DTran (None, 14, 14, 64)   32832       conv2d_161[0][0]
__________________________________________________________________________________________________
concatenate_33 (Concatenate)    (None, 14, 14, 128)  0           conv2d_transpose_33[0][0]
                                                                 conv2d_159[0][0]
__________________________________________________________________________________________________
conv2d_162 (Conv2D)             (None, 14, 14, 64)   73792       concatenate_33[0][0]
__________________________________________________________________________________________________
conv2d_163 (Conv2D)             (None, 14, 14, 64)   36928       conv2d_162[0][0]
__________________________________________________________________________________________________
conv2d_transpose_34 (Conv2DTran (None, 28, 28, 32)   8224        conv2d_163[0][0]
__________________________________________________________________________________________________
concatenate_34 (Concatenate)    (None, 28, 28, 64)   0           conv2d_transpose_34[0][0]
                                                                 conv2d_157[0][0]
__________________________________________________________________________________________________
conv2d_164 (Conv2D)             (None, 28, 28, 32)   18464       concatenate_34[0][0]
__________________________________________________________________________________________________
conv2d_165 (Conv2D)             (None, 28, 28, 32)   9248        conv2d_164[0][0]
__________________________________________________________________________________________________
conv2d_transpose_35 (Conv2DTran (None, 56, 56, 16)   2064        conv2d_165[0][0]
__________________________________________________________________________________________________
concatenate_35 (Concatenate)    (None, 56, 56, 32)   0           conv2d_transpose_35[0][0]
                                                                 conv2d_155[0][0]
__________________________________________________________________________________________________
conv2d_166 (Conv2D)             (None, 56, 56, 16)   4624        concatenate_35[0][0]
__________________________________________________________________________________________________
conv2d_167 (Conv2D)             (None, 56, 56, 11)   1595        conv2d_166[0][0]
==================================================================================================
Total params: 481,003
Trainable params: 481,003
Non-trainable params: 0
```

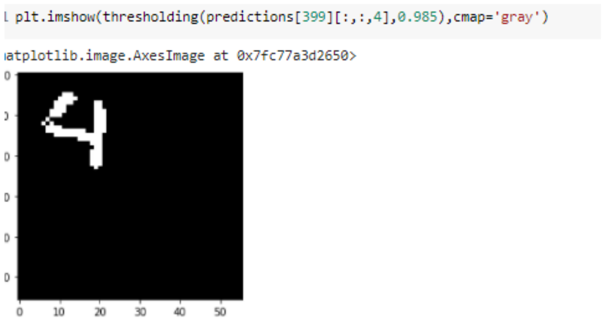
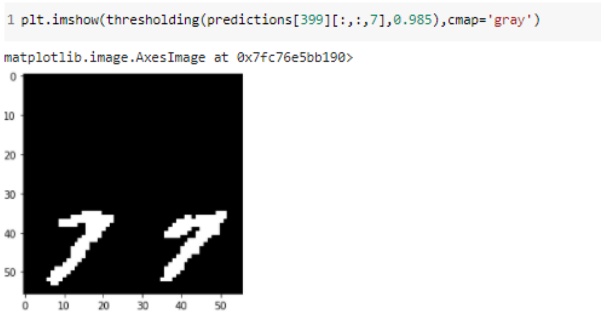
Input-

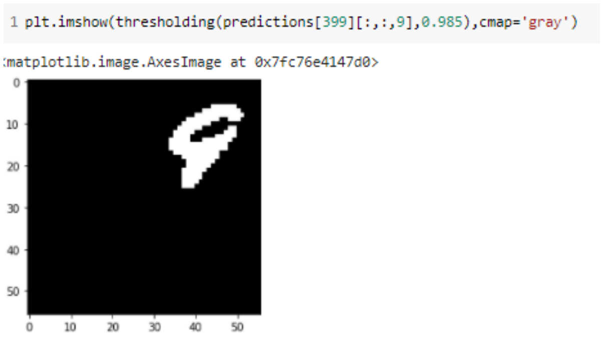


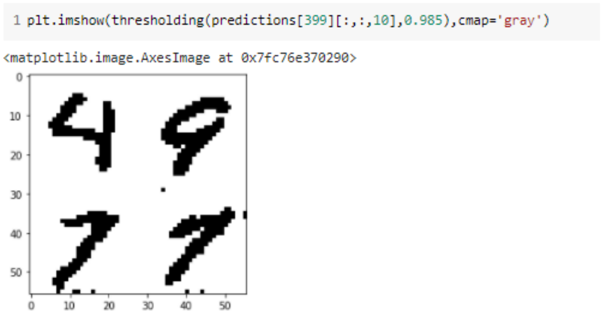
```
plt.imshow(test4imagesx[399],cmap='gray')
```

```
matplotlib.image.AxesImage at 0x7fc6fa01e390>
```

Only showing maps for 4, 9 and 7 ie maps 5,10 and 8 predicted by our model after thresholding. You can have a look at the ipynb for more information.

Map for 4 predicted by model	 `plt.imshow(thresholding(predictions[399][:,:,4],0.985),cmap='gray')` `atplotlib.image.AxesImage at 0x7fc77a3d2650>`
Map for 7 predicted by model	 `plt.imshow(thresholding(predictions[399][:,:,7],0.985),cmap='gray')` `matplotlib.image.AxesImage at 0x7fc76e5bb190>`

Map for 9 predicted by model	
	`plt.imshow(thresholding(predictions[399][:,:,9],0.985),cmap='gray')`
	`<matplotlib.image.AxesImage at 0x7fc76e4147d0>`
	 0
	10
	20
	30
	40
	50
	0
	10
	20
	30
	40
	50

Model for background predicted by model	
	`plt.imshow(thresholding(predictions[399][:,:,10],0.985),cmap='gray')`
	`<matplotlib.image.AxesImage at 0x7fc76e370290>`
	 0
	10
	20
	30
	40
	50
	0
	10
	20
	30
	40
	50

Model produced a jaccard score of 0.9204527383628228 for 1000 test images.(after thresholding model predictions.)