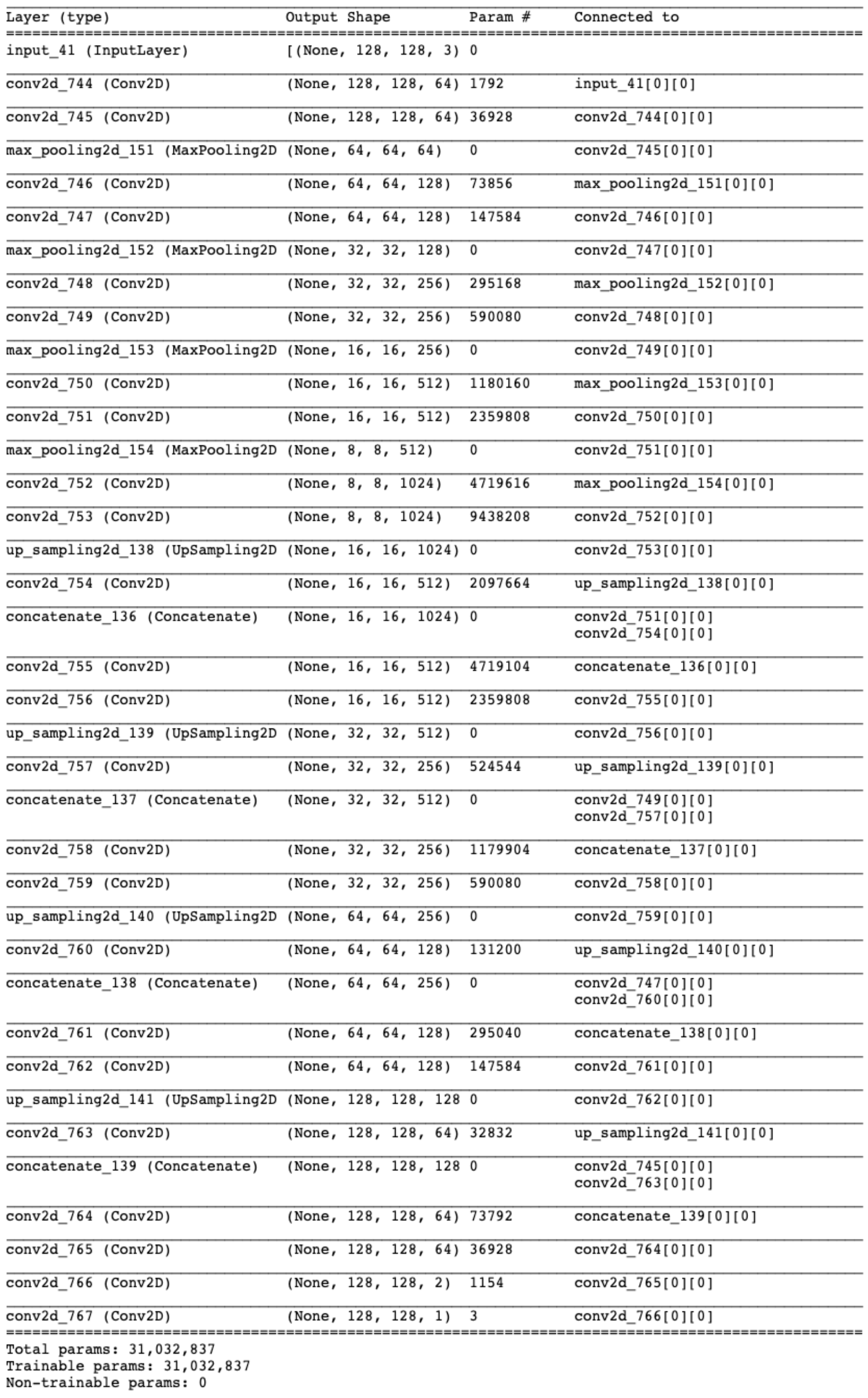
Fan Vincent Ka Chun 1002563380

Link to weights: https://drive.google.com/drive/folders/1stC_wj5IYEg2mrqbamMHWJkAKZlj2QzP?usp=sharing

Question 1

1.1 After implementing my u-net model with the following structure:



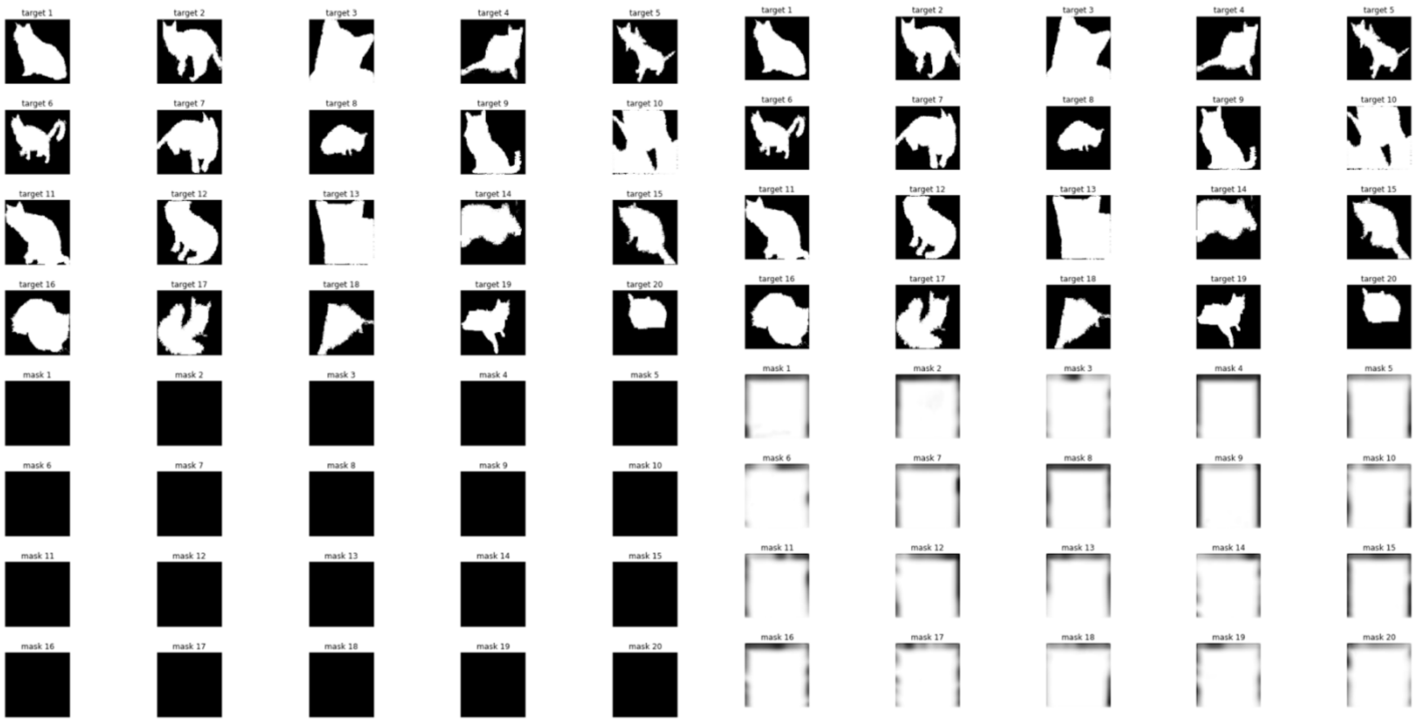
```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_41 (InputLayer)           [(None, 128, 128, 3) 0

conv2d_744 (Conv2D)             (None, 128, 128, 64) 1792        input_41[0][0]

conv2d_745 (Conv2D)             (None, 128, 128, 64) 36928       conv2d_744[0][0]

max_pooling2d_151 (MaxPooling2D (None, 64, 64, 64)   0           conv2d_745[0][0]

conv2d_746 (Conv2D)             (None, 64, 64, 128)  73856       max_pooling2d_151[0][0]

conv2d_747 (Conv2D)             (None, 64, 64, 128)  147584      conv2d_746[0][0]

max_pooling2d_152 (MaxPooling2D (None, 32, 32, 128)  0           conv2d_747[0][0]

conv2d_748 (Conv2D)             (None, 32, 32, 256)  295168      max_pooling2d_152[0][0]

conv2d_749 (Conv2D)             (None, 32, 32, 256)  590080      conv2d_748[0][0]

max_pooling2d_153 (MaxPooling2D (None, 16, 16, 256)  0           conv2d_749[0][0]

conv2d_750 (Conv2D)             (None, 16, 16, 512)  1180160     max_pooling2d_153[0][0]

conv2d_751 (Conv2D)             (None, 16, 16, 512)  2359808     conv2d_750[0][0]

max_pooling2d_154 (MaxPooling2D (None, 8, 8, 512)    0           conv2d_751[0][0]

conv2d_752 (Conv2D)             (None, 8, 8, 1024)   4719616     max_pooling2d_154[0][0]

conv2d_753 (Conv2D)             (None, 8, 8, 1024)   9438208     conv2d_752[0][0]

up_sampling2d_138 (UpSampling2D (None, 16, 16, 1024) 0           conv2d_753[0][0]

conv2d_754 (Conv2D)             (None, 16, 16, 512)  2097664     up_sampling2d_138[0][0]

concatenate_136 (Concatenate)   (None, 16, 16, 1024) 0           conv2d_751[0][0]
                                                                 conv2d_754[0][0]

conv2d_755 (Conv2D)             (None, 16, 16, 512)  4719104     concatenate_136[0][0]

conv2d_756 (Conv2D)             (None, 16, 16, 512)  2359808     conv2d_755[0][0]

up_sampling2d_139 (UpSampling2D (None, 32, 32, 512)  0           conv2d_756[0][0]

conv2d_757 (Conv2D)             (None, 32, 32, 256)  524544      up_sampling2d_139[0][0]

concatenate_137 (Concatenate)   (None, 32, 32, 512)  0           conv2d_749[0][0]
                                                                 conv2d_757[0][0]

conv2d_758 (Conv2D)             (None, 32, 32, 256)  1179904     concatenate_137[0][0]

conv2d_759 (Conv2D)             (None, 32, 32, 256)  590080      conv2d_758[0][0]

up_sampling2d_140 (UpSampling2D (None, 64, 64, 256)  0           conv2d_759[0][0]

conv2d_760 (Conv2D)             (None, 64, 64, 128)  131200      up_sampling2d_140[0][0]

concatenate_138 (Concatenate)   (None, 64, 64, 256)  0           conv2d_747[0][0]
                                                                 conv2d_760[0][0]

conv2d_761 (Conv2D)             (None, 64, 64, 128)  295040      concatenate_138[0][0]

conv2d_762 (Conv2D)             (None, 64, 64, 128)  147584      conv2d_761[0][0]

up_sampling2d_141 (UpSampling2D (None, 128, 128, 128 0           conv2d_762[0][0]

conv2d_763 (Conv2D)             (None, 128, 128, 64) 32832       up_sampling2d_141[0][0]

concatenate_139 (Concatenate)   (None, 128, 128, 128 0           conv2d_745[0][0]
                                                                 conv2d_763[0][0]

conv2d_764 (Conv2D)             (None, 128, 128, 64) 73792       concatenate_139[0][0]

conv2d_765 (Conv2D)             (None, 128, 128, 64) 36928       conv2d_764[0][0]

conv2d_766 (Conv2D)             (None, 128, 128, 2)  1154        conv2d_765[0][0]

conv2d_767 (Conv2D)             (None, 128, 128, 1)  3           conv2d_766[0][0]
==================================================================================================
Total params: 31,032,837
Trainable params: 31,032,837
Non-trainable params: 0
```

We can see that the output is 128 x 128 which is exactly the dimension we want.
I have chosen binary cross entropy and median squared error as my loss functions. The results are following after training:

BCE	`acc: 0.5101 - dice_coef: 0.5550`
MSE	`acc: 0.5101 - dice_coef: 0.5329`

Fan Vincent Ka Chun 1002563380

Here are the predicted masks on the test set for each model:



Left: BCE results Right: MSE results

From the image above, it is easy to see that the predictions are completely wrong and there is some issue within my implementation. However, after many attempts and rebuilding the model with different method. The result is either pitch black or a bright white squared patch in the center. This also explains why I am achieving accuracy and dice coefficient around 50% because there is about half the picture that's all black or all white.

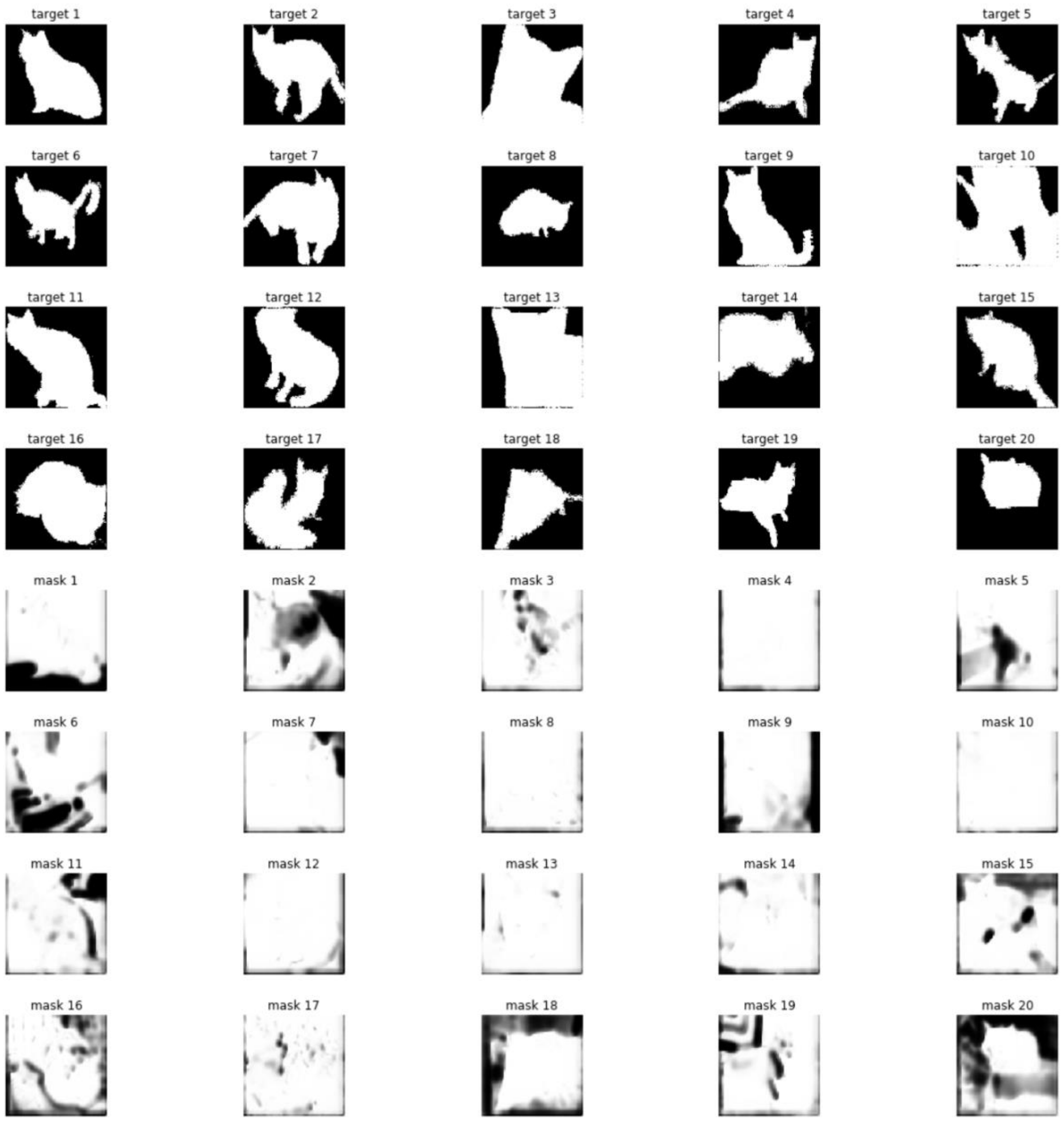
1.2 I have performed rotation, shifting vertically, shifting horizontally, and flipping the image. The procedure to produce more images is as follow: Firstly, I flipped the original image and mask 3 times to get 4 images. Secondly, I take each for these 4 images and masks and perform random rotation, shifting vertically and horizontally 4 times to results in 16 images and 16 masks for every original image and mask.

Of course, I could further increase the number of copies, however due to time constraint and hardware limitation plus late discovery of Google Colab, it would have taken 8 minutes for each epoch on my machine.

Here are the results and predictions on the test set:

```
acc: 0.5434 - dice_coef: 0.5402
```

Fan Vincent Ka Chun 1002563380



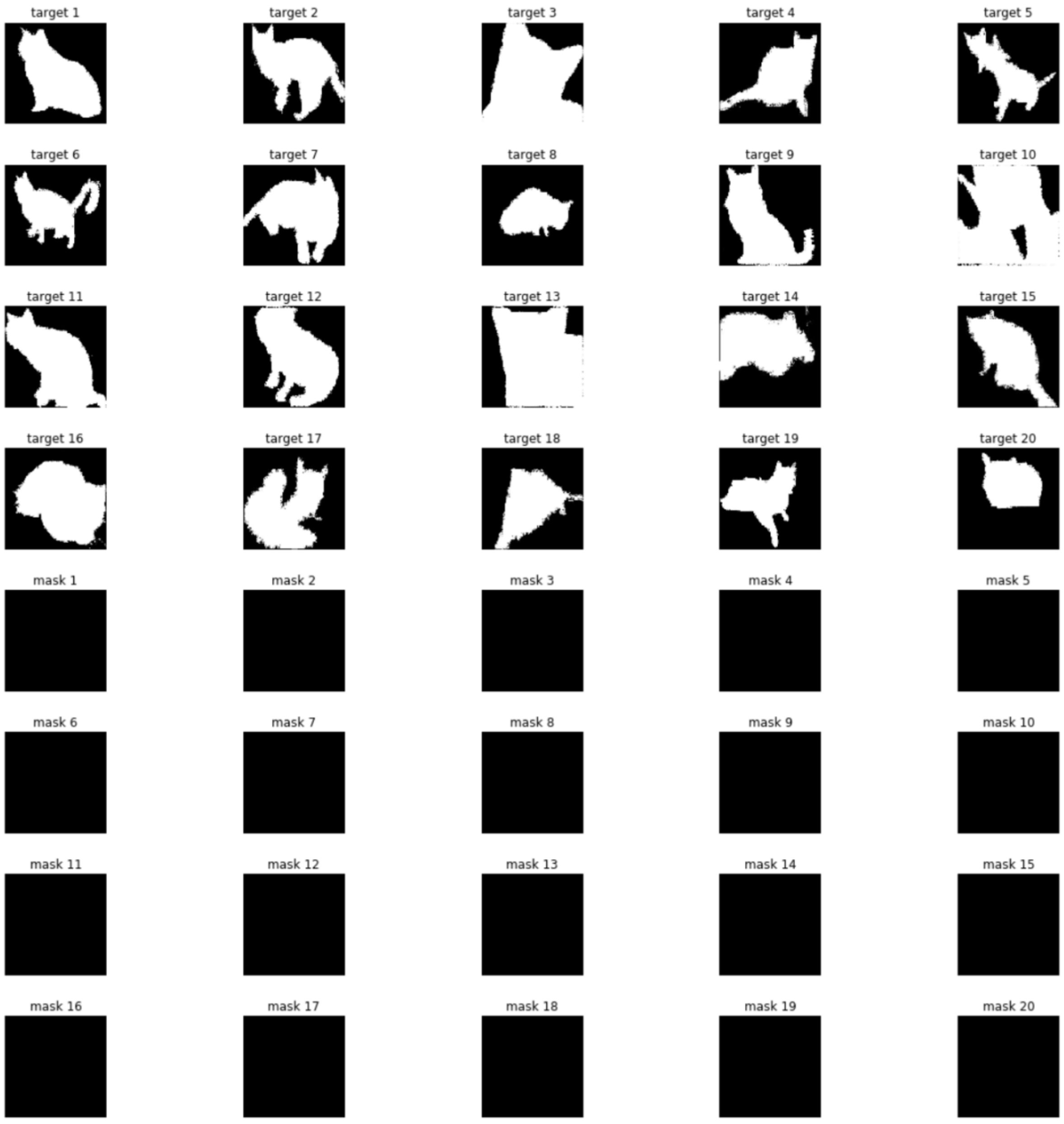
It is worth noticing that there is a slight improvement compared to **1.1** since it's no longer plain white patch. Instead, we can see minor traces of the cats, especially cat 16, 18, 20.

1.3 For this question, I picked car photos and masks taken from the Carvana Image Masking Challenge on Kaggle.

I first use my Unet to train the 2000 images of cars and masks, then I saved the weight. Then I created a new model to inherit the weights and then train it on the cat images again. Since both of these models' last fully connected layers output has the same dimension. I do not need to remove the last layer and I can just load the weight of the cars.

Here are the results:

```
acc: 0.5101 - dice_coef: 0.9998
```



Again, we can see similar behavior with **1.1** and **1.2**, which I know there are flaws in my u-net implementation. However, I was unable to locate the mistakes. One possible source is the weights loading and saving as I am just stacking wrong weights on top of more wrong weights.

1.4 For this part of the question. Since my prediction results from previous sections are obviously wrong, if I apply the masks onto the images it will just get a big patch of green as opposed to a green frame.

Nevertheless, I implanted a function to draw contours given masks by using the cv2 library. However, I did not have enough time or the right mask to draw with.

Fan Vincent Ka Chun 1002563380

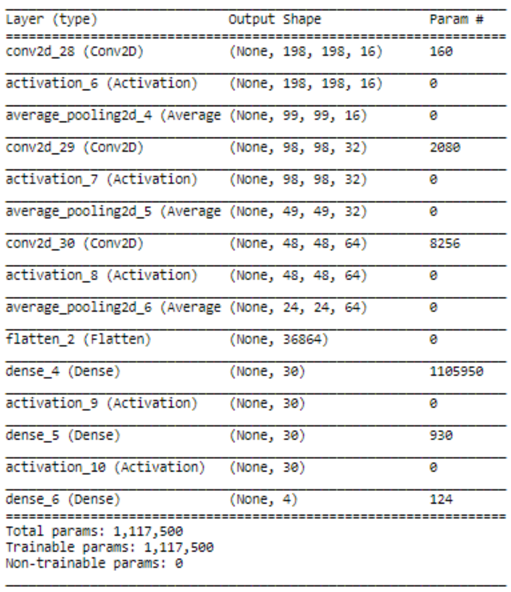
Question 2

Please refer to the python notebook as reference, since I was working on that file and then I moved the code to the starter code python file.

2.1 For this problem, I defined the image as the input of the network in the dimension of (x, 200, 200, 1) where x is the number of total training images. On the other hand, I defined the output of the network as (x, y, x1, y1) where x and y is the coordinate of the center of the circle and x1 and y1 is the coordinate of the corner of the circle (x+r, y+r). Note that I will also normalize the image and coordinates to values between 0 and 1. I then picked L2 distance as my loss function.

The rationale behind is that I want to use a simple convolutional neural network to perform regression on the coordinates. Since there will always be a circle within the image, then the network only needs to localize the circle and L2 distance is a good measure on how close the predicted coordinates and actual coordinates. The reason why I picked to use the corner of the circle as one of the coordinates is that it will allow the network to have more parameters to tune to.

2.2 In my implementation, my neural network is as follows:



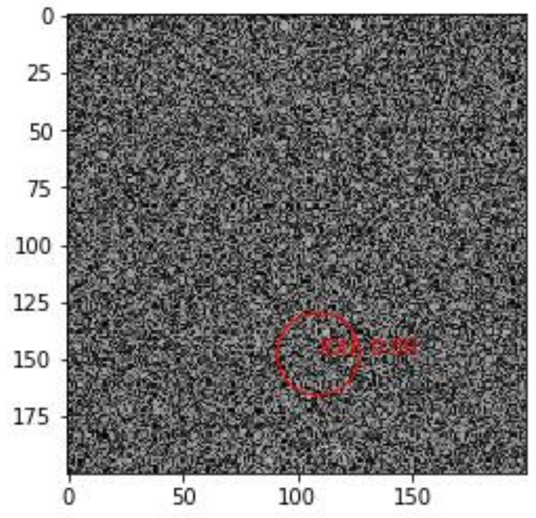
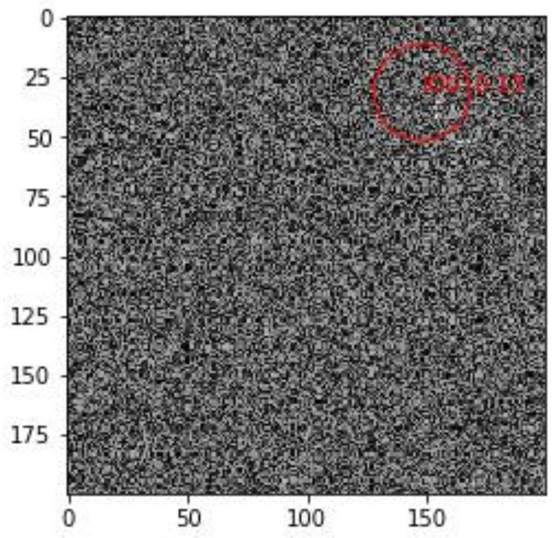
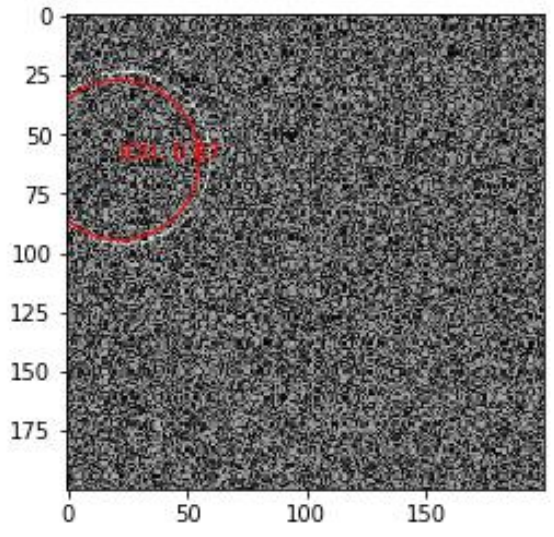
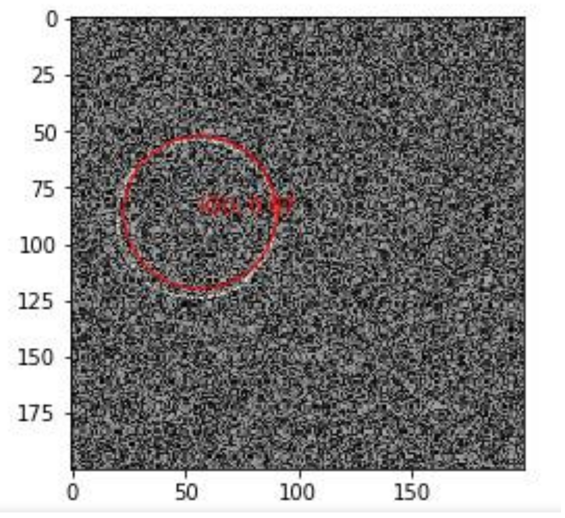
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_28 (Conv2D)           (None, 198, 198, 16)      160
_________________________________________________________________
activation_6 (Activation)    (None, 198, 198, 16)      0
_________________________________________________________________
average_pooling2d_4 (Average (None, 99, 99, 16)        0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 98, 98, 32)        2080
_________________________________________________________________
activation_7 (Activation)    (None, 98, 98, 32)        0
_________________________________________________________________
average_pooling2d_5 (Average (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 48, 48, 64)        8256
_________________________________________________________________
activation_8 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
average_pooling2d_6 (Average (None, 24, 24, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_4 (Dense)              (None, 30)                1105950
_________________________________________________________________
activation_9 (Activation)    (None, 30)                0
_________________________________________________________________
dense_5 (Dense)              (None, 30)                930
_________________________________________________________________
activation_10 (Activation)   (None, 30)                0
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 124
=================================================================
Total params: 1,117,500
Trainable params: 1,117,500
Non-trainable params: 0
_________________________________________________________________
```

I have chosen to use 3 convolution layers at size 16,32,64 to identify all the features, then each followed by a max-pool layer to down sample the input. Finally, to extract the 4 coordinates I need, I have 3 dense layers to allow the network to be more flexible.

I trained my network with 8000 samples with an 80/20 split for validation, I did not choose a larger data set due to lack of computational power and time. The results of my networks has around 50% of the predictions having more than 0.5 IOU and 18% of the predictions having more than 0.7 IOU.

2.3 After trying to use IOU as my loss function, the results were not as good. I think this is largely due to the fact that IOU is not differentiable with respect to the parameters of the network. Therefore, the loss produced is not really useful to the network since it cannot backpropagate well with IOU.

2.4 Here are some of the results I got:

Notice that upper 2 images have very low IOU values (~ 0). In these images, the circles are not as distinguishable from the noise. Also, these images have relatively smaller radius, thus it was affected by the noise by a larger amount.

On the other hand, the bottom 2 images have impressive IOU values (~0.9). If we examine these images, we can easily tell where the circles are located and they both have relatively larger radius. Thus, we can conclude that our network performs better with larger circles.

I have attempt to include a convolutional autoencoder in my implementation. I planned to first parse the images to this autoencoder which will remove the noise from the background and return an image with a circle. This can be achieved by have a u-net like structure with a encoder and decoder chunk inside the network. The noise should theoretically be removed by the max pooling and up sampling layers.

However, after many attempts, the network could not reproduce the noise free input. I believe that it was due to the thickness of the circle is not big enough for it to be distinguished from the heavy noise.

However, I believe to achieve a better result. The noise must be removed first, which would allow the object localization network to be able to localize the circle much easier.

Question 3

A convolutional neural network for binary classification should have a sigmoid activation function as the last layer. Thus, the dimension of its output must be (# of inputs, 1) where the 1 is the predicted probability of whether the image has or has no hotdog. Therefore, we should expect the network to output a scalar when we feed in 1 image, so (1, 1), which it was given as c.

However, since we do not know how the model was set up, we do not know what does 1 or 0 refers to since sigmoid outputs value between 0 and 1.

Therefore, we can say that if c>0.5 then the picture was a hotdog. Otherwise it isn't given that 1 refers to a hotdog. To conclude, we can say that there is a probability of c or (1-c) that the picture was a hotdog.