Heatmap / Density Map Generation and Crowd Counting

Ooi Jun Ming, 15/12/2020

1. Introduction

Crowd Counting has always been an active research within the field of Deep Learning. It is a technique that is used to estimate the number of people within an image or video. In images where there are too many people crammed within, it is hard for us to just look and actually give an accurate estimation. One way that we could do manually is to go from left to right and attempt to count the human heads. However, there is bound to be human error where we could lose track of count or miscount.

Through the use of deep learning, we can simply feed the image into the model and it will attempt to predict the number of people within the image.

2. Real World Applications

In order to perform crowd counting, we also need to be able to generate a heatmap or density map. This allows us to better predict crowd size and determine denseness.

There are many useful real-world applications for this. In the current COVID-19 situation, we want to be able to determiner if an image is dense. By identifying whether the image is dense and looking at the heatmap / density map generated, we are then able to see where are the places that require attention or additional monitoring.

Other real-world applications are:

1. Urban Planning
2. Managing Traffic Flows
3. Video Surveillance

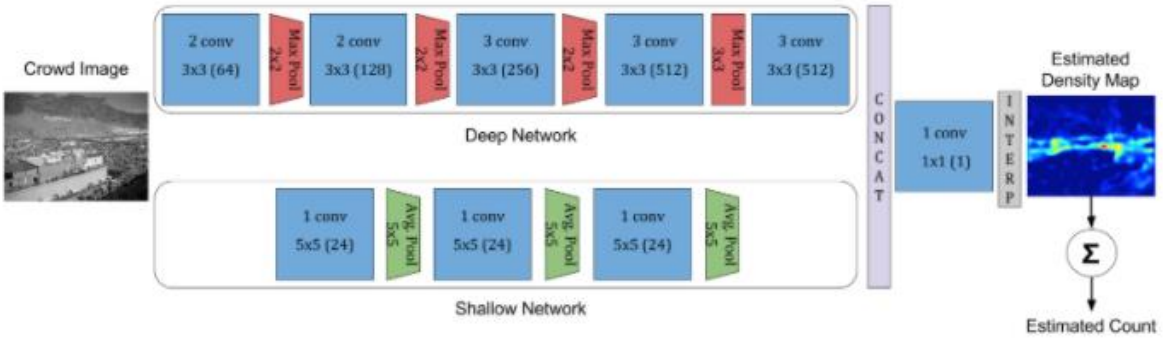
3. Dataset used

The dataset that is used to train the model is the NWPU Crowd Dataset:
https://gjy3035.github.io/NWPU-Crowd-Sample-Code/

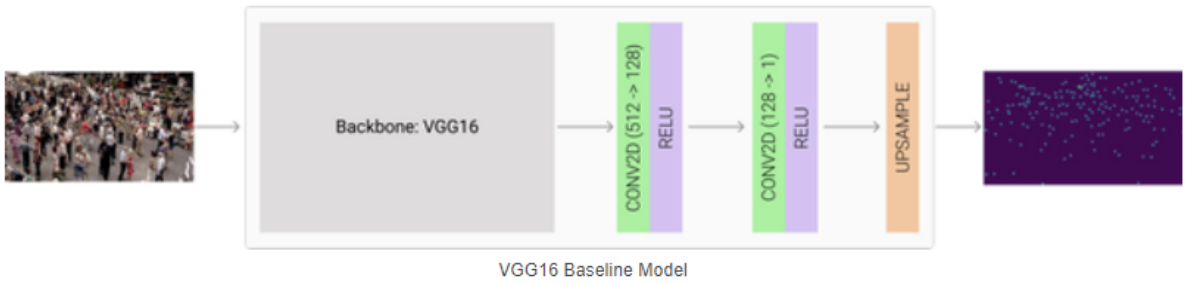
The dataset consists of 3609 images with annotated instances. There are many advantages to using this dataset. Firstly, it combines multiple datasets, which ensures that it has a wide variation. It also contains negative samples where there are no crowds. The last advantage is that it provides the positions of the people within the images, which will be used to help generate the training density map.

However, as a CNN network can only take in a single resolution, all the images will be resized within python to be of 244 pixels by 224 pixels.

4. Model References for Density Map Generation



Model 1 - CrowdNet



Model 2 - Katnoria

The models that I will be referencing are CrowdNet and Katnoria.

CrowdNet is a combination of both shallow and deep FCNN. By having 2 layers, it helps to capture both the high level and low-level features. However, one of the biggest issues with this model is that the output of the neural network is a density map that is 1/32 times the size of the input image. This is probably due to the 4 max-pooling layers within the deep network that drastically reduces the output size of each layer.

Katnoria uses a pretrained VGG16 network followed by 2 Convolutional layers and a final up-sampling layer to get the target density map size.

5. Initial Potential Issues

The first problem that was encountered is that there was no density map given for the training dataset. This means that I had to generate my own density map for the training data to train on.

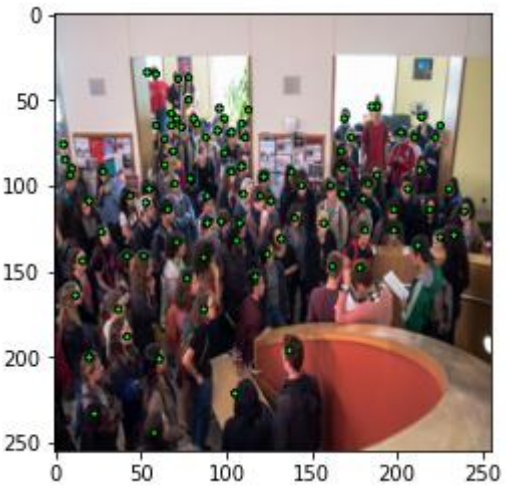
Another issue that I encountered is that due to the large dataset as well as the huge models, many computers have insufficient GPU and CPU space. This means that I have to reduce or remove certain layers within the model.

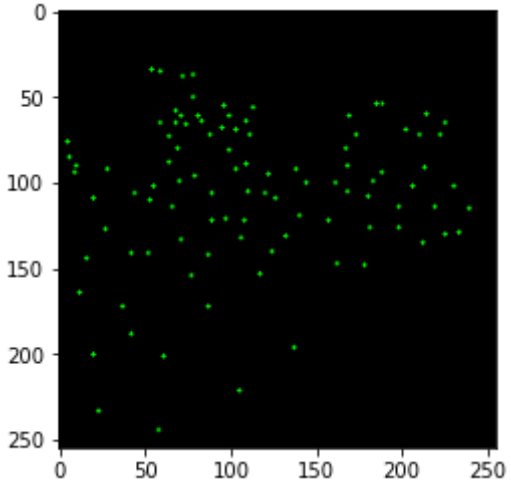
As crowd counting is an estimation model, there is no best model and it can only provide the closest estimation.

6. Custom Density Map Generation for Training and Testing


```
{"img_id": "0009.jpg", "human_num": 100, "points": [[1148.84, 480.96], [1402.28, 515.52], [1537.64, 509.76], [1526.12000000000
```


For the training dataset, I am given a json file for each image which consist of the positions of every human head within the image.



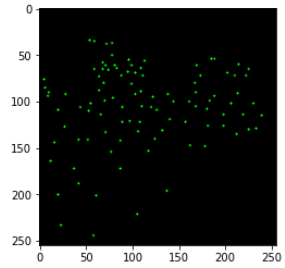



```
img = PIL.Image.new("RGB", (255, 255))
draw = ImageDraw.Draw(img)

for points in image_data:
    draw.ellipse((points[0] - 2, points[1] - 2, points[0] + 2, points[1] + 2), fill=(0, 255, 0), outline=(0, 0, 0))

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x21b96868408>
```



Through the use of the PIL library, I draw each point as a circle within a black image. For better visualization, I use a single colour (green) for each circle. Once, the density map has been drawn, it will then be converted into a single 2D array of floats (224 by 224), which will be used as the output for the model training.

This density map will allow the model to train and learn how to identify the small human heads better. However, there are certain disadvantage with this version which will be discussed later.

7. 1st Model Version and Result

```
model = keras.Sequential()

for i in range(10):
    model.add(vgg16.layers[i])

model.add(keras.layers.Conv2D(128,kernel_size=(3,3),padding="same", activation="relu"))

model.add(keras.layers.Conv2D(1, kernel_size=(3,3), padding="same", activation="relu"))

model.add(keras.layers.UpSampling2D((4, 4)))
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 56, 56, 128)       295040
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 56, 56, 1)         1153
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 224, 224, 1)       0
=================================================================
Total params: 2,031,681
Trainable params: 2,031,681
Non-trainable params: 0
_________________________________________________________________
```

The first model that I created is based on the Katnoria model proposed. For the VGG16 layers, I added the first 10 layers. I then followed it up with 2 Convolutional layers and a final up-sampling layer to increase the output of the density map to 224 by 224, which is the original input size.

However, after training the model, the result was not what I expected. The output of the all the density maps were consistently empty. Every value in the 2D array output was zero.

During the training, there was little to no decrease in MSE throughout every epoch.

Ultimately, the model was considered a failure and could not perform at all.

8. Latest Model Version

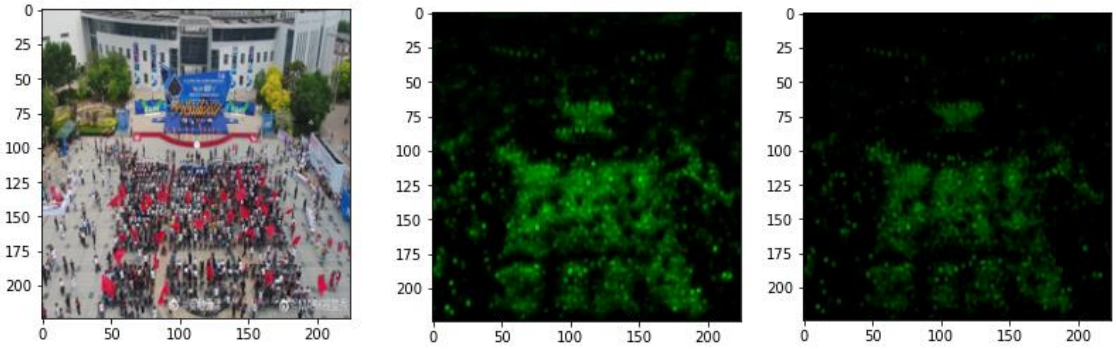
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 64)      1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 224, 224, 128)     73856
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 224, 224, 256)     295168
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 224, 224, 512)     1180160
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 224, 224, 512)     2359808
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 224, 224, 128)     589952
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 224, 224, 1)       1153
=================================================================
Total params: 4,501,889
Trainable params: 4,501,889
Non-trainable params: 0
_________________________________________________________________
```

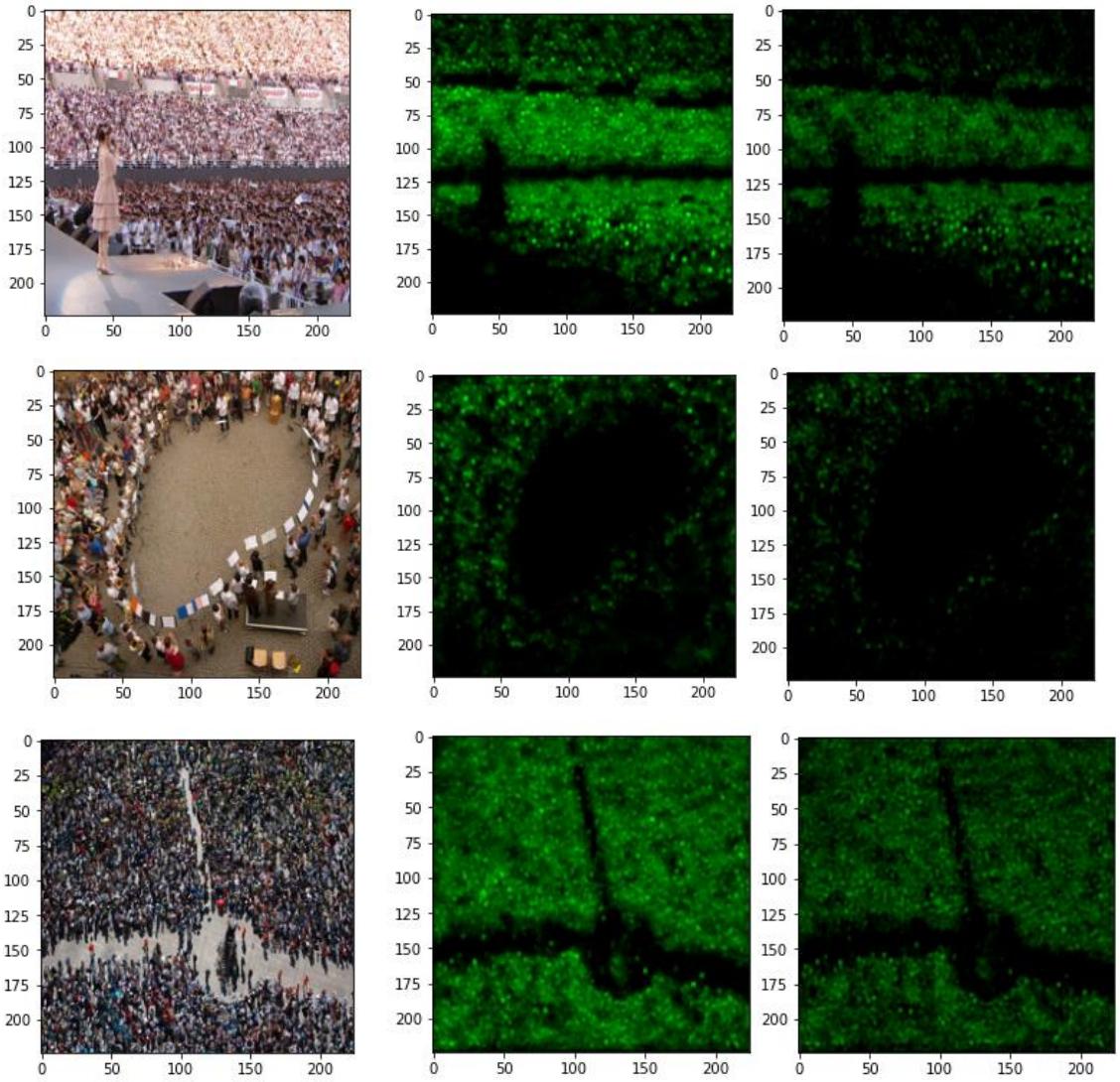
The latest model that I created is still based on the VGG16 model. However, I have removed the max pooling layers. Due to the change in input size for each layer, I am unable to use the VGG16 weights as they are catered for the reduced input size due to the max pooling layers.

By removing the max pooling layers, it ensures that there will be enough features to train on. Furthermore, there is no down-sampling and up-sampling within the model which means that there will be no loss of data.

9. Model Prediction Results

These are some of the density map predictions generated by the model. The 1st density map is trained on 1000 images while the 2nd density map is trained on 2000 images.

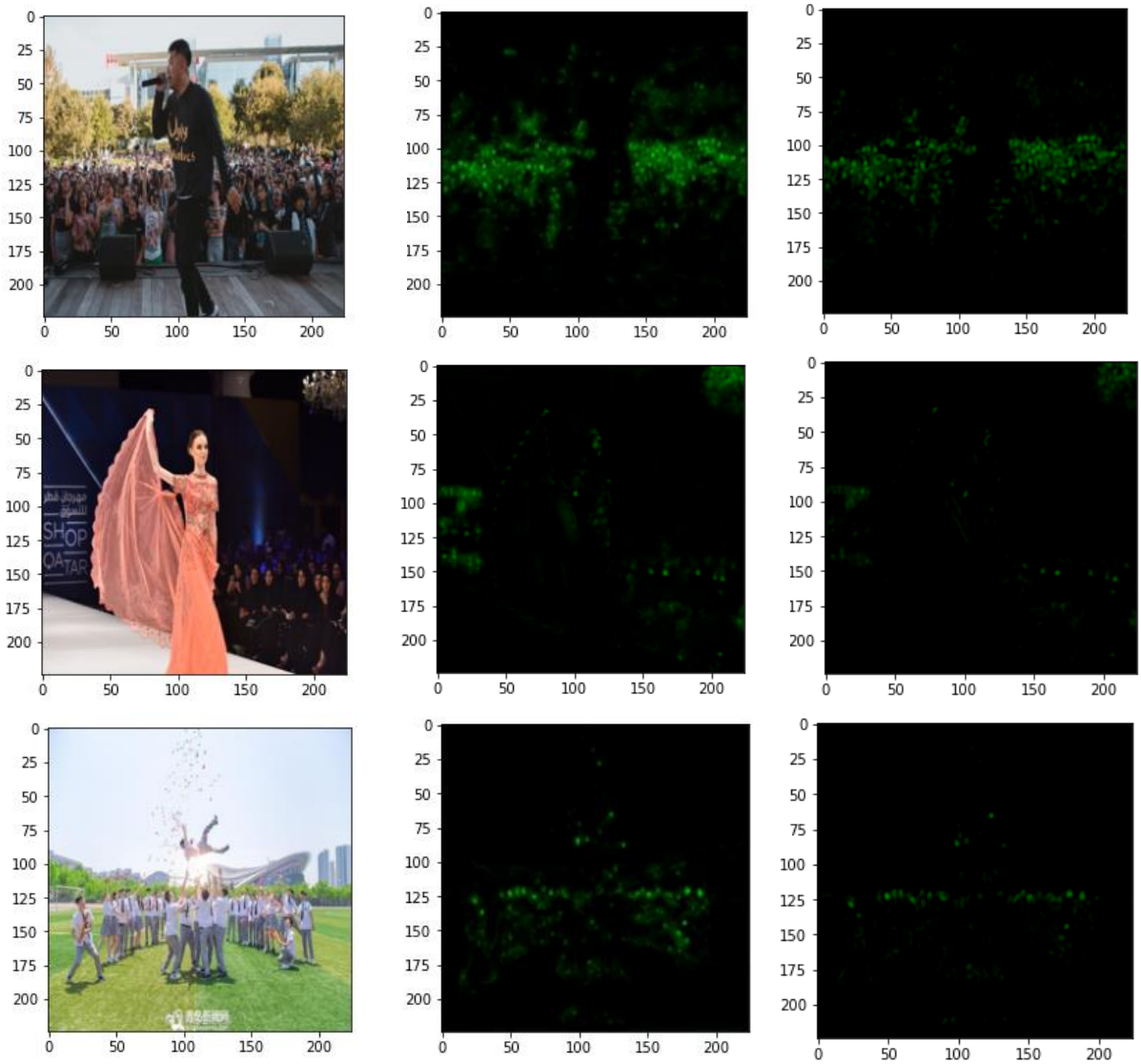




Based on the prediction of the density maps generated, we are able to see that the model is able to consistently determine the small human heads within a crowd. This can be seen by the density maps generated for images that consist of hundreds of people grouped together.

However, what we also realise is that when there are little humans or the images are taken closed up, the model is unable to properly detect the human heads. This is probably due to the way the density map was generated where only a small circle is drawn at the head position but it does not actually highlight the entire head, which the model fails to recognise.

Below are some of the density maps that were predicted wrongly.



10. Potential Issues with the Model

One of the biggest potential issue with the model is that it takes very long to train. For 1000 images, it took 2 days to train. However, for 2000 images, the time taken to train was 6 days. If we want to increase the amount of training data, it could potentially take more than weeks to train.

It also takes quite a while to predict a density map, approximately 5s for a single image. To predict the density maps for 5000 images, it took about 10 hours.

Another potential issue is the loss of image quality as all images must be resized to 224 by 224 pixels.

11. Proposed Solution / Improvement

One way that we could further improve the generation of the training density map would be to use another method that will actually highlight the entire head. By doing so, this will help the model to identify not just small heads, but also close-up images. This could be done by using OpenCV algorithm to perform head detection and highlight them.

We can also improve the quality of the density map by applying more heat density to the density map through the usage of Gaussian Diffusion formula. This would be better for visualization, rather than just using a single value as the density.

We can also train on more images. Despite having 3000 images, we were unable to train on all of them due to the CPU limitations and time constraint.

12. Second Model for Crowd Counting

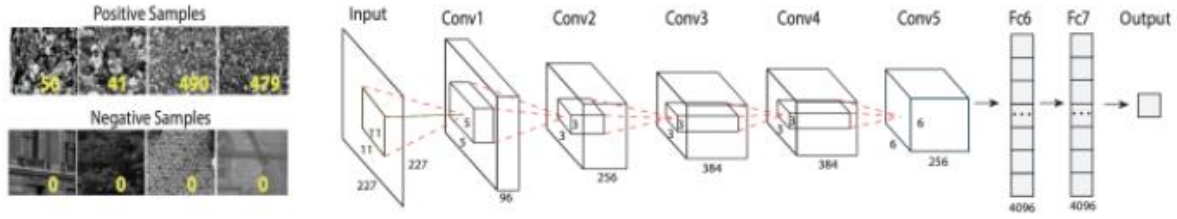


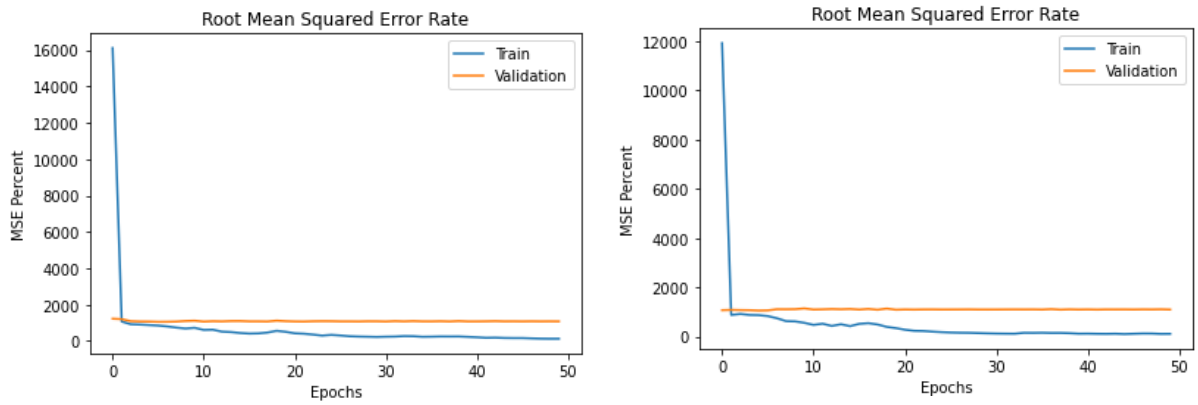
Image Source

The model that we will be referencing for the crowd counting model. It is an initial approach proposed with a CNN regression model. An AlexNet is used for the fully connected network.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_3 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_9 (Dense)              (None, 4028)              202112956
_________________________________________________________________
dense_10 (Dense)             (None, 4028)              16228812
_________________________________________________________________
dense_11 (Dense)             (None, 4028)              16228812
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 4029
=================================================================
Total params: 234,574,609
Trainable params: 234,574,609
Non-trainable params: 0
_________________________________________________________________
```

Instead of the convolutional layer proposed, it will be replaced by the 1st model created to generate the density map. We will then be passing the density map generated into the 2nd model to predict the number of people within the image.

13. RMSE Graph

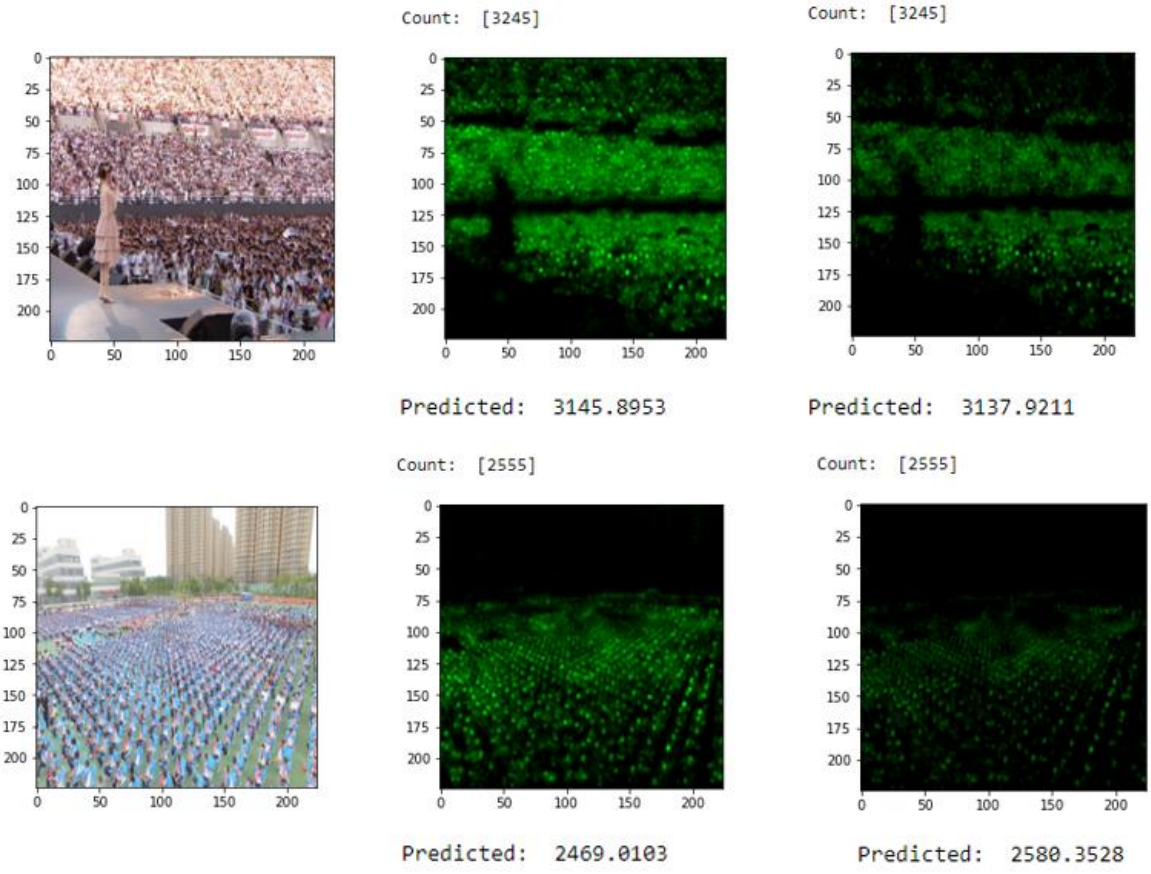


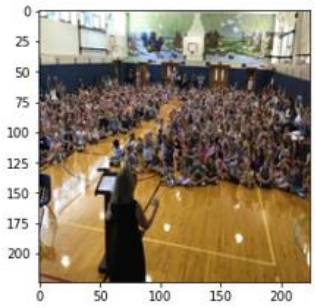
The RMSE graph on the left side is trained on the density maps generated by the 1000 images training. It has a training RMSE of 128.09 and a validation RMSE of 1080.

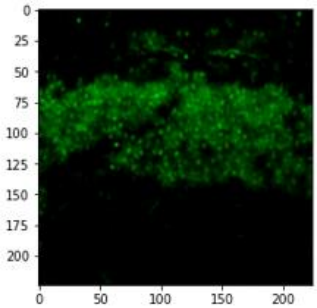
The RMSE graph on the right side is trained on the density maps generated by the 2000 images training. It has a training RMSE of 104.46 and a validation RMSE of 1091.

Unfortunately, even though the RMSE of the training data for both versions are low, both of them have high validation RMSE, which means that both versions of the models are under-fitting.

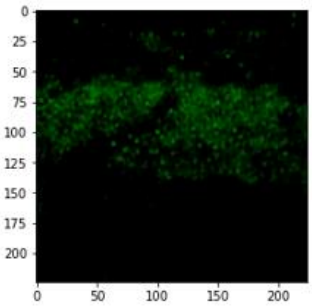
14. Model Predictions



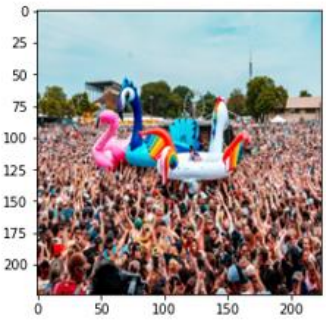


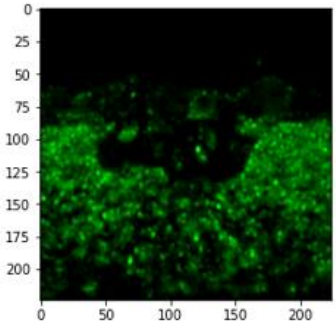


Predicted: 983.78815

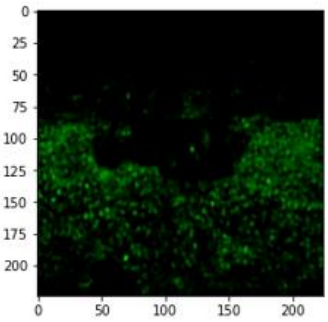


Predicted: 583.2421





Predicted: 2960.8918



Predicted: 2465.549

Based on the predictions, we can see that for the training images, both versions are able to predict closely. However, when we look at the predictions for the last 2 images, we realise that the 1000 images train version tends to over-predict the number of people within the image as compared to the 2000 images version. This is probably due to the way that the density map was generated which affects the counting prediction.

15. Overall Issues

Ultimately, the main issues with the model are due to the fact that the density maps generated are not the most accurate. Due to the certain predictions that are wrong, we actually need to go through the images and remove those whose density maps are wrong.

Another issue could be the possible loss of features due to the down-sizing of images to dimensions of 224 by 224 pixels.

This propagates to the crowd counting model training as it is not training on the most accurate density maps versions.

It is also extremely bad at predicting images with little people or sparse crowds.