Crowd Counting using Multi-Column CNN

Author
Vishnu Bangalore Thirumalesha - 1001829079
FNU Mamta - 1001759457
Shubham Patil - 1001860674

1 Abstract

The goal of the project is to implement a Multi-Column Convolutional Neural Network proposed in Single-Image Crowd Counting via Multi-Column Convolutional Neural Network by Zhang et. al. The paper also proposes a dataset named as Shanghaitech dataset which is used to train the model and generate a Crowd Density Map. Further we aim to test this model on a dataset based on bacteria.

2 Introduction

The aim is to create a method for estimating the crowd count from a single image with random crowd density and random perspective of an image. Method used is the Multi-column Convolutional Neural Network (MCNN) architecture to map the image to its crowd density map to achieve the goal. The proposed MCNN allows us to use any size or resolution of the input image. The features learned by each column CNN are sensitive to variations in people/head size due to perspective effect or image resolution by using filters with receptive fields of different sizes.

3 Method

3.1 Dataset Generation

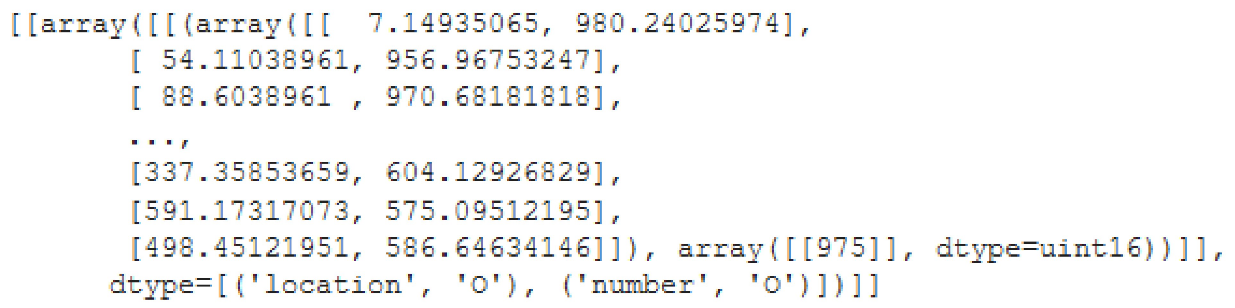
The project is done on the Shanghaitech dataset which has been developed by the authors of our chosen paper. It consists of 1198 annotated images for 330,165 peoples.
It divides in two parts: Part A - 482 images crawled from the internet
Part B - 716 images taken from the streets of Shanghai
Both parts further divided into Train data containing ground-truth in .mat files and crowd images in .jpg format. Test data containing ground-truth in .mat files and crowd images in .jpg format. Once the dataset is collected and we loaded it to the program, we have made few modifications to the dataset.

3.2 Data Preprocessing

The most prominent step while building any machine learning model is data pre-processing as it will directly affect the result of our model. The more we pre-process the data, the more accurate the model performs. To estimate number of people in a given image using CNN the output of the density map (gives approx. of how many people in a square meter or pixel) is used instead of total number of people count in a given image, reason being that the density map preserves more information about an image compared to total number of people in the image due to occlusion and perspective issue. Ground truth data is in .mat format. It has many different information as object format. We segregated the required data and passed it to a Gaussian density filter for returning density of the image and saved it as .npy format in drive for ease of loading it next time.
The ground truth value for an image in .mat looks as below:

.

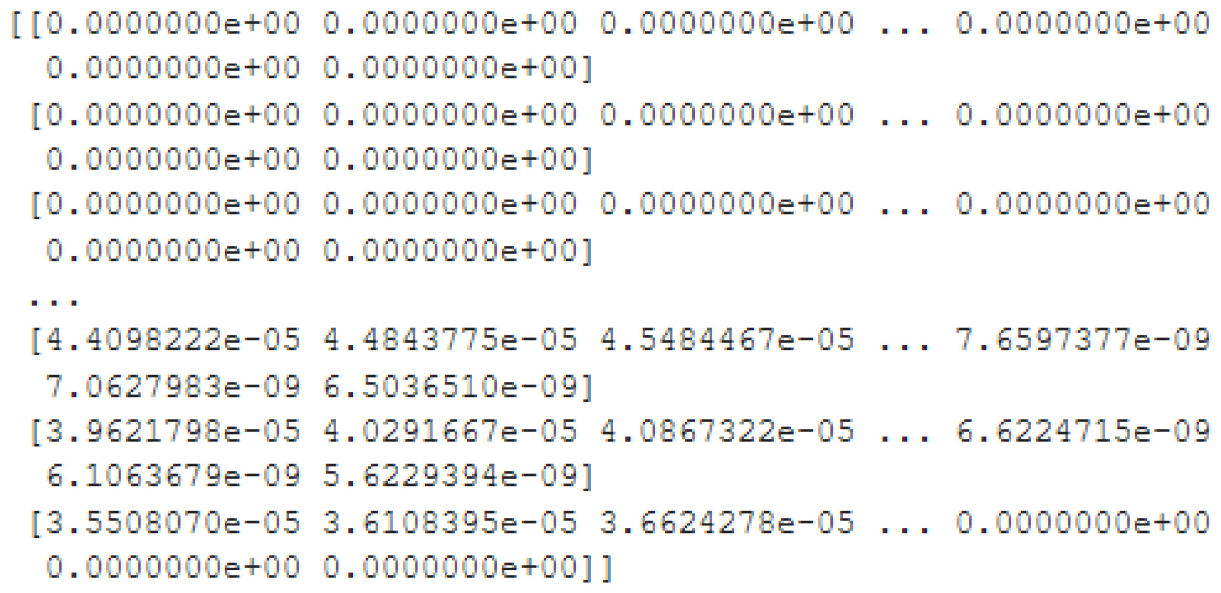
The ground truth value for an image in .mat looks as below:



```
[[array([[(array([[  7.14935065, 980.24025974],
       [ 54.11038961, 956.96753247],
       [ 88.6038961 , 970.68181818],
       ...,
       [337.35853659, 604.12926829],
       [591.17317073, 575.09512195],
       [498.45121951, 586.64634146]]), array([[975]], dtype=uint16))]],
      dtype=[('location', 'O'), ('number', 'O')])]]
```

Figure 1: Ground truth .mat file

The density value of an image in .npy looks as below:



```
[[0.0000000e+00 0.0000000e+00 0.0000000e+00 ... 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 ... 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 ... 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 ...
 [4.4098222e-05 4.4843775e-05 4.5484467e-05 ... 7.6597377e-09
  7.0627983e-09 6.5036510e-09]
 [3.9621798e-05 4.0291667e-05 4.0867322e-05 ... 6.6224715e-09
  6.1063679e-09 5.6229394e-09]
 [3.5508070e-05 3.6108395e-05 3.6624278e-05 ... 0.0000000e+00
  0.0000000e+00 0.0000000e+00]]
```

Figure 2: Ground truth .mat file to numpy

Based on the density generated and image available, the actual images and ground truth values made as numpy are loaded and converted with a down sample value as 4 (down sample value of 1 will return the size same as the actual image) to tensors. These tensors are retuned to form a dataset which will be used for model training.

3.3 MODEL

The aim of this paper is to estimate accurate crowd count from an arbitrary still image, with an arbitrary camera perspective and crowd density. The existing challenges were, the density and the distribution of the crowd vary significantly from image to image and there are tremendous occlusions for most people in each image, hence traditional detection-based methods do not work well on such images and situations. The other challenge faced by the writer of the paper was due to significant

variation in the scale of the people in the images, they needed to utilize features at different scales to accurately estimate crowd counts for different images which was a difficult task to achieve.
To overcome all the above mentioned challenges, the professors of Shanghai Tech university have designed a multi-column convolutional neural network (MCNN) for counting the crowd in an arbitrary still image. The idea of this model is inspired by the mutli-column deep neural network for image classification where the final predictions are obtained by averaging the individual predictions of all deep neural networks.

3.3.1 MCNN

The model used in this paper consists of three columns of convolutional neural networks where each cnn has filters with different sizes. Input to the MCNN is the image and output of the MCNN is a crowd density map. Finally the crowd count is calculated by taking the integral of the density map.
The three columns correspond to filters with receptive fields of different sizes i.e. large, medium and small so that the features learned by each column CNN is adaptive to large variation in people's head size across different image resolutions. The professors have replaced the fully connected layer with a convolutional layer with filter size of 1 x 1. Therefore the input image of the MCNN can be of arbitrary size to avoid distortion. The output of the network is an estimate of the density of the crowd whose integral will derive the crowd count.
Density map preserves more information related to distribution of people in an image than the total number of crowds. Density map gives the spatial distribution of the given image. Such distribution information is useful in many applications, for example, if the density in a small region is much higher than that in other regions, it may indicate something abnormal happens there. Also in learning the density map via CNN, the learned filters of different sizes are more adapted to heads of different sizes, hence more suitable for arbitrary inputs whose perspective effect varies significantly, thus improves the accuracy of crowd count.

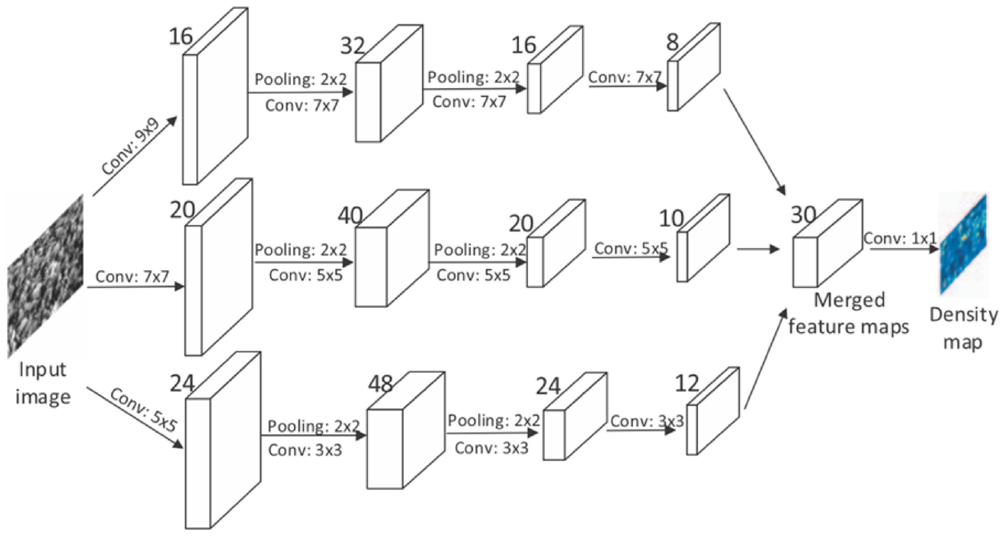
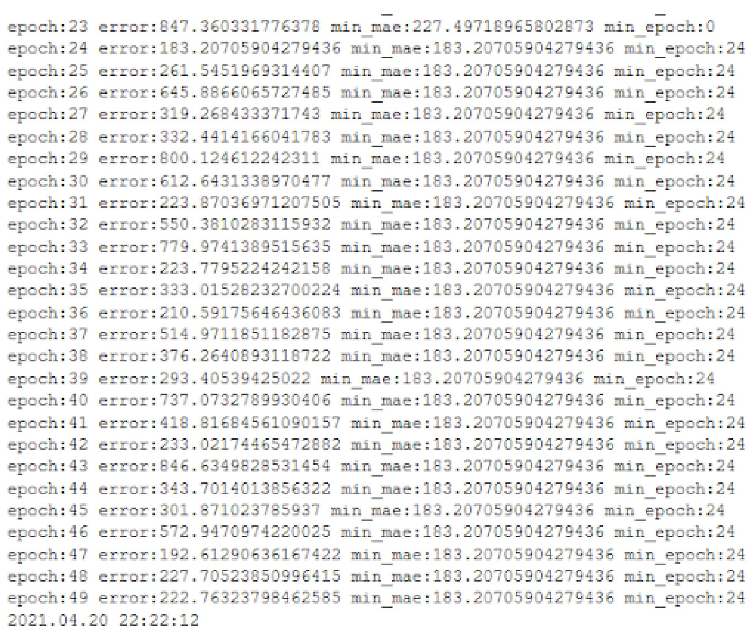


Figure 3: The structure of the multi-column convolutional neural network for crowd density map estimation

It has three parallel CNNs with filters that have different sized local receptive fields. They used the same network architectures for all columns to simplify things (i.e., conv–pooling–conv–pooling). The sizes and numbers of filters for each of these three CNNs, however, differ. Then, for each of the 2x2 regions, Max pooling is used. Rectified linear unit-Relu is the activation function used, and it was chosen because it performs well for cnn. They used less filters for CNNs with larger filters to minimize computational complexity (the number of parameters to be optimized). All of the CNNs' performance function maps are stacked and converted to a density map. Filters of 1 x 1 sizes are used to map the features maps to the density map. The difference between the projected density map and the ground truth is then measured using Euclidean distance.

3.3.2 Milestone 2 results and challenges



```
epoch:23 error:847.360331776378 min_mae:227.49718965802873 min_epoch:0
epoch:24 error:183.20705904279436 min_mae:183.20705904279436 min_epoch:24
epoch:25 error:261.5451969314407 min_mae:183.20705904279436 min_epoch:24
epoch:26 error:645.8866065727485 min_mae:183.20705904279436 min_epoch:24
epoch:27 error:319.268433371743 min_mae:183.20705904279436 min_epoch:24
epoch:28 error:332.4414166041783 min_mae:183.20705904279436 min_epoch:24
epoch:29 error:800.124612242311 min_mae:183.20705904279436 min_epoch:24
epoch:30 error:612.6431338970477 min_mae:183.20705904279436 min_epoch:24
epoch:31 error:223.87036971207505 min_mae:183.20705904279436 min_epoch:24
epoch:32 error:550.3810283115932 min_mae:183.20705904279436 min_epoch:24
epoch:33 error:779.9741389515635 min_mae:183.20705904279436 min_epoch:24
epoch:34 error:223.7795224242158 min_mae:183.20705904279436 min_epoch:24
epoch:35 error:333.01528232700224 min_mae:183.20705904279436 min_epoch:24
epoch:36 error:210.59175646436083 min_mae:183.20705904279436 min_epoch:24
epoch:37 error:514.9711851182875 min_mae:183.20705904279436 min_epoch:24
epoch:38 error:376.2640893118722 min_mae:183.20705904279436 min_epoch:24
epoch:39 error:293.40539425022 min_mae:183.20705904279436 min_epoch:24
epoch:40 error:737.0732789930406 min_mae:183.20705904279436 min_epoch:24
epoch:41 error:418.81684561090157 min_mae:183.20705904279436 min_epoch:24
epoch:42 error:233.02174465472882 min_mae:183.20705904279436 min_epoch:24
epoch:43 error:846.6349828531454 min_mae:183.20705904279436 min_epoch:24
epoch:44 error:343.7014013856322 min_mae:183.20705904279436 min_epoch:24
epoch:45 error:301.871023785937 min_mae:183.20705904279436 min_epoch:24
epoch:46 error:572.9470974220025 min_mae:183.20705904279436 min_epoch:24
epoch:47 error:192.61290636167422 min_mae:183.20705904279436 min_epoch:24
epoch:48 error:227.70523850996415 min_mae:183.20705904279436 min_epoch:24
epoch:49 error:222.76323798462585 min_mae:183.20705904279436 min_epoch:24
2021.04.20 22:22:12
```

Figure 4: Current model results with loss

Currently we have implemented the model proposed by the paper mentioned above. After running it for 50 epochs, above are the results displaying the mean absolute error value for each epoch. Below is the image and the density map generated by our model. Since the error rate is high, the density map generated by the model does not accurately depict the actual density ground truth.

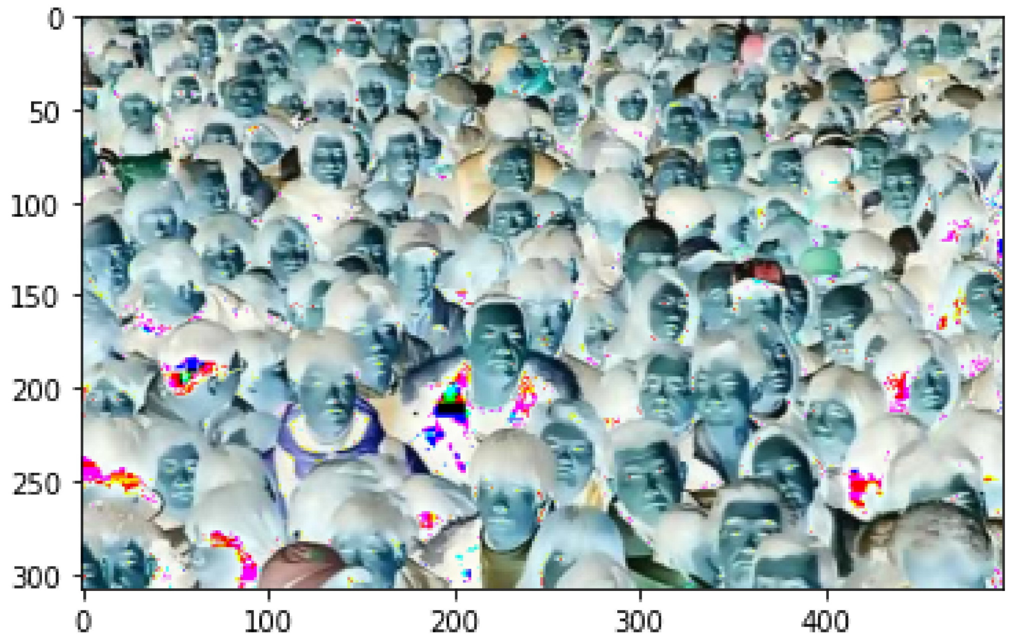


Figure 5: Actual single channel image

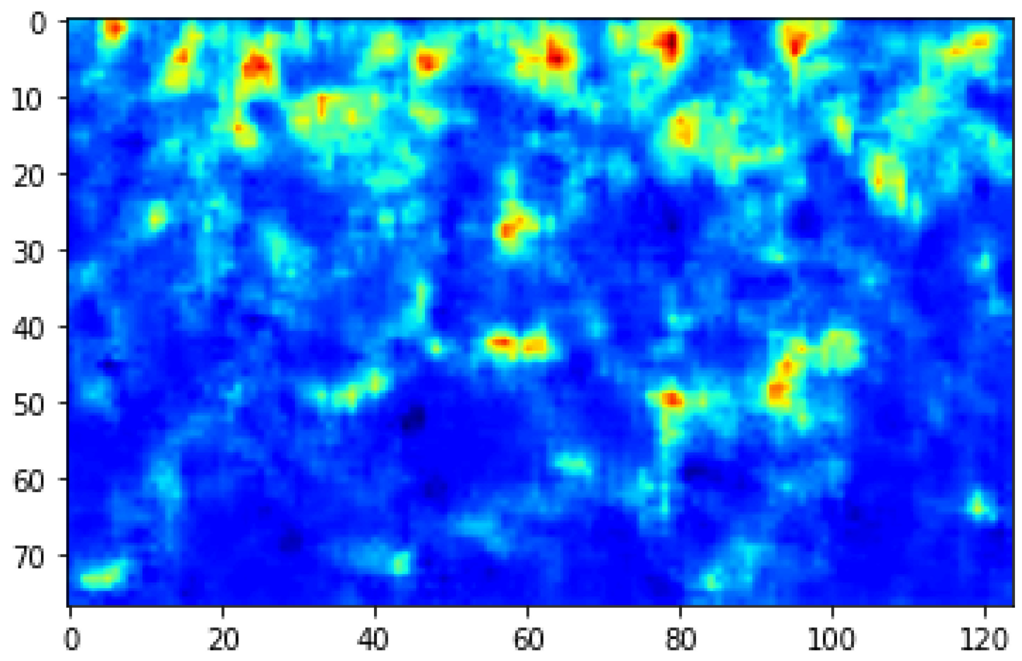
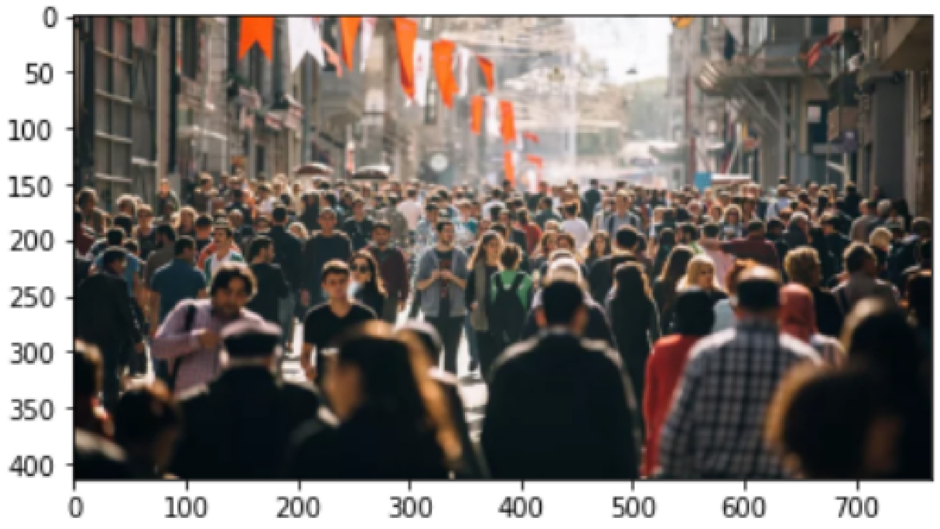


Figure 6: Generated estimate for the actual image

Challenges faced
Reading the mat files and feeding it to the model
Merging the model output into one to generate the estimated density map
Higher error rate

MILESTONE 3 IMPLEMENTATION
To understand the behavior of the model for different sets of training data, we have trained the model on three different sets of training data. Hence based on the result we will be able to do an approximate estimate of the number of people in a given image.

Part 1: Training the model on Part A of the Shanghaitech dataset
As mentioned above the part A consists of images which are sorted for densely populated images. The model was trained for 2002 epochs with the min epoch at 1789th epoch giving a mean absolute error (mae) of 123 which is close to what was achieved in the paper i.e. mean absolute error of 110. The experiments show that the model is able to recognize the dense regions but is not accurately able to find the sparse region of people's headcount.



```
(103, 192)
```

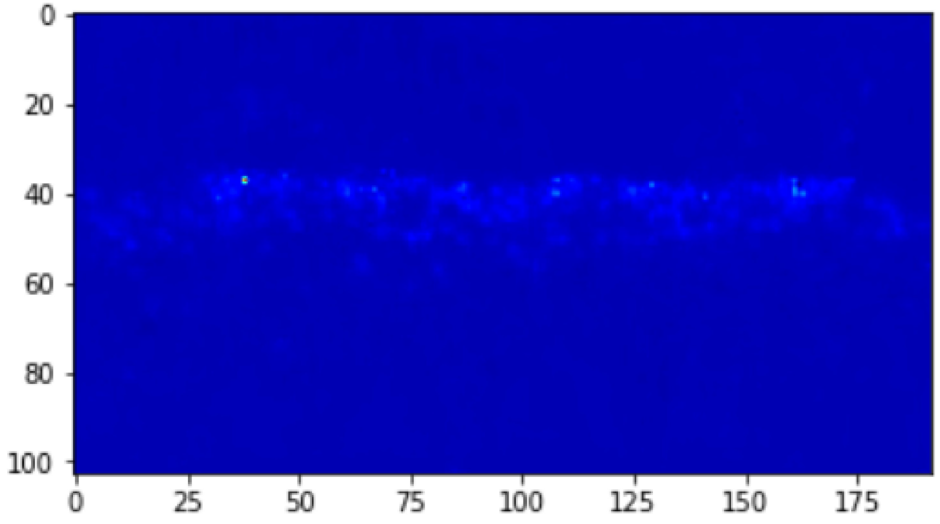
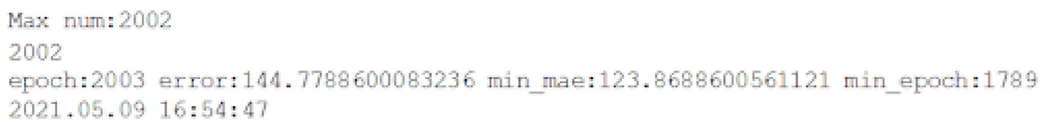


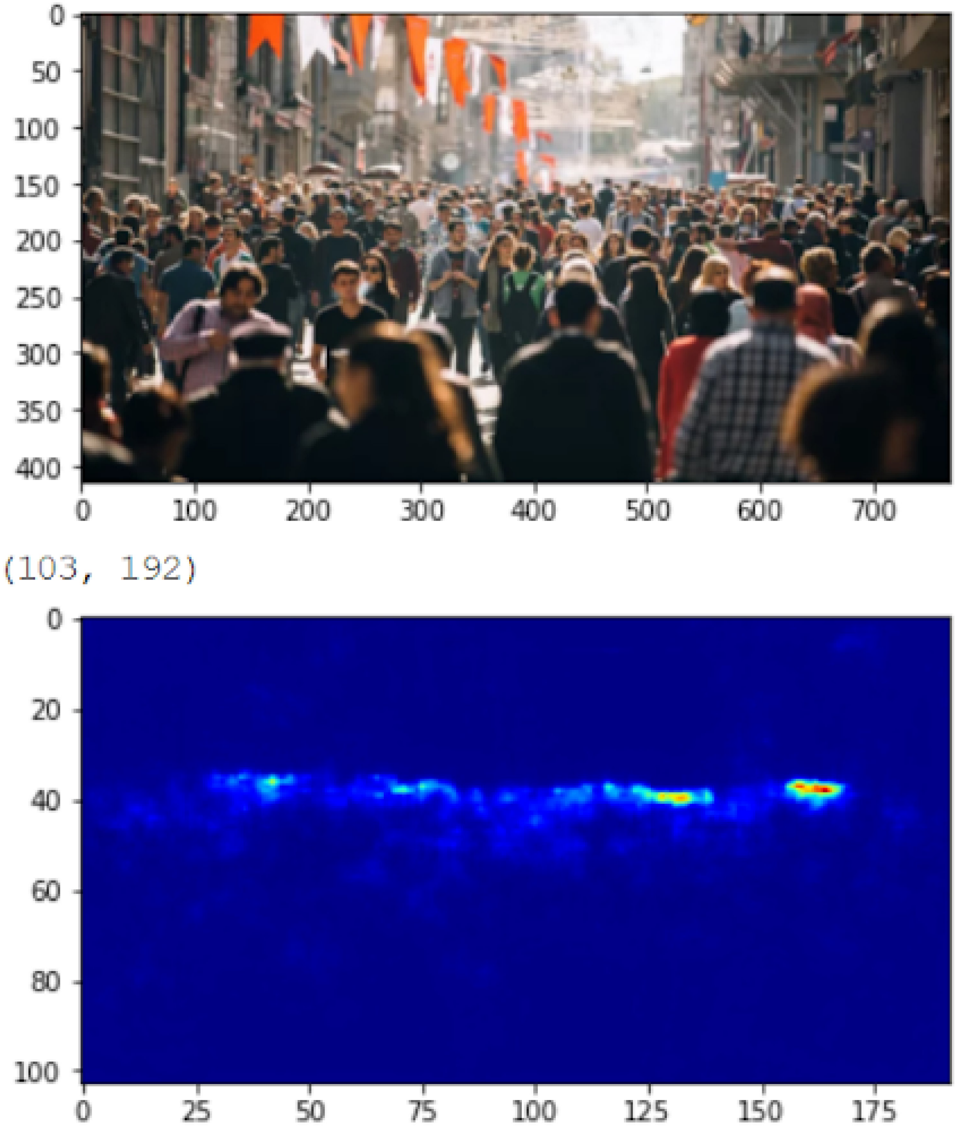
Figure 7: Actual image and estimation from part A training data



```
Max num:2002
2002
epoch:2003 error:144.7788600083236 min_mae:123.8688600561121 min_epoch:1789
2021.05.09 16:54:47
```

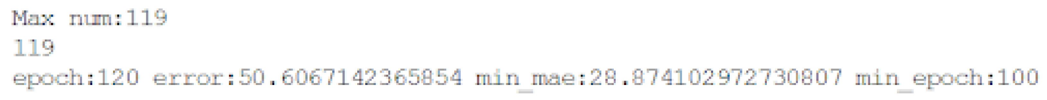
Figure 8: Error for part A training data

Part 2: Training the model on Part B of the Shanghaitech dataset
Similarly Part B, consists of images of sparsely populated crowds from streets of Shanghai. So training on the model, the predicted images is better able to predict the sparse crowds but performs less accurately predicting the dense regions of an image. This model was trained for 120 epochs, with the min epoch at 100th epoch with the mean absolute error at 28.



```
(103, 192)
```

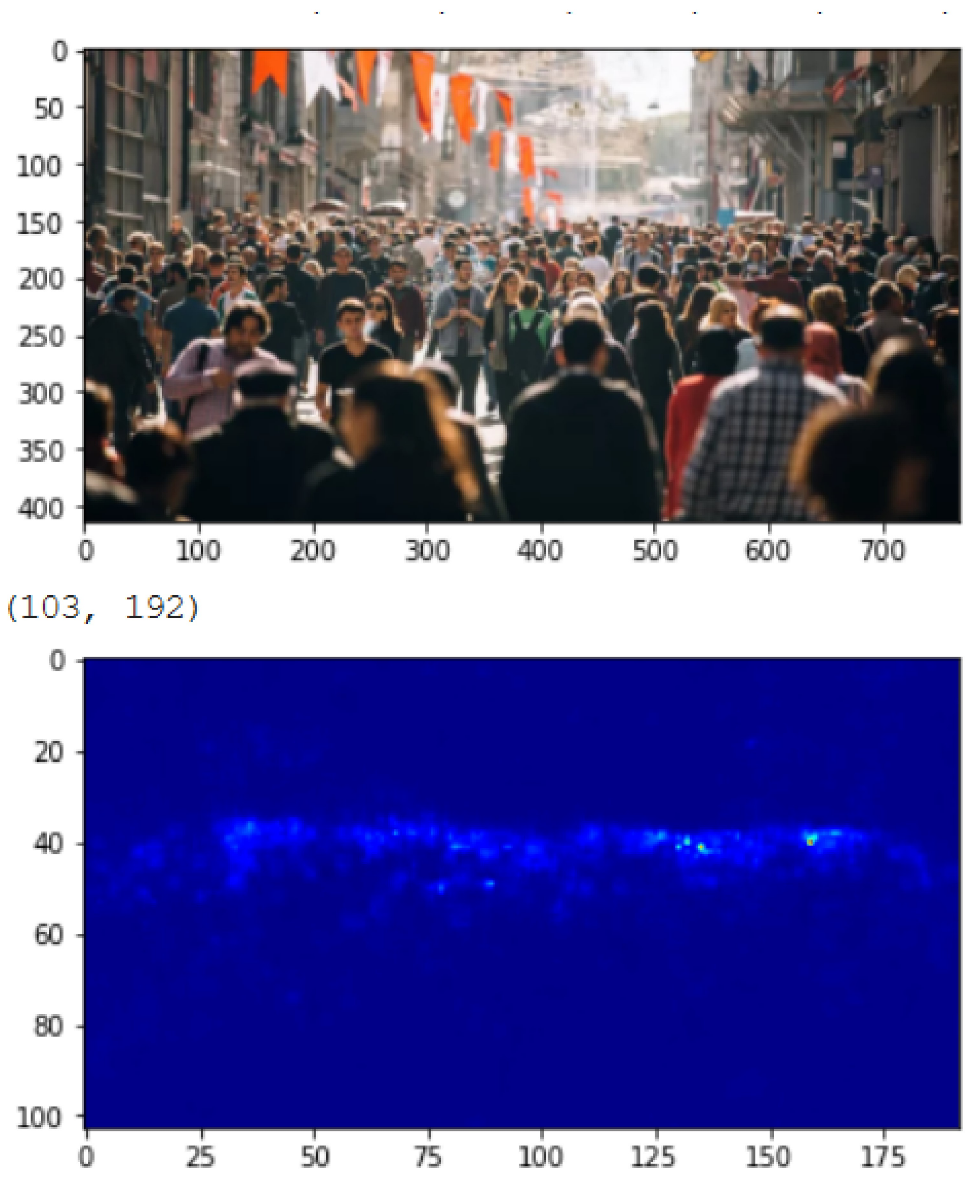
Figure 9: Actual image and estimation from part B training data



```
Max num:119
119
epoch:120 error:50.6067142365854 min_mae:28.874102972730807 min_epoch:100
```

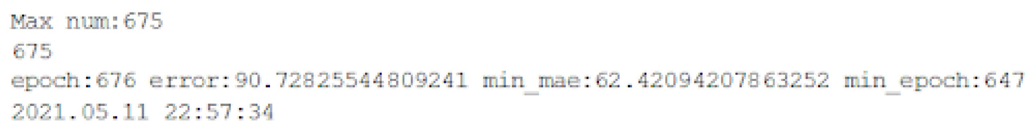
Figure 10: Error for part B training data

Part 3: Training the model on merged Part A and Part B of the Shanghaitech dataset
This model is trained by combining both the part A and part B training data and in this model after training it for 676 epochs we got the mean absolute error of 62. This model gives better error values and the predicted results. The model gives better results for the intermediate density images.



```
(103, 192)
```

Figure 11: Actual image and estimation after merging part A and part B training data



```
Max num:675
675
epoch:676 error:90.72825544809241 min_mae:62.42094207863252 min_epoch:647
2021.05.11 22:57:34
```

Figure 12: Error for merged part A and part B training data

The crowd count can be estimated from the above methods. For example, we can give an input image and based on the density estimates given by all three models we will be able to approximate, if the crowd is in hundreds or thousands. Along with this, we will also be able to recognize the highly populated places in that particular image. This result can be achieved even on a freshly fed image of random resolution with decent quality.

4 Performance Comparison of the model with the paper and other existing techniques

Method	MAE (Part A)	MAE (Part B)
MCNN	123	28.87
MCNN by paper	110.2	26.4
Zhang et al	181	32
LBP+RR	303.2	59.1

Figure 13: Error for part A and part B

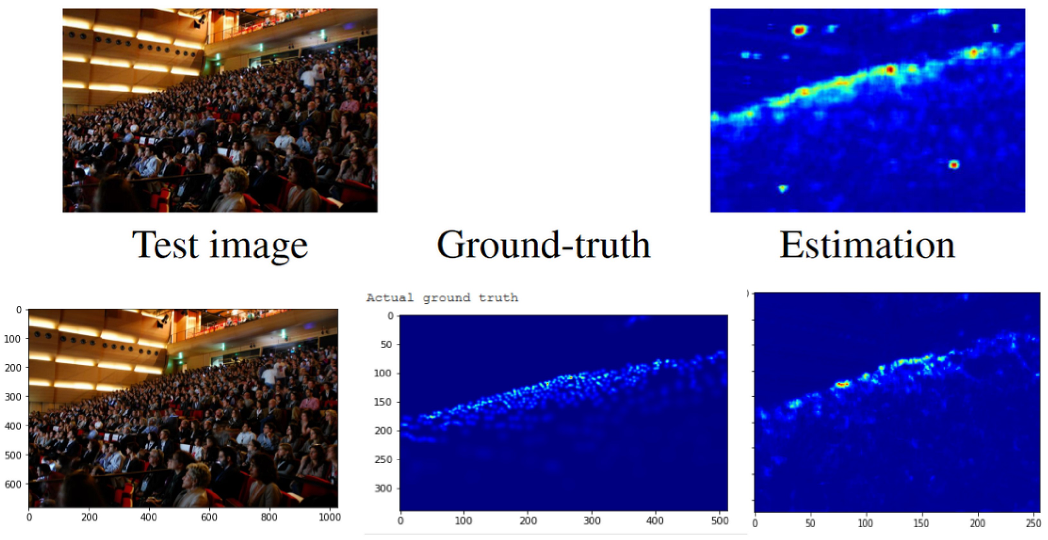


Figure 14: Paper vs our model comparison

In the above image, the first row images are from the paper and second row images are from our model representing actual image, ground truth and estimated density image respectively.

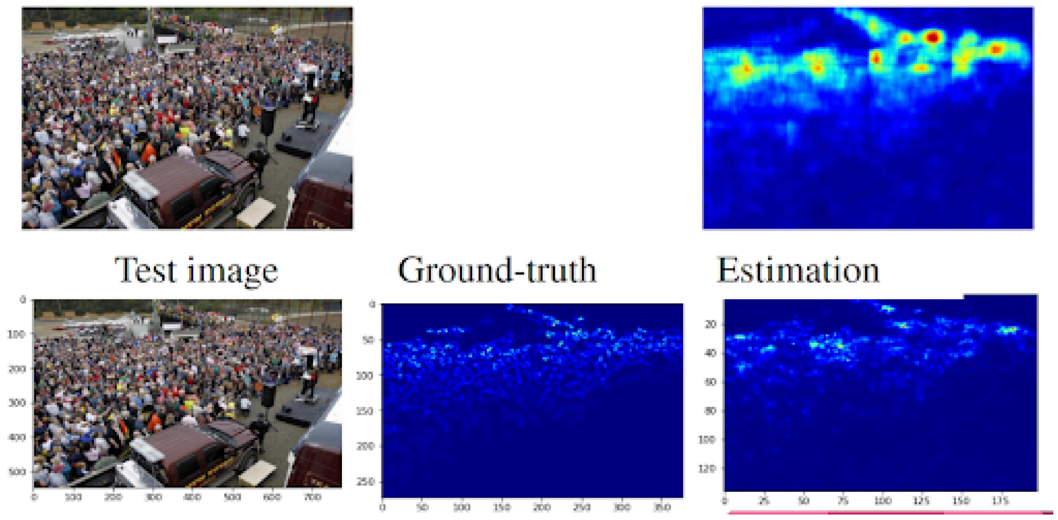


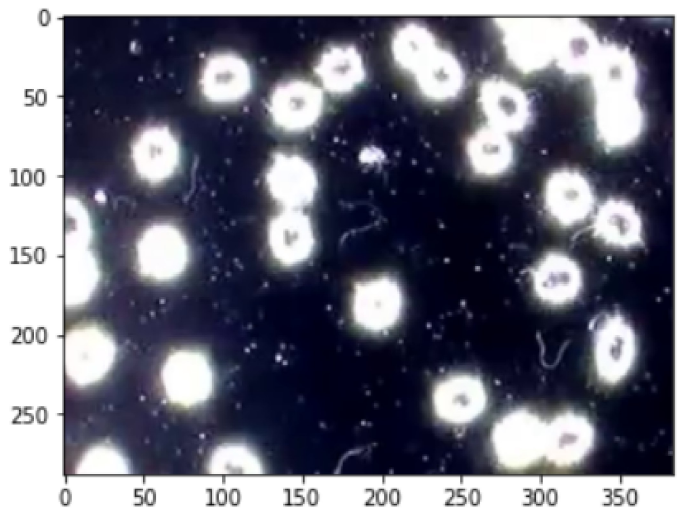
Figure 15: Paper vs our model comparison

In the above image, the first row images are from the paper and second row images are from our model representing actual image, ground truth and estimated density image respectively.

Experiments

Bacteria Dataset

We tested this model on a new raw image and also a new set of similar problem which is bacteria dataset. The model is able to generate a density map showing the density region of the crowd and bacteria.



```
(72, 96)
```

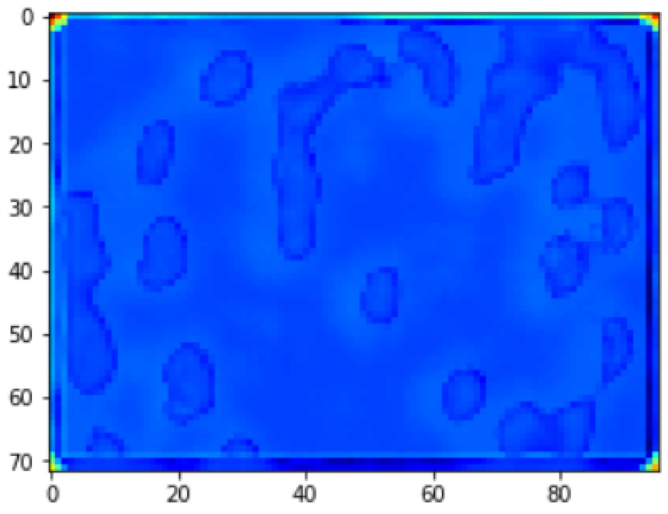
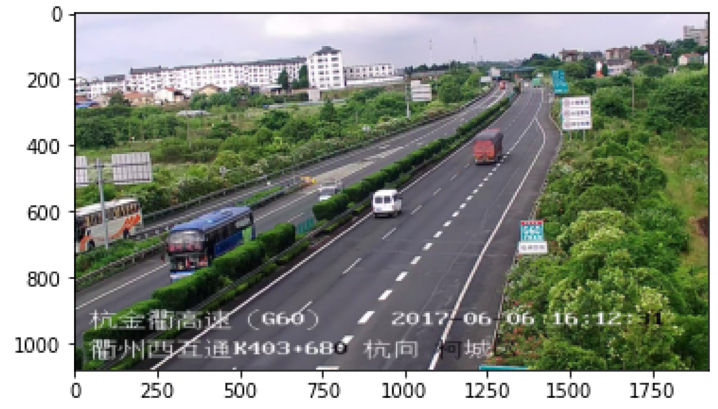


Figure 16: Estimation on Bacteria Dataset

Highway Dataset

From the above bacteria and highway dataset result, we can conclude that though the density is being generated, but to improve this further a transfer learning approach would help the model accurately predict density.

```
Testing on the highway dataset
```


杭金衢高速 (G60) 2017-06-06 16:12:11
衢州西互通K403+680 杭向 柯城
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750

```
(270, 480)
```

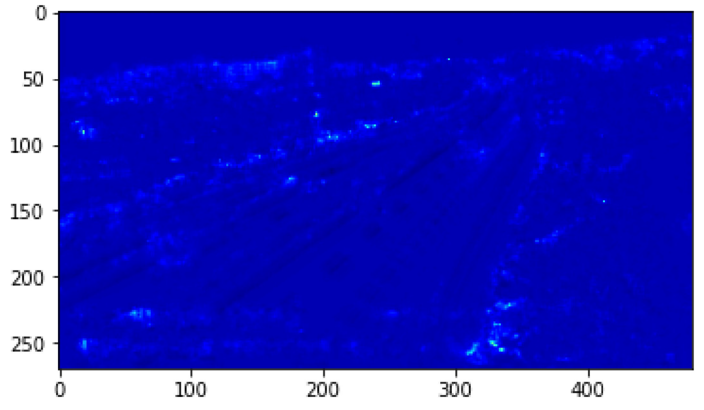


Figure 17: Estimation on Highway Dataset

4.0.1 References:

$https://www.cv - foundation.org/openaccess/content_{c}vpr_{2}016/papers/Zhang_{S}ingle - Image_{C}rowd_{C}ounting_{C}VPR_{2}016_{p}aper.pdf$
$https://www.kaggle.com/tthien/shanghaitec$
$https://github.com/davideverona/deep - crowd - counting_{c}rowdnet$