Course Project
Deep Learning for Computational Data Science

Shashank Tomar (SC17B045) and Kiran L (SC17B150)

January 10, 2021

1 Introduction

The task of visual counting in Computer Vision is to infer the number of objects (people, cars, maize tassels, etc) from an image/video. It has wide applications, such as automatic crowd management, traffic monitoring, and crop yield estimation. Extensive attention has been received in recent years.

Counting is an open-set problem by nature as a count value can range from 0 to ∞ in theory. It is thus typically modeled in a regression manner. Benefiting from the success of convolutional neural networks (CNNs), state-of-the-art deep counting networks often adopt a multi-branch architecture to enhance the feature robustness to dense regions. However, the observed patterns in datasets are limited in practice, which means networks can only learn from a closed set. Crowd counting mechanism can be used to count participants in a protest and crowd coming to an event. The estimate can be used to control in flow of people at an event to prevent overcrowding, stampede (very common in hajj) and to plan evacuation route.

With increasing demands for intelligent video surveillance, public safety and urban planning, improving scene analysis technologies becomes pressing [1], [2]. As an important task of scene analysis, crowd counting has gained more and more attention from multimedia and computer vision communities in recent years for its applications such as crowd control, traffic monitoring and public safety. However, the crowd counting task comes with many challenges such as occlusions, complex backgrounds, non-uniform distributions and variations in scale. Many algorithms have been proposed to address these challenges and increase the accuracy of crowd counting.

2 Literature Review

1. Youmei Zhang et al., 2018, proposed a multi-column CNN (MCNN) is proposed in[5] to address the scale-variation problem by using several CNN branches with different receptive fields to extract multi-scale features. A cascaded CNN learns high-level prior which is incorporated into the crowd density estimation branch of the CNN to boost the performance. They incorporate an attention model into the MCNN [5] to guide the network to focus on head locations during training and testing. Specifically, the attention model learns a probability map in which high probabilities indicate locations where heads are likely to be present. This probability map is used to suppress non-head regions in feature maps from multi-scale feature extraction branches of the MCNN.

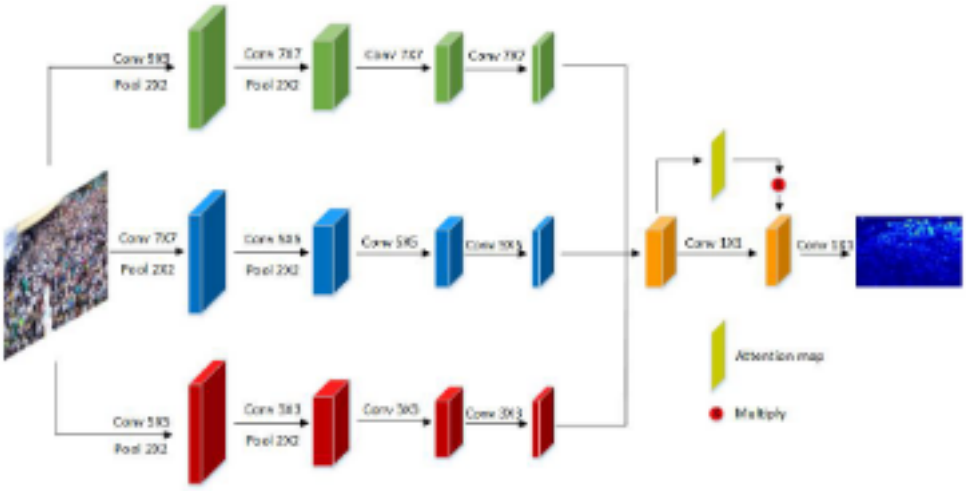


Figure 1: Architecture of the AM-CNN proposed by Youmei Zhang et al., 2018

2. Yuhong Li et al., 2018, propose a network for Congested Scene Recognition called CSRNet to provide a data-driven and deep learning method that can understand highly congested scenes and perform accurate count estimation as well as present high-quality density maps. The proposed CSRNet is composed of two major components: a convolutional neural network (CNN) as the front-end for 2D feature extraction and a dilated CNN for the back-end, which uses dilated kernels to deliver larger reception fields and to replace pooling operations. They demonstrate CSRNet on four datasets (ShanghaiTech dataset, the UCF CC 50 dataset, the WorldEXPO'10 dataset, and the UCSD dataset) and deliver state-of-the-art performance. In the ShanghaiTech Part B dataset, CSRNet achieves 47.3% lower Mean Absolute Error (MAE) than the previous state-of-the-art method.

3. Haipeng Xiong et al., 2019, transform the open set problem into a closed set problem by using the method of spatial divide and conquer to estimate crowd count. A dense region can always be divided until sub-region counts are within the previously observed closed set. Inspired by this idea, they proposed a simple but effective approach, Spatial Divide-and-Conquer Network (S-DCNet). S-DCNet only learns from a closed set but can generalize well to open-set scenarios via S-DC. S-DCNet is also efficient. To avoid repeatedly computing sub-region convolutional features, S-DC is executed on the feature map instead of on the input image. S-DCNet achieves state-of-the-art performance on three crowd counting datasets (ShanghaiTech, UCFCC50 and UCF-QNRF), a vehicle counting dataset (TRANCOS) and a plant counting dataset (MTC). Compared to the previous best methods such as Scale pyramid network for crowd counting (Xinya Chen et al., 2019) and Scale aggregation network for accurate and efficient crowd counting (Xinkun Cao et al., 2018), S-DCNet brings a 20.2% relative improvement on the ShanghaiTech PartB, 20.9% on the UCF-QNRF, 22.5% on the TRANCOS and 15.1 % on the MTC.

Figure 2: Basic Idea behind the architecture propsed by Haipeng Xiong et al., 2019

3 Methodology

Following the approach by Yuhong Li et al., 2018, we have designed the CNN with front-end feature extractor and back-end feature-map transformer whose primary aim is to transform the feature map output by the front-end into a density map which matches the ground-truth density map. Since the complete model is fully convolutional, there is no requirement of a fixed size/dimension of the input image. Since most the input images are color (RGB) images, the

number of channels in the input image is 3.

The complete pipeline has been split into two phases - first is a model density map generation and second is an MLP to aid the inaccuracies in the model prediction. (i.e. fit the generated crowd count). The model prediction of the crowd count is obtained by simply summing up the pixel values. No particular objective function is used to reduce the reconstruction loss (reconstruction of ground-truth density map) since the density map dimensions of ground-truth does not in-general match the model generated density map dimensions. The only loss function is mean square loss of sum of pixel values of model density map and ground-truth density map. The model output is "interpreted" as density map.

Noting that the sum of pixel values of model predicted density map does not in well general match the ground-truth, an MLP is used. The objective of obtaining accurate predictions is now cast as a regression problem, with the x values being the model predicted crowd count and y values being ground-truth crowd count. This is helpful mainly in cases where the model prediction is:

1. Monotonously increasing with ground-truth crowd count
2. Either always under/over-predicts the crowd count.

The first condition is necessary so as to be able to learn a linear regressor and the second condition is due to the limitation of the regressor to simply offset the model predictions by some amount - if under (over) predicted, then increases (decreases) the value for corresponding input to nullify the overall effect leading to final prediction which best matches the ground-truth.

The above described approach is a different approach, since in previous approaches, such as in CSRNet (Yuhong Li et al., 2018), the sum of pixel values of output model density map is taken as a model prediction without use of any regressor to over-come the model inaccuracies.

Thus, we have in essence, retained the model density maps and analysed the effect of sum of pixel values to train an MLP (without using 1×1 convolution).

4 Data Description

ShanghaiTech crowd counting dataset contains 1198 annotated images with a total amount of 330,165 persons.[1] It consists of two parts: PartA and PartB. PartA includes 300 images for training and 182 for testing. This part represents highly congested scenes. PartB contains 716 images in relatively sparse scenes, where 400 images are used for training and 316 for testing. This is a standard dataset commonly used by various deep leaning researchers working on crowd-counting and in a way forms a common basis for comparison of different models

based on their performance in this dataset. Hence, the same was selected for the present course project and analysis.

The major limitation of this dataset is low number of images for training. This can be overcome by data augmentation methods such as random crops, mirroring, noise-addition, and alike. Major benefit of using this dataset is that the performance of the built model as analysed on this dataset is useful for comparison with other models, since most of the models built for crowd-counting report the performance of their models on this dataset.

5 Density Map Generation

Density mapping is simply a way to show where points (or lines) may be concentrated in a given area. A density map is important to visualize the concentration of particular object in a given image (FOV). For example, a density map finds application in census and statistics to visualize population density in a city/country. Similarly, to visualize the forest cover in a given region and in various other applications.

Given the locations of the person (by denoting the location of head), a Gaussian kernel is used to convolve with the delta function like 2-D array to obtain the density maps. Following the method of generating density maps in [2], we use the geometry-adaptive kernels to tackle the highly congested scenes. By blurring each head annotation using a Gaussian kernel (which is normalized to 1), we generate the ground truth considering the spatial distribution of all images from each dataset. The geometry-adaptive kernel is defined as:

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_i) \times G_{\sigma_i}(\boldsymbol{x}),\ with\ \sigma_i = \beta \overline{d_i}$$

For each targeted object x_i in the ground truth δ, we use d_i to indicate the average distance of k nearest neighbors. To generate the density map, we convolve $\delta(x - x_i)$ with a Gaussian kernel with parameter σ_i (standard deviation), where x is the position of pixel in the image. In experiment, we follow the configuration of $\beta = 0.1$ and k = 4. For input with sparse crowd, we adapt the Gaussian kernel to the average head size to blur all the annotations.

6 Code structuring and Explanation

All the implementation was performed using Python programming language on Google Colaboratory platform (with GPU hardware accelerator). The Jupyter Notebook is well written (modular way) to ease comprehension. The same can

be used for future enhancements and modifications by simply modifying the code in different code sections and if necessary adding new code cells/sections.

6.1 Training

The training was performed using PyTorch on Nvidia GPU. Since the input images were of different dimensions, we were constrained to use a batchsize of 1. All models were trained for 100 epochs. The training data was split into training and validation sets in the ratio 4:1. ReLU activation function was used throughout the architecture. Additional training (and validation) data was augmented by mirroring the input images. The difference between the sum of pixel values of ground-truth and model density maps were minimized by MSE Loss function. Adam optimizer was used to accelerate convergence in most cases after comparing the results from Stochastic Gradient Descent (with momentum), RMS Prop and Adam. The early stopping was induced manually by analysing the validation and training loss over 100 epochs (i.e., patience parameter = 100). Validation was performed at every epoch.

Then, with the model density maps for training and validation images saved, the same were used to build an MLP by obtaining the sum of pixel values of the model density maps. The regressor was again trained on GPU using PyTorch module. The regressor was trained for 1000 epochs. Once again ReLU activation function was used throughout the regressor.

6.2 Testing

The testing was performed on the given test-set without any data augmentation. The ground-truth and the complete model predictions were used to plot a scatter plot to observe how well the expected relation of y = x is satisfied. If the ground-truth was available for test set, the MSE loss is computed between ground-truth and prediction.

6.3 Performance Metrics

The following performance metrics were used to analyse multiple models tried:

1. Mean Square Error Loss: defined as $\frac{1}{n}\sum_{i=1} n(\hat{y_i} - y_i)^2$,
2. Mean Average Error Loss: defined as $\frac{1}{n}\sum_{i=1} n|\hat{y_i} - y_i|$,

where $\hat{y_i}$ is the model predicted crowd-count and y_i is ground-truth crowd-count.

7 Initial Results with CSRNet

The CSRNet architecture was implemented as done by Li and Zhang et.al. in their original paper. The front end is taken to be that from VGG-16 because

of its strong transfer learning ability and its flexible architecture for easily concatenating the back-end for density map generation. The classifier portion of the VGG network has been replaced by a custom back-end consisting of dilated convolutions layers for extracting deeper information of saliency as well as maintaining the output resolution. The network was constructed by modifying the original code presented by the authors so that it works on the latest Python platforms in conjunction with the Google Colab online GPU platform.

We go for the SGD+ Momentum loss optimizer (as suggested by the authors of the original paper) and tune its hyperparameters in order to get the desired accuracy. The evaluation metrics were chosen to be the MAE and MSE estimates, calculated as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{GT}|$$

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{GT}|^2}$$

Where N is the number of images in one test sequence and C_i^{GT} is the ground truth of counting. C_i represents the estimated count which is defined as follows:

$$C_i = \sum_{l=1}^{L}\sum_{w=1}^{W} z_{l,w}$$

L and W show the length and width of the density map respectively while $z_{l,w}$ is the pixel at (l,w) of the generated density map. C_i means the estimated counting number for image X_i.

7.1 Case 1: SGD + momentum, learning rate = 1e-2, decay = 5e-4, momentum = 0.95

Result: Model was trained for 100 iterations however the learning rate turned out to be too large as the model predicts zero headcount for all cases. MAE over the entire test dataset: 429.2457.

7.2 Case 2: SGD + momentum, learning rate = 1e-4, decay = 51e-4, momentum = 0.95

Model trained for 50 iterations and it was observed that the model starts learning (has a non-zero head count). However the performance is still very poor, count for most images is less than one. MAE over entire test dataset: 429.2453

7.3 Case 3: SGD + momentum, learning rate = 1e-6, decay = 5e-4, momentum = 0.95

Model trained for 100 iterations after further reducing the learning rate. The learning is much improved compared to previous cases (Although the head count is still lower than required). The density maps slowly begin to seem similar to the ground truth.MAE over entire test dataset 315.5924

7.4 Case 4: Adam optimizer

Initially on training the model with Adam optimizer for 100 iterations, the model learns nothing new (zero headcount on all the images and blank density maps). MAE over dataset was 429.2457
On tuning the Adam optimizer hyper-parameters, the learning was accelerated and the model gave better results than the SGD + momentum optimizer.

7.5 Case 5: SGD + momentum optimizer with frozen layers

The model was trained with pre-trained weights for all the layers except for the last few layers (also varied to find the optimum combination). This is done with the assumption that the lower level feature extraction is same for all cases and therefore by training the last few layers to our specified case will give higher accuracy with minimum computational overhead. The model was trained with frozen layers for 60 epoch, 300 steps per epoch.
Results: Much better results in lesser iterations compared to the last two, the model seems to be learning to correlate the density maps to the head count. Better results where the number of people are closer to the average crowd count (approx. 200-300). Doesn't work well in images where the crowd count is too low or high
MAE over test dataset: 312.1853

However the above implementation draws heavily from previous work and therefore has no novelty. Also training with the VGG-16 as the primary backbone structure is computationally expensive and therefore can't be deployed on actual hardware where crowd counting algorithms need to be practically used. Therefore the focus of the current work will be towards devising a network architecture that is inexpensive to train and deploy, while also giving good accuracy on the head count. In order to do so, different existing networks were tested in order to choose the Primary architecture for our case.

8 Primary Network Architecture Selection

In order to find a network that can act as the backbone of the final network, different existing architectures were implemented from scratch and tested on the aforementioned dataset. The results are as follows:

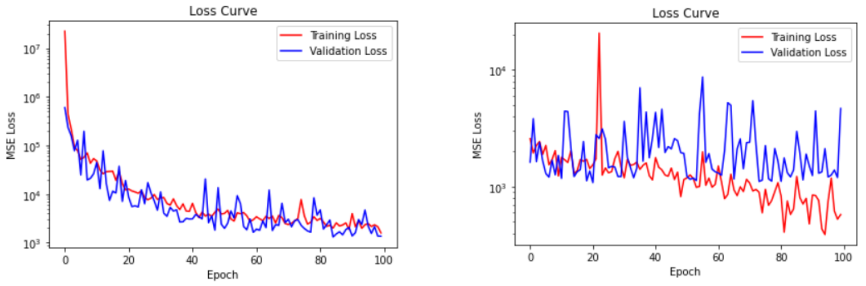
8.1 SqueezeNet

SqueezeNet was developed by researchers at DeepScale, University of California, Berkeley, and Stanford University with the goal to create a smaller neural network with fewer parameters that can more easily fit into computer memory. Therefore it aligns perfectly with our objective.

A squeezenet network was built with Adam optimizer having default learning rate and was trained for 100 iterations on the training set B with a 80:20 training: validation split. In order to investigate the model accuracy only the first three squeeze-net modules were considered in the network front-end. A custom-made network backend with 128, 128, 64 neurons was chosen (after several other combinations) followed by a $1\times$ 1 convolution at the output layer.

Results: The network training is much faster compared to the previous models (The effect of cheaper convolutions is clearly visible) and was trained for 100 epochs (Figure 3). The generated density maps too were found to be better however in certain examples they had negative pixel values, indicating slight overfitting inspite of the overall loss trend. On training the model for a 100 more epochs the overfitting became more pronounced as the training results matched well but the generated density maps were very poor. Similar trends were obtained on training for 95 epochs where the negative pixel value problem was resolved at the expense of deteriorating density maps.

On further tweaking the hyperparameters and training on the A and B datasets it was evident that the model was too sensitive to the training data as well as the network architecture and therefore was not further pursued.



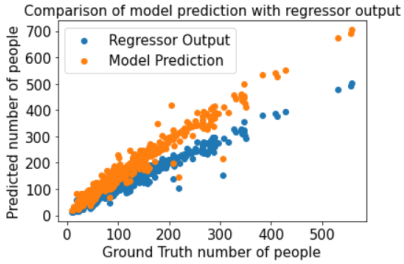
(a) Training loss after 100 iterations

(b) Training loss for 101 to 200 iterations

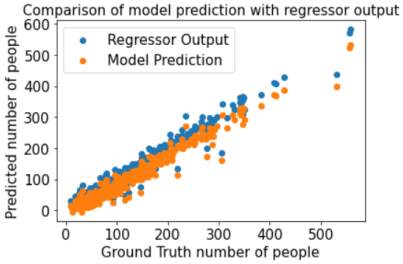
Figure 3: SqueezeNet Training

8.2 DenseNet

In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Thus while it is more

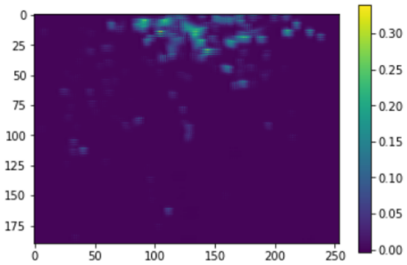


(a) Headcount after 200 iterations

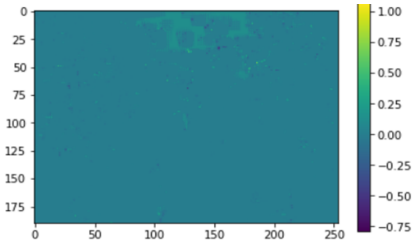


(b) Headcount after 95 iterations

Figure 4: SqueezeNet Training Results



(a) Density map at 100 epoch



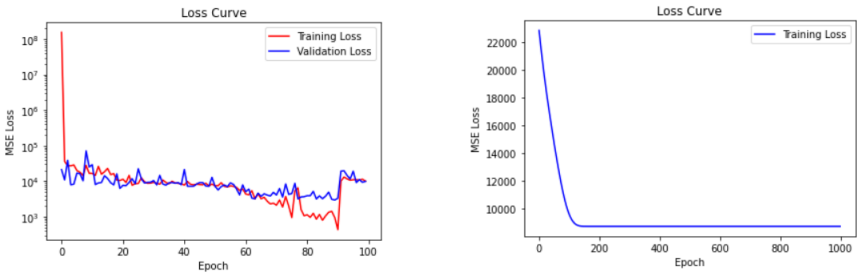
(b) Density map at 95 epochs

Figure 5: Generated density map at different epochs

computationally expensive, each layer receives connective knowledge from all preceding layers. Initially a single Densenet block (consisting of 6 layers) was trained as the front-end, with the backend having three layers with 384, 128, 64 neurons followed by a 1 by 1 convolution layer.

The training takes a very long time inspite of taking a single bock (highest among all models). Adam optimizer (after tuning of hyperparameters) was used with a 80:20 split of training data. On seeing the results (after 100 epochs) it is clear that the model is learning, however the learning rate is really slow compared to the training time. Also on plotting the headcount a bias can be seen in the data, presumably due to the relatively smaller number of layers used in the network.

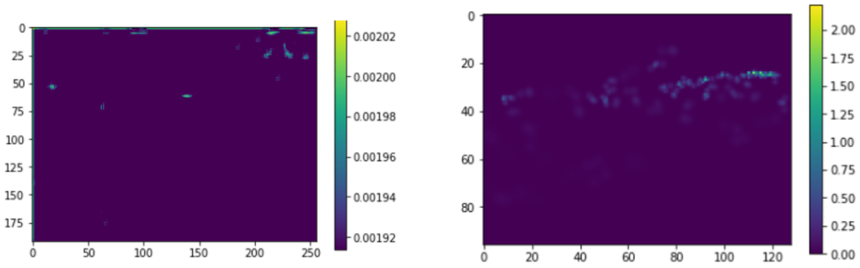
Therefore due to the necessity of higher number of network layers and the resulting very high training time to produce satisfactory results, this network is not further produced.



(a) Training loss variation

(b) Regressor Loss variation

Figure 6: Training loss for DenseNet



(a) Generated density map

(b) Ground truth density map

Figure 7: Density Map comparison for DenseNet after 100 epochs

8.3 VGG-16 with Batch Normalisation

In a modification to the traditional VGG-16 employed in the CSRNet, we add additional batch normalisation layers in order to correct the internal covariate shift that may be occuring in the deep neural network. Internal covariate shift makes the network sensitive to initial layer weights (especially in a deep network) akin to chasing a moving target. Therefore it is expected that adding the batch normalisation layers will reduce the training time too.

end had first 30 layers from VGG16 along with added BN layers, followed by backend with a [512, 512, 512, 256, 128, 64] structure and a 1 times 1 convolution. The network was trained for 100 epochs with Adam optimizer and 80:20 data split. Although the speed increases slightly compared to the CSRNet but it is still higher compared to SqueezeNet. The generated density maps were also of inferior quality, with the output data massively overfitiing on the training dataset. Hence this model too was discarded.

8.4 MobileNet V2

MobileNet was proposed by Google in 2017, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases.

The use of depthwise separable convolutions i.e. a depthwise convolution followed by a pointwise convolution enables exponential reduction in the number of operations and required parameters, which is the primary reason behind it's high speed with good accuracy. The working of the depthwise separable convolutions can be seen in Figure 8

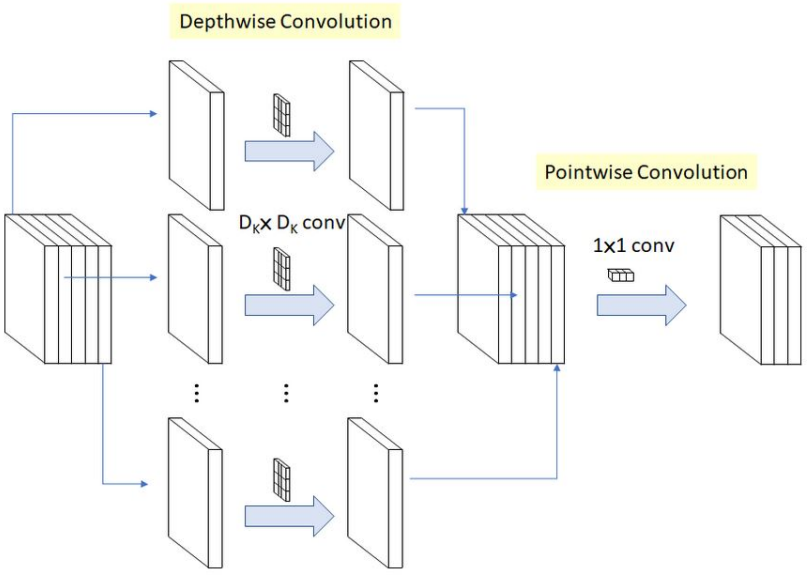


Figure 8: Depthwise Separable Convolutions

Using the depthwise separable convolutions decreases the number of convolutions from $D_K * D_K * M * N * D_F * D_F$ to $D_K * D_K * M * D_F * D_F + M * N * D_F * D_F$.

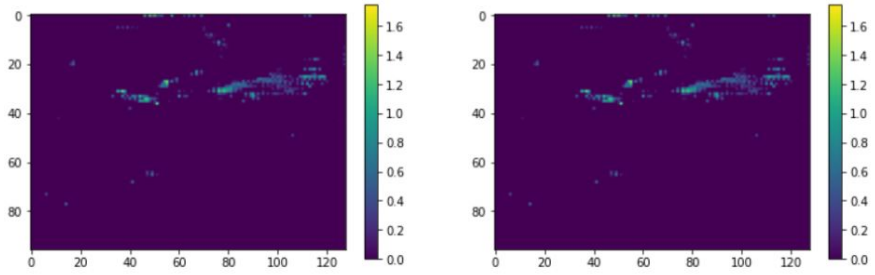
Here M: Number of input channels, N: Number of output channels, D_K: Kernel size, and D_F: Feature map size. Thus the computation cost decreases by a factor of

$$\frac{1}{N} + \frac{1}{D_K^2}$$

For these reasons, MobileNet V2 seems to be the perfect choice for our crowd-counting application.

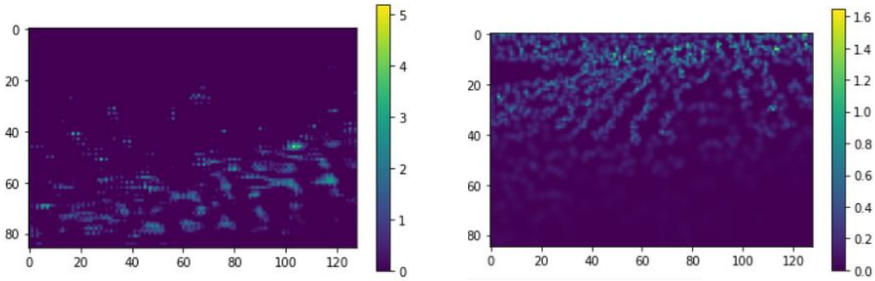
The network was constructed with 5 initial layers from the MobileNet V2 as

the front-end and the back-end with 32,16 neurons (2 layers) followed by a 1 by 1 convolution. The network was initially trained for 50 epochs. The model learnt the density map extraction to a greater extent than all the previous models and the crowd-count too was satisfactory (Linear regressor output didn't differ from the model output, indicating negligible bias Figures 9 and 11). As the validation set loss too had not reached a minimum, the model as trained for 100 epochs next. Here the density map was further improved and the model fit the data well (Figure 10).



(a) Generated density map (b) Ground truth density map

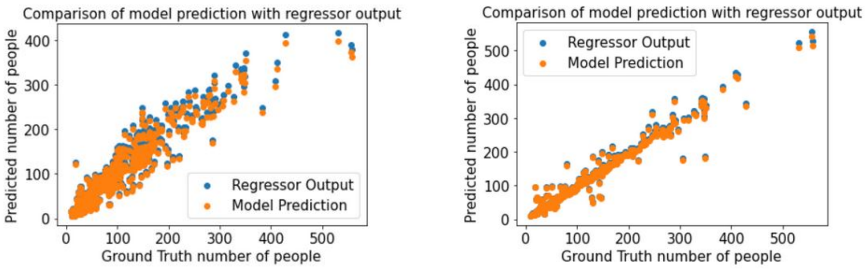
Figure 9: Density Map comparison for Mobilenet V2 after 50 epochs



(a) Generated density map (b) Ground truth density map

Figure 10: Density Map comparison for Mobilenet V2 after 100 epochs

Due to the success obtained from this model on the B training dataset (even on randomly choosing the number of front-end layers) and its fast training time, the model is selected as the primary structure for our network. However the hyperparameters of this network need to be varied and tuned in order to further improve its performance on the current as well as other datasets.



(a) Crowd count after 50 epochs (b) Crowd Count after 100 epochs

Figure 11: Crowd count comparison with ground truth for Mobilenet V2

9 Hyperparameter Tuning

In order to guage the effectiveness of a hyperparameter combination, the assessment parameters used are:

- **Degree of Linearity** of the Generated crowd count versus Ground Truth crowd count curve.
- **Accuracy of the generated density maps** i.e. the density maps should have high local resolution and be able to differentiate individual instances rather than just fit the training data.

9.1 Effect of number of Frontend layers and training dataset

In order to study the effect of the number of front-end layers from the MobileNet V2 that are used in the network, the other hyperparameters were to be kept constant (Adam optimizer with 80:20 data split). The noteworthy results from each of these simulations are shown in the table below (Obtained after 35+ simulations).

The training set A of the ShanghaiTech dataset consists primarily of images with high crowd count and mostly even distribution of crowds throughout the image whereas the B set has images with more localised crowd density and lower crowd count. Therefore the hyperparameter combination that leads to better performance on both the sets individually as well in combination is expected to perform better on general datasets and therefore is desirous.

No. of Layers	Training Dataset	Noteworthy Results
5	A	Good results on Crowd Count at lower crowd densities but poor density map generalisation. Suspected bias in the model
5	B	Poor results compared to dataset A. Model suspected to have poor local resolution of spatial data
5	A+B	Better results compared to dataset B. Model attempts to cluster individual data together using higher average pixel values Poor density map generation
7	A	Excellent agreement on crowd count with less deviation even on testing data but poor density map generation with global clumping of instances
7	B	Better generalisation compared to 5 layer case Good local resolution, density maps with pixel range similar to ground truth density maps
7	A+B	Crowd count on test dataset fits well on high counts too. Poor density maps, high pixel value problem persists
9	A	Good crowd count degree of linearity with slight deviation only at higher crowd counts in the test dataset due to the less number of training samples available for higher counts. Density map generation has good local as well as global resolution.
9	B	Similar results as in dataset A with respect to both the density map generation and comparison of calculated crowd count with the ground truth High linearity in the curve
9	A+B	Best performance yet with the crowd count lying almost exactly on the y=x on the training data while simultaneously having less variance on the test data and less number of outliers even at higher crowd densities. The generated density maps also mimic the ground truth to a higher extent (on the test data) with reasonable pixel value ranges. Even at higher pixel value ranges, the density maps gave a reasonably good representation of data. Recommended for further refinement
11	A	Fits the training data really well but overfits on the test data Poor density map generation with very high tendency of clustering individual instances by increasing the pixel value range much higher than ground truth density maps
11	B	Similar results as that with A dataset but density map generation is poorer, indicating poor local resolution brought upon by overfitting. Need to reduce the number of front-end layers

Table 1: Hyperparameter Tuning Analysis

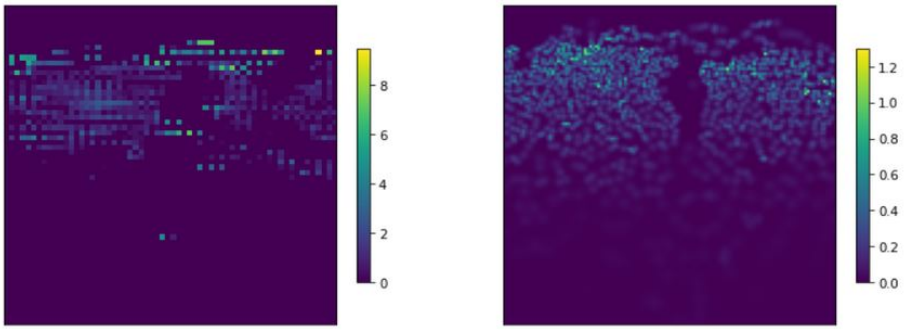


Figure 12: Density map from 9 layer MobileNet network trained on A+B dataset

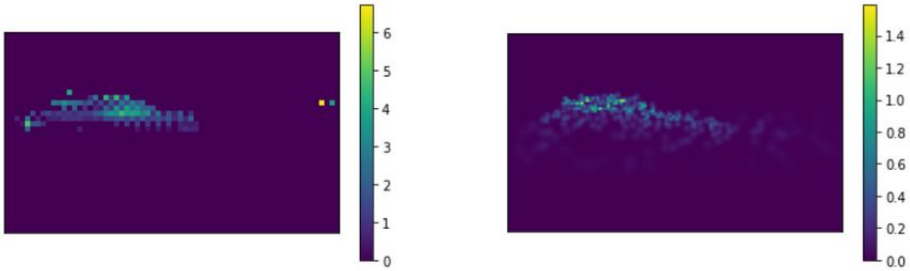
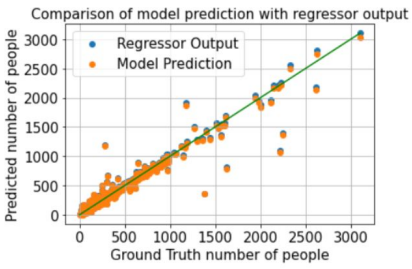
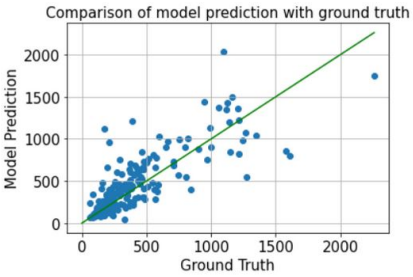


Figure 13: Density map from 9 layer MobileNet network trained on A+B dataset



(a) Crowd count on the training dataset



(b) Crowd count on the test dataset

Figure 14: Crowd count statistics on the network with 9 front-end layers

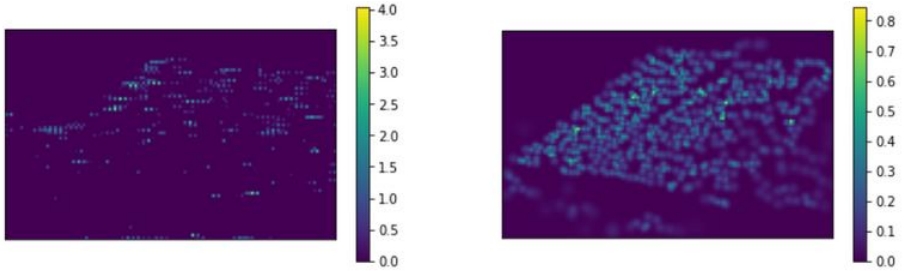
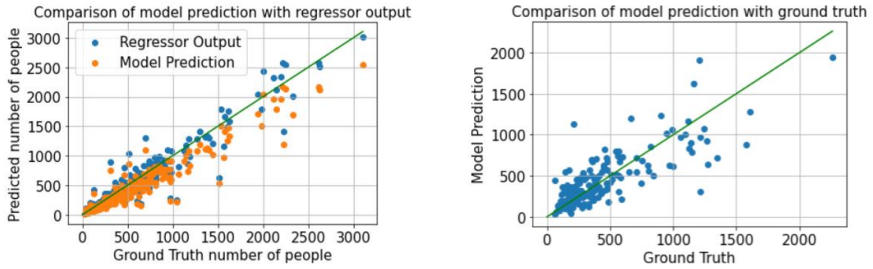


Figure 15: Density map from 5 layer MobileNet network trained on A+B dataset



(a) Crowd count on the training dataset

(b) Crowd count on the test dataset

Figure 16: Crowd count statistics on the network with 5 frontend layers

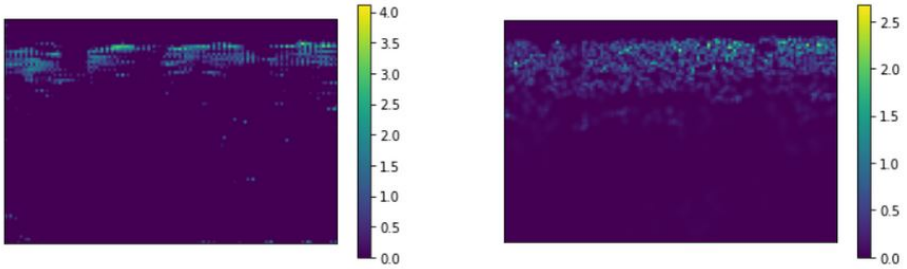
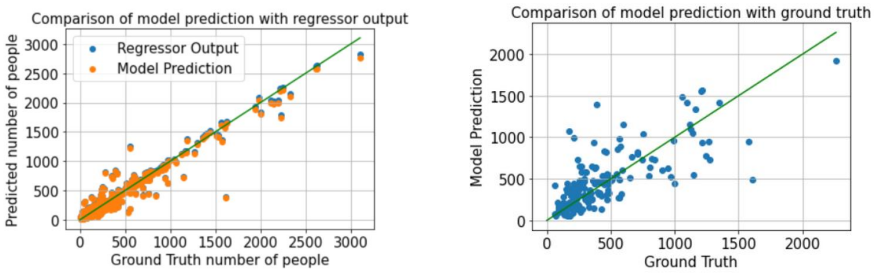


Figure 17: Density map from 7 layer MobileNet network trained on A+B dataset

The pixel values that has been referred to multiple times in Table 1 is important as the density map generation in ground truth has been done to ensure that the sum of pixel values over the entire image equals the crowd count. In line with the same, the following checks need to be done on the obtained density maps:

- As the pixel values directly correlate to number of people, the pixel values can never be negative and therefore models with colourmap values less than zero should be discarded.



(a) Crowd count on the training dataset (b) Crowd count on the test dataset

Figure 18: Crowd count statistics on the network with 7 frontend layers

- The range of the generated density map should be as close to the ground truth as possible.
- If the maximum value of colourmap in the generated density map is much higher compared to the maximum colormap value in the ground truth density map, the model will try to club the surrounding individual instances to a single pixel, reducing the diversity and richness of the generated density map.
- Similarly if the minimum value is too low compared to the ground truth density map, it will force the density map to show an excess of pixels. This generates a density map much more noisier than the original.

Figures 14, 12, 13, 15, 16, 17 and 18 can be used to visualise the findings of Table 1. We see 9 layers of mobilenet as the network end followed by a simple back end of three fully connected connected layers (64,16,1 neurons) gives the best results on both the individual as well as the combined datasets. It gives a good crowd count estimate while producing density maps that faithfully represent the actual crowd density in the image. It is to be noted that each of the 11 configurations in Table 1 was run for multiple epochs and a maximum epoch of 100 was found to work well for the optimum case (early stopping to prevent overfitting). The Adam optimizer with default settings has been used till now but further hyper-parameter search needs to be done over the loss function to obtain the final model.

9.2 Effect of loss function and learning rate

The default learning rate for Adam optimizer 0.001 which has been used until now. This default learning rate (LR) needs to be tweaked in order to suit our specific problem. The network architecture for all the cases is the same, with 9 MobileNet layers in the frontend. The results when tested on different datasets is shown below:

From the results of Table 2, the model on training with Adam optimizer(learning rate of 0.0006, (β_1, β_2)=(0.9, 0.999), ϵ=1e-08, weight decay=0) for 150

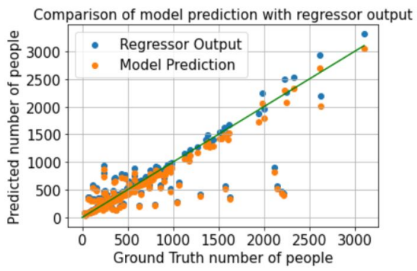
LR	Epochs	Training Dataset	Key Observations
0.005	100	A	Learning rate is too high Constant crowd count and flat density maps
0.002	100	A	Improved learning and crowd count linearity Density maps show global crowd resolution
0.002	100	A+B	Good crowd count linearity, suffers from poor density maps (bad generalisation to new data)
0.0008	150	A+B	Good crowd count linearity, suffers from overfitting and excessive clustering of pixels leading to poor density map generation
0.0006	100	A	Crowd count agreement with ground truth Density map starts representing the crowd density but poor resolution, train more
0.0006	150	A	Excellent crowd count agreement with ground truth. Generated density maps have good global and local resolution
0.0006	150	A+B	Similar results as above works for both low and high density crowds
0.0006	200	A	Excellent crowd count linearity on training data Does not predict well on Test data (saturation at high crowd counts). Generated density maps suffer from poor representation (suspected overfitting).
0.0004	150	A	Good crowd count estimation even on the test dataset density maps display crowd density recognition however the resolution is poor (needs more training) Final results expected to be similar to lr=0.0006 case.

Table 2: Effect of Learning Rate (LR) on Model with Adam optimizer

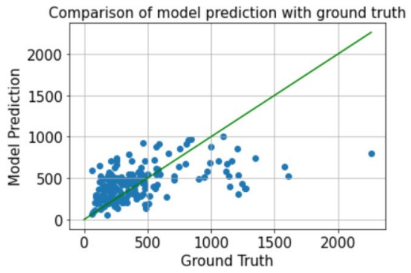
epochs balanced the dual objective of good crowd count (Figure 19) and accurate density map generation (Figures 20 and 21). The performance was undulating on either dataset and showed good generalisation.

The Model was also trained with the SGD and SGD + Momentum optimizer for different hyperparameter combinations. However the optimzer returned NaN values for most case and in the cases where the losses were finite (certain combinations of learning rate, momentum, dampening and weight decay), the model was very sensitive to training data and therefore the optimizer was deemed unfit for further analysis.

The third optimizer tried out was the RMSProp optimizer (chosen due to its prevalent use in Deep Learning literature). The model was initially trained only on the A dataset at different learning rates (0.001, 0.0008, 0.0005, 0.0007, 0.0012, 0.0004 and 0.0006) for 100 and 150 epochs (in order to look for over-



(a) Crowd count on the training dataset



(b) Crowd count on the test dataset

Figure 19: Crowd count statistics on network with 9 frontend layers and Adam optimizer with tuned learning rate

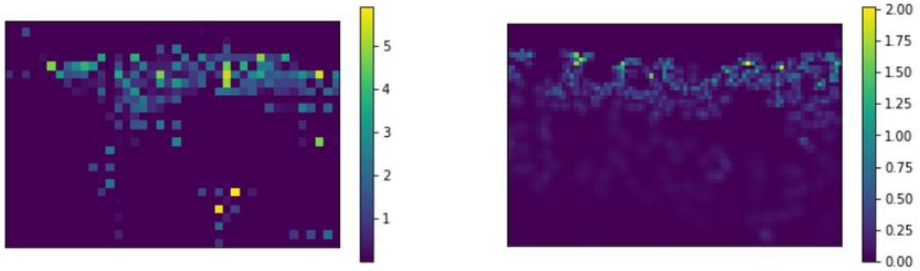


Figure 20: **Left:** Generated density Map **Right:** Ground Truth density map. Model with optimized Adam learning rate

fitting prevention via early stopping). The model with learning rate of 0.0005 performed better on both the crowd count as well as the density map generation parameters and hence was next trained with images of both dataset A and B, to check its generalisation capabilities as well as its performance on a combination of high and low crowd density images. The results showed good fit on the training data and density maps with a semblance to the actual image crowd density (Figures 23, 24). The persisting problem with the RMSProp optimizer though was the bias in crowd count generated on the test samples (Figure 22). All models showed high inaccuracy at higher counts, clustering at median crowd count values and therefore this optmizer too was discarded, going ahead with the Adam optmizer.

10 Data Augmentation

The most common data augmentation techniques that are generally used in Neural Networks cannot be applied to our problems due to its unique nature. Some of those techniques are as follows:

- Cropping: Since the ground truth generation depends on the location of

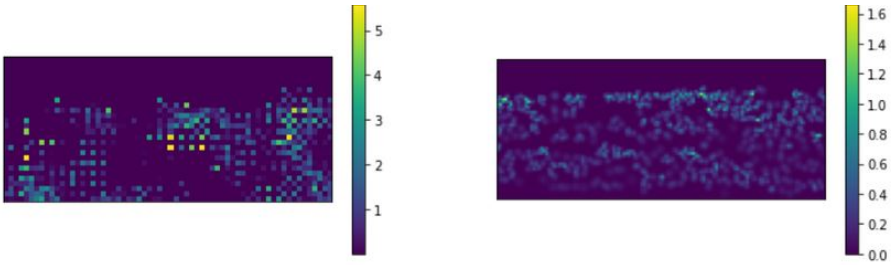
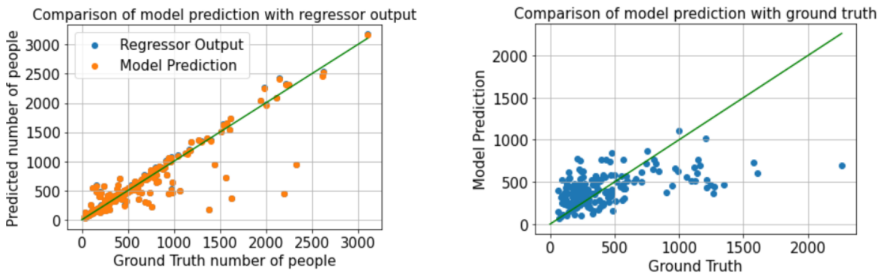


Figure 21: **Left:** Generated density Map **Right:** Ground Truth density map. Model with optimized Adam learning rate



(a) Crowd count on the training dataset

(b) Crowd count on the test dataset

Figure 22: Crowd count statistics on network with 9 frontend layers and RMSProp optimizer with tuned learning rate

the people in the image (calculated manually and available as a .dat file) cropping the image makes the generation of the ground truth very complex as the number of people that will be cropped out in each image will be different and therefore the ground truth generating algorithm will generate wrong density maps (inaccurate total crowd count).

- Rotation: Similar to cropping the image, rotation will change the position of each person in the image (x,y coordinate) and therefore the ground truth generation will return a false density map for each image.
- Translation: Translation shifts the image horizontally and vertically by a certain amount and hence again brings randomness into the number of people present in the actual image, generating errata in the ground truth density maps
- Random Erasing: Random erasing augments data by removing certain portion of each image. Although it is theoretically possible to estimate the number of people removed by considering the bounding box and coordinates of crowd available but due to the practical difficulty of implementing it in a generalized fashion (that too on a dataset with non-uniform image

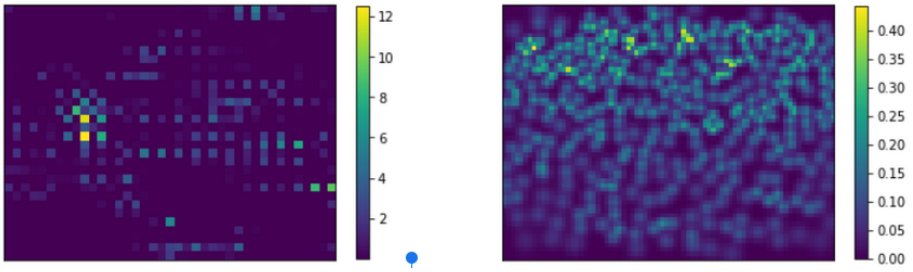


Figure 23: **Left:** Generated density Map **Right:** Ground Truth density map. Model with optimized RMSProp learning rate

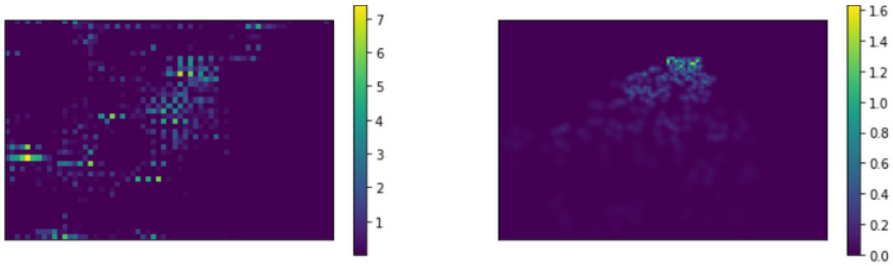


Figure 24: **Left:** Generated density Map **Right:** Ground Truth density map. Model with optimized RMSProp learning rate

sizes) it is not implemented here.

Other advanced data augmentation techniques such as colour space transformations, using specialised Kernels (Kang et. al.), Image mixing, Feature space mixing and many others exist but have not been tried out for the current study. Hence mirroring is the only data augmentation technique that remains to be tried out.

With the 9 layer MobileNet frontend network (which was found to work the best) the training was carried on the A dataset with mirroring, B with mirroring and A+B with mirroring.

- **A with mirroring**: The crowd count showed a high degree of linearity with the ground truth (fits the training data well) but falters on the test dataset as the prediction is close to ground truth only at counts less than 800. Model shows a bias for lower counts on the test datasets. The generated density maps suffer from the same poor crowd representation problem, with a much higher pixel value range than the ground truth.
- **B with mirroring**: Here too the generated crowd count shows an excellent agreement with the ground truth on the training data. On the test data too, barring a few high crowd cases, the result is closely clustered

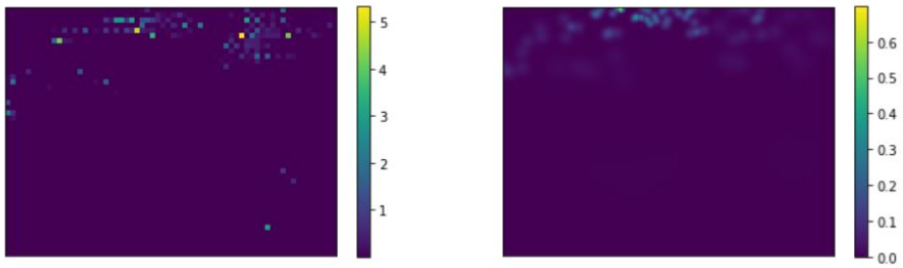


Figure 25: **Left:** Generated density Map **Right:** Ground Truth density map. Model trained on A+B+mirrored B dataset

about the y=x curve (ideal curve). However we see a drastic improvement in the density map generation compared to the A dataset. The density maps show an accurate representation of the actual crowd count density and though the local resolution accuracy still lags, the generated density maps accurately capture the global trends. The reason for this may be due to the nature of a B dataset, which consists of low crowd counts which are spatially localised in the image (i.e. do not cover the entire crowd). This enables the mirrored images to be appreciably different from their parent images avoiding redundancy in the dataset (This redundancy is the expected reason behind the poor performance of the A dataset as the mirrored images are too similar to the original case, leading to high overfit).

- **A+B with mirroring**: Following the trends above the crowd count estimate was better compared to only A dataset and the generated density maps too persisted with the same problems as that of the A datset although too a lesser extent due to the influx of B images in the training data.

Hence the model was trained on a combination on A images with B images and mirrored B images. The resulting density maps for images with low/sparse crowd density is shown in Figures 25 and 26. We see that the model is able to accurately capture the crowd densities for this case also.

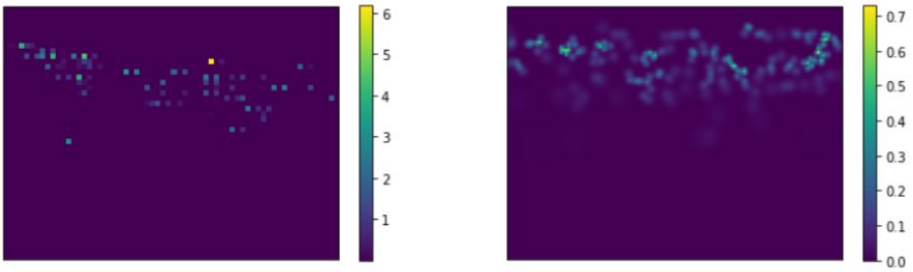


Figure 26: **Left:** Generated density Map **Right:** Ground Truth density map. Model trained on A+B+mirrored B dataset

11 Gabor Filter

A Gabor filter is used for detection of edges and texture changes[4]. There are certain parameters that affect the output of a Gabor filter. In the realms of image processing and computer vision, Gabor filters are generally used in texture analysis, edge detection, feature extraction, disparity estimation (in stereo vision), etc. Gabor filters are special classes of bandpass filters, i.e., they allow a certain 'band' of frequencies and reject the others.

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

In OpenCV Python, following is the structure of the function that is used to create a Gabor kernel:

cv2.getGaborKernel(ksize, sigma, theta, lambda, gamma, psi, ktype)

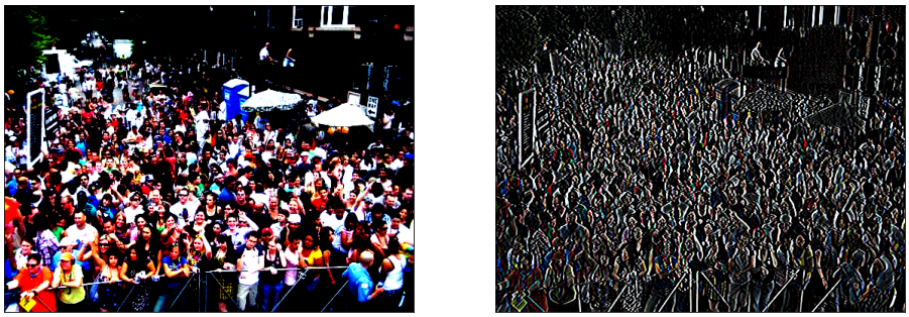
where,

- **ksize** is the Gabor kernel size. If ksize = (a, b), we then have a Gabor kernel of size a x b pixels. As with many other convolution kernels, ksize is preferably odd and the kernel is a square (just for the sake of uniformity).
- σ is the standard deviation of the Gaussian function used in the Gabor filter. This parameter controls the width of the Gaussian envelope used in the Gabor kernel.
- θ is the orientation of the normal to the parallel stripes of the Gabor function. This parameter decides what kind of features the filter responds to. For example, giving θ a value of zero means that the filter is responsive only to horizontal features only. For a θ value of $\frac{\pi}{2}$, the filter responds only to vertical features.
- λ is the wavelength of the sinusoidal factor in the above equation.

- γ is the spatial aspect ratio. Gamma controls the ellipticity of the gaussian. When gamma = 1, the gaussian envelope is circular.
- ψ is the phase offset.
- **ktype** indicates the type and range of values that each pixel in the Gabor kernel can hold.

For the present analysis, the aforementioned parameters of the Gabor filter were varied to obtain a kernel which produces a pre-processed image which is potentially able to out-perform the regular approach (using only the image directly). The following are some of the sample filtered images for the set of parameters: (kernel size = 7, $\sigma = 2$, $\theta = 0$, $\lambda = 5$, $\gamma = 10$, $\phi = 0$).

(a)



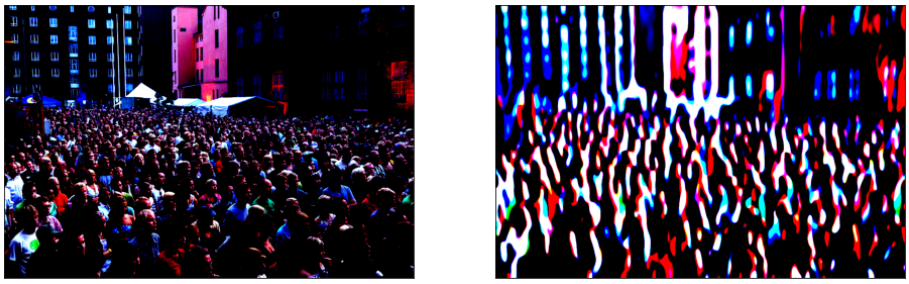
(b)

Figure 27: Gabor filtered samples training images (kernel size = 7, $\sigma = 2$, $\theta = 0$, $\lambda = 5$, $\gamma = 10$, $\phi = 0$)

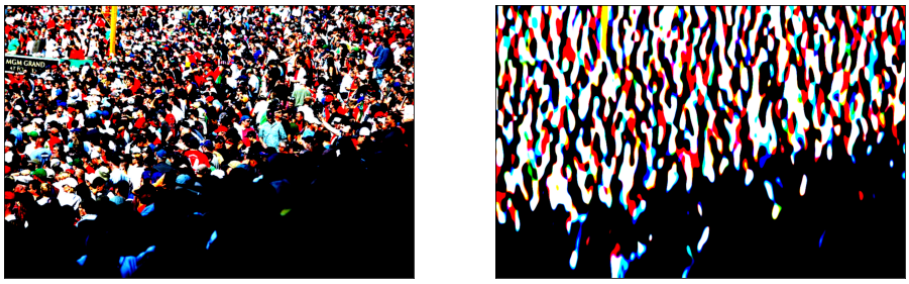
The following are some of the sample filtered images for the set of parameters: (kernel size = 50, $\sigma = 10$, $\theta = 0$, $\lambda = 30$, $\gamma = 1$, $\phi = 0$).

11.1 Effect of Gabor filter on mirrored datasets

Using Gabor filter in conjunction with the best model had minimal changes in the output, however a major contribution of the Gabor filter was seen in



(a)



(b)

Figure 28: Gabor filtered samples training images (kernel size = 50, $\sigma = 10$, $\theta = 0$, $\lambda = 30$, $\gamma = 1$, $\phi = 0$)

combating datasets with large number of repeated images (such as the mirrored dataset of A). It was seen that using a Gabor filter with a smaller kernel size and $\theta = 0$, the resulting model showed better generalisation on the test dataset with less overfit on the training dataset compared to other models (Figure 29) which overfit on the training dataset but have inferior performance on test dataset. This offers scope for future investigations using networks with Gabor filter pre-processing which on sufficient training give better results on the test dataset than in the current study.

12 Ensemble training

Ensemble learning involves training more than one network on the same dataset, then using each of the trained models to make a prediction before combining the predictions in some way to make a final outcome or prediction. Neural network models are nonlinear and have a high variance, which can be frustrating when preparing a final model for making predictions. Ensemble learning combines the predictions from multiple neural network models to reduce the variance of predictions and reduce generalization error. When done correctly it generates predictions that are better than any single model. For our model, ensemble

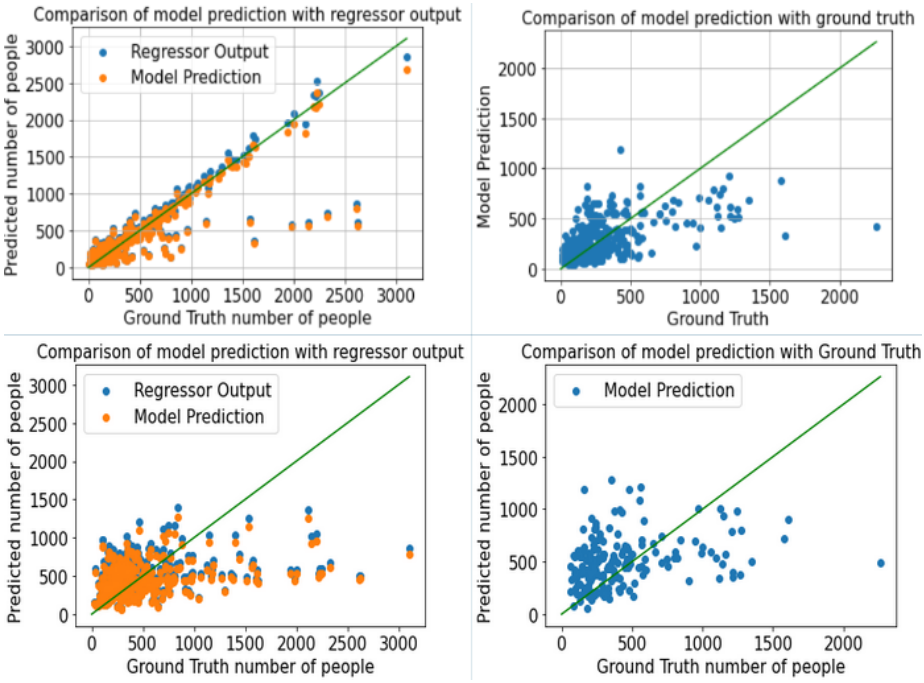


Figure 29: Clockwise from left: Crowd count on training samples without Gabor filter, Crowd Count on test samples without Gabor filter, Crowd count on training samples with Gabor filter, Crowd Count on test samples with Gabor filter

predictions can only be made on the crowd count and not on the density maps as each network has a different final output activation size. The reason behind not forcing the outputs to have a similar size is as follows:

- We work with models having pre-defined architectures (MobileNet, DenseNet, etc) and correspondingly are able to utilize the pre-trained weights that have been obtained after training that network on large datasets such as the ImageNet. In order to obtain images with the same output size, the network architecture has to be changed drastically, which makes it impossible to use the pre-trained weights.

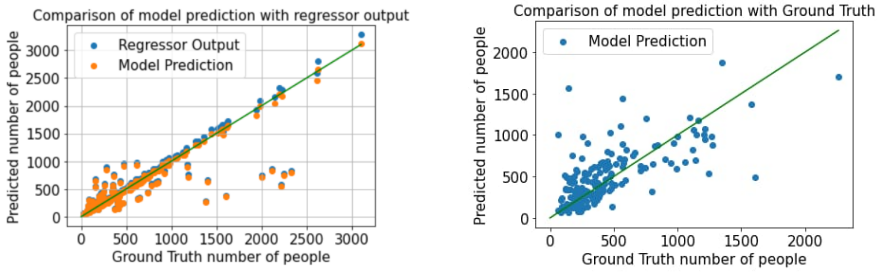
- Without the pre-trained networks for initialisation, the network training will take much longer time and that too won't give accurate results because of the smaller dataset we are training on leading to poor generalisation and overfitting.

- While image compression is easier to do, stretching an image to fit a certain window size without deterioration in quality is a tough task. Add to that the high variability in image sizes and aspect ratios of the dataset makes it

very computationally expensive to resize the images in the pipeline before entering the network. Therefore this approach too is discarded.

In order to find the optimal network combination the models from each network that showed the best results (DenseNet, Mobilenet with different optimizers and VGGNet) were combined with varying weights. The performance metrics were linearity of the generated crowd count with the true crowd count on both the training and test datasets.

The final optimal network had Mobile net with Adam optmizer (9 frontend layers) and DenseNet (6 frontend layers) in 85:15 ratio. We see crowd count agreement on both the train and test datasets (Figure 30/ It is note worthy that although DensenNet alone did not perform as well on the crowd count metric, using it in situ with Mobilenet performed better than either of the networks.)



(a) Crowd count on the training dataset

(b) Crowd count on the test dataset

Figure 30: Crowd count statistics on ensemble network

13 Final Model and Results

13.1 Proposing a new loss function

In each of the density map generated, the primary reason for their deviation from the ground truth density map is the representation of cluster of pixel values in the original map by a single pixel of larger value in the learnt map. This makes the crowd representation sparser than it actually is, leading to inaccurate conclusions. Even after extensive model search and hyper-parameter tuning the above problem persists. The reason for this problem to a certain degree is the faulty loss function too as it penalises the model only if the total crowd count differs from the ground truth and doesn't care how the crowd count is actually generated.

In order to combat the above problem, a new loss function is proposed that penalises the range of pixel values too, in addition to the crowd count. Doing so ensures that model doesn't try to club neighbouring pixels into one pixel of large

value and thus is expected to improve the local resolution and differentiation capability of the generated density maps. The addition of these terms which effectively act as regularizers also don't effect the convexity and convergence property of the loss function while training (verified using the time taken for model convergence). The modified loss function is:

$$Loss = (S_g - S_a)^2 + (P_{max,g} - P_{max,a})^2 + (P_{min,g} - P_{min,a})^2$$

Where:
S_g is sum over pixel values of generated density map (which is the generated crowd count)
S_a is sum over pixel values of ground truth density map (which is the true crowd count)
$P_{max,g}$ is the maximum pixel value in the generated density map
$P_{max,a}$ is the maximum pixel value in the ground truth density map
$P_{min,g}$ is the minimum pixel value in the generated density map
$P_{min,a}$ is the minimum pixel value in the ground truth density map

The last two terms can be thought of as a regularizer, forcing the generated pixel range to equal the ground truth pixel range. This in turn forces the model to represent the actual crowd density, increasing the number of pixels it has to use and thus increasing local resolution.

The optimised model obtained earlier with 9 Mobilenet layers as the front-end and a modified back-end with Adam optimizer was used with this loss function. The generated density maps are shown in Figure 31. We see that although the pixel values come close, they are not matching. Also the improvement in the generated density maps is not too great. This is due to the fact that the additional loss terms are small numerically compared to the main loss term and therefore their effect on the training is not significant. To combat this, additional weight hyperparameters A and B are introduced in the Loss function as shown:

$$Loss = (S_g - S_a)^2 + A * (P_{max,g} - P_{max,a})^2 + B * (P_{min,g} - P_{min,a})^2$$

A and B control the relative importance of the closeness in pixel values to the total crowd count. If A and B are too high, the loss function gives too much importance to matching the pixel values than matching the crowd count, which results in density maps with excess local resolution that fail to capture the global trend (Figure 32).

The final optimised weights were found to be A=B=32 for which both the crowd count as well as the crowd density representation were optimised. This model showed good agreement with the crowd count on test images as well as faithful density map generation (Figures 34 and 35).

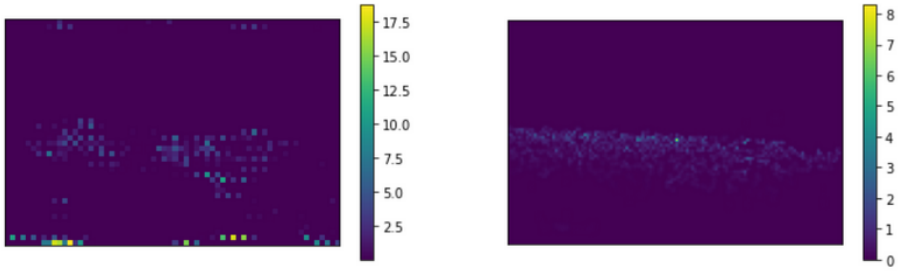


Figure 31: **Left:** Generated density Map **Right:** Ground Truth density map. Model trained on network with modified loss function without weights in the additional loss terms.

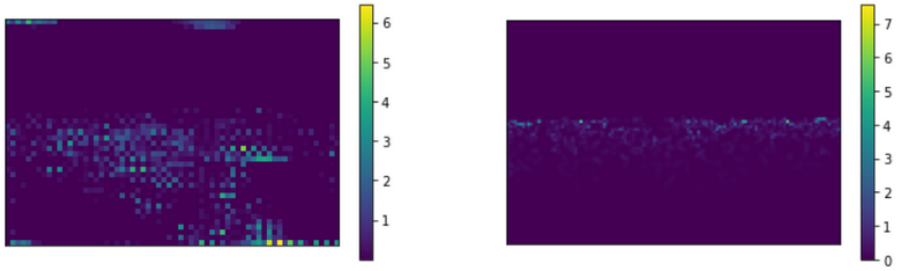
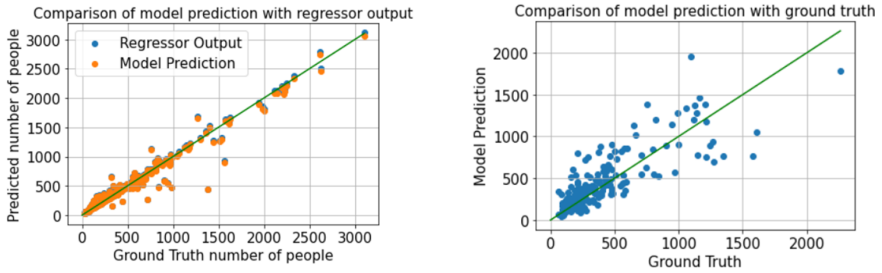


Figure 32: **Left:** Generated density Map **Right:** Ground Truth density map. Model trained on network with modified loss function without high weights (A=B=55)

14 Future Work

The current study found that using a CNN with 9-layer MobileNet frontend and Adam optimizer (with tuned hyperparameters) with modified loss function (optimal weight combination) gave the best combination of both crowd count accuracy and density model generation. However this has been obtained by working within a very limited framework and therefore the following improvements can be carried out in the future:

- Use a bigger dataset as the current study works with limited data due to computational limitations.
- The network can be configured to give output density maps of same output size, either by changing the input data itself or configuring the network to do it. This will require training the network from scratch and therefore is not done here.
- Following the above approach, the loss function can be modified to compute the loss by comparing individual pixel values between the ground



(a) Crowd count on the training dataset (b) Crowd count on the test dataset

Figure 33: Crowd count statistics on final model with weighted loss function

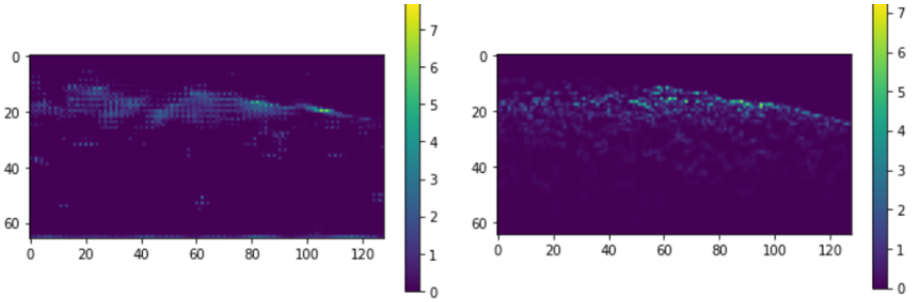


Figure 34: **Left:** Generated density Map **Right:** Ground Truth density map. Final model with modified loss function with optimal weights (A=B=32)

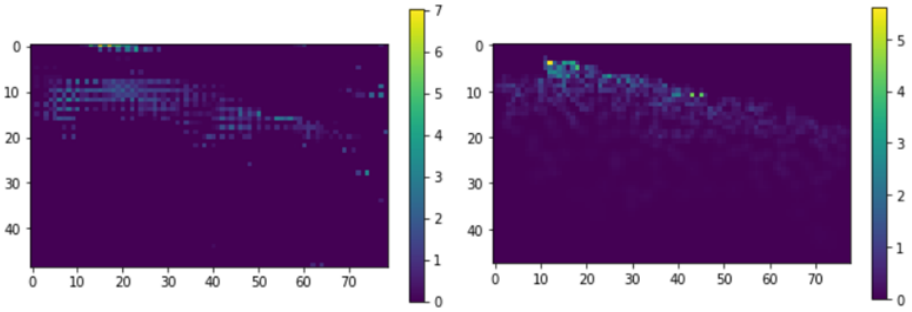


Figure 35: **Left:** Generated density Map **Right:** Ground Truth density map. Final model with modified loss function with optimal weights (A=B=32)

truth and generated image. This will make the density map generation more accurate and reliable as we're explicitly coding in the need to generate density map instead of only matching the crowd count as done here.

- Employ data augmentation other than the geometric transformations used

in this case to both increase the number of training examples as well as the model generalisation. For practical applications, the images will be not as good as those in the current training set (blurring, fog, rain, night time images) and the training set must be modified to account for that.

15 Conclusion

Crowd counting problem was apprehended with a CNN based approach. Initially the CSRNet model was investigated but abandoned due to its high training time and computational cost requirement. Subsequently existing models in literature like MobileNet, DenseNet, VGGNet and SqueezeNet were trained by varying hyperparameters like number of layers, learnning rate, loss function and front-end network until an optimal model and hyperparameter combination that maximised the performance metrics was found. Data augmentation was carried out to aid generalisation. The performance metrics used to assess the models were the generated crowd count agreement with the ground truth and qualitative assessment of generated density maps. The model was trained with two different datasets from the ShanghaiTech database and the results were subsequently analysed. Ensemble approach to training was also tested with excellent results on the crowd count parameter. Finally a novel loss function was proposed which was found to improve on both the crowd count as well as density map generation (with respect to the local and global resolution). Future work which can be carried out is also proposed.

References

[1] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 833–841, 2015.

[2] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 589–597, 2016.

[3] Li, Y., Zhang, X., & Chen, D. (2018). CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1091-1100.

[4] https://cvtuts.wordpress.com/2014/04/27/gabor-filters-a-practical-overview/

16 Appendix

16.1 Visualisation of ensembling on non-optimal models

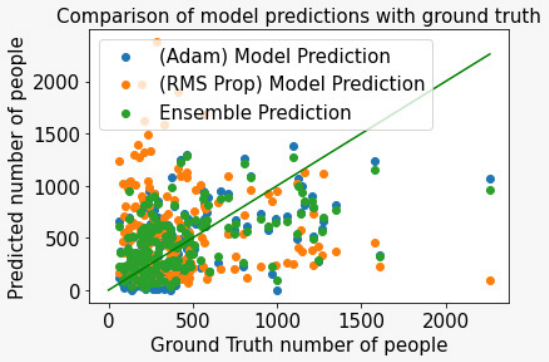


Figure 36: Effect on ensembling on Test data

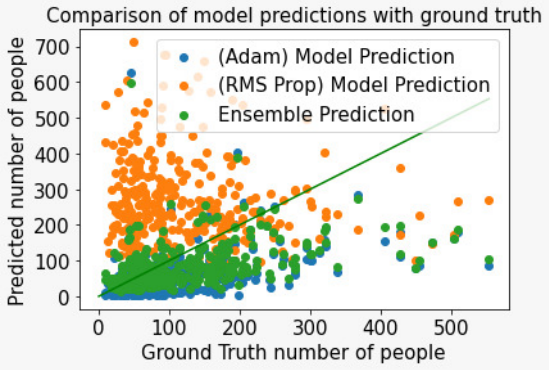
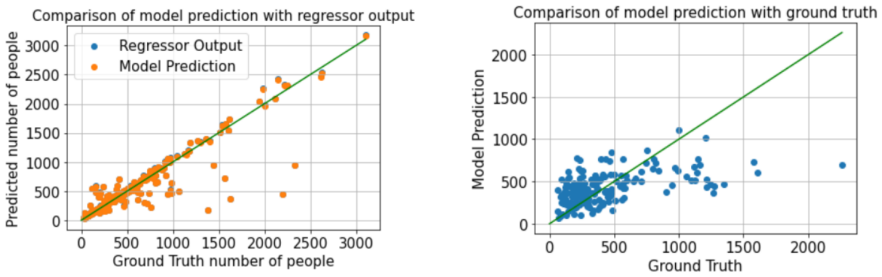


Figure 37: Effect on ensembling on Test data

16.2 Best Results from RMSProp optimizer LR=0.0005

16.3 Effect of overtraining

Results when the 9 layer mobilenet frontend model was overtrained on the Adam optimizer



(a) Crowd count on the training dataset (b) Crowd count on the test dataset

Figure 38: Crowd count statistics on final RMSProp model

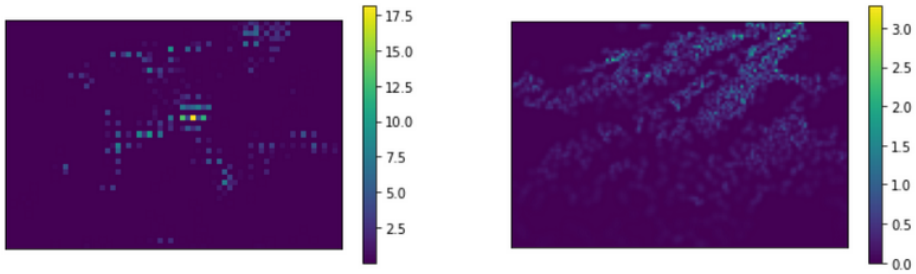


Figure 39: **Left:** Generated density Map **Right:** Ground Truth density map. For the final RMSProp model

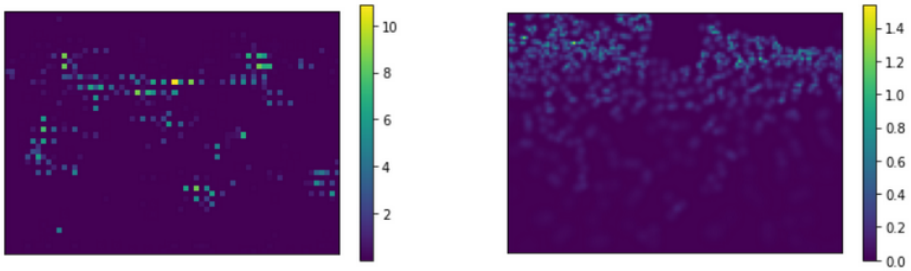
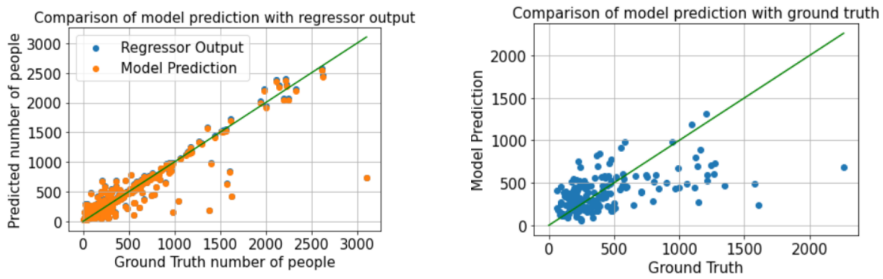


Figure 40: **Left:** Generated density Map **Right:** Ground Truth density map. For the final RMSProp model



(a) Crowd count on the training dataset

(b) Crowd count on the test dataset

Figure 41: Crowd count statistics on overtrained Mobilenet frontend model

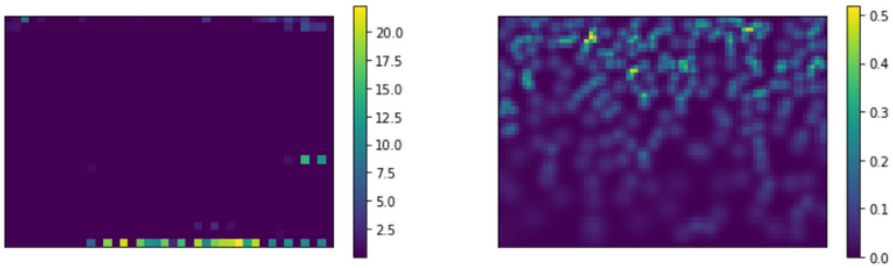


Figure 42: **Left:** Generated density Map **Right:** Ground Truth density map. For overtrained Mobilenet frontend model

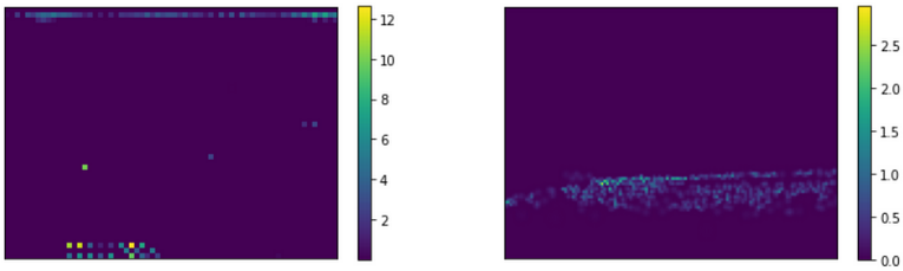


Figure 43: **Left:** Generated density Map **Right:** Ground Truth density map. For overtrained Mobilenet frontend model