Deep Learning For Computational Data Science

Crowd Counting

Under guidance of

Dr. Deepak Mishra (HOD, Dept. of Avionics)

by

Shashank Tomar (SC17B045)

Kiran L (SC17B150)

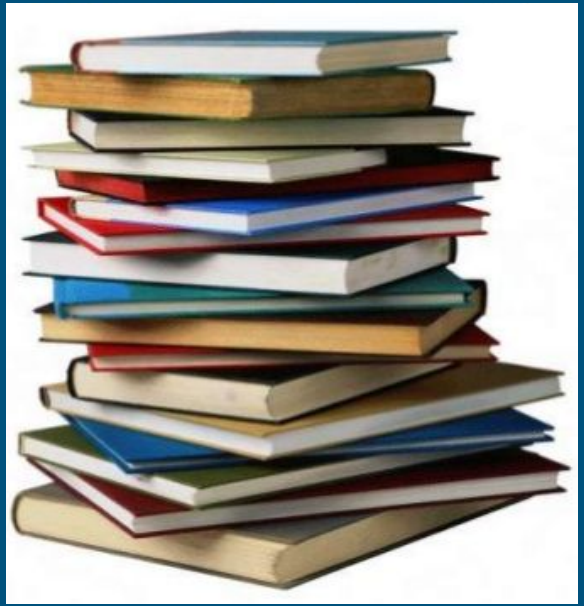
Problem Statement

- Crowd counting - gained more attention from multimedia and computer vision - finds application in **crowd control**, **traffic monitoring** and **public safety**.

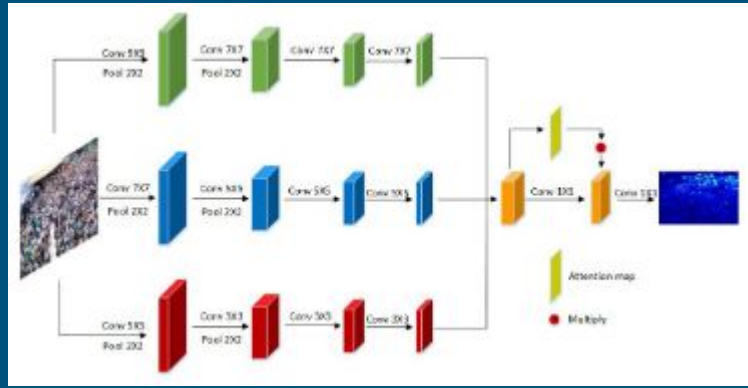
- **Challenges** - occlusions, complex backgrounds, non-uniform distributions and variations in scale.

- Many algorithms proposed to address these challenges and increase the crowd counting accuracy.

Literature Review

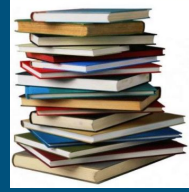


1. **Youmei Zhang et al., 2018**
    a. Multi-column CNN (MCNN) - to address the scale-variation problem
    b. Used several CNN branches with different receptive fields to extract multi-scale features
    c. Incorporate attention mechanism - to guide the network to focus on head locations



2. **Haipeng Xiong et al., 2019**
    a. Spatial Divide-and-Conquer Network (S-DCNet)
    b. Idea: crowd count in a region = sum of crowd count in sub-regions
    c. S-DCNet - state-of-the-art performance major crowd, vehicle and plants counting datasets

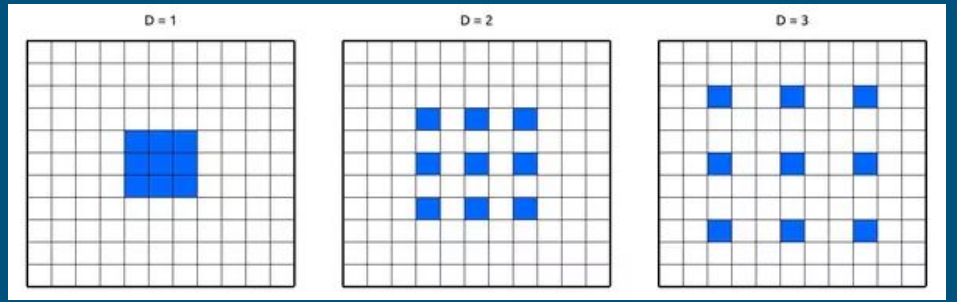
3. Yuhong Li et al., 2018



a. Congested Scene Recognition Network - CSRNet - Aim to provide data-driven and DL method for
   i. accurate count estimation
   ii. high-quality density map generation

b. Two major components:
   i. Front-end CNN - 2D feature extraction
   ii. Back-end Dilated CNN - dilated kernels - larger receptive fields.

c. Four datasets -
   i. ShanghaiTech,
   ii. UCF CC 50,
   iii. WorldEXPO'10,
   iv. UCSD dataset



Data Description

ShanghaiTech crowd counting dataset

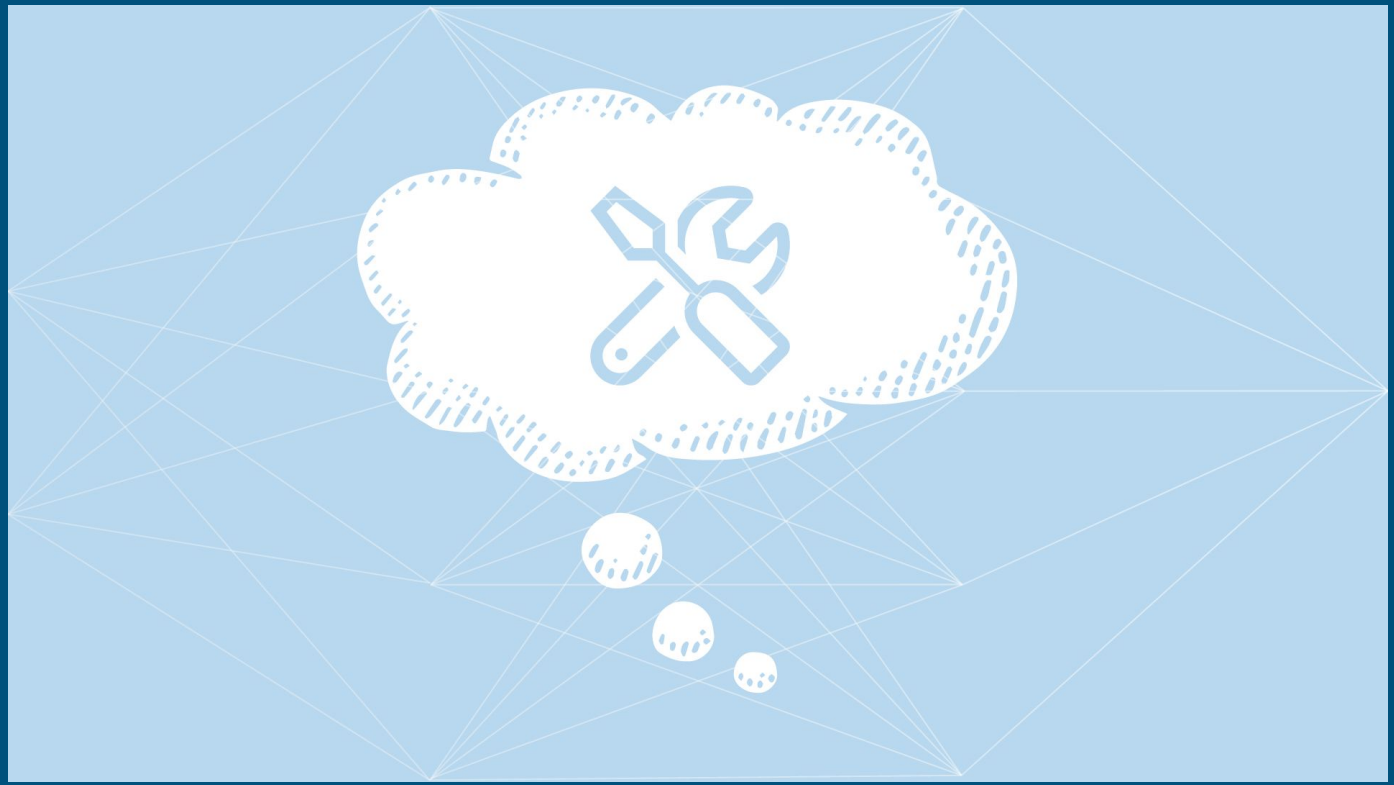
- 198 annotated images - 330,165 persons
- **Part-A** - highly congested scenes - 300 training images and 182 testing images
- **Part-B** - relatively sparse scenes - 400 training images and 316 testing images.
- **Advantage**: Common dataset for crowd-counting model comparison
- **Disadvantage**: Limited number of images - requires data augmentation

ShanghaiTech crowd counting dataset

Part - A
Sample Images

Part - B
Sample Images

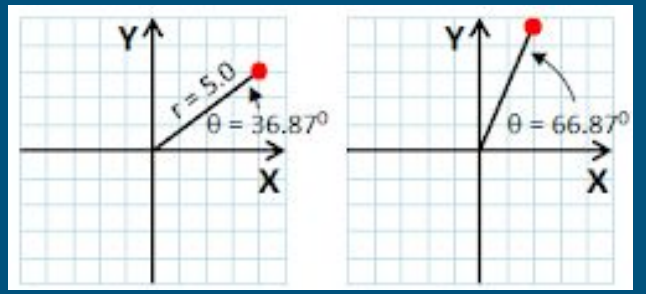
Methodology



Methodology

- Approach - similar to Yuhong Li et al., 2018 (CSRNet) - (Front-end - Back-end)
- Fully Convolutional Neural Network - no constraint on input image dimensions (except a lower bound) - includes diluted kernels
- Pipeline:
  - Model density map (output feature map) generation
  - Crowd-count estimation
- First estimate of crowd-count - sum of pixel values of model density map (not very accurate unless sufficiently trained) - **MLP -** to overcome model shortcomings

Methodology



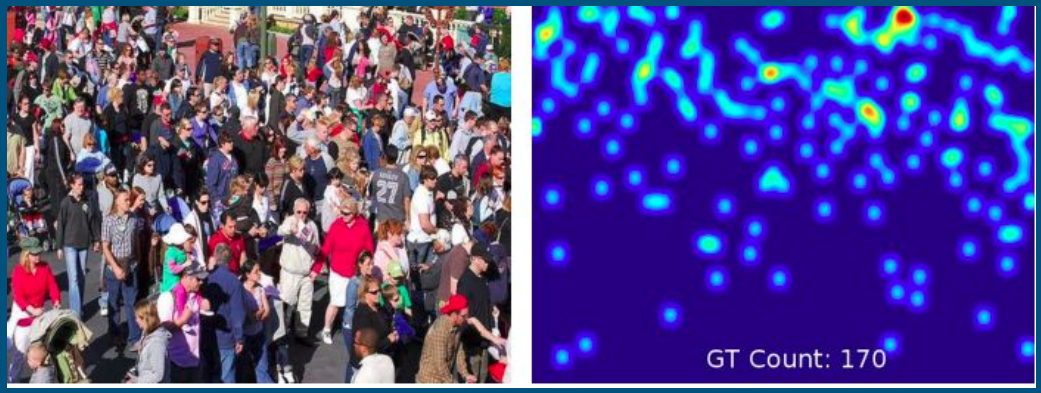
- MLP - not final solution - only useful when model estimate of crowd-count is
  - Monotonous with ground-truth
  - Linear - only need to learn to rotate the line equation (change of slope)

- Basis for using MLP - Universal Approximation Theorem

- Loss functions explored:
  - Mean Square Error (MSE) Loss
  - MSE + L2 Regularization
  - MSE + Range-matching constraint

  (Note: Output feature map only “interpreted” as density map)

Methodology

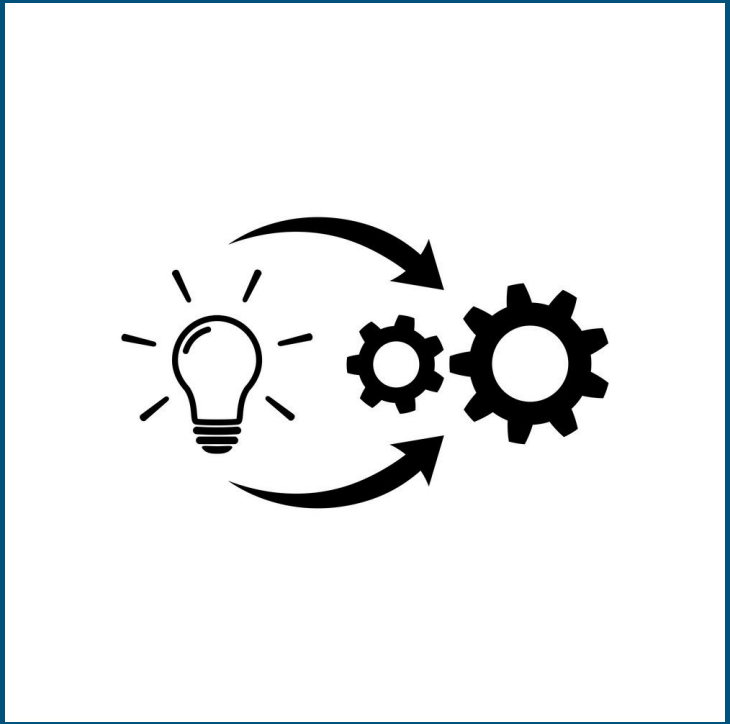


- **Density Map Generation:**
  - Ground-truth Density map generation

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

  - Analogy - Second half of Hogbom CLEAN Algorithm
  - Use 2D-Gaussian Kernel to convolve over the delta-function like array - containing information of position of each person (as annotated)
  - Width (standard deviation) of Gaussian kernel - based on mean distance to k-nearest neighbours

Implementation



Implementation

- Programming language - Python
- Platform - Google Colaboratory (with GPU hardware accelerator)
- Training - using Pytorch (on Nvidia GPU)

- Constraint on batch-size - 1 (why? -> no fixed input image size)
- All models trained for at least 100 epochs - MLP trained for 1000 epochs
- Activation function: ReLU (also for MLP)

Implementation

- Train-validation split - 4:1
- Data augmentation - image mirroring (2x number of training images)
- Optimizers explored:
  - Stochastic Gradient Descent (with momentum)
  - Adam
  - RMS Prop
- Performance Metrics:
  - Mean Square Error
  - Mean Average Error

Analysis

Initial Results with CSRNet

CSRNet as proposed by Li and Zhang et. al. implemented on Pytorch

Optimizer	Hyper-parameter	Comments
SGD+Momentum	LR=1e-2	No learning, MAE=429.24
SGD+Momentum	LR=1e-4	Non-zero head count, MAE=425.32
SGD+Momentum	LR=1e-6	Improvement, better dmaps, long training MAE=302.58
Adam	Default	No learning
SGD+Momentum	Frozen back-end layers	Faster training, MAE=301.16

Takes a lot of time to train - also prone to overfitting due to large VGG-16 backend.
Therefore we look for models that can be trained and deployed faster

Moving Away from CSRNet

CSRNet uses a VGG-16 network as the parent architecture, increasing the training time and computational resources required

We aim to find a network that can train much faster and requires less computational resources to implement on the fly.

Decision to use already existing networks as:

1) Pre-trained weights are available. No need to train from scratch
2) Well tested and literature available
3) Can work with small dataset available to us

Parent Architecture selection:

01 **SqueezeNet**

- Very Fast and inexpensive Training
- Sensitive to training data and number of epochs, overfits easily
- Negative Pixel values and poor density maps

02 **DenseNet**

- Extremely slow training, even with three Fire Modules
- Not too sensitive to training data, positive pixel values
- Prohibitive run time affects hyperparameter search

03 **VGG-16 with Batch Normalisation**

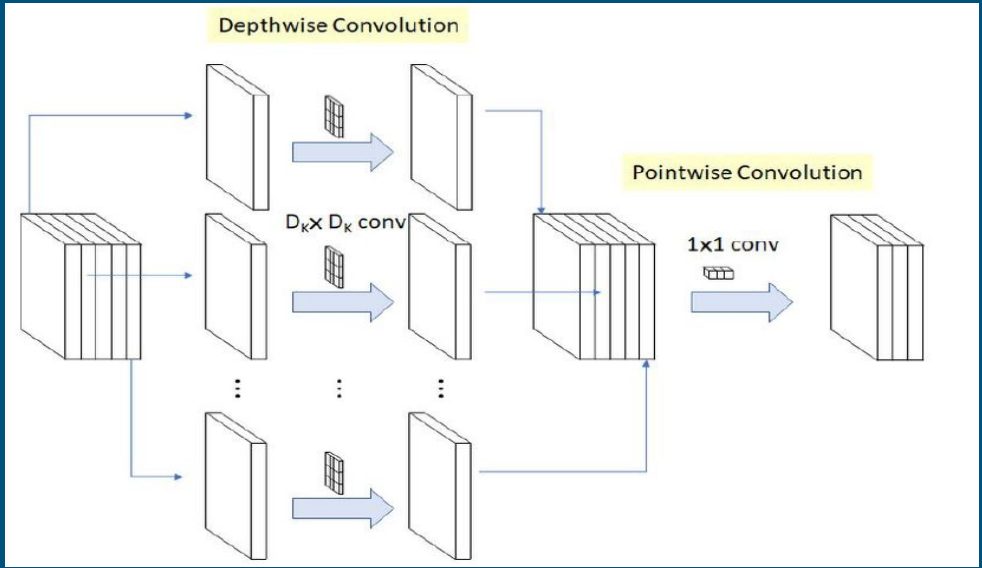
- Large overfitting to training data
- Only slightly faster than CSRNet
- Poor density map and crowd count generation

In all of these parent architectures, hyperparameter variation was carried out by varying number of layers and loss function hyperparameters

MobileNet V2

First proposed By Google in 2017 for high efficiency with minimum resources

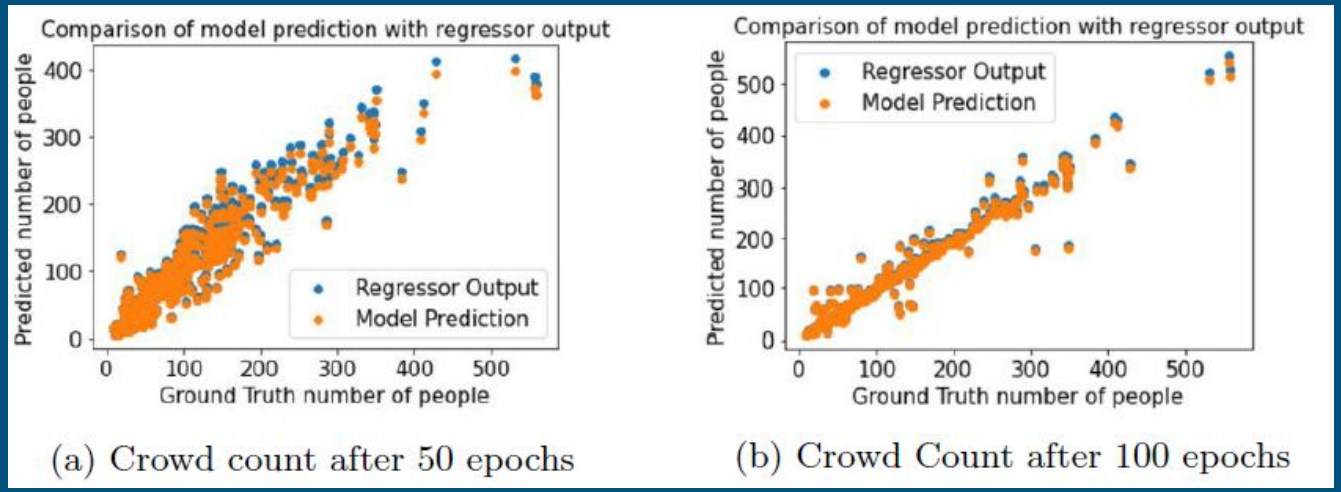
Use of depth-wise separable convolutions (depthwise+pointwise convolution)

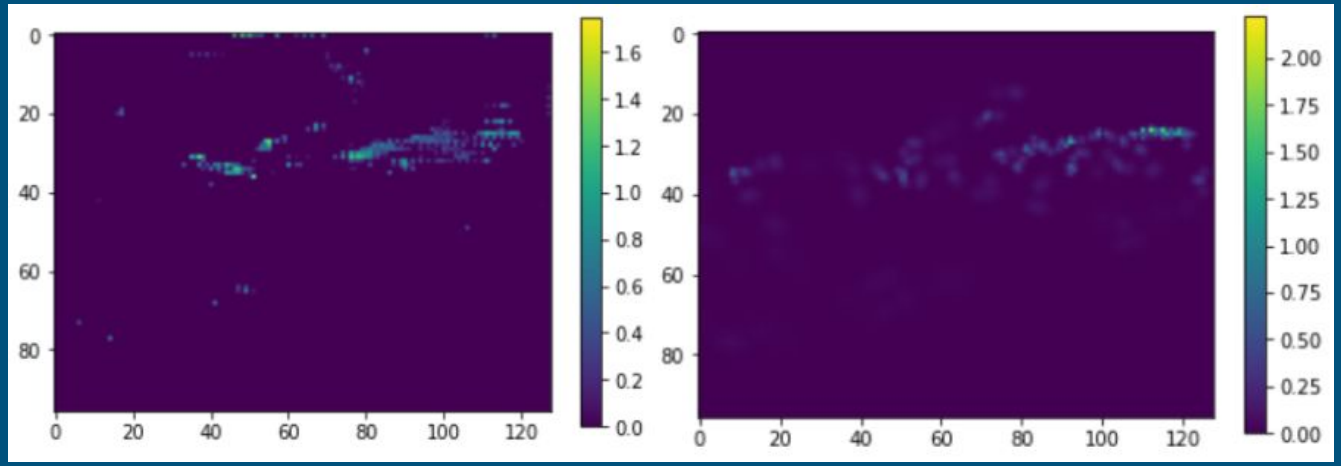


Reduces number of computations by a factor by:

1/N+1/Dk^2

Sample Results from MobilNet without hyper-parameter tuning





Hyper-parameter Tuning

Hyper-parameter Tuning of MobileNet

Hyperparameter combination efficacy measured using two performance metrics:

- Degree of Linearity
- Accuracy of Generated density maps

The following hyperparameters were varied:

- Number of Frontend layers
- Training dataset
- Loss Function and learning rate

Number of Frontend Layers

- Having frontend layers in the range 3-7: Poor density map generation, okayish crowd count - suspected bias for 3-5 layers.
- Having high number of frontend layers (10-13): Poor density map generation, really good fit on training data but high deviation on test data. Falters at high counts.
- 8-9 frontend layers found to be optimal - good density map generation (captures global resolution well) and better degree of linearity on the test data (except at really high counts)

Effect of Training Dataset

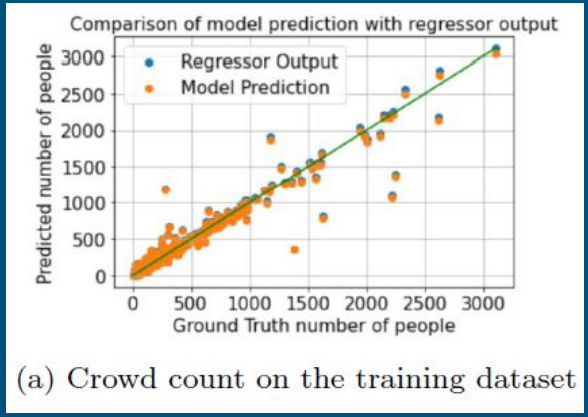
- Need model that on training on A+B gives good results on A and B test samples
- Easier for models to train and predict on dataset A compared to dataset B
- This is because:
  - ShanghaiTech Dataset A - high crowd density, distributed uniformly in image
  - ShanghaiTech Dataset B - localised crowd density, difficult for kernel to represent (need larger kernel size for an entire body than a head)
- Ideal model should perform well on both datasets

Effect of Loss function and learning rate

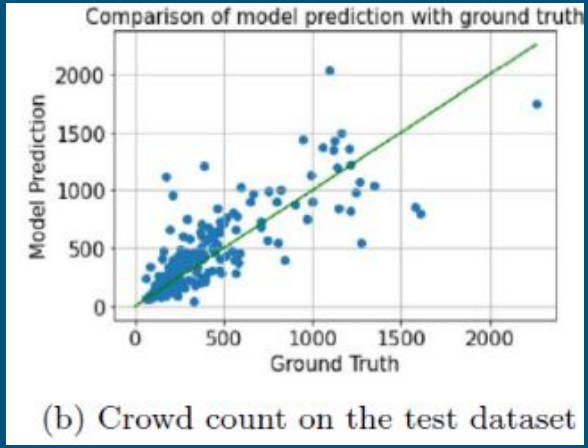
- Extensive hyper-parameter search carried out with 6 learning rates for each of the three optimizers (Adam, SGD and RMSProp).
- SGD + Momentum gave outright wrong results and very sensitive to hyperparameter combination
- RMSProp- LR=0.0005 showed good results on B but poor results on high crowd densities (A and A+B)
- Adam optimizer with learning rate of 0.0006 and 9 layers of mobilnet frontend showed good crowd count results on test set as well as better density map generation

Results

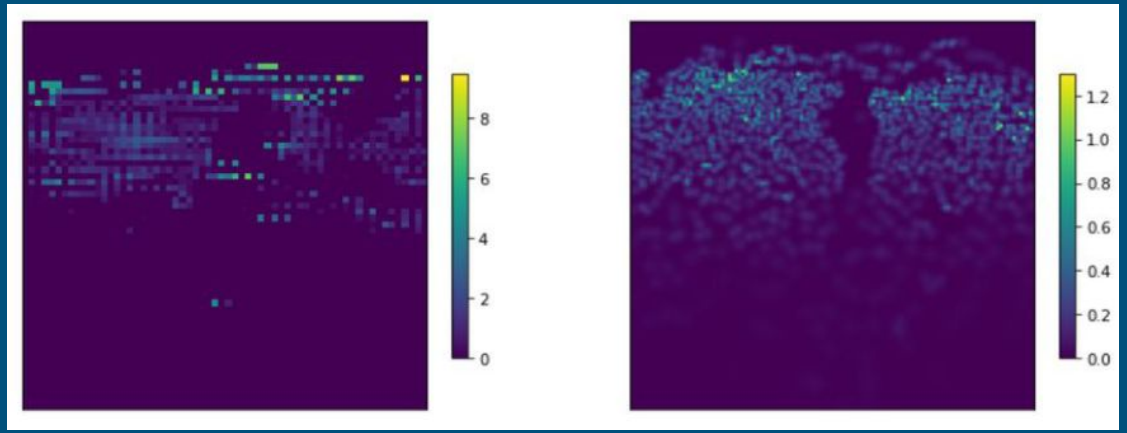
Results

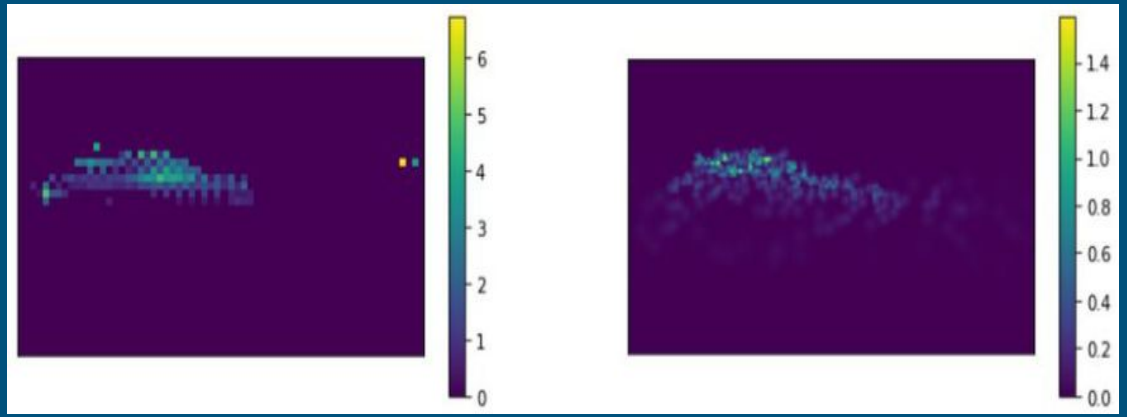


(a) Crowd count on the training dataset



(b) Crowd count on the test dataset





Data Augmentation: Mirroring

- Laborious to go for other geometric image augmentation techniques like cropping, random erasing etc. due to nature of our problem and overfitting concerns
- Excessive care needed for boundary cases

- The best models found out till now were tested with mirrored dataset: A, B and A+B.

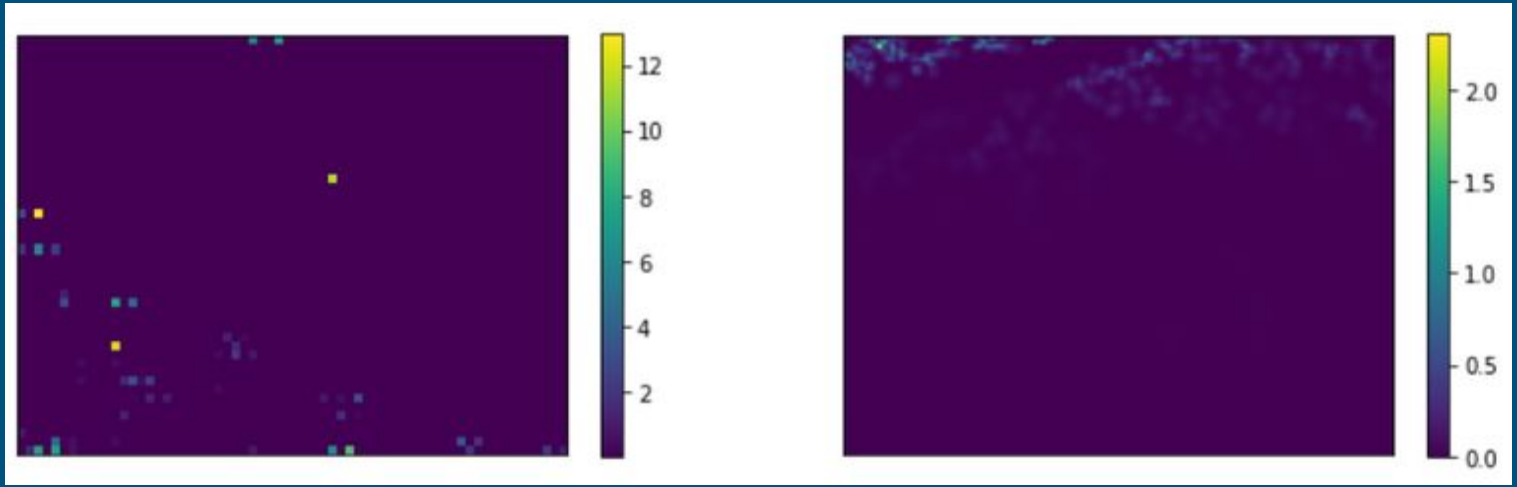
- Models worked well only on B dataset

- B dataset has localised low density images, mirrored images are not redundant.

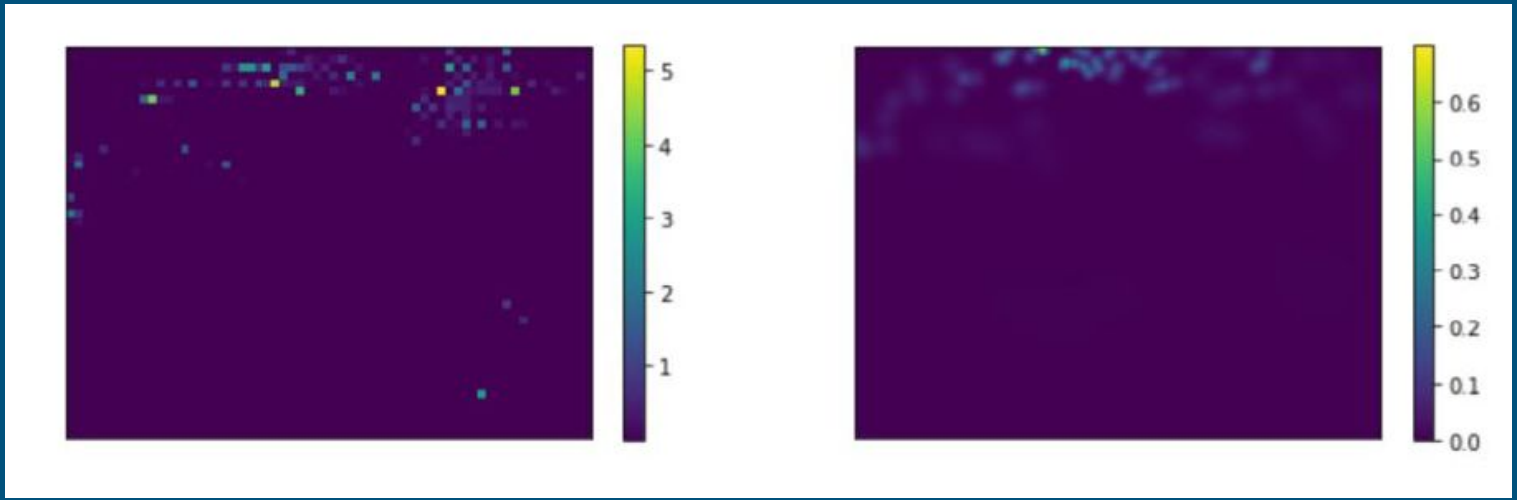
- Hence go for A+B+B mirrored dataset

Results

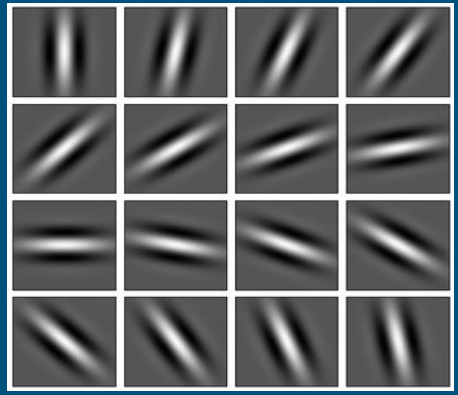
A + B + A mirrored + B mirrored



A + B + B mirrored



Gabor Filter



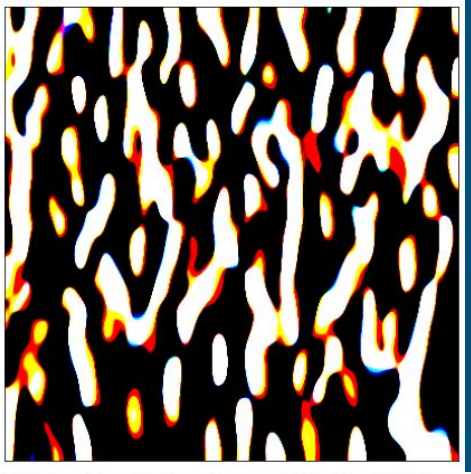
- Used for detection of edges and texture changes
- In OpenCV: cv2.getGaborKernel(ksize, sigma, theta, lambda, gamma, psi, ktype)

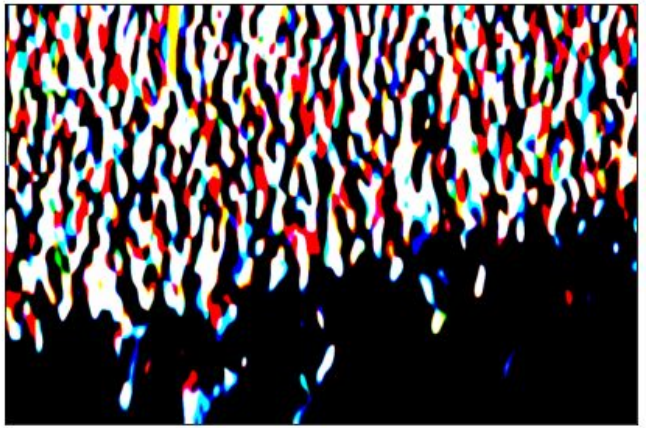
$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

Gabor Filter $(\text{kernel size} = 7, \sigma = 2, \theta = 0, \lambda = 5, \gamma = 10, \phi = 0)$

Gabor Filter

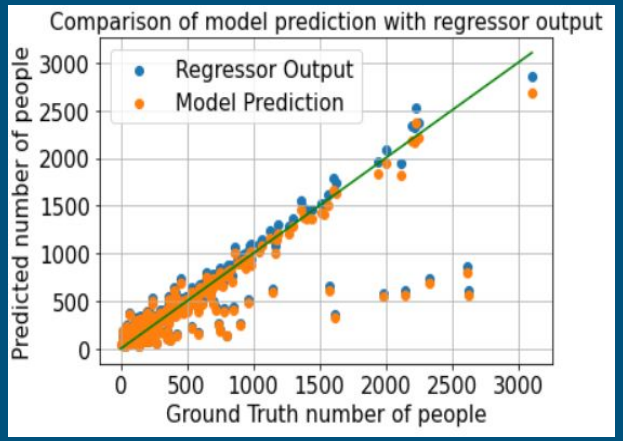
$(\text{kernel size} = 50,\ \sigma = 10,\ \theta = 0,\ \lambda = 30,\ \gamma = 1,\ \phi = 0)$

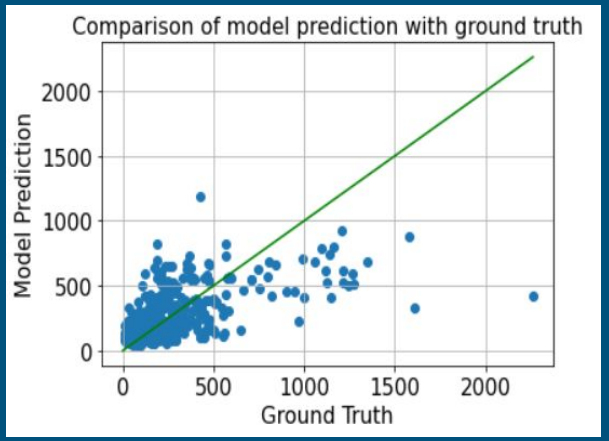


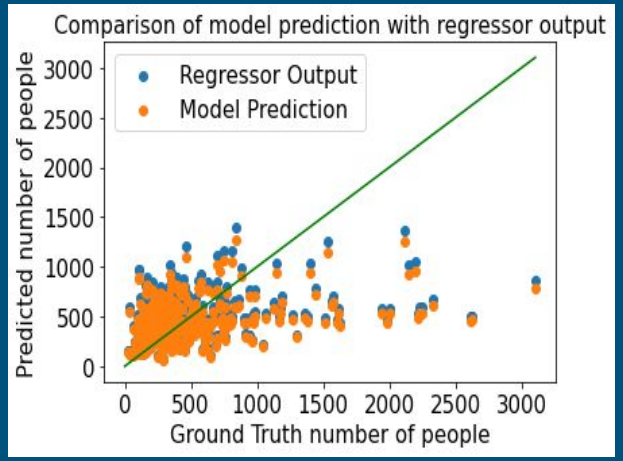


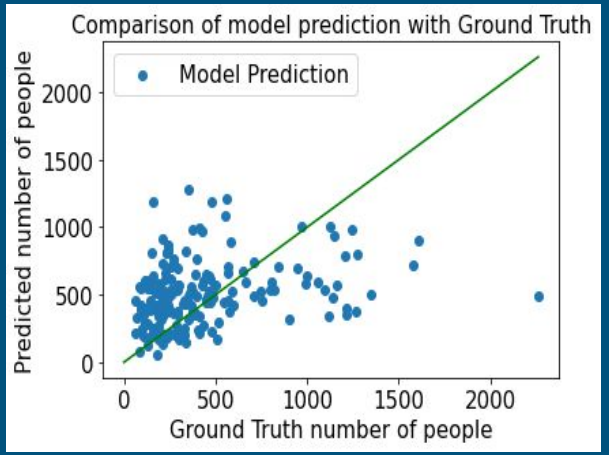
Effect of Gabor filter on mirrored datasets

Smaller kernel (size 7 gives better results)









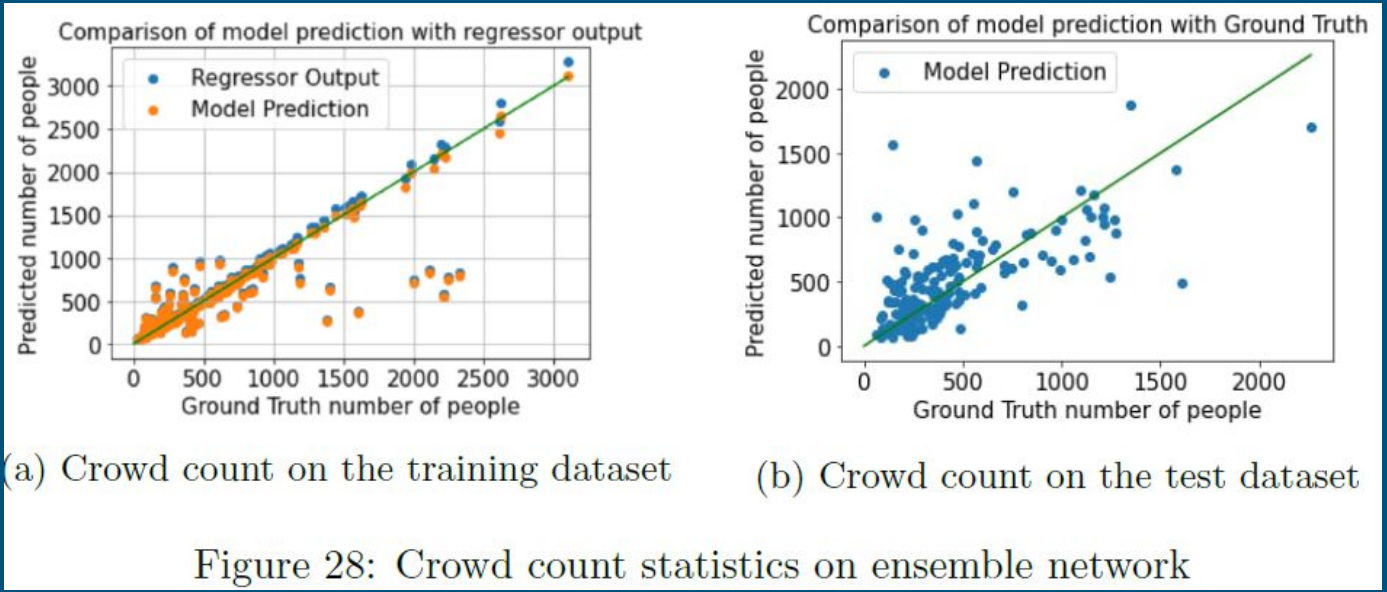
Ensemble Network

- Combining the prediction from two or more models in a way that obtained results are better than either network.

- Can be used only for crowd count in our model not for density map generation due to unequal density map sizes from each network

- Reasons for not enforcing equal sizes on the output maps:
  - Using pre-defined architectures with pre-trained weights
  - High variability of image size and aspect ratios
  - Difficulty in stretching images than compressing

Results from Ensemble Network



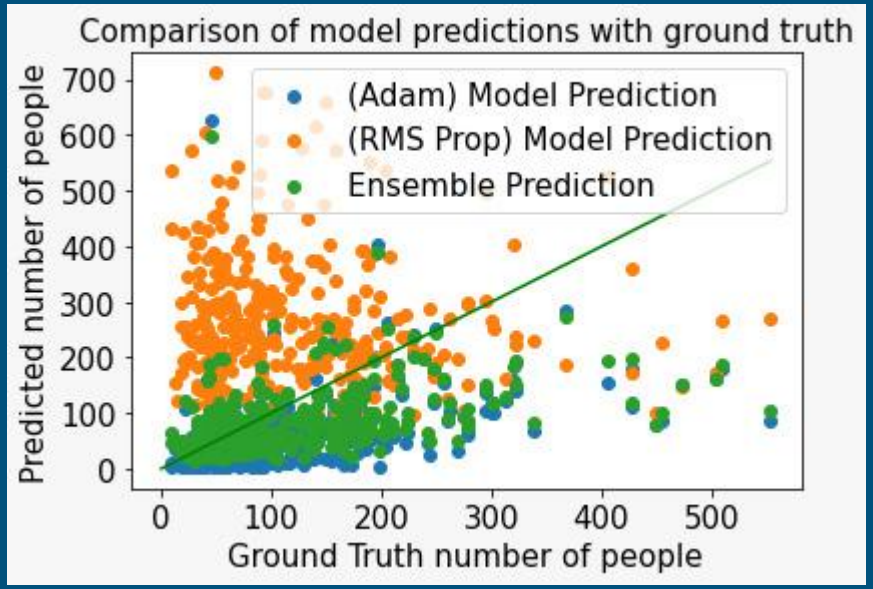
(a) Crowd count on the training dataset

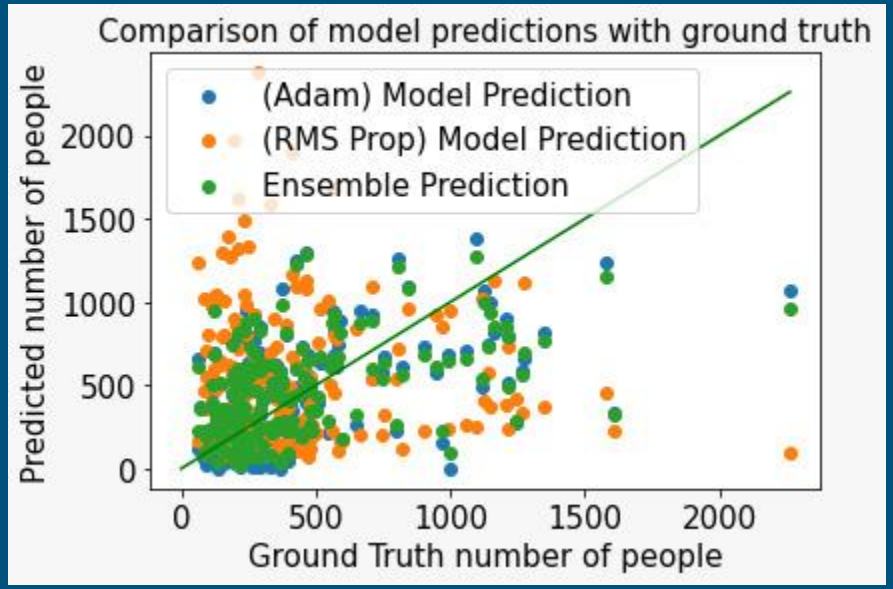
(b) Crowd count on the test dataset

Figure 28: Crowd count statistics on ensemble network

85:15 MobileNet and DenseNet network gave optimal results

Visualising Ensemble networks





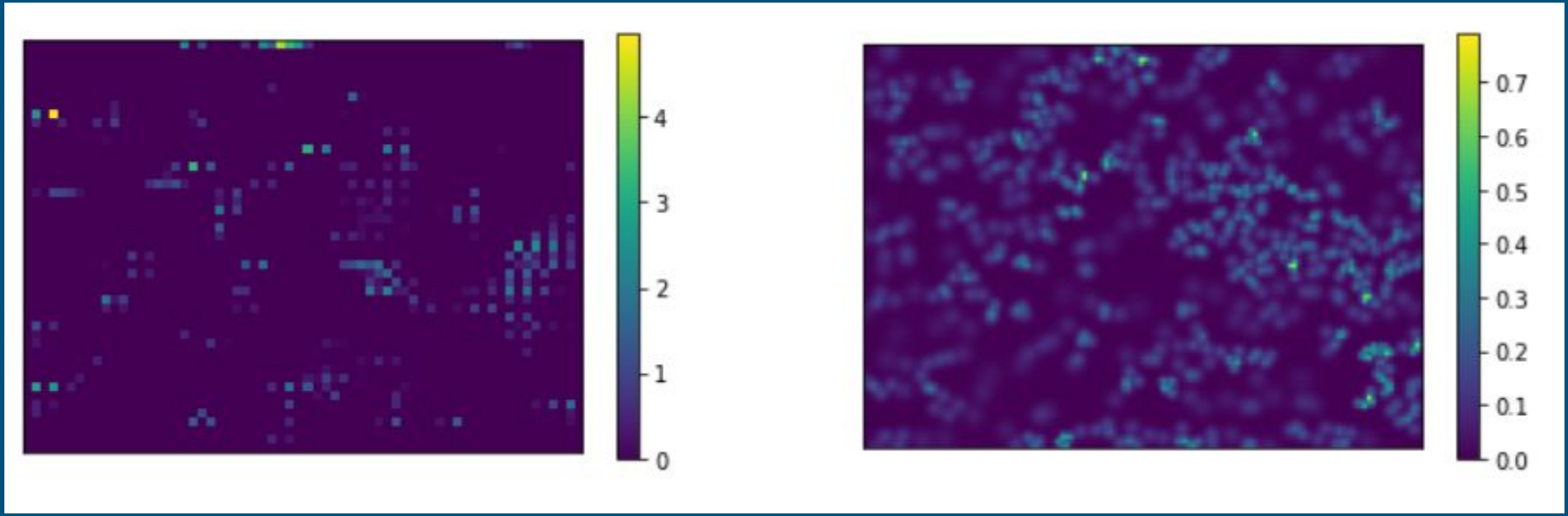
Ensemble of two networks gives better result than either of them with proper choice of weights

Proposing a New Loss Function

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

- Representation of cluster of pixel values in the original map by a single pixel of larger value in the learnt map.
- Need to enforce the learnt pixel values to be in a similar range - similar max-min values
- Enforced using the above loss function, the weights A and B highlight the relative importance of density map accuracy to crowd count accuracy
- Implemented with the optimal models obtained via previous searches

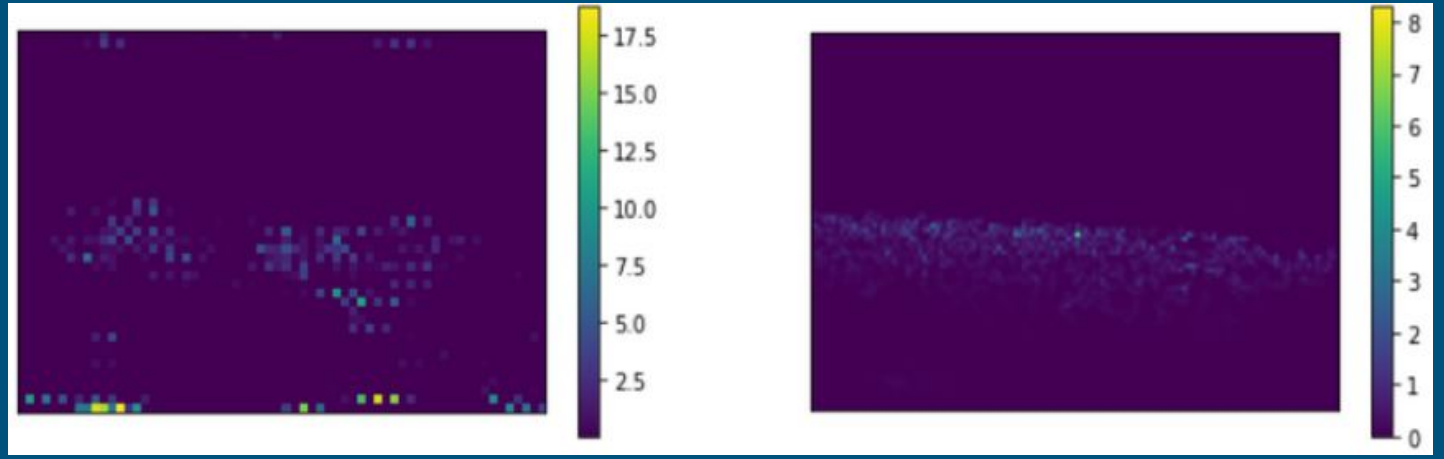
Explaining pixel ranges

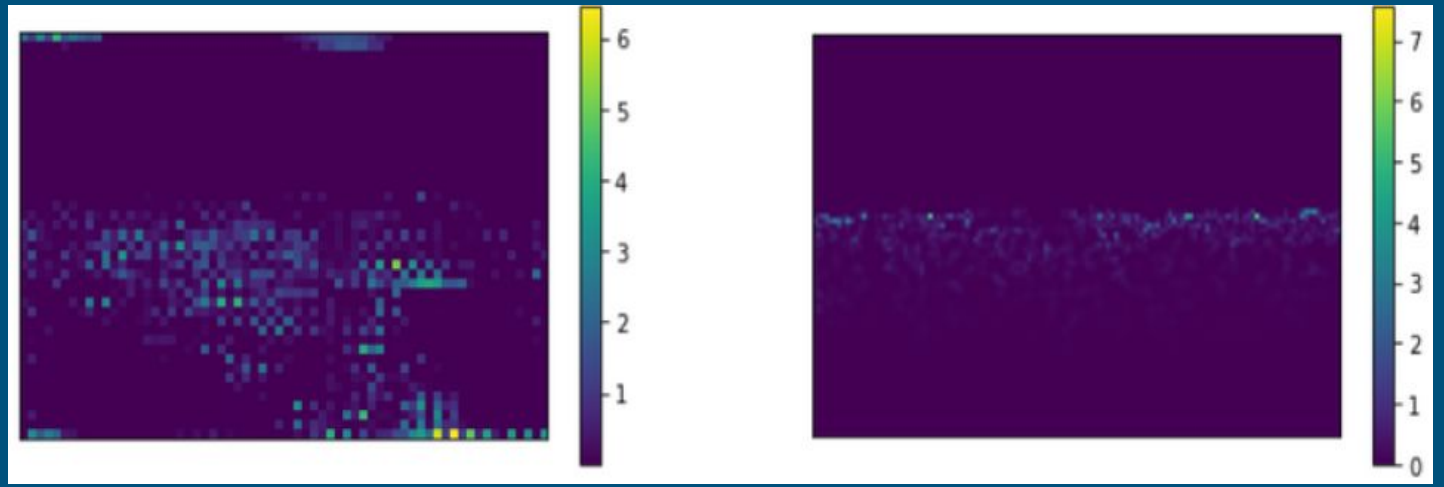


Results

Results with optimal model with old loss function (A=B=0)

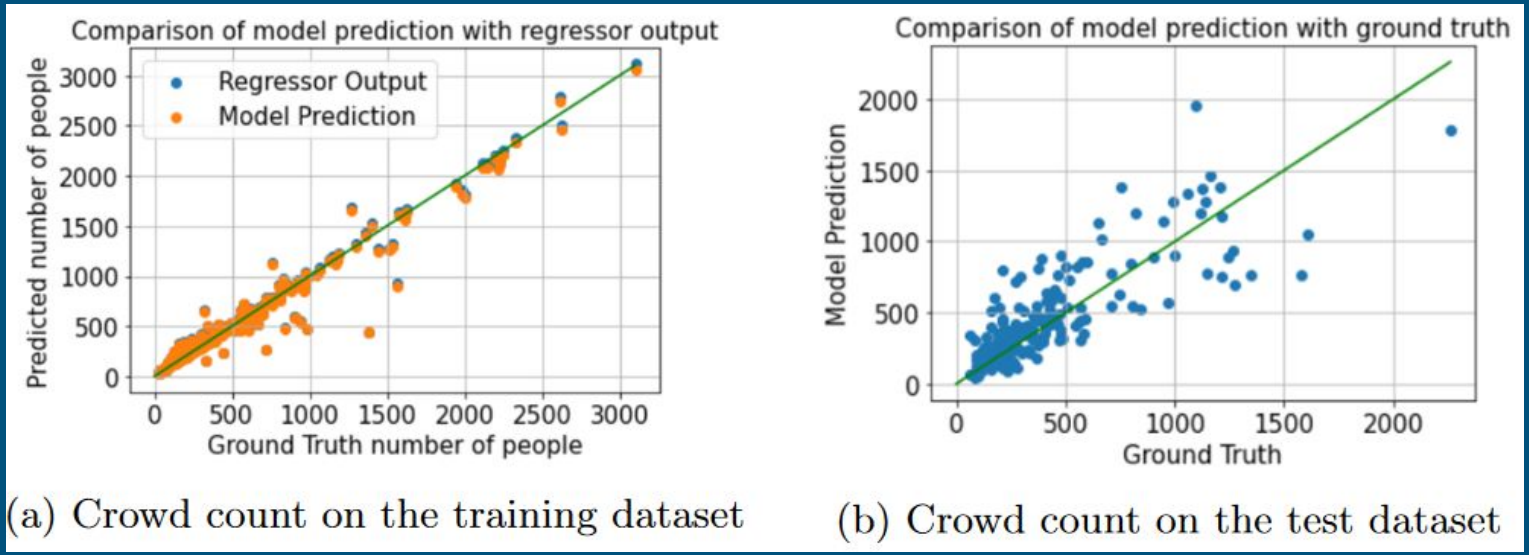
Results with optimal model and high weights (A=B=55)



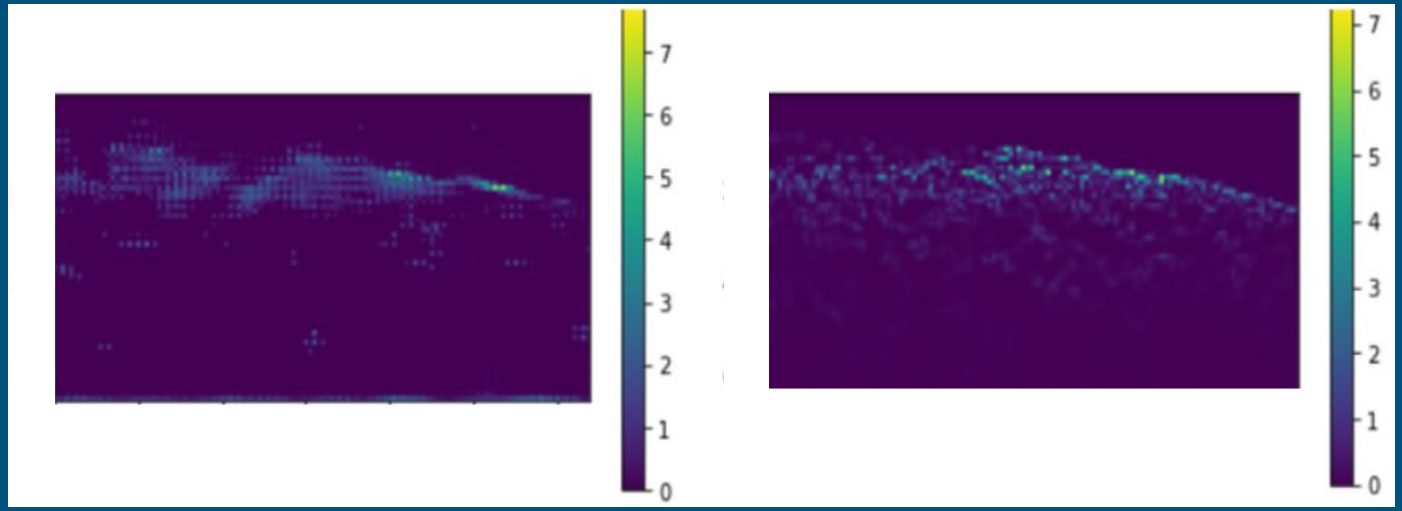


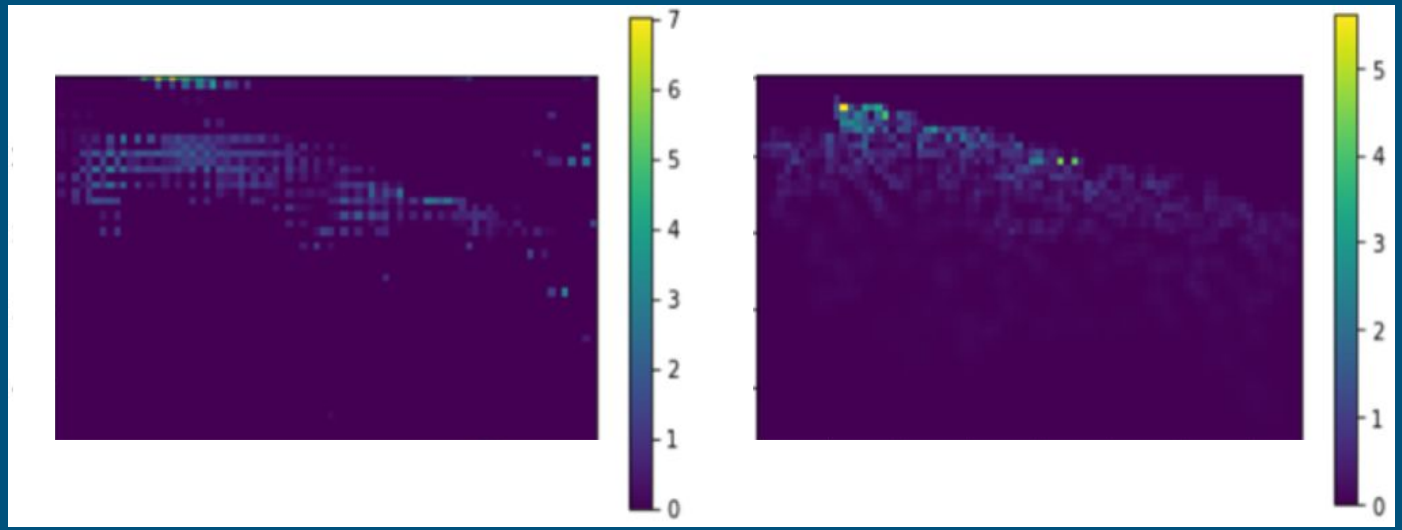
Final model results

9 layers of MobileNet as backend, trained on A+B dataset, Adam optimizer lr=0.0006, Modified loss function with A=B=32



Density Maps from the Final Model





Conclusion

Conclusions

- Crowd counting problem first tried with CSRNet model was investigated
- Existing like MobileNet, DenseNet, VGGNet and SqueezeNet were trained by varying hyperparameters and combination that maximised the performance metrics was found
- Data augmentation was carried out
- The model was trained with two different datasets from the ShanghaiTech
- Ensemble approach to training
- 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 Scope

1) Using a bigger and more general dataset

2) The network can be configured to give output density maps of same size, either by changing the input data itself or configuring the network to do it.

3) Loss function can be modified to compute the loss by comparing individual pixel values between the ground truth and generated image.

4) More accurate metrics for density map characterisation

5) Employ data augmentation other than the geometric transformations used in this case to handle more realistic cases encountered in practical application (blurring, fog, rain, night time images)

References

1. 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.

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

3. 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.

4. 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.


THANK YOU

Questions?