Project Report of
Counting of Crowd Size with the help of Camera

By

Atanu Guin (MT19AI002)

Sandeep Verma (MT19AI003)

Shobhit Sharma (MT19AI010)

Course: Computer Vision

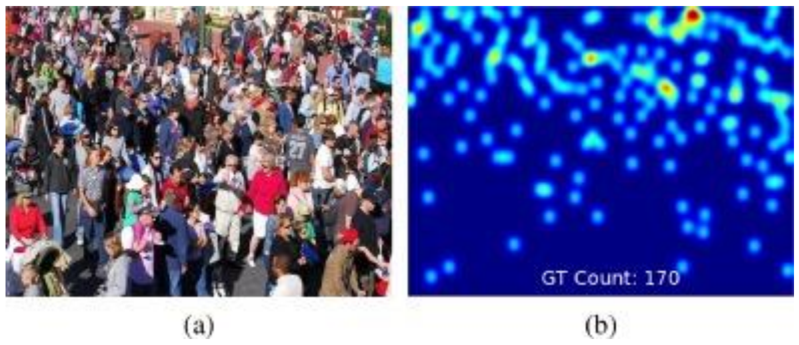
Instructor: Dr. Mayank Vatsa

Motivation:

Counting objects in images is one of the important applications in computer vision. For *video surveillance, anomaly warning, social and political gathering of people and also in biology, physics counting* in extremely dense crowds or objects is an important task. As COVID-19 makes its way across the globe, governments across the world are enforcing lockdown to restrict public gathering on the streets. Local Law enforcement officers are checking round the clock for people's safety. In this current situation when surveillance is not an easy task over a large area, we are trying to build a model based on Computer Vision techniques, on counting how many people are in a video shot.

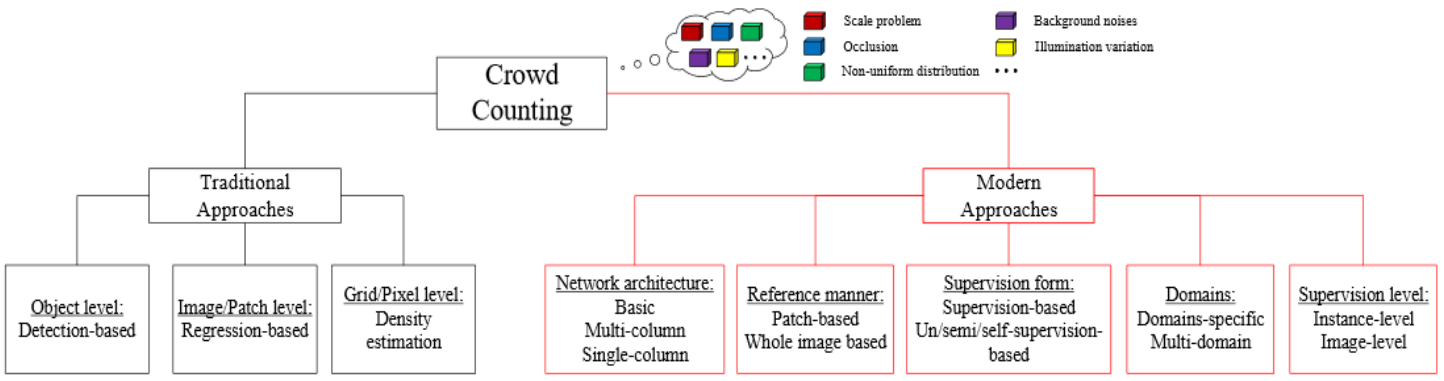
Crowd Images

Problem Specification: The main idea is to count objects indirectly by estimating a density map. The first step is to prepare training samples, so that for every image there is a corresponding density map. A density map is obtained by *applying a convolution with a Gaussian kernel and normalized so that integrating it gives the number of objects.*



Crowd with Density Map

Background work: Various crowd counting methods have been proposed to tackle the problem. The traditional approaches follows 1. *Detection based, 2. Regression method 3. Density estimation based* approaches. *Detection based approaches* first detect the object then do the counting of the bounding boxes. *Regression methods* uses low level features and regression modelling. *Density estimation based* methods takes into account spatial information and estimate density by integrating over an image for counting. Recently *CNN based* approaches become more popular for *scale/context aware, multi-tasking and end-to-end solutions.*

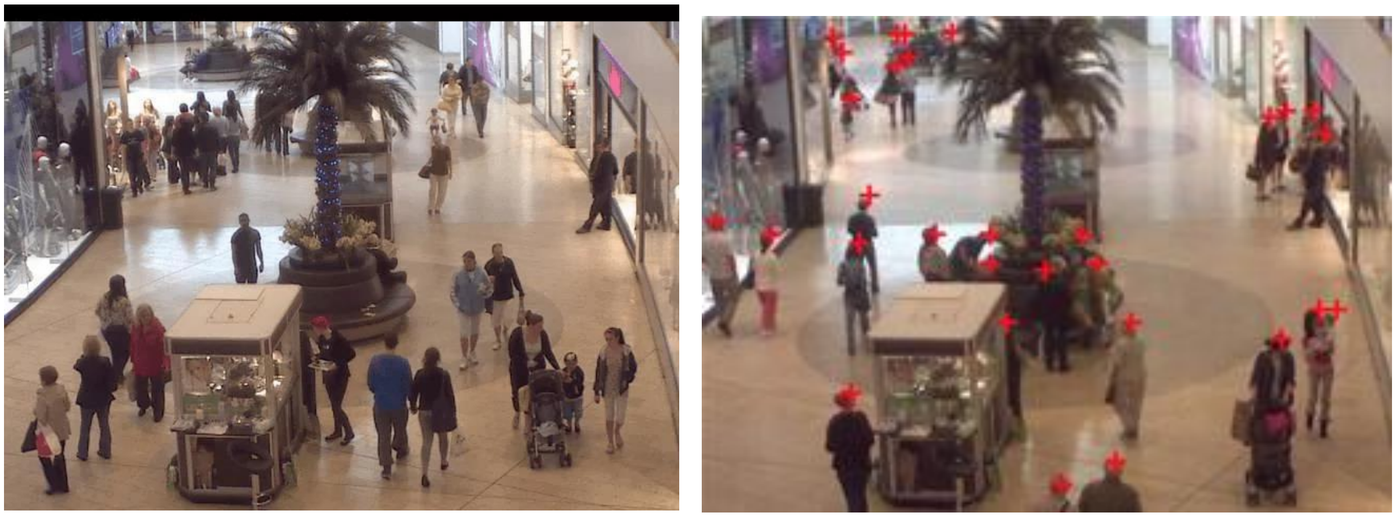

Scale problem
Occlusion
Non-uniform distribution
Background noises
Illumination variation
Crowd Counting
Traditional Approaches
Modern Approaches
Object level: Detection-based
Image/Patch level: Regression-based
Grid/Pixel level: Density estimation
Network architecture: Basic Multi-column Single-column
Reference manner: Patch-based Whole image based
Supervision form: Supervision-based Un/semi/self-supervision-based
Domains: Domains-specific Multi-domain
Supervision level: Instance-level Image-level

Approaches to crowd counting problem

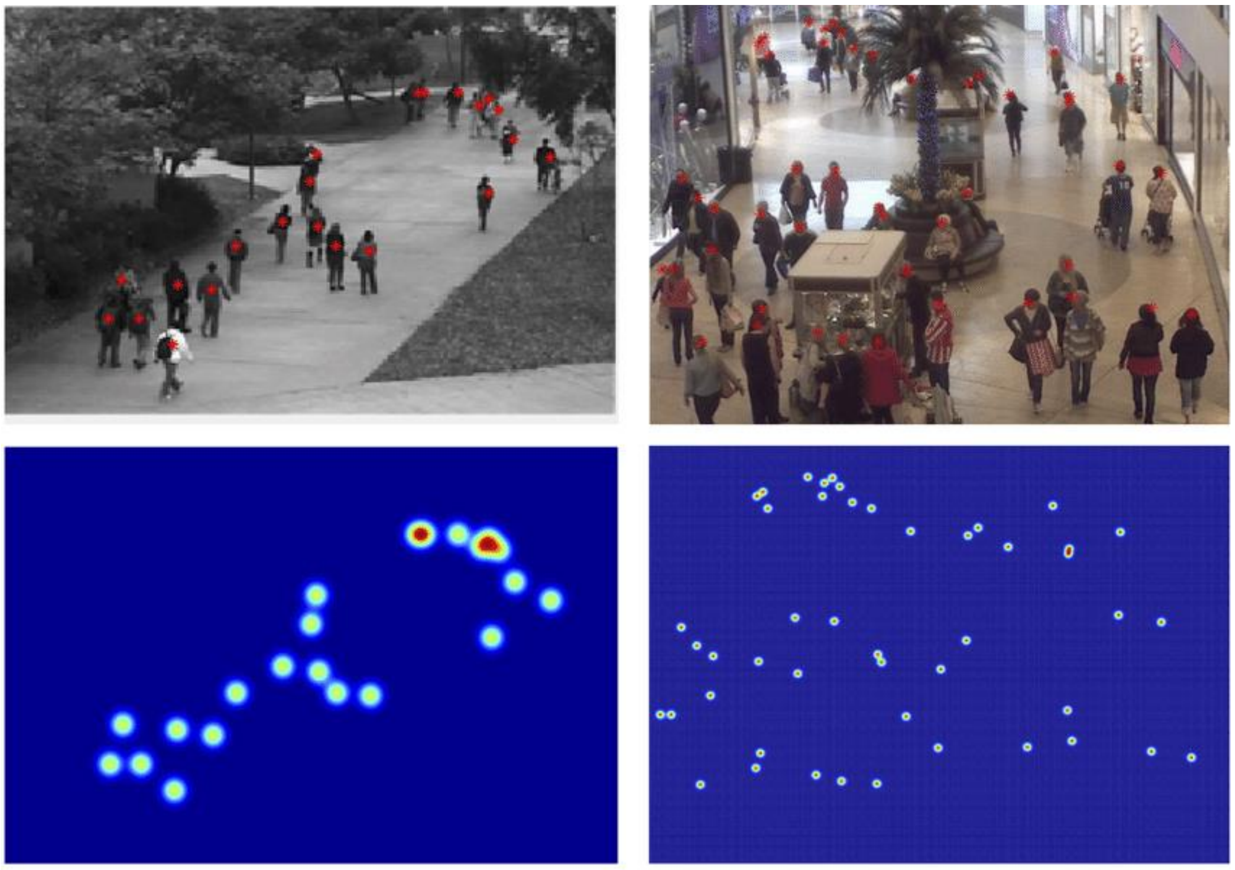
Dataset: We have used mainly 2 dataset in our code. *For training and validation* we have used **UCSD dataset** and *for testing* we have used **Mall dataset**. The UCSD Anomaly Detection Dataset was acquired with a stationary camera mounted at an elevation, overlooking pedestrian walkways. The crowd density in the walkways was variable, ranging from sparse to very crowded. The data was split into 2 subsets, each corresponding to a different scene. The video footage recorded from each scene was split into various clips of around 200 frames. Total number of images is 2000.

UCSD Image **Mall Image**

The mall dataset was collected from a publicly accessible webcam for crowd counting and profiling research. It has over 60,000 pedestrians were labelled in 2000 video frames. We annotated the data exhaustively by labelling the head position of every pedestrian in all frames. Video length 2000 frames and Frame size is 640x480. For testing, we have selected 20 images from Mall dataset.


Annotated Mall Image

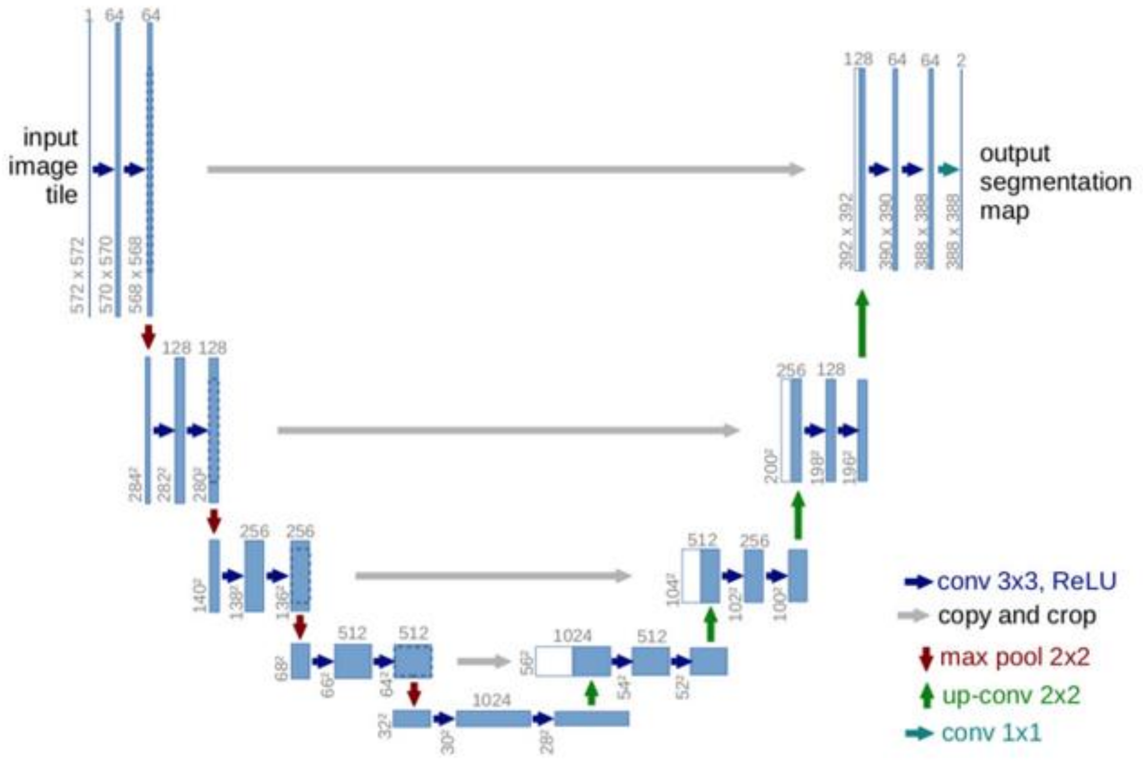
Data processing: We have used the *ground truth generating* matlab file from the internet for both the datasets. These files generates annotation in the form of (x,y) co-ordinates on the images. The annotated points are labelled as 100 before convolution. Then it uses *Gaussian kernel of sigma=1* for finding *the density map. This density map is normalized,* so the annotated points would only become prominent in the images. If we now count the pixel values of the image, we will get the number of people in the images. Now this image data with the annotation are loaded into *hdf5 file format.*


Ground truth Density Map of the Images

Model Architecture:

We have used the *UNet model* for our problem. UNet was originally designed for *segmentation of bio-medical images*. In bio-medical images, not only we have to find if there is a disease but we do have to localize the affected area. In our crowd counting problem, we also *need to localize the crowd in the specific area of the image* for the counting part. For this reason we are using UNet architecture as our preferred model.

The basic UNet model is as follows:



UNet architecture

The UNet model has 2 main parts – 1. Downsampling layers and 2. Up sampling layers.

Downsampling layer is designed with maxpooling and upsampling layers are designed with up-convolution.

Downsampling layer generally consists of -

```
        conv_layer1 -> conv_layer2 -> max_pooling -> dropout(optional)
```

Upsampling layer consists of –

```
        conv_2d_transpose -> concatenate -> conv_layer1 -> conv_layer2
```

Transposed convolution is an upsampling technic that expands the size of images. After the transposed convolution, the image is upsized from 28x28x1024 → 56x56x512, and then, this image is concatenated with the corresponding image from the contracting path and together makes an image of size 56x56x1024.

The last layer is a convolution layer with 1 filter of size 1x1 (notice that there is no dense layer in the whole network). And the rest left is the same for neural network training.

UNet is an architecture which is *robust* *to work on a small dataset* and it is a decent choice for our crowd counting problem.

Experiment and results:

As described we have used UNet model for this project. The project has been implemented in ***Pytorch***. We have used ***NVidia DGX2*** server for training and testing the model.

Model: UNet model contains 3 convolution blocks for down sampling and 3 3 convolution blocks for up sampling. Each convolution block has kernel size (3*3) and single stride. We have used 3 input filters and 64 feature maps for convolution operation. Each convolution block has batch normalization as regularization step and finally the network provides a single tensor output which is the normalized density map for the image. If we sum up all the pixel values in the density map, the normalized density map will give us the counting.

Parameters: Below parameters have been used for the final implementation.

1. learning rate=1e-2
2. epochs=25
3. batch size=32

Learning rate scheduler is used for step size = 5 and coefficient gamma = 0.1.

Loss function: We have used **mean squared error (MSE)** for loss function calculation. The loss is calculated between the predicted output and the ground truth of the density map.

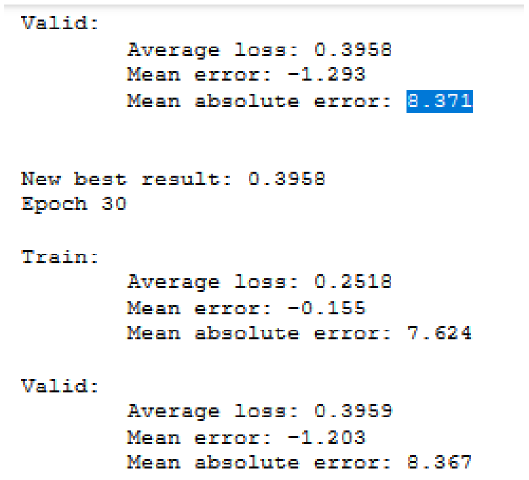
Optimizer: We have used Stochastic Gradient Descent (SGD) optimizer with momentum 0.9.

Metrics: We have used the Mean Error and Mean Absolute Error (MAE) for metrics.

Regularization: There are several regularization techniques that we have used. We have used data augmentation techniques like horizontal flip and vertical flip, learning rate scheduler. We have also used weight decay = 0.9.

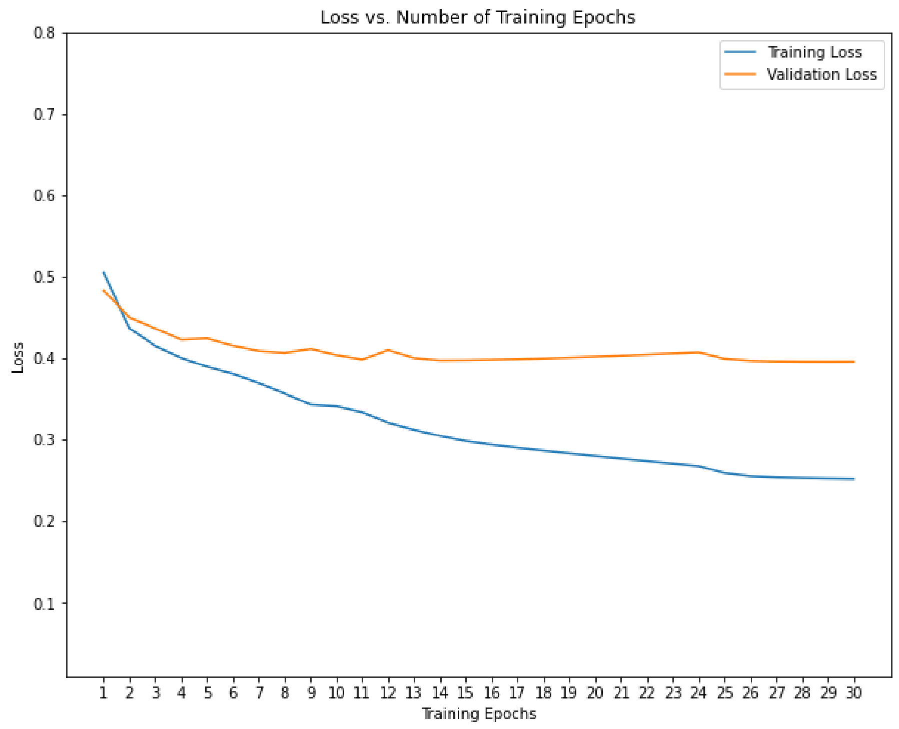
Results – *After successful training, we have achieved best validation loss = 0.3958 with Mean absolute error = 8.371.*

The training results are shown below for last few epochs.

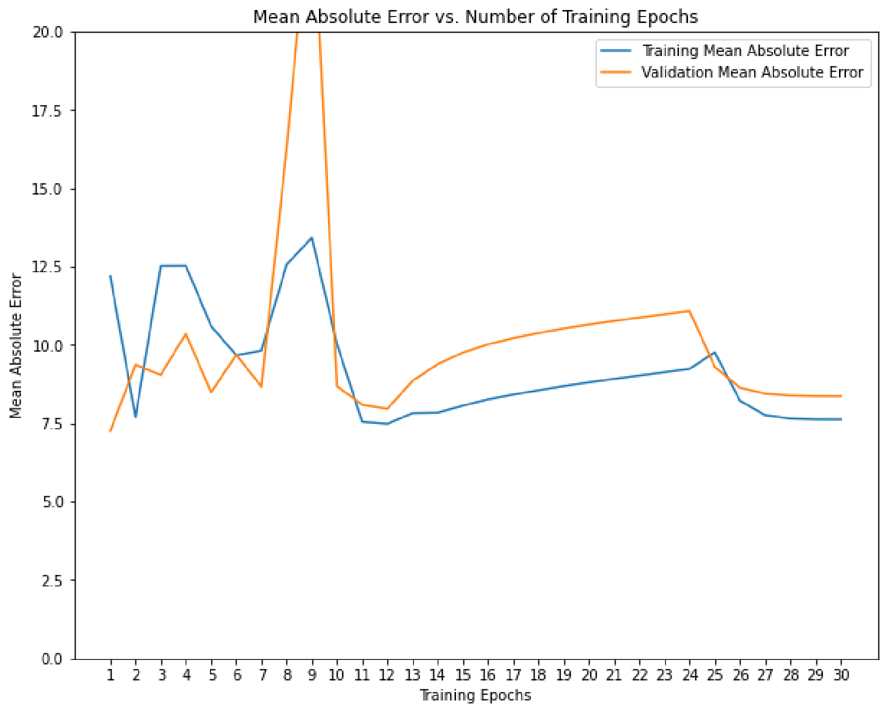


```
Valid:
        Average loss: 0.3958
        Mean error: -1.293
        Mean absolute error: 8.371


New best result: 0.3958
Epoch 30

Train:
        Average loss: 0.2518
        Mean error: -0.155
        Mean absolute error: 7.624

Valid:
        Average loss: 0.3959
        Mean error: -1.203
        Mean absolute error: 8.367
```

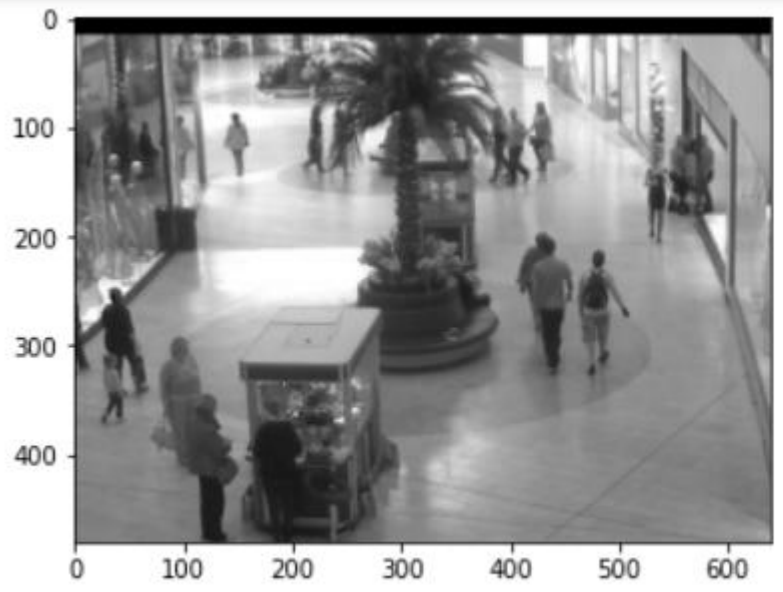
The graph for Training and validation loss are shown below.



The graph for Training and validation Mean Absolute Error (MAE) are shown below.

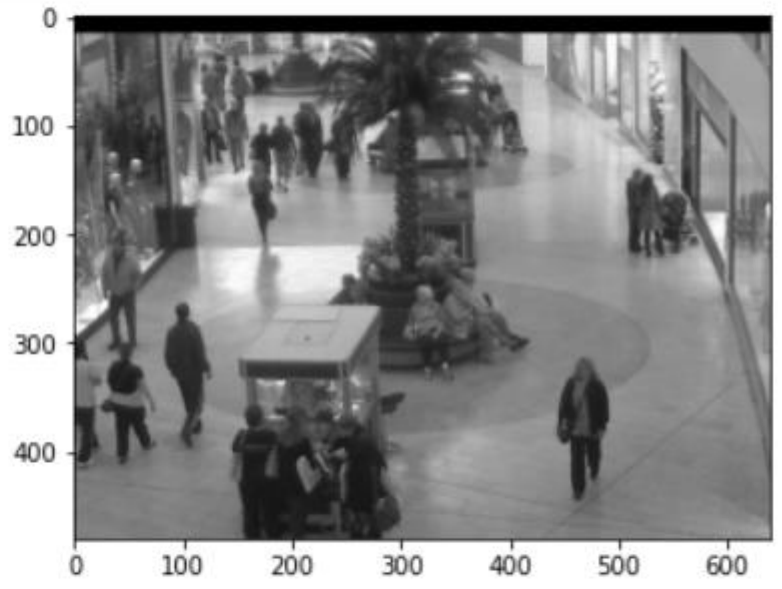


Testing: Finally we have chosen few random images from our testing Mall dataset. The results are shown below.



```
Counted people:45
```

Ground Truth: 22



```
Counted people:63
```

Ground truth: 32

As we can see the results are around 50-60% accurate which is justified for a simple model and shallow neural network model. Areas of Improvements are mentioned below.

Areas for improvement:

This model is performing decently for small crowd size. But for a large crowd size it faces challenges. We can improve the density estimation using more robust techniques and also we can use more deep neural network architecture for better crowd counting. There are many challenges regarding background and foreground disturbances and tiny object detection which can be improved later.

Conclusion:

This project has helped us solve a practical scenario using computer vision and deep learning based technique. Even though the model is performing okay in the scenario, we need to improve some features and modules for better performance of it. However, we have learnt a several new things while implementing the problem and that'll really going to help us approach similar problems in better way.