# DeepVision Crowd Monitor: AI for Density Estimation and Overcrowding Detection

Sanjay Kumar.S

## Table of Contents

# AI DeepVision – Crowd Monitoring Project

## 1. Introduction

Crowd monitoring is an important task for public safety in areas such as malls, stadiums, festivals, and public transport. Manual monitoring is slow and inaccurate. Modern computer vision enables automated crowd density estimation using deep learning.

This project builds an AI-based system that:

- Estimates number of people in an image
- Generates density heatmaps showing crowd distribution
- Detects overcrowding levels

## 2. Dataset Description

We use the ShanghaiTech Crowd Counting Dataset and it has two parts:

| part | characteristics | difficulty |
|---|---|---|
| Part-A | Highly crowded scenes, dense crowds | Hard |
| Part-B | Outdoor scenes, fewer people | Easier |

Each sample contains:

- RGB Image
- Ground truth (.mat file) containing **(x, y)** head locations

The folder structure looks like:

```
|- ShanghaiTech
        |- part-A
                |- test_data
                        |- ground-truth/
                        |- images/
                |- train_data
                        |- ground-truth/
                        |- images/
        |- part-B
                |- test_data
                        |- ground-truth/
                        |- images/
                |- train_data
                        |- ground-truth/
                        |- images/
```

## 3. Problem With Raw Ground Truth

Ground-truth gives points only => not suitable for training CNN directly.

So, we convert points => **density maps**

Each head point contributes to a **Gaussian peak** => sum of density = **count of people**

## 4. Preprocessing Pipeline

Performed on both Part-A and Part-B data using the following steps:

| Step | Purpose |
|---|---|
| Convert BGR – RGB | Standard colour format for deep learning |
| Resize Images (512×512) | Uniform input size |
| Create Impulse Map (zeros + head locations=1) | Binary head map |
| Gaussian Blur (σ=1.5–4) | Converts impulses → density distribution |
| Downsample to 1/8 resolution | To match CSRNet output |
| Save as .npy | Efficient numeric storage |

ƶ Sum of density map = number of people in image

## 5. Density Map Visualization

We displayed:

- Image with head annotations
- Impulse map (binary points)
- Density map (heat style)
- Heatmap overlay on original image

This helps verify dataset correctness.

## 6. Why Downsample Density Map?

CSRNet has pooling layers → output is **1/8 size** of input.

Input: 512 × 512

Output: 64 × 64

So, GT density must be downsampled to same size.

To preserve crowd count,  multiply by 64 (8×8).

## 7. Dataset Splitting

For training the model, split the training data to 80% for training and 20% for testing

And Testing data keeps separate.

## 8. Model Architecture – CSRNet

CSRNet = **Convolutional Neural Network** for crowd counting.

| Component | Description |
|---|---|
| Frontend | VGG-16 style feature extractor |
| Backend | Dilated CNN to enlarge receptive field |
| Output Layer | 1-channel density map |

CSRNet is used because  It

- Works for both dense & sparse crowds
- No detection required → faster
- Accurate counting from density estimation

## 9. Training Setup

| Feature | Value |
|---|---|
| Loss Function | MSE Loss (pixel-wise density difference) |
| Optimizer | Adam |
| Learning Rate | 1e-5 |
| Batch Size | 4 |
| Epochs | 100 |

Monitoring Metrics

- Train Loss
- Validation Loss
- Train MAE
- Validation MAE

MAE → Measures counting accuracy:

MAE = |Predicted Count – Actual Count|