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

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|

For part-A, the test results are
MAE : 98.13
MSE : 19226.94
RMSE : 138.66

For part-B, the test results are
MAE : 22.83
MSE : 1190.72
RMSE : 34.51

10. CSRNet Crowd Counting — Model Loading, Image Testing, Video Processing & Webcam Inference

a. Loading the CSRNet Model & Trained Weights

After training CSRNet on ShanghaiTech Part-A/Part-B datasets, the trained .pth model files are loaded into the system.
This allows the model to perform inference without retraining.

Steps:

1. Import CSRNet model architecture.
2. Move the model to CPU device.

3. Load the trained weights (CSRNet_best.pth or CSRNet_best_2.pth).
4. Set model to evaluation mode.

Purpose:

✓ Load the trained model

✓ Avoid backpropagation

✓ Enable prediction on new images/videos

b. Testing Prediction on a Single Image

To verify that the model works correctly after loading, a simple image is tested.

Steps:

1. Read image using OpenCV.
2. Convert BGR → RGB (CSRNet expects RGB).
3. Normalize using ImageNet mean and std.
4. Convert image into PyTorch tensor.
5. Feed the tensor into CSRNet.
6. Get the density map and compute total count.
7. Overlay heatmap on original image for visualization.

Purpose:

✓ Confirm the model predicts count properly

✓ Understand density map behavior

✓ Visualize crowd intensity on image

c. Video Processing (Frame-by-Frame Crowd Counting)

CSRNet is applied on each frame of a video to generate real-time predictions.

Steps:

1. Open a video file using cv2.VideoCapture()
2. Loop through video frame-by-frame
3. Preprocess each frame
4. Predict density map
5. Sum density values to get person count
6. Generate heatmap and overlay on frame
7. Display frame inside notebook
8. Continue until video ends

Purpose:

✓ Handle dynamic scenes

✓ Demonstrate CSRNet on real video

✓ Produce visual heatmaps for each frame

d. Real-Time Crowd Counting with Live Webcam

After video prediction works, the next step is live webcam inference.

Steps:

1. Open webcam feed (VideoCapture(0))
2. Read frames continuously
3. Resize frame to match CSRNet crowd scale

4. Preprocess frame
5. Predict density map
6. Clip negative density values (avoid negative counts)
7. Compute final count
8. Overlay heatmap + display live
9. Exit when user presses q

Purpose:

✓ Create a real-time crowd monitoring system

✓ Display real-time heatmap and person count

✓ Demonstrate final working model in practical use

11. Fine-Tuning

a. Why Fine-Tuning Is Required

Pre-trained CSRNet models are trained on datasets like ShanghaiTech, which differ from:

- Real-time CCTV footage
- Camera angle
- Lighting conditions
- Crowd distribution

So, directly applying a pre-trained model gives **poor accuracy**.

Fine-tuning adapts the model to our real-world data.

b. Overall Workflow (High-Level)

Video => Frames => Pseudo Head Points

- Gaussian Density Maps
- Dataset Preparation
- CSRNet Fine-Tuning
- Real-Time Inference

c. Dataset Preparation

Video to Frames

- Real-time or recorded crowd videos were collected.
- Videos were converted into individual frames.
- These frames represent real-world crowd scenes.

Purpose:

CSRNet is trained on images, not directly on videos.

d. Pseudo Head Annotation (No Manual Labeling)

Manual annotation of head points is time-consuming and impractical.

Instead, we used:

- A pre-trained CSRNet model
- Generated pseudo head annotations automatically

Steps:

1. Each frame is passed through pre-trained CSRNet
2. A density map is obtained
3. Local peaks in density correspond to head locations

This method is:

- Research-accepted
- Scalable
- Similar to ShanghaiTech labelling

e. Density Map Generation

CSRNet does not train on head points, it trains on density maps.

f. Gaussian Density Map

For each head point:

- A **Gaussian kernel** is placed
- All kernels are summed to form a density map

Properties:

- Integral of density map ≈ crowd count
- Smooth spatial representation
- Stable training

g. Downsampling Density Maps

CSRNet architecture downsamples input images due to pooling layers.

Therefore:

- Ground truth density maps must match model output resolution

Rule:

If input image is H × W

CSRNet output is (H/8) × (W/8)

So:

1. Density map is created at original resolution
2. Then downsampled by factor 8
3. Density values are scaled to preserve count

This step is critical for correct training.

h. Dataset Structure

The final dataset is organized as:

```
dataset/
    ├── images/
    │   ├── frame_1.jpg
    │   ├── frame_2.jpg
    │
    ├── density/
    │   ├── frame_1.npy
    │   ├── frame_2.npy
```

- Each image has a corresponding density map
- Density maps are 2D arrays
- Shapes match CSRNet output

i. CSRNet Model Architecture

CSRNet consists of:

Frontend

- Based on VGG-16
- Extracts low-level and mid-level features

Backend

- Dilated convolutions
- Enlarged receptive field
- Preserves spatial resolution

Output Layer

- Single-channel density map

Output:

Density Map => Sum => Crowd Count

j. Training Configuration

8.1 Loss Function

- Mean Squared Error (MSE)
- Measures difference between predicted and ground-truth density maps

Why MSE?

- Standard loss used in CSRNet
- Stable for density regression

k. Optimizer

- Adam Optimizer
- Very low learning rate

Reason:

We are fine-tuning, not training from scratch.

l. Fine-Tuning Strategy

Fine-tuning is done in two phases.

(i) Phase-1: Backend-Only Training

- Frontend (VGG) layers are frozen
- Only backend layers are trained

Purpose:

- Adapt density estimation to new crowd patterns
- Avoid destroying pre-trained features

Duration:

- Few epochs (5–10)

Expected behavior:

- Loss decreases steadily
- Counts become reasonable

(j) Phase-2: Full Network Fine-Tuning

- Frontend layers are unfrozen
- Entire network is trained

Purpose:

- Adapt model to camera angle, lighting, and perspective
- Improve accuracy further

Duration:

- More epochs (20–40)

m. Training Validation

During training, we verified:

- Loss decreases across epochs
- No shape mismatch errors
- Predicted count ~ Ground truth count
- Density blobs align with heads

This confirms correct fine-tuning.

n. Final Model Output

The fine-tuned CSRNet produces:

- A density map
- Crowd count = sum of density
- Works well on real-world scenes

12. Dashboard

The Crowd Monitoring Dashboard is a web-based application designed to monitor crowd density in videos, estimate the number of people, visualize density heatmaps, and trigger alerts when overcrowding is detected.

The system is built using a fine-tuned CSRNet deep learning model, combined with a Streamlit web interface and an SMTP-based email alert mechanism.
It is suitable for applications such as public safety, event monitoring, campus surveillance, and crowd control systems.

Objectives of the Dashboard

a. Display real-time crowd count
b. Visualize density heatmaps over video frames
c. Detect overcrowding conditions
d. Send email alerts automatically
e. Provide a user-friendly web interface
f. Support dynamic input videos

Architecture Flow

User Uploads Video
|
Frame Extraction (OpenCV)
|
Preprocessing (Resize + Normalize)

|
CSRNet Model Inference
|
Density Map Generation
|
Crowd Count Estimation
|
Heatmap Overlay Visualization
|
Alert Threshold Check
|
SMTP Email Alert (if exceeded)

CSRNet Model Description

CSRNet is a convolutional neural network designed for dense crowd counting.

Key Features:

- Uses VGG-style frontend
- Dilated convolutions in backend
- Outputs density maps, not bounding boxes
- Crowd count obtained by summing density map

Output:

Density Map → Sum of values = Crowd Count

Dashboard Functional Modules

1. Model Loader Module
   Purpose:
   Loads the fine-tuned CSRNet model once and reuses it efficiently.
   Key Points:
   - GPU/CPU auto-detection
   - Cached using @st.cache_resource
   - Prevents repeated model loading
2. Video Upload Module
   Purpose:
   Allows users to upload a video dynamically through the web interface.
   Supported Formats:
   - .mp4
   - .avi
3. Frame Processing Module
   Each video frame undergoes:
   - Resize (performance optimization)
   - Normalization (ImageNet stats)
   - Conversion to tensor
   - Inference using CSRNet
4. Density Map & Heatmap Visualization
   Density Map:
   - Raw output from CSRNet

- Downsampled resolution

Heatmap Overlay:

- Density map resized to frame size
- Color-mapped using JET
- Overlaid on original frame

Purpose:

- Visual understanding of crowd concentration
- Identifies high-density zones clearly

5. Crowd Counting Module

   count = int(density.sum().item())

   - Accurate for dense crowds
   - Works without bounding boxes
   - Robust to occlusions

6. Alert Detection Module

   Alert Trigger Condition:

   If crowd_count ≥ ALERT_THRESHOLD

   Visual Alert:

   - Red alert message on dashboard
   - Status indicator updated

   Threshold:

   - User-adjustable via Streamlit slider

7. Email Alert System (SMTP)

   Purpose:

   Send real-time email alerts when overcrowding is detected.

   Technology:

   - smtplib
   - Gmail SMTP with App Password

   Email Content:

   - Crowd count
   - Timestamp
   - Alert message

   Security:

   - App Password used instead of real Gmail password