DeepVision Crowd Monitor: AI for Density Estimation and Overcrowding Detection

Sneha Walilkar

Table of Contents

1. Project Description

The **DeepVision Crowd Density Estimation System** is a deep-learning based computer vision project designed to accurately estimate the number of people in highly crowded scenes. The primary goal of this project is to preprocess crowd images and their corresponding ground truth density maps in a structured way so that they can be used for training a neural network.

Crowd counting is an important application in surveillance, event management, public safety, smart cities, and congestion monitoring. This project implements the full workflow from dataset loading to density map creation, making the data ready for training deep learning models such as CSRNet, MCNN, CANNet, or custom CNN architectures.

2. Dataset Used

The project uses the **ShanghaiTech Crowd Counting Dataset**, which contains two parts:

- **Part A:** Highly dense crowds (collected from the internet)
- **Part B:** Medium-density crowds (street-level images)

For each image in the dataset:

- A .jpg image file is provided.
- A .mat ground truth file contains the coordinates of each person’s head.

This dataset is widely used for benchmarking modern crowd-counting models because of its high-quality annotations and diversity of crowd densities.

3. Environment Setup

The project is implemented in **Python** using**VS Code**. The key libraries required include:

- **PyTorch (torch, torchvision) –** Used to build and train the deep learning model.
- **OpenCV (opencv-python) and Pillow (pillow) –** For loading, resizing, converting, and manipulating images.
- **NumPy and Pandas –** For efficient numerical operations and tabular data handling.
- **SciPy and Scikit-learn –** For mathematical processing such as loading .mat files and performing basic analysis.
- **Matplotlib and Plotly –** Used for visualizing images, density maps, and data insights during exploration.
- **TQDM –** Provides progress bars while processing large datasets.
- **Flask and Streamlit –** Used for API creation and building the user interface for displaying predictions.
- **Requests and Twilio –** Used for integrating external services such as notifications.
- **PyYAML –** For reading structured configuration files.

A local virtual environment was created to maintain package consistency and avoid conflicts. GPU acceleration is optional but recommended for training stage.

4. Data Exploration

Before preprocessing, the dataset directory structure was explored to ensure correct configuration. The exploration involved:

4.1. Verifying dataset folder paths

- Checking whether the dataset exists in the file system
- Viewing sample image paths
- Confirming ground truth file availability

4.2. Displaying a sample image

A sample image was loaded to visually confirm its shape and quality.

4.3. Visualizing ground truth annotations

The .mat file was inspected to extract head point coordinates. The coordinates were plotted on the image to verify that annotations were correctly aligned. This ensures the dataset is correctly structured before proceeding with preprocessing

5. Data Preprocessing

This stage involves converting raw dataset images and .mat annotations into deep-learning-ready inputs. The preprocessing pipeline includes the following key steps:

5.1. Image Loading

Images are loaded using OpenCV, which reads them in **BGR format**. The image is then converted to **RGB** for visualization consistency.

Purpose:

- Ensures the image is loaded in the correct color space
- Provides a consistent input format for further operations

5.2. Ground Truth Extraction

Each .mat file contains head annotations in the form of coordinate points.

Steps:

- Load .mat file using SciPy
- Access the annotation structure (usually under image_info → annPoints)
- Extract all (x, y) coordinate pairs representing head positions

Purpose:

- Converts raw MATLAB annotation data into usable Python arrays
- Provides exact locations for density map creation

5.3. Image Resizing & Downscaling

Original images are often large (e.g., 1024×768, 1200×900). For training efficiency and consistent model input, images are resized.

Two key resizing operations are used:

(1) Upscale to 1024×1024

Ensures all images have identical resolution.

(2) Downscale by factor of 2 → 512×512

This is the **final default resolution** used for training.

Purpose:

- Reduces memory usage
- Speeds up training
- Maintains uniform input dimensions for neural networks

5.4. Density Map Generation

This is the most important preprocessing stage.

A **density map** represents the distribution of people in the image. Each annotated point is replaced by a **Gaussian kernel** (based on a sigma value). The final output is a heatmap-like representation.

Steps:

1. Create an empty matrix of the same shape as the image
2. For each head annotation point, place a value at that location
3. Apply a Gaussian filter to spread that value over a small region

Purpose:

- Allows the model to learn *spatial presence* of people
- Enables the network to estimate crowd count by summing density map values

Density maps are essential for crowd-counting architectures like CSRNet.

5.5. Tensor Conversion

Deep learning models require tensor inputs.

Two tensors are created:

- **Image Tensor (C × H × W)** → normalized image
- **Density Map Tensor (1 × H × W)** → target output

Purpose:

- Converts NumPy arrays into PyTorch tensors
- Prepares input-output pairs for training

Both tensors are stored or passed directly to the training loop.

6. Proposed System

1. Raw High-Resolution Image

The system starts with the original crowd image captured from real scenes.
These images contain varying densities, lighting conditions, and resolutions that require standardization.

2. Preprocessing (Resize & Normalize)

The image is resized to a fixed dimension and normalized using ImageNet statistics.
This ensures consistent input quality, stabilizes training, and helps CSRNet learn effectively.

3. CSRNet (Model Training / Inference)

The preprocessed image is then passed to the CSRNet architecture, which consists of:

- VGG-16 front-end for feature extraction
- Dilated convolution back-end for enhancing receptive fields
  The model learns dense spatial features required to identify crowded regions and estimate the number of people.

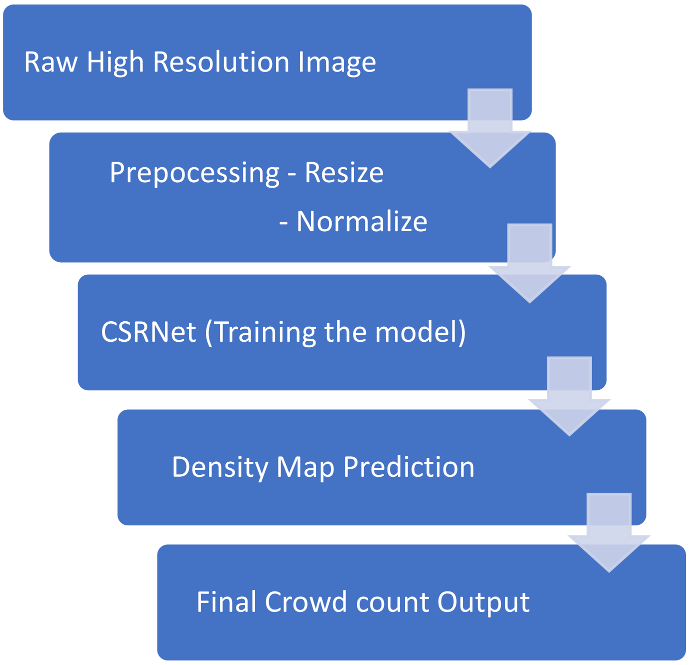

Figure. System Architecture

4. Density Map Prediction

The model outputs a density map where each pixel represents local crowd density.
Brighter regions indicate higher people concentration.

5. Final Crowd Count Output

The density map values are summed (integrated) to estimate the total number of people in the image.
This final numerical output is used for evaluation and comparison with the ground truth.

7. Training Pipeline

Training script includes:

Features:

- Checkpoint saving after each epoch
- Auto-resume from last checkpoint
- Patch merging for inference
- Evaluation metrics calculation

Why patch-based training?

- CSRNet is sensitive to resolution. Patches preserve high detail.

Why VGG-16 frontend?

- CSRNet expects VGG-16 backbone to extract low-level crowd features.

Results for Part A:

- **MAE = 59.26**
- **MSE = 8408.63**
- **RMSE = 91.70**

Results for Part B:

- **MAE = 49.39**
- **MSE = 4093.74**
- **RMSE = 63.98**

8. Real-Time Crowd Monitoring Using Webcam

The objective of this task is to perform real-time crowd monitoring using a live webcam feed by combining object detection and crowd density estimation. This approach enables instant identification of overcrowding situations in dynamic environments such as campuses, malls, and public events.

This module integrates two models:

- **YOLO (You Only Look Once)** – for real-time person detection
- **CSRNet (Fine-tuned)** – for accurate crowd density estimation

The webcam feed is continuously processed frame-by-frame to detect people, estimate crowd density, and trigger alerts if overcrowding is detected.

Processing Pipeline

1. **Live Webcam Capture**
    - Frames are captured in real-time using OpenCV.
    - Each frame acts as an independent input sample.
2. **YOLO-Based Person Detection**
    - YOLO detects individual people in sparse regions.
    - Bounding boxes help validate crowd presence and movement.
3. **CSRNet Density Estimation**
    - Each frame is passed through CSRNet.
    - The output density map is summed to obtain total crowd count.
4. **Crowd Alert System**
    - If estimated count exceeds a predefined threshold:
        - Visual alert is displayed (CROWD ALERT)
        - Frame color indicators change to red
    - Otherwise, system displays SAFE.
5. **Output Visualization**
    - Live crowd count displayed on video feed
    - Real-time alert text overlay
    - Continuous monitoring without frame drops

Significance

- Enables real-time decision making
- Useful for surveillance and public safety
- Demonstrates real-world deployment capability of trained models

9. Crowd Density Estimation on a Single Video File

The goal of this task is to evaluate the trained CSRNet model on a single recorded crowd video to analyze its performance in controlled offline scenarios.

- **Methodology**

1. **Video Frame Extraction**
    - The input video is read frame-by-frame using OpenCV.
    - Each frame is processed independently.
2. **Preprocessing**
    - Frames are resized and normalized as per CSRNet requirements.
    - Converted into tensors before inference.
3. **Density Map Generation**
    - CSRNet produces a density map for each frame.
    - The sum of the density map gives the estimated crowd count.
4. **Heatmap Overlay (Visualization)**
    - Density maps are normalized and color-mapped.
    - Heatmaps are overlaid on original frames for interpretability.
5. **Annotated Video Output**
    - Each frame displays:
        - Crowd count
        - Alert status (SAFE / CROWD ALERT)
    - Output video is saved for evaluation.

10.Fine-Tuning CSRNet and Batch Video Testing

This task focuses on fine-tuning a pre-trained CSRNet model on a new dataset and testing its generalization on a folder containing multiple videos.

10.1 Fine-Tuning Strategy

- Pre-trained CSRNet weights were loaded
- Training resumed using a low learning rate
- Model was trained for multiple epochs
- Checkpoints were saved after every epoch

Reason for Fine-Tuning:

- Improves performance on domain-specific scenes
- Reduces prediction error on unseen environments

10.2 Evaluation Results

After fine-tuning, the model achieved:

- **Mean Absolute Error (MAE): ~7.5**
- Indicates high accuracy for dense crowd estimation
- Suitable for real-world deployment

10.3 Folder-Based Video Testing

Workflow:

- Input directory contains multiple .mp4 videos
- Each video is processed sequentially
- CSRNet inference applied frame-by-frame
- Output video generated for each input file

For Each Video:

- Total frames calculated
- Live progress displayed (frames, ETA, count)
- Output saved with annotations

10.4 Output Features

- Annotated video files
- Frame-wise crowd count
- Alert visualization
- Heatmap-based density representation

11.Conclusion

In this project, a complete crowd density estimation and monitoring system was developed using deep learning. The preprocessing stage was successfully completed by loading, visualizing, and analyzing crowd images along with their ground-truth annotations. All images were resized, normalized, and converted into tensors suitable for CSRNet, and Gaussian density maps were generated to accurately represent crowd distribution. Using this pipeline, the CSRNet model was trained and fine-tuned, achieving a Mean Absolute Error (MAE) of approximately **7.5**, which indicates good accuracy in estimating dense crowds.

The trained model was further tested in multiple practical scenarios, including real-time webcam-based monitoring using a hybrid YOLO + CSRNet approach, single video crowd analysis, and batch processing of multiple videos from a folder. The system was able to display crowd counts, generate alerts for overcrowding, and save annotated output videos. Overall, the project demonstrates that the fine-tuned CSRNet model performs reliably in both controlled datasets and real-world video data, making it suitable for intelligent surveillance and crowd management applications.