DEEPVISION CROWD MONITOR

AI FOR DENSITY ESTIMATION AND OVERCROWDING DETECTION

Submitted by:

Arti Kumari

TABLE OF CONTENTS

1.Introduction

Crowd monitoring and crowd density estimation are fundamental problems in the field of computer vision with direct applications in public safety, smart cities, transportation systems, disaster management, and large-scale event monitoring. With the increasing availability of surveillance cameras and CCTV networks, vast amounts of video data are generated continuously. However, manual monitoring of such data is inefficient, error-prone, and impractical at scale.

Traditional computer vision approaches attempted to solve crowd counting by detecting and counting individual people. While effective in low-density environments, these methods fail in real-world crowded scenes due to severe occlusion, overlapping individuals, perspective distortion, and low-resolution footage. To overcome these limitations, modern approaches rely on **density map regression**, where the objective is to estimate a spatial distribution of people rather than detecting each individual explicitly.

This project, **DeepVision Crowd Monitor**, presents an end-to-end AI-based system that integrates data analysis, deep learning–based crowd counting, real-time video processing, adaptive model switching, fine-tuning using real-world data, and deployment through a web-based dashboard with automated alerting.

1.1 Background and Motivation

Crowd disasters such as stampedes and overcrowding incidents often occur due to delayed detection and poor situational awareness. Early identification of abnormal crowd buildup can significantly reduce risk and improve response time. Automated crowd monitoring systems provide an effective solution by continuously estimating crowd size and density in real time.

Research in crowd counting has evolved from handcrafted feature-based methods to deep learning approaches. Among them, density-based models such as CSRNet have demonstrated superior performance in congested scenes. However, most research solutions remain confined to offline evaluation and lack real-time deployment and alert mechanisms.

The motivation behind this project is to:

- Develop a **robust crowd counting model** for dense scenes
- Extend the model to **real-time video streams**
- Adapt the model to **real-world CCTV environments**
- Deploy the system in a **user-friendly and actionable form**

1.2 Problem Statement

The problem addressed in this project is:

To design and implement a real-time crowd monitoring system that accurately estimates crowd density and count across varying crowd conditions, adapts to real-world surveillance data, and provides automated alerts through an interactive dashboard.

The system must handle challenges such as occlusion, scale variation, domain shift, and real-time performance constraints.

1.3 Objectives of the Project

The objectives of the project are:

- To explore and analyze crowd datasets
- To visualize and preprocess data for density-based learning
- To train and evaluate a deep learning model for crowd counting
- To integrate real-time webcam and CCTV video streams
- To fine-tune the model using pseudo-labeling
- To deploy a real-time dashboard with alert functionality

2.System Overview

The DeepVision Crowd Monitor follows a modular pipeline architecture that separates data processing, model inference, visualization, and alerting. This design improves scalability, maintainability, and extensibility.

2.1 Project Workflow

The workflow consists of the following stages:

1. Dataset exploration and preprocessing
2. Density-based model training and evaluation
3. Real-time video inference and hybrid model switching
4. Fine-tuning using real-world video data
5. Deployment via dashboard and alert system

Each stage builds upon the previous one, forming a complete AI pipeline from data to deployment.

2.2 Overall System Architecture

The system architecture consists of five layers:

- **Input Layer:** Images and video streams
- **Preprocessing Layer:** Resizing, normalization, density generation
- **Model Layer:** CSRNet and YOLOv8
- **Application Layer:** Streamlit dashboard
- **Alert Layer:** SMTP email notifications

This layered approach enables clean separation of concerns.

2.3 Tools, Libraries, and Technologies Used

- **Python** – core programming language
- **PyTorch** – deep learning framework
- **Torchvision** – pretrained VGG-16 backbone
- **OpenCV** – video processing
- **YOLOv8** – person detection
- **Streamlit** – web dashboard
- **SMTP (Gmail)** – email alerts

3. Phase 1: Data Exploration, Data Visualization, and Data Preprocessing

3.1 Understanding Crowd Counting Data

Crowd counting datasets differ from object detection datasets in that annotations often consist of head locations rather than bounding boxes. These annotations are converted into density maps to enable regression-based learning.

3.2 Dataset Structure and Annotation Format

Each image is associated with point annotations (x_i, y_i). The total number of points represents the crowd count. Density maps are generated by convolving these points with Gaussian kernels.

3.3 Data Exploration Techniques

Exploration included:

- Inspecting image resolution and aspect ratio
- Analyzing crowd density distribution
- Identifying sparse vs dense scenes

3.4 Visualization of Crowd Images and Annotations

Visualization helped verify annotation correctness and understand spatial crowd distribution.

3.5 Density Map Generation using Gaussian Kernels

The density map is defined as:

$$D(x) = \sum_{i=1}^{N} G_\sigma (x - x_i)$$

where $G_\sigma$is a Gaussian kernel. The sum of the density map equals the crowd count.

3.6 Image Preprocessing Techniques

- Image resizing
- Normalization to [0، 1]
- Channel reordering

3.7 Challenges Identified During Data Preparation

- Perspective distortion
- Severe occlusion
- Annotation inconsistencies

3.8 Outcome of Phase 1

A clean, structured dataset suitable for training deep learning models was prepared.

4. Phase 2: Model Training and Performance Evaluation

4.1 Overview of Crowd Counting Models

Detection-based methods fail in dense scenes, motivating density-based models like CSRNet.

4.2 CSRNet Architecture and Design Rationale

CSRNet uses:

- VGG-16 frontend
- Dilated convolution backend
- Single-channel density output

4.3 Density Map Regression Approach

The model learns a mapping from image space to density space.

4.4 Loss Function and Optimization Strategy

Mean Squared Error (MSE):

$$L = \frac{1}{N} \sum \parallel D_{pred} - D_{gt} \parallel^2$$

4.5 Training Configuration and Hyperparameters

- Low learning rate
- Small batch size

- Adam optimizer

4.6 Evaluation Metrics (MAE and RMSE)

$$MAE = \frac{1}{N}\sum \mid C_{pred} - C_{gt} \mid$$

$$RMSE = \sqrt{\frac{1}{N}\sum (C_{pred} - C_{gt})^2}$$

4.7 Experimental Results and Analysis

The model achieved low MAE in moderate density scenes and stable performance overall.

4.8 Outcome of Phase 2

A trained CSRNet model ready for real-time deployment.

5. Phase 3: Real-Time Webcam Integration and Model Fine-Tuning

5.1 Need for Real-Time Adaptation

Offline-trained models face domain shift when applied to CCTV footage.

5.2 Real-Time Video Processing Pipeline

Frames are captured, preprocessed, inferred, and visualized in real time.

5.3 Hybrid YOLO–CSRNet Inference Strategy

YOLO handles sparse scenes; CSRNet handles dense scenes.

5.4 Auto-Switching Mechanism Based on Crowd Density

Inference mode is selected dynamically based on estimated crowd count.

5.5 Domain Shift in CCTV Environments

Differences in lighting, resolution, and camera angles affect performance.

5.6 Pseudo-Labeling Technique for Fine-Tuning

CSRNet predictions are reused as pseudo ground truth.

5.7 Fine-Tuning Strategy and Layer Freezing

Frontend layers are frozen; backend layers are updated.

5.8 Visualization and Validation of Fine-Tuned Results

Density maps are visually inspected to ensure correctness.

5.9 Outcome of Phase 3

A fine-tuned model adapted to real-world data.

6. Phase 4: Dashboard Development and SMTP-Based Alert System

6.1 Deployment Objectives

To provide a usable, real-time monitoring interface.

6.2 Streamlit Dashboard Architecture

Streamlit enables fast deployment and live visualization.

6.3 Real-Time Crowd Visualization and Metrics Display

Live feed, count, and mode are displayed.

6.4 Density Map Heatmap Visualization

Density maps are converted into heatmaps for intuitive interpretation.

6.5 Alert Threshold Configuration

User-defined thresholds control alert triggering.

6.6 SMTP Email Alert System Design

Email alerts are sent using Gmail SMTP with TLS.

6.7 Alert Trigger Logic and Session-State Management

Session state prevents repeated alerts.

6.8 Deployment Challenges and Handling

Challenges include camera access and SMTP authentication.

6.9 Outcome of Phase 4

A fully deployable crowd monitoring system.

Results and Observations

7.1 Quantitative Performance Analysis

The system demonstrates stable crowd estimation accuracy.

7.2 Qualitative Analysis using Density Maps

Density maps provide spatial insight.

7.3 Real-Time System Behavior

The system operates smoothly in real time.

7.4 Strengths and Limitations of the System

Strengths include adaptability and robustness; limitations include dependency on camera quality.