Deep Vision Crowd Monitor: AI for Density Estimation and Overcrowding Detection

Intern Name: Nithya Shree S

TABLE OF CONTENT

S.No	Section	Page No.
1	Project Description	3
2	Dataset Description	4
3	Environment Setup	5
4	Data Exploration	6
5	Data Preprocessing	7

1.Project Description

Project Name: Deep Vision Crowd Monitor: AI for Density Estimation and Overcrowding Detection

Mentor: G.K.S Jyoteesh

Objective:
The objective of the project is to develop a real-time deep learning-based system capable of accurately estimating crowd density and detecting overcrowded zones using surveillance video feeds. This system is designed to enhance public safety, assist emergency response teams, and optimize crowd flow management in high-footfall areas, such as transit hubs, public events, religious gatherings, and smart city infrastructures.

Approach:
The project leverages convolutional neural networks (CNNs) such as CSRNet or MCNN for crowd density estimation. It generates density maps from video frames, estimates crowd counts, and flags overcrowded zones. The system also includes visual heatmaps and real-time alerts to authorities for proactive intervention.

Key Features:

- Real-time crowd density estimation
- Overcrowding detection
- Heatmap visualization
- Alert generation through email or SMS

Project Workflow:

1. Video Input: Capture live feed from CCTV/surveillance cameras
2. Frame Extraction: Sample video frames at fixed intervals
3. Preprocessing: Resize, normalize, and clean image frames
4. Crowd Estimation: Generate density maps using CNN-based models
5. Counting & Detection: Estimate crowd count and detect overcrowded zones
6. Alert & Visual Output: Display heatmaps and send real-time alert

2. Dataset Description

Description:

- The dataset contains images of crowds captured in various scenarios, along with their corresponding density maps.
- It is widely used for research in crowd counting and density estimation.

Attributes:

- Image Data: Photographs of crowds in public areas
- Density Maps: Numeric arrays representing crowd density per image
- Annotations: Points marking individual heads for ground truth

Distribution of Data:

- Part A: Highly congested scenes (urban streets, mass gatherings)
- Part B: Sparse crowd scenes (open areas, plazas)

Sample Visualization:

- Sample images can be displayed using Python libraries like OpenCV, Matplotlib, or Pillow.

3. Environment Setup

Python Modules Installed:

torch, torchvision, opencv-python, numpy, pandas, matplotlib, pillow,

scipy, scikit-learn, plotly, tqdm, flask, streamlit, twilio, requests, pyyaml

Steps:

1. Created virtual Python environment
2. Installed required libraries via requirements.txt
3. Optional: CUDA setup for GPU acceleration (for real-time performance)
4. Preferred IDE: Jupyter Notebook / VSCode

Command Used:

```
pip install -r requirements.txt
```

4. Data Exploration

Steps Taken:

1. Loaded sample images from the dataset
2. Observed image dimensions, color channels, and distribution
3. Visualized a few sample images using Matplotlib
4. Verified density maps and annotations

Observations:

- Images have variable resolutions
- Crowd density varies from sparse to very dense
- Density maps are grayscale arrays where higher intensity corresponds to higher crowd density

Visualization Example:

- Display a sample image alongside its density map to understand distribution

5. Data Preprocessing

Data preprocessing is a crucial step in preparing the dataset for deep learning models. It ensures that the input data is clean, consistent, and ready for effective model training. The preprocessing pipeline for the Deep Vision Crowd Monitor project includes the following steps:

1. Image Resizing

- All images are resized to a uniform dimension to match the input requirements of the CNN models (CSRNet/MCNN).
- This standardization ensures consistent feature extraction across all images.
- Example: Images resized to 224x224 or 256x256 pixels depending on model input configuration.

2. Normalization

- Pixel values of images are scaled to a range of 0 to 1 or standardized using mean and standard deviation.
- Normalization helps the model converge faster during training and reduces the risk of exploding gradients.

3. Data Cleaning

- Removed corrupted, duplicate, or low-quality images from the dataset.
- Ensured all density maps have proper correspondence with images.
- Verified the integrity of image annotations for accurate crowd counting.

4. Data Augmentation

- Applied techniques to artificially increase the dataset size and improve model generalization.
- Common augmentations:

    - Horizontal and vertical flips
    - Random rotation and zoom
    - Cropping and padding
    - Color jittering (brightness, contrast adjustments)
- Augmentation prevents overfitting and allows the model to learn variations in crowd scenarios.

5. Conversion to Tensors

- Images and density maps are converted to PyTorch tensors for model training.
- Ensures compatibility with deep learning frameworks and GPU acceleration.

6. Splitting Dataset

- The dataset is split into **training, validation, and testing sets**.
- Typical ratio: 70% training, 15% validation, 15% testing.
- Ensures proper evaluation of model performance and prevents data leakage.

7. Visualization after Preprocessing

- Sample preprocessed images and their density maps are visualized to verify transformations.
- Helps ensure that resizing, normalization, and augmentations have been applied correctly.

Purpose of Preprocessing:

- Ensures consistent input format for deep learning models.
- Reduces noise and improves model accuracy.
- Increases dataset size and variability through augmentation.
- Prepares data for efficient real-time inference on CCTV or surveillance feeds.