Deep Vision Crowd Monitor:

AI-Based Density Estimation and Overcrowding Detection

Anshla Pagdal

TABLE OF CONTENTS

1. **Project Description**
2. **Dataset Description**
3. **Environment Setup**
4. **Data Exploration**
5. **Data Preprocessing**
    - Density Map Generation
    - Visualization
    - Resizing & Downscaling
    - Tensor Conversion
6. **Output Files Generated**

PROJECT DESCRIPTION

Deep Vision Crowd Monitor aims to build an AI-powered system for crowd density estimation and overcrowding detection using surveillance images and video frames. The system uses deep learning models such as CSRNet to generate density maps that estimate the number of individuals in a scene.

The objective is to analyze crowded environments like public gatherings, transit hubs, and events, enabling authorities to make data-driven decisions and manage crowd flow efficiently.
Milestone-1 focuses on the **complete preprocessing workflow**, which prepares the dataset for model training.

DATASET DESCRIPTION

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

Part A

- Contains 300 highly congested images
- Captured from the internet
- Dense crowd regions with hundreds to thousands of people

Part B

- Contains 400 moderately crowded images
- Captured in city streets
- Lower crowd density

Each dataset includes:

- **Original images** (`.jpg`)

- **Ground truth annotations** (`.mat`) containing head coordinates
- **Train/Test splits**

My preprocessing work is applied to **Part A Train** and **Part B Train**, as required for training.

ENVIRONMENT SETUP

The environment was configured using a Python virtual environment to isolate dependencies and ensure reproducible results. Essential modules installed include:

- **NumPy**, **Pandas** – for array and data handling
- **SciPy** – for loading `.mat` annotation files and Gaussian filtering
- **Pillow** – for image operations
- **OpenCV** – for resizing and density map scaling
- **PyTorch** – for dataset creation and tensors
- **Matplotlib** – for visualization
- **tqdm** – for progress bars

A structured workspace was created with folders for:

- Dataset
- Preprocessing scripts
- Jupyter notebooks
- Visualization outputs
- CSV reports
- Density maps

This environment ensures smooth preprocessing and later CSRNet training.

DATA EXPLORATION

Before preprocessing, the dataset was explored by me to understand its structure and properties:

1. Image Shape Analysis

Images in Part A and B have **variable resolutions**, requiring resizing for model compatibility.

2. Ground Truth Examination

Each `.mat` file contains structured fields with head coordinate arrays.
The number of coordinates equals the true crowd count.

3. File Consistency Check

Ensured that every image has a corresponding ground-truth file:

- `IMG_40.jpg` → `GT_IMG_40.mat`

4. Sample Visualization

A few sample images and annotated points were visualized to verify annotation correctness.

This exploration confirmed dataset quality and prepared for preprocessing.

DATA PREPROCESSING

1. Density Map Generation

For each image:

- A blank matrix of size (H × W) was created.
- Ground-truth head coordinates were placed as points.
- A Gaussian filter (`sigma=15`) was applied to generate a smooth density distribution.
- The **sum of the density map equals the total number of people**.

This converts discrete head positions into a continuous density representation.

2. Visualization

For each image, a **three-panel visualization** was generated containing:

1. Original image
2. Ground truth points overlay
3. Density map heatmap with predicted count

These images help verify annotation correctness and preprocessing accuracy.

3. Resizing & Downscaling

To prepare data for CSRNet:

- Images were resized while preserving the **aspect ratio**.
    - The **width was standardized to 1024 pixels**
    - Height was scaled proportionally
- Density maps were resized using interpolation
- Density values were **scaled to preserve the crowd count** after resizing

This ensures that images are uniform for training, while keeping head sizes realistic and counts accurate.

4. Tensor Conversion

For PyTorch model training:

- Each image was converted to a **3-channel normalized tensor**
- Each density map was converted to a **1-channel floating-point tensor**

This enables the DataLoader to load batches efficiently.

So I, verified tensor correctness using a testing script that confirms:

- Image shape: `[1, 3, H, 1024]`
- Density shape: `[1, 1, H, 1024]`
- GT Count preserved

5. CSV Report Generation

For each dataset, a summary report was generated containing:

- Image name
- Ground truth count
- Density map predicted count
- Absolute error
- Dataset name

This provides a quick overview of annotation accuracy.

OUTPUT FILES GENERATED

My preprocessing pipeline generated the following:

Density maps (`.npy`)

Visualization images (`*_viz.png`)

CSV reports (`report.csv`)

Resized + scaled data for training

Tensor-ready dataset loading pipeline

Verified dataset shapes and counts