# DEEP VISION CROWD MONITOR: AI FOR DENSITY ESTIMATION AND OVERCROWDING DETECTION

SARJANA NAGAMANIKANDAN

**Project Description:**

The objective of Deep Vision Crowd Monitor is to develop a realtime deep learning-based system capable of accurately estimating crowd density and detecting overcrowded zones using surveillance video feeds. The system aims to enhance public safety, support emergency response, and optimize crowd flow management in high-footfall areas such as transit hubs, public events, religious gatherings, and smart city infrastructures.

By leveraging convolutional neural networks and crowd estimation algorithms, the project seeks to provide timely insights and automated alerts to authorities, enabling proactive interventions and efficient crowd control.

**DATASET USED:**

ShanghaiTech Crowd Counting Dataset: Includes labeled images and density maps.

**Environment Setup, Data Exploration, and Data Preprocessing:**

The ShanghaiTech Crowd Counting Dataset is a widely used benchmark dataset designed for crowd counting and density estimation tasks. It consists of two parts: Part A, containing images of highly congested urban scenes, and Part B, consisting of relatively sparse crowd scenes from public places. Each image in the dataset is accompanied by ground-truth annotations provided as dot annotations stored inside MATLAB .mat files. These annotations indicate the locations of individuals in the image and are used to generate density maps for training deep learning models. The dataset structure typically includes separate folders for training and testing images along with corresponding ground-truth files. This dataset's diversity and accurate labeling make it ideal for developing robust crowdcounting models.

## 1. Environment Setup

For working with the ShanghaiTech Crowd Counting Dataset downloaded from Kaggle, a proper environment setup is essential to ensure that the dataset can be accessed, visualized, processed, and prepared correctly for building a crowd-counting model. The environment setup includes installing the required libraries, configuring the file paths, and preparing necessary tools to handle image data and MATLAB .mat annotation files.

**Key Components of the Environment Setup Required Python Modules**

The following libraries are essential for handling images, generating density maps, visualization, model training, and deployment:

- torch – for deep learning model development
- torchvision – for pretrained models and image transformations
- opencv-python – for image processing and resizing
- numpy – for numerical computations
- pandas – for handling metadata and tabular data
- matplotlib – for plotting and data visualization
- pillow (PIL) – for loading and manipulating images
- scipy – for reading and processing .mat ground-truth files
- scikit-learn – for splitting dataset and evaluation utilities
- plotly – for interactive visualizations
- tqdm – for progress bars during preprocessing and training
- flask – for deploying the model as a web API
- streamlit – for building interactive UI applications

- twilio – for SMS/notification integration (optional)
- requests – for external API calls
- pyyaml – for configuration file handling

1.Python Installation:

A stable version of Python (such as 3.8–3.12) is required to run the scripts. This allows compatibility with scientific libraries used for data processing.

2. Library Installation:

Kaggle's ShanghaiTech dataset requires specific libraries such as:

- numpy for numerical operations ○ pandas for metadata handling ○ matplotlib for visualization
- Pillow (PIL) for loading and manipulating images ○ scipy for reading .mat ground-truth files ○ opencv-python for image preprocessing and resizing

These dependencies must be installed using pip install -r requirements.txt or individually.

3. Folder Structure Setup:

Once the dataset is downloaded from Kaggle, it must be extracted so that the directory includes folders such as:

- part_A/train_data/images, part_A/train_data/ground_truth
- part_B/test_data/images, etc.
  This ensures that scripts can correctly load images and annotations.

4.IDE and Environment Configuration:

Tools such as VS Code or Jupyter Notebook help in running code efficiently. A virtual environment (e.g., python -m venv venv) ensures clean dependency management.

Overall, this setup ensures that all tools, libraries, and dataset files are ready for exploration and preprocessing.

## 2. **Data Exploration**

Data exploration is the next crucial step after setting up the environment. According to the Kaggle dataset documentation, the ShanghaiTech dataset consists of crowd images along with .mat files containing dot-annotations marking each individual in the scene. Data exploration helps understand dataset size, annotation style, and image characteristics.

**Key Objectives of Data Exploration**

1. Understanding Dataset Parts:
    - Part A: High-density crowd images collected from the internet with heavy congestion.
    - Part B: Sparse crowd scenes captured from streets and campuses.
2. Inspecting Image Properties:
    - Resolution differences between Part A and Part B ○ Image clarity, noise levels, and brightness variations

3.Analyzing Ground-Truth Annotations:

- Each .mat file contains point-level annotations ○ These points represent individual head positions ○ Understanding this helps generate density maps later

4. Dataset Size Overview:
    - Part A: ~482 images (train + test) ○ Part B: ~716 images (train + test)
5. Visualization:
    - Displaying random images
    - Overlaying dot-annotations to understand crowd distribution
    - Checking variations across scenes
6. Identifying Issues:
    - Inconsistent lighting
    - Highly congested scenes causing annotation overlap ○ Large differences in scale between individuals

Data exploration helps determine preprocessing requirements such as resizing, normalization, and density map creation.

## 3. Data Preprocessing

Data preprocessing prepares the downloaded Kaggle dataset for training crowd-counting models. Since the ShanghaiTech dataset includes raw images and .mat files containing human head annotations, several preprocessing steps are needed to convert them into model-readable formats.

**Key Preprocessing Steps**

1. Loading and Validating Dataset Files:
    - Verifying that each image has a corresponding groundtruth .mat file
    - Checking image integrity and removing corrupted samples
2. Annotation Extraction:
    - .mat files contain dot annotations stored under nested structures
    - Extracting these points provides the coordinates of each person in the image
    - These coordinates are later used to create density maps
3. Density Map Generation Using Gaussian Filter:
    - A Gaussian filter is applied to each annotated point
    - Each point is represented by a Gaussian kernel, and summing all kernels forms the density map
    - This technique smooths out the point annotations and helps models learn spatial crowd distribution effectively
4. Handling Density Map Resizing:
    - When images are resized, density maps must also be resized
    - It is crucial to multiply the resized density map by the *scaling factor squared* to preserve the total person count
    - This ensures that resizing does not distort the total density value
5. Image Resizing:
    - Image resolutions vary widely in ShanghaiTech Part A and Part B
    - Standardizing image sizes, such as 512×512 or proportional resizing, helps stabilize training performance

6. Normalization Using ImageNet Statistics:

  - For deep CNN-based models, ImageNet normalization is applied:
    -  Mean = [0.485, 0.456, 0.406]
    -  Std = [0.229, 0.224, 0.225]
  - This aligns image distributions with pretrained backbone networks

7. Data Augmentation Techniques:
To improve generalization and avoid overfitting:

  - Random horizontal/vertical flips ○ Random cropping
  - Brightness and contrast adjustment ○ Random rotation ○

  Gaussian noise addition

  These augmentations help the model learn robust representations under different environmental conditions.

8. Train-Test Split:

  - Kaggle version already contains separate folders for training and testing
  - Ensures fair and consistent evaluation

**Purpose of Preprocessing**

- Ensure dataset uniformity
- Preserve density information during resizing
- Use Gaussian smoothing for spatial awareness
- Normalize images for pretrained networks
- Enhance model generalization through augmentation
- Prepare images and density maps for efficient training

With these preprocessing enhancements, the dataset becomes fully ready for training advanced crowd-counting models.

## 4.Model Training Process (Part A and Part B)

The ShanghaiTech Crowd Counting Dataset is divided into Part A (high-density crowd scenes) and Part B (low-density crowd scenes). Although both parts are trained using similar deep-learning procedures, the characteristics of each subset influence the preprocessing steps, model behavior, and convergence patterns.

4.1 Training Process Overview

The model training pipeline for both parts typically includes:

1. Loading preprocessed images and corresponding density maps
2. Applying transformations and normalization
3. Forward pass through the CNN-based network (e.g., CSRNet, MCNN)
4. Calculating loss between predicted and ground-truth density maps
5. Backpropagation and weight updates
6. Monitoring MAE and MSE for performance evaluation

---

4.2 Training on Part A (High-Density Scenes)

Part A contains highly congested images, making the task more complex. Training on this part requires careful handling of density maps and more robust feature extraction.

Characteristics

- Large number of individuals in each image
- High variation in scale and density distribution
- Requires stronger Gaussian kernels to represent dense regions

Training Considerations

- Smaller batch size due to high computational load
- Stronger data augmentation to generalize over complex scenes
- Careful learning rate tuning because gradients can be large due to dense annotations
- More training epochs to help the network learn fine spatial features

Training Steps

1. Load images and density maps generated with Gaussian filters
2. Apply resizing and ImageNet normalization
3. Feed input into the model
4. Compute loss using
    - MSE Loss for density regression
5. Backpropagate and update the network parameters
6. Monitor validation MAE and RMSE for early stopping

---

4.3 Training on Part B (Low-Density Scenes)

Part B contains sparse crowds with fewer people per image, making the learning process easier and faster.

Characteristics

- Fewer head annotations per image
- Clearer and less congested scenes
- Lower kernel density during map generation

Training Considerations

- Larger batch sizes can be used compared to Part A

- Simpler augmentation since images are clearer
- Fewer epochs required for convergence
- Faster learning due to less complexity

Training Steps

1. Load resized and normalized images
2. Generate density maps using Gaussian filters with appropriate sigma values
3. Forward pass through