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

Bolakonda Anvitha

Table of Contents

1. Project Overview

The “Deep Vision Crowd Monitor” project focuses on preparing data for a deep-learning model that estimates crowd density in highly congested environments.

This preprocessing phase transforms raw images and their annotations into standardized tensors that a neural network can learn from.

The overall goals of this stage are to:

- Understand the dataset through visualization.
- Convert all images into a consistent format.
- Generate reliable density maps from head-point annotations.
- Downsample maps to match CNN output structure.
- Build a PyTorch-compatible dataset for easy training.

This phase establishes the foundation for accurate crowd-count prediction in the advanced stages of the project.

2. Description of Dataset

For this milestone, the ShanghaiTech Part A dataset is used. It contains:

- Images captured in dense urban environments.
- Several hundreds to thousands of people per scene.
- Ground-truth annotations stored in .mat files.
- Each annotation file includes a list of all visible head coordinates.

Part A is challenging due to extremely crowded scenes and large variations in perspective, making it an ideal benchmark for learning crowd estimation.

3. System/Environment Setup

The processing pipeline is built using:

- Python 3.9

- **OpenCV :** computer vision toolkit for handling videos and images
  -it is used to handle video frame extraction and displaying bounding boxes or heat maps in real time.

- **SciPy** for handling **MATLAB** .mat annotation files.

- **NumPy** for vectorized numerical operations
  -it is used to manage images data as a risk and perform quick operations like normalisation and matrix multiplication.

- **PyTorch** and **torchvision** for tensor creation and normalization.

- **Matplotlib** for visualization
  -so showing crowd images density maps and training progress curve.

- **Pillow(pl):** image manipulation
  -image preprocessing before feeding them into the model.

- **Scikit-learn:** machine learning and matrix toolkit
  -to evaluate models accuracy and analyse prediction performance.

- **Plotly:** interactive visualisation library
  -you can use it in streamlit or flask dashboard to show life crowd count graphs or heat maps.

- **tqdm:** for monitoring progress
  -while training a model or looping over images tqdm will help to show progress bar.

- **Torch vision:** pytorch helper library for computer vision using it to load images apply transformation and potentially use transfer learning from pre- trained cnn.

- **Pandas :** data handling and analysis
  -might be used to track model performance log results or manage metadata like image path and predicted crowd count.

- **Flask:** lightweight backend frame
  -can be used to create a real time monitoring dashboard that display crowd count a lots and heat maps in the browser.

- **Streamlit:** fastest way to make web or data apps with python
  -it is used to build a interactive crowd monitoring dashboard using streamlit showing live frames density maps and alerts .

- **Twilio:** communication API for sending SMS calls or WhatsApp alerts
  -when an overcrowded area is detected twilio can send an automatic SMS alert to the control team.

- **Requests:** http library for making web request
  -used for API calls example sending alerts retrieving data from and online feed or integrating with web development.

- **Pyyaml:** reading configuration files
  -used to load configuration parameters examples the growth threshold value data path or model weights.

All steps are executed inside a Jupyter Notebook environment.

4. Exploration of Training Data

Before beginning preprocessing, the dataset was explored to verify structure and consistency.

Image Inspection

A few images were displayed to understand:

- Lighting and clarity
- Crowd density levels
- Range of resolutions

Annotation Verification

Each .mat file was opened and the head-point coordinates were visualized over the image.

This validated that:

- Annotation files are correctly linked
- The points align accurately with visible heads
- All images contain valid ground-truth data

This step ensures there are no missing or corrupted files.

5. Preprocessing Workflow

This section describes how every image is converted into training-ready tensors.

5.1 Loading and Inspecting Images

Images are read using OpenCV.

Since OpenCV loads images in BGR, they are converted into RGB to match the expected input format of deep-learning models.

Pixel values are scaled to the 0–1 range, making them suitable for normalization.

5.2 Reading Ground-Truth Annotation Files

Every image has a corresponding .mat file named in the format:

- GT_IMG_xxx.mat
- These files store all head-point coordinates.
- The coordinates are extracted and used to create Gaussian-based density maps.

5.3 Image Resizing (512 × 512)

To maintain uniform size across the dataset:

- Every image is resized to 512 × 512 pixels.
- All annotation coordinates are rescaled proportionally.
- Resizing ensures that both the input images and their density maps share the same dimensions, which is essential for stable CNN training.

5.4 Generating Gaussian Density Maps

A density map highlights how heavily populated each region is.

The process involves:

1. Creating a blank matrix of zeros.
2. Marking each head location with a value of 1.
3. Applying a Gaussian filter (sigma = 15).

This converts each marked point into a smooth density distribution.

The integral (sum) of the density map equals the number of people in the image.

5.5 Downsampling Density Maps

Neural networks such as CSRNet produce a feature map output that is 1/8th the resolution of the input.

Therefore, each full-resolution (512×512) density map is downsampled to:

➔ 64 × 64

To preserve the total person count, the map is scaled by:

➔ 8 × 8 = 64

This step ensures the model’s predicted density sum correctly reflects the real count.

5.6 Image Normalization

Before converting images into tensors, standard normalization is applied using ImageNet statistics:

- Mean: [0.485, 0.456, 0.406]
- Standard deviation: [0.229, 0.224, 0.225]

This step aligns the images with the distribution expected by pretrained convolutional layers.

5.7 Converting to PyTorch Tensors

Two tensors are created for each sample:

1. Image Tensor

- Shape: [3, 512, 512]
- Contains normalized RGB values.

2. Ground-truth Density Tensor

- Shape: [64, 64]
- Contains the downsampled density map.

These tensors are saved using torch.save() for efficient loading.

5.8 Building the PyTorch Dataset and Dataloader

A custom dataset class is implemented to load paired tensors:

- Input image tensor
- Ground-truth density tensor

The dataloader enables:

- Mini-batching
- Shuffling
- Efficient GPU/CPU data feeding

This structure is essential for training deep CNN models.

6. 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 the same deep-learning approach, the characteristics of each subset affect preprocessing, training behavior, and model convergence.

6.1 Training Process Overview

The model training pipeline for both parts includes:

1. Loading preprocessed images and corresponding density maps
2. Applying ImageNet normalization
3. Forward pass through the CNN-based network (CSRNet)
4. Calculating loss between predicted and ground-truth density maps
5. Backpropagation and weight updates
6. Monitoring MAE and RMSE for performance evaluation
7. Saving the best model as **csrnet_weights.pth**

6.2 Training on Part A (High-Density Scenes)

Part A contains highly crowded images, making the training more challenging. This part requires stronger spatial feature extraction and careful density map handling.

Characteristics

- Very large number of people per image
- High variation in scale and dense crowd distribution
- Requires larger Gaussian kernels during density map creation

Training Considerations

- Smaller batch size due to heavy computational load
- Stronger data augmentation to cover complex scenes
- Careful learning rate tuning because gradients can be large
- More training epochs required to learn fine features

Training Steps

1. Load images and Gaussian-filtered density maps
2. Apply resizing and ImageNet normalization
3. Feed input into the CSRNet model
4. Compute loss using
    - **MSE Loss** for density regression
5. Backpropagate and update model weights
6. Monitor validation MAE and RMSE for best model selection

6.3 Training on Part B (Low-Density Scenes)

Part B contains street-level images with fewer people, making learning easier and faster compared to Part A.

Characteristics

- Fewer head annotations in each image
- Clearer scenes with less congestion
- Smaller Gaussian kernel values during density generation

Training Considerations

- Larger batch size can be used compared to Part A
- Simpler augmentation due to clearer images
- Fewer epochs required for convergence
- Faster learning due to lower scene complexity

Training Steps

1. Load resized and normalized images
2. Generate density maps using Gaussian filters
3. Forward pass through CSRNet
4. Compute loss using MSE
5. Backpropagate and update weights
6. Monitor validation MAE and RMSE to select the best model

7. Live Webcam-Based Crowd Monitoring

The system supports real-time crowd monitoring using a live webcam feed. This mode is useful for continuous surveillance in public locations such as streets, campuses, and event venues.

Working Principle

- The webcam continuously captures video frames in real-time.
- Frames are extracted at fixed intervals to reduce processing load.
- Each frame is preprocessed (resizing and normalization).
- The processed frame is passed to the CSRNet deep learning model.

- CSRNet generates a crowd density map representing the concentration of people.
- The system computes a Crowd Density Index from the density map.
- The live density map and crowd statistics are displayed on the dashboard.

This approach enables continuous real-time monitoring with immediate alert generation if abnormal crowd density is detected.

8. Crowd Monitoring Using Uploaded Video

In addition to live input, the system allows offline analysis using uploaded video files. Supported formats include MP4, AVI, and MOV.

Working Principle

- The user uploads a video through the web interface.
- The system reads the video using OpenCV.
- The video is decoded frame by frame.
- Selected frames are processed using CSRNet to estimate crowd density.
- The results are aggregated to compute an overall Crowd Density Index.
- The final density map and crowd statistics are displayed on the dashboard.

This mode is useful for analyzing recorded CCTV footage or previously captured videos.

9. Intelligent Frame Selection from Video

Processing every frame of a video is computationally expensive. To improve efficiency and maintain accuracy, the system performs intelligent frame sampling.

Working Principle

- The total number of frames in the video is calculated.
- Frames are sampled at regular intervals (e.g., every Nth frame).

- Blurred, duplicate, or low-information frames are automatically ignored.
- Only **high-quality representative frames** are selected for analysis.
- Crowd density is estimated for these selected frames.
- Final results are computed by averaging the density values.

This technique ensures:

- Faster processing
- Reduced hardware load
- Stable and reliable crowd estimation

10. Web-Based Dashboard and Email Alert System

A **web-based dashboard** is designed using Streamlit to provide real-time visualization and alert updates. An **email alert system** is integrated using SMTP to notify authorities when unsafe crowd conditions are detected.

Dashboard Working

- Displays uploaded/live video preview.
- Shows color-coded **crowd density heatmaps**.
- Displays **Crowd Density Index** numerically.
- Indicates crowd status (Safe or Alert).
- Automatically refreshes results as frames are processed.

10.1 Email Alert Working (SMTP)

- Email addresses are stored in a local database.
- A predefined threshold is set for crowd density.
- When density exceeds the threshold:
- Alert is triggered
- Emails are fetched from the database
- Alert messages are sent via Gmail SMTP server.
- Email contains crowd details and safety warning.

This ensures rapid communication and enables preventive crowd control measures.

11. Conclusion

The preprocessing stage of the Deep Vision Crowd Monitor project converted the raw ShanghaiTech dataset into a structured, model-ready format. This phase included visualization, annotation verification, density-map generation, normalization, and dataloader creation, ensuring consistent and reliable input for training. Following preprocessing, the CSRNet model was trained to learn crowd density patterns in both high-density and low-density scenes. The training process involved forward passes, loss computation, backpropagation, and evaluation using MAE and RMSE metrics, and the best model weights were saved for deployment.

After training, the project was extended to real-world applications through live webcam monitoring and uploaded video analysis. Efficient frame selection was used to reduce computation while maintaining accurate density estimation. A web-based dashboard was developed to display crowd density maps, crowd statistics, and alert status in real time. Additionally, an SMTP-based email alert system was integrated to automatically notify registered users whenever crowd density exceeds a defined threshold, enabling timely action and improved public safety.