AI-DeepVision: Crowd Monitoring & Density Map Generation

By: Ishita Deshpande

Table of Contents

Project Description

AI-DeepVision is a research-oriented project aimed at automating crowd monitoring using computer vision and deep learning techniques. The system preprocesses the ShanghaiTech dataset to generate density maps that represent crowd distribution across images, which are essential inputs for modern crowd counting models. The project includes image preprocessing, density map generation, dataset visualization, and modular PyTorch data pipeline development. It prepares high-quality input data for models like CSRNet, MCNN, SANet, and CANNet.

Dataset Used

The project uses the ShanghaiTech Crowd Counting Dataset, which is divided into two parts—Part A and Part B—designed to capture different levels of crowd density. Part A contains highly congested scenes collected from the internet, featuring heavy crowd gatherings where people appear very close to one another. Each image is paired with a ground-truth .mat file containing exact point annotations marking each person's head. Part B, on the other hand, consists of outdoor street scenes with sparse crowds, providing cleaner and less congested visuals. The ground truth for Part B also includes point annotations that reflect head locations but with more spread-out distributions compared to Part A. These annotations are later converted into density maps, which help deep learning models learn how people are distributed spatially across different crowd densities.

Environment Setup

Setting up the environment requires installing several Python libraries that support core project functions such as numerical computation, image processing, file handling, and neural network operations. Below is a list of all required libraries along with their descriptions and functions:

Libraries Used

- **NumPy** – Provides efficient numerical operations and array manipulation, essential for handling images and density maps.
- **SciPy** – Offers scientific functions, including Gaussian filtering and spatial computations needed for density map generation.
- **OpenCV (cv2)** – Used for image loading, resizing, color conversion, and basic preprocessing operations.
- **Matplotlib** – A visualization library for plotting images, density maps, histograms, and dataset statistics.
- **PyTorch** – The core deep learning framework used for building dataset loaders, converting data to tensors, and preparing inputs for CNN models.
- **h5py** – Allows reading and writing .mat or .h5 files that store ground truth point annotations.
- **Pillow** – Used for high-level image operations such as opening, converting, and manipulating image formats.
- **pandas** – Helpful for organizing dataset statistics and producing structured data tables or summaries.
- **scikit-learn** – Offers helper utilities for preprocessing, metrics, and additional data analysis.
- **tqdm** – Provides visually appealing progress bars when processing large datasets.
- **requests** – Enables downloading files or interacting with online resources when needed.
- **PyYAML** – Used to read configuration files (such as paths, preprocessing parameters, and settings) written in YAML format.
- **Flask** – A lightweight web framework that can be used later for deploying crowd counting demos.
- **Streamlit** – Useful for creating interactive visualization dashboards for exploring images and density maps.
- **Plotly** – Provides interactive plotting functions to visualize dataset insights with better clarity.

Data Exploration

Data exploration involves visualizing sample images, annotated points, and generated density maps. This stage helps understand dataset characteristics such as crowd density variation, image sizes, and annotation patterns. Histogram plots of crowd counts reveal the level of congestion and guide model selection. Visual exploration ensures that preprocessing steps are functioning correctly before training a deep learning model.

Data Preprocessing

The preprocessing pipeline prepares both the crowd images and their corresponding ground-truth density maps so they can be used effectively by a VGG-based crowd counting model. From the image, the steps follow this sequence:

1. Load images in RGB and scale pixel values to 0–1.
2. Apply ImageNet mean–std normalization.
3. Downsample the ground-truth density map by **8**× to match CSRNet/VGG output resolution.
4. Multiply the downsampled map by **64** to preserve total person count.
5. Convert both images and ground-truth maps to PyTorch tensors.
6. Build the PyTorch Dataset and Dataloader.

Core Concepts

1. Convolutional Neural Networks (CNNs)

CNNs are neural networks designed for image-based tasks and extract spatial features through convolution operations. They use filters that scan over the image to detect patterns such as edges, textures, and shapes. With layers like convolution, pooling, and activation, CNNs progressively learn high-level features. Their hierarchical structure makes them suitable for dense prediction tasks such as crowd counting. The efficiency and feature-learning capability of CNNs form the backbone of this project.

2. VGG-16 Model (Very Important)

VGG-16 is a 16-layer deep CNN known for its simple architecture of stacked 3×3 convolutions. It is widely used as a feature extractor for crowd counting because of its strong ability to capture spatial patterns. In CSRNet, the pretrained VGG-16 frontend extracts meaningful visual features from input images. VGG-16 requires ImageNet normalization to maintain stable performance. Its deep but uniform structure ensures high-quality feature maps used by subsequent network layers.

3. CSRNet (Convolutional Neural Network for Crowd Counting)

CSRNet is a state-of-the-art model that uses VGG-16 as its frontend and dilated convolutions as its backend. The dilated convolution layers allow the network to capture broader context without increasing computational cost. CSRNet outputs a density map where the sum of pixel values equals the number of people in the image. The model is particularly effective for congested scenes due to its wide receptive

field. Its architecture is simple yet powerful for accurate crowd estimation.

4. Gaussian Density Maps

Density maps are created by placing a 2D Gaussian filter at each annotated head location. They convert point annotations into smooth spatial distributions that represent crowd density. These maps help models learn not just where people are but also how densely populated a region is. The sum of the density map equals the total number of people. Adaptive Gaussian kernels adjust sigma values using nearest-neighbor distances for more realistic density representation.

5. MSE Loss (Mean Squared Error)

MSE loss is the primary loss function used for training crowd counting models. It measures the difference between predicted and ground-truth density maps by averaging squared pixel-wise errors. This loss encourages the model to generate accurate density values at each location. Since density maps are continuous-valued outputs, MSE provides a stable gradient for learning. It directly correlates with improving counting accuracy during training.

6. Optimizers (Eg: Adam)

Optimizers update model parameters to minimize the loss function during training. Adam is widely used due to its adaptive learning rate mechanism, making training stable and efficient. It combines momentum and RMSProp concepts to handle sparse gradients effectively. In CSRNet training, Adam helps maintain consistent

convergence even with high-resolution images. Choosing the right optimizer influences training speed and final model accuracy.

7. Data Normalization Techniques

Normalization adjusts image pixel values to a common scale, improving training stability. ImageNet mean–std normalization ensures compatibility with VGG-16's pretrained weights. It shifts and scales RGB channels such that input distributions match those used during pretraining. Proper normalization avoids biased gradients and accelerates convergence. Without it, the model's feature extraction quality can degrade significantly.

8. Downsampling & Interpolation Methods

CSRNet outputs density maps at 1/8 of the input resolution due to pooling layers in VGG-16. Therefore, ground truth density maps are downsampled by 8× to match output size. Interpolation methods like bicubic interpolation ensure smooth resizing without losing important density variations. Downsampling helps reduce computation while maintaining spatial density structure. This alignment between model output and ground truth is critical for accurate loss computation.

9. KD-Tree (Used in Adaptive Gaussian Kernel)

KD-Tree is a spatial data structure used for efficient nearest-neighbor search. In density map generation, it helps compute adaptive Gaussian sigma values based on crowd locality. Dense crowd regions receive smaller sigma values, while sparse regions receive larger ones. This technique produces more realistic density maps compared to fixed

sigma kernels. KD-Tree significantly speeds up the computation for large datasets.

10. Basic Evaluation Metrics

MAE (Mean Absolute Error) measures the absolute difference between predicted and actual crowd counts. RMSE (Root Mean Squared Error) penalizes larger errors and measures overall prediction stability. These metrics are standard in crowd counting research due to their interpretability. MAE reflects average prediction accuracy, while RMSE indicates robustness. Together, they provide a comprehensive evaluation of model performance.

Summary

To date, the project has successfully progressed through all preliminary stages required before initiating model training. The work began with a detailed examination of the ShanghaiTech Dataset, covering both Part A's high-density crowd scenes and Part B's relatively sparse street-level images, along with their corresponding point-based annotations. The computational environment was systematically configured with the necessary libraries for numerical processing, image handling, visualization, and deep learning workflows. Comprehensive data exploration was conducted to understand image characteristics, annotation patterns, and overall density distribution. Subsequently, a complete preprocessing pipeline was implemented: RGB images were loaded and normalized using ImageNet mean–standard deviation values; ground truth annotations were converted into density maps, downsampled by a factor of eight, and scaled appropriately to preserve total crowd counts. Both the images and density maps were then transformed into PyTorch tensors, and an efficient dataset–dataloader structure was constructed. With the preprocessing phase now completed, the dataset is fully prepared and suitably formatted for training deep learning models such as CSRNet.