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.

Introduction To Model Training

Model training and evaluation form the core phases of any deep learning–based computer vision system. In a crowd counting application using **CSRNet (Convolutional Neural Network with Dilated Convolutions)**, the goal of training is to enable the model to accurately learn the relationship between input images and corresponding **density maps**, which represent the spatial distribution of people in the scene.

The training phase involves feeding preprocessed images and their ground truth density maps into the neural network, optimizing model parameters using a loss function and optimizer, and iteratively refining weights through backpropagation. Evaluation, on the other hand, measures how well the trained model generalizes to unseen data using error metrics such as **MAE (Mean Absolute Error)** and **MSE (Mean Squared Error)**.

This report provides a comprehensive theoretical explanation of:

- Pre-training requirements
- Training workflow
- Loss observation and tracking
- Expected learning behavior
- Epoch planning
- Model saving
- Model evaluation methodology

Model Architecture

The proposed crowd counting system is based on the CSRNet (Convolutional Neural Network with Dilated Convolutions) architecture, which is specifically designed for accurate crowd density estimation in highly congested scenes. CSRNet follows a fully convolutional network (FCN) design, meaning it does not use any fully connected layers and produces a continuous density map instead of a class label. This allows the model to perform pixel-wise regression, which is essential for estimating crowd density. The overall architecture is divided into two main components: the VGG-16 frontend feature extractor and the dilated convolution backend.

VGG-16 is a deep convolutional neural network architecture that is used in CSRNet as the frontend feature extraction network. It consists of 16 weight layers, out of which 13 are convolutional layers and 3 are fully connected layers. In CSRNet, only the convolutional part of VGG-16 is used, while the fully connected layers are removed so that spatial information is preserved for dense prediction. The network uses small 3×3 convolution filters and max-pooling layers to progressively extract meaningful visual features from input images.

The main benefit of using VGG-16 lies in its strong and stable feature extraction capability. It effectively captures low-level features such as edges and textures as well as higher-level features such as shapes and dense patterns, which are crucial for crowd estimation. Additionally, VGG-16 is often used with pretrained ImageNet weights, providing a transfer learning advantage that speeds up model convergence, improves accuracy, and reduces the risk of overfitting in crowd counting tasks.

The backend of CSRNet consists of dilated convolution layers, which expand the receptive field without reducing spatial resolution. Dilated convolutions allow the model to capture both local details and global context, which is extremely important in dense crowd scenarios where people overlap and appear at different scales. Unlike traditional pooling operations that reduce resolution, dilated convolutions preserve fine spatial details while still learning global crowd distribution patterns.

Finally, the output from the dilated convolution layers is passed through a regression layer that generates the predicted density map. Each pixel in this density map represents the estimated crowd density at that location. The total count of people in the image is obtained by integrating (summing) all pixel values in the density map. This architecture enables CSRNet to effectively handle high-density crowds, severe occlusions, perspective variations, and large scene depth, making it highly suitable for real-world crowd counting applications.

Model Training Methodology

Pre-Training Requirements

Before initiating the model training process, several prerequisites must be fulfilled to ensure a stable and meaningful learning process.

1 Preprocessed Images

Raw images must undergo multiple preprocessing steps such as:

- Resizing to a fixed resolution
- Normalization using ImageNet mean and standard deviation
- Conversion to PyTorch tensors

This ensures uniformity and numerical stability during training.

2 Full-Resolution Density Maps (.npy Files)

Ground truth annotations are converted into density maps, where:

- Each person contributes a Gaussian kernel
- The sum of all pixel values equals the total number of people in the image

These maps are stored in NumPy (.npy) format to preserve precision and ensure fast loading during training.

3 Verified Downsampling Logic

CSRNet outputs lower-resolution density maps due to pooling and stride operations. Therefore:

- The original ground truth density maps must be downsampled correctly
- The total crowd count must remain conserved after downsampling

Incorrect downsampling leads to inaccurate learning and unreliable evaluation.

4 Correct Dataset Folder Structure

A properly structured dataset ensures correct data loading:

- Images and density maps must be aligned by filename
- Separate folders for training and testing must exist
  Improper structure leads to mismatched inputs and labels.

5 GPU Environment

Training CSRNet on a CPU is extremely slow due to:

- Large image sizes
- Deep VGG-based architecture
  Using a GPU significantly accelerates:
- Forward propagation
- Backpropagation
- Weight updates

If preprocessing is incomplete, training must **not be started**, as flawed input data directly corrupts learning.

Training Notebook Workflow

The model training process is executed sequentially through structured notebook cells. Each step has a specific theoretical purpose.

1 Importing Required Libraries

Essential deep learning libraries include:

- PyTorch for model creation and training
- NumPy for numerical operations
- OpenCV / PIL for image handling
- TQDM for progress tracking

These libraries together form the backbone of the training environment.

2 Loading the Dataset

The dataset loader:

- Reads preprocessed images
- Loads corresponding density maps
- Applies batching using a DataLoader
- Enables shuffling for better generalization

Batching improves memory efficiency and stabilizes gradient updates.

3 Loading the CSRNet Model

CSRNet is a fully convolutional neural network with:

- A VGG-16 frontend for feature extraction
- A dilated convolution backend for enlarging receptive fields

Dilated convolutions allow the model to:

- Capture both local and global crowd patterns
- Preserve spatial resolution without excessive pooling

Loading CSRNet initializes the network architecture and prepares it for learning.

4 Defining Optimizer and Loss Function

The learning process is guided by two key components:

I] Loss Function

Loss represents the numerical measure of error between the model's predicted output and the ground-truth target values. In crowd counting, the loss is typically computed using Mean Squared Error (MSE) between the predicted density map and the actual density map. A higher loss indicates poor prediction accuracy, while a decreasing loss shows that the model is learning effectively. Batch loss refers to the loss calculated for a single batch, whereas epoch loss is the average loss over all batches in one epoch and is used to monitor overall training performance.

Typically Mean Squared Error (MSE) is used to compute the difference between:

- Predicted density maps
- Ground truth density maps

It penalizes large deviations more heavily, ensuring accurate density estimation.

II] Optimizer

Optimizers like Epoch:

An epoch is defined as one complete pass of the entire training dataset through the neural network. During one epoch, the model processes all training images in batches and updates its internal weights after each batch using backpropagation. Multiple epochs are required so that the model can repeatedly see the data and gradually improve its predictions. As the number of epochs increases, the model learns more refined patterns, leading to better generalization if overfitting is controlled.

The optimizer directly affects training speed and stability.

5 Starting the Training Loop

The training loop consists of:

1. Forward pass through the model
2. Loss computation
3. Backward pass (gradient calculation)
4. Weight update using the optimizer

This loop repeats for all batches across all epochs, gradually refining the model parameters.

Observations During Training

Monitoring training behavior is critical to detect failures or inefficiencies.

1 Batch Loss

Each batch prints:

- The instantaneous learning error
- Helps detect exploding gradients or NaN values early

2 Epoch Loss

The average loss across all batches in one epoch indicates:

- Overall learning progress
- Stability of the training process

3 Key Stability Indicators

During training, the following conditions must hold:

- Loss must not be NaN or infinity
- Training should not freeze midway
- GPU memory should remain stable
- No unexpected runtime errors should occur

CSRNet typically shows a slow and steady loss reduction, which is expected for dense regression tasks like crowd counting.

Tracking Loss Values

Two important logs are maintained:

1 Batch Loss List

- Tracks loss at every training step
- Used for short-term stability analysis

2 Epoch Loss List

- Tracks performance across full dataset passes
- Used to generate learning curves

This allows:

- Learning trend comparison
- Model performance benchmarking
- Debugging unstable training behavior

Expected Loss Pattern

A healthy training process follows this trend:

- Initial loss is high
- Gradual reduction over time
- Minor fluctuations due to batch randomness
- No sudden sharp increase
- No long flat region indicating stagnation

If the loss:

- Increases rapidly → learning rate may be too high
- Stays constant → model may be underfitting or stuck
- Drops too quickly → risk of overfitting

A smooth downward slope confirms correct learning behavior.

Epoch Planning and Training Duration

CSRNet requires long training cycles due to:

- Complex spatial patterns
- High-density crowd variations

Recommended epoch counts:

- 50 epochs → early insights
- 100–150 epochs → reliable learning
- 300 epochs → optimal convergence

Early stopping should only be applied if:

- Loss converges fully
- Overfitting becomes evident

Saving the Trained Model

At the end of training, the learned parameters are stored in:

csrnet_weights.pth

This file contains:

- All trained network weights
- Learned filters for crowd density estimation

This file is:

- Used for inference
- Required for evaluation
- Essential for deployment

Deleting this file means:

- Training must be restarted from scratch

Hence, it must be securely backed up.

Model Evaluation

Evaluation determines how well the trained CSRNet model performs on unseen test data.

1 Purpose of Evaluation

- Measure generalization ability
- Detect overfitting or underfitting
- Quantify real-world performance

2 Evaluation Metrics

Mean Absolute Error (MAE)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Measures average absolute difference between:

- Ground truth count
- Predicted count

Lower MAE means better accuracy.

Mean Squared Error (MSE)

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Penalizes large mistakes more heavily, representing model robustness.

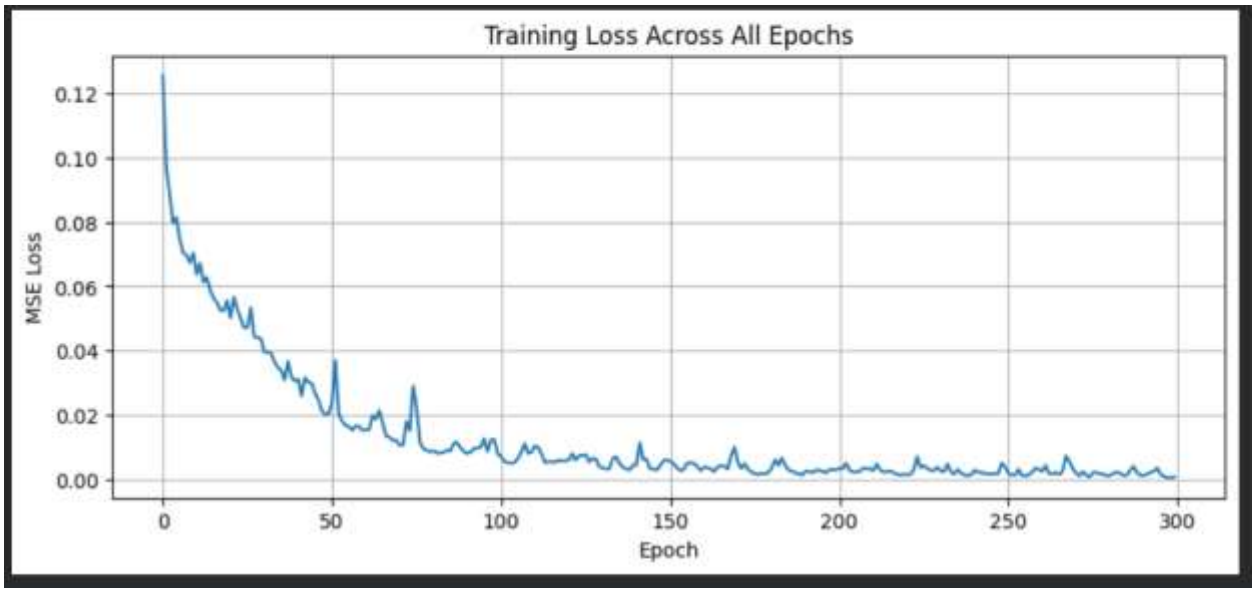
3 Evaluation Workflow

1. Load trained model weights
2. Disable gradient computation
3. Run test images through CSRNet
4. Generate predicted density maps
5. Integrate predicted maps to get person count
6. Compare with ground truth count
7. Compute MAE and MSE

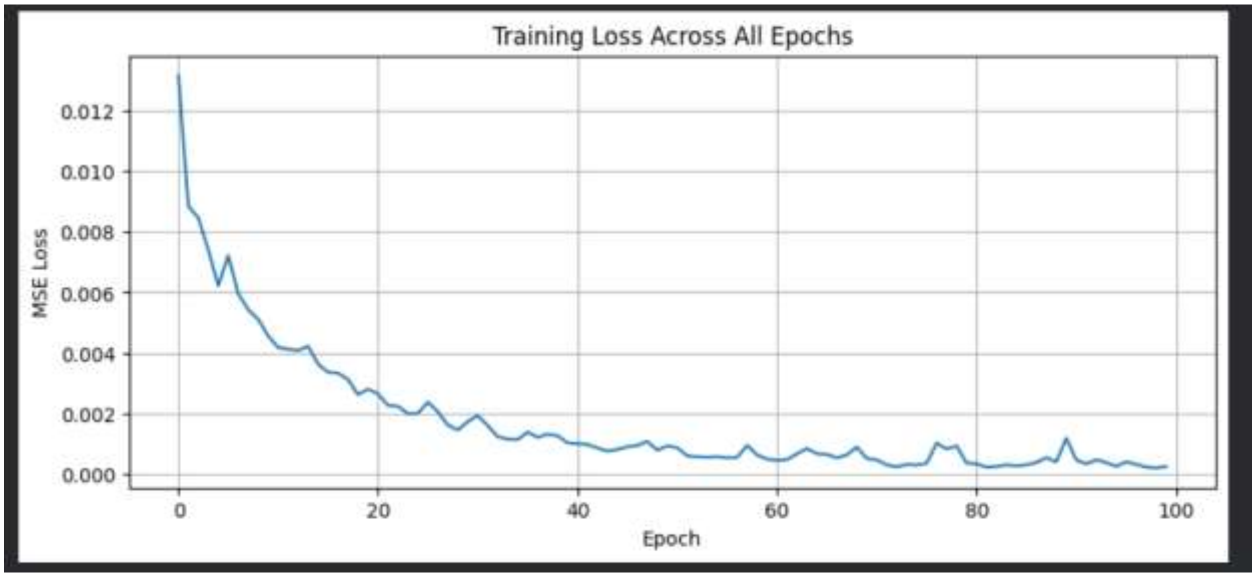
4 Interpretation of Results

- Low MAE & MSE → High prediction reliability
- High MAE → Poor generalization
- Large MSE–MAE gap → Inconsistent predictions

PART A



PART B



Conclusion

The training and evaluation of CSRNet form the backbone of the crowd counting system. Proper preprocessing, verified downsampling, correct dataset structure, and stable GPU execution ensure successful training. Continuous monitoring of batch and epoch losses is essential to prevent learning failures. The trained model, stored as csrnet_weights.pth, represents the final learned knowledge and must be preserved for evaluation and deployment. Model performance is objectively measured using MAE and MSE, which reflect real-world prediction accuracy and robustness.

A well-trained CSRNet with stable loss behavior and low evaluation error confirms the successful implementation of a deep learning–based crowd counting solution.