Final Round: Assignment

CSRNet PyTorch Implementation - by Mohammed Haseeb Ahmed

Introduction

The objective of this project is to implement CSRNet (Convolutional Neural Network for Crowd Counting) using PyTorch. CSRNet is a deep learning architecture tailored for accurately estimating crowd density and counting, especially in dense crowd settings. It uses a combination of convolutional layers and dilated convolutions to efficiently capture high-level features, achieving state-of-the-art performance on various crowd-counting benchmarks. This report provides a summary of the approach, key steps, results, and observations encountered during the implementation of CSRNet using PyTorch on the Shanghai Dataset.

Approach

1. Dataset and Data Preparation

- **Loading and Organizing Data:** The dataset includes images paired with `.mat` files containing annotations of each person's location as a set of coordinates. These annotations serve as the ground truth for density maps.
- **Density Map Generation:** Each annotation point in the image is represented as a Gaussian peak in the density map. The Gaussian function is used to smooth each person's location, helping the model recognize and count clusters of people. The sigma value for Gaussian blur is carefully chosen to ensure peaks are appropriately sized for counting, neither too sharp nor too diffused.
- **Downsampling:** CSRNet requires density maps at 1/8th the original image resolution to match the downsampling in the network. Downsampling is done after generating the density map, and the density values are scaled to maintain total counts, ensuring consistency between training and prediction.
- **Data Augmentation and Transformation:** The dataset images undergo standard image normalization using mean and standard deviation values that align with ImageNet training standards. This step, coupled with tensor conversion, prepares the data for feeding into CSRNet while mitigating overfitting.

2. Dataset and Data Preparation:

CSRNet's architecture consists of two main parts:

- **Frontend (VGG-16 Layers)**:
  - **Feature Extraction**: The first 23 layers of VGG-16 (up to the third convolutional block) act as a feature extractor. These layers focus on capturing low- and mid-level spatial features in the images, which are critical in distinguishing individuals and crowd clusters.
  - **Transfer Learning**: Using a pre-trained VGG-16 model on ImageNet helps leverage learned visual patterns, allowing CSRNet to perform well without extensive training. The fully connected layers of VGG-16 are discarded, as CSRNet focuses on spatial density maps rather than classification.
- **Backend (Dilated Convolutional Layers)**:
  - **Dilated Convolutions**: CSRNet's backend replaces the traditional pooling layers with dilated convolutions, allowing the model to capture a larger receptive field without spatial resolution loss. The dilation rate doubles in successive layers (2, 4, 8, and 16), enabling the model to focus on larger context areas crucial for dense scenes.
  - **Output Layer**: A final convolutional layer with a 1x1 kernel produces a single-channel output density map, where each pixel value indicates the estimated density at that location.
- **Network Configuration**:
  - The frontend and backend combined create a smooth transition from low-level image features to high-level density information. Each dilated convolutional layer in the backend is followed by a ReLU activation, which helps in maintaining non-linearity and stable training.

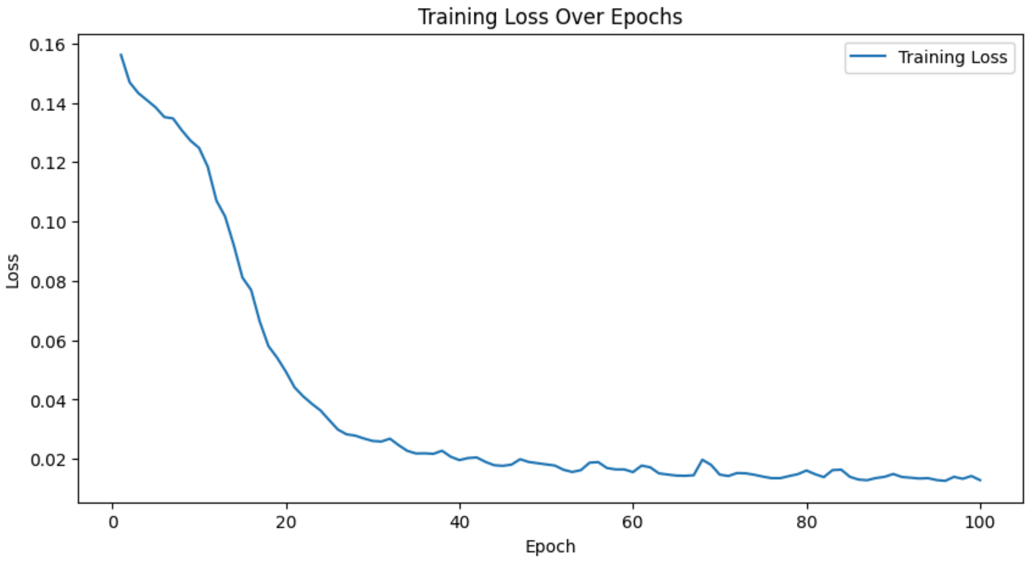
3. Training Process:

- **Loss Function:**
  - **Mean Squared Error** (MSE) loss is used to compare the predicted density map with the ground truth density map. MSE penalizes larger deviations more heavily, which is beneficial when handling density maps with extreme values (e.g., high-density clusters).
- **Optimizer and Learning Rate:**

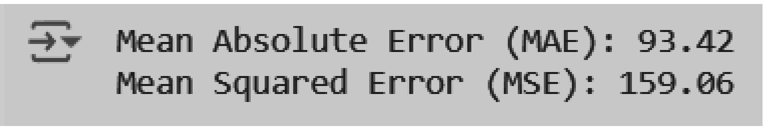
  - **Adam** optimizer with a learning rate of 1×10 ^ −5 is used, providing efficient learning rate adaptation during training. This setting allows smooth convergence, especially important in complex tasks like crowd counting, where high learning rates may cause oscillations in loss.
- **Training Loop:**
  - The model is trained for 100 epochs, with each epoch involving:
    - Forward Pass: Passing the image through the frontend and backend to generate a density map.
    - Backpropagation: Calculating loss and updating model parameters through gradients.
    - Periodic Logging: Training loss is recorded every 10 steps for tracking and adjusting if needed.

Results

- **Shanghai Dataset part_A:**
  - **Training loss vs epochs:**

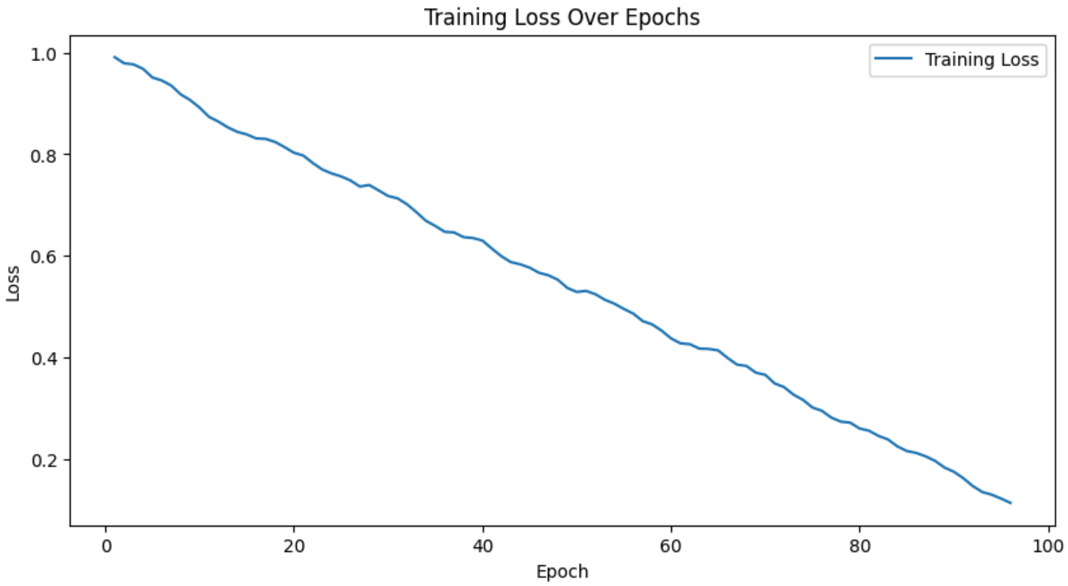


  - **Metrics:**

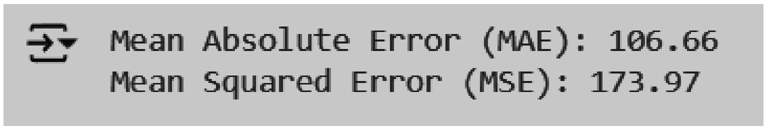


```
Mean Absolute Error (MAE): 93.42
Mean Squared Error (MSE): 159.06
```

- **Shanghai Dataset part_B:**
    - **Training loss vs epochs:**



    - **Metrics:**



```
Mean Absolute Error (MAE): 106.66
Mean Squared Error (MSE): 173.97
```

Observations

- **Dilated Convolutions**: The dilated layers enhanced the model's receptive field, essential for capturing information in dense regions.
- **Data Augmentation**: Normalization helped reduce overfitting on the smaller Part B dataset.
- **Training Stability**: The chosen learning rate and optimizer stabilized training, and training losses consistently decreased over epochs. Adam optimizer performed better than SGD on the part_A dataset.

Resources

Dataset - Shanghai_Dataset

Notebook - CSRNet_PyTorch_Github

Conclusion

The CSRNet implementation successfully replicates results consistent with the original paper, demonstrating CSRNet's capability in dense crowd counting. Further optimizations could explore alternate backbones or data processing techniques to enhance performance in different crowd densities.