Crowd Counting from Drone Footage Using CSRNet

Lauretta Machine Learning Engineer Take-Home Test

Introduction

In surveillance, event monitoring, and public safety applications, crowd counting is essential. Even though traditional object detection models, such as YOLO, are good at identifying individual objects, they frequently have trouble in crowded or dense areas. To solve this issue, I employ a cutting-edge model created especially for crowd counting, called CSRNet (Convolutional Neural Network-based Spatially Regularised Network). This project aims to use CSRNet to count and estimate the number of people on a bridge in drone footage.

Problem Statement

The task was to detect the number of people on a bridge in a drone video. The video footage, which runs from 0:14 to 0:32 seconds, shows the drone taking several pictures from an aerial perspective. The main difficulties are:

1. **Small size of people in the footage**: Due to the high altitude of the drone, people appear very small in the video, which makes detection difficult for conventional models like YOLO.
2. **Crowd density:** The bridge is densely populated, which further complicates the accurate detection of individuals using object detection techniques.
3. **Duplicate frames:** Since the video was shot at a high frame rate, some frames may be very similar, which introduces redundancy in processing.

To overcome these challenges, I use CSRNet for accurate crowd counting and implement a system that:

- **Extracts frame**s from the relevant portion of the video.
- **Processes the frames** using CSRNet to estimate the crowd count.
- **Removes duplicate** frames and selects representative frames for counting.
- **Aggregates** the results for an overall count of people on the bridge.

Approach

- ## Video Preprocessing

  The first step is to extract frames from the drone footage for the time window between 14 seconds and 32 seconds. Using OpenCV, I achieve this by:

  - Downloading the video from YouTube using yt-dlp and renaming it for ease of access.
  - Extracting frames at 1-second intervals from the video by setting the correct frame rate (frames per second, FPS). Removing duplicate frames by processing every 25th frame (roughly 1-second intervals) which is the fps of the given video, thus ensuring that I do not process frames with minor differences.

- ## Model Selection and Setup

  Why CSRNet?

  I selected CSRNet (Convolutional Neural Network-based Spatially Regularized Network) for this project because:

- **CSRNet specializes in crowd counting**: Unlike traditional object detectors, CSRNet uses density maps to estimate the number of people in an image, which makes it more robust in counting small, overlapping objects like people in crowded scenes.
- **Pre-trained weights**: CSRNet comes with pre-trained weights on popular crowd-counting datasets like ShanghaiTech, which helps bypass the expensive process of training the model from scratch.
- **Handling sparse and dense crowds**: CSRNet is effective in both sparse and dense scenes, making it ideal for drone footage where crowd distribution may vary.

  Model Setup

  The model architecture consists of a VGG-16 backbone for feature extraction (with the fully connected layers removed) and dilated convolution layers to generate the density maps. The process involves:

  - Loading the **pre-trained weights** into the CSRNet model (file: weights.pth).
  - **Preprocessing the frames** (normalizing and resizing) before feeding them into the model.
  - Estimating the **crowd count for each frame** using the predicted density map.

Frame Processing and Crowd Counting

Once the model is set up, I perform the following steps:

- **Pass each extracted frame** through the CSRNet model.
- **Generate a density map** for each frame. The density map represents the likelihood of people being present in different regions of the image.
- **Sum the values in the density map** to estimate the total number of people in that frame.

Post-processing and Aggregation

Since crowd size estimates fluctuate between frames due to small variations in the scene, I employ the following strategy:

- **Calculate the maximum count, mean count, and median count** from every 5 consecutive frames (after sampling them 1 frame per second) and store it in a list. This step helps in reducing noise by focusing on frames with the highest crowd density in small time windows. Since, the crowd is not moving at a fast pace and the camera is too far away, most of the people remain the same in 5 consecutive samples. This makes it easier to estimate the total number of people.
- **Sum the max, median, and mean counts** from these groups to get a closer estimate of the total crowd size during the specified time frame.
- Following is the output:

```
Mean:  1122

Median:  1109

Max:  1353
```

Results

The approach yielded a reliable crowd count throughout the first few seconds of the drone footage. Key highlights:

- The model successfully handled small, densely packed objects (people), which traditional object detectors struggle with.
- The use of CSRNet's density maps allowed for smooth and accurate counting, even in cluttered scenes.
- By removing redundant frames and aggregating counts, I ensured that noise in individual frames was minimized.

- However, the model fails significantly as the bridge comes into view as the density of the people increases as well as the size of the people also gets smaller in the video.

Final Crowd Count

The final estimated count of people on the bridge in the 18-second window was

1353, calculated by summing the maximum crowd count
1109, calculated by summing the median crowd count
1122, calculated by summing the median crowd count

from representative frames over the time span.

Conclusion

In this project, I demonstrated the effectiveness of using CSRNet for crowd counting in drone footage, a challenging scenario where individuals appear very small and densely packed. By selecting representative frames, removing duplicates, and using CSRNet's density map-based approach, I obtained a reliable estimate of the number of people on the bridge.

A people-tracking approach would have been a better alternative for the given problem as I are dealing with a video here. Also, a more fine-tuned YOLO8 model or CSRNet model might be able to alleviate the issue that this model faced for the people on the bridge.

Also, using SSIM for better removal of duplicate frames could be a better approach than the one I took.

Proposed Method for Validation

Manual Counting from Sampled Frames:

I would manually count the number of people in selected frames from the video and compare that with the predictions from my **CSRNet model**. To do this, I could choose a subset of frames (e.g., every 5th frame), count the number of people in each frame, and compare the results.

I would calculate metrics like **Mean Absolute Error (MAE)** and **Mean Squared Error (MSE)** to quantify the difference between my manual counts and the model's predictions.

Steps:

- I would extract a set of sample frames, such as from the 14th, 18th, 22nd, and 26th seconds.
- For each frame, I'd manually count the number of people and record the actual count.
- I would then compare my manual counts with the predicted values from the model.

Example Validation Process

1. **Sample Selection**:
    - Sample frames at key time intervals (e.g., at 14s, 18s, 22s, and 26s).
    - Manually count the number of people in each frame using simple visualization tools (like OpenCV).
2. **Model Predictions**:
    - Run the selected frames through CSRNet and get the crowd count estimates for each frame.
3. **Comparison**:
    - Use the formulas for MAE and MSE to calculate the error between the manually counted values and the predicted values.
4. **Interpretation**:
    - A lower MAE indicates that the model is performing well and that the predictions are close to the actual counts.
    - A high MSE might indicate that the model is making large errors in specific frames and might require further tuning or improvements.