Crowd Counting in Restaurants Using Computer Vision

Aaditya Gole Daksh Rajesh

Project Code

Contents

Abstract

Crowd counting is a significant problem in computer vision, particularly in dynamic environments like restaurants where both customers and employees are present. This report presents a structured approach to solving the problem of accurately counting people (customers and employees) in a restaurant setting using video footage. The task is performed using machine learning models and computer vision techniques, iteratively improving the results to overcome challenges such as bounding box flickering (leading to an unstable count), over-detection, and distinguishing between customers and employees. Two main approaches were tested: the YOLOv8 model and a Faster R-CNN model with Non-Maximum Suppression (NMS). The report explains the methodologies, challenges faced, and how the issues were progressively addressed.

1. Problem Statement

The primary objective of this project is to accurately count the number of people present in a cafe in real-time, while excluding the employees from the count. This involves distinguishing between customers and employees in a dynamic environment using video footage. The challenge is to ensure that the count is stable and accurate, despite the presence of overlapping bounding boxes, flickering counts, and the need to differentiate between customers and employees.

2. Introduction

Crowd counting is a critical task with applications in domains such as retail management, safety monitoring, and resource optimization. The goal of this project is to count the number of people accurately in a restaurant video, distinguishing between customers and employees, and minimizing flickering or over-counting caused by overlapping or multiple bounding boxes.

The primary challenges include:

- **Overlapping bounding boxes:** causing inaccurate counts.
- **Flickering counts:** due to frame-to-frame inconsistencies.
- **Distinguishing customers from employees:** in the video provided.

We address these challenges through iterative refinement using machine learning models and techniques such as supervised polygon masking and suppression mechanisms.

3. Methodology

This problem was solved in three iterations. In each iteration, we addressed the challenges faced in the previous step.

3.1. YOLOv8 Pretrained Model

3.1.1. YOLO Architecture

YOLO (You Only Look Once) is a real-time object detection model known for its speed and simplicity. Unlike region-based methods, YOLO treats object detection as a single regression problem. It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell simultaneously.

Key components of YOLO architecture include:

- **Convolutional Backbone:** Extracts features from the input image.
- **Unified Detection:** All predictions (class, location, and confidence) are generated in one pass through the network.

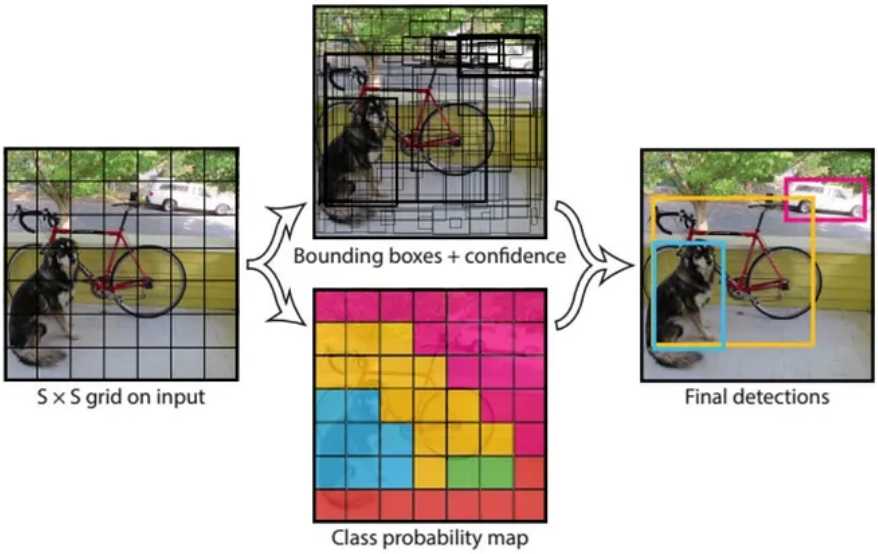

Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Figure 1: How yolo detects images).

Advantages:

- High inference speed, suitable for real-time applications.
- End-to-end training, reducing complexity.

Challenges:

- Struggles with detecting small objects in crowded scenes.
- Generates multiple overlapping bounding boxes, requiring post-processing such as Non-Maximum Suppression (NMS) for refinement.

3.1.2. Implementation and Results

The YOLOv8 pretrained model was selected for object detection due to its speed and efficiency. The methodology and outcomes are as follows:

- **Detection and Counting:** While the YOLOv8 model was able to successfully detect a person when present, it also provided multiple bounding boxes for each person, which caused over-counting. The count flickered between frames due to inconsistencies in bounding box generation.

Challenges Faced:

- Over-detection: Multiple bounding boxes on a single individual.
- Flickering Counts: Frame-to-frame inconsistencies in detection.
- Misclassification: Employees detected within the polygon, leading to customer miscounts.

3.2. Polygon-Based Zone Restriction

3.2.1. Implementation and Results

To minimize detections of employees and focus on customers, a polygonal area was defined to specify the "customer area" within the video frame. This zone was expected to contain only customers, as employees typically remained outside this area. The implementation involved using supervision to create the polygonal zone and restrict detections within it.

However, this approach had limitations. Employees, such as cleaning staff, would occasionally enter the customer area, leading to false-positive detections. This made it challenging to accurately distinguish between customers and employees based solely on their location in the frame.

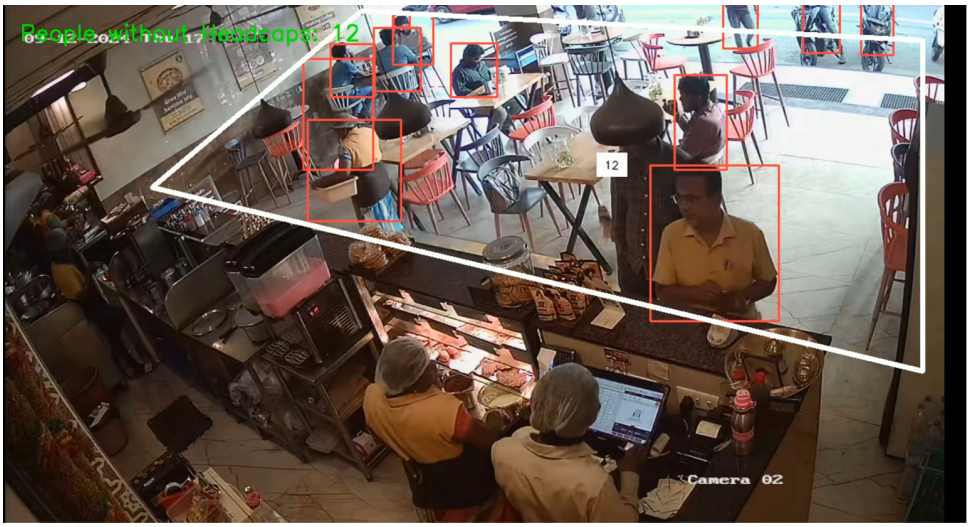

Figure 2: Detection using polygon-based zone restriction (placeholder image).

3.3. Employee Headcap Detection and Subtraction

3.3.1. Implementation and Results

To effectively distinguish between customers and employees, an additional model was employed to detect employees based on their distinctive headcaps, which all employees wear in the video. This approach involved two main steps: detecting all individuals in the frame and then identifying and subtracting those wearing headcaps to isolate the count of customers.

Headcap Detection Methodology The headcap detection was implemented using a custom-trained YOLOv8 model specifically designed to recognize the unique headcaps worn by employees. This model was trained on a dataset comprising images of employees with and without headcaps to ensure accurate differentiation. By leveraging this specialized model, the system can reliably identify employees based on their headgear, even in dynamic and crowded environments.

Detection and Subtraction Process The detection process operates in two stages:

1. **People Detection:** A pre-trained YOLOv8 model detects all individuals in each video frame.

2. **Employee Detection:** The custom-trained YOLOv8 model identifies employees by detecting their headcaps.

By subtracting the detections from the employee model from the total people detected, the system isolates the count of customers. This subtraction ensures that only customers are counted, providing an accurate representation of the number of patrons in the cafe.

Suppression Techniques to Reduce Flickering To minimize the flickering of bounding boxes caused by transient detection inconsistencies, two key suppression techniques were employed:

- **Temporal Averaging:** A buffer was maintained to store the counts from the last 10 frames. By calculating the average count over these frames, transient fluctuations are smoothed out, resulting in a more stable count.
- **Non-Maximum Suppression (NMS):** Applied to the detections to eliminate redundant overlapping bounding boxes. NMS retains the bounding box with the highest confidence score while suppressing others that overlap significantly, thereby reducing multiple detections of the same individual.

These techniques work in tandem to ensure that the bounding boxes remain consistent across frames, providing a reliable count of both customers and employees.

Buffer Implementation for Averaging Counts The use of buffers for averaging counts over multiple frames is crucial in stabilizing the detection output. The 'deque' data structure from Python's 'collections' module was utilized to efficiently manage the frame counts. By maintaining a sliding window of the most recent 10 frames, the system calculates the mean count, thereby mitigating the impact of sporadic detection errors an d ensuring a smooth and consistent count display.

Final Implementation Workflow The overall workflow for processing the video frames is as follows:

1. **Load Models:** Initialize both the people detection model and the employee headcap detection model.
2. **Process Frames:** For each frame in the video:
   (a) Detect all people using the pre-trained YOLOv8 model.
   (b) Detect employees using the custom-trained YOLOv8 model.
   (c) Subtract employee detections from total people detections to isolate customers.
   (d) Update buffers with the current counts.
   (e) Calculate average counts to stabilize the output.
   (f) Apply Non-Maximum Suppression to reduce overlapping bounding boxes.
   (g) Annotate the frame with bounding boxes and count overlays.
3. **Save Processed Video:** Write the annotated frames to the output video file.

This structured approach ensures that the system not only accurately counts the number of customers but also maintains stability and reduces visual inconsistencies in the bounding box detections. The implementation details can be found in the Python notebook available in the GitHub repository.

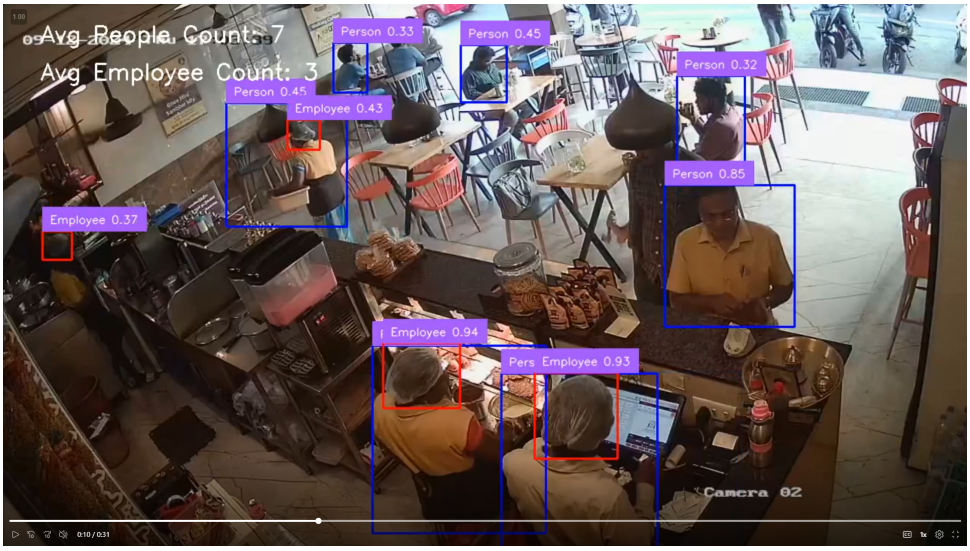

Figure 3: Final detection with employee subtraction and stabilized counts.

3.4. Challenges Addressed

- **Distinguishing between employees and customers:** By detecting employees based on their headcaps f(from the other problem statement) and subtracting them from the total people detected.
- **Flickering Counts:** Flickering solved by implicit NMS and a confidence threshold. Stabilized the count by averaging over the last 10 frames and applying Non-Maximum Suppression to reduce bounding box inconsistencies.

4. Conclusion

This project highlights the iterative refinement required to tackle real-world challenges in crowd counting. By implementing polygon-based zone restriction and employee headcap detection, we significantly improved detection stability and accuracy. Future work will focus on enhancing model generalization and addressing occlusions to further improve the system's robustness.

5. References

- Lin, T.-Y., et al. "Faster R-CNN with Feature Pyramid Networks." *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- Ani Agarwal, ***YOLO explained***