When & Where:
Fine-Tuning Faster R-CNN for Person Detection in Steep Café

Dylan Miner, Katie Hur

Code: https://github.com/katiehur5/cafe-people-detection.git

CPSC 4800: Introduction to Computer Vision

Instructor: Alex Wong

Final Project Report - December 15, 2025

Introduction

Imagine the following scenario. You are a Yale student during Reading Week, walking in a torrential downpour to work at the best (and only) study spot on Science Hill with food and coffee: Steep Café. You arrive in the afternoon to find the café fully occupied.

Understanding and modeling human presence in indoor public spaces is a foundational problem in computer vision. Applications range from occupancy estimation to safety monitoring. Cafés, in particular, present a challenging yet realistic setting for people detection due to variations in lighting conditions throughout the day, frequent occlusions by tables and chairs, cluttered backgrounds, and changes in human pose and scale. In this particular context, subjects are expected to be exhibiting various atypical poses that significantly diverge from a standing figure, from sitting with legs crossed to lying down (sometimes asleep). Accurate and precise detection of people in such environments is a prerequisite for higher-level tasks such as crowd counting and activity analysis.

Recent advances in deep learning have significantly improved object detection performance with the help of large-scale datasets and high-capacity convolutional neural networks. However, models trained on such generic datasets often struggle when deployed in domain-specific environments with a distinct camera viewpoint, illumination, and scene layout.

The goal of this project is to fine-tune a pre-trained object detection model to detect people in one particular café environment. Understanding real-time crowd density could help students and staff manage wait times, improve space usage, and enhance overall campus efficiency. Moreover, the transfer learning approach saves a substantial amount of training time and computational resources. We will use a limited custom dataset, consisting of images that

capture variations in lighting conditions, crowd density, and occlusion patterns. We will annotate images with bounding boxes around each visible person to enable supervised fine-tuning of an existing detection model architecture. By focusing on only one semantic class, “people,” this project isolates the effects of domain adaptation and fine-tuning without the added complexity of multi-class detection.

This project will imitate modern object detection pipelines: dataset preparation, annotation, train-validation-test splitting, model fine-tuning, and quantitative evaluation. Performance will be evaluated using standard metrics such as mean Average Precision (mAP). By demonstrating how a pre-trained detector can be adapted to our specific use case, this work highlights both the strengths and limitations of fine-tuning.

Related Work

Our project sits at the intersection of person detection and crowd counting, focusing on detection-based counting in a constrained indoor environment. While person detection aims to localize and classify individuals within an image, crowd counting focuses on estimating the total number of people present in a scene. Relevant prior work spans general object detection frameworks, specialized approaches for detecting people, and broader crowd counting methodologies. In this section, we discuss the evolution of object detection architectures, specialized person detection techniques, and the various paradigms for crowd counting that inform our approach.

Object Detection Frameworks

Person detection is a fundamental task in computer vision with applications ranging from occupancy estimation to human-computer interaction. Modern approaches generally build off of general object detection frameworks that aim to localize and classify multiple objects within an image.

Early deep learning approaches to object detection adapted CNNs to this task. The R-CNN framework was among the first to successfully introduce region-based detection by extracting region proposals and processing each region independently using a CNN [1]. Although effective, this approach was computationally expensive due to repeated feature extraction.

Fast R-CNN improved efficiency by computing convolutional feature maps only once per image and sharing them across region proposals [2]. However, it still relied on external region proposal methods like selective search which limited speed and prevented end-to-end training. Faster R-CNN addressed this limitation by integrating region proposal directly into the neural network through a Region Proposal Network (RPN) [3]. The RPN generates candidate regions directly from shared convolutional features. This architecture significantly improved detection accuracy and inference speed, making it a strong baseline for tasks that require precise object localization like our project.

Specialized Person Detection Methods

Beyond general object detectors, several specialized approaches have been developed specifically for person detection. The Histogram of Oriented Gradients (HOG) detector was a pioneering method that used hand-crafted features and achieved strong results on pedestrian detection benchmarks [4]. The Deformable Parts Model (DPM) extended this work by modeling people as collections of deformable parts [5].

More recent deep learning approaches include specialized architectures like Pedestrian Detection Networks (PDNs) and adaptations of single-shot detectors (SSD, YOLO) optimized for person detection [6]. These methods often incorporate techniques such as multi-scale feature fusion, attention mechanisms, and specialized data augmentation strategies to handle the challenges inherent in detecting people across varying scales, poses, and occlusion levels.

Crowd Counting

While person detection focuses on identifying individual instances, the related problem of crowd counting aims to estimate the number of people present in an image or video frame. Works in the field of crowd counting are generally categorized into three main approaches: detection based, regression based, and density estimation based methodologies. Chavan and Purohit (2023) survey these methods [7].

1. **Detection-based methods** count individuals by explicitly detecting each person, offering strong interpretability and awareness. These methods apply object detectors like Faster R-CNN or YOLO to localize and count people. While they provide precise localization and work well in low-to-moderate density scenarios, their performance diminishes in extremely dense or highly occluded scenes where individual people become difficult to distinguish.

2. **Regression-based methods** learn a direct mapping from image features to crowd counts without explicitly localizing individuals. Early approaches used hand-crafted features combined with regression models, while modern methods employ deep CNNs to predict counts from learned representations. These methods are computationally efficient but sacrifice spatial information about individual locations.
3. **Density estimation-based methods** represent the current state-of-the-art for high-density crowd counting. These approaches generate density maps where each person is represented as a Gaussian blob, and the total count is obtained by integrating over the density map.

For our cafe monitoring application, we adopt a detection-based approach using Faster R-CNN. While density estimation methods might offer better performance in extremely crowded scenarios, the moderate density typical of cafe environments, combined with our need for individual localization, makes detection-based counting more appropriate.

Methodology

Our methodology mimics modern object detection pipelines: dataset construction, annotation, preprocessing and augmentation, model fine-tuning via transfer learning, and quantitative evaluation.

- **Dataset construction:** The details of the data we used is outlined later in the data section of our report.
- **Model architecture choice:** We adopted Faster R-CNN as our base detection architecture due to its strong localization accuracy and robustness in cluttered scenes. The two-stage detector consists of three main components: (1) a backbone CNN that extracts feature maps from input images, (2) a RPN that generates object candidate regions, and (3) a detection head that classifies bounding boxes for each proposal.
  Our implementation uses the `fasterrcnn_resnet50_fpn` model provided by the TorchVision library, which combines a ResNet-50 backbone with a Feature Pyramid Network (FPN) that enables multi-scale feature fusion (crucial for detecting people at different distances).

- **Transfer Learning and Fine-tuning:** Rather than training the detector from scratch, we employ transfer learning by initializing the model with weights pre-trained on the COCO dataset. COCO contains over 200,000 labeled images and extensive annotations for people across diverse scenes and poses. These pretrained weights provide strong low-level and mid-level feature representations that generalize well to new tasks.
  To adapt the model to our single-class detection task, the original COCO classification head (which predicts 80 object categories) was replaced with a new predictor that distinguishes between "person" and background. All model parameters, including the backbone, RPN, and detection head, were fine-tuned.
- **Objective function and optimization:** The model was trained by minimizing the standard Faster R-CNN multi-task loss, which jointly optimizes classification and localization accuracy. The classification loss is implemented as cross-entropy loss over object and background classes, while the localization error is penalized using smooth L1 loss on bounding box coordinates.
  Optimization was performed using stochastic gradient descent (SGD) with momentum. We optimized all trainable parameters using a learning rate of 0.005, momentum of 0.9, and weight decay of 0.0005. Training was conducted for 10 epochs with a batch size of 2. The relatively small batch size reflects the high memory requirements of two-stage object detectors and ensures compatibility with available GPU resources. The model took approximately six hours to train. Model checkpoints were saved after each epoch to preserve intermediate weights and enable reproducibility.
- **Evaluation:** We evaluated our model's performance on the test set using both qualitative and quantitative metrics. Qualitative evaluation entailed visual inspection of predicted bounding boxes overlaid on test images, assessing localization quality, false positives, and missed detections.
  Quantitative evaluation was conducted using the COCO evaluation protocol and `pycocotools` library. The primary metric reported is Average Precision (AP) at a certain Intersection over Union threshold, which in our case was 0.50 (AP@0.5). In other words, the model considers a detection correct if the predicted bounding box overlaps the ground truth by at least 50%. Precision and recall was computed across object size to analyze performance variation for small, medium, and large individuals.

Data

The Faster R-CNN model used in this project was pretrained on the COCO dataset and finetuned on our custom dataset, which can be accessed here (a Google Drive folder).

We constructed a custom dataset consisting of 101 high-resolution photographs captured during Finals Week on Friday, December 13, between approximately 11:15 A.M. and 7:00 P.M. All images were taken from an elevated mezzanine viewpoint overlooking Steep Café in Yale's Science Building. This fixed camera perspective introduces strong consistency in scene layout while still capturing substantial variation in lighting, occupancy level, and human pose.

Images were captured using a Canon EOS Rebel T6i camera equipped with a 24.2-megapixel CMOS sensor. The camera was operated without flash to preserve natural lighting conditions, which varied significantly throughout the day due to changes in sunlight intensity and interior illumination. Most images were taken through a glass window, introducing realistic challenges such as reflections, glare, and partial occlusion. Guided by the photographer's discretion, only one photograph was captured per distinct scene, which we defined as a moderate increase or decrease in occupancy level, changes in multiple individuals' poses, or movement of people between tables. This sparse pattern of data collection reduced redundancy by preventing duplicates or near-duplicates in the dataset. The images reflect realistic crowd scenarios, including varying lighting conditions (daytime and nighttime), camera viewpoints, crowd densities, and degrees of occlusion.

Figure 1. Example of custom dataset image (1920×1280 pixels)

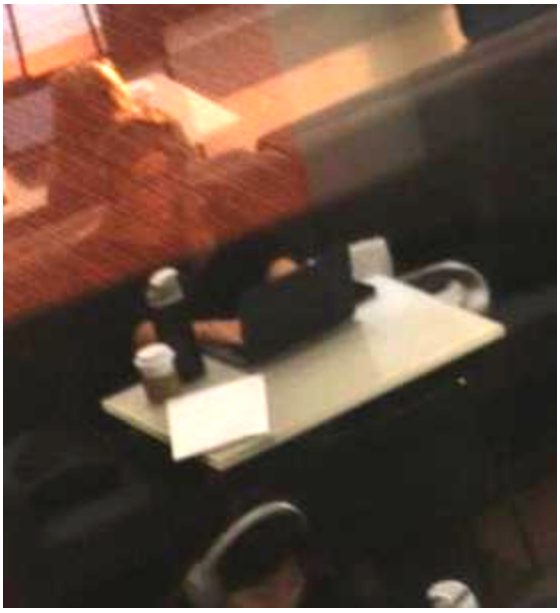

Figure 2. Subwindow of Figure 1 in which window reflection partially occludes person

Implementation Details

Data pre-processing

- **Manual annotation and auto-labeling:** All images were annotated with bounding boxes corresponding to visible people using Roboflow's web-based annotation tool. An initial seed set of fifteen images was manually and carefully annotated. Each visible person was labeled with a bounding box using a single semantic class: "people". The annotator made sure to stay consistent with the boxing method. For each person, a box was drawn around any visible parts of their body, even when partially occluded by furniture or other objects.

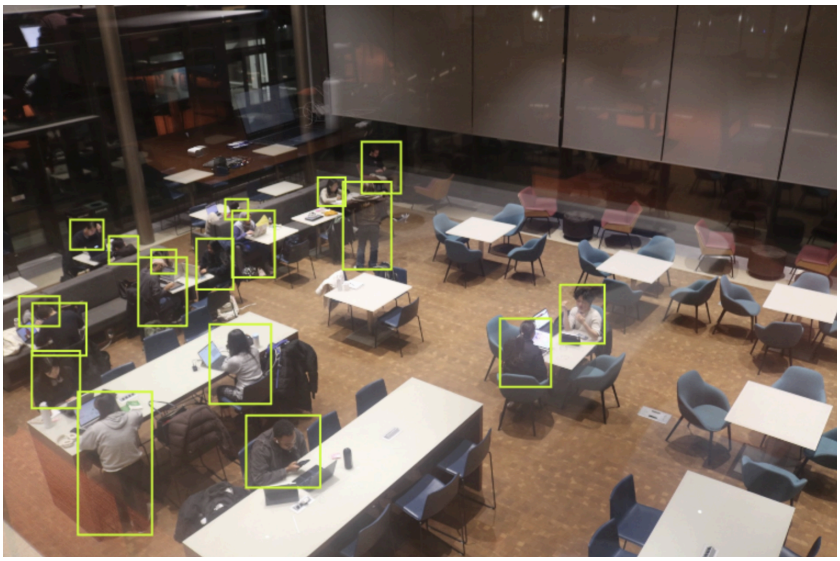

Figure 3. Example of a manually annotated image using the bounding box tool in Roboflow

To streamline the annotation process for the remaining images, we leveraged Roboflow's Instant Model feature. We uploaded our fifteen manually annotated images to a dataset and trained a small Roboflow model on them, creating a Label Assist model that produced preliminary bounding boxes for the remaining eighty-six images. These auto-generated annotations were manually reviewed, with false positives removed and missed detections added. This semi-automated process significantly reduced labeling time while maintaining annotation quality. The final dataset contains 101 fully annotated images.

- **Data splitting:** We split the dataset images according to the following distribution.

Train	Valid	Test
70	16	15

Table 1. Pre-augmentation train/valid/test split

- **Pre-processing:** Each image was pre-processed, which simply involved resizing the image to a fixed 640x640 resolution. Resizing was crucial for standardizing input dimensions, which reduces computational load and ensures consistent feature processing.
- **Augmentation:** To mitigate the limited size of the training set, data augmentation was applied exclusively to the training subset. Each training image was augmented using horizontal flipping and small brightness and exposure jitters (between -3% and +3%). These augmentations simulate realistic variations in lighting conditions without altering scene geometry.

Each original training image produced four total variants, resulting in 280 images in the training set and 311 images total. Validation and test splits remained unaugmented to preserve unbiased evaluation. The new split was as follows:

Train	Valid	Test
280	16	15

Table 2. Post-augmentation train/valid/test split

- **Export:** The 311 unannotated images were uploaded as JPG files into a data folder in a newly initialized Git repository. The annotations were uploaded as three JSON files (one for each split).

Training setup

This project was implemented in Python using the PyTorch and TorchVision libraries. The complete codebase is organized in a public Github repository. The source code directory is structured as follows:

```
src/
└ dataset.py          # for dataset loading
└ model.py            # for model definition
└ train.py            # for training
└ eval.py             # for qualitative evaluation
└ eval_map.py         # for quantitative evaluation
```

1. **`dataset.py`** defines a CafeDataset class with an initialization function that creates a dictionary that maps a raw image to its annotated version through ID. Following TorchVision API, the getitem function returns a tuple of the image tensor and its boxes, labels, and id.
2. **`model.py`** defines a get_model function that loads TorchVision's pretrained `fasterrcnn_resnet50_fpn` architecture with its default weights. The final classification head is replaced to support binary classification (person vs background).
3. **`train.py`** defines the following hyperparameters: `NUM_EPOCHS`, `BATCH_SIZE`, `LEARNING_RATE`, `MOMENTUM`, and `WEIGHT_DECAY`.

Hyperparameter	Value
`NUM_EPOCHS`	`10`

BATCH_SIZE	2
LEARNING_RATE	0.005
MOMENTUM	0.9
WEIGHT_DECAY	0.0005

The main function loads the training and valid datasets, each composed of data and annotations then initializes a PyTorch DataLoader with the training dataset. An SGD optimizer is created with the above hyperparameters. Training proceeds for a fixed number of epochs using backpropagation and gradient descent. Upon completion of each epoch, the average loss is computed and printed, and the model weights are saved to disk as checkpoints files. Due to file size constraints, these checkpoint files are not included in the repository; instructions for reproducing training or using pre-trained checkpoints are provided in the README.

4. **`eval.py`** performs qualitative evaluation by running inference on the test set, drawing predicted bounding boxes, estimating person counts per image, and saving annotated images for visual inspection.
5. **`eval_map.py`**, on the other hand, utilizes the `pycocotools` library to output a quantitative evaluation, comparing predicted detections against ground-truth annotations to compute standard COCO metrics, including mean Average Precision at an Intersection over Union threshold of 0.50 (mAP@0.5), as well as precision and recall.

Results

Evaluation criteria

The primary measure of success for this project was mean Average Precision at an Intersection over Union threshold of 0.50 (mAP@0.5), computed using the COCO evaluation protocol. This metric evaluates both classification correctness and localization accuracy by considering a detection correct if the predicted bounding box overlaps a ground-truth box by at least 50%. mAP@0.5 is widely used in object detection tasks and is particularly appropriate for single-class detection problems with moderate object density.

We also reported precision and recall values stratified by object scale (small, medium, large) to analyze how detection performance varies with subject size. Training performance is assessed using the average training loss per epoch, which reflects convergence behavior but is not itself a measure of detection accuracy.

Training performance

We ran the experiment once, fine-tuning the model's weights on the data in the "train" folder and evaluating the model's performance on the data in the "test" folder. Over ten epochs with a batch size of two, the model's average loss decreased steadily from 0.6538 to 0.1083 (see Figure 4), indicating effective learning and stable convergence.

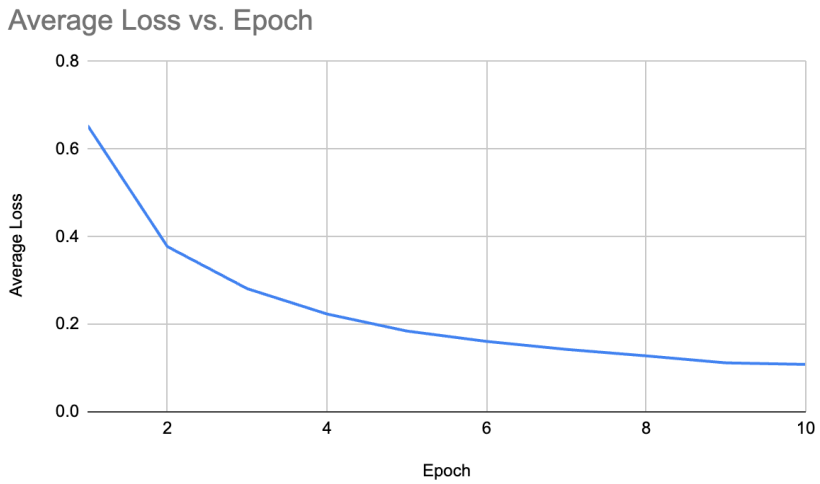

Figure 4. Change in average loss over ten epochs of model training

mAP

The fine-tuned model achieved an mAP@0.5 of 0.89. This score exceeded our initial performance goal of 0.6 and indicates strong detection accuracy.

However, performance varied across object scales:

- **Large objects** (AP = 1.000, AR = 1.000): The model achieves perfect detection for large-scale people; those closest to the camera or occupying substantial image area.
- **Medium objects** (AP = 0.902, AR = 0.937): Performance on medium-scale people is still very strong, but slightly worse than large-scale detection.
- **Small objects** (AP = 0.747, AR = 0.882): Detection accuracy drops for small-scale people. While still achieving reasonable performance, the 15% drop from medium-scale

AP aligns with known challenges in detecting small objects, as fewer pixels provide less information.

This scale-dependent behavior is consistent with known limitations of region-based detectors.

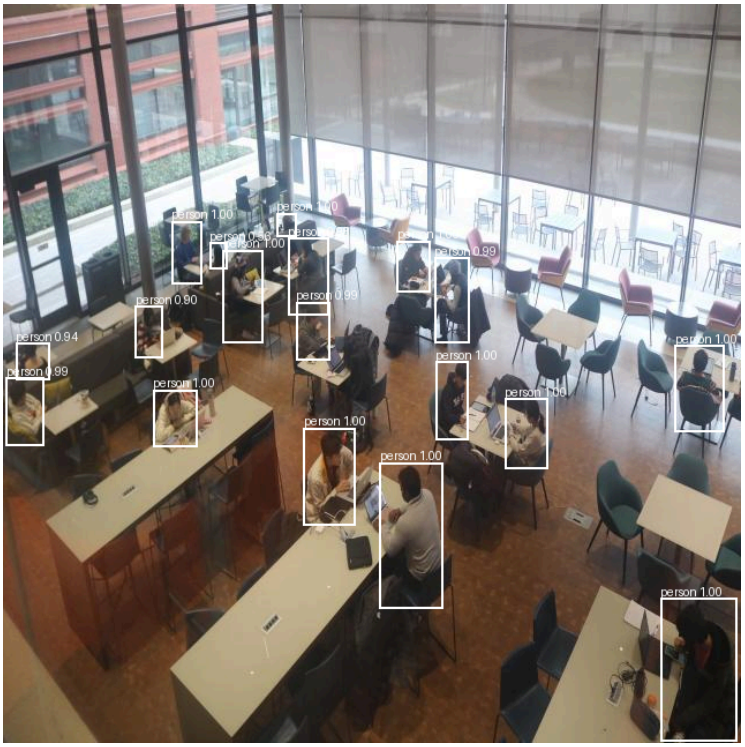

Figure 5. Example of accurate output of fine-tuned model on test image

MAE

We also evaluated counting performance using Mean Absolute Error (MAE) between the predicted number of people and the ground truth count for each image. The fine-tuned model achieves an MAE of 0.87, indicating that predicted occupancy is usually within one person of the true count. This counting error is within the 10–15% range specified in our project proposal as a success metric.

Errors

Two primary error types were observed:

- False positives: The model tended to produce false positives in regions with more complex visual patterns, such as clutter on a student's desk (see figure 6).

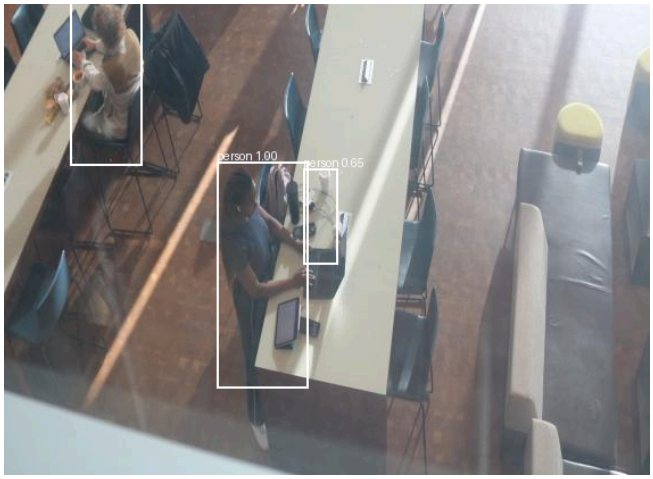

Figure 6. Example of false positive

- False negatives: The model occasionally excluded more distant people, such as those by the windows of Steep (see figure 7).

Figure 7. Example of false negative

Discussion

One key takeaway from this project was the effectiveness of transfer learning for domain-specific tasks. Despite the limited dataset size, fine-tuning a pretrained object detector performed well in a constrained indoor environment.

Challenges

Several challenges were encountered throughout the project. One was the time-consuming nature of manual annotation, particularly of crowded scenes during peak occupancy time. Accurately drawing bounding boxes around each individual required careful inspection and consistency, making full manual labeling impractical at scale. To address this, we adopted a semi-automated labeling approach using a lightweight detection model (made with Roboflow) as a Label Assistant.Training the model on CPU hardware resulted in long runtimes. Careful logging and checkpointing ensured progress could be monitored, preserved, and later replicated.

Limitations

Our dataset is small and collected from a single location, which limits the generalizability of results to other environments or crowd conditions. Additionally, the counting struggled with small or heavily occluded people, which was observed in distant or cluttered regions of the cafe. Future extensions of this work could include expanding the dataset to include a greater variety of images from different study spots around Yale's campus. We could also explore density-based counting methods for higher-density scenes, such as Commons.

Conclusion

Overall, this work illustrates how state-of-the-art object detection models can be effectively adapted to constrained real-world settings and serves as a practical case study in applied computer vision.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2014, pp. 580–587.

[2] R. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2015, pp. 91–99.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627-1645, Sept. 2010, doi: 10.1109/TPAMI.2009.167.

[6] P. Feifel, B. Franke, F. Bonarens, F. Köster, A. Raulf, and F. Schwenker, "Revisiting evaluation of deep neural networks for pedestrian detection," arXiv preprint arXiv:2511.10308, 2025.

[7] D. Chavan and A. Purohit, "Machine learning techniques for crowd counting: A survey," International Journal of Computer Applications, vol. 185, no. 37, pp. 1–6, Oct. 2023.