Object Detection for Sidewalk Trash

Peiying Lin
Department of statistics
Virginia Tech
Peiying@vt.edu

Abstract

This paper evaluates Faster R-CNN and YOLOv8 for sidewalk trash detection, focusing on their speed and accuracy trade-offs and the impact of synthetic data augmentation. Models were trained on the Roboflow "Only Taco" dataset and a merged dataset combining real-world images with AI-generated trash examples. Performance was assessed on real sidewalk photos using mAP@50, mAP@50-95, and inference speed. YOLOv8 achieved near real-time inference (10.8 ms/image) and superior localization accuracy (mAP@50-95: 0.471) but exhibited lower recall (0.737). In contrast, Faster R-CNN prioritized exhaustive detection, achieving higher recall (0.885) at 30× slower speeds (317.8 ms/image) and lower precision (0.697). Merging synthetic data improved YOLOv8's recall by 12% but reduced Faster R-CNN's localization precision (mAP@50-95: 0.351), likely due to its sensitivity to noisy proposals. These results demonstrate YOLOv8's suitability for real-time robotic systems requiring fast, precise detections, while Faster R-CNN excels in recall-critical applications. Future work should explore hybrid architectures and refined synthetic data generation to bridge these trade-offs.[1]

1 Introduction

Trash is commonly seen on sidewalks and often stays there for days or even weeks before being cleaned. This not only affects the environment but also lowers the quality of public spaces. A possible solution is to use cleaning robots, but for them to work effectively, they first need a model that can detect and locate trash on the ground. This paper focuses on training such a model using object detection techniques, with the goal of outlining where trash appears in sidewalk images. To evaluate performance, two well known object detection algorithms, Faster R-CNN and YOLO, are tested on photos of sidewalk trash collected in real outdoor settings.

2 Related work

Several recent studies have explored the use of deep learning for trash detection and robotic collection. In one such project, Kulshreshtha et al. [1] developed an Outdoor Autonomous Trash Collecting Robot (OATCR) that integrates object detection models specifically Mask R-CNN, YOLOv4, and YOLOv4 tiny with a fully automated robotic system capable of navigating outdoor environments, detecting trash, and collecting it using a flap bucket mechanism. The YOLOv4 tiny model was ultimately selected due to its balance between high accuracy (95.2% mAP) and very low inference time (5.21 ms), making it suitable for real-time use on embedded systems like Raspberry Pi. Their system was tested on the TACO dataset and demonstrated effective performance across various terrains. This work illustrates the feasibility of integrating object detection into autonomous robots for environmental cleanup and provides a strong benchmark for evaluating different object detection algorithms in trash identification tasks.

[1] Code available at: https://github.com/Pegi-1224/Sidewalk_Trash_Detection.git

STAT4984 , LateX template borrowed from NIPS 2017.

3 Dataset and Features

This study uses three sources of image data to train and evaluate the object detection models. The first is the Only Taco dataset from Roboflow[3], which contains over 1,000 labeled images. While the dataset focuses on trash, it also includes many scenes where everyday items—such as those on tables or held in hands, appear but are not labeled as trash. Additionally, it contains trash in various environments such as sidewalks and beaches. Despite the inconsistent environments, these images are considered valuable for training, as they help the model generalize what trash looks like across different settings and distinguish it from background objects.

The second dataset was collected manually, consisting of 200 photographs taken along sidewalks near the school. These images reflect real-world conditions and were manually annotated with bounding boxes for visible trash.

The third dataset includes 250 AI generated images created using ChatGPT's image generation tools. These synthetic images were intended to resemble common trash scenarios. However, during labeling, it was observed that the boundaries between trash and background were often vague, which could introduce noise and reduce the model's ability to precisely localize objects.

All images across the three datasets were labeled using a single class: "trash". This decision was made to simplify the classification task and focus the model on recognizing general trash patterns. Since the types of trash can vary greatly and may grow over time, using a unified label helps the model learn broader visual features that define trash, rather than relying on specific subcategories.

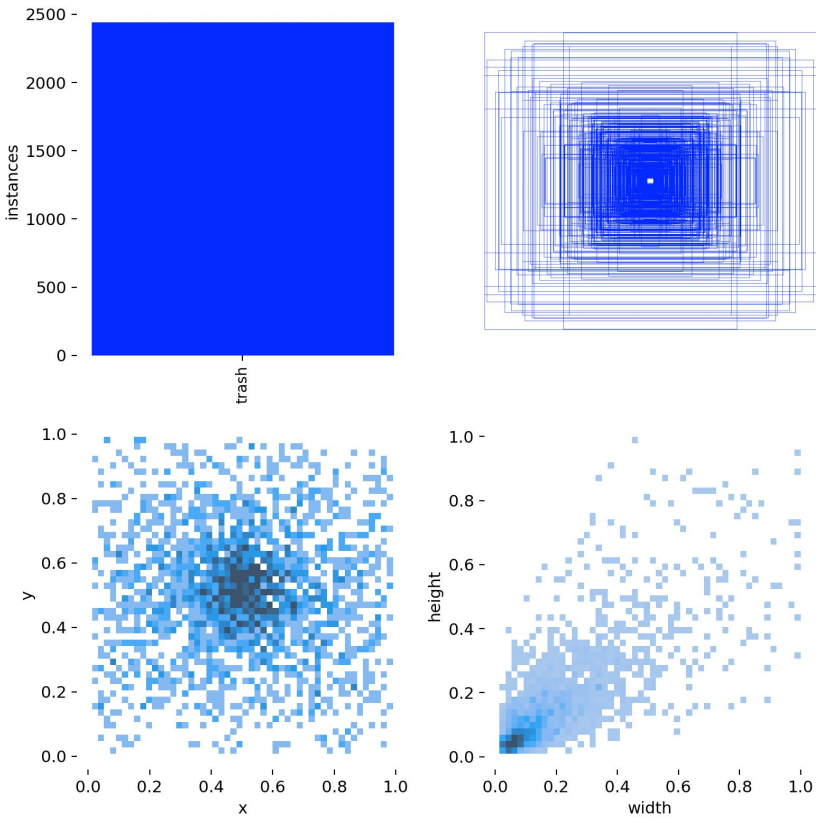

Figure 1: Distribution of Bounding Boxes and Object Locations in Training Dataset

4 Methods

4.1 Faster R-CNN

Faster R-CNN [2] is a two-stage object detection framework that integrates region proposal and classification into a unified architecture. The first stage uses a Region Proposal Network (RPN), a fully convolutional network that slides over shared convolutional feature maps to generate candidate object regions, known as proposals. These proposals are generated using anchor boxes of multiple scales and aspect ratios, allowing the model to detect objects of various sizes in a translation invariant manner. Each anchor is evaluated with an objectness score and refined through bounding box regression. The second stage, based on the Fast R-CNN detector, classifies each proposed region and further refines its bounding box coordinates.

The network is trained using a multitask loss that combines classification loss (softmax over object vs. background) and regression loss (smooth L1) for bounding box coordinates. Positive anchors are selected based on their Intersection over Union (IoU) with ground truth boxes, while those with low IoU are labeled as negative. To train the full system, a four step alternating optimization procedure is used: the RPN and Fast R-CNN are trained sequentially, with convolutional layers gradually shared between them. This shared feature strategy significantly reduces computation, making region proposal nearly cost free and enabling Faster R-CNN to run at near real-time speeds (e.g., about 200 ms per image).

4.2 YOLOv8

YOLOv8 is a one-stage, anchor free object detection model developed by Ultralytics for real-time applications [4]. It integrates three main components: a CSP based backbone that extracts multiscale image features efficiently, a neck composed of an enhanced PANet combined with a Feature Pyramid Network (FPN) to fuse features across scales, and a head that directly predicts bounding boxes and class scores without relying on predefined anchor boxes. This anchor free design simplifies training and improves generalization across objects with varying shapes and sizes.

YOLOv8 includes built in data augmentation techniques such as mosaic and mixup, along with random scaling, flipping, and color adjustments. These augmentations are applied during training and help the model generalize to different backgrounds, object sizes, and occlusion patterns.

The training loss consists of three components: focal loss for classification, which emphasizes hard to classify examples and reduces class imbalance; IoU loss for bounding box regression, which encourages accurate localization; and objectness loss, which guides the model to focus on object containing regions. YOLOv8 also supports mixed precision training, reducing memory usage and improving speed on modern GPUs. With multiple model sizes available from lightweight (YOLOv8n) to high accuracy (YOLOv8x) the framework adapts well to a range of hardware and performance needs.

5 Results

5.1 Experimental Setup

All experiments were conducted on an NVIDIA GeForce RTX 4080 SUPER GPU (16 GB VRAM) with CUDA 12.8 and an Intel Core i9-14900KF CPU (32 GB RAM). Training utilized PyTorch 2.5.1 with mixed precision enabled for GPU acceleration.

5.2 Training Configuration

Two object detection models were trained and evaluated: YOLOv8n and Faster R-CNN. For YOLOv8n, the nano variant defined in yolov8n.yaml was used with randomly initialized weights (no pretrained backbone). The model was trained for 200 epochs using a batch size of 16, with input images resized to 640×640 pixels. The optimizer and loss functions followed Ultralytics' default configuration, which includes classification, bounding box regression, and objectness losses.

For Faster R-CNN, the model was built using the *fasterrcnn resnet50 fpn* backbone, combining a ResNet50 encoder with a Feature Pyramid Network. The model was trained from scratch

(weights=None) with a custom classification head for 2 classes. Training was conducted over 30 epochs with a batch size of 16. The optimizer used was Stochastic Gradient Descent (SGD) with a learning rate of 0.005, momentum of 0.9, and weight decay of 0.0005. The total loss included both classification and bounding box regression components, following the default PyTorch Faster R-CNN implementation.

For both YOLOv8 and Faster R-CNN, models were trained from scratch (i.e., without pretrained weights) to ensure a fair comparison under synthetic data augmentation. This approach isolated the impact of synthetic data on model performance without conflating effects from pretrained feature extraction.

5.3 Model Performance Comparison

Table 1 and Table 2 summarizes the performance of YOLOv8 and Faster R-CNN across different dataset configurations, including training time, test time precision and recall, mean Average Precision (mAP@50 and mAP@50–95), and per image inference speed. YOLOv8 models were trained for 200 epochs, while Faster R-CNN models were trained for 30 epochs, both using a batch size of 16. YOLOv8 models demonstrated consistently faster training and inference speeds, with inference taking around 10ms per image, compared to over 300ms for Faster R-CNN. The merged dataset improved YOLOv8's recall and mAP metrics, while Faster R-CNN showed minimal change, likely due to its sensitivity to synthetic data and limited training epochs.

Table 1: Training configuration and timing for YOLOv8 and Faster R-CNN.

Model	**Dataset**	**Epochs**	**Batch**	**Train Time (s)**
YOLOv8	Original	200	16	884.76
YOLOv8	Merged	200	16	1040.39
Faster R-CNN	Original	30	16	1158.29
Faster R-CNN	Merged	30	16	1869.89
Faster R-CNN	Augmented	30	16	1398.30

Table 2: Evaluation performance metrics on the test set.

Model	**Dataset**	**Precision**	**Recall**	**mAP@50**	**mAP@50-95**	**Infer Time (ms)**
YOLOv8	Original	0.798	0.658	0.758	0.443	10.1
YOLOv8	Merged	0.779	0.737	0.779	0.471	10.8
Faster R-CNN	Original	0.699	0.862	0.699	0.392	311.1
Faster R-CNN	Merged	0.700	0.850	0.700	0.402	317.8
Faster R-CNN	Augmented	0.697	0.885	0.697	0.351	316.4

To better visualize the results, Figure 2 compares each model's test accuracy metrics (precision, recall, mAP@50, and mAP@50–95). Figure 3 shows the training time for each model configuration, highlighting YOLOv8's faster convergence. Finally, Figure 4 presents the inference time per image, illustrating the computational cost differences across architectures.

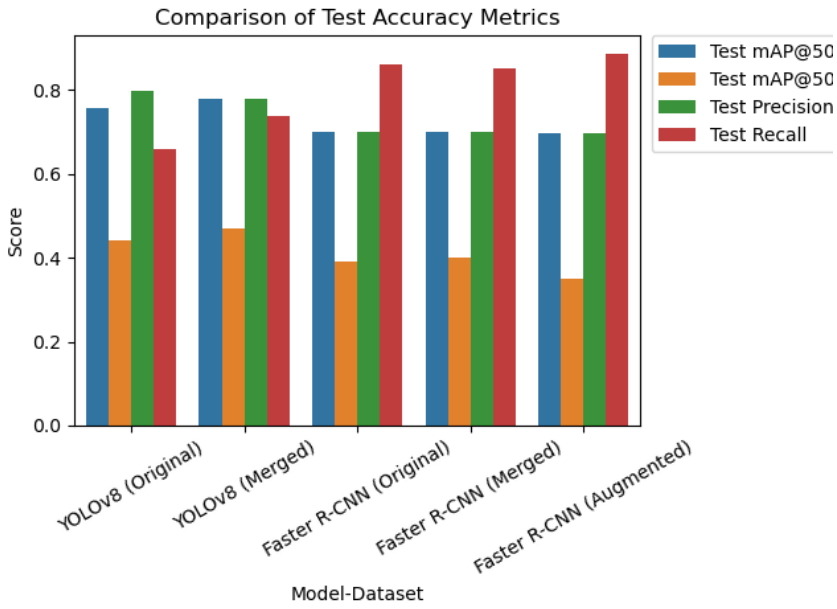

Figure 2: Comparison of test accuracy metrics across models and datasets.

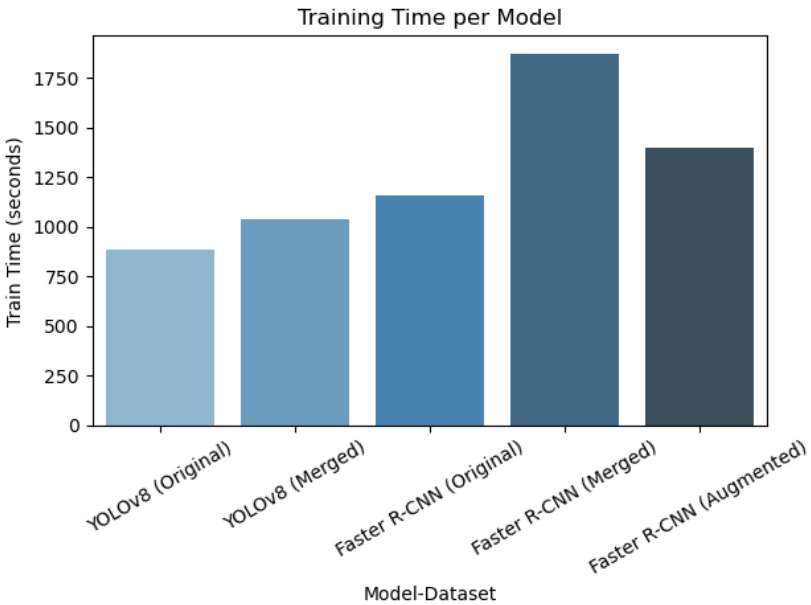

Figure 3: Total training time for each model configuration.

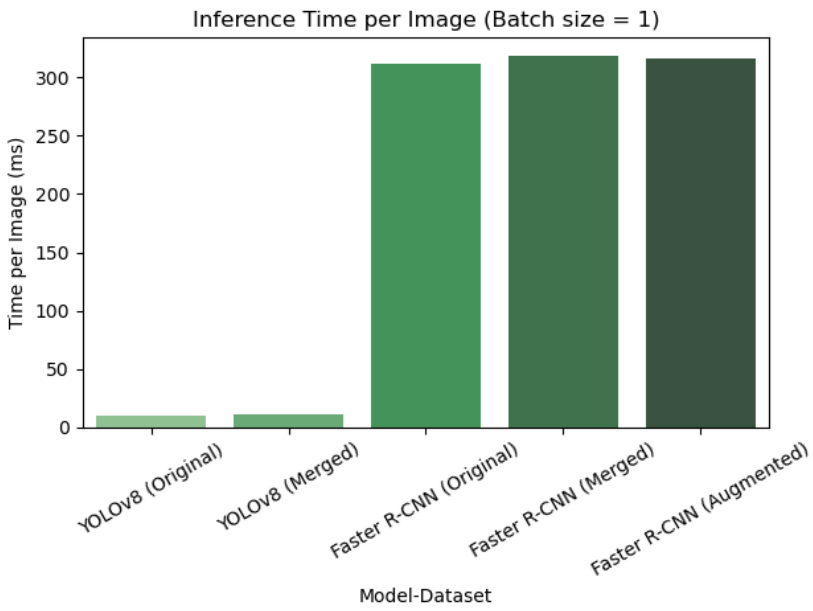

Figure 4: Average inference time per image (batch size = 1).

6 Conclusion and discussion

6.1 YOLOv8 vs. Faster R-CNN

YOLOv8 and Faster R-CNN exemplify the trade-offs between speed and thoroughness in object detection. As a one-stage detector, YOLOv8 processes images in a single forward pass using a lightweight CSPDarknet backbone and a C2f module that efficiently fuses multiscale features through cross stage connections. This streamlined architecture, combined with a single Non-Max Suppression (NMS) step, enables real-time inference speeds, making it ideal for applications like video analysis or robotics. However, its grid based prediction approach, where objects are detected at predefined anchor points or through anchor free coordinate regression, can struggle with objects at irregular scales or off grid positions, leading to slightly lower recall compared to two-stage methods.

In contrast, Faster R-CNN employs a two-stage pipeline optimized for exhaustive object coverage. Its Region Proposal Network generates hundreds of region candidates, which are then refined via RoI Align and a heavier ResNet50 FPN backbone paired with a classification/regression head. This dual stage process ensures Faster R-CNN excels at recall (finding all objects), but its complex pipeline introduces more opportunities for false positives. While the second stage classifier filters out poor proposals, residual background noise or ambiguous regions can persist, marginally reducing precision and mAP (mean Average Precision) compared to YOLOv8 in speed oriented benchmarks. Additionally, the computational overhead of generating and processing proposals, alongside multiple NMS steps, slows inference significantly.

Ultimately, YOLOv8 prioritizes speed and precision, delivering reliable predictions for time-sensitive tasks, while Faster R-CNN sacrifices efficiency for meticulous region analysis, making it better suited for scenarios where missing objects (e.g., medical imaging) is costlier than processing latency.

6.2 AI Generated Image vs. Original Image

In the merged dataset experiments (original images combined with AI generated images), YOLOv8 showed clear improvement, while Faster R-CNN did not benefit significantly. YOLOv8 achieved higher recall and mAP, indicating better object localization and generalization, but with slightly lower precision—suggesting an increase in false positives. This trade-off is expected, as the added data helped YOLOv8 detect more objects, even if some detections were less confident.

On the other hand, Faster R-CNN's performance remained mostly unchanged. This may be due to its heavier architecture (ResNet50 backbone and two-stage detection), which requires more training epochs to fully benefit from additional data. Moreover, Faster R-CNN's Region Proposal Network (RPN) is sensitive to input noise and relies on clean, consistent visual patterns, something AI generated images may not always provide. As a result, the artificial data may not have contributed useful features for generating high quality proposals.

Overall, YOLOv8 appears to be more robust to synthetic data and can effectively leverage it to improve detection performance, while Faster R-CNN may require cleaner data, longer training, or more careful integration of synthetic inputs to show similar gains.

6.3 Discussion

This study demonstrates that YOLOv8 is better suited for real-time sidewalk trash detection tasks compared to Faster R-CNN. While both models were able to detect trash objects with reasonable accuracy, YOLOv8 significantly outperformed Faster R-CNN in terms of inference speed, requiring only around 10 milliseconds per image, compared to over 300 milliseconds for Faster R-CNN. This makes YOLOv8 much more appropriate for deployment in robotic systems where real-time detection is essential.

Due to time constraints, this study did not include experiments with DETR, a transformer-based object detection model. As DETR is known for its strong performance in object localization, especially in complex scenes, it could serve as a valuable comparison in future work.

Overall, the results suggest that YOLOv8 offers the best trade-off between speed and accuracy for trash detection tasks. Its higher mAP values indicate that it can produce tighter and more accurate bounding boxes, which is particularly important for robotic applications where precise localization is needed for object pickup and removal.

References

[1] Medhasvi Kulshreshtha, Sushma S. Chandra, Princy Randhawa, Georgios Tsaramirsis, Adil Khadidos, and Alaa O. Khadidos. Oatcr: Outdoor autonomous trash-collecting robot design using yolov4-tiny. *Electronics*, 10(18):2292, 2021.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[3] Roboflow and TACO-YOLO. Only taco dataset - background0234v2. `https://universe.roboflow.com/tacotoyolo/only-taco-background0234v2`, 2023. Accessed: 2025-05-08.

[4] Muhammad Yaseen. What is yolov8: An in-depth exploration of the internal features of the next-generation object detector, 2024.