CSCI 3485 Lab 5: Object Detection

Rafael Almeida, Ryan Novitski

November 2, 2024

1 Introduction

Object detection is a fundamental principle within the field of computer vision that is important to the foundations of many of its various implementations. Although traditional image classification similarly has the ability to indicate the presence of an object of a finite class, many real-world applications need a higher level of accuracy and speed to be effective. Object detection, or the practice of locating and identifying objects within images or video frames, is essential for tasks such as automated driving, surveillance, and medical imaging. Without any spatial context, these tasks would not be possible to perform accurately enough to ensure safety.

Although these tasks all require some form of object detection to be effective, they also have different purposes and priorities. For instance, medical imaging needs to be as accurate as possible for both its classification and spatial context, but the time it takes to gain these inferences is not critical to its effectiveness(to an extent, of course). However, for the purpose of automated driving, maintaining as close to real-time speed as possible is necessary for its inferences to allow for any micro-adjustments needed to ensure safety for the driver and its surroundings. Because of this tradeoff of speed and accuracy of models, there are different models being used in various applications today. This lab will analyze and compare the effectiveness of the Faster-RCNN and YOLOv5 object detection models using qualitative observations, classification confidence levels, Intersection over Union(IoU), and inference time.

2 Methodology

2.1 Datasets

For this experiment, we used two types of image datasets to evaluate the performance of Faster R-CNN and YOLOv5:

- **Custom Images**: A set of five images collected from the internet, containing various scenes and objects. These images will be used for a qualitative analysis by testing each model's ability to detect diverse object types in unique settings.
- **Bus Dataset**: A pre-defined dataset of bus images used to perform quantitative evaluation. A subset of 50 images of this dataset will be used to quantitatively test each model's performance using classification confidence levels, IoU, and inference times for comparison.

Both datasets were preprocessed to match the input requirements of the models, including resizing and normalization.

2.2 Models and Implementations

Two deep learning models were selected for this lab due to their current relevancy and complementary strengths in object detection:

- **Faster R-CNN**: We used the official PyTorch implementation of Faster R-CNN with a ResNet-50 backbone. Faster R-CNN is a two-stage detector that generates region proposals on the first pass, then performs object classification and bounding box regression on each proposal the next pass.

- **YOLOv5**: The YOLOv5 model was sourced from PyTorch Hub's implementation, which allows for single-pass detection. YOLOv5 performs object detection in a single pass, achieving higher inference speed compared to Faster R-CNN but potentially lower accuracy.

Both models were run on an A100 GPU to ensure efficient and consistent processing of the images.

2.3 Experimental Setup

Each model was evaluated on both datasets with the following steps:

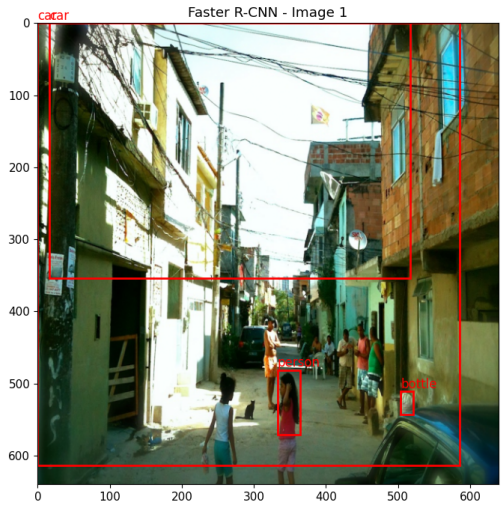
1. **Pre-processing**: All images were resized to match each model's input requirements (640x640 pixels for YOLOv5). Additionally, images were normalized using mean and standard deviation values consistent with ImageNet pre-trained models.
2. **Inference**: We ran inference on each model for both datasets, capturing bounding boxes, object class labels, and confidence scores for each detected object.
3. **Evaluation Metrics**:
    - **Qualitative Analysis**: Observations based on object detections, bounding box accuracy, etc.
    - **Intersection over Union (IoU)**: To measure how well each model's predicted bounding boxes aligned with ground truth. An IoU threshold of 0.5 was used to classify a detection as a true positive.
    - **Confidence Score**: The confidence level associated with each prediction, indicating the model's certainty.
    - **Inference Time**: For each image, we recorded the time taken by each model to process and generate predictions. This metric highlights the speed differences between Faster R-CNN and YOLOv5.

The combination of qualitative and quantitative analyses provides a comprehensive view of each model's performance.

3 Results

3.1 Faster-RCNN

3.1.1 Qualitative Analysis: Custom Images

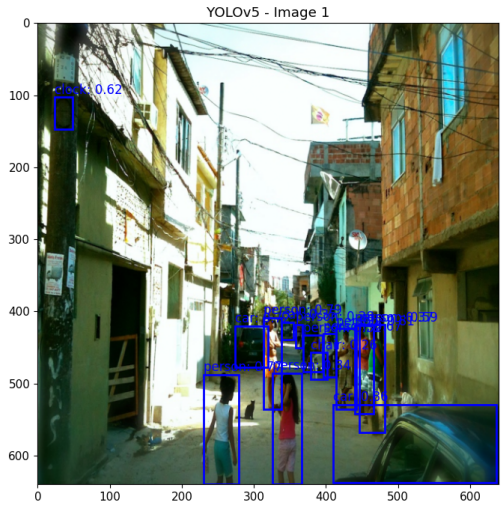


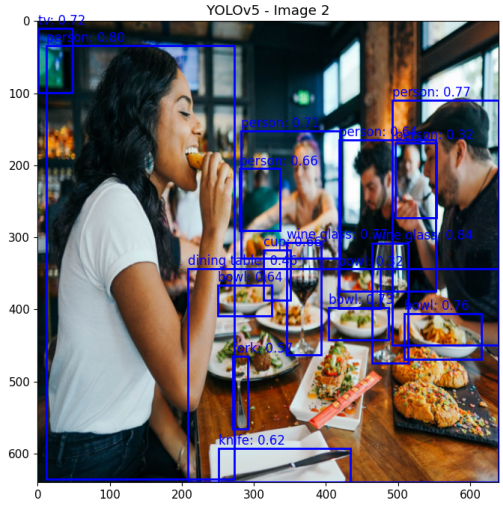
3.1.2 Quantitative Analysis: Bus Dataset

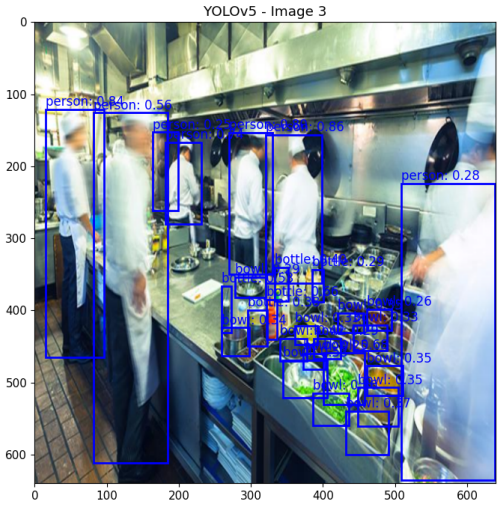
Mean IoU: 0.14
Accuracy: 0.00
Average Inference Time: 2.5964 seconds

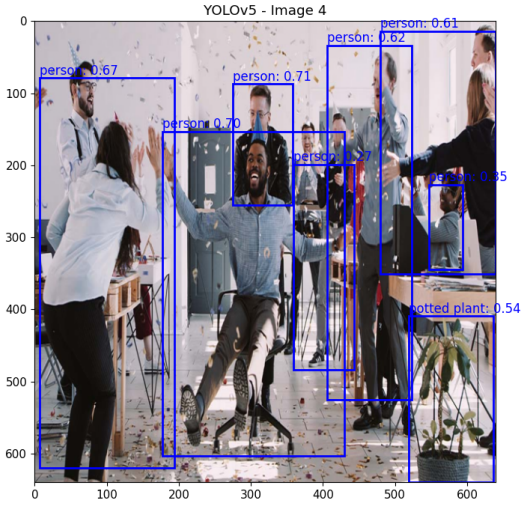
3.2 YOLOv5

3.2.1 Qualitative Analysis: Custom Images









3.2.2 Quantitative Analysis: Bus Dataset

Mean IoU: 0.86
Accuracy: 33.33
Average Inference Time: 0.3074 seconds

4 Discussion

First, looking at the qualitative part of the results, we notice a clear difference between Faster R-CNN and YOLO. Faster R-CNN did not detect all the objects for most images, and most were in the entirely wrong classes. YOLO detected most objects and correctly classified them. There was one instance in the picture of the kitchen where Faster R-CNN detected some of the pans but classified them as traffic lights. YOLO detected more objects and was more likely to be correct in all images by a considerable margin.

The quantitative results reflect what was seen in the qualitative ones. Faster R-CNN's huge mistakes and inability to classify all objects, and not even do so with the correct classes, is reflected in the quantitative results with a Mean IoU of only 0.14 out of 1 and an accuracy of 0. On the other hand, the staggeringly better performance of YOLO in the qualitative section is reflected in the quantitative section with a mean IoU of 0.86, meaning its bounding box covered, on average, 86 percent of the actual bounding box. Moreover, its accuracy was 33 percent, meaning that its predictions were more significant than the given threshold 33 percent of the time.

Finally, the inference time of YOLO, as expected, was much lower than that of Faster R-CNN. This can be explained by YOLO's architecture, which is specifically designed to achieve fast inference times. In contrast, how Faster R-CNN makes its bounding box predictions is not as good, and inferences are not as practical as YOLO.

5 Conclusion

Overall, YOLO outperformed Faster R-CNN in every metric, including qualitative analysis and quantitative metrics like inference time, mean IoU, and accuracy. Faster R-CNN likely performed vastly worse because of the challenging and non-standard images chosen. In the future, this paper can be improved by making qualitative analysis on more standard images from each of the model's training datasets. This way, we can ensure the results are accurate without worrying about the images not being in the correct format

for each model. Thus, YOLO seems to be the better model for accuracy and, most importantly, speed dependent tasks.

6 Code

GitHub Repository