Object Detection With Small YOLOv5

Simone Vaccari 915222
Davide Vettore 868855

1. Objective:

The goal of this project is to train a small YOLOv5 model on the Vehicles-OpenImages dataset for 5-class object detection. The trained network will then be tested on both images and videos downloaded from the internet, as well as videos recorded by ourselves.

2. Introduction:

Object detection is a very important task in computer vision, as it helps us find and locate objects in images. One of the most popular models for this is the YOLO (You Only Look Once) series, especially YOLOv5, by Ultralytics. YOLOv5 is noteworthy because it balances accuracy and speed, making it perfect for real-time applications. There exist different versions of this model, such as YOLOv5s (small) and YOLOv5m (medium), which cater to different needs. Unlike old methods that look at different parts of the image separately, YOLO looks at the whole image at once, predicting bounding boxes and class probabilities for each part of a grid. Furthermore, the input image is passed through the network only once, hence the name of the technique. This method allows YOLO to be fast and accurate at the same time.

The dataset used here includes 627 images where different types of vehicles, like cars, buses, motorcycles, ambulances, and trucks, are detected. These images come from a smaller part focused on vehicles of the Open Images dataset, which is a large open-source collection for computer vision tasks.

In object detection, inference means using a trained model to make predictions on new images. This involves inputting images or video frames into the model and getting predictions about where and what the objects are.

Figure 1: Example of images from the Vehicles-OpenImages dataset

3. Procedure:

First, we prepared the dataset by downloading the Vehicles-OpenImages dataset, which includes images of different vehicle classes. We then cleaned the dataset by removing duplicate images and corresponding label files, ensuring the dataset structure was correct. In total, the dataset contains 439 training images, 125 validation images, and 63 test images. In Figure 2 an example of training batch is showed.

Figure 2: Example of training batch

Next, we cloned the YOLOv5 repository from GitHub and installed the necessary dependencies. We then trained the YOLOv5 small model (yolov5s) for 25 epochs using the prepared dataset and saved the training results in a specified directory. After training, we validated the network by displaying the validation predictions saved during training to visualize the model's performance on the validation set.

For inference, we used the trained model to make predictions on new images and videos downloaded from the internet and visualized the results to evaluate the model's performance on unseen data. Finally, we tested the network on recorded videos containing one or more vehicle classes and analyzed the errors by displaying frames with correct and incorrect detections.

4. Results:

After 25 epochs, the network achieved the results shown in Table 1. For both the training and validation sets, the classification loss, bounding box regression loss, and objectness loss are displayed. Additionally, metrics like precision, recall, and mean average precision (computed both at a single threshold and averaged over multiple thresholds) are reported.

train box loss	train obj loss	train cls loss	precision	recall	mAP 0.5	mAP 0.5:0.95	val box loss	val obj loss	val cls loss
0.0273	0.0171	0.0042	0.7322	0.6041	0.6328	0.4534	0.0408	0.0126	0.0167

Table 1: Results obtained at the end of the training

The confusion matrix in Figure 3 shows the distribution of true positives, false positives, true negatives, and false negatives for each class. Figure 4 provides additional insights into the classification performance of our YOLO network during training by considering changes in the confidence level. These curves help in understanding the trade-offs and selecting an optimal confidence threshold for deploying the model in real-world scenarios.

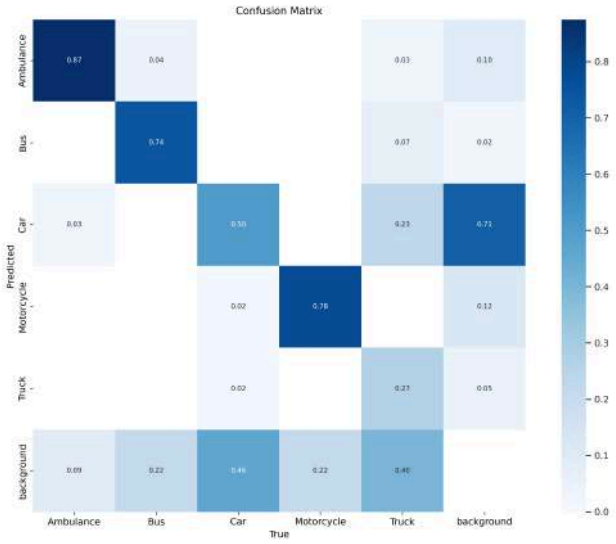

Figure 3: Confusion matrix (percentage)

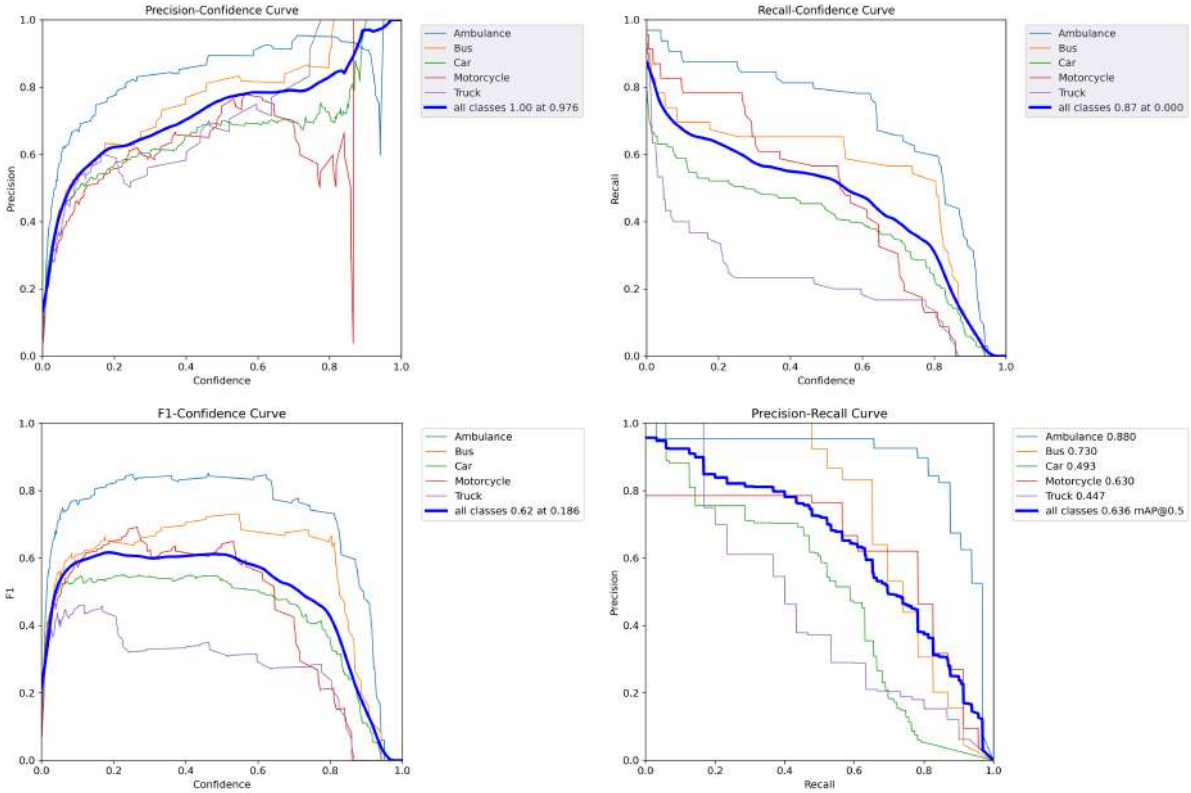

Figure 4: Performances of our YOLO network during training

After observing some important metrics of the model, we can proceed to test the network. First, we will check the predictions of the validation images saved during training. Then, we will also check the inferences on images and videos, both downloaded online and recorded by ourselves. In Figure 5, we can observe an example of a validation batch. Overall, we can say that the network works well. It misses some vehicles in the background, but it also correctly classifies difficult vehicles, like the bus behind the glass.

Figure 5: Example of validation batch

In Figure 6 and Figure 7, the model is tested on images and videos sourced online. As observed before, the model works well, but it often misses a lot of vehicles, especially if they are in the background. It's also interesting to see how the model handles vehicles that were not part of the training dataset, as shown in the second inferenced image. Another interesting observation from the inferenced videos is that the model is more likely to catch cars if they are facing away from the camera rather than towards it.

Figure 6: Inference on images sourced online

Figure 7: Inference on videos sourced online

On the unseen, real-world videos recorded by ourselves at different times of the day, the model performs well. Most of the cars near the camera are correctly detected, especially the parked ones. The model seems to struggle to distinguish between buses and trucks, while it usually correctly identifies motorcycles. It's also interesting to see that the model doesn't misclassify vehicles it wasn't trained on, like the electric scooter and the bicycle.

Figure 8: Inference on videos recorded by ourselves

5. Conclusions:

In this project, we successfully trained a small YOLOv5 model using the Vehicles-OpenImages dataset for detecting five different vehicle classes. During the validation phase, the model demonstrated good performance, accurately detecting and classifying vehicles, though it occasionally missed vehicles in the background or struggled with certain types like buses and trucks.

The inference on new images and videos confirmed that the model performs well on unseen data, correctly identifying vehicles even in varied conditions. It is noteworthy that the model handled vehicles it wasn't trained on without misclassification, which shows its robustness.

Testing on our recorded videos further demonstrated the model's real-world applicability, with strong performance in identifying nearby cars and motorcycles. However, the model encountered challenges with vehicles farther from the camera and distinguishing between similar classes. Additionally, not all classes were present in the inference videos.

Overall, the project demonstrated the effectiveness of the small YOLOv5 model in real-time vehicle detection, highlighting areas for future improvement, particularly in enhancing background vehicle detection and refining class distinctions.