Truck Trailer Detection

Szilveszter Milán
milan.szilveszter@stud.uni-obuda.hu

Abstract

This document presents an approach to truck trailer detection using the YOLOv11 object detection model. The paper outlines the process of data collection, preparation, and model training specifically tailored for identifying truck trailers. A dataset of 719 images was compiled using a custom Python script that captured images from Austrian highway web cameras. The YOLOv11 model was trained using Google Colab, with data split into training, validation, and test sets. The results show promising performance with an average precision of 0.88 and a mean average precision (mAP50) of 0.91. However, the model exhibits signs of overfitting and struggles with precise localization, as indicated by the drop in mAP50-95 to 0.77.

1 Introduction

Object detection is a critical task in computer vision, enabling machines to identify and locate objects within images or video streams. For industries like logistics, transportation, and fleet management, detecting specific objects such as truck trailers is essential for tasks like automated monitoring, traffic analysis, and safety compliance.

The purpose of this documentation is to guide you through the process of collecting the dataset and training the chosen pre-trained object detection model.

2 Choosing a Model

YOLOv11 has been selected for this task due to its high accuracy, speed, and computational efficiency. This document provides a step-by-step approach to ensure successful implementation, from data preparation to model deployment, tailored specifically for truck trailer detection scenarios.

3 Collecting Data

For gathering images for training, I have made a Python script that downloaded images every 15 minutes from Austrian highway web cameras. The script collected 13 camera's pictures.

After sorting the images and deleting useless pictures, the labeling process could be started. I have annotated the pictures with Label Img. After annotation, I have been left with 719 images.

4 Training

YOLO is an easy to use and well documented pre-trained object detection model. The training process is simple, but there are some prerequisites. For the training environment, I used Google Colab.

4.1 Splitting Data

Before training the model, I first split the data into training, validation, and test sets using a dedicated script (the split ratio is 7:2:1 respectively). The primary requirement is that the images and labels

must be stored in separate folders. This is necessary because YOLO uses a YAML configuration file to read the data.

The configuration file must specify the relative paths for the training and validation images. For annotations, YOLO automatically searches for the corresponding label files in the 'labels' directory within each dataset split (train, val).

In addition to specifying the paths for the data, we also need to define the number of classes to be detected and their corresponding names.

4.2 Training YOLO

The training process with YOLO is straightforward. After installing the Ultralytics Python package, training can be easily initiated by providing the necessary parameters.

Parameters used for training:

- epochs: 100
- batch: 32
- imgsz: 640

4.3 Results

The training results for the YOLOv11 model show mixed performance.

4.3.1 Evaluation Metrics

- **Precision:** Averaged at 0.88, peaking at 0.93.
- **mAP50:** The mean average precision reached an average of 0.91.
- **mAP50-95:** Averaged at 0.77, which is a significant drop from mAP50, indicating struggles with tighter bounding boxes.

The model shows signs of overfitting, with consistently higher validation losses compared to training losses. While the mAP50 score is good, the substantial drop in mAP50-95 suggests issues with precise localization. The precision and recall values, while respectable, indicate that the model misses some objects and has false positives. There's clear room for improvement in the model's generalization and accuracy, especially for tighter bounding box requirements. The quality of the training data, with too much repetition, plays a big role in this performance.

4.3.2 Confusion Matrices

- **True Positives:** 901 truck trailers correctly identified.
- **False Positives:** 185 background objects incorrectly classified as trucks.
- **False Negatives:** 95 trucks missed by the model.

The normalized confusion matrix shows that the truck detection rate is 90% which is good, but not perfect recall. 10% of trucks were missed and classified as background.

5 Conclusion

The trained YOLOv11 model for truck trailer detection demonstrates both potential and limitations in its current state. With a truck detection rate of 90% and an average precision of 0.88, the model shows promise for practical applications in transportation and logistics. However, the observed overfitting and the significant drop in performance for tighter bounding box requirements (mAP50-95) indicate areas for improvement.

The model's performance is likely influenced by the quality and diversity of the training data, which may have suffered from repetition due to the image collection method. Future work should focus on expanding and diversifying the dataset to enhance the model's generalization capabilities. Additionally, techniques to address overfitting, such as data augmentation or regularization, could be explored.