Digital Image Processing and Analysis Project Report:

Object Detection using YOLO and Faster R-CNN

Lovro Akmačić

0036537589

Faculty of Electrical Engineering and Computing

lovro.akmacic@fer.hr

Filip Buljan

0036539840

Faculty of Electrical Engineering and Computing

filip.buljan@fer.hr

Lucia Crvelin

0036540219

Faculty of Electrical Engineering and Computing

lucia.crvelin@fer.hr

Tomislav Čupić

00365411259

Faculty of Electrical Engineering and Computing

tomislav.cupic@fer.hr

Lorena Švenjak

0119039893

Faculty of Electrical Engineering and Computing

lorena.svenjak@fer.hr

May 29, 2025

Abstract

This report details a digital image processing and analysis project focused on object detection. We explore and implement state-of-the-art deep learning models, specifically YOLO (You Only Look Once) and Faster R-CNN (Region-based Convolutional Neural Network), for their efficacy in accurately identifying and localizing objects within images. The project aims to compare the performance, advantages, and limitations of these two prominent architectures in a practical application. We will discuss the theoretical underpinnings of each method, their practical implementation, and the insights gained from their application to a relevant dataset.

1 Description of Area and Activity

Digital image processing and analysis is a multidisciplinary field at the intersection of computer science, engineering, and mathematics, focusing on methods for manipulating and analyzing digital images. Its applications are vast, ranging from medical imaging and remote sensing to autonomous vehicles and security systems.

Our project delves into the sub-area of object detection, a fundamental task in computer vision that involves identifying instances of semantic objects of a certain class (e.g., humans, cars, animals) in digital images or videos and localizing them by drawing bounding boxes around them. The primary activity of this project involves the application and comparative analysis of two prominent deep learning-based object detection frameworks: YOLO and Faster R-CNN.

2 Overview of Methods Used

This section will provide a detailed technical overview of the two primary object detection methodologies employed in this project: YOLO and Faster R-CNN.

2.1 YOLO (You Only Look Once)

YOLO is a single-shot detector known for its speed and real-time processing capabilities. Unlike traditional object detection systems that separate the object detection pipeline into distinct stages (e.g., region proposal, classification, non-maximum suppression), YOLO processes the entire image in a single pass. It divides the image into a grid and simultaneously predicts bounding boxes and class probabilities for each grid cell. We will discuss its architectural components, including the backbone network, detection heads, and the loss function used for training.

3 Implementation of YOLO for Car Detection

Our YOLO implementation focuses on efficiently detecting and counting cars in images. The core logic revolves around leveraging a **pre-trained YOLOv8 model** from the `ultralytics` framework. This choice significantly streamlines the development process by utilizing a model already trained on a vast dataset (COCO), thus benefiting from **transfer learning** and eliminating the need for extensive custom training.

The implementation follows a clear workflow:

1. **Model Initialization**: A `YOLO` object is instantiated, loading the pre-trained weights (e.g., `yolov8l.pt`). This prepares the deep learning model for inference.

2. **Inference and Result Parsing**: For each input image, the model performs a single forward pass, predicting all potential objects. The results, which include **bounding box coordinates, class IDs, and confidence scores**, are then extracted.

3. **Car-Specific Filtering**: A crucial step involves filtering these raw detections. Since our goal is car detection, we specifically select detections where the `class_id` corresponds to 'car' (which is class ID 2 in the COCO dataset). This ensures that only relevant objects are processed further.

4. **Visualization and Counting**: For each confirmed car detection, its bounding box is drawn on the image, along with a label indicating its class and confidence score. A running count of detected cars is maintained and displayed on the image. This visual feedback is essential for immediate verification of the model's performance.

5. **Automated Batch Processing**: The system is designed to handle entire datasets. It iterates through specified input directories (e.g., `train`, `val`, `test` subsets), automatically loading each image, applying the detection logic, and saving the annotated output to a designated folder.

In essence, the implementation provides a robust and automated pipeline for car detection, focusing on clear, efficient use of a powerful pre-trained model combined with standard image processing techniques for visualization and data handling.

3.1 Faster R-CNN (Region-based Convolutional Neural Network)

Faster R-CNN is a two-stage object detector that significantly improved upon its predecessors (R-CNN and Fast R-CNN) by introducing the Region Proposal Network (RPN). The RPN is a fully convolutional network that simultaneously predicts object bounds and objectiveness scores at each position. The proposed regions are then fed into the Fast R-CNN detection network for classification and bounding box regression refinement.

4 Implementation of Faster R-CNN: Pre-trained vs. Fine-tuned

Our project implemented Faster R-CNN using two distinct strategies to detect cars: one leveraging a **generically pre-trained model** and another employing a **fine-tuned model**. Both methods use the `torchvision` library for model handling.

4.1 Pre-trained Faster R-CNN

This approach directly utilizes a Faster R-CNN model pre-trained on the **COCO dataset**. The model comes with extensive knowledge of 80 different object categories, including cars.

- **Logic**: The model is loaded with its original weights, allowing it to detect cars based on its general understanding from the diverse COCO training data. Car identification relies on matching detected objects to the 'car' class ID (class ID 3) within COCO's existing classifications.
- **Application**: This method is straightforward, requiring no additional training. It applies a universal object detector to our specific task.

4.2 Fine-tuned Faster R-CNN

This strategy involves adapting a pre-trained Faster R-CNN model to specialize in car detection through **fine-tuning**.

- **Logic**: Instead of using the generic COCO weights entirely, we load a model that has undergone additional training on a car-specific dataset. This process re-trains the model's final layers, adjusting its classification capabilities to primarily distinguish between "background" and "car" (often mapping 'car' to class ID 1 in this specialized setup).
- **Application**: Fine-tuning allows the model to learn more precise and relevant features for car identification, potentially yielding higher accuracy and better performance specifically for our target domain compared to the generically pre-trained version.

By comparing these two implementations, we analyze the impact of **transfer learning** and task-specific adaptation on the model's effectiveness in car detection.

5 Results

This section presents the results of our experiments with YOLO and Faster R-CNN on a chosen dataset.

6 Conclusion

References