UNIVERSITÀ
di VERONA

Master's Degree in Artificial Intelligence

Car Detection using YOLO v8 and Faster R-CNN

Professor:

Professor Vittorio Murino

Course:

Computer Vision and Deep Learning

Author:

Saman Nourani - VR512633

Academic Year 2023-2024

Universita di Verona, Verona, Italy

Contents

Motivation

The ability to recognize cars in urban areas and on public roads and streets using satellite imagery holds significant promise for numerous vital applications.

Protection of the Environment: Quick identification is necessary to stop environmental
Search and Rescue Operations: Helping with unintentional vehicle missions is part of search and rescue operations.

Road management: includes optimizing the asphalt on the road and the car body.

Manage cars:
Air pollution: is the term used to describe the effects of new or old cars on the environment and the air quality, which affects all living things, including plants, animals, and humans.
Noise pollution: Many car horns and outdated vehicles contribute to noise pollution.
The goal of this research is to precisely diagnose the vehicle using satellite imagery, thereby improving road and street safety as well as environmental sustainability.

State of Art

Models:

- CNN is Used for automatic feature extraction
- YOLO
- Faster R-CNN

Techniques:

- Data Augmentation: Enhances training data diversity.
- Transfer Learning: Fine-tuning pre-trained models.
- Cleaning data set

Algorithms:

- YOLO (v8)
- Faster RCNN

Dataset

A gallery of over a thousand images featuring cars. These images are all part of the "Car" class.

Methodology

Two common frameworks used in car detection projects are Faster R-CNN and YOLO v8. You Only Look Once version 8, or YOLO v8, is well-known for its ability to recognize objects in real time. It works by creating a grid out of the input image and using those grid cells to estimate bounding boxes and class probabilities. Because of this method, YOLO v8 may attain remarkable speeds while retaining accuracy, which makes it appropriate for applications that need quick inference, such real-time car recognition in live feeds.

YOLO v8 (version 8 you only look once):

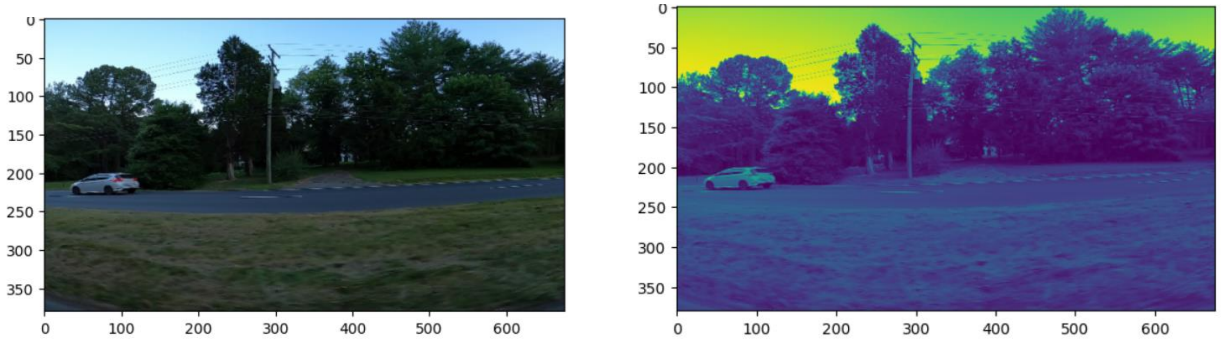
The sophisticated real-time object identification model YOLO v8 is renowned for its simplicity and quickness.

One-shot detection: YOLO v8 predicts bounding boxes and class probabilities in a single pass of the grid by first splitting the input image into a grid.

Speed and efficiency: Suitable for applications like automated driving or traffic monitoring where quick recognition of vehicles and other objects is essential, this system is optimized for real-time inference. While in some instances and earlier iterations,

YOLO's speed and accuracy were marginally slower than Faster R-CNN's, both attributes have greatly improved in the most recent iterations.

First, I get an image to set a RGB, to set and know warm and cold colors to recognize cars.



I have a lot of image dimensions as 676 pixels wide and 380 pixels tall.

	image_name	class	x_centre	y_centre	width	height
192	vid_4_18840.jpg	0	0.532200	0.549469	0.210564	0.140283
441	vid_4_6280.jpg	0	0.755789	0.521799	0.136758	0.115830
464	vid_4_6480.jpg	0	0.096599	0.573279	0.142547	0.097812
277	vid_4_2180.jpg	0	0.259768	0.578427	0.159190	0.115830
506	vid_4_9240.jpg	0	0.325564	0.518661	0.090313	0.076425

I produce a figure and a 2x2 grid of subplots, each measuring 12 by 12 inches. Also I reduce the 2D axis array to a 1D array. It becomes simpler to iterate over every subplot as a result. Select randomly 4 images. For each image; first, constructs the full path to the image file, then read the image using OpenCV and convert it from BGR to RGB color format.

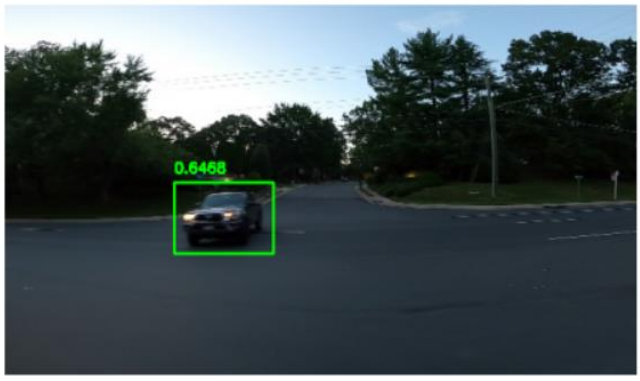
Output result:

The result is a grid of 2x2 subplots with anticipated bounding boxes superimposed over randomly chosen images.

Bounding Boxes: Every subplot has an image with one or more bounding boxes, which show the expected locations and sizes of observed objects, such as cars, drawn around the object.

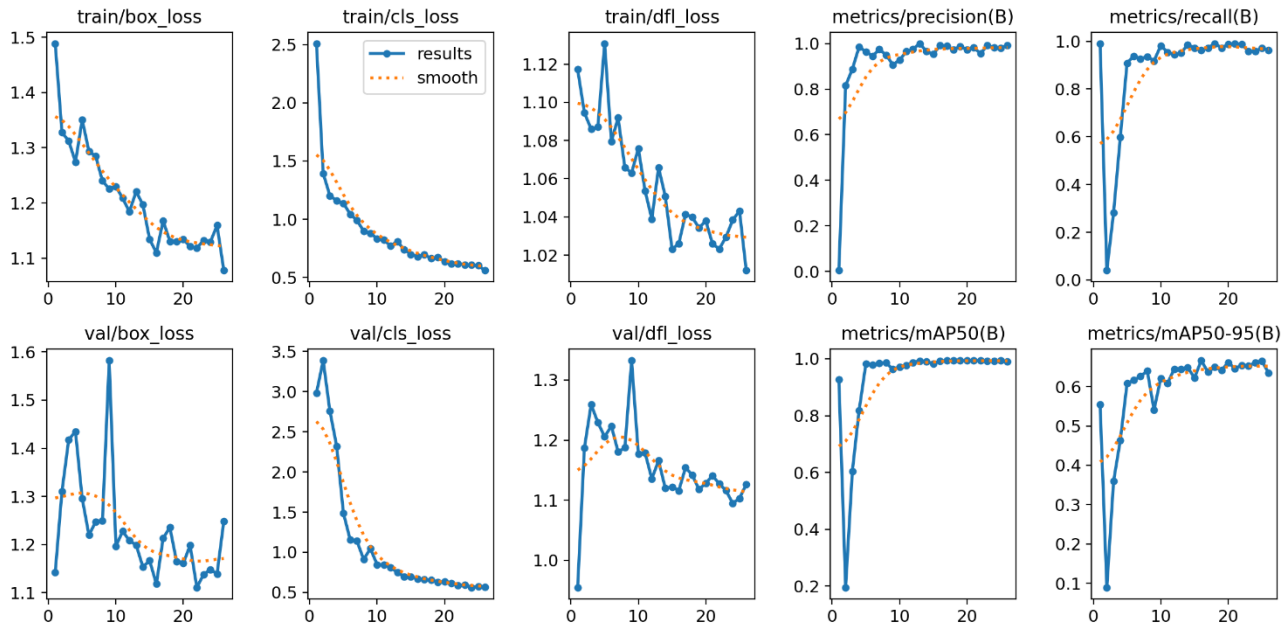
Scores: The numbers next to each box denote the model's level of confidence in the presence of an object at that specific location.

File Output: The finished result is stored as car_detect_yolov8.png, providing a visual depiction for future use or examination.



Loss Function: The loss function in my project is an essential indicator that measures the training performance of my machine learning model. It basically calculates the difference between my model's expected outputs and the actual ground truth labels. In order to maximize the model's capacity for precise prediction, the loss function seeks to reduce this disparity as training progresses.

Generally, typical loss functions for object detection tasks like yours are variations of the mean squared error (MSE), cross-entropy loss, or more specialized losses designed for particular tasks like bounding box regression's Intersection over Union (IoU) loss. In order to reduce overfitting and enhance accuracy, these algorithms compute the difference between expected and actual values, frequently using regularization terms.



Confusion matrix:

I have to calculate some metrics such Precision, Recall, F1 Score and Accuracy. A table that lists a classification model's performance is called a confusion matrix. It measures

the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) by comparing the projected labels with the actual labels.

	Predicted: Car	Predicted: No Car
Actual: Car	500	2
Actual: No Car	0	1

Metrics	Precision	Recall	F1 Score	Accuracy
Numbers	0.75	1.0	0.8571	0.75

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

The precision and resilience of the two-stage Faster R-CNN object detection framework are well established.

Key features include:

- **Two-stage Detection**: Initially, a Region Proposal Network (RPN) is used by Faster R-CNN to suggest regions of interest (RoIs) in the picture that might include objects. After that, it categorizes these suggestions and improves their bounding boxes.
- **Accuracy and Precision**: Its high accuracy is attained by concentrating on accurate object localization and thorough feature extraction, which makes it appropriate for applications where very accurate car detection is essential, like in surveillance systems or manufacturing quality control.
- **Complexity**: Because of its two-stage methodology, Faster R-CNN requires more computing power than YOLO, which could result in slower inference times. But in recent times, improvements in technology and optimization methods have narrowed this disparity.

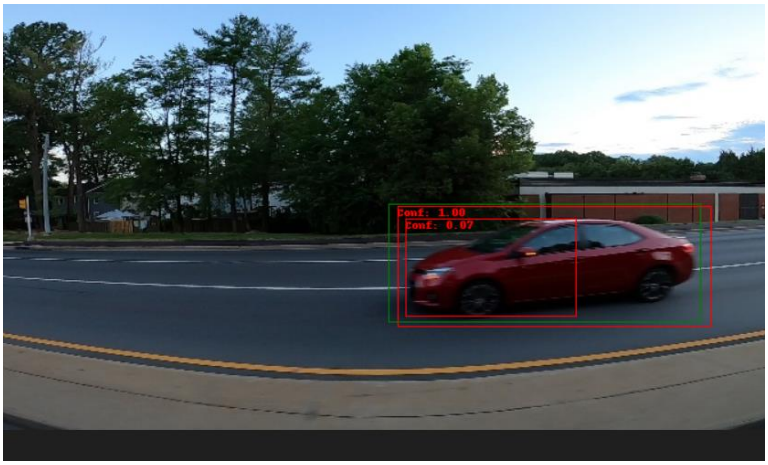
Visualize Batch: The purpose of the visualize batch function is to visually examine how well an object identification model performs on a batch of photographs taken from a dataset. It facilitates comprehension of the model's object detection and bounding box prediction accuracy.

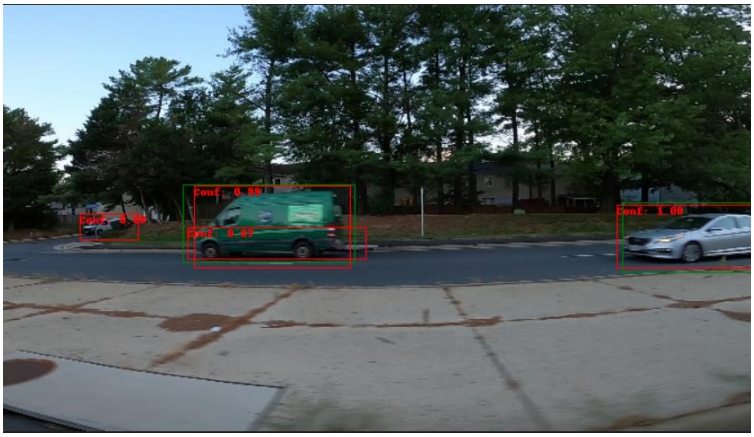
Predicted Items: Bounding boxes that the object detection model predicts to represent the positions of items inside an image are called predicted bounding boxes. A confidence score that expresses the model's level of assurance that the identified object is inside each anticipated bounding box is usually included.

Predicted Classes: Object detection models not only anticipate bounding boxes but also the class labels of objects that are observed. For instance, if the model recognizes an automobile, a human, or any other type of item that it was trained on.

Ground Items: Ground Truth Bounding Boxes: Presented in the dataset for training and maybe evaluation, these are the actual bounding boxes of the objects in the image. Bounding boxes with ground truth indicate the actual positions of items based on human annotations or other trustworthy sources.

Ground Truth Classes: Correct class labels for items in the image are included in ground truth annotations, in addition to bounding boxes. The model uses these labels as a benchmark to compare its predictions to.



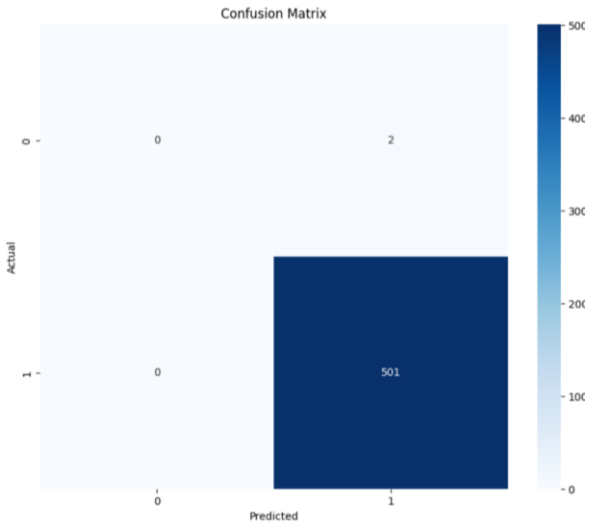


Confusion matrix:

Different prediction outcomes are represented by each cell in the matrix. This is too much important because of first, gives a thorough explanation of the forecast flaws, second, aids in comprehending the model's advantages and the last one disadvantages and beneficial for enhancing class imbalance or modifying the model threshold.

	Predicted: Car	Predicted: No Car
Actual: Car	0	2
Actual: No Car	0	501

Metrics	Precision	F1 Score	Accuracy
Numbers	1.00	0.9980	0.9960



Faster R-CNN Challenges:

Complexity of Implementation: Faster R-CNN, with its two-stage design consisting of a Region Proposal Network (RPN) and subsequent classification and bounding box regression, is more difficult to construct than single-shot detectors such as YOLO. It can be difficult to ensure proper integration and comprehension of various components.

Training Data Preparation: It can be time-consuming and error-prone to efficiently prepare and annotate the training dataset (pictures and related bounding boxes). To train a robust model, it is imperative to ensure that the dataset has enough diversity and high-quality annotations.

Computational Intensity: Faster R-CNN training and inference can be computationally demanding, particularly when handling big datasets and intricate network topologies. This can necessitate having access to strong GPUs and making effective use of processing power.

Hyperparameter Tuning: Achieving optimal performance requires optimizing hyperparameters for the RPN, such as learning rate, batch size, and anchor box designs. It can be difficult to strike the correct balance between convergence and training speed in order to prevent either overfitting or underfitting.

Fine-tuning and Transfer Learning: Given my unique car identification dataset, fine-tuning pre-trained models such as Faster R-CNN necessitates careful consideration of transfer learning approaches. It is crucial to make sure the model generalizes adequately to new data and various environmental situations.

Experiments and Results:

Experiments:

YOLO v8: The Ultralights YOLO framework is used to develop YOLO v8. Predictions are made, the model is trained, the dataset is prepared in YOLO format, and metrics like mean average precision (mAP) are assessed.

Faster R-CNN: The torchvision package in PyTorch is used to construct faster RCNN. The process entails configuring the model, generating a unique dataset class, utilizing the SGD optimizer to train the model, and assessing the results using metrics like confusion matrix, average precision (AP), and precision.

Results:

YOLO v8: Testing images are used to generate predictions for the YOLO v8 model, which was trained using data on automobile object detection. An evaluation is conducted of the average precision measure (mAP) at 0.50 intersection over union (IoU).

Faster R-CNN: Using the same automobile object identification dataset, the Faster RCNN model is trained, and comprehensive assessment metrics like accuracy and AP are calculated. Subsequently, the model is applied to forecast previously unknown test photos.

Conclusion:

Better Model: Depending on certain needs, such the type of the objects being identified and the trade-offs between speed and accuracy, one can choose between Faster RCNN and YOLO v8. For real-time applications that require faster inference times, YOLO might

be the better option, but tasks requiring exact object localization might go for Faster RCNN.

Faster Model: Because it is a single-shot detection model, YOLO inference is usually faster, but Faster RCNN requires more processing because of its two-stage architecture.

Greater Accuracy: The accuracy measure (mAP/AP) for each model can be high, but it depends on the evaluation criteria and dataset that are employed.

References:

Dataset from Kaggle: https://www.kaggle.com/datasets/sshikamaru/car-object-detection/code

GitHub: https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8x.pt