Vehicle Speed Estimation

1. Project Overview

This machine learning project utilizes YOLOv8 to detect and track vehicles in video frames, trained on a meticulously annotated custom dataset. The system incorporates speed estimation by defining a Region of Interest (ROI) within each frame. Utilizing OpenCV, the ROI enables precise speed calculations based on vehicle displacement over time. The outcome is a powerful combination of vehicle identification, tracking, and real-time speed estimation, applicable for traffic monitoring and law enforcement tasks.

2. Requirement

Python `3.8.10`
Ultralytics `8.0.183`
Pandas `2.0.3`
NumPy `1.23.0`
Cv2 `4.8.0`

3. Data Preparation

The data preparation stage of the machine learning project is where we lay the foundation for successful model training. This crucial step involves gathering, cleaning, and organizing the raw data, transforming it into a structured and meaningful format that can be readily utilized for model training.

The data preparation script performs a series of essential tasks, including organizing the dataset into a format suitable for machine learning model training. Following are the functions and logic of the code.

- preprocess: It organizes and copies image files from a source directory (src_root) to a destination directory (dst_root). It creates subdirectories for training and testing images and a directory for labels with IDs. The function traverses the source directory, identifies and copies image files to the appropriate destination directories, and creates a structured hierarchy for an MOT dataset.

- draw_ignore_regions: It takes an input image (IMG) and a list of bounding boxes (boxes) and blacks out the specified regions in the image.

- gen_labels: Processes XML annotations of a multi-object tracking dataset, blackens specified regions in corresponding images, generates tracking labels, and optionally visualizes the results.

- find_file_with_suffix: Recursively finds files with a specific suffix in a root directory and appends them to a list.

- count_files: Counts the number of image files and label (txt) files in specified directories.

- clean_train_set: Cleans up mismatches between the number of image files and label (txt) files in training sets by removing the corresponding problematic files.

Upon successful execution, you can anticipate a well-prepared dataset stored in the specified output directory.

The resultant prepared data should be precisely in the below-mentioned format.

train/ |-- images/
| |-- 1.jpg
| |-- 2.jpg
| |-- ...
|-- labels/
| |-- 1.jpg
| |-- 2.jpg
| |-- ...
|
val/ |-- images/
| |-- 1.jpg
| |-- 2.jpg
| |-- ...
|-- labels/
| |-- 1.jpg
| |-- 2.jpg
| |-- ...
|
test/ |-- images/
| |-- 1.jpg
| |-- 2.jpg
| |-- ...

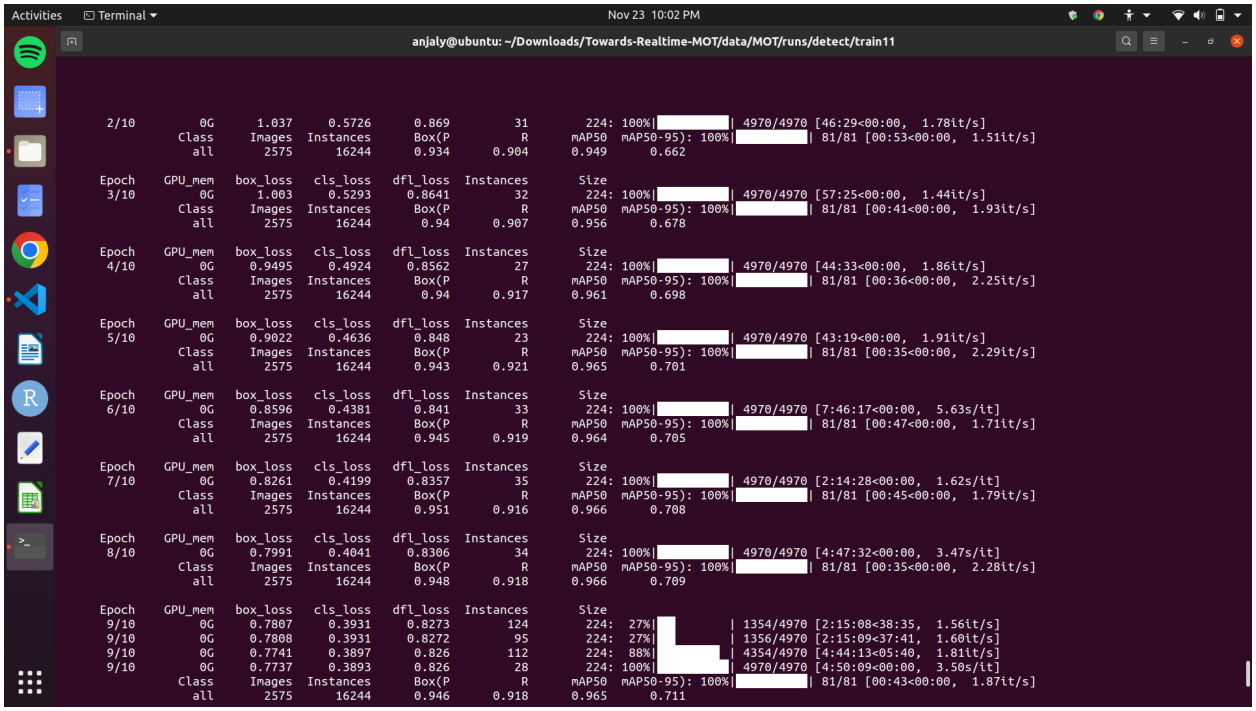
4. Model Training

The model training script orchestrates an iterative learning process over the prepared dataset using the yolov8 model by Ultralytics. It will be trained to detect vehicles like cars and vans, guiding the machine-learning model through multiple epochs to refine its parameters. Notably, the training process is configured to run for 10 epochs, each representing a complete pass through the dataset. Additionally, a specific image size (imgsz) of 244 pixels is employed, contributing to the model's ability to extract meaningful features from the input data.

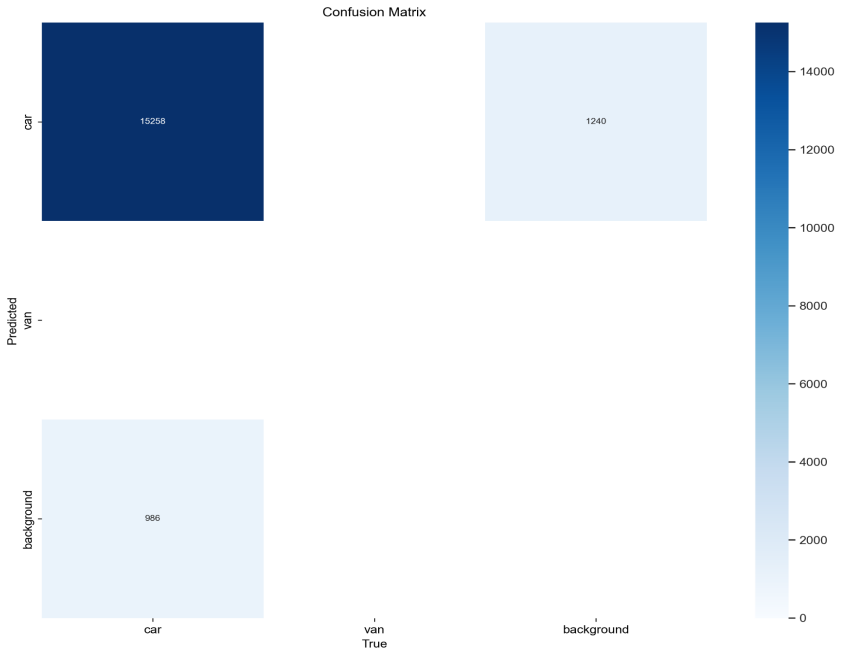
Upon successful execution, expect the script to produce a trained weights file(which will be found in a folder named runs) ready for deployment. The model will have learned from the input data, capturing essential features and relationships, enabling it to make predictions on new, unseen data. The expected

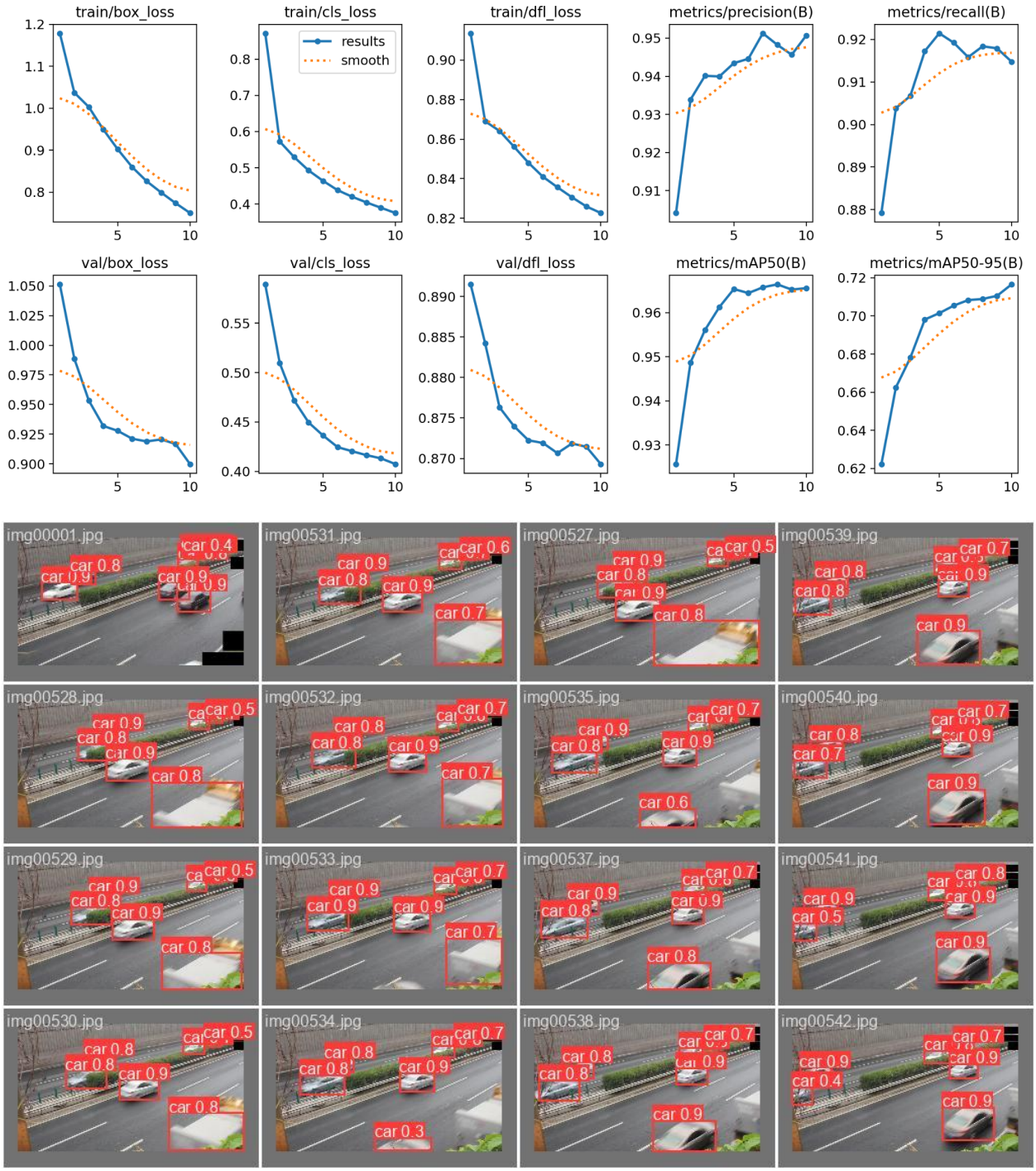
results include model performance metrics, such as accuracy and loss, providing insights into the effectiveness of the trained model. These outcomes are critical for evaluating the model's capability to generalize to real-world scenarios and make accurate predictions.

Example output of the model training:



Training results:





5. Speed Estimation

The speed estimation script performs object detection and tracking on video frames, defining a specific Region of Interest (ROI) where vehicle speed is to be measured. By precisely calculating the displacement of vehicles within the ROI over time, the script accurately finds the speed of each passing vehicle.

Logic of the code:

- Region of Interest (ROI) Definition: The speed estimation begins by defining a Region of Interest (ROI) within each video frame. This region is strategically chosen to encompass the path along which the speed of vehicles will be measured.

- Bounding Box and Center Point Extraction: Object detection identifies and draws bounding boxes around vehicles in each frame. The center point of each detected vehicle is extracted from these bounding boxes.

- Line Placement: Two lines, referred to as Line 1 and Line 2, are drawn on each video frame within the defined ROI. These lines serve as reference points to measure the time a vehicle travels between them.

- Time Recording: The code records when a vehicle's center point touches Line 1 and Line 2. This is achieved by timestamping the frame at each line intersection.

- Speed Calculation: The time elapsed between Line 1 and Line 2 crossings is measured. Simultaneously, the known distance between Line 1 and Line 2 within the ROI calculates the vehicle's speed. The speed is determined using the formula Speed = Distance / Time.

- Repetition Across Frames: This process is repeated for each frame in the video, providing a continuous and dynamic calculation of the speed of vehicles as they traverse the specified ROI.

Upon successful execution, expect a comprehensive output. The script generates a full-fledged video showcasing the seamless detection of objects and the simultaneous display of their respective speeds. A text file is also produced detailing the speed information for each identified vehicle. This holistic approach to speed estimation provides visual insights and creates a structured data output for further analysis and application in diverse domains such as traffic monitoring and law enforcement.

6. Results

In conclusion, the culmination of our efforts in this machine learning project yields compelling results. The trained model, honed through meticulous data preparation and model training phases, showcases a remarkable ability to detect cars and vans accurately. When applied to video data, the model integrates object detection with speed estimation, providing a visually informative output. The generated video dynamically displays the speed of various vehicles, offering a tangible representation of the model's real-world applicability. Simultaneously, the accompanying text file furnishes comprehensive speed details for each detected vehicle, facilitating further analysis and insights. These outcomes collectively underscore the efficacy of our machine learning approach in addressing complex tasks such as vehicle detection and speed estimation, making it a valuable asset in diverse applications ranging from traffic management to public safety.