Problem Statement

Title: Vehicle Cut-In Detection

Objective:

To develop a real-time system for detecting potential vehicle collisions, specifically cut-in scenarios, using machine learning and computer vision techniques. The system aims to enhance road safety by providing early warnings to drivers, thus preventing accidents and improving overall traffic management.

Unique Idea Brief (Solution)

This project provides a unique solution for vehicle cut-in detection by leveraging advanced computer vision and machine learning techniques. Utilizing the YOLOv5 model for real-time vehicle detection, it ensures high accuracy and efficiency in identifying and tracking vehicles. The project implements a custom tracking algorithm to maintain vehicle identities across frames, coupled with a distance calculation mechanism to monitor the proximity of vehicles. The system provides timely warnings for potential cut-ins, enhancing driver safety. By integrating video re-encoding for compatibility and robust preprocessing for varying conditions, the solution offers a comprehensive and reliable approach to real-time vehicle cut-in detection.

Features Offered

 Real-Time Vehicle Detection:

* Utilizes advanced object detection algorithms to identify vehicles in real-time from video streams.
* Capable of handling various environmental conditions such as different lighting, weather, and traffic scenarios.

 Accurate Distance Measurement:

* Calculates the distance between the detected vehicles and the target vehicle with high precision.
* Ensures reliable measurements even at different speeds and in dynamic traffic situations.

 Collision Warning System:

* Generates visual warnings on the video feed when a vehicle gets too close, indicating a potential collision.
* Helps drivers take proactive measures to avoid accidents by providing timely alerts.

 Integration with Pre-Trained Models:

* Leverages the SSD MobileNet model, a state-of-the-art pre-trained model for efficient and accurate vehicle detection.
* Reduces the need for extensive training data and accelerates the development process.

 Scalability and Flexibility:

* Designed to be scalable, allowing integration into various vehicle systems and adaptable to different vehicle types.
* Can be customized and extended to include additional features as needed.

Processflow

 Data Collection:

* Collect video footage from various sources such as traffic cameras, dashcams, or simulation environments.
* Annotate the collected video frames to create labeled datasets for training and testing the model.

 Data Preprocessing:

* Convert the annotated video frames into a suitable format for training the machine learning model.
* Generate TFRecord files from the annotated data to streamline the training process.

 Model Selection and Configuration:

* Choose a pre-trained object detection model (e.g., SSD MobileNet) from TensorFlow’s Model Zoo.
* Configure the model’s hyperparameters and the pipeline configuration file to suit the project’s requirements.

 Training the Model:

* Set up the training environment and dependencies using Python and TensorFlow.
* Train the model using the generated TFRecord files and the configured pipeline.
* Monitor the training process and adjust parameters as needed to improve model performance.

 Model Evaluation:

* Evaluate the trained model using a separate set of test data to assess its accuracy and performance.
* Fine-tune the model based on evaluation results to optimize detection accuracy and reduce false positives.

Technologies used

 Python: Primary programming language for its extensive libraries.

 PyTorch: Deep learning framework for building and training neural networks.

 YOLOv5: Real-time object detection model for vehicle detection.

 OpenCV: Computer vision library for video capture, frame processing, and writing output files.

 NumPy: Library for numerical operations and distance calculations.

 PyCharm: Integrated Development Environment (IDE) for efficient Python development.

 cv2.VideoCapture: Captures video from files, sequences, or cameras.

 cv2.VideoWriter: Writes processed video frames to a new video file.

 Custom SimpleTracker Class: Tracks detected vehicles across video frames using IoU.

 Euclidean Distance Calculation: Calculates distance between vehicle centers using NumPy.

Conclusion

This project has implemented a comprehensive pipeline for vehicle cut-in detection using deep learning and computer vision techniques. The use of YOLOv5 for object detection ensures accurate and efficient vehicle detection. The tracking algorithm maintains the continuity of detected vehicles across frames, and the distance calculation mechanism identifies potential cut-ins, issuing warnings when necessary. This approach can be further extended and refined for real-world applications, such as advanced driver-assistance systems (ADAS) and autonomous driving technologies.