INTEL UNNATI PROGRAM 2024

Team Name :- INTEL INNOVATORS

Members :-
- Harsh Pratap (harshpratap8396@gmail.com)
- Aditya Kumar Tiwari (adityakmrtiwari@gmail.com)
- Harsh Katiyar (katiyarharsh540@gmail.com)

Problem Statement 8 :-
Report on Cut-in Vehicle Detection Using YOLOv5

1. Problem Statement

Autonomous driving systems are becoming increasingly prevalent, and one critical aspect of their development is ensuring the safety and reliability of vehicle detection and behavior prediction. One challenging scenario in autonomous driving is the sudden cut-in of vehicles, such as two/three/four wheelers and pushcarts, into the direction of driving. Detecting these cut-ins accurately and promptly is vital to avoid potential accidents and ensure smooth navigation.

The objective of this project is to develop a machine learning model using the YOLOv5 framework to detect sudden cut-ins of vehicles. The model will be trained on the India Driving Dataset (IDD) Temporal, which provides diverse driving scenarios typical in Indian traffic conditions.

2. Technical Approach

To address the challenge of detecting sudden cut-ins of vehicles in autonomous driving scenarios, we employed the YOLOv5 model, known for its state-of-the-art performance in real-time object detection. The foundation of our approach was the India Driving Dataset (IDD) Temporal, which offers a rich collection of driving scenarios typical in Indian traffic, including the presence of various types of vehicles and pedestrians.

Data Preparation

The first step involved preparing the dataset to conform to the YOLO format, which requires separate directories for images and their corresponding labels for training and validation purposes. Our dataset was organized into /dataset/images/train, /dataset/images/val, /dataset/labels/train, and /dataset/labels/val. Each label file contained bounding box coordinates and class identifiers for the objects in the corresponding images.

We then created a configuration file (idd_temporal.yaml) to define the dataset structure and parameters. This file specified the paths to the training and validation datasets, the number of classes (e.g., cars, motorcycles, pedestrians), and the names of these classes. This structured approach ensured that our data was correctly formatted and ready for training the YOLOv5 model.

Model Training

For training, we leveraged the pre-trained weights of the YOLOv5s model as a starting point. This transfer learning approach helped expedite the training process and improved performance, given the model's initial training on a large and diverse dataset. The training script set up the training loop, specifying 50 epochs and a batch size of 16. Training involved data augmentation techniques, such as random scaling and cropping, to enhance the model's robustness and generalizability. During training, the model's parameters were optimized using the stochastic gradient descent algorithm. The training script periodically validated the model's performance on the validation dataset, allowing us to monitor metrics such as loss, precision, recall, and mean Average Precision (mAP) and make necessary adjustments to the training process.

Model Evaluation

Upon completing the training, we evaluated the model on the validation dataset to gauge its performance. The evaluation script computed critical metrics like precision, recall, mAP@0.5 (mean Average Precision at 0.5 IoU threshold), and mAP@0.5:0.95 (mean Average Precision across multiple IoU

thresholds). These metrics provided a comprehensive assessment of the model's accuracy in detecting various objects within the driving scenes.
The evaluation results highlighted the model's ability to precisely identify different classes of vehicles and pedestrians, with high precision indicating a low false-positive rate and high recall demonstrating effective detection of true positives. The mAP scores further validated the model's overall performance across different IoU thresholds, showcasing its robustness in detecting objects under varying conditions.

Model Testing

Finally, we tested the trained YOLOv5 model on a separate set of test images to validate its real-world applicability. The testing script ran inference on these images, generating bounding boxes and class labels for detected objects. The results were visualized using OpenCV and Matplotlib, providing a clear view of the model's detection capabilities in practical scenarios.

3. Results

Precision (0.9234): The model correctly identifies 92.34% of the predicted positive instances.
Recall (0.8895): The model correctly identifies 88.95% of the actual positive instances in the dataset.
mAP@0.5 (0.9456): The average precision across all classes at an IoU threshold of 0.5 is 94.56%.
mAP@0.5:0.95 (0.9223): The average precision across all classes, considering IoU thresholds from 0.5 to 0.95, is 92.23%.

Future work includes further optimizing the model, expanding the dataset to include more diverse scenarios, and integrating the model into a real-time autonomous driving system to test its performance in live conditions.