Traffic Violation Detection System Report

Project Overview

The Traffic Violation Detection System leverages AI and computer vision to automate traffic law enforcement by detecting vehicle license plates from traffic images or video feeds. The system is built on a deep learning model trained to recognize and extract license plate information from vehicles. The goal of this system is to identify violations such as speeding, running red lights, or illegal parking by cross-referencing detected plate numbers with a violation database.

Technology Stack

- Deep Learning Framework: YOLOv5 (based on PyTorch)
- Model for Detection: YOLOv5 for vehicle license plate detection
- OCR for Plate Number Extraction: Tesseract OCR
- Data Storage: Database for storing violation records
- Programming Language: Python

Training Process

The model was trained using YOLOv5 (a state-of-the-art object detection model based on PyTorch) to detect vehicle license plates. The training process involved the following steps:

1. **Dataset Preparation**:
    - Images of vehicles in various traffic scenarios.
    - Annotations for license plates (bounding box coordinates) for training the model.
    - `.yaml` configuration file to define the dataset and training parameters.
2. **Model Architecture**:
    - YOLOv5 was chosen due to its speed and accuracy in real-time object detection. This model uses convolutional neural networks (CNNs) to detect objects in images, such as license plates in vehicles.
3. **Training Configuration**:
    - Image size: 640x640
    - Batch size: 16
    - Epochs: 70
    - Pre-trained weights: `yolov5l.pt` (YOLOv5 large version)
    - The model was trained on Google Colab with GPU acceleration.
4. **Results**:

- The model was trained successfully, and the best weights (`best.pt`) were saved for future inference. The model learns to detect license plates even in different lighting conditions, angles, and vehicle types.

Detection and Violation Reporting

Once the model is trained, it can be used for inference on new images or video streams to detect vehicles and their license plates.

1. **Inference**:
   - The trained model uses `detect.py` to process new images, outputting detected license plates with bounding boxes.
2. **OCR for Plate Extraction**:
   - **Tesseract OCR** is used to extract text from the detected license plates, providing the vehicle's plate number.
3. **Violation Detection**:
   - The detected plate numbers are checked against a **violation database** (which could contain records of previous violations like running a red light or speeding).
   - If a match is found, the system generates an automatic **violation report**, including the detected plate, timestamp, and type of violation.
4. **Report Generation**:
   - The system generates and stores reports for traffic law enforcement. It can also send automated notifications or fines to the vehicle owner based on the detected violation.

Framework Used

- **YOLOv5**: A cutting-edge object detection model implemented in PyTorch.
  - PyTorch was used for the training and inference of the YOLOv5 model. The model was pre-trained on a large dataset and fine-tuned on your specific traffic dataset to recognize license plates.
  - The YOLOv5 framework uses PyTorch for building, training, and inference, making it a flexible and powerful solution for object detection tasks.

Conclusion

The Traffic Violation Detection System successfully detects license plates in traffic images and cross-references them with violation records to automate law enforcement. By using YOLOv5 (PyTorch) for vehicle detection and Tesseract OCR for plate extraction, the system can

accurately identify vehicles and generate automated violation reports. This system has the potential to enhance traffic monitoring, improve enforcement accuracy, and reduce human intervention in traffic violations.

Next Steps

1. **Fine-Tuning**: Continue fine-tuning the model on a more extensive dataset to improve accuracy.
2. **Real-Time Implementation**: Integrate the model into a real-time traffic monitoring system using video feeds.
3. **Violation Database**: Develop and integrate a robust violation database to manage and store detected violations.
4. **Web Interface**: Build an interface to view and manage violations and generate automated notifications for fines.

Project Completed By: Talha Ahmad & Syed Dilawar
F2019266302 & F2019266273