PLATE-NET AI

Car License Plate Detector using YOLOv8

Submitted by

Shiza Shabbir

National University of Modern Languages (NUML)

Supervisor

Dr. Inayat Ullah Khan

August 25, 2025

Abstract

Automatic vehicle license plate detection plays a vital role in modern intelligent transportation systems, traffic monitoring, parking management, and law enforcement. Traditional computer vision techniques for license plate detection were limited in accuracy and struggled under challenging conditions such as varying lighting, complex backgrounds, and different plate styles.

This project, titled **Plate-Net AI**, proposes a deep learning-based solution utilizing the YOLOv8 model to accurately detect vehicle license plates from static images, recorded videos, and live webcam streams. The dataset was collected from Kaggle and trained using Google Colab with GPU support. The final trained model weights (`best.pt`) are deployed locally in a Python-based system optimized for CPU-only environments.

The system is interactive, allowing users to choose between image, video, or webcam detection, and automatically saves annotated results. Extensive testing confirmed its robustness and efficiency, even when running on CPU hardware without GPU acceleration.

This work demonstrates the potential of YOLO-based object detection models in real-world intelligent systems and lays the foundation for further enhancements, including optical character recognition (OCR) for plate number extraction and integration with IoT-based smart traffic management systems.

Contents

1 Introduction

License plate detection is a crucial task in computer vision, widely used in intelligent transportation systems, traffic management, law enforcement, and surveillance. With the rapid growth of deep learning techniques, models such as YOLO (You Only Look Once) have significantly improved the accuracy and speed of object detection tasks.

This project, named **Plate-Net AI**, applies YOLOv8 to detect car license plates. The project ensures user-friendly deployment by providing a single Python script that supports multiple input modes (image, video, webcam). Unlike resource-heavy GPU setups, this system has been optimized for CPU environments, making it accessible for real-world applications on ordinary machines.

2 Methodology

2.1 Dataset

The dataset used for training was obtained from Kaggle. It consisted of labeled car images with visible license plates. The data was annotated in YOLO format for training purposes.

2.2 Model Training

- Framework: Ultralytics YOLOv8
- Platform: Google Colab (GPU)
- Training Strategy: Transfer learning from pre-trained YOLOv8 model
- Final Output: Trained weights (`best.pt`)

2.3 System Setup

The deployment was designed for Windows systems. Users can set up the environment by installing CPU-only PyTorch, Ultralytics, and OpenCV. Two options were provided:

1. Fresh installation with a new virtual environment.
2. Using an existing virtual environment (faster but less reliable).

2.4 Detection Script

The script `run_detect.py` was developed to provide interactive detection. The user can select:

1. Image detection
2. Video detection
3. Webcam detection

Outputs are displayed on-screen and saved with filenames ending in `_det`.

3 Results

The model performed reliably across all test scenarios:

- **Image Detection:** Accurate bounding boxes drawn on license plates.
- **Video Detection:** Smooth frame-by-frame detection with annotated video output.
- **Webcam Detection:** Real-time license plate detection with stable CPU performance.

4 Troubleshooting

Several challenges were encountered during development and deployment:

- Missing library errors (e.g., `cv2`) were resolved via package installation.
- In remote environments without display support, a save-only mode was suggested.
- For slow performance on CPU, the input image size was reduced for faster inference.
- Environment conflicts were handled by recommending fresh installations.

5 Conclusion

Plate-Net AI successfully demonstrates the use of YOLOv8 for license plate detection. The system is flexible, interactive, and optimized for CPU-based execution. It provides a practical solution for intelligent transportation systems and can be extended with OCR techniques to read plate numbers directly. Future work may also focus on deploying the system as a cloud service or integrating it with IoT-based smart city solutions.

References

- Ultralytics YOLOv8 Documentation: https://docs.ultralytics.com
- Kaggle Dataset: https://www.kaggle.com
- PyTorch: https://pytorch.org