Vehicle Detection System

Submitted by:

Arshia (102317263)

Ragini (102317266)

Ishita Singh Oberoi (102317272)

BE Second Year Batch – 2Q25

Submitted to: Mr. Sukhpal Singh

Computer Science and Engineering Department

Thapar Institute of Engineering & Technology, Patiala

1. Introduction & Problem Statement

The rapid increase in the number of vehicles on the roads has created big challenges in managing traffic, reducing congestion, and enforcing traffic rules. Traditional methods for monitoring traffic rely a lot on people manually observing, which can be slow, tiring, and prone to mistakes.This project aims to build a real-time vehicle detection system using YOLOv8 (You Only Look Once), an object detection model. The system is designed to quickly and accurately detect vehicles in different situations using image data.

2. Dataset Overview

The dataset used in this project is the **Top View Vehicle Detection Image Dataset**, which consists of high-resolution images capturing vehicles from a bird's-eye view.

Dataset Characteristics:

- **Image Resolution**: High-resolution images to capture finer vehicle details.
- **Diversity**: Covers different weather conditions, traffic densities, and lighting variations.
- **Annotations**: Includes bounding boxes marking vehicle locations for supervised learning.
- **Traffic Scenarios**: Contains urban and highway settings with different congestion levels.

Data Preprocessing Steps:

- **Image Processing**: It enhances model performance by resizing (imgsz=640), filtering low-confidence detections (conf=0.5), converting color spaces (cv2.cvtColor), reducing computation with grayscale (cv2.COLOR_BGR2GRAY), smoothing noise (cv2.GaussianBlur), and visualizing detections (results[0].plot()).
- **Data cleaning**: It ensures error-free data by loading structured data (pd.read_csv), removing extra spaces (str.strip()), and handling missing values (fillna(), dropna()) to prevent model bias.
- **Confusion matrix**: It visualizes model performance by loading (cv2.imread) and converting (cv2.cvtColor) the image to RGB for accurate plotting.

3. Technology Stack

- **Programming Language**: Python (for data handling, model training, and evaluation).
- **Deep Learning Framework**: Ultralytics YOLOv8 for object detection.
- **Computer Vision Tools**: OpenCV for image processing, manipulation, and visualization.
- **Data Processing**: NumPy and Pandas for handling image metadata and dataset organization.
- **Visualization Libraries**: Matplotlib and Seaborn for graphical representation of results.

4. ML Model Implementation & Evaluation

Implementation Steps:

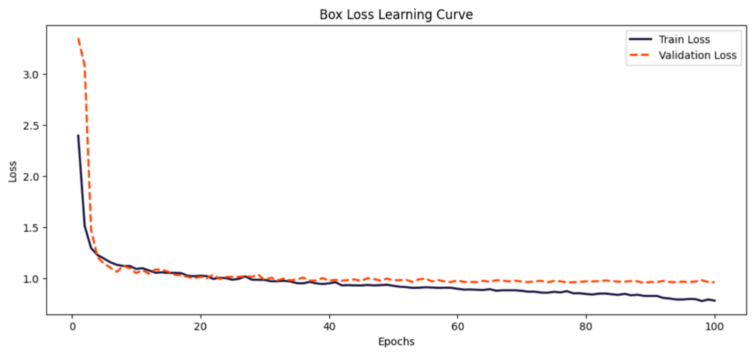
1. **Model Initialization**:
   - Load pre-trained YOLOv8n model.
   - Fine-tune on the dataset.
2. **Dataset Preprocessing**:
   - Resize images, normalize data, and convert bounding box annotations.
3. **Model Training**:
   - Train the YOLOv8 model using GPU acceleration for efficiency.
4. **Inference & Detection**:
   - Predict bounding boxes and confidence scores on test images.
   - Visualize detections with bounding boxes.
5. **Results Analysis**:
   - Evaluate performance using key metrics.

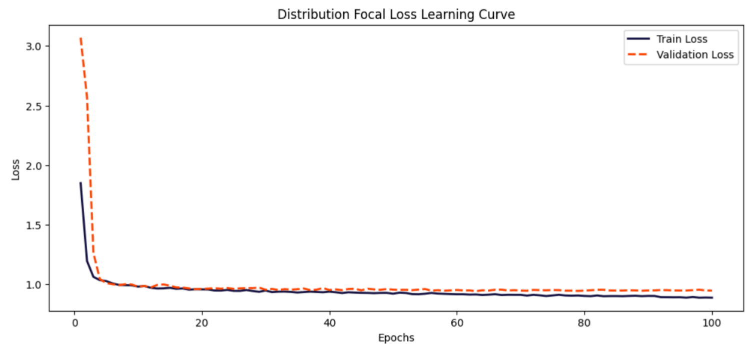
Evaluation Metrics:

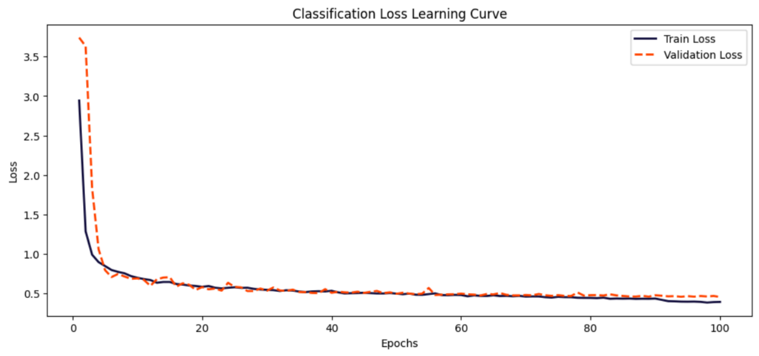
- **Confusion Matrix:** A tool used to visually represent a model's performance by showing the distribution of correct and incorrect predictions across different classes.
- **Learning Curves:** Graphs that track the model's training and validation performance over time to understand learning progress and detect overfitting or underfitting.

5. Results & Insights (Visualizations & Metrics)

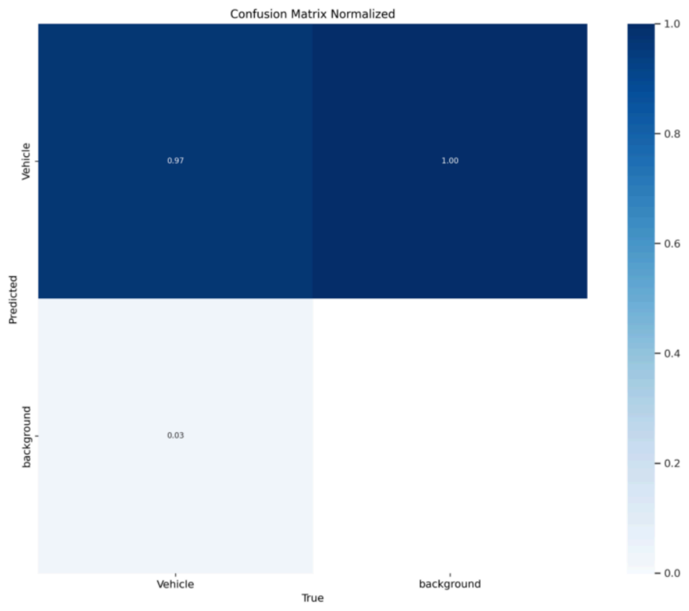
- **Learning Curves:** The graphs show that the YOLOv8 model is learning well, with training and validation losses steadily decreasing, meaning it's improving at detecting objects without major overfitting issues.







- **Confusion Matrix**: The confusion matrix shows that the YOLOv8 model detects vehicles with 97% accuracy, misclassifying 3% of background as vehicles, while perfectly identifying the background (100% accuracy).



6. Challenges & Future Improvements

Challenges:

- **Low-Light Detection Issues**: The model struggles to detect vehicles in nighttime images or poor lighting conditions.
- **Overlapping Vehicles**: Vehicles that are partially hidden or closely packed together may be misclassified or undetected.
- **Dataset Bias**: The dataset primarily consists of **top-view images**, which may limit the model's ability to generalize to other viewpoints.

Future Improvements:

- **Training on a More Diverse Dataset**: Expanding the dataset to include different camera angles and more varied environments.
- **Integrating Multi-Class Classification**: Enhancing detection capabilities to classify vehicles based on type (e.g., sedan, SUV, truck, bus, motorcycle).
- **Implementing Real-Time Tracking**: Adding object tracking features to monitor vehicle movement across video frames.

7. Conclusion & Learnings

This project demonstrates the effectiveness of **YOLOv8** in detecting vehicles in real-time with high accuracy. The model performs well under normal conditions but requires enhancements to handle **low-light environments, occlusions, and diverse viewpoints**.

Key learnings from this project:

- **Deep learning-based object detection significantly improves traffic monitoring** and vehicle classification accuracy.
- **Dataset quality and diversity play a crucial role** in model generalization and robustness.
- **Optimizing models for real-time performance** is essential for practical deployment in smart cities and intelligent transport systems.

8. References

- YOLOv8 Documentation: https://docs.ultralytics.com
- OpenCV Library: https://opencv.org
- Python Official Documentation: https://docs.python.org/