Traffic Sign Detection Using YOLO for Autonomous System : Engineering Track

Aarya Gupta aarya22006@iiitd.ac.in
Sargun Singh Khurana sargun22450@iiitd.ac.in
Abdullah Shujat abdullah22013@iiitd.ac.in

Abstract

This project implements traffic sign detection using YOLOv8, a Deep Learning model for real-time object detection. The model is trained on a traffic sign dataset with preprocessing and augmentation techniques to enhance accuracy. Results show YOLOv8's effectiveness in detecting signs with high precision, highlighting its potential for autonomous driving and road safety applications. '

1. Problem Statement

The implementation of Traffic Sign Detection using YOLOv8 is crucial for enhancing road safety and autonomous navigation. This project aims to develop a real-time detection system capable of accurately identifying various traffic signs from images and videos, leveraging deep learning for precise recognition.

1.1. Scope of the Problem:

- **Input:** Traffic sign images and video streams captured from dashcams, surveillance cameras, or autonomous vehicle sensors. The dataset includes 4,969 samples, categorized into Train, Validation, and Test sets.
- **Output:** Annotated images and video frames with bounding boxes, labels for detected traffic signs, and confidence scores.

1.2. Target Users:

- **Autonomous Vehicles ADAS Systems:** For real-time traffic sign recognition, enabling compliance with speed limits and stop signals.
- **Traffic Monitoring Authorities:** To monitor traffic rule violations using connected CCTV and IoT-based infrastructure.
- **Urban Planners Smart City Developers:** To analyze traffic compliance patterns and improve urban planning.

1.3. System Interface:

The traffic sign detection system will be accessible via a web application, allowing users to upload images and videos for processing. The application will use a backend model powered by YOLOv8 to analyze the uploaded media, detect traffic signs, and return annotated outputs with labeled bounding boxes and confidence scores. This web-based approach ensures ease of use, scalability, and accessibility across different devices.

By leveraging YOLOv8's real-time processing capabilities, this project aims to provide an efficient and scalable solution for traffic sign detection, contributing to autonomous vehicle safety, smart city infrastructure, and transportation research.

2. User Interface & Progress

The traffic sign detection system is a **web application** where users upload images or videos for YOLOv8-based detection. The processed output, with bounding boxes and labels, is displayed on the interface.

2.1. Design & Workflow

1. **Upload**: Users select images or videos.
2. **Processing**: YOLOv8 detects traffic signs.
3. **Output Display**: Annotated results are shown.
4. **(Planned)**: Download option for processed files.

2.2. Progress & Development Tools

- **Tech Stack**: Flask/Django, HTML/CSS/JS, Ultralytics YOLOv8 (model).

2.3. Progress Overview

Completed Components:

- Implemented the YOLOv8 model for traffic sign detection.
- Integrated OpenCV for image and video processing.
- Model training and testing done on images and videos, output images with classification and video generated.

Pending: Basic UI with upload & detection, UI optimization, download feature

The system follows a pipeline where users upload images/videos, which are processed using OpenCV, analyzed by the YOLOv8 model, and displayed via a web-based UI.

By focusing on real-time detection and an intuitive interface, our goal is to make the system accessible for both research and practical applications in traffic management.

3. Related Work

Traffic sign detection and recognition have been extensively studied, with various methodologies developed to improve accuracy and efficiency. Below is a summary of notable approaches, along with key research papers:

3.1. Support Vector Machine (SVM) with Histogram of Oriented Gradient (HOG)

Approach: This method utilizes the HOG feature descriptor to extract features from traffic sign images, which are then classified using an SVM classifier. It is particularly effective in distinguishing traffic signs from other objects based on shape and edge information.

Reference Paper: "Traffic Sign Detection and Recognition Model Using Support Vector Machine and Histogram of Oriented Gradient" [6] by Nabil Ahmed et al. This study presents a model that detects and recognizes Bangladeshi traffic signs from video frames using SVM and HOG, achieving high precision and accuracy.

3.2. Convolutional Neural Networks (CNNs)

Approach: CNNs automatically learn hierarchical feature representations from raw pixel data, making them highly effective for image recognition tasks, including traffic sign detection.

Reference Paper: "CNN based Traffic Sign Detection and Recognition on Real Time Video" [7] by Deepali Patil et al. This paper proposes a system that captures signboards using a camera installed in a vehicle, processes the images using a CNN, and notifies the user of detected traffic signs in real-time.

3.3. Hybrid Approaches Combining GIS and Machine Learning

Approach: Integrating Geographic Information Systems (GIS) with machine learning techniques allows for the detection and positioning of traffic signs using geo-tagged images and videos.

Reference Paper: "Computer Vision-Based Traffic Sign Detection and Extraction: A Hybrid Approach Using GIS And Machine Learning" [8] by Zihao Wu. This research proposes methods that utilize geo-tagged Google Street View images and GoPro videos to detect and map traffic signs accurately.

3.4. Baseline Models for Comparison

For our project, we will consider the following models as baselines to evaluate the performance of our YOLOv8-based traffic sign detection system:

- **SVM with HOG:** Provides a traditional machine learning approach for traffic sign detection, offering a benchmark for evaluating the effectiveness of deep learning models.
- **CNN-based Detection Systems:** Serve as a standard for assessing the improvements in accuracy and efficiency achieved by newer architectures like YOLOv8.

By comparing our system's performance against these established models, we aim to demonstrate the advancements and effectiveness of our approach in traffic sign detection.

4. Datasets

4.1. Datasets

The system is currently trained on the **Traffic Sign Detection Dataset** [5], which contains images of vehicles and their surroundings. This dataset enables the initial development and validation of the YOLOv8 model, allowing for preliminary testing of sign detection capabilities before integrating more specialized datasets to improve accuracy.

To enhance model performance and better generalization across various environments, additional datasets will be integrated in future which include:

- **GTSRB - German Traffic Sign Recognition Benchmark** [1]
- **Road Sign Detection** [2]
- **Indian Traffic Sign Dataset** [3]
- **Traffic Signs Dataset - Indian Roads** [4]

4.2. Exploratory Data Analysis (EDA)

Key Statistics:

- **Class Distribution:** The datasets exhibit class imbalance, where common signs such as speed limits appear more frequently than rare signs.
- **Image Quality:** Some datasets contain low-resolution images or images with motion blur, requiring preprocessing techniques such as noise reduction and image enhancement.
- **Lighting Variations:** The datasets include images taken under different lighting conditions, necessitating data augmentation techniques.
- **Domain Shift:** The datasets cover different countries, meaning models trained on one dataset may need fine-tuning for another region.

Some peculiarities in the dataset include:

- **Class Imbalance:** Some classes, such as stop signs and speed limits, are well-represented, whereas traffic light signs (Red Light, Green Light) have relatively fewer instances.

- **Domain Shifts:** The dataset contains images captured in different lighting conditions (day, night, fog), which may affect detection performance.
- **Varied Image Quality:** Some images have low resolution or are partially occluded, making recognition more challenging.

4.3. Evaluation Metrics

- **Precision:** 94.2%, with Stop Signs (97.3%) and Speed Limit Signs (92-100%) showing strong results. Red (88.1%) and Green Light (82.6%) had slightly lower precision.
- **Recall:** 90.6%, with Stop Signs (98.8%) and Speed Limits (94.1-99.1%) performing well. Lower recall for Red (69.4%) and Green Light (74.1%) suggests occasional missed detections.
- **mAP@0.5:** 95.7%, indicating strong detection capabilities.
- **mAP@0.5-0.95:** 82.97%, reflecting challenges in precise localization.
- **Inference Speed:** 2.4ms per image, supporting real-time applications.
- **Loss Metrics:**
  - Train Box Loss: 0.49017, Validation Box Loss: 0.53427
  - Train Class Loss: 0.38064, Validation Class Loss: 0.35107
  - Train DFL Loss: 0.90189, Validation DFL Loss: 0.93138
- **Speed Metrics:**
  - Inference Speed: 2.4ms per image
  - Preprocessing Speed: 1.5ms per image
  - Postprocessing Speed: 0.9ms per image

5. Compute Resources

- **GPU**: Kaggle T4x2 (2 Tesla T4 GPUs with 16 GB VRAM each, 32 GB total)
- **CPU**: Standard multi-core CPU (exact model unspecified)
- **RAM**: 16 GB or more
- **Operating System**: Linux (Ubuntu-based)
- **CUDA Version**: CUDA 11.x
- **Python Version**: Python 3.10.x
- **Framework**: PyTorch 2.0 (with GPU support for training YOLOv8 model)
- **Training Configuration**: Batch size- 8, 16, 32, 64 , epochs- 10, 50, 100

6. Analysis of Results

The model demonstrated strong performance in most categories, particularly for Stop Signs and Speed Limit Signs. However, performance for Red Light and Green Light traffic signals was comparatively lower.

6.1. Class-wise Performance

- **Stop Sign:** High precision (97.3%) and recall (98.8%) indicate that the model is excellent at detecting stop signs. The mAP@0.5 (99.4%) and mAP@0.5-0.95 (93.3%) further confirm its accuracy.
- **Speed Limit Signs:** Performance varies based on speed limit categories, but mAP@0.5 scores range from 92% to 100%, making them highly reliable.
- **Red and Green Light:** The recall for Red Light (69.4%) and Green Light (74.1%) is lower compared to other categories. This suggests the model occasionally misses these signs, likely due to variations in lighting and occlusions.
- **Speed Limits (20-120):** Most speed limit signs performed well, but the highest category (Speed Limit 120) had slightly lower precision and recall.

7. Next Steps

7.1. Strengths and Areas for Improvement

Strengths:

- High precision (94.2%) and recall (90.6%) indicate strong detection capabilities.
- Excellent performance on Stop Signs and Speed Limits.
- Fast inference speed (2.4ms per image) supports real-time applications.

Areas for Improvement:

- **Red and Green Light Detection:** Lower recall suggests difficulties in identifying these signs, requiring more diverse training data.
- **Localization Accuracy:** The mAP@0.5-0.95 (82.97%) indicates scope for refining bounding box precision.

7.2. Future Improvements

- **Model Enhancement:** Fine-tune YOLOv8 and optimize hyperparameters for improved accuracy.
- **Data Augmentation:** Expand training data, especially for Red and Green Lights, and apply advanced augmentation techniques.
- **User Interface:** Improve UI responsiveness and integrate real-time detection.
- **Deployment:** Develop an API for model inference and connect it to the UI.
- **Performance Testing:** Evaluate speed, accuracy, and resource usage under real-world conditions.
- **Final Documentation:** Compile results, methodologies, and findings into a comprehensive report.

Conclusion: The model is well-optimized for most traffic signs but requires improvements in traffic light detection and localization precision.

8. Member Contribution

- **Aarya Gupta**: Model Implementation, Training, EDA, Performance Evaluation, Report Writing.
- **Abdullah Shujat**: Dataset Collection, Data Preprocessing and Augmentation, Report Writing, Literature Review, Presentation Preparation.
- **Sargun Singh Khurana**: Research and Testing, Literature Review, System Debugging, Dataset Collection, Report Writing, Presentation Preparation.

References

[1] GTSRB - German Traffic Sign Dataset, https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign 2

[2] Road Sign Detection Dataset, https://www.kaggle.com/datasets/andrewmvd/road-sign-detection 2

[3] Indian Traffic Sign Dataset, https://www.kaggle.com/datasets/neelpratiksha/indian-traffic-sign-dataset 2

[4] Traffic Signs Dataset - Indian Roads, https://www.kaggle.com/datasets/kaustubhrastogi17/traffic-signs-dataset-indian-roads 2

[5] Traffic Sign Detection Dataset, https://universe.roboflow.com/selfdriving-car-qtywx/self-driving-cars-lfjou 2

[6] Traffic Sign Detection and Recogni-098 tion Model Using Support Vector Machine and Histogram099 of Oriented Gradient https://www.mecs-press.org/ijitcs/ijitcs-v13-n3/v13n3-5.html 2

[7] CNN based Traffic Sign Detection https://www.ijert.org/cnn-based-traffic-sign-detection-and-recognition-on-real-time-video 2

[8] Computer Vision-Based Traffic Sign Detection https://digitalcommons.georgiasouthern.edu/etd/2039/ 2

9. Visualization of Results

These visualizations help assess the model's strengths and weaknesses, guiding further improvements in detection performance.

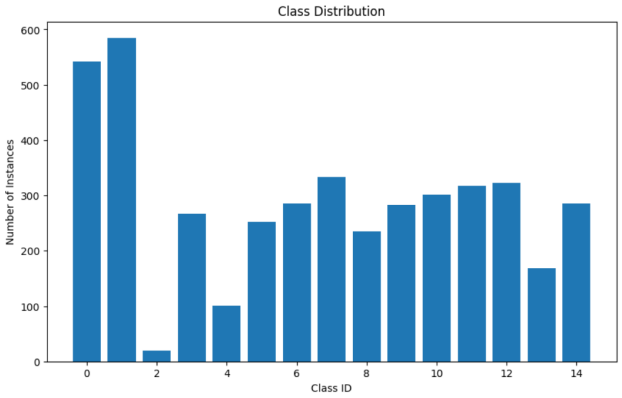

Figure 1. EDA - Bar chart showing class distribution, with Class ID on the x-axis and Number of Instances on the y-axis.

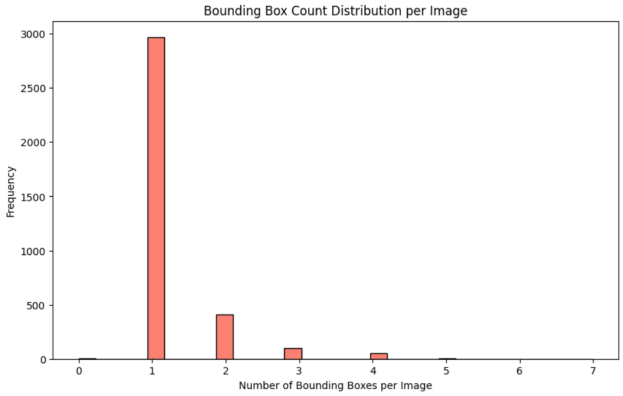

Figure 2. EDA - The histogram shows the distribution of bounding box counts per image. Most images contain a single bounding box, with fewer images having multiple bounding boxes, indicating class imbalance.

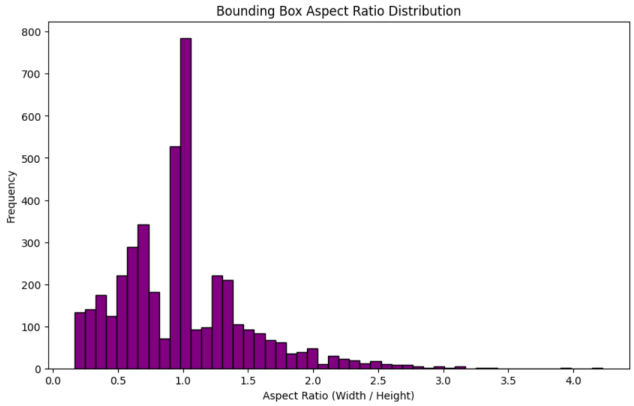

Figure 3. EDA - This histogram shows the distribution of bounding box aspect ratios (Width/Height). The peak at 1.0 suggests many square boxes, while other peaks indicate common shapes.

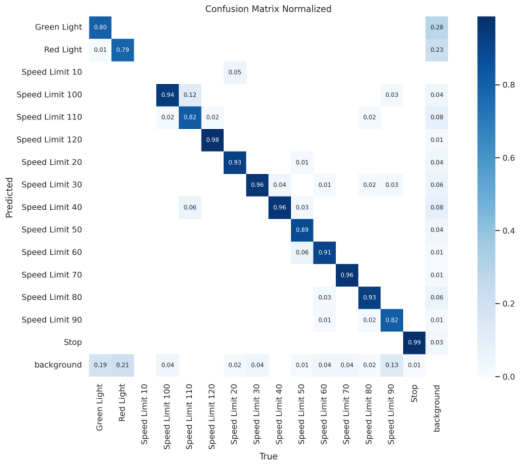

Figure 4. EDA - Normalized Confusion Matrix showing the classification performance across different traffic sign categories.

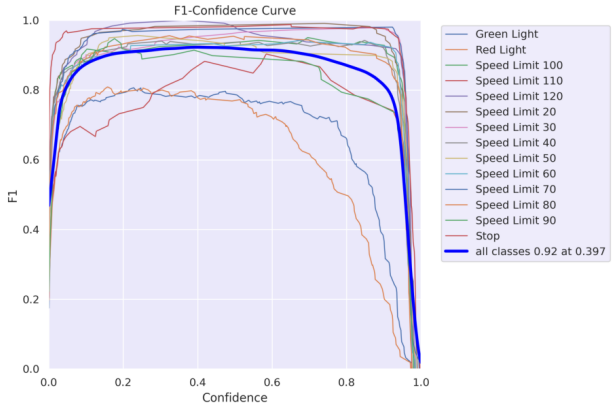

Figure 5. F1-score curve demonstrating the trade-off between precision and recall at different confidence thresholds.

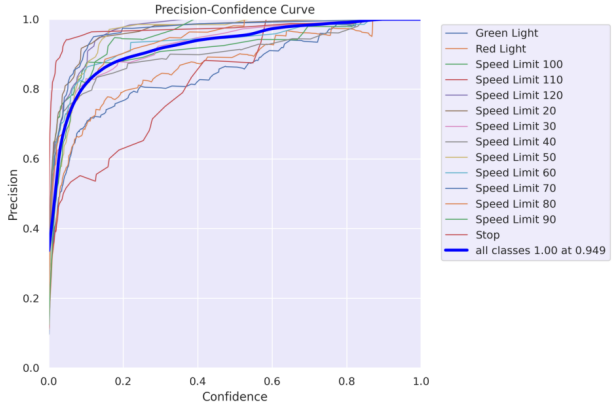

Figure 6. Precision curve illustrating how precision varies across different confidence levels.

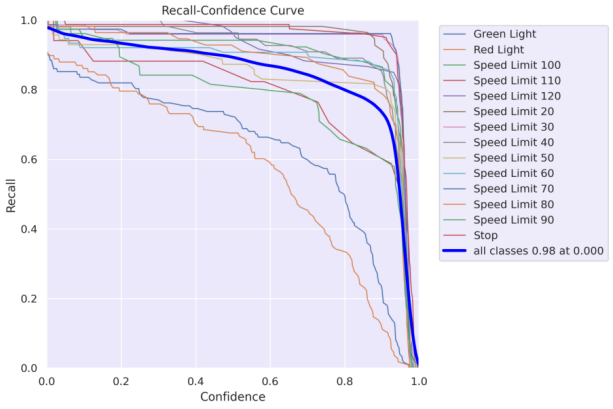

Figure 7. Recall curve showing the variation in recall at different confidence thresholds.

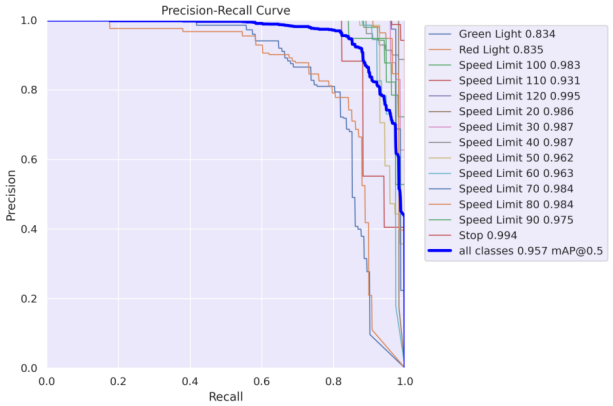

Figure 8. Precision-Recall curve providing insights into the model's ability to balance precision and recall.

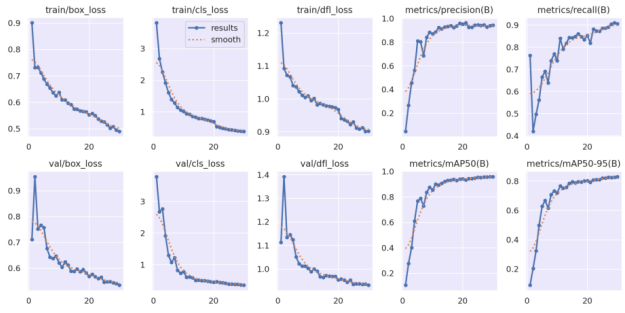

Figure 9. Summary of training results, including loss curves and mAP performance over epochs.