ISSN: 2347-6540

International Journal of Latest Research In Engineering and Computing (IJLREC)
Volume 11, Issue 1, Page No.23-32, 2024
www.ijlrec.com

Traffic Sign Detection Using YOLOv8 Algorithm

Mohd. Aman Shah[1], Arun Negi[2], Arun Negi[3], Harendra Kumar[4], Ms. Aakansha Pundir[5]
Student, Btech. CSE, Shivalik College of Engineering, Dehradun, Uttarakhand, India, reach.mohdamanshah@gmail.com[1]
Student, Btech. CSE, Shivalik College of Engineering, Dehradun, Uttarakhand, India, reacharunnegi@gmail.com[2]
Student, Btech. CSE, Shivalik College of Engineering, Dehradun, Uttarakhand, India, arunnegi989@gmail.com[3]
Student, Btech. CSE, Shivalik College of Engineering, Dehradun, Uttarakhand, India, kumarharender452@gmail.com[4]
Assistant Professor, CSE Department, Shivalik College of Engineering, Dehradun, aakansha.pundir@sce.org.in[5]

Abstract. YOLO is a smart technology used for quickly spotting objects in images or videos. It's popular because it's fast and accurate. Over the years, it's been used to detect all sorts of things like traffic signs, people, cars and many other real world objects.Our goal is to study how YOLO works specifically for spotting **Traffic Signs**. We want to look at five main things: what it's used for, the data it uses, how well it performs, what kind of computers it needs, and the problems it faces.We have opted for the advanced YOLO algorithm i.e YOLOv8 in our project. Compared to other object detection algorithms like Faster R-CNN, SSD, YOLOv8 demonstrates superior accuracy and efficiency in detecting objects.The YOLOv8 model is trained on the dataset obtained from the Roboflow website, specifically formatted as YOLOv5 PyTorch, then the trained model detects traffic signs which are classified into 21 different classes.Also the model is also capable of detecting classified traffic signs in real-time by making predictions over the live camera feed.

Keywords: YOLO – You Only Look Once, CNN – Convolutional Neural Network, R-CNN – Region-based Convolutional Neural Network, SSD – Single Shot Detector

I. INTRODUCTION

Detecting traffic signs is crucial for ensuring road safety, especially in the realm of intelligent transportation systems. As technology continues to advance, so do the methods used for traffic sign detection, enabling the development of more sophisticated driver assistance systems. In practical terms, traffic sign recognition involves identifying and categorizing signs in real-time images or videos captured by cameras installed on vehicles or road-side units. However, this task is challenging due to the diverse range of sign shapes, colors, and conditions, including variations in lighting and weather. AI-driven traffic sign detection systems hold great promise for enhancing road safety and improving the driving experience. By alerting drivers to relevant signs, these systems can help mitigate the risk of accidents..This research paper aims to delve into the realm of AI-based traffic sign detection, offering insights into its effectiveness in practical scenarios. It will provide an overview of state-of-the-art methods, analyzing their strengths and weaknesses through extensive experiments conducted in real-world settings. As the focus shifts towards autonomous and connected vehicles, the demand for reliable traffic sign detection systems continues to rise, while detecting signs in complex environments, handling occlusions, and ensuring robustness under various environmental conditions is still the major issue.

Through this comprehensive exploration, the paper aims to address these challenges, providing valuable insights for scholars and researchers working towards the advancement of intelligent transportation systems. Ultimately, the findings of this study will contribute to the ongoing evolution of this critical field.

II. PROJECT WORKFLOW

The dataset of total 2093 images and categorized into 21 different classes was first pre-processed and divided into test, train and validation sets.After this those train and validation sets are used to train the YOLOv8 Model, following command is used to do so:

```
!yolo task=detect mode=train model=yolov8s.pt data={dataset_location}/data.yaml
epochs=100 imgsz=640
```

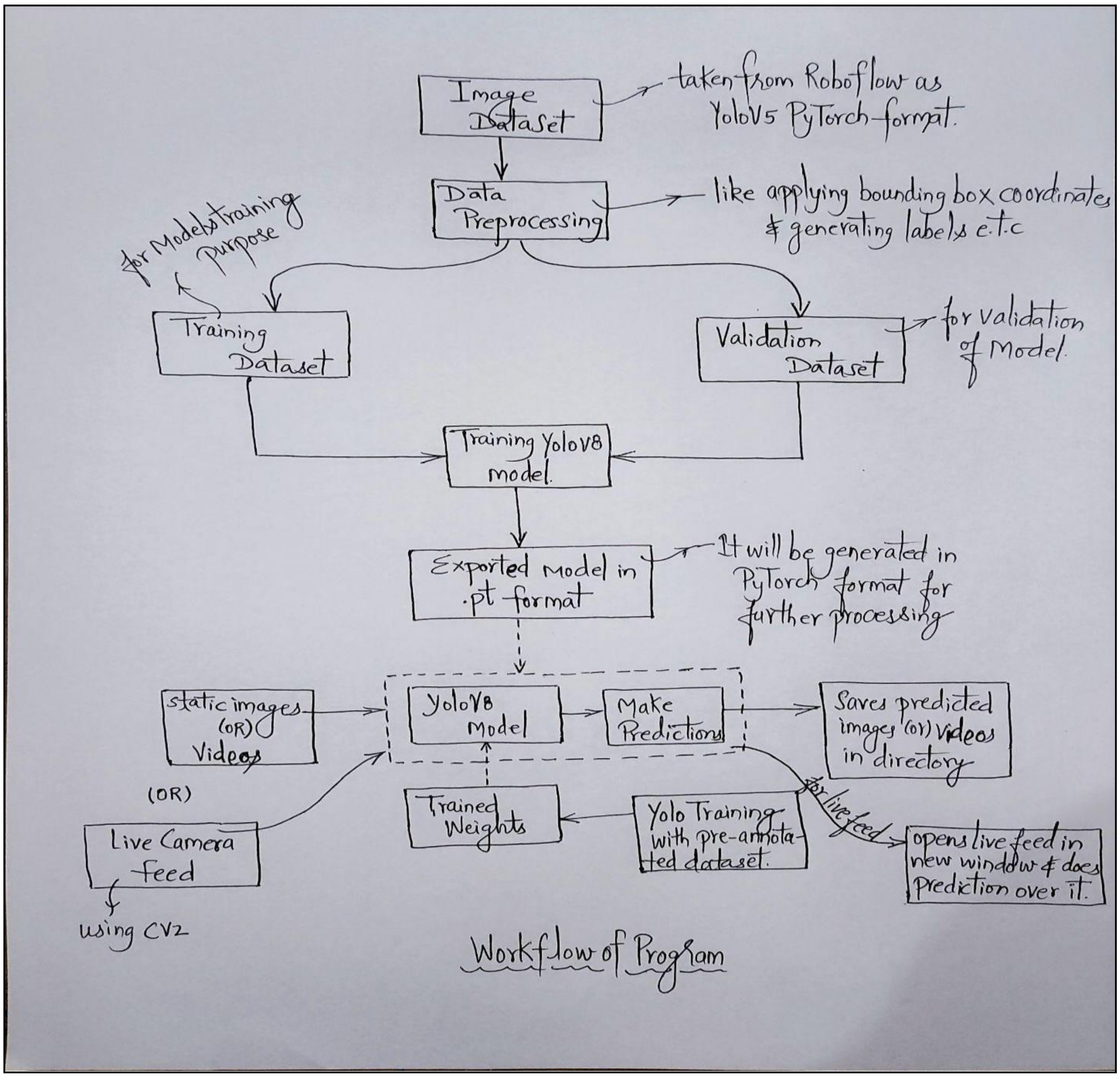


Image Dataset
taken from Roboflow as YoloV5 PyTorch format.
Data Preprocessing
like applying bounding box coordinates & generating labels e.t.c
for Models training purpose
Training Dataset
Validation Dataset
for validation of Model.
Training YoloV8 model.
Exported Model in .pt format
It will be generated in PyTorch format for further processing
static images (OR) Videos
(OR)
Live Camera feed
using CV2
YoloV8 Model
Make Predictions
Saves predicted images (or) videos in directory
for live feed
opens live feed in new window & does Prediction over it.
Trained Weights
Yolo Training with pre-annotated dataset.
Workflow of Program

This will generate and export weights, one of which is having a **.pt** (PyTorch) extension which will be used further for making predictions.Using that weight we can do predictions on static images and videos or on Real-time camera feed. In case of static images and pre-recorded videos given as input , the program will store them in the predict directory after making the prediction, while for Real-time camera feed it will open a new window showing that feed and do prediction over it.

III. METHODOLOGY

1. Dataset Acquisition and Preprocessing:

Dataset contains 2093 images categorized into 21 different classes. These images are preprocessed by dividing them into training and validation sets and resizing, normalization and generating labels are done over to ensure compatibility with the YOLOv8 model.

2. Model Selection and Configuration:

The YOLOv8 model was chosen for traffic sign detection and recognition due to its efficiency, and speed. Its single-stage detection approach allows for real-time processing of images, crucial for timely responses in road safety applications. Leveraging the strengths of YOLOv8, our aim is to develop a robust system contributing to improved road safety and driving experience.

3. Training Procedure:

YOLO is a groundbreaking object detection algorithm known for its real-time processing speed and accuracy. The model was trained using a carefully prepared dataset, where images were annotated beforehand.The training process spanned 100 epochs, each epoch representing a complete pass through the dataset. Images were resized to 640x640 pixels, and training was conducted in batches of 16 images at a time.After training, the model selected the best-performing weights, ensuring optimal performance for subsequent tasks.Ultralytics' YOLOv8 Python library facilitated the training process, providing essential tools and functionalities for model optimization and training.

After training is completed we get a confusion matrix.This matrix provides a visual representation of the model's classification results, detailing the true positive, false positive, true negative, and false negative predictions across all classes.

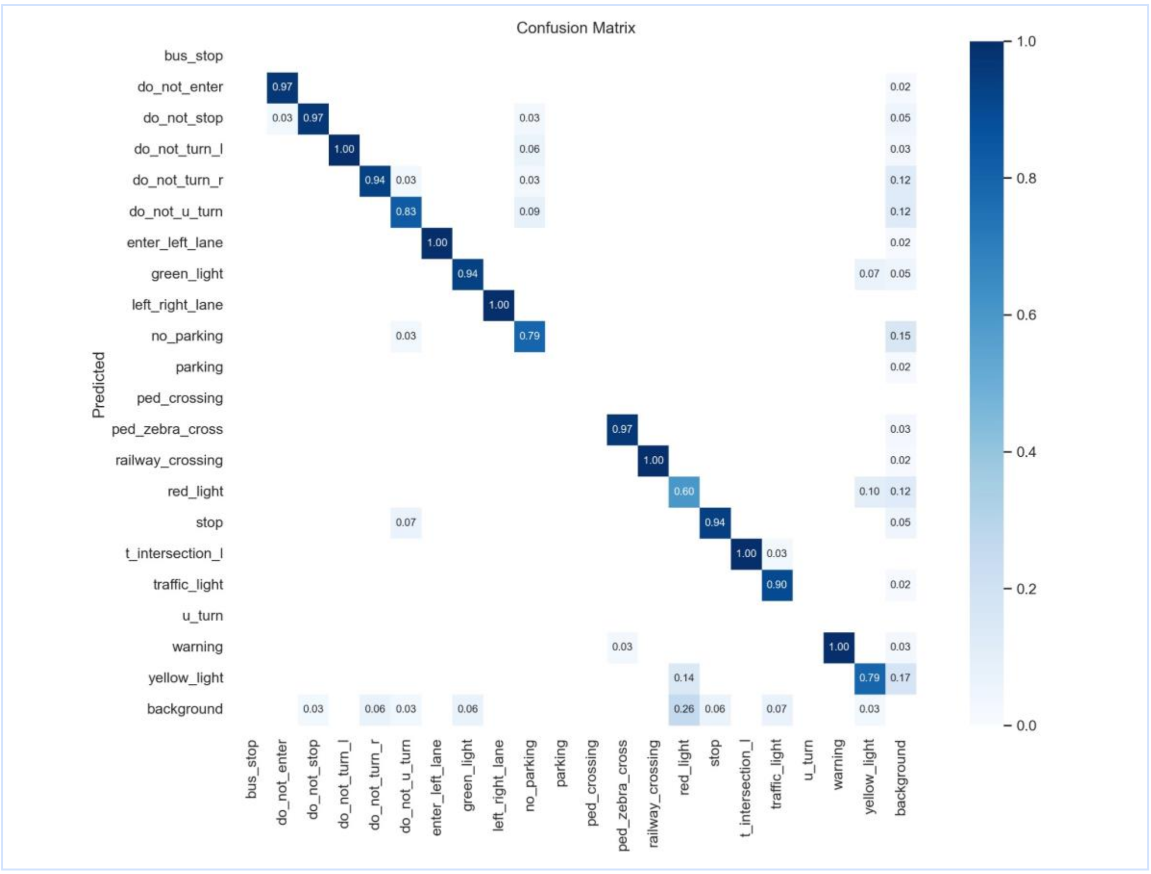


Fig. 2. Confusion Matrix

4. Evaluation Metrics:

To gauge the effectiveness and robustness of our trained model, we employed several standard evaluation metrics commonly used in object detection tasks.

These metrics include precision, recall, and mean average precision (mAP). Precision measures the accuracy of positive predictions, recall evaluates the model's ability to identify all relevant instances, and mAP provides a comprehensive assessment of detection performance across multiple classes.

The precision, recall, F1 score and mAP can be evaluated using the following equations:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

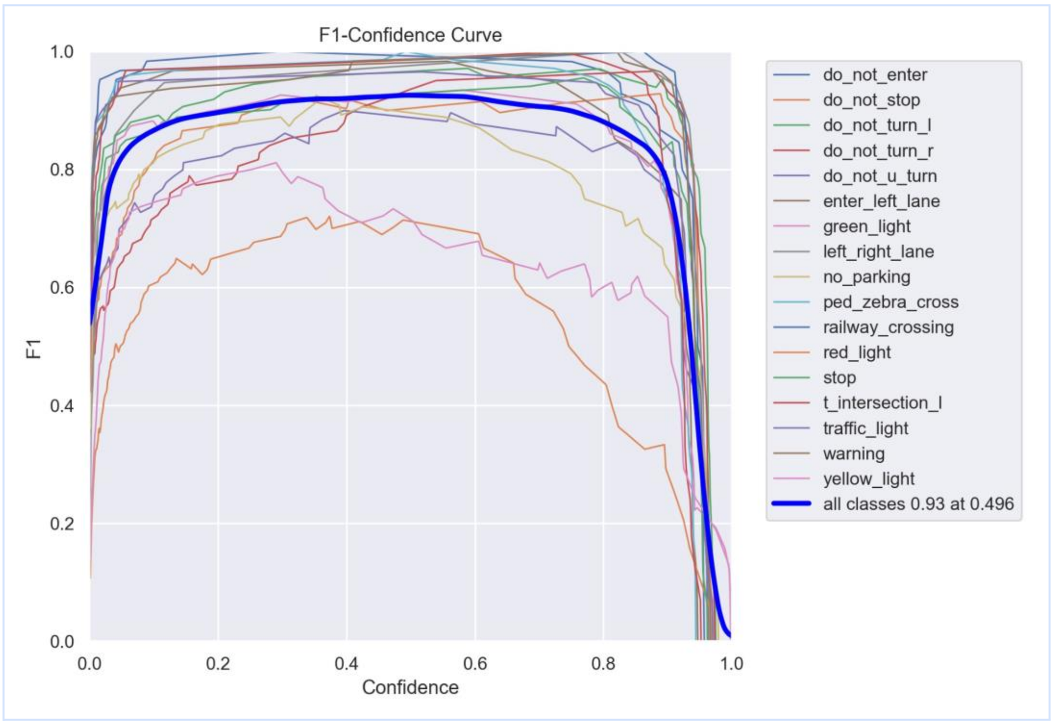


Fig. 3. F1 – Confidence Graph

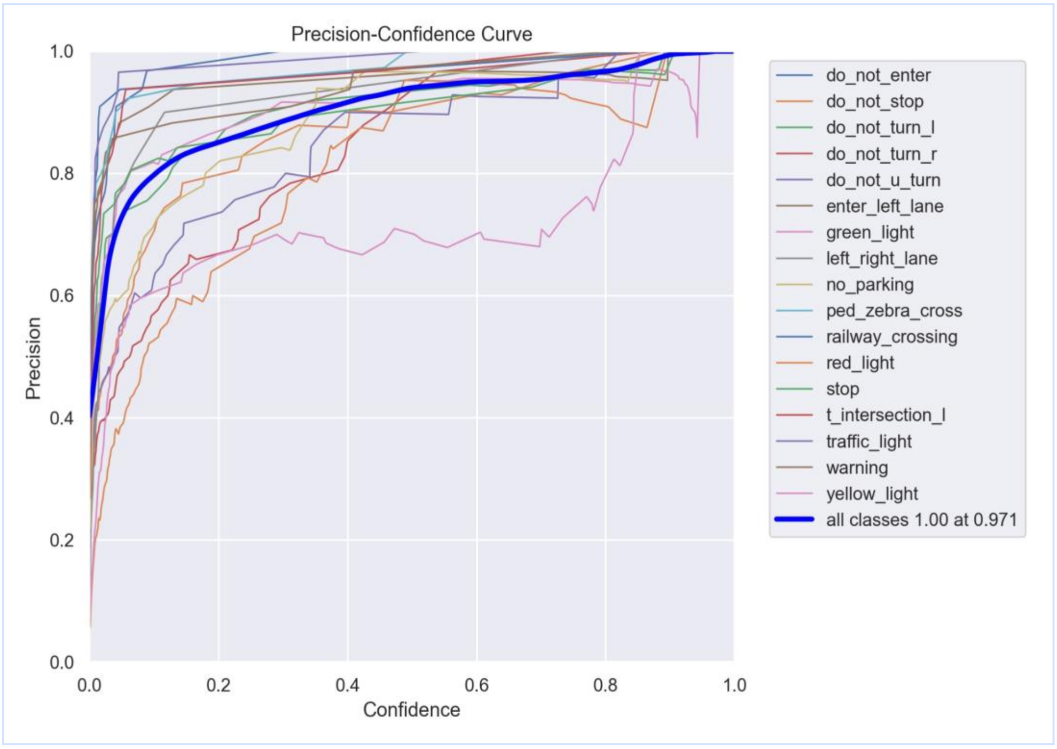


Fig. 4. Precision – Confidence Graph

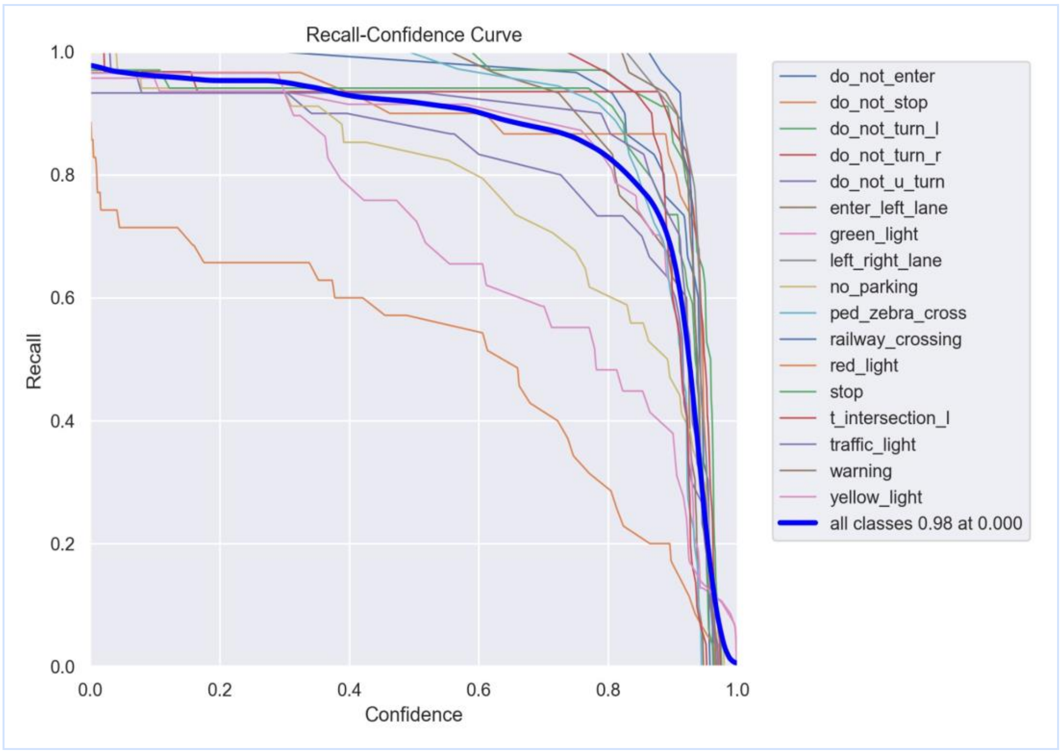


Fig. 5. Recall – Confidence Graph

The model exhibits impressive precision, nearly perfect at a confidence threshold of 0.971. Moreover, the recall rate is exceptionally high, reaching approximately 0.98 at a confidence threshold of 0.000. The Precision-Recall Curve suggests a precision level of around 0.950, with a Mean Average Precision (mAP) of 0.5.

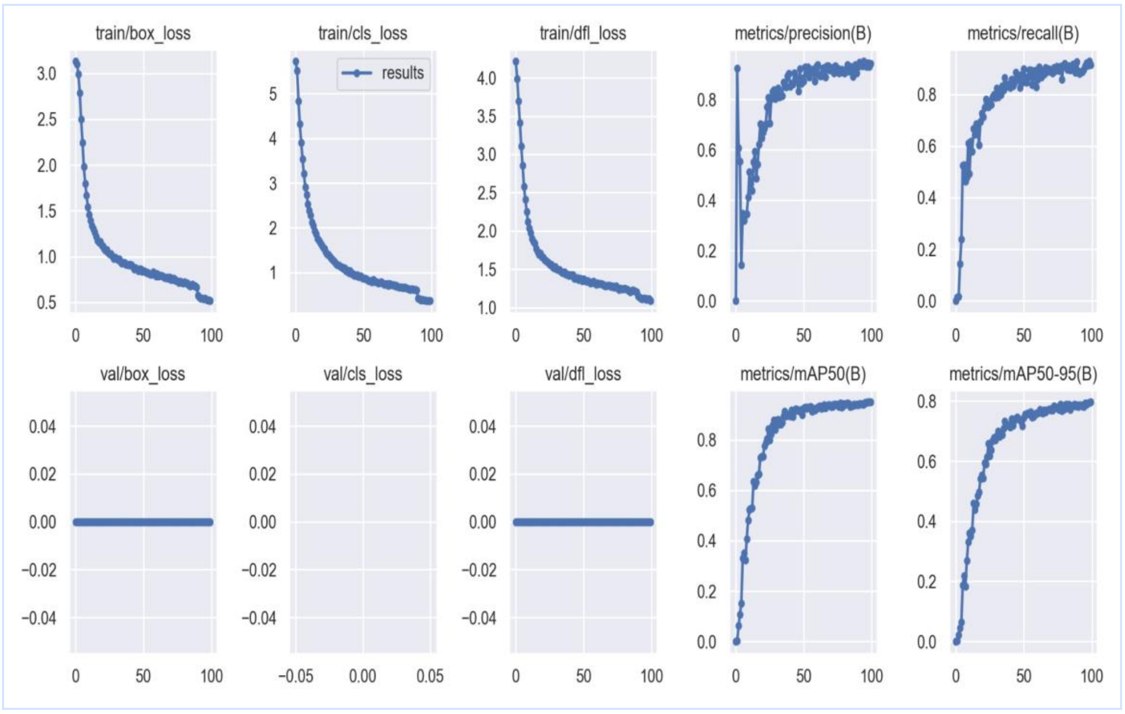


Fig. 6. Training and Validation Losses

5. Experimental Setup: Hardware:

GPU: NVIDIA GeForce RTX 3050 Laptop GPU CPU: 16 CPUs
Memory: 15.3 GB RAM Disk Space: 112.0/293.6 GB

Software:

Python Version: 3.12.3
PyTorch Version: 2.2.2+cu118
Ultralytics YOLOv8 Version: 2.5
CUDA Version: 11.8

6. Future Scope:

Looking ahead, there are exciting possibilities to enhance our project's functionality and accessibility. One avenue we can explore is integrating voice feedback using Python libraries like pyttsx3. This addition would be particularly beneficial for individuals with visual impairments or low eyesight, as it would provide auditory cues about detected traffic signs, thereby improving their overall road safety experience.

Furthermore, we can extend the reach of our project by converting it into an Android application. To achieve this, we can utilize the trained .pt weights from our model and convert them into the .tflite format, which is compatible with TensorFlow Lite. By doing so, our project will become more accessible and portable, allowing users to access its capabilities on their mobile devices. This expansion into the realm of mobile applications opens up new possibilities for reaching a wider audience and making a meaningful impact on road safety.

7. **Limitations and Assumptions:**

a. **Dataset constraints:** Reliability and representativeness of the dataset, potential biases, and variations in traffic sign characteristics may impact model performance.

b. **Model architecture:** YOLOv8's efficiency and accuracy may vary based on traffic sign complexity, image resolution, and overlapping signs.

c. **Training procedure:** Hyperparameters, data augmentation, and optimization techniques influence model performance and generalization.

d. **Real-world application:** Evaluation metrics may not fully reflect model performance under diverse environmental conditions and dynamic scenarios.

e. **Assumptions:** The methodology assumes YOLOv8's suitability for traffic sign detection and relies on the chosen dataset's adequacy for training.

IV. DATASET DESCRIPTION

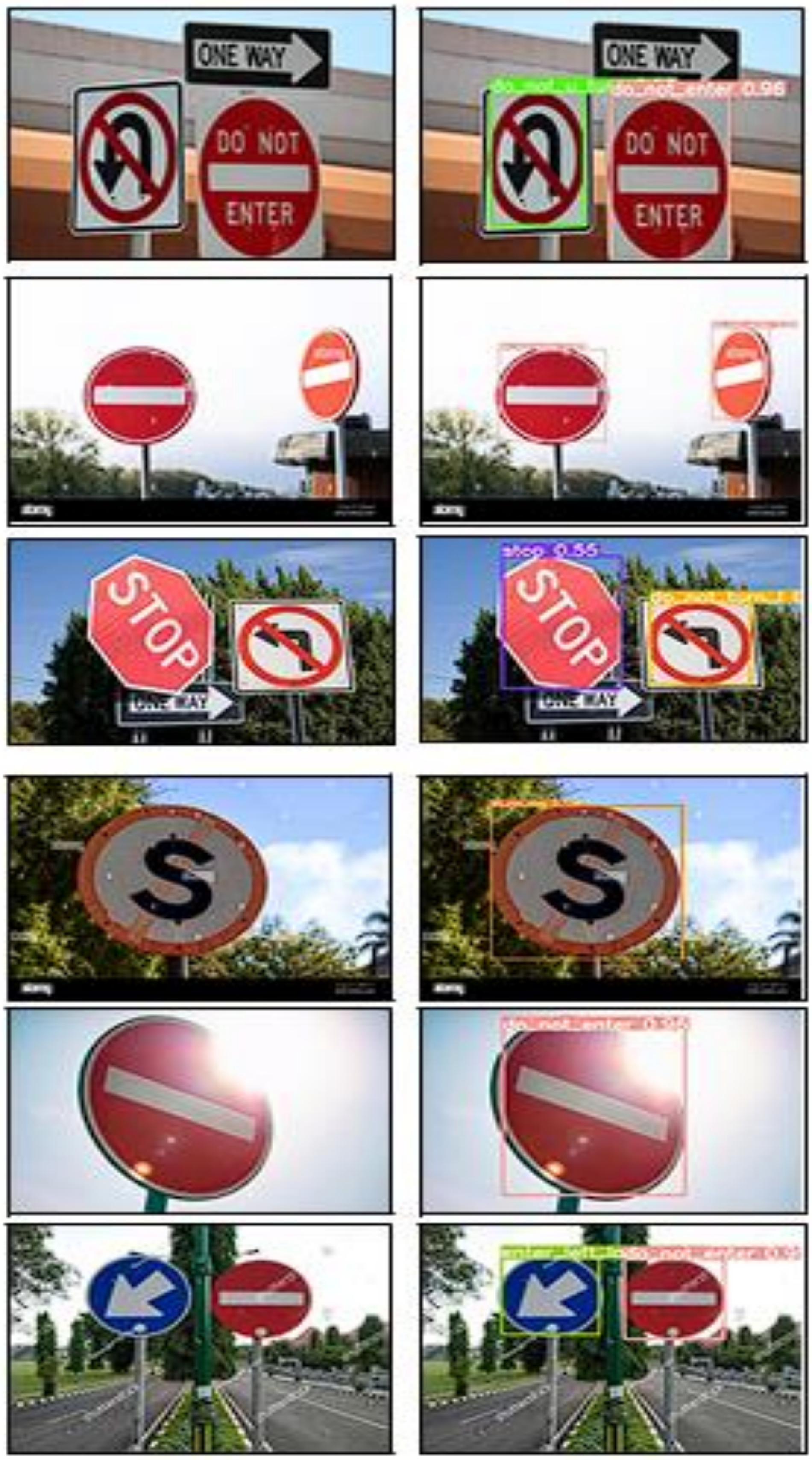
We're using a collection of 2093 images for training our model. These images are divided into three groups: training, validation, and testing.Further these images are categorized into 21 classes namely stop signs, traffic light signs, pedestrian crossing and many more.

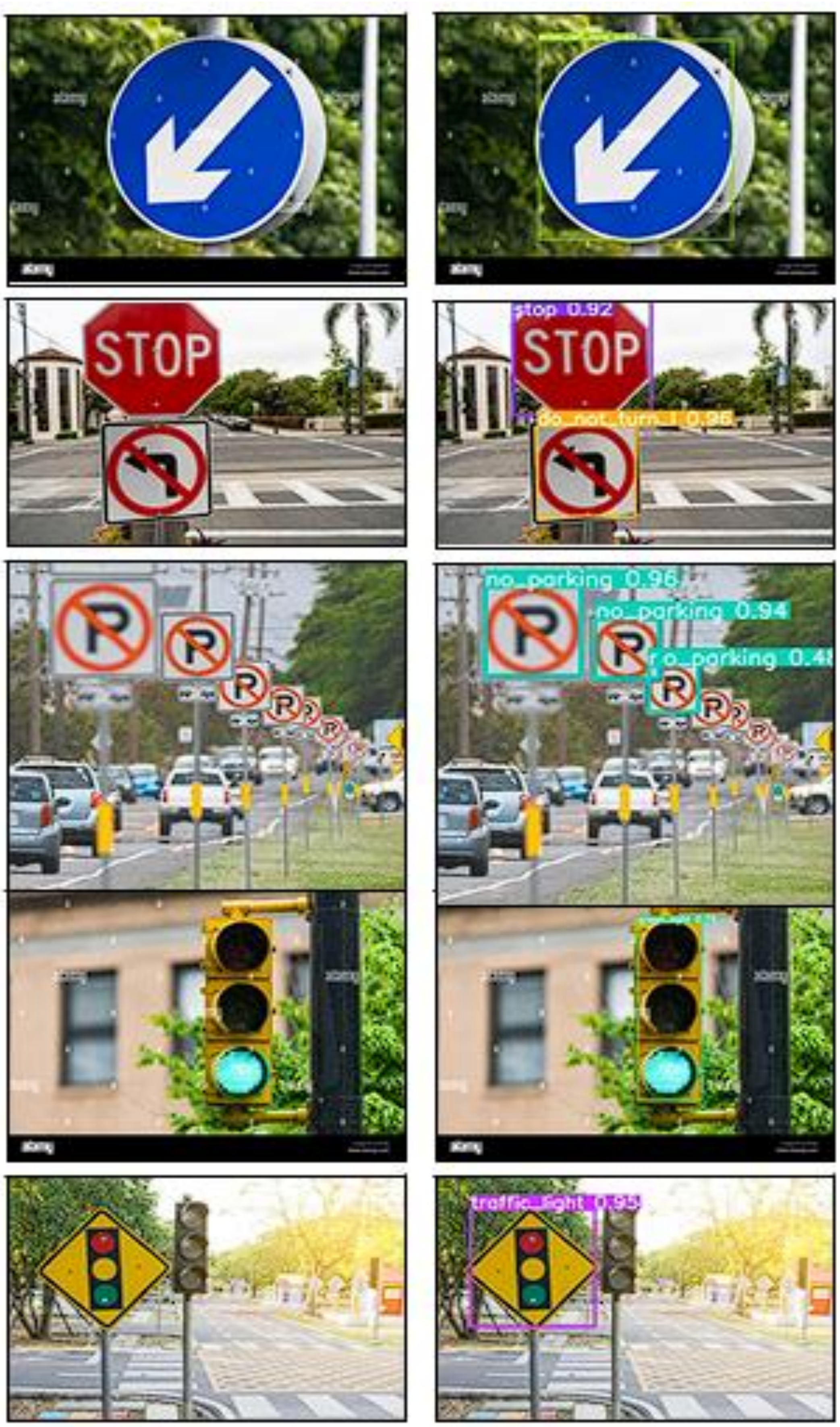
Let us list down all the 21 classes present in our project:

class_ids	**class_names**
0	bus_stop
1	do_not _enter
3	do_not_stop
4	do_not_turn_l
5	do_not_turn_r
6	do_not_u_turn
7	enter_left_lane
8	green_light
9	left_right_lane
10	no_parking
11	parking
12	ped_crossing
13	ped_zebra_crossing
14	railway_crossing
15	red_light
16	stop
17	t_intersection_l
18	traffic_light
19	warning
20	yellow_light

Fig. 7. Class IDs and Class Names

V. PREDICTIONS BY MODEL





VII. CONCLUSION

In conclusion, our project focused on the development of a robust and efficient traffic sign detection and recognition system using the YOLOv8 algorithm. Through extensive experimentation and analysis, we have successfully demonstrated the effectiveness of our approach in accurately detecting and classifying traffic signs in real-time images and videos. By training our model on a meticulously curated dataset comprising 2093 images categorized into 21 different classes, we ensured comprehensive coverage of various traffic sign types commonly encountered on roadways. Looking ahead, there are exciting prospects for further enhancing our project's functionality and accessibility. One avenue for future development involves integrating voice feedback capabilities using Python libraries like pyttsx3, thereby catering to individuals with visual impairments or low eyesight. Additionally, we envision transforming our project into an Android application to extend its reach to a wider audience. By converting our trained model weights into the .tflite format compatible with TensorFlow Lite, we can empower users to access our traffic sign detection system directly on their mobile devices, enhancing road safety and driving experience for all. In summary, our project represents a significant step forward in the domain of intelligent transportation systems, offering a practical solution for improving road safety and enhancing the driving experience through innovative technology and forward-thinking approaches.

ACKNOWLEDGEMENT

The research described in this paper was made possible by the faculty of Computer Science and Engineering, Shivalik College of Engineering, Dehradun, by providing experimental facilities to conduct research and development. We would like to thank Ms. Akansha Pundir for assisting us in various stages of this project and conducting research.

REFERENCES

[1] World Health Organization (WHO). Global Status Report on Road Safety 2018.
[2] TRD-YOLO: A Real-Time, High-Performance Small Traffic Sign Detection Algorithm by Jinqi Chu 1,Chuang Zhang
[3] 1,2,*,Mengmeng Yan 1,Haichao Zhang 1 andTao Ge 1.
[4] Fernando, W. H. D., Sotheeswaran, S.: Automatic Road Traffic Signs Detection and Recognition using 'You Only Look
[5] Once' version 4 (YOLOv4). (2021).
[6] Improved YOLOv5 network for real-time multi-scale traffic sign detection by Junfan Wang, Yi Chen, Mingyu Gao, Zhekang Dong.
[7] Rajesh Kannan Megalingam, Kondareddy Thanigundala, Sreevatsava Reddy Musani, Hemanth Nidamanuru, Lokesh Gadde, Indian traffic sign detection and recognition using deep learning,International Journal of Transportation Science and Technology,Volume 12, Issue 3,2023.