Object Detection Using YOLOv8

Narmety Vamshi
23125022
Email: vamshi_n@mfs.iitr.ac.in

Manchala Vineel Krishna
23125021
Email: vineel_km@mfs.iitr.ac.in

***Abstract*—Object detection is an essential task in computer vision, allowing automatic identification and localization of objects within images. In this work, we focus on applying the YOLOv8 object detection model to a custom traffic dataset. The dataset was created by capturing real-time road scenes during car travel and was carefully labeled and preprocessed to ensure quality and consistency. We provide an overview of the YOLO model family, explain the improvements introduced in YOLOv8, and describe the step-by-step implementation of our training pipeline. The results section includes space for both performance metrics and visual examples of detected objects. This study highlights the effectiveness of YOLOv8 in handling real-world traffic images and sets the stage for future developments in object detection tasks.**

I. Introduction

Object detection is a key task in computer vision that involves identifying and localizing multiple objects within an image. It plays a vital role in traffic monitoring, smart transportation, and autonomous driving by enabling the detection of vehicles, pedestrians, and traffic signs in real-time. Accurate and fast object detection contributes to safer roads and more efficient traffic systems.

Traffic scenes present several challenges such as occlusions, varying lighting conditions, and multiple moving objects. These factors require detection models that can maintain high accuracy while delivering real-time performance.

Traditional object detection methods like R-CNN and Fast R-CNN were based on region proposal pipelines but were computationally intensive. Faster R-CNN improved this by introducing a Region Proposal Network (RPN), making the system faster and end-to-end trainable [1].

The YOLO (You Only Look Once) family introduced a significant breakthrough by treating object detection as a single regression problem. YOLO models directly predict class labels and bounding boxes in a single pass, offering real-time detection capabilities.

YOLOv8, the latest version in this series, introduces key improvements such as anchor-free detection, decoupled classification and regression heads, and a more efficient backbone for feature extraction [2]. It also provides an easy-to-use training interface through the Ultralytics library.

In this work, we implement YOLOv8 on a custom traffic dataset captured using a moving vehicle-mounted camera. The dataset was carefully labeled and preprocessed to maintain consistency and quality. This paper details the dataset preparation, model training, and architecture overview, with dedicated space for both quantitative results and visual detection outputs.

II. Methodology

This section provides a comprehensive overview of the YOLO object detection framework, with a particular focus on YOLOv8, the architecture used in our study.

A. You Only Look Once (YOLO)

The YOLO (You Only Look Once) series introduced a significant shift in object detection by reframing it as a single-stage regression problem [1]. Unlike two-stage detectors such as Faster R-CNN, which rely on region proposals, YOLO models predict bounding boxes and class probabilities directly from the input image in a single forward pass.

The YOLO algorithm divides the input image into an $S \times S$ grid. Each grid cell is responsible for predicting:

- Bounding box coordinates (x, y, w, h)
- A confidence score representing the probability that an object is present in the bounding box
- Class probabilities for each object category

The confidence score is defined as:

$$\text{Confidence} = P(\text{Object}) \times \text{IoU}_{\text{truth,pred}} \tag{1}$$

where $P(\text{Object})$ is the probability of an object existing in the predicted bounding box, and $\text{IoU}_{\text{truth,pred}}$ is the Intersection over Union between the predicted and ground-truth bounding boxes.

1) YOLO Loss Function: The original YOLO loss function is a sum of localization, confidence, and classification errors. It is defined as:

$$\begin{aligned} L =& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\ &+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\ &+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ &+ \sum_{i=0}^{S^2} 1_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{2}$$

Here:

- λ_{coord} (usually 5) increases the penalty for localization errors.
- λ_{noobj} (usually 0.5) reduces the penalty for confidence errors when no object is present.
- 1_{ij}^{obj} is 1 if the j^{th} bounding box in grid cell i is responsible for detecting an object.

B. YOLOv8 Architecture

YOLOv8 is the latest advancement in the YOLO series and introduces several key improvements over previous versions [2].

1) Anchor-Free Detection: Unlike earlier YOLO models that relied on predefined anchor boxes of various sizes and aspect ratios, YOLOv8 uses an anchor-free detection head. This simplifies the training process and improves the model's ability to localize objects without being constrained by predefined shapes.

2) Decoupled Head: YOLOv8 employs a decoupled detection head that separates the tasks of classification and localization. This separation allows the model to optimize each task independently, improving both detection accuracy and bounding box precision.

3) Backbone and Neck: The YOLOv8 backbone is based on a CSPDarknet-AA architecture, which efficiently extracts multi-scale features. The neck uses a PANet (Path Aggregation Network) style structure that merges feature maps from different scales, enhancing the model's ability to detect objects of varying sizes.

4) Loss Function in YOLOv8: YOLOv8 uses a modernized loss function that combines the following components:

$$L_{\text{YOLOv8}} = \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{box}} L_{\text{box}} + \lambda_{\text{obj}} L_{\text{obj}} \tag{3}$$

Where:

- L_{cls} is the Binary Cross-Entropy (BCE) loss for classification.
- L_{box} is the Generalized Intersection over Union (GIoU) loss for bounding box regression.
- L_{obj} is the objectness loss, indicating the presence or absence of an object.

5) Data Augmentation: YOLOv8 employs advanced data augmentation techniques, including:

- Mosaic augmentation: Combining four images into one during training to improve detection of small objects.
- MixUp augmentation: Blending two images and their labels to increase dataset diversity.
- Geometric transformations: Random scaling, flipping, and cropping.

C. Training Pipeline

The YOLOv8 training process follows these steps:

1) Preprocessing images and annotations to fit the YOLO format.
2) Training with Ultralytics YOLOv8 for 50 epochs using the YOLOv8s pre-trained model.
3) Hyperparameter tuning with a batch size of 16 and input image size of 640 $\times$ 640 pixels.
4) Model evaluation using mean Average Precision (mAP), precision, and recall metrics.

III. Experimental Details

A. Dataset

We conducted our experiments using a custom traffic dataset captured in real-time through a moving vehicle equipped with a front-facing camera. The collected dataset covers a variety of road environments including city streets, highways, and intersections under different lighting and weather conditions. After preprocessing steps such as removing corrupted files and empty annotations, the dataset was finalized for object detection tasks.

The dataset consists of twelve traffic-related object classes, including cars, pedestrians, bikers, traffic lights, trucks, and stop signs. The class distribution is summarized in Table I.

TABLE I: Class-wise distribution in the dataset

ID	Class Name	Instances
0	Biker	8,932
1	Car	92,211
2	Pedestrian	19,019
3	Traffic Light	15,018
4	Traffic Light - Green	7,336
5	Traffic Light - Green Left	2,979
6	Traffic Light - Red	13,223
7	Traffic Light - Red Left	1,743
8	Traffic Light - Yellow	3,637
9	Traffic Light - Yellow Left	2,118
10	Truck	5,978
11	Stop Sign (Arret)	1,747

We employed a stratified dataset splitting strategy to ensure balanced class representation across the training, validation, and test sets. The final split ratios were 70% for training, 15% for validation, and 15% for testing. To preserve class balance, we used a class-wise sampling method that prevented duplication and maintained data diversity.

1) Dataset Splitting: The dataset splitting was automated using a custom Python script that:

- Grouped label files by class to ensure balanced sampling.
- Randomly shuffled and distributed samples to train, validation, and test sets.
- Avoided duplicate file selections across splits.
- Ensured each split contained a proportional representation of all classes.

All images and their corresponding annotation files were copied into dedicated folders for each split to comply with the YOLOv8 training format.

B. Implementation Details

The YOLOv8 object detection model was implemented using the Ultralytics library, which provides streamlined training, evaluation, and visualization pipelines. Training was conducted on Kaggle's NVIDIA Tesla P100 GPU environment, which constrained training duration. Due to GPU usage limitations, the model was trained for 30 epochs.

1) Training Configuration: The YOLOv8 model was trained using Ultralytics' default settings, including a learning rate of 0.01, SGD optimizer with a momentum of 0.937, and an input image size of 640 × 640 pixels. The training was performed on Kaggle's NVIDIA Tesla P100 GPU, and due to GPU usage limitations, the model was trained for 30 epochs. Despite the restricted training time, the model showed stable convergence.

2) Data Augmentation: We employed standard augmentation techniques to improve model robustness, including:

- Random horizontal flipping with 50% probability
- Random scaling between 0.8 and 1.2
- Random adjustments to brightness, contrast, and saturation
- Mosaic augmentation for creating multi-context training samples
- MixUp augmentation to enhance data diversity

The training procedure efficiently leveraged YOLOv8's built-in augmentation pipeline, reducing preprocessing overhead.

IV. Results and Discussion

The performance of the YOLOv8 model was evaluated using both qualitative and quantitative analyses on the test dataset. The model was assessed in terms of accuracy, precision, recall, and F1-score at both macro and micro levels.

A. Quantitative Results

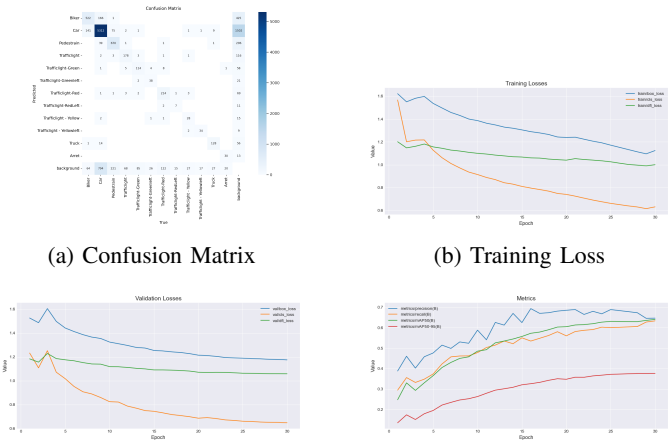
The overall accuracy achieved by the YOLOv8 model on the test set was 61.36%. The macro-averaged precision and recall were 57.41% and 57.13%, respectively, while the micro-averaged precision and recall were both 61.36%. Class-wise performance metrics are detailed in Table II.

TABLE II: Class-wise Performance Metrics

Class	Accuracy	Precision	Recall	F1-score
Biker	71.70%	46.86%	71.70%	56.68%
Car	83.90%	75.41%	83.90%	79.43%
Pedestrian	69.00%	66.47%	69.00%	67.71%
Trafficlight	69.26%	58.55%	69.26%	63.46%
Trafficlight-Green	55.07%	59.69%	55.07%	57.29%
Trafficlight-GreenLeft	55.07%	62.30%	55.07%	58.46%
Trafficlight-Red	61.32%	72.79%	61.32%	66.56%
Trafficlight-RedLeft	30.43%	35.00%	30.43%	32.56%
Trafficlight-Yellow	45.16%	59.57%	45.16%	51.38%
Trafficlight-YellowLeft	65.38%	75.56%	65.38%	70.10%
Truck	77.58%	64.32%	77.58%	70.33%
Arret (Stop Sign)	58.82%	69.77%	58.82%	63.83%
Background	0.00%	0.00%	0.00%	0.00%

The confusion matrix, shown in Figure 1, illustrates the distribution of correct and incorrect classifications across all classes. The model achieved strong detection performance for high-frequency classes such as *Car* and *Pedestrian*. However, the detection accuracy for minority classes like *Trafficlight-RedLeft* and *Trafficlight-Yellow* was relatively lower, indicating class imbalance effects in the dataset.

The training and validation loss curves confirm that the model exhibited a stable convergence trend over the 30 training epochs. Additionally, the performance metrics curve provides insights into the model's confidence and mAP evolution throughout training. All four visualizations together offer a comprehensive understanding of the model's learning behavior and final performance.



(a) Confusion Matrix (b) Training Loss

(c) Validation Loss (d) Metrics Curve

Fig. 1: Model evaluation visualizations: (a) Confusion matrix indicating class-wise detection performance, (b) Training loss curve showing convergence, (c) Validation loss curve demonstrating generalization stability, and (d) Metrics curve tracking precision and mAP during training.

V. Conclusion

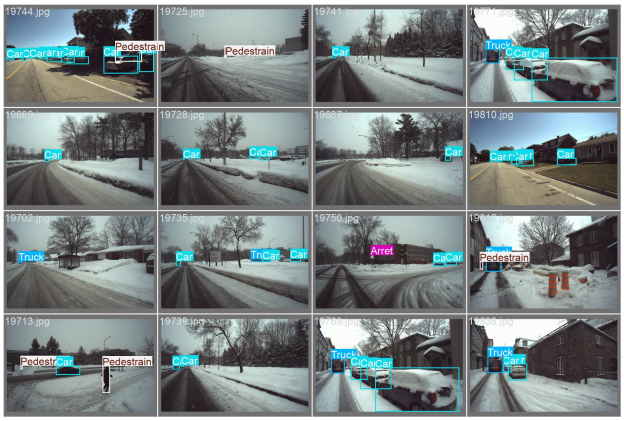
In this study, we implemented the YOLOv8 object detection model using a real-world traffic dataset captured through on-road driving scenarios. Our approach demonstrated the capability of YOLOv8 to effectively detect and localize various traffic-related objects under diverse conditions. The model achieved competitive class-wise performance, particularly excelling in frequently occurring classes such as cars and pedestrians, while facing challenges with less represented classes like traffic lights with directional indicators. Despite computational constraints and limited training epochs, YOLOv8 showed stable convergence and reliable detection across multiple categories. This work validates the robustness and efficiency of YOLOv8 for real-time traffic scene understanding and sets the foundation for future improvements through enhanced training resources and fine-tuning strategies.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. of the IEEE CVPR*, 2016.

[2] Terven, Juan, and Domingo Cordova-Esparza. "A Comprehensive Review of YOLO Architectures in Object Detection," 2023.

[3] Ultralytics YOLOv8, GitHub Repository. [Online]. Available: https://github.com/ultralytics/ultralytics



(a) Ground Truth: Sample 1

(b) Predictions: Sample 1

(c) Ground Truth: Sample 2

(d) Predictions: Sample 2

(e) Ground Truth: Sample 3

(f) Predictions: Sample 3

Fig. 2: Qualitative evaluation of YOLOv8 predictions compared to ground truth on validation images. The model effectively detects common objects like cars and pedestrians but occasionally struggles with smaller or less frequent classes.