EXPLORING YOLO OBJECT DETECTION USING DEEP LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

HELI HATHI

92100103341

HET BUCH

92100103196

BACHELOR OF TECHNOLOGY

in

Computer Engineering

Marwadi University, Rajkot

April, 2024

Marwadi University
Marwadi Chandarana Group

Mini Project (01CE0609)

Marwadi University

Faculty of Technology

Department of Computer Engineering

2023-24

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **EXPLORING YOLO OBJECT DETECTION USING DEEP LEARNING TECHNIQUES** has been carried out by **Heli Hathi** (92100103341) and **Het Buch** (92100103196) under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 6th Semester of Marwadi University, Rajkot during the academic year 2023-24.

Prof. Ravikumar R Natarajan	Dr. Krunal Vaghela
Assistant Professor	Associate Professor
Internal Guide	Head of the Department

Marwadi University

Rajkot

DECLARATION

We hereby declare that the **Mini Project (01CE0609)** report submitted along with the Project entitled **EXPLORING YOLO OBJECT DETECTION USING DEEP LEARNING TECHNIQUES** submitted in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering to Marwadi University, Rajkot, is a bonafide record of original project work carried out by me / us at Marwadi University under the supervision of **Prof. Ravikumar R Natarajan** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

	Name of the Student	Sign of Student
1	_____	_____
2	_____	_____

Acknowledgement

We would like to sincerely thank everyone that contributed to the project's success. Their tremendous advice, assistance, and technical know-how were crucial to the growth process.

We would also want to express our gratitude to our committed mentor, Professor Ravikumar Natrajan, for his important advice and assistance during the project. The effective completion of the project was greatly influenced by his oversight and insightful observations.

We also owe a debt of gratitude to Dr. Krunal Vaghela, the distinguished head of Marwadi University's department of computer engineering, for his persistent support and for fostering a collaborative atmosphere. This made a big difference in the project's success.

We sincerely thank our friends and coworkers for their constant support, insightful criticism, and insightful comments. Their support turned out to be crucial.

Heli Hathi,

Het Buch.

Abstract

Object detection is a fundamental task in computer vision with applications ranging from surveillance to inventory management. This project explores the efficacy of YOLOv8, a state-of-the-art deep learning model, for object detection tasks, focusing on barcode, QR code, and top view vehicle detection. This project evaluates YOLOv8 across different model iterations - Nano, Small, and Medium versions, with meticulous attention to precision, recall, and F1 assessment metrics. Results indicate significant improvements in object detection accuracy with each subsequent model refinement. Specifically, the Nano model achieves 88.95% accuracy, the Small model achieves 97.10% accuracy, and the Medium model achieves 94.10% accuracy for barcode and QR code detection. Additionally, the Medium model achieves a 73% accuracy rate for top view vehicle detection. Furthermore, a web application leveraging Streamlit is developed to demonstrate the practical application of the trained models. The application allows users to detect barcodes, QR codes, and vehicles in real-time through an intuitive interface. This project showcases the effectiveness of YOLOv8 for object detection tasks and underscores the importance of model iteration and evaluation in achieving high accuracy rates. Potential future directions include further fine-tuning of models and integration of real-time processing capabilities into the web application for enhanced utility in various domains.

List of Figures

List of Tables

Abbreviations

YOLO	**You Only Look Once**
CNN	**Convolutional Neural Network**
IOU	**Intersection over Union**
GPU	**Graphics Processing Unit**
GUI	**Graphical User Interface**
ML	**Machine Learning**
DL	**Deep Learning**
CPU	**Central Processing Unit**
mAP	**Mean Average Precision**
cls_loss	**Classification Loss**
dfl_loss	**Distribution Focal Loss**
LR	**Learning Rate**
BS	**Batch Size**
Opt	**Optimizer**
TP	**True Positive**
FP	**False Positive**
TN	**True Negative**
FN	**False Negative**
SSD	**Single Shot Detector**
SSPNet	**Spatial Pyramid Pooling Network**
R-CNN	**Region based Convolutional Neural Network**
PAFPN	**Path Aggregation Feature Pyramid Network**
QR Code	**Quick Response Code**
PR	**Precision-Recall**
BCE	**Binary Cross Entropy**
AP	**Average Precision**
NMS	**Non-maximal Suppression**

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Project Summary

This project proposes a deep learning-based strategy for detecting 1D/2D codes and vehicles, with the focus being on determining the performance of the YOLO algorithm in object detection. To achieve this, we trained the YOLOv8 model with a custom dataset that includes barcodes and QR codes from various scenarios, as well as an external dataset developed for vehicle detection. This project focuses on detecting three unique categories: barcodes (1D codes), QR codes (2D codes), and cars from a top-view perspective utilizing the YOLOv8 architecture. To improve the accuracy of our model, we applied various types of augmentation to the dataset images. We trained three different YOLOv8 architectures for detection of Barcodes and QR codes (nano- YOLOv8n, small- YOLOv8s, and medium- YOLOv8m) on our dataset as well as each of the five YOLOv8 architectures for detecting vehicles (nano- YOLOv8n, small- YOLOv8s, and medium- YOLOv8m, large- YOLOv8l, extra-large- YOLOv8x). We discovered that YOLOv8s detected barcodes and QR codes with an accuracy of 97.1%, while YOLOv8m and YOLOv8l detected vehicles with an accuracy of 95.4% each. The Yolov8 model was then incorporated into a web-based application for real-time detection.

1.2 Purpose

In the realm of computer vision, the rapid and precise identification of objects within images or videos is essential for numerous applications. Machine-learning algorithms have competitive inference speed owing to their relatively easy calculation, but they often have low accuracy since they are trained on specific characteristics. Object detection techniques driven by deep learning architectures have developed rapidly and achieved fruitful results. This project adopts a deep-learning-based object-detection approach to enhance both detection accuracy and real-time performance. This project tests how good the latest version of YOLO from Ultralytics is, specifically the model YOLOv8.

1.3 Objective

Utilizing the real-time detection abilities of YOLOv8, we performed a study aimed at enhancing its talent in swiftly and correctly figuring out objects. Through large training and high-quality-tuning on Kaggle datasets tailored for Barcode and QR code detection as well as small but meticulously pre-processed and augmented Top view vehicle detection, our goal became to optimize YOLOv8's overall performance throughout various situations and environments.

1.4 Scope

YOLOv8, built on the principle of processing entire images through a single-pass CNN, YOLOv8 eliminates the need for painstakingly analysing individual parts of the image. This streamlined approach not only enhances efficiency but also ensures impressive gains in speed. In practical scenarios, where swift and accurate object detection is paramount, YOLOv8 stands as a beacon of promise. Despite these advantages, YOLO struggles to detect small objects. Furthermore, YOLOv8 may encounter challenges in accurately localizing objects, especially when objects are closely packed or overlapping.

1.5 Literature Review

Numerous studies and contributions have made important advances in the field of object detection. Notably, the YOLO strategy stands out as the latest method, valued for its fast computational speed and low parameter needs. A thorough review of recent research, as shown in Table 1.1, demonstrates a strong dependence on the YOLO framework for object detection tasks. Some studies improve their models by incorporating specific modules (e.g., [15] and [17]) or combining them with complementary algorithms (e.g., [13], [14], [19], [20]). While numerous datasets serve as the foundation for object detection tasks, there is a clear trend toward the use of custom datasets customized to specific research aims (e.g., [11], [13], [16], [20], [22]). Furthermore, with rare exceptions, most studies use a learning rate of 0.01 for their investigations ([10], [16], [20]). Further, a common choice of epochs falls between 200 and 500, indicating efforts to improve model efficiency. The most common evaluation measures used in these investigations are accuracy and mAP. Some studies, such as Ref No. 18 (1080×1080) and Ref No. 22 (416×416), use a different image size than the

standard 640×640. These findings demonstrate the importance of YOLO-based techniques in driving progress in object detection research.

Table 1.1 Literature Review

Ref No.	Dataset Used	Model	Parameters Used	Accuracy
[10]	FMD Dataset	YOLOv8s	LR=0.1, BS=16, Epochs=200	mAP(0.95) - 0.93
[11]	US Dataset along with custom Dataset	YOLOv8n, YOLOv8m	LR=0.01, Epochs=50, Opt.=SGD	99.8%
[12]	TT100k Dataset	YOLOv8 (Customized)	LR=0.01, BS=32, Epochs=200	80.8%
[13]	Custom Dataset	YOLOv8 (n, m, s) and CenterNet model	LR=0.01, BS=2, Opt.=SGD	95%
[14]	Pascal VOC Dataset	SSB-YOLO	LR = 0.01, BS=16, Opt.=SGD, Epochs=300	87.5%
[15]	COD10K, CAMO Dataset	YOLOv8+EFM+ EEM	LR=0.01, BS=32, Opt.=SGD, Epochs=500	70.5% 64%
[16]	Weeds Public Dataset along with Custom Dataset	EfficientDet, YOLOv5m, YOLOv6l, YOLOv7, YOLOv8l	LR=0.0001, BS=8, Epochs=100, Opt.=SGD	88.23% (Weeds) 52.25% (Custom)
[17]	BDD100K, NEXET Dataset	YOLOv8 (Modified: SnakeVision)	LR=0.01, BS=32, Epochs=1000, Layers=168	mAP(50-95)-0.449 mAP(50-95)-0.495
[18]	Obj. Detection and localization of image Dataset	YOLOv8 (with Transfer Learning)	LR=0.01, Epochs=20	mAP(50) of 0.874 at 66.7 FPS
[19]	LSI X-Ray, OPI X-Ray Dataset	SC-YOLOv8	Not Provided	LSI X-Ray-82.7% OPI X-Ray-89.2%
[20]	Custom Dataset	SHFP-YOLO	LR=0.001, BS=16, Epochs=500, Opt.=SGD	mAP(50-95)-0.42

[21]	MASK, Face (FDDB) Dataset	YOLOv8	Epochs = 100	84.6%
[22]	Custom Dataset	YOLOv8	Epochs=100, BS=64	84.7%
[23]	Open Source Dataset	YOLOv8n	Not Provided	>=95%
[24]	RailSem19 Dataset	YOLOv8x	LR=0.01, Opt.=SGD	mAP(50)-0.88

1.6 Project Scheduling

Our project followed a disciplined strategy to ensure proper execution and early completion. We methodically planned project phases using Gantt charts, from the beginning of the YOLOv8 study to the final journal paper writing. This enabled clear visualization and resource allocation, allowing for more effective cooperation. In addition, regular team meetings with our project guide encouraged open communication, cooperation, and rapid resolution of any developing difficulties. By adhering to these principles, we kept the project on track and achieved results that met our goals.

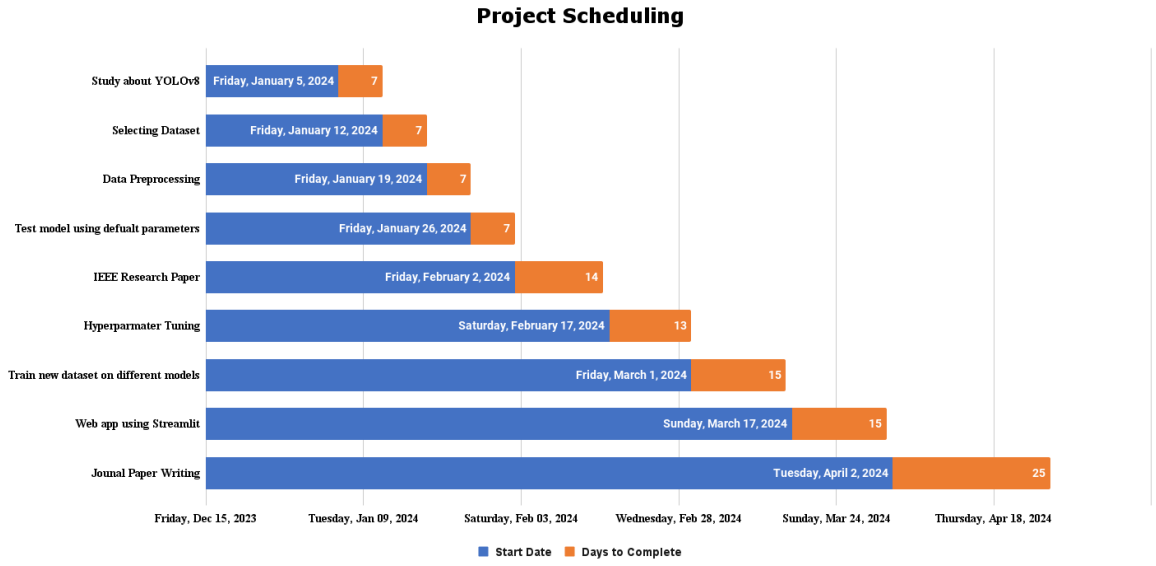

Fig. 1.1 Gantt Chart of Project Workflow

CHAPTER 2

SYSTEM ANALYSIS

2.1 Study of Current System

At present, there are two types of computer-vision based detection: traditional machine learning ways and deep-learning-based methods. Machine-learning based techniques have low accuracy as they are trained on selected features and do not perform well if the image is not seen before training. Object detection techniques driven by deep learning architectures have developed rapidly and achieved fruitful results. It can be categorized into two-stage object detectors and one-stage object detectors. Examples of two-stage models include R-CNN, Fast R-CNN, Faster R-CNN, SPPNet etc., Two stage detectors are designed flexibly and covers a wide range of work.

2.2 Problem and Weaknesses of Current System

Although, two-stage detectors perform better, but due to the need to generate a large number of candidate regions, it increases computational complexity and reduces detection speed. In contrast, one-stage object detection models directly extract visual features to predict the object class and location. Typical one-stage object detection methods are SSD, YOLO, and their variants, CenterNet, M2det, etc.

2.3 Requirements of New System

In order to reduce the high computational cost and improve the slow detection speed caused by obtaining higher detection accuracy, researches were carried on discarding the object generation phase and investigated one-stage object-detection algorithms. Of all the one-stage object detection algorithms, state-of-the-art one-stage object detection network YOLOv8 is chosen as the backbone in this project. Below are the steps of installing and executing YOLO algorithm.

1. Install YOLOv8 using pip
2. Select open-source dataset with labelled images
3. Export your dataset for use with YOLO v8
4. Use the yolo command line utility to run train a model
5. Run inference with the YOLO command line application

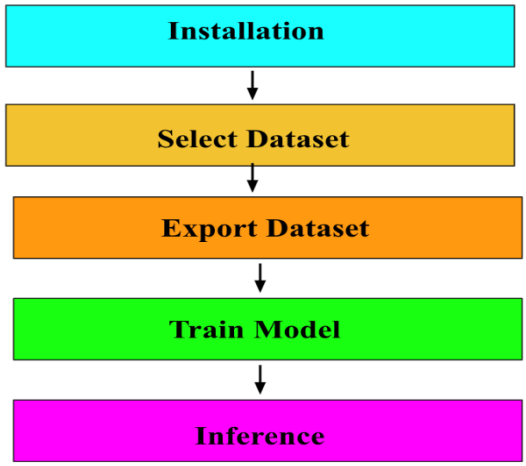

Fig 2.1 Steps of using YOLOv8

2.4 System Feasibility

2.4.1 Does the system contribute to the overall objectives of the organization?

Ans. Yes

2.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints?

Ans. Yes

2.4.3 Can the system be integrated with other systems which are already in place?

Ans. Yes

2.5 Proposed System

The main advantage of using YOLO is to faster execution time because of less parameters and simple architecture. The two main criteria that the project has focused on are the precision and speed of the model, which also leads to the goal of the project. The goal here is to see if the YOLOv8 object detection algorithm is fast and precise enough to be used in real-time detection or not. YOLOv8 combines the advantages of fast detection speed and a small number of network parameters.

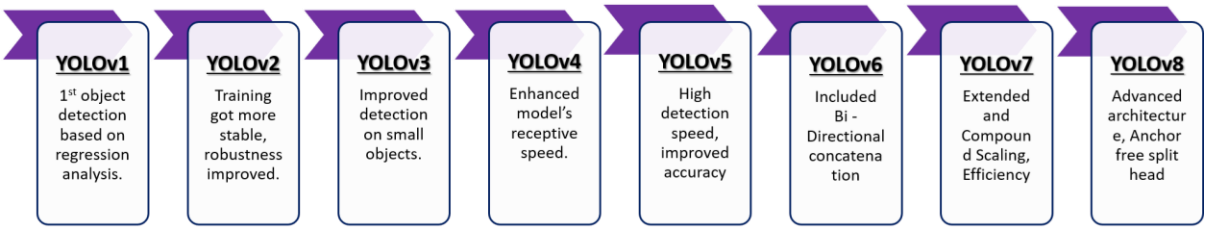

Fig. 2.2 Versions of YOLO

2.6 Features of Proposed System

A. Advantages of using YOLOv8

- **Better mAP:** YOLOv8 surpasses previous versions in mean average precision, excelling in detecting smaller objects and challenging conditions.
- **Efficiency:** Achieves high performance with less parameters, potentially reducing learning occasions.
- **Developer-friendly:** Enhancements like multi-GPU aid, greater serialization simplifying improvement techniques.
- **Robust:** Handles scale variations and other challenges successfully, ensuring dependable outcomes across diverse scenarios.
- **Faster Inference:** YOLOv8 gives you fast inference speeds, perfect for real-time packages which include self-driving motors and video evaluation.

B. Applications of YOLOv8

- **Autonomous Vehicles:** Detect objects on roads in real time, crucial for independent automobile systems.
- **Security and Surveillance:** Monitors high-protection places like airports, enhancing surveillance structures.
- **Medical Imaging:** Aids in decoding medical scans for prognosis and remedy planning.
- **Industrial Automation:** Tracks gadgets on manufacturing strains, detects defects, ensures best management.

C. Goals/Outcomes from YOLOv8

- Evaluate the overall performance of YOLOv8 in object detection, especially in appropriately across smaller gadgets and in hard conditions.
- Examine the performance of YOLOv8 in parameter utilization and training times, aiming to streamline the model and improve computational efficiency.
- Analyse the flexibility offered by way of special variants of YOLOv8 (YOLOv8n, YOLOv8s, YOLOv8m) for various needs, balancing elements along with accuracy, velocity, and useful resource utilization.
- Provide insights into the strengths and boundaries of YOLOv8 based on empirical evaluations and actual international use instances, aiming to manual researchers and practitioners in leveraging this technology effectively.
- Contribute to the continuing discourse surrounding similar approaches in Computer Vision, advancing the knowledge of modern-day object-detection techniques and their effects on numerous programs.

2.7 Components of Proposed System

Ultralytics' YOLOv8, which was launched in January 2023, follows the release of YOLOv5 in 2020. It works with CLI and PIP packages and has integrations for labelling, training, and deployment. To improve accuracy, YOLOv8 uses more augmentation during training, such as online image augmentation and mosaic augmentation, which combines four images every iteration. These strategies help the

model train more successfully. YOLOv8 outperforms previous versions due to the use of a bigger feature map and a more efficient convolutional network. YOLOv8 comprises three components: a backbone (Extraction of features), a neck for combining multi-level features, and a head for outputting classification and localization predictions. Compared to YOLOv5, YOLOv8 eliminates the top-down up-sampling phase of the PA-FPN in the neck layer, replacing the C3 module with the more gradient-enriched C2f. The system's backbone consists of three major components: the Conv, C2f, and SPPF modules. These modules interact to complete a variety of tasks. First, two Conv modules are used to determine essential traits. Subsequently, the C2f module strengthens these characteristics. The SPPF module refines the features by examining different parts of the image. The backbone structure follows the PA-FPN architecture, which uses separate parts for semantic comprehension and deep analysis. Finally, the major module produces three unique maps that depict different parts of the image. The head module decouples the classification and localization processes and replaces the anchor-based concept with Anchor-free, reducing model computations and improving convergence speed and effectiveness. It mostly includes loss calculation and bounding box filtering. YOLOv8 uses a positive-negative pattern matching approach similar to Task Aligned Assigner. This technique, when combined with classification and regression, picks positive samples based on scores. The loss function combines BCE Loss and Bounding Box Regression Loss, as well as dfl_loss and complete IoU. To extract bounding boxes, YOLOv8 filters predicted boxes using the NMS method. YOLOv8 comes in numerous variants, including YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, which varies in parameter size, depth, and computing needs. YOLOv8s, chosen for its quicker detection speed and greater accuracy, is recommended because of its low computational cost. The main objective for using this algorithm was for detecting objects, with the help of a single neural network. The algorithm is straight-forward to build and can be moulded precisely on an entire image. Overall, the architecture of YOLOv8 is designed to achieve a balance between speed and accuracy, making it well-suited for real-time object detection tasks in various applications.

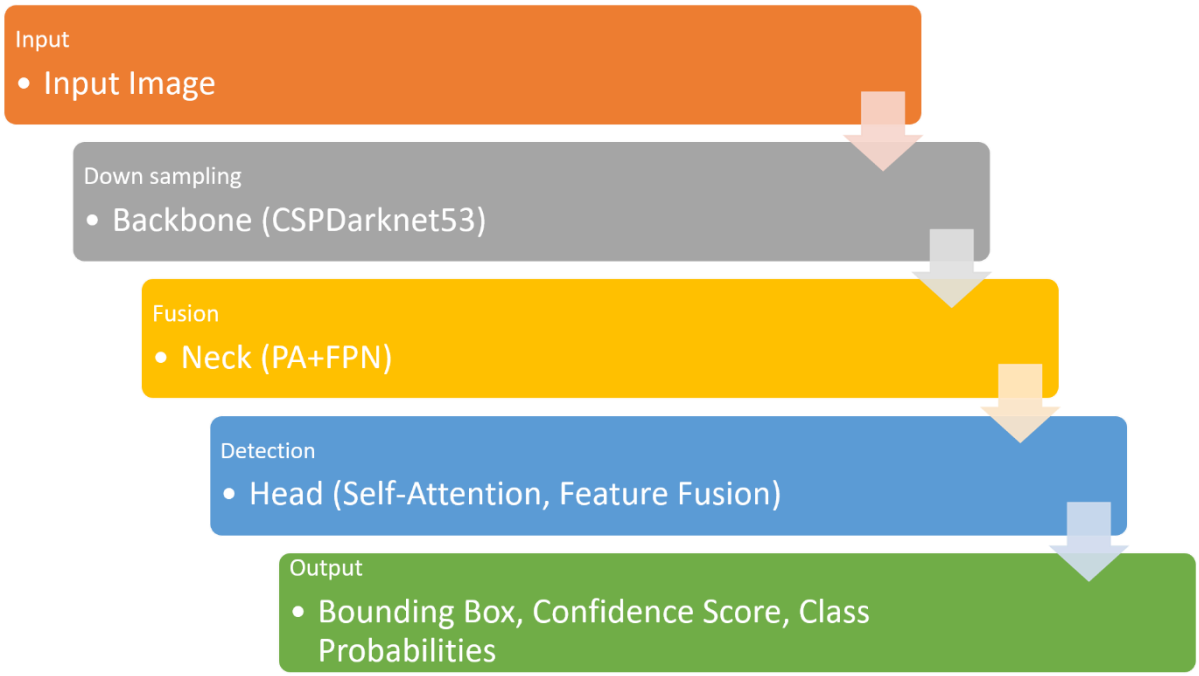

Fig. 2.3 Architecture of YOLOv8 n/s/m/x/l

2.8 Selection of Hardware and Algorithm

During the course of this project, we have used various algorithms, modules in a carefully crafted environment. For this project we have used following technologies and hardware configuration:

A. Software Requirements

- Roboflow – For image annotation
- PyTorch – 2.x
- Python 3.11x
- YOLOv8 (From Ultralytics)
- Tensorflow
- Streamlit – For web-application
- Weights & Biases – For result visuals

B. Hardware Requirements

- GPU – T4 (×2)

CHAPTER 3

SYSTEM DESIGN

3.1 Methodology

Methodology is a systematic approach or collection of procedures that aim to simplify and improve the flow of activities, information, and processes inside an organization or project. It describes the methods, regulations, and resources needed to efficiently fulfil a specified activity or objective. The process consists of six steps: data collection, preprocessing, annotation, training a YOLO model, evaluation, and detection and classification.

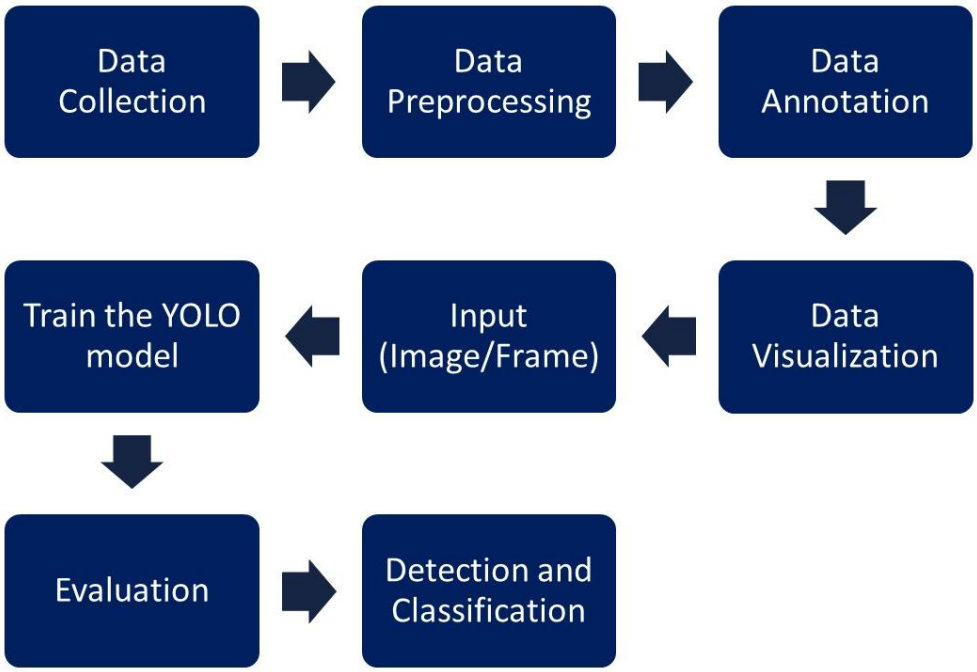

Fig. 3.1 Workflow Methodology

A. Data Collection

The first part of the procedure involves data gathering, in which a custom dataset (Barcode and QR code) of 31,078 images is selected from Kaggle and divided into training (28,696) and validation (2,382) images and another one is top view vehicle detection dataset which consists of 626 total images of which training images are 536 and validation image set contains of 90 images.

B. Data Preprocessing

The obtained data is then pre-processed to clean and optimize it for training. Following preprocessing and data annotation is performed using RoboFlow. Preprocessing- Auto-Oriented: Applied, Resize: Stretch to 416×416, Modify Classes: 4 remapped, 0 dropped.

C. Data Annotation

In the augmentation process, each training example undergoes four distinct transformations to enhance the diversity of the dataset. These transformations include horizontal and vertical flips, as well as rotations of 90° clockwise, counterclockwise, and upside down. Additionally, cropping is applied with a range from 0% minimum zoom to 22% maximum zoom to vary the scale and composition of the images. Rotation within the range of -15° to +15°, along with horizontal and vertical shearing within ±15°, further contribute to the variability of the dataset. Moreover, a blur effect of up to 3.25 pixels and noise of up to 5% of pixels are introduced to simulate real-world imperfections. Furthermore, cutout augmentation is performed with three randomly placed boxes, each covering 10% of the image's area, to encourage robustness in the model's learning process. By applying these augmentations, the dataset becomes enriched with diverse variations, thereby improving the model's ability to generalize effectively to unseen data.

D. Training the YOLO model

For barcode and QR code - After data preprocessing and annotation, the YOLOv8 model is used for training. Three variants of the YOLOv8 model are used for this purpose: YOLOv8n, YOLOv8s, and YOLOv8m, with a learning rate of 0.01 and training epochs set to 10.

For top view vehicle detection - Five variants of the YOLOv8 model are used for this purpose: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l and YOLOv8xl with a learning rate of 0.01 and training epochs set to 10 and 20 both cases.

Image size taken for this study is 416×416.

E. Input (Image/Frame)

Input is the images provided through dataset

F. Data Visualization

The YOLOv8 methodology's data visualisation portion usually includes methods for examining and assessing the dataset that was used to train the model for YOLOv8 object recognition. In order to better understand the distribution and properties of the objects included in the photos, this approach frequently involves visualising annotated bounding boxes superimposed on sample photographs from the collection. Furthermore, the class distribution and aspect ratios of objects can be analysed using histograms or other visual aids, which can help choose the best augmentation techniques and model configurations. Moreover, the model's performance can be understood and possible areas for improvement can be found by visualising the predictions on photos from a validation or test set. Through thorough data visualisation, practitioners and researchers can learn important insights about the dataset and make informed decisions throughout the model development process.

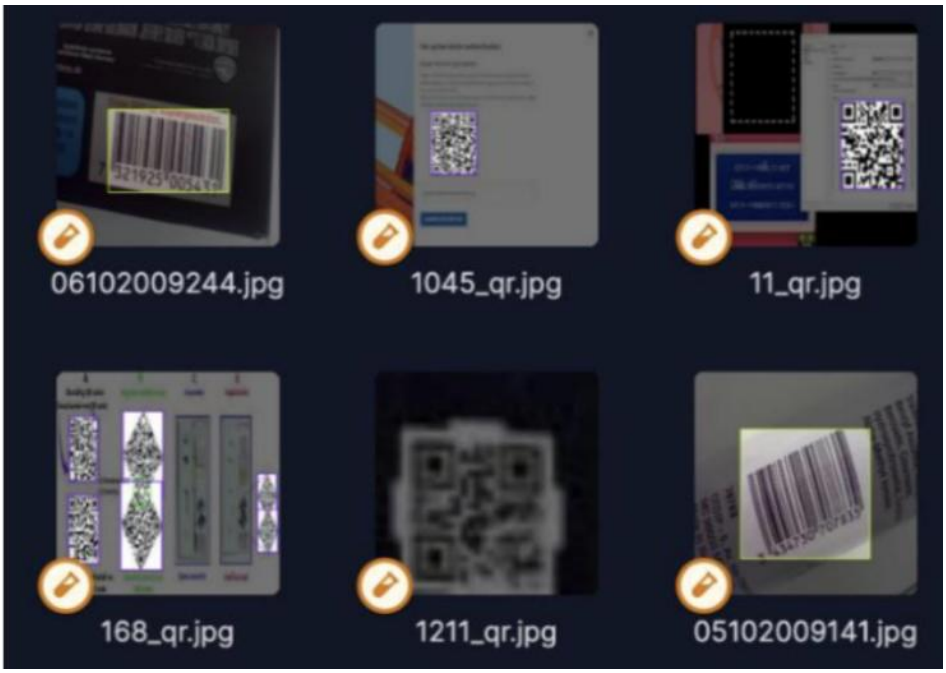

Fig. 3.2 Barcode and QR code dataset

G. Evaluation

Evaluate the model's performance on unseen data using metrics like Precision, Recall, AP, and mAP and analyze the model's strengths and weaknesses based on the

evaluation results and iteratively refine the training process or model architecture for better performance.

1. Confusion Matrix:

   A confusion matrix is a tabular representation of actual vs. predicted classes. It consists of four categories: TP, TN, FP, and FN. These values can be calculated by comparing the ground truth labels with the predicted labels.

2. Precision:

   Precision measures the proportion of true positive detections among all positive predictions. It is calculated as TP / (TP + FP).

3. Recall:

   Recall (also known as sensitivity) measures the proportion of true positive detections among all actual positives. It is calculated as TP / (TP + FN).

4. IoU:

   IoU measures the overlap between the ground truth bounding box and the predicted bounding box. It is calculated as the area of overlap between the two bounding boxes divided by the area of their union.

5. F1 Score:

   F1 score is the harmonic mean of precision and recall, providing a single metric that balances both. It is calculated as 2 × (precision × recall) / (precision × recall).

6. Classification Loss (cls_loss):

   Classification loss measures the discrepancy between predicted class probabilities and ground truth class labels. It is commonly computed using a softmax or sigmoid function followed by a cross-entropy loss.

7. Confidence Loss (dfl_loss): Confidence loss penalizes incorrect confidence scores assigned to predicted bounding boxes. It is typically computed using a binary cross-entropy loss, where the target values indicate whether an object is present in a grid cell.

8. Localization Loss (box loss):

   Localization loss measures the discrepancy between predicted bounding box coordinates and ground truth bounding box coordinates. Commonly, smooth L1 loss or squared L2 loss is used for this purpose.

9. Mean Average Precision (mAP):

   mAP is a popular metric for object detection that calculates the average precision (AP) across multiple classes and then averages these values to obtain the mean. AP is computed by first calculating precision and recall values at different confidence score thresholds and then computing the area under the curve (PR curve).

10. Accuracy:

    Accuracy measures the proportion of correctly predicted bounding boxes among all predicted bounding boxes. For object detection, accuracy is often computed using IoU thresholds, where a predicted bounding box is considered correct if its IoU with the ground truth bounding box exceeds a certain threshold.

H. Detection and Classification

A series of convolutional and linear layers process the enhanced features to generate the final object detection outputs. These outputs typically include:

- **Bounding Box Predictions:** The model predicts bounding boxes that enclose detected objects in the image.
- **Confidence Scores:** For each predicted bounding box, a confidence score indicates the likelihood that the box contains a valid object.
- **Class Probabilities:** The model predicts the probability of each object class belonging within a bounding box.

3.2 Output Interface Design

After making the object detection model using YOLOv8 on both the datasets i.e. Barcode and QR code detection and Top View vehicle detection, we tried to use those models in real-time application. So, we created a web-application using Streamlit where a user can choose model (Barcode/QR code or Top View detection) based on the input choice, and get the output with the confidence score of that object along with the bounding box. Streamlit is a Python framework enabling quick building of interactive web applications. Streamlit simplifies interface creation by allowing developers to use Python functions to add widgets like sliders, buttons, and text inputs. Developers may easily construct dynamic interfaces because of its declarative syntax and vast widget library. Its easy connection with data science libraries and simple deployment choices makes it a popular choice for developing interactive data visualization and machine learning applications.

The main reasons for choosing Streamlit were:

- Simplicity and ease of use
- Rapid prototyping
- Built-in widgets and components
- Automatic layout optimization

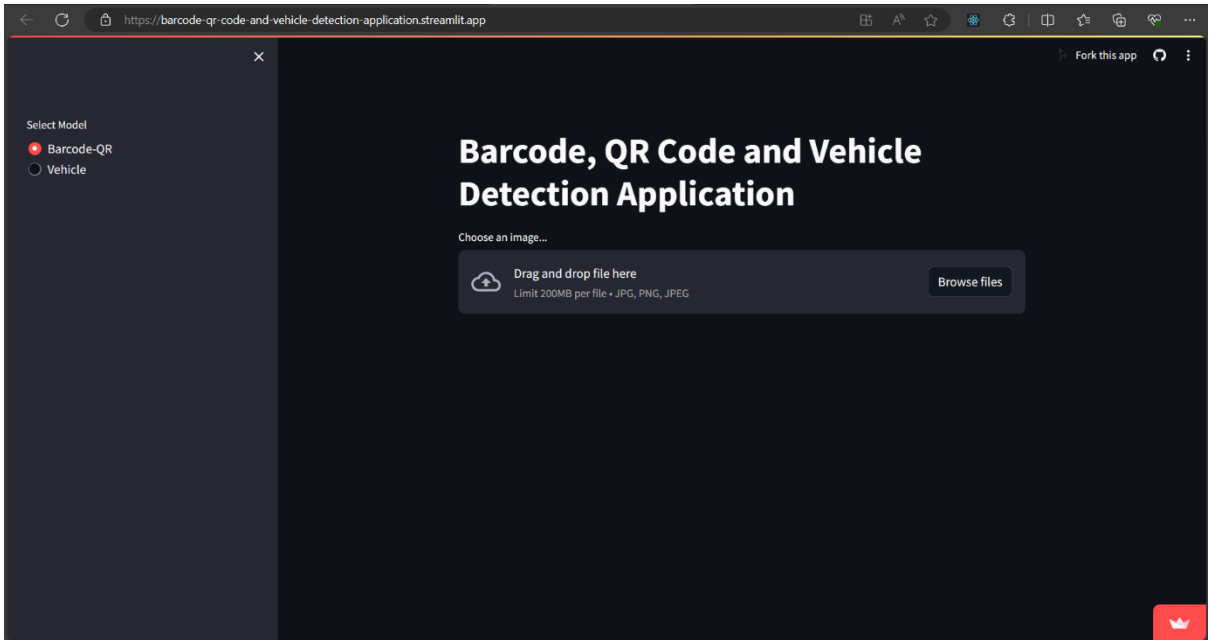

Fig. 3.3 Front-end Interface of web-application

CHAPTER 4

IMPLEMENTATION

4.1 Implementation

During implementation, we first tried to make a YOLOv8 model on default parameters to checked how the model performs. We used YOLOv8 pre-trained model on Barcode and QR code dataset to check the results on default values.

```
from ultralytics import YOLO
import torch

# Initialize the YOLO model
model = YOLO("yolov8n.pt")

# Train the model using the provided data and for the specified number of epochs, using the GPU on your
if torch.cuda.is_available():
  device = "cuda"
else:
  device = "cpu"

model.train(data="/kaggle/input/barcode-detection/data.yaml", epochs=20, device=[0,1], visualize=True)
#model.train(data="/kaggle/input/barcode-detection/data.yaml", epochs=3, device=[0, 1])
```

```
from ultralytics import YOLO

# Load an official model
model = YOLO('yolov8n.pt')

# Or load a custom model
model = YOLO('/kaggle/working/runs/detect/train/weights/best.pt')

# Validate the model
metrics = model.val()
metrics.box.map
metrics.box.map50
metrics.box.map75
metrics.box.maps
```

Fig. 4.1 Implementation using default parameters

Default hyperparameters taken are:

- Input image size (default: 640×640)
- Number of images per batch (default: 16)
- Optimizer- AdamW
- Learning Rate - 0.000714
- Momentum - 0.937

Based on the results we further used hyper-parameter tuning on the model to make it robust and finetuned it for optimal results. Hyperparameter tuning is an iterative process aimed at finding the optimal configuration for the model to achieve the best possible performance on your specific dataset. These configurations are settings that control the learning process of the model but are set before the training begins and remain constant throughout.

- lr = 0.01
- epochs - 10 and 20
- batch_size = 16
- img_size = 416×416

```
from ultralytics import YOLO
import torch

# Initialize the YOLO model
model = YOLO()
model = YOLO("yolov8x.pt")

# Train the model using the provided data and for the specified number of epochs, using the GPU on your Mac
if torch.cuda.is_available():
  device = "cuda"
else:
  device = "cpu"

model.train(data="/kaggle/input/top-view-vehicle-detection-image-dataset/Vehicle_Detection_Image_Dataset/data.yaml",
            optimizer='AdamW',epochs=20, lr0 = 0.01,
            project = 'Top view vehicle Detection',
            verbose = True, device=device)
```

```
from ultralytics import YOLO

# Load an official model
model = YOLO('yolov8n.pt')

# Or load a custom model
model = YOLO('/kaggle/working/runs/detect/train/weights/best.pt')

# Validate the model
metrics = model.val()
metrics.box.map
metrics.box.map50
metrics.box.map75
metrics.box.maps
```

Fig. 4.2 Implementation using hyperparameter tuning

After the completion of model training using hyperparameter tuning, we started further expanding our project. So, we integrated our models in a web-based application which we created using Streamlit, which a Python framework for build web-based applications. Below is the source code of our web-app.

```
import streamlit as st
from PIL import Image
from ultralytics import YOLO
import dill

# Load YOLO models
vehicle_model = YOLO('vehicle.pt')
barcode_model = YOLO('barcode.pt')

# Function to perform inference based on model selection
def perform_inference(model, uploaded_image):
```

```
    image = Image.open(uploaded_image)
    results = model(image)
    return results

# Streamlit app
def main():
    st.title("Barcode, QR Code and Vehicle Detection Application")

    # Sidebar - Model selection
    selected_model = st.sidebar.radio("Select Model", ("Barcode-QR", "Vehicle"))

    # Upload image
    uploaded_image = st.file_uploader("Choose an image...", type=["jpg", "png", "jpeg"])

    if uploaded_image is not None:
        st.image(uploaded_image, caption="Uploaded Image", use_column_width=True)

        # Run inference based on selected model
        if selected_model == "Barcode-QR":
            model = barcode_model
        else:
            model = vehicle_model

        if st.button("Run Detection"):
            with st.spinner('Running inference...'):
                try:
                    # Perform inference
                    results = perform_inference(model, uploaded_image)

                    # Visualize and save results
                    for i, r in enumerate(results):
```

```
                st.subheader(f"Result")

                # Plot results image
                im_bgr = r.plot()  # BGR-order numpy array
                im_rgb = Image.fromarray(im_bgr[..., ::-1])  # RGB-order PIL image
                st.image(im_rgb, caption=f"Result", use_column_width=True)

                # Save results to disk
                #filename = f'results_{selected_model}_{i}.jpg'
                #r.save(filename=filename)
                #st.markdown(f"Download [**{filename}**](/{filename})")

            except Exception as e:
                st.error(f"Error: {e}")

if __name__ == "__main__":
    main()
```

After the hyperparameter tuning was completed, we started developing a real time web-application which uses the pre-trained models we made in backend to detect the unseen user input.

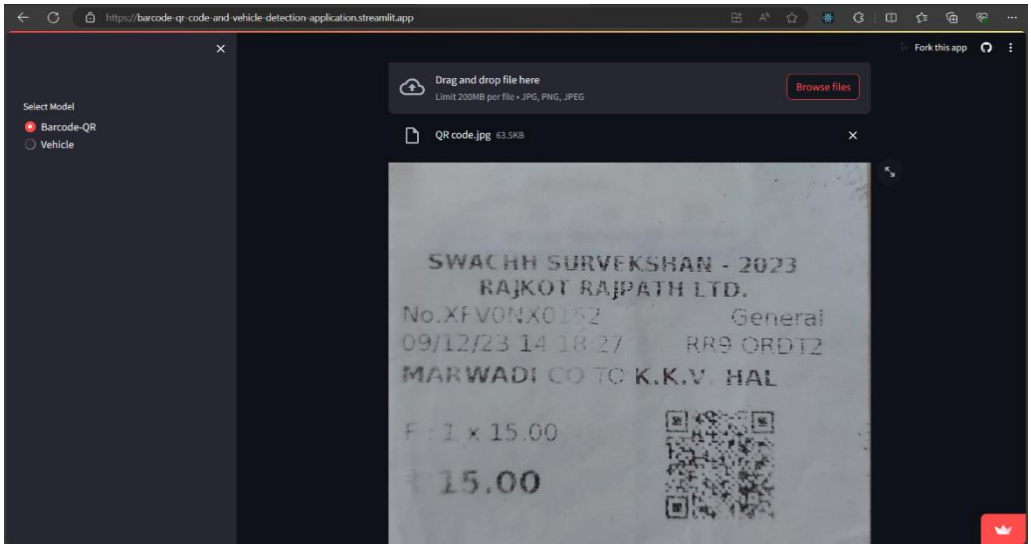

Fig. 4.3 Implementation of our web-application (QR code detection)

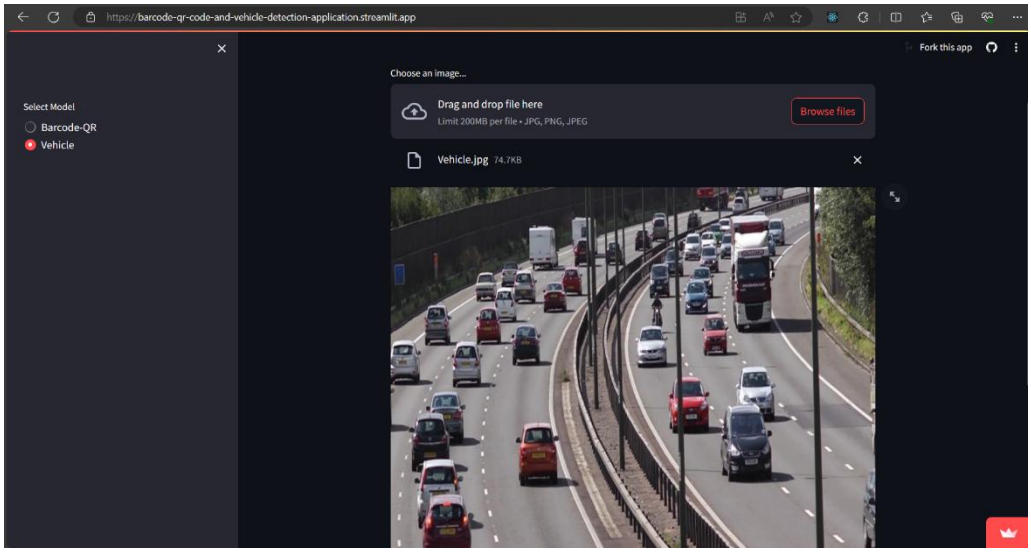

Fig. 4.4 Implementation of our web-application (Vehicle detection)

4.2 Results

In YOLOv8 object detection, result analysis is the process of interpreting the information the model outputs after it examines an image or video. It is basically figuring out what the model "saw" and how confident it is in its findings. Here is a breakdown of what we analyse:

- **Bounding Boxes:** These are the rectangular boxes YOLOv8 draws around detected objects. By examining these boxes, we understand the location and size of the objects within the image.
- **Class Labels:** YOLOv8 assigns a label to each object. Analysing these labels tells us what kind of objects the model identified.
- **Confidence Scores:** These scores represent how certain the model is about its detections. Scores closer to 1 indicate high confidence, while lower scores mean the model is less sure.

A. Results Analysis of Barcode and QR code detection

- Graphs – 10 epochs

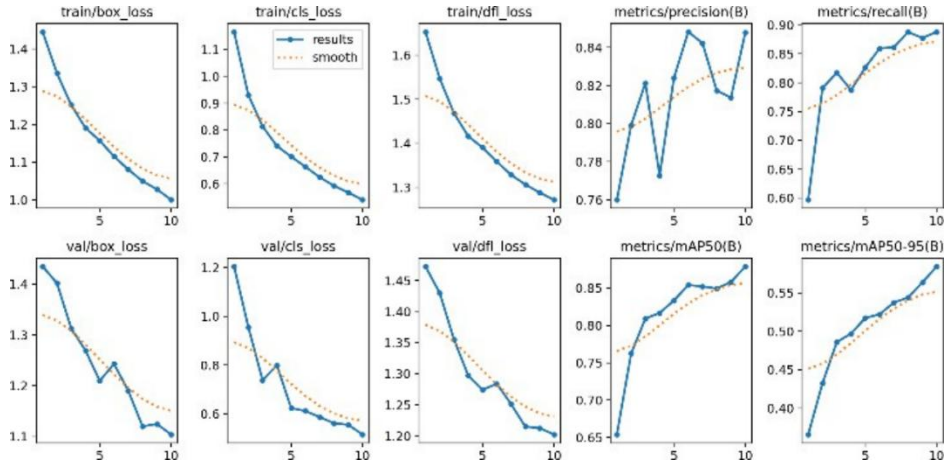

Fig. 4.5 Result Analysis for YOLOv8n

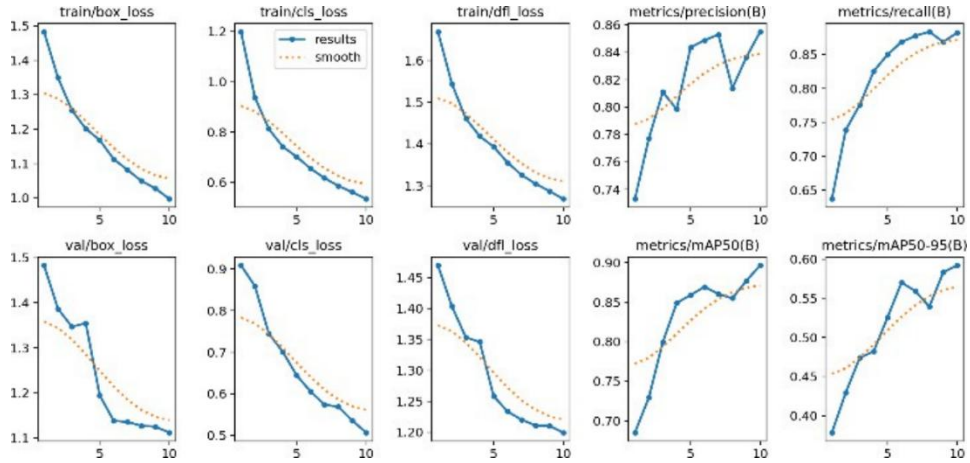

Fig. 4.6 Result Analysis for YOLOv8s

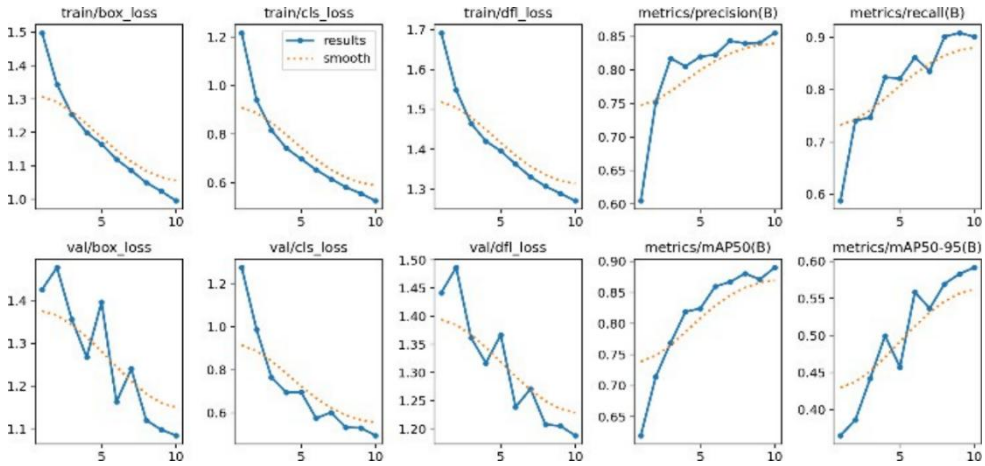

Fig. 4.7 Result Analysis for YOLOv8m

- Confusion Matrix – 10 epochs

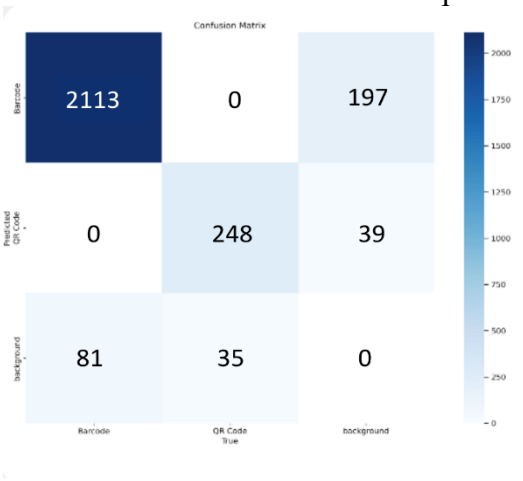

Fig. 4.8 Confusion Matrix YOLOv8n

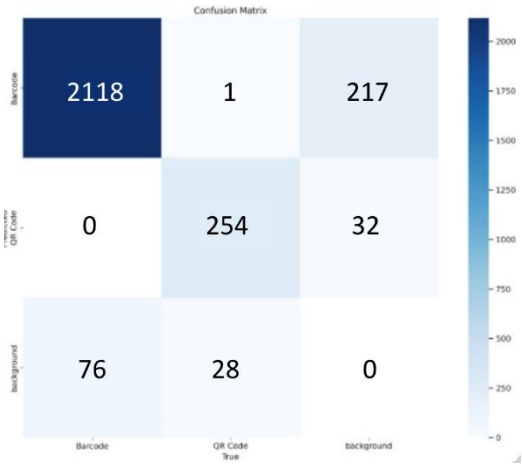

Fig. 4.9 Confusion Matrix YOLOv8s

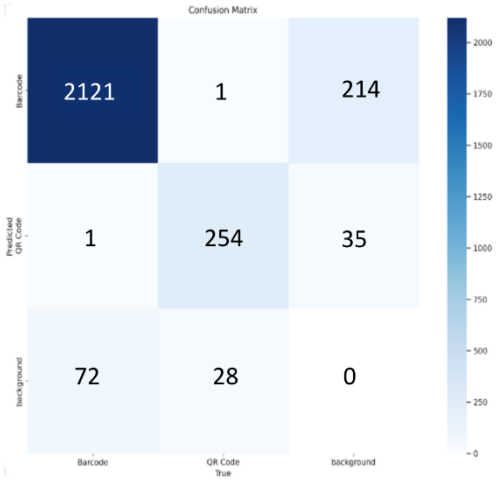

Fig. 4.10 Confusion Matrix YOLOv8m

B. Result Analysis of Top View Vehicle Detection

- Graphs – 10 epochs

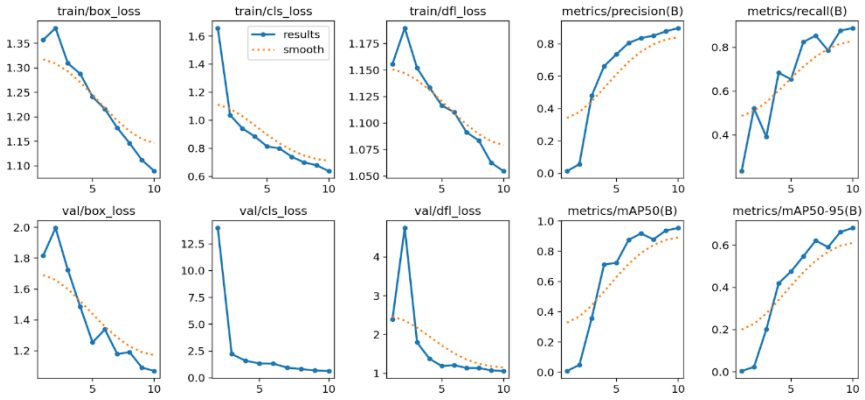

Fig. 4.11 Result Analysis for YOLOv8n

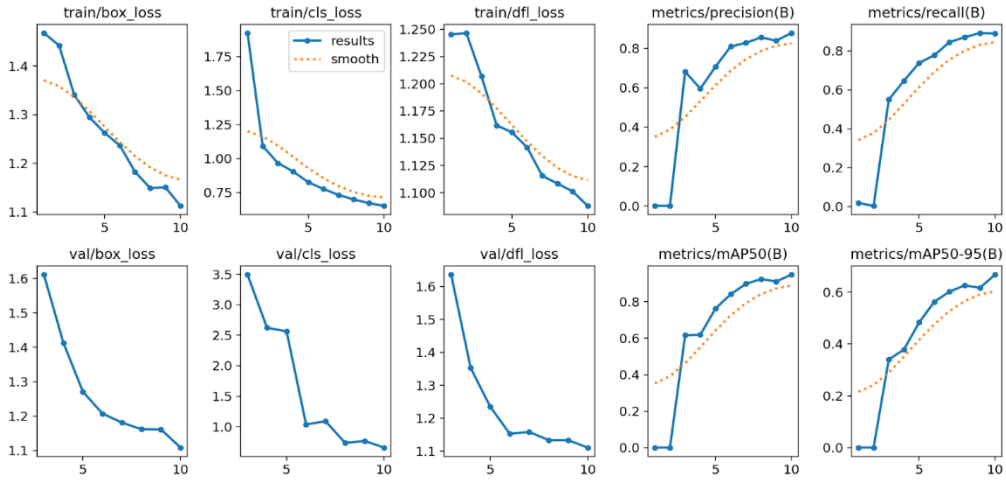

Fig. 4.12 Result Analysis for YOLOv8s

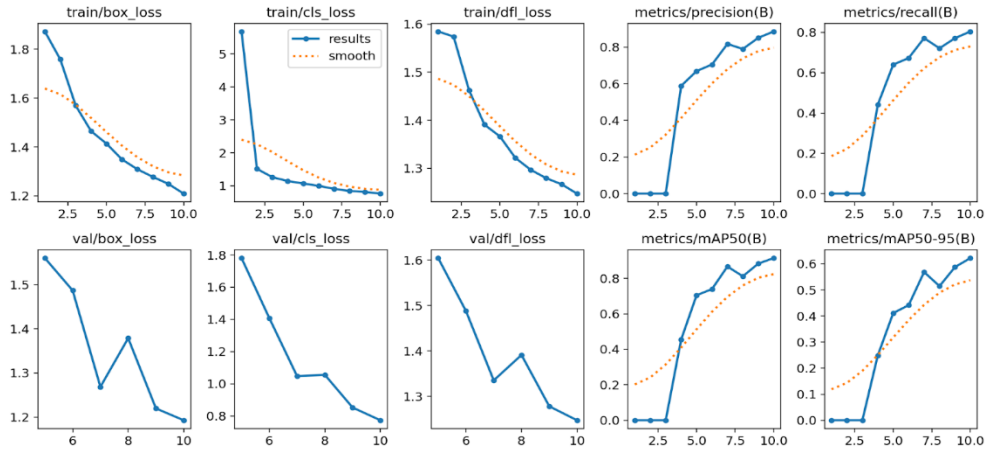

Fig. 4.13 Result Analysis for YOLOv8m

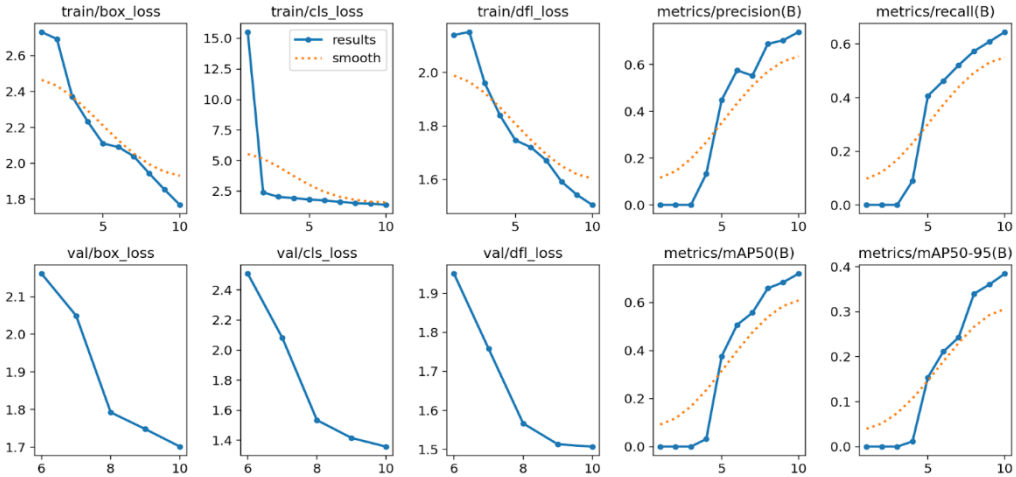

Fig. 4.14 Result Analysis for YOLOv8l

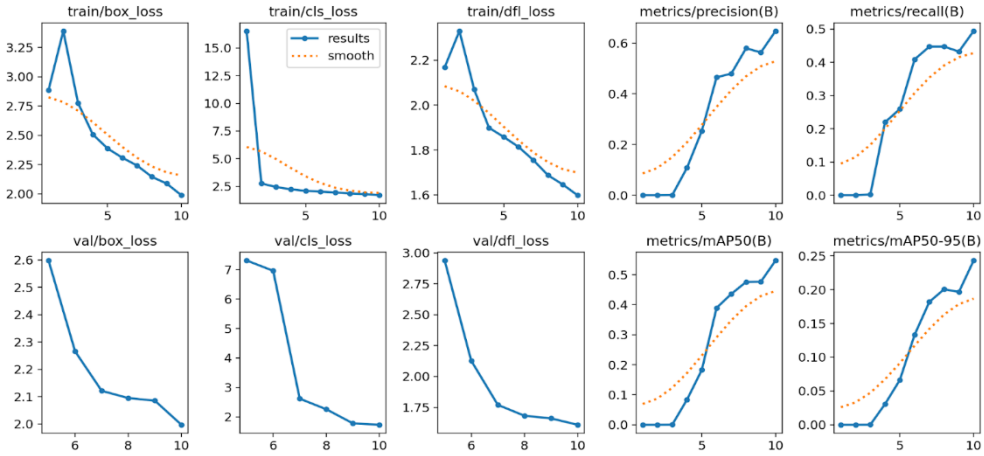

Fig. 4.15 Result Analysis for YOLOv8x

- Graphs – 20 epochs

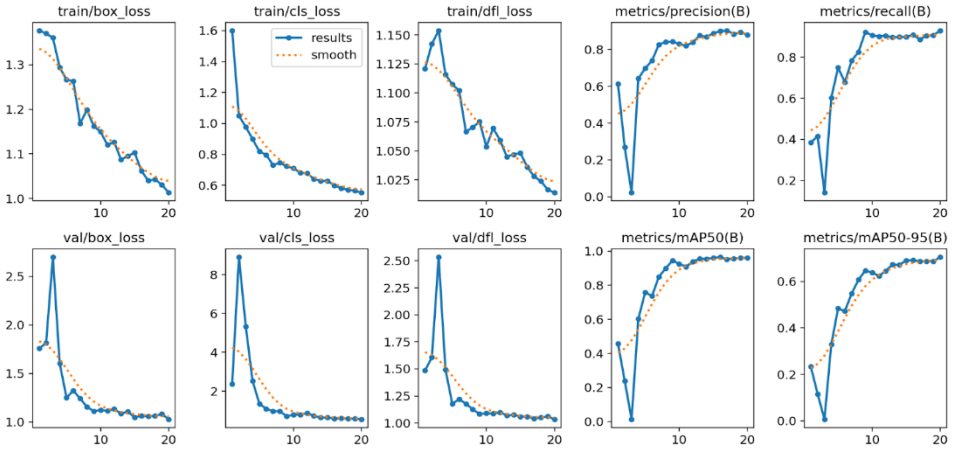

Fig. 4.16 Result Analysis for YOLOv8n

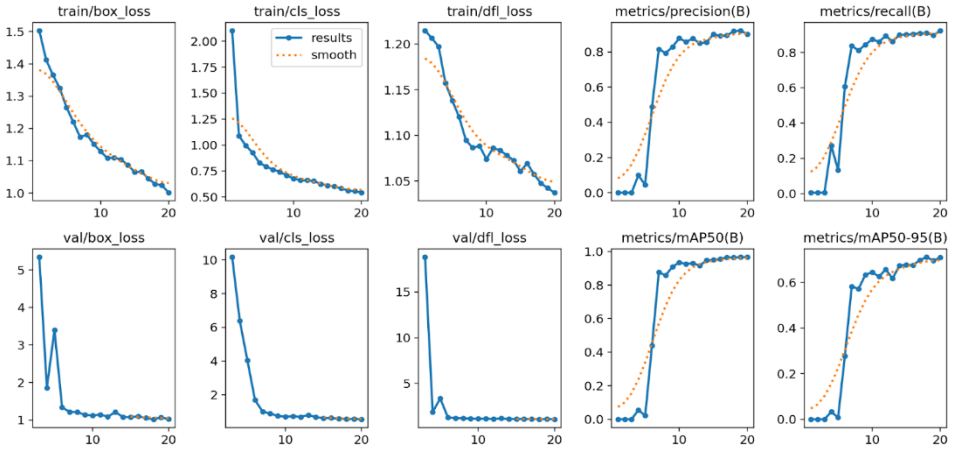

Fig. 4.17 Result Analysis for YOLOv8s

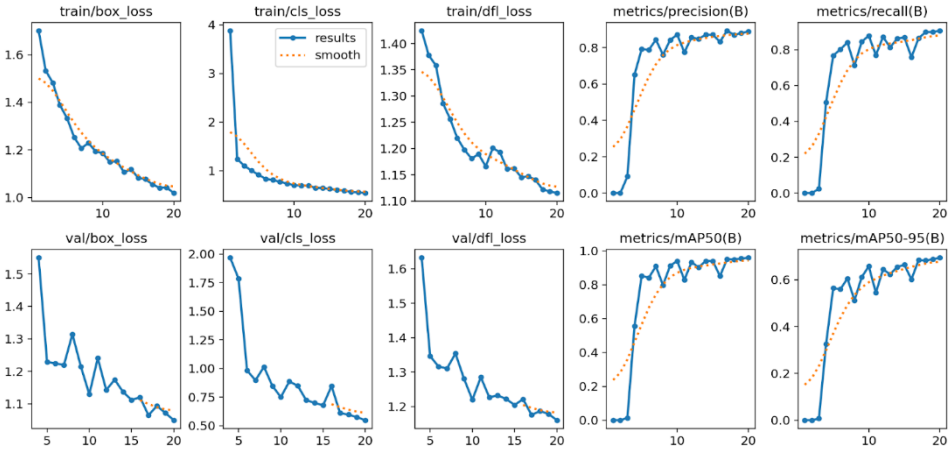

Fig. 4.18 Result Analysis for YOLOv8m

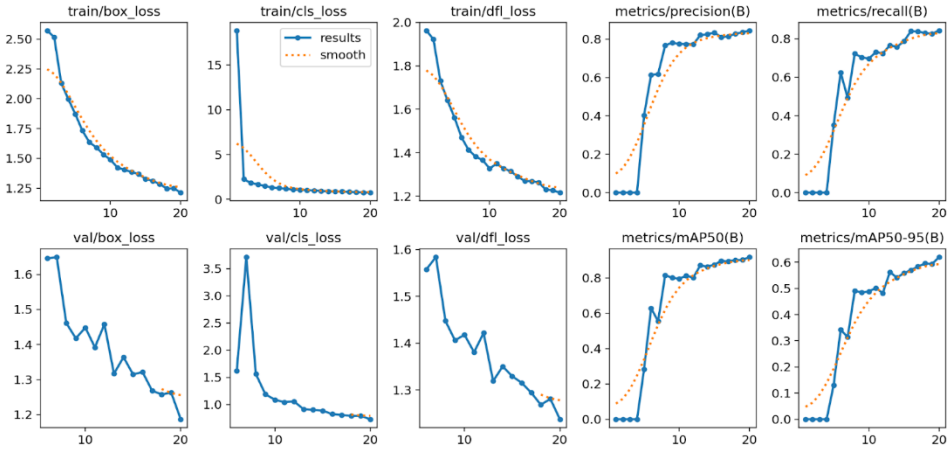

Fig. 4.19 Result Analysis for YOLOv8l

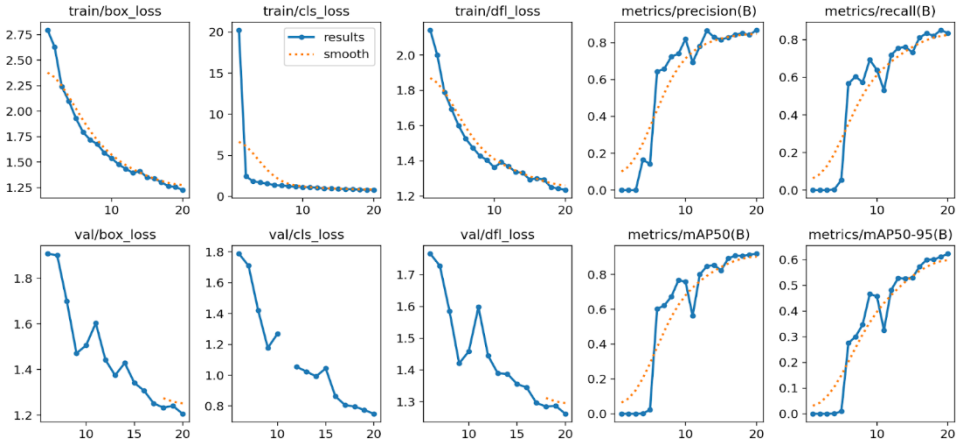

Fig. 4.20 Result Analysis for YOLOv8x

- Confusion Matrix – 10 epochs

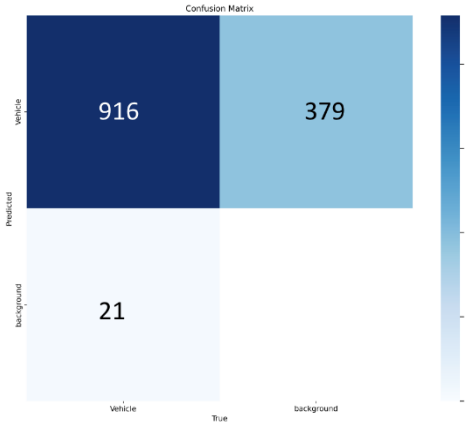

Fig. 4.21 Confusion Matrix YOLOv8n

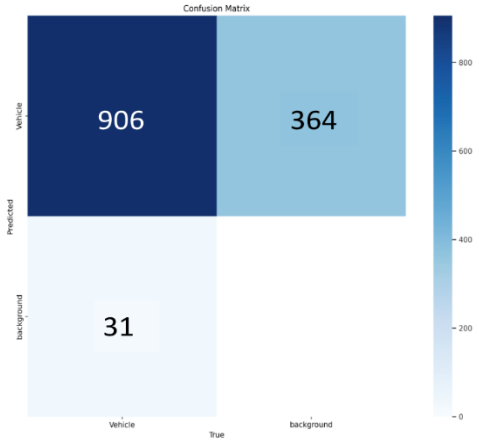

Fig. 4.22 Confusion Matrix YOLOv8s

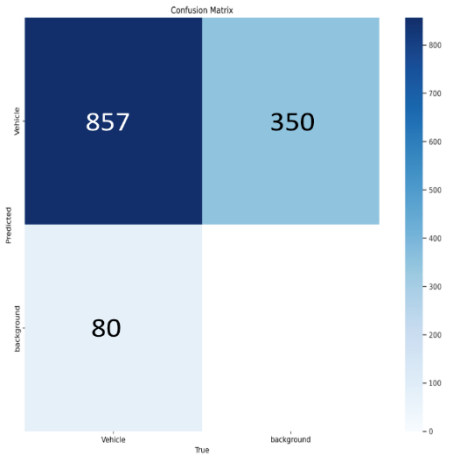

Fig. 4.23 Confusion Matrix YOLOv8m

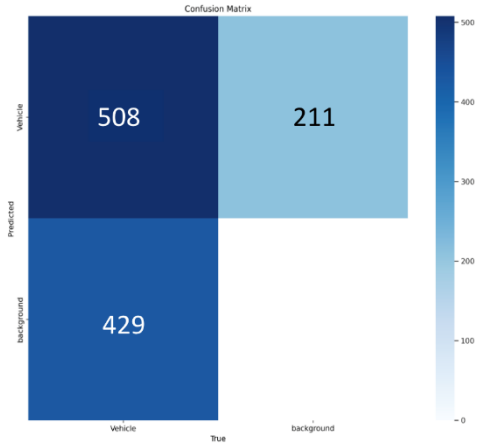

Fig. 4.24 Confusion Matrix YOLOv8l

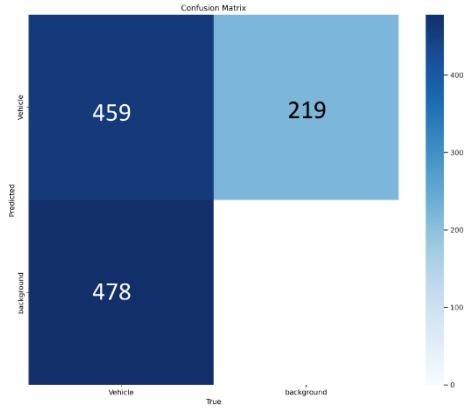

Fig. 4.25 Confusion Matrix YOLOv8x

- Confusion Matrix – 20 epochs

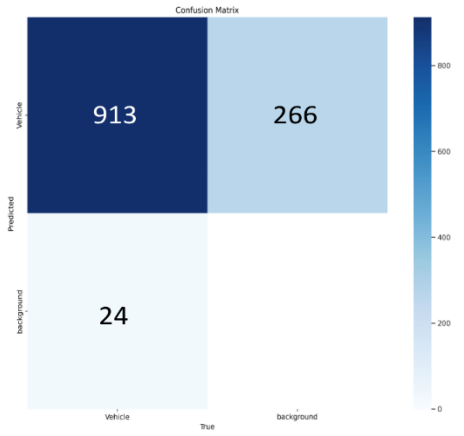

Fig. 4.26 Confusion Matrix YOLOv8n

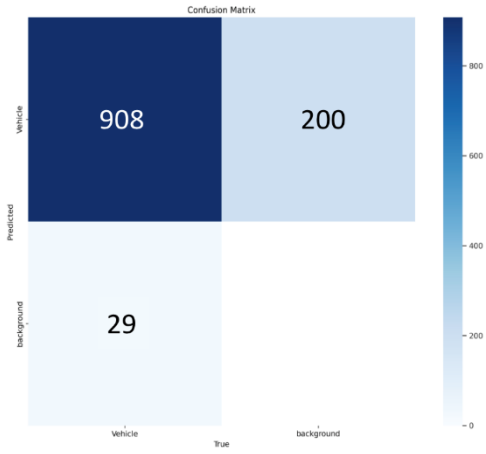

Fig. 4.27 Confusion Matrix YOLOv8s

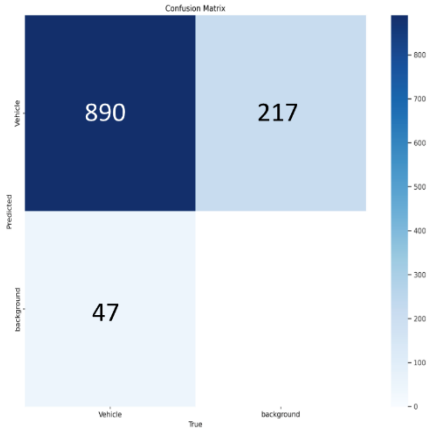

Fig. 4.28 Confusion Matrix YOLOv8m

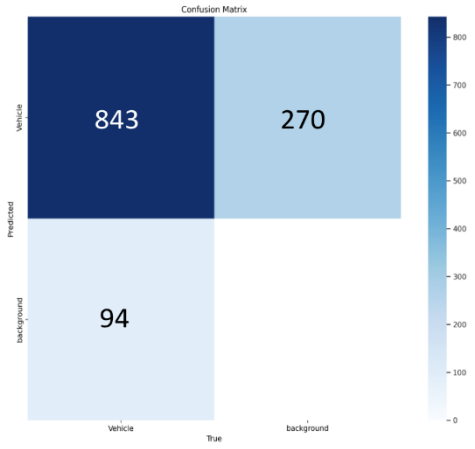

Fig. 4.29 Confusion Matrix YOLOv8l

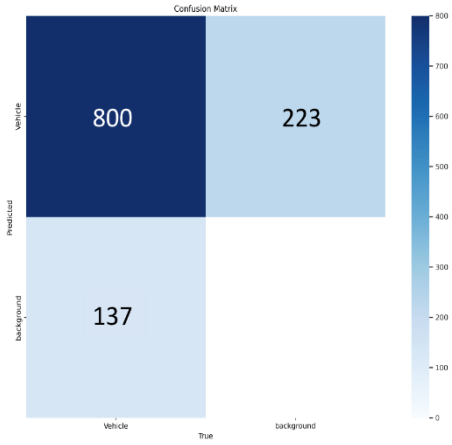

Fig. 4.30 Confusion Matrix YOLOv8x

4.3 Output

Finally, after the model training, we feed the model with the images that model has not seen during its training. Based on model's prediction and confidence scores, we check which model is best suited for application. After finding which model is best suited, we use those models in our web application for accurate real-time detection.

Fig. 4.31 Detection of Barcode on unseen images

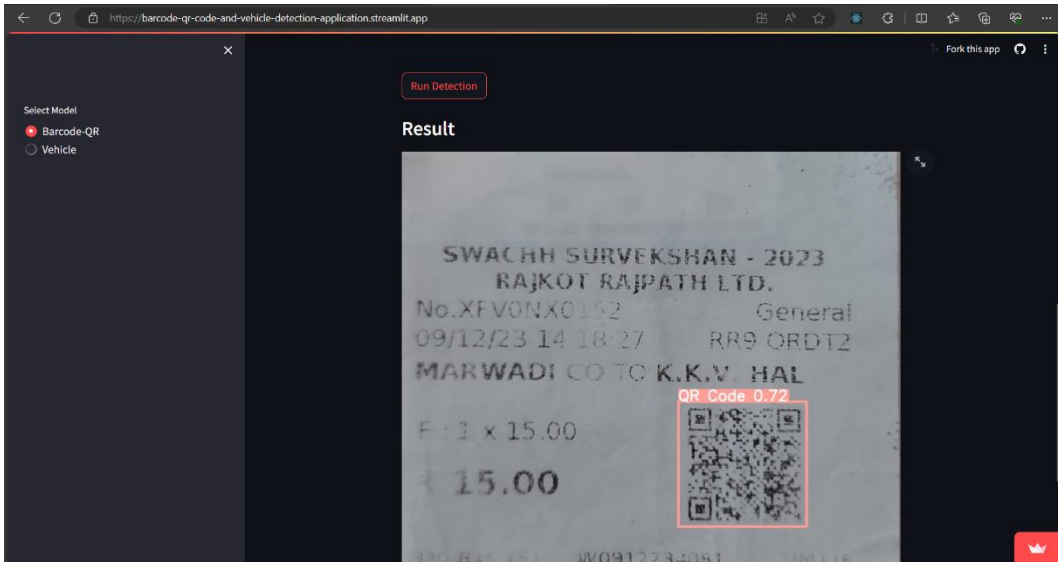

Fig. 4.32 QR code live detection in website

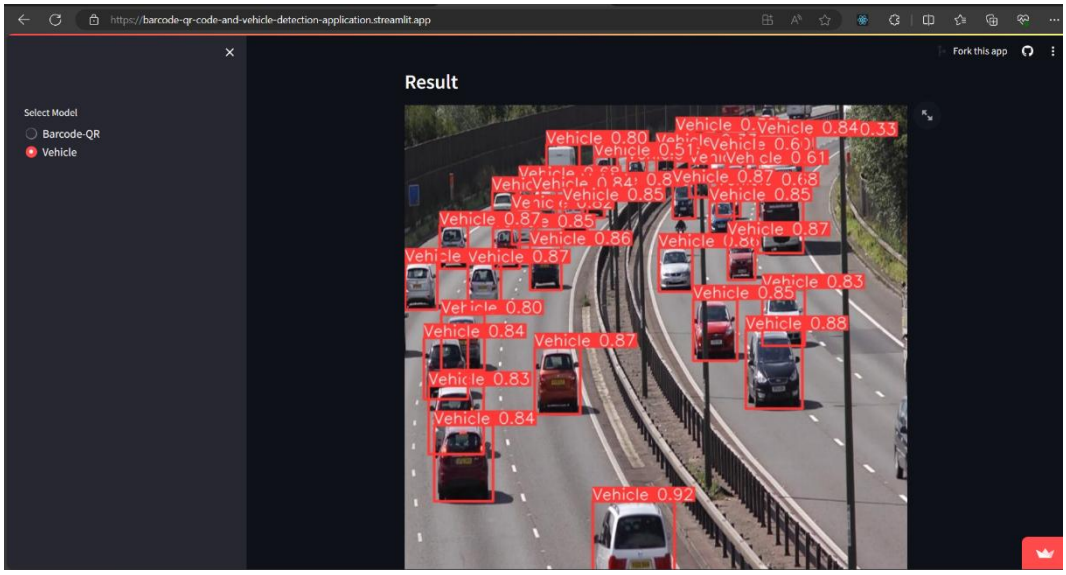

Fig. 4.33 Live vehicle detection in website

CHAPTER 5

TESTING

5.1 Result Analysis

Based on the all the different model training and on various scenarios (10 epochs and 20 epochs), we obtained the result as mentioned in Table 5.1, Table 5.2, Table 5.3

Table 5.1 Barcode and QR code detection result table

Models	Recall	Accuracy	Precision	F1 Score	mAP		Train Loss			Validation Loss		
					mAP 50	mAP 50-95	cls	dfl	box	cls	dfl	box
YOLOv8n	0.89	0.89	0.85	0.87	0.88	0.59	0.54	1.01	1.27	0.52	1.20	1.10
YOLOv8s	0.88	0.97	0.85	0.87	0.90	0.59	0.53	1.27	1.00	0.51	1.20	1.11
YOLOv8m	0.90	0.94	0.86	0.88	0.89	0.59	0.53	1.00	1.27	0.50	1.19	1.09

Table 5.2 Top View Vehicle Detection result table (10 epochs)

Models	Recall	Accuracy	Precision	F1 Score	mAP		Train Loss			Validation Loss		
					mAP 50	mAP 50-95	cls	dfl	box	cls	dfl	box
YOLOv8n	0.89	0.66	0.89	0.81	0.95	0.68	0.63	1.05	1.09	0.64	1.06	1.07
YOLOv8s	0.89	0.51	0.87	0.66	0.94	0.66	0.65	1.09	1.11	0.65	1.11	1.11
YOLOv8m	0.80	0.73	0.88	0.80	0.91	0.62	0.76	1.25	1.21	0.77	1.25	1.19
YOLOv8l	0.54	0.44	0.69	0.62	0.59	0.30	1.37	1.50	1.76	1.35	1.50	1.70
YOLOv8x	0.49	0.47	0.64	0.64	0.54	0.24	1.70	1.60	1.99	1.73	1.61	1.99

Table 5.3 Top View Vehicle Detection result table (20 epochs)

Models	Recall	Accuracy	Precision	F1 Score	mAP		Train Loss			Validation Loss		
					mAP 50	mAP 50-95	cls	dfl	box	cls	dfl	box
YOLOv8n	0.93	0.77	0.87	0.80	0.96	0.70	0.55	1.01	1.01	0.55	1.03	1.02
YOLOv8s	0.91	0.79	0.91	0.78	0.97	0.71	0.54	1.03	1.00	0.54	1.06	1.02
YOLOv8m	0.90	0.95	0.88	0.95	0.96	0.69	0.54	1.11	1.02	0.54	1.16	1.05
YOLOv8l	0.84	0.95	0.84	0.95	0.92	0.62	0.75	1.21	1.21	0.73	1.23	1.18
YOLOv8x	0.80	0.71	0.85	0.76	0.86	0.59	0.77	1.23	1.22	0.75	1.26	1.20

CHAPTER 6

CONCLUSION & OUTCOMES

6.1 Overall Analysis of Project Viabilities

As provided in the table (Table 5.1) the performance of the three YOLOv8 variants (YOLOv8n, YOLOv8s, and YOLOv8m) for Barcode and QR code detection and five YOLOv8 variants (YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x) for Top View vehicle detection in object detection tasks was evaluated based on several key metrics.

For Barcode and QR code detection, YOLOv8m achieved the highest recall (0.90) and accuracy (0.94) among the models, indicating its effectiveness in correctly identifying objects. YOLOv8s followed closely with a recall of 0.88 and an impressive accuracy of 0.97. YOLOv8n exhibited slightly lower recall (0.88) and accuracy (0.89) compared to the other two models. YOLOv8s demonstrated the highest precision (0.8541), followed by YOLOv8m (0.85) and YOLOv8n (0.84). However, when considering the balance between precision and recall, YOLOv8m achieved the highest F1 score (0.87), indicating its ability to maintain a balance between precision and recall. YOLOv8s and YOLOv8n had F1 scores of 0.86 and 0.86, respectively. YOLOv8s achieved the highest mAP score (0.89), followed closely by YOLOv8m (0.89) and YOLOv8n (0.87). The mAP50 and mAP50-95 scores were also highest for YOLOv8s, indicating its effectiveness in detecting objects across different thresholds. YOLOv8m exhibited the lowest training loss (0.99) and validation loss (1.27) among the models, suggesting its robustness and ability to generalize well to unseen data. YOLOv8s had slightly higher losses (training loss: 1.26, validation loss: 0.99), while YOLOv8n demonstrated the highest losses (training loss: 1.01, validation loss: 1.27).

For Top View Vehicle Detection, training was done in two ways: Firstly, 10 epochs for each five YOLOv8 variants were evaluated, then 20 epochs were set for training of all variants. For 10 epochs, YOLOv8n outperforms with strong recall (0.89), mAP (0.95), and F1 score (0.81) in 10 epochs, but its training loss implies potential for improvement. YOLOv8m strikes a compromise between accuracy (0.88) and recall (0.80) while minimizing training loss (0.62). YOLOv8s promotes speed, but YOLOv8l and

YOLOv8x provide high accuracy at the cost of processing time and resource needs. Overall, YOLOv8n is better for high recall, whereas YOLOv8m finds a compromise between accuracy, recall, and training efficiency. For 20 epochs, YOLOv8n excels with a recall of 0.93, a great mAP (0.96), and an F1 score of 0.80. It strikes a good mix between accuracy (0.87) and recall, resulting in solid performance. YOLOv8s also offers an appealing balance, with accuracy and recall both at 0.91 and the lowest training loss (0.70) of any model, suggesting fast learning. YOLOv8m, while equivalent to YOLOv8n in accuracy (0.88), has a little poorer recall (0.90) and a larger training loss (0.69). YOLOv8l and YOLOv8x exhibit inferior performance across measures and are less efficient learners. In summary, YOLOv8n excels at decreasing missed detections, but YOLOv8s strikes a balance between precision, recall, and training efficiency, making it an outstanding pick. Consider YOLOv8s for real-time processing, whereas YOLOv8n may be better suited to resource-constrained situations.

6.2 Problems encountered and possible solutions

During, the training of models for Top view vehicle detection, the dataset size appeared to be small, resulting in machine learning with fewer images for training. This led to inefficient work and decreased accuracy during testing. Furthermore, as the models grow in size from large to extra-large, with numerous additional parameters (in several millions) and the dataset size remaining constant, the accuracy does not necessarily improve. The restricted dataset size tends to cause accuracy to peak at small to medium models.

6.3 Project Outcomes

In conclusion, this project evaluates YOLOv8's barcode and QR code detection along with detection of vehicles from top view perspective in images using a dataset that contains both categories. Experimentation shows that sophisticated YOLOv8 family designs improve detection. After hyperparameter fine-tuning and based on the results, YOLOv8m is the most effective at detecting barcodes and QR codes, with the highest recall (0.90) and accuracy (0.94), as well as a noteworthy F1 score (0.87). YOLOv8s follows closely, with high accuracy (0.8541) and mAP score (0.89), making it acceptable for a wide range of threshold values. Meanwhile, for Top View Vehicle Detection, YOLOv8n outperforms with a strong recall (0.93), mAP (0.96), and F1 score (0.80), demonstrating consistent performance even after 20 epochs. However,

YOLOv8s appears as an appealing option for balancing accuracy, recall, and training efficiency, with precision and recall both at 0.91 and the lowest training loss (0.70). As a result, although YOLOv8n excels at decreasing missed detections, YOLOv8s provides a comprehensive solution for a wide range of detection tasks, particularly in real-time applications.

6.4 Future Scope

By leveraging the insights gained from evaluating different hyperparameters and model configurations, we ensured the app's efficiency in accurately identifying objects of interest. This project not only highlights the versatility of YOLOv8 in different detection tasks but also demonstrates the practical application of deep learning models in real-world scenarios through intuitive web interfaces. Experimenting on an even larger dataset, using hybrid algorithms and optimizing the model’s accuracy further through algorithmic and hardware upgrades remains our future goals

References

[1] H. Rodríguez-Rangel, L. A. Morales-Rosales, R. Imperial-Rojo, M. A. Roman-Garay, G. E. Peralta-Peñuñuri, and M. Lobato-Báez, "Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO," Appl. Sci., vol. 12, no. 6, pp. 1–20, 2022, doi: 10.3390/app12062907.

[2] S. Huang, Y. He, and X. A. Chen, "M-YOLO: A Nighttime Vehicle Detection Method Combining Mobilenet v2 and YOLO v3," J. Phys. Conf. Ser., vol. 1883, no. 1, 2021, doi: 10.1088/1742-6596/1883/1/012094.

[3] W. Li, "Vehicle detection in foggy weather based on an enhanced YOLO method," J. Phys. Conf. Ser., vol. 2284, no. 1, 2022, doi: 10.1088/1742-6596/2284/1/012015.

[4] H. V. Koay, J. H. Chuah, C. O. Chow, Y. L. Chang, and K. K. Yong, "Yolo-rtuav: Towards real-time vehicle detection through aerial images with low-cost edge devices," Remote Sens., vol. 13, no. 21, pp. 1–26, 2021, doi: 10.3390/rs13214196.

[5] J. Sang et al., "An improved YOLOv2 for vehicle detection," Sensors (Switzerland), vol. 18, no. 12, 2018, doi: 10.3390/s18124272.

[6] J. Zhao et al., "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4," IEEE Access, vol. 10, no. i, pp. 8590–8603, 2022, doi: 10.1109/ACCESS.2022.3143365.

[7] T. Han, T. Cao, Y. Zheng, L. Chen, Y. Wang, and B. Fu, "Improving the Detection and Positioning of Camouflaged Objects in YOLOv8," Electron., vol. 12, no. 20, pp. 1–17, 2023, doi: 10.3390/electronics12204213.

[8] Mingda Wang, Luyuan Ren. SSB-YOLO: A vehicle object detection algorithm based on improved YOLOv8, 13 December 2023, PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-3743453/v1]

[9] C. Kwan et al., "Real-time and deep learning based vehicle detection and classification using pixel-wise code exposure measurements," Electron., vol. 9, no. 6, pp. 1–21, 2020, doi: 10.3390/electronics9061014.

[10] S. Tamang, B. Sen, A. Pradhan, K. Sharma, and V. K. Singh, "International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Enhancing COVID-19 Safety: Exploring YOLOv8 Object Detection for Accurate Face Mask Classification," Orig. Res. Pap. Int. J. Intell. Syst. Appl. Eng. IJISAE, vol. 2023, no. 2, pp. 892–897, 2023, [Online]. Available: www.ijisae.org

[11] A. Inui et al., "Detection of Elbow OCD in the Ultrasound Image by Artificial Intelligence Using YOLOv8," Appl. Sci., vol. 13, no. 13, 2023, doi: 10.3390/app13137623.

[12] L. J. Zhang, J. J. Fang, Y. X. Liu, H. Feng Le, Z. Q. Rao and J. X. Zhao, "CR-YOLOv8: Multiscale Object Detection in Traffic Sign Images," in IEEE Access, vol. 12, pp. 219-228, 2024, doi: 10.1109/ACCESS.2023.3347352.

[13] B. Xiao, M. Nguyen, and W. Q. Yan, "Fruit ripeness identification using YOLOv8 model," Multimed. Tools Appl., no. February, 2023, doi: 10.1007/s11042-023-16570-9.

[14] Mingda Wang, Luyuan Ren. SSB-YOLO: A vehicle object detection algorithm based on improved YOLOv8, 13 December 2023, PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-3743453/v1]

[15] T. Han, T. Cao, Y. Zheng, L. Chen, Y. Wang, and B. Fu, "Improving the Detection and Positioning of Camouflaged Objects in YOLOv8," Electron., vol. 12, no. 20, pp. 1–17, 2023, doi: 10.3390/electronics12204213.

[16] M. Sportelli et al., "Evaluation of YOLO Object Detectors for Weed Detection in Different Turfgrass Scenarios," Appl. Sci., vol. 13, no. 14, 2023, doi: 10.3390/app13148502.

[17] Q. Liu, Y. Liu, and D. Lin, "Revolutionizing Target Detection in Intelligent Traffic Systems: YOLOv8-SnakeVision," Electron., vol. 12, no. 24, 2023, doi: 10.3390/electronics12244970.

[18] J. Jönsson Hyberg and A. Sjöberg, 'Investigation regarding the Performance of YOLOv8 in Pedestrian Detection', Dissertation, 2023.

[19] L. Han, C. Ma, Y. Liu, J. Jia, and J. Sun, "SC-YOLOv8: A Security Check Model for the Inspection of Prohibited Items in X-ray Images," Electron., vol. 12, no. 20, 2023, doi: 10.3390/electronicsmu12204208.

[20] G. Yu, T. Wang, G. Guo, and H. Liu, "SFHG-YOLO: A Simple Real-Time Small-Object-Detection Method for Estimating Pineapple Yield from Unmanned Aerial Vehicles," Sensors, vol. 23, no. 22, 2023, doi: 10.3390/s23229242.

[21] N. P. Motwani and S. S, "Human Activities Detection using DeepLearning Technique-YOLOv8," ITM Web Conf., vol. 56, p. 03003, 2023, doi: 10.1051/itmconf/20235603003.

[22] S. Khalid, H. M. Oqaibi, M. Aqib, and Y. Hafeez, "Small Pests Detection in Field Crops Using Deep Learning Object Detection," Sustain., vol. 15, no. 8, pp. 1–19, 2023, doi: 10.3390/su15086815.

[23] D. Sudharson et al., "Proactive Headcount and Suspicious Activity Detection using Proactive Headcount and YOLOv8 Suspicious Activity Detection using," Procedia Comput. Sci., vol. 230, no. 2023, pp. 61–69, 2024, doi: 10.1016/j.procs.2023.12.061.

[24] SEVİ, M., & AYDIN, İ. (2023). Detection of Foreign Objects Around the Railway Line with YOLOv8. Computer Science, IDAP-2023 : International Artificial Intelligence and Data Processing Symposium(IDAP-2023), 19-23. https://doi.org/10.53070/bbd.1346317.

Marwadi University
Marwadi Chandarana Group

Department of Computer Engineering

Marwadi University

Academic Year : 2023-24

Semester : 6

Mini Project (01CE0609)

Project Report Diary

Team ID : MU_6CE_089

Project Title : Exploring Yolo Objection using Deep Learning Techniques.

Sr. No.	Student Full Name	Student En. No.	Class
1	Heli. Hathi	92100103341	6TC2
2	Het. Buch	92100103196	6TC2
3			

Guide Name:

Weekly Mini Project Report Diary – JANUARY

Week	Project Activity	Guide's Comments/Suggestions/Remarks	Date	Guide Signature
1	(i) Try yolov8 on any dataset and check the performance (ii) Find 5 recent research papers.	Good work Discussed hybrid algorithms.	5/1/24	5/1/24
2	(i) Fix errors (ii) Complete Literature Review (iii) Come with one algorithm.	Figured datasets & algorithms. Implement CNN.	12/1/24	12/1/24
3	(i) Execute / Implement yolov8 on 2 different datasets. (ii) Literature Review on 5 Research Papers.	Shown results of 2 datasets. Good. → try Nms alg, Hyperparameters & plan Custom dataset + add 5 more papers	19/1/24	19/1/24
4	(i) Literature Review on next 5 Research paper (ii) Complete PPt for review.	Done with 15 LR. PPT Completed Glass & Bar Code & QR Code dataset tested Try Hyperparameters	2/Feb/24	2/Feb/24

Weekly Mini Project Report Diary – FEBRUARY

Week	Project Activity	Guide's Comments/Suggestions/Remarks	Date	Guide Signature
1	Comparison on different models (nano, small and medium).	Evaluation metrics & graphs Checked. Good work.	9/2/24	
2	→ Write Research Paper (IEEE).	Written one IEEE Conference Paper on yolov8 → Start work on Glass dataset & perform Hyperparameter tuning.	16/2/24	
3	→ Complete comparison of nano and small model in glass detection dataset.	2 models Completed. Try other models.	23/2/24	
4	→ Complete medium, large and extra large model comparison in glass dataset.	Glass dataset results finalized. Develop front End & Connect BarCode & Glassmodel. ~~Streamlit / Django / Flask~~	1/3/24	

Weekly Mini Project Report Diary – MARCH

Week	Project Activity	Guide's Comments/Suggestions/Remarks	Date	Guide Signature
1	→ Perform Result Analysis on both dataset and complete ppt for 2 review.	Glass dataset is imbalance. try other dataset PPT done. Implement Front End.	7/3/24	
2	→ Compare all models in vehicle detection dataset and analyze its result.	Good.	19/3/24	
3	→Create Web application using Streamlit.	Deployment done. working fine. Start Journal work	22/3/24	
4	→ Start Journal work. → Presentation for final Review.	Started LR. Complete LR with 15 papers. Final PPT prepared. Start Report work.	29/3/24	

Weekly Mini Project Report Diary – APRIL

Week	Project Activity	Guide's Comments/Suggestions/Remarks	Date	Guide Signature
1	→ Complete ppt of final Review.	Done with final demo. Good work. 3 datasets compared with you.	5/4/24	Roy
2				
3				
4				

Marwadi University
Marwadi Chandarana Group

Mini Project (01CE0609)

Marwadi University

Faculty of Technology

Department of Computer Engineering

2023-24

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **EXPLORING YOLO OBJECT DETECTION USING DEEP LEARNING TECHNIQUES** has been carried out by **Heli Hathi** (92100103341) and **Het Buch** (92100103196) under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 6th Semester of Marwadi University, Rajkot during the academic year 2023-24.

Prof. Ravikumar R Natarajan
Assistant Professor
Internal Guide

Dr. Krunal Vaghela
Associate Professor
Head of the Department