2D Object Detection on KITTI Dataset

INTERNSHIP REPORT

Submitted by:

VELIDE SRI MANASWINI (CS22B2030)

Computer Science with major in Artificial Intelligence

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

July 26, 2024

CERTIFICATE

This is to certify that the report titled **2D Object Detection on Kitti Dataset (with modified YOLO v8 model)**, submitted by **Velide Sri Manaswini (CS22B2030)**, to the Indian Institute of Information Technology, Design and Manufacturing Kancheepuram, in partial fulfilment of requirements for Summer Internship 2024 is a bonafide record of the work done by him/her under my supervision.

Dr. Ram Prasad Padhy
Assistant Professor
Department of Computer Science and Engineering
IIITDM Kancheepuram, Chennai - 600 127

Place: Chennai

Date: July 26, 2024

2D Object Detection on Kitti Dataset

1 Summary of Internship

1.1 Motivation

The motivation for 2D object detection stems from its critical role in enabling machines to understand and interact with the visual world, which is fundamental for various applications across various domains. 2D object detection is its vital role in the development of self-driving cars. In autonomous driving, accurate object detection ensures the safety and efficiency of vehicles by identifying pedestrians, other vehicles, cyclists, traffic signals, and road signs and obstacles in real-time, 2D object detection systems enhance the safety and reliability of autonomous driving.

In the field of surveillance, it enhances security by enabling the automatic monitoring of environments and detecting unusual activities. Beyond self-driving cars, this technology also powers applications in surveillance, healthcare, robotics, and retail, making it an essential component in the advancement of intelligent systems and automation. The continuous advancements in deep learning and computational power have made 2D object detection increasingly accurate and efficient, driving its adoption and further research to overcome challenges such as occlusion, real-time processing, and small object detection.

1.2 Objectives

The primary objective of this report is to evaluate the performance and applicability of the YOLOv8 (You Only Look Once, Version 8) algorithm for 2D object detection tasks. Specifically, the goals are:

1.2.1 Implementation and Training:

To document the implementation process of YOLOv8, including data preparation, model training, and hyperparameter tuning. To provide insights and recommendations for optimizing YOLOv8 performance based on experimental results.

1.2.2 Performance Evaluation:

To measure the accuracy, precision, recall, and F1 score of modified YOLOv8 on a KITTI dataset. To evaluate the computational efficiency and resource requirements of YOLOv8 for real-time object detection applications. To compare the performance of modified YOLOv8 with versions of YOLOv8

1.2.3 Use Case Demonstrations:

To demonstrate the use of YOLOv8 in various applications such as autonomous driving, surveillance, and image-based quality control. To showcase real-time detection capabilities through practical examples and visualizations. By achieving these objectives, this report aims to provide a comprehensive understanding of the strengths, weaknesses, and practical considerations of using YOLOv8 for 2D object detection tasks.

2 Contribution

Object Detection : Object detection is a technique used in computer vision for the identification and localization of objects within an image or a video. Image Localization is the process of identifying the correct location of one or multiple objects using bounding boxes, which correspond to rectangular shapes around the objects.

2.1 Object Detection

2.1.1 Anchor Based:

SDD Series: The Single Shot MultiBox Detector (SSD) series employs a single neural network for object detection, leveraging anchor boxes at multiple scales and aspect ratios. It balances speed and accuracy, making it suitable for real-time applications, distinguishing itself from models like Faster R-CNN with its simplicity and efficiency.

RCNN Series: The Region-based Convolutional Neural Network (R-CNN) series involves generating region proposals and then classifying each. Models like Faster R-CNN improve upon the original by integrating region proposal networks, enhancing speed and accuracy, becoming a cornerstone in object detection by efficiently localizing and classifying objects in images.

YOLO Series: The You Only Look Once (YOLO) series is renowned for its real-time object detection capabilities. By framing detection as a single regression problem, YOLO processes images extremely quickly. It predicts bounding boxes and class probabilities directly from full images, trading some accuracy for speed, and has evolved through multiple versions to enhance performance.

2.1.2 Anchor Free:

Key Point Based: Key point-based detection methods identify objects by detecting their key points (e.g., corners, center points). These methods bypass traditional anchor boxes, improving detection accuracy and reducing computational complexity. They excel in applications requiring precise localization, such as human pose estimation and facial landmark detection.

Anchor Point Based: Anchor point-based methods forego predefined anchor boxes, focusing instead on predicting key anchor points directly from the image. This approach simplifies the network architecture and reduces computational overhead, enhancing efficiency and potentially improving detection accuracy, especially in cases with varied object sizes and shapes.

YOLO Series: In its later iterations, the YOLO series has incorporated anchor-free techniques, blending the benefits of both anchor-based and anchor-free approaches. This hybrid model aims to improve detection accuracy while maintaining the high-speed processing characteristic of YOLO, making it versatile for various real-time object detection tasks.

2.1.3 Transformer Based:

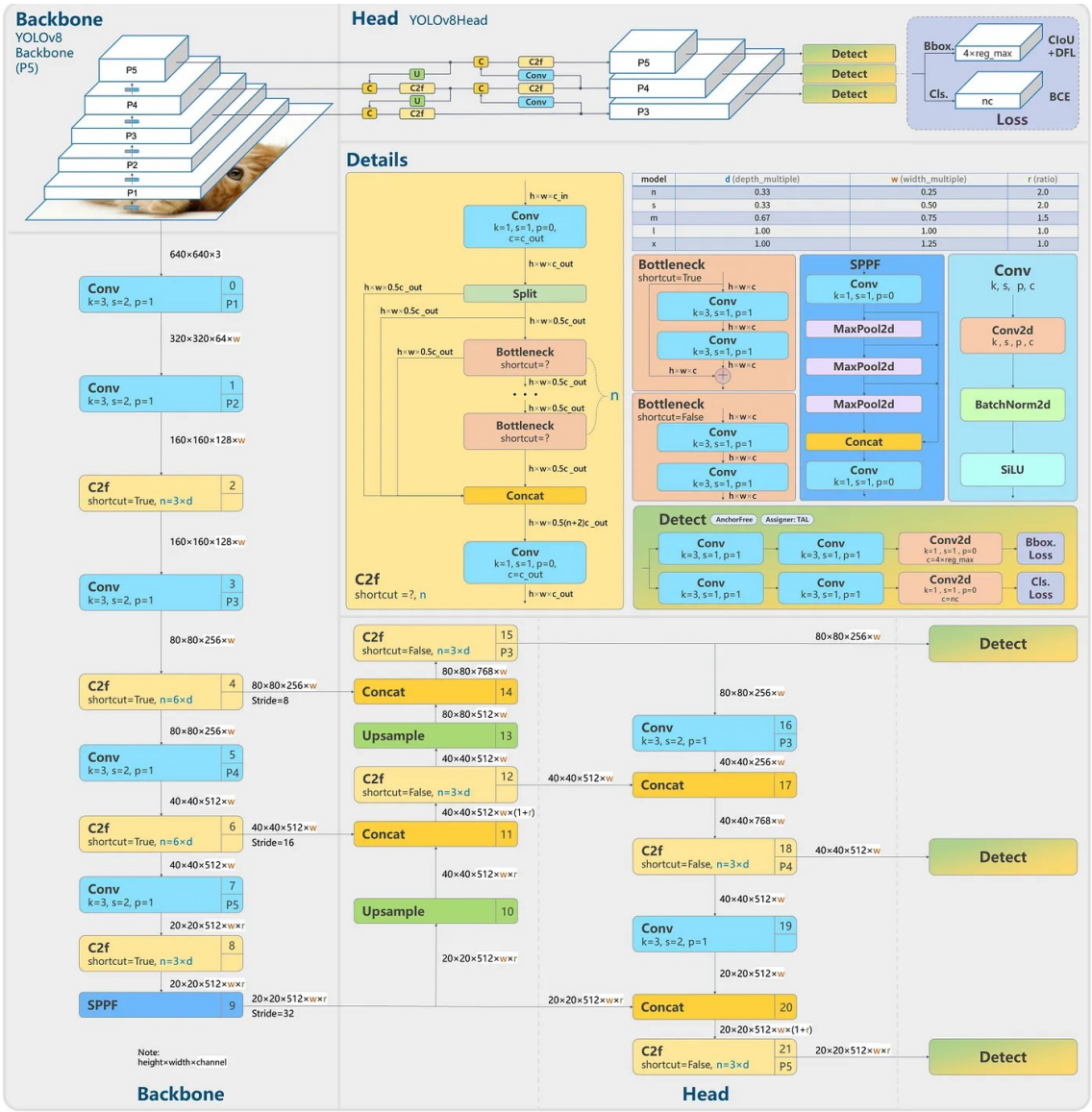
DETR Series: The Detection Transformer (DETR) series introduces transformers to object detection, utilizing a set-based global loss to enforce unique predictions. DETR models combine CNNs with transformer encoders and decoders, excelling in end-to-end object detection without needing traditional anchor boxes or complex post-processing steps.

VIT Series: The Vision Transformer (ViT) series adapts the transformer architecture, originally designed for natural language processing, to image classification tasks. By treating image patches as tokens, ViTs leverage self-attention mechanisms to model long-range dependencies, demonstrating competitive performance and scalability on large-scale image datasets.

2.2 Yolo v8 :

YOLO is a convulsional neural network that predicts bounding boxes and class probabilities of an image in a single evaluation. YOLO v8 Architecture has 3 essential blocks in the algorithm and everything will occur in these blocks, which are:

2.2.1 Yolo v8 Architeture:



```
scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml'
  will call yolov8.yaml with scale 'n'
    [depth, width, max_channels]
n: [0.33, 0.25, 1024] # YOLOv8n summary:
225 layers,  3157200 parameters, 3157184 gradients,   8.9 GFLOPs

s: [0.33, 0.50, 1024] # YOLOv8s summary:
225 layers, 11166560 parameters, 11166544 gradients,  28.8 GFLOPs

m: [0.67, 0.75, 768] # YOLOv8m summary:
295 layers, 25902640 parameters, 25902624 gradients,  79.3 GFLOPs

l: [1.00, 1.00, 512] # YOLOv8l summary:
365 layers, 43691520 parameters, 43691504 gradients, 165.7 GFLOPs

x: [1.00, 1.25, 512] # YOLOv8x summary:
365 layers, 68229648 parameters, 68229632 gradients, 258.5 GFLOPs
```

Backbone :

The main function of the backbone architecture is to extract meaning full features from the input image. It mainly captures simple patterns like edges and textures in the initial layers. It will provide a rich, hierarchical representation of the input.

Neck:

Neck is the middle part of the YOLO V8 Architecture. It acts as a bridge between the backbone and the head. The main function of the neck is performing feature fusion operations and integrating contextual information. Basically the Neck assembles feature pyramids by aggregating feature maps obtained by the Backbone, in other words, the neck collects feature maps from different stages of the backbone. It Perform concatenation or fusion of features of different scales to ensure that the network can detect objects of different sizes and colors.

Neck Architecture integrates contextual information to improve detection accuracy by considering the broader context of the scene. It reduces the spatial resolution and dimensionality of resources to facilitate computation, a fact that increases speed but can also reduce the quality of the model.

Head:

The head is the final part of the YOLO V8 Architecture Its main function is to generate the network's outputs, such as bounding boxes and confidence scores for object detection. Head Architecture generates bounding boxes in the image for objects and also assigns confidence score to each bounding box to indicate how likely an object is present in the image.It also sorts the objects in the input image according to their categories

2.2.2 Modified Yolo v8 Architeture:

```
# Parameters
nc: 9 # number of classes

# YOLOv8.0n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, Conv, [2048, 3, 2]] # 9-P6/32
  - [-1, 3, C2f, [2048, True]]
  - [-1, 1, SPPF, [2048, 5]] # 11

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 8], 1, Concat, [1]] # cat backbone P5
  - [-1, 3, C2, [1024, False]] # 14

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2, [512, False]] # 17

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2, [256, False]] # 20 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 17], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2, [512, False]] # 23 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 14], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2, [1024, False]] # 26 (P5/32-large)

  - [-1, 1, Conv, [1024, 3, 2]]
  - [[-1, 11], 1, Concat, [1]] # cat head P6
  - [-1, 3, C2, [2048, False]] # 29 (P6/64-xlarge)
```

```
- [[18, 23, 26, 29], 1, Detect, [nc]] # Detect(P3, P4, P5, P6)

YOLOv8modifiedmodel summary:
386 layers, 53330084 parameters, 53330068 gradients, 120.2 GFLOPs
```

2.3 Conversion of KITTI labels to YOLO format

Converting KITTI labels to YOLO format involves translating the bounding box information and class labels into the format expected by YOLO.

2.3.1 KITTI Labels:

Car 0.0 0 1.57 712.40 143.00 810.73 307.92 1.89 1.64 4.67 1.84 1.47 8.41 0.01

```
Car - Type
0.0 - Truncated
0 - Occluded
1.57 - Alpha
712.40 - BBox_x1
143.00 - BBox_y1
810.73 - BBox_x2
307.92 - BBox_y2
1.89 - Dimensions_3D_x
1.64 - Dimensions_3D_y
4.67 - Dimensions_3D_z
1.84 - Location_3D_x
1.47 - Location_3D_y
8.41 - Location_3D_z
0.01 - Rotation_y
```

2.3.2 YOLO Labels:

```
(class_id  x_center  y_center  width  height)
```

Conversion of Kitti labels format to yolo format :

$The formula for$
x_center is:

$$x\_center = \frac{(BBox\_x1 + BBox\_x2)}{2 \times (image\_width)}$$

y_center is:

$$y\_center = \frac{(BBox\_y1 + BBox\_y2)}{2 \times (image\_height)}$$

$width$ is:

$$width = \frac{(BBox\_x2 - BBox\_x1)}{(image\_width)}$$

$height$ is:

$$height = \frac{(BBox\_y2 - BBox\_y1)}{(image\_height)}$$

3 Graphs

3.1 Train :

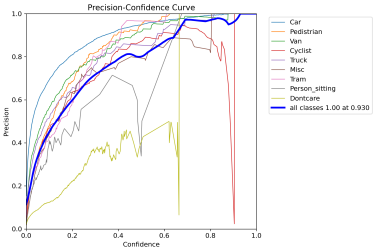


Figure 1: Precision-Confidence Curve

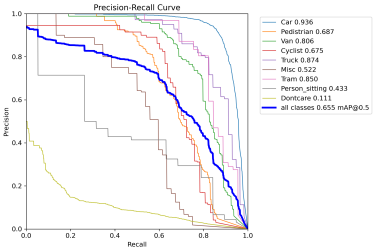


Figure 2: Precision-Recall Curve

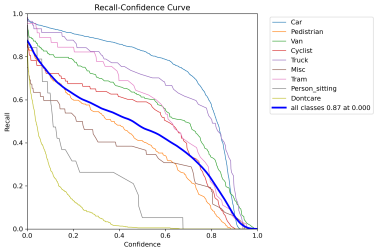


Figure 3: Recall-Confidence Curve

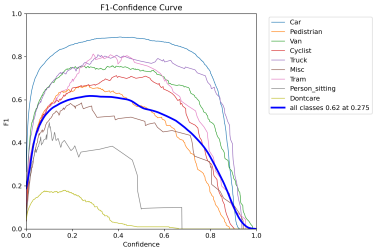


Figure 4: F1-Confidence Curve

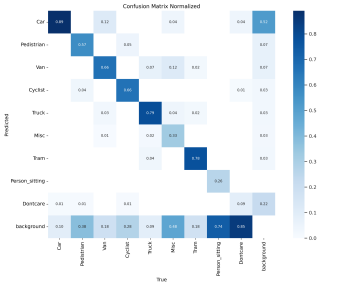


Figure 5: confusion matrix normalized

3.2 Validation :

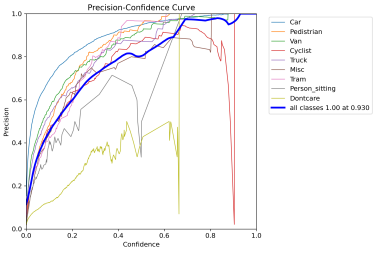


Figure 6: Precision-Confidence Curve

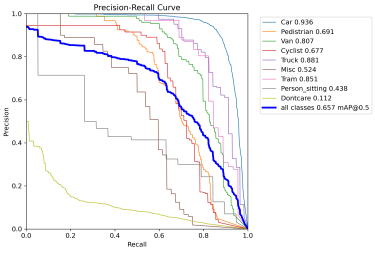


Figure 7: Precision-Recall Curve

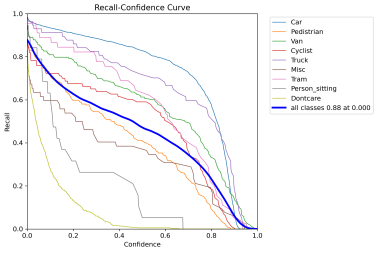


Figure 8: Recall-Confidence Curve

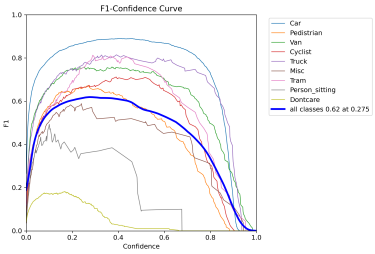


Figure 9: F1-Confidence Curve

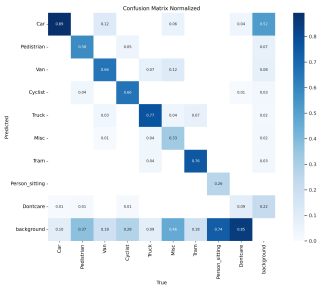


Figure 10: Confusion matrix normalized

4 Results:

To train YOLOv8 for optimal object detection performance, a systematic approach was employed. We began by understanding the architecture of YOLOv8 and its layers. We added some layers to the YOLOv8 model to enhance its feature extraction capabilities and improve its ability to detect smaller objects. These additional layers included convolutional layers with different kernel sizes and dilations, allowing the model to capture features at various scales. This helped the model to better understand complex scenes and accurately localize objects. Furthermore, we fine-tuned the model using a KITTI dataset, for about 50 epoches. Experimentation involved monitoring the training process through epoch vs. loss and metric curves. Hyperparameters such as learning rate, batch size, and anchor box sizes were carefully tuned to achieve the best performance. By iteratively adjusting training parameters and observing the resulting curves, we aimed to achieve a well-balanced modified YOLOv8 model tailored to our specific needs.

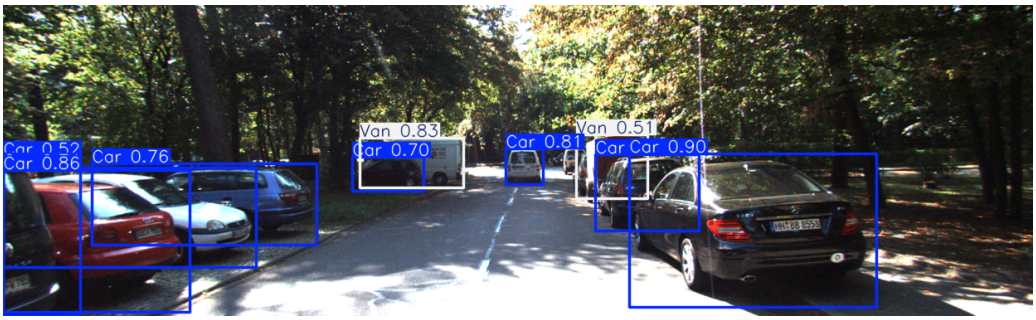


Figure 11: 7 Cars and 2 Vans were detected with bounding boxes

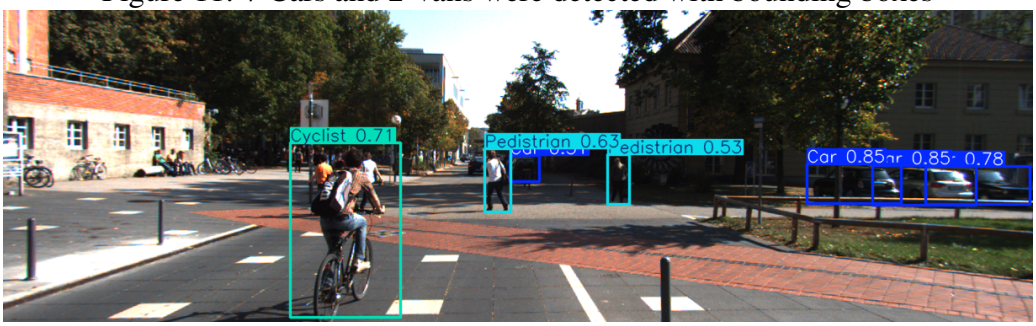


Figure 12: 1 Cyclist 4 Cars and 2 Pedistrians were detected with bounding boxes

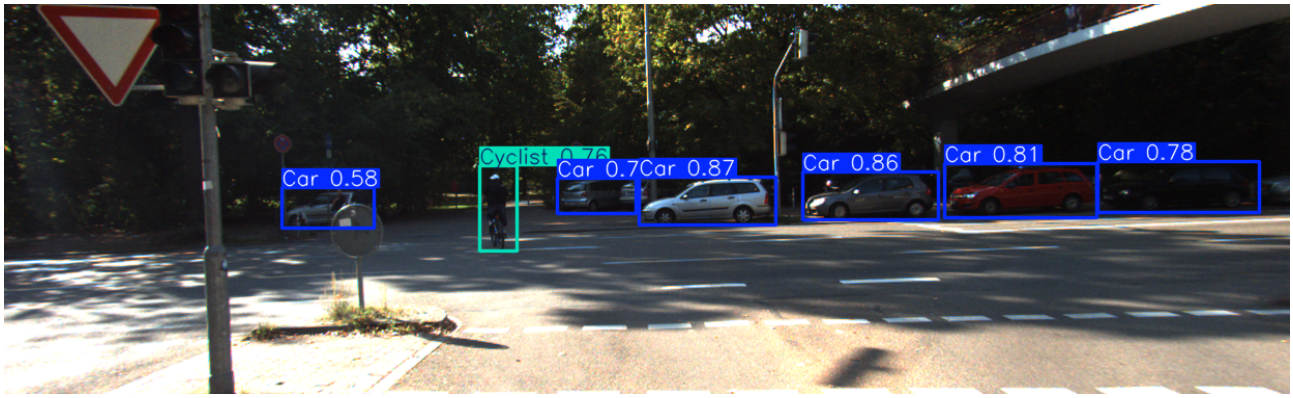


Figure 13: 6 Cars and 1 Cyclist were detected with bounding boxes

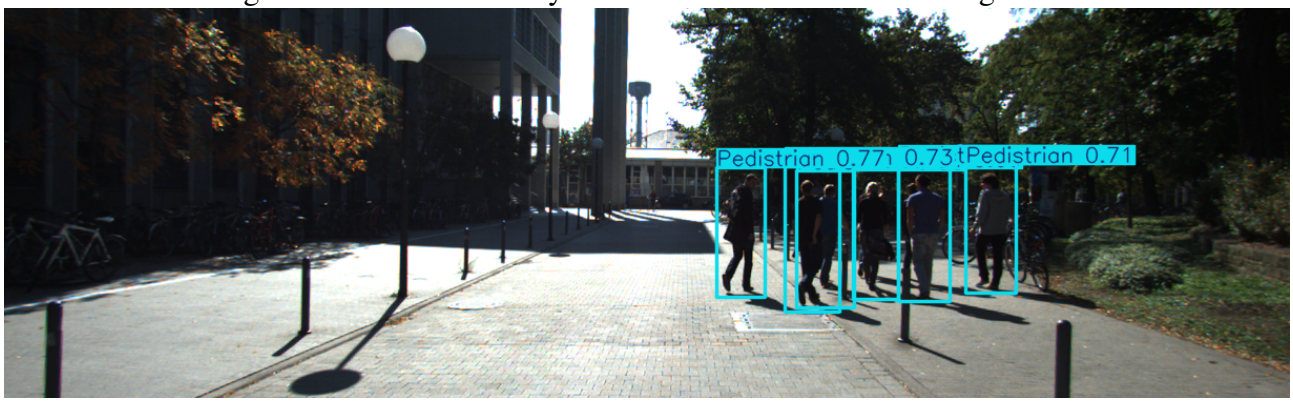


Figure 14: Pedistrians were detected with bounding boxes

These image demonstrates YOLO's object detection capabilities. Bounding boxes corresponding class labels are visualized, indicating successful object identification and classification across the image. This suggests strong model performance.

After the modifications done to yolov8 and trained for 50 epoches the mean average precision occured and the actual mean average precision for yolov8 models:

YOLOv8	NANO	SMALL	MEDIUM	LARGE	XLARGE
mAP@0.5	0.5	0.6	0.65	0.70	0.75
mAP@0.5:0.95	0.3	0.4	0.45	0.5	0.55

Table 1: performances of models of Yolov8

YOLOv8	MODIFIED
mAP@0.5	0.718
mAP@0.5:0.95	0.492

Table 2: performance of modified model of yolo v8

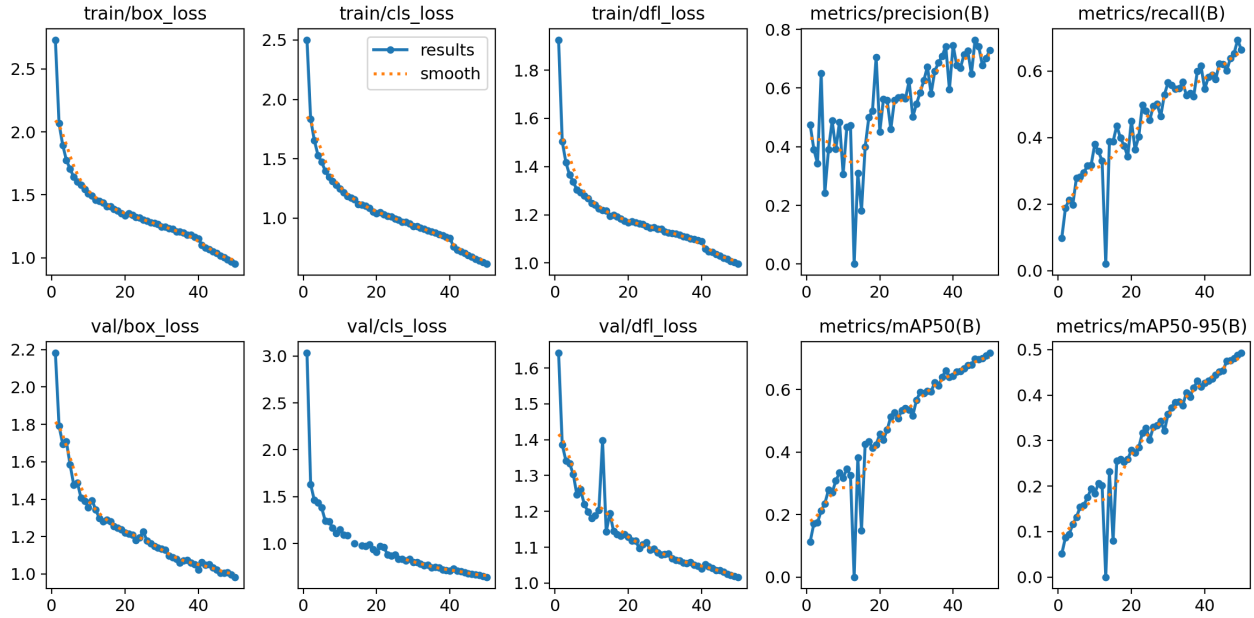


Figure 15: Epoch vs Loss Metric curves for Modified YOLO v8

5 Conclusions and Future Scope :

5.1 Conclusion:

By outlining clear objectives and leveraging comprehensive training data, we have successfully trained and fine-tuned deep learning models to detect KITTI dataset attributes accurately. This project successfully detecting the objects by giving input while testing. The foundation was laid through a comprehensive training process over 50 epoches, leveraging a KITTI dataset of 7,415 images. Notably, the trained model achieved a mean Average Precision (mAP) of 71.8,and mAP (50-95) of 49.2 on the KITTI benchmark dataset. This demonstrates the model's strong potential for Object detection.

5.2 Future Scope

we can develop a user-friendly application for real-time Vehicles, Cyclist, and Pedistrian detection using live camera feeds. To further improve the resilience precision and utility of our detection system, we must vary our training data and increase the size of our attribute collection.Training a larger model, potentially a variant of YOLOv8, on a significantly bigger dataset encompassing a wider range of attributes. This will enhance the model's ability to handle real-world scenarios with greater complexity. Although the method as it stands only handles one input , handling live video streams is a fascinating next step.

6 References

- "A Survey of Computer Vision Methods for 2D Object Detection from Unmanned Aerial Vehicles" by Dario Cazzato ,Claudio Cimarelli, Jose Luis Sanchez-Lopez ,Holger Voos and Marco

Leo

- W Chen, Y Li, Z Tian, F Zhang - Array, 2023 - Elsevier
- D Cazzato, C Cimarelli, JL Sanchez-Lopez, H Voos... - Journal of ..., 2020 - mdpi.com
- M Irani, P Anandan - IEEE transactions on pattern analysis and ..., 1998 - ieeexplore.ieee.org
- ChatGPT
- Science Direct
- H Wang, C Liu, Y Cai, L Chen... - IEEE Transactions on ..., 2024 - ieeexplore.ieee.org