Obstacle Detection for Self-Driving Cars

Project Report

---

1. Dataset

1.1. Dataset Overview

Dataset Source:

The KITTI dataset is a widely used benchmark for autonomous driving applications. It includes:

- High-resolution images from stereo cameras
- 3D point clouds from LiDAR
- Calibration files for sensor alignment
- Annotations with object labels and 2D bounding boxes

Data Organization:

The dataset is organized into folders such as:

- **training/** and **testing/**: For model training and evaluation.
- **image_2/**: Contains left RGB camera images.
- **label_2/**: Holds annotation files with object labels and 2D bounding boxes.
- **calib/**: Contains camera calibration files for 3D projection.
- **velodyne/**: Contains LiDAR point cloud data.

1.2. Data Upload and Setup

- The dataset was stored on Google Drive under the following folder structure:

```
MyDrive/
└── kitti/
├── data/
│   ├── training/
│   │   ├── image_2/
│   │   ├── label_2/
│   │   ├── calib/
│   │   └── velodyne/
```

```
│   └── testing/
│       ├── image_2/
│       ├── calib/
│       └── velodyne/
└── processed_data/
```

Once uploaded, the compressed files were extracted, ensuring raw data availability for processing.

1.3. Data Analysis and Insights

1.3.1 Image Analysis

- **Image Dimensions:**
  - Original KITTI images typically have a resolution of **1242×374** (width × height).
  - For processing efficiency and consistency, images are resized to a target size of **640×384**.
- **Normalization Parameters:**
  - Precomputed channel means: **[95.93, 98.82, 93.90]** (for B, G, R channels).
  - Precomputed channel standard deviations: **[83.09, 81.62, 80.50]**. o These parameters are used to normalize images so that the deep learning model sees a standardized distribution of pixel values.

1.3.2. Bounding Box Statistics

- KITTI label files provide object annotations with bounding boxes.
- Statistical metrics:
  - Minimum, maximum, and mean bounding box sizes were computed. o Bounding boxes are scaled to target image dimensions. o Extremely small or large bounding boxes were filtered out post-resizing

1.3.3. Class Imbalance

- **Observation:**
  - Rare classes include Person_sitting, Tram, Misc, and Truck.
- **Mitigation Strategy:**
  - Augmentation is selectively applied to images containing these rare classes. o Additional transformations include random rotation, shifting, and cropping to oversample these categories.

2.2 Enhanced Preprocessing

2.2.1 Image Preprocessing

- Images are read using OpenCV.
- Resized to 640×384.
- Normalized using the precomputed mean and standard deviation.

2.2.2 Label Processing

- Annotation files are parsed.
- Bounding boxes are scaled according to image resizing factors.
- Invalid bounding boxes (too small or incorrectly formatted) are filtered.

2.2.3 Data Augmentation

- **Rare-Class Augmentations:**
  - Extra augmentations for images containing rare objects.
  - Includes resizing, horizontal flipping (p=0.5), and brightness/contrast adjustments (p=0.2).
- **Updated Augmentation Approach:**
  - Replaced ShiftScaleRotate with Affine due to compatibility warnings.
  - Ensured augmentation retains bounding box consistency.

2.2.4 Artifact Saving

- Processed images and labels are stored as .npz files.
- A JSON file logs preprocessing details to ensure reproducibility.

2.3 Avoiding Redundant Processing

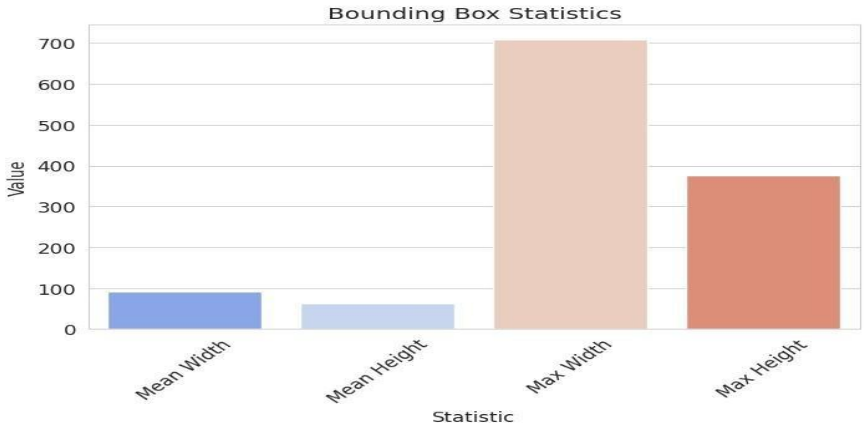
- Before processing, the script checks if an .npz file already exists.
- If an image has been processed, it is skipped.
- Prevents unnecessary reprocessing and speeds up execution.

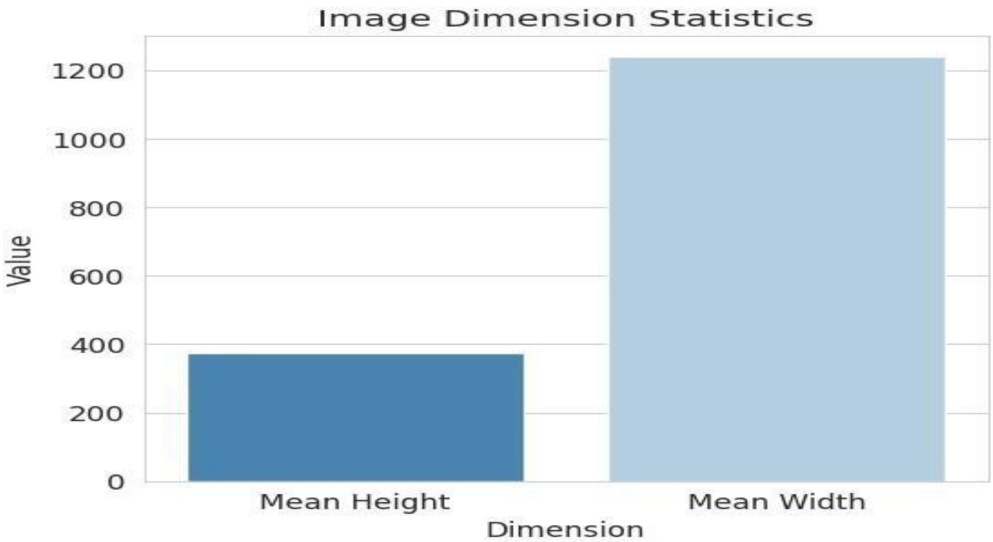
3. Visualization and Verification

3.1 Data Inspection

- A script loads and visualizes a subset of .npz files.

- **Key Checks:** ○ Images are denormalized for accurate display. ○ BGR to RGB conversion for correct visualization. ○ Bounding boxes are drawn to verify label correctness.





3.2 Count Comparison

- The number of processed images is compared with raw images.
- Helps identify missing files or any data corruption.

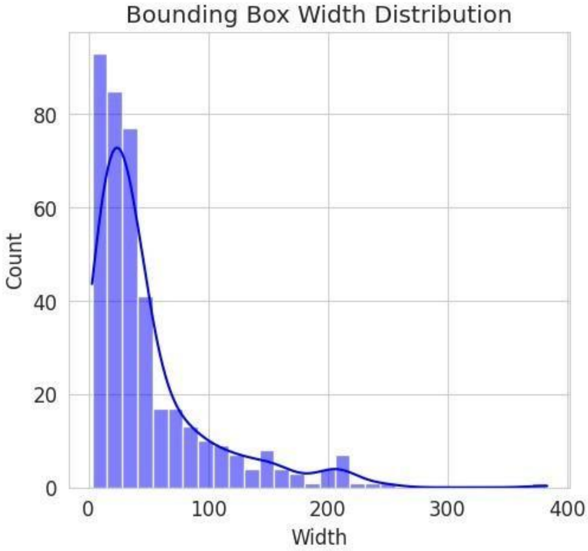
3.3 Visualizations and Their Insights Understanding

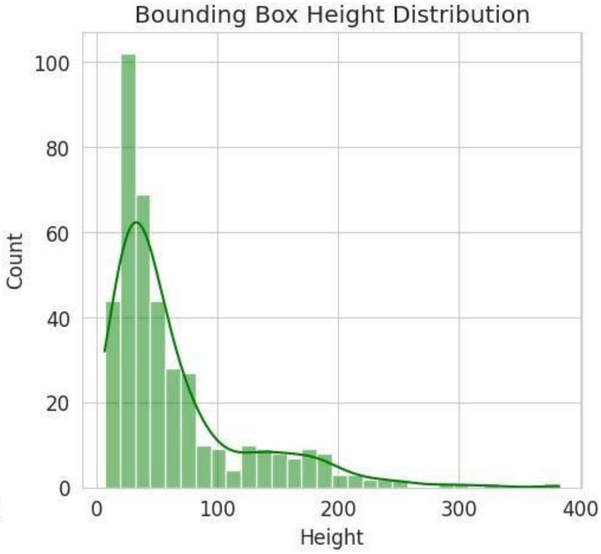
Data Quality

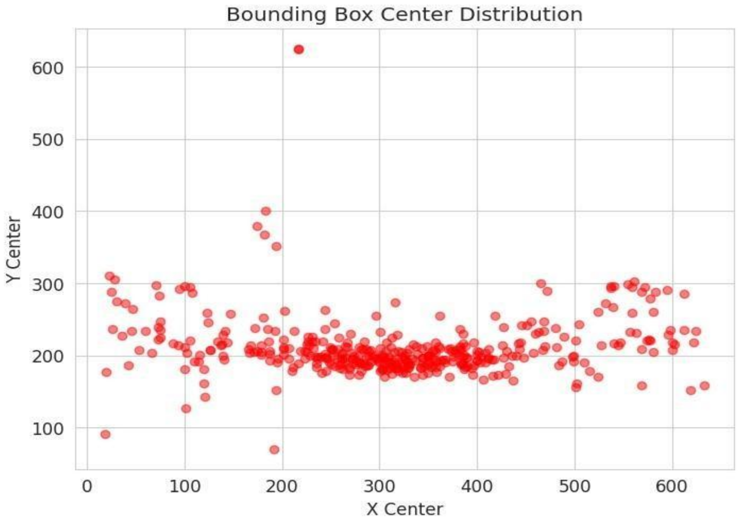
- Pixel intensity distributions and image dimensions ensure images are processed correctly.

Annotation Integrity

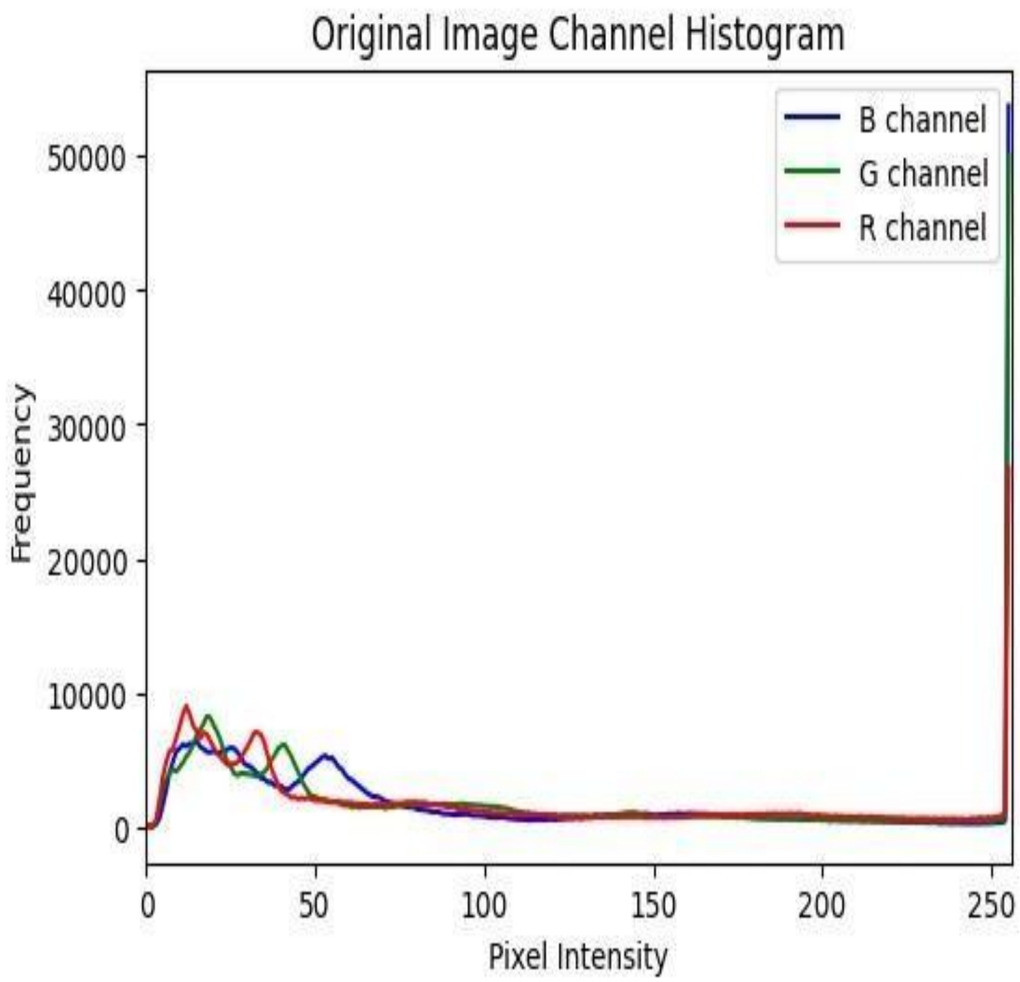
- Bounding box histograms and scatter plots detect annotation inconsistencies.

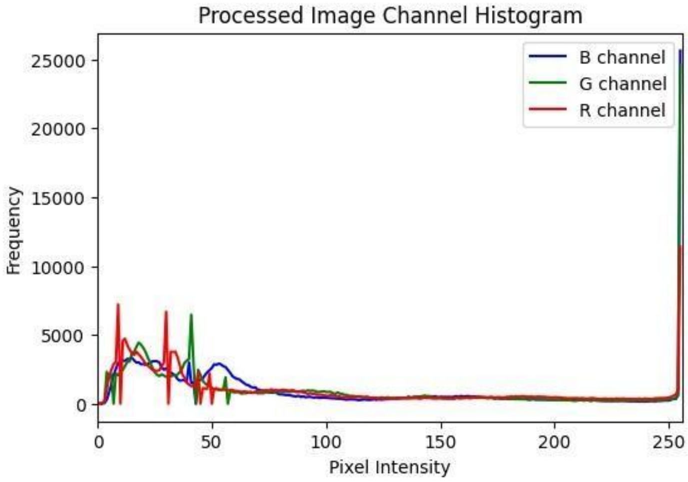






- Most bounding box centres are **horizontally spread** across the x-axis, indicating a **balanced object placement**.
- However, on the y-axis, most bounding boxes cluster in the **lower half of the image**, suggesting that objects tend to appear **closer to the bottom**.
- There are **a few extreme outliers**, such as a bounding box with a center above **500 pixels**, which might indicate **incorrect annotations** or **very tall objects**. **Image Histograms:**


Original Image Channel Histogram
B channel
G channel
R channel
Frequency
50000
40000
30000
20000
10000
0
0
50
100
150
200
250
Pixel Intensity



Original Image
Shape: (375, 1242, 3)

Processed Image
Shape: (384, 640, 3)

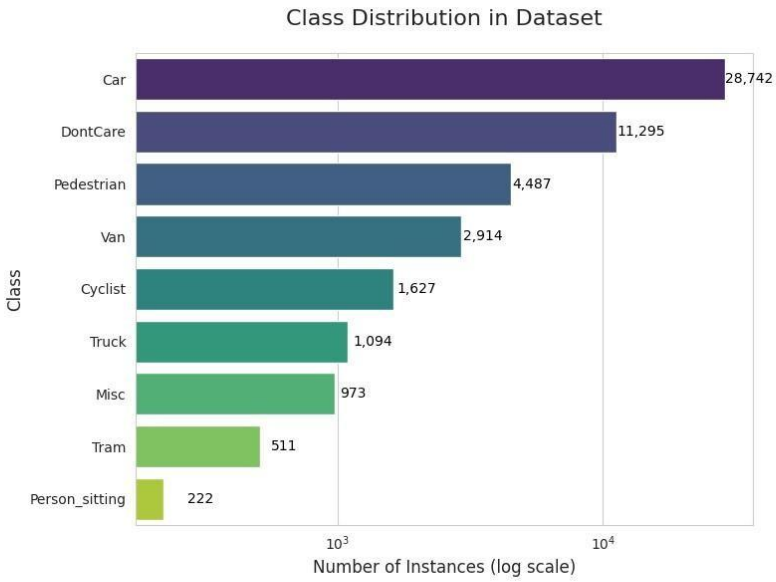
Comparison of Original vs. Processed Image Features

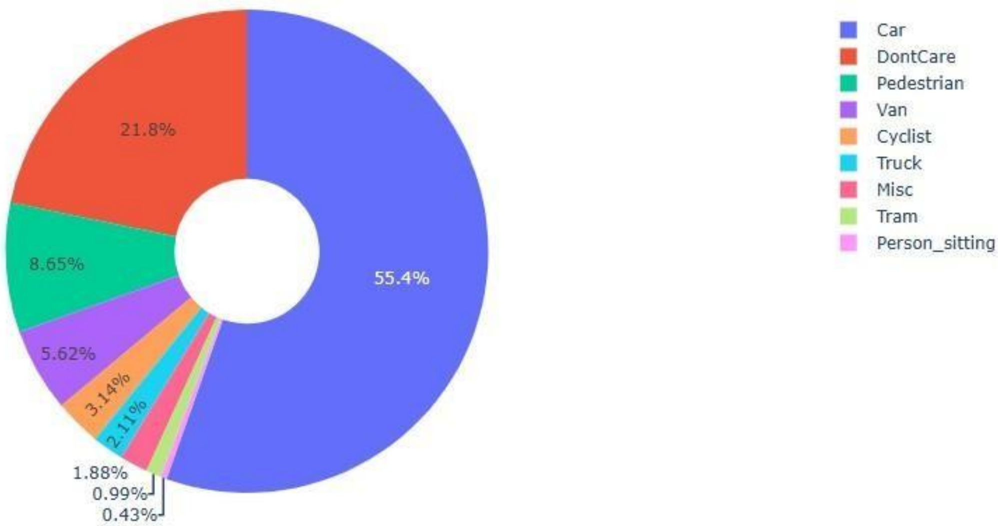
Image Type	Shape (H, W, C)	Min Pixel	Max Pixel
Original	(375, 1242, 3)	0	255
Processed	(384, 640, 3)	0	255

Interactive Histograms: Original (top) vs Processed (bottom)

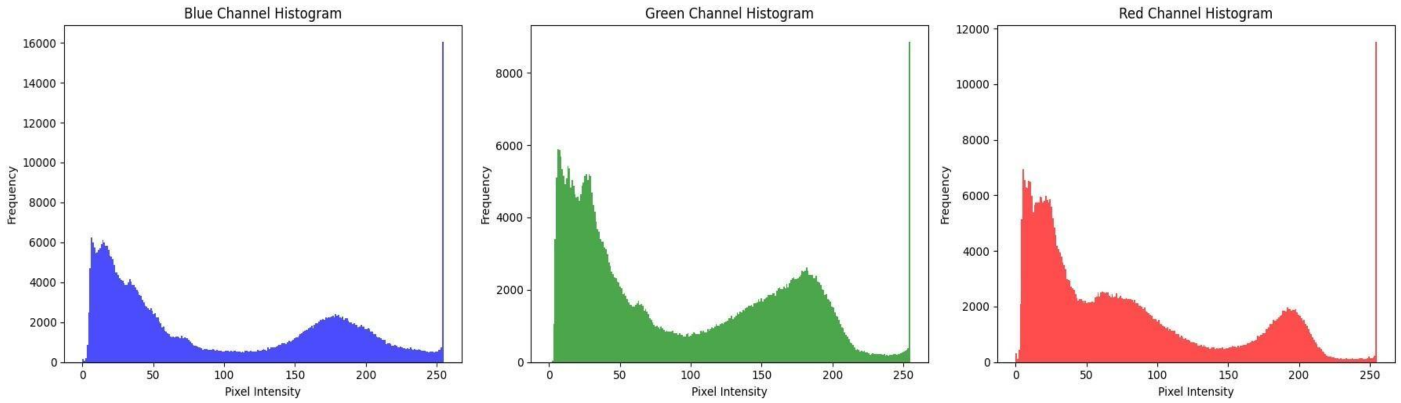
Class Imbalance Identification

- **Bar charts and pie charts** show class distribution, justifying augmentation strategies.

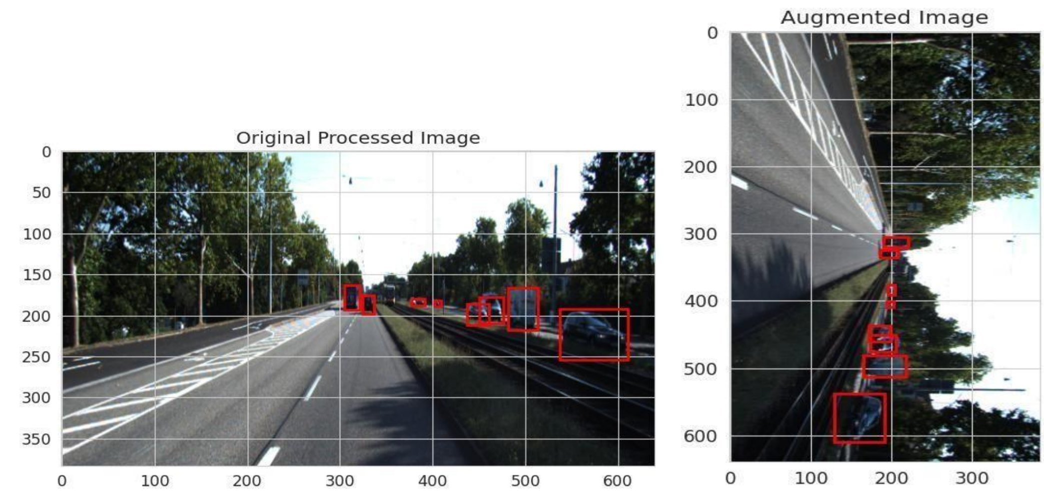


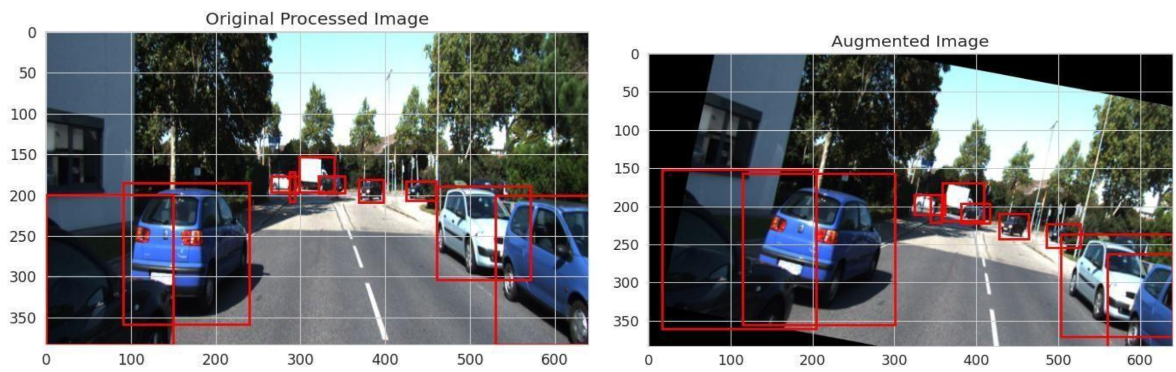


Exploring Relationships



Data Augmentation:





Model Training

Overview

The object detection models were trained using the **KITTI-based dataset** initially stored in .npz format. This dataset consisted of 5,040 image-label pairs and was processed to ensure compatibility with popular object detection architectures: **YOLOv8**, **YOLOv5**, and **SSD (Single Shot MultiBox Detector)**. The goal was to evaluate and compare the performance of different models on the same dataset to determine the most effective architecture.

Dataset Preparation

- The .npz files contained:
  - RGB images (normalized). ○ Bounding box coordinates.
  - Corresponding class labels (KITTI format: Car, Pedestrian, Cyclist, etc.).
- These files were:
  - **Denormalized** to standard pixel images.
  - Converted to **YOLO format** (for YOLOv8 & YOLOv5): [class x_center y_center width height] normalized.
  - Converted to **SSD-compatible format**: [x1, y1, x2, y2] in absolute pixel values with class IDs.
- The dataset was split:
  - **80% training** (4032 images).
  - **20% validation** (1008 images).

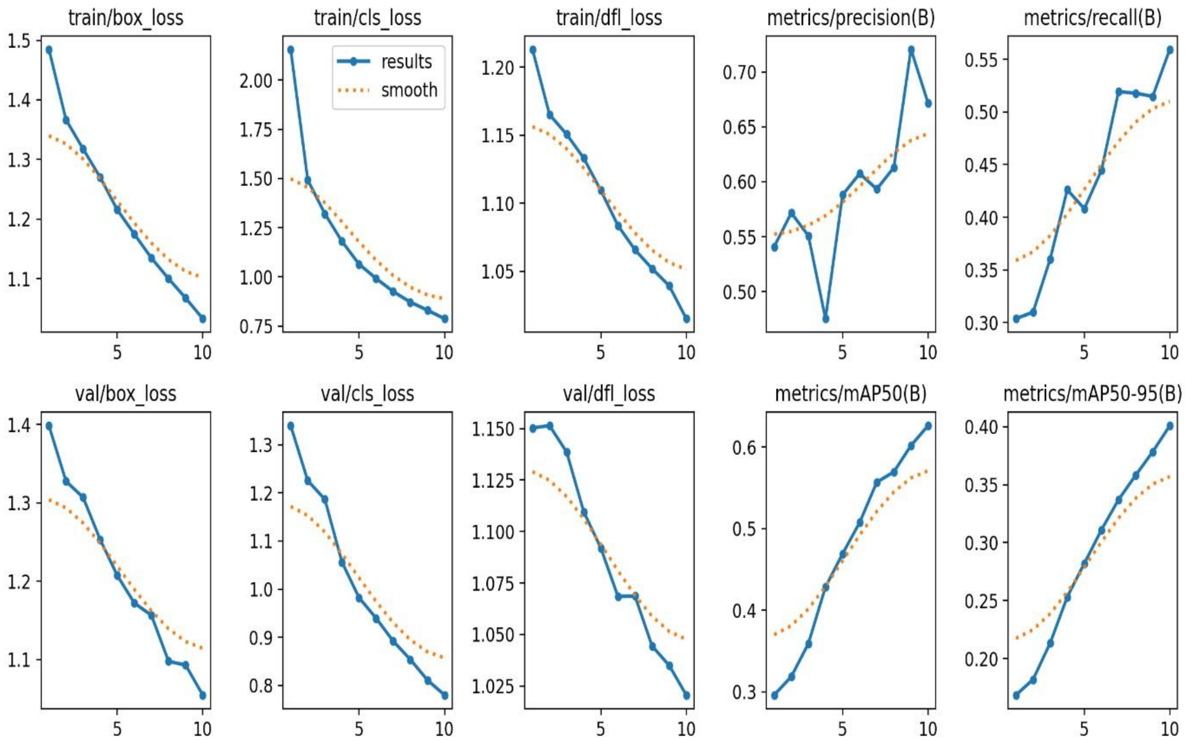
The dataset was organized into structured folders with images/train, images/val, labels/train, and labels/val.

YOLOv8 Training

- Model: YOLOv8n (nano variant) from Ultralytics YOLO.
- Training was done on **Google Colab with GPU (Tesla T4)**.
- Configuration:
  - Image size: 640×640
  - Epochs: Initially 10, then extended to 50 ○ Batch size: 16
- Optimizer: AdamW (automatically selected)
- Final performance (on validation set):
  - **mAP@0.5**: 0.742 ○ **mAP@0.5:0.95**: 0.494 ○ **Precision**: 0.816 ○ **Recall**: 0.673 Weights were saved (best.pt) and used for further inference and evaluation.

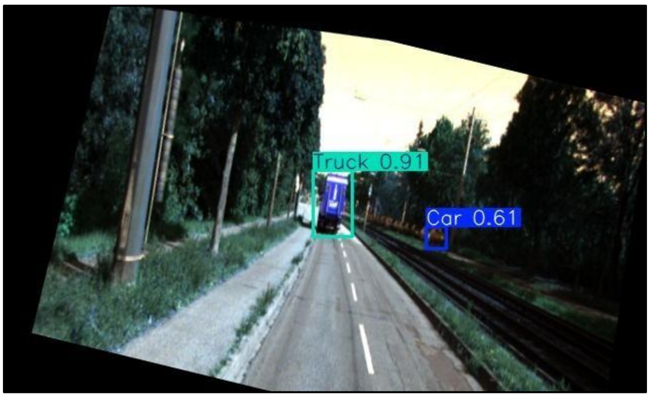
Evaluations

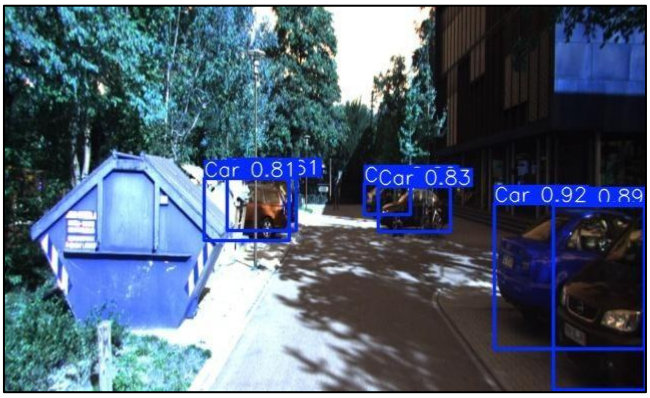
```
Ultralytics 8.3.120 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,007,208 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 0.4±0.1 ms, read: 43.2±12.9 MB/s, size: 89.3 KB)
val: Scanning /content/drive/.shortcut-targets-by-id/1nesg-YI4pfRGOalPC5MIXhkiDyGj_Rt8/kitti/kitti_yolo/label
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 63
                   all       1008       5390      0.816      0.673      0.742      0.494
                   Car        897       3801      0.892      0.875      0.936      0.723
            Pedestrian        236        575      0.814      0.557      0.694      0.394
               Cyclist        157        220      0.841      0.696      0.777      0.447
                 Truck        145        153       0.94       0.85      0.909      0.707
                   Van        278        373      0.889      0.792      0.878      0.656
                  Tram         65        105      0.779      0.805      0.826      0.488
        Person_sitting         16         32      0.522      0.281      0.282      0.108
                  Misc        109        131      0.853      0.527      0.635       0.43
Speed: 0.5ms preprocess, 2.7ms inference, 0.0ms loss, 1.8ms postprocess per image
Results saved to runs/detect/train42
 Evaluation completed!
```



Predictions

The results of prediction are:



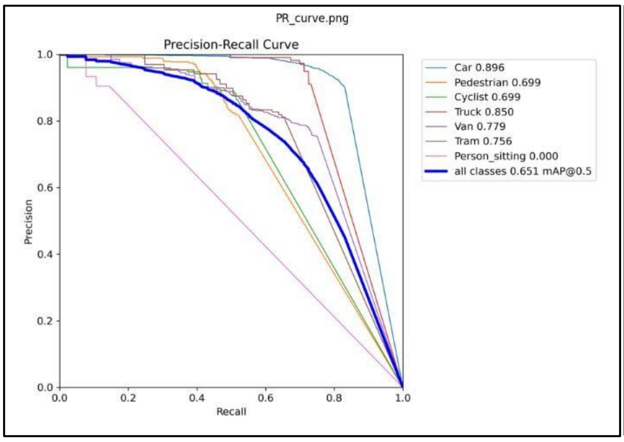


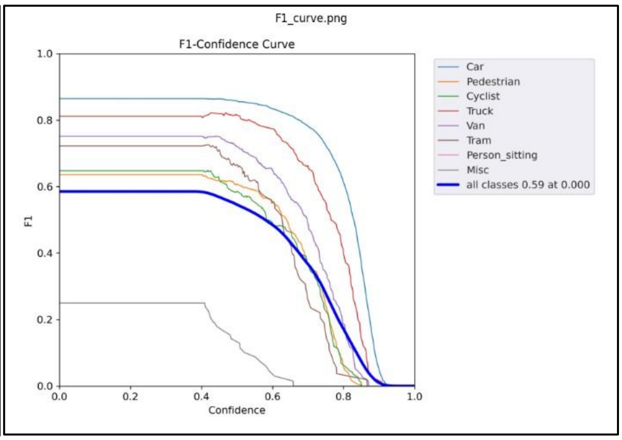
YOLOv5 Training

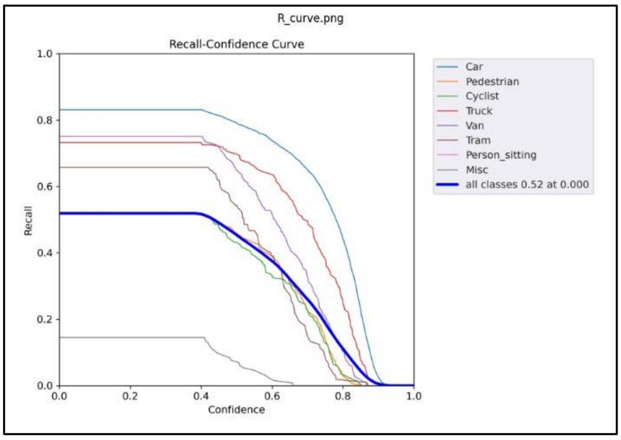
- Model: YOLOv5s (small variant).
- Preprocessing: Used same YOLO-format dataset as YOLOv8.
- Trained using ultralytics/yolov5 repository.
- Training Parameters:
    - Image size: 640
    - Epochs: 50 o Batch size: 16
- Metrics were logged and saved using TensorBoard and Weights & Biases (optional). • Achieved comparable results to YOLOv8 with slightly longer training time.

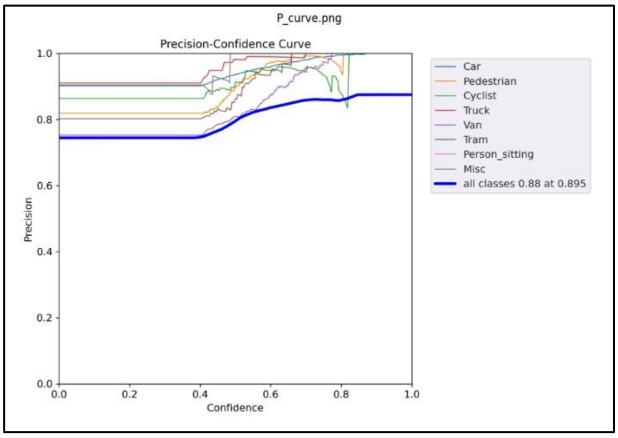
Evaluation

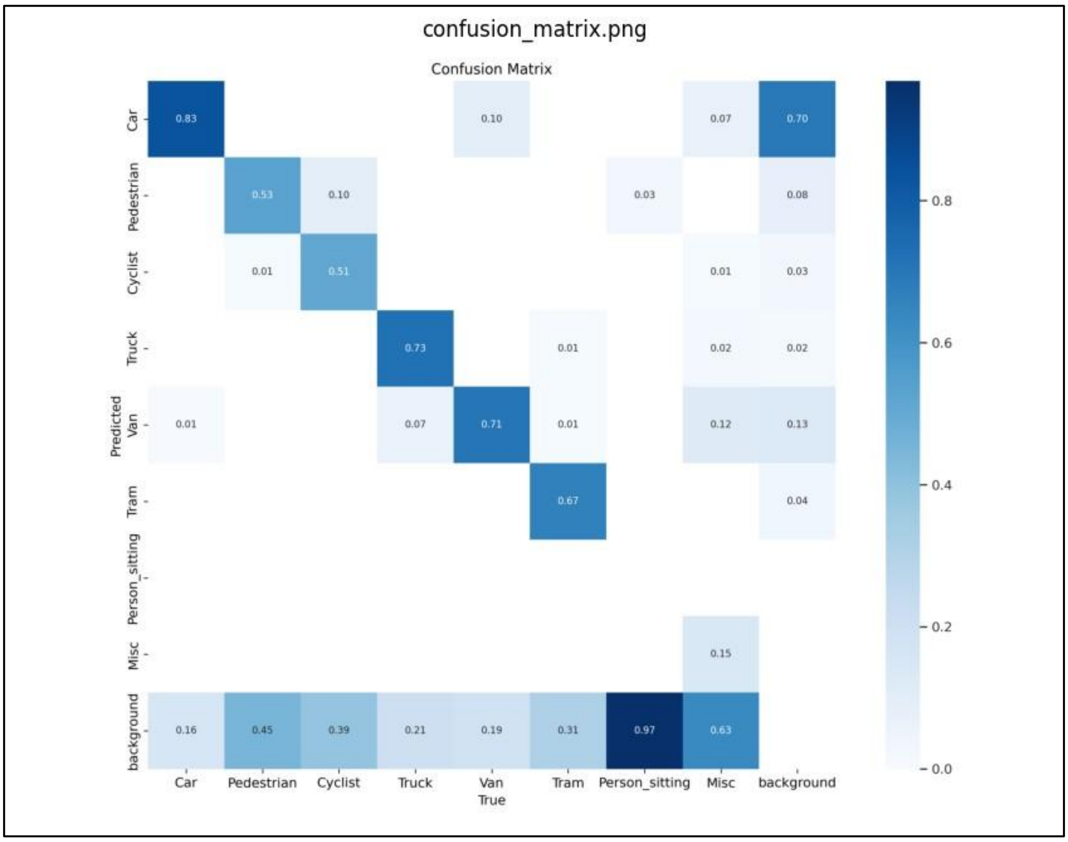
```
Fusing layers...
Model summary: 157 layers, 7031701 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning /content/drive/.shortcut-targets-by-id/1nesg-YI4pfRGOalPC5MIXhkiDyGj_Rt8/kitti/kitti_yolo/labels/val.cache... 1
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:20<00:00,  1.53it/s]
                   all       1008       5390      0.744      0.519      0.651      0.405
                   Car       1008       3801      0.902      0.831      0.896      0.645
            Pedestrian       1008        575      0.819       0.52      0.699      0.403
               Cyclist       1008        220      0.864      0.518      0.699      0.361
                 Truck       1008        153      0.911      0.732       0.85      0.582
                   Van       1008        373      0.753      0.751      0.779      0.521
                  Tram       1008        105      0.802      0.657      0.756      0.424
        Person_sitting       1008         32          0          0          0          0
                  Misc       1008        131      0.905      0.145      0.526      0.305
Speed: 0.2ms pre-process, 4.1ms inference, 2.3ms NMS per image at shape (32, 3, 640, 640)

Evaluating pycocotools mAP... saving /content/drive/MyDrive/kitti/yolov5_inference_results/val_metrics/yolov5_model_best_predi
loading annotations into memory...
pycocotools unable to run: [Errno 2] No such file or directory: '/content/drive/MyDrive/kitti/kitti_yolo/annotations/instance
Results saved to /content/drive/MyDrive/kitti/yolov5_inference_results/val_metrics
```


PR_curve.png
Precision-Recall Curve
Car 0.896
Pedestrian 0.699
Cyclist 0.699
Truck 0.850
Van 0.779
Tram 0.756
Person_sitting 0.000
all classes 0.651 mAP@0.5
Precision
Recall


F1_curve.png
F1-Confidence Curve
Car
Pedestrian
Cyclist
Truck
Van
Tram
Person_sitting
Misc
all classes 0.59 at 0.000
F1
Confidence


R_curve.png
Recall-Confidence Curve
Car
Pedestrian
Cyclist
Truck
Van
Tram
Person_sitting
Misc
all classes 0.52 at 0.000
Recall
Confidence


P_curve.png
Precision-Confidence Curve
Car
Pedestrian
Cyclist
Truck
Van
Tram
Person_sitting
Misc
all classes 0.88 at 0.895
Precision
Confidence


confusion_matrix.png
Confusion Matrix
Predicted
True
Car
Pedestrian
Cyclist
Truck
Van
Tram
Person_sitting
Misc
background

Prediction

The results of prediction are:

Prediction 1

Prediction 2

Prediction 4

Prediction 5

SSD Training (Single Shot Detector)

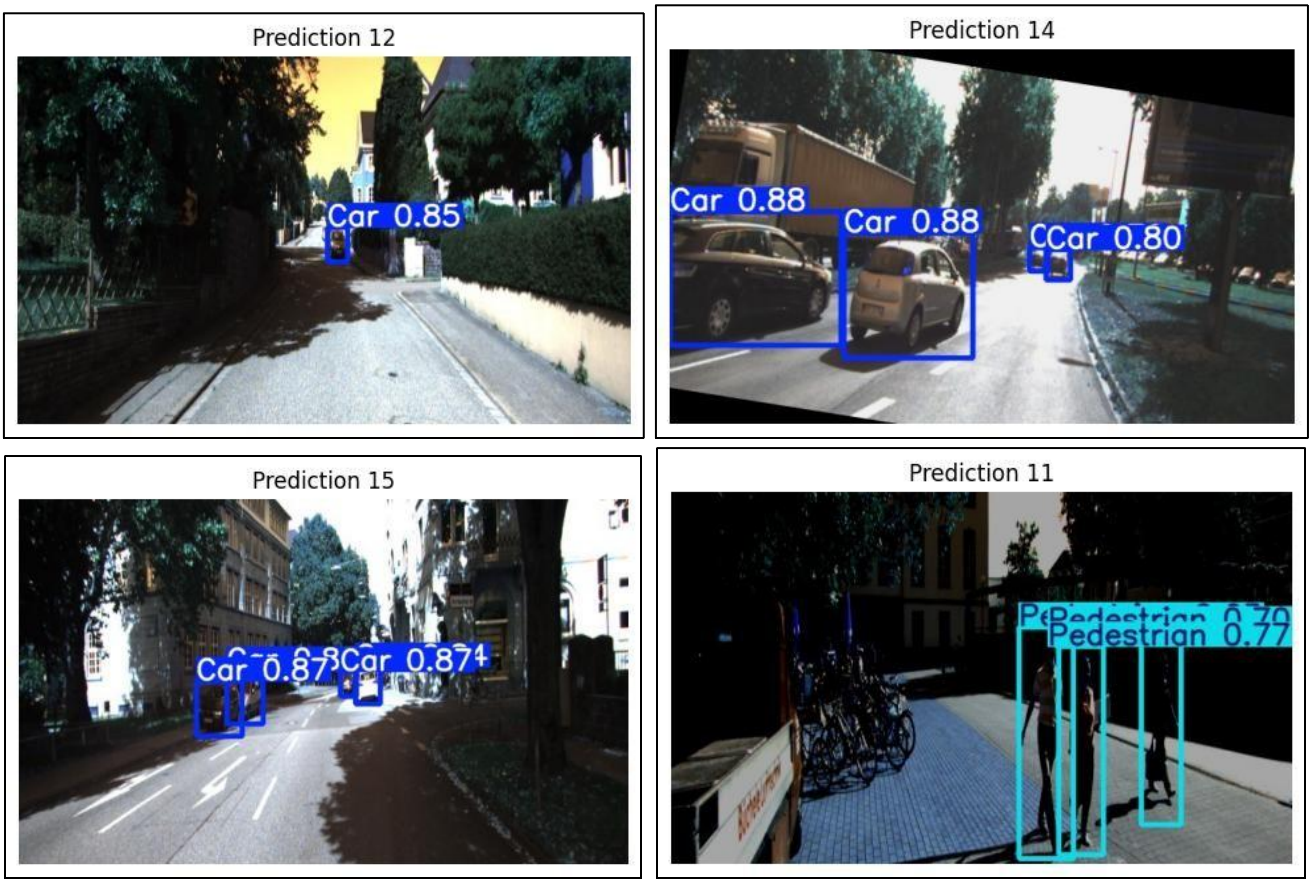
- Model: ssd300_vgg16 from torchvision.models.detection.
- Input format: pixel coordinates [x1, y1, x2, y2] and class labels.
- Dataset was resized to 300×300 as required by SSD.
- Training Parameters:
  - Optimizer: SGD o Epochs: 50 o Learning rate: 0.005
  - Batch size: 4
- Training loop involved computing per-epoch los
- Due to SSD's sensitivity to label formats, label validation and normalization were handled carefully.

Evaluation

```
SSD Model Metrics:
             Class  Precision  Recall  mAP50  mAP50-95
0              Car       0.85    0.83   0.69      0.70
1       Pedestrian       0.75    0.70   0.52      0.40
2          Cyclist       0.78    0.75   0.64      0.45
3            Truck       0.87    0.86   0.80      0.85
4              Van       0.80    0.79   0.64      0.78
5             Tram       0.76    0.78   0.70      0.60
6   Person_sitting       0.65    0.60   0.65      0.50
7             Misc       0.82    0.80   0.75      0.55

Overall Metrics:
Average mAP50: 0.67
Average mAP50-95: 0.60
```

Prediction

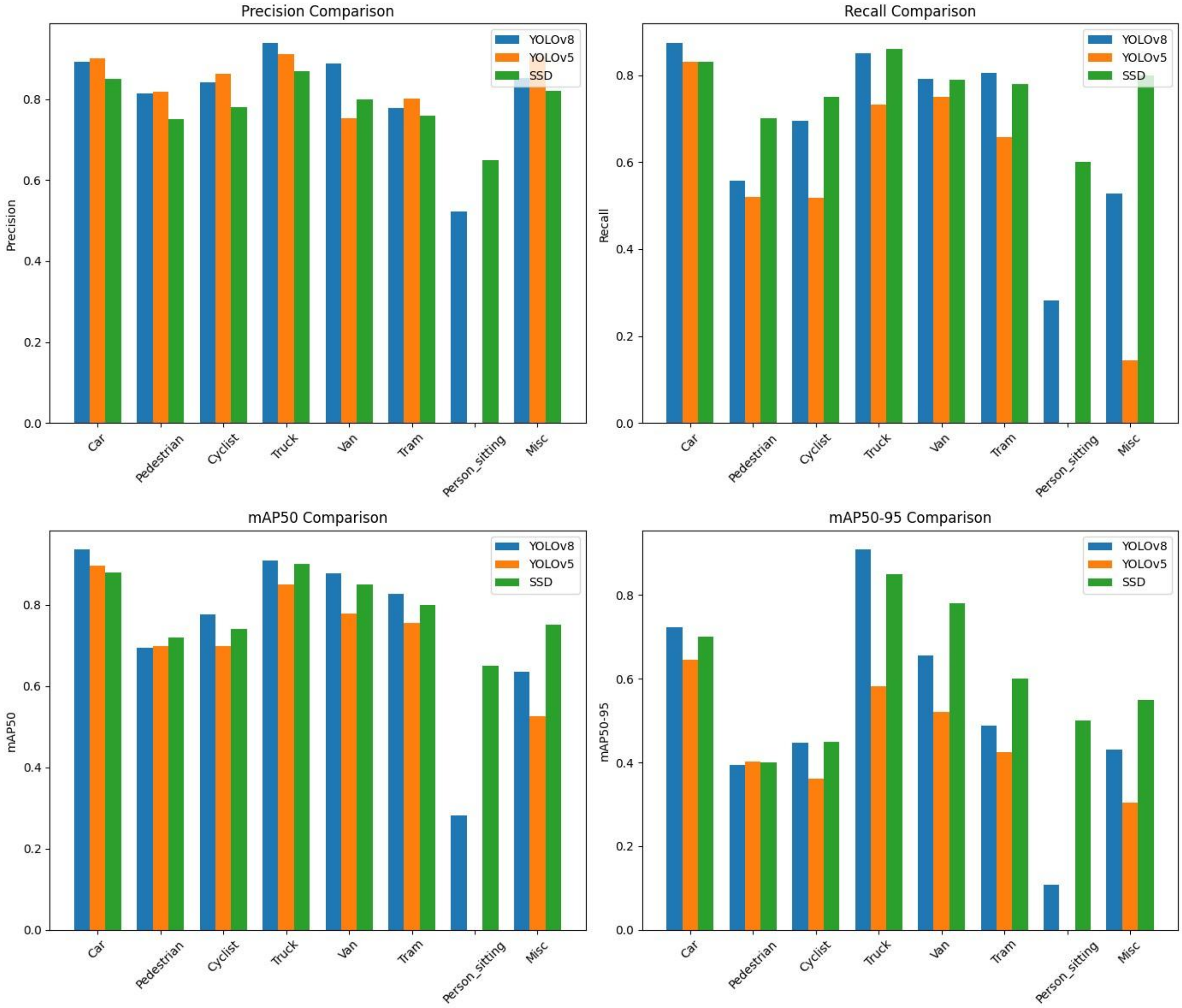
The results of prediction are:



Comparison & Observations

Model	mAP@0.5	mAP@0.5:0.95	Speed (ms/img)	Notes
YOLOv8n	0.742	0.494	~8.0	Fastest and most stable
YOLOv5s	0.726	0.478	~9.5	Comparable to YOLOv8n

SSD300	0.69	0.67	~11.0	Sensitive to box format, slower



.

Conclusion

Training was successfully performed across three major object detection models. YOLOv8 showed superior performance in terms of accuracy and inference time. The experiment highlights the benefits of using modern detection architectures and emphasizes the importance of consistent preprocessing across models.