Configuration Manual

MSc Research Project
Data Analytics

Alphons Zacharia James
Student ID: x23169702

School of Computing
National College of Ireland

Supervisor: Furqan Rustam

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Alphons Zacharia James
Student ID:	x23169702
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Furqan Rustam
Submission Due Date:	12/12/2024
Project Title:	Wrong-Way Vehicle Detection Using YOLOv7 for Enhanced Traffic Safety
Word Count:	XXX
Page Count:	13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Alphons Zacharia James
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Alphons Zacharia James
x23169702

1 Introduction

This Manuel provides instructions for configuring and deploying the research project "Wrong-Way Vehicle Detection Using YOLOv7 for Enhanced Traffic Safety".This helps to recreate experimental setup used for the research.The research experiments model training with three different versions of YOLO models YOLOv5,YOLOv7 and YOLOv8.The manual contains the hardware,software and cloud requirements needed for the successful recreation.

2 System Requirements

The model training part of the research is done using with the help of cloud computing and the implementation of the application developed for wrong way vehicle detection is executed locally.

2.1 Cloud Requirement

T4 GPU enabled Google Colab environment is used for model training process.

2.2 Hardware Requirements

The application part of the proposed system is executed locally in a Microsoft Windows 11 OS enabled MI Notebook Pro,the device specifications are given below .

- **Processor:** 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz
- **Installed RAM:** 16.0 GB (15.8 GB usable)
- **System Type:** 64-bit operating system, x64-based processor

2.3 Software Requirement

The following software's are required to be installed locally for the execution.

- **Python:**Version 3.11.7
- **Anaconda:**Distribution of Python with pre-installed libraries and tools.

2.4 Python Packages Required

2.4.1 'Execution of Notebook File in Google Colab'

Following libraries are used in Google-Colab for successful model training.

Library	Purpose
Open CV	Library for real-time computer vision tasks
Imgaug	Used for image augmentations
numpy	Numerical computations
google.colab.files	To upload and download from Colab

Table 1: Libraries and Their Purposes

2.4.2 Local execution of 'app.py'

The following python libraries and packages are required for successful execution of the application. These libraries are included in file "requirments.txt" along with code-artifact shared.

Package Name	Version
absl-py	2.1.0
asttokens	2.4.1
astunparse	1.6.3
cattrs	23.2.3
certifi	2024.8.30
charset-normalizer	3.4.0
colorama	0.4.6
contourpy	1.3.1
cycler	0.12.1
Cython	3.0.10
decorator	5.1.1
dnspython	2.6.1
executing	2.1.0
filelock	3.16.1
filetype	1.2.0
flatbuffers	24.3.25
fonttools	4.55.0
fsspec	2024.10.0
gast	0.5.4
google-pasta	0.2.0
grpcio	1.64.0
h5py	3.11.0
idna	3.7
imbalanced-learn	0.12.2
imutils	0.5.4
ipython	8.29.0
jedi	0.19.2

Package Name	**Version**
Jinja2	3.1.4
keras	3.3.3
kiwisolver	1.4.7
libclang	18.1.1
Markdown	3.7
MarkupSafe	3.0.2
matplotlib	3.9.2
matplotlib-inline	0.1.7
ml-dtypes	0.3.2
mpmath	1.3.0
namex	0.0.8
networkx	3.4.2
numpy	1.23.5
opencv-python	4.10.0.84
opencv-python-headless	4.10.0.84
opt-einsum	3.3.0
optree	0.11.0
packaging	24.2
pandas	2.2.3
parso	0.8.4
pillow	11.0.0
pmdarima	2.0.4
prompt_toolkit	3.0.48
protobuf	4.21.2
psutil	6.1.0
psycopg2	2.9.9
pure_eval	0.2.3
Pygments	2.18.0
pymongo	4.6.3
pyparsing	3.2.0
python-dateutil	2.9.0.post0
pytz	2024.2
PyYAML	6.0.2
requests	2.32.3
requests-cache	1.2.0
retry-requests	2.0.0
roboflow	1.1.48
scipy	1.14.1
seaborn	0.13.2
six	1.16.0
stack-data	0.6.3
sympy	1.13.1
tensorboard	2.16.2
tensorboard-data-server	0.7.2
tensorflow	2.16.1

Package Name	Version
tensorflow-intel	2.16.1
tensorflow-io-gcs-filesystem	0.31.0
termcolor	2.4.0
thop	0.1.1.post2209072238
torch	2.5.0
torchaudio	2.5.0
torchvision	0.20.0
tqdm	4.67.0
traitlets	5.14.3
typing_extensions	4.12.2
tzdata	2024.2
url-normalize	1.4.3
urllib3	2.2.3
wcwidth	0.2.13
Werkzeug	3.1.3

2.5 Steps to Install and Setup Environment

1. Install Anaconda from website `https://www.anaconda.com/download`
2. Open Anaconda prompt and create conda environment using command:

```
conda create --name env python=3.11.7
```

3. Activate the environment using command:

```
conda activate env
```

4. Run the command:

```
conda install pip
```

5. Install requirements from file requirements.txt using command:

```
pip install -r requirements.txt
```

3 CodeArtifact Folder

The contains the contents of the shared folder.The table 3 explains the contents of the folder and its purpose.

Name	Type	Description
Results model training	Folder	Contains results of model training
Saved_model	Folder	Contains best.pt loaded in app.py
violations	Folder	Violation images
Yolov7	Folder	Cloned from git to load the model in app.py
app.py	Python file	Application
dataset.ipynb	Jupyter Notebook file	Contain dataset description
dataseturl.txt	Txt file	URL of dataset
requirements.txt	Txt file	Packages for pip install
t1.mov	Video file	For Testing application
T2.mp4	Video file	For Testing application
v5model.ipynb	Jupyter Notebook file	Model training YOLOv5
v7model.ipynb	Jupyter Notebook file	Model training YOLOv7
V8model.ipynb	Jupyter Notebook file	Model training YOLOv8

Table 3: Summary of files and their descriptions.

4 Code Files

The project contains four files: three Jupyter Notebook files which are executed in Google Colab, and one Python file that can be executed locally. The details of each file are as follows:

1. `dataset.ipynb`: This file contains code to understand the dataset, its classes, and annotations.
2. `v5model.ipynb`: This file is used for model training with YOLOv5. The best weights and results are downloaded after training the model.
3. `v7model.ipynb`: This file is used for model training with YOLOv5. The best weights and results are downloaded after training the model.
4. `v8model.ipynb`: This file is used for model training with YOLOv5. The best weights and results are downloaded after training the model.
5. `app.py`: This file loads the best-trained model weights and detects violations from input videos. When running the application, the user will be asked to enter the video file name and the reference direction. The allowed reference direction is set by clicking two points on the screen, and it will display the allowed direction.

5 Data Sources

The dataset used in for model training is downloaded from Roboflow.The URL for the dataset .[1] .It contains 1961 images with 6 classes of vehicles annotated.

The video footage used for testing of the system is downloaded which are publicly available.Both videos t1.mov [2] and t2.mp4 [3] are added with the code folder shared.

[1]Dataset url: `https://universe.roboflow.com/aliyahhalim/vehicle-detection-q8q4n`
[2]t1.mov:`https://www.pexels.com/video/a-double-lane-highway-for-road-travelers-4261446/`
[3]t2.mp4:`https://www.videezy.com/travel/5651-cars-pass-under-an-overpass`

6 Code Execution

The jupyter notebook files can be executed in Google Colab (https://colab.google/).you can upload the 'ipynb' files and execute the code directly .The dataset will be downloaded to the cloud resource directly and processed.There is no need to upload dataset. The python application 'app.py' file can be executed locally in the activated Anaconda python environment which is discussed before.

1. Change directory to the saved unzipped code-artifact folder using the `cd` command:

```
cd "path"
```

2. Run the code using the following command:

```
python app.py
```

One the application is up and running user will be prompted to give file name give name of the video files with format of video(eg: t1.mov) and then user will be prompted to enter reference direction.The direction has to be entered by clicking two points on screen which has to align with the vehicle movement direction.If the first click is considered as A and second is considered as B.The allowed movement direction is from A to B and the opposite direction is considered as wrong way.

7 Codes

The file dataset.ipynb contains the exploration of dataset before training .File is executed on Google Colab and dataset is loaded directly into the cloud resource and understanding of the dataset and the augmentation which is done on each model training code.

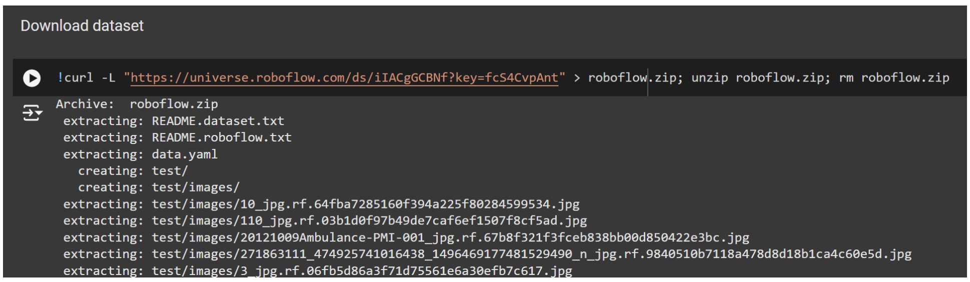

Figure 1: Dataset downloading and unzipping .

7.1 Augmentation

Using imgaug library augmentation for the train image set of the dataset is done.It includes making changes to the dataset images along with updating the label file.For label file initial label file is parsed and after making sufficient changes it is converted back to YOLO format.

```python
#Function to count images
def count_images_in_folder(folder_path):
    image_extensions = {'.jpg', '.jpeg', '.png', '.gif', '.bmp', '.tiff'}
    image_count = 0
    for filename in os.listdir(folder_path):
        file_extension = os.path.splitext(filename)[1].lower()
        if file_extension in image_extensions:
            image_count += 1
    return image_count
folder_path = r'/content/train/images'
folder_path1 = r'/content/valid/images'
folder_path2 = r'/content/test/images'
image_count = count_images_in_folder(folder_path)
image_count =image_count+count_images_in_folder(folder_path1)
image_count =image_count+count_images_in_folder(folder_path2)
print(f"Number of images in the folder: {image_count}")
```

```
Number of images in the folder: 1961
```

Figure 2: Code for counting number of images .

Figure 3: Sample data from Dataset.

```python
augmentation = iaa.Sequential([
    iaa.Multiply((0.8, 1.2), per_channel=0.2),    # Random brightness change
    iaa.LinearContrast((0.75, 1.5)),              # Random contrast change
    iaa.AddToHueAndSaturation((-20, 20)),         # Random hue and saturation change
    iaa.Affine(scale=(0.8, 1.2),                  # Scale images
               translate_percent=(-0.2, 0.2),     # Random translations
               rotate=(-30, 30),                  # Rotate images between -30 to 30 degrees
               shear=(-10, 10)),                  # Shear images between -10 to 10 degrees
    iaa.GaussianBlur(sigma=(0, 1.5)),             # Gaussian blur
    iaa.MotionBlur(k=5),                          # Motion blur
    iaa.AdditiveGaussianNoise(scale=(10, 50)),    # Add Gaussian noise
    iaa.JpegCompression(compression=(70, 99)),    # Simulate JPEG compression
])
```

Figure 4: Augmentations applied to images.

```python
# Function to parse YOLO label file
def parse_yolo_label(label_path, img_width, img_height):
    boxes = []
    with open(label_path, 'r') as file:
        for line_number, line in enumerate(file, start=1):
            parts = line.strip().split()
            if len(parts) != 5:
                continue  # Skip lines
            try:
                class_id, x_center, y_center, width, height = map(float, parts)
                x_center *= img_width
                y_center *= img_height
                width *= img_width
                height *= img_height
                x_min = x_center - (width / 2)
                y_min = y_center - (height / 2)
                x_max = x_center + (width / 2)
                y_max = y_center + (height / 2)
                boxes.append((int(class_id), x_min, y_min, x_max, y_max))
            except ValueError as e:
                continue  # Skip lines with invalid numbers
    return boxes
```

Figure 5: Parsing label file for updating labels after augmentation

```python
# Convert bounding boxes to YOLO format after augmentation
def convert_to_yolo_format(bbs, img_width, img_height):
    yolo_boxes = []
    for bb in bbs:
        x_center = ((bb.x1 + bb.x2) / 2) / img_width
        y_center = ((bb.y1 + bb.y2) / 2) / img_height
        width = (bb.x2 - bb.x1) / img_width
        height = (bb.y2 - bb.y1) / img_height
        yolo_boxes.append((x_center, y_center, width, height))
    return yolo_boxes
```

Figure 6: Converting Back to YOLO Format

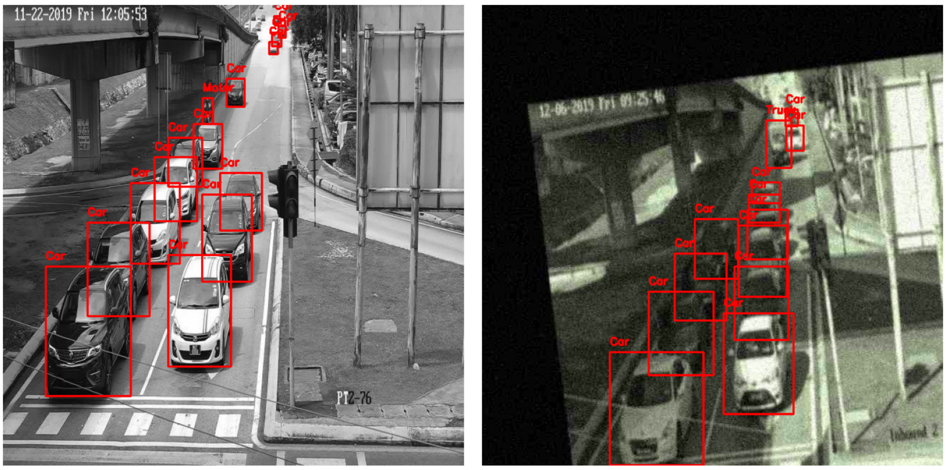

Figure 7: Sample image after augmentation

7.2 Model Training

The model training for each Model versions of YOLO are also done in cloud resource with T4 GPU.The ’v5model.ipynb’,’v7model.ipynb’ and ’v8model.ipynb’ does the model training and after sucessful completion of execution the results of the training is downloaded from cloud system as zip file and saved for evaluation

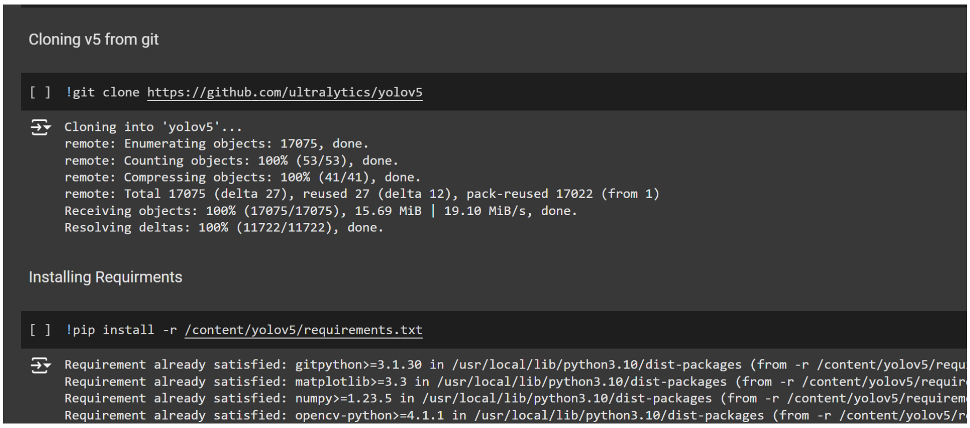

Figure 8: Cloning YOLOv5 from original repository

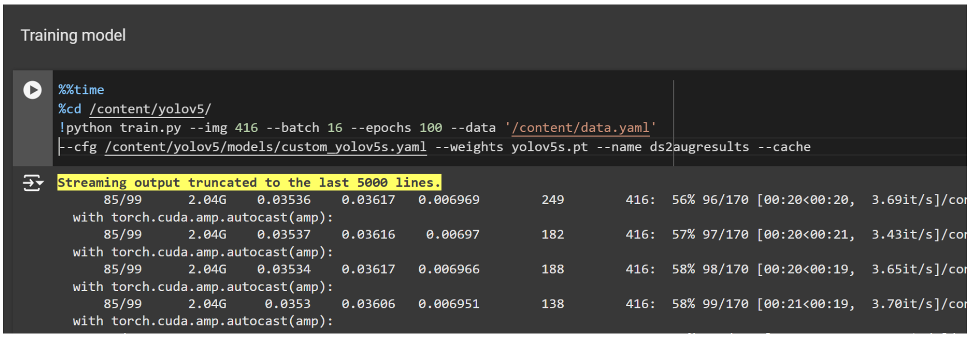

Figure 9: Sample image after augmentation

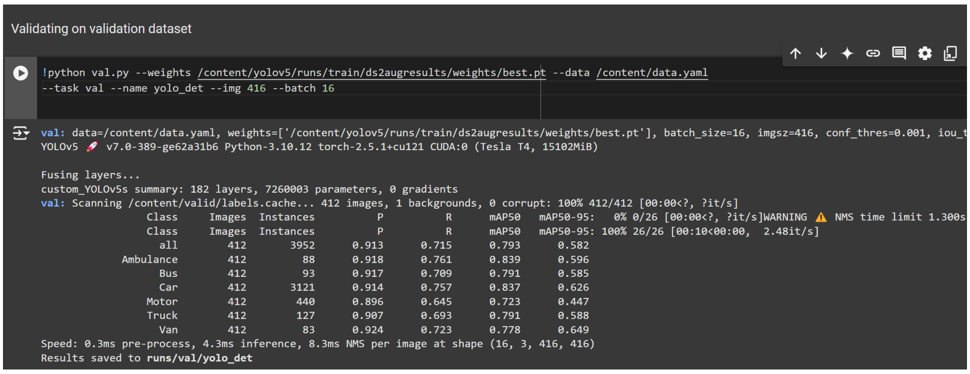

Figure 10: Model Evaluation Code YOLOv5

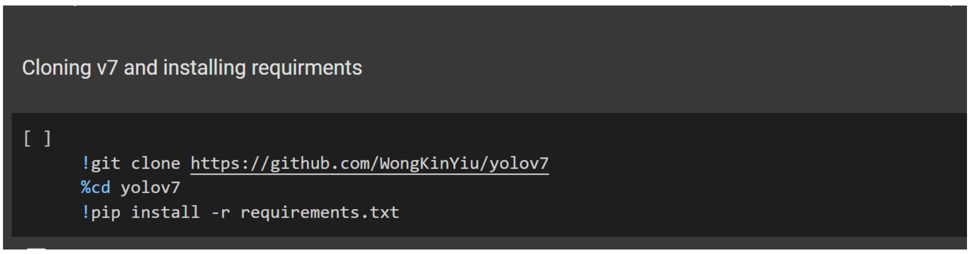

Figure 11: Cloning v7 from git repository

Training dataset

```
[ ]
    %cd /content/yolov7
    !python train.py --batch 16 --img 416 --epochs 100
    --data /content/yolov7/vehicle-detection-7/data.yaml --weights 'yolov7_training.pt' --device 0 --project "/content/yolov7/runs/train" --name "v750"

    /content/yolov7
    2024-12-09 17:24:46.551630: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to register cuFFT factory: Attempting to register factory
    2024-12-09 17:24:46.571095: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to register cuDNN factory: Attempting to register facto
    2024-12-09 17:24:46.576990: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to register cuBLAS factory: Attempting to register fac
    2024-12-09 17:24:46.590926: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in
    To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
    2024-12-09 17:24:47.632660: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```

Figure 12: Model training Code YOLOv7

Evalating model

```
#model evaluation
!python test.py --img-size 416 --batch 16 --weights /content/yolov7/runs/train/v750/weights/best.pt
--data /content/yolov7/vehicle-detection-7/data.yaml --device 0 --name validresult

Namespace(weights=['/content/yolov7/runs/train/v750/weights/best.pt'], data='/content/yolov7/vehicle-detection-7
YOLOR 🚀 v0.1-128-ga207844 torch 2.5.1+cu121 CUDA:0 (Tesla T4, 15102.0625MB)
```

Figure 13: Code for validating using best.pt model after training

```
  cache, exists = torch.load(cache_path), True  # load
val: Scanning 'vehicle-detection-7/valid/labels.cache' images and labels... 412 found, 0 missing, 1 empt
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 26/26
                 all         412        3952       0.914       0.815       0.876       0.717
           Ambulance         412          88       0.929       0.894       0.929       0.768
                 Bus         412          93       0.875       0.828        0.86       0.704
                 Car         412        3121       0.898       0.835        0.89       0.713
               Motor         412         440       0.904        0.75       0.818       0.581
               Truck         412         127       0.962       0.787       0.879       0.722
                 Van         412          83       0.919       0.795       0.877       0.811
Speed: 6.6/3.7/10.3 ms inference/NMS/total per 416x416 image at batch-size 16
Results saved to runs/test/validresult
```

Figure 14: Screenshot of validation result of YOLOv7

Installing ultralytics and importing

```
# Pip install method (recommended)

!pip install ultralytics==8.2.103 -q

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

Ultralytics YOLOv8.2.103 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 32.6/235.7 GB disk)
```

Figure 15: importing YOLOv8 to python Environment

Training custom dataset

```
[ ] !yolo task=detect mode=train model=yolov8s.pt data=/datasets/vehicle-detection-7/data.yaml epochs=100 imgsz=416 batch=16 plots=True

    Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8s.pt to 'yolov8s.pt'...
    100% 21.5M/21.5M [00:00<00:00, 132MB/s]
    New https://pypi.org/project/ultralytics/8.3.48 available 😃 Update with 'pip install -U ultralytics'
    Ultralytics YOLOv8.2.103 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
    engine/trainer: task=detect, mode=train, model=yolov8s.pt, data=/datasets/vehicle-detection-7/data.yaml, epochs=100, time=None, patience=100, batch=16, im
    Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
    100% 755k/755k [00:00<00:00, 38.6MB/s]
    Overriding model.yaml nc=80 with nc=6
```

Figure 16: SModel training for YOLOv8

Validating with validation dataset

```
!yolo task=detect \
mode=val \
model=/datasets/runs/detect/train/weights/best.pt\
data=/datasets/vehicle-detection-7/data.yaml
```

```
Ultralytics YOLOv8.2.103 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11,127,906 parameters, 0 gradients, 28.4 GFLOPs
val: Scanning /datasets/vehicle-detection-7/valid/labels.cache... 412 images, 1 backgrounds, 0 corrupt: 100% 412/412 [00:00<?, ?it/s]
WARNING ⚠ Box and segment counts should be equal, but got len(segments) = 240, len(boxes) = 3952. To resolve this only boxes will be used and all se
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 26/26 [00:07<00:00,  3.43it/s]
                   all        412       3952      0.911      0.807      0.866      0.708
             Ambulance         82         88      0.885      0.875      0.938      0.749
                   Bus         92         93      0.824      0.849      0.868       0.71
                   Car        342       3121      0.913      0.814      0.897      0.734
                 Motor        252        440      0.887      0.709       0.78      0.559
                 Truck        114        127      0.978      0.787      0.872      0.714
                   Van         71         83       0.98      0.807       0.84      0.781
Speed: 1.0ms preprocess, 4.4ms inference, 0.0ms loss, 3.0ms postprocess per image
Results saved to runs/detect/val
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Figure 17: Model Validation YOLOv8

7.3 Python File (app.py)

The app.py contains the detection part of the project and will load the model and process videos frame by frame and detect violations .Few snapshots of the application is attched.

```
DESIRED_HEIGHT = 720

#loading yolov7 model
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = torch.hub.load('yolov7', 'custom', f'./saved_model/best.pt', source='local')  # loading the model


video_path = input("Please enter the path to the video file: ")
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    print(f"Error: Could not open video file {video_path}")
    exit()
```

Figure 18: Loding the saved YOLOv7 model 'best.pt'

The 'best.pt' five which is saved in the folder saved_model is the same 'best.pt' which is available in`resultmodeltraining\v7100\train` ,obtained after training of model.

```
# Function to capture mouse clicks
def select_points(event, x, y, flags, param):
    global points, reference_direction
    if event == cv2.EVENT_LBUTTONDOWN:
        points.append((x, y))
        if len(points) == 2:
            reference_vector = (points[1][0] - points[0][0], points[0][1] - points[1][1])
            norm = np.linalg.norm(reference_vector)
            if norm != 0:
                reference_direction = np.array(reference_vector) / norm  # Normalize the reference direction
            print(f"Reference Direction Set: {reference_direction}")
```

Figure 19: Code to get mouse click from user for reference direction

```python
                    # Process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame, (DESIRED_WIDTH, DESIRED_HEIGHT))

                    # Inference using YOLOv7
    results = model(frame)
    detections = results.xyxy[0].numpy()
                # Prepare the final detections after filtering confidence
    final_detections = []
    for detection in detections:
        x1, y1, x2, y2, confidence, class_id = detection[:6]
        if confidence > 0.5 and class_id in [0,1,2,4,5]:
            final_detections.append(((x1 + x2) // 2, (y1 + y2) // 2, int(x1), int(y1), int(x2), int(y2), int(class_id)))

    #Assign unique IDs to detected vehicles
    updated_vehicle_ids = {}
    for detection in final_detections:
        center_x, center_y, x1, y1, x2, y2, class_id = detection

        # Find the closest previous centroid
        closest_id = None
        min_distance = float('inf')
        for vehicle_id, prev_center in vehicle_ids.items():
            distance = np.linalg.norm(np.array([center_x, center_y]) - np.array(prev_center))
            if distance < min_distance and distance < 50:  # Threshold to match centroids
                closest_id = vehicle_id
                min_distance = distance
```

Figure 20: Processing video Frame by Frame

8 Results

The results both train and validation of the model training done using respective YOLO models are saved in 'results model training' folder.These are directly downloaded from the Google Colab after execution of the model training code.It contains figures precision-recall curve,precision-confidence curve ,Recall confidence curve,F1-Confidence curve and normalized confusion matrix. Also the weights folder contains the saved model with the best result with name 'best.pt'.It contains both train results and validation results.

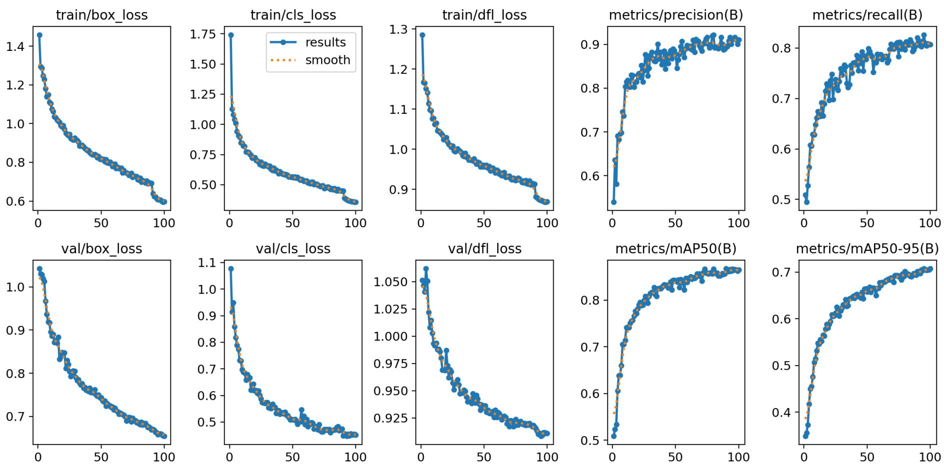

Figure 21: Training and Validation Metric Plotted over epochs YOLOV8

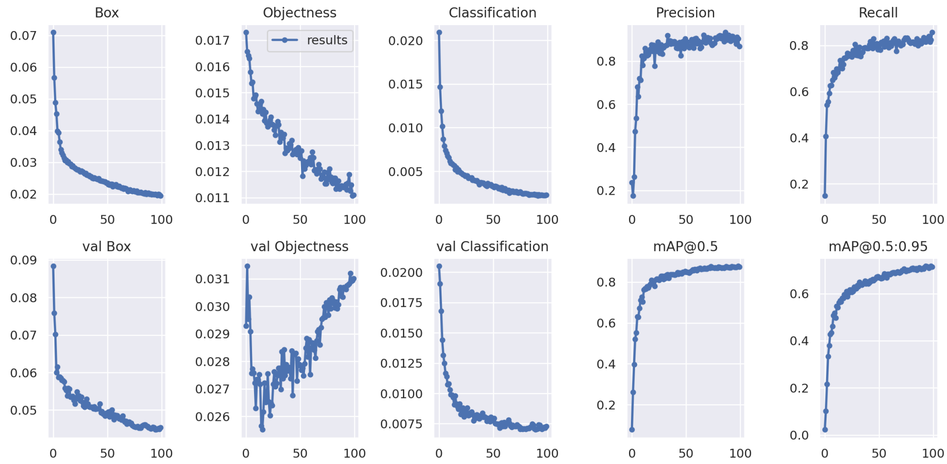

Figure 22: Training and Validation Metric Plotted over epochs YOLOV7

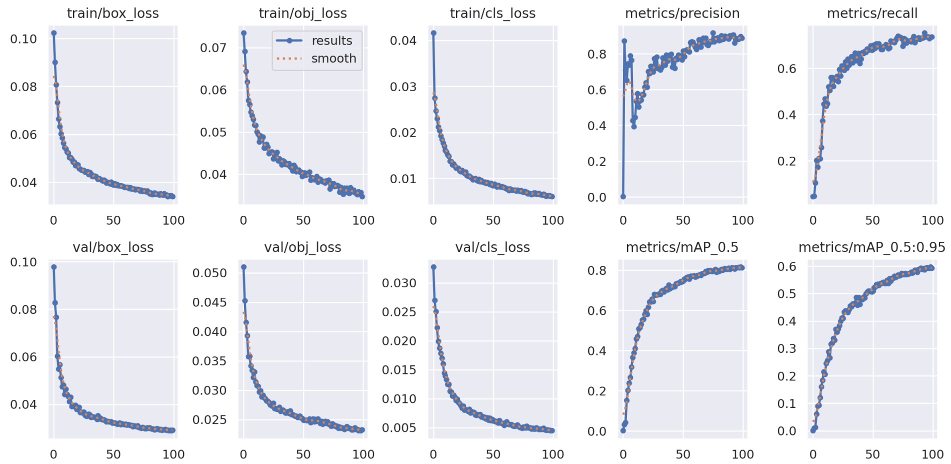

Figure 23: Training and Validation Metric Plotted over epochs YOLOV5