Internship Project Report

Intern name: Smruthi

Project Manager: Narra Anitha

Project Mentor: Naresh Adepu

Project Objective

To gain hands-on experience in computer vision and deep learning techniques through practical implementation and weekly learning modules.

Project Title

1. Image classification: Classify Images of Pedestrian and Auto
2. Object Detection: Detect Pedestrian and Auto

Project Scope

The internship explores basic to advanced concepts in image processing, neural networks, and object detection, concluding with a final implementation project.

Deliverables

Weekly learning summaries, implementation tasks, and a final report detailing the internship outcomes.

Timeline

Duration: 5 Weeks

From 19.05.25 to 20.06.25

Tools and Technologies Used

Python, OpenCV, TensorFlow/Keras, ARAS, CNNs, YOLO, and other relevant libraries and frameworks.

Week 1: Understanding ARAS and Learning ML Fundamentals

During the first week, the following ARAS Features were researched and studied:

1. Blind Spot Detection (BSD)
2. Lane Departure Warning (LDW) and Lane Keeping Assistance (LKA)
3. Lane Change Assist (LCA)
4. Forward Collision Warning (FCW)
5. Autonomous Emergency Braking (AEB)
6. Rear Collision Warning (RCW)
7. Rear Cross Traffic Alert (RCTA)
8. Traffic Sign Recognition (TSR) and Traffic Light Detection
9. Rider Drowsiness / Fatigue Monitoring System
10. Traction Control System (TCS)
11. Anti-lock Braking System (ABS) and Cornering ABS
12. Hill Hold Control (HHC)
13. Adaptive Headlights
14. V2X Communication
15. Adaptive Cruise Control (ACC)

The focus was on understanding the functionality, implementation, challenges encountered in implementation, and various other aspects of Advanced Rider Assistance Systems (ARAS).

To effectively carry out the projects, it was essential to have a foundational understanding of Machine Learning (ML), Neural networks (NN), and Convolutional Neural Networks (CNN). Therefore, I also focused on learning the fundamentals of ML in the first week.

Topics Covered:

- Structural Hierarchy in Artificial Intelligence
- Types ML systems:

  a) Supervised learning

  - Regression
  - classification: binary classification, multi-class classification

  b) Unsupervised learning

  c) Reinforcement learning

  d) Semi-Supervised learning

Week 2: Neural Networks and CNNs

During the second week, I studied the architecture, working, and fundamental concepts of neural networks, with a focus on Feedforward Neural Networks, Multi-Layer Perceptrons (MLPs), and Convolutional Neural Networks (CNNs)

Key Concepts Covered:

1. Neural Networks (NN)
    - Network architecture: input, hidden, and output layers
    - Forward propagation mechanics: parameterization by weights and biases, computation of weighted sums
    - Backpropagation
    - Hyperparameter tuning learning rate, batch size, number of epochs
    - Activation functions: Tanh, Sigmoid, ReLU, Leaky ReLU, SoftMax
    - Loss functions: regression losses (MSE, MAE, Huber), classification losses (Binary and Categorical Cross-Entropy)
2. Explored CNNs and specifically the LeNet-5 architecture which included:

    LeNet-5 by Yann LeCun is a CNN for image classification:
    Layers:
    - Convolutional layers,
    - Average pooling layers (max and average pooling)
    - Fully connected layers

    Architecture:
    - Consists of seven layers: two convolutional, two average pooling, one flattening convolutional, two fully connected
    - Activation function used is tanh for all layers except the output layer which uses Softmax
3. Learned how to develop a convolutional neural network based on LeNet5 architecture using Keras

Implementation steps:

1. Data collection

2. Data processing: split into training, validation, and test sets,

   normalization, resizing, one-hot encoding

3. Building the LeNet-5 model
4. Compiling the model
5. Training the model
6. Evaluating on test dataset

Week 3: Image Classification with CNN

The objective for Week 3 was to build a binary image classification model to distinguish between two classes: Pedestrian and Auto-rickshaw.

1. Setup:
Necessary tools and modules installed: Python, Anaconda, TensorFlow, PyTorch, and LabelMe.

These were used for data preprocessing, annotation, model building, and evaluation.

2. Data Collection:
Images were collected using Google Image Search and Kaggle datasets.

Indian roads pedestrian - Google Search

Auto rickshaw in india - Google Search

Pedestrians Dataset

IIT Delhi Campus Pedestrian Dataset (Detection)

3. Implementation using Keras (TensorFlow):

Hyperparameters:

- Image Size: 256x256
- Batch Size: 32
- Epochs: 15
- Optimizer: Adam

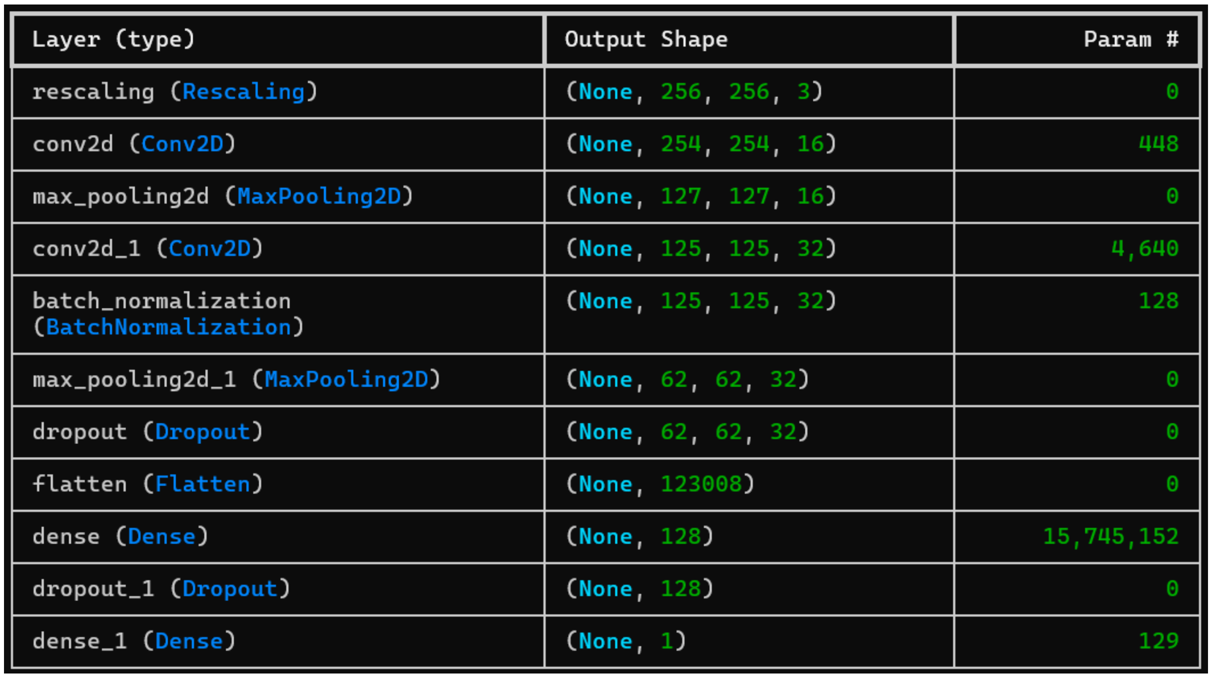
- Loss Function: Binary Cross-Entropy

The dataset was split in the ratio of 70:20:10 for training, validation, and testing respectively.

Model Summary:

Built a Custom model:

After experimenting with multiple model architectures, various hyperparameters, and layer configurations, this model configuration provided the best results in terms of accuracy and general performance:

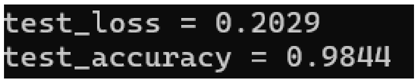


Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 256, 256, 3)	0
conv2d (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 125, 125, 32)	4,640
batch_normalization (BatchNormalization)	(None, 125, 125, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
dropout (Dropout)	(None, 62, 62, 32)	0
flatten (Flatten)	(None, 123008)	0
dense (Dense)	(None, 128)	15,745,152
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

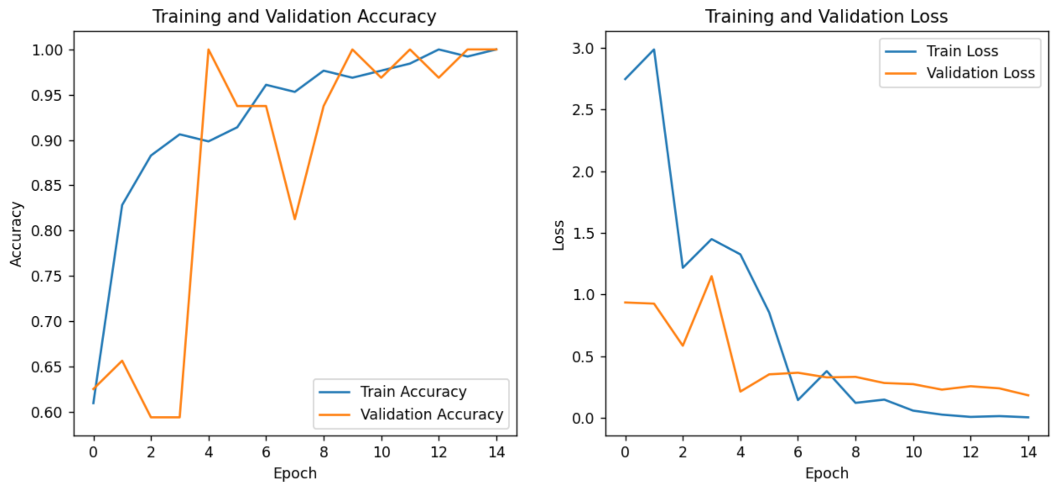
Model size = 184 MB

Test Results:

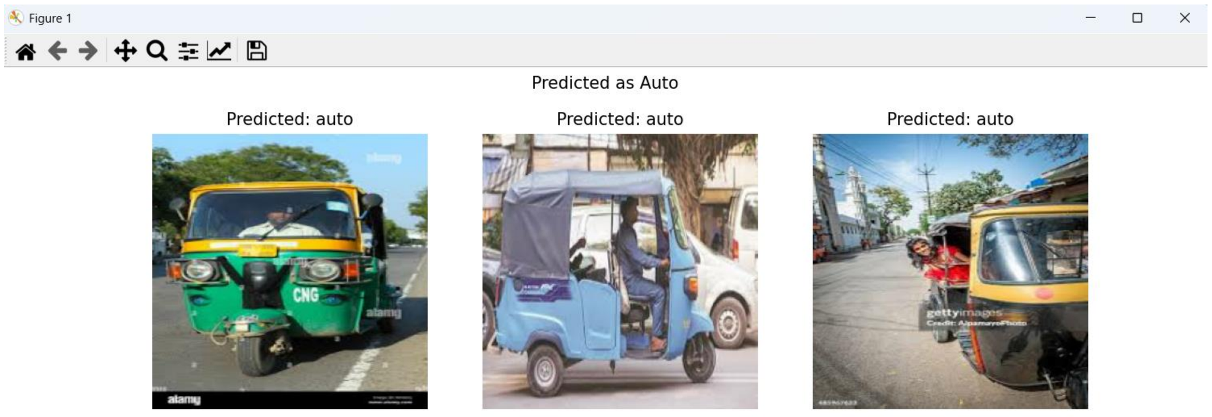
On testing the trained model on validation and test dataset, the following results were obtained:

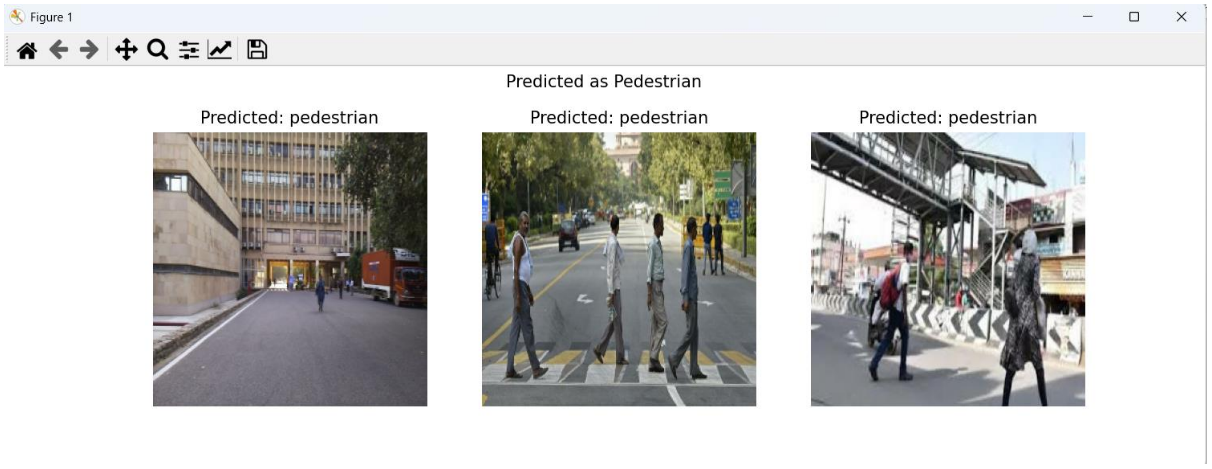


```
test_loss = 0.2029
test_accuracy = 0.9844
```



OUTPUT





To gain understanding of the implementation process using PyTorch, the same model was reimplemented using the PyTorch framework.

Week 4: Object Detection – Understanding concepts and training model

In the fourth week, the objective was to implement an object detection model. Before practical implementation, I focused on understanding the necessary concepts required for the task.

Below is a summary of the topics covered:

- Dataset Annotation:
  Learned how to create custom datasets using the LabelMe tool for manually annotating images with bounding boxes. I also assisted in annotating a custom dataset for another project.
- Object Detection Fundamentals:
  Studied the two main categories of object detection algorithms:
  - Two-Stage Detectors:
    These algorithms first generate region proposals and then classify them. They offer higher accuracy but are slower.
    Examples: R-CNN (Region based CNN), Fast R-CNN, Faster R-CNN, Mask R-CNN, G-RCNN.
  - One-Stage Detectors:
    These models detect and classify objects in a single step, offering real-time performance.
    Examples: YOLO (You Only Look Once), SSD (Single shot detection), RetinaNet, YOLOR (You Only Learn One Representation).
- YOLO:
  Focused on understanding the YOLO architecture, which is widely used for fast object detection tasks and the implementation: Feature extraction, grid division, bounding box prediction, class prediction, and post-processing.

Implementation:

A pre-trained model - Yolo11n was used for object detection task:

Yolo11n is a high-speed object detection model trained on datasets such as COCO, which contains a wide range of everyday object categories including people and vehicles. Yolo11 is highly suitable for real-time and small object detection tasks.

Hence, I used Yolo11n to train on a small custom dataset of 516 images.

Yolo11n Model Summary:

```
YOLO11n summary: 181 layers, 2,590,230 parameters, 0 gradients, 6.4 GFLOPs
```

1. Dataset collection and Annotation:

- Images were collected from Google Images and Kaggle.

  Indian roads pedestrian - Google Search

  Auto rickshaw in india - Google Search

  Pedestrians Dataset

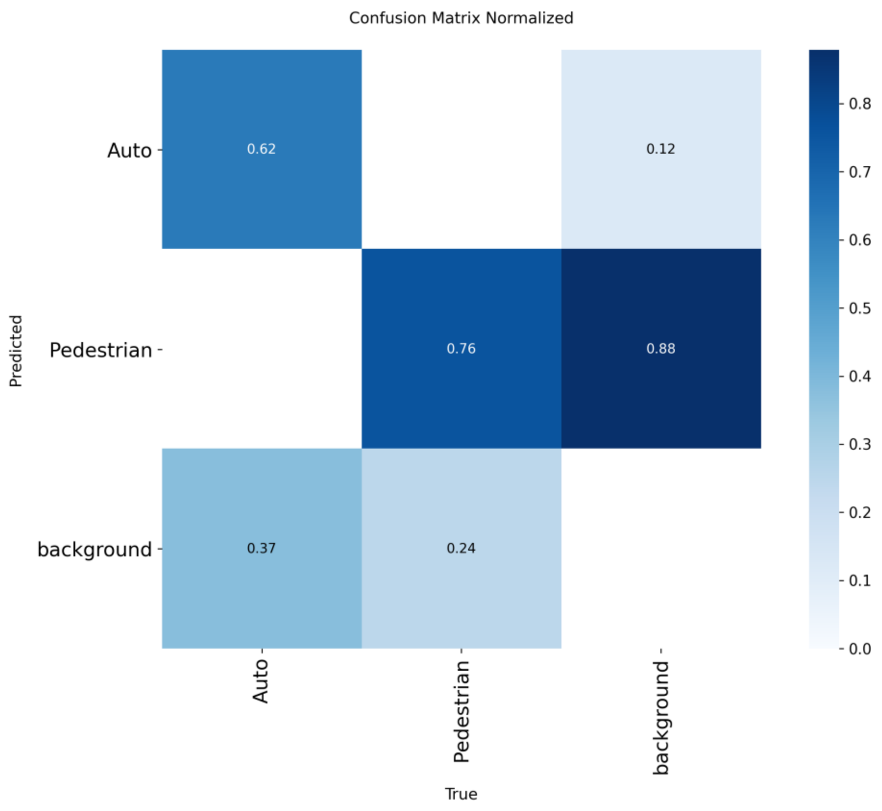
  IIT Delhi Campus Pedestrian Dataset (Detection)

- Annotation was performed using Roboflow.
- Classes were labeled as pedestrian and auto
- Datasets were split into 3 – training, validation and test dataset in the ratio 70:20:10 respectively.

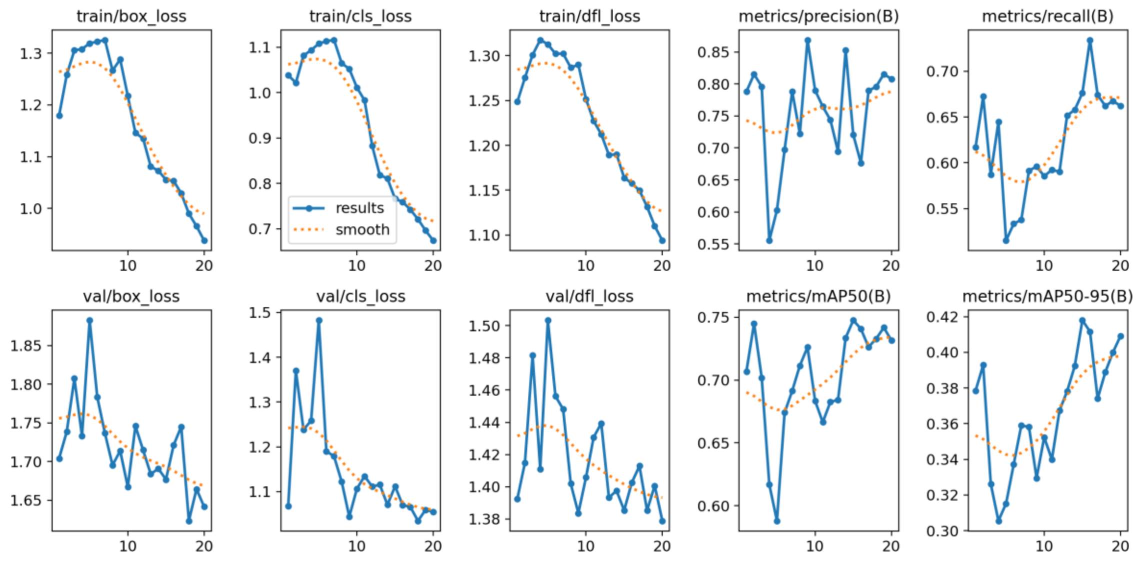
2. Training

After training for 90 epochs, the following results were observed:

- Confusion Matrix:

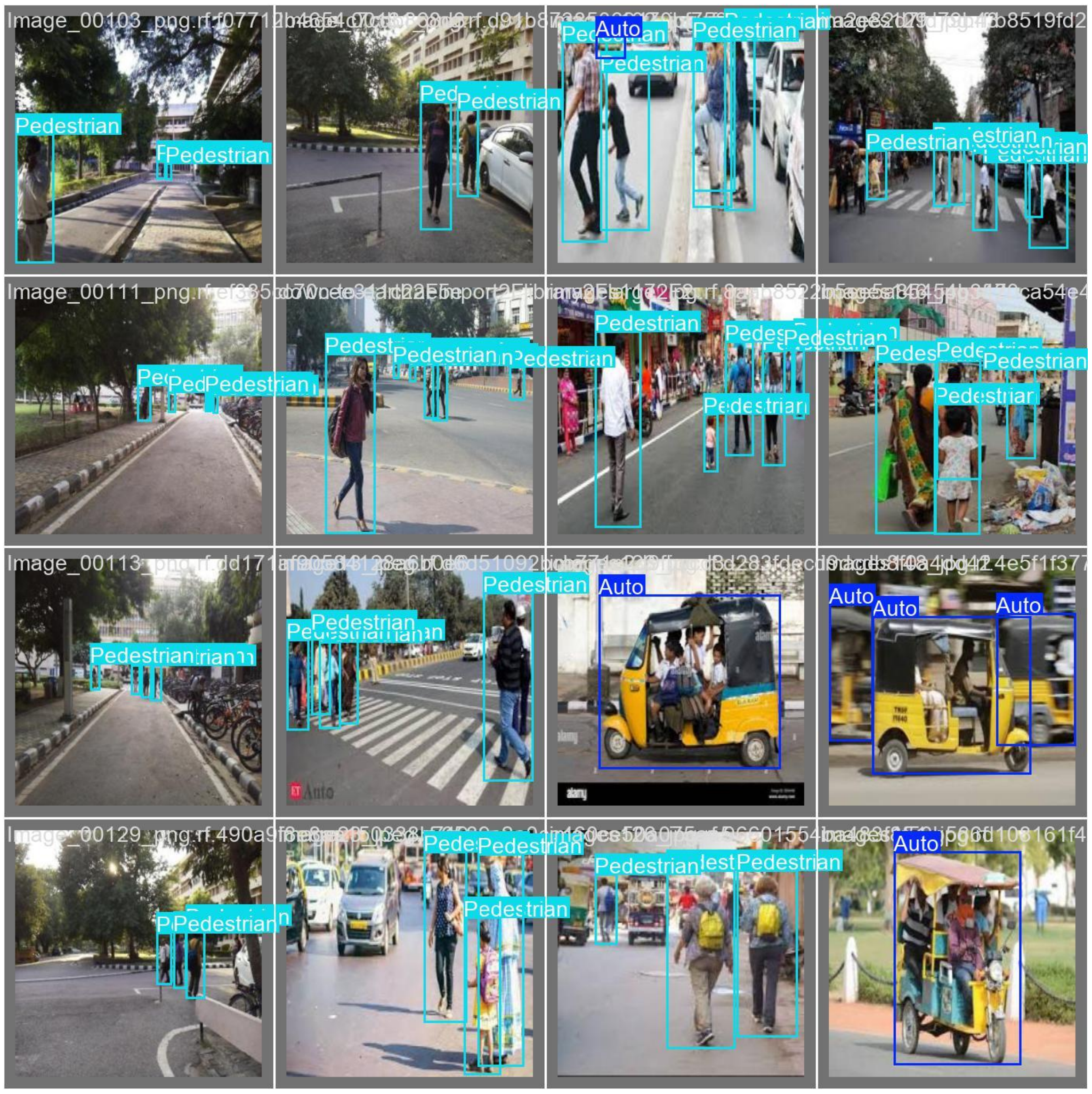


- Loss curves, Precision, Recall and mAP:

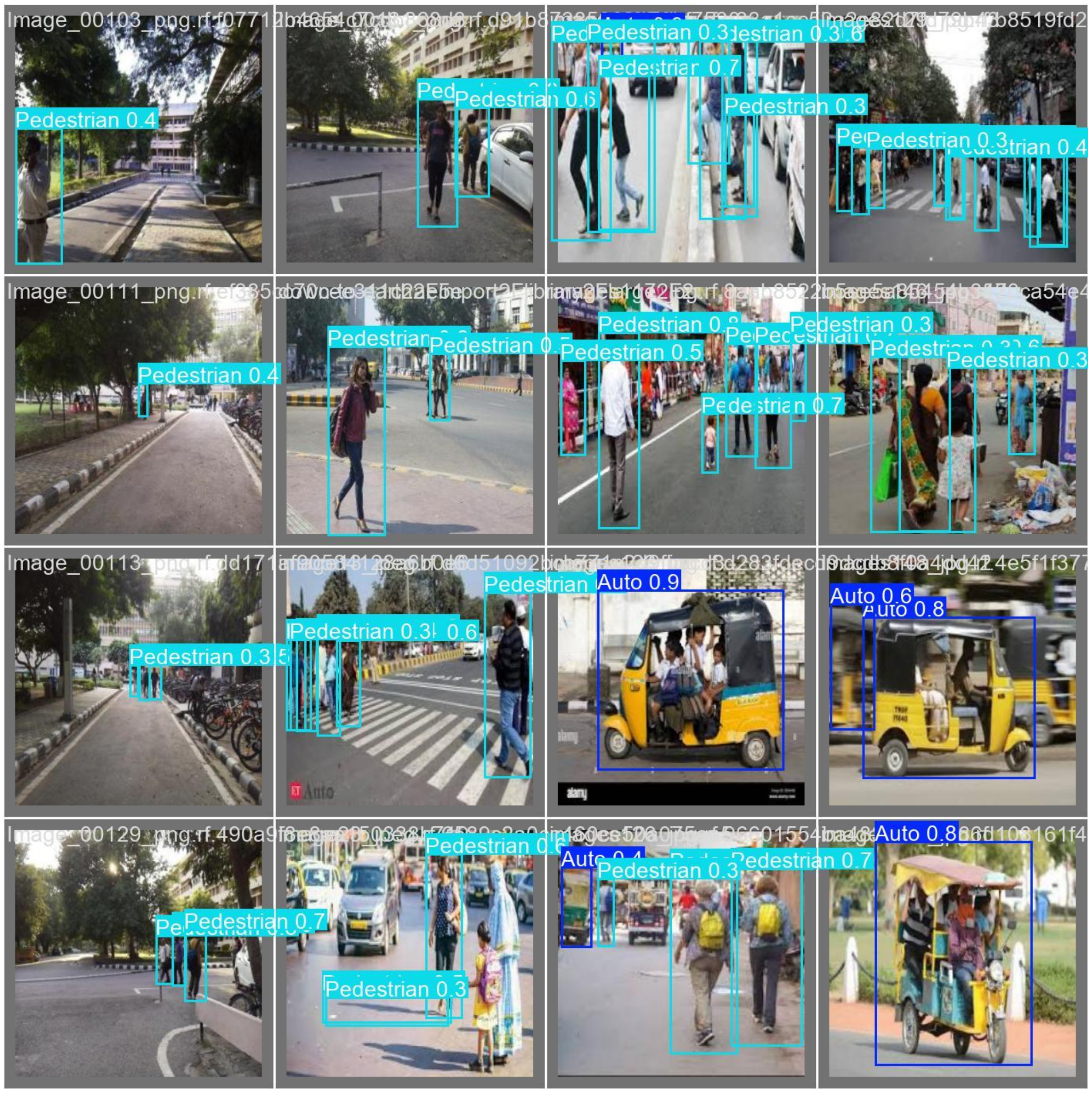


- Predicted Results of validation carried out during the training

  Labelled Images from validation set:



Predictions made by the model:



- Model size = 5300 KB

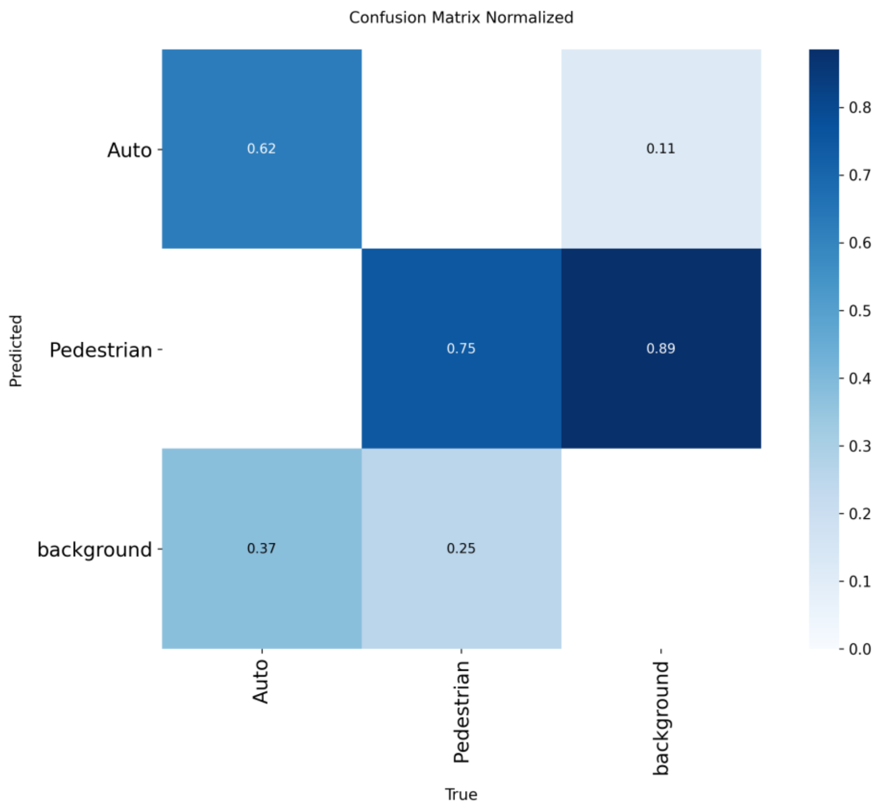
Week 5: Object detection: Validation and Prediction, and Drafting of Final Report

3. Validation

On Validation, the following results were obtained:

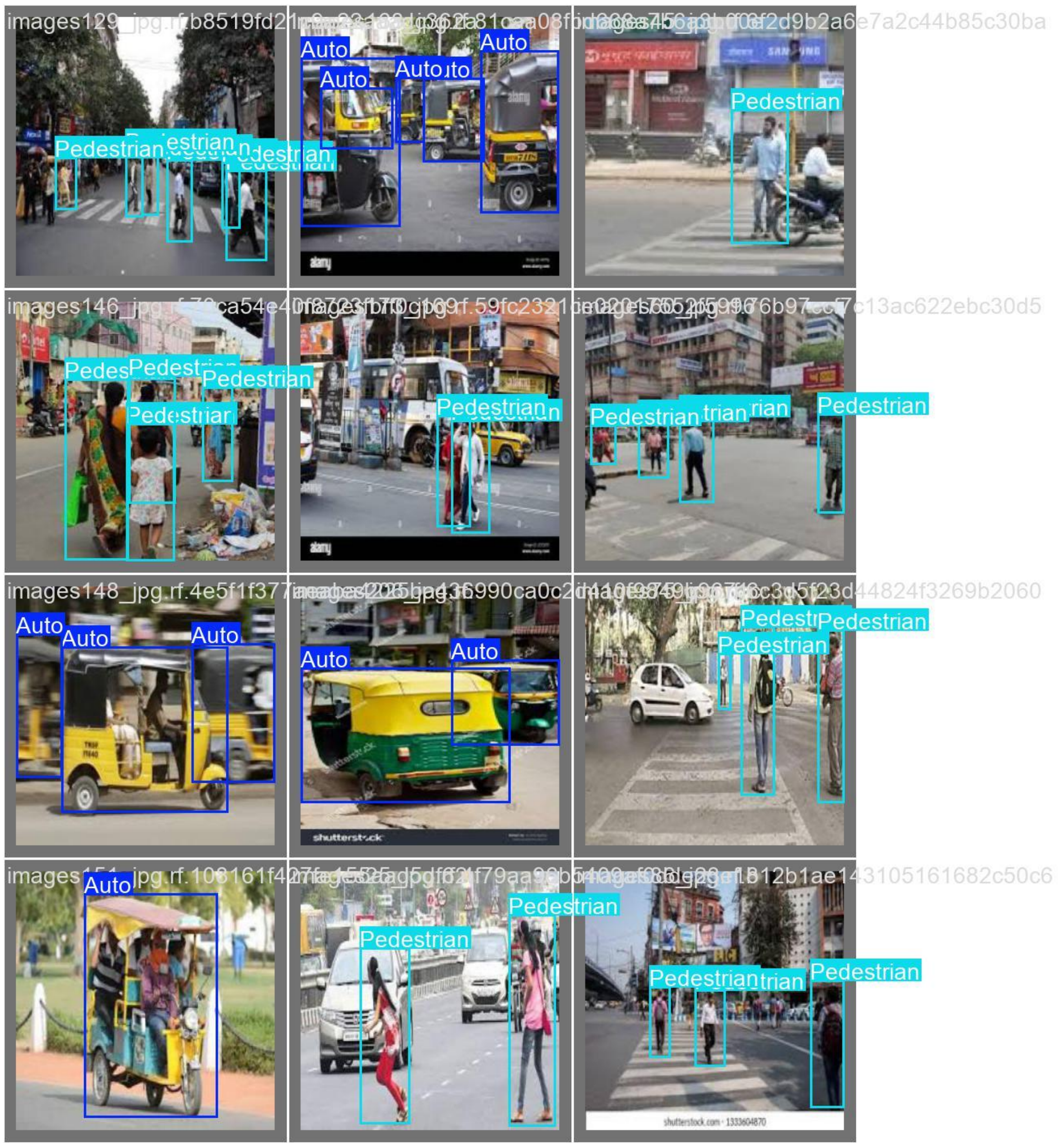
- Confusion Matrix:

(Slight variations when compared to the validation done during training process)

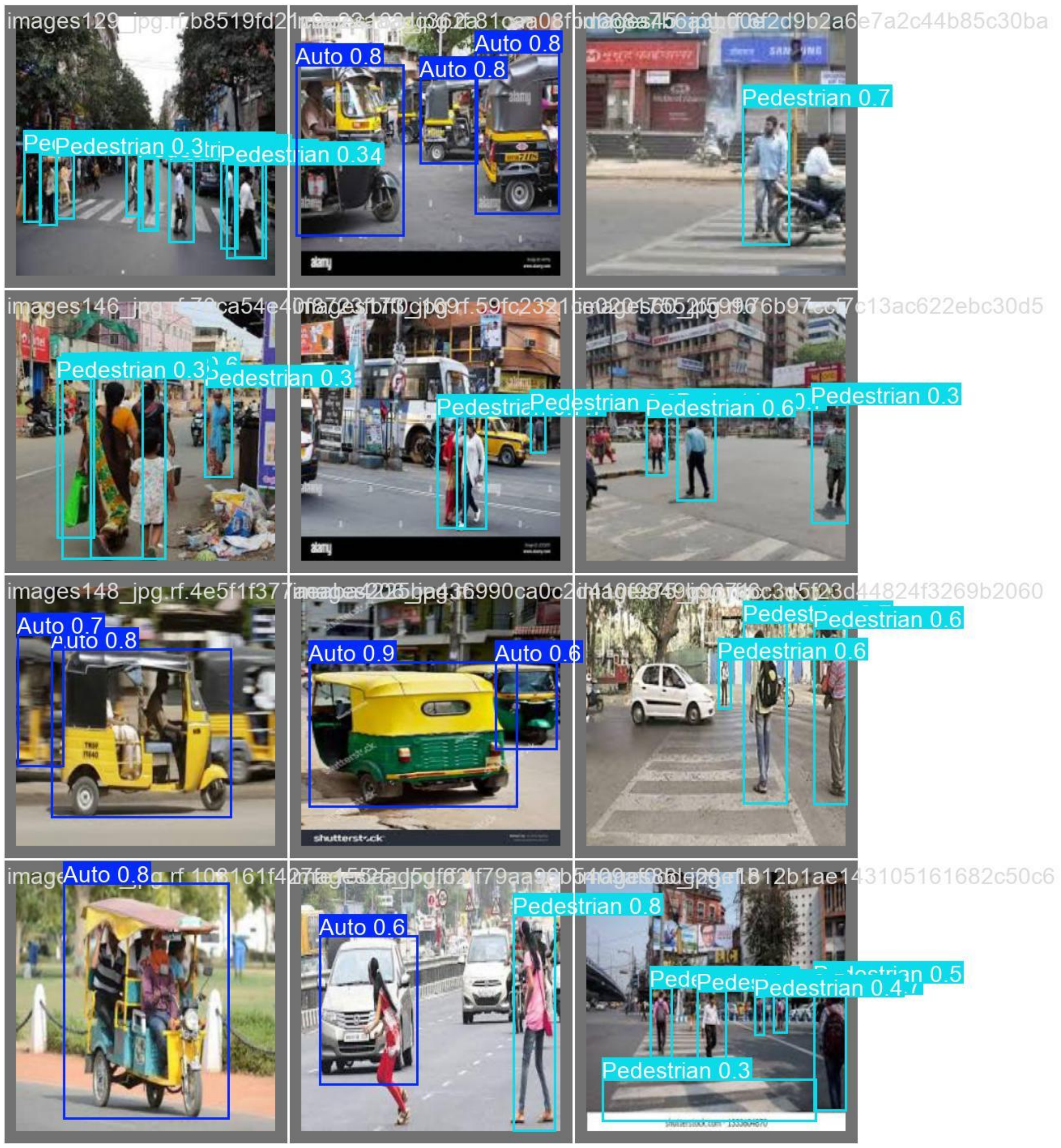


- Validation results:

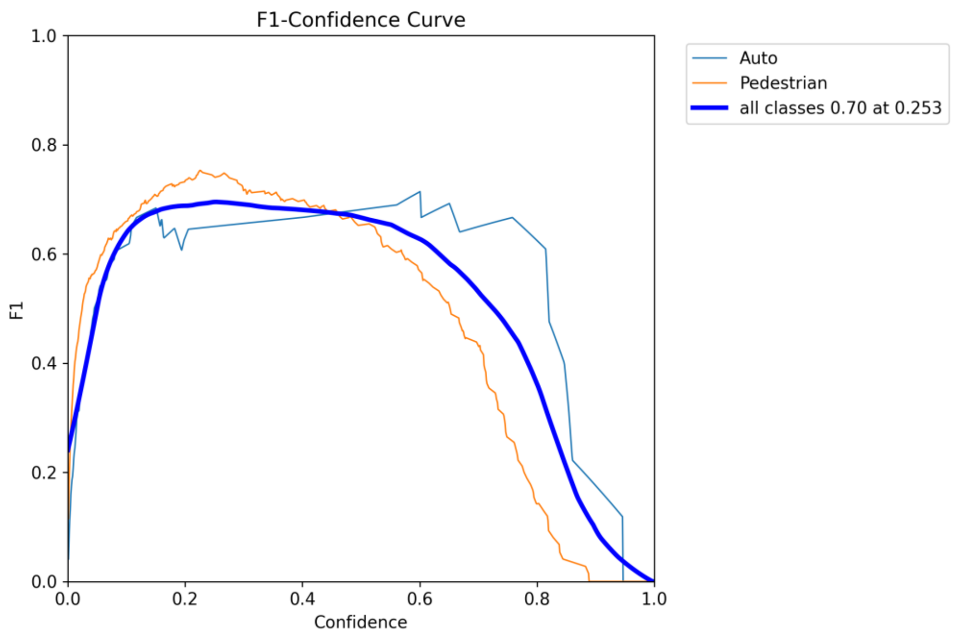
  Labelled images from validation set:



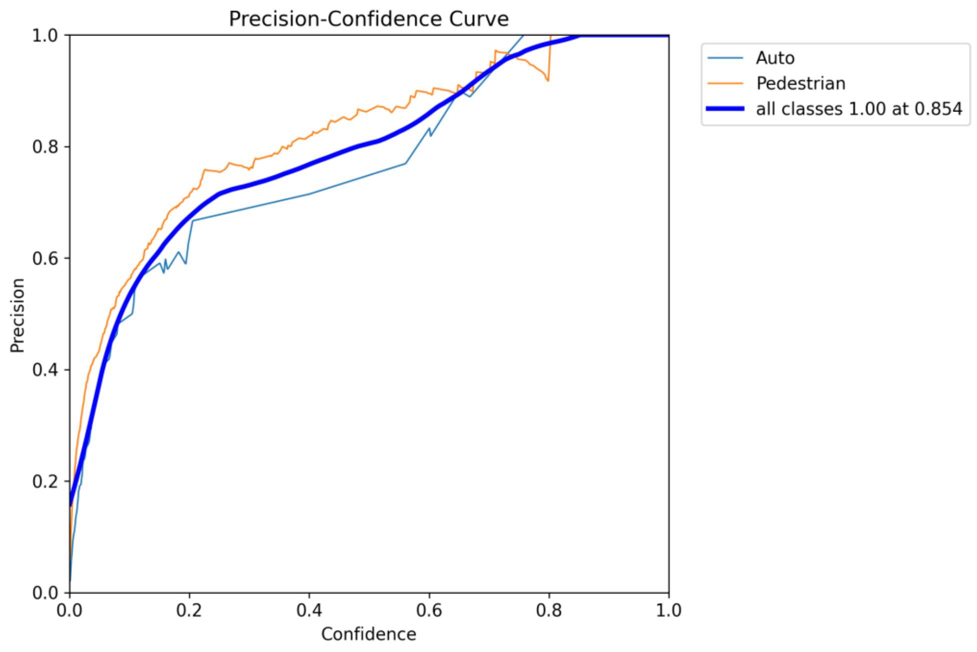
Predictions made by the model:


Auto 0.8
Auto 0.8
Auto 0.8
Pedestrian 0.7
Pedestrian 0.3
Pedestrian 0.34
Pedestrian 0.3
Pedestrian 0.6
Pedestrian 0.3
Auto 0.7
Auto 0.8
Auto 0.9
Auto 0.6
Pedestrian 0.6
Pedestrian 0.6
Auto 0.8
Auto 0.6
Pedestrian 0.8
Pedestrian 0.5
Pedestrian 0.3

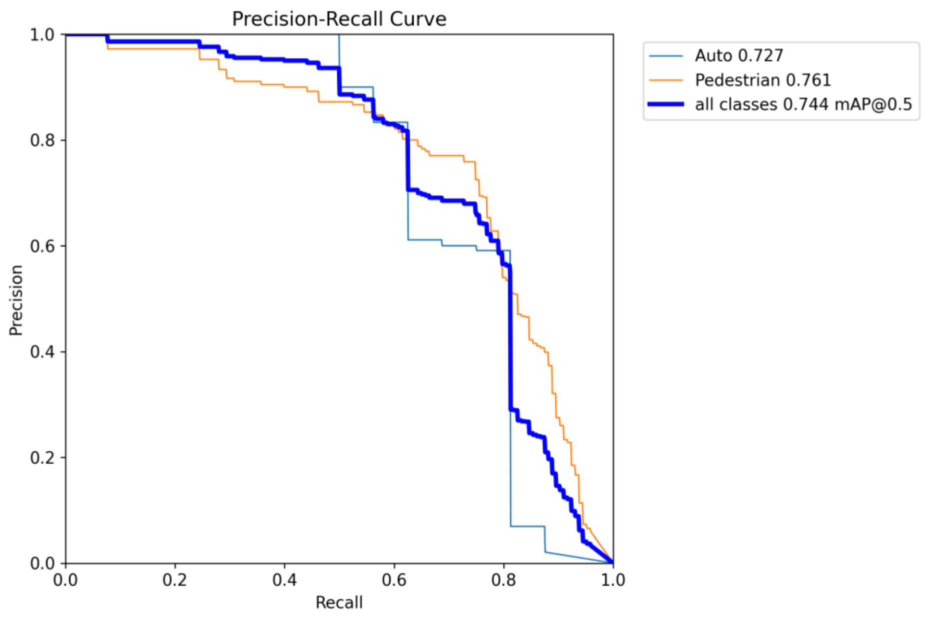
F1 CURVE



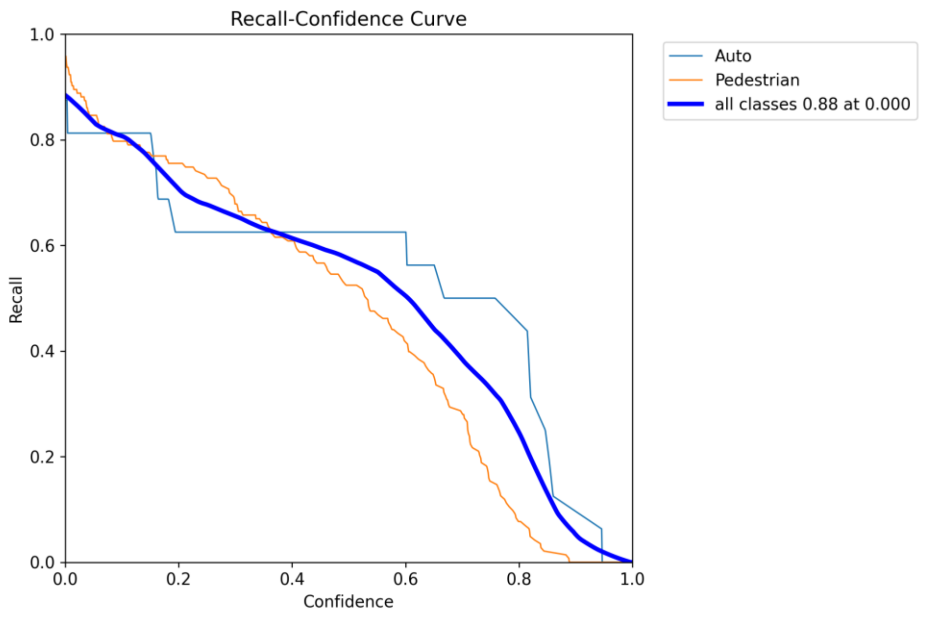
P CURVE



PR CURVE



R CURVE



Through the validation process, I learned about the performance metrics such as accuracy, precision, recall, F1 Score, mAP and IOU, confusion matrix, and loss curves

4. Prediction

Predictions made by the model on some images from test dataset:

Image with Pedestrian:

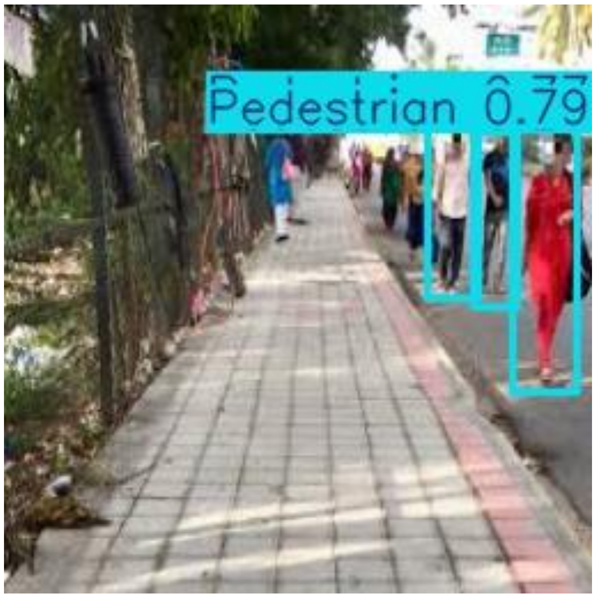


Image with Auto:

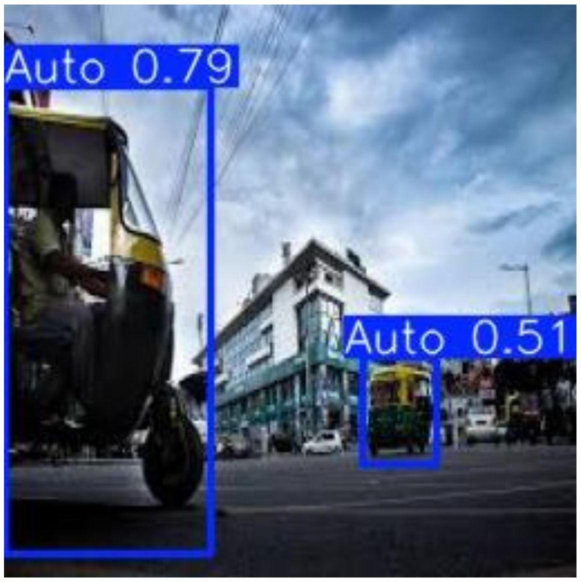
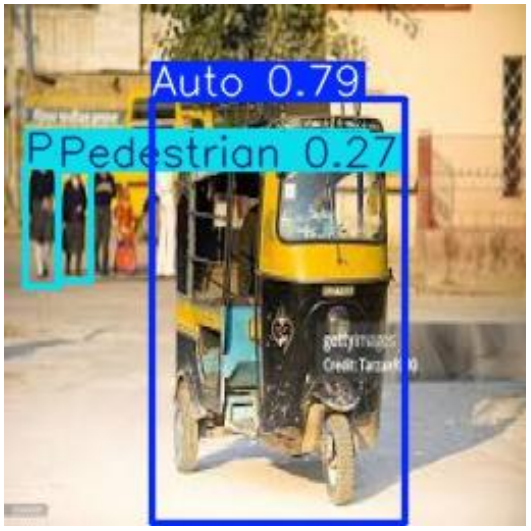


Image consisting of both Pedestrian and Auto:



Upon completing the training, validation, and evaluation of the Yolo11n model on a custom dataset, this final weekly report was drafted to summarize the work carried out each week.

References

https://www.researchgate.net/publication/339819211_PASSENGER_DETECTION_AND_COUNTING_FOR_PUBLIC_TRANSPORT_SYSTEM

https://viso.ai/applications/pothole-detection/

https://github.com/hoanglehaithanh/Traffic-Sign-Detection

https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/

https://ieeexplore.ieee.org/document/8662019

https://developers.google.com/machine-learning/intro-to-ml

https://developers.google.com/machine-learning/crash-course/linear-regression

https://www.geeksforgeeks.org/machine-learning/types-of-machine-learning/

https://www.geeksforgeeks.org/machine-learning/python-image-classification-using-keras/

https://www.kaggle.com/code/garginirmal/cnn-keras-image-classification

https://www.kaggle.com/code/shtrausslearning/pytorch-cnn-binary-image-classification

Image Classification using CNN Keras | Full implementation

Build a Deep CNN Image Classifier with ANY Images

https://docs.ultralytics.com/modes/predict/#boxes

https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z

https://www.superannotate.com/blog/yolo-object-detection

https://www.datacamp.com/blog/yolo-object-detection-explained

https://encord.com/blog/yolo-object-detection-guide/

https://youtu.be/UL2cfTTqdNo?si=-Frg49aH_i72I_Xp

https://youtu.be/m9fH9OWn8YM?si=0_FOQjOZztVGv1ff