# Internship Final Report

# Week 1: Understanding ARAS and Learning ML Fundamentals

During the first week, the focus was on understanding Advanced Rider Assistance Systems (ARAS) and building fundamental knowledge in Machine Learning (ML).

**ARAS Features 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)

**Machine Learning Topics Covered:**

* Types of Machine Learning: Supervised, Unsupervised, Reinforcement, Semi-supervised
* Concepts of Classification and Regression

**Referred Links:**

For ARAS Features:

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>

For ML Fundamentals:

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/>

**Week 2: Neural Networks and CNNs**

During the second week, the architecture and working of neural networks, particularly Feedforward Neural Networks, MLPs and Convolutional Neural Networks (CNNs) were studied.

Key Concepts Covered:

* Structure of Neural Networks: Input, Hidden, and Output layers
* Forward and Backward Propagation
* Activation Functions: Tanh, Sigmoid, ReLU, Softmax
* Loss Functions: MSE, MAE, Cross-Entropy
* Optimization Techniques: Stochastic Gradient Descent (SGD), Batch, Mini-Batch

Explored CNNs and specifically the LeNet-5 architecture which included:

* Layers: Convolution, Pooling, Fully Connected
* Activation Functions used: Tanh, Softmax
* Implementation steps: Data preprocessing, model building, compiling, training, evaluation

**Referred Links**

https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

<https://www.geeksforgeeks.org/machine-learning/neural-networks-a-beginners-guide/>

<https://paravisionlab.co.in/lenet-5-architecture/>

<https://www.youtube.com/watch?v=aircAruvnKk>  
<https://www.youtube.com/watch?v=YRhxdVk_sIs>  
<https://www.youtube.com/watch?v=5tvmMX8r_OM>

[Hot Dog or Not Hot Dog – Convolutional Neural Network Course for Beginners](https://www.youtube.com/watch?v=nVhau51w6dM)

# Week 3: Image Classification with CNN

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

**Setup:**  
Essential tools were installed including Anaconda, TensorFlow, PyTorch, and LabelMe. These were used for data preprocessing, annotation, model building, and evaluation.

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

[Indian roads pedestrian - Google Search](https://www.google.com/search?q=indian+roads+pedestrian&sca_esv=f1d46e54a177805b&udm=2&sxsrf=AE3TifPrOA4kiAHktmtGQEBPNqpTE_eoHg%3A1750074017450&ei=oQJQaO2lG_XIwN4PorbDiAI&ved=0ahUKEwit-o_h7fWNAxV1JNAFHSLbECEQ4dUDCBE&uact=5&oq=indian+roads+pedestrian&gs_lp=EgNpbWciF2luZGlhbiByb2FkcyBwZWRlc3RyaWFuMgcQIxgnGMkCSIQ-UIgKWMc7cAF4AJABAJgBiAOgAaErqgEEMy0xN7gBA8gBAPgBAZgCC6AC5xzCAgYQABgHGB7CAgoQABiABBhDGIoFwgIFEAAYgATCAgcQABiABBgKwgIEEAAYHsICBhAAGAoYHsICBhAAGAgYHpgDAIgGAZIHBDMtMTGgB_UwsgcEMy0xMbgH5xzCBwUwLjguM8gHIA&sclient=img)

[Auto rickshaw in india - Google Search](https://www.google.com/search?sca_esv=f1d46e54a177805b&hl=en&q=auto+rickshaw+in+india&uds=AOm0WdE2fekQnsyfYEw8JPYozOKzzx1qfahp2UWthC9jUZ8uQaDneKKOP26tmpGTey03WG2KEMD1lMQr_36j1zCkKjuZRpXKmc9F4yThPb4VInuCXqyTd_gf7nz99L39tBCflFbVZe84A7RyoJyvX0EUsIU-Bk9owD8rWFojus-eH4Lqt3VhY_4&udm=2&sa=X&ved=2ahUKEwjmo4fW7vWNAxXQ4MkDHWEfH5kQxKsJegUInAEQAQ&ictx=0&biw=1528&bih=738&dpr=1.25)

[Pedestrians Dataset](https://www.kaggle.com/datasets/alincijov/penn-fudan)

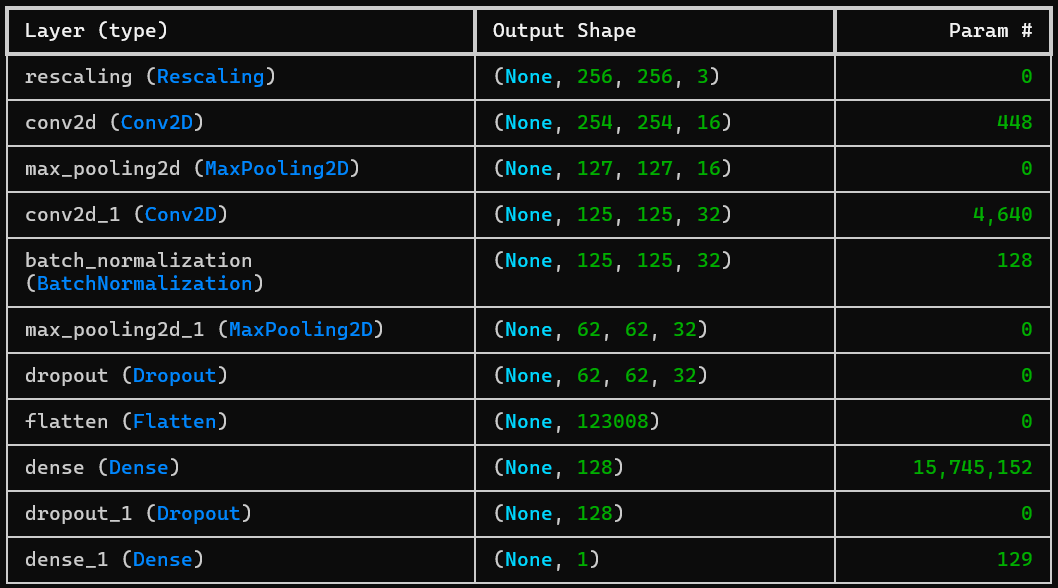
[IIT Delhi Campus Pedestrian Dataset (Detection)](https://www.kaggle.com/datasets/aryangarg01/iit-delhi-campus-pedestrian-dataset-detection)

**Implementation using Keras (TensorFlow):**

**Hyperparameters:**

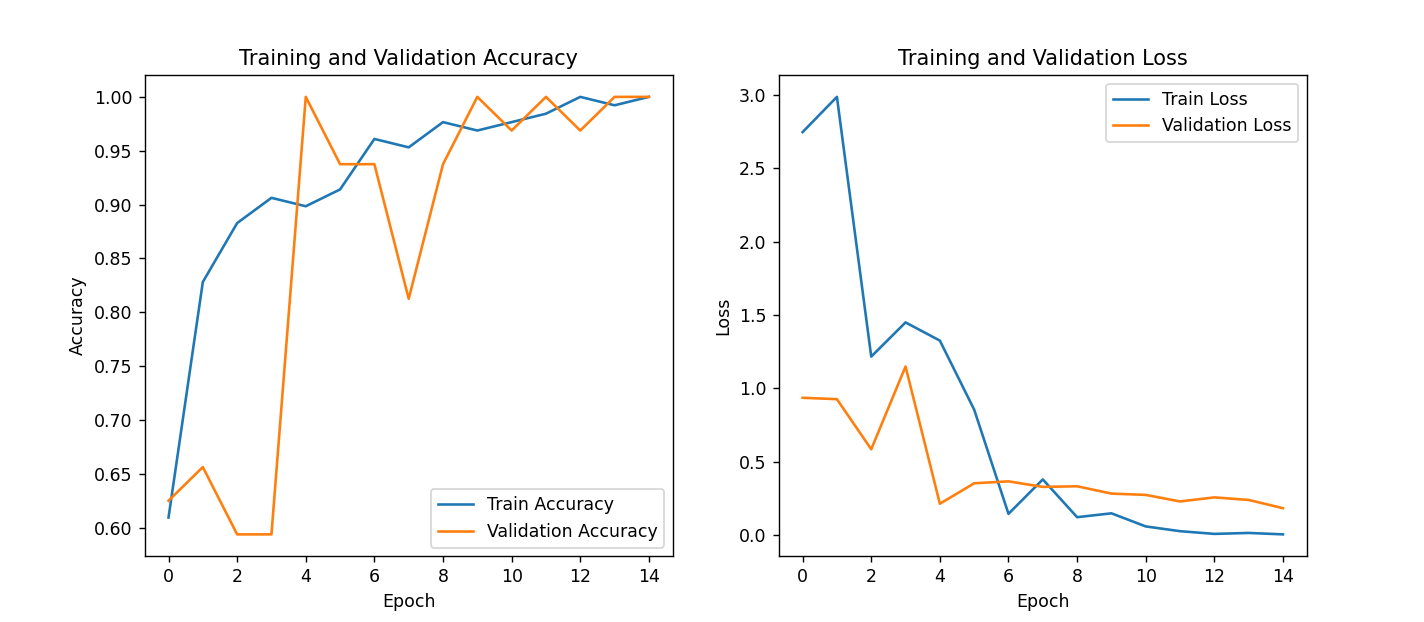
- Image Size: 256x256  
- Batch Size: 32  
- Epochs: 15  
- Optimizer: Adam  
- Loss Function: Binary Cross-Entropy

**Model Summary:**

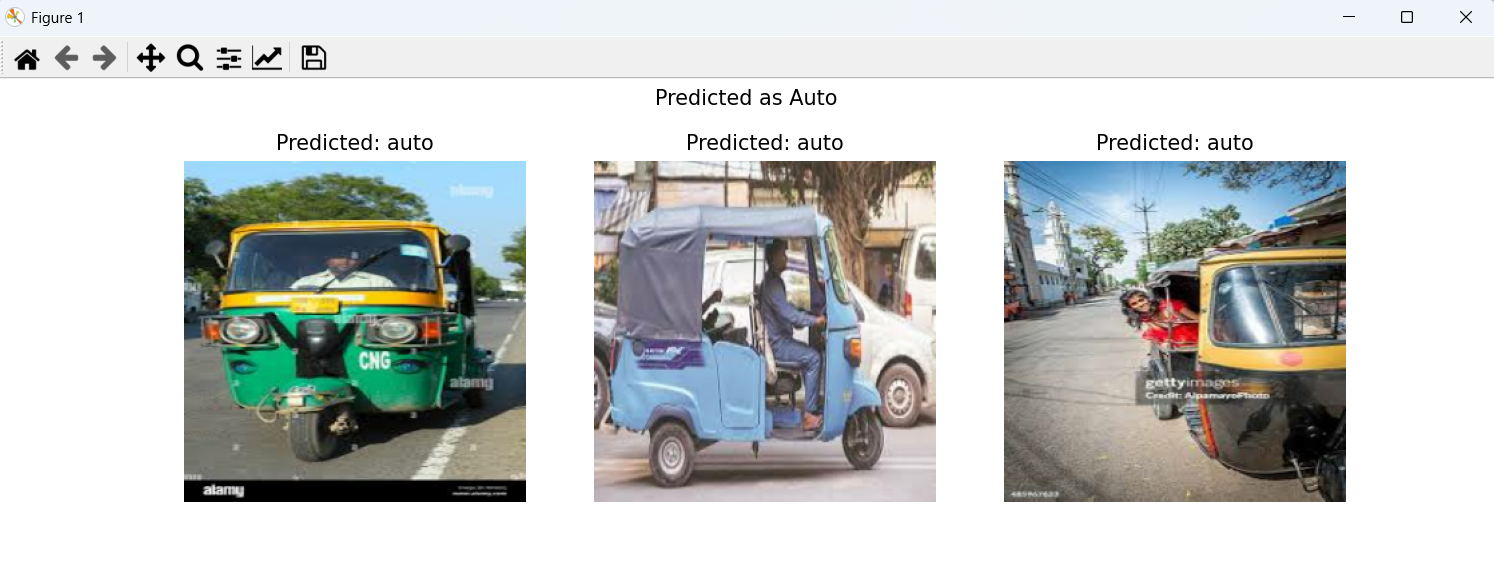


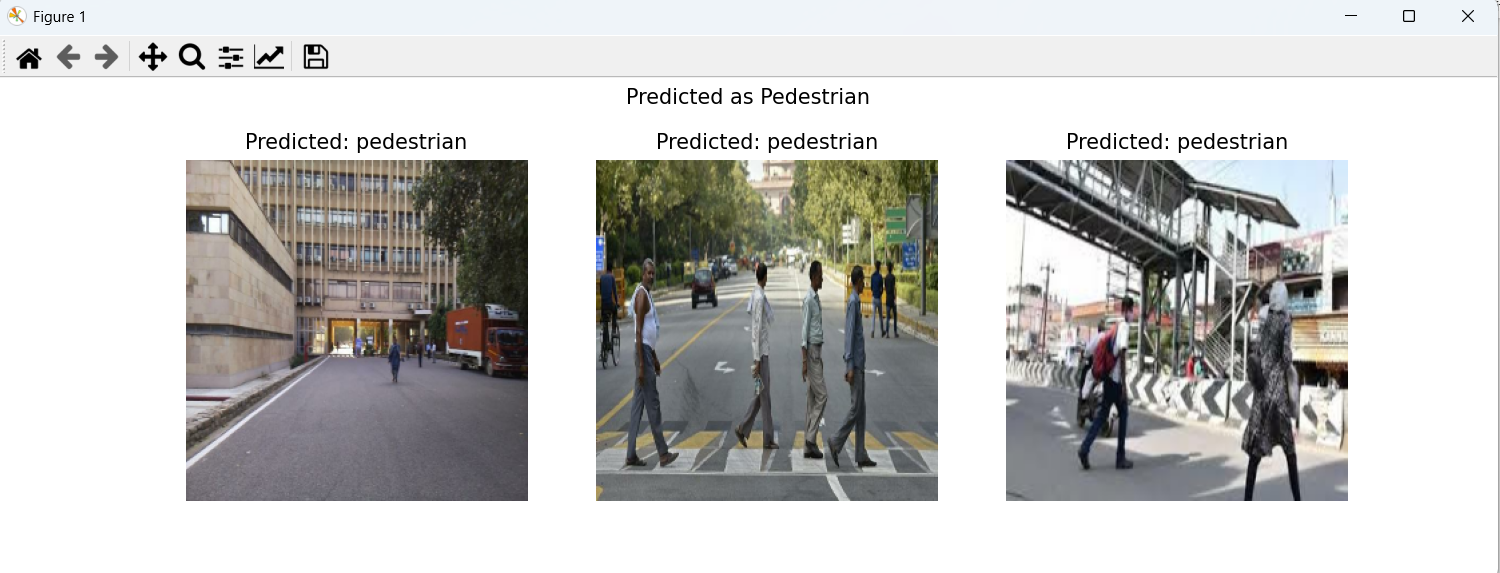
**Test Results:**





**OUTPUT**

****

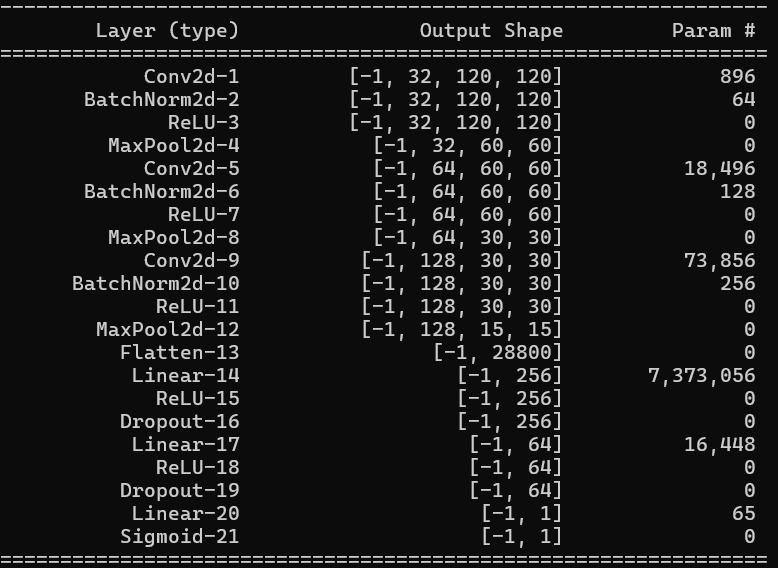
****

**Implementation using PyTorch:**

**Hyperparameters:**

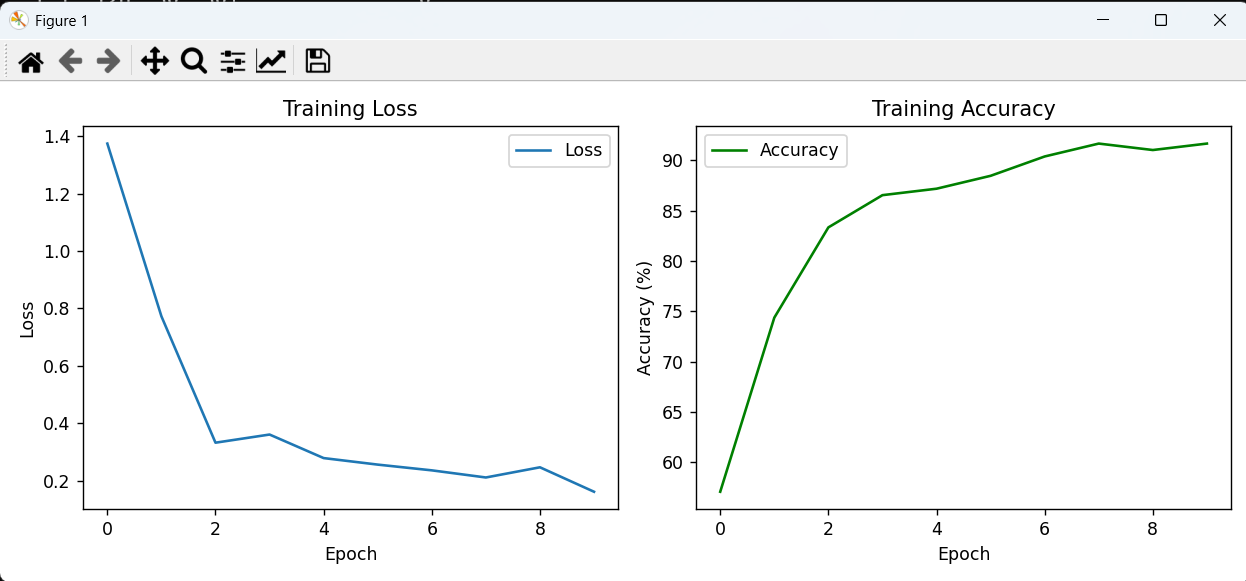
- Image Size: 120x120  
- Batch Size: 16  
- Epochs: 10  
- Learning Rate: 0.01  
- Optimizer: Adam  
- Loss Function: Binary Cross-Entropy

**Model Summary:**

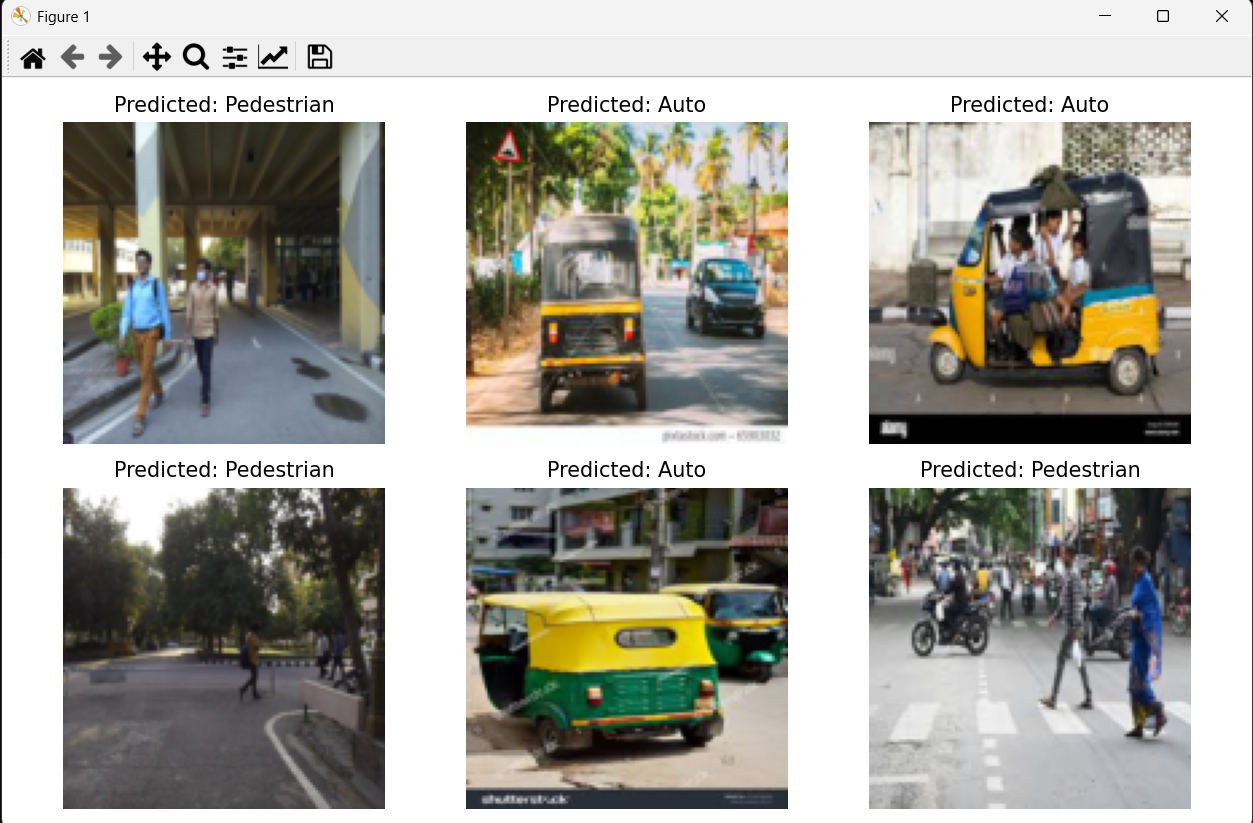


**Test Results:**





**OUTPUT**



**Referred Links**

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](https://www.youtube.com/watch?v=J1jhfAw5Uvo)

[Build a Deep CNN Image Classifier with ANY Images](https://www.youtube.com/watch?v=jztwpsIzEGc)

# Week 4: Object Detection – Pedestrian and Auto-Rickshaw

The fourth week involved the following tasks:

* Learnt Annotation and annotated a dataset using the tool Labelme.
* Studied object detection algorithms and YOLO (You Only Look Once) in particular.

Learning Outcomes:

- Studied object detection methods: Two-stage (e.g., Faster R-CNN) and One- stage (e.g., YOLO, SSD)  
- Understood YOLO architecture: Feature extraction, grid division, bounding box prediction, class prediction, and post-processing.

**Implementation:**

The goal was to implement a model for detecting pedestrian and auto

Dataset collection and Annotation:  
Images were collected from Google Images and Kaggle.

* [Indian roads pedestrian - Google Search](https://www.google.com/search?q=indian+roads+pedestrian&sca_esv=f1d46e54a177805b&udm=2&sxsrf=AE3TifPrOA4kiAHktmtGQEBPNqpTE_eoHg%3A1750074017450&ei=oQJQaO2lG_XIwN4PorbDiAI&ved=0ahUKEwit-o_h7fWNAxV1JNAFHSLbECEQ4dUDCBE&uact=5&oq=indian+roads+pedestrian&gs_lp=EgNpbWciF2luZGlhbiByb2FkcyBwZWRlc3RyaWFuMgcQIxgnGMkCSIQ-UIgKWMc7cAF4AJABAJgBiAOgAaErqgEEMy0xN7gBA8gBAPgBAZgCC6AC5xzCAgYQABgHGB7CAgoQABiABBhDGIoFwgIFEAAYgATCAgcQABiABBgKwgIEEAAYHsICBhAAGAoYHsICBhAAGAgYHpgDAIgGAZIHBDMtMTGgB_UwsgcEMy0xMbgH5xzCBwUwLjguM8gHIA&sclient=img)
* [Auto rickshaw in india - Google Search](https://www.google.com/search?sca_esv=f1d46e54a177805b&hl=en&q=auto+rickshaw+in+india&uds=AOm0WdE2fekQnsyfYEw8JPYozOKzzx1qfahp2UWthC9jUZ8uQaDneKKOP26tmpGTey03WG2KEMD1lMQr_36j1zCkKjuZRpXKmc9F4yThPb4VInuCXqyTd_gf7nz99L39tBCflFbVZe84A7RyoJyvX0EUsIU-Bk9owD8rWFojus-eH4Lqt3VhY_4&udm=2&sa=X&ved=2ahUKEwjmo4fW7vWNAxXQ4MkDHWEfH5kQxKsJegUInAEQAQ&ictx=0&biw=1528&bih=738&dpr=1.25)
* [Pedestrians Dataset](https://www.kaggle.com/datasets/alincijov/penn-fudan)
* [IIT Delhi Campus Pedestrian Dataset (Detection)](https://www.kaggle.com/datasets/aryangarg01/iit-delhi-campus-pedestrian-dataset-detection)

Annotation was performed using Roboflow.

Classes were labeled and datasets were split into 3 – training, validation and test dataset (in the ratio 70:20:10 respectively).

Model Summary:

Training and Validation results:

# Referred Links

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/