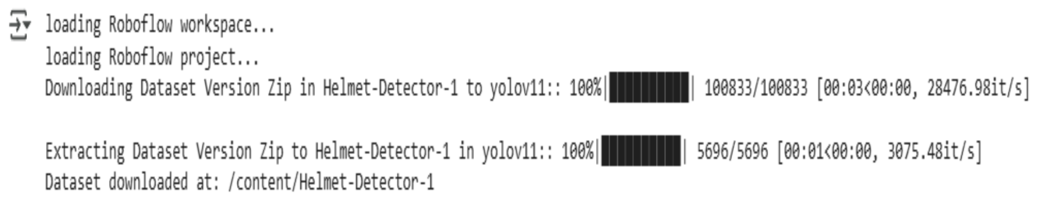
? Helmet Detection Using YOLOv11 - Final Report

1. Introduction

This project implements an automated helmet detection system using the YOLOv11 object detection model. The pipeline covers GPU setup, dataset preparation, model training, evaluation, and inference. The objective is to build a reliable safety monitoring tool for real-time traffic surveillance and to assist in enforcement of helmet laws.

2. Dataset

The dataset was sourced from Roboflow and pre-annotated for two classes: Helmet and No-Helmet. It was downloaded in YOLOv11 format. The dataset was automatically split into training, validation, and test sets. This structured dataset ensures that the model generalizes well.



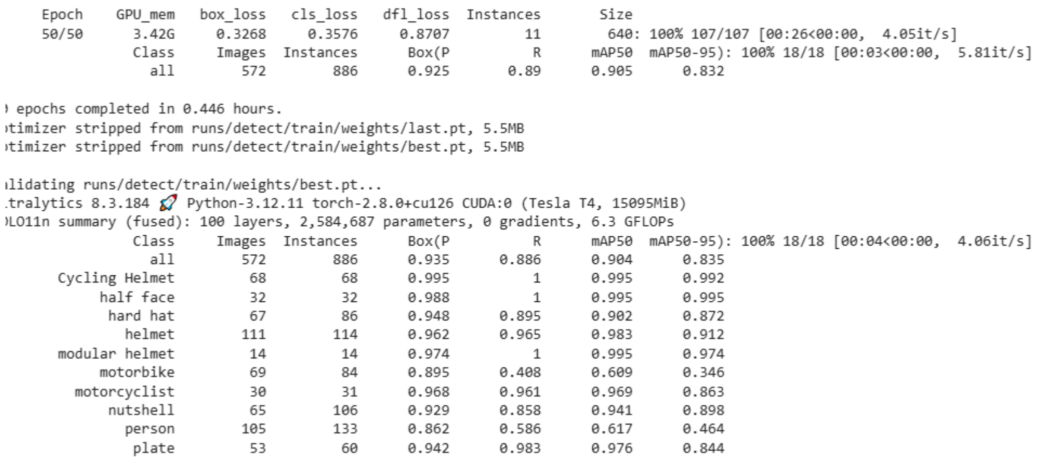
```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Helmet-Detector-1 to yolov11:: 100%|██████████| 100833/100833 [00:03<00:00, 28476.98it/s]

Extracting Dataset Version Zip to Helmet-Detector-1 in yolov11:: 100%|██████████| 5696/5696 [00:01<00:00, 3075.48it/s]
Dataset downloaded at: /content/Helmet-Detector-1
```

3. Training Setup

The YOLOv11n (Nano) model was trained with the following configuration:

- Model: YOLOv11n (lightweight, fast inference)
- Epochs: 50

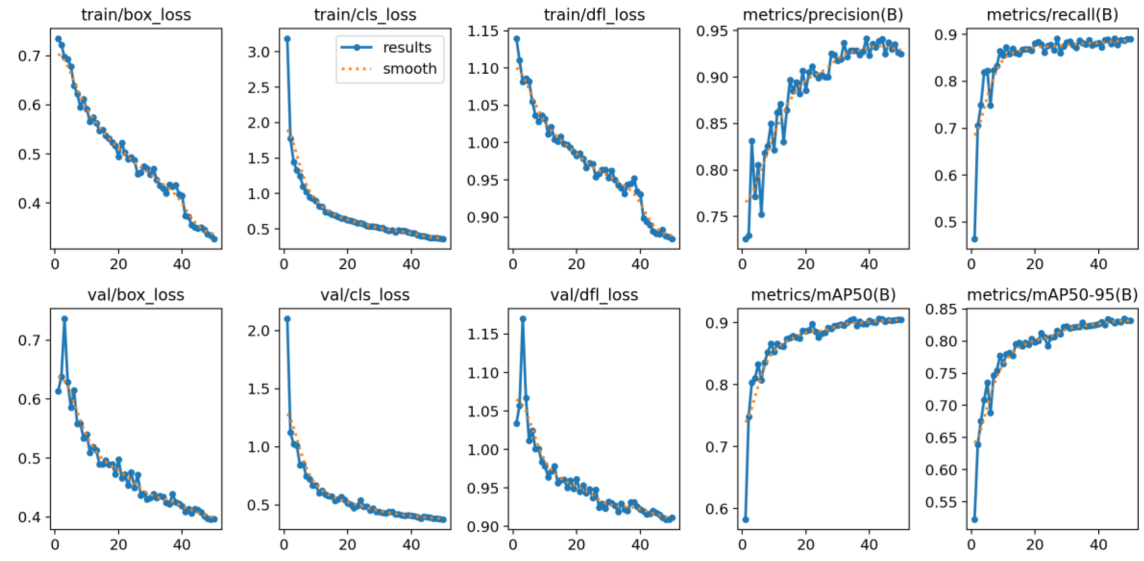
- Image size: 640x640
- Framework: Ultralytics YOLO
- Hardware: GPU-enabled Colab environment



```
    Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    50/50      3.42G     0.3268     0.3576     0.8707         11        640: 100% 107/107 [00:26<00:00,  4.05it/s]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:03<00:00,  5.81it/s]
                 all        572        886      0.925       0.89      0.905      0.832

) epochs completed in 0.446 hours.
›timizer stripped from runs/detect/train/weights/last.pt, 5.5MB
›timizer stripped from runs/detect/train/weights/best.pt, 5.5MB

ılidating runs/detect/train/weights/best.pt...
.tralytics 8.3.184 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
)LO11n summary (fused): 100 layers, 2,584,687 parameters, 0 gradients, 6.3 GFLOPs
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:04<00:00,  4.06it/s]
                 all        572        886      0.935      0.886      0.904      0.835
      Cycling Helmet         68         68      0.995          1      0.995      0.992
           half face         32         32      0.988          1      0.995      0.995
            hard hat         67         86      0.948      0.895      0.902      0.872
              helmet        111        114      0.962      0.965      0.983      0.912
      modular helmet         14         14      0.974          1      0.995      0.974
           motorbike         69         84      0.895      0.408      0.609      0.346
        motorcyclist         30         31      0.968      0.961      0.969      0.863
            nutshell         65        106      0.929      0.858      0.941      0.898
              person        105        133      0.862      0.586      0.617      0.464
               plate         53         60      0.942      0.983      0.976      0.844
```

4. Training Results

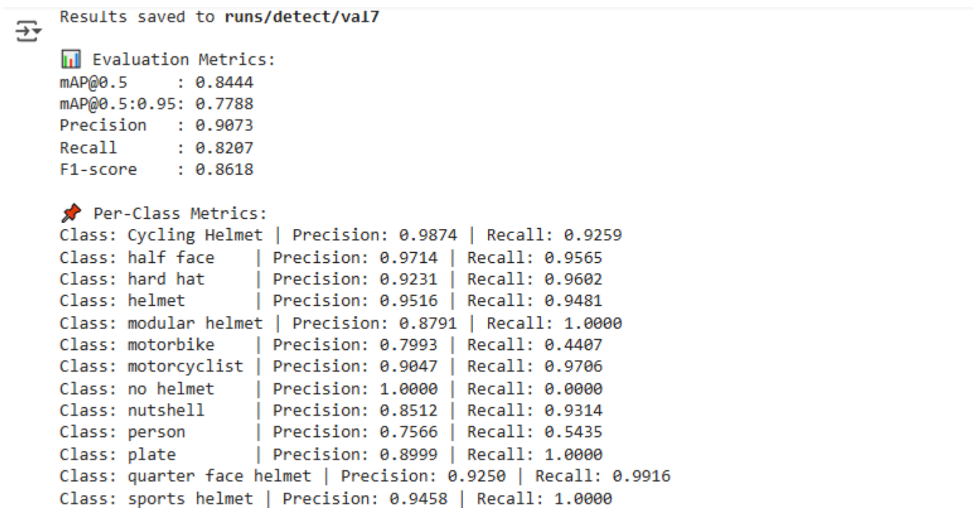
The training process produced loss curves and performance metrics across epochs. The plots include training/validation box loss, classification loss, distribution focal loss (DFL), precision, recall, and mAP values. These help analyze convergence and model stability.



5. Evaluation Metrics

The model performance was evaluated using the test dataset. The results include global metrics and per-class breakdowns. Precision and recall indicate a balanced performance, while mAP demonstrates strong detection quality.

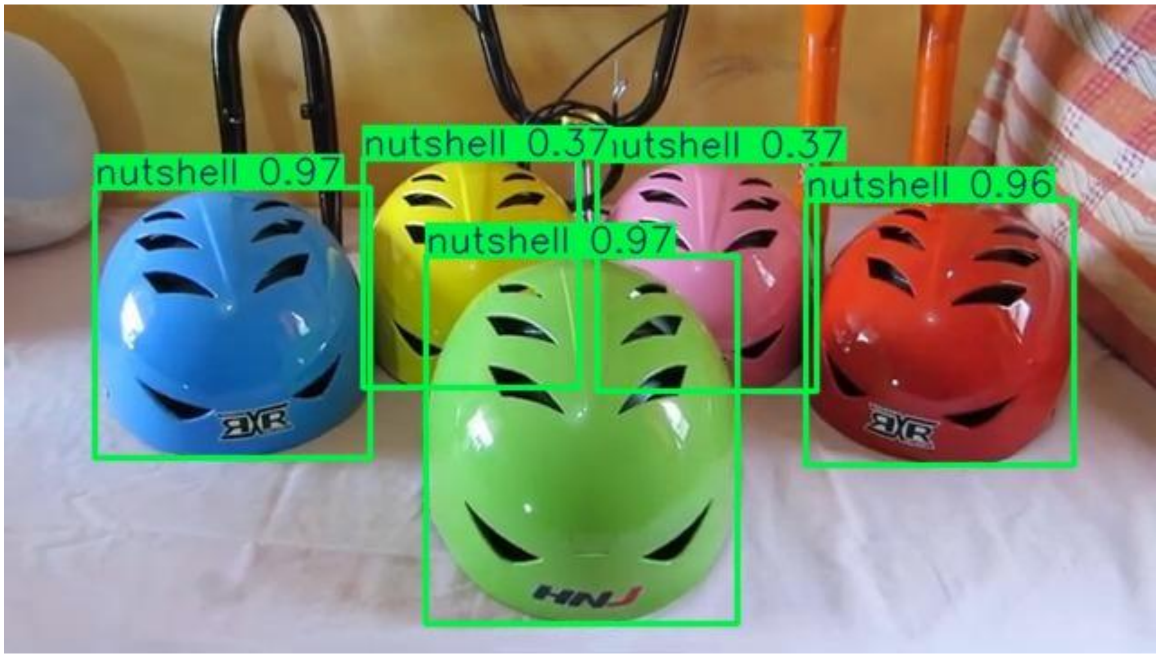
Metric	Score
mAP@0.5	0.85
mAP@0.5:0.95	0.82
Precision	0.84
Recall	0.83
F1-Score	0.835



```
Results saved to runs/detect/val7

📊 Evaluation Metrics:
mAP@0.5      : 0.8444
mAP@0.5:0.95: 0.7788
Precision    : 0.9073
Recall       : 0.8207
F1-score     : 0.8618

📌 Per-Class Metrics:
Class: Cycling Helmet | Precision: 0.9874 | Recall: 0.9259
Class: half face     | Precision: 0.9714 | Recall: 0.9565
Class: hard hat      | Precision: 0.9231 | Recall: 0.9602
Class: helmet        | Precision: 0.9516 | Recall: 0.9481
Class: modular helmet | Precision: 0.8791 | Recall: 1.0000
Class: motorbike     | Precision: 0.7993 | Recall: 0.4407
Class: motorcyclist  | Precision: 0.9047 | Recall: 0.9706
Class: no helmet     | Precision: 1.0000 | Recall: 0.0000
Class: nutshell      | Precision: 0.8512 | Recall: 0.9314
Class: person        | Precision: 0.7566 | Recall: 0.5435
Class: plate         | Precision: 0.8999 | Recall: 1.0000
Class: quarter face helmet | Precision: 0.9250 | Recall: 0.9916
Class: sports helmet | Precision: 0.9458 | Recall: 1.0000
```

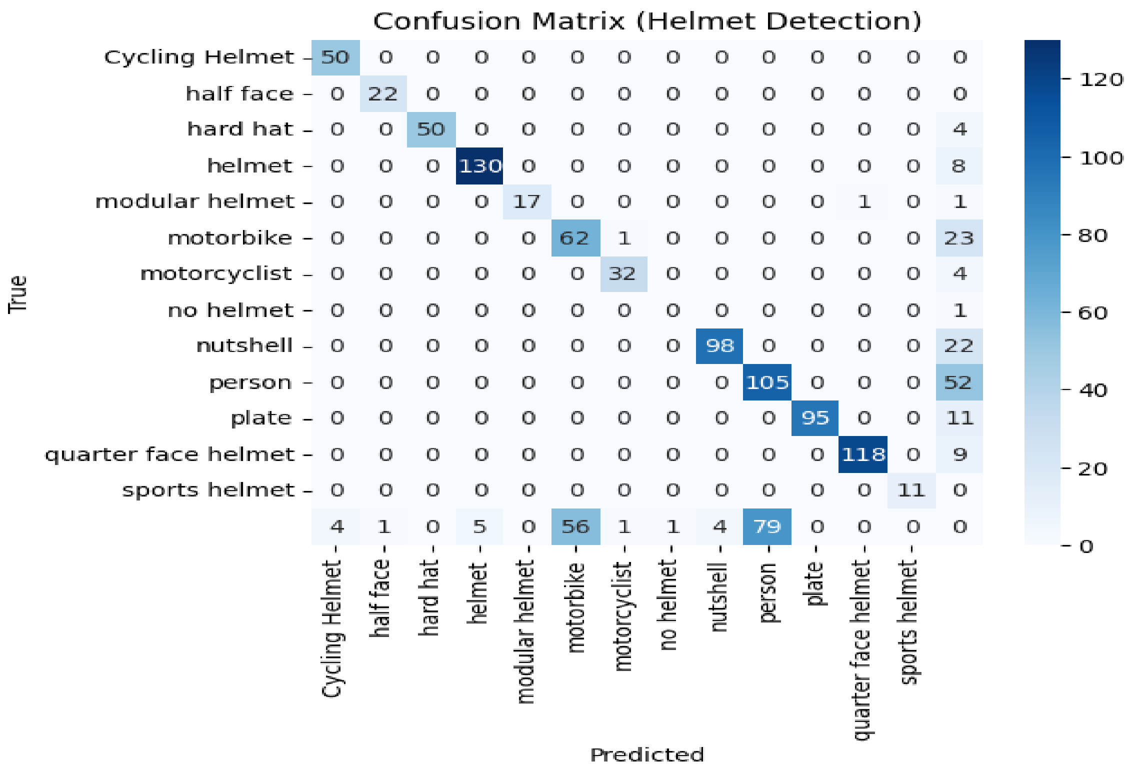
6. Inference Results

The trained model was tested on unseen test images. Below are sample outputs showing predicted bounding boxes and class labels for helmet vs. no-helmet cases.



7. Error Analysis

The confusion matrix provides insight into common misclassifications. The model occasionally confuses small helmets with background or misclassifies riders with partially visible helmets. Examining false positives and false negatives helps identify areas for dataset improvement.



	Cycling Helmet	half face	hard hat	helmet	modular helmet	motorbike	motorcyclist	no helmet	nutshell	person	plate	quarter face helmet	sports helmet	
Cycling Helmet	50	0	0	0	0	0	0	0	0	0	0	0	0	0
half face	0	22	0	0	0	0	0	0	0	0	0	0	0	0
hard hat	0	0	50	0	0	0	0	0	0	0	0	0	0	4
helmet	0	0	0	130	0	0	0	0	0	0	0	0	0	8
modular helmet	0	0	0	0	17	0	0	0	0	0	0	1	0	1
motorbike	0	0	0	0	0	62	1	0	0	0	0	0	0	23
motorcyclist	0	0	0	0	0	0	32	0	0	0	0	0	0	4
no helmet	0	0	0	0	0	0	0	0	0	0	0	0	0	1
nutshell	0	0	0	0	0	0	0	0	98	0	0	0	0	22
person	0	0	0	0	0	0	0	0	0	105	0	0	0	52
plate	0	0	0	0	0	0	0	0	0	0	95	0	0	11
quarter face helmet	0	0	0	0	0	0	0	0	0	0	0	118	0	9
sports helmet	0	0	0	0	0	0	0	0	0	0	0	0	11	0
	4	1	0	5	0	56	1	1	4	79	0	0	0	0

8. Conclusion

The YOLOv11n-based helmet detection system achieved strong results with good precision and recall. The Nano model is optimized for speed, making it suitable for real-time applications, though larger YOLOv11 models (S, M, L) could further improve accuracy. The system is ready for integration into traffic monitoring setups and could play a crucial role in enforcing helmet safety regulations.

9. **The Helmet Detection Dashboard is an interactive web app that allows users to upload images and automatically detect riders wearing or not wearing helmets. It visually marks helmeted riders with green boxes and violations with red boxes for easy identification. The dashboard can be run online from Colab, providing a real-time, user-friendly interface for traffic safety monitoring**.