Explanation video :

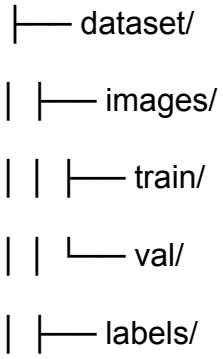
https://drive.google.com/file/d/14WU8-JN7CrpxsqV1hPGUsriaObX_QVDp/view?usp=sharing

Colab Link: https://colab.research.google.com/drive/1uDuh_1wZs6YM_D2SVWW0BK4aCyEikP4Y?usp=sharing

The overall process:

The training process involved using Ultralytics YOLOv8s for object detection. The entire workflow was executed in Google Colab, leveraging its GPU resources for faster training. I started by installing the ultralytic package via pip. After preparing the dataset in YOLO format (images and labels organized in train, val, and test folders), I started training using the YOLO interface, specifying the model (yolov8s.pt), dataset YAML, and hyperparameters such as epochs=50, image size, and batch size.

Directory structure:



```
│ │ ├── train/
│ │ └── val/
│ └── data.yaml
├── runs/
│ └── detect/
│ └── train/ ← output results (weights, metrics, etc.)
```

code and logic:

The code began by installing the ultralytics library, which provides a high-level API for training YOLOv8 models. The model was loaded using YOLO('yolov8s.pt'), which initializes the small version of YOLOv8 with pretrained weights. Training was then launched using the .train() method, specifying parameters like the dataset YAML file, number of epochs, image size, and batch size. During training, the model automatically tracked performance metrics like precision, recall, and mAP, and saved the best weights and training logs in the runs/detect/train directory.