Name : Prajwal Dilip Ganar

Class : TY – T2

PRN : 202201040034

Assignment 3

Colab link :

https://colab.research.google.com/drive/1JRMvU-9tjcjXn628UomFcEv98simeMdh?usp=sharing

Github link :

https://github.com/Prajwal435/DL_Assignment3.git

This assignment focuses on implementing and evaluating a YOLOv11 model for real time object detection using the COCO dataset. The project is executed on Google Colab, where the environment is set up with the required libraries (Roboflow, Ultralytics, PyTorch, OpenCV, etc.). The COCO dataset is acquired via Roboflow, preprocessed, and then used to train a YOLOv11 model. Finally, the trained model is tested on unseen data, and its performance is evaluated using metrics such as Mean Average Precision (mAP), Precision, Recall, and F1 Score. The goal is to understand the complete workflow from data acquisition and model training to inference and performance evaluation.

Methodology

1. Environment Setup and Installation:

-- Install Required Libraries: Set up your Google Colab environment by installing all necessary libraries such as Roboflow, Ultralytics, PyTorch, and OpenCV.

-- Verify Installation: Ensure that the libraries are installed correctly by running initial checks, which prepares the environment for subsequent steps.

2. Dataset Preparation & Preprocessing:

-- Dataset Acquisition: Use the Roboflow API to download the COCO dataset (version 34) in the YOLOv11 format. This step involves authenticating with an API key and accessing the dataset from a specific workspace.

-- Directory Structure Verification: Confirm that the dataset is organized into distinct folders for training, validation, and testing, with corresponding YOLO format annotations. -- Data Quality Check: Review sample images and their annotations to ensure that the dataset is correctly formatted and ready for training.

3. Model Training:

-- Model Initialization: Load the YOLOv11 model using the pre-trained weights provided (e.g., the "yolo11n.pt" model). This step initializes the model architecture for training.

-- Training Configuration: Set up key training parameters, including the number of epochs (e.g., 100 epochs), batch size, learning rate, and input image size. These parameters are critical for model convergence.

-- Monitoring Training: As training progresses, monitor metrics such as loss, mean average precision (mAP), precision, and recall. The training process is continuously adjusted based on these metrics to improve model performance.

-- Model Saving: Save the best-performing model weights based on validation performance for later inference.

4. Model Inference and Evaluation:

-- Load the Trained Model: Once training is complete, load the best saved model weights.

-- Run Inference: Test the model on unseen test images. In the Colab environment, ensure that GUI-based display methods are disabled and results are saved instead.

-- Result Visualization: Retrieve the saved inference outputs and visualize them using image display tools such as matplotlib or PIL. This helps in verifying the presence of bounding boxes and detection quality.

-- Performance Evaluation: Evaluate the model using key performance metrics such as mAP (at IoU 0.5 and 0.5–0.95), precision, recall, and F1 score. Analyze these metrics to understand the balance between correctly identified objects and missed detections.