**Deep Learning Project Workflow: Object Detection**

**1.Data Ingestion**

* **Download dataset zip file from Kaggle and unzip and download it.**
* **On doing that I found that data is already split into training, validation and test files.**
* **A yaml file also present .**

**2. Data Cleanup**

* **Remove Irrelevant Data**: Remove data having the wrong filetype.(i.e Data not in the format of .txt or.jpg)
* **Check for Anomalies**: Handle missing items such as images without labels or labels without images.
* **Handle Missing Items**: Identify and log any missing labels or images by comparing their basenames.

**3. Data Preparation**

* **Data Splitting**: Data is already separated in three different files as mentioned before . So we can directly use the training data file for the training phase.
* **Data exploration**:
  + Plot graphs for clear understanding of number of classes and their respective distributions.
  + Plot a graph for the number of bounding boxes per class(5 classes are present in the dataset according to yaml file) .
  + Test by plotting a bounding box on an image from the training data using respective label.

**4. Data Augmentation**

**We shall check the need of data augmentation, if needed we can use techniques like:-**

* **Image Augmentation**: Apply transformations to images like rotations, flips, zooming, and color adjustments to artificially increase the variety in the dataset.
* **Bounding Box Augmentation**: Ensure that the bounding boxes associated with images are transformed accordingly to match the augmented images.

**5. Model Building**

* **Select Model Architecture**: Have chosen to use YOLOv8 as the dataset seems to be of that respective format.
* **Configure Model**: Set up the model’s architecture, optimizer, and loss function.
* **Pre-trained Weights**: The one question we have to decide is to whether we use a pretrained model or start fresh.( as a yaml file is present in the data.)

**6. Model Training**

* **Train Model**: Use the prepared training data to train the model. Monitor the performance on the validation set to avoid overfitting.
* **Hyperparameter Tuning**: Experiment with different hyperparameters (e.g., learning rate, batch size) to find the best configuration.

**7. Model Evaluation**

* **Evaluate on Test Set**: Evaluate the model using the test data to measure performance we can judge using certain metrics such as mAP (mean Average Precision) for object detection.We even have a test video present in the dataset.
* **Visualize Results**: Check the results by visualizing predictions overlaid on test images to understand the model's behavior.

**8. Model Optimization**

* **Post-Training Quantization**: Apply model quantization techniques to reduce the size and improve the speed of the model.
* **Model Compression**: Consider model compression techniques to make the model lightweight without sacrificing too much accuracy.

**9. Model Deployment**

* **Web Application**: Deploy the model using frameworks like FastAPI, or Streamlit to serve predictions via a simple web interface.