Vehicle Detection and Classification Project

Introduction

The goal of this project is to detect and classify vehicles—such as cars, buses, motorbikes, and trucks—in images, using annotated datasets. The annotations are provided in XML format, describing bounding boxes and class labels. The project involves data preprocessing, dataset analysis, model training with YOLOv8, and evaluation.

Problem Statement

Given a dataset of images with bounding box annotations in XML format, the task is to create an object detection model that predicts vehicle locations and classifies them into selected categories. The dataset contains multiple vehicle classes with varying image resolutions and conditions. The model should generalize well to unseen images.

Data Preparation & Cleaning

- The dataset directory is explored and validated by checking image-annotation pairs.
- The XML annotations are parsed to validate bounding box correctness (xmin < xmax, ymin < ymax, bounding box within image size).
- Unmatched or invalid files are skipped, and only clean image-annotation pairs are used further.
- Cleaned images and annotations are copied to separate working directories.

Data Visualization

- Vehicle instances are extracted from annotations into a DataFrame.
- Class distributions, bounding box widths, heights, and areas are visualized using bar charts and histograms.
- 3-4 major vehicle classes (e.g., car, bus, motorbike, truck) are selected based on frequency for model training.

Data Splitting & Format Conversion

- Train and validation splits are created ensuring class stratification.
- Bounding boxes are converted to YOLO format (normalized center coordinates and dimensions).

- Corresponding images and labels are organized into train/val folders.
- A dataset YAML file specifying paths, classes, and number of classes is created.

Model Training

- A pretrained YOLOv8 large model (`yolov8l.pt`) is loaded with the Ultralitycs implementation.
- Training configuration includes batch size, augmentations, learning rate scheduling, optimizer, and early stopping.
- The model is trained on the preprocessed dataset for multiple epochs.

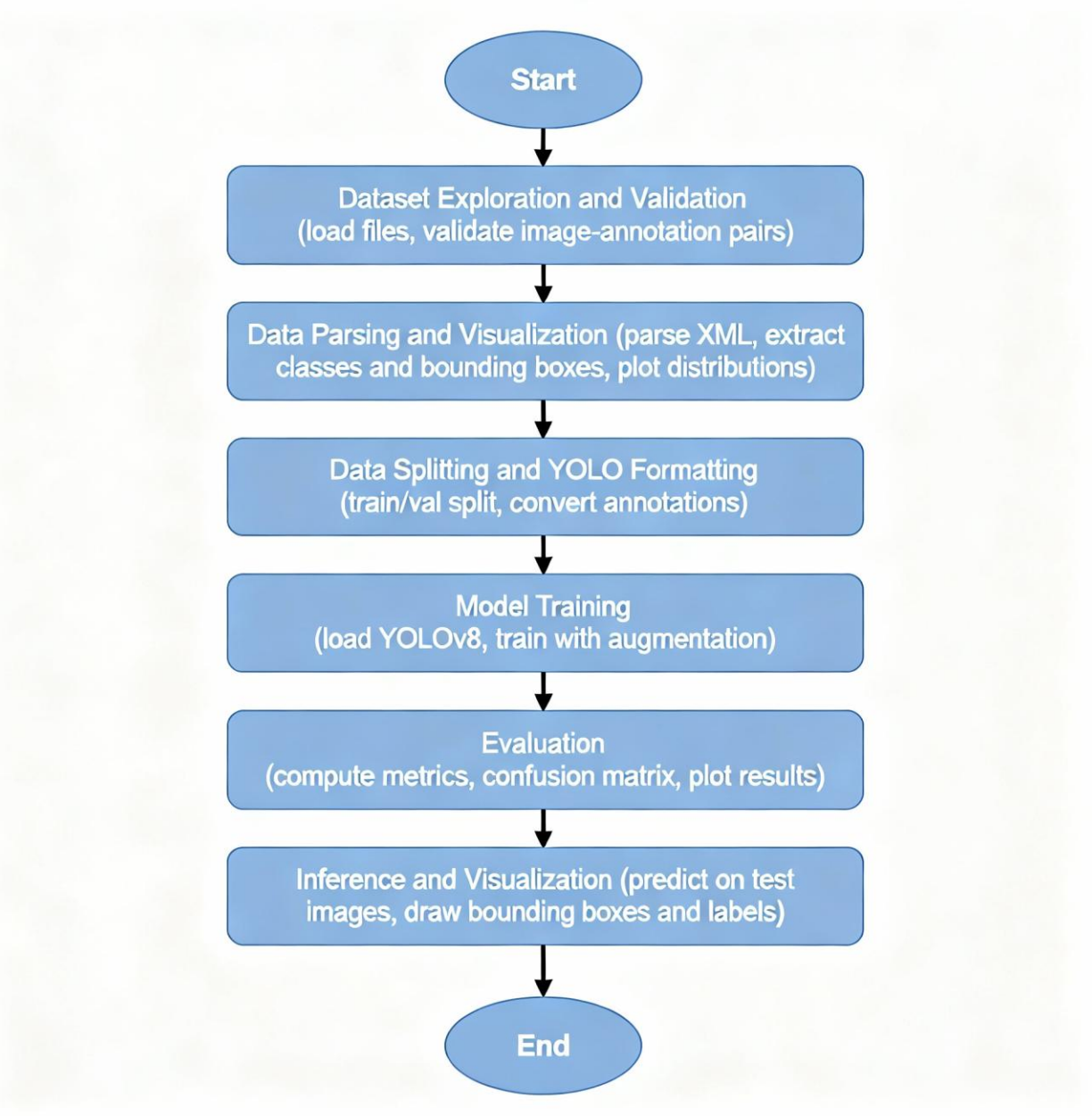
Model Evaluation & Metrics

- After training, the model is evaluated on the validation set.
- Evaluation metrics such as class-wise precision, recall, mAP@0.5, and mAP@0.5:0.95 are computed and displayed in tabular form.
- Confusion matrices for selected classes are generated and visualized using heatmaps.
- Training and validation losses and accuracies are plotted over epochs for performance monitoring.

Inference & Visualization

- The trained model is used to predict classes and bounding boxes on sample test images.
- Bounding boxes and class-confidence labels are drawn on images for visualization.
- Only selected classes are shown to make output focused and clear.

Flowchart:



The flowchart should include these main blocks and data flow:

1. **Dataset Exploration & Validation**
    - Load files, verify image-annotation pairs, filter invalid files.
2. **Data Parsing & Visualization**
    - Parse XML files to extract bounding boxes and classes; analyze data distribution.
3. **Data Splitting & YOLO Formatting**

- Create train/validation splits, convert annotations to YOLO format.

4. **Model Training**
    - Load pretrained YOLOv8 model, train on dataset with augmentation.
5. **Evaluation**
    - Compute metrics, confusion matrix, and plot results.
6. **Inference and Visualization**
    - Predict on test images, display predictions with bounding boxes and labels.