Microplastic Detection Project Report

# Project Overview

This project fine-tunes a YOLOv8 object detection model to identify microplastics in microscopy imagery. The workflow covers dataset preparation, model training, evaluation, and real-time inference.

# Workflow Summary

* Convert Roboflow CSV annotations into YOLO format using scripts/prepare\_dataset.py.
* Define the Ultralytics dataset configuration in configs/microplastics.yaml.
* Train YOLOv8 via train.py with configurable hyperparameters (epochs, batch size, image size).
* Evaluate the trained weights on the validation split and review precision/recall/mAP metrics.
* Run live\_inference.py to perform real-time detection on webcam or video sources.

# Evaluation Metrics

Model: YOLOv8n fine-tuned for 50 epochs on CPU (torch-2.8.0+cpu).

* Precision (P): 0.780
* Recall (R): 0.663
* mAP@0.50: 0.735
* mAP@0.50:0.95: 0.355
* Per-image timing: 0.9 ms preprocess, 30.7 ms inference, 0.3 ms postprocess.

Inference speed indicates the model can operate near real time on CPU, with further gains possible on GPU.

# Testing on Images

To run the trained model on a still image use the Ultralytics API:

from ultralytics import YOLO  
  
model = YOLO('runs/microplastics/weights/best.pt')  
model.predict('path/to/image.jpg', conf=0.3, save=True)

Annotated results are written to runs/detect/predict/. Adjust the confidence threshold or pass device='cuda:0' for GPU inference.

# Live Inference

Execute live\_inference.py to process frames from a webcam or video stream:

python live\_inference.py --weights runs/microplastics/weights/best.pt --source 0

Press 'q' to close the OpenCV window. The script supports alternate sources such as RTSP URLs or video files.

# Repository & Git Notes

* Initialise the repository with git init, add files, then commit with an informative message.
* Rename the default branch to main using git branch -M main when needed.
* Add the GitHub remote (git remote add origin <url>) and push via git push -u origin main.
* Include .gitignore to exclude generated datasets, runs/, virtual environments, and model checkpoints.

# Problem-Solution Highlights

1. Dataset conversion: Implemented scripts/prepare\_dataset.py to translate Roboflow CSV annotations to YOLO labels and copy imagery.
2. Training guidance: Added train.py entrypoint with arguments for base checkpoint, epochs, batch size, image size, and device selection.
3. Evaluation interpretation: Explained precision, recall, mAP@0.50, and mAP@0.50:0.95 metrics from Ultralytics validation output.
4. Inference usage: Described how to run predictions on individual images and operate the live inference script.
5. Git troubleshooting: Resolved push issues by committing, renaming the branch to main, and pushing to the remote origin.
6. Documentation: Prepared README.md and this Word report to summarise the project and captured solutions.