Microplastic Detection Project Guide

Microplastic Detection leverages YOLOv8 to identify microplastics from microscopy imagery. This document summarizes the project architecture along with the key setup, training, and inference commands.

# Architecture Overview

* **data/:** Roboflow export with train/ and valid/ splits used as the raw source dataset.
* **dataset/:** YOLO-formatted dataset generated by scripts/prepare\_dataset.py (images/ and labels/ per split).
* **scripts/prepare\_dataset.py:** Converts Roboflow CSV annotations into YOLO-normalised labels and copies images.
* **configs/microplastics.yaml:** Ultralytics dataset definition that points training to dataset/images and dataset/labels.
* **train.py:** Training entry point that fine-tunes a YOLOv8 checkpoint and saves artifacts under runs/<name>.
* **runs/:** Auto-generated directory that stores checkpoints, metrics, and prediction visualisations from Ultralytics.
* **live\_inference.py and testapp.py:** Runnable scripts for webcam/video inference and single-image evaluation.
* **requirements.txt and yolov8n.pt:** Python dependencies and the default base weights used to initialise training.

# Environment Setup

Run the following steps once per machine to prepare a clean Python environment:

1. Create a dedicated virtual environment.

python -m venv .venv

1. Activate the environment (PowerShell).

.venv\Scripts\Activate.ps1

1. Upgrade pip to the latest version.

pip install --upgrade pip

1. Install project requirements.

pip install -r requirements.txt

Install the matching PyTorch build for your CPU or GPU by copying the command suggested at https://pytorch.org/get-started/locally/ after activating the environment.

# Dataset Preparation

Convert the Roboflow export into YOLO format before training. This copies images into dataset/images/<split> and creates YOLO label files under dataset/labels/<split>.

python scripts/prepare\_dataset.py

Optional arguments you can supply:

* **--data-root:** Override where the Roboflow export is located (defaults to data/).
* **--target-root:** Write the YOLO-formatted dataset to a different directory (defaults to dataset/).
* **--class-id:** Set the numeric class ID if you introduce additional classes later.

# Training the Model

train.py wraps Ultralytics YOLO to fine-tune a pretrained checkpoint using the dataset definition above.

python train.py --model yolov8n.pt --epochs 50 --batch 16 --imgsz 640

Important flags:

* **--data-config:** Path to the dataset YAML (defaults to configs/microplastics.yaml).
* **--model:** Base YOLOv8 weights to fine-tune, e.g. yolov8n.pt or yolov8s.pt.
* **--device:** Specify compute hardware such as 'cpu', 'cuda', or a GPU index like '0,1'.
* **--project:** Parent directory for Ultralytics run artifacts (default runs/).
* **--name:** Run name used as the subfolder under the project directory.
* **--resume:** Resume the latest run with the same name instead of starting from scratch.

Resume a previous training run after an interruption:

python train.py --model yolov8n.pt --resume --project runs --name microplastics

# Running Inference

## Single Image Testing

Update the image path inside testapp.py if needed, then run the script to create annotated outputs in runs/detect/predict/.

python testapp.py

Key tips:

* Adjust the confidence threshold via the conf= argument inside testapp.py.
* Set device='cuda:0' after installing a GPU-enabled PyTorch build.

## Live or Video Inference

Stream from a webcam, RTSP source, or video file. Press 'q' in the display window to exit.

python live\_inference.py --weights runs/detect/microplastics/weights/best.pt --source 0 --conf 0.3

* Use --source <path or URL> to stream from a file or remote camera.
* Provide --device cuda to force GPU inference when available.
* Lower --conf to see more detections or raise it to filter noise.

# Operational Tips

* Check runs/ for metrics, model checkpoints, and prediction imagery after each experiment.
* Version control configs/microplastics.yaml and keep track of dataset revisions for reproducibility.
* Archive the best.pt weights generated in runs/ when promoting a model to production.