Microplastic Detection Project Technical Document

This document captures the architecture, code behaviour, evaluation metrics, and generated artefacts for the microplastic detection project located.

# 1. Project Structure and Workflow

* **configs/:** Holds Ultralytics dataset definitions, chiefly configs/microplastics.yaml.
* **data/:** Roboflow export with train/ and valid/ splits used as raw input for conversion.
* **dataset/:** Destination directory populated by scripts/prepare\_dataset.py with YOLO-formatted images and labels.
* **runs/:** Ultralytics output tree containing training artefacts (runs/microplastics) and inference results (runs/detect).
* **scripts/prepare\_dataset.py:** Utility to convert Roboflow CSV annotations into YOLO label files (see section 2).
* **train.py:** Entrypoint for fine-tuning YOLOv8 checkpoints.
* **live\_inference.py:** Real-time webcam or stream inference client.
* **testapp.py:** Quick single-image inference helper exporting annotated outputs.
* **requirements.txt:** Pinned Python dependencies: ultralytics>=8.2.0 and opencv-python>=4.8.0.
* **yolov8n.pt:** Default pretrained checkpoint referenced during fine-tuning.
* **README.md:** High-level instructions for setup, dataset preparation, and usage.

Workflow summary: prepare\_dataset.py rewrites the Roboflow export into YOLO format, train.py fine-tunes the network while logging metrics under runs/microplastics, and inference is carried out via testapp.py or live\_inference.py, which write their artefacts to runs/detect.

# 2. Codebase Walkthrough

## 2.1 train.py

train.py orchestrates fine-tuning through two functions:

* **parse\_args() (train.py:11-50):** Builds an argparse parser for dataset configuration, hyperparameters, and runtime options. The script expects a --model argument (used via args.model), which means an explicit model path must be injected before execution.
* **main() (train.py:53-74):** Instantiates YOLO with the supplied checkpoint, handles optional resume mode, and forwards training arguments to model.train. Ultralytics automatically logs metrics, checkpoints, and visual artefacts to runs/microplastics.

## 2.2 scripts/prepare\_dataset.py

Key functions ensuring the dataset matches YOLO conventions:

* **load\_annotations (lines 22-50):** Parses Roboflow CSV rows, normalises bounding boxes into YOLO centre/width/height format, and clamps them for numerical stability.
* **copy\_images (lines 53-60):** Copies JPEGs into target/images/<split>, skipping files that already exist.
* **write\_label\_file (lines 63-69):** Writes YOLO label text files with class id and normalised box coordinates.
* **process\_split (lines 72-99):** Coordinates annotation loading, image copying, and label writing per split; returns image and label counts for logging.
* **main (lines 101-137):** Parses CLI arguments (--data-root, --target-root, --class-id) and iterates over train/valid splits to call process\_split.

## 2.3 live\_inference.py

(No need to run this file, can also be removed)

* **parse\_args() (lines 12-38):** Collects weights path, video source, confidence threshold, and device string. Defaults point to the best checkpoint saved under runs.
* **main() (lines 41-77):** Opens the video stream via OpenCV, instantiates YOLO, and performs frame-by-frame prediction. Results are rendered with results[0].plot() and displayed until the user presses q. Errors are surfaced via sys.stderr.

## 2.4 testapp.py

* **Standalone inference snippet (lines 1-11):** Loads the trained checkpoint (runs/microplastics/weights/best.pt) and calls model.predict with save=True, writing annotated images to runs/detect/predict\*. The optional print outputs bounding boxes debugging.

## 2.5 Supporting Configuration

* **configs/microplastics.yaml:** Defines dataset root (dataset/) and class mapping for Microplastic (class id 0).
* **requirements.txt:** Lists ultralytics and opencv-python, the only Python dependencies required by the scripts.

# 3. End-to-End Working

1. **Dataset conversion:** Run scripts/prepare\_dataset.py to populate dataset/images and dataset/labels from Roboflow exports.
2. **Training:** Invoke train.py with an explicit --model argument (for example yolov8n.pt) to fine-tune the detector while logging metrics into runs/microplastics.
3. **Evaluation artefacts:** Ultralytics automatically exports metrics, confusion matrices, and qualitative batches during training for inspection.
4. **Batch or live inference:** Use testapp.py for saved images and live\_inference.py for webcams/streams; both rely on YOLO.predict to produce annotated frames.

# 4. Evaluation Metrics (results.csv)

* **train/box\_loss, train/cls\_loss, train/dfl\_loss:** Bounding-box regression, classification, and distribution focal losses recorded per epoch on the training split.
* **val/box\_loss, val/cls\_loss, val/dfl\_loss:** Equivalent losses measured on the validation split to monitor generalisation.
* **metrics/precision(B):** Bounding-box precision computed on the validation set.
* **metrics/recall(B):** Validation recall measuring how many ground-truth boxes were recovered.
* **metrics/mAP50(B):** Mean Average Precision at IoU 0.50 for the Microplastic class.
* **metrics/mAP50-95(B):** mAP averaged across IoU thresholds from 0.50 to 0.95 (COCO-style).
* **lr/pg0-2:** Learning rate for Ultralytics parameter groups recorded for reproducibility.
* **time:** Per-epoch runtime in seconds, tracking training throughput.

These metrics are necessary to quantify localisation accuracy (precision/recall, mAP) and to debug optimisation dynamics (loss curves and learning rates). Monitoring both loss terms and detection metrics ensures the detector improves without overfitting.

# 5. Observed Results from results.csv

**Training span:** 50 epochs recorded.

* Best precision 0.81761 achieved at epoch 44
* Best recall 0.66725 achieved at epoch 20
* Best mAP50 0.73422 achieved at epoch 34
* Best mAP50-95 0.35409 achieved at epoch 34

Final epoch (epoch 50) metrics:

* Precision(B): 0.79526
* Recall(B): 0.64668
* mAP50(B): 0.72655
* mAP50-95(B): 0.34844
* Validation box loss: 1.83383
* Validation cls loss: 1.10035
* Validation dfl loss: 1.28409

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Interpretation** | **Comment** |
| **Precision (0.79)** | Good | Model makes relatively few false positives. |
| **Recall (0.65)** | Moderate | Model misses some objects — can improve detection coverage. |
| **mAP@50 (0.73)** | Solid | Model has decent detection ability. |
| **mAP@50–95 (0.35)** | Moderate | Localization could be sharper. |
| **Losses (~1–1.8)** | Acceptable | Model is learning, but can be fine-tuned further. |

**1. Precision (B): 0.79526**

* **Meaning:** Of all the predicted bounding boxes labeled as positive (detections), about **79.5% are correct**.
* **Interpretation:**
  + High precision → Few **false positives** (the model doesn’t predict boxes when there’s nothing there).
  + Moderate precision (around 0.8) → Reasonably good; the model is fairly confident in its predictions.

**2. Recall (B): 0.64668**

* **Meaning:** Of all the true objects in the validation set, the model correctly detected about **64.7%** of them.
* **Interpretation:**
  + Lower recall → The model **misses some objects** (false negatives).
  + This suggests that the model could benefit from improvements in sensitivity — perhaps more training data, better augmentation, or tuning anchor sizes.

The best checkpoints (epochs 34 and 44) correspond to peaks in mAP and precision, indicating improved localisation as training progressed.

# 6. runs/ Artefact Review

The following tables capture every image stored under runs/, explaining the generation pathway and the diagnostic value of each artefact.

## 6.1 Training Artefacts (runs/microplastics)

|  |  |  |  |
| --- | --- | --- | --- |
| Image | Generated By | Representation | Significance |
| runs/microplastics/labels.jpg | Ultralytics dataloader routine within `YOLO.train` (train.py:65-74). | Composite mosaic of training samples with ground-truth Microplastic bounding boxes. | Quick label audit produced automatically at training start to verify annotations were parsed correctly. |
| runs/microplastics/train\_batch0.jpg | Augmented batch preview emitted by `YOLO.train` (train.py:65-74). | Grid of images showing on-the-fly augmentation and labels for the first dataloader batch. | Confirms augmentation pipeline and class coverage without inspecting data manually. |
| runs/microplastics/train\_batch1.jpg | Same `YOLO.train` batching preview as above. | Second snapshot of augmented training samples with bounding boxes. | Ensures variability across batches and checks for annotation drift. |
| runs/microplastics/train\_batch2.jpg | Same `YOLO.train` batching preview as above. | Third snapshot of augmented training samples. | Demonstrates consistency of label application throughout early batches. |
| runs/microplastics/train\_batch1480.jpg | Later-epoch batch preview saved during `YOLO.train` loop. | Augmented samples captured mid-training (iteration ~1480). | Allows comparison of augmentation behaviour later in training. |
| runs/microplastics/train\_batch1481.jpg | Later-epoch batch preview saved during `YOLO.train` loop. | Additional iteration snapshot of training data. | Helps verify batch-to-batch diversity at the same stage. |
| runs/microplastics/train\_batch1482.jpg | Later-epoch batch preview saved during `YOLO.train` loop. | Further snapshot from iteration series around 1482. | Completes the set of consecutive batches for diagnostic comparison. |
| runs/microplastics/val\_batch0\_labels.jpg | Validation loader inside `YOLO.train` during final evaluation. | Ground-truth labels for the zeroth validation batch. | Reference view for comparing model predictions against human annotations. |
| runs/microplastics/val\_batch0\_pred.jpg | Validation predictions computed in `YOLO.train` (Ultralytics evaluation). | Model predictions for the same validation batch, rendered with bounding boxes and confidences. | Visual check for detection quality on held-out data. |
| runs/microplastics/val\_batch1\_labels.jpg | Validation loader ground-truth export. | Labels for validation batch 1. | Supports manual inspection of additional validation samples. |
| runs/microplastics/val\_batch1\_pred.jpg | Validation prediction export from `YOLO.train`. | Predicted detections for validation batch 1. | Facilitates side-by-side comparison with labels.jpg counterpart. |
| runs/microplastics/val\_batch2\_labels.jpg | Validation loader ground-truth export. | Labels for validation batch 2. | Extends coverage of validation samples available for review. |
| runs/microplastics/val\_batch2\_pred.jpg | Validation prediction export from `YOLO.train`. | Predicted detections for validation batch 2. | Demonstrates model behaviour across more validation imagery. |
| runs/microplastics/BoxP\_curve.png | Ultralytics metrics export triggered by `YOLO.train`. | Precision curve across confidence thresholds for bounding box detections. | Used to understand precision trade-offs across thresholds. |
| runs/microplastics/BoxR\_curve.png | Ultralytics metrics export triggered by `YOLO.train`. | Recall curve across confidence thresholds. | Shows how recall varies with decision thresholds. |
| runs/microplastics/BoxF1\_curve.png | Ultralytics metrics export triggered by `YOLO.train`. | F1 score curve for detection thresholds. | Highlights threshold yielding best F1 balance between precision and recall. |
| runs/microplastics/BoxPR\_curve.png | Ultralytics metrics export triggered by `YOLO.train`. | Precision-recall curve aggregated across evaluation set. | Summarises detector performance by mapping recall vs precision. |
| runs/microplastics/confusion\_matrix.png | Confusion matrix plotting inside Ultralytics validation. | Absolute count confusion matrix for predictions vs ground-truth. | Quantifies false positives/negatives for the Microplastic class. |
| runs/microplastics/confusion\_matrix\_normalized.png | Same validation routine as above. | Normalized confusion matrix (per-class percentages). | Shows proportional accuracy, independent of class frequency. |
| runs/microplastics/results.png | Training history plot saved by `YOLO.train`. | Curves for losses and mAP across 50 epochs. | Visual reference for training convergence and validation trends. |

## 6.2 Inference Artefacts (runs/detect)

|  |  |  |  |
| --- | --- | --- | --- |
| Image | Generated By | Representation | Significance |
| runs/detect/predict/tests.jpg | `YOLO(...).predict` with `save=True` (see testapp.py:4-9). | Annotated inference for input image tests.jpg saved under the default predict run. | Verifies that single-image prediction writes visual overlays to runs/detect. |
| runs/detect/predict2/dscn0569.jpg | Same `model.predict` invocation path as testapp.py (or CLI equivalent). | Detection result overlay for source image dscn0569.jpg. | Demonstrates model transfer to a different microscope capture. |
| runs/detect/predict3/images.jpg | Ultralytics prediction pipeline. | Detection overlay for images.jpg. | Another qualitative sample for checking inference output. |
| runs/detect/predict4/micro.jpg | Ultralytics prediction pipeline. | Detection overlay for micro.jpg. | Used to inspect predictions on an alternate microscope frame. |
| runs/detect/predict5/LOW.jpg | Ultralytics prediction pipeline. | Detection overlay for LOW.jpg. | Captures inference on a low-magnification or low-quality input to gauge robustness. |

Each detection image contains the annotated output from Ultralytics model.predict, showing the Microplastic detections on the corresponding input frame. The training artefacts originated during YOLOv8 training and document dataset integrity as well as performance trends.