Plastic Waste Segregation System – Engineering Handover Report

Author: Shahan Sahay

Date: July 2025

# 1. Executive Summary

This report outlines the design, development, and implementation of a real-time plastic waste classification system, created as part of an internship project. The goal was to build an intelligent segregation system that uses computer vision and deep learning (YOLOv8) to identify and categorize plastic waste materials like PET, HDPE, LDPE, PP, PS, and CAN from a conveyor belt stream.  
  
To reduce manual labeling overhead and improve retraining accuracy, the system incorporates automatic verification using Google Gemini and OpenAI GPT APIs. The entire system is modular, script-driven, and designed for real-time inference, dataset growth, and future hardware deployment. This documentation serves as a handover guide for the next engineer to maintain and extend the system.

# 2. System Overview

The system is deployed using a Logitech C920 webcam, fixed at a height of 70 cm above a circular conveyor system. This placement ensures a top-down, stable view of the waste flow.  
  
The architecture follows this flow:  
- Frame capture via webcam or video  
- Inference via a YOLOv8 model trained on six plastic categories  
- Confidence thresholding (detections below 0.5 confidence are flagged)  
- Gemini/GPT-based verification of flagged results  
- Verified samples are logged and stored for retraining  
  
The aim is continuous improvement through semi-automated feedback.

# 3. Dataset Collection and Labeling

1. Video Acquisition:  
 - Videos were recorded from the actual conveyor setup using the script `scripts/capture\_dataset.py`. It records Full HD video and saves it in `captured\_videos/`.  
2. Frame Extraction:  
 - Every 15th frame was extracted from recorded videos to ensure dataset diversity.  
3. Manual Labeling:  
 - Uploaded to Roboflow and annotated into PET, HDPE, LDPE, PP, PS, CAN.  
4. Dataset Statistics:  
 - ~4900 images labeled.  
 - YOLOv8 format with .txt and .jpg.

# 4. YOLOv8 Model Training

Model Training was performed using YOLOv8m from Ultralytics with the following configuration:  
- Model: yolov8m.pt  
- Epochs: 250  
- Image Size: 640x640  
- Batch Size: 16  
- GPU: NVIDIA RTX 4060  
- Output Weights: best.pt  
- Augmentation via Roboflow: flipping, rotation, brightness

# 5. Inference and Verification Pipeline

Scripts and their functions:  
- `scripts/detect\_live\_cam.py`: Real-time webcam detection using best.pt  
- `scripts/detect\_and\_queue.py`: Detect objects from videos, store results in `queue/`  
- `scripts/verify\_with\_gemini.py`: Verify multi-labels using Gemini, store verified data  
- `scripts/verify\_single\_label.py`: Validates single-label files with Gemini  
- `scripts/verify\_confidence\_fallback.py`: Verifies with Gemini, falls back to GPT if unsure  
- `scripts/camera\_check.py`: Identifies working camera index  
- `scripts/checker\_valid.py`: Sanity check script to test code functionality

# 6. Windows Environment Setup (New PC)

1. Install Python 3.10+ from: https://www.python.org (Add to PATH)  
2. Install Git from: https://git-scm.com  
3. Clone the repository:  
 git clone https://github.com/shahansahay/plastic-waste-segregation.git  
4. Navigate:  
 cd plastic-waste-segregation  
5. Create virtual environment:  
 python -m venv .venv  
 .venv\Scripts\activate  
6. Install dependencies:  
 pip install --upgrade pip  
 pip install ultralytics opencv-python torch rich requests  
7. Create `.env` with:  
 GEMINI\_API\_KEY=your\_gemini\_key  
 GPT\_API\_KEY=your\_gpt\_key

# 7. Verification Automation

`verify\_confidence\_fallback.py` automates label validation:  
- Crops objects from images  
- Sends image + labels to Gemini  
- If Gemini is unsure, falls back to GPT-4o  
- If verified, stores in: dataset/images/verified/ and dataset/labels/verified/

# 8. Folder Structure Requirements

Required folders:  
- queue/  
- VIDEOS/  
- captured\_videos/  
- dataset/images/verified/  
- dataset/labels/verified/

# 9. Future Recommendations

- Integrate air jets or robotic arms for physical sorting  
- Add OCR (Tesseract or PaddleOCR) for brand-level classification  
- Build a Streamlit or Dash dashboard for live classification  
- Expand dataset under various lighting and angles  
- Deploy optimized version on NVIDIA Jetson Orin Nano or Xavier NX