|  |
| --- |
| Title: Waste Detection by using YOLOv8 |
| Done by : Pritesh Ram Keshri Project Number : 05, Submission Date: 16th June 2025 |
| **Overview:**  This project presents an intelligent waste segregation system that utilizes the **YOLOv8 object detection** model to identify and classify waste items captured via a live webcam stream. The waste items are categorized into **recyclable**, **non-recyclable**, and **hazardous** types. The model was trained using a curated dataset and deployed through a **Streamlit** application for real-time monitoring and detection. |
| **Libraries used:**   * **Ultralytics YOLOv8 –** For object detection and model training * **OpenCV (cv2) –** Video frame capture and manipulation * **Streamlit –** Web interface for real-time detection * **NumPy** – Array operations * **Matplotlib –** Visualization of results * **Pandas –** Dataframe support (if needed during analysis) |
| **Dataset Details:**   * **Source:** [**Roboflow Waste Detection Dataset**](https://universe.roboflow.com/ai-project-i3wje/waste-detection-vqkjo/model/3) * **Format:** Labeled dataset for YOLO object detection * **Classes:** * **Recyclable:** cardboard\_box, can, plastic\_bottle\_cap, plastic\_bottle, reusable\_paper * **Non-Recyclable:** plastic\_bag, scrap\_paper, stick, plastic\_cup, snack\_bag, plastic\_box, straw, plastic\_cup\_lid, scrap\_plastic, cardboard\_bowl, plastic\_cutlery * **Hazardous:** battery, chemical\_spray\_can, chemical\_plastic\_bottle, chemical\_plastic\_gallon, light\_bulb, paint\_bucket |
| **APIs Integrated:**  No external APIs integrated. Application is deployed directly using Streamlit. |
| **Source code 1: File Name : Clone & Run Setup (setup.sh)**  git clone https://github.com/boss4848/waste-detection.git  cd waste-detection  pip install -r requirements.txt  streamlit run app.py |
| **Source code 2: File name: Streamlit App (app.py)**  import streamlit as st  from ultralytics import YOLO  import cv2  model = YOLO("best.pt")  st.title("Intelligent Waste Detection")  frame = st.camera\_input("Capture a Frame")  if frame:  img = cv2.imdecode(np.frombuffer(frame.read(), np.uint8), 1)  results = model(img)  annotated\_img = results[0].plot()  st.image(annotated\_img, caption="Detection Output") |
| **Source code 3: Helper Functions (helper.py)**  def load\_model(model\_path="best.pt"):  from ultralytics import YOLO  return YOLO(model\_path)  def predict\_and\_draw(model, frame):  results = model(frame)  return results[0].plot() |
| **Source Code 4: Model Training (train.py)**    yolo task=detect mode=train model=yolov8n.pt data=your\_dataset.yaml epochs=50 imgsz=640 |
|  |
| **Output screenshots:**  **1. Streamlit Web Interface:** Below image shows the streamlit web interface of the waste  Detection **.**    **2. Prediction Samples**:Here’s how you can see the prediction sample of the waste classification using YOLOv8.    **3. Real-Time Detection**: Given two images shows the real-time detection of the waste which also classifies the hazardous items. |
| **What you learned:**  Through this project, I gained practical knowledge of:   * Object detection with YOLOv8 * Real-time video processing using OpenCV * Designing and deploying applications with Streamlit * Understanding class imbalance and annotation techniques * Model training, validation, and testing using custom datasets |
| **What the Skills you gained:**   * Built real-time computer vision applications * Trained object detection models using YOLO * Handled labeled datasets and YOLO formatting * Web deployment using Streamlit * Visualized and interpreted model predictions |
| **Real Time Applications:**   * **Smart Dustbins:** Automated waste sorting systems * **Urban Sanitation Monitoring:** Detect improper waste disposal * **Recycling Plants:** Categorize incoming waste streams * **Public Surveillance:** Identify hazardous material disposal * **Educational Tools:** Promote waste awareness through demos |
| **Further Enhancement Suggestions:**   * Integrate robotic arm controls for automated sorting * Include voice alerts or LED indicators * Deploy on edge devices like Raspberry Pi * Implement cloud-based dashboards for municipal monitoring * Add feedback system for missed/wrong detections * Train with larger and diversified datasets   **Training Performance (Example):**   | **Epoch** | **Training mAP50** | **Validation mAP50** | **Loss** | | --- | --- | --- | --- | | **1** | **42.1%** | **40.8%** | **0.72** | | **10** | **81.6%** | **78.9%** | **0.25** | | **50** | **92.3%** | **91.0%** | **0.09** | |