**Machine Learning Project Documentation Deployment Report**

**Title - Real-Time Recycling Sorting Using Deep Learning**

**Group no: Group 8**

**Group Members:**

1. Tekia Tekle

2. Firehiwet Zerihun

3. Biniyam Demissew

4. Daniel Alemu

5. Eyob Sisay

6. Haymanot Tamrat

# Machine Learning Project Documentation

## Deployment

## 1. Overview

The deployment phase of this project transforms our machine learning models into a functional, real-time waste classification system. The goal is to build a practical and user-accessible application that uses deep learning to classify images of trash into predefined categories such as plastic, paper, metal, glass, cardboard, and general waste. This contributes to more efficient recycling efforts and smart city initiatives.

Our deployment leverages a YOLOv8 (You Only Look Once version 8) object detection model, trained on the TrashNet dataset, along with a lightweight web application built using Streamlit. This setup allows users to either upload an image or capture one directly from their device’s camera. Once an image is submitted, the application returns a real-time classification prediction with annotated bounding boxes.

The application was hosted using **ngrok**, which creates a secure tunnel from the Colab environment to the web, allowing live testing of the application through a public URL. This setup simulates a real deployment environment without requiring a full backend infrastructure.

**2. Model Serialization**

To enable deployment and reusability of the trained YOLOv8 model, we used model serialization. Serialization refers to saving the model’s architecture, parameters, and weights into a file format that can be reloaded without retraining.

* **Model Used:** YOLOv8n (nano variant for efficiency on low-resource devices)
* **Training Dataset:** TrashNet (containing 6 waste classes)
* **Saved Format:** .pt (PyTorch checkpoint format)

After training, the best-performing model was saved using:



This file preserves the model’s learned weights and can be easily reloaded during inference for real-time classification.

**3. Model Serving**

The serialized model was served via a user-facing application created using **Streamlit**. Streamlit is an open-source Python framework that allows quick deployment of machine learning models into interactive web apps. It was chosen because:

* It supports image upload and camera input.
* It’s compatible with the Google Colab environment.
* It requires minimal code to create powerful interfaces.

The frontend included:

* Tabs for **image upload** and **camera input**.
* Real-time display of prediction results using bounding boxes.
* Class label display with confidence scores.

We integrated the YOLOv8 model directly into the Streamlit app:



After receiving the image, the model performs inference and displays results.

**4. API Integration**

To make the app accessible without a dedicated web server, **ngrok** was used to tunnel the Streamlit server. This allowed testing the web app remotely, even from mobile devices.

Steps:

1. Start the Streamlit app:
2. !streamlit run app.py &> /dev/null &
3. Launch ngrok tunnel:
4. from pyngrok import ngrok
5. public\_url = ngrok.connect(8501)
6. print ("App is live at:", public\_url)

Users could visit the ngrok-generated public URL to interact with the app in real time.

**5. Security Considerations**

Although ngrok is primarily used for development, security precautions were taken:

* Access was limited to authorized test users.
* No personal data was collected.
* Uploaded images were used solely for classification and were not stored.

For production deployment, it’s recommended to:

* Use HTTPS on all endpoints.
* Integrate authentication.
* Secure model endpoints using API gateways and usage logs.

**7. Monitoring and Logging**

During this deployment phase:

* Logs were viewed via the Colab terminal and Streamlit console.
* Performance metrics like mAP, precision, and recall were printed during model training.
* User interactions (upload/camera inputs) were handled in real-time without backend storage.

In future iterations, proper logging and monitoring tools (e.g., TensorBoard, Prometheus) can be integrated to track:

* Model inference times
* Usage patterns
* Accuracy over time

**8. Future Improvements**

This deployment sets a strong foundation for intelligent waste classification systems. In future versions, we aim to:

* Deploy on a cloud platform (e.g., Streamlit Cloud, AWS, or Hugging Face Spaces)
* Extend to real-time video streams from CCTV
* Add multilingual support and audio guidance
* Export predictions to CSV or database logs