Waste Sorting ML Project Report

1. Introduction

With the continuous increase in global waste production due to industrialization and population growth, the need for intelligent waste management systems has become more crucial than ever. Traditional waste sorting methods rely heavily on manual labor, which makes the process time-consuming, error-prone, and economically inefficient.

This project aims to develop an automated recycling waste-sorting system using classical machine learning techniques to classify garbage images into predefined categories such as cardboard, glass, metal, paper, plastic, and trash. By using texture-based features and lightweight models, this system seeks to provide a practical and efficient solution that can be deployed in low-resource environments, without requiring deep learning infrastructure.

The proposed solution includes a web-based interface that allows users to upload waste images and receive instant predictions of the waste type. This system could be integrated into smart recycling bins or sorting lines to improve the accuracy and speed of waste classification, ultimately supporting better recycling practices and environmental sustainability.

2. Literature Review

Waste classification has been a widely studied problem in the field of computer vision and machine learning. Many existing solutions rely on deep learning models such as Convolutional Neural Networks (CNNs), which provide high accuracy but require large datasets, GPU processing power, and long training times. However, for systems targeting low-cost deployment and real-time response, classical machine learning algorithms offer a viable alternative.

One of the widely used datasets for garbage classification is the TrashNet dataset, which contains labeled images across six categories: cardboard, glass, metal, paper, plastic, and trash. Previous work using TrashNet often involves deep learning, such as fine-tuned CNNs

or transfer learning models like ResNet or VGGNet, achieving accuracies upwards of 85–90%.

In contrast, traditional methods using hand-crafted features like color histograms, texture descriptors, and edge detection filters combined with machine learning classifiers (e.g., K-Nearest Neighbors, SVM, Decision Trees) have shown reasonable performance with far lower computational costs. Among these, Gray-Level Co-occurrence Matrix (GLCM) is a well-established method for extracting texture features, and it has been successfully applied in medical image analysis, remote sensing, and industrial inspection.

This project adopts a classical approach by combining GLCM for texture feature extraction with a manually implemented K-Nearest Neighbors (KNN) classifier. This methodology is computationally efficient, interpretable, and suitable for educational and resource-constrained applications.

3. Methodology

The implementation of the recycling waste-sorting system followed a structured classical machine learning pipeline, consisting of data acquisition, preprocessing, feature extraction, model training, evaluation, and deployment via a web interface.

3.1 Data Collection and Augmentation

We used the TrashNet dataset, which includes 2,527 labeled images categorized into six classes: cardboard, glass, metal, paper, plastic, and trash. Each image was loaded and resized to a fixed dimension of 128x128 pixels, and then converted to grayscale to simplify the computation of texture-based features.

To improve the model's robustness and reduce overfitting, basic image augmentation techniques were applied, including:

- Horizontal flipping

- Vertical flipping

This effectively doubled the dataset size and introduced more variations in image orientations.

3.2 Feature Extraction using GLCM

We used Gray-Level Co-occurrence Matrix (GLCM) to extract texture-based features from the grayscale images. GLCM captures the spatial relationships of pixel intensities and provides several statistical descriptors. From each GLCM, we extracted the following features:

- Energy
- Contrast
- Dissimilarity
- Homogeneity
- Correlation
- Angular Second Moment (ASM)

These six features form the final feature vector for each image.

3.3 Classification using K-Nearest Neighbors (KNN)

Instead of relying on scikit-learn's built-in classifiers, we implemented a custom KNN classifier from scratch. For each test sample:

- Euclidean distances were computed between the test feature and all training features.
- The k nearest neighbors (in our case, k=1) were selected.
- The majority label among these neighbors was assigned as the prediction.

This manual implementation ensured a deeper understanding of the algorithm and its behavior with real data.

3.4 Model Training and Evaluation

The dataset was split into training and testing sets using an 80-20 split. The training set was used to build the feature set and train the KNN model, while the test set was used for performance evaluation. Accuracy was calculated by comparing predicted labels with actual labels.

3.5 Web Deployment

To provide a user-friendly interface, we developed a Flask-based web application. The app allows users to upload an image of waste, which is then:

- Preprocessed
- Converted to grayscale
- Features extracted using GLCM
- Classified using the trained model

The app returns the predicted waste category along with the model accuracy and a preview of the uploaded image.

4. Result Analysis

4.1 Classification Accuracy

After training and testing the model using an 80-20 data split, the system achieved a classification accuracy of approximately 82.45% on the test set.

Metric	Value
Accuracy	82.45%
Classes	6 (cardboard, glass, metal, paper, plastic, trash)
Feature Type	GLCM (6 features per image)
Classifier	Manual KNN (k=1)

4.2 Strengths of the Approach

- Lightweight: Suitable for embedded or low-power devices.

- Explainable: Offers interpretability and transparency.

- Custom Implementation: Educational and controllable.

4.3 Limitations and Challenges

- Limited Generalization: Struggles with visually similar materials.

- Sensitivity to Noise: Performance drops with poor-quality images.

- Imbalanced Classes: May hinder learning for underrepresented classes.

4.4 Future Improvements

- Feature Fusion: Combine GLCM with other features like color or edges.

- Hyperparameter Tuning: Try other k values.

5. Implementation

This section describes the system's architecture, code structure, and key implementation details used to build the recycling automation waste-sorting system.

5.1 System Overview

The system follows a modular design consisting of the following major components:

Data Loader: Responsible for loading, resizing, converting, and augmenting images from the dataset.

Feature Extractor: Computes GLCM matrices and derives statistical features.

Classifier: A custom implementation of the K-Nearest Neighbors algorithm.

Evaluation Module: Measures the accuracy of the model.

Flask Web App: Provides a simple interface for image upload and real-time classification.

5.2 Directory Structure

```
project/
│
├──── archive/                # Dataset folder
│    └──── Garbage classification/
│        ├──── cardboard/
│        ├──── glass/
│        ├──── metal/
│        ├──── paper/
│        ├──── plastic/
│        └──── trash/
│
├──── app.py                  # Main Flask application
├──── glcm_utils.py           # GLCM feature extraction functions
├──── knn_classifier.py       # Manual KNN classifier implementation
├──── requirements.txt        # Python dependencies
```

└──── templates/

└──── index.html # Web interface HTML template

5.3 Key Functions and Modules

- Preprocessing

  preprocess_image(image): Resizes the image to 128x128 and converts it to grayscale.

- Feature Extraction

  compute_glcm(image, distance, angle): Computes a Gray-Level Co-occurrence Matrix for a given direction and distance.

  extract_features_from_glcm(glcm): Extracts six statistical texture features.

- Model Training and Prediction

  train_model(X_train, y_train): Encodes labels and extracts GLCM features for the training data.

  evaluate_model(...): Compares predicted vs. true labels to calculate accuracy.

  knn_predict(...): A custom function to predict a test sample's class using Euclidean distance and majority voting.

- Web Application

  Built using Flask, with a minimal HTML template.

  Users can upload an image, which is classified on the backend and the result is displayed on the same page.

The classification output includes:

- Predicted class

- Model accuracy
- Uploaded image preview

5.4 Technologies Used

Technology	Purpose
Python	Core programming language
OpenCV	Image processing
NumPy	Numerical computations
Scikit-learn	Label encoding and accuracy metrics
Imgaug	Image augmentation
Flask	Web app framework
PIL / base64	Image handling and display in HTML

6. Conclusion

This project demonstrated the feasibility of using classical machine learning techniques for automated waste classification based on image data. By leveraging texture features extracted using GLCM and a custom KNN classifier, the system achieved a classification accuracy of 82.45% while maintaining low computational overhead.

The manual implementation approach ensured transparency, educational value, and adaptability, making it ideal for use in academic settings or low-resource environments. Furthermore, the web-based interface provided an accessible way for users to interact with the model and receive real-time feedback on waste categorization.

Although the system has limitations, such as sensitivity to lighting and difficulty distinguishing visually similar materials, future enhancements like feature fusion and real-time deployment could significantly improve its performance and usability.