Machine Learning Project
Material Stream Identification System

Team Members:

Name	ID
Osama Yasser Farouk	20220057
Hasnaa Ahmed	20221232
Nada Ali Hassan	20221180
Shahd Mohammed Mahmoud	20221086
Abdelrahman Bakry	20221103

Github Repo Link:
https://github.com/osamayasser19/ML-Project

Introduction

The efficient and automated sorting of post-consumer waste represents a critical bottleneck in achieving global circular economic goals. Traditional manual sorting is often slow, inconsistent, and unable to keep up with the increasing volume of recyclable materials. To address this challenge, this project involves the development of an Automated Material Stream Identification (MSI) System.

This project aims to implement a vision-based system for the automated classification of seven material classes. The following sections document the complete machine learning workflow: from enhancing the dataset through data augmentation and converting pixel data into numerical feature vectors, to conducting a comparative analysis between k-NN and SVM classifiers. The report concludes with the details of our real-time system deployment.

1. Data Cleansing and Augmentation

Data cleaning: Implemented a function **(cleanImages)** to loop over all classes folders and remove corrupted images or images with size equal to zero

Augmentation: Implemented Using **Augmentor** library to prevent overfitting. The operations done on images are as follows:

- Random flipping was prioritized as it represents the most frequent change for materials
- random 90-degree rotations and random cropping
- Random brightness, distortion and color adjustments were applied to simulate varying lighting conditions

Classes Sizing: I set the sample generated to be the max between (600 -photos count in class) and (30% of photos count in class), I choose 600 because the class with most photos needs to be at approx. 600 images total

2. Feature Extraction (Image to Vector)

To represent waste material images effectively, several feature extraction techniques were implemented. Both handcrafted descriptors and deep learning–based features were explored.Several handcrafted feature descriptors were implemented and evaluated during development to analyze texture, shape, and color information:

HOG features were extracted from grayscale images resized to 128×128 pixels. The descriptor captures local edge and shape information by computing gradient orientations over small spatial regions.

LBP features were used to capture local texture patterns, which are particularly relevant for material recognition tasks. The grayscale image was encoded using:

- Number of neighbors (P): 8
- Radius (R): 1
- Method: Uniform

Color information was extracted using a 3D histogram in the HSV color space, which is more robust to lighting changes than RGB. The histogram was computed with:

- 8 bins per channel (H, S, V)
- Histogram normalization applied

To obtain high-level semantic representations, deep features were extracted using a **pre-trained ResNet50 convolutional neural network**, trained on the ImageNet dataset.Each image was preprocessed using the ResNet50 preprocessing function and passed through the network to obtain a fixed-length feature vector. These deep features capture complex visual patterns such as texture, shape, and object-level semantics, making them highly suitable for material classification.

Before training the classifiers, all extracted features were standardized using **StandardScaler**, ensuring zero mean and unit variance. This step is crucial for distance-based classifiers such as k-Nearest Neighbors and margin-based classifiers like Support Vector Machines.

The processed data were saved as:

- `X_features.npy` — standardized feature matrix
- `y_labels.npy` — corresponding numeric class labels

The fitted scaler was also saved (`scaler.save`) to ensure consistency between training and real-time inference.

3. Classifier Implementation

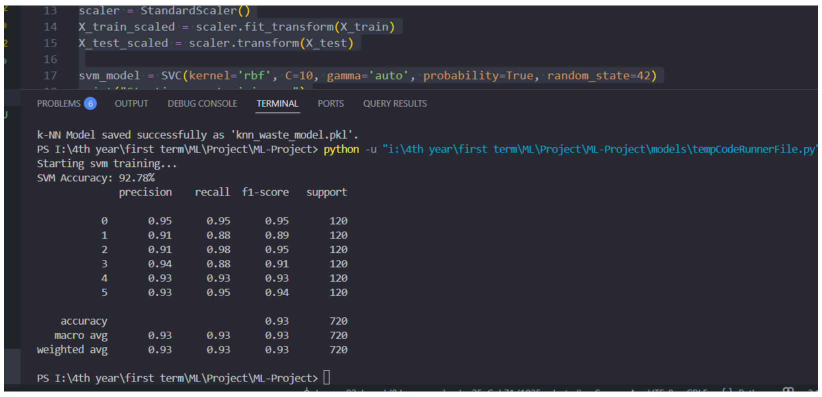
The extracted deep feature vectors were used to train and evaluate two supervised classifiers: k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM). The dataset was split into training and testing sets using an 80/20 stratified split to preserve class distribution. Feature standardization was applied using a StandardScaler to improve convergence and ensure fair distance computation.

- **Support Vector Machine (SVM):**

Support Vector Machine (SVM) classifier was employed to provide a margin-based, discriminative classification approach. The same standardized feature vectors and stratified train–test split were used to ensure a fair comparison between classifiers. Feature scaling was essential to optimize the SVM's performance and convergence behavior.The SVM model was implemented using:

- Radial Basis Function (RBF) kernel
- Regularization parameter C=10C = 10C=10
- Kernel coefficient γ=auto\gamma = \text{auto}γ=auto
- Probability estimation enabled

The RBF kernel allows the classifier to handle non-linear class boundaries commonly present in visual feature spaces. Model performance was evaluated using accuracy and detailed classification metrics. The trained SVM model and its corresponding scaler were saved to support consistent real-time inference and deployment with accuracy of **92.78%** .


```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm_model = SVC(kernel='rbf', C=10, gamma='auto', probability=True, random_state=42)
```

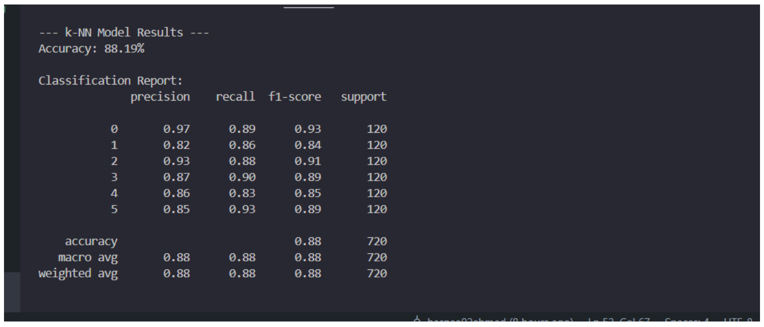
```
k-NN Model saved successfully as 'knn_waste_model.pkl'.
PS I:\4th year\first term\ML\Project\ML-Project> python -u "i:\4th year\first term\ML\Project\ML-Project\models\tempCodeRunnerFile.py"
Starting svm training...
SVM Accuracy: 92.78%
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       120
           1       0.91      0.88      0.89       120
           2       0.91      0.98      0.95       120
           3       0.94      0.88      0.91       120
           4       0.93      0.93      0.93       120
           5       0.93      0.95      0.94       120

    accuracy                           0.93       720
   macro avg       0.93      0.93      0.93       720
weighted avg       0.93      0.93      0.93       720

PS I:\4th year\first term\ML\Project\ML-Project> 
```

- ## K-Nearest Neighbors (k-NN):

To classify waste material images based on the extracted feature representations, a k-Nearest Neighbors (k-NN) classifier was implemented and evaluated. The standardized feature dataset was divided into training and testing subsets using an 80%–20% stratified split to maintain class distribution consistency. Feature standardization was applied prior to training to ensure reliable distance computation, which is essential for distance-based classifiers.

The k-NN classifier was configured with:

- Number of neighbors (k): 9
- Distance-based weighting, giving higher influence to closer samples


```
--- k-NN Model Results ---
Accuracy: 88.19%

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       120
           1       0.82      0.86      0.84       120
           2       0.93      0.88      0.91       120
           3       0.87      0.90      0.89       120
           4       0.86      0.83      0.85       120
           5       0.85      0.93      0.89       120

    accuracy                           0.88       720
   macro avg       0.88      0.88      0.88       720
weighted avg       0.88      0.88      0.88       720
```

During inference, the classifier determines the label of an input sample by analyzing the most similar feature vectors in the training set. Model performance was assessed using classification accuracy and detailed class-wise precision, recall, and F1-score metrics. After training, the k-NN model was serialized and saved for integration into the real-time material identification system with accuracy of **88.19%**.

4. Rejection Mechanism (The "Unknown" Class)

To handle inputs that do not belong to any of the trained classes, a rejection mechanism was implemented to identify unknown materials during real-time inference.

For the **k-NN classifier**, the rejection mechanism is based on the **average normalized distance** between the test sample and its nearest neighbors used in deployment. Specifically:

- Distances to the k nearest neighbors are computed in the standardized feature space.
- Distances are normalized by the minimum and maximum distances among neighbors.
- The average normalized distance is compared against a threshold (empirically,like 0.9).

If the average distance exceeds the threshold, the sample is classified as **“Unknown”**; otherwise, the standard k-NN prediction is used. This approach leverages the distance-based nature of k-NN to quantify similarity and detect outliers effectively.

For the **SVM classifier,** probability estimates are used to implement the rejection mechanism:

- The SVM is trained with probability estimation enabled (`probability=True`).
- For each test sample, the maximum predicted class probability is computed.
- If this maximum probability falls below a pre-defined confidence threshold, the sample is labeled as **“Unknown”**; otherwise, the predicted class is returned.

This approach allows the SVM to reject uncertain predictions by quantifying confidence in its classification decision, providing robustness against out-of-distribution inputs.

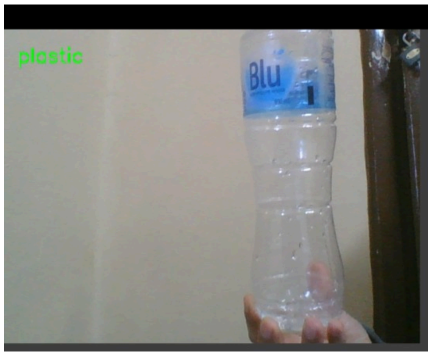
5. System Deployment

The real-time Material Stream Identification (MSI) system was deployed using a live camera feed to classify waste materials based on the extracted deep feature representations. The deployment pipeline integrates feature extraction, classification, and an "Unknown" class rejection mechanism to ensure reliable operation in practical scenarios.

The system workflow is as follows:

1. **Camera Input**
   A webcam captures live frames for real-time analysis.

2. **Feature Extraction**
   Each frame is processed to extract a fixed-length feature vector.All feature vectors are standardized using the previously saved `StandardScaler` to ensure consistency with the trained SVM model.

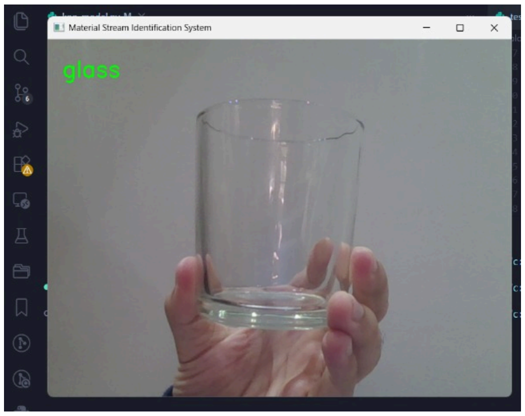


3. **Classification with Rejection**

   The standardized feature vector is input to the trained SVM classifier.If the maximum probability falls below the threshold, the sample is labeled as **“Unknown”**; otherwise, the corresponding material class is predicted.

4. **Label Mapping and Display**

   Predicted class indices are mapped to human-readable labels based on the dataset folder names.



5. **Real-Time Operation**

The system continuously captures frames, performs feature extraction and classification, and updates the display until the user exits (by pressing 'q').

This deployment approach allows robust, real-time classification while handling out-of-distribution materials via the rejection mechanism. All trained models and scalers are serialized and loaded at runtime to ensure consistent behavior between training and deployment phases.