Cairo University
Faculty of Computers and Artificial Intelligence

Material Stream Identification System

Name:	ID:
Joseph sameh	20220099
Youssef Joseph	20220389
Rana Ibrahim	20220130
Salma Mohamed	20220152
Jonathan Mokhles	20220100

1. Data Preparation & Augmentation

What Does It Do and The Main Goal?

It takes a collection of material images (cardboard, glass, metal, paper, plastic, and trash) and creates more variations of them through transformations. like having one photo and making multiple copies with slight differences - rotated, flipped, brightened, etc. This helps train better machine learning models by giving them more examples to learn from.

Ensures each material class has exactly 500 images by creating augmented (modified) copies of the original images. This balances the dataset and prevents the machine learning model from being biased toward classes with more images.

How It Works

1. Taking Inventory

The program starts by:

- Checking the data folder for original images
- Counting how many images exist for each material type
- Creating a new augmented_data folder to store the results

2. The Augmentation Toolkit

The program uses several image transformations:

- **Random Rotation**: Rotates images up to 10 degrees left or right
- **Horizontal Flip**: 50% chance to flip the image left-to-right
- **Vertical Flip**: 30% chance to flip the image top-to-bottom
- **Resize**: Standardizes all images to 224×224 pixels
- **Color Jitter**: Randomly adjusts brightness, contrast, saturation, and hue slightly

3. The Augmentation Process

The augment_class_folder() function for each material class:

1. **Copies originals**: First, it copies all original images to the augmented folder
2. **Validates images**: Checks that each image is valid before processing
3. **Creates variations**: Applies random transformations to create new images
4. **Keeps count**: Continues creating augmented images until reaching 500 total
5. **Names carefully**: Adds _aug1, _aug2, etc. to filenames to track generations
6. **Avoids duplicates**: Skips files that already exist

5. Smart Safety Features

The program includes several safety checks:

- **Image validation**: Uses img.verify() to ensure images aren't corrupted
- **Skip bad images**: If an image can't be opened, it skips it and continues
- **No overwrites**: Checks if a file exists before saving
- **Progress updates**: Prints status for each class so you can track progress

Note: we added extra data images so their count is more than that in the original dataset on classroom

Output Example

```
cardboard: 881 images
glass: 960 images
metal: 1221 images
paper: 1026 images
plastic: 1405 images
trash: 813 images

Total images before augmentation: 6306

Max class count: 1405
Target count for all classes: 1405

cardboard: 1762 images (round 2)
cardboard augmented to 1827 images.
glass augmented to 1827 images.
metal augmented to 1827 images.
paper augmented to 1827 images.
plastic augmented to 1827 images.
trash: 1626 images (round 2)
trash augmented to 1827 images.


✅ All augmented data saved in: D:\...\augmented_data
```

2. Feature Extraction

What Does It Do and The Main Goal?

This program looks at pictures of different materials (cardboard, glass, metal, paper, plastic, and trash) and converts them into a language that computers can understand (numbers)

The code takes images from the augmented_data folder and extracts features (characteristics) from each image using a pre-trained model. These features are then saved so other programs can use them to train classifiers that identify what material is in a picture.

How It Works

1. Setting Up

- Defining 6 material categories: cardboard, glass, metal, paper, plastic, and trash
- Creating a mapping system (like a dictionary) where each material gets a number ID
- Checking if your computer has a GPU (graphics card) to speed things up

2. The Feature Extractor

The CNNFeatureExtractor class uses a powerful pre-trained model called ResNet50:

- **ResNet50**: A neural network that has already learned to recognize patterns
- It takes the last layer off the ResNet50 model to get a 2048-dimensional feature vector
- Each image is resized to 224×224 pixels and normalized before processing
- The model extracts 2,048 numbers from each image that describe what it “sees”

3. Processing All Images

The process_dataset() function:

- Goes through each material folder in augmented_data
- Reads every image file (PNG, JPG, JPEG)
- Feeds each image to the feature extractor
- Collects all the feature vectors, labels, and filenames

4. Normalizing the Data

After extraction:

- Uses StandardScaler to normalize all features (makes them comparable)
- This ensures features have mean=0 and standard deviation=1
- Helps machine learning models work better

5. Saving Everything

The program saves three important files in the features folder:

- **features.csv**: Contains all 2,048 features for each image, and the label, class name, and filename
- **class_mapping.json**: The dictionary that maps material names to their ID numbers
- **scaler_params.csv**: Parameters needed to normalize new images the same way

6. Loading Features

The load_features() function:

- Reads back the saved features from disk
- Reconstructs the scaler so you can normalize new images consistently
- Useful when you want to use the features later without re-extracting them

Output Example

```
Loading features from disk...

Features successfully loaded:
  Features shape: (10962, 2048)
  Labels shape: (10962,)
  Feature mean: -0.000000
  Feature std: 1.000000
  Filenames loaded: 10962

Class mapping:
  cardboard (ID 0): 1827 samples
  glass (ID 1): 1827 samples
  metal (ID 2): 1827 samples
  paper (ID 3): 1827 samples
  plastic (ID 4): 1827 samples
  trash (ID 5): 1827 samples
```

3. Architecture Comparison (SVM vs k-NN)

Support Vector Machine (SVM) and k-Nearest Neighbors (k-NN). Both models were trained and tested using the same extracted feature vectors with dimensionality 2048 and the same stratified 80% training and 20% validation split to ensure a fair comparison.

Overall Performance

The SVM classifier achieved a validation accuracy of **95%**, while the k-NN classifier achieved a lower validation accuracy of **83%**. This shows that SVM was more effective at separating the material classes in the feature space. The margin-based nature of SVM makes it more suitable for high-dimensional features compared to the distance-based approach used by k-NN.

Class-Level Analysis

SVM showed consistently high precision and recall across all six material classes, indicating stable and reliable performance. On the other hand, k-NN performance varied between classes. While k-NN achieved high precision for some classes such as Glass and Cardboard, it had lower recall values for these classes, meaning that some samples were misclassified. The Metal class showed the weakest performance for k-NN, which may be due to visual similarity with other materials.

Unknown Class Handling

Both models included a mechanism to handle the Unknown class. The SVM used a confidence-based rejection method, while k-NN relied on a distance threshold derived from validation data. The SVM showed more stable rejection behavior, whereas k-NN was more sensitive to noise and feature-space density.

Final Model Selection

Based on the higher accuracy, better class-level stability, and faster inference performance, the **SVM classifier** was selected as the final model for deployment in the real-time system.

Aspect	**SVM**	**k-NN**
Feature Vector Size	2048	2048
Kernel / k Value	RBF kernel	k = 3
Validation Accuracy	**0.92**	**0.87**
Precision (Average)	0.92	0.87
Recall (Average)	0.92	0.87
F1-Score (Average)	0.92	0.87
Class-Level Stability	High	Medium
Unknown Class Handling	Confidence-based rejection	Distance-based threshold
Training Cost	High	Low
Inference Speed	Fast	Slower
Suitability for Real-Time	**High**	Medium