Cairo University

Faculty of Computers and Artificial Intelligence

Machine Learning Project

1. Importing the required modules:

- **Numpy:** for efficient mathematical calculations on datasets (2D array).
- **Pandas:** for data manipulation and data analysis.
- **OS:** for file/folder manipulation.
- **Cv2:** for image processing and computer vision.
- **Pathlib:** Cleaner and faster way to read/write files than os
- **Tqdm:** for tracking progress of loops or any tasks that take long time.
- **Sklearn:** for training and testing machine learning models on our dataset.
- **Torch:** used for training,running and testing neural networks.

2. Preparing and cleaning the dataset:

- **Image Collection:** Finds all jpg images in each category and put it in its corresponding category.
- **Image count per category:** Find how many images per category.
- **Dataset Validation:** Delete any corrupted images in each category and display how many images are valid and not valid.
- **Augmentation:** For a single image, it generates multiple variations: Horizontal and vertical flips,rotations, brightness changes ,zooming. Lastly, we perform the augmentation to the whole dataset to keep each category balanced so that each category has the same number of images.

3. Preparing the hardware:

- To ensure that the environment is reproducible and GPU-ready.
- To use DenseNet-201 CNN for feature extraction.
- Hyperparameters are defined.

4. Preprocessing Dataset:

- Create a new organized directory of train and test datasets for each category.
- Prepare data for ML-Classification.
- Split the dataset into 80% training and 20% validation.
- Convert images into torches for faster training and stable performance.
- Save images to disk.

5. Loading Preprocessed Torches: This ensures

- A **high-performance dataset loader** for deep learning.
- A **baseline dataset** for comparison.
- Faster training startup.
- Lower CPU overhead during training.
- Better scalability for large datasets.

6. Using pretrained CNN model:

- Model Setup

- Loads a pretrained DenseNet-201 model (ImageNet weights).
- Replaces the final classifier layer to match the number of dataset categories.
- Moves the model to GPU if available.

- Uses transfer learning to improve accuracy and reduce training time.

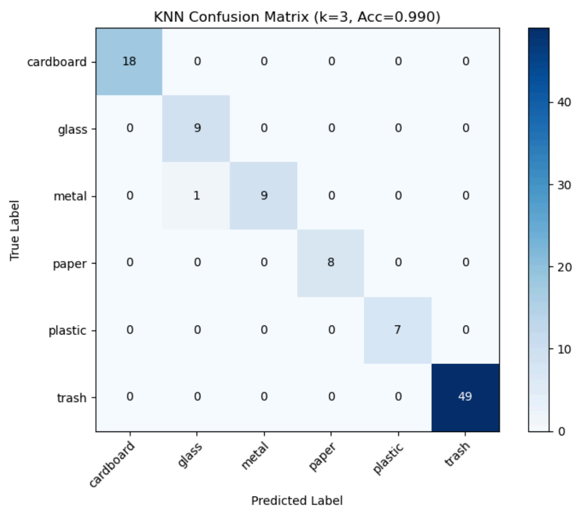
- **Purpose:**
    - The model is trained efficiently using transfer learning.
    - The best DenseNet-201 weights are saved to disk.
    - Training stops automatically when performance plateaus.
    - Final metrics are reported clearly.

7. Using CNN as a Feature Extractor:

- Reads images using OpenCV.
- Converts BGR to RGB, then to PIL.
- Applies validation preprocessing.
- Passes image through CNN (forward pass).
- Flattens and returns a 1D NumPy array of size 1920.
- This produces feature vectors that can be used for traditional ML classifiers (KNN, SVM).

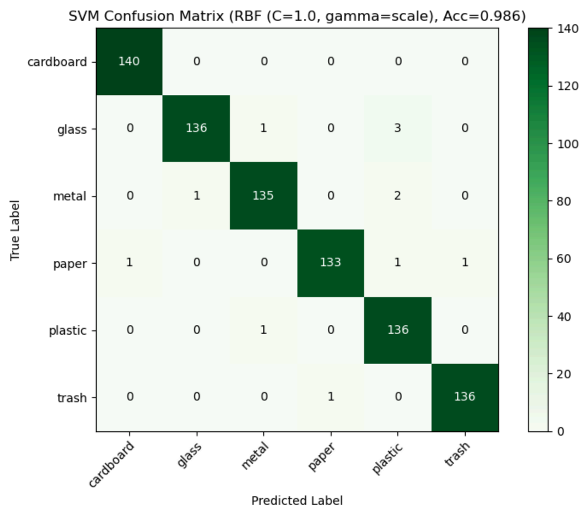
8. Models Evaluation and Comparison:

	Train Acc.	**Test Acc.**	**Best Config.**
KNN	**100%**	**99.01%**	**K = 3**
SVM	**100%**	**98.55%**	**rbf,C=1.0, gamma = scale**



KNN Confusion Matrix (k=3, Acc=0.990)

True Label \ Predicted Label	cardboard	glass	metal	paper	plastic	trash
cardboard	18	0	0	0	0	0
glass	0	9	0	0	0	0
metal	0	1	9	0	0	0
paper	0	0	0	8	0	0
plastic	0	0	0	0	7	0
trash	0	0	0	0	0	49



SVM Confusion Matrix (RBF (C=1.0, gamma=scale), Acc=0.986)

True Label \ Predicted Label	cardboard	glass	metal	paper	plastic	trash
cardboard	140	0	0	0	0	0
glass	0	136	1	0	3	0
metal	0	1	135	0	2	0
paper	1	0	0	133	1	1
plastic	0	0	1	0	136	0
trash	0	0	0	1	0	136

0
20
40
60
80
100
120
140

- **KNN performs better by 0.46% on the test set.**