TÉCNICO
LISBOA

Automated Deep Learning Waste Classification

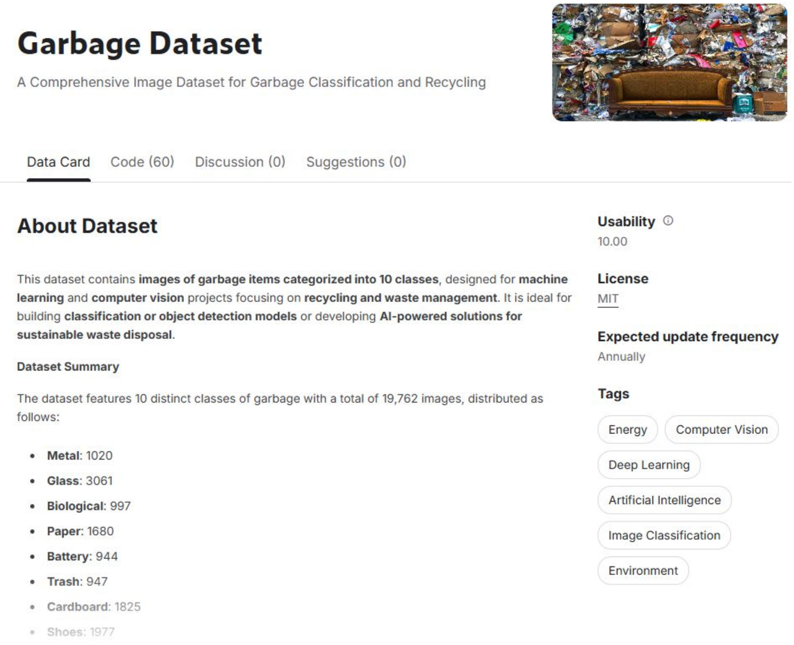
Intelligent Systems

Final Project

Isabella Silva 100195
Rita Barco 100259

Dataset Description

- ➔ 19762 images divided in 10 classes:
  `['battery','biological','cardboard','clothes','glass', 'metal','paper','plastic','shoes','trash']`
- ➔ 80/20 ratio for training/test sets
- ➔ test set was subdivided in studio-like images and real-world-like images



Methods implemented

→ MLP + Autoencoder

→ CNN

MLP data preprocessing

- ➔ Image resizing (128x128) and extraction of features (encoder)
- ➔ Standardizing of the extracted features
- ➔ Flattening of the Data

MLP implementation overview

- ➔ Trained encoder is extracted
- ➔ Standardize and flatten encoder extracted features serve as inputs to the MLP

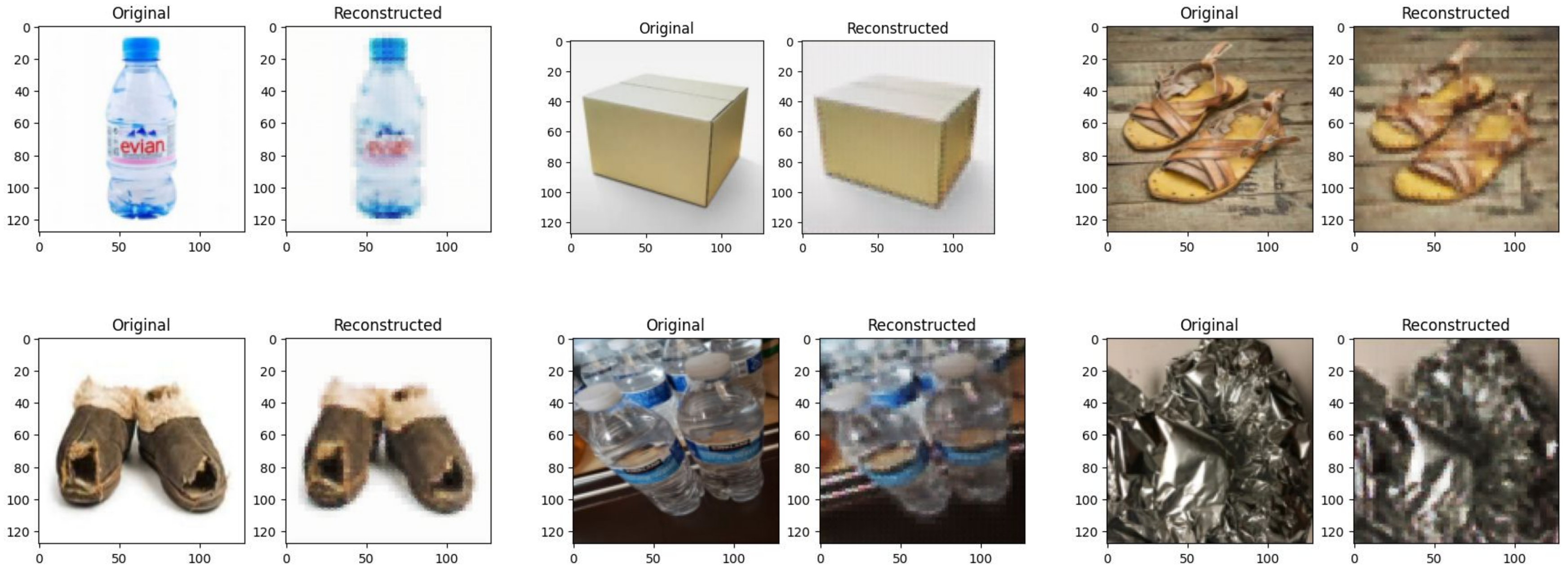
Autoencoder design and training

- **Encoder:** <u>2 convolutional layers</u> each one followed by <u>ReLU</u> activation and <u>max-pooling</u> to reduce spatial dimensions

- **Decoder:** uses <u>2 deconvolution layers</u> followed by nonlinear activations then (<u>ReLU</u> in 1st layer and <u>sigmoid</u> in 2nd layer)
- Training minimizes mean squared error (MSE)

Autoencoder reconstructions

➔ The model learns meaningful features in both studio-like and real-world-like images



Developed MLP

MLP_Simple

- ➔ single hidden layer (128 neurons) followed by:

  ReLU activation function + dropout layer of 0.1 (lessen overfitting in training)

- ➔ Fully connected Output layer (linear transformation)

MLP_Improved

- ➔ 3 layers:
  - ◆ linear layer (reduces the per-sample feature vector)
  - ◆ 2 Hidden layer, each followed by a ReLU activation and dropout with a rate of 0.2 (lessen overfitting in training)
- ➔ Fully connected Output layer (linear transformation)

MLP training procedure for both models

- ➔ During each iteration, the model computed predictions for each batch, compared them to the true labels using the loss function and updated its parameters through backpropagation
- ➔ Adam optimizer with a learning rate of 0.0005
- ➔ Cross-Entropy Loss function
- ➔ Ran through a 100 epochs

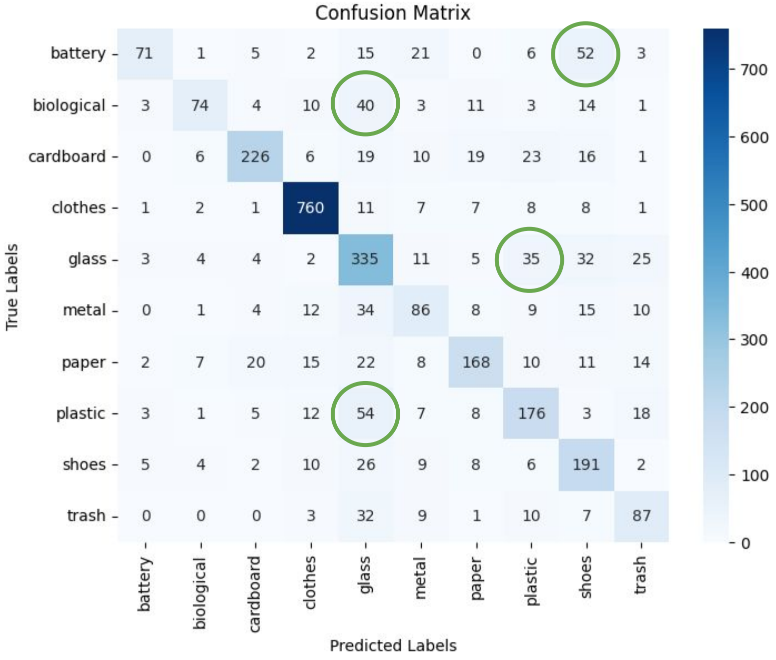
MLP testing and results

- ➔ Evaluation metrics: accuracy, precision, recall, F1-score

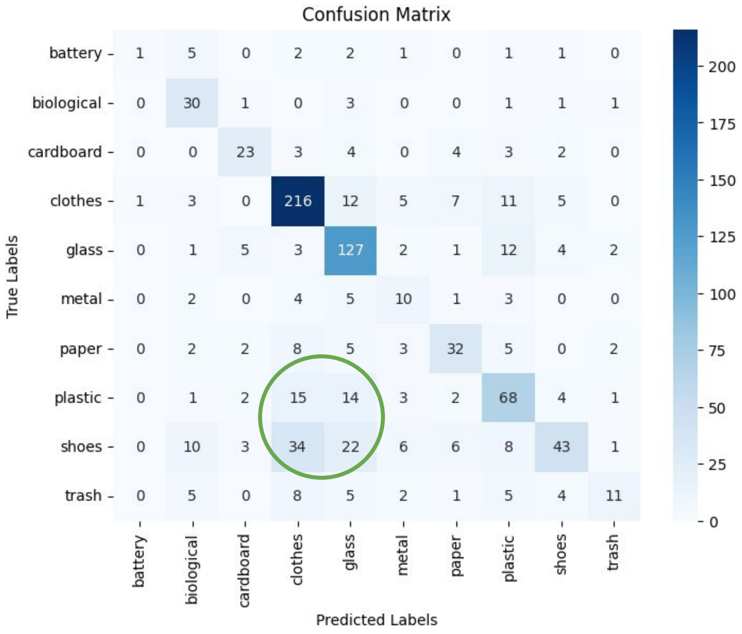
- ➔ **MLP_Improved** outperformed the simple variant, achieving 71% of accuracy on studio-like images and 64% on real-world images (these values indicate moderate generalization)

	MLP (S) T	MLP (I) T	MLP (S) TeS	MLP (I) TeS	MLP(S) TeR	MLP (I) TeR
Accuracy	0.9789	0.9477	0.6862	0.7054	0.6247	0.6419
Precision	0.9791	0.9503	0.6942	0.7200	0.6595	0.6444
Recall	0.9789	0.9477	0.6862	0.7054	0.6247	0.6419
F1 Score	0.9789	0.9473	0.6876	0.7032	0.6213	0.6240

MLP testing and results



Studio-like



Real-world-like

CNN implementation - data preprocessing

- ➔ This model performs direct image classification from raw RGB images
- ➔ Images are resized to 128x128 pixels
- ➔ ToTensor() is used to convert the image to a tensor and normalizes pixel values to [0,1]

CNN architecture details

IMAGE INPUT — 3x128x128

Conv1 + ReLU + MaxPool — 8x64x64

Conv2 + ReLU + MaxPool — 64x32x32

Conv3 + ReLU + MaxPool — 128x16x16

Flatten — 32768 features

Fully connected layer (fc1) + ReLU + dropout — 128 neurons

Fully connected layer (fc2) — 10 logits

Output (class logits)

Kernel used for the convolutional layers

3x3 kernel with padding=1

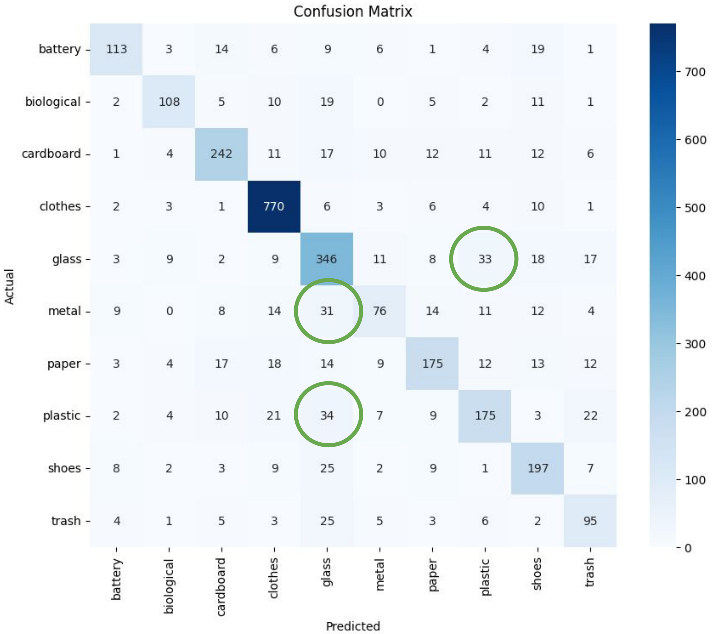
Training Process

- Optimizer: Adam
- Learning rate: 0.001
- Loss function: CrossEntropyLoss (used for multi-class classification)
- Epochs: 50

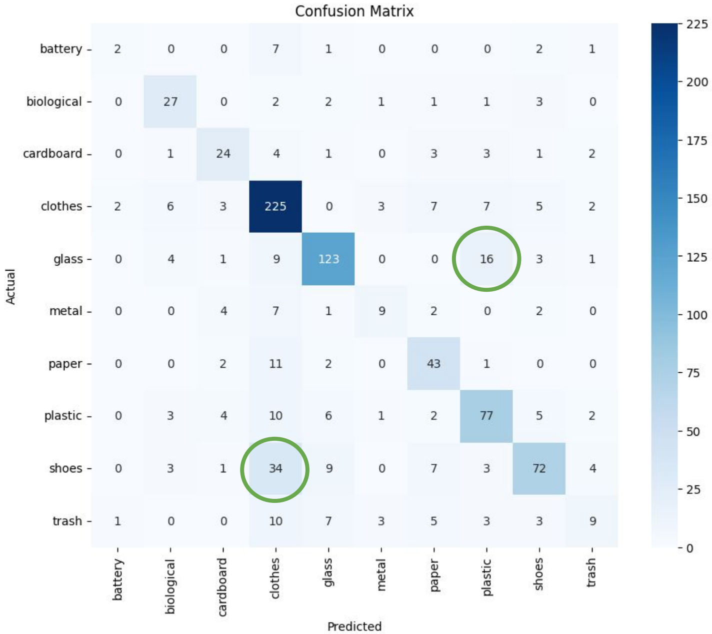
CNN testing results

- ➔ Near-perfect training
- ➔ Large difference between training and test performance suggest overfit
- ➔ Confusion matrices reveal frequent misclassifications (glass vs. plastic the most common)

	Train	TeS	TeR
Accuracy	0.9970	0.7453	0.6991
Precision	0.9970	0.7436	0.6927
Recall	0.9970	0.7453	0.6991
F1 Score	0.9970	0.7414	0.6875



Studio-like



Real-world-like

Model Comparison: MLP vs CNN

- ➔ The CNN outperforms the Encoder+MLP model on both studio-like and real-world test sets, achieving higher accuracy and F1-Score
- ➔ CNN shows stronger overfitting with near-perfect training scores
- ➔ Both models generalize worse on real-world images (on CNN the performance drop is larger) - as expected
- ➔ Visual similarity and class imbalance affect both models

THANK YOU!

NOTA: Ambas queremos ir para exame