Relatório Projeto – Meta I

Machine Learning - Reciclagem

Inteligência Computacional

Lucas Pagnano Golobovante - 2023140728
Diogo Valdemar Melo Estimado - 2023130197

Índice

Lucas Pagnano Golobovante – 2023140728
Diogo Valdemar Melo Estimado - 2023130197

Descrição do caso de estudo e objetivos do problema

O projeto insere-se na área da reciclagem, com o objetivo de classificar automaticamente imagens de resíduos em cinco categorias: Metal, Orgânico, Papel, Plástico e Vidro.

A maior aplicação do projeto veio de uma pesquisa que indica que: "7 em cada 10 portugueses já separam as suas embalagens, mas destes só 1 o faz de forma correta.", ou seja, da população portuguesa que separa as suas embalagens apenas 15% aproximadamente o faz de forma correta.

A tarefa é um problema de classificação multiclasse de imagens. O objetivo é desenvolver um modelo de *Deep Learning* capaz de identificar corretamente o tipo de material, auxiliando processos de triagem e reciclagem.

Com isso, temos o intuito de:

- Implementar uma Rede Neuronal Convolucional (CNN) para classificação de imagens.
- Avaliar o desempenho do modelo em treino, validação e teste.

Descrição do modelo e implementação dos algoritmos

Bibliotecas utilizadas

- TensorFlow / Keras → construção e treino da rede neuronal.
- NumPy → manipulação de arrays e cálculos numéricos.
- Matplotlib e Seaborn → visualização de gráficos e matrizes de confusão.
- scikit-learn → métricas de avaliação (classification report, confusion matrix, roc_auc_score, label_binarize).

```
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score
from sklearn.preprocessing import label_binarize
```

Configuração do dataset

Estrutura

- dataset_train → imagens para treino e validação (80/20 split).
- dataset_test → imagens para teste final.

Classes

- Metal: 750 (1500)
- Vidro: 750 (1500)
- Papel: 1000 (2000)
- Plástico: 1000 (2000)
- Orgânico: 1000 (2000)

Balanceamento: Dataset relativamente equilibrado, mas com pequenas variações no número de imagens por classe.

Representação: Imagens RGB redimensionadas para 128x128 pixels, cada imagem é representada como uma matriz tridimensional de valores normalizados entre 0 e 1 (ImageDataGenerator(rescale=1./255, …)).

```
train_dir = "dataset_train"

train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=10, width_shift_range=0.1,
                                   height_shift_range=0.1, zoom_range=0.1, validation_split=0.2)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    batch_size=16,
    class_mode="categorical",
    subset="training",
    shuffle=True
)

val_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    batch_size=16,
    class_mode="categorical",
    subset="validation",
    shuffle=False
)
```

Configuração do modelo

Optamos por utilizar Redes Neurais Convolucionais (CNNs), pois preservam a estrutura espacial das imagens e aprendem padrões locais como bordas, texturas e formas, sendo assim uma melhor opção.

Arquitetura/Camadas

- Entrada: (128x128x3)
- Conv2D(32 filtros, ReLU) + MaxPooling
- Conv2D(64 filtros, ReLU) + MaxPooling
- Conv2D(128 filtros, ReLU) + MaxPooling
- Flatten
- Dense(128, ReLU)
- Dropout(0.5)
- Dense(5, Softmax)

Função de perda: categorical_crossentropy

Otimizador: Adam (learning rate padrão 0.001)

Métrica principal: Accuracy

Épocas: 50

Batch size: 16

```
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(128,128,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(5, activation="softmax")
])

model.compile(optimizer="adam",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
```

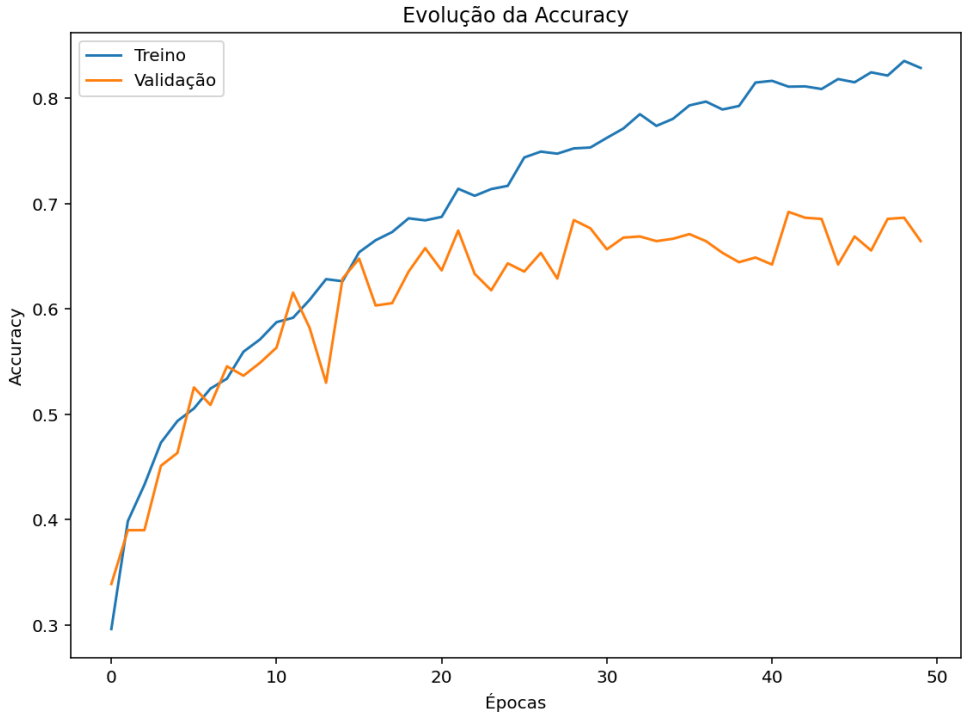
```
history = model.fit(
    train_gen,
    epochs=50,
    validation_data=val_gen
)
```

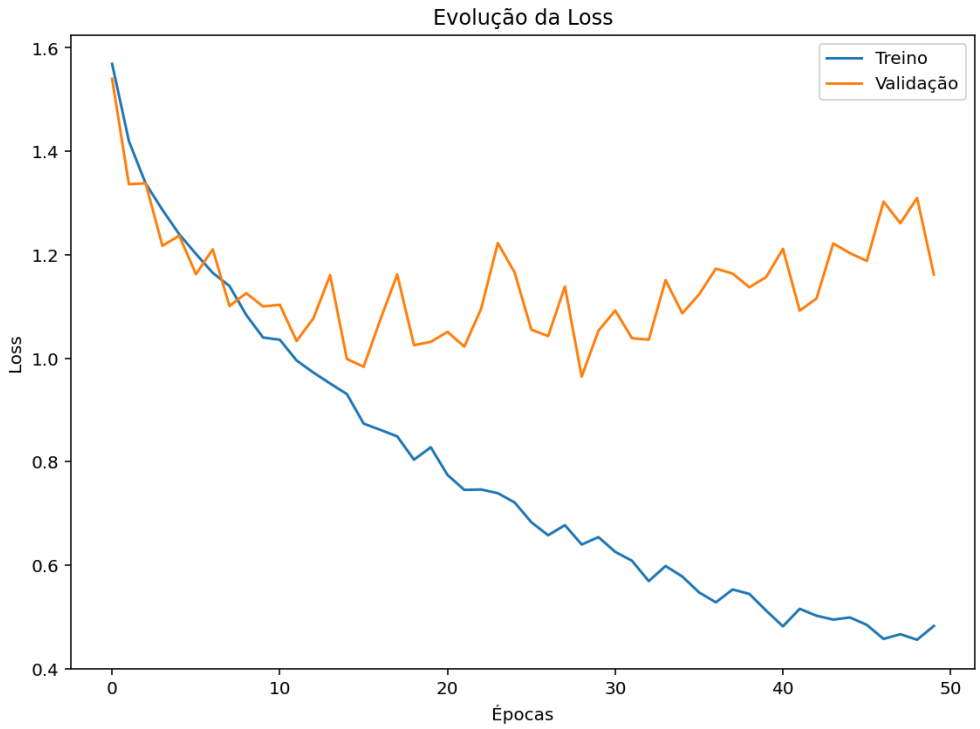
Análise de resultados

Análise do treino

- Accuracy no treino: ~82%
- Accuracy na validação: ~66%

```
Epoch 45/50
225/225 ━━━━━━━━━━━━━━━━━━━━ 31s 138ms/step - accuracy: 0.8183 - loss: 0.4987 - val_accuracy:
0.6422 - val_loss: 1.2030
Epoch 46/50
225/225 ━━━━━━━━━━━━━━━━━━━━ 30s 133ms/step - accuracy: 0.8153 - loss: 0.4844 - val_accuracy:
0.6689 - val_loss: 1.1879
Epoch 47/50
225/225 ━━━━━━━━━━━━━━━━━━━━ 30s 134ms/step - accuracy: 0.8247 - loss: 0.4572 - val_accuracy:
0.6556 - val_loss: 1.3027
Epoch 48/50
225/225 ━━━━━━━━━━━━━━━━━━━━ 29s 130ms/step - accuracy: 0.8217 - loss: 0.4663 - val_accuracy:
0.6856 - val_loss: 1.2609
Epoch 49/50
225/225 ━━━━━━━━━━━━━━━━━━━━ 31s 138ms/step - accuracy: 0.8356 - loss: 0.4556 - val_accuracy:
0.6867 - val_loss: 1.3099
Epoch 50/50
225/225 ━━━━━━━━━━━━━━━━━━━━ 29s 130ms/step - accuracy: 0.8289 - loss: 0.4823 - val_accuracy:
0.6644 - val_loss: 1.1617
```





O modelo aprendeu bem os dados de treino, mas não conseguiu transferir totalmente esse desempenho para os dados de validação, indicando um overfitting moderado.

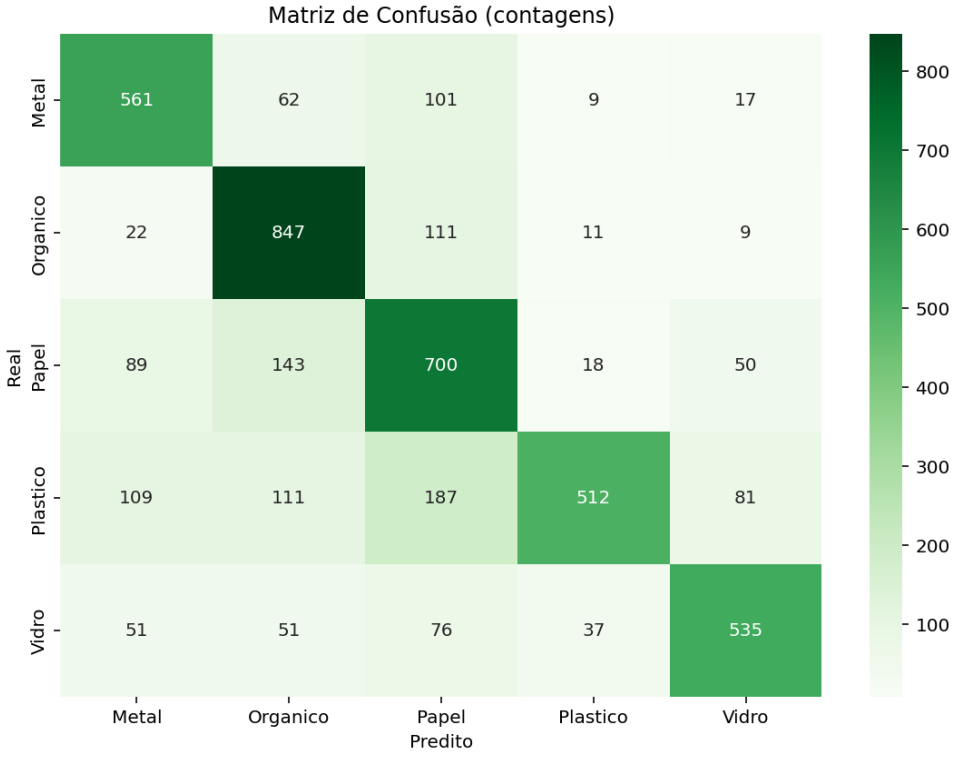
O desempenho em validação mostra que o modelo já tem alguma capacidade de generalização, mas ainda há espaço para melhorias.

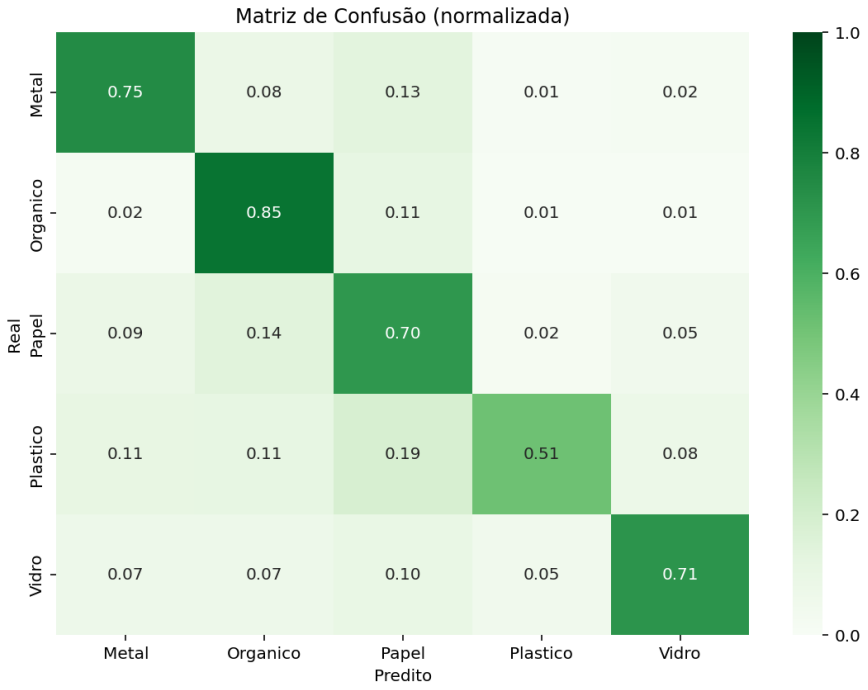
Análise do teste

- Accuracy: ~70%
- Recall: ~70%
- F1-score médio: ~70%
- Especificidade: ~92%
- AUC: ~91%

	precision	recall	f1-score	support
Metal	0.67	0.75	0.71	750
Organico	0.70	0.85	0.77	1000
Papel	0.60	0.70	0.64	1000
Plastico	0.87	0.51	0.65	1000
Vidro	0.77	0.71	0.74	750
accuracy			0.70	4500
macro avg	0.72	0.70	0.70	4500
weighted avg	0.72	0.70	0.70	4500

Especificidade média (macro): 0.9248
AUC média (macro): 0.9125





Plástico: alta precisão (0.87), mas recall baixo (0.51) → o modelo acerta quando prevê plástico, mas deixa escapar muitos plásticos reais.

Papel: desempenho mais fraco, com F1-score de 0.64

Orgânico e Vidro: desempenho equilibrado, com F1-scores acima de 0.70.

Metal: recall alto (0.75), mas precisão moderada → o modelo identifica bem metais, mas confunde outras classes como sendo metal.

Conclusões

Os resultados obtidos revelaram uma acurácia global de 70%, com destaque para a AUC média de 0.91 e especificidade média de 0.92, indicando excelente capacidade de separação entre classes e baixo índice de falsos positivos. A análise da matriz de confusão normalizada evidenciou que classes como orgânico e vidro foram bem reconhecidas, enquanto plástico e papel apresentaram maior taxa de confusão, possivelmente devido à semelhança visual entre os materiais.

Além disso, o relatório de classificação mostrou que o modelo possui desempenho equilibrado entre precisão e recall, com F1-scores superiores a 0.70 em três das cinco classes. A classe plástico, apesar de alta precisão (0.87), apresentou recall mais baixo (0.51), sugerindo a necessidade de ajustes para melhorar a sensibilidade.

A partir dessas observações, conclui-se que o modelo desenvolvido é tecnicamente sólido e funcional, servindo como base promissora para aplicações práticas em sistemas de triagem automatizada. A próxima etapa do projeto poderá inclui otimizar e melhorar o projeto de forma a obter um modelo mais robusto.

Melhorias previstas para a Meta II

- Data Augmentation: aumentar diversidade artificial do dataset.
- Batch Normalization: estabilizar treino.
- Transfer Learning: usar redes pré-treinadas (MobileNetV2, ResNet50).
- Class Weights: compensar desbalanceamento.
- Early Stopping e Checkpoints: evitar overfitting e guardar o melhor modelo.

Bibliografia

https://www.pontoverde.pt/comunicacao-e-educacao/noticias/estamos-a-chegar-ao-ponto-ponto-verde-estreia-nova-campanha-de-comunicacao/

Copilot