FIA - Prof Edjard Mota	2o Trabalho	Entrega: 16/Nov

Projeto 2: Classificação de Lixo para Reciclagem

Este projeto se conecta diretamente aos desafios do mundo real em reciclagem e gestão de resíduos, um tema relevante para o foco na COP 30 do material de referência. A triagem automatizada pode aumentar a eficiência e reduzir a contaminação nos fluxos de reciclagem. O objetivo é construir um classificador de imagens multiclasse para categorizar imagens de lixo em uma de seis classes: glass (vidro), paper (papel), cardboard (papelão), plastic (plástico), metal e trash (lixo orgânico/rejeito).

1. Problema e Objetivo: A triagem automática de resíduos é um passo importante para otimizar a reciclagem. O seu objetivo é construir um classificador de imagens multiclasse para categorizar imagens de lixo em seis tipos: vidro, papel, papelão, plástico, metal e lixo orgânico.

2. Análise do Dataset

- **Fonte:** O dataset TrashNet, disponível no GitHub e espelhado no Kaggle e Hugging Face.
- **Conteúdo:** Consiste em 2.527 imagens, divididas em 6 pastas,, redimensionadas (512x384 pixels) divididas em seis categorias. O dataset é relativamente pequeno e um pouco desbalanceado (a classe trash possui apenas 137 imagens), o que torna o aumento de dados (*data augmentation*) uma técnica necessária e valiosa a ser ensinada.
- **Nome:** TrashNet
- **Link:** https://github.com/garythung/trashnet ou no Kaggle https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification

3. Notebook Básico a ser Adaptado (com implementações da Terefa Principal)

- **Link:**(https://www.kaggle.com/code/farnazmirfeizi/trash-garbage-type-detection-using-cnn)
- **Fluxo de Atividades do Notebook:**
  1. **Carregamento de Dados:** Utiliza o ImageDataGenerator do Keras para carregar imagens diretamente das pastas do diretório, uma prática comum em visão computacional.
  2. **Aumento de Dados (*Data Augmentation*):** Configura o ImageDataGenerator para criar novas imagens de treino em tempo real através de transformações como rotação, zoom e inversão, combatendo o *overfitting*.
  3. **Construção do Modelo CNN:** Define uma arquitetura de CNN Sequential com camadas Conv2D, MaxPooling2D, Flatten e Dense.
  4. **Treinamento e Avaliação:** Treina o modelo com os geradores de dados e avalia sua performance.

4. Tarefa Principal

- **Entrada:** Um tensor de imagem (ex: 224x224x3).
- **Saída:** Uma camada densa com 6 neurônios e uma função de ativação softmax, produzindo uma distribuição de probabilidade sobre as classes.
- **Modelo:** Uma arquitetura CNN padrão (ex: 2-3 blocos de Conv2D -> ReLU ->

MaxPooling2D) seguida por um classificador (Flatten -> Dense -> Dropout -> Dense).
- **Avaliação:** Acurácia Categórica e uma Matriz de Confusão para visualizar quais classes são mais frequentemente confundidas entre si.

Lista de Entregas

- Um repositório no GitHub contendo:
  - Um arquivo README.md com a descrição do projeto e os resultados.
  - O Jupyter Notebook (.ipynb) com o código, a definição do modelo e a análise dos resultados.
  - Uma seção no notebook para a avaliação final, incluindo a acurácia e a matriz de confusão para entender quais classes o modelo confunde mais.
  - Uma breve conclusão sobre o impacto do *data augmentation* no desempenho do modelo.

‘