Projeto 2: Classificação de Lixo para Reciclagem

Descrição do Projeto

Este projeto, parte da disciplina de Aprendizado de Máquinas baseado em Redes Neurais, conecta-se diretamente aos desafios do mundo real em reciclagem e gestão de resíduos, um tema relevante para o foco na COP 30.

O objetivo principal é construir um classificador de imagens multiclasse para categorizar automaticamente imagens de lixo em uma de seis classes: glass (vidro), paper (papel), cardboard (papelão), plastic (plástico), metal e trash (lixo orgânico/rejeito).

O Dataset: TrashNet

O projeto utiliza o dataset TrashNet, disponível no Kaggle.

- Fonte: Kaggle - Garbage Classification
- Conteúdo: 2.527 imagens (redimensionadas para 512x384 pixels).
- Classes: 6 pastas, cada uma representando uma classe de lixo.
- O Desafio: O dataset é considerado pequeno e desbalanceado. A classe trash, por exemplo, possui apenas 137 imagens. Isso torna o overfitting (quando o modelo "decora" os dados de treino) o principal desafio a ser superado.

Ferramentas e Metodologia

Para desenvolver este classificador, foram utilizadas as seguintes ferramentas e técnicas:

- Linguagem: Python 3.10
- Ambiente: Conda (ambiente tf-final com suporte a GPU)
- Bibliotecas Principais: TensorFlow (Keras), Scikit-learn, Pandas, Matplotlib, Seaborn.
- Modelo: Uma Rede Neural Convolucional (CNN) padrão, seguindo o que foi solicitado:
    - 3 Blocos de Conv2D -> ReLU -> Dropout -> MaxPooling2D.
    - 1 Classificador Flatten -> Dense -> Dropout -> Dense.
- Técnica Chave (Combate ao Overfitting): Para lidar com o dataset pequeno, foi aplicado um Data Augmentation agressivo no gerador de dados de treino (ImageDataGenerator). Isto cria novas imagens "falsas" em tempo real para o modelo treinar, aumentando a variabilidade:

    - rescale=1./255
    - validation_split=0.15
    - horizontal_flip=True
    - vertical_flip=True
    - zoom_range = 0.5
    - width_shift_range = 0.3
    - height_shift_range = 0.3
    - rotation_range=50

- Estratégia de Treino: O modelo foi treinado usando o callback EarlyStopping, que monitora a val_loss (perda na validação) e encerra o treino automaticamente se o modelo parar de aprender. O treino foi interrompido após 52 épocas.

Resultados e Análise

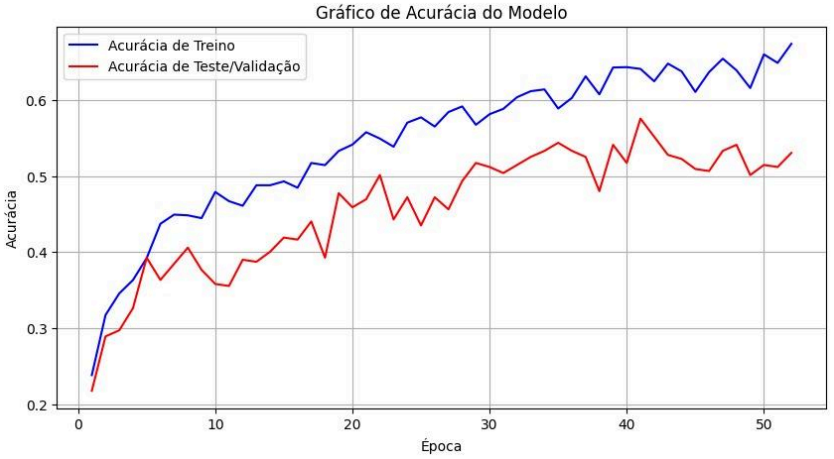
O modelo foi treinado e avaliado, gerando os seguintes resultados:

- Acurácia Final: O modelo atingiu uma acurácia de validação final de 57.6%.

Análise do Treinamento

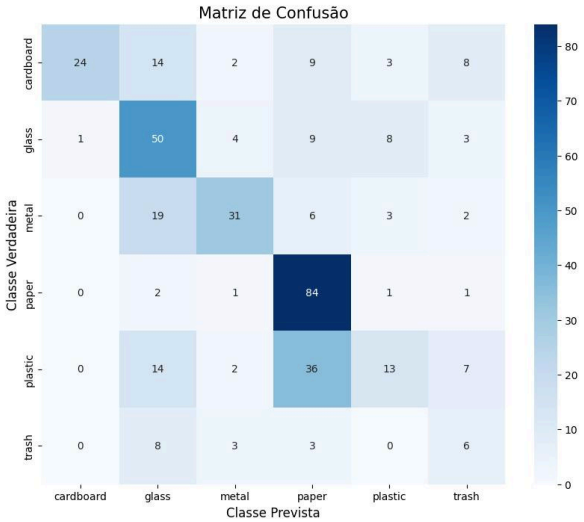
O gráfico de acurácia de treino vs. validação (teste) mostra um claro "gap" (abertura) entre as duas linhas. A linha azul (treino) continua subindo, enquanto a vermelha (validação) estagna por volta dos 50-57%.

Isso é a prova visual do overfitting: o modelo estava ficando muito bom em "decorar" as imagens de treino, mas não conseguia generalizar esse conhecimento para imagens novas.



Análise da Matriz de Confusão

A Acurácia Categórica (57.6%) nos diz que o modelo acertou, mas a Matriz de Confusão foi a chave para entender onde ele acertou e por que ele errou.



Analisando a matriz e o Relatório de Classificação (F1-score), podemos ver:

- Pontos Fortes: O modelo é excelente em identificar paper (F1-score de 71%) e metal (F1 de 60%). Quando o modelo prevê "paper", ele tem 94% de precisão.
- Pontos Fracos: O modelo é muito ruim em identificar plastic (F1-score de 26%) e trash (F1 de 26%).
- A Causa da Confusão: A matriz revela o porquê:
    - Plástico: De 72 imagens de plástico, o modelo classificou 36 delas como paper e 14 como glass.
    - Trash: De 20 imagens de lixo orgânico, o modelo acertou apenas 6, confundindo o resto com glass (8 vezes) e metal (3 vezes).

Conclusão

- Impacto do Data Augmentation: O data augmentation foi crucial. Sem ele, o overfitting teria sido muito mais severo e rápido. Ele foi essencial para permitir que o modelo aprendesse classes fáceis (como paper), mas não foi suficiente para resolver dois problemas centrais: (1) o desbalanceamento extremo da classe trash e (2) a alta similaridade visual entre plásticos transparentes, vidro e papel amassado.
- Resultado Final: A acurácia de ~57% não é um "fracasso" do modelo, mas sim a prova de que este dataset específico é um desafio significativo para uma arquitetura CNN simples, destacando a necessidade de mais dados ou técnicas mais avançadas (como Transfer Learning) para melhorias futuras.

Como Executar o Projeto

Clone este repositório:
git clone https://github.com/icompgerrar/Trabalho-2-FIA---Machine-Learning-com-CNN-.git
cd https://github.com/icompgerrar/Trabalho-2-FIA---Machine-Learning-com-CNN-.git

1. Instale as dependências restantes:
   pip install matplotlib pandas seaborn jupyterlab ipykernel scikit-learn
2. Execute o Jupyter Notebook jupyter lab
3. Abra o arquivo .ipynb e execute as células. (Lembre-se de ajustar o data_path no Bloco 2 para o local onde você baixou o dataset).

AUTORES

- Micael Gerrar micael.gerrar@icomp.ufam.edu.br
- Franscico Brilhante fransciscobraga99@gmail.com
- Tharcio Assunção tharcio.assuncao@icomp.ufam.edu.br
- Davi Emanuel Davi.emanuel@icomp.ufam.edu.br
- Hanna Mesquita hanna.mesquita@icomp.ufam.edu.br