Documentação do Projeto: Classificação de Imagens com CNN

Autor: Gabriel Castelo **Algoritmo:** Rede Neural Convolucional (CNN) **Dataset:** CIFAR-10 **Framework:** TensorFlow/Keras

1. Visão Geral e Definição do Problema

Este projeto constrói um modelo de Aprendizado Profundo (Deep Learning) supervisionado para classificar imagens pequenas em 10 categorias distintas.

O Dataset CIFAR-10

O CIFAR-10 é um conjunto de dados amplamente utilizado para treinar algoritmos de visão computacional.

- **Tamanho:** 60.000 imagens coloridas.
- **Resolução:** 32x32 pixels.
- **Classes:** Avião, Automóvel, Pássaro, Gato, Veado, Cão, Sapo, Cavalo, Navio, Caminhão.

2. A Solução: Rede Neural Convolucional (CNN)

Redes Neurais Convolucionais são a arquitetura padrão-ouro para análise de imagens. Ao contrário de redes densas tradicionais que achatam a imagem imediatamente, as CNNs preservam a estrutura espacial 2D dos pixels.

Componentes Principais Utilizados:

1. **Camadas de Convolução (`Conv2D`):** Funcionam como filtros que deslizam sobre a imagem para extrair características (bordas, texturas, formas complexas). No código, usamos filtros 3x3 e função de ativação **ReLU**.
2. **Camadas de Pooling (`MaxPooling2D`):** Reduzem a dimensionalidade da imagem (downsampling), mantendo apenas as informações mais importantes e tornando o modelo menos sensível à posição exata dos objetos.
3. **Camadas Densas (`Dense`):** No final da rede, a matriz 3D é "achatada" (`Flatten`) em um vetor 1D e passada para camadas comuns para realizar a classificação final.

3. Estrutura do Código

O script `CNN.py` executa o pipeline completo:

1. **Pré-processamento:**
    - Carrega os dados.
    - Normaliza os pixels (divide por 255.0) para que os valores fiquem entre 0 e 1, facilitando o treinamento da rede.
2. **Construção do Modelo:**
    - Empilha camadas `Conv2D` e `MaxPooling2D` sequencialmente.
    - Adiciona camadas `Dense` no topo para a saída de 10 classes (`softmax`).
3. **Treinamento:**
    - Compila com otimizador `Adam`.
    - Treina por 10 épocas.
4. **Visualização:**
    - Gera gráficos de Acurácia x Época.
    - Mostra previsões visuais em um grid de imagens de teste.

4. Exemplo de Uso e Resultados

Ao rodar o script, você verá dois outputs principais:

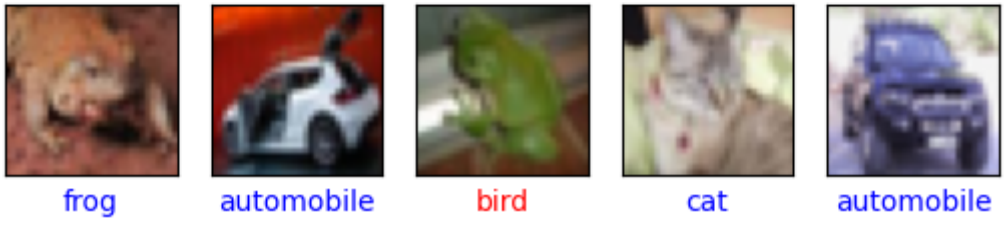
1. Gráfico de Evolução

Um gráfico que mostra a acurácia de treino e validação subindo ao longo das épocas. Isso ajuda a diagnosticar se o modelo está aprendendo bem ou sofrendo *overfitting*.

2. Previsões Visuais

O script exibe um grid com imagens do conjunto de teste e a classe predita pelo modelo.

- **Legenda Azul:** Previsão Correta.
- **Legenda Vermelha:** Previsão Incorreta.



5. Requisitos

- Python 3.x
- TensorFlow 2.x
- NumPy
- Matplotlib