Documentação train.py

Objetivo

Este script constrói, treina e avalia uma rede neural convolucional (CNN) usando o conjunto de dados MNIST para reconhecimento de dígitos manuscritos.

Carregamento dos Dados

(x_train, y_train), (x_test, y_test) = mnist.load_data()

- Carrega os dados de treino e teste do conjunto MNIST.

Normalização e Formatação

- As imagens são convertidas para o tipo float32 e normalizadas para o intervalo [0, 1].
- As dimensões são ajustadas para (28, 28, 1), exigido pelas camadas convolucionais.

Codificação One-Hot

- y_train e y_test são convertidos em vetores one-hot com 10 posições (uma para cada dígito de 0 a 9).

Construção do Modelo

- Sequential(): modelo sequencial do Keras. -
- Conv2D: camada convolucional com filtros 3x3 e ativação ReLU.
- MaxPooling2D: reduz dimensionalidade com pooling 2x2.
- Flatten: transforma saída 2D em vetor 1D.
- Dense: camadas densas (fully connected), com:
- 128 neurônios + ReLU
- 10 neurônios + Softmax (saída)

Compilação do Modelo

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

- Otimizador: Adam
- Função de perda: Entropia cruzada categórica
- Métrica: Acurácia

Avaliação

loss, acc = model.evaluate(x_test, y_test_cat)

- Avalia o desempenho no conjunto de teste.
- Imprime a acurácia final.

Salvamento do Modelo

model.save('mnist_number_recognition_keras.h5')

- Salva o modelo treinado para uso futuro.

Visualização de Previsões

- Mostra 5 imagens do conjunto de teste com seus respectivos rótulos reais e predições feitas pelo modelo.

Requisitos

- Python 3.x
- Bibliotecas: keras, numpy, matplotlib

Instalação

pip install keras numpy matplotlib