Handwritten Recognition

Mohammad Jaafar Ali Farhat

Step 1: Data Collection

I'll use the MNIST dataset from TensorFlow/Keras.

Step 2: Data Preprocessing

Normalize the data and prepare it for the model.

Step 3: Model Training and Evaluation

Train a Convolutional Neural Network (CNN) and evaluate its performance.

Step 4: Save and Download the Model and Results

Save the trained model and evaluation results, and provide download links.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.utils import to\_categorical

# Load dataset

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Normalize data

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# One-hot encode labels

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Reshape data for the model

X\_train = X\_train.reshape(-1, 28, 28, 1)

X\_test = X\_test.reshape(-1, 28, 28, 1)

# Build the model

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Flatten(),

Dense(128, activation='relu'),

Dense(10, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train, epochs=10, validation\_data=(X\_test, y\_test))

# Evaluate the model

evaluation\_results = model.evaluate(X\_test, y\_test, verbose=0)

# Save the model

model.save('handwritten\_character\_recognition\_model.h5')

# Save evaluation results to CSV

evaluation\_results\_df = pd.DataFrame([evaluation\_results], columns=['Loss', 'Accuracy'])

evaluation\_results\_df.to\_csv('model\_evaluation\_results.csv', index=False)

# Plot training history

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Loss')

plt.plot(history.history['val\_loss'], label='Val Loss')

plt.legend()

plt.title('Loss over epochs')

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'], label='Accuracy')

plt.plot(history.history['val\_accuracy'], label='Val Accuracy')

plt.legend()

plt.title('Accuracy over epochs')

plt.savefig('training\_history.png')

plt.show()

more details in the repo

