LEBANESE AMERICAN UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS

LAS 205 – Digital Cultures

Lab 2_4

Lara Tawbeh
ID#: 202102927

Date: 03-21-2024

Table of Contents:

A. Part I:

1. The MNIST dataset is a collection of 70,000 grayscale images of handwritten digits ranging from 0 to 9. Its origin can be traced back to Yann LeCun (Courant Institute, NYU) and Corinna Cortes (Google Labs, New York)

2. The size of the MNIST dataset is 70,000 images. It is managed using Python's tensor flow specifically the keras api.

3. The MNIST dataset includes grayscale images of handwritten digits (0 to 9) as well as their corresponding labels, already partitioned into images and labels for both training and validation.

4. The training set is made of 60,000 images, while the test set contains 10,000 images. The training set is 6x larger than the test set.

5. The dimension specifications of the MNIST dataset are as follows: each image is represented as a 28x28 pixel array which means there is a total of 784 pixels per image.

B. Part II:

1. The labels in the dataset represent digits from 0-9 which correspond to the handwritten images. Ex: a label of 6 means that the image contains the digit 6.

2. The digit contained in the handwritten image.

3. There are 10 classes in the dataset where each class represents a digit from 0 to 9. They are encoded using one-hot encoding, where each digit is represented as a binary matrix with a '1' meaning that the digit exists in that class and 0 its absence.

C. Part III:

1. In data processing, normalization means scaling the data to a standard range usually between 0 and 1. It is important to make sure that features with bigger magnitudes don't dominate the smaller ones during model training. Normalization contributes to a better training process by improving its stability and convergence.

2. Data normalization in natural language processing (NLP) could be essential when handling text data. For instance, the length of text in sentiment analysis changes significantly, as such, representing words as vectors or normalizing the size of input sequences could lead to boost the performance of models like recurrent neural networks (RNNs) or transformers.

D. Part IV:

To improve the accuracy of my model I changed the epochs number to 11 after some trial and error. Here's is the before and after accuracies:

BEFORE:

```
[ ] history = model.fit(
        x_train, y_train, epochs=5, verbose=1, validation_data=(x_valid, y_valid)
    )

    Epoch 1/5
    1875/1875 [==============================] - 22s 11ms/step - loss: 0.1876 - accuracy: 0.9423 - val_loss: 0.0998 - val_accuracy: 0.9701
    Epoch 2/5
    1875/1875 [==============================] - 21s 11ms/step - loss: 0.0846 - accuracy: 0.9757 - val_loss: 0.0936 - val_accuracy: 0.9751
    Epoch 3/5
    1875/1875 [==============================] - 20s 10ms/step - loss: 0.0597 - accuracy: 0.9837 - val_loss: 0.0990 - val_accuracy: 0.9758
    Epoch 4/5
    1875/1875 [==============================] - 21s 11ms/step - loss: 0.0454 - accuracy: 0.9876 - val_loss: 0.1079 - val_accuracy: 0.9764
    Epoch 5/5
    1875/1875 [==============================] - 20s 11ms/step - loss: 0.0351 - accuracy: 0.9901 - val_loss: 0.0963 - val_accuracy: 0.9790
```

The code above trains the model for 5 epochs on the training dataset while also evaluating it on the validation dataset after each epoch.

✓ Connected to Python 3 Google Compute Engine backend

AFTER:

```
history = model.fit(
    x_train, y_train, epochs=11, verbose=1, validation_data=(x_valid, y_valid)
)

Epoch 1/11
1875/1875 [==============================] - 14s 7ms/step - loss: 9.9837e-04 - accuracy: 0.9997 - val_loss: 0.1211 - val_accuracy: 0.9849
Epoch 2/11
1875/1875 [==============================] - 14s 7ms/step - loss: 4.3002e-04 - accuracy: 0.9999 - val_loss: 0.1279 - val_accuracy: 0.9856
Epoch 3/11
1875/1875 [==============================] - 14s 7ms/step - loss: 3.9056e-04 - accuracy: 0.9999 - val_loss: 0.1159 - val_accuracy: 0.9862
Epoch 4/11
1875/1875 [==============================] - 14s 8ms/step - loss: 4.1851e-04 - accuracy: 0.9998 - val_loss: 0.1240 - val_accuracy: 0.9855
Epoch 5/11
1875/1875 [==============================] - 14s 7ms/step - loss: 1.3106e-04 - accuracy: 0.9999 - val_loss: 0.1236 - val_accuracy: 0.9858
Epoch 6/11
1875/1875 [==============================] - 14s 7ms/step - loss: 2.0064e-06 - accuracy: 1.0000 - val_loss: 0.1223 - val_accuracy: 0.9863
Epoch 7/11
1875/1875 [==============================] - 14s 7ms/step - loss: 7.9653e-07 - accuracy: 1.0000 - val_loss: 0.1226 - val_accuracy: 0.9860
Epoch 8/11
1875/1875 [==============================] - 14s 7ms/step - loss: 6.4655e-07 - accuracy: 1.0000 - val_loss: 0.1226 - val_accuracy: 0.9860
Epoch 9/11
1875/1875 [==============================] - 14s 8ms/step - loss: 5.7649e-07 - accuracy: 1.0000 - val_loss: 0.1226 - val_accuracy: 0.9860
Epoch 10/11
1875/1875 [==============================] - 14s 7ms/step - loss: 5.2454e-07 - accuracy: 1.0000 - val_loss: 0.1227 - val_accuracy: 0.9860
Epoch 11/11
1875/1875 [==============================] - 13s 7ms/step - loss: 4.8580e-07 - accuracy: 1.0000 - val_loss: 0.1227 - val_accuracy: 0.9860
```

✓ 2m 32s completed at 11:24PM