Akash Varughese Joseph

<u>Task 2 – Number Recognition using a Neural Network</u>

Overview:

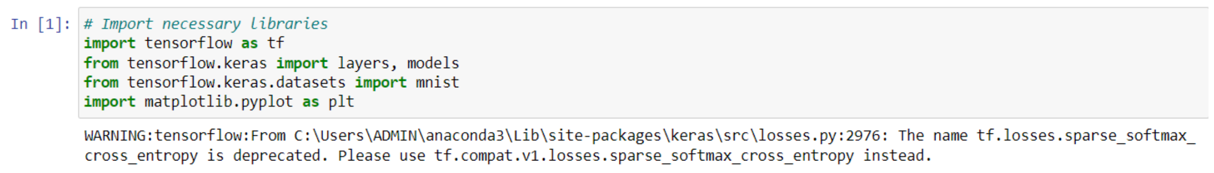
This project consists of the following components:

- Introduction
- Library Import and Dataset Loading
- Building the model
- Training the model
- Evaluate the test set with the model
- Visualize the training history
- Displaying Predicted and True labels
- Conclusion

Introduction:

In the realm of computer vision and machine learning, the task or recognizing handwritten digits holds paramount significance. Accurate digit recognition serves as a fundamental building block for diverse applications ranging from postal code recognition in mail sorting systems to automatic processing of bank cheques. In this project, we delve into the fascinating domain of digit recognition, employing a neural network to decipher handwritten numerals with precision. The dataset under consideration, known as the MNIST dataset, has emerged as a benchmark in the field of machine learning. Comprising a vast collection of 28x28 pixel grayscale images of handwritten digits (0 through 9), MNIST provides an ideal playground for training and evaluating machine learning models. The primary objective of our project is to develop an efficient and reliable system that can discern the nuances of handwritten digits and accurately predict the corresponding numerical values. Throughout this project, we address critical aspects such as data pre-processing, model building, training, and evaluation. The neural network is trained on a subset of the MNIST dataset, and its performance is rigorously assessed using a separate set of unseen data to ensure its generalization capabilities.

Library Import:

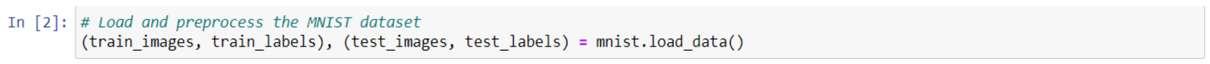


```
In [1]: # Import necessary libraries
        import tensorflow as tf
        from tensorflow.keras import layers, models
        from tensorflow.keras.datasets import mnist
        import matplotlib.pyplot as plt
```

```
WARNING:tensorflow:From C:\Users\ADMIN\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_
cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

The code imports essential libraries for a handwritten digit recognition project using TensorFlow and Keras. It leverages the MNIST dataset, a benchmark in machine learning, and includes visualization with matplotlib.

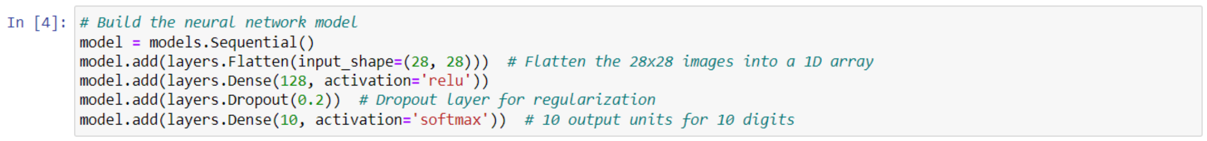
Loading Dataset:



```
In [2]: # Load and preprocess the MNIST dataset
        (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

This code snippet loads and pre-processes the MNIST dataset, splitting it into training and testing sets with corresponding labels. The dataset comprises of 28x28 pixel images of handwritten digits, laying the foundation for training a neural network for digit recognition.

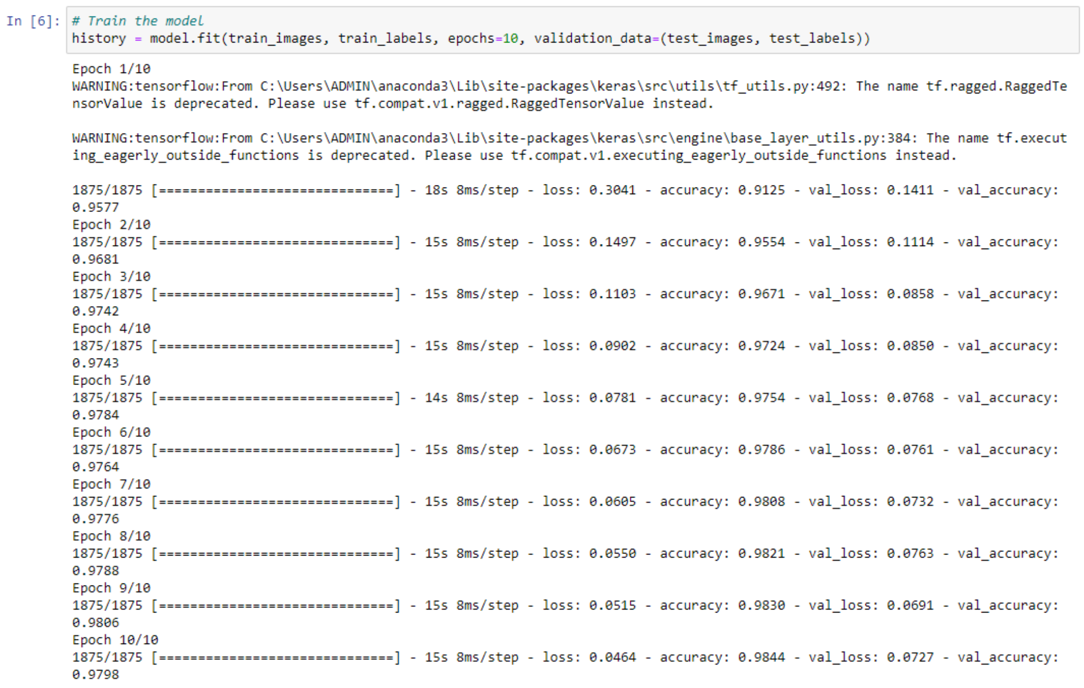
Building the model:



```
In [4]: # Build the neural network model
        model = models.Sequential()
        model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images into a 1D array
        model.add(layers.Dense(128, activation='relu'))
        model.add(layers.Dropout(0.2))  # Dropout layer for regularization
        model.add(layers.Dense(10, activation='softmax'))  # 10 output units for 10 digits
```

This code defines a feedforward neural network, specifically a multi-layer perception (MLP), for handwritten digit recognition. It consists of an input layer flattening 28x28 images, a hidden layer with 128 neurons and ReLU activation, a dropout layer for regularization, and an output layer with 10 units using softmax activation for classifying 10 digits.

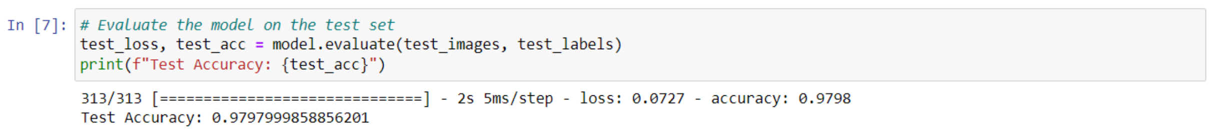
Training the model:



```
In [6]: # Train the model
        history = model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))

        Epoch 1/10
        WARNING:tensorflow:From C:\Users\ADMIN\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTe
        nsorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

        WARNING:tensorflow:From C:\Users\ADMIN\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.execut
        ing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

        1875/1875 [==============================] - 18s 8ms/step - loss: 0.3041 - accuracy: 0.9125 - val_loss: 0.1411 - val_accuracy:
        0.9577
        Epoch 2/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.1497 - accuracy: 0.9554 - val_loss: 0.1114 - val_accuracy:
        0.9681
        Epoch 3/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.1103 - accuracy: 0.9671 - val_loss: 0.0858 - val_accuracy:
        0.9742
        Epoch 4/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.0902 - accuracy: 0.9724 - val_loss: 0.0850 - val_accuracy:
        0.9743
        Epoch 5/10
        1875/1875 [==============================] - 14s 8ms/step - loss: 0.0781 - accuracy: 0.9754 - val_loss: 0.0768 - val_accuracy:
        0.9784
        Epoch 6/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.0673 - accuracy: 0.9786 - val_loss: 0.0761 - val_accuracy:
        0.9764
        Epoch 7/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.0605 - accuracy: 0.9808 - val_loss: 0.0732 - val_accuracy:
        0.9776
        Epoch 8/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.0550 - accuracy: 0.9821 - val_loss: 0.0763 - val_accuracy:
        0.9788
        Epoch 9/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.0515 - accuracy: 0.9830 - val_loss: 0.0691 - val_accuracy:
        0.9806
        Epoch 10/10
        1875/1875 [==============================] - 15s 8ms/step - loss: 0.0464 - accuracy: 0.9844 - val_loss: 0.0727 - val_accuracy:
        0.9798
```

This code trains the neural network model on the MNIST dataset for 10 epochs, utilizing the training images and labels. The validation data (test set) is used to assess model performance during training, and the training history is stored for later analysis.

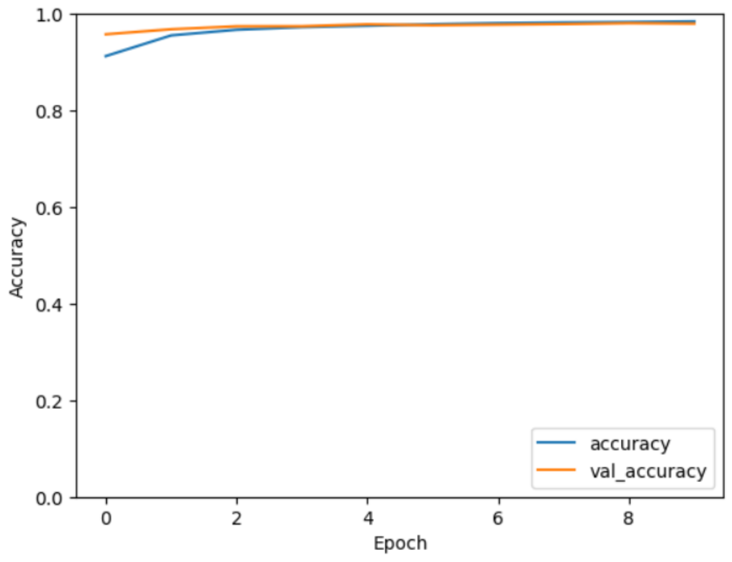
Evaluate the test set with the model:



```
In [7]: # Evaluate the model on the test set
        test_loss, test_acc = model.evaluate(test_images, test_labels)
        print(f"Test Accuracy: {test_acc}")

        313/313 [==============================] - 2s 5ms/step - loss: 0.0727 - accuracy: 0.9798
        Test Accuracy: 0.9797999858856201
```

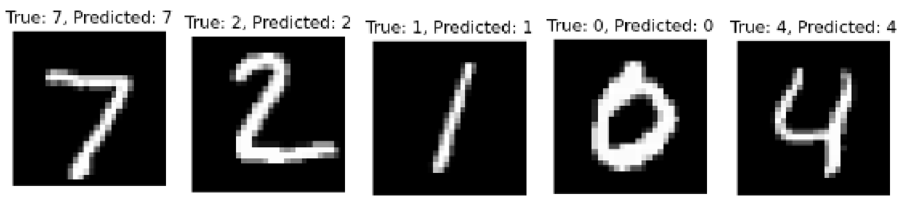
The code evaluates the trained neural network model on the test set, calculating the test loss and accuracy. The obtained test accuracy provides insights into the model's performance on unseen data.

Visualize the training history:



This code visualizes the training history of the neural network, plotting both training and validation accuracies over epochs. The similarity between the two lines indicates effective learning without significant overfitting or underfitting.

Displaying Predicted and True labels:



This code snippet visually presents a sample of five test images alongside their true labels and corresponding predictions made by the trained neural network. The model's ability to accurately predict digits is showcased, providing insights into its real-world applicability.

Conclusion:

In summary, this project successfully employed a well-structured neural network for handwritten digit recognition using the MNIST dataset. The model demonstrated robust learning, achieving commendable accuracy on both training and validation sets. Notably, the balanced training history indicates effective learning without overfitting. The real-world applicability was evident in accurate predictions on test images. This project highlights the significance of neural networks in image recognitions tasks and sets for future advancements in character recognition and document processing.