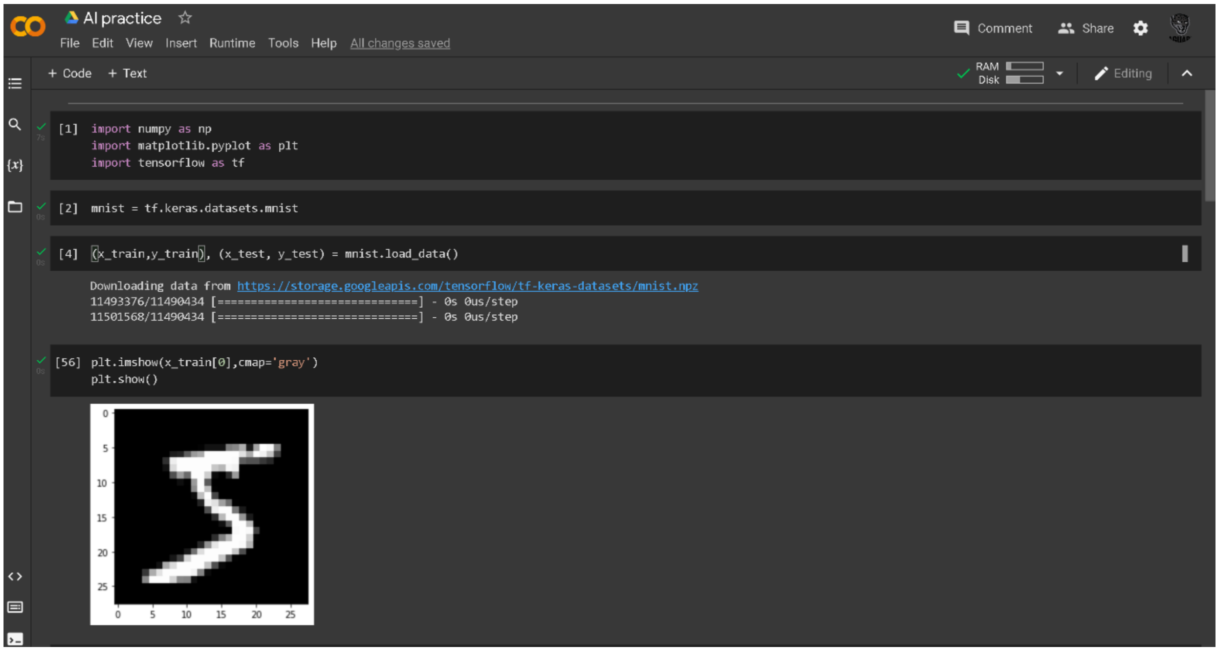
Shelby Chapa

AI Homework

For this exercise we will use TensorFlow which is an open source library used for building and training neural network deep learning models. We will create a neural network structure to run data through and the neural network that will learn from the data and create a model. We will then look at the output and analyze the output of the accuracy. The model must have a good level of accuracy or we must tune or change the structure of the deep learning model. This is significant because in neural networks understanding the topology is the root of building deep learning neural networks.

The problem we are trying to address is to accurately identify handwritten letters/numbers using Tensorflow. We will be using the principals of neural networks-neurons, Model Building, Model Training, Model Evaluation, and Prediction. The model identifies several data points in an image and the model will be able to identify the number/letter with a high amount of accuracy. The following are steps used to complete the problem.

The data set used below is the mnist dataset. We are training the AI to recognize hand written data. We displayed the image contents of the data which displays a hand written number 5 in 28X28 pixels.



```
[1]  import numpy as np
     import matplotlib.pyplot as plt
     import tensorflow as tf
```

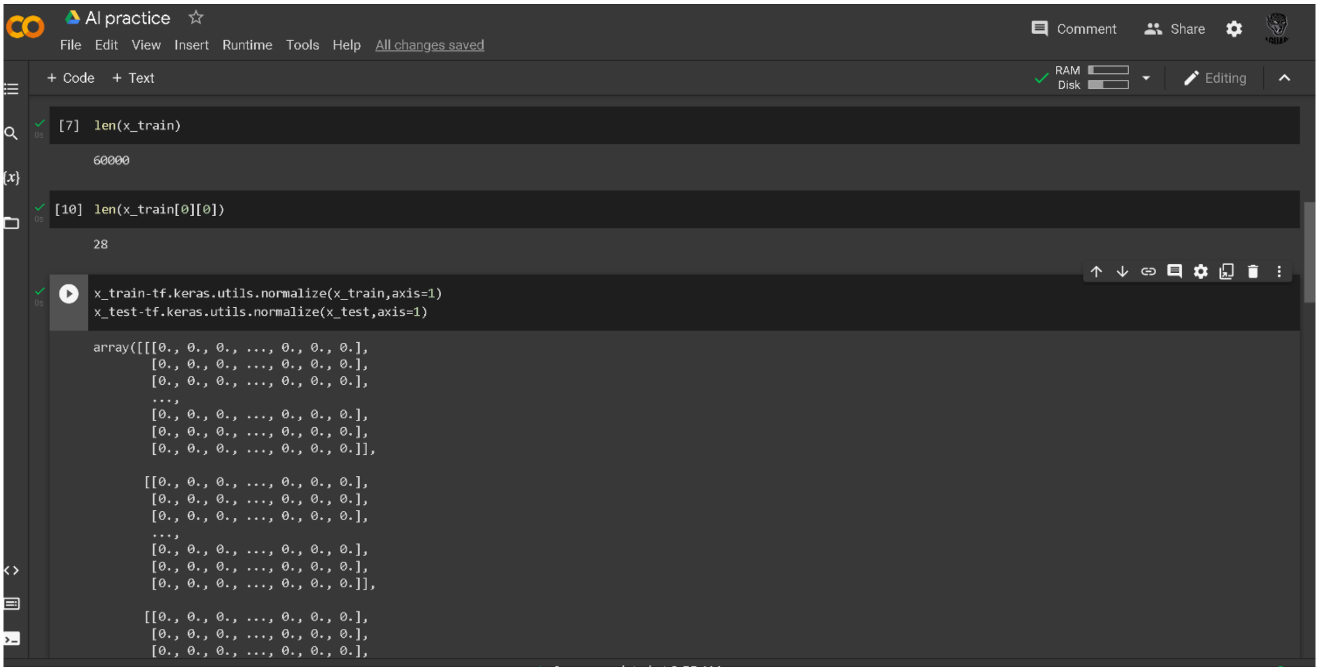
```
[2]  mnist = tf.keras.datasets.mnist
```

```
[4]  (x_train,y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

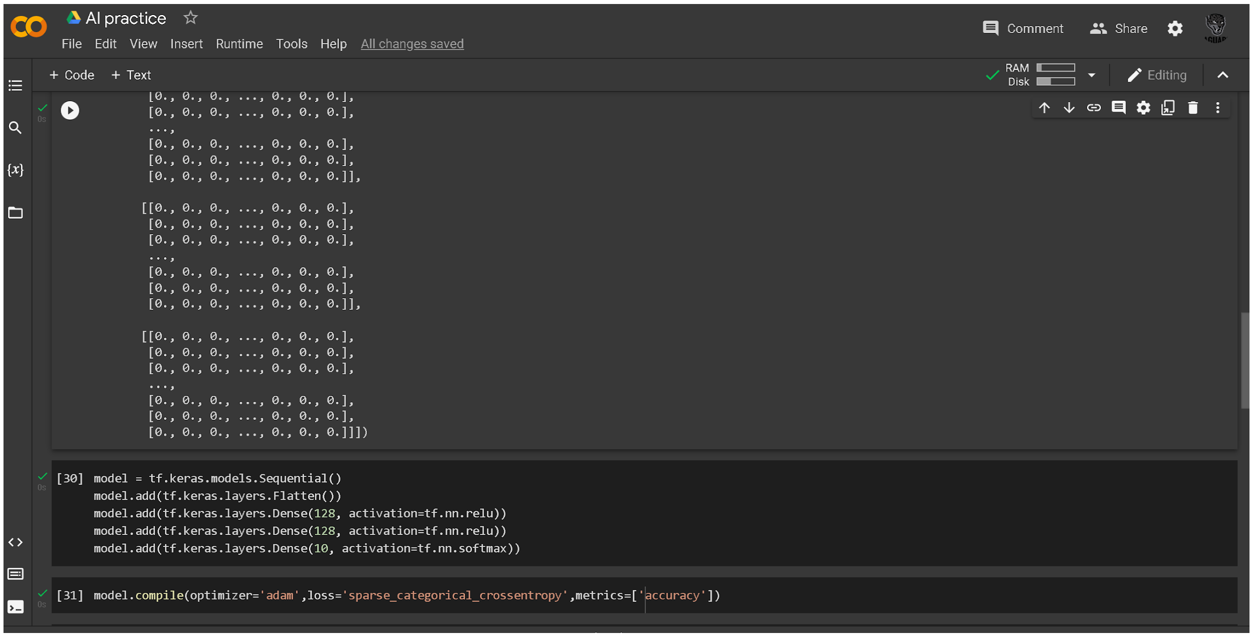
```
[56] plt.imshow(x_train[0],cmap='gray')
     plt.show()
```

The training data has 60,000 values, and the length of the first row is also 28. We were able to confirm that the table was 28X28. For each image we have there are 784 data points, because each pixel represents something in the data. Then the data is normalized in which outliers are removed.



```
[7]  len(x_train)

     60000

[10] len(x_train[0][0])

     28

     x_train-tf.keras.utils.normalize(x_train,axis=1)
     x_test-tf.keras.utils.normalize(x_test,axis=1)

     array([[[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]],

            [[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]],

            [[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
```

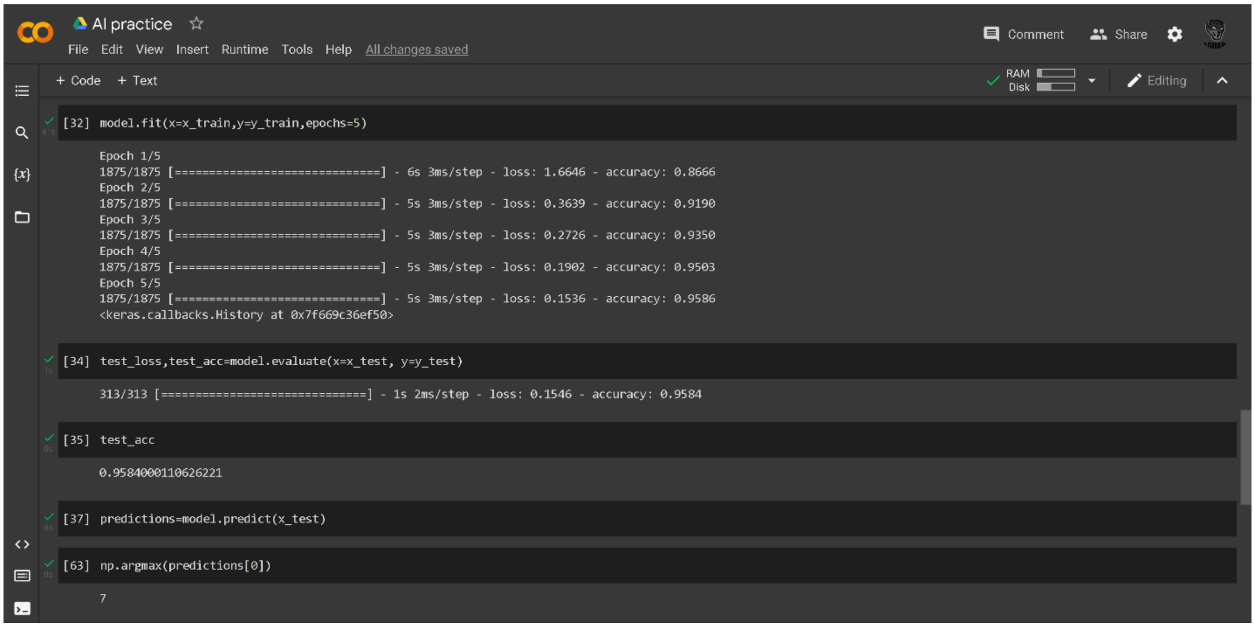
Then it's important to start to identify the topology of the model by creating a model variable. Multuple layers are added from flat to dense, and a sequential model is created, the results tell you how long the neurons are. The output layer will range to 10 values.



```
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]],

            [[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]],

            [[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]]])

[30] model = tf.keras.models.Sequential()
     model.add(tf.keras.layers.Flatten())
     model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
     model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
     model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

[31] model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

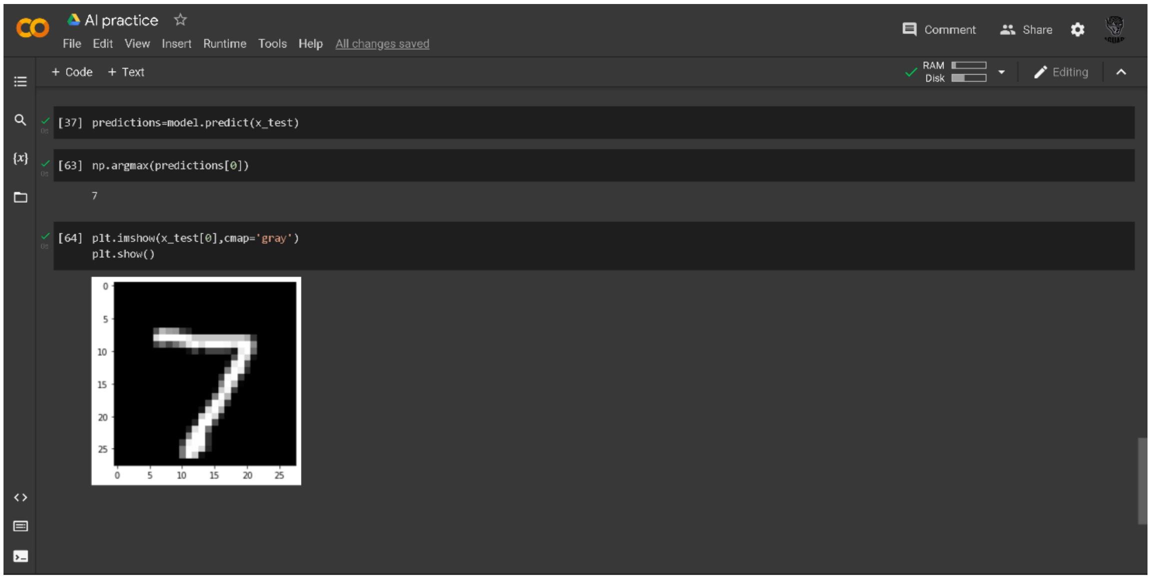
Also in the figure above, the data is then compiled to codify everyting is working and no other changes need to be made. We use an optimizer that will help figure out if there are any errors, loss is added to calculate the difference between the prediction and the output. Finally, the accuracy of the matrix is used to calculate the functioning of the model.

The fit function below will train the data and test the data. We use epochs to identify how many times to train the model which also increases accuracy. The accuracy was identified to be 95.8% in which the model will be able to identify the handwritten number. The prediction number was changed from the exercise. Instead of obtaining 9, I set the prediction number as [0] which resulted as 7.



```
[32] model.fit(x=x_train,y=y_train,epochs=5)

     Epoch 1/5
     1875/1875 [==============================] - 6s 3ms/step - loss: 1.6646 - accuracy: 0.8666
     Epoch 2/5
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.3639 - accuracy: 0.9190
     Epoch 3/5
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.2726 - accuracy: 0.9350
     Epoch 4/5
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.1902 - accuracy: 0.9503
     Epoch 5/5
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.1536 - accuracy: 0.9586
     <keras.callbacks.History at 0x7f669c36ef50>

[34] test_loss,test_acc=model.evaluate(x=x_test, y=y_test)

     313/313 [==============================] - 1s 2ms/step - loss: 0.1546 - accuracy: 0.9584

[35] test_acc

     0.9584000110626221

[37] predictions=model.predict(x_test)

[63] np.argmax(predictions[0])

     7
```

The model was accurately able to identify the handwriting as the number 7 from the prediction with the accuracy of 95%



```
[37] predictions=model.predict(x_test)

[63] np.argmax(predictions[0])

     7

[64] plt.imshow(x_test[0],cmap='gray')
     plt.show()
```

In conclusion, the model was able to accurately identify handwritten letters/numbers using Tensorflow. After several steps, such as normalzing the data set and testing several times to train the model which also increased accuracy. It was able to use several data points and model to identify the handwriting as the number 7 from the prediction with the accuracy of 95% in which we achieved a solution to our initial problem.