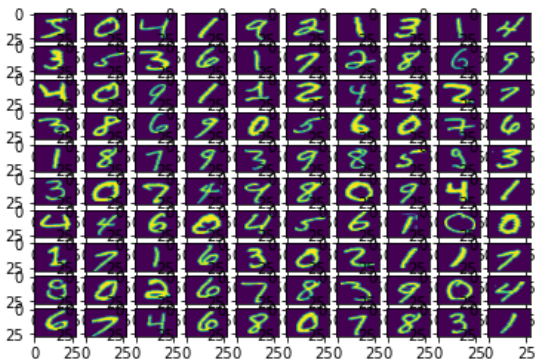
```
# Step 1: Import the necessary libraries.
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Activation
import matplotlib.pyplot as plt


# Step 2: Download the dataset.
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()



# Cast the records into float values
# Step 3: Now we will convert the pixels into floating-point values.
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

# normalize image pixel values by dividing
# by 255
gray_scale = 255
```



```
# Step 4: Understand the structure of the dataset
print("Feature matrix:", x_train.shape)
print("Target matrix:", x_test.shape)
print("Feature matrix:", y_train.shape)
print("Target matrix:", y_test.shape)
```

```
Feature matrix: (60000, 28, 28)
Target matrix: (10000, 28, 28)
Feature matrix: (60000,)
Target matrix: (10000,)
```

```
# Step 5: Visualize the data.
fig, ax = plt.subplots(10, 10)
k = 0
for i in range(10):
    for j in range(10):
        ax[i][j].imshow(x_train[k].reshape(28, 28),
                        aspect='auto')
        k += 1
plt.show()
```



```
# Step 6: Form the Input, hidden, and output layers.
model = Sequential([

    # reshape 28 row * 28 column data to 28*28 rows
    Flatten(input_shape=(28, 28)),

    # dense layer 1
    Dense(256, activation='sigmoid'),
```

```
    # dense layer 2
    Dense(128, activation='sigmoid'),

    # output layer
    Dense(10, activation='sigmoid'),
])



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



# Step 7: Compile the model.
model.fit(x_train, y_train, epochs=10,
        batch_size=2000,
        validation_split=0.2)
```

```
Epoch 1/10
24/24 [==============================] - 2s 59ms/step - loss: 2.0911 - accuracy: 0.4239 - val_loss: 1.7432 - val_accuracy: 0.6598
Epoch 2/10
24/24 [==============================] - 1s 54ms/step - loss: 1.4018 - accuracy: 0.7297 - val_loss: 1.0434 - val_accuracy: 0.8112
Epoch 3/10
24/24 [==============================] - 1s 53ms/step - loss: 0.8700 - accuracy: 0.8226 - val_loss: 0.6726 - val_accuracy: 0.8625
Epoch 4/10
24/24 [==============================] - 1s 54ms/step - loss: 0.6043 - accuracy: 0.8687 - val_loss: 0.4946 - val_accuracy: 0.8904
Epoch 5/10
24/24 [==============================] - 1s 54ms/step - loss: 0.4712 - accuracy: 0.8884 - val_loss: 0.4046 - val_accuracy: 0.8994
Epoch 6/10
24/24 [==============================] - 1s 54ms/step - loss: 0.3989 - accuracy: 0.8986 - val_loss: 0.3533 - val_accuracy: 0.9087
Epoch 7/10
24/24 [==============================] - 1s 53ms/step - loss: 0.3542 - accuracy: 0.9067 - val_loss: 0.3195 - val_accuracy: 0.9137
Epoch 8/10
24/24 [==============================] - 1s 53ms/step - loss: 0.3230 - accuracy: 0.9122 - val_loss: 0.2955 - val_accuracy: 0.9193
Epoch 9/10
24/24 [==============================] - 1s 53ms/step - loss: 0.3000 - accuracy: 0.9164 - val_loss: 0.2773 - val_accuracy: 0.9234
Epoch 10/10
24/24 [==============================] - 1s 53ms/step - loss: 0.2812 - accuracy: 0.9210 - val_loss: 0.2619 - val_accuracy: 0.9281
<keras.callbacks.History at 0x7f0d32ee6d10>
```



```
# Step 9: Find Accuracy of the model.
results = model.evaluate(x_test, y_test, verbose = 0)
print('test loss, test acc:', results)
```

```
test loss, test acc: [0.26626068353652954, 0.9232000112533569]
```