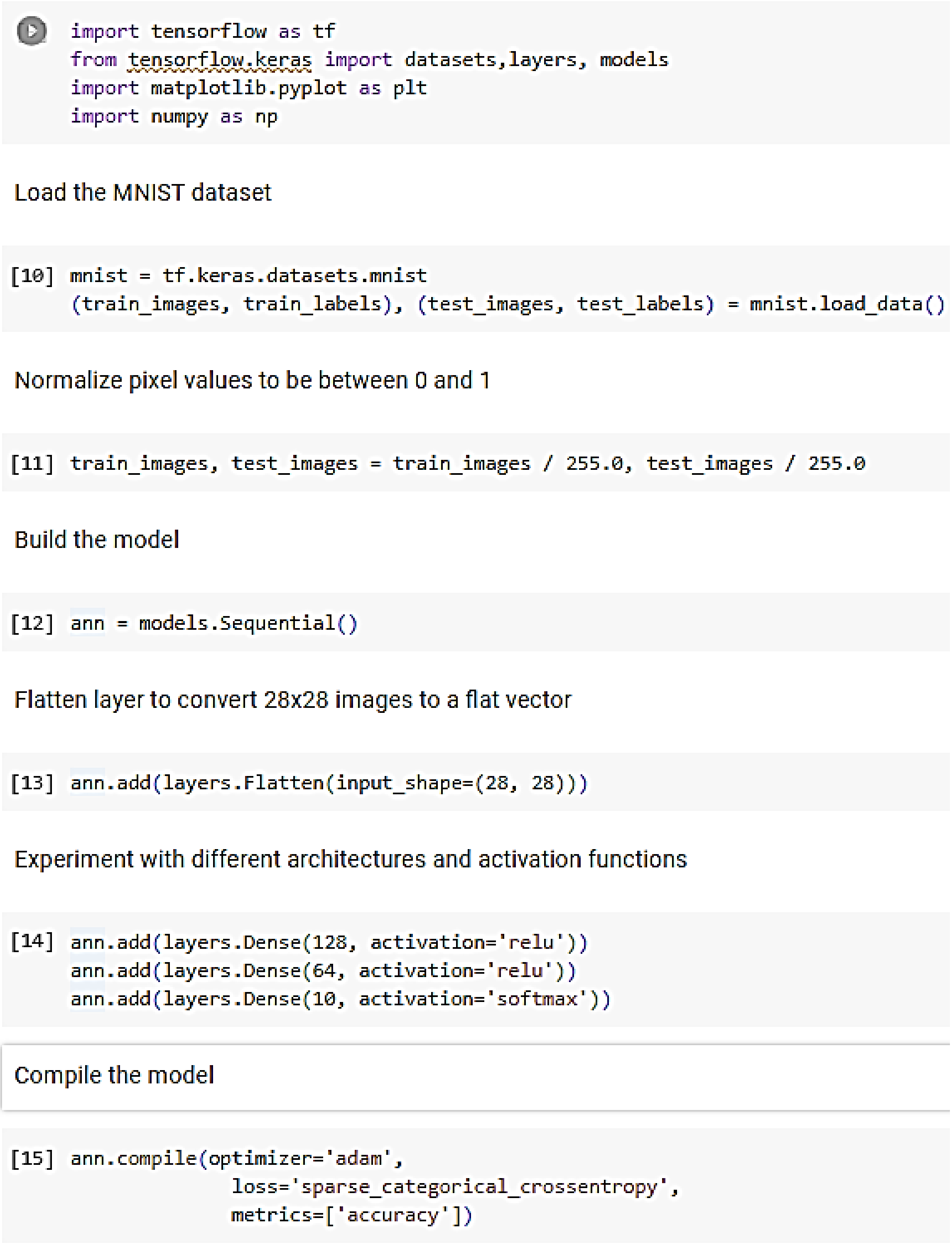


```
import tensorflow as tf
from tensorflow.keras import datasets,layers, models
import matplotlib.pyplot as plt
import numpy as np
```

Load the MNIST dataset

```
[10] mnist = tf.keras.datasets.mnist
     (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Normalize pixel values to be between 0 and 1

```
[11] train_images, test_images = train_images / 255.0, test_images / 255.0
```

Build the model

```
[12] ann = models.Sequential()
```

Flatten layer to convert 28x28 images to a flat vector

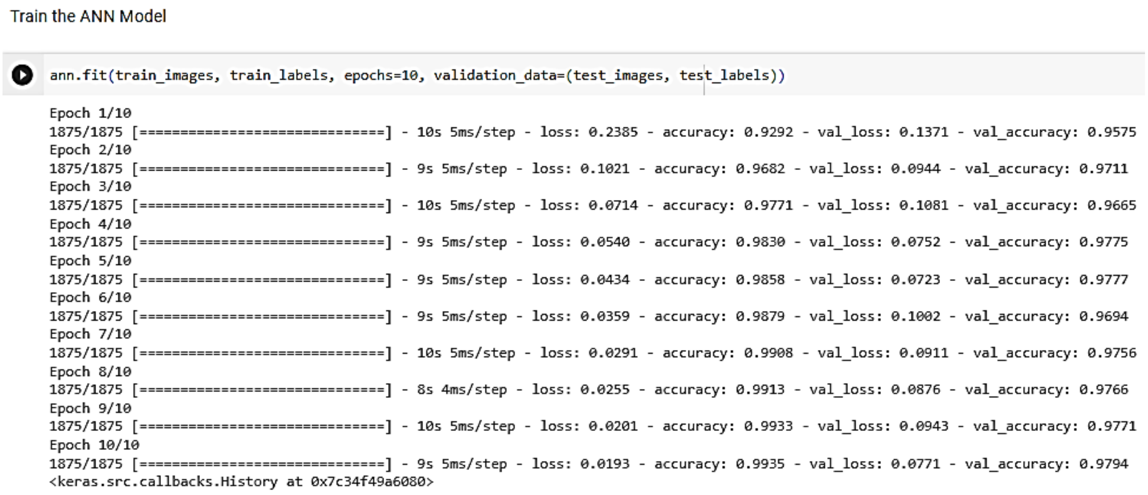
```
[13] ann.add(layers.Flatten(input_shape=(28, 28)))
```

Experiment with different architectures and activation functions

```
[14] ann.add(layers.Dense(128, activation='relu'))
     ann.add(layers.Dense(64, activation='relu'))
     ann.add(layers.Dense(10, activation='softmax'))
```

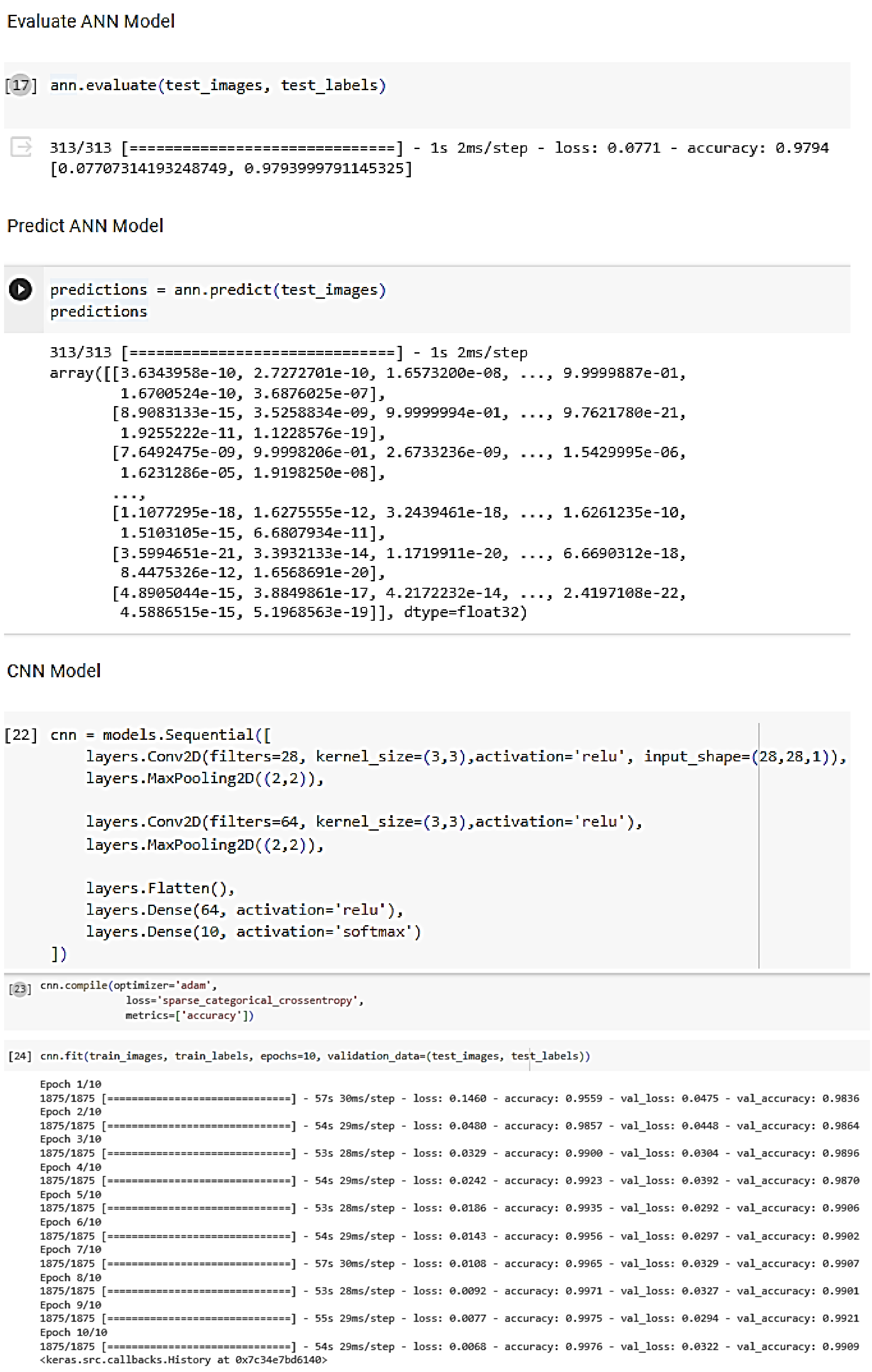
Compile the model

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



Train the ANN Model

```
ann.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))

Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2385 - accuracy: 0.9292 - val_loss: 0.1371 - val_accuracy: 0.9575
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1021 - accuracy: 0.9682 - val_loss: 0.0944 - val_accuracy: 0.9711
Epoch 3/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0714 - accuracy: 0.9771 - val_loss: 0.1081 - val_accuracy: 0.9665
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0540 - accuracy: 0.9830 - val_loss: 0.0752 - val_accuracy: 0.9775
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0434 - accuracy: 0.9858 - val_loss: 0.0723 - val_accuracy: 0.9777
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0359 - accuracy: 0.9879 - val_loss: 0.1002 - val_accuracy: 0.9694
Epoch 7/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0291 - accuracy: 0.9908 - val_loss: 0.0911 - val_accuracy: 0.9756
Epoch 8/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0255 - accuracy: 0.9913 - val_loss: 0.0876 - val_accuracy: 0.9766
Epoch 9/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0201 - accuracy: 0.9933 - val_loss: 0.0943 - val_accuracy: 0.9771
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0193 - accuracy: 0.9935 - val_loss: 0.0771 - val_accuracy: 0.9794
<keras.src.callbacks.History at 0x7c34f49a6080>
```



Evaluate ANN Model

```
[17] ann.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0771 - accuracy: 0.9794
[0.07707314193248749, 0.9793999791145325]
```

Predict ANN Model

```
predictions = ann.predict(test_images)
predictions
```

```
313/313 [==============================] - 1s 2ms/step
array([[3.6343958e-10, 2.7272701e-10, 1.6573200e-08, ..., 9.9999887e-01,
        1.6700524e-10, 3.6876025e-07],
       [8.9083133e-15, 3.5258834e-09, 9.9999994e-01, ..., 9.7621780e-21,
        1.9255222e-11, 1.1228576e-19],
       [7.6492475e-09, 9.9998206e-01, 2.6733236e-09, ..., 1.5429995e-06,
        1.6231286e-05, 1.9198250e-08],
       ...,
       [1.1077295e-18, 1.6275555e-12, 3.2439461e-18, ..., 1.6261235e-10,
        1.5103105e-15, 6.6807934e-11],
       [3.5994651e-21, 3.3932133e-14, 1.1719911e-20, ..., 6.6690312e-18,
        8.4475326e-12, 1.6568691e-20],
       [4.8905044e-15, 3.8849861e-17, 4.2172232e-14, ..., 2.4197108e-22,
        4.5886515e-15, 5.1968563e-19]], dtype=float32)
```

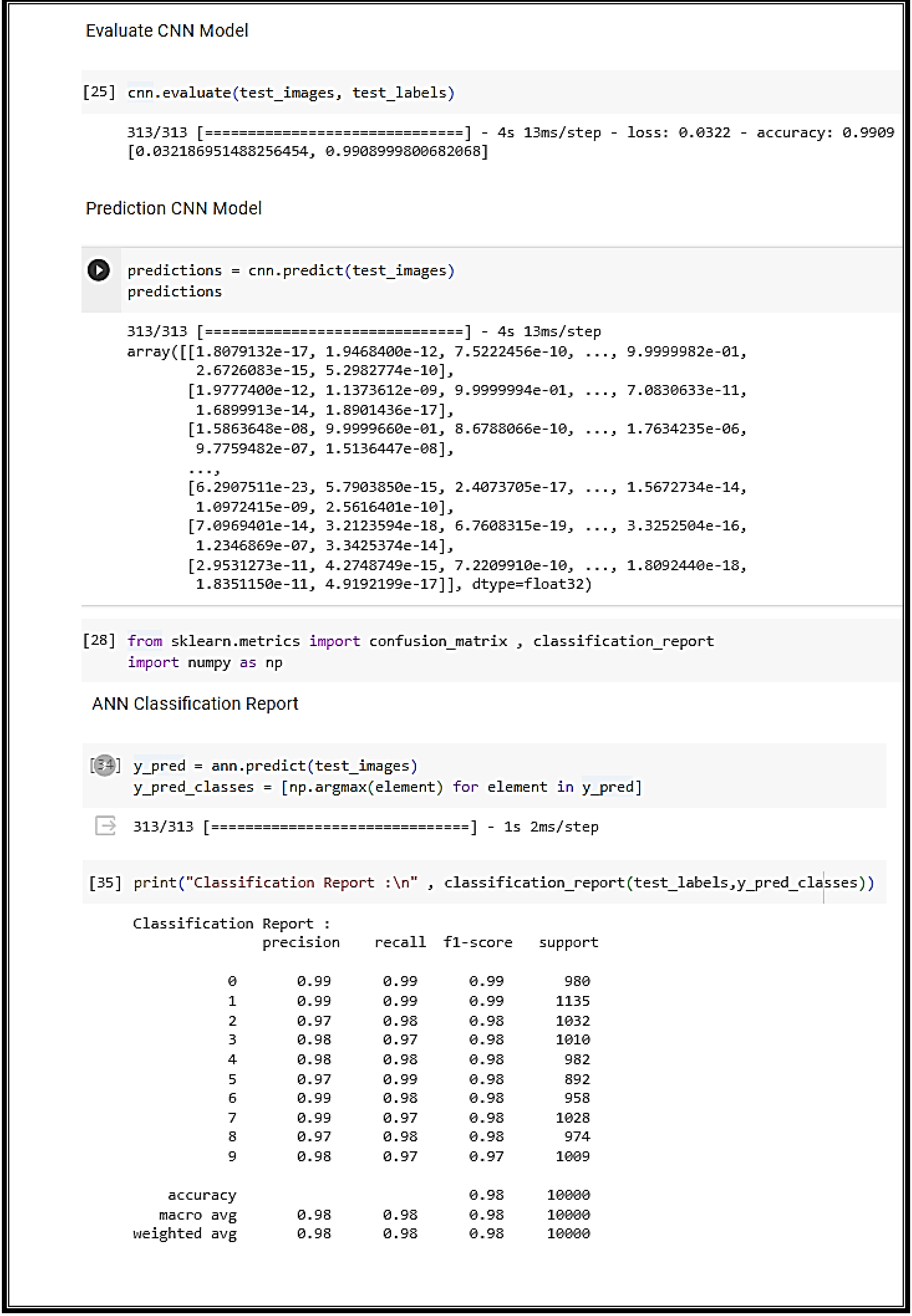
CNN Model

```
[22] cnn = models.Sequential([
         layers.Conv2D(filters=28, kernel_size=(3,3),activation='relu', input_shape=(28,28,1)),
         layers.MaxPooling2D((2,2)),

         layers.Conv2D(filters=64, kernel_size=(3,3),activation='relu'),
         layers.MaxPooling2D((2,2)),

         layers.Flatten(),
         layers.Dense(64, activation='relu'),
         layers.Dense(10, activation='softmax')
     ])
```

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

```
[24] cnn.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))
```

```
Epoch 1/10
1875/1875 [==============================] - 57s 30ms/step - loss: 0.1460 - accuracy: 0.9559 - val_loss: 0.0475 - val_accuracy: 0.9836
Epoch 2/10
1875/1875 [==============================] - 54s 29ms/step - loss: 0.0480 - accuracy: 0.9857 - val_loss: 0.0448 - val_accuracy: 0.9864
Epoch 3/10
1875/1875 [==============================] - 53s 28ms/step - loss: 0.0329 - accuracy: 0.9900 - val_loss: 0.0304 - val_accuracy: 0.9896
Epoch 4/10
1875/1875 [==============================] - 54s 29ms/step - loss: 0.0242 - accuracy: 0.9923 - val_loss: 0.0392 - val_accuracy: 0.9870
Epoch 5/10
1875/1875 [==============================] - 53s 28ms/step - loss: 0.0186 - accuracy: 0.9935 - val_loss: 0.0292 - val_accuracy: 0.9906
Epoch 6/10
1875/1875 [==============================] - 54s 29ms/step - loss: 0.0143 - accuracy: 0.9956 - val_loss: 0.0297 - val_accuracy: 0.9902
Epoch 7/10
1875/1875 [==============================] - 57s 30ms/step - loss: 0.0108 - accuracy: 0.9965 - val_loss: 0.0329 - val_accuracy: 0.9907
Epoch 8/10
1875/1875 [==============================] - 53s 28ms/step - loss: 0.0092 - accuracy: 0.9971 - val_loss: 0.0327 - val_accuracy: 0.9901
Epoch 9/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.0077 - accuracy: 0.9975 - val_loss: 0.0294 - val_accuracy: 0.9921
Epoch 10/10
1875/1875 [==============================] - 54s 29ms/step - loss: 0.0068 - accuracy: 0.9976 - val_loss: 0.0322 - val_accuracy: 0.9909
<keras.src.callbacks.History at 0x7c34e7bd6140>
```



Evaluate CNN Model

```
[25] cnn.evaluate(test_images, test_labels)

     313/313 [==============================] - 4s 13ms/step - loss: 0.0322 - accuracy: 0.9909
     [0.032186951488256454, 0.9908999800682068]
```

Prediction CNN Model

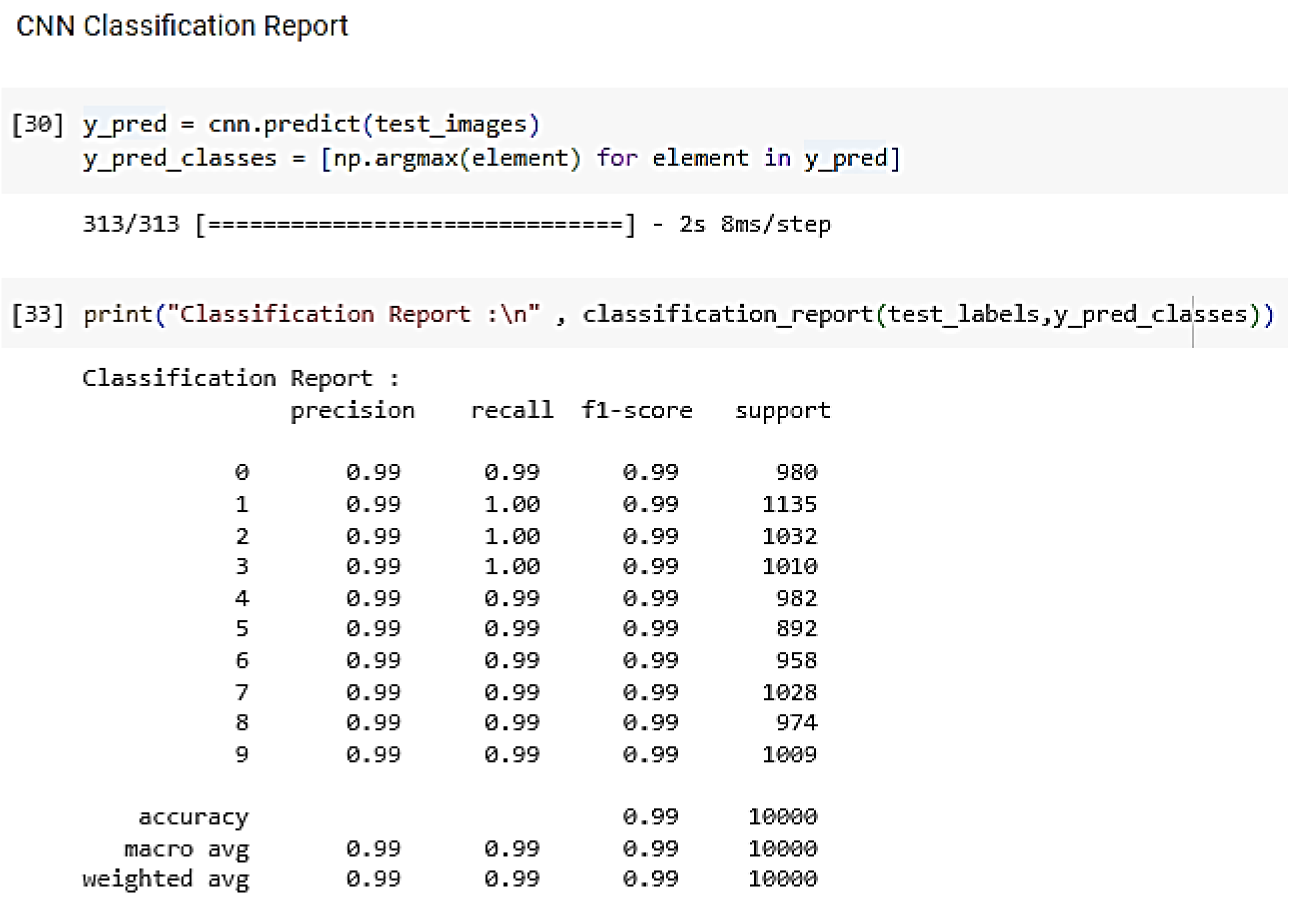
```
predictions = cnn.predict(test_images)
predictions

313/313 [==============================] - 4s 13ms/step
array([[1.8079132e-17, 1.9468400e-12, 7.5222456e-10, ..., 9.9999982e-01,
        2.6726083e-15, 5.2982774e-10],
       [1.9777400e-12, 1.1373612e-09, 9.9999994e-01, ..., 7.0830633e-11,
        1.6899913e-14, 1.8901436e-17],
       [1.5863648e-08, 9.9999660e-01, 8.6788066e-10, ..., 1.7634235e-06,
        9.7759482e-07, 1.5136447e-08],
       ...,
       [6.2907511e-23, 5.7903850e-15, 2.4073705e-17, ..., 1.5672734e-14,
        1.0972415e-09, 2.5616401e-10],
       [7.0969401e-14, 3.2123594e-18, 6.7608315e-19, ..., 3.3252504e-16,
        1.2346869e-07, 3.3425374e-14],
       [2.9531273e-11, 4.2748749e-15, 7.2209910e-10, ..., 1.8092440e-18,
        1.8351150e-11, 4.9192199e-17]], dtype=float32)
```

```
[28] from sklearn.metrics import confusion_matrix , classification_report
     import numpy as np
```

ANN Classification Report

```
[34] y_pred = ann.predict(test_images)
     y_pred_classes = [np.argmax(element) for element in y_pred]

     313/313 [==============================] - 1s 2ms/step
```

```
[35] print("Classification Report :\n" , classification_report(test_labels,y_pred_classes))

     Classification Report :
                    precision    recall  f1-score   support

                0       0.99      0.99      0.99       980
                1       0.99      0.99      0.99      1135
                2       0.97      0.98      0.98      1032
                3       0.98      0.97      0.98      1010
                4       0.98      0.98      0.98       982
                5       0.97      0.99      0.98       892
                6       0.99      0.98      0.98       958
                7       0.99      0.97      0.98      1028
                8       0.97      0.98      0.98       974
                9       0.98      0.97      0.97      1009

         accuracy                           0.98     10000
        macro avg       0.98      0.98      0.98     10000
     weighted avg       0.98      0.98      0.98     10000
```



CNN Classification Report

```
[30] y_pred = cnn.predict(test_images)
     y_pred_classes = [np.argmax(element) for element in y_pred]

     313/313 [==============================] - 2s 8ms/step

[33] print("Classification Report :\n" , classification_report(test_labels,y_pred_classes))

     Classification Report :
                   precision    recall  f1-score   support

                0       0.99      0.99      0.99       980
                1       0.99      1.00      0.99      1135
                2       0.99      1.00      0.99      1032
                3       0.99      1.00      0.99      1010
                4       0.99      0.99      0.99       982
                5       0.99      0.99      0.99       892
                6       0.99      0.99      0.99       958
                7       0.99      0.99      0.99      1028
                8       0.99      0.99      0.99       974
                9       0.99      0.99      0.99      1009

         accuracy                           0.99     10000
        macro avg       0.99      0.99      0.99     10000
     weighted avg       0.99      0.99      0.99     10000
```