21BDS0340

Abhinav Dinesh Srivatsa

Deep Learning Lab

Assessment – II

Aim:

To train a model using the MNIST digits dataset and predict values from test data using convolutional layers.

Procedure:

1. Load the MNIST digits dataset from keras
2. Create a utility function to plot a random digit
3. Create a utility function to plot a confusion matrix with a model and testing data
4. Create, compile, and fit the base CNN model
5. Evaluate the model using a confusion matrix
6. Create, compile, and fit the updated CNN model
7. Evaluate the model using a confusion matrix
8. Create, compile, and fit the updated CNN model
9. Evaluate the model using a confusion matrix

Code:

Interactive Python notebook attached below:

21BDS0340 - Abhinav Dinesh Srivatsa

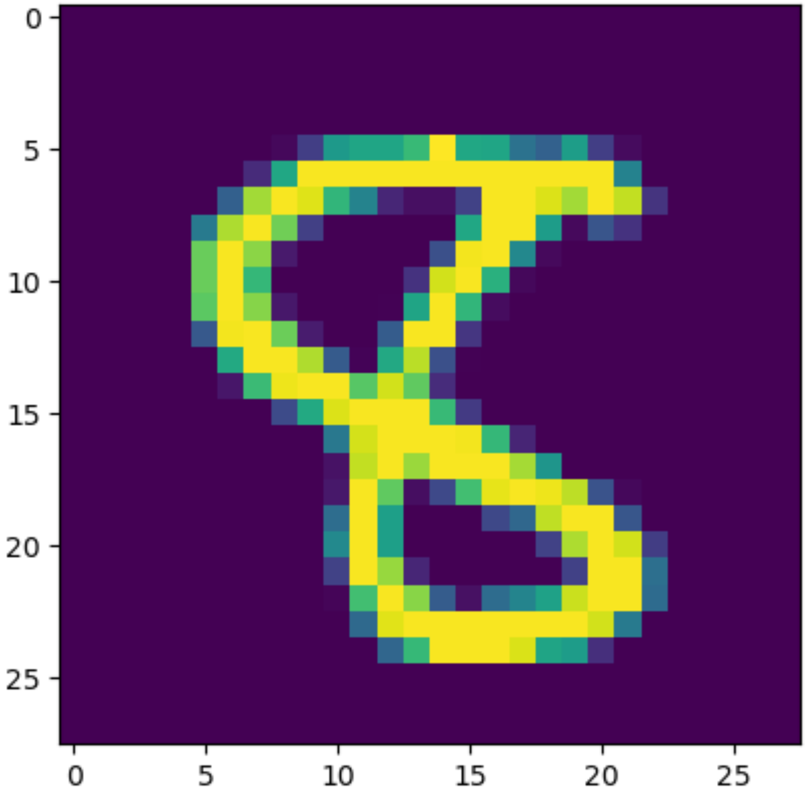
In [ ]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

In [ ]:
```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
X_train.shape
```

Out[ ]:
```
(60000, 28, 28)
```

In [ ]:
```
def plot_rand_digit(data):
    i = int(random.random() * len(data))
    plt.imshow(data[i])
```

In [ ]:
```
plot_rand_digit(X_train)
```



In [ ]:
```
# normalisation
X_train_norm = X_train / 256
X_test_norm = X_test / 256
X_train_norm
```

```
Out[ ]: 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.],
                ...,
                [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.],
                [0., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.]]])
```

In [ ]:

```
def confusion_matrix(model, X_test, y_test):
    y_pred = model.predict(X_test)
    y_pred = [row.argmax() for row in y_pred]
    mat = np.zeros((10, 10), dtype=int)
    for i in range(len(y_pred)):
        pred = y_pred[i]
        real = y_test[i]
        mat[pred][real] += 1
    fig, ax = plt.subplots(figsize=(10, 9))
    sns.heatmap(mat, cmap='mako', annot=True, fmt='', ax=ax)
```

In [ ]:

```
# convolutional model 1
model1 = tf.keras.Sequential([
    tf.keras.layers.Input((28, 28, 1)),

    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),

    tf.keras.layers.Conv2D(48, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(10, activation="softmax")
])

model1.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model1.fit(X_train_norm, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.2338 - accuracy
: 0.9268
Epoch 2/5
1875/1875 [==============================] - 14s 8ms/step - loss: 0.0903 - accuracy
: 0.9725
Epoch 3/5
1875/1875 [==============================] - 14s 8ms/step - loss: 0.0721 - accuracy
: 0.9775
Epoch 4/5
1875/1875 [==============================] - 14s 8ms/step - loss: 0.0609 - accuracy
: 0.9815
Epoch 5/5
1875/1875 [==============================] - 14s 8ms/step - loss: 0.0558 - accuracy
: 0.9826
```

Out[ ]:

```
<keras.src.callbacks.History at 0x21c260b5100>
```

In [ ]:

```
model1.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d_4 (MaxPoolin  (None, 13, 13, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_5 (Conv2D)           (None, 11, 11, 48)        13872     
                                                                 
 max_pooling2d_5 (MaxPoolin  (None, 5, 5, 48)          0         
 g2D)                                                            
                                                                 
 dropout_2 (Dropout)         (None, 5, 5, 48)          0         
                                                                 
 flatten_2 (Flatten)         (None, 1200)              0         
                                                                 
 dense_6 (Dense)             (None, 10)                12010     
                                                                 
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d_4 (MaxPoolin  (None, 13, 13, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_5 (Conv2D)           (None, 11, 11, 48)        13872     
                                                                 
 max_pooling2d_5 (MaxPoolin  (None, 5, 5, 48)          0         
 g2D)                                                            
                                                                 
 dropout_2 (Dropout)         (None, 5, 5, 48)          0         
                                                                 
 flatten_2 (Flatten)         (None, 1200)              0         
                                                                 
 dense_6 (Dense)             (None, 10)                12010     
                                                                 
```

```
=================================================================
Total params: 26202 (102.35 KB)
Trainable params: 26202 (102.35 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```
model1.evaluate(X_test_norm, y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0288 - accuracy:
0.9896
```

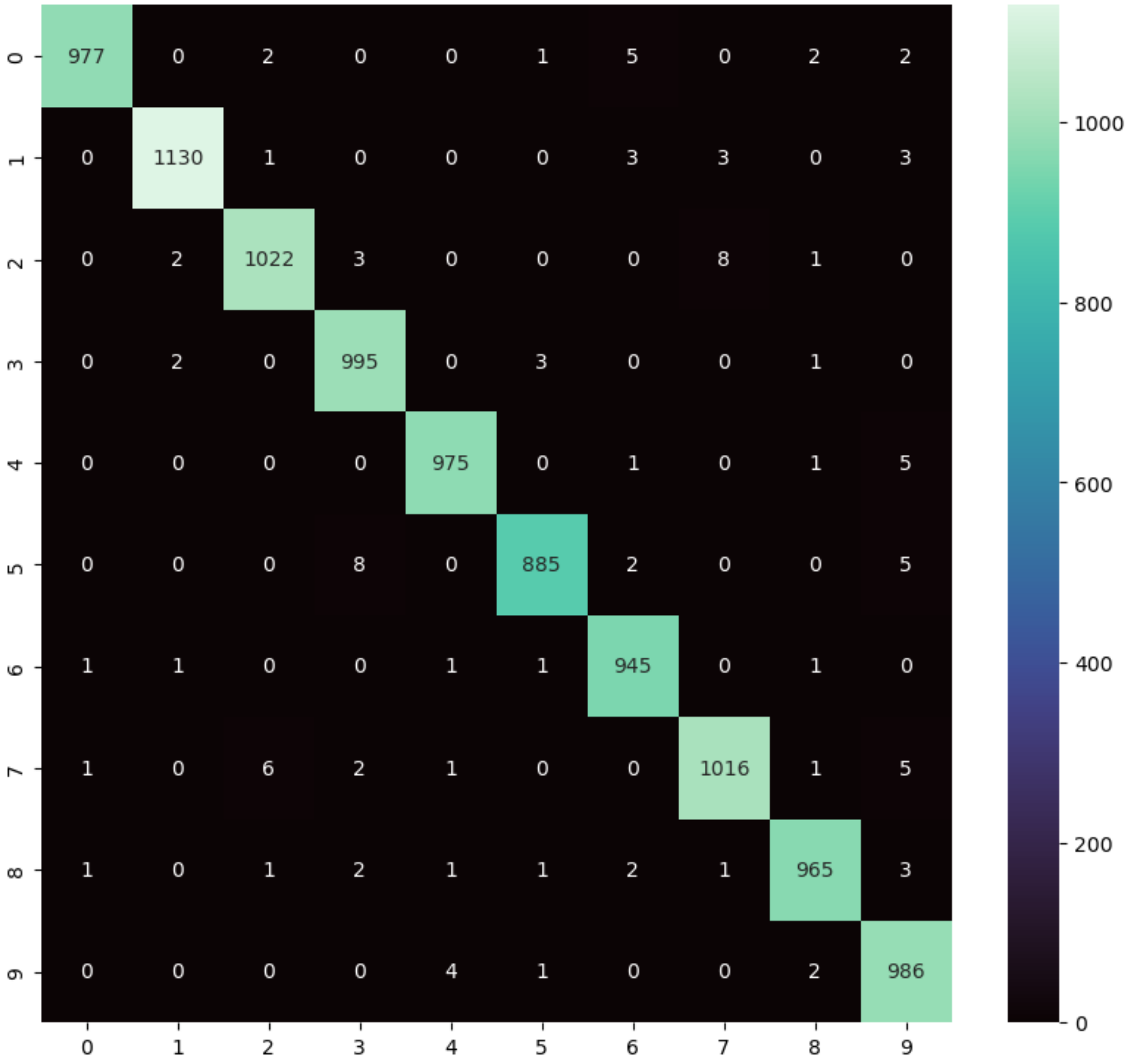
Out[ ]:

```
[0.028803551569581032, 0.9896000027656555]
```

In [ ]:

```
confusion_matrix(model1, X_test_norm, y_test)
```

```
313/313 [==============================] - 1s 3ms/step
```



	0	1	2	3	4	5	6	7	8	9
0	977	0	2	0	0	1	5	0	2	2
1	0	1130	1	0	0	0	3	3	0	3
2	0	2	1022	3	0	0	0	8	1	0
3	0	2	0	995	0	3	0	0	1	0
4	0	0	0	0	975	0	1	0	1	5
5	0	0	0	8	0	885	2	0	0	5
6	1	1	0	0	1	1	945	0	1	0
7	1	0	6	2	1	0	0	1016	1	5
8	1	0	1	2	1	1	2	1	965	3
9	0	0	0	0	4	1	0	0	2	986

In [ ]:

```
# convolutional model 2
model2 = tf.keras.Sequential([
    tf.keras.layers.Input((28, 28, 1)),

    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),

    tf.keras.layers.Conv2D(48, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(100, activation="softmax"),
    tf.keras.layers.Dense(10, activation="softmax")
])

model2.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model2.fit(X_train_norm, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 17s 9ms/step - loss: 1.2371 - accuracy
: 0.7348
Epoch 2/5
1875/1875 [==============================] - 16s 8ms/step - loss: 0.5417 - accuracy
: 0.7897
Epoch 3/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.4166 - accuracy
: 0.8001
Epoch 4/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.3499 - accuracy
: 0.8403
Epoch 5/5
1875/1875 [==============================] - 16s 8ms/step - loss: 0.2384 - accuracy
: 0.8972
```

Out[ ]:

```
<keras.src.callbacks.History at 0x21c458ea8e0>
```

In [ ]:

```
model2.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_8 (MaxPoolin  (None, 13, 13, 32)        0
 g2D)

 conv2d_9 (Conv2D)           (None, 11, 11, 48)        13872

 max_pooling2d_9 (MaxPoolin  (None, 5, 5, 48)          0
 g2D)

 dropout_4 (Dropout)         (None, 5, 5, 48)          0

 flatten_4 (Flatten)         (None, 1200)              0

 dense_9 (Dense)             (None, 100)               120100

 dense_10 (Dense)            (None, 10)                1010

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_8 (MaxPoolin  (None, 13, 13, 32)        0
 g2D)

 conv2d_9 (Conv2D)           (None, 11, 11, 48)        13872

 max_pooling2d_9 (MaxPoolin  (None, 5, 5, 48)          0
 g2D)

 dropout_4 (Dropout)         (None, 5, 5, 48)          0

 flatten_4 (Flatten)         (None, 1200)              0

 dense_9 (Dense)             (None, 100)               120100

 dense_10 (Dense)            (None, 10)                1010

=================================================================
Total params: 135302 (528.52 KB)
Trainable params: 135302 (528.52 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```
model2.evaluate(X_test_norm, y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.1928 - accuracy:
0.9130
```

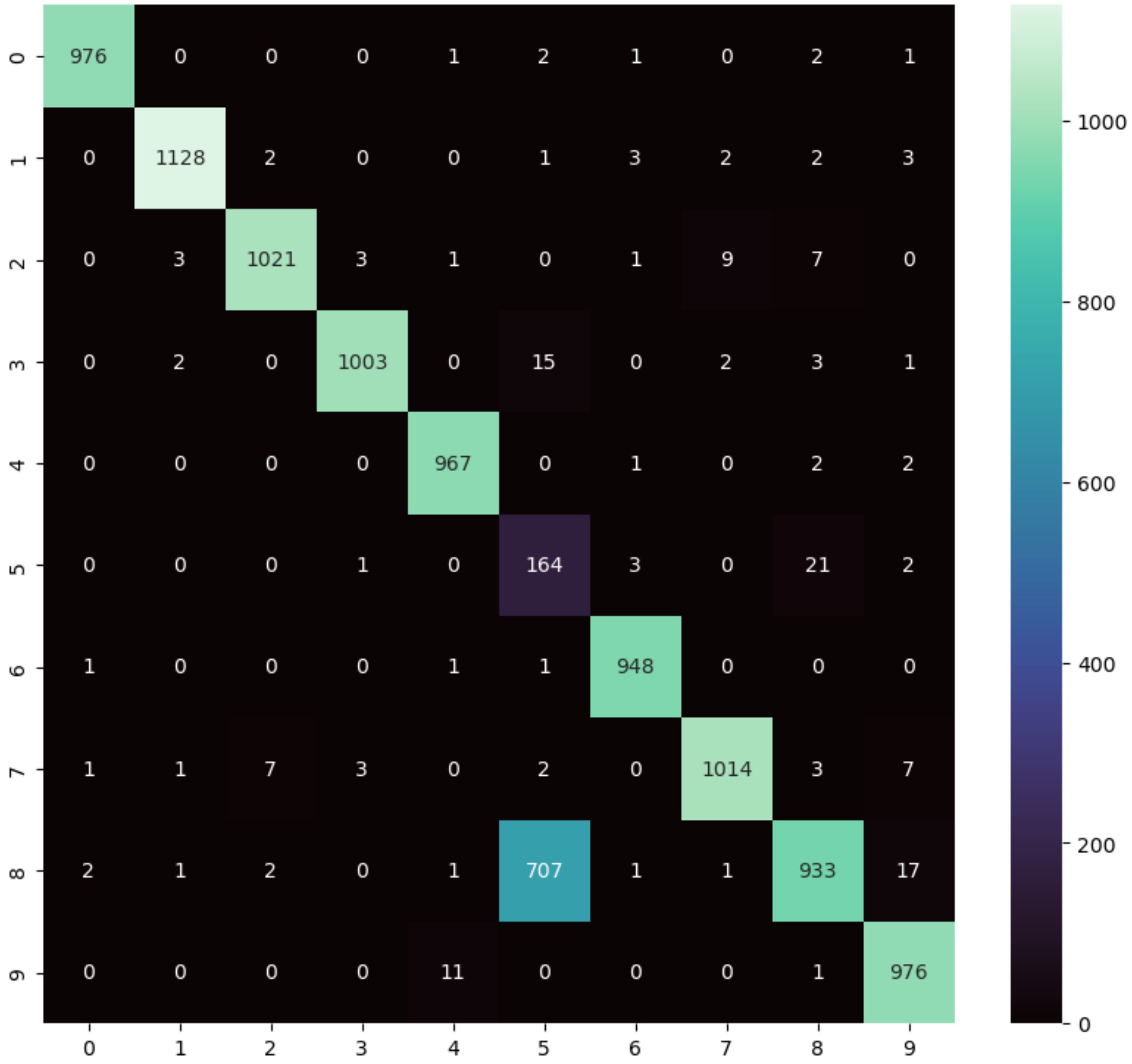
Out[ ]:

```
[0.19282568991184235, 0.9129999876022339]
```

In [ ]:

```
confusion_matrix(model2, X_test_norm, y_test)
```

```
313/313 [==============================] - 1s 3ms/step
```



	0	1	2	3	4	5	6	7	8	9
0	976	0	0	0	1	2	1	0	2	1
1	0	1128	2	0	0	1	3	2	2	3
2	0	3	1021	3	1	0	1	9	7	0
3	0	2	0	1003	0	15	0	2	3	1
4	0	0	0	0	967	0	1	0	2	2
5	0	0	0	1	0	164	3	0	21	2
6	1	0	0	0	1	1	948	0	0	0
7	1	1	7	3	0	2	0	1014	3	7
8	2	1	2	0	1	707	1	1	933	17
9	0	0	0	0	11	0	0	0	1	976

In [ ]:

```
# convolutional model 3
model3 = tf.keras.Sequential([
    tf.keras.layers.Input((28, 28, 1)),

    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),

    tf.keras.layers.Conv2D(16, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.Conv2D(16, kernel_size=(3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(10, activation="softmax")
])

model3.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model3.fit(X_train_norm, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 36s 19ms/step - loss: 0.1806 - accurac
y: 0.9444
Epoch 2/5
1875/1875 [==============================] - 35s 18ms/step - loss: 0.0613 - accurac
y: 0.9809
Epoch 3/5
1875/1875 [==============================] - 35s 18ms/step - loss: 0.0461 - accurac
y: 0.9854
Epoch 4/5
1875/1875 [==============================] - 37s 20ms/step - loss: 0.0362 - accurac
y: 0.9880
Epoch 5/5
1875/1875 [==============================] - 39s 21ms/step - loss: 0.0304 - accurac
y: 0.9904
```

Out[ ]:

```
<keras.src.callbacks.History at 0x21c2627a910>
```

In [ ]:

```
model3.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_10 (Conv2D)          (None, 26, 26, 32)        320       
                                                                 
 conv2d_11 (Conv2D)          (None, 24, 24, 32)        9248      
                                                                 
 max_pooling2d_10 (MaxPooli  (None, 12, 12, 32)        0         
 ng2D)                                                           
                                                                 
 conv2d_12 (Conv2D)          (None, 10, 10, 16)        4624      
                                                                 
 conv2d_13 (Conv2D)          (None, 8, 8, 16)          2320      
                                                                 
 max_pooling2d_11 (MaxPooli  (None, 4, 4, 16)          0         
 ng2D)                                                           
                                                                 
 flatten_5 (Flatten)         (None, 256)               0         
                                                                 

_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_10 (Conv2D)          (None, 26, 26, 32)        320       
                                                                 
 conv2d_11 (Conv2D)          (None, 24, 24, 32)        9248      
                                                                 
 max_pooling2d_10 (MaxPooli  (None, 12, 12, 32)        0         
 ng2D)                                                           
                                                                 
 conv2d_12 (Conv2D)          (None, 10, 10, 16)        4624      
                                                                 
 conv2d_13 (Conv2D)          (None, 8, 8, 16)          2320      
                                                                 
 max_pooling2d_11 (MaxPooli  (None, 4, 4, 16)          0         
 ng2D)                                                           
                                                                 
 flatten_5 (Flatten)         (None, 256)               0         
                                                                 
 dense_11 (Dense)            (None, 10)                2570      
                                                                 
=================================================================
Total params: 19082 (74.54 KB)
Trainable params: 19082 (74.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```
model3.evaluate(X_test_norm, y_test)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.0358 - accuracy: 
0.9896
```

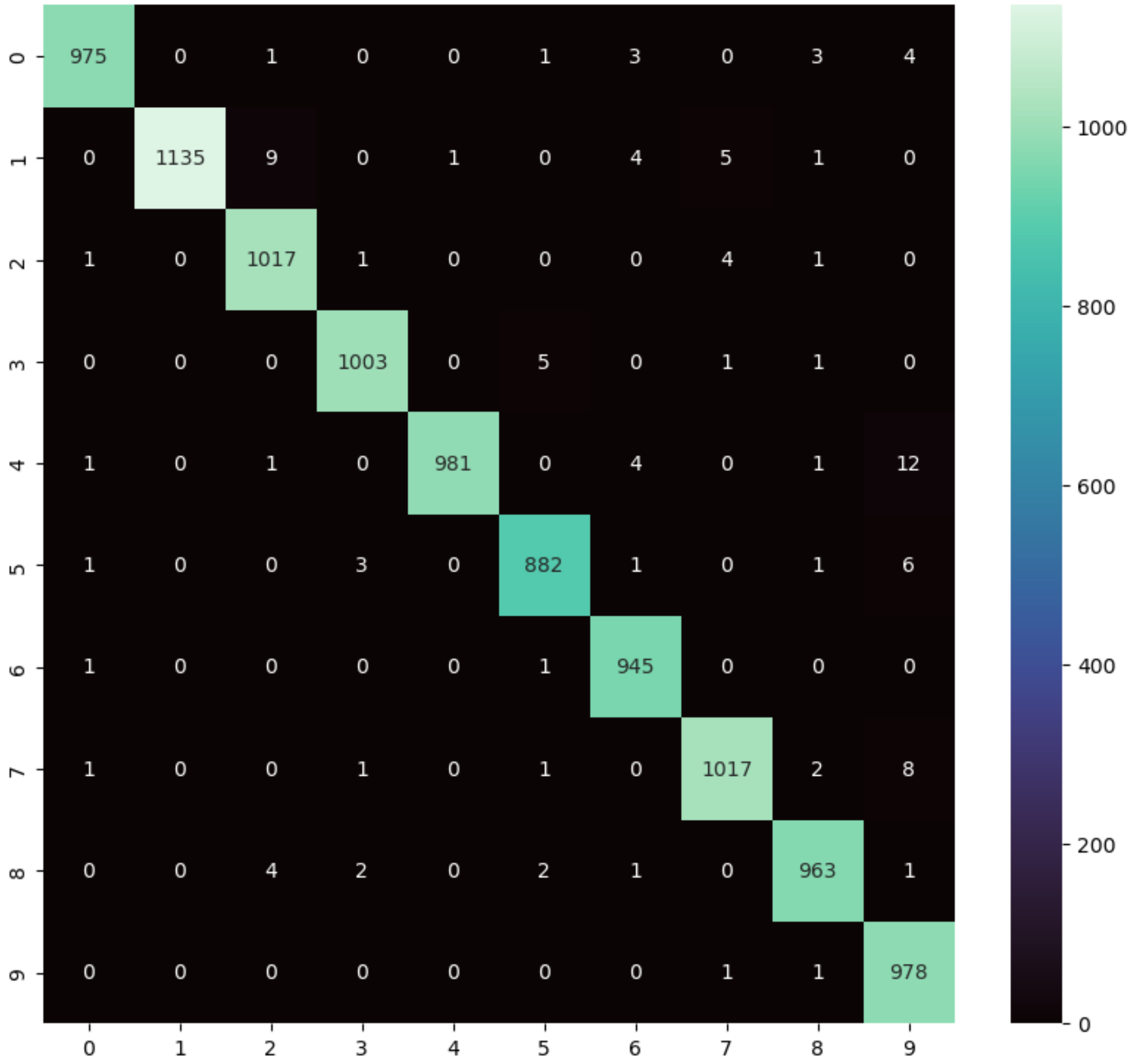
Out[ ]:

```
[0.035829197615385056, 0.9896000027656555]
```

In [ ]:

```
confusion_matrix(model3, X_test_norm, y_test)
```

```
313/313 [==============================] - 2s 5ms/step
```



	0	1	2	3	4	5	6	7	8	9
0	975	0	1	0	0	1	3	0	3	4
1	0	1135	9	0	1	0	4	5	1	0
2	1	0	1017	1	0	0	0	4	1	0
3	0	0	0	1003	0	5	0	1	1	0
4	1	0	1	0	981	0	4	0	1	12
5	1	0	0	3	0	882	1	0	1	6
6	1	0	0	0	0	1	945	0	0	0
7	1	0	0	1	0	1	0	1017	2	8
8	0	0	4	2	0	2	1	0	963	1
9	0	0	0	0	0	0	0	1	1	978