21BDS0340 - Abhinav Dinesh Srivatsa

Deep Learning Lab

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

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

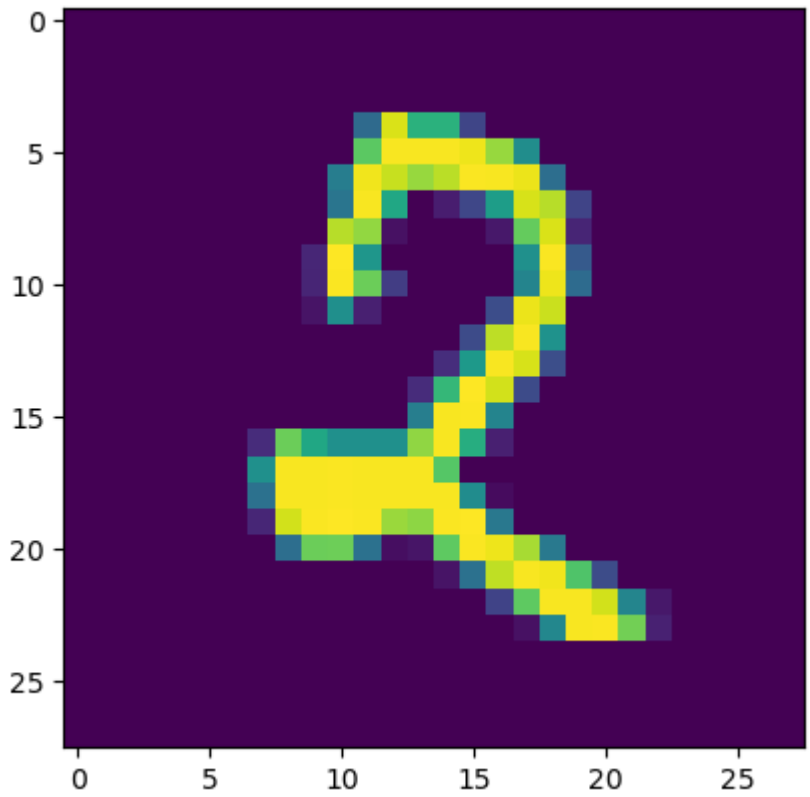
Out[ ]:
```
(60000, 28, 28)
```

In [ ]:
```
X_train_flat = X_train.reshape(len(X_train), 28 * 28)
X_test_flat = X_test.reshape(len(X_test), 28 * 28)
X_train_flat.shape
```

Out[ ]:
```
(60000, 784)
```

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

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



In [ ]:
```
def confusion_matrix(model, X_test, y_test):
    y_pred = model.predict(X_test_flat)
    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)
```

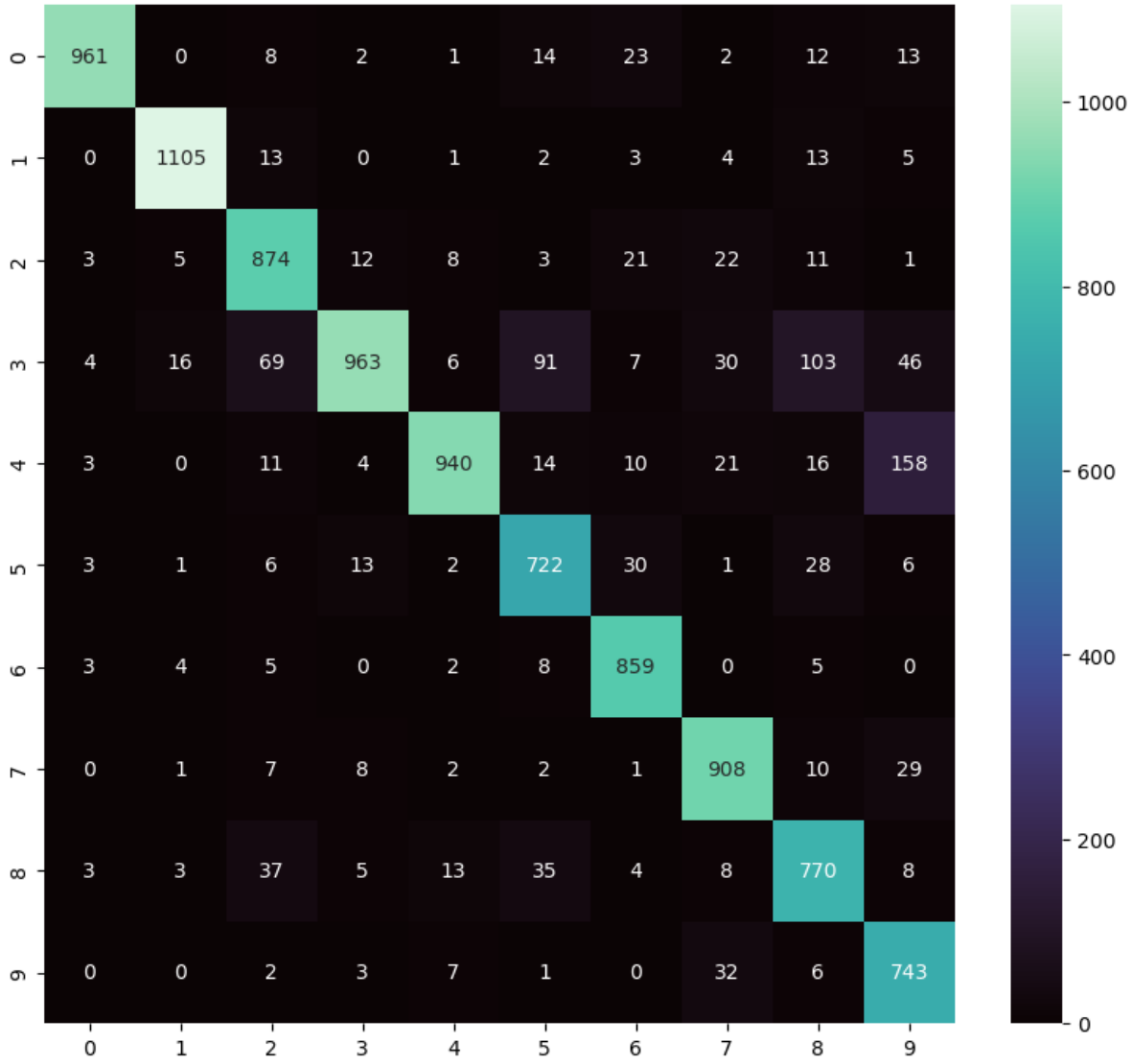
In [ ]:
```
model1 = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(784,)),
    tf.keras.layers.Dense(10, activation='softmax'),
])

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

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

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 539us/step - accuracy: 0.7781 - loss: 16.0441
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 547us/step - accuracy: 0.8745 - loss: 6.2690
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 528us/step - accuracy: 0.8822 - loss: 5.7668
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 522us/step - accuracy: 0.8887 - loss: 5.3183
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 533us/step - accuracy: 0.8873 - loss: 5.2703
```

Out[ ]:
```
<keras.src.callbacks.history.History at 0x2914ff9ae50>
```

In [ ]:
```
confusion_matrix(model1, X_test_flat, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 522us/step
```



In [ ]:
```
# scaling
X_train_flat_scaled = X_train_flat / 256
X_test_flat_scaled = X_test_flat / 256
```

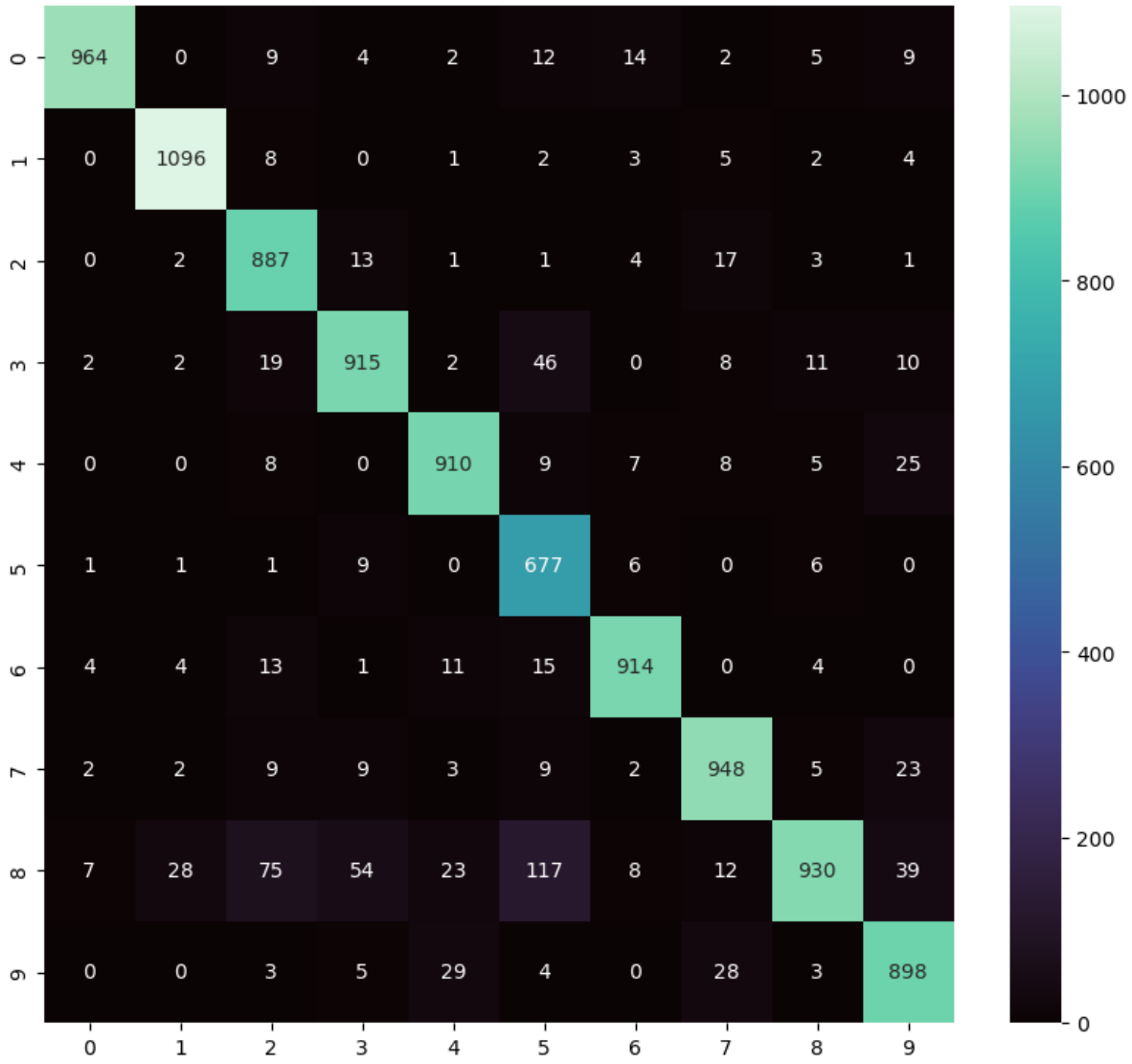
In [ ]:
```
model2 = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(784,)),
    tf.keras.layers.Dense(10, activation='softmax'),
])

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

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

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 529us/step - accuracy: 0.8216 - loss: 0.7004
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 534us/step - accuracy: 0.9149 - loss: 0.3091
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 532us/step - accuracy: 0.9209 - loss: 0.2832
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 530us/step - accuracy: 0.9240 - loss: 0.2745
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 534us/step - accuracy: 0.9270 - loss: 0.2656
```

Out[ ]: <keras.src.callbacks.history.History at 0x291501057f0>

In [ ]:
```
confusion_matrix(model2, X_test_flat_scaled, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 522us/step
```



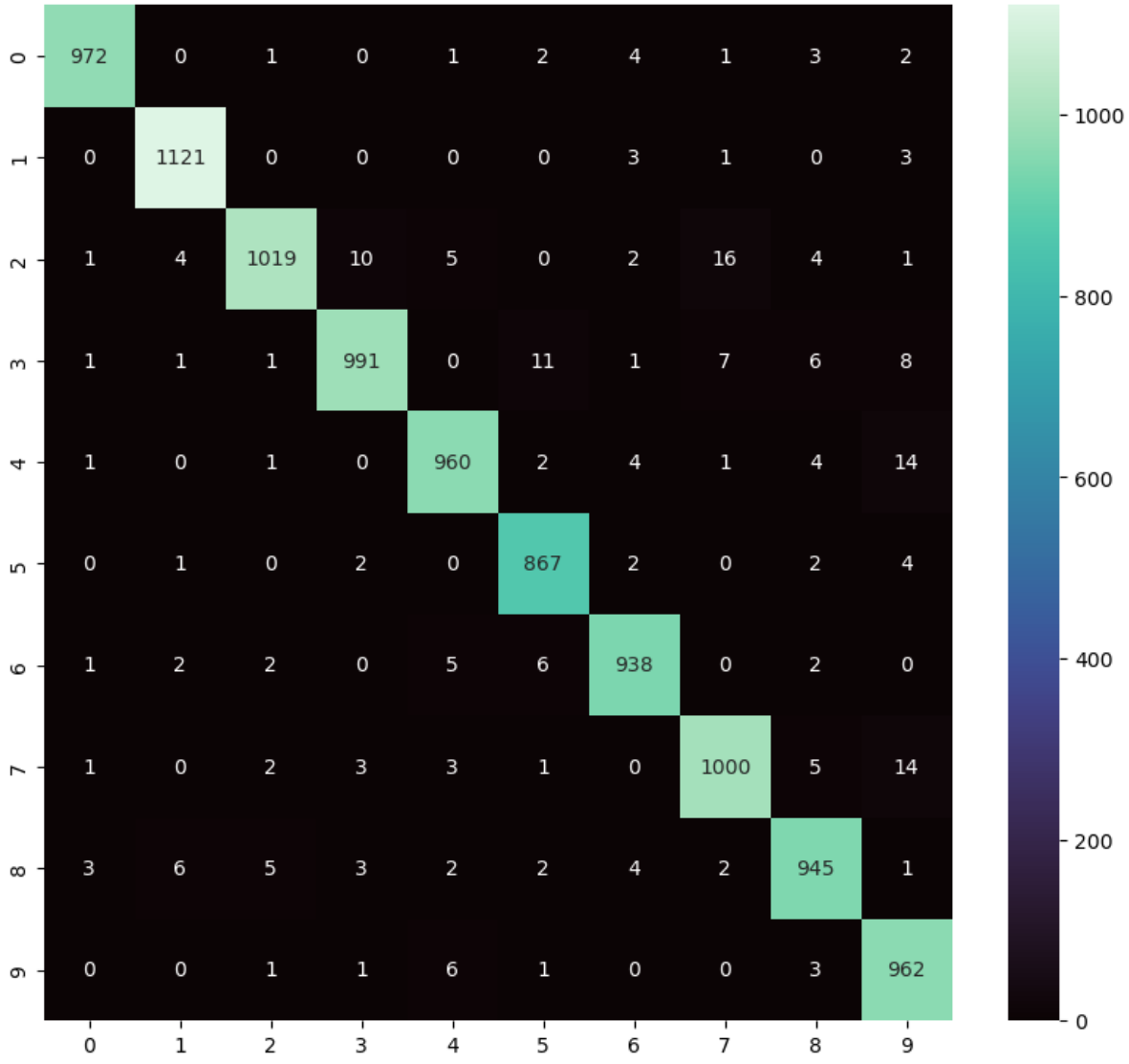
In [ ]:
```
model3 = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(784,)),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

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

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

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 779us/step - accuracy: 0.8743 - loss: 0.4465
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 777us/step - accuracy: 0.9602 - loss: 0.1345
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 780us/step - accuracy: 0.9747 - loss: 0.0858
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 807us/step - accuracy: 0.9808 - loss: 0.0628
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 794us/step - accuracy: 0.9843 - loss: 0.0515
```

Out[ ]:
```
<keras.src.callbacks.history.History at 0x29150298fd0>
```

In [ ]:
```
confusion_matrix(model3, X_test_flat_scaled, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 593us/step
```


0 972 0 1 0 1 2 4 1 3 2
1 0 1121 0 0 0 0 3 1 0 3
2 1 4 1019 10 5 0 2 16 4 1
3 1 1 1 991 0 11 1 7 6 8
4 1 0 1 0 960 2 4 1 4 14
5 0 1 0 2 0 867 2 0 2 4
6 1 2 2 0 5 6 938 0 2 0
7 1 0 2 3 3 1 0 1000 5 14
8 3 6 5 3 2 2 4 2 945 1
9 0 0 1 1 6 1 0 0 3 962
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

In [ ]: