digits-classification

August 21, 2024

importing libraries

[57]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
%matplotlib inline
```

1 data pre processing

[58]:
```
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

[60]:
```
X_train.shape
```

[60]:
```
(60000, 28, 28)
```

[61]:
```
len(X_train)
```

[61]:
```
60000
```

[62]:
```
len(X_test)
```

[62]:
```
10000
```

[63]:
```
X_train[0]
```

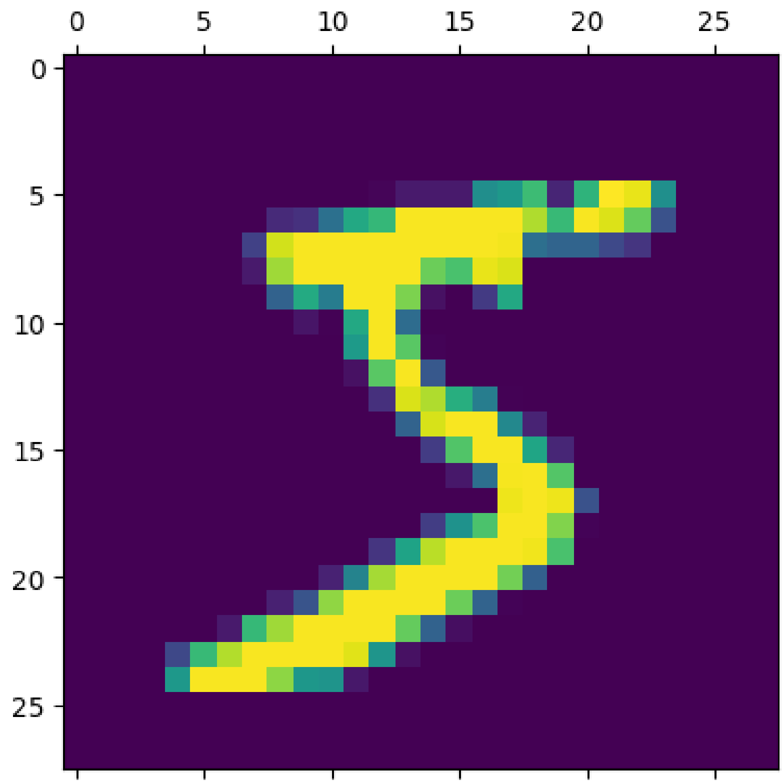
[63]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   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,   3,
  18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
 253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
 253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
 253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
 205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
  90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
 190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
 253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
 241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
   0,   0],
```

```
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
         148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
         253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
         253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
         195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
          11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0]], dtype=uint8)
```

[64]:
```
plt.matshow(X_train[0])
```

[64]:
```
<matplotlib.image.AxesImage at 0x7b7a2e305fc0>
```

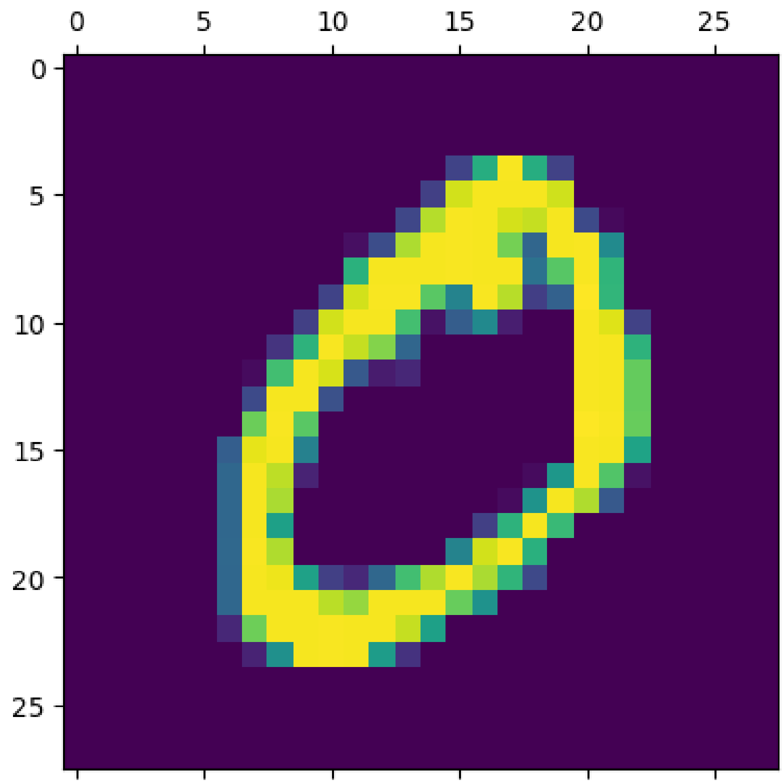


[65]:
```
y_train[0]
```

[65]: 5

[66]:
```
plt.matshow(X_train[1])
```

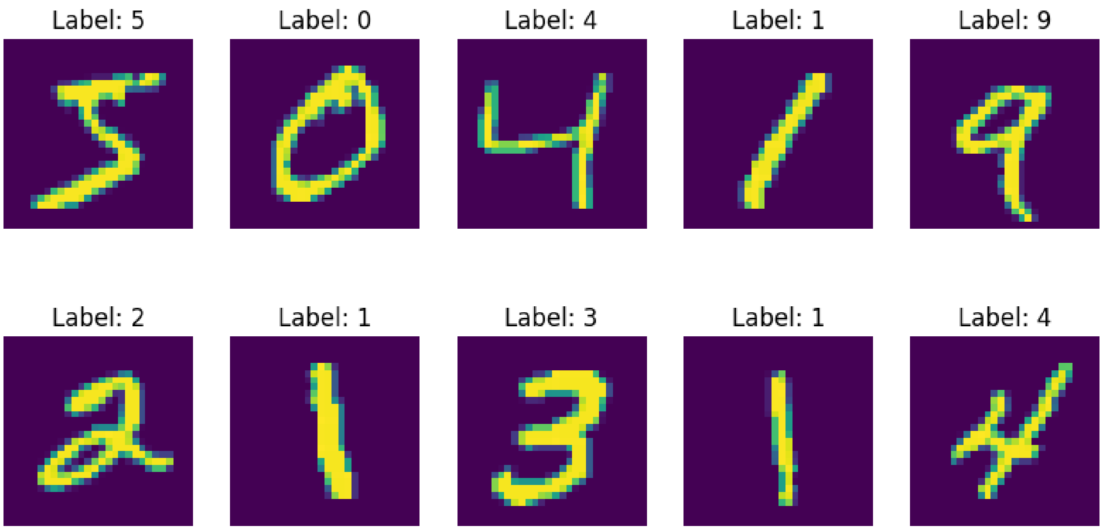
[66]: <matplotlib.image.AxesImage at 0x7b7a2e3ab760>



[67]:
```
y_train[1]
```

[67]: 0

[68]:
```
plt.figure(figsize=(10, 5))
for i in range(10):
  plt.subplot(2, 5, i+1)
  plt.imshow(X_train[i])
  plt.title(f'Label: {y_train[i]}')
  plt.axis('off')
plt.show()
```



[69]:
```
X_train = X_train / 255
X_test = X_test / 255
```

[70]:
```
X_train[0]
```

[70]:
```
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.        , 0.01176471, 0.07058824, 0.07058824,
 0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
 0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.11764706, 0.14117647,
 0.36862745, 0.60392157, 0.66666667, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.88235294, 0.6745098 ,
 0.99215686, 0.94901961, 0.76470588, 0.25098039, 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.19215686, 0.93333333, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.98431373, 0.36470588, 0.32156863,
 0.32156863, 0.21960784, 0.15294118, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.07058824, 0.85882353, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.77647059,
 0.71372549, 0.96862745, 0.94509804, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.31372549, 0.61176471,
 0.41960784, 0.99215686, 0.99215686, 0.80392157, 0.04313725,
 0.        , 0.16862745, 0.60392157, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.05490196,
 0.00392157, 0.60392157, 0.99215686, 0.35294118, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.54509804, 0.99215686, 0.74509804, 0.00784314,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.04313725, 0.74509804, 0.99215686, 0.2745098 ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.1372549 , 0.94509804, 0.88235294,
 0.62745098, 0.42352941, 0.00392157, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.31764706, 0.94117647,
 0.99215686, 0.99215686, 0.46666667, 0.09803922, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.17647059,
 0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.0627451 , 0.36470588, 0.98823529, 0.99215686, 0.73333333,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.97647059, 0.99215686, 0.97647059,
 0.25098039, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.18039216,
 0.50980392, 0.71764706, 0.99215686, 0.99215686, 0.81176471,
 0.00784314, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.15294118, 0.58039216, 0.89803922,
 0.99215686, 0.99215686, 0.99215686, 0.98039216, 0.71372549,
```

```
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.09019608, 0.25882353,
 0.83529412, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
 0.77647059, 0.31764706, 0.00784314, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.07058824, 0.67058824, 0.85882353, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,
 0.03529412, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.21568627,
 0.6745098 , 0.88627451, 0.99215686, 0.99215686, 0.99215686,
 0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.53333333,
 0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,
 0.51764706, 0.0627451 , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ]])
```

[71]:
```
#Reshape the data to 1D Array
X_train_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flattened = X_test.reshape(len(X_test), 28*28)
```

[72]:
```
X_train_flattened.shape
```

[72]:
```
(60000, 784)
```

2 building and compiling the model

[74]:
```
model = keras.Sequential([
        keras.layers.Dense(64, input_shape=(784,), activation='relu',␣
 ↪kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.2),  # Added dropout for regularization
        keras.layers.Dense(10, activation='softmax')
])

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

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

3 fitting the model

[75]:
```
model.fit(X_train_flattened,y_train,epochs=20)
```

```
Epoch 1/20
1875/1875            8s 4ms/step -
accuracy: 0.8210 - loss: 0.6841
Epoch 2/20
1875/1875            7s 2ms/step -
accuracy: 0.9346 - loss: 0.3073
Epoch 3/20
1875/1875            7s 4ms/step -
accuracy: 0.9415 - loss: 0.2712
Epoch 4/20
```

```
1875/1875              8s 2ms/step -
accuracy: 0.9479 - loss: 0.2491
Epoch 5/20
1875/1875              6s 3ms/step -
accuracy: 0.9510 - loss: 0.2321
Epoch 6/20
1875/1875              5s 2ms/step -
accuracy: 0.9536 - loss: 0.2236
Epoch 7/20
1875/1875              6s 3ms/step -
accuracy: 0.9553 - loss: 0.2218
Epoch 8/20
1875/1875              6s 3ms/step -
accuracy: 0.9563 - loss: 0.2142
Epoch 9/20
1875/1875              10s 3ms/step -
accuracy: 0.9570 - loss: 0.2097
Epoch 10/20
1875/1875              5s 2ms/step -
accuracy: 0.9586 - loss: 0.2038
Epoch 11/20
1875/1875              4s 2ms/step -
accuracy: 0.9578 - loss: 0.2048
Epoch 12/20
1875/1875              7s 3ms/step -
accuracy: 0.9591 - loss: 0.1990
Epoch 13/20
1875/1875              9s 2ms/step -
accuracy: 0.9609 - loss: 0.1948
Epoch 14/20
1875/1875              6s 3ms/step -
accuracy: 0.9593 - loss: 0.1982
Epoch 15/20
1875/1875              4s 2ms/step -
accuracy: 0.9612 - loss: 0.1904
Epoch 16/20
1875/1875              6s 3ms/step -
accuracy: 0.9593 - loss: 0.1985
Epoch 17/20
1875/1875              5s 3ms/step -
accuracy: 0.9615 - loss: 0.1927
Epoch 18/20
1875/1875              4s 2ms/step -
accuracy: 0.9610 - loss: 0.1912
Epoch 19/20
1875/1875              5s 3ms/step -
accuracy: 0.9569 - loss: 0.1991
Epoch 20/20
```

```
1875/1875 9s 2ms/step -
accuracy: 0.9601 - loss: 0.1934
```

[75]: <keras.src.callbacks.history.History at 0x7b7a3d9c65f0>

4 Model Evaluating

[76]:
```
model.evaluate(X_test_flattened, y_test)
```

```
313/313 2s 5ms/step -
accuracy: 0.9668 - loss: 0.1782
```

[76]: [0.15831340849399567, 0.9721999764442444]

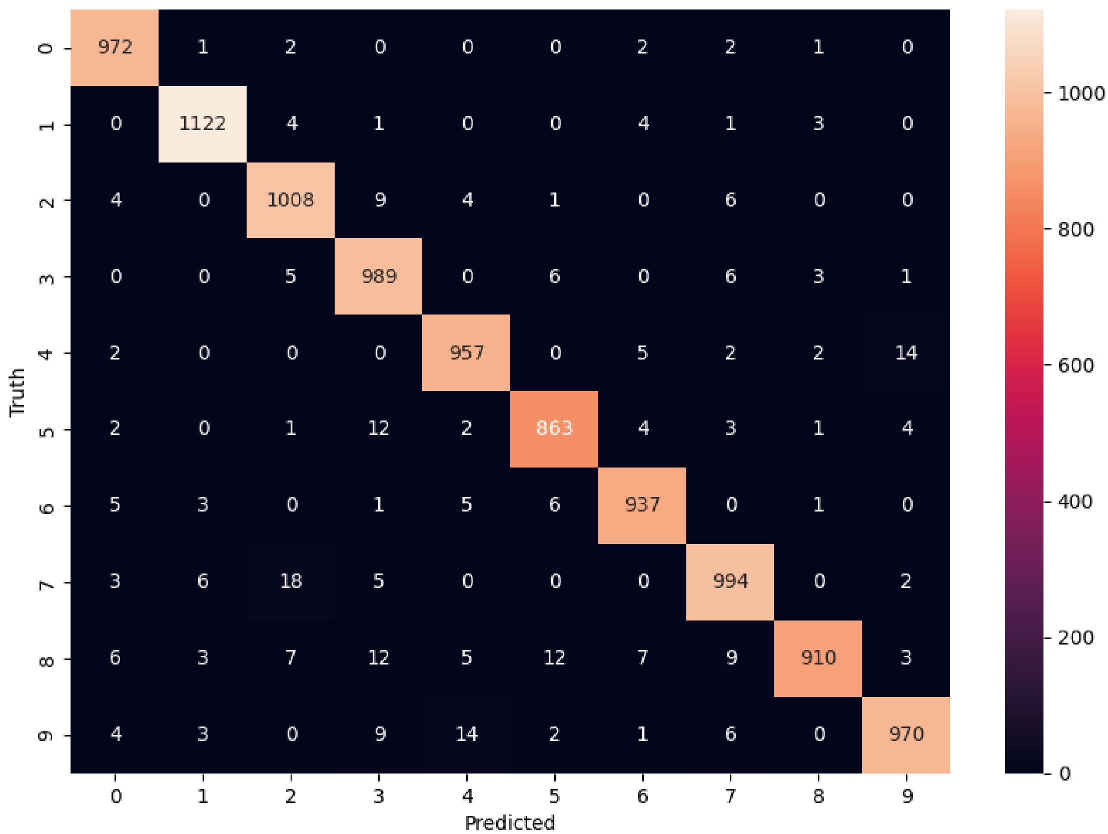
[77]:
```
y_predicted = model.predict(X_test_flattened)
y_predicted_labels = [np.argmax(i) for i in y_predicted]
cm = tf.math.confusion_matrix(labels=y_test, predictions=y_predicted_labels)
```

```
313/313 1s 3ms/step
```

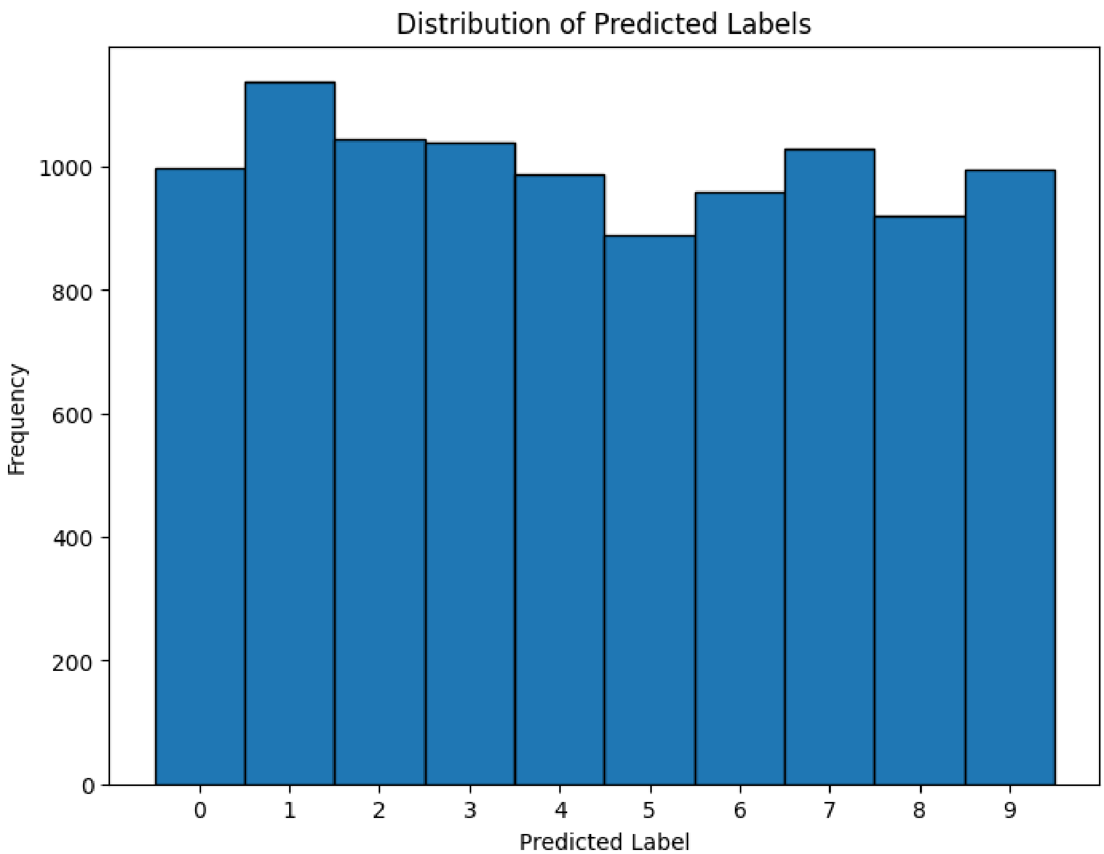
5 First performance visualization

[78]:
```
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

[78]: Text(95.72222222222221, 0.5, 'Truth')



[79]:
```python
# Plotting the distribution of predicted labels
plt.figure(figsize=(8, 6))

plt.hist(y_predicted_labels, bins=np.arange(11) - 0.5, edgecolor='black')
plt.xticks(range(10))
plt.xlabel('Predicted Label')
plt.ylabel('Frequency')
plt.title('Distribution of Predicted Labels')
plt.show()
```

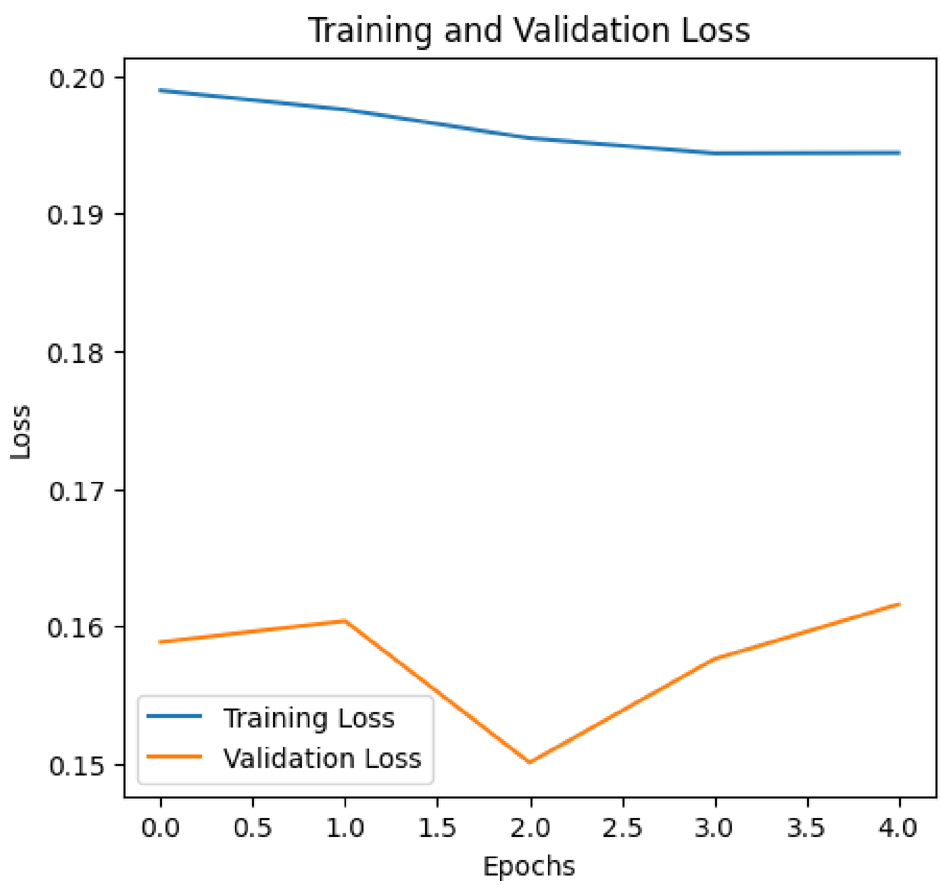


[80]:
```python
# Plotting the loss and accuracy curves
history = model.fit(X_train_flattened, y_train, epochs=5,␣
↪validation_data=(X_test_flattened, y_test))
# Plotting the loss
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
```

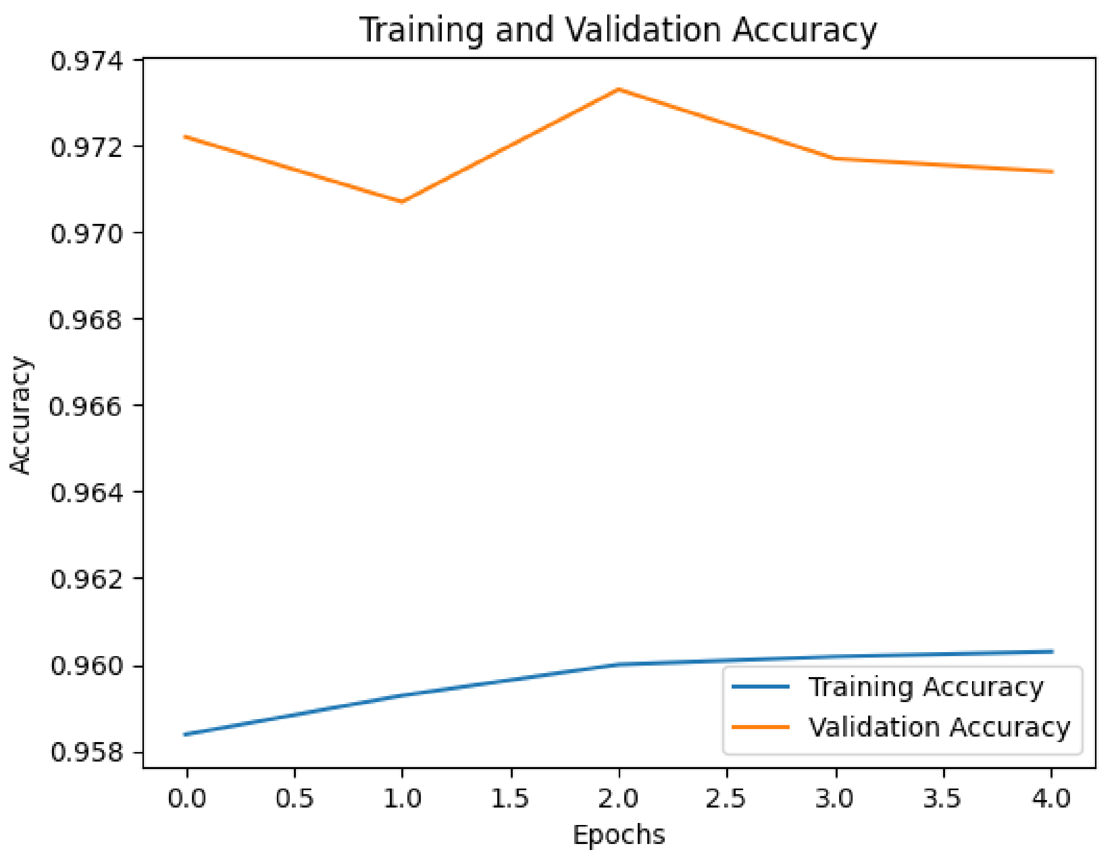
```
Epoch 1/5
1875/1875            15s 8ms/step -
accuracy: 0.9588 - loss: 0.1986 - val_accuracy: 0.9722 - val_loss: 0.1589
Epoch 2/5
1875/1875            11s 3ms/step -
accuracy: 0.9608 - loss: 0.1917 - val_accuracy: 0.9707 - val_loss: 0.1604
Epoch 3/5
```

```
1875/1875                7s 4ms/step -
accuracy: 0.9609 - loss: 0.1902 - val_accuracy: 0.9733 - val_loss: 0.1501
Epoch 4/5
1875/1875                5s 3ms/step -
accuracy: 0.9623 - loss: 0.1889 - val_accuracy: 0.9717 - val_loss: 0.1576
Epoch 5/5
1875/1875                6s 3ms/step -
accuracy: 0.9610 - loss: 0.1904 - val_accuracy: 0.9714 - val_loss: 0.1616
```

[80]: <matplotlib.legend.Legend at 0x7b7a5dd0ff10>



[81]:
```
# Plotting the accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



6 Improving Model

[84]:
```
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.
↪2, random_state=42)

X_train = X_train.reshape(-1, 784)
X_val = X_val.reshape(-1, 784)

# Model Definition with L2 Regularization
model = keras.Sequential([
    keras.layers.Dense(128, input_shape=(784,), activation='relu',␣
↪kernel_regularizer=regularizers.l2(0.001)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation='relu', kernel_regularizer=regularizers.
↪l2(0.001)),
```

```
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation='softmax')
])

# Compile the Model
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Callbacks for Early Stopping and Learning Rate Reduction
early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3,␣
 ↪min_lr=0.00001)

# Train the Model
history_new = model.fit(X_train, y_train, validation_data=(X_val, y_val),
                    epochs=50, batch_size=64, callbacks=[early_stopping,␣
 ↪reduce_lr])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

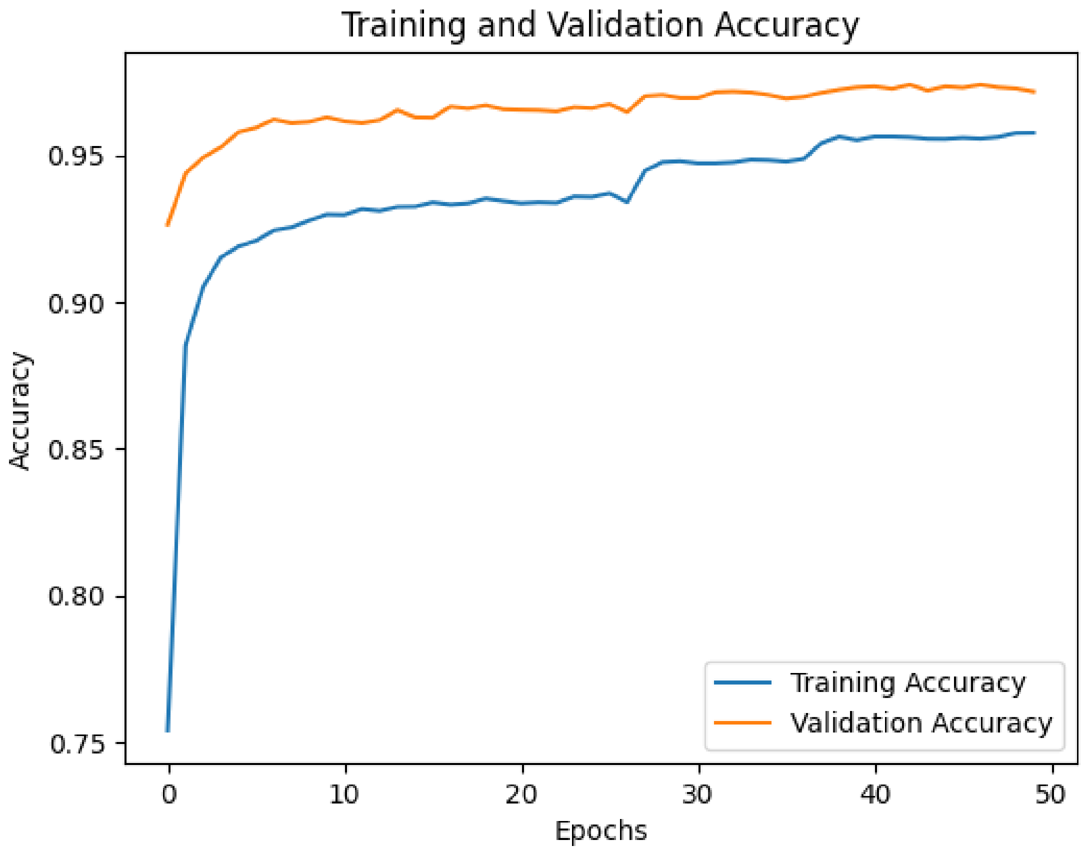
```
Epoch 1/50
600/600              12s 13ms/step -
accuracy: 0.6010 - loss: 1.4049 - val_accuracy: 0.9262 - val_loss: 0.4347 -
learning_rate: 0.0010
Epoch 2/50
600/600              5s 5ms/step -
accuracy: 0.8792 - loss: 0.6010 - val_accuracy: 0.9436 - val_loss: 0.3612 -
learning_rate: 0.0010
Epoch 3/50
600/600              5s 5ms/step -
accuracy: 0.9018 - loss: 0.5071 - val_accuracy: 0.9491 - val_loss: 0.3292 -
learning_rate: 0.0010
Epoch 4/50
600/600              7s 7ms/step -
accuracy: 0.9144 - loss: 0.4538 - val_accuracy: 0.9526 - val_loss: 0.3148 -
learning_rate: 0.0010
Epoch 5/50
600/600              4s 5ms/step -
accuracy: 0.9200 - loss: 0.4332 - val_accuracy: 0.9577 - val_loss: 0.2886 -
learning_rate: 0.0010
Epoch 6/50
600/600              5s 5ms/step -
```

```
accuracy: 0.9197 - loss: 0.4227 - val_accuracy: 0.9593 - val_loss: 0.2807 -
learning_rate: 0.0010
Epoch 7/50
600/600            5s 8ms/step -
accuracy: 0.9247 - loss: 0.3999 - val_accuracy: 0.9621 - val_loss: 0.2655 -
learning_rate: 0.0010
Epoch 8/50
600/600            3s 5ms/step -
accuracy: 0.9280 - loss: 0.3865 - val_accuracy: 0.9608 - val_loss: 0.2635 -
learning_rate: 0.0010
Epoch 9/50
600/600            3s 5ms/step -
accuracy: 0.9272 - loss: 0.3809 - val_accuracy: 0.9613 - val_loss: 0.2613 -
learning_rate: 0.0010
Epoch 10/50
600/600            6s 7ms/step -
accuracy: 0.9306 - loss: 0.3724 - val_accuracy: 0.9628 - val_loss: 0.2549 -
learning_rate: 0.0010
Epoch 11/50
600/600            4s 5ms/step -
accuracy: 0.9330 - loss: 0.3670 - val_accuracy: 0.9615 - val_loss: 0.2590 -
learning_rate: 0.0010
Epoch 12/50
600/600            3s 5ms/step -
accuracy: 0.9320 - loss: 0.3661 - val_accuracy: 0.9608 - val_loss: 0.2587 -
learning_rate: 0.0010
Epoch 13/50
600/600            3s 5ms/step -
accuracy: 0.9308 - loss: 0.3640 - val_accuracy: 0.9619 - val_loss: 0.2545 -
learning_rate: 0.0010
Epoch 14/50
600/600            5s 8ms/step -
accuracy: 0.9328 - loss: 0.3516 - val_accuracy: 0.9652 - val_loss: 0.2438 -
learning_rate: 0.0010
Epoch 15/50
600/600            3s 5ms/step -
accuracy: 0.9339 - loss: 0.3562 - val_accuracy: 0.9628 - val_loss: 0.2504 -
learning_rate: 0.0010
Epoch 16/50
600/600            5s 5ms/step -
accuracy: 0.9351 - loss: 0.3518 - val_accuracy: 0.9627 - val_loss: 0.2473 -
learning_rate: 0.0010
Epoch 17/50
600/600            6s 11ms/step -
accuracy: 0.9340 - loss: 0.3557 - val_accuracy: 0.9665 - val_loss: 0.2409 -
learning_rate: 0.0010
Epoch 18/50
600/600            3s 5ms/step -
```

```
accuracy: 0.9349 - loss: 0.3522 - val_accuracy: 0.9658 - val_loss: 0.2405 -
learning_rate: 0.0010
Epoch 19/50
600/600             5s 5ms/step -
accuracy: 0.9369 - loss: 0.3489 - val_accuracy: 0.9669 - val_loss: 0.2427 -
learning_rate: 0.0010
Epoch 20/50
600/600             7s 7ms/step -
accuracy: 0.9351 - loss: 0.3535 - val_accuracy: 0.9655 - val_loss: 0.2444 -
learning_rate: 0.0010
Epoch 21/50
600/600             3s 5ms/step -
accuracy: 0.9352 - loss: 0.3473 - val_accuracy: 0.9653 - val_loss: 0.2378 -
learning_rate: 0.0010
Epoch 22/50
600/600             3s 5ms/step -
accuracy: 0.9372 - loss: 0.3366 - val_accuracy: 0.9652 - val_loss: 0.2412 -
learning_rate: 0.0010
Epoch 23/50
600/600             7s 7ms/step -
accuracy: 0.9333 - loss: 0.3528 - val_accuracy: 0.9648 - val_loss: 0.2382 -
learning_rate: 0.0010
Epoch 24/50
600/600             3s 5ms/step -
accuracy: 0.9355 - loss: 0.3492 - val_accuracy: 0.9663 - val_loss: 0.2354 -
learning_rate: 0.0010
Epoch 25/50
600/600             5s 5ms/step -
accuracy: 0.9387 - loss: 0.3406 - val_accuracy: 0.9659 - val_loss: 0.2389 -
learning_rate: 0.0010
Epoch 26/50
600/600             7s 8ms/step -
accuracy: 0.9403 - loss: 0.3344 - val_accuracy: 0.9673 - val_loss: 0.2358 -
learning_rate: 0.0010
Epoch 27/50
600/600             3s 5ms/step -
accuracy: 0.9347 - loss: 0.3417 - val_accuracy: 0.9646 - val_loss: 0.2374 -
learning_rate: 0.0010
Epoch 28/50
600/600             5s 5ms/step -
accuracy: 0.9441 - loss: 0.3145 - val_accuracy: 0.9700 - val_loss: 0.2126 -
learning_rate: 5.0000e-04
Epoch 29/50
600/600             6s 7ms/step -
accuracy: 0.9491 - loss: 0.2889 - val_accuracy: 0.9704 - val_loss: 0.2089 -
learning_rate: 5.0000e-04
Epoch 30/50
600/600             4s 5ms/step -
```

```
accuracy: 0.9489 - loss: 0.2879 - val_accuracy: 0.9694 - val_loss: 0.2099 -
learning_rate: 5.0000e-04
Epoch 31/50
600/600            5s 4ms/step -
accuracy: 0.9489 - loss: 0.2807 - val_accuracy: 0.9694 - val_loss: 0.2050 -
learning_rate: 5.0000e-04
Epoch 32/50
600/600            5s 8ms/step -
accuracy: 0.9482 - loss: 0.2764 - val_accuracy: 0.9714 - val_loss: 0.1982 -
learning_rate: 5.0000e-04
Epoch 33/50
600/600            3s 5ms/step -
accuracy: 0.9496 - loss: 0.2739 - val_accuracy: 0.9716 - val_loss: 0.1955 -
learning_rate: 5.0000e-04
Epoch 34/50
600/600            5s 5ms/step -
accuracy: 0.9472 - loss: 0.2809 - val_accuracy: 0.9712 - val_loss: 0.1923 -
learning_rate: 5.0000e-04
Epoch 35/50
600/600            6s 6ms/step -
accuracy: 0.9493 - loss: 0.2694 - val_accuracy: 0.9704 - val_loss: 0.1950 -
learning_rate: 5.0000e-04
Epoch 36/50
600/600            3s 5ms/step -
accuracy: 0.9491 - loss: 0.2736 - val_accuracy: 0.9692 - val_loss: 0.1985 -
learning_rate: 5.0000e-04
Epoch 37/50
600/600            6s 5ms/step -
accuracy: 0.9475 - loss: 0.2767 - val_accuracy: 0.9698 - val_loss: 0.1947 -
learning_rate: 5.0000e-04
Epoch 38/50
600/600            5s 5ms/step -
accuracy: 0.9534 - loss: 0.2558 - val_accuracy: 0.9711 - val_loss: 0.1875 -
learning_rate: 2.5000e-04
Epoch 39/50
600/600            5s 5ms/step -
accuracy: 0.9562 - loss: 0.2463 - val_accuracy: 0.9722 - val_loss: 0.1866 -
learning_rate: 2.5000e-04
Epoch 40/50
600/600            4s 6ms/step -
accuracy: 0.9540 - loss: 0.2465 - val_accuracy: 0.9730 - val_loss: 0.1812 -
learning_rate: 2.5000e-04
Epoch 41/50
600/600            4s 6ms/step -
accuracy: 0.9567 - loss: 0.2397 - val_accuracy: 0.9733 - val_loss: 0.1776 -
learning_rate: 2.5000e-04
Epoch 42/50
600/600            4s 5ms/step -
```

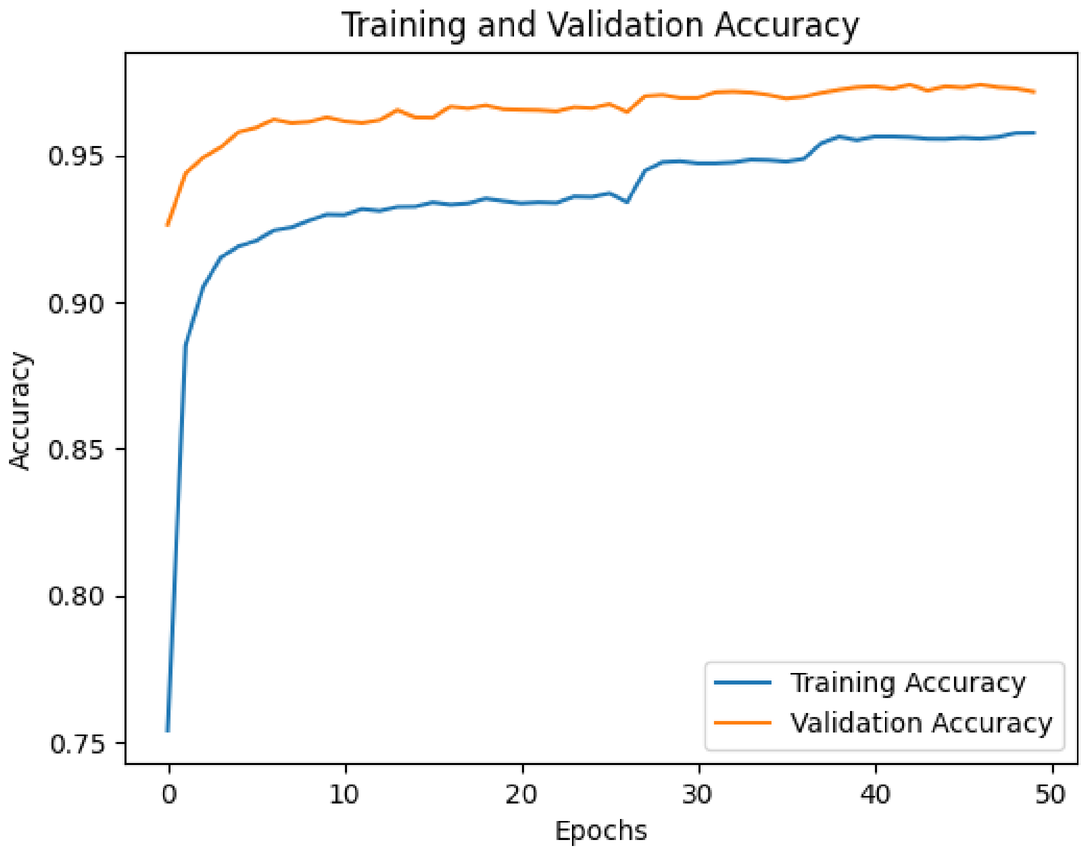
```
accuracy: 0.9568 - loss: 0.2426 - val_accuracy: 0.9725 - val_loss: 0.1771 -
learning_rate: 2.5000e-04
Epoch 43/50
600/600             3s 4ms/step -
accuracy: 0.9565 - loss: 0.2389 - val_accuracy: 0.9740 - val_loss: 0.1758 -
learning_rate: 2.5000e-04
Epoch 44/50
600/600             6s 6ms/step -
accuracy: 0.9558 - loss: 0.2364 - val_accuracy: 0.9719 - val_loss: 0.1774 -
learning_rate: 2.5000e-04
Epoch 45/50
600/600             3s 5ms/step -
accuracy: 0.9549 - loss: 0.2332 - val_accuracy: 0.9733 - val_loss: 0.1741 -
learning_rate: 2.5000e-04
Epoch 46/50
600/600             3s 5ms/step -
accuracy: 0.9568 - loss: 0.2359 - val_accuracy: 0.9729 - val_loss: 0.1773 -
learning_rate: 2.5000e-04
Epoch 47/50
600/600             3s 5ms/step -
accuracy: 0.9564 - loss: 0.2322 - val_accuracy: 0.9740 - val_loss: 0.1752 -
learning_rate: 2.5000e-04
Epoch 48/50
600/600             4s 7ms/step -
accuracy: 0.9567 - loss: 0.2314 - val_accuracy: 0.9730 - val_loss: 0.1737 -
learning_rate: 2.5000e-04
Epoch 49/50
600/600             4s 5ms/step -
accuracy: 0.9579 - loss: 0.2288 - val_accuracy: 0.9726 - val_loss: 0.1743 -
learning_rate: 2.5000e-04
Epoch 50/50
600/600             3s 5ms/step -
accuracy: 0.9583 - loss: 0.2326 - val_accuracy: 0.9716 - val_loss: 0.1781 -
learning_rate: 2.5000e-04
```



7 Second Performance Visualization

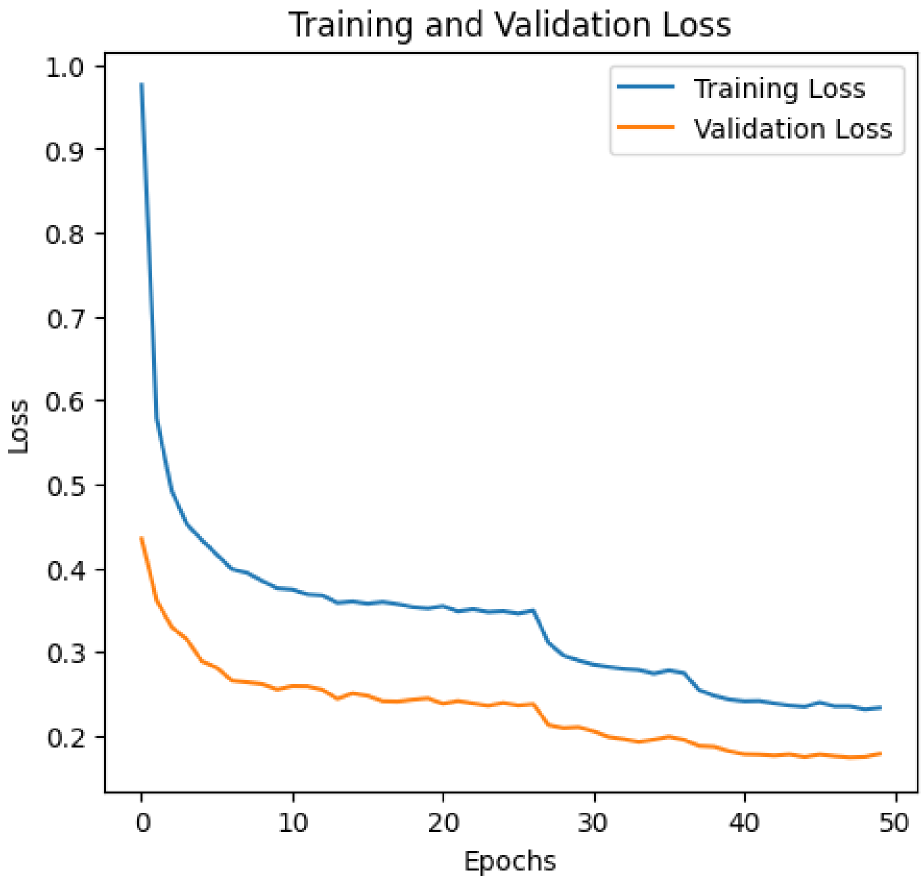
[87]:
```
model.evaluate(X_test_flattened, y_test)

# Plotting the accuracy
plt.plot(history_new.history['accuracy'], label='Training Accuracy')
plt.plot(history_new.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
313/313              1s 3ms/step -
accuracy: 0.9697 - loss: 0.1867
```



[88]:
```
# Plotting the loss and accuracy curves
# Plotting the loss
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history_new.history['loss'], label='Training Loss')
plt.plot(history_new.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
```

[88]:
```
<matplotlib.legend.Legend at 0x7b7a472fbbe0>
```

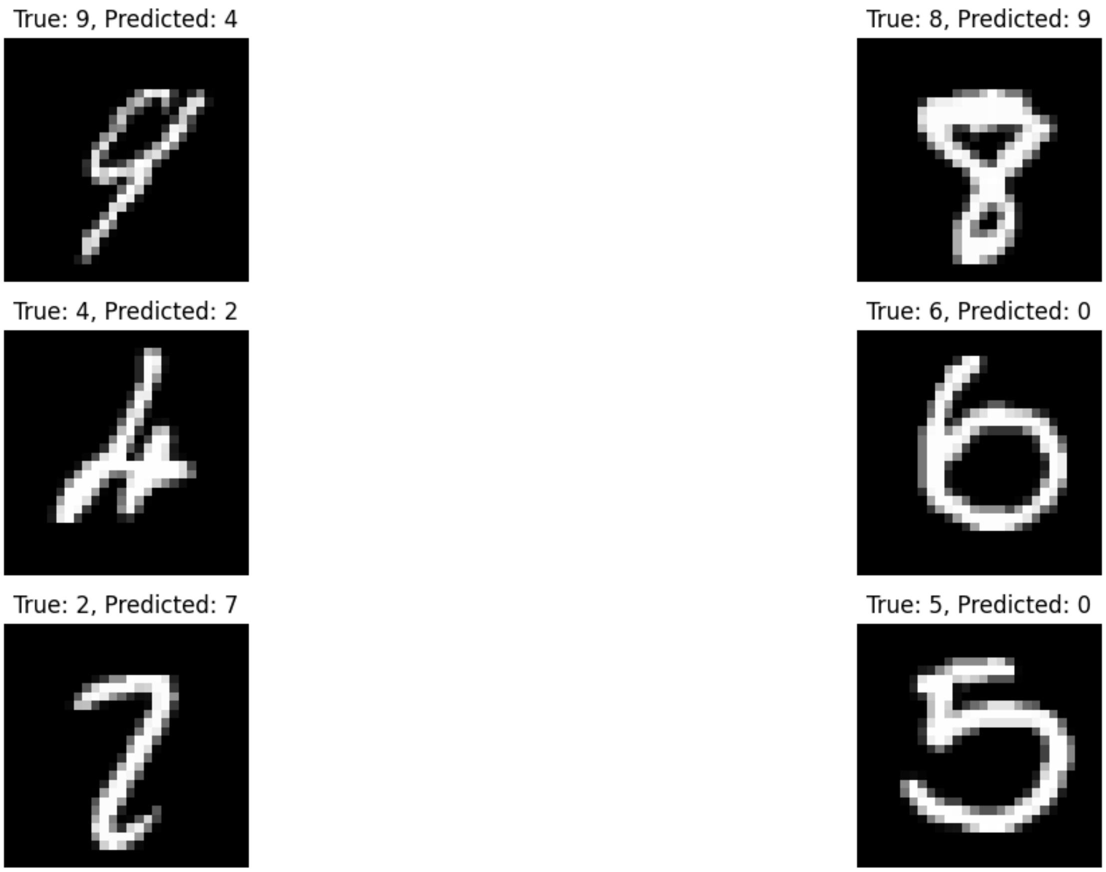


8 Visualize some incorrect predictions

[92]:
```
import matplotlib.pyplot as plt
import numpy as np
y_predicted = model.predict(X_test_flattened)
y_predicted_labels = [np.argmax(i) for i in y_predicted]

# Find incorrect predictions
incorrect_indices = np.where(y_predicted_labels != y_test)[0]

# Display some incorrect predictions
plt.figure(figsize=(15, 8))
for i, incorrect_index in enumerate(incorrect_indices[:6]):
    plt.subplot(3, 2, i + 1)
    plt.imshow(X_test[incorrect_index], cmap='gray')
    plt.title(f"True: {y_test[incorrect_index]}, Predicted:␣
 ↪{y_predicted_labels[incorrect_index]}")
    plt.axis('off')
```

```
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
```




```
[ ]:
```