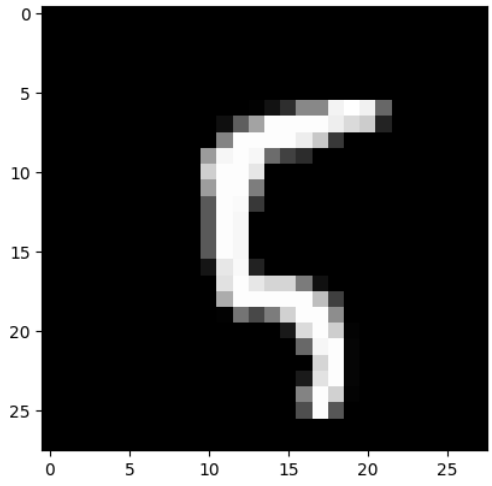
In [ ]:
```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
```

In [ ]:
```
#model / data parameters

num_classes = 10
input_shape = (28,28,1)

(Xtrain, ytrain), (Xtest, ytest) = keras.datasets.mnist.load_data()
```

In [ ]:
```
print(Xtrain.shape)
print(ytrain.shape)
print(Xtest.shape)
print(ytest.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

In [ ]:
```
sample = Xtrain[100]
plt.imshow(sample, cmap='gray')
plt.show()
print(ytrain[100])
```




```
5
```

In [ ]:
```
Xtrain[100]//255
```

Out[ ]:
```
ndarray (28, 28) show data
```

In [ ]:
```
#normalize the images -> values bet 0-1
Xtrain = Xtrain/255
Xtrest = Xtest/255
```

In [ ]:
```
Xtrain[100]
```

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.        , 0.00784314, 0.07058824,
        0.18039216, 0.53333333, 0.53333333, 0.95686275, 1.        ,
        0.94509804, 0.40392157, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.05882353, 0.36862745, 0.63921569, 0.99215686,
        0.99215686, 0.99215686, 0.99215686, 0.93333333, 0.85490196,
        0.8       , 0.1372549 , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.51372549, 0.99215686, 0.99215686, 0.99215686,
        0.99215686, 0.92941176, 0.78431373, 0.22352941, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.60784314, 0.96470588, 0.99215686, 0.96862745, 0.42352941,
        0.25490196, 0.17647059, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.81176471, 0.99215686, 0.99215686, 0.90196078, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.61568627, 0.99215686, 0.99215686, 0.49019608, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.34901961, 0.99215686, 0.98039216, 0.22352941, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.34901961, 0.99215686, 0.96862745, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.34901961, 0.99215686, 0.96862745, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.34901961, 0.99215686, 0.96862745, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.08235294, 0.90588235, 0.97647059, 0.13333333, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.88235294, 0.99215686, 0.90588235, 0.83529412,
        0.83529412, 0.48235294, 0.0627451 , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.6745098 , 0.99215686, 0.99215686, 0.99215686,
        0.99215686, 0.99215686, 0.74509804, 0.24705882, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.00784314, 0.45490196, 0.28235294, 0.48627451,
        0.81960784, 0.99215686, 0.99215686, 0.55294118, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.09803922, 0.85882353, 0.99215686, 0.80784314, 0.01176471,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.40784314, 0.96470588, 0.99215686, 0.01960784,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.83529412, 0.99215686, 0.01960784,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.10196078, 0.88627451, 0.99215686, 0.01960784,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.51764706, 0.99215686, 0.81960784, 0.01176471,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.30588235, 0.99215686, 0.3372549 , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 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 [ ]:
```
#reshape the images -> shape = (28,28,1)
Xtrain = np.expand_dims(Xtrain, -1)
Xtest = np.expand_dims(Xtest, -1)

print(Xtrain.shape)
print(Xtest.shape)

#0th index = no of samples
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

In [ ]:
```
ytrain
```

Out[ ]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

In [ ]:
```
# #convert class vectors to binary class metrices
# ytrain = keras.utils.to_categorical(ytrain, num_classes)
# ytest = keras.utils.to_categorical(ytest, num_classes)
```

In [ ]:
```
model = keras.Sequential(
    [
      keras.Input(shape=input_shape),
      layers.Conv2D(32, kernel_size=(3,3), activation='relu'),
      layers.MaxPooling2D(pool_size=(2,2)), #down sample
      layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
      layers.MaxPooling2D(pool_size=(2,2)),

      layers.Flatten(),
      layers.Dropout(0.4),
      #Layers.Dense(15, activation='relu'),#add more dense layers
      #Layers.Dropout(0.4),#trying using dropout
      layers.Dense(num_classes, activation='softmax')
    ]
)
#28-3+1
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 26, 26, 32)        320

 max_pooling2d_10 (MaxPooli  (None, 13, 13, 32)        0
 ng2D)

 conv2d_11 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_11 (MaxPooli  (None, 5, 5, 64)          0
 ng2D)

 flatten_5 (Flatten)         (None, 1600)              0

 dropout_5 (Dropout)         (None, 1600)              0

 dense_5 (Dense)             (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
batch_size = 128
epochs = 15
```

In [ ]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])#multiclass - loss= categorical, else binary or sparse cc
```

In [ ]:
```
#model training
hist = model.fit(Xtrain, ytrain, batch_size=batch_size, epochs = epochs, validation_split=0.1)
```

```
Epoch 1/15
422/422 [==============================] - 4s 6ms/step - loss: 0.3467 - accuracy: 0.8935 - val_loss: 0.0808 - val_accuracy: 0.9783
Epoch 2/15
422/422 [==============================] - 2s 4ms/step - loss: 0.1055 - accuracy: 0.9672 - val_loss: 0.0567 - val_accuracy: 0.9838
Epoch 3/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0816 - accuracy: 0.9748 - val_loss: 0.0496 - val_accuracy: 0.9860
Epoch 4/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0671 - accuracy: 0.9794 - val_loss: 0.0462 - val_accuracy: 0.9875
Epoch 5/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0604 - accuracy: 0.9817 - val_loss: 0.0380 - val_accuracy: 0.9892
Epoch 6/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0509 - accuracy: 0.9840 - val_loss: 0.0364 - val_accuracy: 0.9893
Epoch 7/15
422/422 [==============================] - 2s 5ms/step - loss: 0.0483 - accuracy: 0.9846 - val_loss: 0.0361 - val_accuracy: 0.9888
Epoch 8/15
422/422 [==============================] - 2s 5ms/step - loss: 0.0435 - accuracy: 0.9862 - val_loss: 0.0350 - val_accuracy: 0.9905
Epoch 9/15
422/422 [==============================] - 3s 6ms/step - loss: 0.0406 - accuracy: 0.9869 - val_loss: 0.0337 - val_accuracy: 0.9905
Epoch 10/15
422/422 [==============================] - 2s 6ms/step - loss: 0.0370 - accuracy: 0.9877 - val_loss: 0.0309 - val_accuracy: 0.9907
Epoch 11/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0358 - accuracy: 0.9880 - val_loss: 0.0306 - val_accuracy: 0.9908
Epoch 12/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0337 - accuracy: 0.9891 - val_loss: 0.0337 - val_accuracy: 0.9905
Epoch 13/15
422/422 [==============================] - 2s 5ms/step - loss: 0.0314 - accuracy: 0.9898 - val_loss: 0.0292 - val_accuracy: 0.9925
Epoch 14/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0288 - accuracy: 0.9909 - val_loss: 0.0303 - val_accuracy: 0.9923
Epoch 15/15
422/422 [==============================] - 2s 4ms/step - loss: 0.0280 - accuracy: 0.9909 - val_loss: 0.0306 - val_accuracy: 0.9910
```

In [ ]:
```
print(hist.history)
```


```
{'loss': [0.34670016169548035, 0.10547356307506561, 0.08160374313592911, 0.0670543685555458, 0.060424238443374634, 0.05093512311577797, 0.04825188964
6053314, 0.04346991702914238, 0.04064791649580002, 0.0369662307202816, 0.03583152964711189, 0.033714693039655685, 0.031443625688552856, 0.02878768555
8199883, 0.027996234595775604], 'accuracy': [0.8935370445251465, 0.9671666622161865, 0.9747777581214905, 0.9794074296951294, 0.9817036986351013, 0.98
40185046195984, 0.9845555424690247, 0.9862037301063538, 0.9868888854980469, 0.9877036809921265, 0.9880370497703552, 0.9891481399536133, 0.98979628086
09009, 0.9900974306488037, 0.9909444451332092], 'val_loss': [0.08079592138528824, 0.05668488144874573, 0.04962928220629692, 0.046180836856365204, 0.0
38036007434129715, 0.03636574745178223, 0.03606457635760307, 0.034988511353731155, 0.033744510263204575, 0.03085474856197834, 0.030576961115002632,
0.033670615404844284, 0.029184773564338684, 0.03034009039402008, 0.03060285933315754], 'val_accuracy': [0.9783333539962769, 0.9838333129882812, 0.986
0000014305115, 0.987500011920929, 0.9891666769981384, 0.9893333315849304, 0.9888333082199097, 0.9904999732971191, 0.9904999732971191, 0.9906666874885
559, 0.9908333420753479, 0.9904999732971191, 0.9925000071525574, 0.9923333525657654, 0.9909999966621399]}
```


In [ ]:
```
score = model.evaluate(Xtest, ytest, verbose=0)
print("test loss: ", score[0])
print("test accuracy: ", score[1])
```

```
test loss:  8.114225387573242
test accuracy:  0.9858999848365784
```

In [ ]:
```
y_pred = model.predict(Xtest)
```

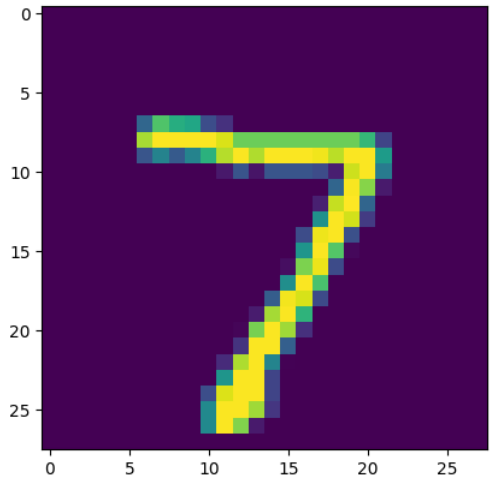
```
313/313 [==============================] - 1s 2ms/step
```

In [ ]:
```
y_pred
```

Out[ ]:
```
array([[0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [ ]:
```
y_pred_classes = np.argmax(y_pred, axis=1)
```

In [ ]:
```
plt.imshow(Xtest[0])
plt.show()
```



In [ ]:
```
y_pred_classes
```

Out[ ]:
```
array([7, 2, 1, ..., 4, 5, 6])
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred_classes)
cm
```

Out[ ]:
```
array([[ 977,    0,    0,    0,    0,    0,    0,    1,    2,    0],
       [   0, 1123,    3,    1,    0,    0,    4,    0,    4,    0],
       [   1,    0, 1027,    0,    0,    0,    1,    1,    2,    0],
       [   0,    0,    6,  994,    0,    3,    0,    0,    6,    1],
       [   0,    0,    1,    0,  977,    0,    1,    0,    3,    0],
       [   1,    0,    0,    3,    0,  878,    3,    0,    7,    0],
       [   1,    1,    1,    0,    1,    1,  948,    0,    5,    0],
       [   0,    2,   20,    0,    6,    0,    0,  988,    9,    3],
       [   4,    0,    0,    0,    0,    0,    0,    0,  970,    0],
       [   2,    0,    0,    0,   14,    2,    0,    1,   13,  977]])
```

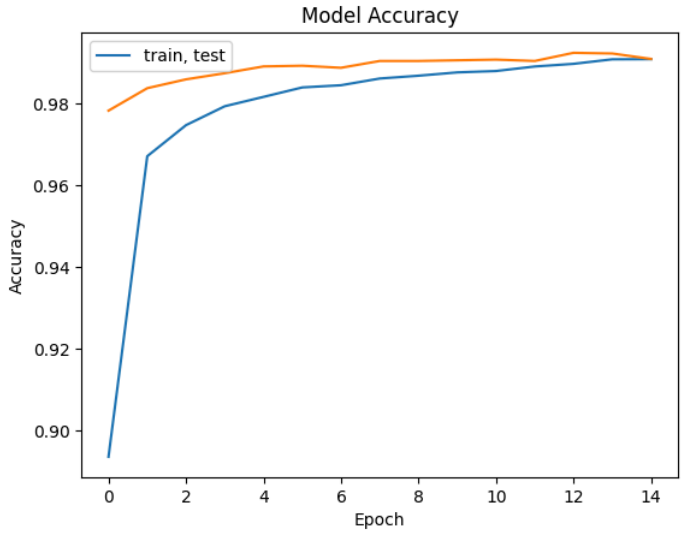
In [ ]:
```
from sklearn.metrics import classification_report
cr = classification_report(ytest, y_pred_classes)
print(cr)
```

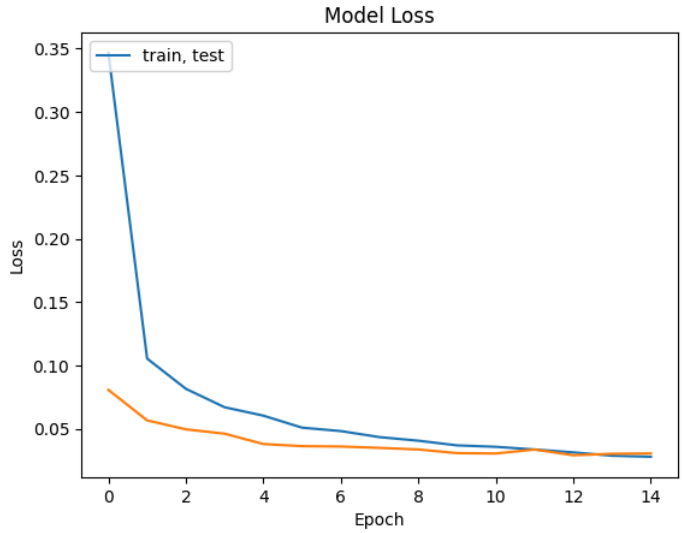
```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       1.00      0.99      0.99      1135
           2       0.97      1.00      0.98      1032
           3       1.00      0.98      0.99      1010
           4       0.98      0.99      0.99       982
           5       0.99      0.98      0.99       892
           6       0.99      0.99      0.99       958
           7       1.00      0.96      0.98      1028
           8       0.95      1.00      0.97       974
           9       1.00      0.97      0.98      1009

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

In [ ]:
```
model.save('mnist_t1.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [ ]:
```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title("Model Accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(['train, test'], loc='upper left')
plt.show()

#training
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("Model Loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend(['train, test'], loc='upper left')
plt.show()
```





In [ ]:
```
import tensorflow as tf
mn_model = tf.keras.models.load_model("/content/mnist_t1.h5")
```

In [ ]:
```
import cv2 as cv
```

In [ ]:
```
image = cv.imread("/content/1.jpg")
```

In [ ]:
```
image.shape
```

Out[ ]:
```
(170, 170, 3)
```

In [ ]:
```
gray_image=cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

In [ ]:
```
gray_image.shape
```

Out[ ]:
```
(170, 170)
```

In [ ]:
```
img = cv.resize(gray_image, (28,28))
```

In [ ]:
```
img = np.expand_dims(img, -1)
img = np.expand_dims(img, 0)
```

In [ ]:
```
img.shape
```

Out[ ]:
```
(1, 28, 28, 1)
```

In [ ]:
```
pred = (mn_model.predict(img))
```

```
1/1 [==============================] - 0s 40ms/step
```

In [ ]:
```
print(np.argmax(pred, axis=1))
```

```
[8]
```