```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten


(xtrain,ytrain),(xtest,ytest)=keras.datasets.mnist.load_data()


xtrain[0].shape
```

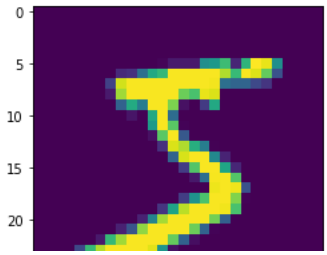
```
(28, 28)
```

```
xtrain[0]
```

```
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,  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)
```

```
import matplotlib.pyplot as plt
plt.imshow(xtrain[0])
```

```
<matplotlib.image.AxesImage at 0x7f6e016a8070>
```




```
xtest[0]
```

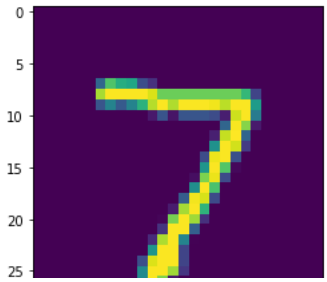
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,  67, 114,  72, 114, 163, 227, 254,
        225, 254, 254, 254, 250, 229, 254, 254, 140,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  17,  66,
         14,  67,  67,  67,  59,  21, 236, 254, 106,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,  83, 253, 209,  18,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,  22, 233, 255,  83,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 129, 254, 238,  44,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,  59, 249, 254,  62,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0, 133, 254, 187,   5,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   9, 205, 248,  58,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0, 126, 254, 182,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  75, 251, 240,  57,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         19, 221, 254, 166,   0,   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,
        203, 254, 219,  35,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  38,
        254, 254,  77,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  31, 224,
        254, 115,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 133, 254,
        254,  52,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  61, 242, 254,
        254,  52,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 121, 254, 254,
        219,  40,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 121, 254, 207,
         18,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   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)
```

```
ytest
```

```
array([7, 2, 1, ..., 4, 5, 6], dtype=uint8)
```

```
plt.imshow(xtest[0])
```

```
<matplotlib.image.AxesImage at 0x7f6dfe56d9d0>
```



```
xtrain=xtrain/255
xtest=xtest/255
```

```
xtrain.shape
```

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

```
ytrain
```

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

```
#model building
model=Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128,activation="relu"))
model.add(Dense(32,activation="relu"))
#add ouput layer.
model.add(Dense(10,activation="softmax"))

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

history=model.fit(xtrain,ytrain,epochs=25,validation_split=0.20)
```

```
Epoch 1/25
1500/1500 [==============================] - 5s 3ms/step - loss: 0.2845 - accuracy: 0.9172 - val_loss: 0.1535 - val_accuracy: 0.9549
Epoch 2/25
1500/1500 [==============================] - 5s 3ms/step - loss: 0.1182 - accuracy: 0.9641 - val_loss: 0.1284 - val_accuracy: 0.9627
Epoch 3/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0831 - accuracy: 0.9748 - val_loss: 0.0998 - val_accuracy: 0.9713
Epoch 4/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0614 - accuracy: 0.9819 - val_loss: 0.1008 - val_accuracy: 0.9711
Epoch 5/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0493 - accuracy: 0.9841 - val_loss: 0.0975 - val_accuracy: 0.9734
Epoch 6/25
1500/1500 [==============================] - 4s 2ms/step - loss: 0.0380 - accuracy: 0.9881 - val_loss: 0.1167 - val_accuracy: 0.9693
Epoch 7/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0303 - accuracy: 0.9906 - val_loss: 0.1088 - val_accuracy: 0.9733
Epoch 8/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0274 - accuracy: 0.9912 - val_loss: 0.1030 - val_accuracy: 0.9741
Epoch 9/25
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0216 - accuracy: 0.9929 - val_loss: 0.1104 - val_accuracy: 0.9742
Epoch 10/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0210 - accuracy: 0.9928 - val_loss: 0.1145 - val_accuracy: 0.9711
Epoch 11/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0169 - accuracy: 0.9943 - val_loss: 0.1033 - val_accuracy: 0.9752
Epoch 12/25
1500/1500 [==============================] - 4s 2ms/step - loss: 0.0135 - accuracy: 0.9955 - val_loss: 0.1184 - val_accuracy: 0.9748
Epoch 13/25
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0176 - accuracy: 0.9941 - val_loss: 0.1107 - val_accuracy: 0.9763
Epoch 14/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0124 - accuracy: 0.9958 - val_loss: 0.1197 - val_accuracy: 0.9766
Epoch 15/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0125 - accuracy: 0.9956 - val_loss: 0.1531 - val_accuracy: 0.9709
Epoch 16/25
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0097 - accuracy: 0.9969 - val_loss: 0.1277 - val_accuracy: 0.9762
Epoch 17/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0112 - accuracy: 0.9961 - val_loss: 0.1515 - val_accuracy: 0.9704
Epoch 18/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0114 - accuracy: 0.9962 - val_loss: 0.1316 - val_accuracy: 0.9764
Epoch 19/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0102 - accuracy: 0.9969 - val_loss: 0.1478 - val_accuracy: 0.9732
Epoch 20/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0081 - accuracy: 0.9972 - val_loss: 0.1382 - val_accuracy: 0.9775
Epoch 21/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0095 - accuracy: 0.9970 - val_loss: 0.1486 - val_accuracy: 0.9750
```

```
Epoch 22/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0088 - accuracy: 0.9974 - val_loss: 0.1547 - val_accuracy: 0.9741
Epoch 23/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0096 - accuracy: 0.9969 - val_loss: 0.1426 - val_accuracy: 0.9773
Epoch 24/25
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0036 - accuracy: 0.9991 - val_loss: 0.1672 - val_accuracy: 0.9750
Epoch 25/25
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0119 - accuracy: 0.9961 - val_loss: 0.1515 - val_accuracy: 0.9756
```

```
xtest[0]
```

```
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.        , 0.32941176, 0.7254902 , 0.62352941, 0.59215686,
        0.23529412, 0.14117647, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.87058824, 0.99607843, 0.99607843, 0.99607843,
        0.99607843, 0.94509804, 0.77647059, 0.77647059, 0.77647059,
        0.77647059, 0.77647059, 0.77647059, 0.77647059, 0.77647059,
        0.66666667, 0.20392157, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.2627451 , 0.44705882, 0.28235294, 0.44705882,
        0.63921569, 0.89019608, 0.99607843, 0.88235294, 0.99607843,
        0.99607843, 0.99607843, 0.98039216, 0.89803922, 0.99607843,
```

```
yprob=model.predict(xtest)
yprob[0]
```

```
313/313 [==============================] - 1s 2ms/step
array([8.3662664e-14, 1.6219028e-14, 1.1767820e-11, 3.0221433e-06,
       5.2607122e-25, 4.9262149e-15, 4.3939224e-21, 9.9999696e-01,
       1.2495416e-12, 2.6428618e-10], dtype=float32)
```

```
ypred=yprob.argmax(axis=1)
ypred
```

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

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

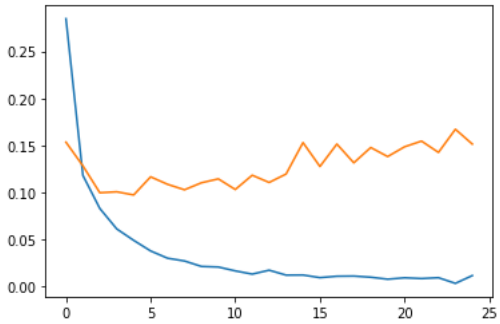
```
              precision    recall  f1-score   support

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

```
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
```

```
[<matplotlib.lines.Line2D at 0x7f6dcdb8fca0>]
```



✓ 0s completed at 3:22 PM