Import the Libraries

```
import numpy as np
import pandas as pd
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Flatten
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Loading the File

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
xtrain.shape
```

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

```
xtest.shape
```

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

```
xtrain.ndim
```

```
3
```

```
xtrain[1]
```

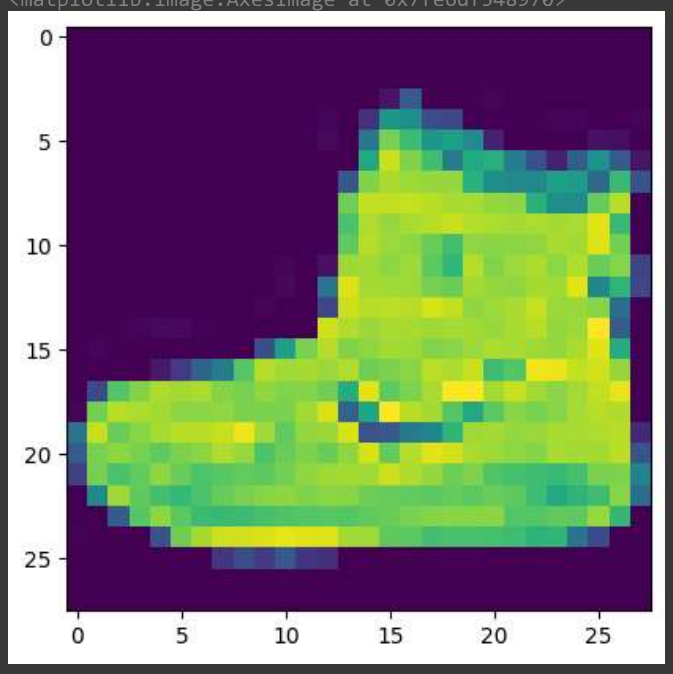
```
          0,   0],
       [  0,   0,   0, 164, 235, 214, 211, 220, 216, 201,  52,  71,  89,
         94,  83,  78,  70,  76,  92,  87, 206, 207, 222, 213, 219, 208,
          0,   0],
       [  0,   0,   0, 106, 187, 223, 237, 248, 211, 198, 252, 250, 248,
        245, 248, 252, 253, 250, 252, 239, 201, 212, 225, 215, 193, 113,
          0,   0],
       [  0,   0,   0,   0,   0,  17,  54, 159, 222, 193, 208, 192, 197,
        200, 200, 200, 200, 201, 203, 195, 210, 165,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  47, 225, 192, 214, 203, 206,
        204, 204, 205, 206, 204, 212, 197, 218, 107,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   1,   6,   0,  46, 212, 195, 212, 202, 206,
        205, 204, 205, 206, 204, 212, 200, 218,  91,   0,   3,   1,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,  11, 197, 199, 205, 202, 205,
        206, 204, 205, 207, 204, 205, 205, 218,  77,   0,   5,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   3,   0,   2, 191, 198, 201, 205, 206,
        205, 205, 206, 209, 206, 199, 209, 219,  74,   0,   5,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   2,   0,   0, 188, 197, 200, 207, 207,
        204, 207, 207, 210, 208, 198, 207, 221,  72,   0,   4,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   2,   0,   0, 215, 198, 203, 206, 208,
        205, 207, 207, 210, 208, 200, 202, 222,  75,   0,   4,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 212, 198, 209, 206, 209,
        206, 208, 207, 211, 206, 205, 198, 221,  80,   0,   3,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 204, 201, 205, 208, 207,
        205, 211, 205, 210, 210, 209, 195, 221,  96,   0,   3,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 202, 201, 205, 209, 207,
        205, 213, 206, 210, 209, 210, 194, 217, 105,   0,   2,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 204, 204, 205, 208, 207,
        205, 215, 207, 210, 208, 211, 193, 213, 115,   0,   2,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0, 204, 207, 207, 208, 206,
        206, 215, 210, 210, 207, 212, 195, 210, 118,   0,   2,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 198, 208, 208, 208, 204,
        207, 212, 212, 210, 207, 211, 196, 207, 121,   0,   1,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 198, 210, 207, 208, 206,
        209, 213, 212, 211, 207, 210, 197, 207, 124,   0,   1,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0, 172, 210, 203, 201, 199,
        204, 207, 205, 204, 201, 205, 197, 206, 127,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0, 188, 221, 214, 234, 236,
        238, 244, 244, 244, 240, 243, 214, 224, 162,   0,   2,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   1,   0,   0, 139, 146, 130, 135, 135,
        137, 125, 124, 125, 121, 119, 114, 130,  76,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

```
xtrain[1].shape
```

```
(28, 28)
```

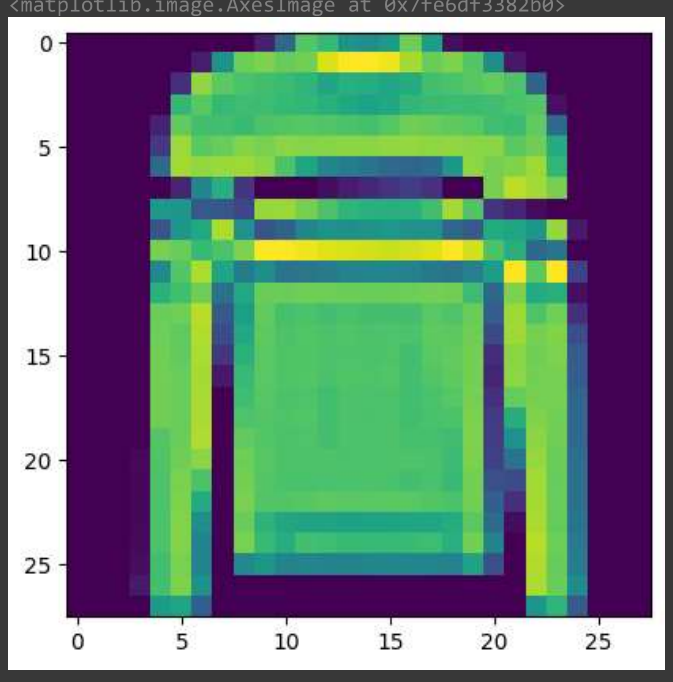
```
# plooting the first image from the xtrain
plt.imshow(xtrain[0])
```

```
<matplotlib.image.AxesImage at 0x7fe6df548970>
```




```
plt.imshow(xtrain[5])
```

```
<matplotlib.image.AxesImage at 0x7fe6df3382b0>
```




```
# Scale the 'X' features
xtrain = xtrain/255
xtest = xtest/255
```

```
xtrain
```

```
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.]]])
```

```
xtrain[0]
```

```
 0.80392157, 0.8627451 , 0.94117647, 0.31372549, 0.58823529,
 1.        , 0.89803922, 0.86666667, 0.7372549 , 0.60392157,
 0.74901961, 0.82352941, 0.8       , 0.81960784, 0.87058824,
 0.89411765, 0.88235294, 0.        ],
[0.38431373, 0.91372549, 0.77647059, 0.82352941, 0.87058824,
 0.89803922, 0.89803922, 0.91764706, 0.97647059, 0.8627451 ,
 0.76078431, 0.84313725, 0.85098039, 0.94509804, 0.25490196,
 0.28627451, 0.41568627, 0.45882353, 0.65882353, 0.85882353,
 0.86666667, 0.84313725, 0.85098039, 0.8745098 , 0.8745098 ,
 0.87843137, 0.89803922, 0.11372549],
[0.29411765, 0.8       , 0.83137255, 0.8       , 0.75686275,
 0.80392157, 0.82745098, 0.88235294, 0.84705882, 0.7254902 ,
 0.77254902, 0.80784314, 0.77647059, 0.83529412, 0.94117647,
 0.76470588, 0.89019608, 0.96078431, 0.9372549 , 0.8745098 ,
 0.85490196, 0.83137255, 0.81960784, 0.87058824, 0.8627451 ,
 0.86666667, 0.90196078, 0.2627451 ],
[0.18823529, 0.79607843, 0.71764706, 0.76078431, 0.83529412,
 0.77254902, 0.7254902 , 0.74509804, 0.76078431, 0.75294118,
 0.79215686, 0.83921569, 0.85882353, 0.86666667, 0.8627451 ,
 0.9254902 , 0.88235294, 0.84705882, 0.78039216, 0.80784314,
 0.72941176, 0.70980392, 0.69411765, 0.6745098 , 0.70980392,
 0.80392157, 0.80784314, 0.45098039],
[0.        , 0.47843137, 0.85882353, 0.75686275, 0.70196078,
 0.67058824, 0.71764706, 0.76862745, 0.8       , 0.82352941,
 0.83529412, 0.81176471, 0.82745098, 0.82352941, 0.78431373,
 0.76862745, 0.76078431, 0.74901961, 0.76470588, 0.74901961,
 0.77647059, 0.75294118, 0.69019608, 0.61176471, 0.65490196,
 0.69411765, 0.82352941, 0.36078431],
[0.        , 0.        , 0.29019608, 0.74117647, 0.83137255,
 0.74901961, 0.68627451, 0.6745098 , 0.68627451, 0.70980392,
 0.7254902 , 0.7372549 , 0.74117647, 0.7372549 , 0.75686275,
 0.77647059, 0.8       , 0.81960784, 0.82352941, 0.82352941,
 0.82745098, 0.7372549 , 0.7372549 , 0.76078431, 0.75294118,
 0.84705882, 0.66666667, 0.        ],
[0.00784314, 0.        , 0.        , 0.        , 0.25882353,
 0.78431373, 0.87058824, 0.92941176, 0.9372549 , 0.94901961,
 0.96470588, 0.95294118, 0.95686275, 0.86666667, 0.8627451 ,
 0.75686275, 0.74901961, 0.70196078, 0.71372549, 0.71372549,
 0.70980392, 0.69019608, 0.65098039, 0.65882353, 0.38823529,
 0.22745098, 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.15686275, 0.23921569, 0.17254902,
 0.28235294, 0.16078431, 0.1372549 , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ]])
```

```
xtrain.shape
```

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

```
xtest.shape
```

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

```
#### Model Building

model = Sequential()

# adding Flatten for the input data with the shape of 28 by 28
# 28 rows and 28 columns
model.add(Flatten(input_shape = (28,28)))

# Adding 1st Hidden layer with 128 neurons with activation function called 'relu
model.add(Dense(units = 128, activation = 'relu'))

# Adding 2nd Hidden Layer with 32 neurons and with activation function of 'relu
model.add(Dense(units = 32, activation = 'relu'))

# Adding output Layer
model.add(Dense(units = 10, activation = 'softmax'))

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

history = model.fit(xtrain,ytrain, epochs = 25, validation_split = 0.2)
```

```
Epoch 1/25
1500/1500 [==============================] - 9s 5ms/step - loss: 0.5398 - accuracy: 0.8117 - val_loss: 0.4013 - val_accuracy: 0.8610
Epoch 2/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.3878 - accuracy: 0.8599 - val_loss: 0.3771 - val_accuracy: 0.8666
Epoch 3/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.3469 - accuracy: 0.8739 - val_loss: 0.3642 - val_accuracy: 0.8702
Epoch 4/25
1500/1500 [==============================] - 7s 4ms/step - loss: 0.3231 - accuracy: 0.8810 - val_loss: 0.3333 - val_accuracy: 0.8777
Epoch 5/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.3016 - accuracy: 0.8875 - val_loss: 0.3463 - val_accuracy: 0.8781
Epoch 6/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.2889 - accuracy: 0.8916 - val_loss: 0.3343 - val_accuracy: 0.8806
Epoch 7/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.2747 - accuracy: 0.8973 - val_loss: 0.3259 - val_accuracy: 0.8813
Epoch 8/25
1500/1500 [==============================] - 11s 7ms/step - loss: 0.2639 - accuracy: 0.9009 - val_loss: 0.3491 - val_accuracy: 0.8795
Epoch 9/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.2531 - accuracy: 0.9053 - val_loss: 0.3293 - val_accuracy: 0.8829
Epoch 10/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.2454 - accuracy: 0.9076 - val_loss: 0.3470 - val_accuracy: 0.8809
Epoch 11/25
1500/1500 [==============================] - 9s 6ms/step - loss: 0.2371 - accuracy: 0.9117 - val_loss: 0.3444 - val_accuracy: 0.8777
Epoch 12/25
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2293 - accuracy: 0.9138 - val_loss: 0.3177 - val_accuracy: 0.8898
Epoch 13/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.2218 - accuracy: 0.9162 - val_loss: 0.3180 - val_accuracy: 0.8867
```

```
Epoch 14/25
1500/1500 [==============================] - 7s 4ms/step - loss: 0.2154 - accuracy: 0.9187 - val_loss: 0.3266 - val_accuracy: 0.8878
Epoch 15/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.2091 - accuracy: 0.9209 - val_loss: 0.3233 - val_accuracy: 0.8919
Epoch 16/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.2033 - accuracy: 0.9226 - val_loss: 0.3225 - val_accuracy: 0.8870
Epoch 17/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1962 - accuracy: 0.9267 - val_loss: 0.3346 - val_accuracy: 0.8848
Epoch 18/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.1925 - accuracy: 0.9276 - val_loss: 0.3341 - val_accuracy: 0.8920
Epoch 19/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1841 - accuracy: 0.9297 - val_loss: 0.3585 - val_accuracy: 0.8888
Epoch 20/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.1819 - accuracy: 0.9303 - val_loss: 0.3516 - val_accuracy: 0.8913
Epoch 21/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1756 - accuracy: 0.9335 - val_loss: 0.3575 - val_accuracy: 0.8903
Epoch 22/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1706 - accuracy: 0.9349 - val_loss: 0.3559 - val_accuracy: 0.8888
Epoch 23/25
1500/1500 [==============================] - 7s 4ms/step - loss: 0.1672 - accuracy: 0.9361 - val_loss: 0.3685 - val_accuracy: 0.8876
Epoch 24/25
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1614 - accuracy: 0.9379 - val_loss: 0.3792 - val_accuracy: 0.8855
Epoch 25/25
1500/1500 [==============================] - 8s 5ms/step - loss: 0.1592 - accuracy: 0.9391 - val_loss: 0.3714 - val_accuracy: 0.8935
```

```
model.history.history
```

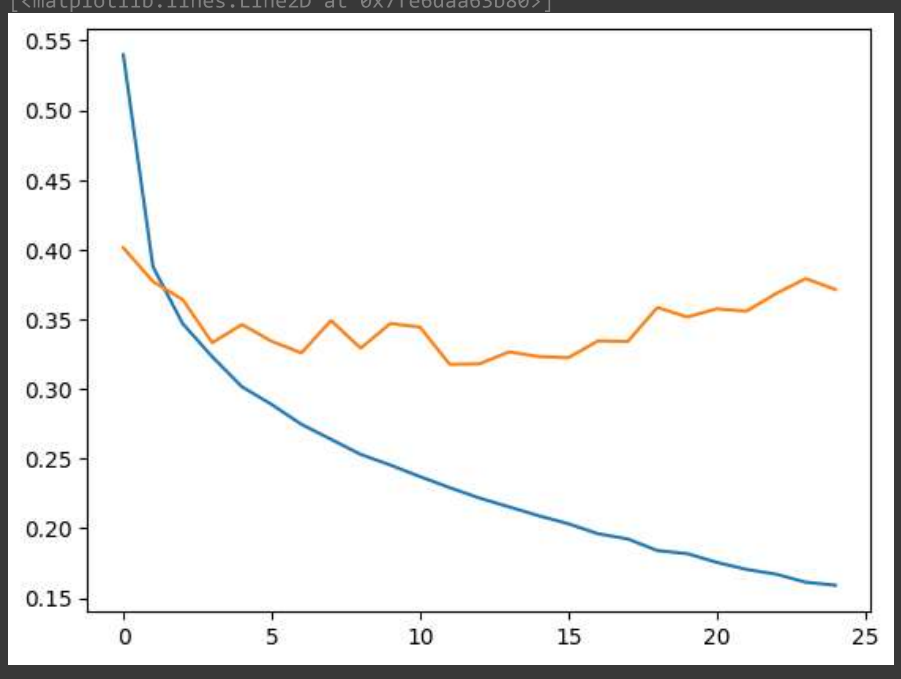

```
 0.9276041388511658,
 0.929729163646698,
 0.9303333163261414,
 0.9335208535194397,
 0.9348541498184204,
 0.9361458420753479,
 0.9378541707992554,
 0.9390833377838135],
'val_loss': [0.4013245105743408,
 0.37713655829429626,
 0.3641831576824188,
 0.3332516551017761,
 0.34627336263656616,
 0.3342668116092682,
 0.32586216926574707,
 0.34914299845695496,
 0.3293168842792511,
 0.3470301628112793,
 0.34441205859184265,
 0.3177095651626587,
 0.3180163502693176,
 0.326634019613266,
 0.32327428460121155,
 0.3224915564060211,
 0.3345533609390259,
 0.3340749740600586,
 0.35849544405937195,
 0.35164740681648254,
 0.35752108693122864,
 0.3558826148509979,
 0.36850476264953613,
 0.37921902537345886,
 0.3713984489440918],
'val_accuracy': [0.8610000014305115,
 0.8665833473205566,
 0.8702499866485596,
 0.8777499794960022,
 0.878083348274231,
 0.8805833458900452,
 0.8813333511352539,
 0.8794999718666077,
 0.8829166889190674,
 0.8809166550636292,
 0.8776666522026062,
 0.8897500038146973,
 0.8867499828338623,
 0.8878333568572998,
 0.8919166922569275,
 0.8870000243186951,
 0.8848333358764648,
 0.8920000195503235,
 0.8888333439826965,
 0.8913333415985107,
 0.890250027179718,
 0.8887500166893005,
 0.887583315372467,
 0.8855000138282776,
 0.89349997043609621}
```


```
pd.DataFrame(model.history.history)
```

	loss	accuracy	val_loss	val_accuracy
0	0.539779	0.811729	0.401325	0.861000
1	0.387770	0.859896	0.377137	0.866583
2	0.346916	0.873937	0.364183	0.870250
3	0.323076	0.881042	0.333252	0.877750
4	0.301581	0.887500	0.346273	0.878083
5	0.288931	0.891646	0.334267	0.880583
6	0.274695	0.897313	0.325862	0.881333
7	0.263917	0.900875	0.349143	0.879500
8	0.253092	0.905312	0.329317	0.882917

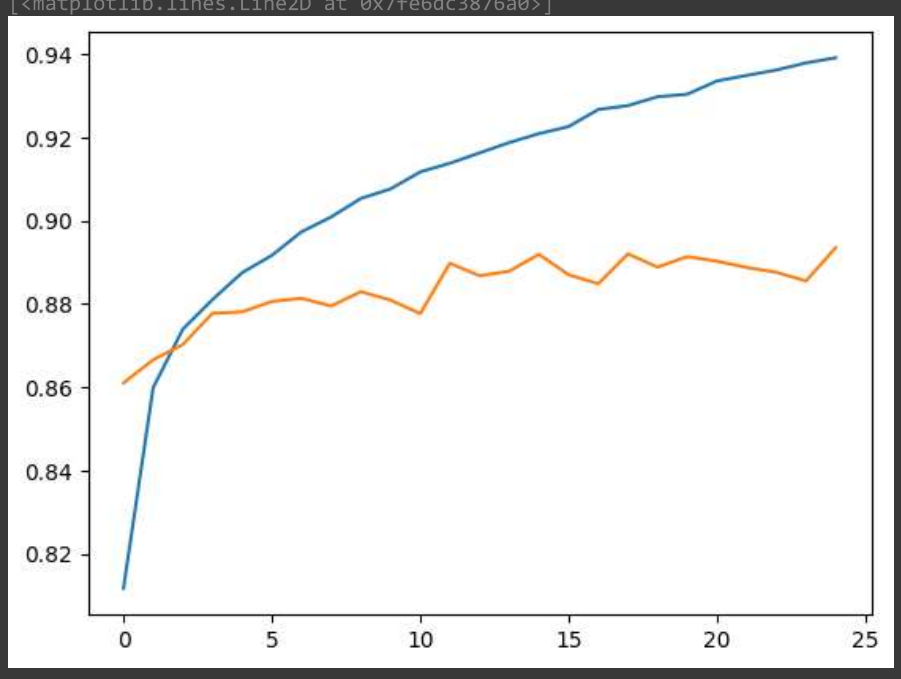
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7fe6daa63b80>]
```




```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

```
[<matplotlib.lines.Line2D at 0x7fe6dc3876a0>]
```




```
yprob = model.predict(xtest)
```

```
yprob
# the probabilityfor each image would be because the image hold 0 to 9
```

```
313/313 [==============================] - 1s 4ms/step
array([[5.9739646e-06, 9.0776155e-08, 1.0418838e-07, ..., 3.1115396e-03,
        2.1421345e-06, 9.9686658e-01],
       [4.4418261e-06, 2.0802497e-29, 9.9978709e-01, ..., 8.5245548e-26,
        5.1381476e-15, 2.5444031e-22],
       [5.3217633e-18, 9.9999994e-01, 3.2869661e-21, ..., 5.3920342e-27,
        9.0596932e-21, 2.4692541e-21],
       ...,
       [1.4404990e-16, 8.7817884e-22, 6.9923119e-14, ..., 1.7170681e-16,
        9.9999994e-01, 5.4965073e-15],
       [4.5986292e-12, 9.9999994e-01, 9.8653445e-15, ..., 9.6074030e-17,
        2.5231279e-14, 1.1103196e-16],
       [1.3218482e-09, 1.5410868e-14, 7.0234205e-09, ..., 2.6203052e-08,
        3.3480099e-08, 1.5272267e-09]], dtype=float32)
```

```
yprob[0]
```

```
array([5.9739646e-06, 9.0776155e-08, 1.0418838e-07, 5.2817725e-08,
       9.7353627e-09, 5.1395045e-07, 1.2894427e-05, 3.1115396e-03,
       2.1421345e-06, 9.9686658e-01], dtype=float32)
```

```
'''
0 T-shirt/top
1 Trouser
2 Pullover
3 Dress
4 Coat
5 Sandal
6 Shirt
7 Sneaker
8 Bag
9 Ankle boot
'''
```

```
# argmax will return the value of inex holding the maximum value
ypred = yprob.argmax(axis = 1)
ypred
```

```
array([9, 2, 1, ..., 8, 1, 5])
```

```
ypred[0]
```

```
9
```

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

```
              precision    recall  f1-score   support

           0       0.85      0.82      0.83      1000
           1       0.98      0.97      0.97      1000
           2       0.80      0.80      0.80      1000
           3       0.88      0.89      0.88      1000
           4       0.77      0.84      0.80      1000
           5       0.97      0.97      0.97      1000
           6       0.71      0.67      0.69      1000
           7       0.93      0.97      0.95      1000
           8       0.98      0.96      0.97      1000
           9       0.98      0.93      0.95      1000

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

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

```
[<matplotlib.lines.Line2D at 0x7fe6a400bc10>]
```