```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

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

+ Code + Text

```
(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 [==============================] - 2s 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)
```

```
xtrain[0].shape
```

```
(28, 28)
```

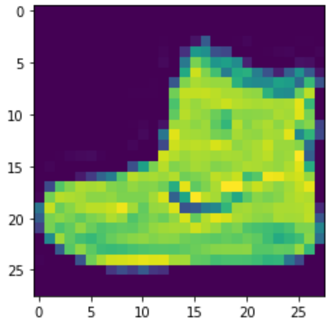
```
xtrain[0]
```

```
        229,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
        183, 225, 216, 223, 228, 235, 227, 224, 222, 224, 221, 223, 245,
        173,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
        193, 228, 218, 213, 198, 180, 212, 210, 211, 213, 223, 220, 243,
        202,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   1,   3,   0,  12,
        219, 220, 212, 218, 192, 169, 227, 208, 218, 224, 212, 226, 197,
        209,  52],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   6,   0,  99,
        244, 222, 220, 218, 203, 198, 221, 215, 213, 222, 220, 245, 119,
        167,  56],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   4,   0,   0,  55,
        236, 228, 230, 228, 240, 232, 213, 218, 223, 234, 217, 217, 209,
         92,   0],
       [  0,   0,   1,   4,   6,   7,   2,   0,   0,   0,   0,   0, 237,
        226, 217, 223, 222, 219, 222, 221, 216, 223, 229, 215, 218, 255,
         77,   0],
       [  0,   3,   0,   0,   0,   0,   0,   0,   0,  62, 145, 204, 228,
        207, 213, 221, 218, 208, 211, 218, 224, 223, 219, 215, 224, 244,
        159,   0],
       [  0,   0,   0,   0,  18,  44,  82, 107, 189, 228, 220, 222, 217,
```

```
       [  0,   0,   0,   0,   0,   0,   0,  40,  61,  44,  72,  41,  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,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   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)
```

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

```
<matplotlib.image.AxesImage at 0x7f08ed49d760>
```




```
ytrain.shape
```

```
(60000,)
```

```
xtest.shape
```

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

```
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,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   3,   1,   0,   0,   7,   0,  37,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          1,   2,   0,  27,  84,  11,   0,   0,   0,   0,   0,   0, 119,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          1,   0,   0,  88, 143, 110,   0,   0,   0,   0,  22,  93, 106,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          4,   0,  53, 129, 120, 147, 175, 157, 166, 135, 154, 168, 140,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   2,
          0,  11, 137, 130, 128, 160, 176, 159, 167, 178, 149, 151, 144,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   1,   0,   2,   1,   0,   3,   0,
          0, 115, 114, 106, 137, 168, 153, 156, 165, 167, 143, 157, 158,
         11,   0],
       [  0,   0,   0,   0,   1,   0,   0,   0,   0,   0,   3,   0,   0,
         89, 139,  90,  94, 153, 149, 131, 151, 169, 172, 143, 159, 169,
         48,   0],
       [  0,   0,   0,   0,   0,   0,   2,   4,   1,   0,   0,   0,  98,
        136, 110, 109, 110, 162, 135, 144, 149, 159, 167, 144, 158, 169,
        119,   0],
       [  0,   0,   2,   2,   1,   2,   0,   0,   0,   0,  26, 108, 117,
         99, 111, 117, 136, 156, 134, 154, 154, 156, 160, 141, 147, 156,
        178,   0],
       [  3,   0,   0,   0,   0,   0,   0,  21,  53,  92, 117, 111, 103,
```

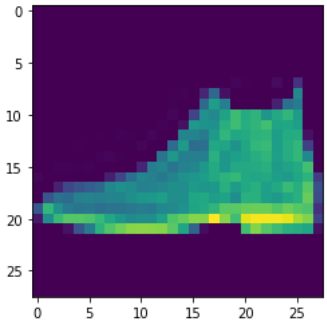
```
       115, 129, 134, 143, 154, 165, 170, 154, 151, 154, 143, 138, 150,
       165,  43],
      [  0,   0,  23,  54,  65,  76,  85, 118, 128, 123, 111, 113, 118,
       127, 125, 139, 133, 136, 160, 140, 155, 161, 144, 155, 172, 161,
       189,  62],
      [  0,  68,  94,  90, 111, 114, 111, 114, 115, 127, 135, 136, 143,
       126, 127, 151, 154, 143, 148, 125, 162, 162, 144, 138, 153, 162,
       196,  58],
      [ 70, 169, 129, 104,  98, 100,  94,  97,  98, 102, 108, 106, 119,
```

```
ytest
```

```
array([9, 2, 1, ..., 8, 1, 5], dtype=uint8)
```

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

```
<matplotlib.image.AxesImage at 0x7f08eab3d4f0>
```




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

```
# model building

model = Sequential()
model.add(Flatten(input_shape = (28,28)))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))

# add output layer
model.add(Dense(10, activation = 'softmax'))

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

history = model.fit(xtrain, ytrain, epochs = 4, validation_split = 0.2)
```

```
Epoch 1/4
1500/1500 [==============================] - 11s 6ms/step - loss: 0.5210 - accuracy: 0.8170 - val_loss: 0.4362 - val_accuracy: 0.84
Epoch 2/4
1500/1500 [==============================] - 9s 6ms/step - loss: 0.3882 - accuracy: 0.8606 - val_loss: 0.3885 - val_accuracy: 0.859
Epoch 3/4
1500/1500 [==============================] - 10s 7ms/step - loss: 0.3460 - accuracy: 0.8740 - val_loss: 0.3695 - val_accuracy: 0.86
Epoch 4/4
1500/1500 [==============================] - 8s 5ms/step - loss: 0.3242 - accuracy: 0.8802 - val_loss: 0.3794 - val_accuracy: 0.866
```

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

```
313/313 [==============================] - 1s 2ms/step
array([4.3102315e-08, 2.3085565e-07, 3.0462271e-08, 1.8243986e-08,
       4.1405138e-10, 6.9130175e-03, 1.1727066e-07, 9.8088365e-03,
       1.5109680e-05, 9.8326254e-01], dtype=float32)
```

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

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

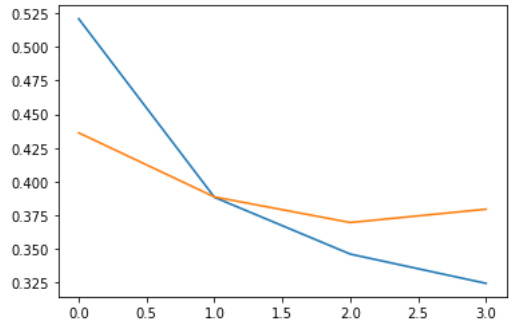
```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.82      1000
           1       0.99      0.96      0.97      1000
           2       0.69      0.86      0.76      1000
```

```
              3       0.83      0.90      0.86      1000
              4       0.75      0.77      0.76      1000
              5       0.97      0.95      0.96      1000
              6       0.79      0.42      0.55      1000
              7       0.94      0.94      0.94      1000
              8       0.94      0.97      0.95      1000
              9       0.94      0.96      0.95      1000

       accuracy                           0.86     10000
      macro avg       0.86      0.86      0.85     10000
   weighted avg       0.86      0.86      0.85     10000
```

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

```
[<matplotlib.lines.Line2D at 0x7f08bf8ef610>]
```



Colab paid products - Cancel contracts here