```
## import libraries


import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras


#load data


(X_train,y_train),(X_test,y_test)=tf.keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
40960/29515 [========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
16384/5148 [================================================================================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

```
y_train[0]
```

```
9
```

Saved successfully!

```
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,   1,
          0,   0,  13,  73,   0,   0,   1,   4,   0,   0,   0,   0,   1,
          1,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
          0,  36, 136, 127,  62,  54,   0,   0,   0,   1,   3,   4,   0,
          0,   3],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   6,
          0, 102, 204, 176, 134, 144, 123,  23,   0,   0,   0,   0,  12,
         10,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0, 155, 236, 207, 178, 107, 156, 161, 109,  64,  23,  77, 130,
         72,  15],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1,   0,
         69, 207, 223, 218, 216, 216, 163, 127, 121, 122, 146, 141,  88,
        172,  66],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   1,   1,   1,   0,
        200, 232, 232, 233, 229, 223, 223, 215, 213, 164, 127, 123, 196,
        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,
```

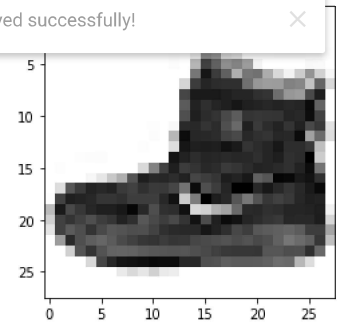
```
        159,   0],
       [  0,   0,   0,   0,  18,  44,  82, 107, 189, 228, 220, 222, 217,
        226, 200, 205, 211, 230, 224, 234, 176, 188, 250, 248, 233, 238,
        215,   0],
       [  0,  57, 187, 208, 224, 221, 224, 208, 204, 214, 208, 209, 200,
        159, 245, 193, 206, 223, 255, 255, 221, 234, 221, 211, 220, 232,
        246,   0],
       [  3, 202, 228, 224, 221, 211, 211, 214, 205, 205, 205, 220, 240,
         80, 150, 255, 229, 221, 188, 154, 191, 210, 204, 209, 222, 228,
        225,   0],
```

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

```
'\n0 T-shirt/top\n1 Trouser\n2 Pullover\n3 Dress\n4 Coat\n5 Sandal\n6 Shirt\n7 Sneaker\n8 Bag\n9 Ankle
boot\n'
```

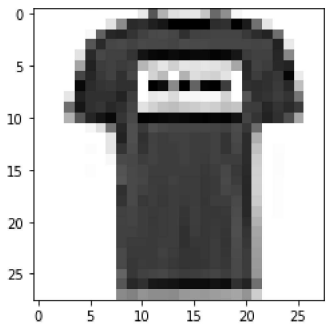
```
#show image


plt.imshow(X_train[0],cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x7f8ad3333210>
```


```
plt.imshow(X_train[1],cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x7f8ad3319b10>
```




```
class_labels = ["T-shirt/top","Trouser","Pullover","Dress","Coat","Sandal","Shirt","sneaker","Bag","Ankle boot"]
plt.figure(figsize=(16,16))
j=1
for i in np.random.randint(0,1000,25):
    plt.subplot(5,5,j); j+=1
    plt.imshow(X_train[i],cmap='Greys')
    plt.axis('off')
    plt.title('{} / {}'.format(class_labels[y_train[i]],y_train[i]))
```

Saved successfully!

```
X_train.shape
```

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

```
X_train.ndim
```

```
3
```

```
X_train= np.expand_dims(X_train,-1)
```

```
X_train.shape
```

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

```
X_test.ndim
```

```
3
```

```
# feature scaling
```

```
X_train[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]],
```



```
X_train = X_train/255
X_test = X_test/255


from sklearn.model_selection import train_test_split
X_train,X_validation,y_train,y_validation=train_test_split(X_train,y_train,test_size=0.2,random_state=2020)


X_train.shape,X_validation.shape,y_train.shape,y_validation.shape
```

```
((48000, 28, 28, 1), (12000, 28, 28, 1), (48000,), (12000,))
```

```
# Build CNN Model


model = keras.models.Sequential([
                                keras.layers.Conv2D(filters=32,kernel_size=3,strides=(1,1),padding='valid',activation='relu',input_shape=[28
                                keras.layers.MaxPooling2D(pool_size=(2,2)),
                                keras.layers.Flatten(),
                                keras.layers.Dense(units=128,activation='relu'),
                                keras.layers.Dense(units=10,activation='softmax')
])


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
```

```
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
flatten (Flatten)            (None, 5408)              0
dense (Dense)                (None, 128)               692352
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
```

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

```
model.fit(X_train,y_train,epochs=10,batch_size=512,verbose=1,validation_data=(X_validation,y_validation))
```

```
Epoch 1/10
94/94 [==============================] - 19s 193ms/step - loss: 0.6307 - accuracy: 0.7839 - val_loss: 0.4303 - val_accuracy: 0.8517
Epoch 2/10
94/94 [==============================] - 18s 192ms/step - loss: 0.3789 - accuracy: 0.8685 - val_loss: 0.3660 - val_accuracy: 0.8727
Epoch 3/10
94/94 [==============================] - 18s 193ms/step - loss: 0.3318 - accuracy: 0.8840 - val_loss: 0.3516 - val_accuracy: 0.8792
Epoch 4/10
94/94 [==============================] - 18s 192ms/step - loss: 0.3039 - accuracy: 0.8922 - val_loss: 0.3257 - val_accuracy: 0.8842
Epoch 5/10
94/94 [==============================] - 18s 192ms/step - loss: 0.2808 - accuracy: 0.8994 - val_loss: 0.3225 - val_accuracy: 0.8882
Epoch 6/10
94/94 [==============================] - 18s 193ms/step - loss: 0.2640 - accuracy: 0.9061 - val_loss: 0.2936 - val_accuracy: 0.8964
Epoch 7/10
94/94 [==============================] - 18s 193ms/step - loss: 0.2461 - accuracy: 0.9130 - val_loss: 0.3073 - val_accuracy: 0.8897
Epoch 8/10
94/94 [==============================] - 18s 191ms/step - loss: 0.2364 - accuracy: 0.9150 - val_loss: 0.2822 - val_accuracy: 0.9032
Epoch 9/10
94/94 [==============================] - 18s 191ms/step - loss: 0.2234 - accuracy: 0.9201 - val_loss: 0.2792 - val_accuracy: 0.9007
Epoch 10/10
94/94 [==============================] - 18s 192ms/step - loss: 0.2109 - accuracy: 0.9251 - val_loss: 0.2726 - val_accuracy: 0.9051
<keras.callbacks.History at 0x7f8acf3f3210>
```

Saved successfully!

```
#Test the model
```

```
X_test = np.expand_dims(X_test,-1)
```

```
X_test.shape
```

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

```
y_pred = model.predict(X_test).round(2)
y_pred
np.argmax(y_pred[0])
```

```
9
```

```
model.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.2807 - accuracy: 0.8971
[0.2807179093360901, 0.8970999717712402]
```

```
X_test.ndim
```

```
5
```

```
#Visualize output
```

```
plt.figure(figsize=(16,16))
j=1
for i in np.random.randint(0,1000,25):
  plt.subplot(5,5,j); j+=1
  plt.imshow(X_test[i].reshape(28,28),cmap='Greys')
  plt.axis('off')
  plt.title('{} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]],y_test[i],class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i]))
```

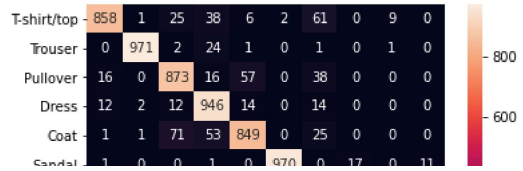
Saved successfully!

```
from sklearn.metrics import confusion_matrix
plt.figure(figsize=(16,9))
y_pred_labels = [np.argmax(label) for label in y_pred]
cm = confusion_matrix(y_test,y_pred_labels)
cm
```

```
array([[858,   1,  25,  38,   6,   2,  61,   0,   9,   0],
       [  0, 971,   2,  24,   1,   0,   1,   0,   1,   0],
       [ 16,   0, 873,  16,  57,   0,  38,   0,   0,   0],
       [ 12,   2,  12, 946,  14,   0,  14,   0,   0,   0],
       [  1,   1,  71,  53, 849,   0,  25,   0,   0,   0],
       [  1,   0,   0,   1,   0, 970,   0,  17,   0,  11],
       [140,   1, 113,  43,  80,   0, 613,   0,  10,   0],
       [  0,   0,   0,   0,   0,   9,   0, 954,   1,  36],
       [  2,   1,   4,   8,   3,   2,   1,   5, 974,   0],
       [  1,   0,   0,   0,   0,   7,   0,  28,   0, 964]])
<Figure size 1152x648 with 0 Axes>
```

```
sns.heatmap(cm,annot=True,fmt='d',xticklabels=class_labels,yticklabels=class_labels)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8acb4512d0>
```




```
from sklearn.metrics import classification_report
cr = classification_report(y_test,y_pred_labels,target_names=class_labels)
print(cr)
```

```
                precision    recall  f1-score   support

   T-shirt/top       0.83      0.86      0.84      1000
       Trouser       0.99      0.97      0.98      1000
      Pullover       0.79      0.87      0.83      1000
         Dress       0.84      0.95      0.89      1000
          Coat       0.84      0.85      0.84      1000
        Sandal       0.98      0.97      0.97      1000
         Shirt       0.81      0.61      0.70      1000
       sneaker       0.95      0.95      0.95      1000
           Bag       0.98      0.97      0.98      1000
    Ankle boot       0.95      0.96      0.96      1000

      accuracy                           0.90     10000
     macro avg       0.90      0.90      0.90     10000
  weighted avg       0.90      0.90      0.90     10000
```

```
model.save('fashion_mnsit_data_classification.h5')
```

Saved successfully!

```
shion_mnsit_data_classification.h5')
```

```
model2.predict(X_test).round(2)
```

```
array([[0.  , 0.  , 0.  , ..., 0.01, 0.  , 0.99],
       [0.  , 0.  , 1.  , ..., 0.  , 0.  , 0.  ],
       [0.  , 1.  , 0.  , ..., 0.  , 0.  , 0.  ],
       ...,
       [0.  , 0.  , 0.  , ..., 0.  , 0.99, 0.  ],
       [0.  , 1.  , 0.  , ..., 0.  , 0.  , 0.  ],
       [0.  , 0.  , 0.  , ..., 0.07, 0.05, 0.01]], dtype=float32)
```

```
y_pred_sample = model2.predict(X_test).round(2)
```

```
np.argmax(y_pred_sample[0])
```

```
9
```

```
y_test
```

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

Saved successfully!