Fashion MNIST Data classification using CNN

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

```
(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-datase
32768/29515 [==================================] - 0s 0us/step
40960/29515 [==========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
16384/5148 [====================================================================
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

```
X_train.shape,y_train.shape
```

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

```
X_test.shape, y_test.shape
```

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

```
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,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   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],
       [ 98, 233, 198, 210, 222, 229, 229, 234, 249, 220, 194, 215, 217,
        241,  65,  73, 106, 117, 168, 219, 221, 215, 217, 223, 223, 224,
```

```
y_train[0]
```

```
     9
```

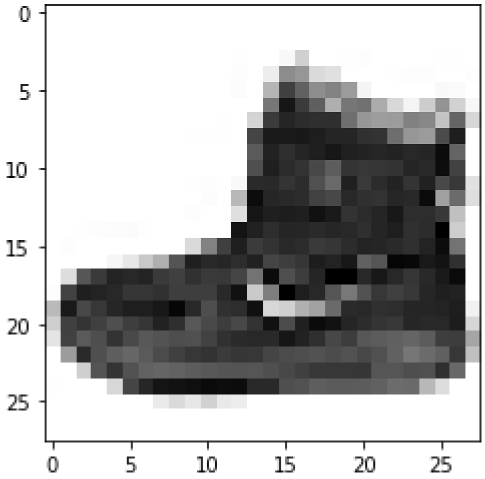
```
'''
0   T-shirt/top
1   Trouser
2   Pullover
3   Dress
4   Coat
5   Sandal
6   Shirt
7   Sneaker
8   Bag
```

```
9   Ankle boot
'''
```

```
      '\n0\tT-shirt/top\n1\tTrouser\n2\tPullover\n3\tDress\n4\tCoat\n5\tSandal\n6\tSh
```

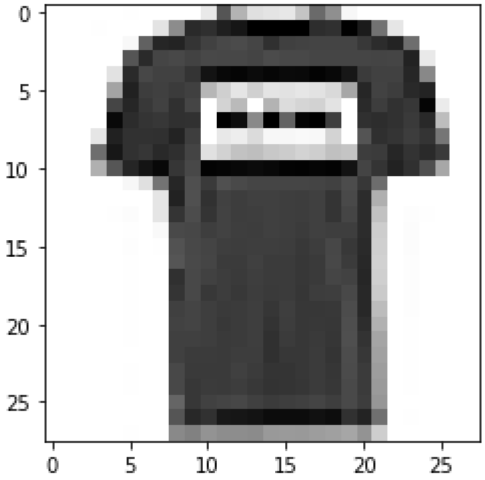
```
plt.imshow(X_train[0],cmap='Greys')
```

```
      <matplotlib.image.AxesImage at 0x7f05d8002b10>
```

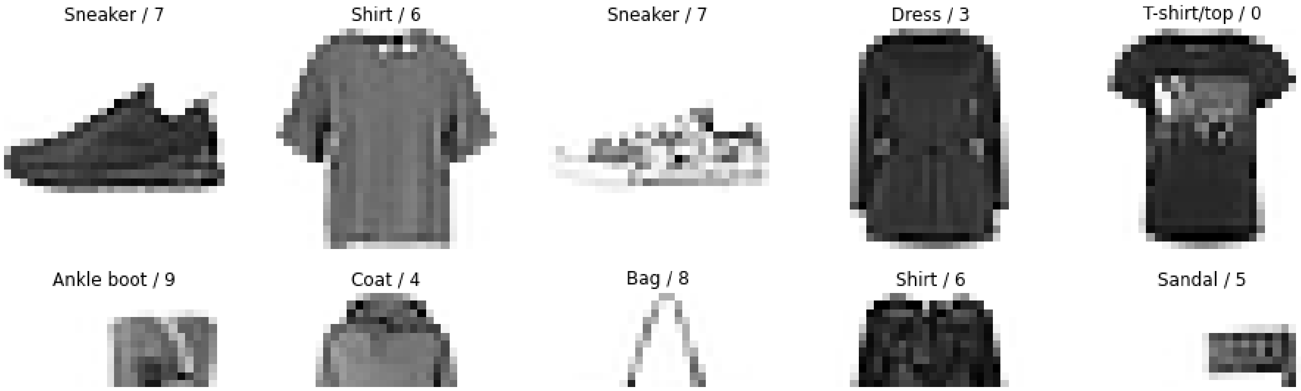


```
plt.imshow(X_train[1],cmap='Greys')
```

```
      <matplotlib.image.AxesImage at 0x7f05d7af4750>
```



```
class_labels = ["T-shirt/top","Trouser",    "Pullover","Dress","Coat",  "Sandal",   "Shirt
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]))
```



```
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_train.ndim
```

```
      4
```

```
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],
              [  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,r
```

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

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

Building CNN model

[ ] ↳ *6 cells hidden*

Test the model

```
X_test.shape, y_test.shape
```

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

```
X_test[0]
```

```
        0.44705882, 0.43529412, 0.44705882, 0.45098039, 0.49803922,
        0.52941176, 0.53333333, 0.56078431, 0.49411765, 0.49803922,
        0.59215686, 0.60392157, 0.56078431, 0.58039216, 0.49019608,
        0.63529412, 0.63529412, 0.56470588, 0.54117647, 0.6       ,
        0.63529412, 0.76862745, 0.22745098],
       [0.2745098 , 0.6627451 , 0.50588235, 0.40784314, 0.38431373,
        0.39215686, 0.36862745, 0.38039216, 0.38431373, 0.4       ,
        0.42352941, 0.41568627, 0.46666667, 0.47058824, 0.50588235,
        0.58431373, 0.61176471, 0.65490196, 0.74509804, 0.74509804,
        0.76862745, 0.77647059, 0.77647059, 0.73333333, 0.77254902,
        0.74117647, 0.72156863, 0.14117647],
       [0.0627451 , 0.49411765, 0.67058824, 0.7372549 , 0.7372549 ,
        0.72156863, 0.67058824, 0.6       , 0.52941176, 0.47058824,
        0.49411765, 0.49803922, 0.57254902, 0.7254902 , 0.76470588,
        0.81960784, 0.81568627, 1.        , 0.81960784, 0.69411765,
        0.96078431, 0.98823529, 0.98431373, 0.98431373, 0.96862745,
        0.8627451 , 0.80784314, 0.19215686],
       [0.        , 0.        , 0.        , 0.04705882, 0.2627451 ,
        0.41568627, 0.64313725, 0.7254902 , 0.78039216, 0.82352941,
        0.82745098, 0.82352941, 0.81568627, 0.74509804, 0.58823529,
        0.32156863, 0.03137255, 0.        , 0.        , 0.        ,
        0.69803922, 0.81568627, 0.7372549 , 0.68627451, 0.63529412,
        0.61960784, 0.59215686, 0.04313725],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.         , 0.         , 0.         ],
       [0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,

        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         ],
       [0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         , 0.         , 0.         ,
        0.         , 0.         , 0.         ]])
```

```
y_test[0]
```

```
      9
```

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

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

```
class_labels = ["T-shirt/top","Trouser",    "Pullover","Dress","Coat",  "Sandal",   "Shirt
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],cmap="Greys")
    plt.axis('off')
    plt.title('{} / {}'.format(class_labels[y_test[i]],y_test[i]))
```

```
X_test.shape
```

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

```
X_test.ndim
```

```
    3
```

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

```
X_test[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.        ],
```

```
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,r
```

```
X_test.shape,X_validation.shape,y_test.shape,y_validation.shape
```

```
((10000, 28, 28, 1), (9600, 28, 28, 1), (10000,), (9600,))
```

```
model=keras.models.Sequential([
                         keras.layers.Conv2D(filters=32,kernel_size=3,strides=(1,1),paddin
                         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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 5408)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 128)               692352    
_________________________________________________________________
dense_3 (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_test,y_test,epochs=10,batch_size=512,verbose=1,validation_data=(X_validation,y
```

```
Epoch 1/10
20/20 [==============================] - 6s 244ms/step - loss: 1.1317 - accurac
Epoch 2/10
20/20 [==============================] - 5s 233ms/step - loss: 0.5949 - accurac
Epoch 3/10
20/20 [==============================] - 5s 235ms/step - loss: 0.4994 - accurac
Epoch 4/10
20/20 [==============================] - 5s 234ms/step - loss: 0.4551 - accurac
Epoch 5/10
20/20 [==============================] - 5s 234ms/step - loss: 0.4254 - accurac
Epoch 6/10
20/20 [==============================] - 5s 235ms/step - loss: 0.4057 - accurac
Epoch 7/10
20/20 [==============================] - 5s 235ms/step - loss: 0.3807 - accurac
Epoch 8/10
20/20 [==============================] - 5s 236ms/step - loss: 0.3587 - accurac
Epoch 9/10
20/20 [==============================] - 5s 234ms/step - loss: 0.3405 - accurac
Epoch 10/10
20/20 [==============================] - 5s 235ms/step - loss: 0.3214 - accurac
<keras.callbacks.History at 0x7f05d79e2f90>
```