In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
import keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Flatten, Dense
import random
%matplotlib inline
```

In [12]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

x_train.shape, y_train.shape, " ", x_test.shape, y_test.shape
```

Out[12]:
```
((60000, 28, 28), (60000,), ' ', (10000, 28, 28), (10000,))
```

In [13]:
```
print("the shape of x_train is: {} and y_train is:{}".format(x_train.shape,y_train.
print("the shape of x_test is: {} and y_test is:{}".format(x_test.shape,y_test.shap
```

```
the shape of x_train is: (60000, 28, 28) and y_train is:(60000,)
the shape of x_test is: (10000, 28, 28) and y_test is:(10000,)
```

In [14]:
```
x_train[0]
```

Out[14]:
```
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],
       [ 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,
        229,  29],
       [ 75, 204, 212, 204, 193, 205, 211, 225, 216, 185, 197, 206, 198,
        213, 240, 195, 227, 245, 239, 223, 218, 212, 209, 222, 220, 221,
        230,  67],
       [ 48, 203, 183, 194, 213, 197, 185, 190, 194, 192, 202, 214, 219,
```

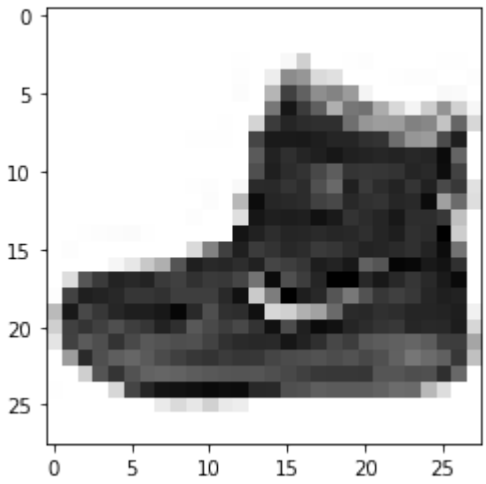
```
       221, 220, 236, 225, 216, 199, 206, 186, 181, 177, 172, 181, 205,
       206, 115],
      [  0, 122, 219, 193, 179, 171, 183, 196, 204, 210, 213, 207, 211,
       210, 200, 196, 194, 191, 195, 191, 198, 192, 176, 156, 167, 177,
       210,  92],
      [  0,   0,  74, 189, 212, 191, 175, 172, 175, 181, 185, 188, 189,
       188, 193, 198, 204, 209, 210, 210, 211, 188, 188, 194, 192, 216,
       170,   0],
      [  2,   0,   0,   0,  66, 200, 222, 237, 239, 242, 246, 243, 244,
       221, 220, 193, 191, 179, 182, 182, 181, 176, 166, 168,  99,  58,
         0,   0],
      [  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)
```

In [15]:
```
class_labels = ["Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker
class_labels
```

Out[15]:
```
['Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker']
```

In [17]:
```
plt.imshow(x_train[0], cmap="Greys")
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x18a3e32d430>
```



In [18]:
```
x_train.shape
```

Out[18]:
```
(60000, 28, 28)
```

In [19]:
```
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]))
```

Out[19]:
```
Text(0.5, 1.0, 'Sandal / 4')
```

In [20]:
```
x_train.ndim
```

Out[20]:
```
3
```

In [21]:
```
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
```

In [22]:
```
x_train.ndim
```

Out[22]:
```
4
```

In [23]:
```
x_train = x_train/255
x_test = x_test/255
```

In [24]:
```
from sklearn.model_selection import train_test_split
x_train, x_validation, y_train, y_validation = train_test_split(x_train, y_train)
```

In [25]:
```
x_train.shape, y_train.shape, x_validation.shape, y_validation.shape
```

Out[25]:
```
((45000, 28, 28, 1), (45000,), (15000, 28, 28, 1), (15000,))
```

In [26]:
```
cnn = keras.models.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=3,strides=(1,1),padding='valid',
    tf.keras.layers.MaxPooling2D((2,2)),
```

```
    tf.keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2,2),padding='same',
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(256,activation='relu'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dense(10,activation='softmax'),
])
```

```
C:\Users\hpcnd\anaconda3\lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [28]:
```
cnn.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])
```

In [29]:
```
cnn.fit(x_train, y_train, epochs=20, batch_size=16, verbose=1, validation_data=(x_v
```

```
Epoch 1/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 26s 8ms/step - accuracy: 0.6958 - loss: 0.8025 - va
l_accuracy: 0.8607 - val_loss: 0.3819
Epoch 2/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.8547 - loss: 0.3967 - va
l_accuracy: 0.8825 - val_loss: 0.3217
Epoch 3/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.8766 - loss: 0.3328 - va
l_accuracy: 0.8961 - val_loss: 0.2784
Epoch 4/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.8899 - loss: 0.3003 - va
l_accuracy: 0.8971 - val_loss: 0.2819
Epoch 5/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 24s 8ms/step - accuracy: 0.8969 - loss: 0.2812 - va
l_accuracy: 0.8993 - val_loss: 0.2792
Epoch 6/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 24s 8ms/step - accuracy: 0.9049 - loss: 0.2574 - va
l_accuracy: 0.9018 - val_loss: 0.2676
Epoch 7/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.9113 - loss: 0.2427 - va
l_accuracy: 0.9033 - val_loss: 0.2659
Epoch 8/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.9144 - loss: 0.2339 - va
l_accuracy: 0.9057 - val_loss: 0.2715
Epoch 9/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.9195 - loss: 0.2219 - va
l_accuracy: 0.9035 - val_loss: 0.2680
Epoch 10/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.9250 - loss: 0.2064 - va
l_accuracy: 0.9127 - val_loss: 0.2581
Epoch 11/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9231 - loss: 0.2050 - va
l_accuracy: 0.9044 - val_loss: 0.2785
Epoch 12/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 21s 8ms/step - accuracy: 0.9288 - loss: 0.1971 - va
l_accuracy: 0.9093 - val_loss: 0.2657
Epoch 13/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9304 - loss: 0.1861 - va
l_accuracy: 0.9080 - val_loss: 0.2751
Epoch 14/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 21s 7ms/step - accuracy: 0.9313 - loss: 0.1846 - va
l_accuracy: 0.9088 - val_loss: 0.2713
Epoch 15/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 23s 8ms/step - accuracy: 0.9310 - loss: 0.1842 - va
l_accuracy: 0.9115 - val_loss: 0.2651
Epoch 16/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9372 - loss: 0.1675 - va
l_accuracy: 0.9105 - val_loss: 0.2722
Epoch 17/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9388 - loss: 0.1659 - va
l_accuracy: 0.9107 - val_loss: 0.2799
Epoch 18/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9414 - loss: 0.1596 - va
l_accuracy: 0.9120 - val_loss: 0.2762
Epoch 19/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9439 - loss: 0.1533 - va
l_accuracy: 0.9131 - val_loss: 0.2747
Epoch 20/20
2813/2813 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - accuracy: 0.9404 - loss: 0.1574 - va
l_accuracy: 0.9072 - val_loss: 0.3003
```

Out[29]:
```
<keras.src.callbacks.history.History at 0x18a40e5b070>
```

In [30]:
```
y_pred = cnn.predict(x_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```

In [31]:
```
cnn.evaluate(x_test, y_test)
```

Out[31]:
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8996 - loss: 0.3582
[0.33889663219451904, 0.8996000289916992]
```

In [32]:
```
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]))
```

Out[32]:
```
Text(0.5, 1.0, 'Dress / 2')
```

In [ ]:
```
```