In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
%matplotlib inline
```

In [ ]:
```
fashion_train_df= pd.read_csv('/content/drive/MyDrive/archive/fashion-mnist_train.csv')
```

In [ ]:
```
fashion_test_df = pd.read_csv('/content/drive/MyDrive/archive/fashion-mnist_test.csv')
```

In [ ]:
```
fashion_train_df.head()
```

Out[6]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pix
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	6	0	0	0	0	0	0	0	5	0	...	0	0	0	30	43	
3	0	0	0	0	1	2	0	0	0	0	...	3	0	0	0	0	
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 785 columns

In [ ]:
```
fashion_train_df.tail()
```

Out[7]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779
59995	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
59996	1	0	0	0	0	0	0	0	0	0	...	73	0	0	0	0
59997	8	0	0	0	0	0	0	0	0	0	...	160	162	163	135	94
59998	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
59999	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 785 columns

In [ ]:
```
fashion_train_df.shape
```

Out[8]:
```
(60000, 785)
```

In [ ]:
```
fashion_test_df.shape
```

Out[9]:
```
(10000, 785)
```

In [ ]:
```
training = np.array(fashion_train_df,dtype='float32')
testing = np.array(fashion_test_df,dtype='float32')
```

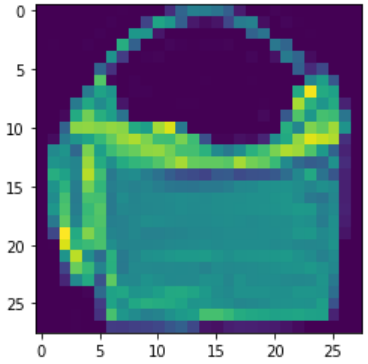
In [ ]:
```
training.shape
```

Out[11]:
```
(60000, 785)
```

In [ ]:
```
import random
```

In [ ]:
```
i = random.randint(0,60001)
plt.imshow(training[i,1:].reshape(28,28))
label = training[i,1]
label
```

Out[31]: 0.0



i = random.randint(0,60001) plt.imshow(training[i,1:].reshape(28,28)) label = training[i,1] label

In [ ]:
```
W_grid = 7
L_grid = 7

fig,axes = plt.subplots(L_grid,W_grid,figsize =(17,17))

axes = axes.ravel()
n_training = len(training)



for i in np.arange(0,W_grid*L_grid):
        index = np.random.randint(0,n_training)
        axes[i].imshow(training[index,1:].reshape((28,28)))
        axes[i].set_title(training[index,0],fontsize = 8)
        axes[i].axis('off')

plt.subplots_adjust(hspace=0.4)
```

In [ ]:
```
X_train = training[:,1:]/255
y_train = training[:,0]
X_test = testing[:,1:]/255
y_test = testing[:,0]
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_validate, y_train, y_validate = train_test_split(X_train, y_train,test_size = 0.2,random_state
```

In [ ]:
```
X_train = X_train.reshape(X_train.shape[0],*(28,28,1))
X_test = X_test.reshape(X_test.shape[0],*(28,28,1))
X_validate = X_validate.reshape(X_validate.shape[0],*(28,28,1))
```

In [ ]:
```
X_train.shape
```

Out[37]:
```
(48000, 28, 28, 1)
```

In [ ]:
```
X_test.shape
```

Out[38]:
```
(10000, 28, 28, 1)
```

In [ ]:
```
X_validate.shape
```

Out[39]:
```
(12000, 28, 28, 1)
```

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import TensorBoard
```

In [ ]:
```
cnn_model = Sequential()
cnn_model.add(Conv2D(32,3,3,input_shape = (28,28,1),activation = 'relu'))
cnn_model.add(MaxPooling2D(pool_size= (2,2)))
cnn_model.add(Flatten())
cnn_model.add(Dense(32,activation = 'relu'))
cnn_model.add(Dense(10,activation = 'sigmoid'))
cnn_model.compile(loss ='sparse_categorical_crossentropy',optimizer = Adam(learning_rate=0.001),metrics= [
```

In [ ]:
```
epochs = 200
```

In [ ]:
```
cnn_model.fit(X_train,y_train,batch_size =512,epochs = epochs,verbose = 1,validation_data = (X_validate,y_
```

```
Epoch 1/200
94/94 [==============================] - 3s 23ms/step - loss: 1.4013 - accuracy: 0.5608 - val_loss: 0.
7930 - val_accuracy: 0.7260
Epoch 2/200
94/94 [==============================] - 2s 21ms/step - loss: 0.6888 - accuracy: 0.7555 - val_loss: 0.
6148 - val_accuracy: 0.7827
Epoch 3/200
94/94 [==============================] - 2s 21ms/step - loss: 0.5766 - accuracy: 0.7950 - val_loss: 0.
5493 - val_accuracy: 0.8041
Epoch 4/200
94/94 [==============================] - 2s 21ms/step - loss: 0.5242 - accuracy: 0.8117 - val_loss: 0.
5121 - val_accuracy: 0.8207
Epoch 5/200
94/94 [==============================] - 2s 21ms/step - loss: 0.4940 - accuracy: 0.8219 - val_loss: 0.
4920 - val_accuracy: 0.8216
Epoch 6/200
94/94 [==============================] - 2s 21ms/step - loss: 0.4723 - accuracy: 0.8293 - val_loss: 0.
4667 - val_accuracy: 0.8342
Epoch 7/200
```

In [ ]:
```
evaluation = cnn_model.evaluate(X_test,y_test)
print('Test Accuracy : {:.3f}'.format(evaluation[1]))
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3451 - accuracy: 0.8813
Test Accuracy : 0.881
```

In [ ]:
```
predicted_classes = np.argmax(cnn_model.predict(X_test),axis=-1)
```

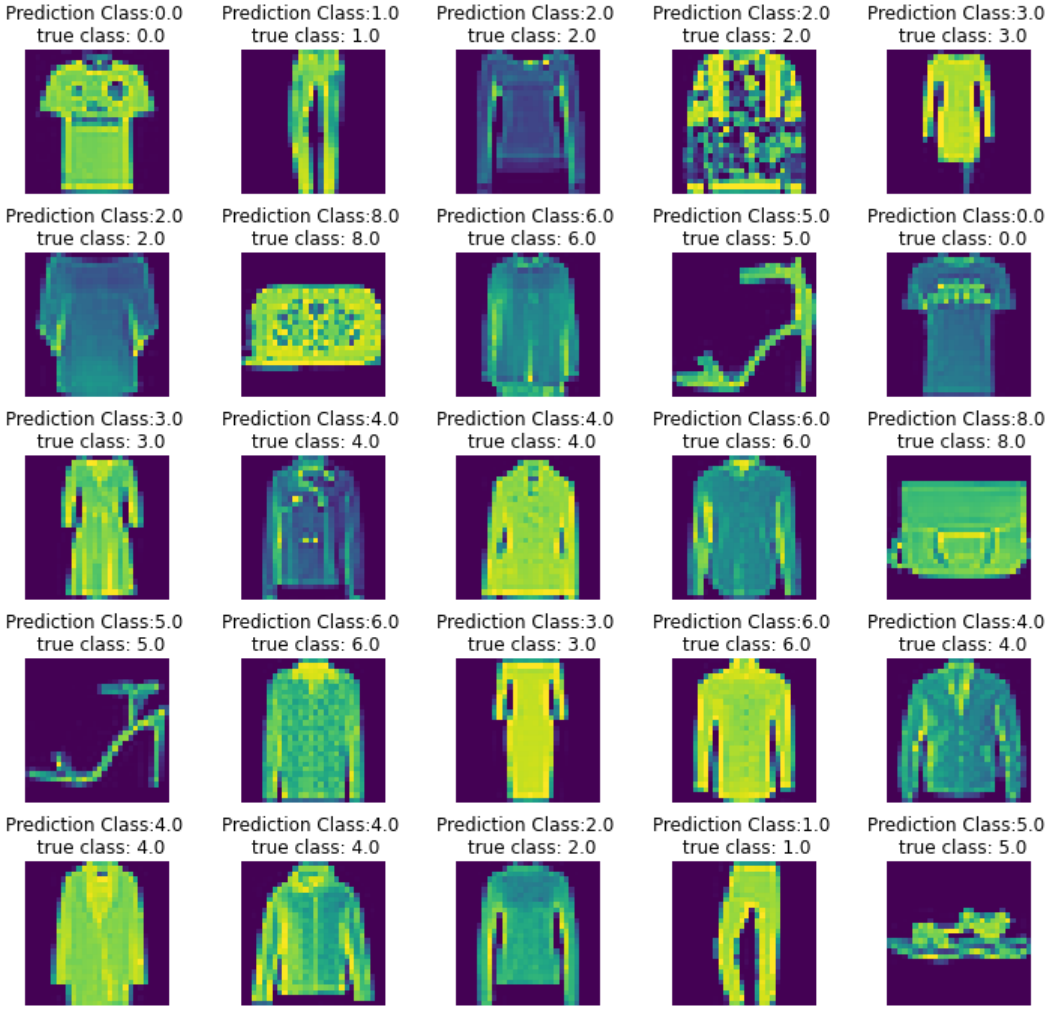
```
313/313 [==============================] - 0s 2ms/step
```

In [ ]:
```
predicted_classes
```

Out[71]:
```
array([0, 1, 2, ..., 8, 8, 1])
```

In [ ]:
```
L = 5
W = 5

fig,axes = plt.subplots(L,W,figsize = (12,12))
axes = axes.ravel()
for i in np.arange(0,L*W):
    axes[i].imshow(X_test[i].reshape(28,28))
    axes[i].set_title('Prediction Class:{1} \n true class: {1}'.format(predicted_classes[i],y_test[i]))
    axes[i].axis('off')
plt.subplots_adjust(wspace = 0.5)
```



In [ ]:
```
from sklearn.metrics import classification_report

classes = 10
targets = ["Class {}".format(i) for i in range(classes)]
print(classification_report(y_test, predicted_classes, target_names = targets))
```

```
              precision    recall  f1-score   support

     Class 0       0.80      0.86      0.83      1000
     Class 1       0.96      0.98      0.97      1000
     Class 2       0.84      0.80      0.82      1000
     Class 3       0.88      0.91      0.89      1000
     Class 4       0.82      0.80      0.81      1000
     Class 5       0.96      0.94      0.95      1000
     Class 6       0.69      0.66      0.68      1000
     Class 7       0.94      0.92      0.93      1000
     Class 8       0.97      0.98      0.97      1000
     Class 9       0.93      0.97      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
```