In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Dense, Flatten
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
from keras.utils import to_categorical
```

In [2]:

```
fashion_train_df = pd.read_csv(r'C:\Users\rohit\Desktop\Fashion\fashion-mnist-datasets
fashion_test_df = pd.read_csv(r'C:\Users\rohit\Desktop\Fashion\fashion-mnist-datasets\
```

In [3]:

```
fashion_train_df.shape
```

Out[3]:

```
(60000, 785)
```

In [4]:

```
fashion_train_df.columns
```

Out[4]:

```
Index(['label', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       ...
       'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779', 'pixel780',
       'pixel781', 'pixel782', 'pixel783', 'pixel784'],
      dtype='object', length=785)
```

In [5]:

```
print(set(fashion_train_df['label']))
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [6]:

```
print([fashion_train_df.drop(labels='label', axis=1).min(axis=1).min(),
      fashion_train_df.drop(labels='label', axis=1).max(axis=1).max()])
```

```
[0, 255]
```

In [7]:

```
fashion_train_df.head()
```

Out[7]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776
0	2	0	0	0	0	0	0	0	0	0	...	0	0
1	9	0	0	0	0	0	0	0	0	0	...	0	0
2	6	0	0	0	0	0	0	0	5	0	...	0	0
3	0	0	0	0	1	2	0	0	0	0	...	3	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 785 columns

In [8]:

```
fashion_test_df.shape
```

Out[8]:

```
(10000, 785)
```

In [9]:

```
fashion_test_df.head()
```

Out[9]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776
0	0	0	0	0	0	0	0	0	9	8	...	103	87
1	1	0	0	0	0	0	0	0	0	0	...	34	0
2	2	0	0	0	0	0	0	14	53	99	...	0	0
3	2	0	0	0	0	0	0	0	0	0	...	137	126
4	3	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 785 columns

In [10]:

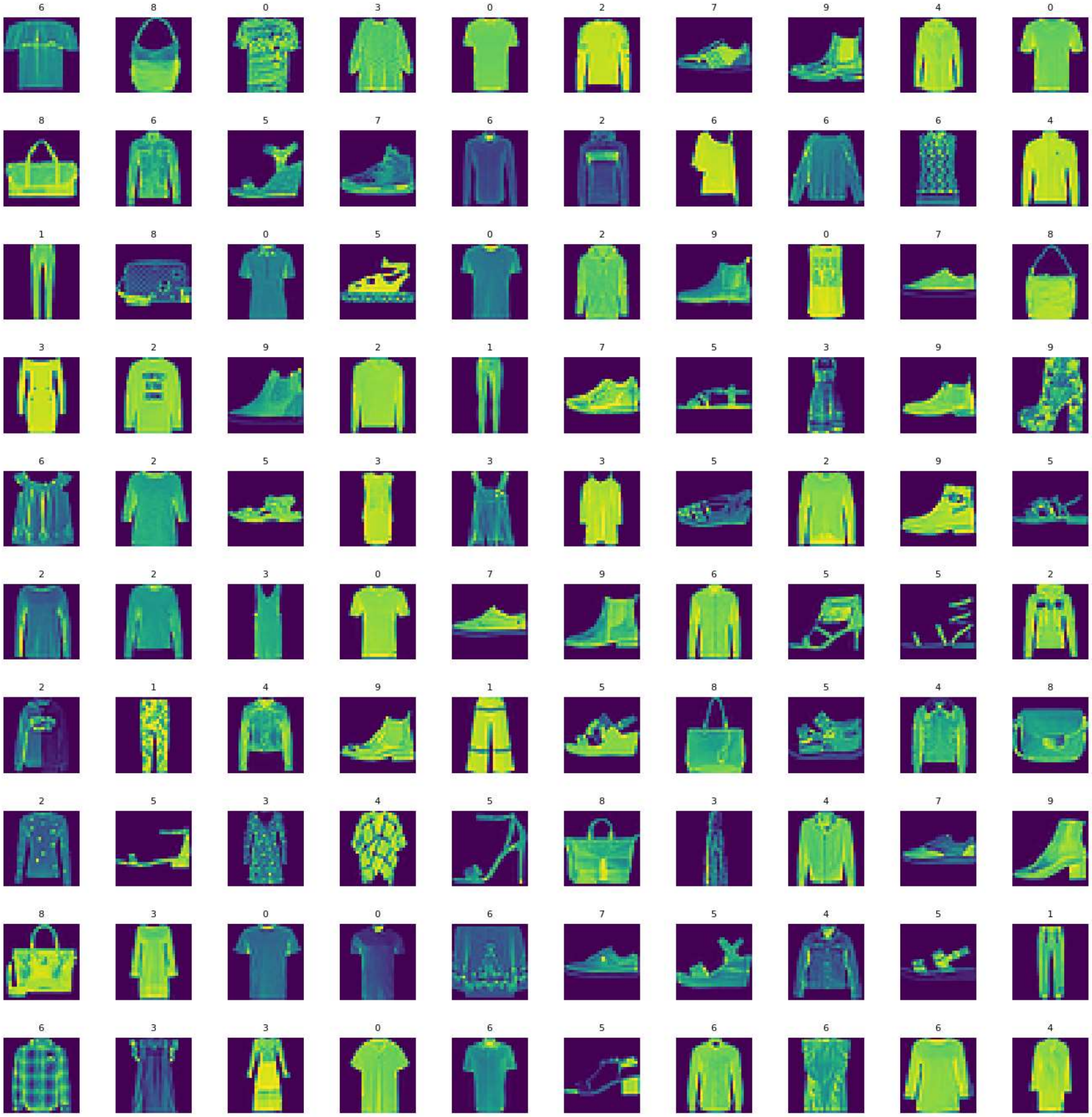
```
# Convert the dataframe ti numpy array
training = np.asarray(fashion_train_df, dtype='float32')

# Lets show multiple images in a 15x15 grid
height = 10
width = 10

fig, axes = plt.subplots(nrows=width, ncols=height, figsize=(17,17))
axes = axes.ravel()  # this flattens the 15x15 matrix into 225
n_train = len(training)

for i in range(0, height*width):
    index = np.random.randint(0, n_train)
    axes[i].imshow(training[index, 1:].reshape(28,28))
    axes[i].set_title(int(training[index, 0]), fontsize=8)
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.5)
```



In [11]:
```
# convert to numpy arrays and reshape
training = np.asarray(fashion_train_df, dtype='float32')
X_train = training[:, 1:].reshape([-1,28,28,1])
X_train = X_train/255   # Normalizing the data
y_train = training[:, 0]

testing = np.asarray(fashion_test_df, dtype='float32')
X_test = testing[:, 1:].reshape([-1,28,28,1])
X_test = X_test/255    # Normalizing the data
y_test = testing[:, 0]
```

In [12]:
```
# Split the training set into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, rar
```

In [13]:
```
print(X_train.shape, X_val.shape, X_test.shape)
print(y_train.shape, y_val.shape, y_test.shape)
```

```
(48000, 28, 28, 1) (12000, 28, 28, 1) (10000, 28, 28, 1)
(48000,) (12000,) (10000,)
```

In [14]:

```
cnn_model = Sequential()
cnn_model.add(Conv2D(filters=64, kernel_size=(3,3), input_shape=(28,28,1), activation=
cnn_model.add(MaxPooling2D(pool_size = (2,2)))
cnn_model.add(Dropout(rate=0.3))
cnn_model.add(Flatten())
cnn_model.add(Dense(units=32, activation='relu'))
cnn_model.add(Dense(units=10, activation='sigmoid'))
```

In [15]:

```
cnn_model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', me
cnn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 13, 13, 64)        0         
                                                                 
 flatten (Flatten)           (None, 10816)             0         
                                                                 
 dense (Dense)               (None, 32)                346144    
                                                                 
 dense_1 (Dense)             (None, 10)                330       
                                                                 
=================================================================
Total params: 347,114
Trainable params: 347,114
Non-trainable params: 0
_________________________________________________________________
```

```
C:\Users\rohit\AppData\Roaming\Python\Python39\site-packages\keras\optimizers\optimiz
er_v2\adam.py:117: UserWarning: The `lr` argument is deprecated, use `learning_rate` 
instead.
  super().__init__(name, **kwargs)
```

In [16]:

```
cnn_model.fit(x=X_train, y=y_train, batch_size=512, epochs=5, validation_data=(X_val,
```

```
Epoch 1/5
94/94 [==============================] - 55s 581ms/step - loss: 0.7152 - accuracy: 0.
7566 - val_loss: 0.4528 - val_accuracy: 0.8416
Epoch 2/5
94/94 [==============================] - 76s 811ms/step - loss: 0.4289 - accuracy: 0.
8508 - val_loss: 0.3921 - val_accuracy: 0.8638
Epoch 3/5
94/94 [==============================] - 87s 922ms/step - loss: 0.3782 - accuracy: 0.
8679 - val_loss: 0.3650 - val_accuracy: 0.8706
Epoch 4/5
94/94 [==============================] - 121s 1s/step - loss: 0.3432 - accuracy: 0.87
94 - val_loss: 0.3257 - val_accuracy: 0.8867
Epoch 5/5
94/94 [==============================] - 128s 1s/step - loss: 0.3233 - accuracy: 0.88
70 - val_loss: 0.3158 - val_accuracy: 0.8914
```

Out[16]:

```
<keras.callbacks.History at 0x228000c75e0>
```

In [17]:

```
eval_result = cnn_model.evaluate(X_test, y_test)
print("Accuracy : {:.3f}".format(eval_result[1]))
```

```
313/313 [==============================] - 6s 18ms/step - loss: 0.3106 - accuracy: 0.
8956
Accuracy : 0.896
```

In [25]:

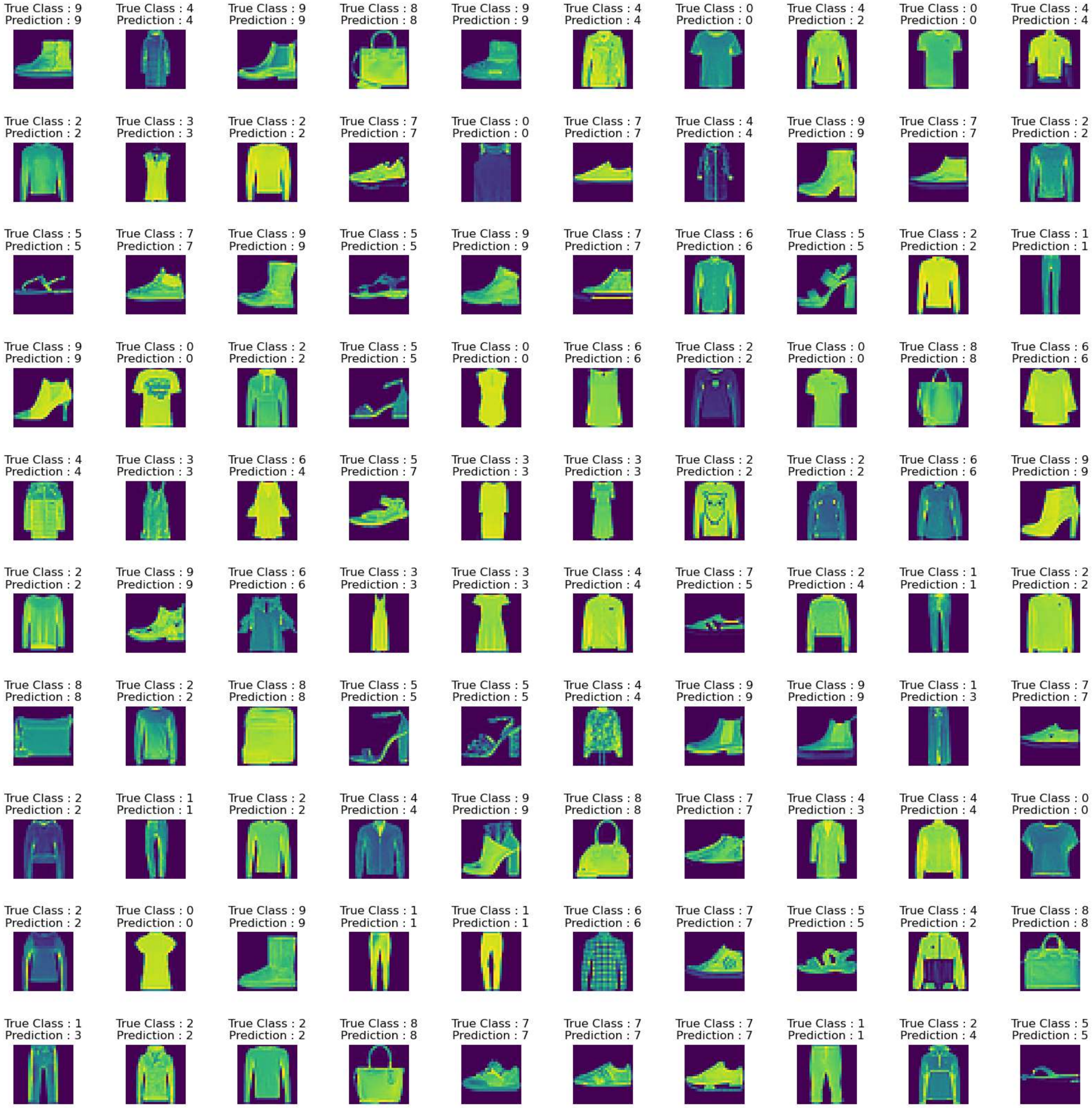
```
y_predict = np.argmax(cnn_model.predict(x=X_test), axis=-1)
```

```
313/313 [==============================] - 5s 14ms/step
```

In [26]:

```
height = 10
width = 10

fig, axes = plt.subplots(nrows=width, ncols=height, figsize=(20,20))
axes = axes.ravel()
for i in range(0, height*width):
    index = np.random.randint(len(y_predict))
    axes[i].imshow(X_test[index].reshape((28,28)))
    axes[i].set_title("True Class : {:0.0f}\nPrediction : {:d}".format(y_test[index],y
    axes[i].axis('off')
plt.subplots_adjust(hspace=0.9, wspace=0.5)
```

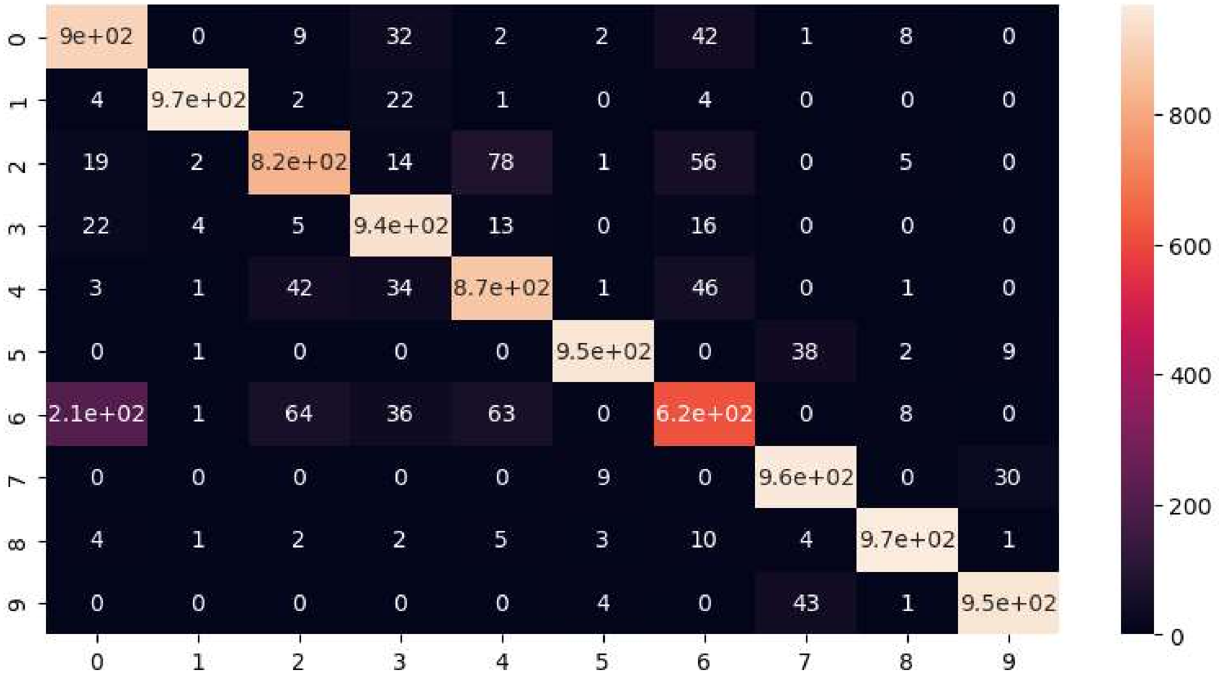


In [27]:

```
cm = confusion_matrix(y_test, y_predict)
plt.figure(figsize=(10,5))
sbn.heatmap(cm, annot=True)
```

Out[27]:

```
<AxesSubplot:>
```



In [29]:

```
num_classes = 10
class_names = ["class {}".format(i) for i in range(num_classes)]
cr = classification_report(y_test, y_predict, target_names=class_names)
print(cr)
```

```
              precision    recall  f1-score   support

     class 0       0.77      0.90      0.83      1000
     class 1       0.99      0.97      0.98      1000
     class 2       0.87      0.82      0.85      1000
     class 3       0.87      0.94      0.90      1000
     class 4       0.84      0.87      0.86      1000
     class 5       0.98      0.95      0.96      1000
     class 6       0.78      0.62      0.69      1000
     class 7       0.92      0.96      0.94      1000
     class 8       0.97      0.97      0.97      1000
     class 9       0.96      0.95      0.96      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.89     10000
weighted avg       0.90      0.90      0.89     10000
```

In [ ]: