

## # Submitted by Navneet Das 3433 Comp A

```
***  
Assignment 3-A Deep Learning  
Convolutional neural network (CNN)  
Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into different categories.  
***
```

```
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('DeepLearningData/fashion_minst/fashion-mnist_train.csv', sep=',')  
fashion_test_df = pd.read_csv('DeepLearningData/fashion_minst/fashion-mnist_test.csv', sep=',')
```

```
In [3]: fashion_train_df.shape # Shape of the dataset
```

```
Out[3]: (60000, 785)
```

```
In [4]: fashion_train_df.columns # Name of the columns of the DataSet.
```

```
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}
```

So we have 10 different labels. from 0 to 9.

Now let's find out what is the min and max of values of in the other columns.

```
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]
```

So we have 0 to 255 which is the color values for grayscale. 0 being white and 255 being black.

Now let's check some of the rows in tabular format

```
In [7]: fashion_train_df.head()
```

```
Out[7]:
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	2	0	0	0	0	0	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	0	0	0	0	0
2	6	0	0	0	0	0	0	0	5	0	...	0	0	0	30	43	0	0	0	0	0
3	0	0	0	0	1	2	0	0	0	0	...	3	0	0	0	0	1	0	0	0	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

So every other thing of the test dataset are going to be the same as the train dataset except the shape.

```
In [8]: fashion_test_df.shape
```

```
Out[8]: (10000, 785)
```

So here we have 10000 images instead of 60000 as in the train dataset.

Lets check first few rows.

```
In [9]: fashion_test_df.head()
```

Out[9]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	0	0	0	0	0	0	0	0	9	8	...	103	87	56	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	34	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	14	53	99	...	0	0	0	0	63	53	31	0	0
3	2	0	0	0	0	0	0	0	0	0	...	137	126	140	0	133	224	222	56	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

5 rows × 785 columns



```

In [10]: training = np.asarray(fashion_train_df, dtype='float32')

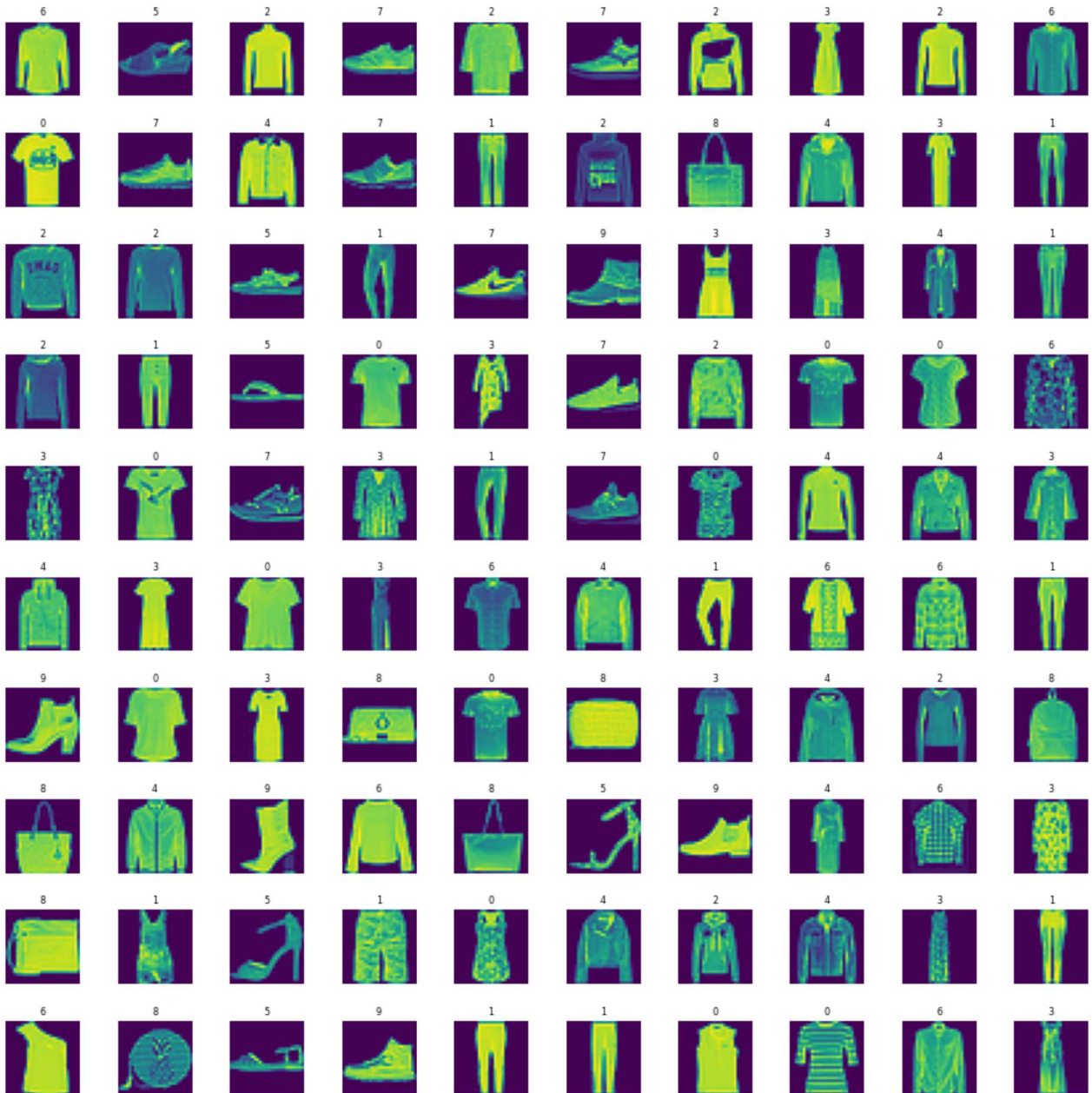
height = 10
width = 10

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]: training = np.asarray(fashion_train_df, dtype='float32')
X_train = training[:, 1:].reshape([-1,28,28,1])
X_train = X_train/255
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
y_test = testing[:, 0]
```

```
In [12]: X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=5) # TODO : change the random
```

```
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='relu'))
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'))
```

#### compile the model

```
In [15]: cnn_model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
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\navne\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:110: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

```
In [16]: from keras.utils.vis_utils import plot_model
plot_model(cnn_model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

You must install pydot (`pip install pydot`) and install graphviz (see instructions at <https://graphviz.gitlab.io/download/>) (<https://graphviz.gitlab.io/download/>) for plot\_model to work.

#### Train the model

```
In [17]: cnn_model.fit(x=X_train, y=y_train, batch_size=512, epochs=10, validation_data=(X_val, y_val))
```

```
Epoch 1/10
94/94 [=====] - 21s 222ms/step - loss: 0.7145 - accuracy: 0.7613 - val_loss: 0.4710 - val_accuracy: 0.8338
Epoch 2/10
94/94 [=====] - 20s 212ms/step - loss: 0.4167 - accuracy: 0.8553 - val_loss: 0.4016 - val_accuracy: 0.8612
Epoch 3/10
94/94 [=====] - 21s 224ms/step - loss: 0.3663 - accuracy: 0.8725 - val_loss: 0.3636 - val_accuracy: 0.8727
Epoch 4/10
94/94 [=====] - 21s 222ms/step - loss: 0.3304 - accuracy: 0.8847 - val_loss: 0.3369 - val_accuracy: 0.8813
Epoch 5/10
94/94 [=====] - 21s 224ms/step - loss: 0.3105 - accuracy: 0.8911 - val_loss: 0.3217 - val_accuracy: 0.8851
Epoch 6/10
94/94 [=====] - 22s 236ms/step - loss: 0.2949 - accuracy: 0.8974 - val_loss: 0.3209 - val_accuracy: 0.8887
Epoch 7/10
94/94 [=====] - 22s 236ms/step - loss: 0.2825 - accuracy: 0.9012 - val_loss: 0.2952 - val_accuracy: 0.8978
Epoch 8/10
94/94 [=====] - 22s 235ms/step - loss: 0.2736 - accuracy: 0.9039 - val_loss: 0.2961 - val_accuracy: 0.8955
Epoch 9/10
94/94 [=====] - 20s 210ms/step - loss: 0.2653 - accuracy: 0.9060 - val_loss: 0.2906 - val_accuracy: 0.8977
Epoch 10/10
94/94 [=====] - 21s 228ms/step - loss: 0.2565 - accuracy: 0.9097 - val_loss: 0.2969 - val_accuracy: 0.8956
```

```
Out[17]: <keras.callbacks.History at 0x181666ab730>
```

```
In [18]: eval_result = cnn_model.evaluate(X_test, y_test)
print("Accuracy :", (eval_result[1]))
```

```
313/313 [=====] - 1s 4ms/step - loss: 0.2806 - accuracy: 0.9013
Accuracy : 0.9013000130653381
```

```
In [19]: y_pred = cnn_model.predict(x=X_test)
```

```
313/313 [=====] - 1s 4ms/step
```

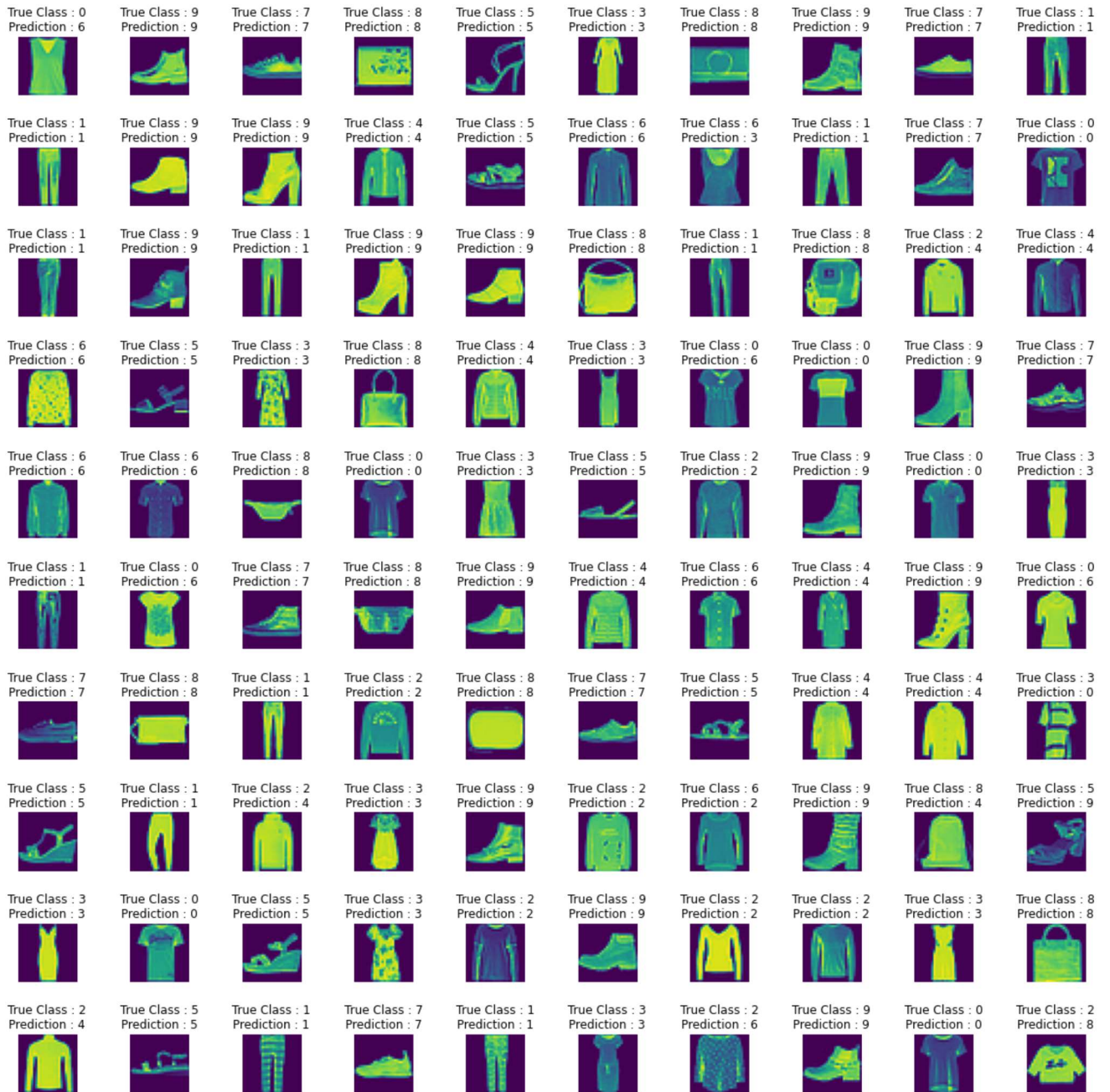
```
In [20]: y_pred[2].argmax()
```

```
Out[20]: 6
```

```
In [21]: y_pred = [pred.argmax() for pred in y_pred]
```

```
In [22]: 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_pred))
    axes[i].imshow(X_test[index].reshape((28,28)))
    axes[i].set_title("True Class : {:.0f}\nPrediction : {:.0f}".format(y_test[index],y_pred[index]))
    axes[i].axis('off')
plt.subplots_adjust(hspace=0.9, wspace=0.5)
```

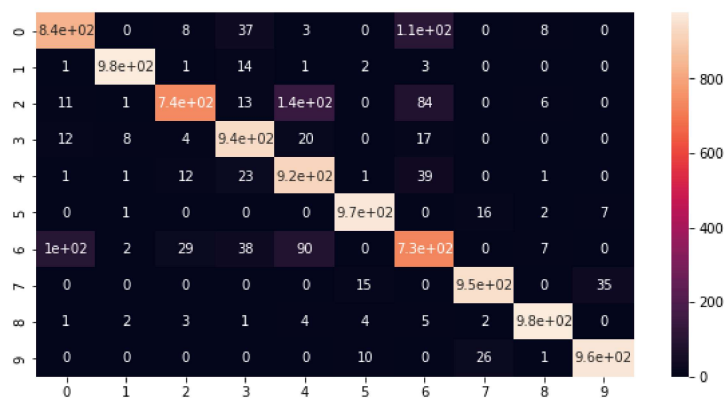


In [ ]:

**\*\*Plot Confusion Matrix\*\***

```
In [23]: cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10,5))
sbn.heatmap(cm, annot=True)
```

Out[23]: <AxesSubplot:>



### Classification Report

```
In [24]: num_classes = 10
class_names = ["class {}".format(i) for i in range(num_classes)]
cr = classification_report(y_test, y_pred, target_names=class_names)
print(cr)
```

	precision	recall	f1-score	support
class 0	0.87	0.83	0.85	1000
class 1	0.98	0.98	0.98	1000
class 2	0.93	0.74	0.82	1000
class 3	0.88	0.94	0.91	1000
class 4	0.78	0.92	0.84	1000
class 5	0.97	0.97	0.97	1000
class 6	0.74	0.73	0.74	1000
class 7	0.96	0.95	0.95	1000
class 8	0.98	0.98	0.98	1000
class 9	0.96	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