Name : SHEIK PAREETH

Fashion ANN

Deep learning

In [1]:

```
# Deep Learning libraries
import tensorflow as tf
```

In [2]:

```
# Basic Libraries
import numpy as np
import pandas as pd

# Visualization libraries
import matplotlib.pyplot as plt
import pydot
import seaborn as sns

#Evaluation library
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV

# Deep Learning libraries
import tensorflow as tf
from tensorflow.keras import layers
import keras
from keras.models import Sequential
from keras.layers.core import Dense,Activation,Dropout
from keras.datasets import mnist
from keras.utils.np_utils import to_categorical
from keras.wrappers.scikit_learn import KerasClassifier
```

In [3]:

```
# load the dataset
fashion_train=pd.read_csv("fashion-mnist_train.csv")
fashion_test=pd.read_csv("fashion-mnist_test.csv")
```

In [4]:

```
fashion_train.shape
```

Out[4]:

```
(60000, 785)
```

In [5]:

```
fashion_test.shape
```

Out[5]:

```
(10000, 785)
```

In [6]:

```
X_train_fashion = fashion_train.drop('label',axis = 1)
y_train_fashion = fashion_train['label']
X_test_fashion = fashion_test.drop('label',axis = 1)
y_test_fashion = fashion_test['label']
```

In [7]:

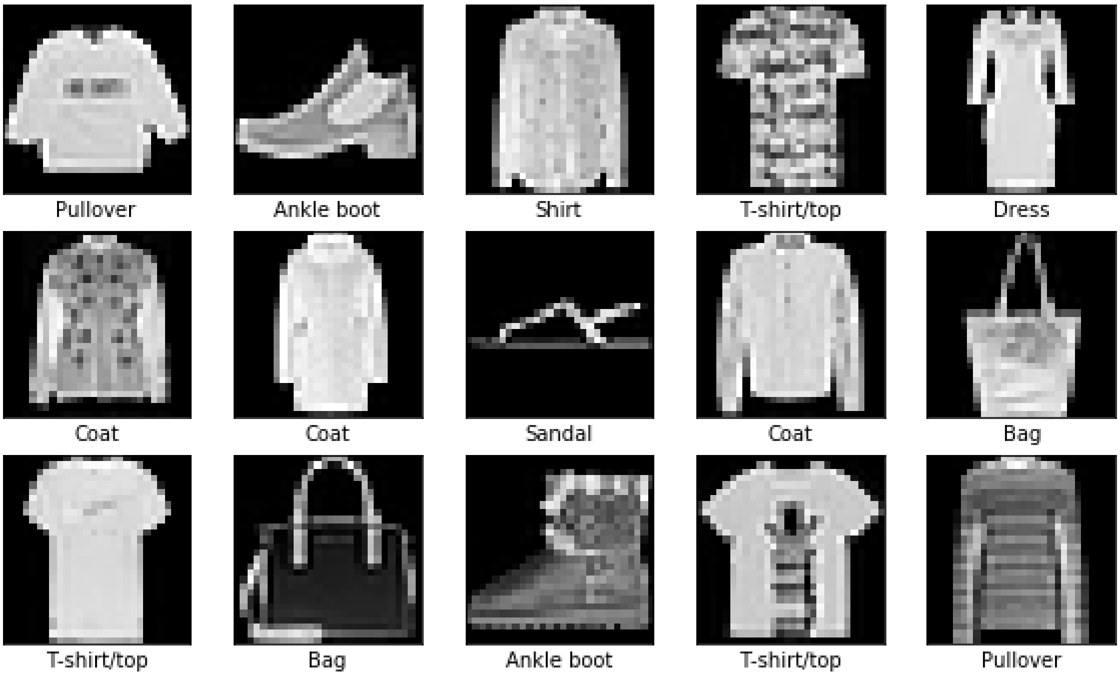
```
#Reshaping the dataset
x_train_reshape = X_train_fashion.values.reshape(-1,28,28)
x_test_reshape = X_test_fashion.values.reshape(-1,28,28)
```

In [8]:

```
#Names of clothing accessories in order
col_names = ['T-shirt/top','Trouser','Pullover','Dress','Coat','Sandal','Shirt','Sneaker','
```

In [10]:

```
#Visualizing the images
plt.figure(figsize=(10,10))
for i in range(15):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train_reshape[i], cmap='gray')
    plt.xlabel(col_names[y_train_fashion[i]])
plt.show()
```



In [11]:

```
y_train_fashion = to_categorical(y_train_fashion, num_classes=10)
y_test_fashion = to_categorical(y_test_fashion, num_classes=10)
```

In [12]:

```
#Creating base neural network
model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(784,)),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(24, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(24, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(10,activation='softmax'),])
```

In [13]:

```
#Compiling the model
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics = ['accuracy'])
```

In [14]:

```
#Fitting the model
history=model.fit(X_train_fashion, y_train_fashion, batch_size=100, epochs=30,validation_da
```

```
Epoch 1/30
600/600 [==============================] - 19s 17ms/step - loss: 1.1148 - ac
curacy: 0.6205 - val_loss: 0.5428 - val_accuracy: 0.8077
Epoch 2/30
600/600 [==============================] - 9s 14ms/step - loss: 0.7390 - acc
uracy: 0.7517 - val_loss: 0.4815 - val_accuracy: 0.8451
Epoch 3/30
600/600 [==============================] - 7s 11ms/step - loss: 0.6700 - acc
uracy: 0.7729 - val_loss: 0.4781 - val_accuracy: 0.8370
Epoch 4/30
600/600 [==============================] - 9s 16ms/step - loss: 0.6356 - acc
uracy: 0.7871 - val_loss: 0.4921 - val_accuracy: 0.8518
Epoch 5/30
600/600 [==============================] - 6s 10ms/step - loss: 0.6165 - acc
uracy: 0.7927 - val_loss: 0.4968 - val_accuracy: 0.8543
Epoch 6/30
600/600 [==============================] - 7s 12ms/step - loss: 0.5951 - acc
uracy: 0.8041 - val_loss: 0.4974 - val_accuracy: 0.8543
Epoch 7/30
600/600 [==============================] - 7s 11ms/step - loss: 0.5849 - acc
uracy: 0.8069 - val_loss: 0.5135 - val_accuracy: 0.8545
Epoch 8/30
600/600 [==============================] - 5s 8ms/step - loss: 0.5842 - accu
racy: 0.8069 - val_loss: 0.5040 - val_accuracy: 0.8600
Epoch 9/30
600/600 [==============================] - 5s 9ms/step - loss: 0.5740 - accu
racy: 0.8105 - val_loss: 0.5340 - val_accuracy: 0.8551
Epoch 10/30
600/600 [==============================] - 5s 8ms/step - loss: 0.5706 - accu
racy: 0.8120 - val_loss: 0.5135 - val_accuracy: 0.8630
Epoch 11/30
600/600 [==============================] - 7s 11ms/step - loss: 0.5712 - acc
uracy: 0.8131 - val_loss: 0.4892 - val_accuracy: 0.8670
Epoch 12/30
600/600 [==============================] - 7s 12ms/step - loss: 0.5549 - acc
uracy: 0.8169 - val_loss: 0.4873 - val_accuracy: 0.8684
Epoch 13/30
600/600 [==============================] - 8s 13ms/step - loss: 0.5445 - acc
uracy: 0.8218 - val_loss: 0.4806 - val_accuracy: 0.8644
Epoch 14/30
600/600 [==============================] - 7s 12ms/step - loss: 0.5406 - acc
uracy: 0.8234 - val_loss: 0.4938 - val_accuracy: 0.8684
Epoch 15/30
600/600 [==============================] - 8s 13ms/step - loss: 0.5369 - acc
uracy: 0.8240 - val_loss: 0.4555 - val_accuracy: 0.8695
Epoch 16/30
600/600 [==============================] - 7s 11ms/step - loss: 0.5316 - acc
uracy: 0.8247 - val_loss: 0.4792 - val_accuracy: 0.8664
Epoch 17/30
600/600 [==============================] - 6s 10ms/step - loss: 0.5250 - acc
uracy: 0.8273 - val_loss: 0.5000 - val_accuracy: 0.8592
Epoch 18/30
600/600 [==============================] - 6s 9ms/step - loss: 0.5260 - accu
racy: 0.8256 - val_loss: 0.4433 - val_accuracy: 0.8717
Epoch 19/30
600/600 [==============================] - 6s 10ms/step - loss: 0.5296 - acc
```

```
uracy: 0.8239 - val_loss: 0.4470 - val_accuracy: 0.8674
Epoch 20/30
600/600 [==============================] - 6s 11ms/step - loss: 0.5248 - acc
uracy: 0.8270 - val_loss: 0.4440 - val_accuracy: 0.8700
Epoch 21/30
600/600 [==============================] - 7s 11ms/step - loss: 0.5289 - acc
uracy: 0.8268 - val_loss: 0.4261 - val_accuracy: 0.8690
Epoch 22/30
600/600 [==============================] - 9s 14ms/step - loss: 0.5152 - acc
uracy: 0.8292 - val_loss: 0.4341 - val_accuracy: 0.8665
Epoch 23/30
600/600 [==============================] - 6s 9ms/step - loss: 0.5107 - accu
racy: 0.8299 - val_loss: 0.4290 - val_accuracy: 0.8619
Epoch 24/30
600/600 [==============================] - 8s 14ms/step - loss: 0.5134 - acc
uracy: 0.8310 - val_loss: 0.3950 - val_accuracy: 0.8755
Epoch 25/30
600/600 [==============================] - 7s 12ms/step - loss: 0.5073 - acc
uracy: 0.8310 - val_loss: 0.4117 - val_accuracy: 0.8732
Epoch 26/30
600/600 [==============================] - 7s 12ms/step - loss: 0.5063 - acc
uracy: 0.8326 - val_loss: 0.3954 - val_accuracy: 0.8728
Epoch 27/30
600/600 [==============================] - 6s 9ms/step - loss: 0.4996 - accu
racy: 0.8342 - val_loss: 0.4092 - val_accuracy: 0.8736
Epoch 28/30
600/600 [==============================] - 7s 12ms/step - loss: 0.4995 - acc
uracy: 0.8356 - val_loss: 0.3998 - val_accuracy: 0.8743
Epoch 29/30
600/600 [==============================] - 5s 9ms/step - loss: 0.5003 - accu
racy: 0.8352 - val_loss: 0.4010 - val_accuracy: 0.8717
Epoch 30/30
600/600 [==============================] - 5s 8ms/step - loss: 0.4990 - accu
racy: 0.8325 - val_loss: 0.3806 - val_accuracy: 0.8698
```

In [15]:

```
test_loss_fashion, test_acc_fashion = model.evaluate(X_test_fashion, y_test_fashion)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.3806 - accu
racy: 0.8698
```

In [16]:

```
print('Fashion MNIST Test accuracy:', round(test_acc_fashion,4))
```

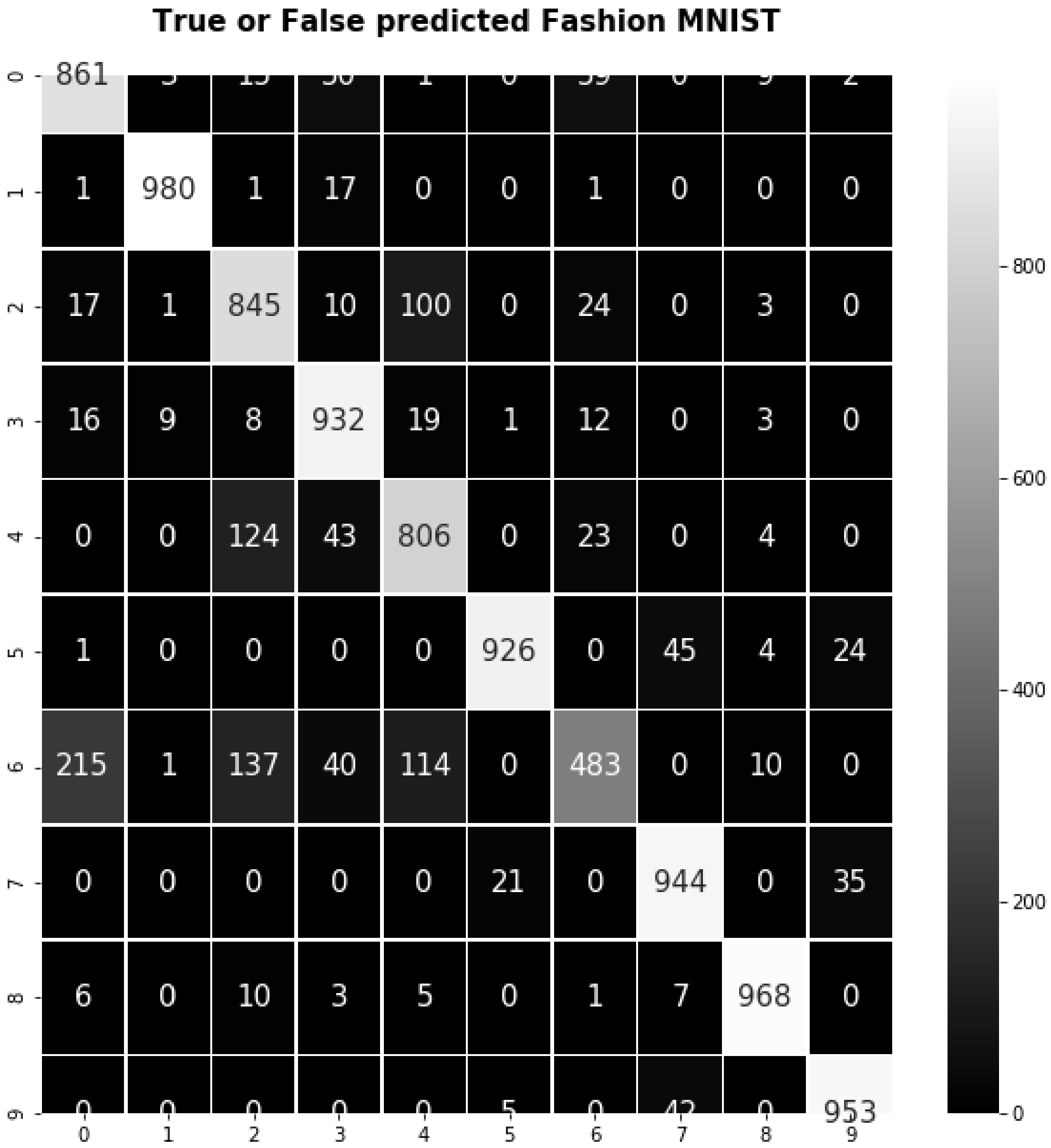
```
Fashion MNIST Test accuracy: 0.8698
```

In [17]:

```
#Predicting the labels-Fashion
y_predict_fash = model.predict(X_test_fashion)
y_predict_fash=np.argmax(y_predict_fash, axis=1)
y_test_fash_eval=np.argmax(y_test_fashion, axis=1)
```

In [18]:

```
con_mat=confusion_matrix(y_test_fash_eval,y_predict_fash)
plt.style.use('seaborn-deep')
plt.figure(figsize=(10,10))
sns.heatmap(con_mat,annot=True,annot_kws={'size': 15},linewidths=0.5,fmt="d",cmap="gray")
plt.title('True or False predicted Fashion MNIST\n',fontweight='bold',fontsize=15)
plt.show()
```



In [19]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test_fash_eval,y_predict_fash))
```

```
              precision    recall  f1-score   support

           0       0.77      0.86      0.81      1000
           1       0.99      0.98      0.98      1000
           2       0.74      0.84      0.79      1000
           3       0.85      0.93      0.89      1000
           4       0.77      0.81      0.79      1000
           5       0.97      0.93      0.95      1000
           6       0.80      0.48      0.60      1000
           7       0.91      0.94      0.93      1000
           8       0.97      0.97      0.97      1000
           9       0.94      0.95      0.95      1000

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000
```

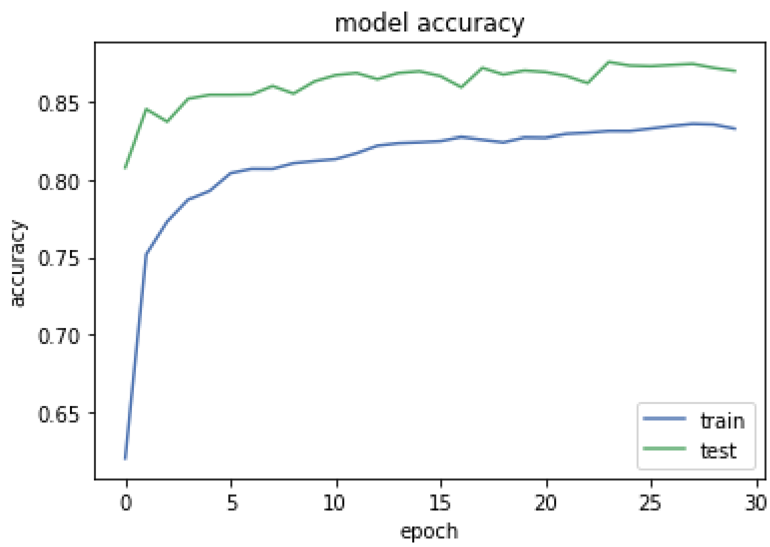
In [20]:

```
print(history.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

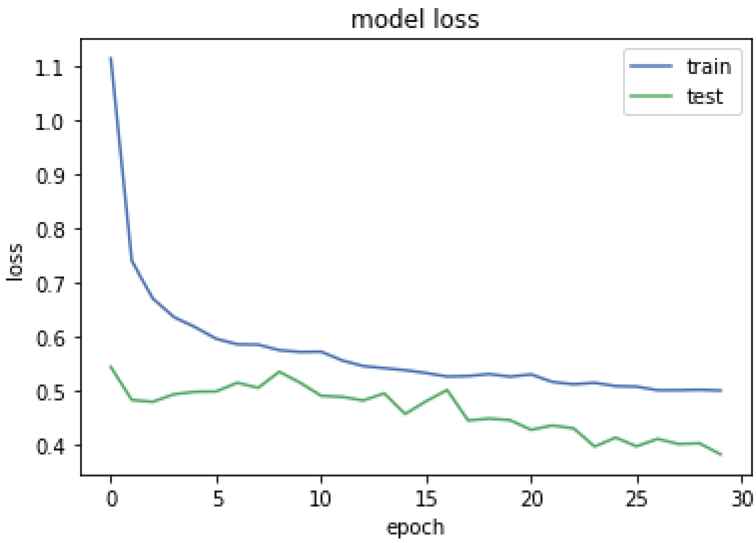
In [21]:

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
```



In [22]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
```



In [23]:

```
#tf.expand_dims(X_test_digit[0])
y_predict = model.predict(X_test_fashion.loc[[0],:].values)
y_predict=np.argmax(y_predict, axis=1) # Here we get the index of maximum value in the enco
y_test_digit_eval=np.argmax(y_test_fashion, axis=1)
```

In [28]:

```
y_predict[0]
```

Out[28]:

6

In [25]:

```
#Names of clothing accessories in order
col_names = ['T-shirt/top','Trouser','Pullover','Dress','Coat','Sandal','Shirt','Sneaker','
```

In [29]:

```
#Visualizing the images
plt.figure(figsize=(10,10))
plt.imshow(x_train_reshape[0], cmap='gray')
plt.xlabel("Actual:{},Pred:{}".format(col_names[np.argmax(y_test_fashion[0])],col_names[y_p
plt.show()
```