LAB6 : Multi-class Classification of Fashion Apparels using DNN

NAME : ANNAPOORNIMA S

225229101

Steps

1.Open fashion_mnist dataset from keras

In [2]:
```
import tensorflow as tf
import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten
```

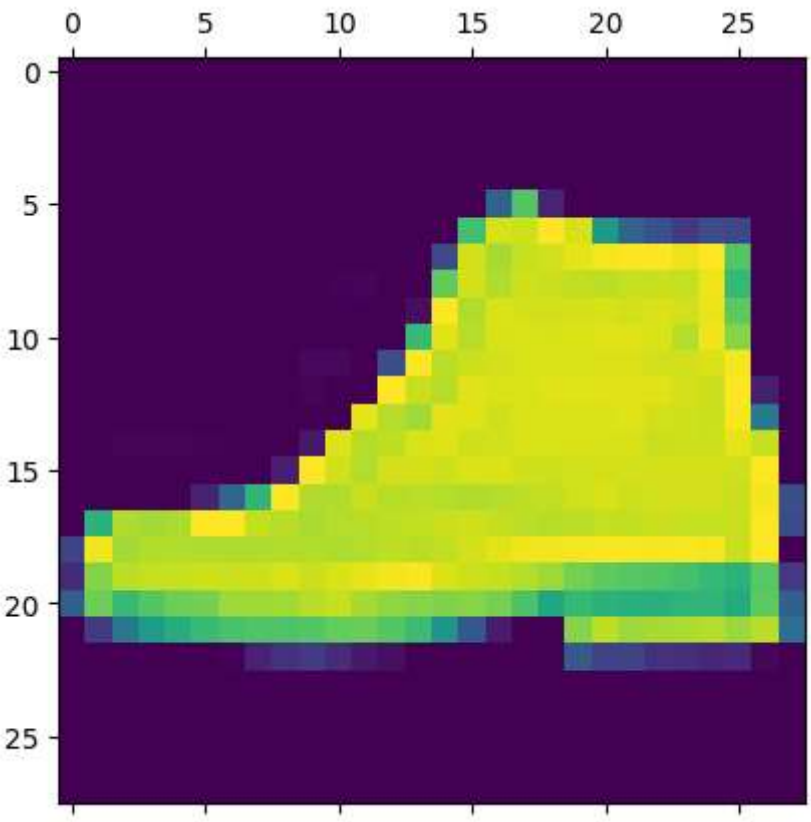
In [3]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data(
```

2. Perform basic EDA

In [4]:
```
print('x_train shape: ',x_train.shape,'       ''x_train size: ',x_train.size)
print('y_train shape: ',y_train.shape,'             ''y_train size: ',y_train.si
```

```
x_train shape:  (60000, 28, 28)        x_train size:  47040000
y_train shape:  (60000,)               y_train size:  60000
```

In [11]:
```
plt.matshow(x_train[42])
plt.show()
```



3.Normalize

In [5]:
```
X_train = x_train.astype('float32')/255
X_test = x_test.astype('float32')/255
```

In [6]:
```
X_train[45]
```

Out[6]:
```
array([[0.        , 0.        , 0.        , 0.00392157, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.13333334,
        0.5803922 , 0.7921569 , 0.41960785, 0.32941177, 0.63529414,
        0.7764706 , 0.3529412 , 0.10196079, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.08235294, 0.41960785, 0.6509804 , 0.65882355,
        0.59607846, 0.8       , 0.93333334, 0.972549  , 0.92941177,
        0.74509805, 0.5568628 , 0.69803923, 0.61960787, 0.38431373,
        0.06666667, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.00392157, 0.        ,
        0.21568628, 0.7372549 , 0.54901963, 0.5137255 , 0.4745098 ,
        0.46666667, 0.45490196, 0.57254905, 0.6156863 , 0.5176471 ,
        0.4509804 , 0.53333336, 0.49019608, 0.54901963, 0.56078434,
        0.70980394, 0.2       , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
```

4.Build a simple baseline model

In [7]:
```
model=Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(512,input_dim=1,activation='relu'))
model.add(Dense(10,activation='softmax'))

model.compile(loss='mean_squared_error',
              optimizer='RMSprop',
              metrics=['accuracy'])
```

In [8]:
```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 27.6105 - acc
uracy: 0.1006
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - accu
racy: 0.1010
Epoch 3/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0985
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.1010
Epoch 5/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0989
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0997
Epoch 7/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0995
Epoch 8/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0989
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0992
Epoch 10/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0980
```

Out[8]:
```
<keras.callbacks.History at 0x22ce424de80>
```

In [9]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 512)               401920

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

5. Performance Analysis

In [10]:
```
model=Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))

model.compile(loss='mean_squared_error',
              optimizer='RMSprop',
              metrics='accuracy')
```

In [11]:
```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - accu
racy: 0.0991
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 27.6101 - accu
racy: 0.0977
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss: 27.6101 - accu
racy: 0.1001
Epoch 4/10
1875/1875 [==============================] - 3s 1ms/step - loss: 27.6101 - accu
racy: 0.0986
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 27.6101 - accu
racy: 0.0989
Epoch 6/10
1875/1875 [==============================] - 2s 1ms/step - loss: 27.6101 - accu
racy: 0.0996
Epoch 7/10
1875/1875 [==============================] - 2s 1ms/step - loss: 27.6101 - accu
racy: 0.0997
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 27.6101 - accu
racy: 0.0994
Epoch 9/10
1875/1875 [==============================] - 3s 1ms/step - loss: 27.6101 - accu
racy: 0.0997
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 27.6101 - accu
racy: 0.1001
```

Out[11]:
```
<keras.callbacks.History at 0x22ce4f3b730>
```

In [12]:
```
model=Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(256,input_dim=1,activation='relu'))
model.add(Dense(256,input_dim=1,activation='relu'))
model.add(Dense(10,activation='softmax'))

model.compile(loss='mean_squared_error',
              optimizer='RMSprop',
              metrics=['accuracy'])
```

In [13]:
```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 5s 2ms/step - loss: 27.6101 - accu
racy: 0.1011
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - accu
racy: 0.0988
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - accu
racy: 0.1002
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - accu
racy: 0.0996
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6100 - accu
racy: 0.1003
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - accu
racy: 0.0981
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - accu
racy: 0.0996
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - accu
racy: 0.0988
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - accu
racy: 0.0982
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6102 - accu
racy: 0.0985
```

Out[13]:
```
<keras.callbacks.History at 0x22ce4e95e80>
```

In [14]:
```
model=Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(512,input_dim=1,activation='relu'))
model.add(Dense(512,input_dim=1,activation='relu'))
model.add(Dense(10,activation='softmax'))

model.compile(loss='mean_squared_error',
              optimizer='RMSprop',
              metrics=['accuracy'])
```

In [15]:
```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6103 - acc
uracy: 0.0983
Epoch 2/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - acc
uracy: 0.0997
Epoch 3/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - acc
uracy: 0.1007
Epoch 4/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - acc
uracy: 0.0994
Epoch 5/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - acc
uracy: 0.1008
Epoch 6/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - acc
uracy: 0.1004
Epoch 7/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - acc
uracy: 0.0984
Epoch 8/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - acc
uracy: 0.1018
Epoch 9/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - acc
uracy: 0.0971
Epoch 10/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - acc
uracy: 0.0985
```

Out[15]:
```
<keras.callbacks.History at 0x22ce5123700>
```

In [ ]:
```
```

In [16]:
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2
```

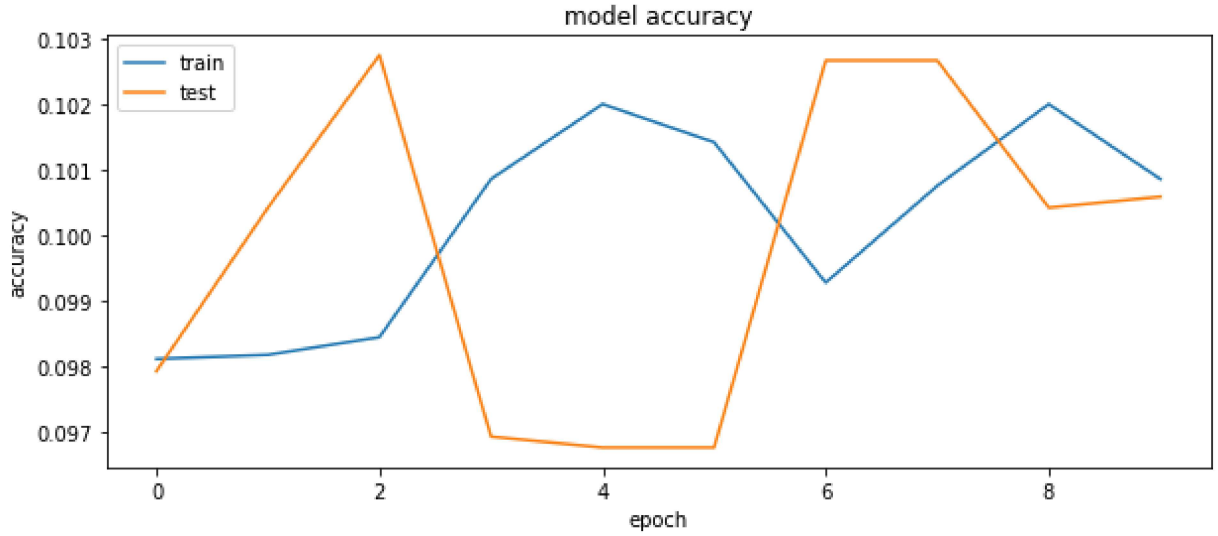
In [17]:
```
history = model.fit(X_train,y_train,epochs=10,validation_data=(X_val, y_val))
```

```
Epoch 1/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.0981 - val_loss: 27.7079 - val_accuracy: 0.0979
Epoch 2/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.0982 - val_loss: 27.7079 - val_accuracy: 0.1004
Epoch 3/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.0984 - val_loss: 27.7079 - val_accuracy: 0.1028
Epoch 4/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.1009 - val_loss: 27.7079 - val_accuracy: 0.0969
Epoch 5/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.1020 - val_loss: 27.7079 - val_accuracy: 0.0967
Epoch 6/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.1014 - val_loss: 27.7079 - val_accuracy: 0.0967
Epoch 7/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.0993 - val_loss: 27.7079 - val_accuracy: 0.1027
Epoch 8/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.1007 - val_loss: 27.7079 - val_accuracy: 0.1027
Epoch 9/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.1020 - val_loss: 27.7079 - val_accuracy: 0.1004
Epoch 10/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - acc
uracy: 0.1009 - val_loss: 27.7079 - val_accuracy: 0.1006
```

In [18]:
```
print(history.history.keys())


figure(figsize=(10, 4))
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='upper left')
plt.show()
```

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



In [ ]:
```
```