# NAME : ASHIKA.C

# ROLL NO : 225229105

# LAB - 6 MULTI-CLASS CLASSIFICATION OF FASHION APPARELS USING DNN

### STEPS

### *1 . OPEN*

In [1]:
```
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
```

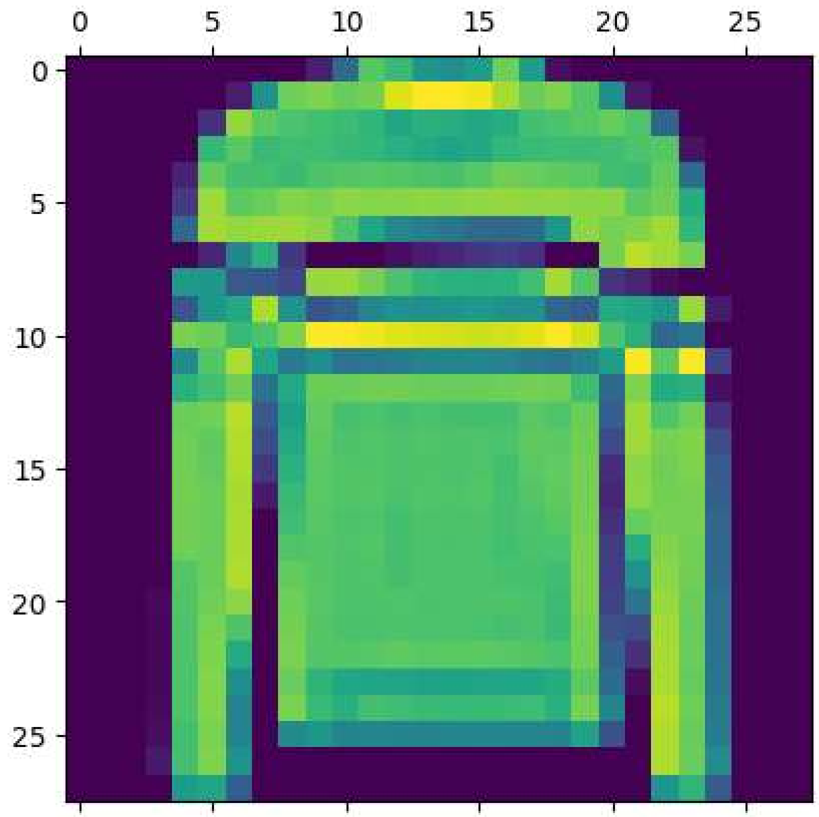
In [2]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_dat
```

### *2 . PERFORM BASIC EXPLORATORY DATA ANALYSIS (EDA)*

In [3]:
```
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.size)
```

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

In [22]:
```
plt.matshow(x_train[5])
plt.show()
```




### *3 . NORMALIZE*

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

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

Out[6]:
```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 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 [==============================] - 8s 3ms/step - loss: 27.6104 - ac
curacy: 0.1018
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.0991
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.0995
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.1006
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.1006
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.1004
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.0989
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.0996
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.1020
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.1016
```

Out[8]:
```
<keras.callbacks.History at 0x1d160fc1270>
```

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 - ac
curacy: 0.1007
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.0996
Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.0994
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.0975
Epoch 5/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.0973
Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.1014
Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.1011
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.1020
Epoch 9/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.1024
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 27.6101 - ac
curacy: 0.1015
```

Out[11]:
```
<keras.callbacks.History at 0x1d1619d50f0>
```

In [12]:
```
model=Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(256,activation='relu'))
model.add(Dense(256,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 [==============================] - 6s 3ms/step - loss: 27.6101 - ac
curacy: 0.0998
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.0996
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.0990
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.1012
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.0983
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.0997
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.1012
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.1009
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.0989
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 27.6101 - ac
curacy: 0.0993
```

Out[13]:
```
<keras.callbacks.History at 0x1d1625831c0>
```

In [14]:
```
model=Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(512,activation='relu'))
model.add(Dense(512,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 [==============================] - 10s 5ms/step - loss: 27.6104 - a
ccuracy: 0.1002
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0994
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.1008
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0993
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.1004
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0995
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.1002
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0997
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0999
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.1012
```

Out[15]:
```
<keras.callbacks.History at 0x1d162987490>
```

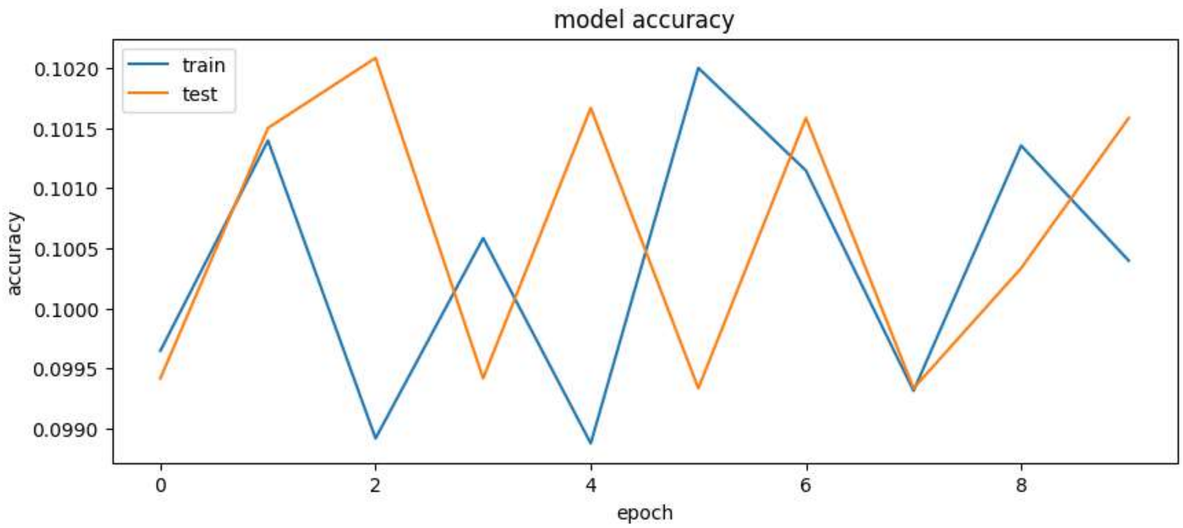
In [16]:
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0
```

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

```
Epoch 1/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.0996 - val_loss: 27.5198 - val_accuracy: 0.0994
Epoch 2/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.6326 - ac
curacy: 0.1014 - val_loss: 27.5198 - val_accuracy: 0.1015
Epoch 3/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.0989 - val_loss: 27.5198 - val_accuracy: 0.1021
Epoch 4/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.1006 - val_loss: 27.5198 - val_accuracy: 0.0994
Epoch 5/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.0989 - val_loss: 27.5198 - val_accuracy: 0.1017
Epoch 6/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.1020 - val_loss: 27.5198 - val_accuracy: 0.0993
Epoch 7/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.1011 - val_loss: 27.5198 - val_accuracy: 0.1016
Epoch 8/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.0993 - val_loss: 27.5198 - val_accuracy: 0.0993
Epoch 9/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.1014 - val_loss: 27.5198 - val_accuracy: 0.1003
Epoch 10/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.6326 - ac
curacy: 0.1004 - val_loss: 27.5198 - val_accuracy: 0.1016
```

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 [19]:
```
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('epoch')
plt.xlabel('accuracy')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

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