NAME:R.RAHINI ROLL NO:215229128

LAB 5:MULTI-CLASS CLASSIFICATION OF FASHION APPARELS USING DNN

STEPS

1.open data set

In [1]:

```
import tensorflow as tf
import keras
```

In [2]:

```
fmnist=tf.keras.datasets.fashion_mnist.load_data()
fmnist
```

Out[2]:

```
((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, 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, 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]]], dtype=uint8),
  array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)),
 (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, 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, 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]]], dtype=uint8),
  array([9, 2, 1, ..., 8, 1, 5], dtype=uint8)))
```

2.Perform basic EDA

In [3]:

```
(x_train,y_train),(x_test,y_test) = fmnist
```

In [4]:

```
print("x_train shape:", x_train.shape, "\ny_train shape:", y_train.shape,
      "\nx_test shape:",x_test.shape,"\ny_test shape:",y_test.shape)
```

```
x_train shape: (60000, 28, 28)
y_train shape: (60000,)
x_test shape: (10000, 28, 28)
y_test shape: (10000,)
```

In [5]:

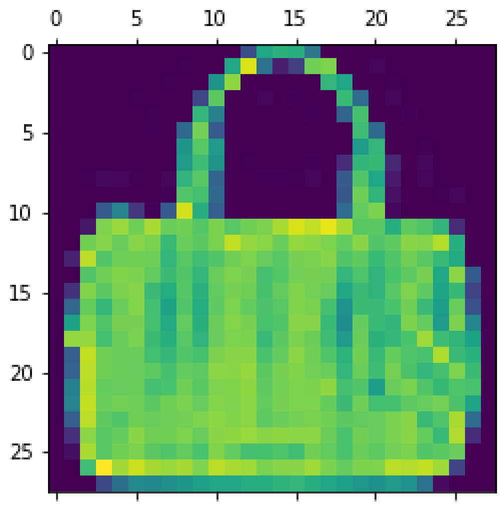
```
import matplotlib.pyplot as plt
```

In [6]:

```
plt.matshow(x_train[4000])
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x1f66d32e790>
```



Step-3:Normalize

In [7]:

```
X_train=x_train.astype('float32') /255
X_test=x_test.astype('float32') /255
```

In [8]:

```
X_train.size
```

Out[8]:

```
47040000
```

In [9]:

```
X_test.size
```

Out[9]:

```
7840000
```

Step-4:Build a simple baseline model

In [10]:

```
from keras.models import Sequential
from keras.layers.core import Dense,Flatten
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(512,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

In [11]:

```
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
_________________________________________________________________
```

In [12]:

```
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 [==============================] - 60s 28ms/step - loss: 27.6106
- accuracy: 0.1021
Epoch 2/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101
- accuracy: 0.1020
Epoch 3/10
1875/1875 [==============================] - 53s 28ms/step - loss: 27.6101
- accuracy: 0.1005
Epoch 4/10
1875/1875 [==============================] - 54s 29ms/step - loss: 27.6101
- accuracy: 0.1002
Epoch 5/10
1875/1875 [==============================] - 44s 24ms/step - loss: 27.6101
- accuracy: 0.1003
Epoch 6/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101
- accuracy: 0.1018
Epoch 7/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101
- accuracy: 0.0998
Epoch 8/10
1875/1875 [==============================] - 43s 23ms/step - loss: 27.6101
- accuracy: 0.1010
Epoch 9/10
1875/1875 [==============================] - 47s 25ms/step - loss: 27.6101
- accuracy: 0.0997
Epoch 10/10
1875/1875 [==============================] - 40s 22ms/step - loss: 27.6101
- accuracy: 0.0993
```

Out[13]:

```
<keras.callbacks.History at 0x1f66d59edf0>
```

In [22]:

```
model.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 3s 9ms/step - loss: 27.6100 - a
ccuracy: 0.0999
```

Out[22]:

```
[27.609987258911133, 0.09989999979734421]
```

5. Performance Analysis

In [23]:

```
model1=Sequential()
model1.add(Flatten(input_shape=(28, 28)))
model1.add(Dense(128,activation='relu'))
model1.add(Dense(128,activation='relu'))
model1.add(Dense(10,activation='softmax'))
model1.compile(loss='mean_squared_error',optimizer='RMSprop',
              metrics='accuracy')
```

In [24]:

```
model1.fit(x_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 15s 9ms/step - loss: 27.7225
- accuracy: 0.1006
Epoch 2/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 3/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 4/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 5/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 6/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 7/10
1500/1500 [==============================] - 11s 7ms/step - loss: 27.7226
- accuracy: 0.1003
Epoch 8/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 9/10
1500/1500 [==============================] - 11s 7ms/step - loss: 27.7225
- accuracy: 0.1003
Epoch 10/10
1500/1500 [==============================] - 10s 7ms/step - loss: 27.7225
- accuracy: 0.1003
```

Out[24]:

```
<keras.callbacks.History at 0x1f66dd90c70>
```

In [25]:

```
model2=Sequential()
model2.add(Flatten(input_shape=(28, 28)))
model2.add(Dense(512,input_dim=1,activation='relu'))
model2.add(Dense(512,input_dim=1,activation='relu'))
model2.add(Dense(10,activation='softmax'))
model2.compile(loss='mean_squared_error',
             optimizer='RMSprop',
             metrics=['accuracy'])
```

In [26]:

```
model2.fit(x_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 57s 36ms/step - loss: 27.7226
- accuracy: 0.0997
Epoch 2/10
1500/1500 [==============================] - 53s 35ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 3/10
1500/1500 [==============================] - 55s 37ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 4/10
1500/1500 [==============================] - 56s 38ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 5/10
1500/1500 [==============================] - 56s 37ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 6/10
1500/1500 [==============================] - 55s 36ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 7/10
1500/1500 [==============================] - 54s 36ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 8/10
1500/1500 [==============================] - 54s 36ms/step - loss: 27.7225
- accuracy: 0.0998
Epoch 9/10
1500/1500 [==============================] - 54s 36ms/step - loss: 27.7226
- accuracy: 0.0998
Epoch 10/10
1500/1500 [==============================] - 51s 34ms/step - loss: 27.7225
- accuracy: 0.0998
```

Out[26]:

```
<keras.callbacks.History at 0x1f6790a3310>
```

In [27]:

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, rand
```

In [28]:

```
history = model.fit(x_train,y_train,epochs=10,validation_data=(x_val, y_val))
```

```
Epoch 1/10
1200/1200 [==============================] - 28s 23ms/step - loss: 27.5287
- accuracy: 0.1013 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 2/10
1200/1200 [==============================] - 31s 26ms/step - loss: 27.5288
- accuracy: 0.1001 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 3/10
1200/1200 [==============================] - 33s 27ms/step - loss: 27.5287
- accuracy: 0.1015 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 4/10
1200/1200 [==============================] - 29s 24ms/step - loss: 27.5287
- accuracy: 0.1001 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 5/10
1200/1200 [==============================] - 32s 27ms/step - loss: 27.5287
- accuracy: 0.0998 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 6/10
1200/1200 [==============================] - 32s 27ms/step - loss: 27.5287
- accuracy: 0.0998 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 7/10
1200/1200 [==============================] - 32s 27ms/step - loss: 27.5288
- accuracy: 0.0991 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 8/10
1200/1200 [==============================] - 32s 27ms/step - loss: 27.5288
- accuracy: 0.1015 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 9/10
1200/1200 [==============================] - 32s 26ms/step - loss: 27.5287
- accuracy: 0.1009 - val_loss: 28.0479 - val_accuracy: 0.1001
Epoch 10/10
1200/1200 [==============================] - 32s 27ms/step - loss: 27.5287
- accuracy: 0.0978 - val_loss: 28.0479 - val_accuracy: 0.1001
```

In [29]:

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
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'])
```