Lab 6_Multi-class Classification of Fashion Apparels using DNN

Name:P.Asha Belcilda

Rollno:225229104

Steps

1. Open fashion_mnist dataset from keras

In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
C:\Users\ashac\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarnin
g: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:

```python
dataset = tf.keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/train-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf
-keras-datasets/train-labels-idx1-ubyte.gz)
29515/29515 [==============================] - 0s 4us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/train-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf
-keras-datasets/train-images-idx3-ubyte.gz)
26421880/26421880 [==============================] - 6s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/t10k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz)
5148/5148 [==============================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/t10k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz)
4422102/4422102 [==============================] - 1s 0us/step
```

2.Perform basic Exploratory Data Analysis(EDA)

In [4]:

```
(X_train,y_train),(X_test,y_test)=dataset
```

In [5]:

```
print("X_train shape:",X_train.shape)
print("y_train shape:",y_train.shape)
print("X_test shape:",X_test.shape)
print("y_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 [6]:

```
print("X_train size:",X_train.size)
print("y_train size:", y_train.size)
print("X_test size:",X_test.size)
print("y_test size:",y_test.size)
```

```
X_train size: 47040000
y_train size: 60000
X_test size: 7840000
y_test size: 10000
```

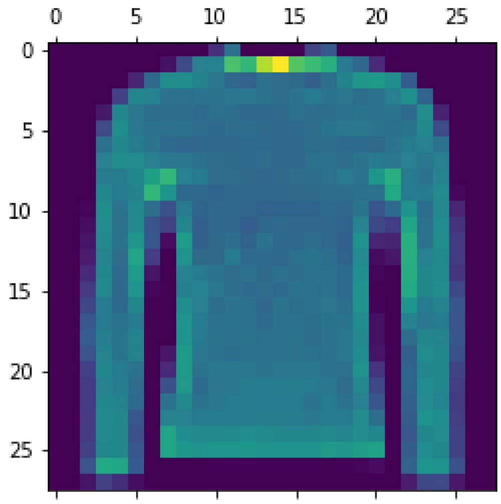
```
In [7]: X_train[37]
```

```
Out[7]: array([[  0,   0,   1,   1,   0,   1,   0,   0,   0,   0,  34,  95,   0,
                  0,   0,   0,  52,  70,   0,   0,   0,   2,   1,   0,   1,   0,
                  0,   0],
               [  0,   0,   0,   1,   1,   0,   0,  11,  59, 111, 113, 182, 169,
                226, 255, 188, 175, 162, 105,  85,  31,   0,   0,   0,   1,   0,
                  0,   0],
               [  0,   0,   1,   0,   0,  27,  89, 127, 127, 115, 101,  86,  81,
                 95,  91,  88,  78,  92, 115, 136, 139, 126,  73,   1,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,  49, 117, 113,  95,  94,  97,  98, 102, 101,
                 98,  91,  97, 104,  97, 101,  92,  95, 111, 128,  82,   1,   0,
                  0,   0],
               [  0,   0,   0,  18, 118, 102,  92,  92,  92,  89,  94,  92,  86,
                 85,  88,  94,  92,  92,  95,  99,  95,  98,  89, 126,  24,   0,
                  0,   0],
               [  0,   0,   0,  59, 127, 102,  95,  94,  97,  91,  86,  91,  92,
                 88,  86,  89,  91,  99, 102, 101,  98,  94,  99, 121,  57,   0,
                  0,   0],
               [  0,   0,   0,  95, 118, 107,  98,  89,  84,  86,  86,  86,  89,
                 89,  85,  85,  92,  92,  89,  89,  88,  97, 107, 111,  97,   0,
                  0,   0],
               [  0,   0,   0, 111, 126, 123, 111, 102, 102,  94,  91,  88,  89,
                 91,  86,  86,  95,  97,  91,  98, 104, 102, 111, 102, 111,   0,
                  0,   0],
               [  0,   0,   0, 108, 107, 117, 146, 169, 111, 105,  91,  91,  88,
                 84,  88,  91,  92,  94, 105,  97, 136, 162, 104,  97, 114,   0,
                  0,   0],
               [  0,   0,   1, 118, 104, 114, 169, 130,  85,  86,  82,  85,  85,
                 85,  86,  88,  88,  92,  92,  94,  95, 155, 104, 104, 123,   5,
                  0,   0],
               [  0,   0,   4, 126,  97, 120,  92,  63, 104,  82,  86,  86,  84,
                 81,  82,  82,  82,  89,  84,  99,  65,  89, 130,  92, 120,  27,
                  0,   0],
               [  0,   0,  10, 123,  95, 121,  66,  52, 117,  78,  86,  84,  76,
                 86,  88,  84,  84,  85,  81, 113,  55,  47, 149,  94, 124,  37,
                  0,   0],
               [  0,   0,  14, 121,  98, 136,  70,   7, 140,  79,  88,  92,  81,
                 97,  85,  85,  91,  85,  79, 130,  27,  24, 160,  98, 127,  43,
                  0,   0],
               [  0,   0,  20, 115,  91, 149,  46,   0, 130,  88,  88, 105,  89,
                 89,  85,  94,  99,  92,  82, 123,   8,  15, 160, 111, 121,  43,
                  0,   0],
               [  0,   0,  31, 118,  89, 140,  13,   0, 113, 105,  97,  91,  94,
                 88,  94,  92,  97,  95,  89, 128,   4,   0, 159, 118, 121,  39,
                  0,   0],
               [  0,   0,  42, 120,  86, 133,   4,   0, 110, 113, 101,  92,  89,
                 91,  89,  97,  94,  97,  89, 131,   0,   0, 155, 117, 126,  55,
                  0,   0],
               [  0,   0,  55, 104,  86, 117,   1,   0, 127, 111,  92,  97,  94,
                 84,  98,  92,  95, 101,  92, 130,   2,   0, 134, 104, 114,  68,
                  0,   0],
               [  0,   0,  70, 104,  97, 105,   0,   0, 130, 105,  95,  97,  95,
                 89,  97,  95,  92,  99,  98, 124,  11,   0, 113, 101, 108,  66,
                  0,   0],
               [  0,   0,  76, 110,  99, 115,   0,   1, 131, 104,  98,  95,  98,
                 94,  95,  98,  91,  97, 102, 120,  11,   0,  91, 118, 114,  68,
                  0,   0],
       [  0,   0,  65, 115,  99, 114,   0,  15, 139, 102,  97,  92,  95,
         95,  94,  98,  94,  97, 102, 124,  14,   0,  79, 126, 118,  63,
          0,   0],
       [  0,   0,  49, 118, 105,  97,   0,  31, 139,  98, 102, 102, 101,
         98, 101, 101, 101,  95,  94, 131,  39,   0,  82, 118, 118,  52,
          0,   0],
       [  0,   0,  43, 113, 108,  91,   0,  63, 137,  89, 101, 102, 107,
        105, 107, 107, 104, 102,  91, 120,  55,   0,  82, 124, 118,  47,
          0,   0],
       [  0,   0,  43, 118, 114,  89,   0,  97, 121,  95, 102, 104, 107,
        104, 104, 104,  97, 104,  95, 111,  68,   0,  79, 123, 115,  44,
          0,   0],
       [  0,   0,  42, 120, 118,  69,   0, 114, 111, 104, 104, 105, 108,
        108, 108, 108, 102, 105, 104, 123,  92,   0,  75, 130, 124,  46,
          0,   0],
       [  0,   0,  44, 120, 117,  63,   0, 149, 124, 120, 120, 121, 124,
        121, 118, 118, 117, 120, 115, 123, 117,   0,  69, 117, 123,  49,
          0,   0],
       [  0,   0,  50, 117, 117,  69,   1, 149, 136, 131, 134, 134, 136,
        131, 130, 131, 131, 137, 133, 143, 147,   0,  75, 124, 117,  50,
          0,   0],
       [  0,   0,  65, 156, 150,  73,   0,   1,   0,   1,   2,   2,   2,
          2,   2,   2,   2,   1,   2,   4,   8,   0,  81, 134, 131,  63,
          0,   0],
       [  0,   0,  14,  50,  33,  10,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,  20,  55,  59,  17,
          0,   0]], dtype=uint8)
```

In [8]:
```
y_train[37]
```

Out[8]:
```
2
```

In [10]:
```
plt.matshow(X_train[37])
plt.show()
```



3.Normalize

```
In [11]: X_train = X_train.reshape((X_train.shape[0], 28*28)).astype('float32')
         X_test = X_test.reshape((X_test.shape[0], 28*28)).astype('float32')
```

```
In [12]: X_train = X_train / 255
         X_test = X_test / 255
```

```
In [13]: X_train[37]
```

```
Out[13]: array([0.        , 0.        , 0.00392157, 0.00392157, 0.        ,
                0.00392157, 0.        , 0.        , 0.        , 0.        ,
                0.13333334, 0.37254903, 0.        , 0.        , 0.        ,
                0.        , 0.20392157, 0.27450982, 0.        , 0.        ,
                0.        , 0.00784314, 0.00392157, 0.        , 0.00392157,
                0.        , 0.        , 0.        , 0.        , 0.        ,
                0.        , 0.00392157, 0.00392157, 0.        , 0.        ,
                0.04313726, 0.23137255, 0.43529412, 0.44313726, 0.7137255 ,
                0.6627451 , 0.8862745 , 1.        , 0.7372549 , 0.6862745 ,
                0.63529414, 0.4117647 , 0.33333334, 0.12156863, 0.        ,
                0.        , 0.        , 0.00392157, 0.        , 0.        ,
                0.        , 0.        , 0.        , 0.00392157, 0.        ,
                0.        , 0.10588235, 0.34901962, 0.49803922, 0.49803922,
                0.4509804 , 0.39607844, 0.3372549 , 0.31764707, 0.37254903,
                0.35686275, 0.34509805, 0.30588236, 0.36078432, 0.4509804 ,
                0.53333336, 0.54509807, 0.49411765, 0.28627452, 0.00392157,
                0.        , 0.        , 0.        , 0.        , 0.        ,
                0.        , 0.        , 0.        , 0.19215687, 0.45882353,
                0.44313726, 0.37254903, 0.36862746, 0.38039216, 0.38431373,
                0.4       , 0.39607844, 0.38431373, 0.35686275, 0.38039216
```

4.Build a simple baseline model

```
In [14]: from keras.models import Sequential
```

```
In [15]: from keras.layers import Activation,Dense
```

```
In [16]: from tensorflow.keras.models import Sequential
```

```
In [17]: model=Sequential()
         model.add(Dense(512,input_dim=28*28,activation='relu'))
         model.add(Dense(10,activation='softmax'))
```

```
In [18]: model.compile(loss='mean_squared_error', metrics=['accuracy'])
```

In [19]:
```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 24s 12ms/step - loss: 27.6102
- accuracy: 0.1028
Epoch 2/10
1875/1875 [==============================] - 22s 11ms/step - loss: 27.6101
- accuracy: 0.1044
Epoch 3/10
1875/1875 [==============================] - 22s 12ms/step - loss: 27.6101
- accuracy: 0.1037
Epoch 4/10
1875/1875 [==============================] - 23s 12ms/step - loss: 27.6101
- accuracy: 0.1037
Epoch 5/10
1875/1875 [==============================] - 22s 12ms/step - loss: 27.6101
- accuracy: 0.1040
Epoch 6/10
1875/1875 [==============================] - 22s 11ms/step - loss: 27.6101
- accuracy: 0.1024
Epoch 7/10
```

In [20]:
```
model.evaluate(X_test, y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 27.6100 - accu
racy: 0.1036
```

Out[20]:
```
[27.6099910736084, 0.10360000282526016]
```

In [21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 407050 (1.55 MB)
Trainable params: 407050 (1.55 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

5. Performance Analysis

2layers

In [22]:
```
model1 = Sequential()
model1.add(Dense(512, input_dim=28*28, activation='relu'))
model1.add(Dense(512, input_dim=28*28, activation='relu'))
model1.add(Dense(10,activation='softmax'))
model1.compile(loss='mean_squared_error', metrics=['accuracy'])
model1.fit(X_train,y_train,epochs=10)
model1.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 31s 16ms/step - loss: 27.6102 -
accuracy: 0.0998
Epoch 2/10
1875/1875 [==============================] - 28s 15ms/step - loss: 27.6101 -
accuracy: 0.0999
Epoch 3/10
1875/1875 [==============================] - 29s 16ms/step - loss: 27.6101 -
accuracy: 0.1013
Epoch 4/10
1875/1875 [==============================] - 29s 16ms/step - loss: 27.6101 -
accuracy: 0.1033
Epoch 5/10
1875/1875 [==============================] - 30s 16ms/step - loss: 27.6101 -
accuracy: 0.1040
Epoch 6/10
1875/1875 [==============================] - 29s 15ms/step - loss: 27.6101 -
accuracy: 0.1035
Epoch 7/10
1875/1875 [==============================] - 30s 16ms/step - loss: 27.6101 -
accuracy: 0.1031
Epoch 8/10
1875/1875 [==============================] - 30s 16ms/step - loss: 27.6101 -
accuracy: 0.1032
Epoch 9/10
1875/1875 [==============================] - 29s 15ms/step - loss: 27.6101 -
accuracy: 0.1043
Epoch 10/10
1875/1875 [==============================] - 28s 15ms/step - loss: 27.6101 -
accuracy: 0.1035
313/313 [==============================] - 2s 6ms/step - loss: 27.6100 - accu
racy: 0.1020
```

Out[22]:
```
[27.609987258911133, 0.10199999809265137]
```

In [23]:

```
model2 = Sequential()
model2.add(Dense(256, input_dim=28*28, activation='relu'))
model2.add(Dense(256, input_dim=28*28, activation='relu'))
model2.add(Dense(10,activation='softmax'))
model2.compile(loss='mean_squared_error', metrics=['accuracy'])
model2.fit(X_train,y_train,epochs=10)
model2.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0994
Epoch 2/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1018
Epoch 3/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1002
Epoch 4/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0996
Epoch 5/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0991
Epoch 6/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0997
Epoch 7/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0992
Epoch 8/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0989
Epoch 9/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1004
Epoch 10/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1010
313/313 [==============================] - 1s 4ms/step - loss: 27.6100 - accu
racy: 0.1015
```

Out[23]:

```
[27.609987258911133, 0.1014999970793724]
```

In [24]:

```
model3 = Sequential()
model3.add(Dense(128, input_dim=28*28, activation='relu'))
model3.add(Dense(128, input_dim=28*28, activation='relu'))
model3.add(Dense(10,activation='softmax'))
model3.compile(loss='mean_squared_error', metrics=['accuracy'])
model3.fit(X_train,y_train,epochs=10)
model3.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 27.6101 - a
ccuracy: 0.1057
Epoch 2/10
1875/1875 [==============================] - 10s 5ms/step - loss: 27.6101 - a
ccuracy: 0.1076
Epoch 3/10
1875/1875 [==============================] - 10s 6ms/step - loss: 27.6101 - a
ccuracy: 0.1053
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.1023
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0984
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0989
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0977
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0969
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0983
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0982
313/313 [==============================] - 1s 3ms/step - loss: 27.6100 - accu
racy: 0.0992
```

Out[24]:

```
[27.609987258911133, 0.09920000284910202]
```

In [25]:

```
model4 = Sequential()
model4.add(Dense(512, input_dim=28*28, activation='relu'))
model4.add(Dense(512, input_dim=28*28, activation='relu'))
model4.add(Dense(512, input_dim=28*28, activation='relu'))
model4.add(Dense(10,activation='softmax'))
model4.compile(loss='mean_squared_error', metrics=['accuracy'])
model4.fit(X_train,y_train,epochs=10)
model4.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 42s 21ms/step - loss: 27.6101 -
accuracy: 0.1005
Epoch 2/10
1875/1875 [==============================] - 39s 21ms/step - loss: 27.6101 -
accuracy: 0.0940
Epoch 3/10
1875/1875 [==============================] - 39s 21ms/step - loss: 27.6101 -
accuracy: 0.0931
Epoch 4/10
1875/1875 [==============================] - 39s 21ms/step - loss: 27.6101 -
accuracy: 0.0930
Epoch 5/10
1875/1875 [==============================] - 39s 21ms/step - loss: 27.6101 -
accuracy: 0.0934
Epoch 6/10
1875/1875 [==============================] - 39s 21ms/step - loss: 27.6101 -
accuracy: 0.0941
Epoch 7/10
1875/1875 [==============================] - 40s 21ms/step - loss: 27.6101 -
accuracy: 0.0962
Epoch 8/10
1875/1875 [==============================] - 40s 21ms/step - loss: 27.6101 -
accuracy: 0.0962
Epoch 9/10
1875/1875 [==============================] - 40s 21ms/step - loss: 27.6101 -
accuracy: 0.0965
Epoch 10/10
1875/1875 [==============================] - 40s 21ms/step - loss: 27.6101 -
accuracy: 0.0978
313/313 [==============================] - 3s 8ms/step - loss: 27.6100 - accu
racy: 0.0994
```

Out[25]:

```
[27.609987258911133, 0.09939999878406525]
```

In [26]:

```
model5 = Sequential()
model5.add(Dense(256, input_dim=28*28, activation='relu'))
model5.add(Dense(256, input_dim=28*28, activation='relu'))
model5.add(Dense(256, input_dim=28*28, activation='relu'))
model5.add(Dense(10,activation='softmax'))
model5.compile(loss='mean_squared_error', metrics=['accuracy'])
model5.fit(X_train,y_train,epochs=10)
model5.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.1043
Epoch 2/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1031
Epoch 3/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1038
Epoch 4/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1063
Epoch 5/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1081
Epoch 6/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1103
Epoch 7/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1110
Epoch 8/10
1875/1875 [==============================] - 13s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1119
Epoch 9/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - a
ccuracy: 0.1126
Epoch 10/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.1141
313/313 [==============================] - 2s 4ms/step - loss: 27.6100 - accu
racy: 0.1140
```

Out[26]:

```
[27.609987258911133, 0.11400000005960464]
```

In [27]:

```
model6 = Sequential()
model6.add(Dense(128, input_dim=28*28, activation='relu'))
model6.add(Dense(128, input_dim=28*28, activation='relu'))
model6.add(Dense(128, input_dim=28*28, activation='relu'))
model6.add(Dense(10,activation='softmax'))
model6.compile(loss='mean_squared_error', metrics=['accuracy'])
model6.fit(X_train,y_train,epochs=10)
model6.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 27.6101 - a
ccuracy: 0.1065
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0930
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0867
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0887
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0925
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0949
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0964
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0990
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: 0.1022
313/313 [==============================] - 1s 3ms/step - loss: 27.6100 - accu
racy: 0.1065
```

Out[27]:

```
[27.609987258911133, 0.10649999976158142]
```

In [28]:

```
model7 = Sequential()
model7.add(Dense(512, input_dim=28*28, activation='relu'))
model7.add(Dense(512, input_dim=28*28, activation='relu'))
model7.add(Dense(512, input_dim=28*28, activation='relu'))
model7.add(Dense(512, input_dim=28*28, activation='relu'))
model7.add(Dense(10,activation='softmax'))
model7.compile(loss='mean_squared_error', metrics=['accuracy'])
model7.fit(X_train,y_train,epochs=10)
model7.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 52s 27ms/step - loss: 27.6101 -
accuracy: 0.0904
Epoch 2/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101 -
accuracy: 0.1005
Epoch 3/10
1875/1875 [==============================] - 50s 26ms/step - loss: 27.6101 -
accuracy: 0.1015
Epoch 4/10
1875/1875 [==============================] - 50s 26ms/step - loss: 27.6101 -
accuracy: 0.1043
Epoch 5/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101 -
accuracy: 0.1060
Epoch 6/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101 -
accuracy: 0.1057
Epoch 7/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101 -
accuracy: 0.1063
Epoch 8/10
1875/1875 [==============================] - 51s 27ms/step - loss: 27.6101 -
accuracy: 0.1061
Epoch 9/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101 -
accuracy: 0.1057
Epoch 10/10
1875/1875 [==============================] - 50s 27ms/step - loss: 27.6101 -
accuracy: 0.1060
313/313 [==============================] - 3s 9ms/step - loss: 27.6100 - accu
racy: 0.1031
```

Out[28]:

```
[27.609987258911133, 0.1031000018119812]
```

```
In [*]: model8 = Sequential()
        model8.add(Dense(256, input_dim=28*28, activation='relu'))
        model8.add(Dense(256, input_dim=28*28, activation='relu'))
        model8.add(Dense(256, input_dim=28*28, activation='relu'))
        model8.add(Dense(256, input_dim=28*28, activation='relu'))
        model8.add(Dense(10,activation='softmax'))
        model8.compile(loss='mean_squared_error', metrics=['accuracy'])
        model8.fit(X_train,y_train,epochs=10)
        model8.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 20s 10ms/step - loss: 27.6101 -
accuracy: 0.0901
Epoch 2/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0866
Epoch 3/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0853
Epoch 4/10
1875/1875 [==============================] - 16s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0843
Epoch 5/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0848
Epoch 6/10
1346/1875 [====================>.........] - ETA: 4s - loss: 27.5904 - accura
cy: 0.0859
```

```
In [30]: model9 = Sequential()
         model9.add(Dense(128, input_dim=28*28, activation='relu'))
         model9.add(Dense(128, input_dim=28*28, activation='relu'))
         model9.add(Dense(128, input_dim=28*28, activation='relu'))
         model9.add(Dense(128, input_dim=28*28, activation='relu'))
         model9.add(Dense(10,activation='softmax'))
         model9.compile(loss='mean_squared_error', metrics=['accuracy'])
         model9.fit(X_train,y_train,epochs=10)
         model9.evaluate(X_test,y_test)
```

```
1875/1875 [==============================] - 7s 4ms/step - loss: 27.6101 - ac
curacy: 0.0974
Epoch 8/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.6101 - ac
curacy: 0.0989
Epoch 9/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.6101 - ac
curacy: 0.1019
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.6101 - ac
curacy: 0.1050
313/313 [==============================] - 1s 2ms/step - loss: 27.6100 - accu
racy: 0.1101
```

```
Out[30]: [27.609987258911133, 0.11010000109672546]
```

In [31]:

```
model10 = Sequential()
model10.add(Dense(512, input_dim=28*28, activation='relu'))
model10.add(Dense(512, input_dim=28*28, activation='relu'))
model10.add(Dense(512, input_dim=28*28, activation='relu'))
model10.add(Dense(512, input_dim=28*28, activation='relu'))
model10.add(Dense(512, input_dim=28*28, activation='relu'))
model10.add(Dense(10,activation='softmax'))
model10.compile(loss='mean_squared_error', metrics=['accuracy'])
model10.fit(X_train,y_train,epochs=10)
model10.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 57s 30ms/step - loss: 27.6101 -
accuracy: 0.0717
Epoch 2/10
1875/1875 [==============================] - 55s 29ms/step - loss: 27.6101 -
accuracy: 0.0779
Epoch 3/10
1875/1875 [==============================] - 57s 31ms/step - loss: 27.6101 -
accuracy: 0.0808
Epoch 4/10
1875/1875 [==============================] - 56s 30ms/step - loss: 27.6101 -
accuracy: 0.0828
Epoch 5/10
1875/1875 [==============================] - 59s 31ms/step - loss: 27.6101 -
accuracy: 0.0851
Epoch 6/10
1875/1875 [==============================] - 68s 36ms/step - loss: 27.6101 -
accuracy: 0.0870
Epoch 7/10
1875/1875 [==============================] - 65s 35ms/step - loss: 27.6101 -
accuracy: 0.0890
Epoch 8/10
1875/1875 [==============================] - 60s 32ms/step - loss: 27.6101 -
accuracy: 0.0897
Epoch 9/10
1875/1875 [==============================] - 60s 32ms/step - loss: 27.6101 -
accuracy: 0.0920
Epoch 10/10
1875/1875 [==============================] - 57s 31ms/step - loss: 27.6101 -
accuracy: 0.0933
313/313 [==============================] - 3s 8ms/step - loss: 27.6100 - accu
racy: 0.0936
```

Out[31]:

```
[27.609987258911133, 0.09359999746084213]
```

In [32]:

```
model11 = Sequential()
model11.add(Dense(256, input_dim=28*28, activation='relu'))
model11.add(Dense(256, input_dim=28*28, activation='relu'))
model11.add(Dense(256, input_dim=28*28, activation='relu'))
model11.add(Dense(256, input_dim=28*28, activation='relu'))
model11.add(Dense(256, input_dim=28*28, activation='relu'))
model11.add(Dense(10,activation='softmax'))
model11.compile(loss='mean_squared_error', metrics=['accuracy'])
model11.fit(X_train,y_train,epochs=10)
model11.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0996
Epoch 2/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0913
Epoch 3/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0913
Epoch 4/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0919
Epoch 5/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0918
Epoch 6/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0910
Epoch 7/10
1875/1875 [==============================] - 14s 7ms/step - loss: 27.6101 - a
ccuracy: 0.0894
Epoch 8/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0898
Epoch 9/10
1875/1875 [==============================] - 14s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0898
Epoch 10/10
1875/1875 [==============================] - 15s 8ms/step - loss: 27.6101 - a
ccuracy: 0.0909
313/313 [==============================] - 2s 4ms/step - loss: 27.6100 - accu
racy: 0.0876
```

Out[32]:

```
[27.609987258911133, 0.08760000020265579]
```

In [33]:

```
model12 = Sequential()
model12.add(Dense(128, input_dim=28*28, activation='relu'))
model12.add(Dense(128, input_dim=28*28, activation='relu'))
model12.add(Dense(128, input_dim=28*28, activation='relu'))
model12.add(Dense(128, input_dim=28*28, activation='relu'))
model12.add(Dense(128, input_dim=28*28, activation='relu'))
model12.add(Dense(10,activation='softmax'))
model12.compile(loss='mean_squared_error', metrics=['accuracy'])
model12.fit(X_train,y_train,epochs=10)
model12.evaluate(X_test,y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 11s 5ms/step - loss: 27.6101 - a
ccuracy: 0.0797
Epoch 2/10
1875/1875 [==============================] - 10s 5ms/step - loss: 27.6101 - a
ccuracy: 0.0879
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0930
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0919
Epoch 5/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - ac
curacy: 0.0901
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - ac
curacy: 0.0869
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 27.6101 - ac
curacy: 0.0853
Epoch 8/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - ac
curacy: 0.0852
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - ac
curacy: 0.0845
Epoch 10/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.6101 - ac
curacy: 0.0862
313/313 [==============================] - 1s 2ms/step - loss: 27.6100 - accu
racy: 0.0841
```

Out[33]:

```
[27.609987258911133, 0.08410000056028366]
```