Name : Arul Kumar ARK

Roll No. : 225229103

In [ ]:

```
                    Lab : 6 :  Multi-class Classification of Fashion Apparels using DNN
```

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

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

In [3]:

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

In [4]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten
```

In [5]:

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
```

In [6]:

```
from keras.datasets import fashion_mnist
```

In [7]:

```
from tensorflow.keras.optimizers import Adam
```

In [8]:

```
from sklearn.model_selection import train_test_split
```

Dataset Split :

In [9]:

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz)
29515/29515 [==============================] - 0s 3us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz)
26421880/26421880 [==============================] - 66s 3us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/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-datasets/t10k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz)
4422102/4422102 [==============================] - 1s 0us/step
```

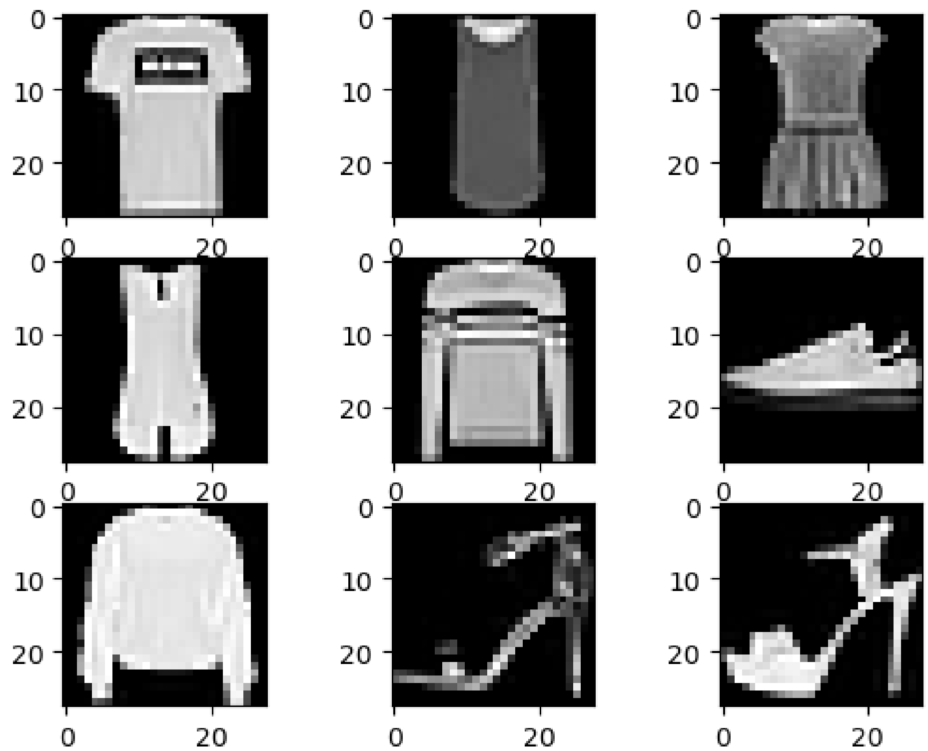
In [10]:

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

Img

In [11]:

```
for i in range(1, 10):
    plt.subplot(3, 3, i)
    plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))
plt.show()
```



Shape

In [12]:

```
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,)
```

Size

In [13]:

```
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 [14]:

```
print(X_train[37])
```

```
   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]
```

In [15]:

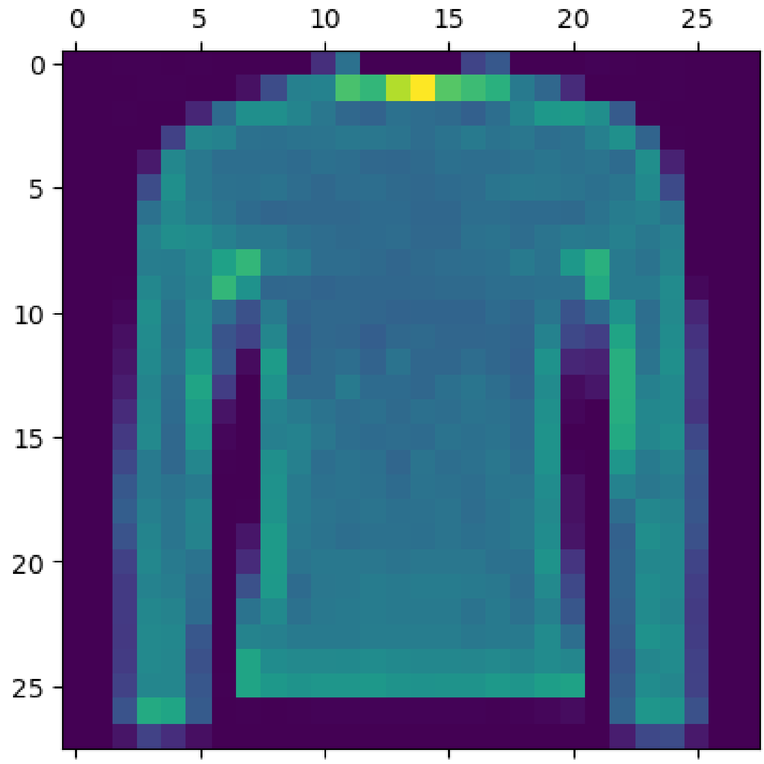
```
y_train[37]
```

Out[15]:

2

In [16]:

```
plt.matshow(X_train[37])
plt.show()
```



Normalize

In [17]:

```
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 [18]:

```
X_train = X_train / 255
X_test = X_test / 255
```

Baseline model

In [19]:

```
model = Sequential()
model.add(Dense(512, input_dim=28*28, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

In [20]:

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

In [21]:

```
model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 23s 12ms/step - loss: 27.6102 - acc
uracy: 0.1043
Epoch 2/10
1875/1875 [==============================] - 24s 13ms/step - loss: 27.6101 - acc
uracy: 0.0989
Epoch 3/10
1875/1875 [==============================] - 25s 13ms/step - loss: 27.6101 - acc
uracy: 0.0975
Epoch 4/10
1875/1875 [==============================] - 28s 15ms/step - loss: 27.6101 - acc
uracy: 0.0978
Epoch 5/10
1875/1875 [==============================] - 27s 14ms/step - loss: 27.6101 - acc
uracy: 0.0970
Epoch 6/10
1875/1875 [==============================] - 26s 14ms/step - loss: 27.6101 - acc
uracy: 0.0969
Epoch 7/10
```

In [22]:

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

```
313/313 [==============================] - 1s 3ms/step - loss: 27.6100 - accuracy:
0.0942
```

Out[22]:

```
[27.60999298095703, 0.094200000166893]
```

In [23]:

```
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)
_________________________________________________________________
```

In [24]:

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=1
```

In [25]:

```
history = model.fit(X_train,y_train,epochs=10,validation_data=(X_val, y_val))
```

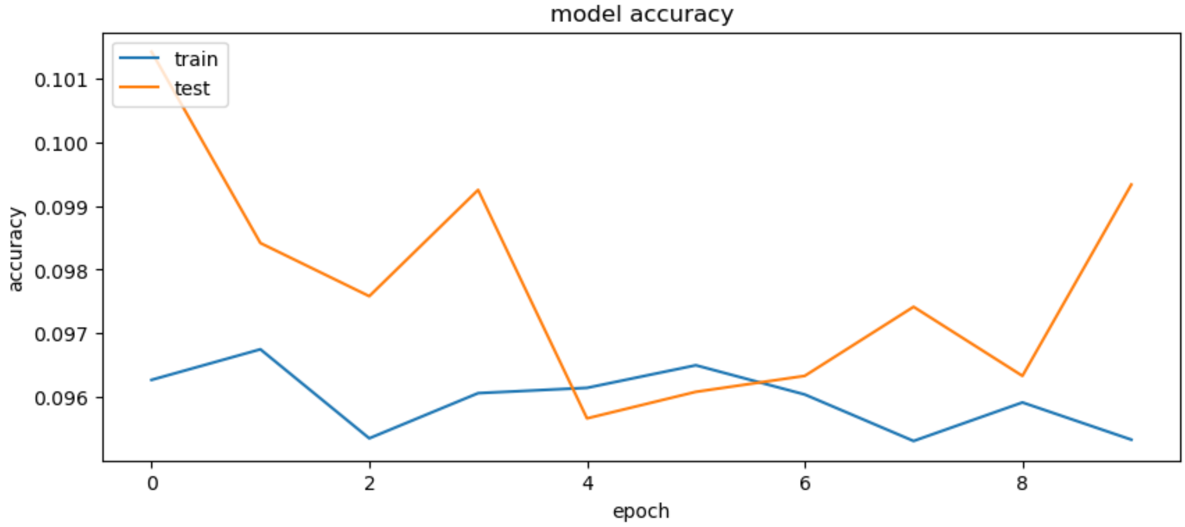
```
Epoch 1/10
1500/1500 [==============================] - 21s 14ms/step - loss: 27.5856 - acc
uracy: 0.0963 - val_loss: 27.7079 - val_accuracy: 0.1014
Epoch 2/10
1500/1500 [==============================] - 23s 15ms/step - loss: 27.5856 - acc
uracy: 0.0967 - val_loss: 27.7079 - val_accuracy: 0.0984
Epoch 3/10
1500/1500 [==============================] - 22s 15ms/step - loss: 27.5856 - acc
uracy: 0.0954 - val_loss: 27.7079 - val_accuracy: 0.0976
Epoch 4/10
1500/1500 [==============================] - 23s 15ms/step - loss: 27.5856 - acc
uracy: 0.0961 - val_loss: 27.7079 - val_accuracy: 0.0993
Epoch 5/10
1500/1500 [==============================] - 23s 15ms/step - loss: 27.5856 - acc
uracy: 0.0961 - val_loss: 27.7079 - val_accuracy: 0.0957
Epoch 6/10
1500/1500 [==============================] - 21s 14ms/step - loss: 27.5856 - acc
uracy: 0.0965 - val_loss: 27.7079 - val_accuracy: 0.0961
Epoch 7/10
```

In [26]:

```
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'])
```



Performance Analysis

Layer : 2 : 512

In [27]:

```
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'])
```

In [28]:

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

```
Epoch 1/10
1500/1500 [==============================] - 27s 17ms/step - loss: 27.5856 - acc
uracy: 0.1027
Epoch 2/10
1500/1500 [==============================] - 26s 17ms/step - loss: 27.5856 - acc
uracy: 0.1010
Epoch 3/10
1500/1500 [==============================] - 30s 20ms/step - loss: 27.5856 - acc
uracy: 0.0981
Epoch 4/10
1500/1500 [==============================] - 31s 21ms/step - loss: 27.5856 - acc
uracy: 0.0989
Epoch 5/10
1500/1500 [==============================] - 30s 20ms/step - loss: 27.5856 - acc
uracy: 0.1003
Epoch 6/10
1500/1500 [==============================] - 27s 18ms/step - loss: 27.5856 - acc
uracy: 0.1019
Epoch 7/10
```

In [29]:

```
model1.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 27.6100 - accuracy:
0.1066
```

Out[29]:

```
[27.609987258911133, 0.10660000145435333]
```

Layer : 2 : 256

In [30]:

```
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'])
```

In [31]:

```
model2.fit(X_train,y_train,epochs=10)
model2.evaluate(X_test,y_test)
```

```
Epoch 1/10
1500/1500 [==============================] - 12s 7ms/step - loss: 27.5857 - accu
racy: 0.1154
Epoch 2/10
1500/1500 [==============================] - 18s 12ms/step - loss: 27.5856 - acc
uracy: 0.1052
Epoch 3/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.1058
Epoch 4/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - accu
racy: 0.1083
Epoch 5/10
1500/1500 [==============================] - 15s 10ms/step - loss: 27.5856 - acc
uracy: 0.1079
Epoch 6/10
1500/1500 [==============================] - 10s 7ms/step - loss: 27.5856 - accu
racy: 0.1110
Epoch 7/10
```

In [32]:

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

```
Epoch 1/10
1500/1500 [==============================] - 15s 10ms/step - loss: 27.5856 - acc
uracy: 0.1067
Epoch 2/10
1500/1500 [==============================] - 13s 9ms/step - loss: 27.5856 - accu
racy: 0.1058
Epoch 3/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.1050
Epoch 4/10
1500/1500 [==============================] - 11s 7ms/step - loss: 27.5856 - accu
racy: 0.1059
Epoch 5/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.1044
Epoch 6/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.1064
Epoch 7/10
```

In [33]:

```
model2.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 27.6100 - accuracy:
0.1064
```

Out[33]:

```
[27.609987258911133, 0.10639999806880951]
```

Layer : 2 : 128

In [34]:

```
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'])
```

In [35]:

```
model3.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 10s 6ms/step - loss: 27.5856 - accu
racy: 0.1088
Epoch 2/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.5856 - accur
acy: 0.1266
Epoch 3/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accur
acy: 0.1260
Epoch 4/10
1500/1500 [==============================] - 10s 7ms/step - loss: 27.5856 - accu
racy: 0.1191
Epoch 5/10
1500/1500 [==============================] - 10s 6ms/step - loss: 27.5856 - accu
racy: 0.1121
Epoch 6/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accur
acy: 0.1057
Epoch 7/10
```

In [36]:

```
model3.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 27.6100 - accuracy:
0.0937
```

Out[36]:

```
[27.609987258911133, 0.09369999915361404]
```

Layer : 3 : 512

In [37]:

```
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'])
```

In [38]:

```
model4.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 40s 26ms/step - loss: 27.5856 - acc
uracy: 0.1001
Epoch 2/10
1500/1500 [==============================] - 34s 23ms/step - loss: 27.5856 - acc
uracy: 0.0927
Epoch 3/10
1500/1500 [==============================] - 34s 23ms/step - loss: 27.5856 - acc
uracy: 0.0952
Epoch 4/10
1500/1500 [==============================] - 38s 25ms/step - loss: 27.5856 - acc
uracy: 0.0967
Epoch 5/10
1500/1500 [==============================] - 33s 22ms/step - loss: 27.5856 - acc
uracy: 0.0979
Epoch 6/10
1500/1500 [==============================] - 31s 21ms/step - loss: 27.5856 - acc
uracy: 0.0967
Epoch 7/10
```

In [39]:

```
model4.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 27.6100 - accuracy:
0.0976
```

Out[39]:

```
[27.609987258911133, 0.09759999811649323]
```

Layer : 3 : 256

In [40]:

```
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'])
```

In [41]:

```
model5.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 13s 8ms/step - loss: 27.5856 - accu
racy: 0.0962
Epoch 2/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.0938
Epoch 3/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.0959
Epoch 4/10
1500/1500 [==============================] - 12s 8ms/step - loss: 27.5856 - accu
racy: 0.0967
Epoch 5/10
1500/1500 [==============================] - 13s 9ms/step - loss: 27.5856 - accu
racy: 0.0951
Epoch 6/10
1500/1500 [==============================] - 13s 9ms/step - loss: 27.5856 - accu
racy: 0.0935
Epoch 7/10
```

In [42]:

```
model5.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 4ms/step - loss: 27.6100 - accuracy:
0.0948
```

Out[42]:

```
[27.609987258911133, 0.09480000287294388]
```

Layer : 4 : 512

In [43]:

```
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'])
```

In [44]:

```
model7.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 43s 28ms/step - loss: 27.5856 - acc
uracy: 0.0951
Epoch 2/10
1500/1500 [==============================] - 41s 27ms/step - loss: 27.5856 - acc
uracy: 0.0947
Epoch 3/10
1500/1500 [==============================] - 42s 28ms/step - loss: 27.5856 - acc
uracy: 0.0979
Epoch 4/10
1500/1500 [==============================] - 47s 31ms/step - loss: 27.5856 - acc
uracy: 0.1001
Epoch 5/10
1500/1500 [==============================] - 54s 36ms/step - loss: 27.5856 - acc
uracy: 0.1020
Epoch 6/10
1500/1500 [==============================] - 46s 31ms/step - loss: 27.5856 - acc
uracy: 0.1025
Epoch 7/10
```

In [45]:

```
model7.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 27.6100 - accuracy:
0.1025
```

Out[45]:

```
[27.609987258911133, 0.10249999910593033]
```

Layer : 4 : 256

In [46]:

```
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'])
```

In [47]:

```
model8.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 15s 9ms/step - loss: 27.5856 - accu
racy: 0.1345
Epoch 2/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.1187
Epoch 3/10
1500/1500 [==============================] - 15s 10ms/step - loss: 27.5856 - acc
uracy: 0.1129
Epoch 4/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accu
racy: 0.1113
Epoch 5/10
1500/1500 [==============================] - 14s 10ms/step - loss: 27.5856 - acc
uracy: 0.1111
Epoch 6/10
1500/1500 [==============================] - 15s 10ms/step - loss: 27.5856 - acc
uracy: 0.1118
Epoch 7/10
```

In [48]:

```
model8.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 27.6100 - accuracy:
0.1144
```

Out[48]:

```
[27.609987258911133, 0.1143999993801117]
```

Layer : 4 : 128

In [49]:

```
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'])
```

In [50]:

```
model9.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 9s 5ms/step - loss: 27.5856 - accur
acy: 0.1103
Epoch 2/10
1500/1500 [==============================] - 10s 6ms/step - loss: 27.5856 - accu
racy: 0.1074
Epoch 3/10
1500/1500 [==============================] - 9s 6ms/step - loss: 27.5856 - accur
acy: 0.1024
Epoch 4/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.5856 - accur
acy: 0.0956
Epoch 5/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accur
acy: 0.0921
Epoch 6/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.5856 - accur
acy: 0.0918
Epoch 7/10
```

In [51]:

```
model9.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 4ms/step - loss: 27.6100 - accuracy:
0.0926
```

Out[51]:

```
[27.609987258911133, 0.09260000288486481]
```

Layer : 5 : 512

In [52]:

```
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'])
```

In [53]:

```
model10.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 47s 30ms/step - loss: 27.5856 - accura
cy: 0.0960
Epoch 2/10
1500/1500 [==============================] - 50s 33ms/step - loss: 27.5856 - accura
cy: 0.1028
Epoch 3/10
1500/1500 [==============================] - 55s 37ms/step - loss: 27.5856 - accura
cy: 0.1066
Epoch 4/10
1500/1500 [==============================] - 58s 39ms/step - loss: 27.5856 - accura
cy: 0.1088
Epoch 5/10
1500/1500 [==============================] - 48s 32ms/step - loss: 27.5856 - accura
cy: 0.1084
Epoch 6/10
1500/1500 [==============================] - 45s 30ms/step - loss: 27.5856 - accura
cy: 0.1074
Epoch 7/10
1500/1500 [==============================] - 49s 32ms/step - loss: 27.5856 - accura
cy: 0.1080
Epoch 8/10
1500/1500 [==============================] - 55s 37ms/step - loss: 27.5856 - accura
cy: 0.1066
Epoch 9/10
1500/1500 [==============================] - 57s 38ms/step - loss: 27.5856 - accura
cy: 0.1059
Epoch 10/10
1500/1500 [==============================] - 45s 30ms/step - loss: 27.5856 - accura
cy: 0.1036
```

Out[53]:

```
<keras.src.callbacks.History at 0x22f85f37510>
```

In [54]:

```
model10.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 7ms/step - loss: 27.6100 - accuracy:
0.1044
```

Out[54]:

```
[27.609987258911133, 0.10440000146627426]
```

Layer : 5 : 256

In [55]:

```
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'])
```

In [56]:

```
model11.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 17s 10ms/step - loss: 27.5856 - accura
cy: 0.0886
Epoch 2/10
1500/1500 [==============================] - 16s 11ms/step - loss: 27.5856 - accura
cy: 0.0955
Epoch 3/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accurac
y: 0.0972
Epoch 4/10
1500/1500 [==============================] - 14s 10ms/step - loss: 27.5856 - accura
cy: 0.0994
Epoch 5/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accurac
y: 0.0983
Epoch 6/10
1500/1500 [==============================] - 14s 10ms/step - loss: 27.5856 - accura
cy: 0.0981
Epoch 7/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accurac
y: 0.0965
Epoch 8/10
1500/1500 [==============================] - 14s 9ms/step - loss: 27.5856 - accurac
y: 0.0960
Epoch 9/10
1500/1500 [==============================] - 15s 10ms/step - loss: 27.5856 - accura
cy: 0.0945
Epoch 10/10
1500/1500 [==============================] - 15s 10ms/step - loss: 27.5856 - accura
cy: 0.0940
```

Out[56]:

```
<keras.src.callbacks.History at 0x22f8c1ad690>
```

In [57]:

```
model11.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 4ms/step - loss: 27.6100 - accuracy:
0.0921
```

Out[57]:

```
[27.609987258911133, 0.09210000187158585]
```

Layer : 5 : 128

In [58]:

```
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'])
```

In [59]:

```
model12.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.5856 - accurac
y: 0.0921
Epoch 2/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.5856 - accurac
y: 0.1023
Epoch 3/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.1038
Epoch 4/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.1044
Epoch 5/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.1035
Epoch 6/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.1015
Epoch 7/10
1500/1500 [==============================] - 8s 5ms/step - loss: 27.5856 - accurac
y: 0.1005
Epoch 8/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.0986
Epoch 9/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.0982
Epoch 10/10
1500/1500 [==============================] - 7s 5ms/step - loss: 27.5856 - accurac
y: 0.0960
```

Out[59]:

```
<keras.src.callbacks.History at 0x22f95fa6610>
```

In [60]:

```
model12.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 27.6100 - accuracy:
0.0914
```

Out[60]:

```
[27.609987258911133, 0.09139999747276306]
```