

**#Apeksha Phapale Roll NO: 4231 Div:B**

**# DL Practical NO.3B**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
class_names=['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag',
'Ankleboot']
df1 = pd.read_csv(r'C:\Users\apeksha\Downloads\DL\pract_3\fashion-mnist_train.csv')
print(df1)
```

```
x_train = df1.drop("label", axis=1).values
y_train = df1["label"].values
print("x_train shape: ",x_train.shape)
print("y_train shape: ",y_train.shape)
print(np.unique(y_train))
df2 = pd.read_csv(r'C:\Users\apeksha\Downloads\DL\pract_3\fashion-mnist_test.csv')
print(df2)
```

```
x_test = df2.drop("label", axis=1).values
y_test = df2["label"].values
print("x_test shape: ",x_test.shape)
print("y_test shape: ",y_test.shape)
x_train = x_train.reshape(60000, 28, 28)
x_test = x_test.reshape(10000, 28, 28)
print(x_train[0])
print(y_train[0])
plt.imshow(x_train[0])
print(x_test[10])
print(y_test[10])
plt.imshow(x_test[10])
```

**#normalization and reshaping**

```
x_train = x_train/255
x_test = x_test/255
x_train = x_train.reshape(60000, 28, 28, 1)
x_test = x_test.reshape(10000, 28, 28, 1)
```

```

print("Train Shape :",x_train.shape)
print("Test Shape :",x_test.shape)
print("y_train shape :",y_train.shape)
print("y_test shape :",y_test.shape)

#building model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
model=Sequential()
model.add(Conv2D(64, (3,3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.summary()

#model training

model.fit(x_train, y_train, epochs=3, verbose=1,validation_data=(x_test,y_test))

#model testing
predictions = model.predict(x_test)

import numpy as np
index=10
print(predictions[index])
final_value=np.argmax(predictions[index])
print("Actual label :",y_test[index])
print("Predicted label :",final_value)
print("Class :",class_names[final_value])

plt.imshow(x_test[10])

#evaluation of model
loss, accuracy = model.evaluate(x_test, y_test)

```

```
print("Loss :",loss)
print("Accuracy (Test Data) :",accuracy*100)
```

### **Output:**

label	pixel1	pixel2	pixel3	...	pixel781	pixel782	pixel783	pixel784
0	2	0	0	0 ...	0	0	0	0
1	9	0	0	0 ...	0	0	0	0
2	6	0	0	0 ...	0	0	0	0
3	0	0	0	0 ...	0	0	0	0
4	3	0	0	0 ...	0	0	0	0
...	...	...	...	...	...	...	...	...
59995	9	0	0	0 ...	0	0	0	0
59996	1	0	0	0 ...	0	0	0	0
59997	8	0	0	0 ...	0	0	0	0
59998	8	0	0	0 ...	0	0	0	0
59999	7	0	0	0 ...	0	0	0	0

[60000 rows x 785 columns]

x\_train shape: (60000, 784)

y\_train shape: (60000,)

[0 1 2 3 4 5 6 7 8 9]

label	pixel1	pixel2	pixel3	...	pixel781	pixel782	pixel783	pixel784
0	0	0	0	0 ...	0	0	0	0
1	1	0	0	0 ...	0	0	0	0
2	2	0	0	0 ...	31	0	0	0
3	2	0	0	0 ...	222	56	0	0
4	3	0	0	0 ...	0	0	0	0
...	...	...	...	...	...	...	...	...
9995	0	0	0	0 ...	1	0	0	0
9996	6	0	0	0 ...	28	0	0	0
9997	8	0	0	0 ...	42	0	1	0
9998	8	0	1	3 ...	0	0	0	0
9999	1	0	0	0 ...	0	0	0	0

[10000 rows x 785 columns]

x\_test shape: (10000, 784)

y\_test shape: (10000,)

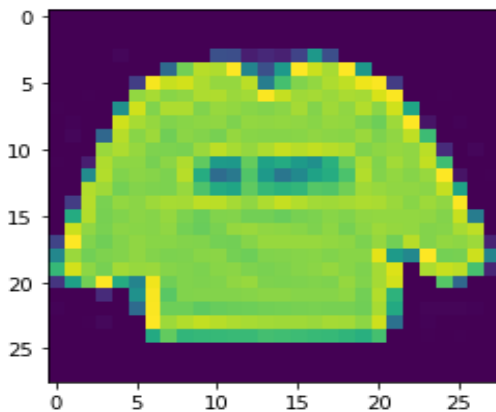
[[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0]  
[ 0 0 0 0 4 0 0 0 0 0 62 61 21 29 23 51 136 61  
0 0 0 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 88 201 228 225 255 115 62 137 255 235 222  
255 135 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 47 252 234 238 224 215 215 229 108 180 207 214 224  
231 249 254 45 0 0 0 0 0 0]  
[ 0 0 1 0 0 214 222 210 213 224 225 217 220 254 233 219 221 217  
223 221 240 254 0 0 1 0 0 0]  
[ 1 0 0 0 128 237 207 224 224 207 216 214 210 208 211 221 208 219  
213 226 211 237 150 0 0 0 0 0]  
[ 0 2 0 0 237 222 215 207 210 212 213 206 214 213 214 213 210 215  
214 206 199 218 255 13 0 2 0 0]  
[ 0 4 0 85 228 210 218 200 211 208 203 215 210 209 209 210 213 211  
210 217 206 213 231 175 0 0 0 0]  
[ 0 0 0 217 224 215 206 205 204 217 230 222 215 224 233 228 232 228  
224 207 212 215 213 229 31 0 4 0]  
[ 1 0 21 225 212 212 203 211 225 193 139 136 195 147 156 139 128 162  
197 223 207 220 213 232 177 0 0 0]  
[ 0 0 123 226 207 211 209 205 228 158 90 103 186 138 100 121 147 158  
183 226 208 214 209 216 255 13 0 1]  
[ 0 0 226 219 202 208 206 205 216 184 156 150 193 170 164 168 188 186  
200 219 216 213 213 211 233 148 0 0]  
[ 0 45 227 204 214 211 218 222 221 230 229 221 213 224 233 226 220 219  
221 224 223 217 210 218 213 254 0 0]  
[ 0 157 226 203 207 211 209 215 205 198 207 208 201 201 197 203 205 210  
207 213 214 214 214 213 208 234 107 0]  
[ 0 235 213 204 211 210 209 213 202 197 204 215 217 213 212 210 206 212  
203 211 218 215 214 208 209 222 230 0]  
[ 52 255 207 200 208 213 210 210 208 207 202 201 209 216 216 216 216 214  
212 205 215 201 228 208 214 212 218 25]  
[118 217 201 206 208 213 208 205 206 210 211 202 199 207 208 209 210 207  
210 210 245 139 119 255 202 203 236 114]

```

[171 238 212 203 220 216 217 209 207 205 210 211 206 204 206 209 211 215
 210 206 221 242  0 224 234 230 181 26]
[ 39 145 201 255 157 115 250 200 207 206 207 213 216 206 205 206 207 206
 215 207 221 238  0  0 188 85  0  0]
[  0  0  0 31  0 129 253 190 207 208 208 208 209 211 211 209 209 209
 212 201 226 165  0  0  0  0  0  0]
[  2  0  0  0  0 89 254 199 199 192 196 198 199 201 202 203 204 203
 203 200 222 155  0  3  3  3  2  0]
[  0  0  1  5  0  0 255 218 226 232 228 224 222 220 219 219 217 221
 220 212 236 95  0  2  0  0  0  0]
[  0  0  0  0  0  0 155 194 168 170 171 173 173 179 177 175 172 171
 167 161 180  0  0  1  0  1  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0]
[  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0]]

```

2



```

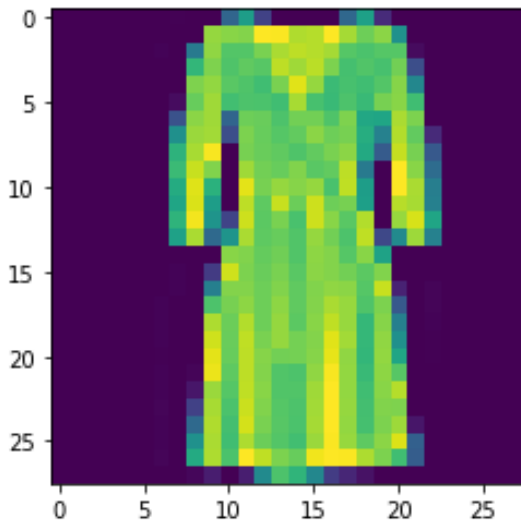
[[ 0  0  0  0  0  0  0  0  1  0  0 83 142 50  0  0  0  0  85
 145 31  0  0  0  0  0  0  0  0  0]
[  0  0  0  0  0  0  0  0  0  0 215 210 208 255 254 225 227 255 221
 199 211 129  0  0  0  0  0  0  0]
[  0  0  0  0  0  0  2  0 105 213 187 187 204 223 230 227 221 188
 183 188 188  7  0  0  0  0  0  0]
[  0  0  0  0  0  0  0  0  0 169 206 185 193 189 230 219 229 205 180
 186 181 201 61  0  0  0  0  0  0]
[  0  0  0  0  0  0  0  0  0 206 214 190 185 177 204 244 215 174 181
 177 187 209 118  0  0  0  0  0  0]]

```

[ 0 0 0 0 0 0 0 8 196 219 178 184 183 177 222 181 173 184  
173 203 210 177 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 64 211 219 83 199 197 184 201 201 185 206  
153 150 223 205 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 128 217 220 61 205 196 188 194 211 199 203  
159 112 226 194 30 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 165 222 253 0 203 197 193 185 194 204 211  
155 73 233 203 71 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 174 234 207 0 219 201 196 207 190 194 230  
105 0 255 210 90 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 157 243 163 0 245 203 215 209 215 182 231  
142 0 255 223 109 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 150 241 142 0 230 192 234 198 236 199 203  
144 0 228 222 111 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 166 251 132 52 236 191 204 182 236 210 190  
226 0 216 240 150 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 146 223 87 132 223 192 196 186 215 201 184  
231 55 122 218 112 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 0 0 0 210 207 195 200 186 212 208 188  
210 147 0 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 2 0 44 237 205 197 204 190 211 208 201  
191 207 0 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 0 2 0 110 208 208 199 207 193 207 213 211  
188 234 24 0 3 0 0 0 0 0]  
[ 0 0 0 0 0 0 2 0 0 184 203 212 199 212 193 208 223 216  
185 205 71 0 3 0 0 0 0 0]  
[ 0 0 0 0 0 0 2 0 0 224 198 226 199 215 191 210 231 216  
170 209 110 0 2 0 0 0 0 0]  
[ 0 0 0 0 0 0 2 0 0 237 197 231 204 215 202 208 244 220  
170 213 128 0 1 0 0 0 0 0]  
[ 0 0 0 0 0 0 1 0 0 245 196 230 209 201 202 209 246 213  
169 214 150 0 1 0 0 0 0 0]  
[ 0 0 0 0 0 0 2 0 3 248 192 230 208 186 184 213 253 214  
173 212 189 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 2 0 15 217 188 231 210 186 186 219 255 214  
177 210 227 0 0 0 0 0 0 0]  
[ 0 0 0 0 0 0 3 0 49 222 183 235 207 188 184 220 255 215  
179 207 206 0 0 0 0 0 0 0]

```
[ 0 0 0 0 0 0 2 0 87 225 179 239 204 189 183 221 255 214
180 205 218 15 0 0 0 0 0 0]
[ 0 0 0 0 0 0 2 0 129 223 177 224 198 187 178 217 254 216
192 211 242 78 0 0 0 0 0 0]
[ 0 0 0 0 0 0 3 0 156 224 183 255 231 205 196 250 255 254
224 205 177 75 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 4 20 0 21 122 184 167 118 45 27
12 0 0 0 0 0 0 0 0 0]]
```

3



Train Shape : (60000, 28, 28, 1)

Test Shape : (10000, 28, 28, 1)

y\_train shape : (60000,)

y\_test shape : (10000,)

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0

flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204928
dense_1 (Dense)	(None, 10)	1290

=====

Total params: 243,786

Trainable params: 243,786

Non-trainable params: 0

---

Epoch 1/3

1875/1875 [=====] - 40s 21ms/step - loss: 0.4556 - accuracy: 0.8336 - val\_loss: 0.3248 - val\_accuracy: 0.8850

Epoch 2/3

1875/1875 [=====] - 41s 22ms/step - loss: 0.3029 - accuracy: 0.8895 - val\_loss: 0.2959 - val\_accuracy: 0.8954

Epoch 3/3

1875/1875 [=====] - 46s 24ms/step - loss: 0.2564 - accuracy: 0.9050 - val\_loss: 0.2488 - val\_accuracy: 0.9089

313/313 [=====] - 2s 6ms/step

[6.3769415e-04 1.5261188e-06 2.7142337e-04 9.9646181e-01 1.6428892e-03  
1.6867384e-06 9.5685822e-04 2.2876253e-07 2.5509478e-05 3.7153288e-07]

Actual label : 3

Predicted label : 3

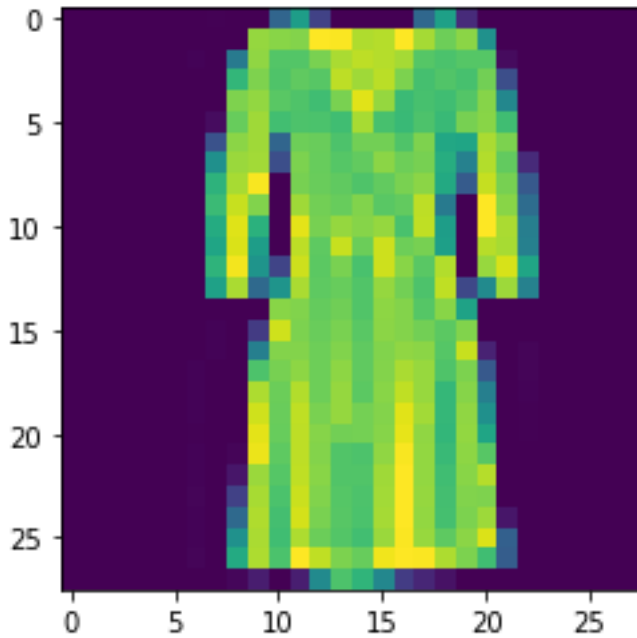
Class : Dress

313/313 [=====] - 2s 6ms/step - loss: 0.2488 - accuracy: 0.9089

Loss : 0.24884097278118134

Accuracy (Test Data) : 90.89000225067139





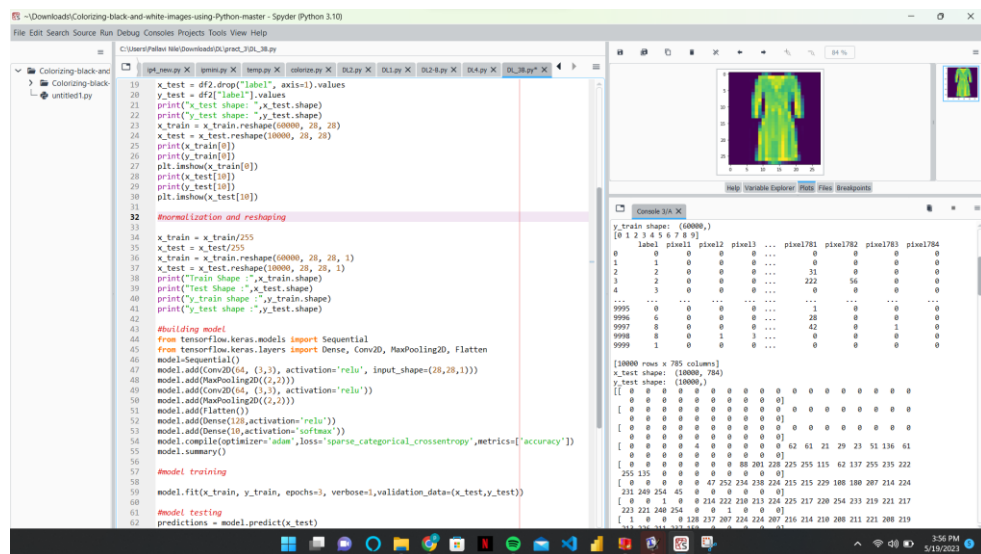
```

-Downloads/Coloring-black-and-white-images-using-Python-master - Spyder (Python 3.10)
File Edit Search Source Run Debug Console Projects Tools View Help
C:\Users\Pellaw\Downloads\DL\print_2D_3D.py

37 x_test = x_test.reshape((10000, 28, 28, 1))
38 print("Train Shape :", x_train.shape)
39 print("Test Shape :", x_test.shape)
40 print("y_train shape :", y_train.shape)
41 print("y_test shape :", y_test.shape)
42
43
44 #Building model
45 from tensorflow.keras.models import Sequential
46 from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
47 model = Sequential()
48 model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
49 model.add(MaxPooling2D((2, 2)))
50 model.add(Conv2D(64, (3, 3), activation='relu'))
51 model.add(MaxPooling2D((2, 2)))
52 model.add(Flatten())
53 model.add(Dense(10, activation='softmax'))
54 model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
55 model.summary()
56
57 #Model training
58 model.fit(x_train, y_train, epochs=1, verbose=1, validation_data=(x_test, y_test))
59
60 #Model testing
61 predictions = model.predict(x_test)
62
63 import numpy as np
64 index=10
65 print(predictions[index])
66 final_value=np.argmax(predictions[index])
67 print("Actual label :", y_test[index])
68 print("Predicted label :", final_value)
69 print("Class :", class_names[final_value])
70
71 plt.imshow(x_test[10])
72
73 #Evaluation of model
74 loss, accuracy = model.evaluate(x_test, y_test)
75 print("Loss :", loss)
76 print("Accuracy (Test Data) :", accuracy*100)
77
78
79
80

```

Epoch 1/1  
1875/1875 [=====] - 40s 21ms/step - loss: 0.4556 - accuracy: 0.8936 - val\_loss: 0.3248 - val\_accuracy: 0.8859  
Epoch 2/3  
1875/1875 [=====] - 40s 22ms/step - loss: 0.3829 - accuracy: 0.8895 - val\_loss: 0.2959 - val\_accuracy: 0.8954  
Epoch 3/3  
1875/1875 [=====] - 40s 24ms/step - loss: 0.2564 - accuracy: 0.9058 - val\_loss: 0.2488 - val\_accuracy: 0.9089  
313/313 [=====] - 2s 6ms/step  
[6.3709415e-04 1.5261189e-06 2.7342107e-04 9.3648181e-01 1.6428892e-03 1.480158e-06 5.505822e-04 2.267623e-07 2.509479e-05 3.7211286e-07]  
Actual label : 3  
Predicted label : 3  
Class : Dress  
313/313 [=====] - 2s 6ms/step - loss: 0.2488 - accuracy: 0.9089  
Loss : 0.2488489778118134  
Accuracy (Test Data) : 90.89080223807319



Download\Colorizing-black-and-white-images-using-Python-master - Spyder (Python 3.10)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Pallavi Nile\Downloads\DL\pract\_3\DL\_38.py

```
1 #File Pallavi Nile NO: 4217 Div:B
2 # DL Practical MG-38
3
4 import pandas as pd
5 import numpy as np
6 import matplotlib.pyplot as plt
7 class_names= ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker']
8 df1 = pd.read_csv(r"C:\Users\Pallavi Nile\Downloads\DL\pract_3\Fashion-mnist_train.csv")
9 print(df1)
10
11 x_train = df1.drop("label", axis=1).values
12 y_train = df1["label"].values
13 print("x_train shape: ", x_train.shape)
14 print("y_train shape: ", y_train.shape)
15 print(np.unique(y_train))
16 df2 = pd.read_csv(r"C:\Users\Pallavi Nile\Downloads\DL\pract_3\Fashion-mnist_test.csv")
17 print(df2)
18
19 x_test = df2.drop("label", axis=1).values
20 y_test = df2["label"].values
21 print("x_test shape: ", x_test.shape)
22 print("y_test shape: ", y_test.shape)
23 x_train = x_train.reshape(60000, 28, 28)
24 x_test = x_test.reshape(10000, 28, 28)
25 print(x_train[0])
26 print(y_train[0])
27 plt.imshow(x_train[0])
28 print(x_test[10])
29 print(y_test[10])
30 plt.imshow(x_test[10])
31
32 #normalization and reshaping
33
34 x_train = x_train/255
35 x_test = x_test/255
36 x_train = x_train.reshape(60000, 28, 28, 1)
37 x_test = x_test.reshape(10000, 28, 28, 1)
38 print("Train Shape :", x_train.shape)
39 print("Test Shape :", x_test.shape)
40 print("y_train shape :", y_train.shape)
41 print("y_test shape :", y_test.shape)
42
43 #Building model
44 from tensorflow.keras.models import Sequential
```

Help Variable Explorer Plot File Breakpoints

In [1]: runfile('C:/Users/Pallavi Nile/Downloads/DL/pract\_3/DL\_38.py', wdir='C:/Users/Pallavi Nile/Downloads/DL/pract\_3')

label	pixel1	pixel2	pixel3	...	pixel1781	pixel1782	pixel1783	pixel1784
0	2	0	0	...	0	0	0	0
1	9	0	0	...	0	0	0	0
2	6	0	0	...	0	0	0	0
3	0	0	0	...	0	0	0	0
4	3	0	0	...	0	0	0	0
...	...	...	...	...	...	...	...	...
59995	9	0	0	...	0	0	0	0
59996	1	0	0	...	0	0	0	0
59997	8	0	0	...	0	0	0	0
59998	8	0	0	...	0	0	0	0
59999	7	0	0	...	0	0	0	0

[60000 rows x 785 columns]  
x\_train shape: (60000, 784)  
y\_train shape: (60000,)  
[0 1 2 3 4 5 6 7 8 9]

label	pixel1	pixel2	pixel3	...	pixel1781	pixel1782	pixel1783	pixel1784
0	0	0	0	...	0	0	0	0
1	1	0	0	...	0	0	0	0
2	2	0	0	...	31	0	0	0
3	2	0	0	...	222	56	0	0
4	3	0	0	...	0	0	0	0

