Deep Learning Lab Assignment 03

Name of Student: Aniket Yashvant Sandbhor
Branch: Computer Engineering
Exam number: B190424404

0.1 Implement Classification using Convolution Neural Network.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from keras.datasets import fashion_mnist
import tensorflow.keras as tk
```

[2]:
```
%matplotlib inline
```

[3]:
```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

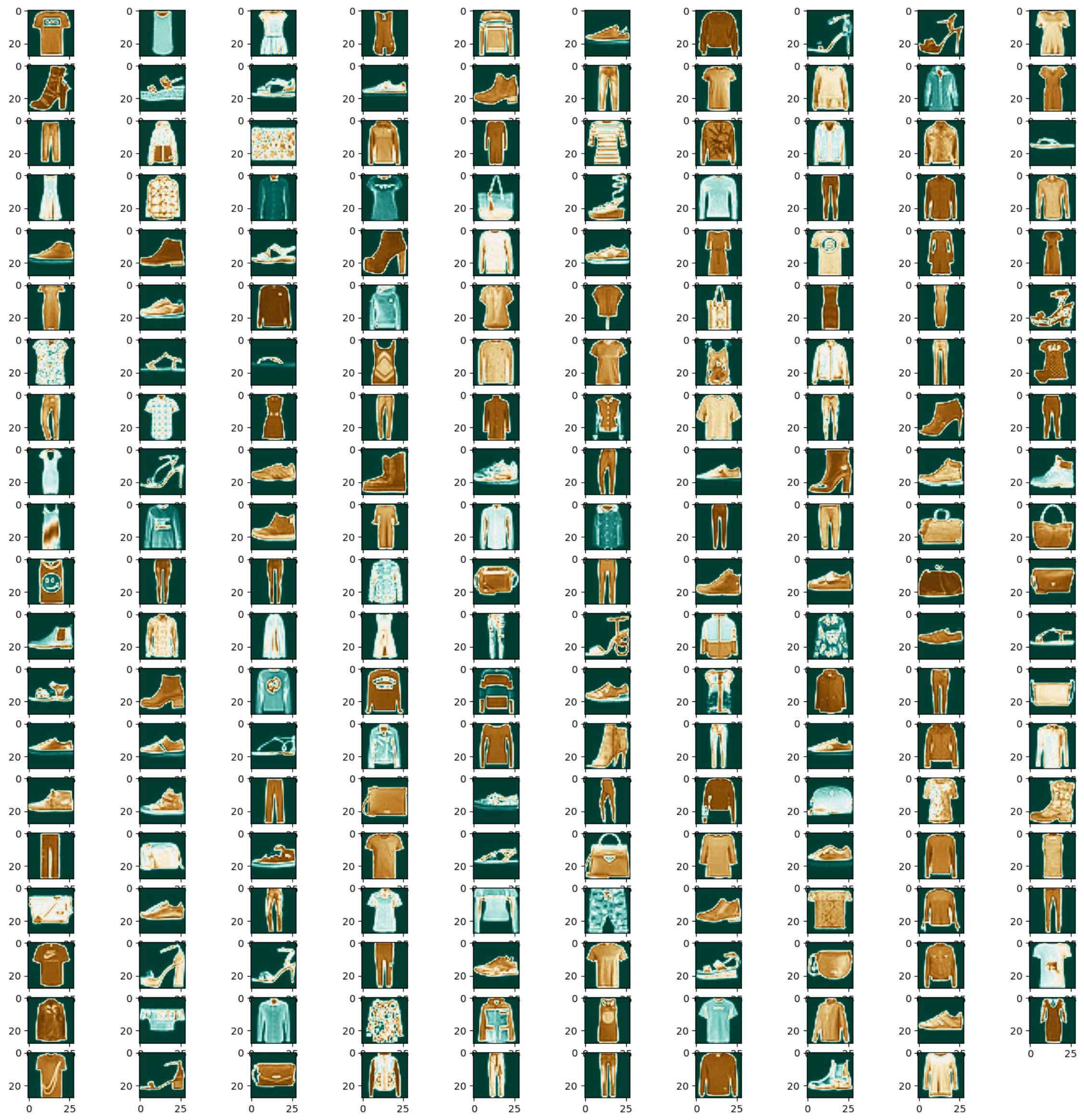
```
Downloading data from https://storage.googleapis.com/tensorflow/
 ↪tf-keras-
datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/
 ↪tf-keras-
datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 3s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/
 ↪tf-keras-
datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/
 ↪tf-keras-
datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
[4]: display(x_train.shape,x_test.shape)
```

```
(60000, 28, 28)

(10000, 28, 28)
```

```
[5]: figure=plt.figure(figsize=(20,20))
     for i in range(1,200):
       plt.subplot(20,10,i)
       plt.imshow(x_train[i],cmap=plt.get_cmap('BrBG_r'))
     plt.show()
```



```
[6]: cnn_model = tk.Sequential()

     cnn_model.add(tk.layers.Conv2D(32,3,3,input_shape =
      ↪(28,28,1),activation = 'relu'))

         # Max pooling will reduce the
         # size with a kernal size of 2x2
     cnn_model.add(tk.layers.MaxPooling2D(pool_size= (2,2)))

         # Once the convolutional and pooling
         # operations are done the layer
         # is flattened and fully connected layers
         # are added
     cnn_model.add(tk.layers.Flatten())

     cnn_model.add(tk.layers.Dense(32,activation = 'relu'))
     cnn_model.add(tk.layers.Dense(10,activation = 'softmax'))
```

```
[7]: cnn_model.
      ↪compile(optimizer='Adam',loss='sparse_categorical_crossentropy',metrics=['accura
```

```
[8]: cnn_model.fit(x=x_train,y=y_train,batch_size =512,epochs = 50,verbose
      ↪= 1,validation_data = (x_test,y_test))
```

```
Epoch 1/50
  1/118 [...] - ETA: 17s - loss: 17.5993 - accuracy:
0.1309

2024-04-11 03:04:05.027603: W
tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get
 ↪CPU
frequency: 0 Hz

118/118 [==============================] - 1s 5ms/step - loss: 1.9728 -
accuracy: 0.5292 - val_loss: 1.1078 - val_accuracy: 0.6329
Epoch 2/50
118/118 [==============================] - 1s 5ms/step - loss: 0.9289 -
accuracy: 0.6850 - val_loss: 0.7940 - val_accuracy: 0.7185
Epoch 3/50
118/118 [==============================] - 1s 5ms/step - loss: 0.6656 -
accuracy: 0.7391 - val_loss: 0.6508 - val_accuracy: 0.7529
Epoch 4/50
118/118 [==============================] - 1s 5ms/step - loss: 0.5816 -
accuracy: 0.7768 - val_loss: 0.5854 - val_accuracy: 0.7793
Epoch 5/50
118/118 [==============================] - 1s 5ms/step - loss: 0.5196 -
accuracy: 0.8068 - val_loss: 0.5426 - val_accuracy: 0.7991
Epoch 6/50
118/118 [==============================] - 1s 5ms/step - loss: 0.4810 -
```

```
accuracy: 0.8217 - val_loss: 0.5286 - val_accuracy: 0.8077
Epoch 7/50
118/118 [==============================] - 1s 5ms/step - loss: 0.4595 -
accuracy: 0.8292 - val_loss: 0.5076 - val_accuracy: 0.8134
Epoch 8/50
118/118 [==============================] - 1s 5ms/step - loss: 0.4416 -
accuracy: 0.8361 - val_loss: 0.5128 - val_accuracy: 0.8169
Epoch 9/50
118/118 [==============================] - 1s 5ms/step - loss: 0.4253 -
accuracy: 0.8417 - val_loss: 0.4860 - val_accuracy: 0.8241
Epoch 10/50
118/118 [==============================] - 1s 5ms/step - loss: 0.4133 -
accuracy: 0.8467 - val_loss: 0.4783 - val_accuracy: 0.8239
Epoch 11/50
118/118 [==============================] - 1s 5ms/step - loss: 0.4039 -
accuracy: 0.8502 - val_loss: 0.4684 - val_accuracy: 0.8301
Epoch 12/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3964 -
accuracy: 0.8530 - val_loss: 0.4725 - val_accuracy: 0.8244
Epoch 13/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3897 -
accuracy: 0.8551 - val_loss: 0.4620 - val_accuracy: 0.8315
Epoch 14/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3847 -
accuracy: 0.8566 - val_loss: 0.4587 - val_accuracy: 0.8346
Epoch 15/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3758 -
accuracy: 0.8605 - val_loss: 0.4562 - val_accuracy: 0.8370
Epoch 16/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3721 -
accuracy: 0.8620 - val_loss: 0.4603 - val_accuracy: 0.8324
Epoch 17/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3664 -
accuracy: 0.8627 - val_loss: 0.4527 - val_accuracy: 0.8375
Epoch 18/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3618 -
accuracy: 0.8648 - val_loss: 0.4482 - val_accuracy: 0.8424
Epoch 19/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3562 -
accuracy: 0.8673 - val_loss: 0.4494 - val_accuracy: 0.8402
Epoch 20/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3545 -
accuracy: 0.8665 - val_loss: 0.4537 - val_accuracy: 0.8367
Epoch 21/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3481 -
accuracy: 0.8699 - val_loss: 0.4436 - val_accuracy: 0.8411
Epoch 22/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3473 -
```

```
accuracy: 0.8698 - val_loss: 0.4531 - val_accuracy: 0.8404
Epoch 23/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3444 -
accuracy: 0.8704 - val_loss: 0.4519 - val_accuracy: 0.8391
Epoch 24/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3397 -
accuracy: 0.8722 - val_loss: 0.4420 - val_accuracy: 0.8465
Epoch 25/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3347 -
accuracy: 0.8742 - val_loss: 0.4430 - val_accuracy: 0.8440
Epoch 26/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3326 -
accuracy: 0.8753 - val_loss: 0.4493 - val_accuracy: 0.8449
Epoch 27/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3320 -
accuracy: 0.8745 - val_loss: 0.4545 - val_accuracy: 0.8428
Epoch 28/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3286 -
accuracy: 0.8764 - val_loss: 0.4512 - val_accuracy: 0.8459
Epoch 29/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3235 -
accuracy: 0.8776 - val_loss: 0.4514 - val_accuracy: 0.8439
Epoch 30/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3212 -
accuracy: 0.8789 - val_loss: 0.4408 - val_accuracy: 0.8450
Epoch 31/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3201 -
accuracy: 0.8796 - val_loss: 0.4501 - val_accuracy: 0.8474
Epoch 32/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3156 -
accuracy: 0.8802 - val_loss: 0.4715 - val_accuracy: 0.8405
Epoch 33/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3185 -
accuracy: 0.8790 - val_loss: 0.4506 - val_accuracy: 0.8452
Epoch 34/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3123 -
accuracy: 0.8826 - val_loss: 0.4596 - val_accuracy: 0.8431
Epoch 35/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3109 -
accuracy: 0.8824 - val_loss: 0.4415 - val_accuracy: 0.8486
Epoch 36/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3120 -
accuracy: 0.8823 - val_loss: 0.4478 - val_accuracy: 0.8474
Epoch 37/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3082 -
accuracy: 0.8834 - val_loss: 0.4474 - val_accuracy: 0.8486
Epoch 38/50
118/118 [==============================] - 1s 5ms/step - loss: 0.3039 -
```

```
accuracy: 0.8850 - val_loss: 0.4508 - val_accuracy: 0.8476
Epoch 39/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3027 -
accuracy: 0.8855 - val_loss: 0.4547 - val_accuracy: 0.8492
Epoch 40/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3006 -
accuracy: 0.8858 - val_loss: 0.4634 - val_accuracy: 0.8449
Epoch 41/50
118/118 [==============================] - 1s 6ms/step - loss: 0.3031 -
accuracy: 0.8848 - val_loss: 0.4587 - val_accuracy: 0.8475
Epoch 42/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2985 -
accuracy: 0.8866 - val_loss: 0.4581 - val_accuracy: 0.8462
Epoch 43/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2931 -
accuracy: 0.8880 - val_loss: 0.4561 - val_accuracy: 0.8481
Epoch 44/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2949 -
accuracy: 0.8878 - val_loss: 0.4621 - val_accuracy: 0.8486
Epoch 45/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2921 -
accuracy: 0.8888 - val_loss: 0.4610 - val_accuracy: 0.8504
Epoch 46/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2933 -
accuracy: 0.8877 - val_loss: 0.4712 - val_accuracy: 0.8457
Epoch 47/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2910 -
accuracy: 0.8891 - val_loss: 0.4748 - val_accuracy: 0.8474
Epoch 48/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2882 -
accuracy: 0.8906 - val_loss: 0.4740 - val_accuracy: 0.8486
Epoch 49/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2857 -
accuracy: 0.8911 - val_loss: 0.4709 - val_accuracy: 0.8480
Epoch 50/50
118/118 [==============================] - 1s 6ms/step - loss: 0.2855 -
accuracy: 0.8915 - val_loss: 0.4680 - val_accuracy: 0.8486
```

```
[8]: <keras.callbacks.History at 0x30905aed0>
```

```
[9]: evaluation = cnn_model.evaluate(x_test,y_test)
     print('Test Accuracy : {:.3f}'.format(evaluation[1]))
```

```
313/313 [==============================] - 0s 600us/step - loss: 0.
 ↪4680 -
accuracy: 0.8486
Test Accuracy : 0.849
```

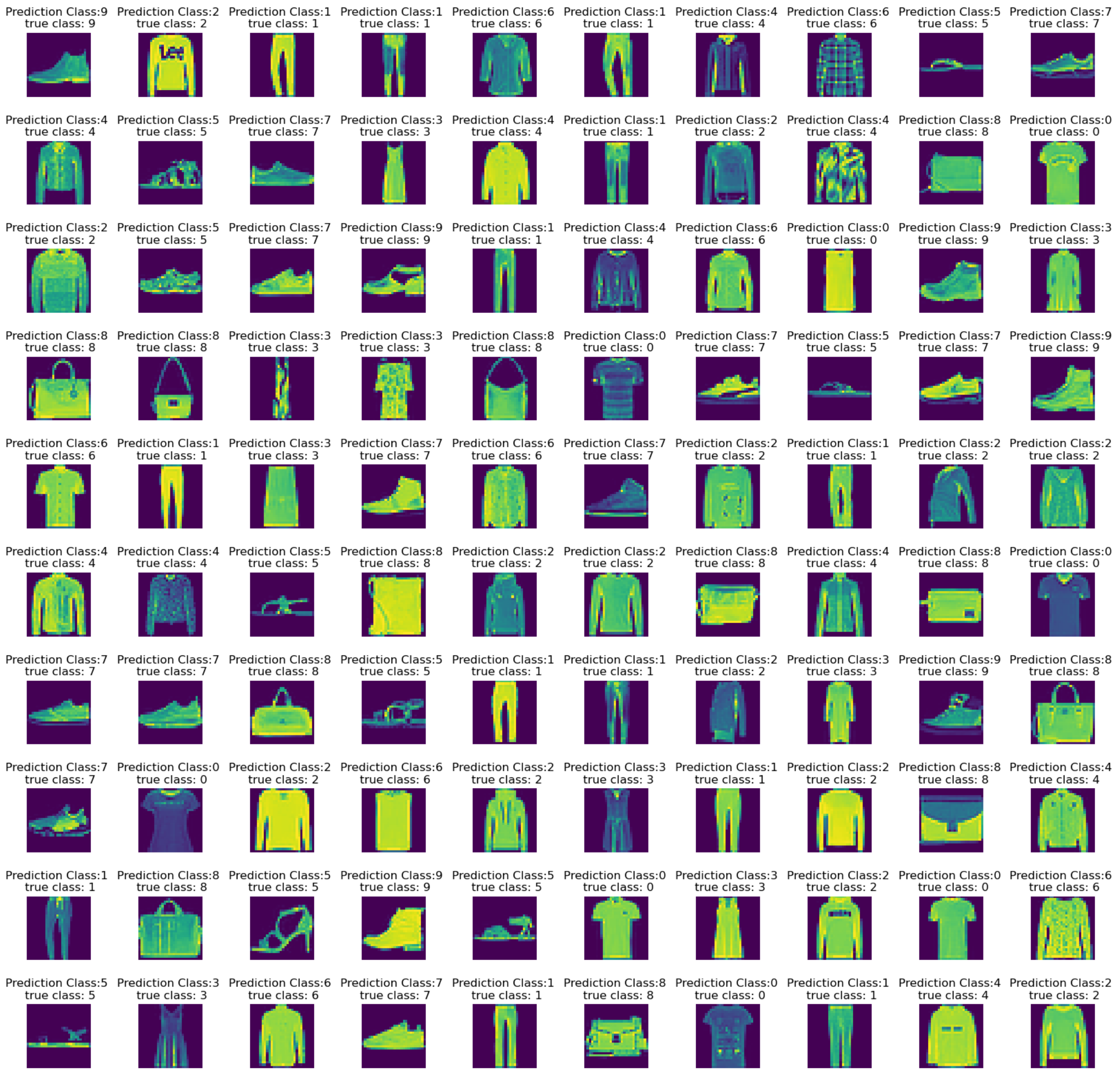
```
[10]: predicted_classes = np.argmax(cnn_model.predict(x_test),axis=-1)

      313/313 [==============================] - 0s 568us/step

[11]: predicted_classes

[11]: array([9, 2, 1, ..., 8, 1, 5])

[12]: L = 10
      W = 10
      fig,axes = plt.subplots(L,W,figsize = (20,20))
      axes = axes.ravel()
      for i in np.arange(0,L*W):
          axes[i].imshow(x_test[i].reshape(28,28))
          axes[i].set_title('Prediction Class:{1} \n true class: {1}'.
       ↪format(predicted_classes[i], y_test[i]))
          axes[i].axis('off')
      plt.subplots_adjust(wspace = 0.75)
```


Prediction Class:9
true class: 9
Prediction Class:2
true class: 2
Prediction Class:1
true class: 1
Prediction Class:1
true class: 1
Prediction Class:6
true class: 6
Prediction Class:1
true class: 1
Prediction Class:4
true class: 4
Prediction Class:6
true class: 6
Prediction Class:5
true class: 5
Prediction Class:7
true class: 7
Prediction Class:4
true class: 4
Prediction Class:5
true class: 5
Prediction Class:7
true class: 7
Prediction Class:3
true class: 3
Prediction Class:4
true class: 4
Prediction Class:1
true class: 1
Prediction Class:2
true class: 2
Prediction Class:4
true class: 4
Prediction Class:8
true class: 8
Prediction Class:0
true class: 0
Prediction Class:2
true class: 2
Prediction Class:5
true class: 5
Prediction Class:7
true class: 7
Prediction Class:9
true class: 9
Prediction Class:1
true class: 1
Prediction Class:4
true class: 4
Prediction Class:6
true class: 6
Prediction Class:0
true class: 0
Prediction Class:9
true class: 9
Prediction Class:3
true class: 3
Prediction Class:8
true class: 8
Prediction Class:8
true class: 8
Prediction Class:3
true class: 3
Prediction Class:3
true class: 3
Prediction Class:8
true class: 8
Prediction Class:0
true class: 0
Prediction Class:7
true class: 7
Prediction Class:5
true class: 5
Prediction Class:7
true class: 7
Prediction Class:9
true class: 9
Prediction Class:6
true class: 6
Prediction Class:1
true class: 1
Prediction Class:3
true class: 3
Prediction Class:7
true class: 7
Prediction Class:6
true class: 6
Prediction Class:7
true class: 7
Prediction Class:2
true class: 2
Prediction Class:1
true class: 1
Prediction Class:2
true class: 2
Prediction Class:2
true class: 2
Prediction Class:4
true class: 4
Prediction Class:4
true class: 4
Prediction Class:5
true class: 5
Prediction Class:8
true class: 8
Prediction Class:2
true class: 2
Prediction Class:2
true class: 2
Prediction Class:8
true class: 8
Prediction Class:4
true class: 4
Prediction Class:8
true class: 8
Prediction Class:0
true class: 0
Prediction Class:7
true class: 7
Prediction Class:7
true class: 7
Prediction Class:8
true class: 8
Prediction Class:5
true class: 5
Prediction Class:1
true class: 1
Prediction Class:1
true class: 1
Prediction Class:2
true class: 2
Prediction Class:3
true class: 3
Prediction Class:9
true class: 9
Prediction Class:8
true class: 8
Prediction Class:7
true class: 7
Prediction Class:0
true class: 0
Prediction Class:2
true class: 2
Prediction Class:6
true class: 6
Prediction Class:2
true class: 2
Prediction Class:3
true class: 3
Prediction Class:1
true class: 1
Prediction Class:2
true class: 2
Prediction Class:8
true class: 8
Prediction Class:4
true class: 4
Prediction Class:1
true class: 1
Prediction Class:8
true class: 8
Prediction Class:5
true class: 5
Prediction Class:9
true class: 9
Prediction Class:5
true class: 5
Prediction Class:0
true class: 0
Prediction Class:3
true class: 3
Prediction Class:2
true class: 2
Prediction Class:0
true class: 0
Prediction Class:6
true class: 6
Prediction Class:5
true class: 5
Prediction Class:3
true class: 3
Prediction Class:6
true class: 6
Prediction Class:7
true class: 7
Prediction Class:1
true class: 1
Prediction Class:8
true class: 8
Prediction Class:0
true class: 0
Prediction Class:1
true class: 1
Prediction Class:4
true class: 4
Prediction Class:2
true class: 2