```
print("Name : Bhagvat Nivrutti Mutthe ")
print("Roll No : BCB-76")
print("Assignment no.3")
print("CONVOLUTIONAL NEURAL NETWORK (CNN)")
```

```
Name : Bhagvat Nivrutti Mutthe
Roll No : BCB-76
Assignment no.3
CONVOLUTIONAL NEURAL NETWORK (CNN)
```

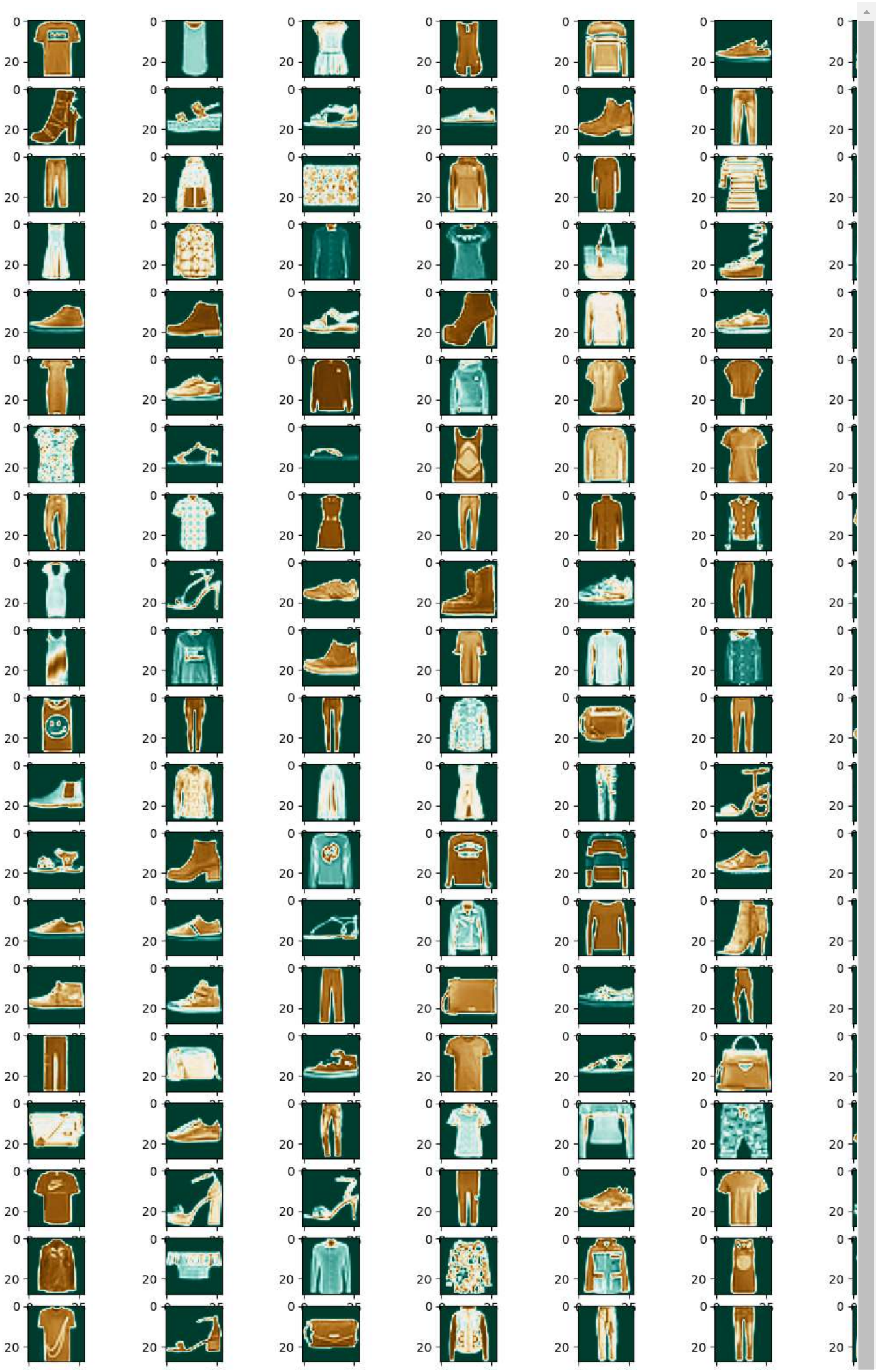
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from keras.datasets import fashion_mnist
import tensorflow.keras as tk
%matplotlib inline
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
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 [==============================] - 0s 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
```

```
%matplotlib inline
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
(60000, 28, 28)
(10000, 28, 28)
```

```
(10000, 28, 28)
```

```
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()
```



```
0  25  0  25  0  25  0  25  0  25  0  25  0
```

```
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'))
cnn_model.compile(optimizer='Adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
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
118/118 [==============================] - 3s 20ms/step - loss: 5.1860 - accuracy: 0.3564 - val_loss: 1.4453 - val_accuracy: 0.49
Epoch 2/50
118/118 [==============================] - 2s 19ms/step - loss: 1.1896 - accuracy: 0.5916 - val_loss: 1.0197 - val_accuracy: 0.64
Epoch 3/50
118/118 [==============================] - 3s 26ms/step - loss: 0.8962 - accuracy: 0.6879 - val_loss: 0.8493 - val_accuracy: 0.70
Epoch 4/50
118/118 [==============================] - 3s 21ms/step - loss: 0.7511 - accuracy: 0.7314 - val_loss: 0.7438 - val_accuracy: 0.72
Epoch 5/50
118/118 [==============================] - 2s 19ms/step - loss: 0.6649 - accuracy: 0.7584 - val_loss: 0.6731 - val_accuracy: 0.75
Epoch 6/50
118/118 [==============================] - 2s 19ms/step - loss: 0.6113 - accuracy: 0.7788 - val_loss: 0.6303 - val_accuracy: 0.77
Epoch 7/50
118/118 [==============================] - 2s 19ms/step - loss: 0.5717 - accuracy: 0.7916 - val_loss: 0.6188 - val_accuracy: 0.78
Epoch 8/50
118/118 [==============================] - 3s 23ms/step - loss: 0.5448 - accuracy: 0.8009 - val_loss: 0.5851 - val_accuracy: 0.78
Epoch 9/50
118/118 [==============================] - 3s 24ms/step - loss: 0.5162 - accuracy: 0.8109 - val_loss: 0.5680 - val_accuracy: 0.80
Epoch 10/50
118/118 [==============================] - 2s 19ms/step - loss: 0.4940 - accuracy: 0.8187 - val_loss: 0.5569 - val_accuracy: 0.80
Epoch 11/50
118/118 [==============================] - 2s 21ms/step - loss: 0.4826 - accuracy: 0.8220 - val_loss: 0.5332 - val_accuracy: 0.81
Epoch 12/50
118/118 [==============================] - 4s 30ms/step - loss: 0.4639 - accuracy: 0.8302 - val_loss: 0.5256 - val_accuracy: 0.81
Epoch 13/50
118/118 [==============================] - 3s 29ms/step - loss: 0.4517 - accuracy: 0.8337 - val_loss: 0.5154 - val_accuracy: 0.81
Epoch 14/50
118/118 [==============================] - 3s 23ms/step - loss: 0.4388 - accuracy: 0.8383 - val_loss: 0.5094 - val_accuracy: 0.82
Epoch 15/50
118/118 [==============================] - 2s 19ms/step - loss: 0.4303 - accuracy: 0.8420 - val_loss: 0.5120 - val_accuracy: 0.82
Epoch 16/50
118/118 [==============================] - 2s 19ms/step - loss: 0.4229 - accuracy: 0.8460 - val_loss: 0.4851 - val_accuracy: 0.83
Epoch 17/50
118/118 [==============================] - 3s 25ms/step - loss: 0.4105 - accuracy: 0.8493 - val_loss: 0.4807 - val_accuracy: 0.83
Epoch 18/50
118/118 [==============================] - 4s 37ms/step - loss: 0.4058 - accuracy: 0.8507 - val_loss: 0.4774 - val_accuracy: 0.83
Epoch 19/50
118/118 [==============================] - 2s 19ms/step - loss: 0.3973 - accuracy: 0.8544 - val_loss: 0.4671 - val_accuracy: 0.83
Epoch 20/50
118/118 [==============================] - 2s 19ms/step - loss: 0.3919 - accuracy: 0.8558 - val_loss: 0.4802 - val_accuracy: 0.83
Epoch 21/50
118/118 [==============================] - 2s 19ms/step - loss: 0.3848 - accuracy: 0.8578 - val_loss: 0.4649 - val_accuracy: 0.83
Epoch 22/50
118/118 [==============================] - 3s 26ms/step - loss: 0.3775 - accuracy: 0.8612 - val_loss: 0.4599 - val_accuracy: 0.83
Epoch 23/50
118/118 [==============================] - 3s 22ms/step - loss: 0.3734 - accuracy: 0.8626 - val_loss: 0.4681 - val_accuracy: 0.83
Epoch 24/50
118/118 [==============================] - 2s 19ms/step - loss: 0.3710 - accuracy: 0.8636 - val_loss: 0.4855 - val_accuracy: 0.83
Epoch 25/50
118/118 [==============================] - 2s 19ms/step - loss: 0.3648 - accuracy: 0.8645 - val_loss: 0.4575 - val_accuracy: 0.84
Epoch 26/50
118/118 [==============================] - 3s 24ms/step - loss: 0.3580 - accuracy: 0.8677 - val_loss: 0.4945 - val_accuracy: 0.82
Epoch 27/50
118/118 [==============================] - 4s 32ms/step - loss: 0.3597 - accuracy: 0.8667 - val_loss: 0.4529 - val_accuracy: 0.84
Epoch 28/50
118/118 [==============================] - 4s 30ms/step - loss: 0.3512 - accuracy: 0.8711 - val_loss: 0.4622 - val_accuracy: 0.84
Epoch 29/50
```

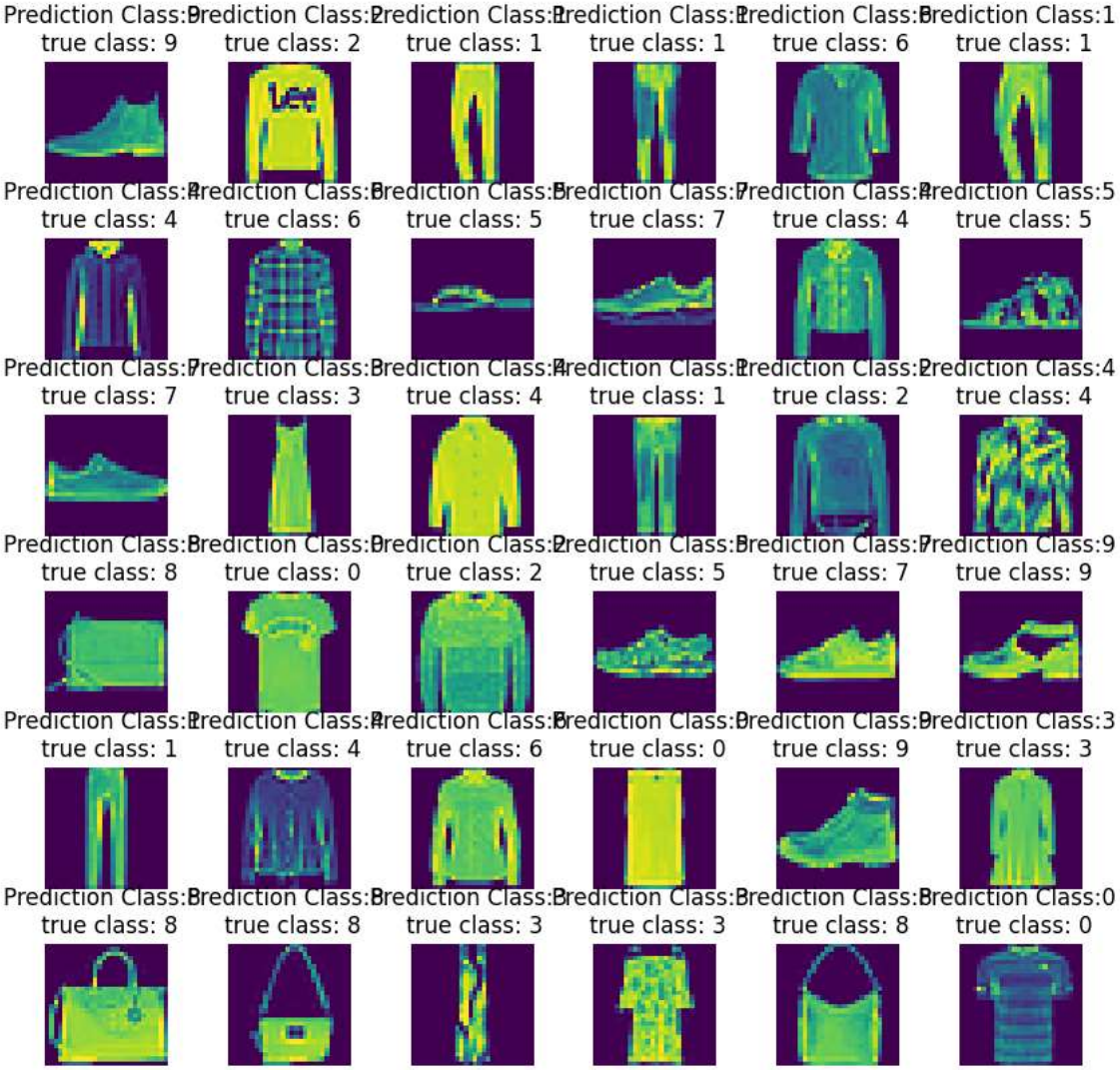
```
evaluation = cnn_model.evaluate(x_test,y_test)
print('Test Accuracy : {:.3f}'.format(evaluation[1]))
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.4578 - accuracy: 0.8458
Test Accuracy : 0.846
```

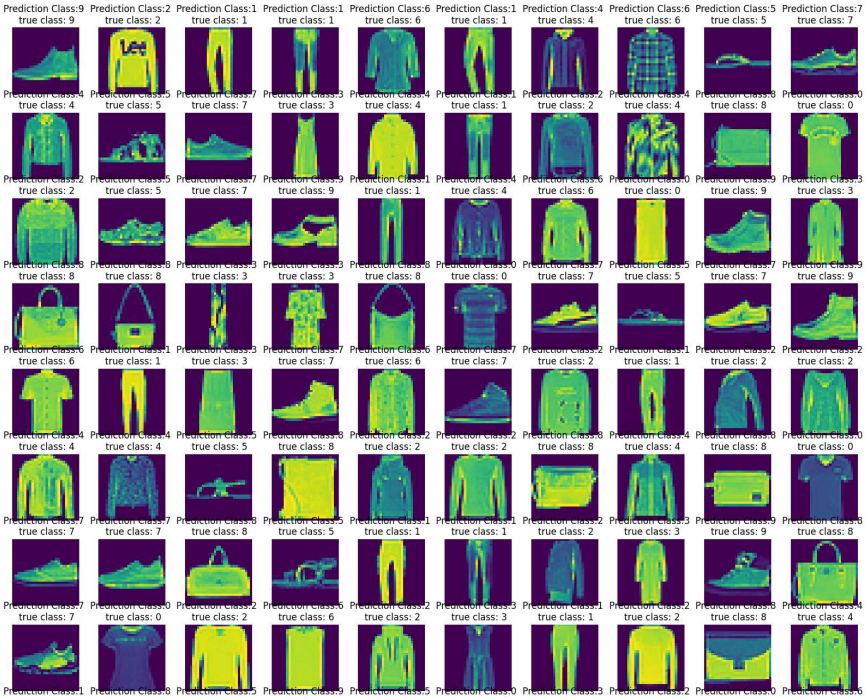
```
predicted_classes = np.argmax(cnn_model.predict(x_test),axis=-1)
```

```
313/313 [==============================] - 1s 2ms/step
```

```
L = 6
W = 6
fig,axes = plt.subplots(L,W,figsize = (10,10))
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 = 0.50)
```




```
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 = 0.30)
```


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