CS5590/490 - Python and Deep Learning Programming
In Class Programming Report - 8
Class ID 24 - Anurag Thantharate
Date Submitted: 04/05/2020

Video Link: https://www.loom.com/share/3921c34d9d9f429aa0af1df8f7e94505

---

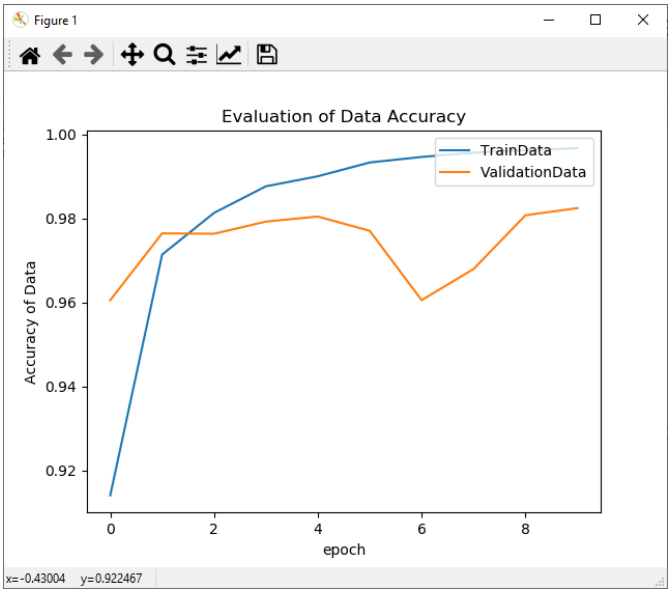
1. Using the history object in the source code, plot the loss and accuracy for both training data and validation data.

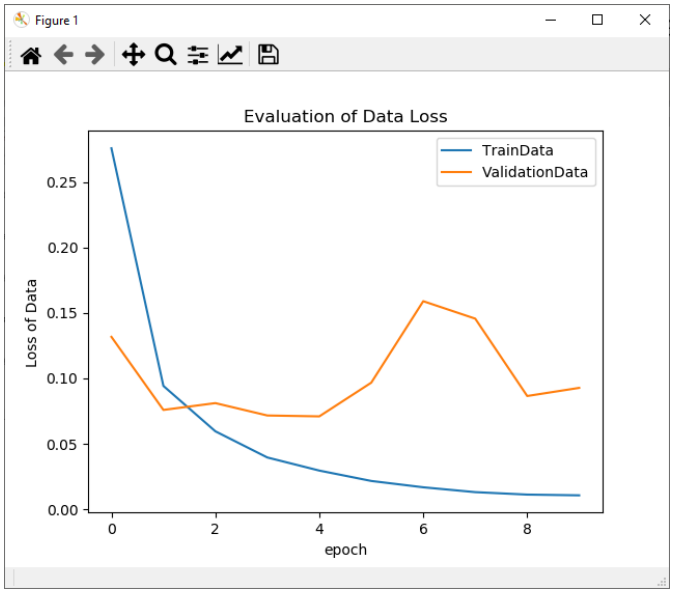
1_History_Loass_Accuracy.py | 2_SingleImage_Loss_Accuracy.py | 3_Change_Activation.py | 4_NoScaling.py | Bonus.py

```python
# Importing libraries
from keras import Sequential
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense
from keras.utils import to_categorical

# Loading input data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Display the first image in the training data
print(train_images.shape[1:])

# Process the data
# 1. Convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')

# Scale data
train_data /=255.0
test_data /=255.0

# Change the labels from integer to one-hot encoding
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1, validation_data=(
    test_data, test_labels_one_hot))

```

```python
# Graphical evaluation of accuracy associated with training and validation data
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Evaluation of Data Accuracy')
plt.xlabel('epoch')
plt.ylabel('Accuracy of Data')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()

# Graphical evaluation of loss associated with training and validation data
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('Loss of Data')
plt.title('Evaluation of Data Loss')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()
```

```
57600/60000 [===========================>..] - ETA: 0s - loss: 0.0099 - accuracy: 0.9966
58112/60000 [===========================>.] - ETA: 0s - loss: 0.0099 - accuracy: 0.9967
58624/60000 [===========================>.] - ETA: 0s - loss: 0.0098 - accuracy: 0.9967
59136/60000 [===========================>.] - ETA: 0s - loss: 0.0099 - accuracy: 0.9967
59648/60000 [===========================>.] - ETA: 0s - loss: 0.0098 - accuracy: 0.9967
60000/60000 [==============================] - 7s 114us/step - loss: 0.0098 - accuracy: 0.9967 - val_loss: 0.0857 - val_accuracy: 0.9826

Process finished with exit code 0
```



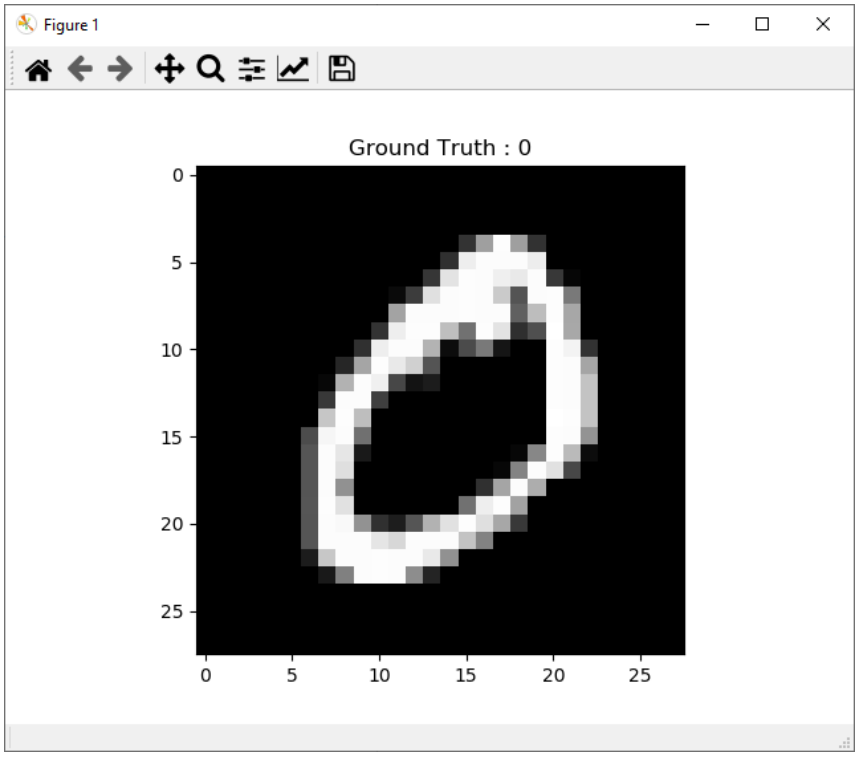


2. Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image in the test data

DLICP2 〉 2_SingleImage_Loss_Accuracy.py

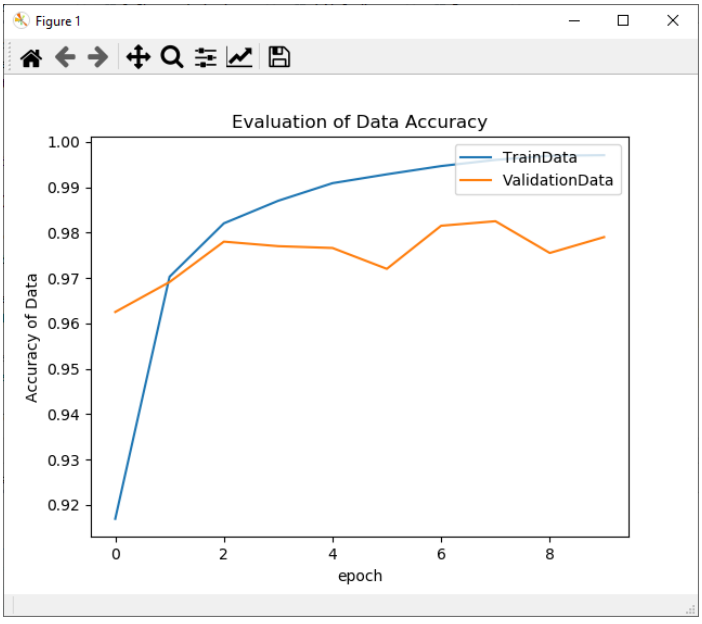
1_History_Loass_Accuracy.py | 2_SingleImage_Loss_Accuracy.py | 3_Change_Activation.py | 4_NoScaling.py | Bonus.py

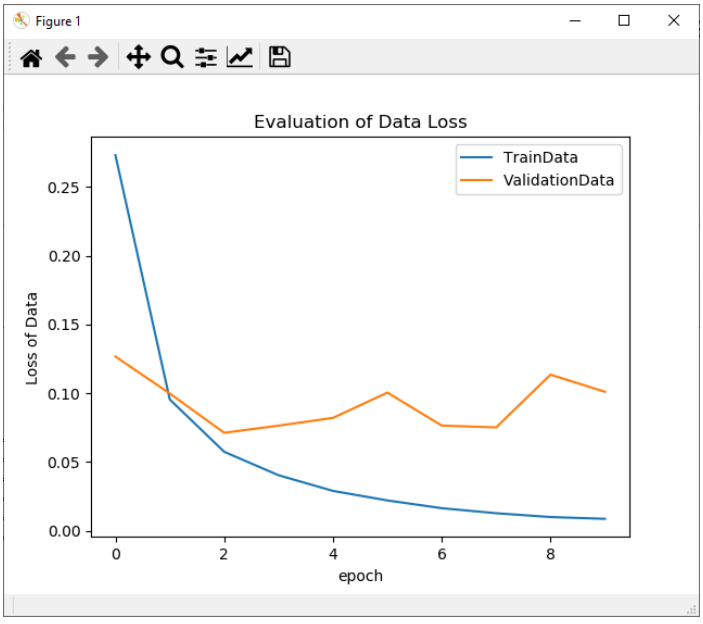
```python
# Importing libraries
from keras import Sequential
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense
from keras.utils import to_categorical

# Loading input data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Display the second image in the training data
plt.imshow(train_images[1, :, :], cmap='gray')
plt.title('Ground Truth : {}'.format(train_labels[1]))
plt.show()

# Process the data
# 1. Convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')

# Scale data
train_data /=255.0
test_data /=255.0

# Change the labels frominteger to one-hot encoding
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

```



```
# Creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1, validation_data=(
    test_data, test_labels_one_hot))
[test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))

# Listing all the components of data present in history
print('The data components present in history are', history.history.keys())

# Graphical evaluation of accuracy associated with training and validation data
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Evaluation of Data Accuracy')
plt.xlabel('epoch')
plt.ylabel('Accuracy of Data')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()

# Graphical evaluation of loss associated with training and validation data
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('Loss of Data')
plt.title('Evaluation of Data Loss')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()
```

```
 7456/10000 [=====================>........] - ETA: 0s
 7904/10000 [======================>.......] - ETA: 0s
 8352/10000 [=======================>.....] - ETA: 0s
 8832/10000 [========================>....] - ETA: 0s
 9280/10000 [=========================>...] - ETA: 0s
 9760/10000 [==========================>.] - ETA: 0s
10000/10000 [==============================] - 1s 113us/step
Evaluation result on Test Data : Loss = 0.10106841758542136, accuracy = 0.9789999723434448
The data components present in history are dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```



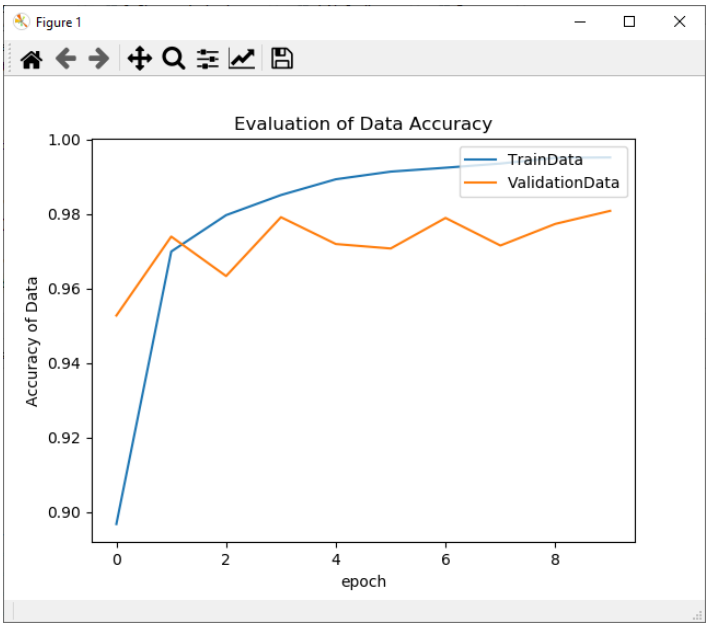


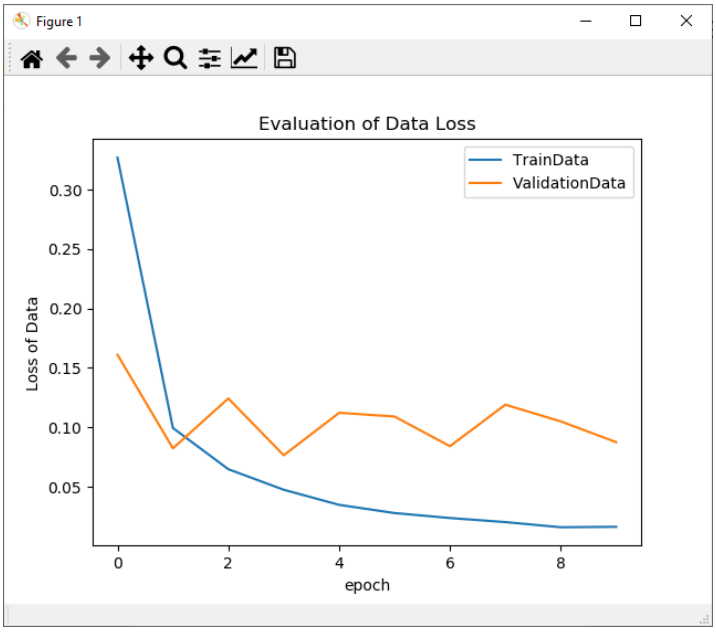
3. We had used 2 hidden layers and relu activation:
    a. Try to change the number of hidden layer and the activation to tanh or sigmoid and report what happens.

```
# Creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(512, activation='tanh'))
model.add(Dense(512, activation='sigmoid'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=2, verbose=1, validation_data=(test_data, test_labels_one_hot))
[test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))
```

```
 7712/10000 [======================>.......] - ETA: 0s
 8096/10000 [=======================>......] - ETA: 0s
 8448/10000 [========================>.....] - ETA: 0s
 8832/10000 [=========================>....] - ETA: 0s
 9216/10000 [==========================>...] - ETA: 0s
 9600/10000 [===========================>..] - ETA: 0s
 9984/10000 [============================>.] - ETA: 0s
10000/10000 [==============================] - 1s 138us/step
Evaluation result on Test Data : Loss = 0.08754560220184503, accuracy = 0.98089998960495
The data components present in history are dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

Accuracy Increased after adding more activation function.





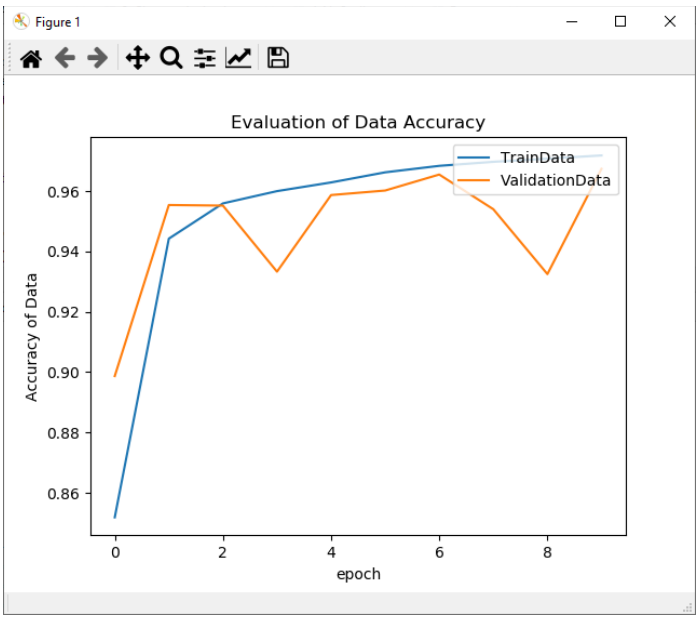
4. Run the same code without scaling the images, how the accuracy changes?

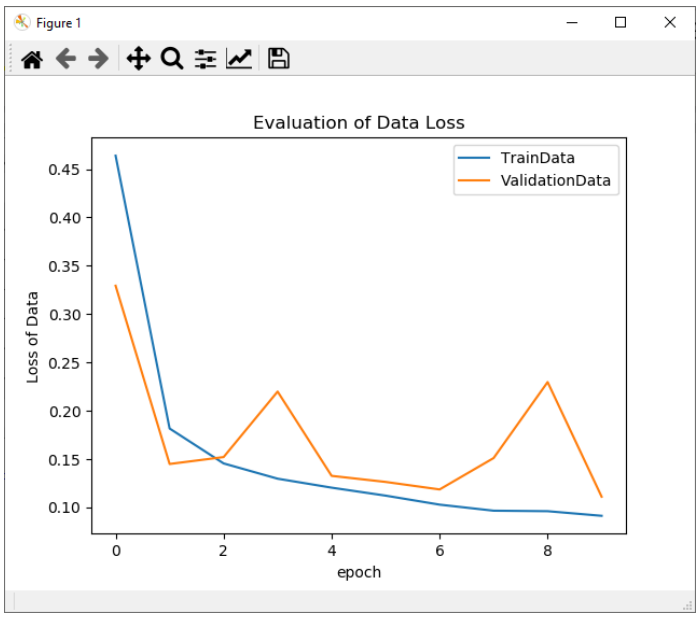
DLICP2 〉 4_NoScaling.py

1_History_Loass_Accuracy.py | 2_SingleImage_Loss_Accuracy.py | 3_Change_Activation.py | 4_NoScaling.py | Bonus.py

```
# Importing libraries
from keras import Sequential
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense
from keras.utils import to_categorical

# Loading input data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Display the second image in the training data
plt.imshow(train_images[1, :, :], cmap='gray')
plt.title('Ground Truth : {}'.format(train_labels[1]))
plt.show()

# Process the data
# 1. Convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')

# Change the labels frominteger to one-hot encoding
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(512, activation='tanh'))
model.add(Dense(512, activation='sigmoid'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1, validation_data=(
    test_data, test_labels_one_hot))
[test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))

# Listing all the components of data present in history
print('The data components present in history are', history.history.keys())
```

```
 8224/10000 [=======================>......] - ETA: 0s
 8608/10000 [========================>.....] - ETA: 0s
 8992/10000 [=========================>....] - ETA: 0s
 9376/10000 [==========================>...] - ETA: 0s
 9760/10000 [==========================>..] - ETA: 0s
10000/10000 [==============================] - 1s 146us/step
Evaluation result on Test Data : Loss = 0.11110130823198706, accuracy = 0.9674000144004822
The data components present in history are dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```





** Bonus point:

5. Convert the sequential model to API model.

```
# Creating network
a=Input(shape=(dimData,))
b= Dense(512, activation='relu')(a)
b= Dense(512, activation='relu')(b)
b= Dense(10, activation='softmax')(b)
model = Model(input=a,output=b)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=2, verbose=1,
                  validation_data=(test_data, test_labels_one_hot))
```

```
58880/60000 [============================>.] - ETA: 0s - loss: 0.0946 - accuracy: 0.9710
59392/60000 [============================>.] - ETA: 0s - loss: 0.0946 - accuracy: 0.9709
59904/60000 [============================>.] - ETA: 0s - loss: 0.0946 - accuracy: 0.9709
60000/60000 [==============================] - 8s 136us/step - loss: 0.0946 - accuracy: 0.9709 - val_loss: 0.0767 - val_accuracy: 0.9745
```
