

Introduction of Computer

Homework 5 Report

Name: 徐子程

Student ID: B06602037

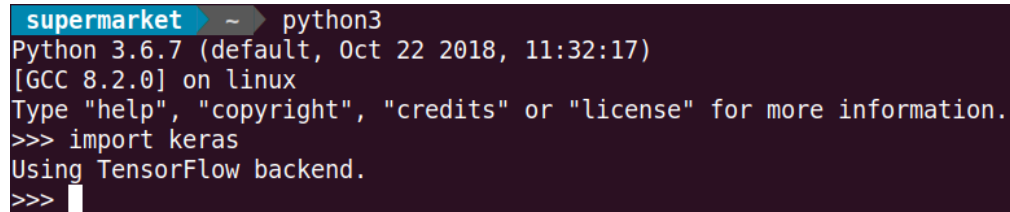
Your tensorflow version: 1.13.1 (With GPU support , GPU : NVIDIA GTX 1060 6G)

Keras version: 2.2.4

Environment (Windows 10 / MacOS / Linux): Ubuntu 18.04.2 LTS

- **Verify Installation (30%)**

To verify your tensorflow installation, open the python shell and import the package “keras”. There should be a message showing that keras is using the tensorflow backend. Please paste the screenshot here.



```
supermarket ~ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>>
```

- **Classification with Neural Network**

1. The structure of your best model. (10%)

3 flatten layers

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1024)	803840
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 512)	524800
dropout_3 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 10)	5130

2. Training accuracy. (5%)

Up to 0.94

```
Total params: 2,383,370
Trainable params: 2,383,370
Non-trainable params: 0

60000/60000 [=====] - 2s 34us/step
Train Acc: 0.94595
```

3. How do you determine the termination of training ? (10%)

We set the number of training epochs.

4. How do you determine which model structure is better ? (10%)

By comparing the validation data accuracy, if the accuracy is better than previous training epoch , we save the model into a file.

After all training epochs were done, we load the best model for testing data.

- **Bonus : Convolutional Neural Network**

Model structure:

4 convolution layers, 2 pooling layers, 2 flatten layers

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_1 (Batch Normalization)	(None, 28, 28, 32)	128
conv2d_2 (Conv2D)	(None, 28, 28, 64)	18496
batch_normalization_2 (Batch Normalization)	(None, 28, 28, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_3 (Conv2D)	(None, 14, 14, 64)	36928
batch_normalization_3 (Batch Normalization)	(None, 14, 14, 64)	256
conv2d_4 (Conv2D)	(None, 14, 14, 128)	73856
batch_normalization_4 (Batch Normalization)	(None, 14, 14, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 128)	0
dropout_1 (Dropout)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
dropout_3 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 10)	5130

Observations:

If dropout value is not big enough, overfitting will occur. The better value is around 50%.

Best validation accuracy: Up to 0.94

```
Epoch 26/30
54000/54000 [=====] - 13s 240us/step - loss: 0.1220 - acc: 0.9562 - val_loss: 0.1882 - val_acc: 0.9382

Epoch 00026: val_acc did not improve from 0.93933
Epoch 27/30
54000/54000 [=====] - 13s 241us/step - loss: 0.1173 - acc: 0.9581 - val_loss: 0.1973 - val_acc: 0.9292

Epoch 00027: val_acc did not improve from 0.93933
Epoch 28/30
54000/54000 [=====] - 13s 240us/step - loss: 0.1175 - acc: 0.9585 - val_loss: 0.1963 - val_acc: 0.9352

Epoch 00028: val_acc did not improve from 0.93933
Epoch 29/30
54000/54000 [=====] - 13s 240us/step - loss: 0.1109 - acc: 0.9608 - val_loss: 0.2283 - val_acc: 0.9283

Epoch 00029: val_acc did not improve from 0.93933
Epoch 30/30
54000/54000 [=====] - 13s 240us/step - loss: 0.1096 - acc: 0.9608 - val_loss: 0.1769 - val_acc: 0.9425

Epoch 00030: val_acc improved from 0.93933 to 0.94250, saving model to best.h5
```