

# 计算机科学与技术学院自然语言处理课程实验报告

实验题目：Homework3		学号：201600122057
日期：2019.3.29	班级：智能16	姓名：贝仕成
Email：337263318@qq.com		
实验目的：熟悉华为云的基本使用方法。		
<b>实验软件和硬件环境：</b> Python 3.5.6 Jupyter notebook 5.0.0  神舟战神 Z7M-KP7S1 NVIDIA GTX1050Ti  华为云		
<b>实验原理和方法：</b> 通过将 jupyter notebook 上运行成功的代码另存为 py 文件来建立训练作业，并输出结果以及一系列图。		
<b>实验步骤：（不要求罗列完整源代码）</b> 1. 准备数据 · 建立凭证 · 开启 OBS 服务 · 上传数据		



ModelArts

[ 新用户福利大放送 ] 价值约400元礼包 免费送。 [领取礼包](#)


最新动态



自动学习

暂未创建

[立即体验](#)



数据管理

mnist

2019/03/28 23:46:08



训练作业

train\_mnist

2019/03/29 00:29:12

样例库



找云宝

云宝是华为云吉祥物，本样例详细介绍如何使用ModelArts自动学习功能和放置算法快速生成物体检测...




花卉识别

本样例详细介绍如何使用花卉数据集对ModelArts预置算法ResNet\_v1\_50重训练，快速生成...



冰山识别

本样例详细介绍如何使用华为MoXing框架编写模型训练代码，快速生成图像分类模型，实现...



OBS Browser

bashine mnist

上传 新建文件夹 下载 移动 复制 粘贴 更多

输入对象名前缀搜索

桶名查询	名称	存储类别	恢复状态	大小	类型	修改时间	操作
bashine	train-1...	标准存储	--	58.601 KB	文件	2019/3/28 23:46:10 GMT+08:00	
	train-1...	标准存储	--	28.204 KB	文件	2019/3/28 23:46:10 GMT+08:00	
	t10k-1...	标准存储	--	7.476 MB	文件	2019/3/28 23:46:09 GMT+08:00	
	t10k-1...	标准存储	--	1.572 MB	文件	2019/3/28 23:46:09 GMT+08:00	
	t10k-1...	标准存储	--	9.773 KB	文件	2019/3/28 23:46:09 GMT+08:00	
	t10k-1...	标准存储	--	4.435 KB	文件	2019/3/28 23:46:09 GMT+08:00	
	train-1...	标准存储	--	44.860 MB	文件	2019/3/28 23:46:09 GMT+08:00	
	train-1...	标准存储	--	9.453 MB	文件	2019/3/28 23:46:09 GMT+08:00	

对象数量 41 存储用量 150.918 MB

数据上传成功

## 2. 调试代码

- 建立调试环境
- 修改调试代码
- 查看运行情况

jupyter Untitled (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code Convert to Python File

```
    '--log_dir',
    type=str,
    default=os.path.join(os.getenv('TEST_TMPDIR', '/tmp'),
                        'tensorflow/mnist/logs/mnist_with_summaries'),
    help='Summaries log directory')
parser.add_argument(
    '--data_url',
    type=str,
    default='s3://bashine/mnist',
    help='Directory for storing input data')
parser.add_argument(
    '--train_url',
    type=str,
    default='s3://bashine/log',
    help='summaries log directory')
FLAGS, unparsed = parser.parse_known_args()
FLAGS.log_dir = FLAGS.train_url
FLAGS.data_dir = FLAGS.data_url
tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)
```

Accuracy at step 650: 0.9605  
Accuracy at step 660: 0.9612  
Accuracy at step 670: 0.9625  
Accuracy at step 680: 0.9606  
Accuracy at step 690: 0.9631  
Adding run metadata for 699  
Accuracy at step 700: 0.9621  
Accuracy at step 710: 0.963  
Accuracy at step 720: 0.9631  
Accuracy at step 730: 0.9604  
Accuracy at step 740: 0.9629  
Accuracy at step 750: 0.9635  
Accuracy at step 760: 0.9659  
Accuracy at step 770: 0.9655  
Accuracy at step 780: 0.9668  
Accuracy at step 790: 0.9664  
Adding run metadata for 799  
Accuracy at step 800: 0.9659  
Accuracy at step 810: 0.9653

### 3. 训练模型

- 建立训练作业
- 建立 tensorboard

实验结果:

版本管理

溯源图

▼ 版本过滤

查看对比结果

2019/03/29 00:29:12

🟢

⏮ V0002 状态 运行成功 运行时间 00:01:11 ⏭

创建TensorBoard 创建模型 修改 更多操作

配置信息

日志 结果 资源占用情况

名称

train\_mnist | jobe7cd9a87

AI引擎

TensorFlow | TF-1.8.0-python3.6

状态

运行成功

代码目录

/bashine/codes/

运行版本

V0002

启动文件

/bashine/codes/train\_mnist.py

开始运行时间

2019/03/29 00:29:20

训练数据集

/bashine/mnist/

运行时间

00:01:11

主要运行参数

资源池

Computing GPU(P100) instance

训练输出位置

/bashine/log/

计算节点个数

1

描述

-

日志输出位置

-

NAS 地址

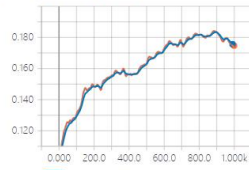
-

NAS 挂载路径

-

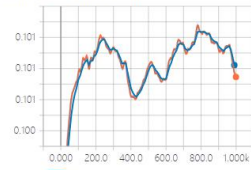
layer1

layer1/biases/summaries/max



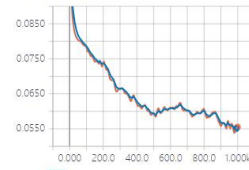
🔍 📊 📉

layer1/biases/summaries/mean



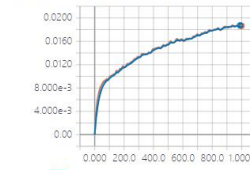
🔍 📊 📉

layer1/biases/summaries/min



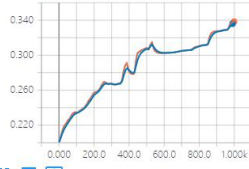
🔍 📊 📉

layer1/biases/summaries/stddev\_1



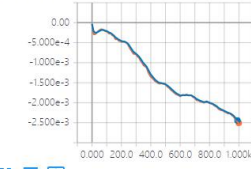
🔍 📊 📉

layer1/weights/summaries/max



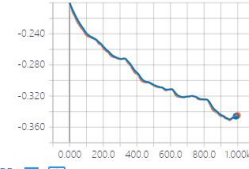
🔍 📊 📉

layer1/weights/summaries/mean



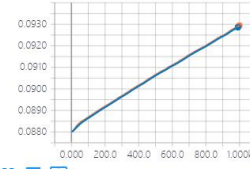
🔍 📊 📉

layer1/weights/summaries/min



🔍 📊 📉

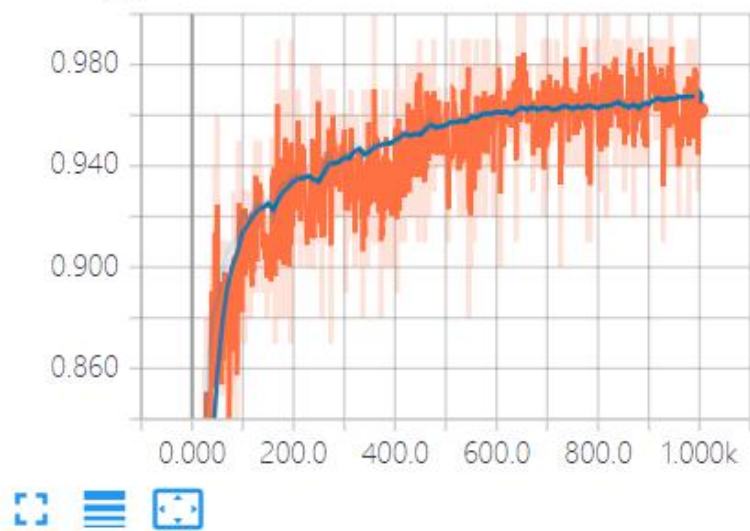
layer1/weights/summaries/stddev\_1



🔍 📊 📉

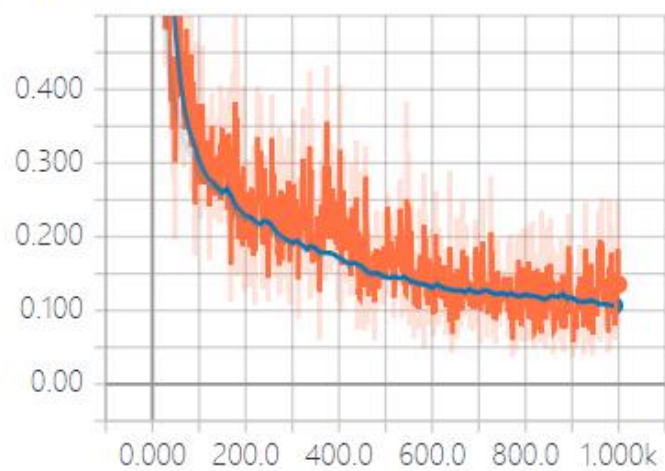
accuracy\_1

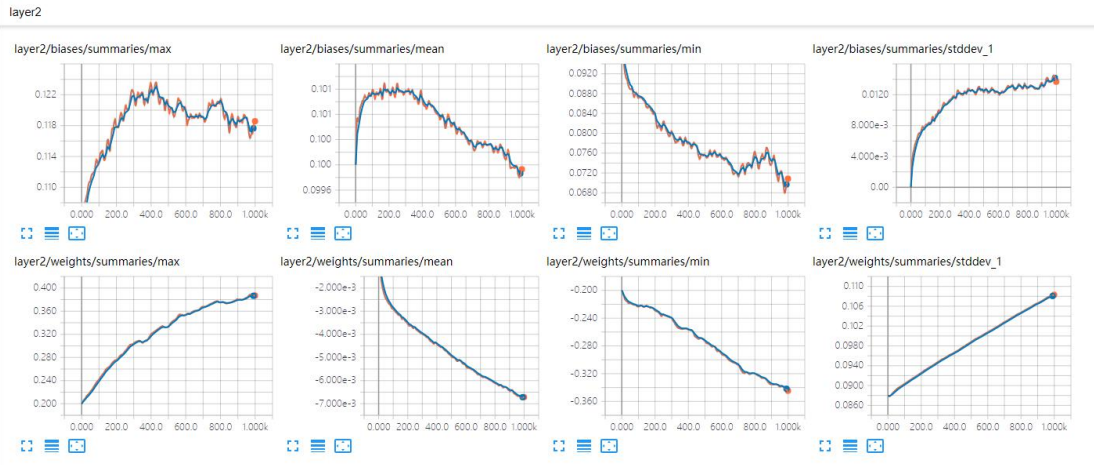
accuracy\_1



cross\_entropy\_1

cross\_entropy\_1





#### 4. 后续任务

尝试用 pytorch 框架在 ModelArts 上完成 MNIST

```
In [2]: from __future__ import print_function
import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
import os
import sys
import mxing as mx

_S3_SECRET_ACCESS_KEY=(os.environ.get('SECRET_ACCESS_KEY',None)
                        or os.environ.get('S3_SECRET_ACCESS_KEY',None)
                        or os.environ.get('AWS_SECRET_ACCESS_KEY',None))
_S3_ACCESS_KEY_ID=(os.environ.get('ACCESS_KEY_ID',None)
                   or os.environ.get('S3_ACCESS_KEY_ID',None)
                   or os.environ.get('AWS_ACCESS_KEY_ID',None))
mx.file.set_auth(ak=_S3_ACCESS_KEY_ID,sk=_S3_SECRET_ACCESS_KEY)
```

```
for epoch in range(1, args.epochs + 1):
    train(args, model, device, train_loader, optimizer, epoch)
    test(args, model, device, test_loader)

    if (args.save_model):
        torch.save(model.state_dict(), "mnist_cnn.pt")

if __name__ == '__main__':
    main()
```

```
Train Epoch: 1 [24320/60000 (41%)] Loss: 0.215614
Train Epoch: 1 [24960/60000 (42%)] Loss: 0.152429
Train Epoch: 1 [25600/60000 (43%)] Loss: 0.224683
Train Epoch: 1 [26240/60000 (44%)] Loss: 0.263283
Train Epoch: 1 [26880/60000 (45%)] Loss: 0.232621
Train Epoch: 1 [27520/60000 (46%)] Loss: 0.263381
Train Epoch: 1 [28160/60000 (47%)] Loss: 0.212269
Train Epoch: 1 [28800/60000 (48%)] Loss: 0.133421
Train Epoch: 1 [29440/60000 (49%)] Loss: 0.278359
Train Epoch: 1 [30080/60000 (50%)] Loss: 0.093708
Train Epoch: 1 [30720/60000 (51%)] Loss: 0.127433
Train Epoch: 1 [31360/60000 (52%)] Loss: 0.246622
Train Epoch: 1 [32000/60000 (53%)] Loss: 0.338711
Train Epoch: 1 [32640/60000 (54%)] Loss: 0.152402
Train Epoch: 1 [33280/60000 (55%)] Loss: 0.090370
Train Epoch: 1 [33920/60000 (57%)] Loss: 0.144396
Train Epoch: 1 [34560/60000 (58%)] Loss: 0.197434
```

结论分析与体会：

华为云是个功能强大的平台，把自己的工作安排上去可以很快地完成（当然要付出一点经济代价），或许是第一次接触所以不太熟练，觉得前期的部署挺费事的。（自己电脑条件够的话还是想在本地跑）。这次只体验了其中一部分功能，还有其他功能等待我们去发掘，或许日后它将成为我们研究深度学习不可或缺的工具。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1—3 道问答题：

1. 实验指导中并没有提到 `mnist_train.py` 是怎么来的，怎么办？

将实验指导中所说的代码在 jupyter notebook 中修改后另存为 py 文件，将这个 py 文件重命名为 `mnist_train.py`，就得到我们接下来训练作业所要要到的文件。