

# # Deep Learning Course Experiment Report

## Experiment 1

### 实验学生基本信息：

- 姓名：陈佳睿
- 学号：201700301042

### 实验内容：

- 熟悉华为云ModelArts
- 参考官网例子，使用TensorFlow实现手写数字识别
- 根据手写数字识别例子，完成实验报告

描述步骤(简略)

整理和总结使用平台的问题 (3个)

### 1. 熟悉华为云ModelArts

首先注册华为云的账号，完善了个人信息，进行了实名认证，确保了后续老师发放算力资源的顺利进行。

进入我的凭证，生成访问密钥（Access Key Id和Secret Access Key）用于后续的obs数据上传

我们将训练模型所需的数据需要通过obs工具进行上传，通过在华为云的obs管理界面创建桶来实现

| 工具                           | 说明   |
|------------------------------|--|
| <a href="#">OBS Browser+</a> | OBS Browser+是一款运行在Windows系统上的对象存储服务管理工具，OBS Browser+的图形化界面可以非常方便地让用户在本地对OBS进行管理。   |
| <a href="#">OBS Browser</a>  | OBS Browser提供类似于Windows资源管理器的功能，以及桶和对象的基本属性管理功能，可以非常方便地让用户在本地进行对象存储管理，例如：浏览文件、上传下载文件、配置生命周期规则等。                              |
| <a href="#">obsutil</a>      | obsutil是一款用于访问管理OBS的命令行工具，可以对OBS进行常用的配置管理操作，如创建桶、上传文件/文件夹、下载文件/文件夹、删除文件/文件夹等。对于熟悉命令行程序的用户，obsutil是执行批量处理、自动化任务的好选择。          |
| <a href="#">obsftp</a>       | obsftp工具利用pyftpdlib库的FTP server能力和对象存储云端存储能力，提供出具有FTP接入的云上存储使用能力。在企业实际业务中，无需单独搭建FTP服务器和存储池，实现业务和运维的轻量化，极大降低了原有的FTP访问方式的技术成本。 |
| <a href="#">obsfs</a>        | obsfs是OBS提供的一款基于FUSE的文件系统工具，主要用于将OBS桶挂载至Linux系统，让用户能够在本地像操作文件系统一样直接使用OBS海量的存储空间。   |

华为对于obs桶数据上传的工具一共提供了5种选择

但是由于我使用的是macOS系统，所以直接被剥夺了使用OBS Browser+和OBS Browser的机会，于是我依次尝试了obsftp和obsutil，最终通过obsutil实现了成功上传，这个没有用户界面，纯粹是使用命令来实现上传的工具其实对于用户并不是很友好，但是通过阅读开发和使用文档，还是成功了。

---

尝试使用ModelArts在公共OBS桶中提供的花卉的示例数据集，进行模型训练和环境熟悉  
对于花蕊类别的图像判别任务，我们使用华为云已经预置的算法“ResNet\_v1\_50”来进行



ModelArts

## 训练作业

帮助 | 常见问题

训练作业 预置算法 作业参数管理 可视化作业

|                        | 名称                     | 用途        | 引擎类型                         | 精度                      | P4推理速度 (ms) | 大小...  | 操作                   |
|------------------------|------------------------|-----------|------------------------------|-------------------------|-------------|--------|----------------------|
| 数据管理 <span>Beta</span> | yolo_v3                | 检测物体类别和位置 | MXNet, MXNet-1.2.1-pyth...   | 81.7%(mAP)              | 40          | 235.31 | <a href="#">创建训练</a> |
| 开发环境                   | retinanet_resnet_v1_50 | 检测物体类别和位置 | TensorFlow, TF-1.8.0-pyth... | 83.15%(mAP)             | 112         | 255.15 | <a href="#">创建训练</a> |
| 训练管理                   | mobilenet_v1           | 图像分类      | TensorFlow, TF-1.8.0-pyth... | 70.90%(top1), 89.90...  | 36          | 100.93 | <a href="#">创建训练</a> |
| · 训练作业                 | inception_v3           | 图像分类      | TensorFlow, TF-1.8.0-pyth... | 78.00%(top1), 93.90...  | 58          | 103.78 | <a href="#">创建训练</a> |
| · 自动化搜索作业              | hard_example_mining    | 图像分类      | TensorFlow, TF-1.8.0-pyth... | -                       | -           | 4.77   | <a href="#">创建训练</a> |
| 模型管理                   | feature_cluster        | 图嵌入       | TensorFlow, TF-1.8.0-pyth... | -                       | -           | 200.84 | <a href="#">创建训练</a> |
| 部署上线                   | darknet_53             | 图像分类      | MXNet, MXNet-1.2.1-pyth...   | 78.56%(top1), 94.43...  | 17          | 158.89 | <a href="#">创建训练</a> |
| AI市场 <span>Beta</span> | SegNet_VGG_BN_16       | 图像语义分割    | MXNet, MXNet-1.2.1-pyth...   | 89%(pixel acc)          | 27          | 112.41 | <a href="#">创建训练</a> |
| 订阅列表                   | ResNet_v2_50           | 图像分类      | MXNet, MXNet-1.2.1-pyth...   | 75.55%(top1), 92.6%(... | 15          | 97.76  | <a href="#">创建训练</a> |
| 专属资源池                  | ResNet_v1_50           | 图像分类      | TensorFlow, TF-1.8.0-pyth... | 74.2%(top1), 91.7%(...  | 44          | 200.84 | <a href="#">创建训练</a> |
| 全局配置                   |                        |           |                              |                         |             |        |                      |

10 总条数: 12 < 1 2 >

根据实验指导书的指引，配置了训练数据集、生成模型和结果、训练日志的路径

## 确认创建训练的作业

创建训练作业 [返回作业列表](#)

1 服务选型 2 规格确认 3 完成

### 规格确认

| 产品名称          | 产品规格  | 计费模式 | 价格         |
|---------------|---|------|------------|
| trainjob-c29f | 名称: trainjob-c29f<br>描述: -<br>训练数据集: /exp-000/dataset-flowers/Flowers-Data-Set/<br>算法: ResNet_v1_50<br>运行参数: split_spec=train:0.8,eval:0.2<br>num_gpus=1<br>batch_size=32<br>eval_batch_size=32<br>learning_rate_strategy=0.002<br>evaluate_every_n_epochs=1<br>save_interval_secs=2000000<br>max_epochs=100<br>log_every_n_steps=10<br>save_summaries_steps=5<br>训练输出位置: /exp-000/model-test/<br>日志输出位置: /exp-000/train-log/<br>规格: CPU: 8 核 64GiB GPU: 1 * nvidia-v100 32GiB<br>计算节点个数: 1 | 按需计费 | ¥ 28.00/小时 |

配置费用 **¥28.00/小时** [?](#)

作业创建后，将立即启动，运行过程中按需计费。

[上一步](#) [提交](#)

## 接下来创建可视化作业

训练作业 预置算法 作业参数管理 可视化作业

创建

全部状态

请输入名称查询



名称

状态

运行时长...

日志路径

创建时间

描述

操作

tensor-80d9

30dff60-4b2a-40ad-a...

运行中 (59分钟后停...

00:00:17

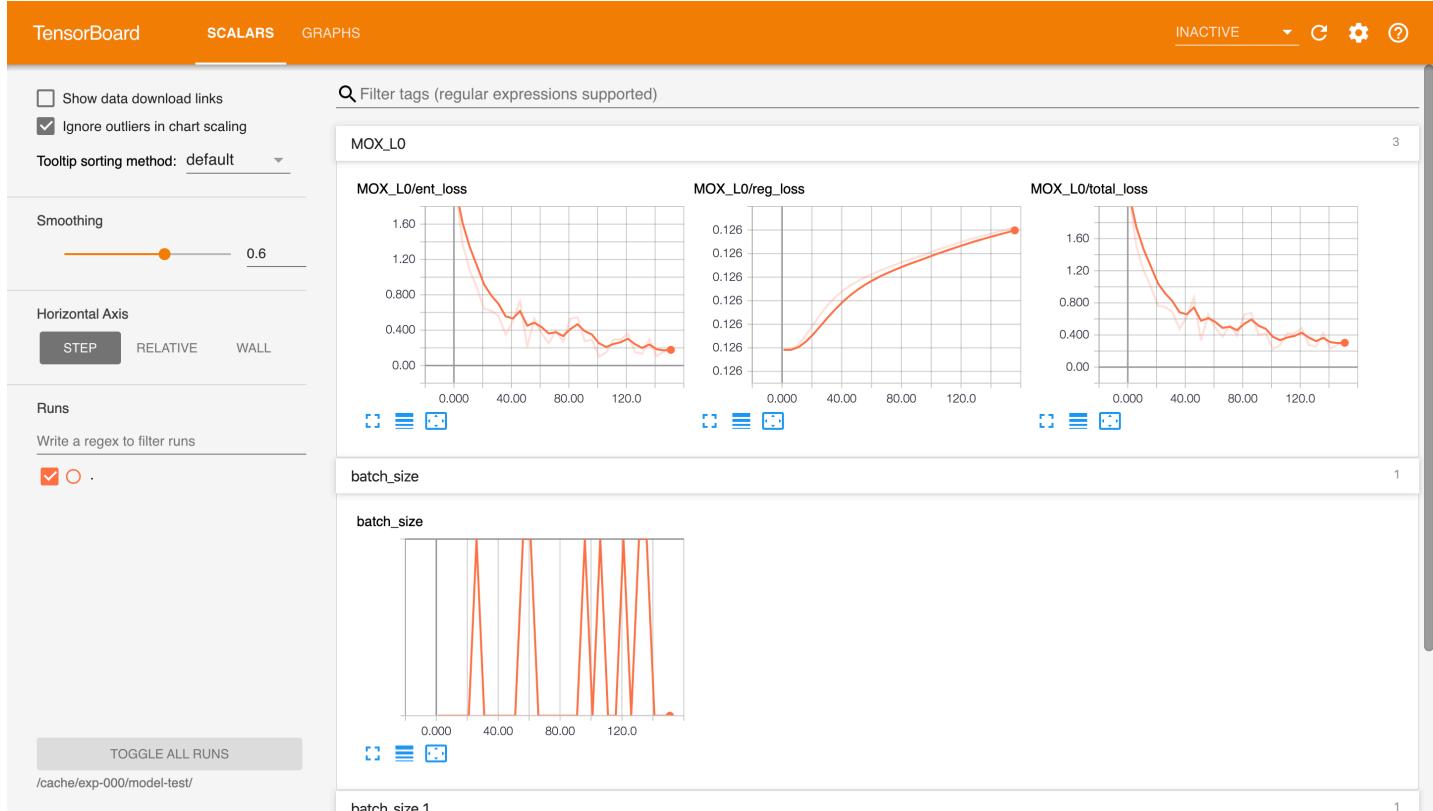
/exp-000/model-test/

2020/02/25 18:21:59 GMT+08:00



停止

删除



这里算是初步体验了使用华为云来进行模型训练的大体过程，但是自己没有进行具体的编码，只是看到了模型确实使用python2.x版本的支持的tensorflow跑起来了。

## 导入模型

模型

帮助

常见问题

导入

全部类型

请输入名称查询



模型名称

最新版本

部署类型

版本数量

创建时间

描述

操作

model-7c8f

0.0.1

在线服务/批量服务/边缘服务

1

2020/02/27 19:38:27...

创建新版本 删除

溯源图

请输入版本查询



版本

状态

部署类型

模型大小

模型来源

创建时间

描述

操作

0.0.1

导入中

在线服务/批量服务/边缘...

0 KB

-

2020/02/27 19:38:27...



部署 ▾ 发布 ▾ 更多 ▾



# 部署上线

部署

2 规格确认 3 完成

详情

| 产品类型   | 产品规格                  |                 | 计费模式 | 数量 | 价格        |
|--------|-----------------------|-----------------|------|----|-----------|
| 在线服务   | 名称                    | service-913b    | 按需计费 | 1  | ¥ 0.80/小时 |
|        | 参数配置                  | 模型名称 model-7c8f |      |    |           |
|        | 模型版本 0.0.1            |                 |      |    |           |
|        | 分流 (%) 100 %          |                 |      |    |           |
|        | 计算节点规格 CPU: 2 核 8 GiB |                 |      |    |           |
|        | 计算节点个数 1              |                 |      |    |           |
| 环境变量 - |                       |                 |      |    |           |

配置费用 **¥0.80**/小时

上一步 提交

## 部署为在线服务

在线服务 > service-913b

修改 停止 删除 C

|            |                |      |                                      |
|------------|----------------|------|--------------------------------------|
| 名称         | service-913b   | 服务ID | 7394322a-7622-4685-b9b7-b753bf721aa9 |
| 状态         | 运行中 (54 分钟后停止) | 来源   | 我的部署                                 |
| 调用失败次数/总次数 | 0 / 0          | 网络配置 | 未设置                                  |
| 描述         | -              | 数据采集 | 未开启                                  |

调用指南 预测 配置更新记录 数据采集 监控信息 事件 日志 共享 源码

API接口地址 <https://4c7298efd5d44a09b1affdafcf7> 说明: 支持本租户AK/SK以及token认证方式 [接口调用指南](#)

模型: model-7c8f 0.0.1

参数配置

POST /

输入参数

| 名称     | 类型   |
|--------|------|
| images | file |

输出参数

预测成功

请求路径: /

选择预测图片文件

上传

重新预测

案例反馈

预测图片预览



预测结果显示

预测成功

```

1 {
2   "predicted_label": "sunflowers",
3   "scores": [
4     [
5       "sunflowers",
6       "1.000"
7     ],
8     [
9       "tulips",
10      "0.000"
11    ],
12    [
13      "roses",
14      "0.000"
15    ],
16    [
17      "dandelion",
18      "0.000"
19    ],
20

```

## 2. 参考官网例子，使用TensorFlow实现手写数字识别

**体会使用Notebook构建模型**

首先要做好准备工作

- 将 `mnist_example.ipynb` 下载下来并传到自己创建的桶的 `mnist-MoXing-code` 文件夹下
- 自己创建一张尺寸为“28px\*28px”黑底白字的手写数字图片，我使用的是ipad上的绘



图软件procreate绘制的。如图：

- 将 `Mnist-Data-Set` 上传到 `dataset-mnist` 的obs路径下

\* 计费模式  按需计费

\* 名称

描述   
0/512

自动停止

! 开启该选项后, 该Notebook实例将在运行时长超出您所选择的时长后, 自动停止。

自动停止时间  1小时后  2小时后  4小时后  6小时后  自定义

\* 工作环境  Python3  Python2

\* 资源池  公共资源池  专属资源池

\* 类型  CPU  GPU

\* 规格  GPU: 1\*v100 NV32 CPU: 8 核 64GiB  GPU: 1\*p100 CPU: 8 核 64GiB

适合场景: GPU计算型, 适合标准深度学习在GPU上的代码运行与调测

配置费用 **¥7.60**/小时 ②

下一步

## 这里我们创建训练用的notebook

Notebook > notebook-785e

! 温馨提示: 该Notebook设置了自动停止时间, 即将在 52 分钟 后自动停止。点击 [这里](#) 修改

Jupyter

Files  Running [ModelArts Examples](#)

Select items to perform actions on them. Sync OBS contents to Notebook? [Click here](#)

|  | Name | Last Modified         | File size |
|--|------|-----------------------|-----------|
| <input type="checkbox"/> 0                   | /    |                       |           |
| <input type="checkbox"/> mnist_example.ipynb |      | Running 5 minutes ago |           |

10 Total Records: 1 < 1 >

点开我们上传的用于训练手写数字训练集的.ipynb文件  
 运行加载和训练数据集的代码  
 得到如下提示:

```
INFO:tensorflow:Saving checkpoints for 1000 into ./cache/log/model.ckpt.
INFO:tensorflow:Ignoring --checkpoint_path because a checkpoint already exists in ./cache/log/
INFO:tensorflow>No assets to save.
INFO:tensorflow>No assets to write.
INFO:tensorflow:Restoring parameters from ./cache/log/model.ckpt-1000
INFO:tensorflow:SavedModel written to: b'./cache/log/model/saved_model.pb'
```

后面我用预测的代码对于我上传的数字图像进行识别  
 得到结果:

```
INFO:tensorflow:Graph was finalized.  
INFO:tensorflow:Restoring parameters from ./cache/log/model.ckpt-1000  
INFO:tensorflow:Running local_init_op.  
INFO:tensorflow:Done running local_init_op.  
INFO:tensorflow:      [1 examples]
```

```
The result: [1]
```

```
An exception has occurred, use %tb to see the full traceback.
```

但是这与我写的数字7不一致，按道理讲，对于mnist数据集进行训练之后，应该不论是在自己的训练集，还是外来图像数据精度已经挺高了，结果还是错误的。思考了一下，感觉可能是我写的数字不太清楚，当然也有可能是这里搭建的模型对于外部数据的精度不高。

### 整理和总结使用平台的问题 (3个)

1. 首先，从一个macbook使用者的角度出发，上传文件的工具上手的学习曲线比较高，没有两个小时去琢磨和配置这个obsutil工具是不可能掌握方法的，这无疑是刚开始使用这个平台的一个坎。  
至于有多难用，我为此写了一篇博客放在简书上，个人感觉可以帮刚上手的macOS用户加快至少20分钟的速度。
2. 不管是创建notebook还是创建已有模型进行训练，平台的响应速度都很慢，基本是我出去倒了一杯热水回来他还在加载。这就很拖实验的时间，如果本来快速的平台响应可以做到2小时完成实验的话，那在华为云上最终拖的时间就可能达到双倍的时间。
3. 总的系统给人很零散的感觉，特别是obs的系统和ModelArts的界面不在一起，而且没有特别方便的链接，在自己的浏览器中总是要点开好多的标签栏才能开始实验，给人感觉并不优雅。