

计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目： 熟悉华为云 ModelArts		学号： 202000130047																																																		
日期： 2022-11-5	班级： 智能 20	姓名： 夏再禹																																																		
Email： 842649082@qq.com																																																				
<p>实验目的：</p> <p>熟悉华为云 ModelArts</p> <p>参考例子，练习使用自动学习训练模型</p> <p>参考例子，练习使用 notebook 训练模型</p> <p>阅读 TensorFlow 实现手写数字识别代码</p>																																																				
<p>实验软件和硬件环境：</p> <p>软件： jupyter notebook</p> <p>硬件： cpu: Intel i5</p>																																																				
<p>实验原理和方法：</p> <p>ModelArts 是面向 AI 开发者的一站式开发平台,提供海量数据预处理及半自动化标注、大规模分布式训练、自动化模型生成及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期 AI 工作流。</p> <p>ModelArts 使用对象存储服务（Object Storage Service，简称 OBS）进行数据存储以及模型的备份和快照，实现安全、高可靠和低成本的数据需求。因此，在使用 ModelArts 之前需要创建一个 OBS 桶，然后在 OBS 桶中创建文件夹用于存放数据。</p>																																																				
<p>实验步骤：（不要求罗列完整源代码）</p> <p>一、使用自动学习训练模型</p> <p>1. 创建 obs 桶，并创建好文件夹</p> <table border="1"><thead><tr><th><input type="checkbox"/></th><th>名称</th><th>存储类别</th><th>大小</th><th>加密状态</th><th>恢复状态</th><th>最后修改时间</th><th>操作</th></tr></thead><tbody><tr><td><input type="checkbox"/></td><td>dataset-yun...</td><td>--</td><td>--</td><td>--</td><td>--</td><td>--</td><td>分享 复制路径 更多 ▾</td></tr><tr><td><input type="checkbox"/></td><td>model-test</td><td>--</td><td>--</td><td>--</td><td>--</td><td>--</td><td>分享 复制路径 更多 ▾</td></tr><tr><td><input type="checkbox"/></td><td>train-log</td><td>--</td><td>--</td><td>--</td><td>--</td><td>--</td><td>分享 复制路径 更多 ▾</td></tr></tbody></table> <p>2. 利用 OBS Browser+ 将“Yunbao-Data-Custom”文件夹下的所有文件上传至“yourname2022/dataset-yunbao” OBS 路径下</p> <div><div>桶列表 / maiqiobs / dataset-yunbao / Yunbao-Data-Custom</div><div>华北-北京四 桶内对象总数: 100 存储总用量: 32.08 MB</div><div><div>上传 新建文件夹 下载 复制 更多 粘贴(1)</div><div>输入对象名前缀搜索</div></div><table border="1"><thead><tr><th><input type="checkbox"/></th><th>对象名称</th><th>存储类别</th><th>大小</th><th>最后修改时间</th><th>操作</th></tr></thead><tbody><tr><td><input type="checkbox"/></td><td>eval</td><td>--</td><td>--</td><td>--</td><td>↓ 🔍 ...</td></tr><tr><td><input type="checkbox"/></td><td>train</td><td>--</td><td>--</td><td>--</td><td>↓ 🔍 ...</td></tr></tbody></table></div> <p>3. 创建训练项目。</p>			<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时间	操作	<input type="checkbox"/>	dataset-yun...	--	--	--	--	--	分享 复制路径 更多 ▾	<input type="checkbox"/>	model-test	--	--	--	--	--	分享 复制路径 更多 ▾	<input type="checkbox"/>	train-log	--	--	--	--	--	分享 复制路径 更多 ▾	<input type="checkbox"/>	对象名称	存储类别	大小	最后修改时间	操作	<input type="checkbox"/>	eval	--	--	--	↓ 🔍 ...	<input type="checkbox"/>	train	--	--	--	↓ 🔍 ...
<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时间	操作																																													
<input type="checkbox"/>	dataset-yun...	--	--	--	--	--	分享 复制路径 更多 ▾																																													
<input type="checkbox"/>	model-test	--	--	--	--	--	分享 复制路径 更多 ▾																																													
<input type="checkbox"/>	train-log	--	--	--	--	--	分享 复制路径 更多 ▾																																													
<input type="checkbox"/>	对象名称	存储类别	大小	最后修改时间	操作																																															
<input type="checkbox"/>	eval	--	--	--	↓ 🔍 ...																																															
<input type="checkbox"/>	train	--	--	--	↓ 🔍 ...																																															

选择“新建数据集”填写“数据集名称”，设置“数据集输入位置”为 OBS 桶中”dataset/yunbao/train”路径、“数据集输出位置”设置为 OBS 桶中”model-test”路径。

4. 训练模型

训练设置

数据集版本名称

V001

训练验证比例 ?

训练集比例: 0.8 ?

验证集比例: 0.2

增量训练版本 ?

不选择版本

最大训练时长 (分钟)

6

训练偏好 ?

balance

计算规格

自动学习免费规格 (GPU)

1、自动学习训练免费规格用于使用体验，训练作业会在1小时（不包括模型发布时间）后自动停止，请勿在训练设置中使用超过1小时的最大训练时长。

2、自动学习训练免费规格资源不包含对象存储服务（OBS）存储资源费用，对象存储服务（OBS）计费标准详见如下链接：[对象存储服务（OBS）计费详情](#)。

☐ 我已阅读并同意以上内容

配置费用

¥0.00/小时 ?

下一步

5. 训练结束后，查看训练结果

版本管理

✓

已完成

V001 (26677a49-1145-4b...

2022/11/05 21:23:44 GMT+...

准确率: 89%

部署

删除

6. 单击“部署”，将模型部署上线。

<input type="checkbox"/>	名称/ID 下拉	状态	调用失败次数...	创建时间 下拉	更新时间 下拉	描述	操作
<input type="checkbox"/>	exeML-2022_ExeML_16... dd23c09c-c591-44ca-86bc-ece...	<div>● 运行中 (41 分钟 后停止)</div>	0/0	2022/11/05 21:41:41 GM...	2022/11/05 21:49:44 GM...	Created by...	修改 预测 启动 更多 下拉

7. 进行测试

预测图片预览



预测结果显示

预测成功

```
1 [{
2   "detection_classes": [
3     "yunbao",
4     "yunbao",
5     "yunbao"
6   ],
7   "detection_boxes": [
8     [
9       "29.197529",
10      "195.50885",
11      "405.43854",
12      "466.69513"
13     ],
14     [
15       "44.67538",
16       "541.0391",
17       "442.1521",
18       "803.1568"
19     ],
20     [
21       "458.07397",
22       "428.11813",
23       "658.05774",
```

8. 删除自动学习与 obs 桶

二、使用 notebook 训练模型

1. 创建 obs 桶，上传代码文件

桶列表 / notebook-maiqi / mnist-MoXing-code

华北-北京四 | 桶内对象总数: 4 | 存储总用量: 15.58 KB

上传 新建文件夹 下载 复制 更多

输入对象名前缀搜索

对象名称	存储类别	大小	最后修改时间	操作
mnist_example.ipynb	标准存储	15.58 KB	2022/11/05 21:28:51 GMT+...	下载 删除 ...

2. 上传数据集

桶列表 / notebook-maiqi / dataset-mnist

华北-北京四 | 桶内对象总数: 14 | 存储总用量: 63.48 MB

上传 新建文件夹 下载 复制 更多

输入对象名前缀搜索

对象名称	存储类别	大小	最后修改时间	操作
110k-labels-idx1-ubyte.gz	标准存储	4.43 KB	2022/11/05 21:38:28 GMT+...	下载 删除 ...
train-images-idx3-ubyte	标准存储	44.86 MB	2022/11/05 21:38:28 GMT+...	下载 删除 ...
train-labels-idx1-ubyte	标准存储	58.60 KB	2022/11/05 21:38:28 GMT+...	下载 删除 ...
train-images-idx3-ubyte.gz	标准存储	9.45 MB	2022/11/05 21:38:28 GMT+...	下载 删除 ...
train-labels-idx1-ubyte.gz	标准存储	28.20 KB	2022/11/05 21:38:28 GMT+...	下载 删除 ...
110k-images-idx3-ubyte.gz	标准存储	1.57 MB	2022/11/05 21:38:29 GMT+...	下载 删除 ...
110k-images-idx3-ubyte	标准存储	7.47 MB	2022/11/05 21:38:29 GMT+...	下载 删除 ...
110k-labels-idx1-ubyte	标准存储	9.77 KB	2022/11/05 21:38:33 GMT+...	下载 删除 ...

3. 创建 Notebook 开发环境

创建 Notebook

[< 返回 Notebook 列表](#)

① 服务选型 — ② 规格确认 — ③ 完成

◎ 评价 | 📖 使用指南

产品名称	产品规格	计费模式	价格
notebook-4990	自动停止时间	1小时	
	描述	notebook	
	工作环境	Multi-Engine 1.0 (Python3, Recommended)	
	资源池	公共资源池	
	类型	CPU	
	规格	CPU: 2 核 8GB	
	存储配置	OBS (/notebook-maiqi/mnist-MoXing-code/)	
		按需计费	Notebook: ¥0.80/小时

4. 进入 jupyter 页面，打开代码文件

将“data_url”修改为步骤 1：准备数据中数据集所在 OBS 路径

```
In [ ]: ##### your coding place: begin #####
# 此处必须修改为用户数据桶位置

#数据在OBS的存储位置。
# eg. s3:// : 统一路径输入
#      /uBucket : 桶名, 用户的私有桶的名称 eg. bucket
#      /notebook/data/: 文件路径

data_url = 's3://notebook-maiqi/dataset-mnist/'

##### your coding place: end #####
```

5. 运行代码

可看到训练过程为：

```
INFO:tensorflow:step: 900(global step: 900) sample/sec: 22112.526 loss: 0.634 accuracy: 0.880
INFO:tensorflow:step: 910(global step: 910) sample/sec: 54457.336 loss: 0.580 accuracy: 0.860
INFO:tensorflow:step: 920(global step: 920) sample/sec: 47233.153 loss: 0.505 accuracy: 0.920
INFO:tensorflow:step: 930(global step: 930) sample/sec: 55819.856 loss: 0.631 accuracy: 0.840
INFO:tensorflow:step: 940(global step: 940) sample/sec: 60038.706 loss: 0.638 accuracy: 0.840
INFO:tensorflow:step: 950(global step: 950) sample/sec: 58367.715 loss: 0.516 accuracy: 0.880
INFO:tensorflow:step: 960(global step: 960) sample/sec: 28696.661 loss: 0.484 accuracy: 0.940
INFO:tensorflow:step: 970(global step: 970) sample/sec: 52958.384 loss: 0.734 accuracy: 0.860
INFO:tensorflow:step: 980(global step: 980) sample/sec: 54106.089 loss: 0.732 accuracy: 0.840
INFO:tensorflow:step: 990(global step: 990) sample/sec: 58060.687 loss: 0.591 accuracy: 0.880
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'
```

6. 进行测试


测试给定的图片 ，预测结果：7

```
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: [7]

我手绘了两张图片（28*28，黑底白字）用于测试：

测试图片 ，预测结果：2

测试图片 ，预测结果：3

可见我绘制的这两张预测的都错了，但模型在训练集上的 accuracy 确实还不错，由此可见模型在训练集上过拟合严重。

三、阅读 TensorFlow 实现手写数字识别代码

我通过阅读代码，把代码中主要的流程或是特殊的处理总结如下：

1. 代码首先将数据拷贝到本地，这样能够加快训练，后续也经常进行这个操作。这个算是这里比较要特别注意的操作，因为在我们自己电脑上不需这样做。

```
mox.file.copy_parallel(data_url, local_url)
```

2. 通过函数 `tf.flags.DEFINE_string('train_url', None, 'Train Url')` 来定义日志以及生产模型的存储路径。（tensorflow 的使用技巧）

3. 定义模型：

- (1) 阅读模型的前向传播部分的代码（函数 `model_fn`）

可发现模型**仅由一层全连接层组成**，且没用用激活函数，也没用 softmax。

输入的一张 28*28 的图片，模型直接将其拉伸成长 784 的向量，然后通过一层全连接层，变成 10 维向量，代表对 10 个类别的预测得分，然后取其中值最大的类别作为预测出的类别。

- (2) 模型的损失函数使用 tensorflow 中的 `softmax_cross_entropy_with_logits`。

- (3) 阅读 `mox.run()` 函数的定义，可知模型采用 `sgd` 进行训练，学习率为 0.01。

4. 模型测试

这里又用模型的全连接层的参数 `W`、`b` 再次实现了模型的前向传播，和模型定义中类似。

从模型的简单结构上来看，就可以很好解释为什么测试时准确率极低，因为模型采用一层 784 维到 10 维的全连接层，参数有 $784 \times 10 + 10$ 个，参数足够多从而使模型可以在训练集上拟合到准确率较高，但模型的简单结构显然不足以应对手写数字识别的任务，因此进行测试时可发现效果很差，过拟合现象严重。

结论分析与体会：

通过本次实验，熟悉了华为云平台的使用。

对于数据存储以及模型的备份，需要创建一个 OBS 桶，可以把文件上传上去或下载下来。

本次实验首先创建了一个自动学习训练模型，我们传上数据集，通过华为云平台的 ModelArts 可直接进行标注、训练、部署。

然后使用了 notebook 训练了一个手写字识别模型，将数据集传上 OBS 桶，利用 jupyter 开发环境进行代码的编写与运行。

阅读了 TensorFlow 实现手写数字识别代码，理解了模型以及一些 tensorflow 的语法，通过模型的定义（一层全连接层）解释了模型在测试中的表现不好的原因。

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：
(整理和总结使用平台的三个问题)

1. 直接在网页上上传训练数据非常慢，且不能上传文件夹

解决：安装 OSB Browser+，用其进行上传。

2. 部署上线一直不成功，状态总是从准备中就到了停止。

解决：把部署时选择的“计算节点规格”选项由选的免费的改为付费的，就成功运行了。

3. 访问密钥无法再次下载

解决：访问密钥只能下载一次，必须保存好，否则若丢失就重新创建一个。