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：华为云 ModelArts 使用		学号：201900301174
日期：2021.9.20	班级：　智能 19	姓名：　韩旭
Email：hanx@mail.sdu.edu.cn		

实验目的：
1，熟悉华为云 ModelArts
2，使用预训练模型实现花卉识别
3，使用 tensorflow 实现手写数字识别

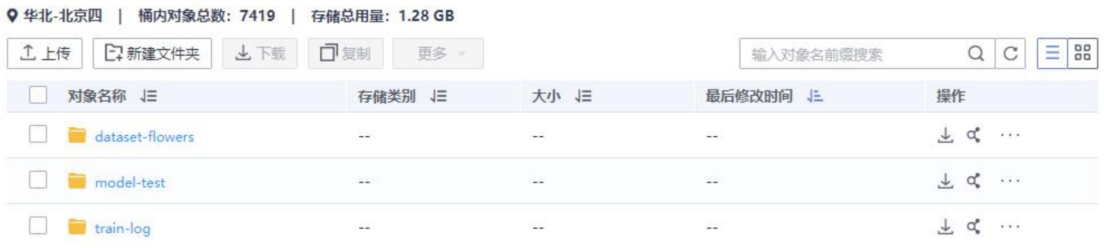
实验软件和硬件环境：
华为云 ModelArts

实验原理和方法：
1，使用预置模型 ResNet50 识别花卉种类，并部署模型
2，使用 tensorflow 在 ModelArts 的 Notebook 中训练并且测试模型，实现手写数字识别

实验步骤：（不要求罗列完整源代码）

一：使用预置模型 ResNet50 识别花卉种类

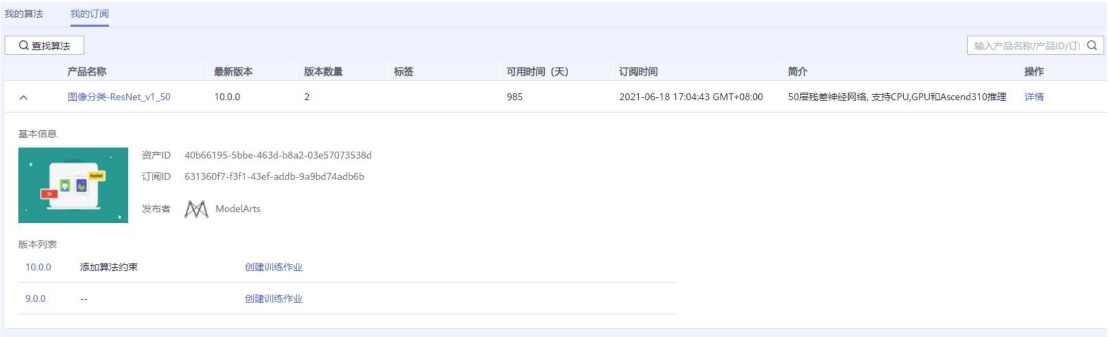
创建 obs 桶以及如下文件夹



华北-北京四 | 桶内对象总数：7419 | 存储总用量：1.28 GB

对象名称	存储类别	大小	最后修改时间	操作
dataset-flowers	--	--	--	
model-test	--	--	--	
train-log	--	--	--	

1，添加订阅算法 ResNet_v1_50，并且创建训练作业



	产品名称	最新版本	版本数量	标签	可用时间（天）	订阅时间	简介	操作
^	图像分类-ResNet_v1_50	10.0.0	2		985	2021-06-18 17:04:43 GMT+08:00	50层残差神经网络, 支持CPU,GPU和Ascend310推理	详情

基本信息

资产ID 40b66195-5bbe-463d-b8a2-03e57073538d

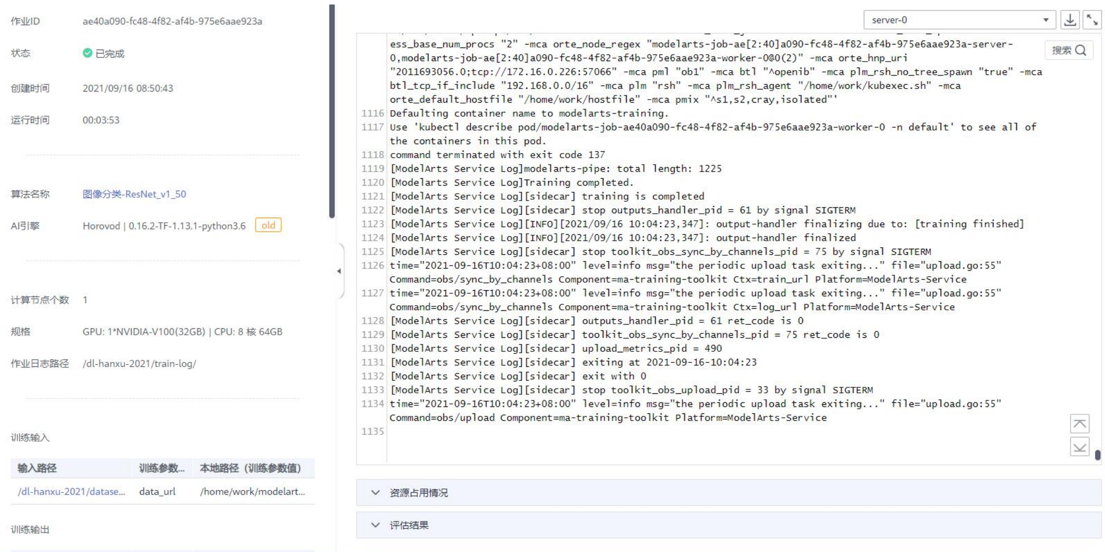
订阅ID 631360f7-f3f1-43ef-addb-9a9bd74adb6b

发布者 ModelArts

版本列表

10.0.0	添加算法约束	创建训练作业
9.0.0	--	创建训练作业

2，使用 ResNet_v1_50 模型训练过程如下



作业ID ae40a090-fc48-4f82-af4b-975e6aae923a

状态 已完成

创建时间 2021/09/16 08:50:43

运行时间 00:03:53

算法名称 图像分类-ResNet_v1_50

AI引擎 Horovod | 0.16.2-TF-1.13.1-python3.6 old

计算节点个数 1

规格 GPU: 1*NVIDIA-V100(32GB) | CPU: 8 核 64GB

作业日志路径 /dl-hanxu-2021/train-log/

训练输入

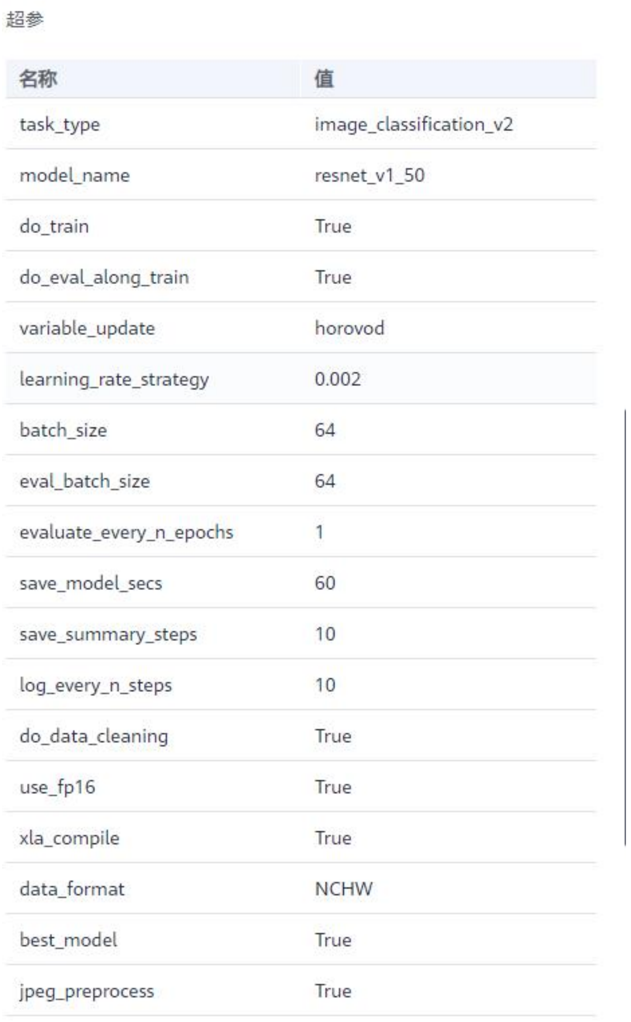
输入路径	训练参数...	本地路径（训练参数值）
/dl-hanxu-2021/datase...	data_url	/home/work/modelart...

训练输出

```
ess_base_num_procs "2" -mca orte_node_regex "modelarts-job-ae[2:40]a090-fc48-4f82-af4b-975e6aae923a-server-
0,modelarts-job-ae[2:40]a090-fc48-4f82-af4b-975e6aae923a-worker-0@0(2)" -mca orte_hnp_uri
"2011693056.0;tcp://172.16.0.226:57066" -mca pml "ob1" -mca btl "^openib" -mca plm_rsh_no_tree_spawn "true" -mca
btl_tcp_if_include "192.168.0.0/16" -mca plm "rsh" -mca plm_rsh_agent "/home/work/kubexec.sh" -mca
orte_default_hostfile "/home/work/hostfile" -mca pmix "^s1,s2,cray,isolated"'
Defaulting container name to modelarts-training.
Use 'kubectl describe pod/modelarts-job-ae40a090-fc48-4f82-af4b-975e6aae923a-worker-0 -n default' to see all of
the containers in this pod.
command terminated with exit code 137
[ModelArts Service Log]modelarts-pipe: total length: 1225
[Modelarts Service Log]Training completed.
[ModelArts Service Log][sidecar] training is completed
[ModelArts Service Log][sidecar] stop outputs_handler_pid = 61 by signal SIGTERM
[ModelArts Service Log][INFO][2021/09/16 10:04:23,347]: output-handler finalizing due to: [training finished]
[ModelArts Service Log][INFO][2021/09/16 10:04:23,347]: output-handler finalized
[ModelArts Service Log][sidecar] stop toolkit_obs_sync_by_channels_pid = 75 by signal SIGTERM
time="2021-09-16T10:04:23+08:00" level=info msg="the periodic upload task exiting..." file="upload.go:55"
Command=obs/sync_by_channels Component=ma-training-toolkit Ctx=train_url Platform=ModelArts-Service
time="2021-09-16T10:04:23+08:00" level=info msg="the periodic upload task exiting..." file="upload.go:55"
Command=obs/sync_by_channels Component=ma-training-toolkit Ctx=log_url Platform=ModelArts-Service
[ModelArts Service Log][sidecar] outputs_handler_pid = 61 ret_code is 0
[ModelArts Service Log][sidecar] toolkit_obs_sync_by_channels_pid = 75 ret_code is 0
[ModelArts Service Log][sidecar] upload_metrics_pid = 490
[ModelArts Service Log][sidecar] exiting at 2021-09-16-10:04:23
[ModelArts Service Log][sidecar] exit with 0
[ModelArts Service Log][sidecar] stop toolkit_obs_upload_pid = 33 by signal SIGTERM
time="2021-09-16T10:04:23+08:00" level=info msg="the periodic upload task exiting..." file="upload.go:55"
Command=obs/upload Component=ma-training-toolkit Platform=ModelArts-Service
```

可以看到模型训练时间为 3 分 53 秒。

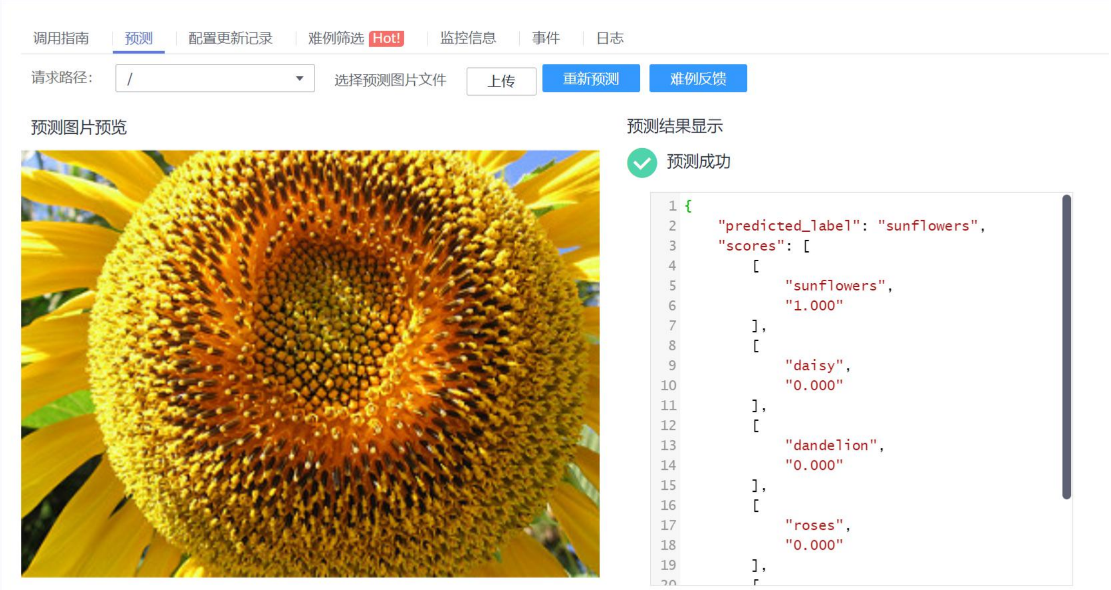
3，训练使用超参如下：



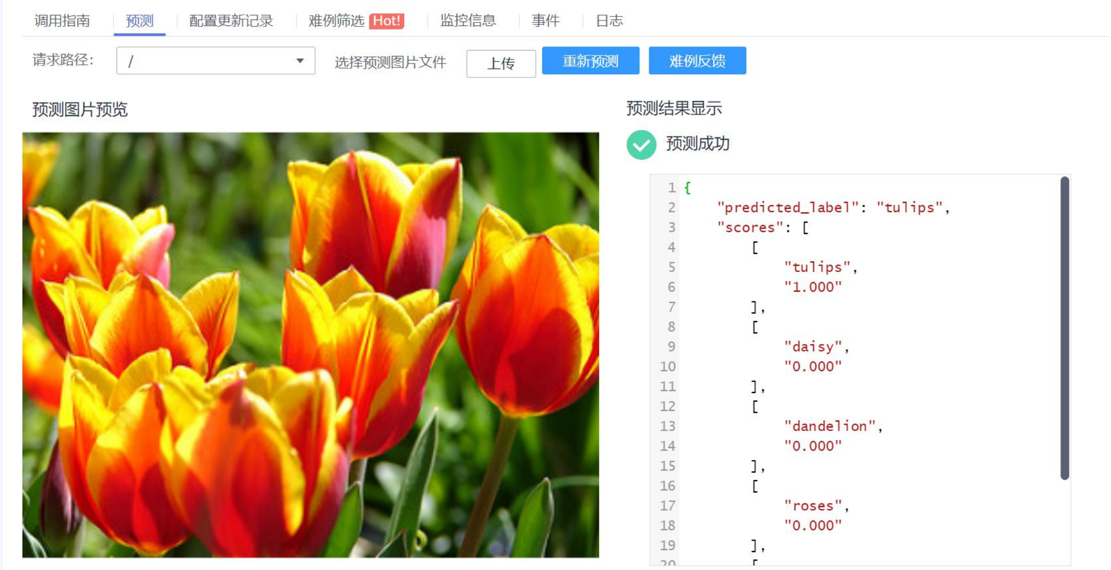
超参

名称	值
task_type	image_classification_v2
model_name	resnet_v1_50
do_train	True
do_eval_along_train	True
variable_update	horovod
learning_rate_strategy	0.002
batch_size	64
eval_batch_size	64
evaluate_every_n_epochs	1
save_model_secs	60
save_summary_steps	10
log_every_n_steps	10
do_data_cleaning	True
use_fp16	True
xla_compile	True
data_format	NCHW
best_model	True
jpeg_preprocess	True

4，训练结束后完成部署，预测结果如下



```
{
    "predicted_label": "sunflowers",
    "scores": [
        [
            "sunflowers",
            "1.000"
        ],
        [
            "daisy",
            "0.000"
        ],
        [
            "dandelion",
            "0.000"
        ],
        [
            "roses",
            "0.000"
        ],
        [
```



请求路径： /

选择预测图片文件

预测图片预览

预测结果显示

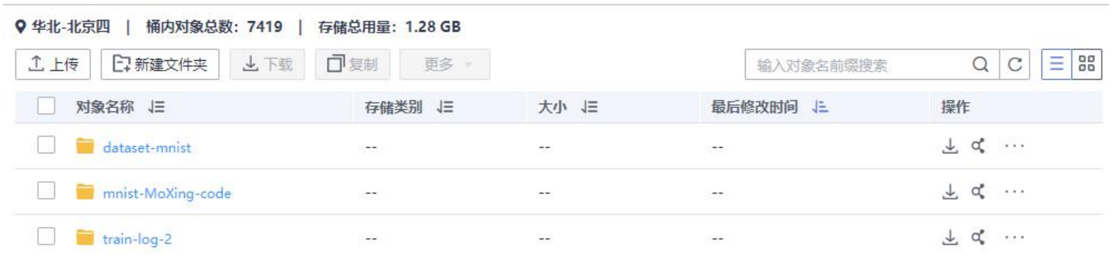
预测成功

```
{
    "predicted_label": "tulips",
    "scores": [
        [
            "tulips",
            "1.000"
        ],
        [
            "daisy",
            "0.000"
        ],
        [
            "dandelion",
            "0.000"
        ],
        [
            "roses",
            "0.000"
        ],
        [
```

可以看到模型预测成功。

二：使用 tensorflow 在 ModelArts 的 Notebook 中实现手写数字识别

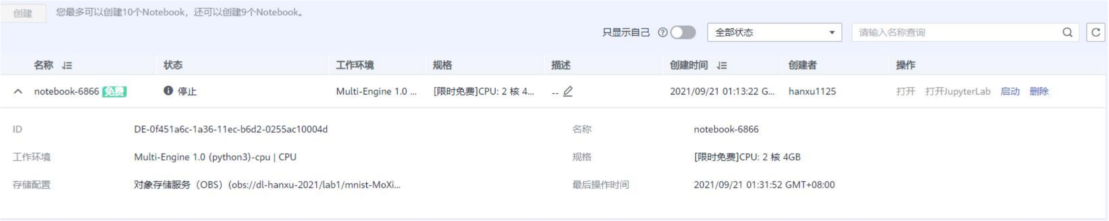
创建 obs 桶以及如下文件夹



华北-北京四 | 桶内对象总数：7419 | 存储总用量：1.28 GB

□	对象名称	存储类别	大小	最后修改时间	操作
□	dataset-mnist	--	--	--	
□	mnist-MoXing-code	--	--	--	
□	train-log-2	--	--	--	

1，创建 Notebook

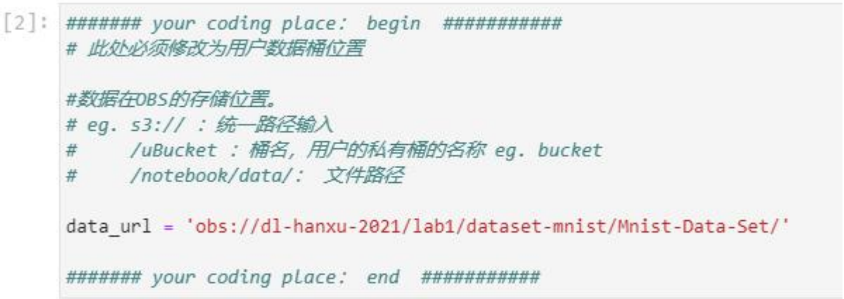


您最多可以创建10个Notebook，还可以创建9个Notebook。

名称	状态	工作环境	规格	描述	创建时间	创建者	操作
notebook-6866 免费	停止	Multi-Engine 1.0 ...	[限时免费]CPU: 2 核 4...	--	2021/09/21 01:13:22 G...	hanxu1125	打开 打开JupyterLab 启动 删除

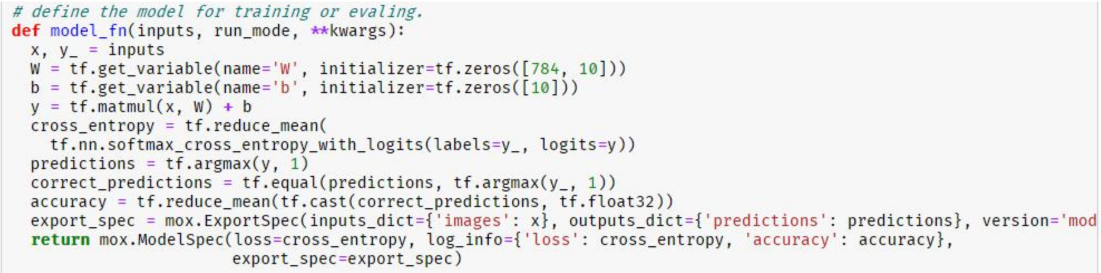
ID	DE-0f451a6c-1a36-11ec-b6d2-0255ac10004d	名称	notebook-6866
工作环境	Multi-Engine 1.0 (python3)-cpu | CPU	规格	[限时免费]CPU: 2 核 4GB
存储配置	对象存储服务（OBS）(obs://dl-hanxu-2021/lab1/mnist-MoXi...	最后操作时间	2021/09/21 01:31:52 GMT+08:00

2，使用 Notebook 训练模型，修改训练数据集路径为自己桶中

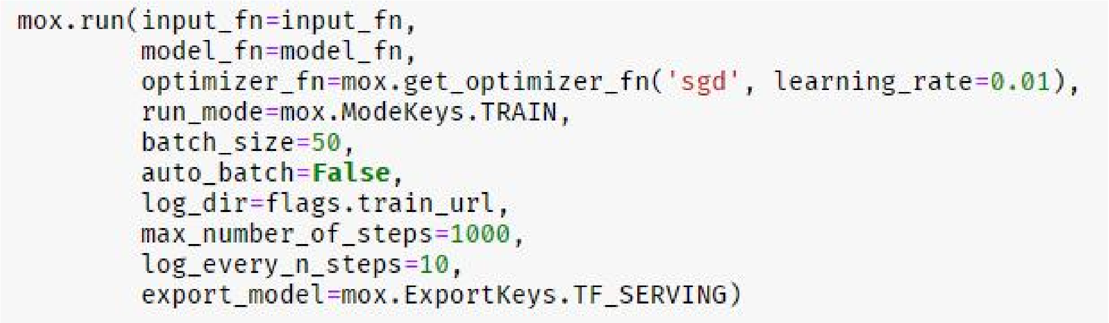


```
[2]: ####### your coding place:  begin  ###########
     # 此处必须修改为用户数据桶位置

     #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 = 'obs://dl-hanxu-2021/lab1/dataset-mnist/Mnist-Data-Set/'

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

模型结构以及训练超参：



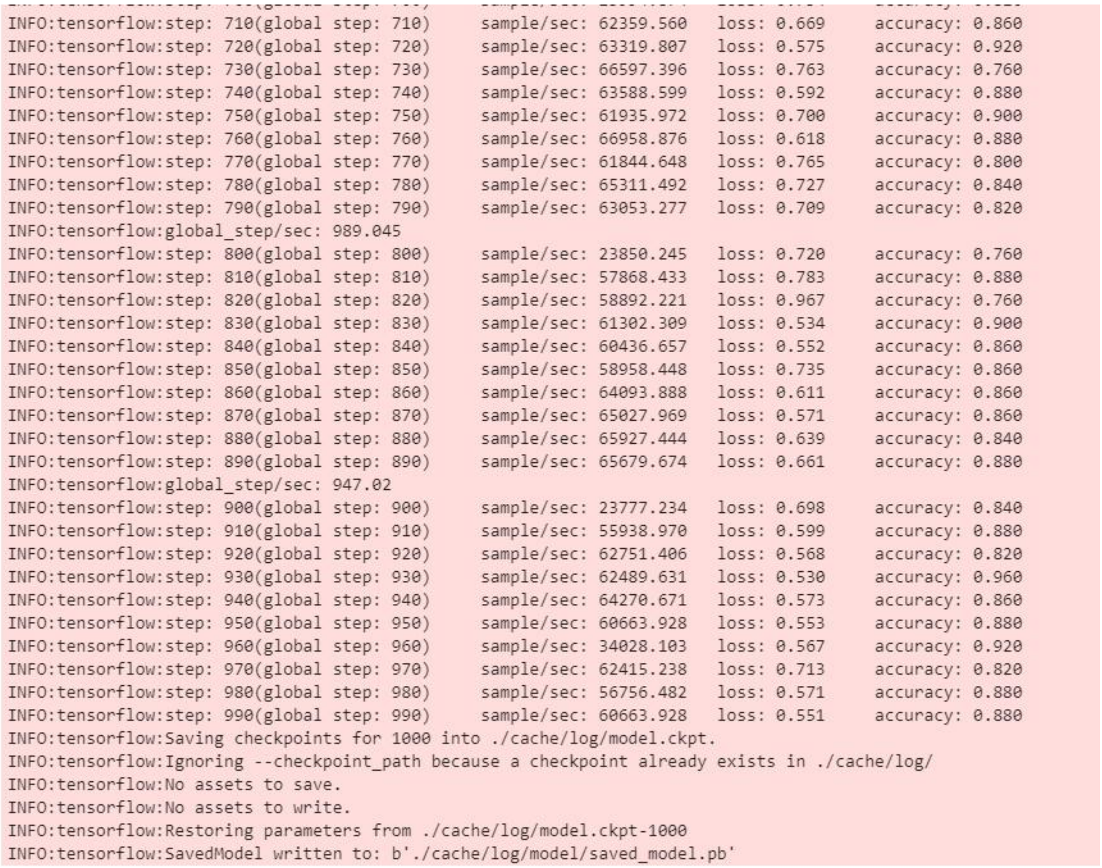
```
# define the model for training or evaling.
def model_fn(inputs, run_mode, **kwargs):
  x, y_ = inputs
  W = tf.get_variable(name='W', initializer=tf.zeros([784, 10]))
  b = tf.get_variable(name='b', initializer=tf.zeros([10]))
  y = tf.matmul(x, W) + b
  cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))
  predictions = tf.argmax(y, 1)
  correct_predictions = tf.equal(predictions, tf.argmax(y_, 1))
  accuracy = tf.reduce_mean(tf.cast(correct_predictions, tf.float32))
  export_spec = mox.ExportSpec(inputs_dict={'images': x}, outputs_dict={'predictions': predictions}, version='mod
  return mox.ModelSpec(loss=cross_entropy, log_info={'loss': cross_entropy, 'accuracy': accuracy},
                       export_spec=export_spec)
```



```
mox.run(input_fn=input_fn,
        model_fn=model_fn,
        optimizer_fn=mox.get_optimizer_fn('sgd', learning_rate=0.01),
        run_mode=mox.ModeKeys.TRAIN,
        batch_size=50,
        auto_batch=False,
        log_dir=flags.train_url,
        max_number_of_steps=1000,
        log_every_n_steps=10,
        export_model=mox.ExportKeys.TF_SERVING)
```

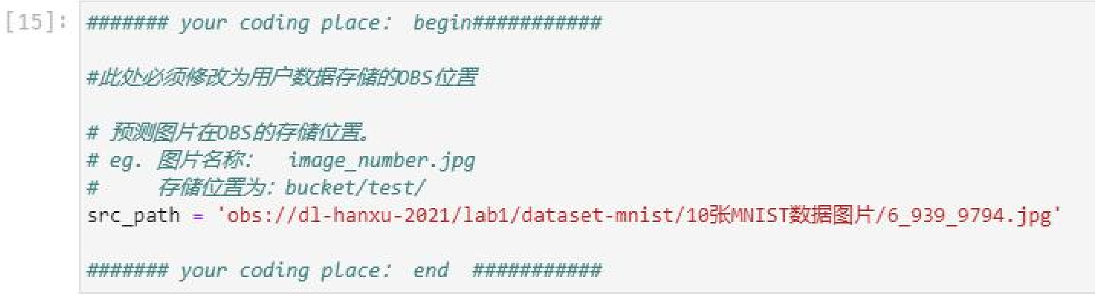
可以看到用 softmax 的交叉熵作为损失函数，优化器是 SGD，学习率 0.01，batch_size 为 50

3，训练过程如下



```
INFO:tensorflow:step: 710(global step: 710)     sample/sec: 62359.560   loss: 0.669     accuracy: 0.860
INFO:tensorflow:step: 720(global step: 720)     sample/sec: 63319.807   loss: 0.575     accuracy: 0.920
INFO:tensorflow:step: 730(global step: 730)     sample/sec: 66597.396   loss: 0.763     accuracy: 0.760
INFO:tensorflow:step: 740(global step: 740)     sample/sec: 63588.599   loss: 0.592     accuracy: 0.880
INFO:tensorflow:step: 750(global step: 750)     sample/sec: 61935.972   loss: 0.700     accuracy: 0.900
INFO:tensorflow:step: 760(global step: 760)     sample/sec: 66958.876   loss: 0.618     accuracy: 0.880
INFO:tensorflow:step: 770(global step: 770)     sample/sec: 61844.648   loss: 0.765     accuracy: 0.800
INFO:tensorflow:step: 780(global step: 780)     sample/sec: 65311.492   loss: 0.727     accuracy: 0.840
INFO:tensorflow:step: 790(global step: 790)     sample/sec: 63053.277   loss: 0.709     accuracy: 0.820
INFO:tensorflow:global_step/sec: 989.045
INFO:tensorflow:step: 800(global step: 800)     sample/sec: 23850.245   loss: 0.720     accuracy: 0.760
INFO:tensorflow:step: 810(global step: 810)     sample/sec: 57868.433   loss: 0.783     accuracy: 0.880
INFO:tensorflow:step: 820(global step: 820)     sample/sec: 58892.221   loss: 0.967     accuracy: 0.760
INFO:tensorflow:step: 830(global step: 830)     sample/sec: 61302.309   loss: 0.534     accuracy: 0.900
INFO:tensorflow:step: 840(global step: 840)     sample/sec: 60436.657   loss: 0.552     accuracy: 0.860
INFO:tensorflow:step: 850(global step: 850)     sample/sec: 58958.448   loss: 0.735     accuracy: 0.860
INFO:tensorflow:step: 860(global step: 860)     sample/sec: 64093.888   loss: 0.611     accuracy: 0.860
INFO:tensorflow:step: 870(global step: 870)     sample/sec: 65027.969   loss: 0.571     accuracy: 0.860
INFO:tensorflow:step: 880(global step: 880)     sample/sec: 65927.444   loss: 0.639     accuracy: 0.840
INFO:tensorflow:step: 890(global step: 890)     sample/sec: 65679.674   loss: 0.661     accuracy: 0.880
INFO:tensorflow:global_step/sec: 947.02
INFO:tensorflow:step: 900(global step: 900)     sample/sec: 23777.234   loss: 0.698     accuracy: 0.840
INFO:tensorflow:step: 910(global step: 910)     sample/sec: 55938.970   loss: 0.599     accuracy: 0.880
INFO:tensorflow:step: 920(global step: 920)     sample/sec: 62751.406   loss: 0.568     accuracy: 0.820
INFO:tensorflow:step: 930(global step: 930)     sample/sec: 62489.631   loss: 0.530     accuracy: 0.960
INFO:tensorflow:step: 940(global step: 940)     sample/sec: 64270.671   loss: 0.573     accuracy: 0.860
INFO:tensorflow:step: 950(global step: 950)     sample/sec: 60663.928   loss: 0.553     accuracy: 0.880
INFO:tensorflow:step: 960(global step: 960)     sample/sec: 34028.103   loss: 0.567     accuracy: 0.920
INFO:tensorflow:step: 970(global step: 970)     sample/sec: 62415.238   loss: 0.713     accuracy: 0.820
INFO:tensorflow:step: 980(global step: 980)     sample/sec: 56756.482   loss: 0.571     accuracy: 0.880
INFO:tensorflow:step: 990(global step: 990)     sample/sec: 60663.928   loss: 0.551     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'
```

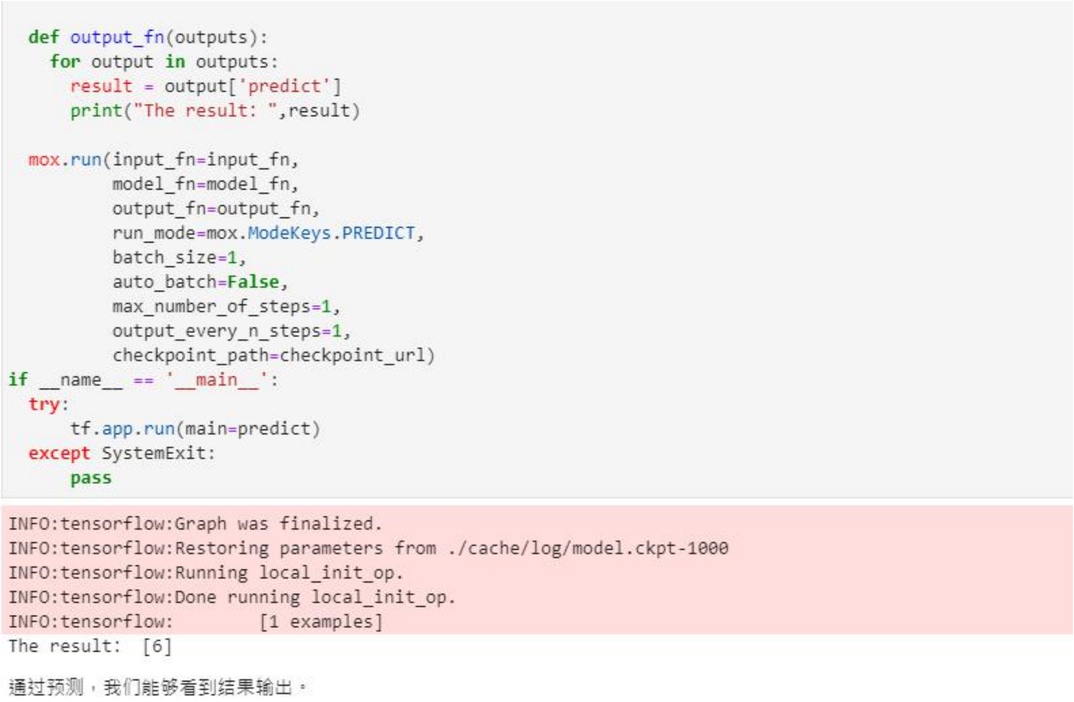
4，修改测试图片路径



[15]:

```
####### your coding place:  begin###########

#此处必须修改为用户数据存储的OBS位置

# 预测图片在OBS的存储位置。
# eg. 图片名称:   image_number.jpg
#     存储位置为: bucket/test/
src_path = 'obs://dl-hanxu-2021/lab1/dataset-mnist/10张MNIST数据图片/6_939_9794.jpg'

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

5，完成预测，结果如下



```
  def output_fn(outputs):
    for output in outputs:
      result = output['predict']
      print("The result: ",result)

  mox.run(input_fn=input_fn,
          model_fn=model_fn,
          output_fn=output_fn,
          run_mode=mox.ModeKeys.PREDICT,
          batch_size=1,
          auto_batch=False,
          max_number_of_steps=1,
          output_every_n_steps=1,
          checkpoint_path=checkpoint_url)
if __name__ == '__main__':
  try:
      tf.app.run(main=predict)
  except SystemExit:
      pass
```

```
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:  [6]
```

通过预测，我们能够看到结果输出。

可以看到模型预测成功，结果正确。

结论分析与体会：

1，学会使用 ModelArts 训练模型以及模型部署，最后预测

2，学会使用 Notebook 训练模型并且完成预测

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，新版 Notebook 的运行环境镜像过少，要想使用 tf1.15 以下只能想办法进到老版 Notebook

2，华为云的示例文档没有更新，有些步骤无法匹配新版功能，只能舍去或者找到旧版界面

3，有的时候显卡会有些紧张，要排一个多小时的队才有资源