

**人工智能实验报告**

题 目 MNIST手写体识别

专 业 信息安全

学　　 号 2021111000

姓 名 李卓凌

1. **背景简介/问题描述**
   1. 基于Mindspore框架的模型本地训练及预测
2. 定义LeNet网络及损失函数
3. 加载数据集并进行训练，查看结果并保存模型
4. 加载保存模型并进行推理
   1. 基于Modelarts平台和Tensorflow框架的模型训练及部署
5. 准备MNIST数据集
6. 创建OBS桶并上传训练文件和推理文件
7. 创建训练作业并推理部署
8. **算法介绍**

2.1 基于Mindspore框架的模型本地训练及预测

1. 导入python库和模块并配置运行信息

使用context.set\_context来配置运行所需的信息，包括运行模式、后端信息等信息。

1. 数据处理

使用MindSpore中的create\_dataset函数来创建数据集，并在函数进行数据增强和处理操作，具体包括

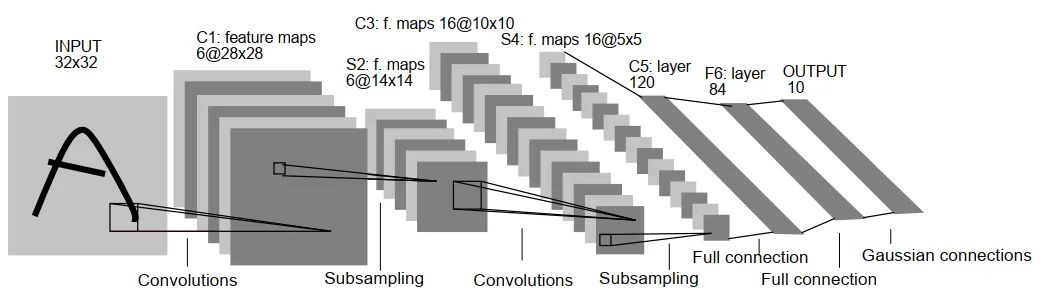
1. 定义数据集
2. 定义进行数据增强和处理所需的参数
3. 根据参数生成对用的数据增强操作
4. 使用map映射函数将数据操作应用到数据集
5. 对生成的数据集进行处理
6. 定义LetNet网络

LeNet-5共有7层，不包含输入层，每层都包含可训练参数；每个层有多个Feature Map，每个Feature Map通过一种卷积滤波器提取输入的一种特征，然后每个Feature Map有多个神经元。

(1) INPUT层 首先是数据 INPUT 层，输入图像的尺寸统一归一化为 32x32。 注：INPUT层不属于LeNet-5的网络结构，传统上，一般不将INPUT层视为网络层次结构之一。

(2) C1层 (卷积层) 输入图片：32x32 卷积核大小：5x5 卷积核种类：6 输出6个特征图(feature map)，每个特征图的大小：28x28

网络构造如图1所示



图表 1 LetNet\_5网络构造

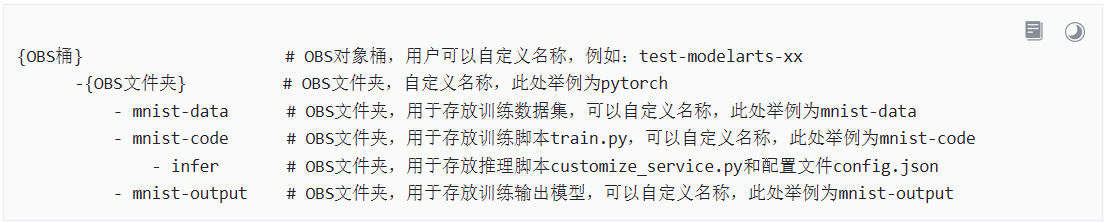
1. 定义损失函数及优化器

损失函数使用MindSpore中的SoftmaxCrossEntropyWithLogits

1. 进行训练及验证

2.2 基于Modelarts平台和Tensorflow框架的模型训练及部署

1. 准备数据集和train.py,customize\_serve.py代码文件
2. 创建OBS桶并上传文件，OBS桶格式如图2所示



图表 2 OBS桶

1. 创建训练作业

启动方式选择“预置框架”，下拉框中选择PyTorch，pytorch\_1.8.0-cuda\_10.2-py\_3.7-ubuntu\_18.04-x86\_64。

代码目录选择已创建的代码目录路径“/test-modelarts-xx/pytorch/mnist-code/”。

启动文件选择代码目录下上传的训练脚本“train.py”。

输入单击“添加”，设置训练输入的“参数名称”为“data\_url”。设置数据存储位置为 “/test-modelarts-xx/pytorch/mnist-data/”。

输出单击“添加”，设置训练输出的“参数名称”为“train\_url”。设置数据存储位置为 “/test-modelarts-xx/pytorch/mnist-output/”。

资源类型选择 GPU 单卡的规格，如“GPU: 1\*NVIDIA-V100(16GB) | CPU: 8 核 64GB 780GB”

1. 预测结果，选择“multipart/form-data”，上传图片并进行预测
2. **算法实现**

3.1 tain.py代码如下

# Copyright 2020 Huawei Technologies Co., Ltd

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

# ============================================================================

"""

######################## train lenet example ########################

train lenet and get network model files(.ckpt) :

python train.py --data\_path /YourDataPath

"""

import os

import ast

import argparse

from src.config import cfg as cfg

from src.dataset import create\_dataset

from src.lenet import LeNet5

import mindspore.nn as nn

from mindspore import context

from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor, TimeMonitor

from mindspore.train import Model

from mindspore.nn.metrics import Accuracy

from mindspore.common import set\_seed

# 设置MindSpore的执行模式和设备

context.set\_context(mode=context.GRAPH\_MODE, device\_target='CPU') # Ascend, CPU, GPU

set\_seed(1)

if \_\_name\_\_ == "\_\_main\_\_":

parser = argparse.ArgumentParser(description='MindSpore Lenet Example')

# 设备设置

parser.add\_argument('--device\_target', type=str, default='CPU', choices=['Ascend', 'GPU', 'CPU'],

help='device where the code will be implemented (default: Ascend)')

parser.add\_argument('--data\_path', type=str, default="./MNIST\_Data",

help='path where the dataset is saved')

parser.add\_argument('--ckpt\_path', type=str, default="./ckpt", help='if is test, must provide\

path where the trained ckpt file')

parser.add\_argument('--dataset\_sink\_mode', type=ast.literal\_eval, default=True,

help='dataset\_sink\_mode is False or True')

args = parser.parse\_args()

context.set\_context(mode=context.GRAPH\_MODE, device\_target=args.device\_target)

ds\_train = create\_dataset(os.path.join(args.data\_path, "train"),

cfg.batch\_size)

network = LeNet5(cfg.num\_classes)

#设定loss函数

net\_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")

#设定优化器

net\_opt = nn.Momentum(network.trainable\_params(), cfg.lr, cfg.momentum)

time\_cb = TimeMonitor(data\_size=ds\_train.get\_dataset\_size())

config\_ck = CheckpointConfig(save\_checkpoint\_steps=cfg.save\_checkpoint\_steps,

keep\_checkpoint\_max=cfg.keep\_checkpoint\_max)

ckpoint\_cb = ModelCheckpoint(prefix="checkpoint\_lenet", directory=args.ckpt\_path, config=config\_ck)

#编译形成模型

model = Model(network, net\_loss, net\_opt, metrics={"Accuracy": Accuracy()})

print("============== Starting Training ==============")

# 训练网络 train.py

model.train(cfg['epoch\_size'], ds\_train, callbacks=[time\_cb, ckpoint\_cb, LossMonitor()]

, dataset\_sink\_mode=args.dataset\_sink\_mode)

3.2 eval.py代码如下

import os

import ast

import argparse

import mindspore.nn as nn

from mindspore import context

from mindspore.train.serialization import load\_checkpoint, load\_param\_into\_net

from mindspore.train import Model

from mindspore.nn.metrics import Accuracy

from src.dataset import create\_dataset

from src.config import cfg as cfg

from src.lenet import LeNet5

# 设置MindSpore的执行模式和设备

context.set\_context(mode=context.GRAPH\_MODE, device\_target='CPU') # Ascend, CPU, GPU

if \_\_name\_\_ == "\_\_main\_\_":

parser = argparse.ArgumentParser(description='MindSpore Lenet Example')

# 设备设置

parser.add\_argument('--device\_target', type=str, default='CPU', choices=['Ascend', 'GPU', 'CPU'],

help='device where the code will be implemented (default: Ascend)')

parser.add\_argument('--data\_path', type=str, default="./MNIST\_Data",

help='path where the dataset is saved')

parser.add\_argument('--ckpt\_path', type=str, default="", help='if mode is test, must provide\

path where the trained ckpt file')

parser.add\_argument('--dataset\_sink\_mode', type=ast.literal\_eval,

default=False, help='dataset\_sink\_mode is False or True')

args = parser.parse\_args()

context.set\_context(mode=context.GRAPH\_MODE, device\_target=args.device\_target)

ds\_eval = create\_dataset(os.path.join(args.data\_path, "test"), cfg.batch\_size, 1)

network = LeNet5(cfg.num\_classes)

#设定loss函数

net\_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")

repeat\_size = cfg.epoch\_size

#设定优化器

net\_opt = nn.Momentum(network.trainable\_params(), cfg.lr, cfg.momentum)

#编译形成模型

model = Model(network, net\_loss, net\_opt, metrics={"Accuracy": Accuracy()})

print("============== Starting Testing ==============")

param\_dict = load\_checkpoint("./ckpt/checkpoint\_lenet-6\_1875.ckpt") # args.ckpt\_path

load\_param\_into\_net(network, param\_dict)

ds\_eval = create\_dataset(os.path.join(args.data\_path, "test"),

cfg.batch\_size,

1)

acc = model.eval(ds\_eval, dataset\_sink\_mode=args.dataset\_sink\_mode)

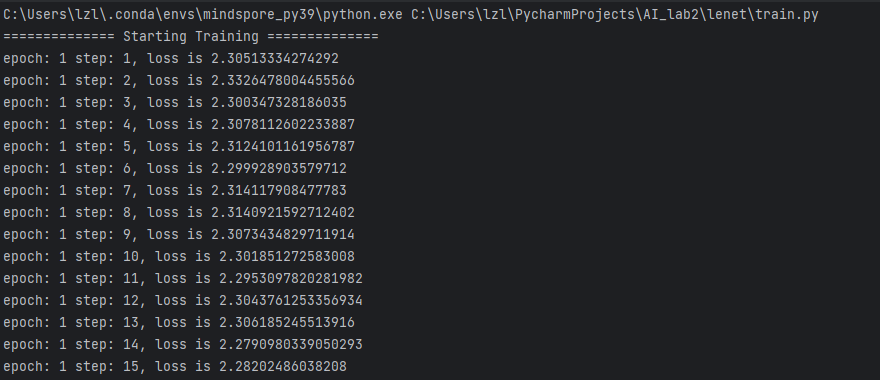
print("============== {} ==============".format(acc)

1. **讨论及结论**

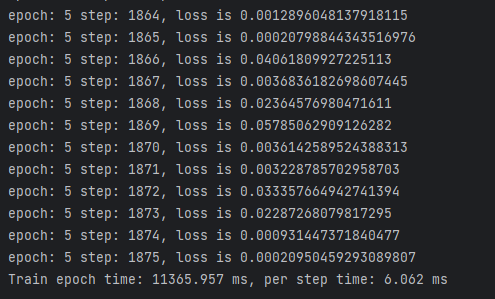
4.1 基于MindSpore框架的实现结果如图3，4，5所示

可以看到数据损失从最初的2.3，经过五层网络学习后，数据损失降低到0.0002

观察eval.py结果，看到准确率高达0.99，正确率较高，说明模型有效，准确率较好



图表 3 train.py1



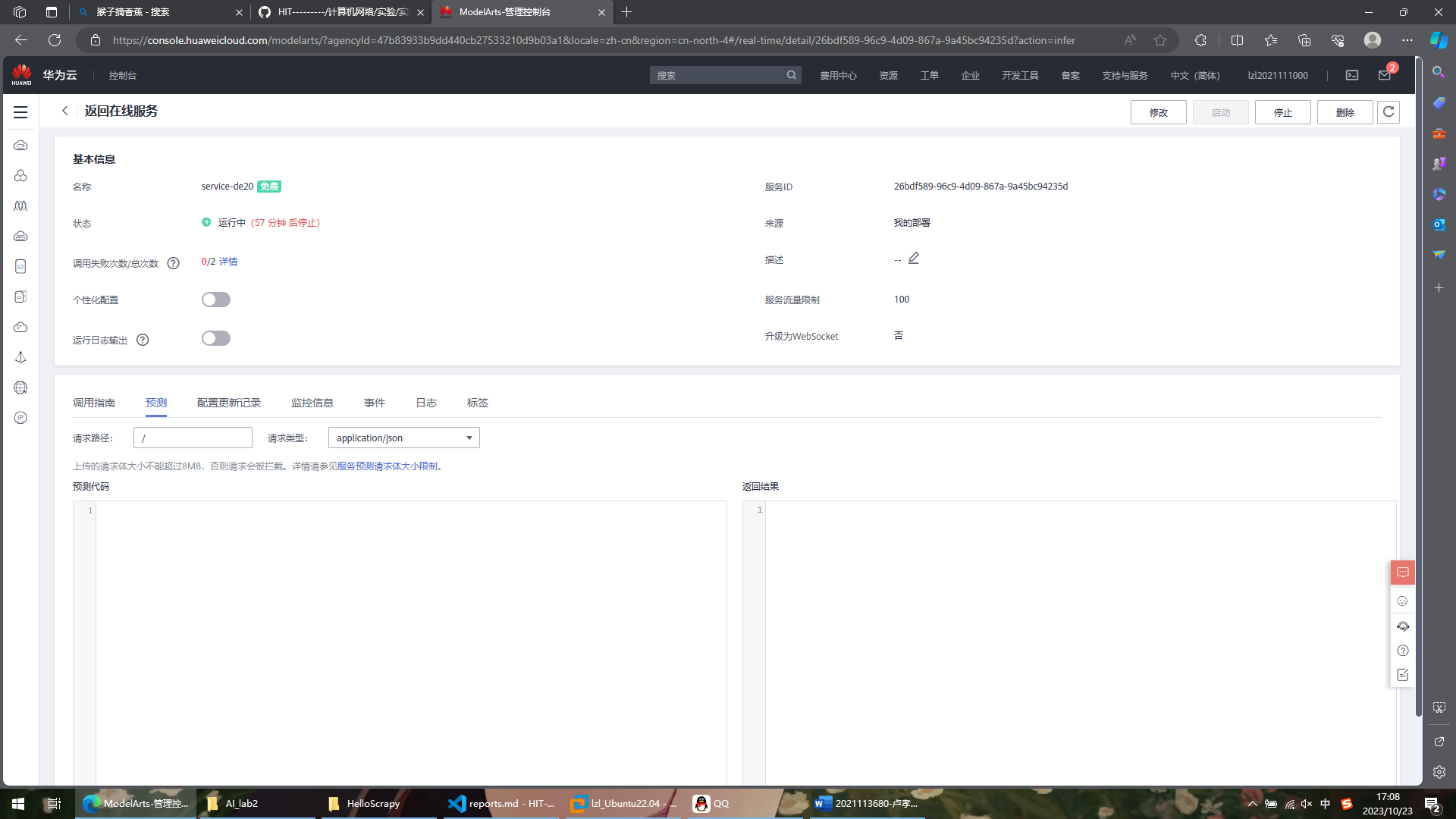
图表 4 train.py2



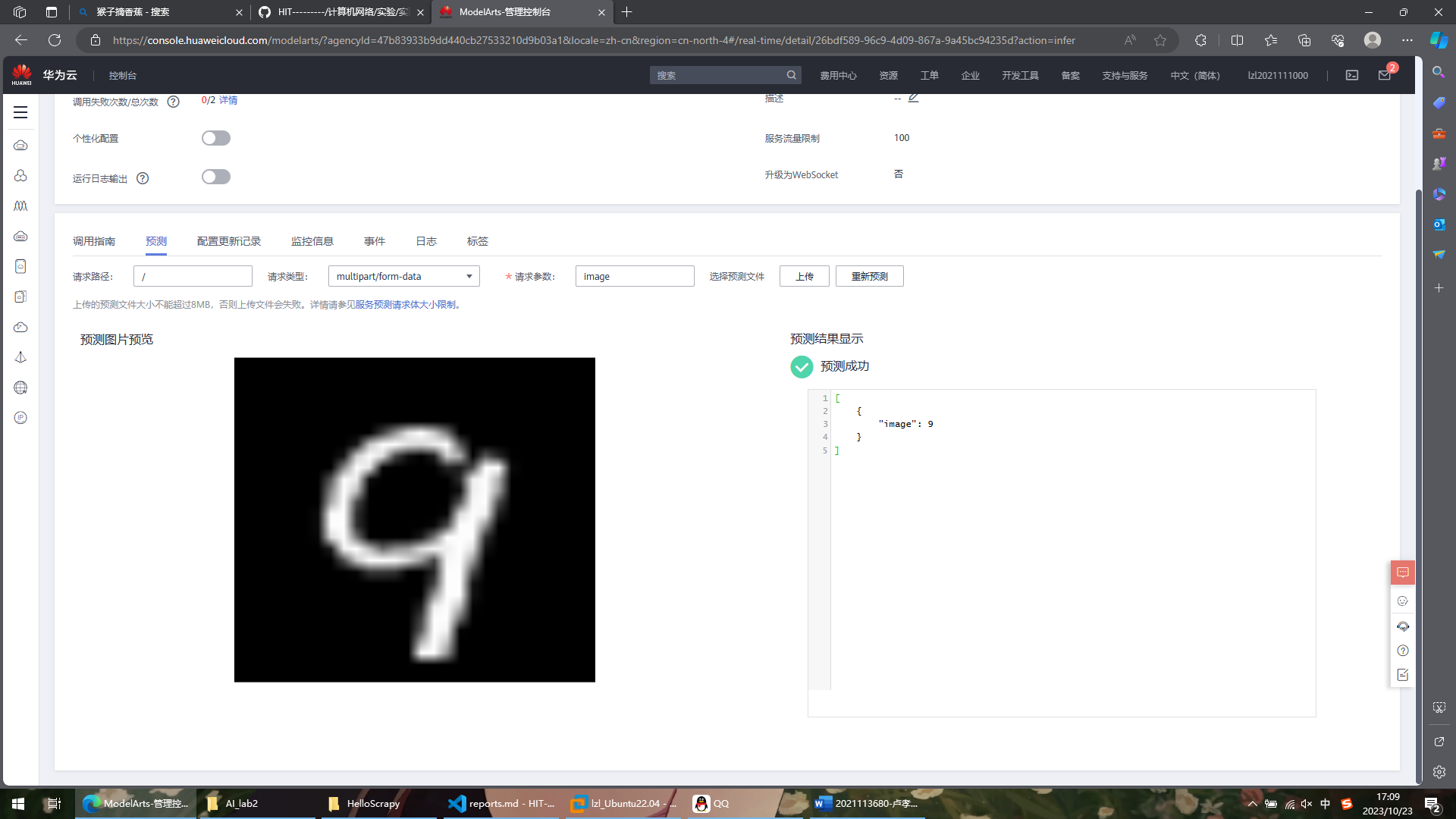
图表 5 eval.py 结果

4.2 基于Modelarts平台和Tensorflow框架的模型训练及部署

如图6所示，成功完成了作业训练以及项目部署，如图7所示，上传手写数字2，成功识别，可见模型训练的可靠性。



图表 6 部署成功截图



图表 7 预测实验成功截图

1. **参考文献**
2. **[LeCun et al., 1998]Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998. Gzipped PostScript, 75 pages, 709897 bytes**
3. **Bottou et al., 1997]L. Bottou, Y. LeCun, and Y. Bengio. Global training of document processing systems using graph transformer networks. In Proc. of Computer Vision and Pattern Recognition, Puerto-Rico, 1997. IEEE. DjVu Gzipped PostScript, 6 pages, 61286 bytes**