

**人工智能实验报告**

题 目 基于Mindspore框架与ModelArts 平台的MNIST手写体识别实验

专 业 人工智能（视听觉信息处理）

学　　 号 7203610121

姓 名 刘天瑞

同 组 人 员 无

1. **背景简介/问题描述**

**实验介绍**

本例子会实现一个简单的图片分类的功能，整体流程如下：

1、处理需要的数据集，这里使用了MNIST数据集。

1. 定义一个网络，这里我们使用LeNet网络。
2. 定义损失函数和优化器。
3. 加载数据集并进行训练，训练完成后，查看结果及保存模型文件。
4. 加载保存的模型，进行推理。

验证模型，加载测试数据集和训练后的模型，验证结果精度。

**实验准备**

在动手进行实践之前，确保，你已经正确安装了MindSpore。如果没有，可以通过MindSpore官网安装页面：<https://www.mindspore.cn/install/> ，将MindSpore安装在你的电脑当中。

同时希望你拥有Python编码基础和概率、矩阵等基础数学知识。

**实验详细设计与实现**

**数据准备**

我们示例中用到的MNIST数据集是由10类28\*28的灰度图片组成，训练数据集包含60000张图片，测试数据集包含10000张图片。

MNIST数据集下载页面：http://yann.lecun.com/exdb/mnist/。页面提供4个数据集下载链接，其中前2个文件是训练数据需要，后2个文件是测试结果需要。

将数据集下载并解压到本地路径下，这里将数据集解压分别存放到工作区的./MNIST\_Data/train、./MNIST\_Data/test路径下。

目录结构如下：

└─MNIST\_Data

├─ test

│ t10k-images.idx3-ubyte

│ t10k-labels.idx1-ubyte

│

└─ train

train-images.idx3-ubyte

train-labels.idx1-ubyte

为了方便样例使用，我们在样例脚本中添加了自动下载数据集的功能。

**实验步骤**

1. 导入Python库&模块并配置运行信息

在使用前，导入需要的Python库。

目前使用到os库，为方便理解，其他需要的库，我们在具体使用到时再说明。

详细的MindSpore的模块说明，可以在MindSpore API页面中搜索查询。

可以通过context.set\_context来配置运行需要的信息，譬如运行模式、后端信息、硬件等信息。

导入context模块，配置运行需要的信息。

parser.add\_argument('--device\_target', type=str, default=" Ascend", choices=['Ascend', 'GPU', 'CPU'], help='device where the code will be implemented (default: Ascend)')

在样例中我们配置样例运行使用图模式。再train.py和eval.py根据实际情况配置硬件信息，譬如代码运行在Ascend AI处理器上，则device\_target选择Ascend，代码运行在CPU、GPU同理。详细参数说明，请参见context.set\_context接口说明。使用本地和华为云处理器完成模型的训练。

1. 数据处理

#根据数据集存储地址，生成数据集，构建训练、验证函数

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

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

数据集对于训练非常重要，好的数据集可以有效提高训练精度和效率。在加载数据集前，我们通常会对数据集进行一些处理。步骤二的以下内容里，mindspore 提供了更多数据处理的方法，供大家以后使用mindspore时参考：

/\*--------定义数据集及数据操作

我们定义一个函数create\_dataset来创建数据集。在这个函数中，我们定义好需要进行的数据增强和处理操作：

1. 定义数据集。

2. 定义进行数据增强和处理所需要的一些参数。

3. 根据参数，生成对应的数据增强操作。

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。

#根据数据集存储地址，生成数据集

def create\_dataset(data\_dir, training=True, batch\_size=32, resize=(32, 32),

rescale=1/(255\*0.3081), shift=-0.1307/0.3081, buffer\_size=64):

#生成训练集和测试集的路径

data\_train = os.path.join(data\_dir, 'train') # train set

data\_test = os.path.join(data\_dir, 'test') # test set

#利用MnistDataset方法读取mnist数据集，如果training是True则读取训练集

ds = ms.dataset.MnistDataset(data\_train if training else data\_test)

#map方法是非常有效的方法，可以整体对数据集进行处理，resize改变数据形状，rescale进行归一化，HWC2CHW改变图像通道

ds = ds.map(input\_columns=["image"], operations=[CV.Resize(resize), CV.Rescale(rescale, shift), CV.HWC2CHW()])

#利用map方法改变数据集标签的数据类型

ds = ds.map(input\_columns=["label"], operations=C.TypeCast(ms.int32))

# shuffle是打乱操作，同时设定了batchsize的大小，并将最后不足一个batch的数据抛弃

ds = ds.shuffle(buffer\_size=buffer\_size).batch(batch\_size, drop\_remainder=True)

return ds

其中，

batch\_size：每组包含的数据个数，现设置每组包含32个数据。

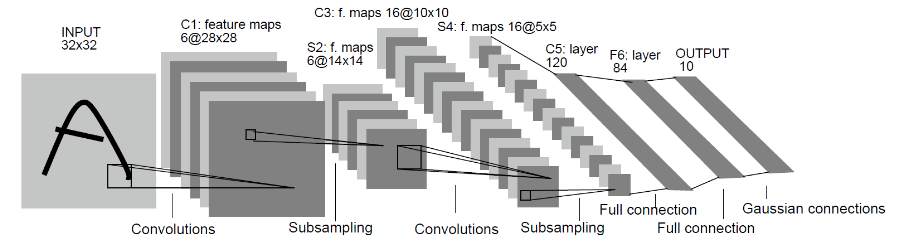
先进行修改图片尺寸，归一化，修改图像频道数等工作，再修改标签的数据类型。最后进行shuffle操作，同时设定batch\_size，设置drop\_remainder为True，则数据集中不足最后一个batch的数据会被抛弃。

MindSpore支持进行多种数据处理和增强的操作，各种操作往往组合使用，具体可以参考数据处理与数据增强章节。--------\*/

1. 定义网络

我们选择相对简单的LeNet网络。LeNet网络不包括输入层的情况下，共有7层：2个卷积层、2个下采样层（池化层）、3个全连接层。每层都包含不同数量的训练参数，如下图所示：

LeNet-5



更多的LeNet网络的介绍不在此赘述，希望详细了解LeNet网络，可以查询http://yann.lecun.com/exdb/lenet/。

使用MindSpore定义神经网络需要继承mindspore.nn.cell.Cell。Cell是所有神经网络（Conv2d等）的基类。

神经网络的各层需要预先在\_\_init\_\_方法中定义，然后通过定义construct方法来完成神经网络的前向构造。按照LeNet的网络结构，定义网络各层如下：

#定义模型结构，MindSpore中的模型时通过construct定义模型结构，在\_\_init\_\_中初始化各层的对象

class LeNet5(nn.Cell):

def \_\_init\_\_(self, num\_class=10, num\_channel=1):

super(LeNet5, self).\_\_init\_\_()

#定义卷积层，ReLU激活函数，平坦层和全连接层

#conv2d的输入通道为1维，输出为6维，卷积核尺寸为5\*5，步长为1，不适用padding

self.conv1 = nn.Conv2d(num\_channel, 6, 5, pad\_mode='valid')

self.conv2 = nn.Conv2d(6, 16, 5, pad\_mode='valid')

self.fc1 = nn.Dense(16 \* 5 \* 5, 120, weight\_init=Normal(0.02))

self.fc2 = nn.Dense(120, 84, weight\_init=Normal(0.02))

self.fc3 = nn.Dense(84, num\_class, weight\_init=Normal(0.02))

self.relu = nn.ReLU()

self.max\_pool2d = nn.MaxPool2d(kernel\_size=2, stride=2)

self.flatten = nn.Flatten()

#构建Lenet5架构，x代表网络的输入

def construct(self, x):

x = self.max\_pool2d(self.relu(self.conv1(x)))

x = self.max\_pool2d(self.relu(self.conv2(x)))

x = self.flatten(x)

x = self.relu(self.fc1(x))

x = self.relu(self.fc2(x))

x = self.fc3(x)

return x

1. 定义损失函数及优化器

在进行定义之前，先简单介绍损失函数及优化器的概念。

损失函数：又叫目标函数，用于衡量预测值与实际值差异的程度。深度学习通过不停地迭代来缩小损失函数的值。定义一个好的损失函数，可以有效提高模型的性能。

优化器：用于最小化损失函数，从而在训练过程中改进模型。

定义了损失函数后，可以得到损失函数关于权重的梯度。梯度用于指示优化器优化权重的方向，以提高模型性能。

定义损失函数

MindSpore支持的损失函数有SoftmaxCrossEntropyWithLogits、L1Loss、MSELoss等。这里使用SoftmaxCrossEntropyWithLogits损失函数。

MindSpore提供了callback机制，可以在训练过程中执行自定义逻辑，这里使用框架提供的ModelCheckpoint为例。 ModelCheckpoint可以保存网络模型和参数，以便进行后续的fine-tuning（微调）操作。

#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)

#编译形成模型

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

# 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)

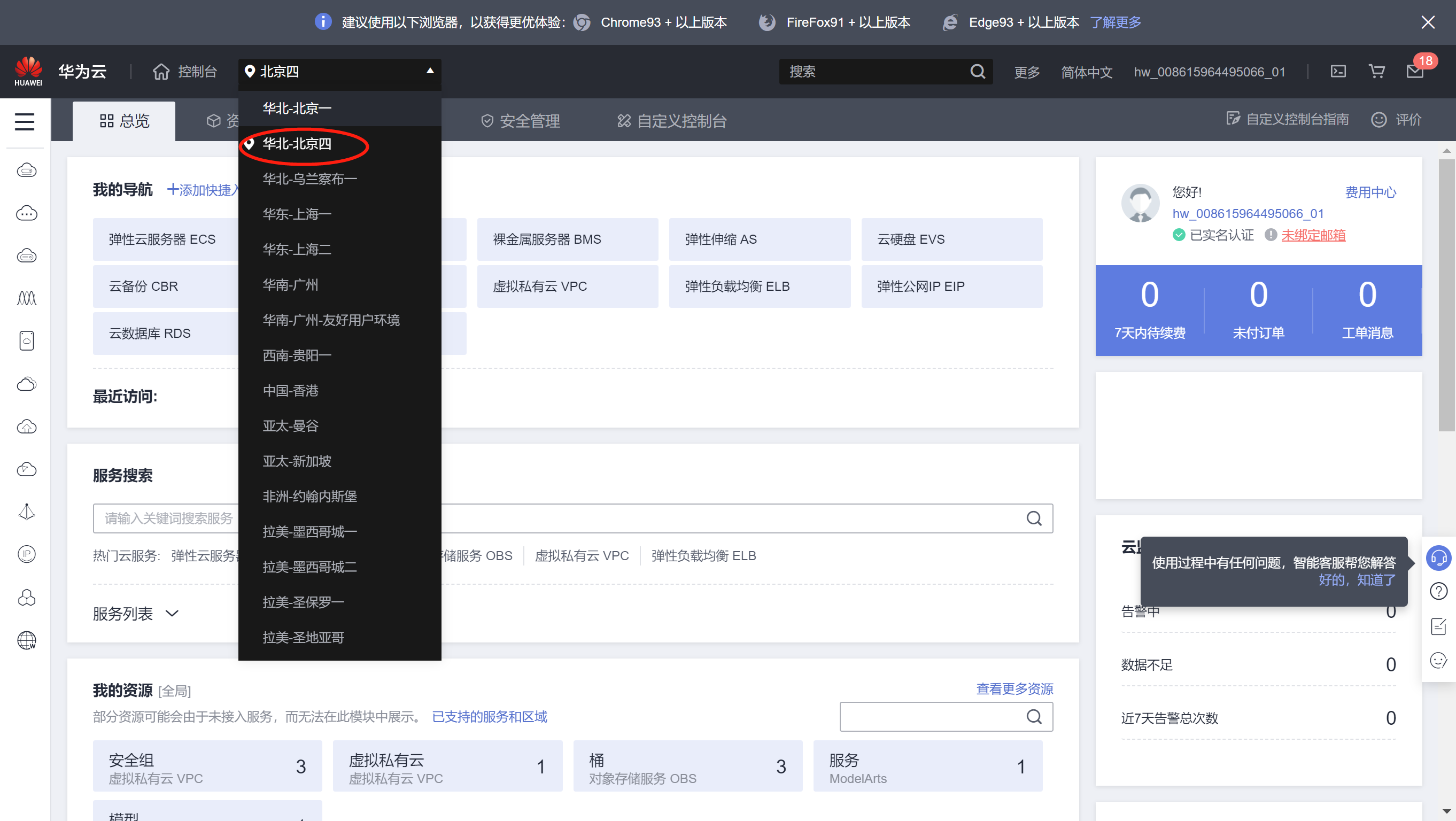
1. 开始训练及验证过程

利用train.py和eval.py完成模型的训练及验证过程。

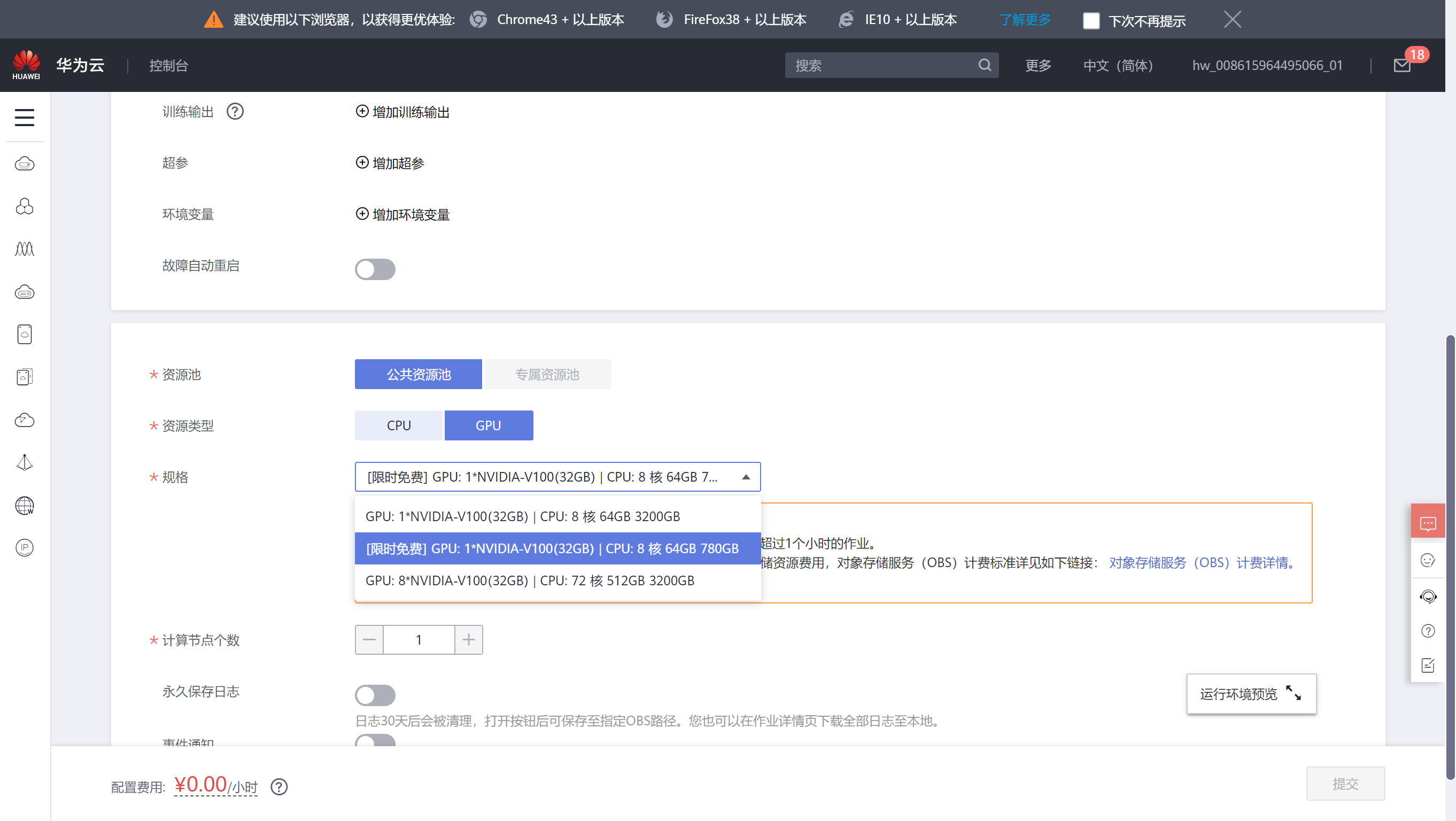
训练过程中会打印loss值。loss值会波动，但总体来说loss值会逐步减小，精度逐步提高。每个人运行的loss值有一定随机性，不一定完全相同。

1. **基于Modelarts平台和Tensorflow框架的模型训练及部署（请勿付费）**

**注：控制台服务器要选择：华北-北京四，如下图。**



**模型训练：请选择免费GPU资源，如下图。请勿付费。**



参考华为官方网页实例（<https://support.huaweicloud.com/bestpractice-modelarts/modelarts_10_0080.html>）在昇腾服务器上实现Tensorflow模型的训练部署及测试

一、这其中待解决问题的解释：

该实验包含了 3 部分内容：

1. 基于 Mindspore 框架的模型本地训练及预测；

2. 基于 Modelarts 平台和 Tensorflow 框架的模型训练及部署(请勿付费)。

二、问题的形式化描述

1. 基于 Mindspore 框架的模型本地训练及预测；

2. 处理需要的数据集，这里使用了 MNIST 数据集；

3. 定义一个网络，这里我们使用 LeNet 网络；

4. 定义损失函数和优化器；

5. 加载数据集并进行训练，训练完成后，查看结果及保存模型文件；

6. 加载保存的模型，进行推理；

7. 验证模型，加载测试数据集和训练后的模型，验证结果精度；

8. 基于 Modelarts 平台和 Tensorflow 框架的模型训练及部署(请勿付费)本部分只需要根据指导书中网址里面的步骤执行即可。

三、解决方案介绍

1. 基于 Mindspore 框架的模型本地训练及预测：

该部分需要根据指导书中的步骤下载好准备的文件，然后再补全代码即可；

2. 基于 Modelarts 平台和 Tensorflow 框架的模型训练及部署(请勿付费)：

该部分须在华为云完成，按照网页上的步骤完成即可。

1. **算法介绍**

2.1 所用算法及的解题思路的一般介绍

由实验目的可知在实验中会使用到的MNIST数据集，它是由10类28\*28的灰度图片组成，训练数据集总共包含60000张图片，测试数据集总共包含10000张图片。在代码中定义一个函数create\_dataset()来创建数据集。在这个函数中定义明确需要进行的数据增强和处理操作：

1. 定义数据集；

2. 定义进行数据增强和处理所需要的一些参数；

3. 根据参数，生成对应的数据增强操作；

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。

在处理完数据之后便开始设计神经网络，这里选择的是相对简单的lenet网络。lenet网络在不包括输入层的情况下，共有以下7层分别是：2个卷积层、2个下采样层（池化层）以及3个全连接层，其中每层都包含不同数量的训练参数。

最后还定义损失函数，从这里开始就可以开始训练了。

1. **算法实现**

3.1 实验结果（按照指导书一步步执行）

以下各截图为代码实现部分：

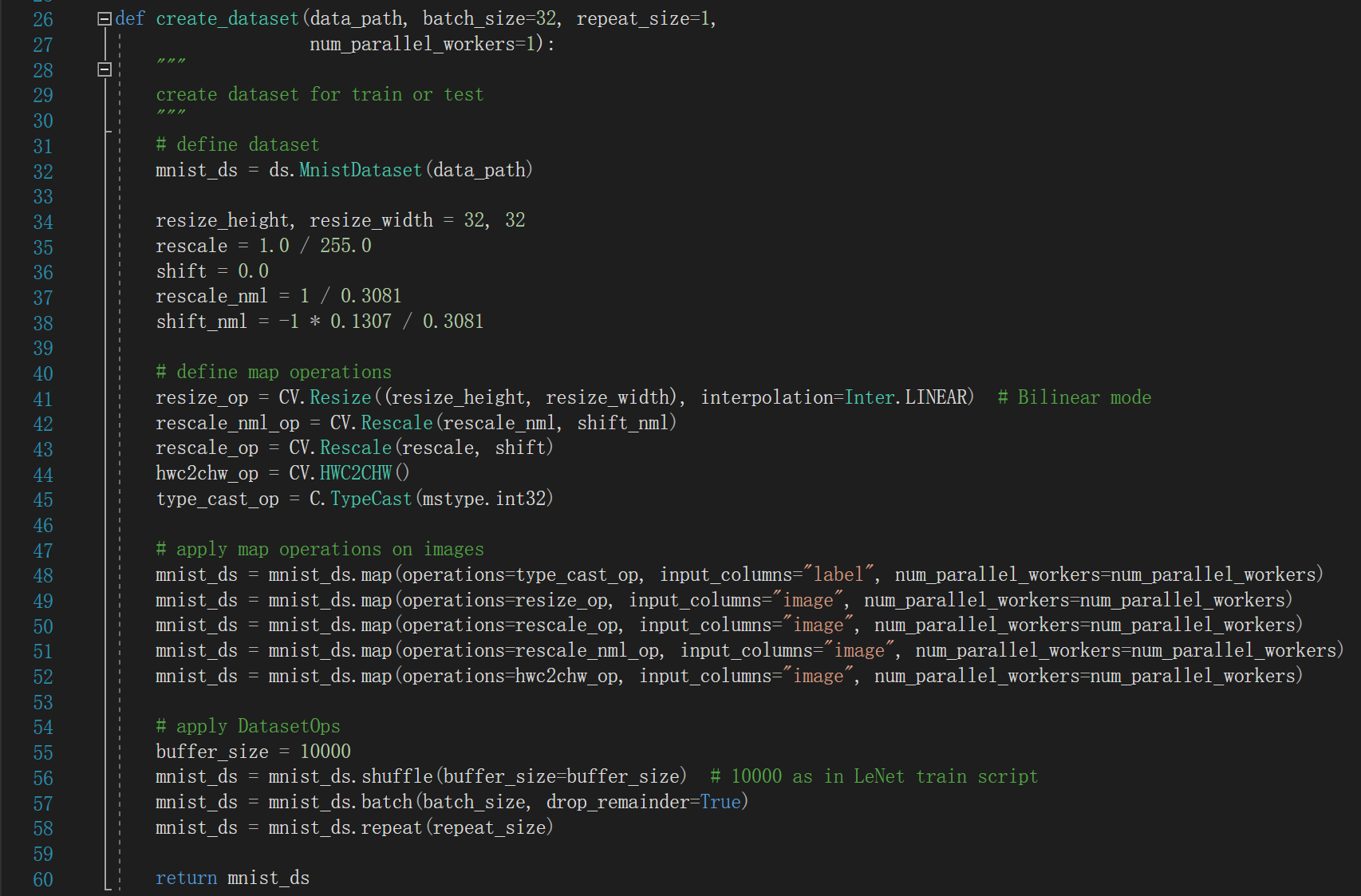


图1. 数据集处理部分的create\_dataset()函数

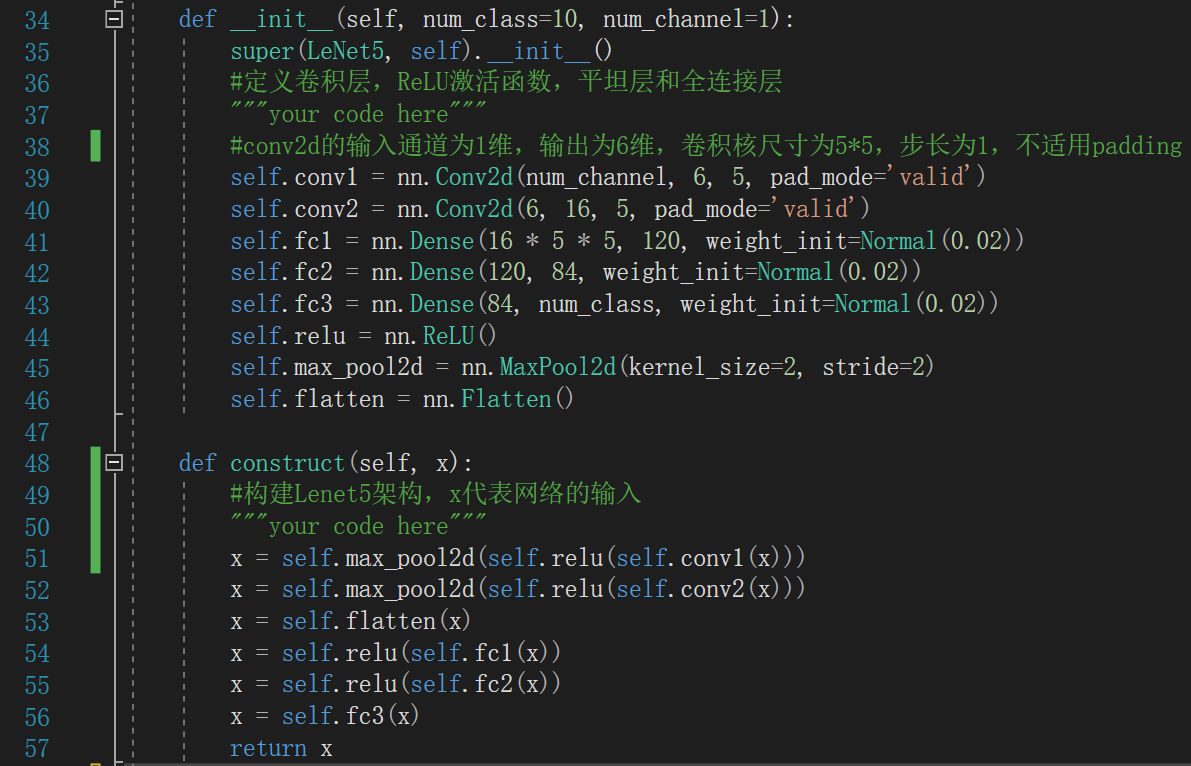


图2. 神经网络lenet网络各层的具体定义

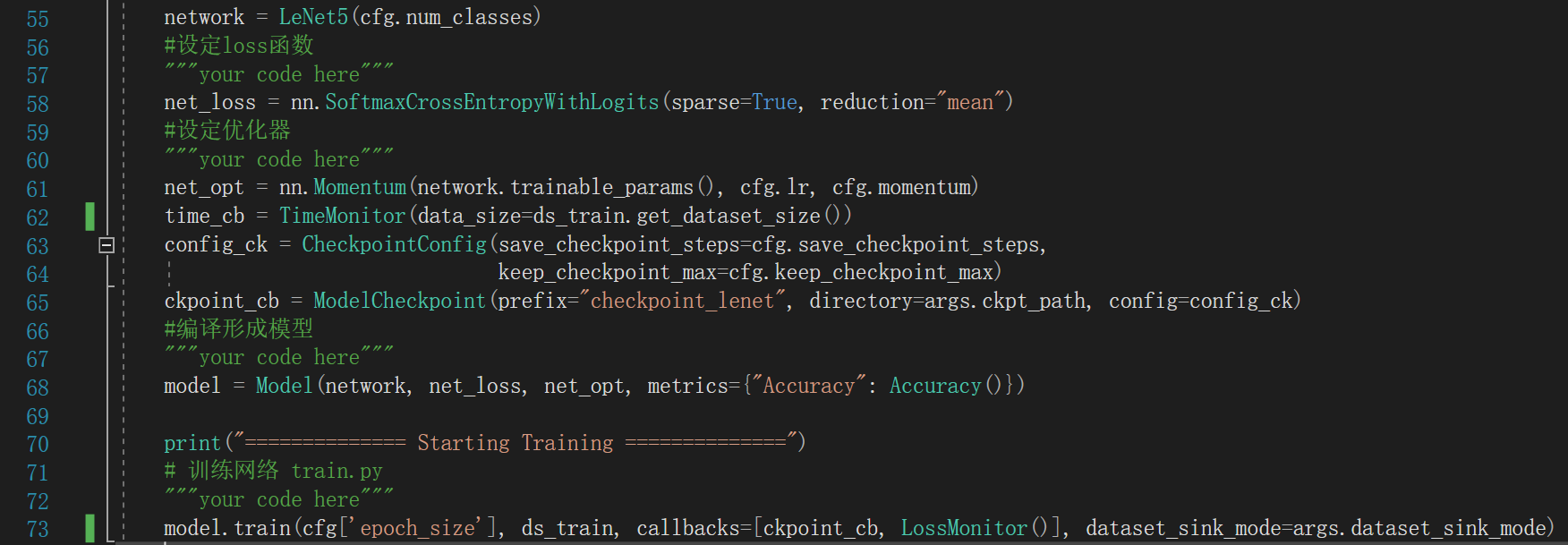


图3. 设定损失函数、优化器，编译形成模型已经训练网络

当所有需填入的函数其各个内容均已经填充完毕之后可以开始进行训练（train）和评估测试（evaluate）了：

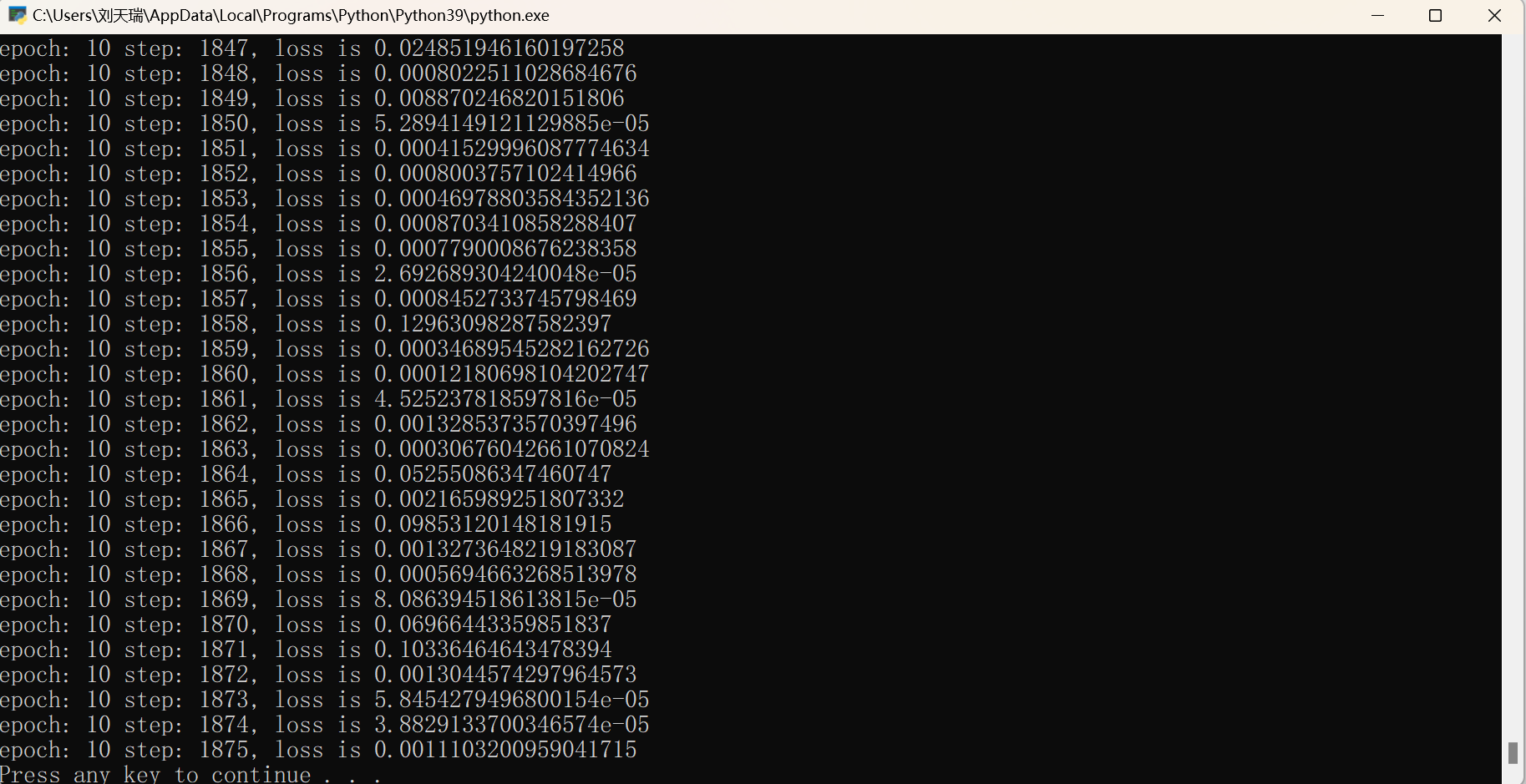


图4. 进行模型的训练

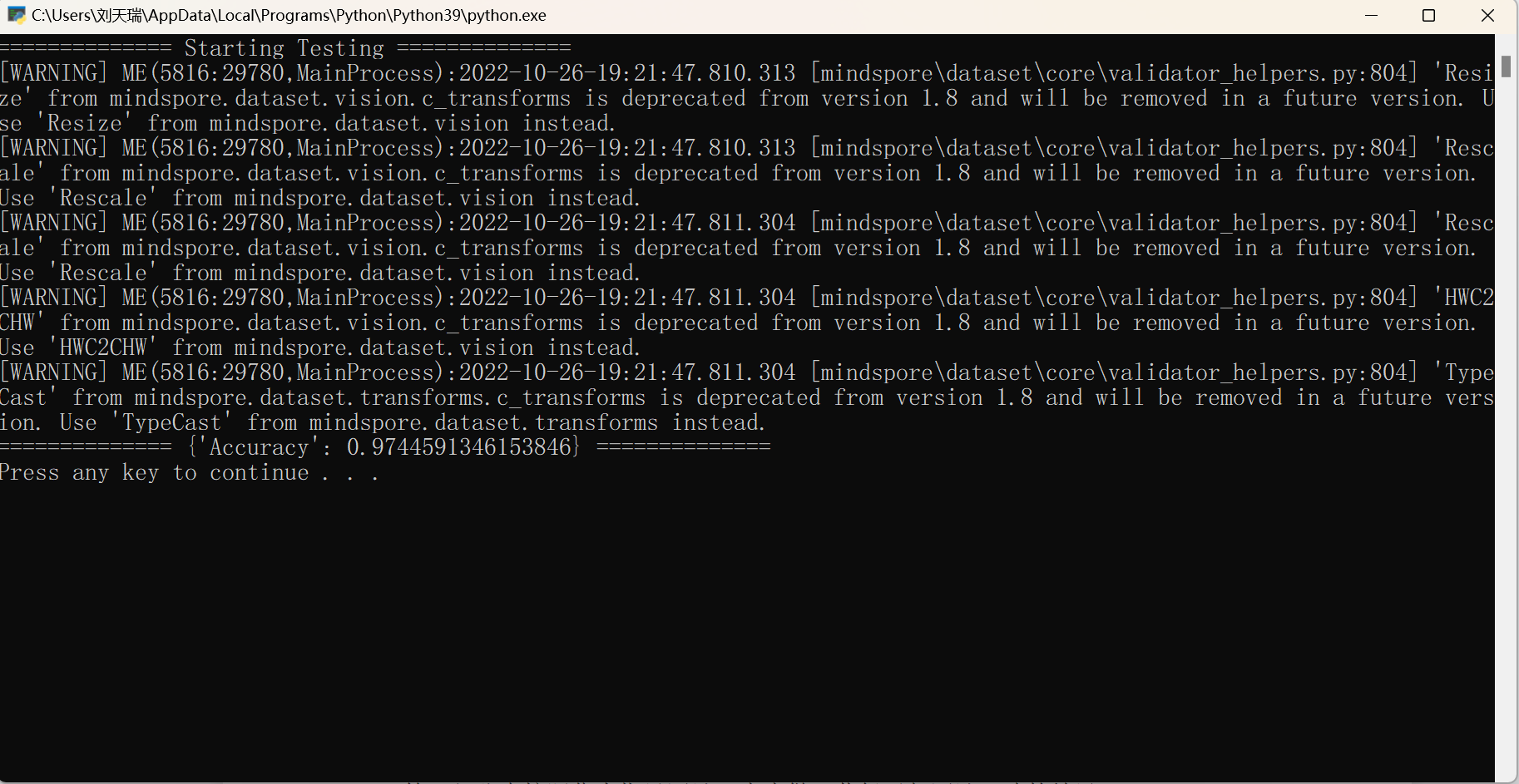


图5. 进行模型的评估测试

实验进行的第二部分也按照华为官方的步骤指导网站里的阐述依次做下去，最终可以获得正确的预测成功结果。



图6. 创建了桶



图7. OBS文件目录





图8. 已生成目标文件

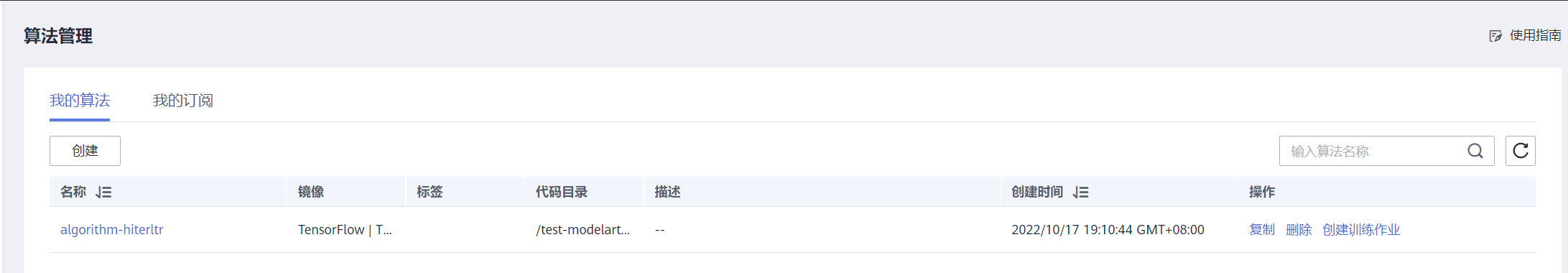
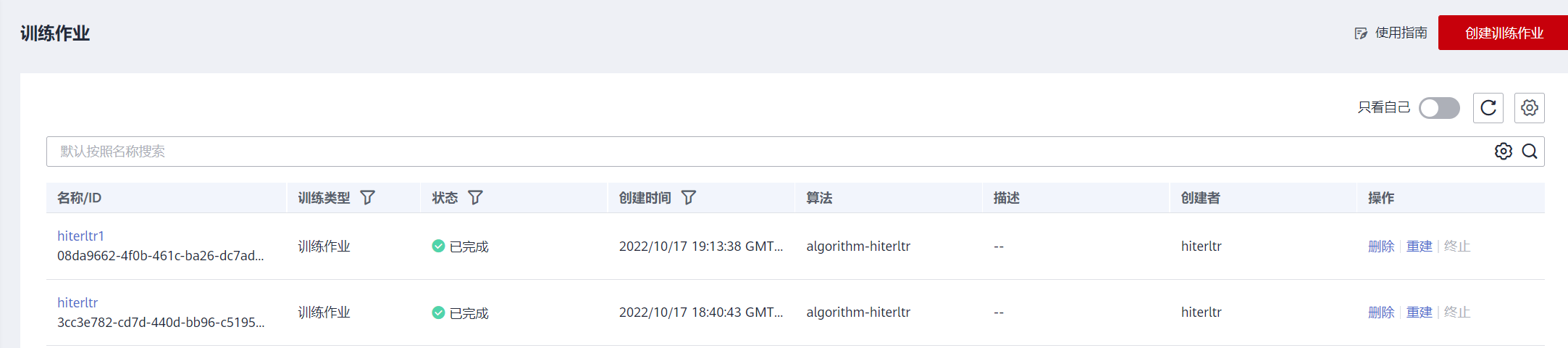


图9. 创建了算法



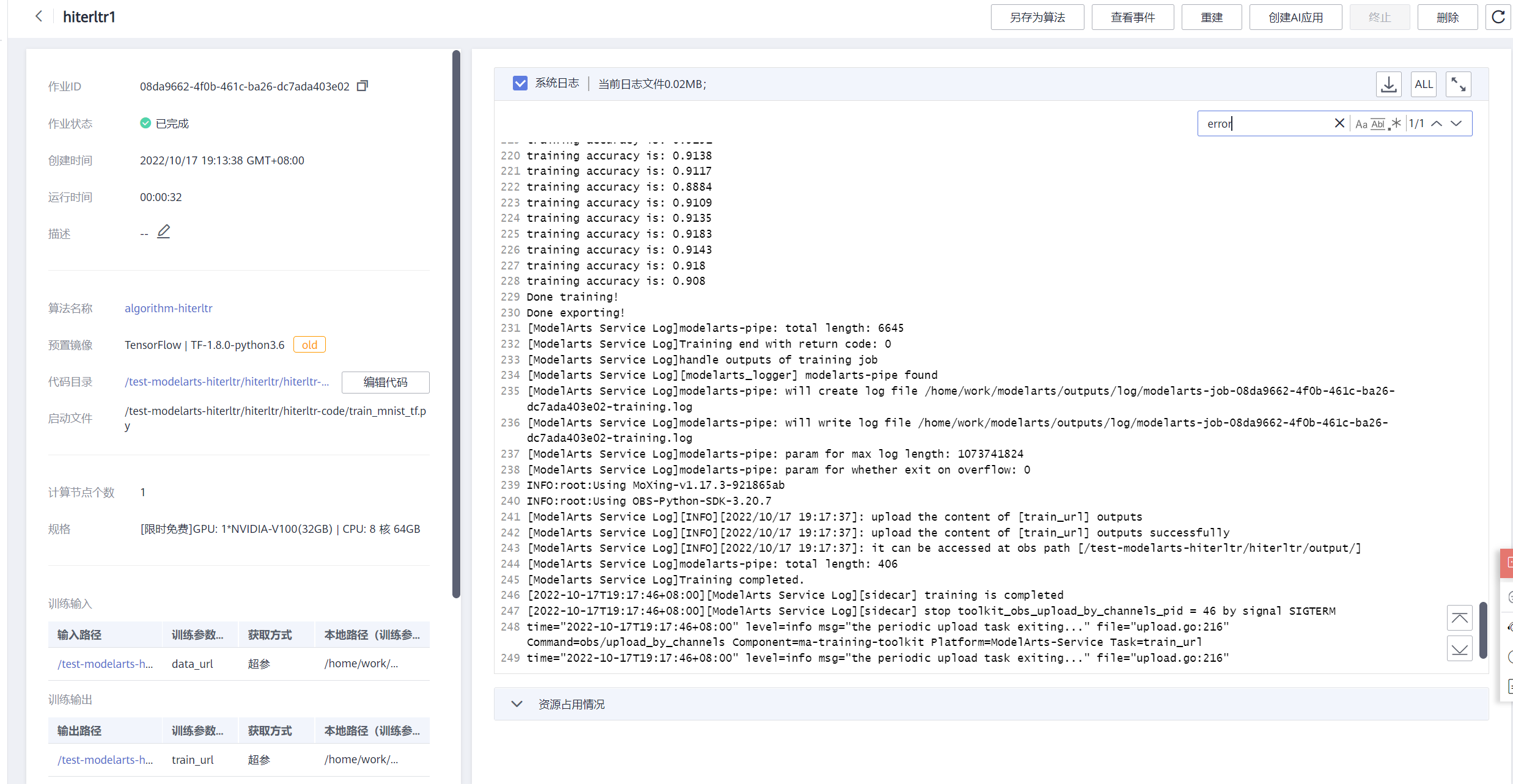


图10. 训练作业成功





图11. 模型AI应用成功

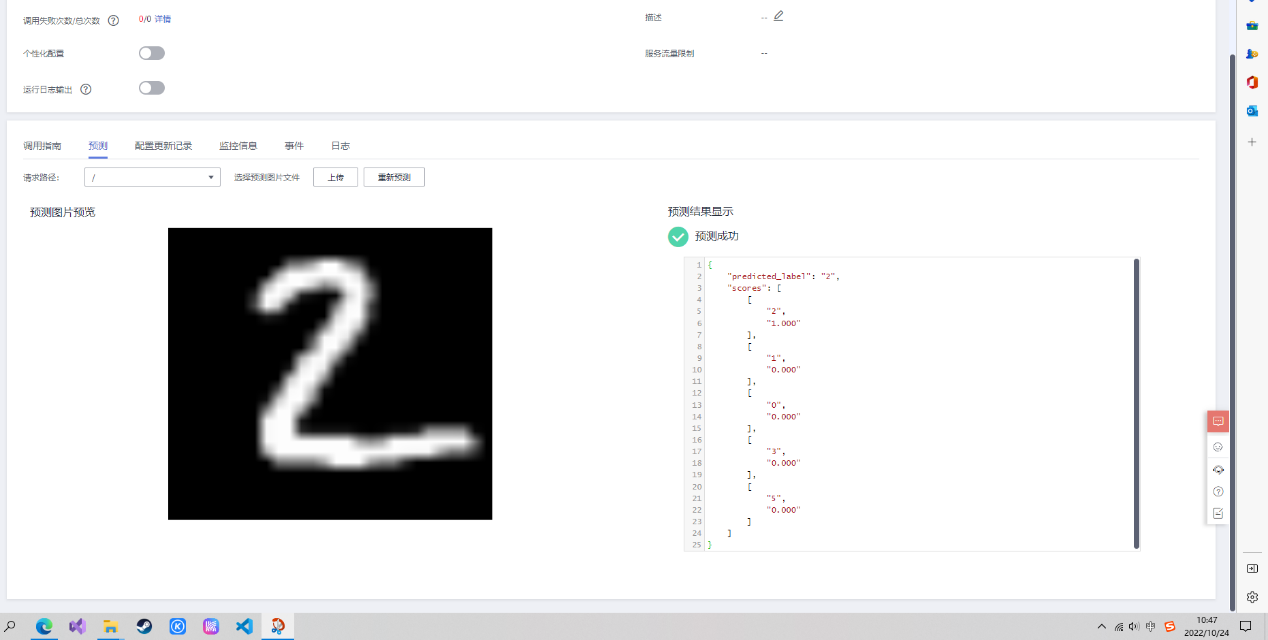


图12. ModelArts实验结果

1. **讨论及结论**

4.1 讨论

本案例中使用的MNIST数据集是比较简单的用做demo的数据集，配套算法也是比较简单的常用于教学的神经网络算法。这样的数据和算法生成的模型仅适用于教学模式，并不能应对复杂的预测场景。即生成的模型对预测图片有一定范围和要求，就像研究测试后初步粗略发现该华为云平台训练出的模型似乎只支持28\*28像素的手写识别预测，其他手写图片的最终识别结果是不准确的，这说明其余图片可能并不适配。并且图片必须和训练集中的图片相似到一定程度才有可能预测准确。本次实验也主要是为了给我们开拓视野，所以我觉得依照实验要求每个步骤严谨遵循实行就会较为顺利地得到理想结果。

**参考文献**

[1] [使用自定义算法构建模型（手写数字识别）\_AI开发平台ModelArts\_最佳实践\_模型训练（自定义算法-新版训练）\_华为云 (huaweicloud.com)](https://support.huaweicloud.com/bestpractice-modelarts/modelarts_10_0080.html) https://support.huaweicloud.com/bestpractice-modelarts/modelarts\_10\_0080.html

注：报告中图的下方要有图题（如图1. XXXX），表格需要用三线表，表头需列在表的上方（如表1. XXXX）。图表居中排列。