

华为“智能基座”系列课程

《深度学习》 前馈神经网络

版本：2.2




华为技术有限公司

版权所有 © 华为技术有限公司 2021。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

 HUAWEI 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <http://e.huawei.com>

目录

1 实验介绍	5
1.1 实验目的	5
1.2 实验清单	5
1.3 实验开发环境	5
1.4 开发平台介绍	6
1.5 背景知识	6
2 手写体图像识别实验	8
2.1 实验介绍	8
2.1.1 简介	8
2.1.2 实验目的	8
2.2 实验环境要求	8
2.3 实验总体设计	8
2.4 实验过程	9
2.4.1 创建实验环境	9
2.4.2 导入实验所需模块	10
2.4.3 导入实验数据集	11
2.4.4 模型搭建与训练	13
2.4.5 模型评估	14
2.4.6 停止实验环境	14
2.5 实验总结	15
3 FashionMnist 图像分类实验	16
3.1 实验介绍	16
3.1.1 简介	16
3.1.2 实验目的	16
3.2 实验环境要求	16
3.3 实验总体设计	16
3.4 实验过程	17
3.4.1 数据集准备	17
3.4.2 创建实验环境	17
3.4.3 导入实验所需模块	18
3.4.4 变量定义	18

3.4.5 读取并处理数据	19
3.4.6 定义前馈神经网络	22
3.4.7 训练	22
3.4.8 评估测试	23
3.4.9 对预测结果可视化	24
3.4.10 停止实验环境	26
3.5 实验总结	26
4 附录：ModelArts 开发环境搭建	28

1

实验介绍

前馈神经网络是一种最简单的神经网络，各神经元分层排列。每个神经元只与前一层的神经元相连。接收前一层的输出，并输出给下一层，各层间没有反馈。是目前应用最广泛、发展最迅速的人工神经网络之一。

本章实验主要专注于理解全连接网络（Fully Connected Neural Network）原理，并利用网络实现分类与回归任务。同时，对网络中应用到的优化器进行深入学习。

本章实验主要分为两个难度：初级和中级。

初级实验：①手写体图像识别实验；②FashionMnist 图像分类实验；③汽车里程数预测实验。

中级实验：①鸢尾花分类任务对比实验。

1.1 实验目的

通过实验了解全连接神经网络的结构，应用全连接网络处理分类和回归任务。

1.2 实验清单

表 1-1

实验	简述	难度	软件环境	开发环境
手写体图像识别实验	基于MindSpore框架，在ModelArts平台进行Mnist手写体图像识别实验。	初级	MindSpore-1.5-python3.7	ModelArts
FashionMnist 图像分类实验	基于MindSpore框架，在本地进行FashionMnist图像分类实验。	初级	MindSpore-1.5-python3.7	ModelArts

1.3 实验开发环境

- Mindspore

若选择在华为云 ModelArts 上快速搭建开发环境，可参考文末附录：ModelArts 开发环境搭建。

1.4 开发平台介绍

- MindSpore 最佳匹配昇腾芯片的开源 AI 计算框架，支持 Ascend、GPU、CPU 平台。
MindSpore 官网：<https://www.mindspore.cn>
- ModelArts 是面向开发者的一站式 AI 开发平台，为机器学习与深度学习提供海量数据预处理及半自动化标注、大规模分布式 Training、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。

具体内容请参考平台介绍 ppt。

1.5 背景知识

前馈神经网络（Feedforward neural network，FNN），简称前馈网络，是人工神经网络的一种。前馈神经网络采用一种单向多层结构，如下图所示。图中每一个圆圈代表一个神经元，每一层包含若干个神经元。在此种神经网络中，各神经元可以接收前一层神经元的信号，并产生输出到下一层。第 0 层叫输入层，最后一层叫输出层，其他中间层叫做隐含层（或隐藏层、隐层）。隐层可以是一层，也可以是多层。

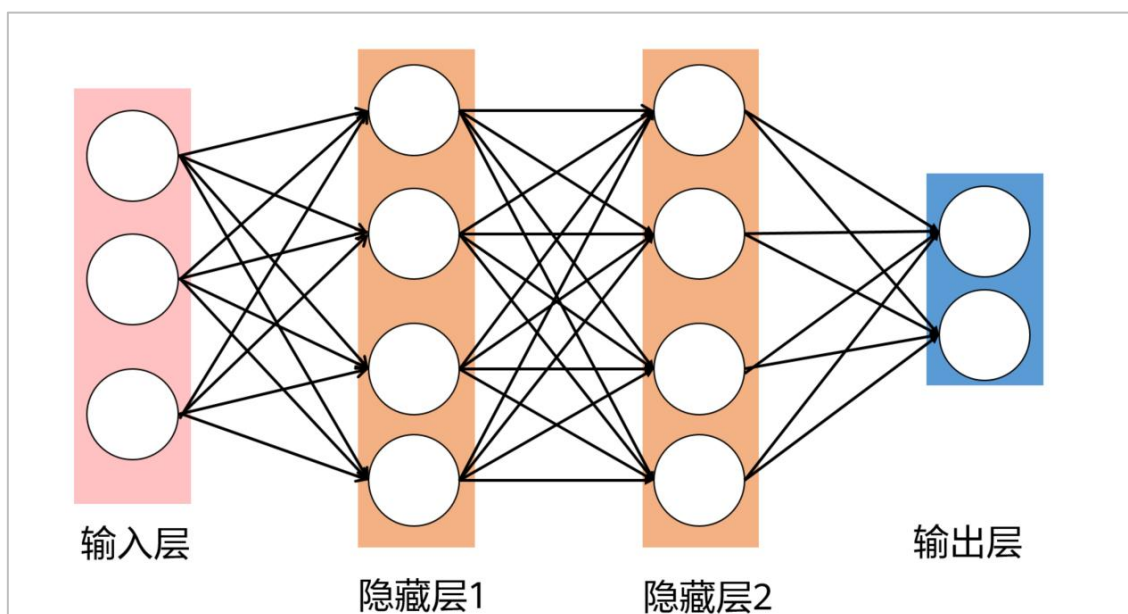


图 1-2 前馈网络结构图

网络中的每一个神经元都完成了一次计算工作，如下图所示。接受高维的输入($x_0, x_1, x_2, \dots, x_n$)与同维度的参数($w_0, w_1, w_2, \dots, w_n$)对应相乘，累加后，经过激活函数得到该神经元的输出，传递至下一层神经元。

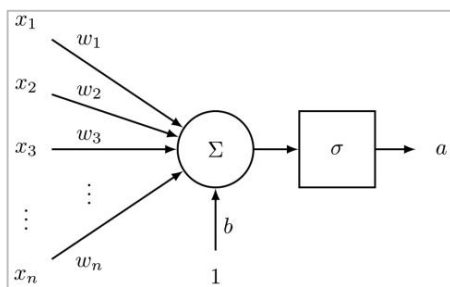


图 1-3 神经元结构图

神经网络中输入数据的信息是沿着正向传播的,而预测结果和标签的误差则是通过反向传播的,在完成反向传播后就可以对所有参数进行更新。这个阶段也就是训练神经网络的过程。网络在建立时往往会随机生成所有参数。可想而知,此时的参数并不是理想参数,需要通过调整参数来达到我们想要的理想效果。这就需要训练,或者说反向传播更新参数来达到。

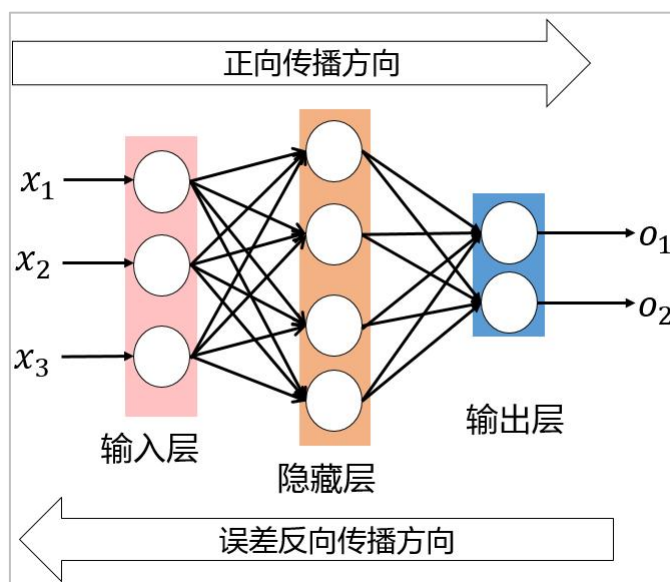


图 1-4 反向传播示例

当第一个样本输入刚刚初始化的网络,会得到一个输出值 o_d ,而对于该样本,我们知道他本该得到输出值 t_d ,因此,通过下面的损失函数,我们可以得出这个样本的损失值。当然,这只是一种损失函数:二次代价函数。

$$E(W) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

然后,就可以利用梯度下降法对网络参数进行更新。梯度下降法的思想是让损失函数沿着负梯度的方向进行搜索,不断迭代更新参数,最终使得损失值最小化。此处,会用到求偏导数和链式法则。

2 手写体图像识别实验

2.1 实验介绍

2.1.1 简介

Mnist 手写体图像识别实验是深度学习入门经典实验。Mnist 数据集包含 60,000 个用于训练的示例和 10,000 个用于测试的示例。这些数字已经过尺寸标准化并位于图像中心，图像是固定大小(28x28 像素)，其值为 0 到 255，为简单起见，每个图像都被平展并转换为 784(28*28)个特征的一维 numpy 数组。

参考 mindspore/course 仓库: <https://gitee.com/mindspore/course/tree/master/lenet5>

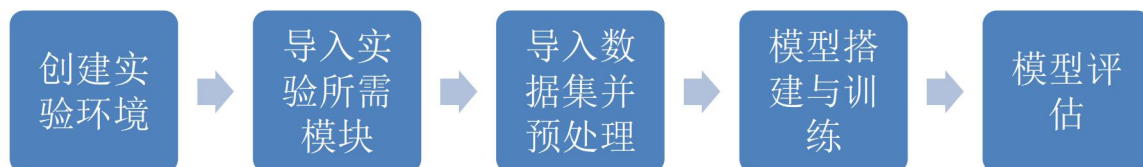
2.1.2 实验目的

- 学会如何搭建全连接神经网络。
- 掌握搭建网络过程中的关键点。
- 掌握分类任务的整体流程。

2.2 实验环境要求

推荐在华为云 ModelArts 实验平台完成实验，也可在本地搭建 python3.7.5 和 MindSpore1.5 环境完成实验。

2.3 实验总体设计



创建实验环境：在 ModelArts 平台创建 Ascend+MindSpore 环境。

导入实验所需模块：该步骤通常都是程序编辑的第一步，将实验代码所需要用到的模块包用 import 命令进行导入。

导入数据集并预处理：神经网络的训练离不开数据，这里对数据进行导入。同时，因为全连接网络只能接收固定维度的输入数据，所以，要对数据集进行预处理，以符合网络的输入维度要求。同时，设定好每一次训练的 Batch 的大小，以 Batch Size 为单位进行输入。

模型搭建：利用 mindspore.nn 的 cell 模块搭建全连接网络，包含输入层，隐藏层，输出层。同时，配置好网络需要的优化器，损失函数和评价指标。传入数据，并开始训练模型。

模型评估：利用测试集进行模型的评估。

2.4 实验过程

2.4.1 创建实验环境

在开始本实验前，需要完成实验环境搭建工作。

步骤 1 进入 ModelArts 开发环境

参考文末附录，创建 ModelArts 上的开发环境 Notebook 并进入。

步骤 2 上传数据集

实验中需要用到 MNIST 手写体字符的数据集：<http://yann.lecun.com/exdb/mnist/>

创建 MNIST 文件夹，在 MNIST 文件夹下设置 train 和 test 文件夹，将下载的训练集和测试集数据解压后分别置于 train 和 test 文件夹下：

- 训练集 train: train-images-idx3-ubyte, train-labels-idx1-ubyte
- 测试集 test: t10k-images-idx3-ubyte, t10k-labels-idx1-ubyte

如下图所示：

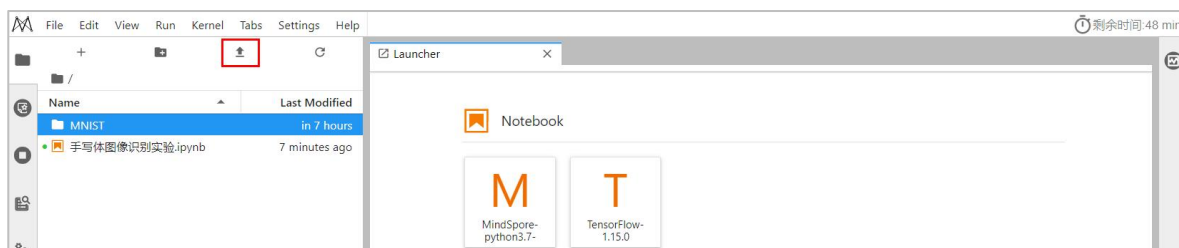


图 2-1 数据上传

步骤 3 打开 Notebook

打开 Notebook 控制台后，新建或打开 ipynb 文件，选择 MindSpore 环境作为 Kernel，即可开始编辑实验代码。

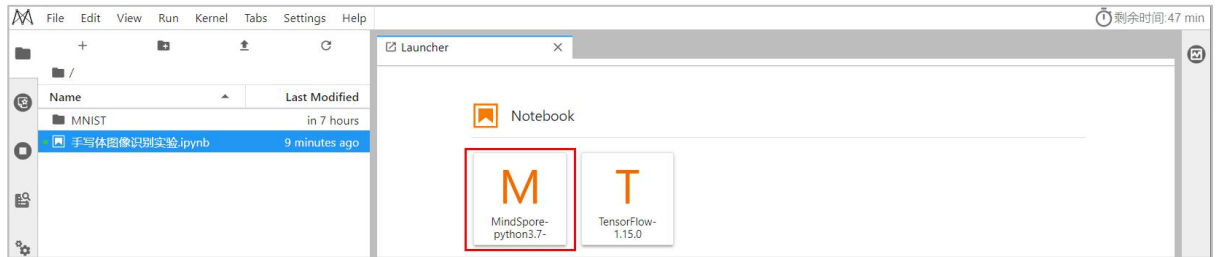


图 2-2 创建 ipynb 文件

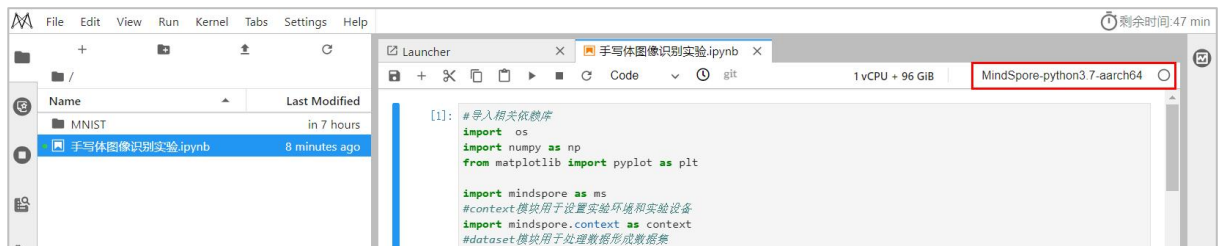


图 2-3 打开 ipynb 文件

2.4.2 导入实验所需模块

os 模块主要用于对系统路径和文件进行处理。Numpy 模块主要用于数据的基本运算操作。Matplotlib 模块主要用于画图。MindSpore 相关模块主要用于搭建网络、调用优化器、读取数据集和将数据集处理成网络的标准输入格式。

```
# 导入相关依赖库
import os
import numpy as np
from matplotlib import pyplot as plt

import mindspore as ms
# context 模块用于设置实验环境和实验设备
import mindspore.context as context
# dataset 模块用于处理数据形成数据集
import mindspore.dataset as ds
# c_transforms 模块用于转换数据类型
import mindspore.dataset.transforms.c_transforms as C
# vision.c_transforms 模块用于转换图像, 这是一个基于 opencv 的高级 API
import mindspore.dataset.vision.c_transforms as CV
# 导入 Accuracy 作为评价指标
from mindspore.nn.metrics import Accuracy
# nn 中有各种神经网络层如: Dense, ReLU
from mindspore import nn
# Model 用于创建模型对象, 完成网络搭建和编译, 并用于训练和评估
from mindspore.train import Model
# LossMonitor 可以在训练过程中返回 LOSS 值作为监控指标
from mindspore.train.callback import LossMonitor
# 设定运行模式为动态图模式, 并且运行设备为昇腾芯片
```

```
context.set_context(mode=context.GRAPH_MODE, device_target='Ascend')
```

2.4.3 导入实验数据集

MNIST 是一个手写数字数据集，训练集包含 60000 张手写数字，测试集包含 10000 张手写数字，共 10 类。可在 MNIST 数据集的官网下载数据集，解压到当前代码目录下。MindSpore 的 dataset 模块有专门用于读取和解析 MNIST 数据集的源数据集，可直接读取并生成训练集和测试集。

步骤 1 加载并查看数据集

```
# MindSpore 内置方法读取 MNIST 数据集
ds_train = ds.MnistDataset(os.path.join(r'./MNIST', "train"))
ds_test = ds.MnistDataset(os.path.join(r'./MNIST', "test"))

print('训练数据集数量: ', ds_train.get_dataset_size())
print('测试数据集数量: ', ds_test.get_dataset_size())
# 该数据集可以通过 create_dict_iterator() 转换为迭代器形式，然后通过 __next__() 一个个输出样本
image = ds_train.create_dict_iterator().__next__()
print(type(image))
print('图像长/宽/通道数: ', image['image'].shape)
# 一共 10 类，用 0-9 的数字表达类别。
print('一张图像的标签样式: ', image['label'])
```

输出结果：

```
训练数据集数量: 60000
测试数据集数量: 10000
<class 'dict'>
图像长/宽/通道数: (28, 28, 1)
一张图像的标签样式: 7
```

步骤 2 生成测试集和训练集

创建数据集，为训练集设定 Batch Size，这是因为我们通常会采用小批量梯度下降法（MBGD）来训练网络，所以 Batch Size 作为一个非常重要的超参数需要提前设定好。在本代码中，Batch Size 为 128，意味着每一次更新参数，我们都用 128 个样本的平均损失值来进行更新。

```
DATA_DIR_TRAIN = "./MNIST/train" # 训练集信息
DATA_DIR_TEST = "./MNIST/test" # 测试集信息

def create_dataset(training=True, batch_size=128, resize=(28, 28), rescale=1/255, shift=-0.5,
buffer_size=64):
    ds = ms.dataset.MnistDataset(DATA_DIR_TRAIN if training else DATA_DIR_TEST)

    # 定义改变形状、归一化和更改图片维度的操作。
    # 改为 (28, 28) 的形状
    resize_op = CV.Resize(resize)
```

```
# rescale 方法可以对数据集进行归一化和标准化操作，这里就是将像素值归一到 0 和 1 之间，shift 参数
# 可以让值域偏移至-0.5 和 0.5 之间
rescale_op = CV.Rescale(rescale, shift)
# 由高度、宽度、深度改为深度、高度、宽度
hwc2chw_op = CV.HWC2CHW()

# 利用 map 操作对原数据集进行调整
ds = ds.map(input_columns="image", operations=[resize_op, rescale_op, hwc2chw_op])
ds = ds.map(input_columns="label", operations=C.TypeCast(ms.int32))
# 设定洗牌缓冲区的大小，从一定程度上控制打乱操作的混乱程度
ds = ds.shuffle(buffer_size=buffer_size)
# 设定数据集的 batch_size 大小，并丢弃剩余的样本
ds = ds.batch(batch_size, drop_remainder=True)

return ds
```

步骤 3 查看数据

```
# 显示前 10 张图片以及对应标签,检查图片是否是正确的数据集
dataset_show = create_dataset(training=False)
data = dataset_show.create_dict_iterator().__next__()
images = data['image'].asnumpy()
labels = data['label'].asnumpy()

for i in range(1,11):
    plt.subplot(2, 5, i)
    # 利用 squeeze 方法去掉多余的一个维度
    plt.imshow(np.squeeze(images[i]))
    plt.title('Number: %s' % labels[i])
    plt.xticks([])
plt.show()
```

输出结果：

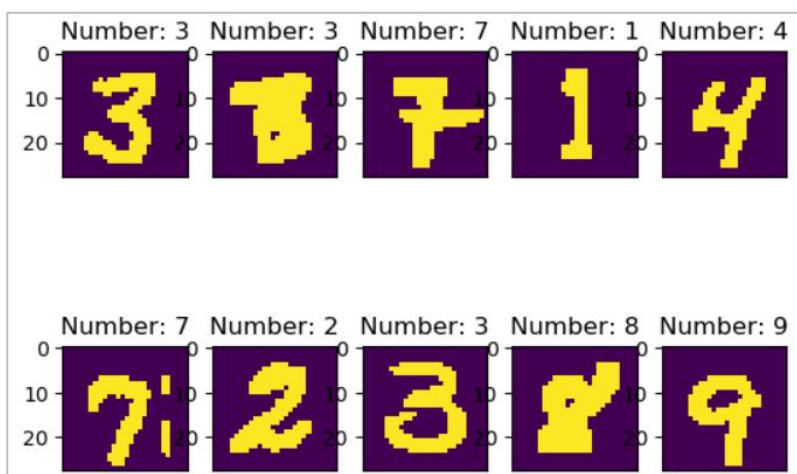


图 2-4 输出图片结果

2.4.4 模型搭建与训练

手写数字图像数据集准备完成，接下来我们就需要构建训练模型，本实验采用的是全连接神经网络算法，所以我们首先需要建立初始化的神经网络。

步骤 1 创建网络

手写数字图像数据集准备完成，接下来我们就需要构建训练模型，本实验采用的是全连接神经网络算法，所以我们首先需要建立初始化的神经网络。nn.cell 能够用来组成网络模型;模型包括 5 个全连接层和 RELU 激活函数，一个全连接输出层并使用 softmax 进行多分类，共分成 (0-9) 10 类

```
# 利用定义类的方式生成网络，Mindspore 中定义网络需要继承 nn.cell。在 init 方法中定义该网络需要的神经网络层
# 在 construct 方法中梳理神经网络层与层之间的关系。
class ForwardNN(nn.Cell):
    def __init__(self):
        super(ForwardNN, self).__init__()
        self.flatten = nn.Flatten()
        self.relu = nn.ReLU()
        self.fc1 = nn.Dense(784, 512, activation='relu')
        self.fc2 = nn.Dense(512, 256, activation='relu')
        self.fc3 = nn.Dense(256, 128, activation='relu')
        self.fc4 = nn.Dense(128, 64, activation='relu')
        self.fc5 = nn.Dense(64, 32, activation='relu')
        self.fc6 = nn.Dense(32, 10, activation='softmax')

    def construct(self, input_x):
        output = self.flatten(input_x)
        output = self.fc1(output)
        output = self.fc2(output)
        output = self.fc3(output)
        output = self.fc4(output)
        output = self.fc5(output)
        output = self.fc6(output)
        return output
```

步骤 2 设定参数

指定模型所需的损失函数、评估指标、优化器等参数。

```
lr = 0.001
num_epoch = 10
momentum = 0.9

net = ForwardNN()
# 定义 loss 函数，改函数不要求导，可以给离散的标签值，且 loss 值为均值
```

```
loss = nn.loss.SoftmaxCrossEntropyWithLogits( sparse=True, reduction='mean')
# 定义准确率为评价指标，用于评价模型
metrics={"Accuracy": Accuracy()}
# 定义优化器为 Adam 优化器，并设定学习率
opt = nn.Adam(net.trainable_params(), lr)
```

步骤 3 训练模型并保存

将创建好的网络、损失函数、评估指标、优化器等参数装入模型中对模型进行训练。

```
# 生成验证集，验证集不需要训练，所以不需要 repeat
ds_eval = create_dataset(False, batch_size=32)
# 模型编译过程，将定义好的网络、loss 函数、评价指标、优化器编译
model = Model(net, loss, opt, metrics)

# 生成训练集
ds_train = create_dataset(True, batch_size=32)
print("===== Starting Training =====")
# 训练模型，用 loss 作为监控指标，并利用昇腾芯片的数据下沉特性进行训练
model.train(num_epoch, ds_train, callbacks=[LossMonitor()], dataset_sink_mode=True)
```

输出结果：

```
===== Starting Training =====
epoch: 1 step: 1875, loss is 1.7190123
epoch: 2 step: 1875, loss is 1.5236794
epoch: 3 step: 1875, loss is 1.5861311
epoch: 4 step: 1875, loss is 1.6486512
epoch: 5 step: 1875, loss is 1.5861502
epoch: 6 step: 1875, loss is 1.5549002
epoch: 7 step: 1875, loss is 1.6174002
epoch: 8 step: 1875, loss is 1.6176393
epoch: 9 step: 1875, loss is 1.9296615
epoch: 10 step: 1875, loss is 1.5236502
```

2.4.5 模型评估

利用模型对测试集的数据进行预测，并与标签对比，用准确率 accuracy 进行评估。

```
# 使用测试集评估模型，打印总体准确率
metrics_result=model.eval(ds_eval)
print(metrics_result)
```

输出结果：

```
{'Accuracy': 0.8710666666666667}
```

2.4.6 停止实验环境

试验完成之后请及时停止实验环境，避免资源浪费，如下图：



图 2-5 停止实验环境

2.5 实验总结

本章提供了一个基于华为 MindSpore 框架的手写体图像识别实验。展示了如何构建全连接网络实现图像识别的全流程。

3

FashionMnist 图像分类实验

3.1 实验介绍

3.1.1 简介

该实验主要内容是进行 fashion mnist 数据集标签的识别，是一个替代 MNIST 手写数字集的图像数据集。其涵盖了来自 10 种类别的共 7 万个不同商品的正面图片。Fashion-MNIST 的大小、格式和训练集/测试集划分与原始的 MNIST 完全一致。60000/10000 的训练测试数据划分，28x28x1 的灰度图片。

参考 mindspore/course 仓库：<https://gitee.com/mindspore/course/tree/master/feedforward>

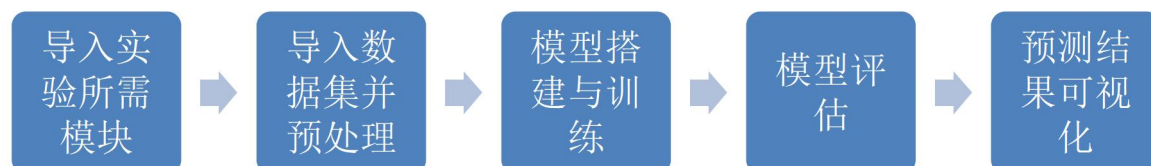
3.1.2 实验目的

- 掌握如何使用 MindSpore 进行简单前馈神经网络的开发。
- 了解如何使用 MindSpore 进行简单图片分类任务的训练。
- 了解如何使用 MindSpore 进行简单图片分类任务的测试和预测。

3.2 实验环境要求

- ModelArts 平台：MindSpore

3.3 实验总体设计



3.4 实验过程

3.4.1 数据集准备

Fashion-MNIST 是一个替代 MNIST 手写数字集的图像数据集。它是由 Zalando（一家德国的时尚科技公司）旗下的研究部门提供。其涵盖了来自 10 种类别的共 7 万个不同商品的正面图片。Fashion-MNIST 的大小、格式和训练集/测试集划分与原始的 MNIST 完全一致。60000/10000 的训练测试数据划分，28x28x1 的灰度图片。

这里介绍一下经典的 MNIST（手写字母）数据集。经典的 MNIST 数据集包含了大量的手写数字。十几年来，来自机器学习、机器视觉、人工智能、深度学习领域的研究员们把这个数据集作为衡量算法的基准之一。实际上，MNIST 数据集已经成为算法作者的必测的数据集之一，但是 MNIST 数据集太简单了。很多深度学习算法在测试集上的准确率已经达到 99.6%。

从 [Fashion-MNIST GitHub 仓库](#) 下载数据集，创建 Fashion-MNIST 文件夹，在 Fashion-MNIST 文件夹下设置 train 和 test 文件夹，将下载的训练集和测试集数据解压后分别置于 train 和 test 文件夹下：

- 训练集 train: train-images-idx3-ubyte, train-labels-idx1-ubyte;
- 测试集 test: t10k-images-idx3-ubyte, t10k-labels-idx1-ubyte;

3.4.2 创建实验环境

在开始本实验前，需要完成实验环境搭建工作。

步骤 1 进入 ModelArts 开发环境

参考文末附录，创建 ModelArts 上的开发环境 Notebook 并进入。

步骤 2 上传数据集

将 Fashion-MNIST 数据集的压缩包文件（zip 格式）上传至环境项目内，新建终端解压缩，输入如下命令进行解压缩，如下图所示：

```
cd work
unzip Fashion-MNIST.zip
```

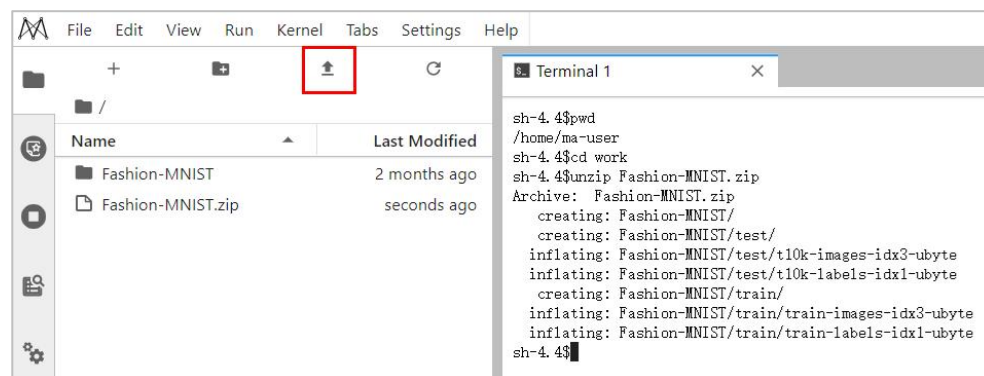


图 3-1 数据上传，解压缩

步骤 3 打开 Notebook

打开 Notebook 控制台后，新建或打开 ipynb 文件，选择 MindSpore 环境作为 Kernel，即可开始编辑实验代码。

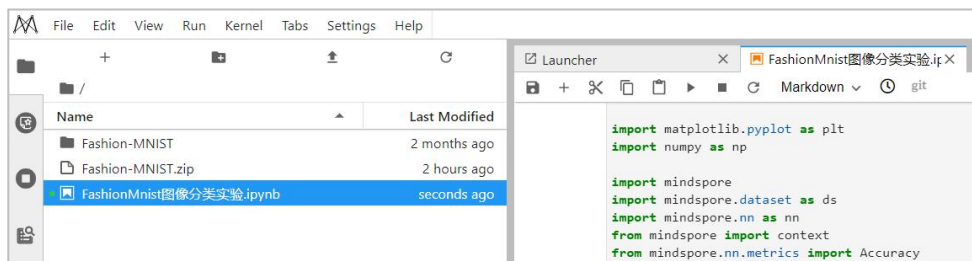


图 3-2 打开 ipynb 文件

3.4.3 导入实验所需模块

用到的框架主要包括：

- mindspore，用于神经网络的搭建；
- numpy，用于处理一些数据；
- matplotlib，用于画图、图像展示；
- struct，用于处理二进制文件。

```
import os
import struct
from easydict import EasyDict as edict

import matplotlib.pyplot as plt
import numpy as np

import mindspore
import mindspore.dataset as ds
import mindspore.nn as nn
from mindspore import context
from mindspore.nn.metrics import Accuracy
from mindspore.train import Model
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor
from mindspore import Tensor

context.set_context(mode=context.GRAPH_MODE, device_target='Ascend')
```

3.4.4 变量定义

```
cfg = edict({
    'train_size': 60000, # 训练集大小
    'test_size': 10000, # 测试集大小
```

```
'channel': 1, # 图片通道数
'image_height': 28, # 图片高度
'image_width': 28, # 图片宽度
'batch_size': 60,
'num_classes': 10, # 分类类别
'lr': 0.001, # 学习率
'epoch_size': 20, # 训练次数
#此处应该改为自己数据集存储的路径，分别用 train 和 test 两个文件夹存储训练数据集和测试数据集
'data_dir_train': os.path.join('./Fashion-MNIST/train/'),
'data_dir_test': os.path.join('./Fashion-MNIST/test/'),
'save_checkpoint_steps': 1, # 多少步保存一次模型
'keep_checkpoint_max': 3, # 最多保存多少个模型
'output_directory': './model_fashion', # 保存模型路径
'output_prefix': "checkpoint_fashion_forward" # 保存模型文件名字
})
```

3.4.5 读取并处理数据

步骤 1 读取数据

```
def read_image(file_name):
    """
    :param file_name: 文件路径
    :return: 训练或者测试数据
    如下是训练的图片的二进制格式
    [offset] [type]          [value]          [description]
    0000     32 bit integer  0x00000803(2051) magic number
    0004     32 bit integer  60000          number of images
    0008     32 bit integer  28             number of rows
    0012     32 bit integer  28             number of columns
    0016     unsigned byte   ??             pixel
    0017     unsigned byte   ??             pixel
    .....
    xxxx     unsigned byte   ??             pixel
    """
    file_handle = open(file_name, "rb") # 以二进制打开文档
    file_content = file_handle.read() # 读取到缓冲区中
    head = struct.unpack_from('>IIII', file_content, 0) # 取前 4 个整数，返回一个元组
    offset = struct.calcsize('>IIII')
    imgNum = head[1] # 图片数
    width = head[2] # 宽度
    height = head[3] # 高度
    bits = imgNum * width * height # data 一共有 60000*28*28 个像素值
    bitsString = '>' + str(bits) + 'B' # fmt 格式: '>47040000B'
    imgs = struct.unpack_from(bitsString, file_content, offset) # 取 data 数据，返回一个元组
```

```

    imgs_array = np.array(imgs).reshape((imgNum, width * height)) # 最后将读取的数据 reshape 成
    【图片数，图片像素】二维数组
    return imgs_array

def read_label(file_name):
    """
    :param file_name:
    :return:
    标签的格式如下：
    [offset] [type]          [value]          [description]
    0000      32 bit integer  0x00000801 (2049) magic number (MSB first)
    0004      32 bit integer  60000          number of items
    0008      unsigned byte  ??              label
    0009      unsigned byte  ??              label
    .....
    xxxx      unsigned byte  ??              label
    The labels values are 0 to 9.
    """
    file_handle = open(file_name, "rb") # 以二进制打开文档
    file_content = file_handle.read() # 读取到缓冲区中
    head = struct.unpack_from('>II', file_content, 0) # 取前 2 个整数，返回一个元组
    offset = struct.calcsize('>II')
    labelNum = head[1] # label 数
    bitsString = '>' + str(labelNum) + 'B' # fmt 格式: '>47040000B'
    label = struct.unpack_from(bitsString, file_content, offset) # 取 data 数据，返回一个元组
    return np.array(label)

def get_data():
    # 文件获取
    train_image = os.path.join(cfg.data_dir_train, 'train-images-idx3-ubyte')
    test_image = os.path.join(cfg.data_dir_test, "t10k-images-idx3-ubyte")
    train_label = os.path.join(cfg.data_dir_train, "train-labels-idx1-ubyte")
    test_label = os.path.join(cfg.data_dir_test, "t10k-labels-idx1-ubyte")
    # 读取数据
    train_x = read_image(train_image)
    test_x = read_image(test_image)
    train_y = read_label(train_label)
    test_y = read_label(test_label)
    return train_x, train_y, test_x, test_y

```

步骤 2 数据预处理和处理结果图片展示

查看当前目录：

```
print(os.getcwd())
```

输出结果：

```
/home/ma-user/work
```

数据预处理和图片展示：

```
train_x, train_y, test_x, test_y = get_data()
#第一维度是 batchsize 的数据,第二维度是图像的 channel 数, 第三, 第四维度是高和宽
train_x = train_x.reshape(-1, 1, cfg.image_height, cfg.image_width)
test_x = test_x.reshape(-1, 1, cfg.image_height, cfg.image_width)
#归一化至 0 和 1 之间
train_x = train_x / 255.0
test_x = test_x / 255.0
#修改数据格式
train_x = train_x.astype('float32')
test_x = test_x.astype('float32')
train_y = train_y.astype('int32')
test_y = test_y.astype('int32')
print('训练数据集样本数: ', train_x.shape[0])
print('测试数据集样本数: ', test_y.shape[0])
print('通道数/图像长/宽: ', train_x.shape[1:])
print('一张图像的标签样式: ', train_y[0]) # 一共 10 类, 用 0-9 的数字表达类别。

plt.figure()
plt.imshow(train_x[0,0,...])
plt.colorbar()
plt.grid(False)
plt.show()
```

输出结果：

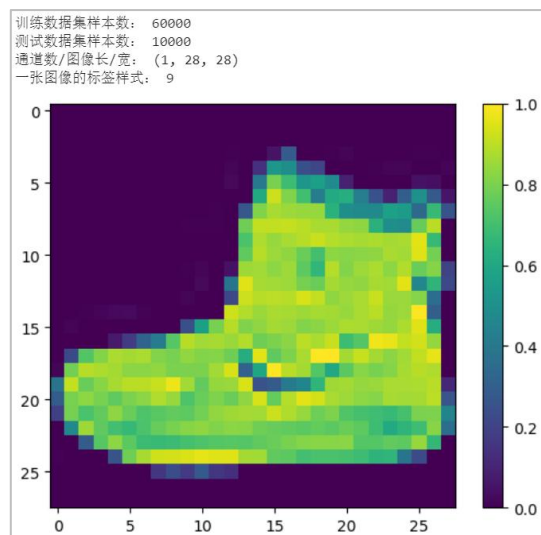


图 3-3 输出结果

步骤 3 使用 MindSpore GeneratorDataset 接口将 numpy.ndarray 类型的数据转换为 Dataset

```
# 转换数据类型为 Dataset
XY_train = list(zip(train_x, train_y))
#转换数据和标签为 dataset 类型，并制定数据为 x，标签为 y
ds_train = ds.GeneratorDataset(XY_train, ['x', 'y'])
ds_train = ds_train.shuffle(buffer_size=cfg.train_size).batch(cfg.batch_size, drop_remainder=True)
XY_test = list(zip(test_x, test_y))
ds_test = ds.GeneratorDataset(XY_test, ['x', 'y'])
ds_test = ds_test.shuffle(buffer_size=cfg.test_size).batch(cfg.batch_size, drop_remainder=True)
```

3.4.6 定义前馈神经网络

前馈神经网络是一种最简单的神经网络，各神经元分层排列（其中每一层包含若干个神经元）。每个神经元只与前一层的神经元相连，接收前一层的输出，并输出给下一层，各层间没有反馈。是目前应用最广泛、发展最迅速的人工神经网络之一。第 0 层叫输入层，最后一层叫输出层，其他中间层叫做隐含层（或隐藏层、隐层）。隐层可以是一层，也可以是多层，是由全连接层堆叠而成。

```
# 定义前馈神经网络
class Forward_fashion(nn.Cell):
    def __init__(self, num_class=10): # 一共分十类，图片通道数是 1
        super(Forward_fashion, self).__init__()
        self.num_class = num_class
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(cfg.channel * cfg.image_height * cfg.image_width, 128)
        self.relu = nn.ReLU()
        self.fc2 = nn.Dense(128, self.num_class)

    def construct(self, x):
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

3.4.7 训练

使用 Fashion-MNIST 数据集对上述定义的前馈神经网络模型进行训练。训练策略如下表所示，可以调整训练策略并查看训练效果。

batch size	number of epochs	learning rate	input shape	optimizer
60	20	0.001	(1, 28, 28)	Adam

```
# 构建网络
network = Forward_fashion(cfg.num_classes)
# 定义模型的损失函数，优化器
```

```
net_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
net_opt = nn.Adam(network.trainable_params(), cfg.lr)
# 训练模型
model = Model(network, loss_fn=net_loss, optimizer=net_opt, metrics={"acc"})
loss_cb = LossMonitor(per_print_times=int(cfg.train_size / cfg.batch_size))
# 设定每多少 step 保存一个 checkpoint, 并且设定最多保存多少个 checkpoint
config_ck = CheckpointConfig(save_checkpoint_steps=cfg.save_checkpoint_steps,
                              keep_checkpoint_max=cfg.keep_checkpoint_max)
ckpt_cb = ModelCheckpoint(prefix=cfg.output_prefix, directory=cfg.output_directory, config=config_ck)
print("===== Starting Training =====")
model.train(cfg.epoch_size, ds_train, callbacks=[ckpt_cb, loss_cb], dataset_sink_mode=False)
```

输出结果:

```
===== Starting Training =====
epoch: 1 step: 1000, loss is 0.47615242
epoch: 2 step: 1000, loss is 0.6278242
epoch: 3 step: 1000, loss is 0.4200313
epoch: 4 step: 1000, loss is 0.342624
epoch: 5 step: 1000, loss is 0.20733497
epoch: 6 step: 1000, loss is 0.37861693
epoch: 7 step: 1000, loss is 0.35047245
epoch: 8 step: 1000, loss is 0.27327073
epoch: 9 step: 1000, loss is 0.23978877
epoch: 10 step: 1000, loss is 0.31236422
epoch: 11 step: 1000, loss is 0.19281301
epoch: 12 step: 1000, loss is 0.2834559
epoch: 13 step: 1000, loss is 0.1653405
epoch: 14 step: 1000, loss is 0.27971604
epoch: 15 step: 1000, loss is 0.24855426
epoch: 16 step: 1000, loss is 0.23632649
epoch: 17 step: 1000, loss is 0.18741903
epoch: 18 step: 1000, loss is 0.13822958
epoch: 19 step: 1000, loss is 0.095432095
epoch: 20 step: 1000, loss is 0.38744873
```

3.4.8 评估测试

```
# 使用测试集评估模型, 打印总体准确率
metric = model.eval(ds_test)
print(metric)
```

输出结果:

```
{'acc': 0.8877510040160642}
```

实例预测:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
#从测试集中取出一组样本，输入模型进行预测
test_ = ds_test.create_dict_iterator().__next__()
#利用 key 值选出样本
test = Tensor(test_['x'], mindspore.float32)
predictions = model.predict(test)
softmax = nn.Softmax()
predictions = softmax(predictions)
predictions = predictions.asnumpy()
true_label = test_['y'].asnumpy()

for i in range(15):
    p_np = predictions[i, :]
    pre_label = np.argmax(p_np)
    print('第' + str(i) + '个 sample 预测结果: ', class_names[pre_label], ' 真实结果: ',
          class_names[true_label[i]])
```

输出结果：

```
第 0 个 sample 预测结果: Coat    真实结果: Coat
第 1 个 sample 预测结果: Shirt   真实结果: Shirt
第 2 个 sample 预测结果: Bag     真实结果: Bag
第 3 个 sample 预测结果: Trouser  真实结果: Trouser
第 4 个 sample 预测结果: Ankle boot  真实结果: Ankle boot
第 5 个 sample 预测结果: Dress    真实结果: Dress
第 6 个 sample 预测结果: Sneaker   真实结果: Sneaker
第 7 个 sample 预测结果: Bag      真实结果: Bag
第 8 个 sample 预测结果: Coat     真实结果: Coat
第 9 个 sample 预测结果: Dress    真实结果: Dress
第 10 个 sample 预测结果: Sneaker  真实结果: Sneaker
第 11 个 sample 预测结果: T-shirt/top  真实结果: T-shirt/top
第 12 个 sample 预测结果: Trouser  真实结果: Trouser
第 13 个 sample 预测结果: Sneaker  真实结果: Sneaker
第 14 个 sample 预测结果: Sneaker  真实结果: Sneaker
```

3.4.9 对预测结果可视化

```
# -----定义可视化函数-----
# 输入预测结果序列，真实标签序列，以及图片序列
# 目标是根据预测值对错，让其标签显示为红色或者蓝色。对：标签为蓝色；错：标签为红色
def plot_image(predicted_label, true_label, img):
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    # 显示对应图片
    plt.imshow(img, cmap=plt.cm.binary)
    # 显示预测结果的颜色，如果对上了是蓝色，否则为红色
    if predicted_label == true_label:
```



```
        color = 'blue'
    else:
        color = 'red'
    # 显示对应标签的格式，样式
    plt.xlabel('{}({})'.format(class_names[predicted_label],
                                class_names[true_label]), color=color)
# 将预测的结果以柱状图形状显示蓝对红错
def plot_value_array(predicted_label, true_label, predicted_array):
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    this_plot = plt.bar(range(10), predicted_array, color='#777777')
    plt.ylim([0, 1])
    this_plot[predicted_label].set_color('red')
    this_plot[true_label].set_color('blue')

# 预测 15 个图像与标签，并展现出来
num_rows = 5
num_cols = 3
num_images = num_rows * num_cols
plt.figure(figsize=(2 * 2 * num_cols, 2 * num_rows))

for i in range(num_images):
    plt.subplot(num_rows, 2 * num_cols, 2 * i + 1)
    pred_np_ = predictions[i, :]
    predicted_label = np.argmax(pred_np_)
    image_single = test_['x'][i, 0, ...].asnumpy()
    plot_image(predicted_label, true_label[i], image_single)
    plt.subplot(num_rows, 2 * num_cols, 2 * i + 2)
    plot_value_array(predicted_label, true_label[i], pred_np_)
plt.show()
```

输出结果：



图 3-4 输出结果

3.4.10 停止实验环境

试验完成之后请及时停止实验环境，避免资源浪费，如下图：



图 3-5 停止实验环境

3.5 实验总结

该实验主要是对全连接神经网络学习的巩固，在学习了第一个实验后，利用本实验对设计神经网络训练到评估的整体流程进行巩固。



4 附录：ModelArts 开发环境搭建

在华为云 ModelArts 平台上创建 AI 框架为 Mindspore-1.5，硬件环境为 Ascend 910+ARM 的开发环境。

步骤 1 进入华为云 ModelArts 控制台

在[华为云 ModelArts 主页](#)，点击“管理控制台”进入 ModelArts 的管理页面。

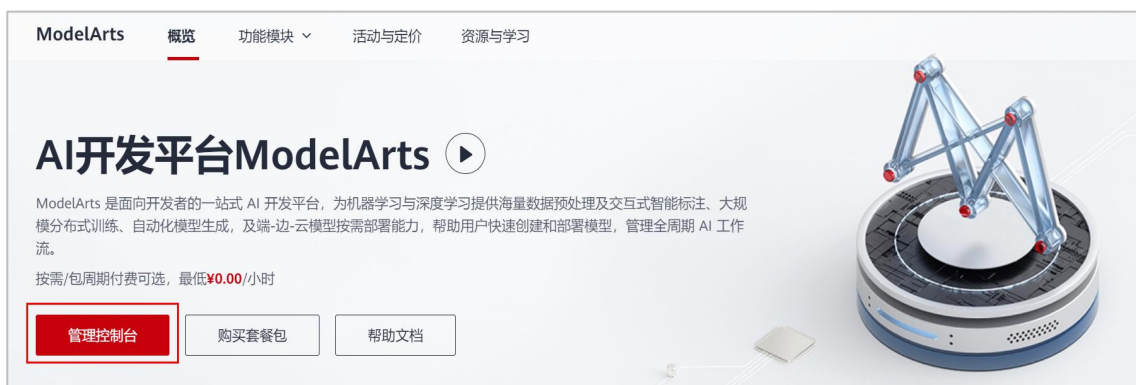


图 4-1 华为云 ModelArts 主页

步骤 2 创建 Notebook 训练作业

控制台区域选择“华北-北京四”，在左侧菜单栏中选择“开发环境”的“Notebook”，点击进入 Notebook 创建页面。

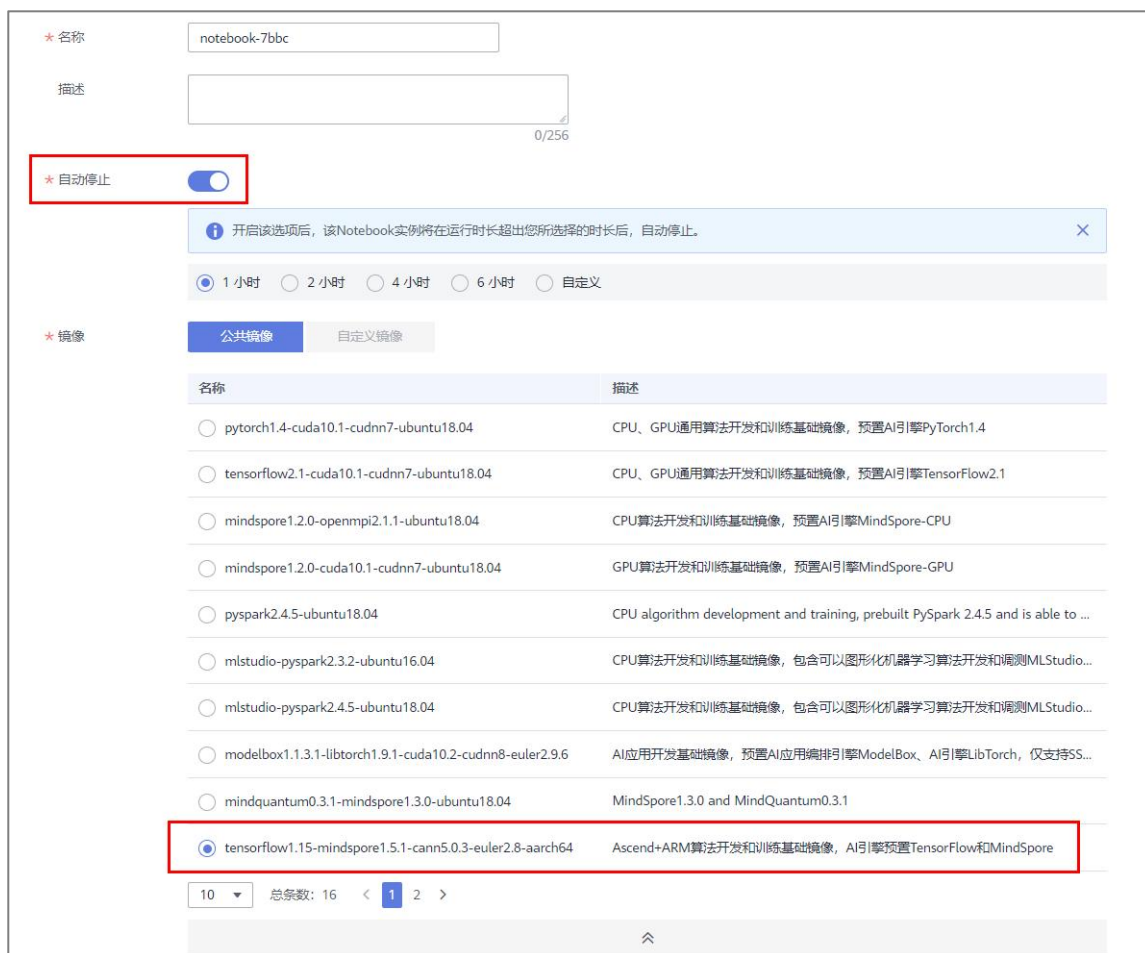


图 4-2 ModelArts 控制台

点击“创建”按钮，创建一个新的 Notebook，其配置如下：

- 名称：自定义。
- 自动停止：建议 1 小时。
- 镜像：Ascend+ARM 算法开发和训练基础镜像。
- 资源池：公共资源池。
- 类型：ASCEND。
- 规格：Ascend: 1*Ascend910|CPU: 24 核 96GB。
- 存储配置：默认存储（50GB），亦可选择 EVS，支持自定义存储规格且为专属资源。

如图所示：



* 名称

描述 0/256

* 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。

☒ 1 小时 ☐ 2 小时 ☐ 4 小时 ☐ 6 小时 ☐ 自定义

* 镜像 ☒ 公共镜像 ☐ 自定义镜像

名称	描述
<input type="radio"/> pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	CPU、GPU通用算法开发和训练基础镜像，预置AI引擎PyTorch1.4
<input type="radio"/> tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04	CPU、GPU通用算法开发和训练基础镜像，预置AI引擎TensorFlow2.1
<input type="radio"/> mindspore1.2.0-openmpi2.1.1-ubuntu18.04	CPU算法开发和训练基础镜像，预置AI引擎MindSpore-CPU
<input type="radio"/> mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04	GPU算法开发和训练基础镜像，预置AI引擎MindSpore-GPU
<input type="radio"/> pyspark2.4.5-ubuntu18.04	CPU algorithm development and training, prebuilt PySpark 2.4.5 and is able to ...
<input type="radio"/> mlstudio-pyspark2.3.2-ubuntu16.04	CPU算法开发和训练基础镜像，包含可以图形化机器学习算法开发和预测MLStudio...
<input type="radio"/> mlstudio-pyspark2.4.5-ubuntu18.04	CPU算法开发和训练基础镜像，包含可以图形化机器学习算法开发和预测MLStudio...
<input type="radio"/> modelbox1.1.3.1-libtorch1.9.1-cuda10.2-cudnn8-euler2.9.6	AI应用开发基础镜像，预置AI应用编排引擎ModelBox、AI引擎LibTorch，仅支持SS...
<input type="radio"/> mindquantum0.3.1-mindspore1.3.0-ubuntu18.04	MindSpore1.3.0 and MindQuantum0.3.1
<input checked="" type="radio"/> tensorflow1.15-mindspore1.5.1-cann5.0.3-euler2.8-aarch64	Ascend+ARM算法开发和训练基础镜像，AI引擎预置TensorFlow和MindSpore

10 总条数: 16 < 1 2 >



图 4-3 Notebook 创建配置

配置完成之后“立即创建”，规格确认无误之后“提交”。

步骤 3 启动 Notebook 进入开发环境

当上一步创建好 Notebook 状态显示为“运行中”时，在右侧“操作”栏中“打开”，即可进入在线编程页面。

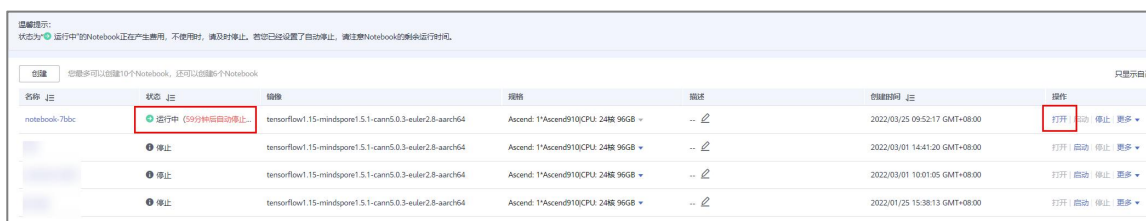


图 4-4 打开 Notebook 环境

可以在此页面创建或编辑 MindSpore 的项目，如图所示：

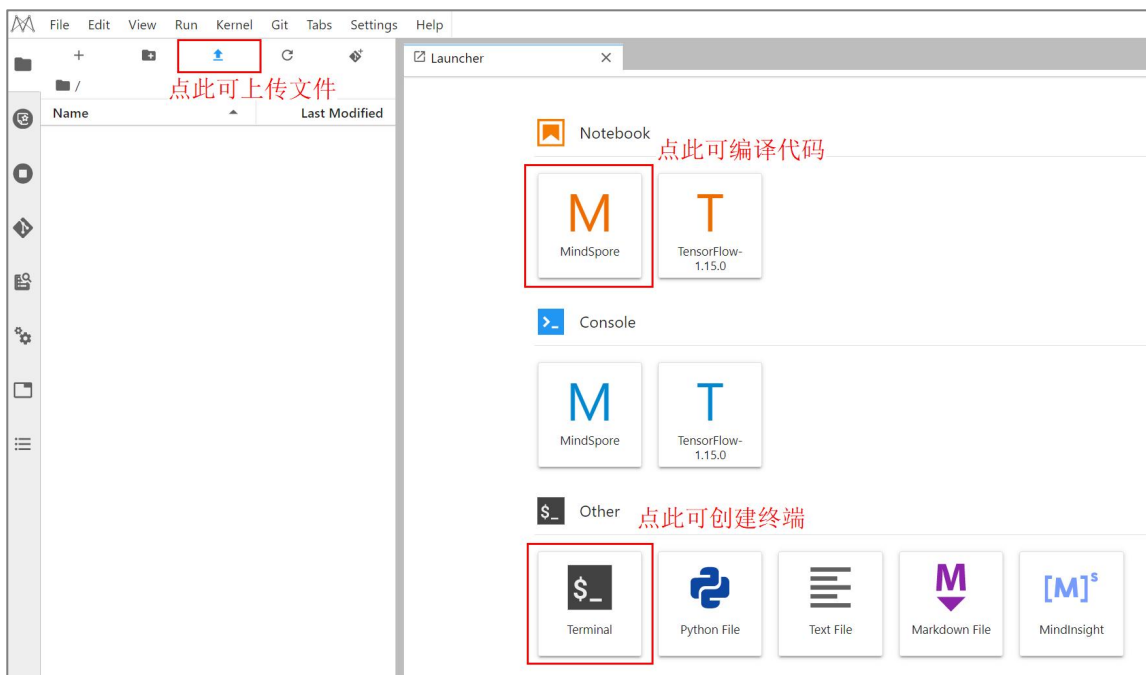


图 4-5 Notebook 开发页面

*注意：Notebook 环境内上传、创建和编辑的文件均在/home/ma-user/work 目录下。

步骤 4 停止 Notebook 训练作业

实验完成之后，请及时关闭 Notebook 训练作业，避免产生不必要的资源浪费。

登录[华为云 ModelArts 控制台](#)，在“操作”栏选择“停止”操作。

如下图所示：



图 4-6 及时停止 Notebook

至此训练用的线上 Notebook 环境搭建完成。