基于MindSpore的手写体数字识别



华为技术有限公司

**目录**

[1 实验介绍 1](#_Toc157591381)

[1.1 实验目的 1](#_Toc157591382)

[1.2 实验清单 1](#_Toc157591383)

[1.3 开发平台介绍 1](#_Toc157591384)

[2 基于MindSpore的手写体数字识别 2](#_Toc157591385)

[2.1 实验介绍 2](#_Toc157591386)

[2.2 环境准备 2](#_Toc157591387)

[2.3 实验过程 2](#_Toc157591388)

[2.4 手写数字识别 2](#_Toc157591389)

[2.5 实验总结 9](#_Toc157591390)

[3 附录：ModelArts开发环境搭建 10](#_Toc157591391)

# 实验介绍

## 实验目的

本章实验的主要目的是掌握Python在人工智能开发中的常用操作，并以手写体识别案例为例，基于深度学习框架MindSpore熟悉AI开发流程。

## 实验清单

表格：实验、简述、难度、软件环境、硬件环境。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | **简述** | **难度** | **软件环境** | **开发环境** |
| 基于MindSpore的手写体数字识别 | 了解华为自研框架MindSpore，掌握使用MindSpore构建模型并训练 | 中级 | MindSpore2.2 | ModelArts |

## 开发平台介绍

MindSpore是最佳匹配昇腾芯片的开源AI计算框架，支持Asend、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。

ModelArts成长地图：<https://support.huaweicloud.com/modelarts/index.html>

# 基于MindSpore的手写体数字识别

## 实验介绍

Mindspore是最佳匹配昇腾AI处理器算力的全场景深度学习框架，为数据科学家和算法工程师提供设计友好、运行高效的开发体验，推动人工智能软硬件应用生态繁荣发展。

本章将通过手写数字识别的案例讲述华为AI开发框架Mindspore的基本使用。

## 环境准备

本实验在华为云ModelArts上进行，开发环境快速搭建可参考：



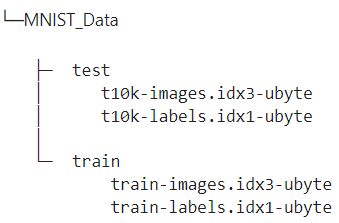
## 实验过程

获取数据集

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数据集下载地址："<https://mindspore-website.obs.cn-north-4.myhuaweicloud.com/notebook/datasets/MNIST_Data.zip>" 。

下载后的数据文件夹目录结构如下：



pip install download

%env no\_proxy='a.test.com,127.0.0.1,2.2.2.2'

from download import download

# 下载MNIST数据集

url = "https://mindspore-website.obs.cn-north-4.myhuaweicloud.com/" \

"notebook/datasets/MNIST\_Data.zip"

path = download(url, "./", kind="zip", replace=True)

导入Python库和MindSpore模块

由在使用前，导入需要的Python库和MindSpore模块。

详细的MindSpore模块说明，可以在MindSpore API页面中搜索查询：<https://www.mindspore.cn/docs/zh-CN/r2.1/index.html>

可以通过mindspore.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。

import numpy as np

import matplotlib.pyplot as plt

import mindspore as ms

from mindspore import nn

from mindspore.dataset import vision,transforms,MnistDataset

from mindspore.dataset.vision import Inter

from mindspore.common.initializer import Normal

from mindspore.train import Model

#设定运行模式为图模式，运行硬件为昇腾芯片

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

数据处理

利用MnistDataset方法读取mnist数据集，并整体对数据集进行处理。

读取数据集时指定训练集和测试集，并进行shuffle操作， 再进行修改图片尺寸，归一化，修改图像频道等工作。修改标签的数据类型，最后设定batch\_size。

# 定义预处理操作的流程，整体对数据集进行处理

def datapipe(path, batch\_size=32):

image\_transform = [

vision.Resize(size=32, interpolation=Inter.LINEAR), # Resize： 以双线性插值方式调整图像尺寸大小

vision.Rescale(1.0 / 255.0, 0.0), # Rescale: 缩放图像的像素值大小，将像素值统一除255，数据类型由unit8转为float32

vision.Normalize(mean=(0.1307,),std=(0.3081,)), # Mormalize：将像素值归一化

vision.HWC2CHW(), # HWC2CHW：将张量格式从（height,width,channel）转换成(channel,height,width)

]

label\_transform = transforms.TypeCast(ms.int32)

# 利用MnistDataset接口读取解压后的MNIST的训练集和测试集，并进行shuffle操作

dataset = MnistDataset(path, shuffle=True)

# 通过map方法对每张图片应用数据处理操作

dataset = dataset.map(operations=image\_transform, input\_columns=["image"])

# 将每个标签的数据类型转换为int32

dataset = dataset.map(operations=label\_transform, input\_columns=["label"])

# 对数据集进行分批处理；当最后一个批处理数据包含的数据条目小于batch\_size时，drop\_remainder表示是否将该批处理丢弃，不传递给下一个操作。默认值：False，不丢弃。

dataset = dataset.batch(batch\_size, drop\_remainder=True)

return dataset

dataset\_train = datapipe("MNIST\_Data/train")

dataset\_eval = datapipe("MNIST\_Data/test")

通过如下代码读取任意一个数据内容，观察打印结果。

data\_next = dataset\_train.create\_dict\_iterator(output\_numpy=True).\_\_next\_\_()

print('Batch Size/通道数/图像长/宽：', data\_next['image'].shape)

print('图像的标签样式：', data\_next['label'])

plt.figure()

plt.imshow(data\_next['image'][1,...].squeeze(), cmap="gray")

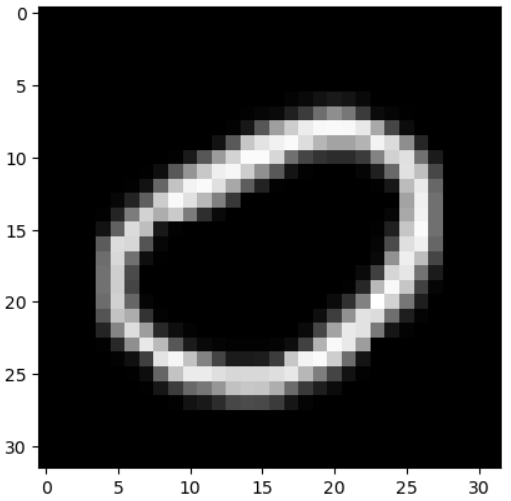
plt.grid(False)

plt.show()

输出：

Batch Size/通道数/图像长/宽： (32, 1, 32, 32)

图像的标签样式： [2 0 1 4 7 7 2 6 0 1 4 8 7 7 8 0 8 4 2 9 3 0 4 4 3 7 1 8 2 5 6 6]



构造神经网络

使用MindSpore定义神经网络需要继承mindspore.nn.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。原始模型激活函数为sigmoid，池化层为平均池化层。本实验采用ReLU激活函数和最大池化层，以获得更好的效果。

#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):

"""LeNet5"""

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

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

# 卷积层，输入的通道数为num\_channel,输出的通道数为6,卷积核大小为5\*5

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

# 卷积层，输入的通道数为6，输出的通道数为16,卷积核大小为5\*5

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

# ReLU激活函数

self.relu = nn.ReLU()

# 池化层

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

# 多维数组展平为一维数组

self.flatten = nn.Flatten()

# 全连接层，输入个数为16\*5\*5，输出个数为120

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

# 全连接层，输入个数为120，输出个数为84

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

# 全连接层，输入个数为84，分类的个数为num\_class

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

def construct(self, x):

# 使用定义好的运算构建前向网络

x = self.conv1(x)

x = self.relu(x)

x = self.max\_pool2d(x)

x = self.conv2(x)

x = self.relu(x)

x = self.max\_pool2d(x)

x = self.flatten(x)

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

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

x = self.fc3(x)

return x

搭建训练网络并进行训练

在进行正式的训练之前，需要对模型进行实例化，定义优化器和损失函数，首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。

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

训练过程中会打印loss值，类似以下代码块的运行结果所示。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。

训练的每个epoch后都打印了模型在测试集上的测试精度，从打印的精度值分析LeNet5模型的预测能力在不断提升。

定义用于训练的train\_loop函数和用于测试的test\_loop函数。

使用函数式自动微分，需先定义正向函数forward\_fn，使用mindspore.value\_and\_grad获得微分函数grad\_fn。然后，我们将微分函数和优化器的执行封装为train\_step函数，接下来循环迭代数据集进行训练即可。# Training and testing related modules

# 对LeNet5进行实例化

network = LeNet5()

# 定义损失函数

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

# 定义优化器，通过model.trainable\_params()方法获得模型的可训练参数，并传入学习率超参来初始化优化器

net\_opt = nn.Momentum(network.trainable\_params(), learning\_rate=0.01, momentum=0.9)

# 定义用于训练的train\_loop函数。

def train\_loop(model, dataset, loss\_fn, optimizer):

# 定义正向计算函数

def forward\_fn(data, label):

logits = model(data)

loss = loss\_fn(logits, label)

return loss

# 定义微分函数，使用mindspore.value\_and\_grad获得微分函数grad\_fn,输出loss和梯度。

# 由于是对模型参数求导,grad\_position 配置为None，传入可训练参数。

grad\_fn = ms.value\_and\_grad(forward\_fn, None, optimizer.parameters)

# 定义 one-step training函数

def train\_step(data, label):

loss, grads = grad\_fn(data, label)

optimizer(grads)

return loss

size = dataset.get\_dataset\_size()

model.set\_train()

for batch, (data, label) in enumerate(dataset.create\_tuple\_iterator()):

loss = train\_step(data, label)

if batch % 100 == 0:

loss, current = loss.asnumpy(), batch

print(f"loss: {loss:>7f} [{current:>3d}/{size:>3d}]")

# 定义用于测试的test\_loop函数。

def test\_loop(model, dataset, loss\_fn):

num\_batches = dataset.get\_dataset\_size()

model.set\_train(False)

total, test\_loss, correct = 0, 0, 0

for data, label in dataset.create\_tuple\_iterator():

pred = model(data)

total += len(data)

test\_loss += loss\_fn(pred, label).asnumpy()

correct += (pred.argmax(1) == label).asnumpy().sum()

test\_loss /= num\_batches

correct /= total

print(f"Test: \n Accuracy: {(100\*correct):>0.1f}%, Avg loss: {test\_loss:>8f} \n")

epochs = 3

for t in range(epochs):

print(f"Epoch {t+1}\n-------------------------------")

train\_loop(network, dataset\_train, net\_loss, net\_opt)

ms.save\_checkpoint(network, "./save\_direct.ckpt")

test\_loop(network, dataset\_eval, net\_loss)

print("Done!")

预测可视化

# 将模型参数存入parameter的字典中，采用load\_checkpoint接口加载模型参数

param\_dict = ms.load\_checkpoint("./save\_direct.ckpt")

# 重新定义一个LeNet5神经网络

net = network

# 将参数加载到网络中

ms.load\_param\_into\_net(net, param\_dict)

model = Model(net)

data\_test = dataset\_eval.create\_dict\_iterator()

data = next(data\_test)

images = data["image"].asnumpy()

labels = data["label"].asnumpy()

# 使用函数model.predict预测image对应分类

output = model.predict(ms.Tensor(data['image']))

pred = np.argmax(output.asnumpy(), axis=1)

plt.figure()

for i in range(1, 9):

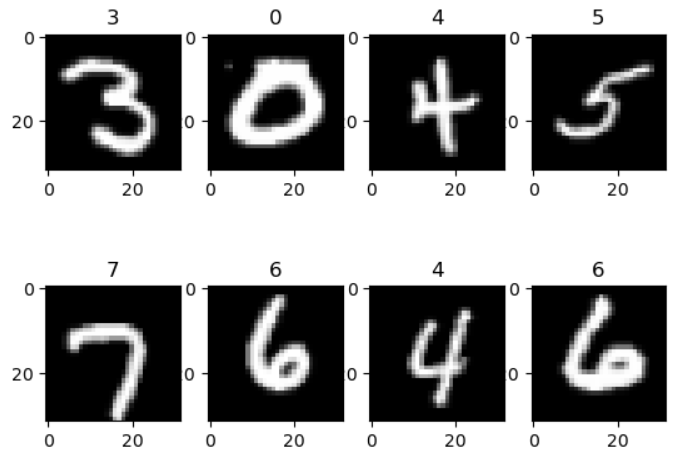
plt.subplot(2, 4, i)

plt.imshow(images[i-1].squeeze(), cmap="gray")

plt.title(pred[i-1])

plt.show()

输出：



## 实验总结

本章节主要通过华为深度学习框架MindSpore搭建LeNet-5网络模型完成手写数字识别的全流程。