昇腾创新实践课

MNIST手写体识别实验手册



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2022。 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  C:\Users\jwx341670\Desktop\华为标志 Huawei Logo 2018\竖版标志Vertical Version\PNG\HW_POS_RBG_Vertical-150ppi.png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](http://e.huawei.com/).huawei.com |

目录

[1 实验环境介绍 2](#_Toc142051086)

[1.1 实验介绍 2](#_Toc142051087)

[1.1.1 关于本实验 2](#_Toc142051088)

[1.1.2 实验环境介绍 2](#_Toc142051089)

[2 MNIST手写体识别实验 3](#_Toc142051090)

[2.1 实验介绍 3](#_Toc142051091)

[2.1.1 关于本实验 3](#_Toc142051092)

[2.1.2 实验目的 4](#_Toc142051093)

[2.1.3 背景知识 4](#_Toc142051094)

[2.1.4 实验设计 4](#_Toc142051095)

[2.2 实验过程 4](#_Toc142051096)

[2.2.1 环境准备 4](#_Toc142051097)

[2.2.2 数据处理 5](#_Toc142051098)

[2.2.3 网络定义 6](#_Toc142051099)

[2.2.4 模型训练 8](#_Toc142051100)

[2.2.5 模型评估 9](#_Toc142051101)

[2.2.6 效果展示 9](#_Toc142051102)

[2.3 实验总结 11](#_Toc142051103)

[2.4 思考题 11](#_Toc142051104)

# 实验环境介绍

## 实验介绍

### 关于本实验

本实验使用MindSpore深度学习框架，进行网络搭建、数据处理、网络训练和测试，完成MNIST手写体识别任务。

### 实验环境介绍

实验、介绍、难度、软件环境、硬件环境：

实验环境介绍

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | **实验介绍** | **难度** | **软件环境** | **开发环境** |
| MNIST手写体识别实验 | 基于MindSpore的进阶操作，使用MNIST数据集搭建卷积神经网络LeNet5，进行模型训练及评估； | 简单 | Python3.7 MindSpore1.9 | Modelarts |

# MNIST手写体识别实验

## 实验介绍

### 关于本实验

本实验使用MindSpore深度学习框架，进行网络搭建、数据处理、网络训练和测试，完成MNIST手写体识别任务。

数据集介绍：

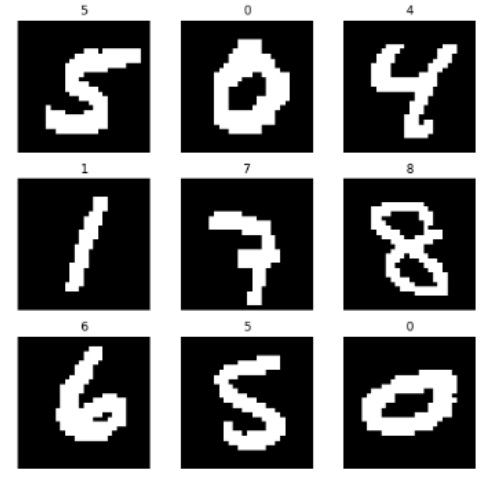
MNIST数据集来自美国国家标准与技术研究所，National Institute of Standards and Technology(NIST),数据集由来自250个不同人手写的数字构成，其中50%是高中学生，50%来自人口普查局（the Census Bureau）的工作人员。

训练集：60000，测试集：10000

MNIST数据集可在 http://yann.lecun.com/exdb/mnist/ 获取

本手册提供将训练、测试数据分好的数据集，链接：<https://zhuanyejianshe.obs.cn-north-4.myhuaweicloud.com/chuangxinshijianke/MindSpore/data-MNIST.rar>

下载后解压缩，获得data文件夹，将此文件夹放置于项目开发文件夹便可使用。



### 实验目的

理解MindSpore开发基本流程。

理解MindSpore常用模块的功能。

掌握MindSpore的基础操作。

掌握卷积神经网络的搭建。

### 背景知识

卷积神经网络知识，MindSpore基础知识，MindSpore进阶知识，图像数据预处理，LeNet5卷积神经网络结构。

### 实验设计



## 实验过程

### 环境准备

MindSpore模块主要用于本次实验卷积神经网络的构建，包括很多子模块。

mindspore.dataset：包括MNIST数据集的载入与处理，也可以自定义数据集。

mindspore.common：包中会有诸如type形态转变、权重初始化等的常规工具。

mindspore.nn：主要包括网络可能涉及到的各类网络层，诸如卷积层、池化层、全连接层，也包括损失函数，激活函数等。

Model：承载网络结构，并能够调用优化器、损失函数、评价指标。

代码：

import mindspore

from mindspore import dtype

from mindspore import context

from mindspore.dataset import MnistDataset

from mindspore.dataset.vision import Resize, Rescale, HWC2CHW

from mindspore.dataset.transforms import TypeCast

from mindspore import nn

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

from mindspore.train import Model

from mindspore.nn.metrics import Accuracy

### 数据处理

定义数据预处理函数。

函数功能包括：

1. 加载数据集

2. 打乱数据集

3. 图像特征处理（标准化、通道转换等）

4. 批量输出数据

5. 重复

下载数据集

!wget https://zhuanyejianshe.obs.cn-north-4.myhuaweicloud.com/chuangxinshijianke/cv-nlp/MNIST.zip

!unzip MNIST.zip

打印数据集列名

train\_dataset = MnistDataset('MNIST/train')

test\_dataset = MnistDataset('MNIST/test')

print(train\_dataset.get\_col\_names()) # 打印数据集中包含的数据列名，用于dataset的预处理

输出

['image', 'label']

使用create\_tuple\_iterator对数据集进行迭代，打印数据集shape

for image, label in test\_dataset.create\_tuple\_iterator():

print(f"Shape of image [N, C, H, W]: {image.shape} {image.dtype}")

print(f"Shape of label: {label.shape} {label.dtype}")

break

输出

Shape of image [N, C, H, W]: (28, 28, 1) UInt8

Shape of label: () UInt32

数据预处理函数实现

def create\_dataset(data\_path, batch\_size=32, num\_parallel\_workers=1):

"""

create dataset for train or test

"""

# define dataset

mnist\_ds = MnistDataset(data\_path)

resize\_height, resize\_width = 32, 32

rescale = 1.0 / 255.0

rescale\_nml = 1 / 0.3081

shift\_nml = -1 \* 0.1307 / 0.3081

# define map operations

resize\_op = Resize((resize\_height, resize\_width)) # Bilinear mode

rescale\_nml\_op = Rescale(rescale\_nml \* rescale, shift\_nml)

hwc2chw\_op = HWC2CHW()

type\_cast\_op = TypeCast(dtype.int32)

# apply map operations on images

mnist\_ds = mnist\_ds.map(operations=type\_cast\_op, input\_columns="label", num\_parallel\_workers=num\_parallel\_workers)

mnist\_ds = mnist\_ds.map(operations=resize\_op, input\_columns="image", num\_parallel\_workers=num\_parallel\_workers)

mnist\_ds = mnist\_ds.map(operations=rescale\_nml\_op, input\_columns="image", num\_parallel\_workers=num\_parallel\_workers)

mnist\_ds = mnist\_ds.map(operations=hwc2chw\_op, input\_columns="image", num\_parallel\_workers=num\_parallel\_workers)

# apply DatasetOps

mnist\_ds = mnist\_ds.shuffle(buffer\_size=1024)

mnist\_ds = mnist\_ds.batch(batch\_size, drop\_remainder=True)

return mnist\_ds

加载训练数据集，并打印数据shape

ds\_train = create\_dataset("./MNIST/train", 32)

for data in ds\_train.create\_dict\_iterator():

print(f"Shape of image [N, C, H, W]: {data['image'].shape} {data['image'].dtype}")

print(f"Shape of label: {data['label'].shape} {data['label'].dtype}")

break

输出：

Shape of image [N, C, H, W]: (32, 1, 32, 32) Float32

Shape of label: (32,) Int32

### 网络定义

参考LeNet网络结构，构建网络：

LeNet-5出自论文《Gradient-Based Learning Applied to Document Recognition》，原本是一种用于手写体字符识别的非常高效的卷积神经网络，包含了深度学习的基本模块：卷积层，池化层，全连接层。

mindspore.nn类是构建所有网络的基类，也是网络的基本单元。当用户需要自定义网络时，可以继承nn.Cell类，并重写\_\_init\_\_方法和construct方法。\_\_init\_\_包含所有网络层的定义，construct中包含数据（Tensor）的变换过程（即计算图的构造过程）

import mindspore.nn as nn

class LeNet5(nn.Cell):

"""

LeNet-5网络结构

"""

def \_\_init\_\_(self, num\_class=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')

# 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)

# 全连接层，输入个数为120，输出个数为84

self.fc2 = nn.Dense(120, 84)

# 全连接层，输入个数为84，分类的个数为num\_class

self.fc3 = nn.Dense(84, num\_class)

# 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()

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.fc1(x)

x = self.relu(x)

x = self.fc2(x)

x = self.relu(x)

x = self.fc3(x)

return x

network = LeNet5(num\_class=10)

实例化LeNet5对象，并查看其结构

model = LeNet5()

print(model)

输出

LeNet5<

(conv1): Conv2d<input\_channels=1, output\_channels=6, kernel\_size=(5, 5), stride=(1, 1), pad\_mode=valid, padding=0, dilation=(1, 1), group=1, has\_bias=False, weight\_init=normal, bias\_init=zeros, format=NCHW>

(conv2): Conv2d<input\_channels=6, output\_channels=16, kernel\_size=(5, 5), stride=(1, 1), pad\_mode=valid, padding=0, dilation=(1, 1), group=1, has\_bias=False, weight\_init=normal, bias\_init=zeros, format=NCHW>

(fc1): Dense<input\_channels=400, output\_channels=120, has\_bias=True>

(fc2): Dense<input\_channels=120, output\_channels=84, has\_bias=True>

(fc3): Dense<input\_channels=84, output\_channels=10, has\_bias=True>

(relu): ReLU<>

(max\_pool2d): MaxPool2d<kernel\_size=2, stride=2, pad\_mode=VALID>

(flatten): Flatten<>

>

### 模型训练

载入数据集

构建网络，包含：构建网络、选择损失函数、优化器、模型

训练模型

context.set\_context(mode=context.GRAPH\_MODE)

ds\_train = create\_dataset("./MNIST/train", 32)

network = LeNet5(10)

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

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

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

config\_ck = CheckpointConfig(save\_checkpoint\_steps=1875,

keep\_checkpoint\_max=10)

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

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

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

model.train(10, ds\_train, callbacks=[time\_cb, ckpoint\_cb, LossMonitor()], dataset\_sink\_mode=True)

输出：

============== Starting Training ==============

epoch: 1 step: 1875, loss is 2.2961819171905518

Train epoch time: 47902.712 ms, per step time: 25.548 ms

epoch: 2 step: 1875, loss is 2.2973110675811768

Train epoch time: 2669.290 ms, per step time: 1.424 ms

epoch: 3 step: 1875, loss is 2.312480926513672

Train epoch time: 2681.684 ms, per step time: 1.430 ms

epoch: 4 step: 1875, loss is 0.17669112980365753

Train epoch time: 3041.176 ms, per step time: 1.622 ms

epoch: 5 step: 1875, loss is 0.1346883475780487

Train epoch time: 3321.495 ms, per step time: 1.771 ms

epoch: 6 step: 1875, loss is 0.11107778549194336

Train epoch time: 3286.046 ms, per step time: 1.753 ms

epoch: 7 step: 1875, loss is 0.05199004337191582

Train epoch time: 3106.234 ms, per step time: 1.657 ms

epoch: 8 step: 1875, loss is 0.11010876297950745

Train epoch time: 3457.695 ms, per step time: 1.844 ms

epoch: 9 step: 1875, loss is 0.0002306260576006025

Train epoch time: 3462.788 ms, per step time: 1.847 ms

epoch: 10 step: 1875, loss is 7.86668824730441e-05

Train epoch time: 3224.511 ms, per step time: 1.720 ms

### 模型评估

查看模型在测试集的准确率

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

network = LeNet5(10)

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

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

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

param\_dict = load\_checkpoint("checkpoint\_lenet-10\_1875.ckpt")

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

ds\_eval = create\_dataset("./MNIST/test", 32)

acc = model.eval(ds\_eval)

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

输出：

============== Starting Testing ==============

============== {'Accuracy': 0.9889823717948718} ==============

### 效果展示

import matplotlib.pyplot as plt

import numpy as np

ds\_test = create\_dataset("./MNIST/test", 9)

ds\_test = ds\_test.create\_dict\_iterator()

data = next(ds\_test)

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

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

plt.figure()

for i in range(1, 10):

plt.subplot(3, 3, i)

plt.imshow(images[i-1][0], interpolation="None", cmap="gray")

plt.show()

# 使用函数model.predict预测image对应分类

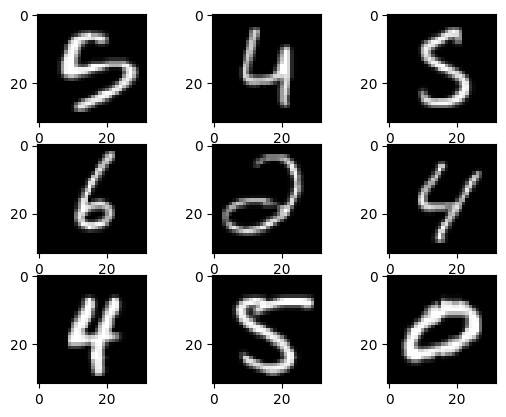
output = model.predict(mindspore.Tensor(data['image']))

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

# 输出预测分类与实际分类

print(f'Predicted: "{predicted}", Labels: "{labels}"')

输出：



Predicted: "[5 4 5 6 2 4 4 5 0]", Labels: "[5 4 5 6 2 4 4 5 0]"

模型导出

from mindspore import Tensor, context, load\_checkpoint, load\_param\_into\_net, export

network = LeNet5(10)

# load network checkpoint

param\_dict = load\_checkpoint("checkpoint\_lenet-10\_1875.ckpt")

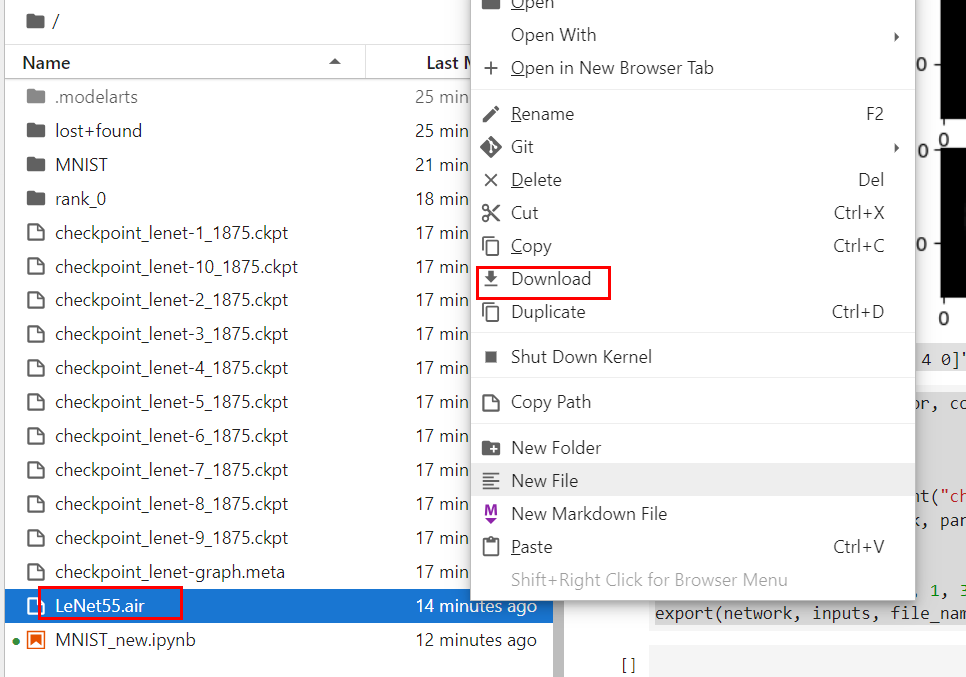
load\_param\_into\_net(network, param\_dict)

# export network

inputs = Tensor(np.ones([1, 1, 32, 32]), mindspore.float32) # batch size 1, image height and weight 32

export(network, inputs, file\_name="LeNet5", file\_format="AIR")

模型下载



## 实验总结

本实验介绍了MindSpore在图像数据集上的应用，使用MNIST手写体数据集搭建LeNet5卷积神经网络，让学员熟悉MindSpore的进阶用法，掌握MindSpore开发的流程。

## 思考题

1. 请描述MindSpore的基础数据处理流程。

答：数据加载 > shuffle > map > batch > repeat。

2. 定义网络时需要继承哪一个基类？

答：mindspore.nn.Cell。

3. 定义网络时有哪些必须编写哪两个函数？

答：\_\_init\_\_()，construct()。

4. 思考3中提到的两个函数有什么用途？

答：一般会在\_\_init\_\_()中定义算子，然后在construct()中定义网络结构。\_\_init\_\_()中的语句由Python解析执行；construct()中的语句由MindSpore接管，有语法限制。