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

**商标声明**

 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 实验环境介绍

## 1.1 实验介绍

### 1.1.1 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。

### 1.1.2 实验环境介绍

实验、介绍、难度、软件环境、硬件环境：

**表 1-1 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<br>MindSpore1.2.1<br>Numpy 1.17.5 | PC机 |

# 2 MNIST 手写体识别实验

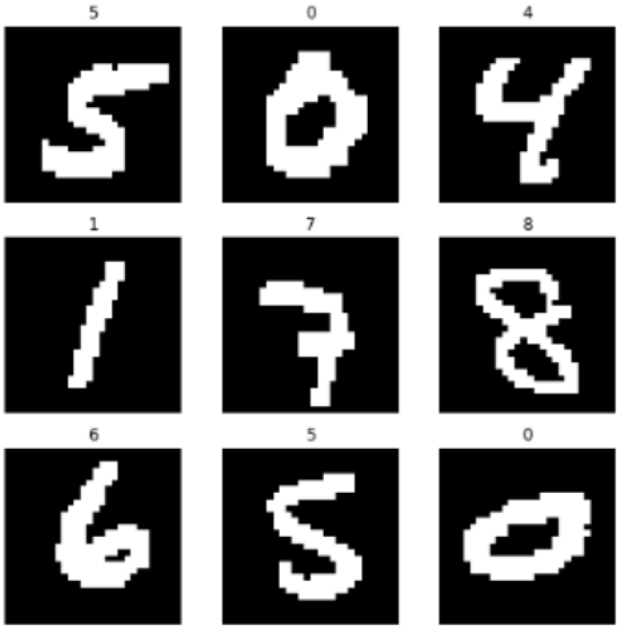
## 2.1 实验介绍

### 2.1.1 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。




### 2.1.2 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。

### 2.1.3 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。

### 2.1.4 实验设计


## 2.2 实验过程

### 2.2.1 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：

```
# mindspore.dataset
import mindspore.dataset as ds #  数据集的载入
import mindspore.dataset.transforms.c_transforms as C #  常用转化算子
import mindspore.dataset.vision.c_transforms as CV #  图像转化算子

# mindspore.common
from mindspore.common import dtype as mstype #  数据形态转换
from mindspore.common.initializer import Normal #  参数初始化

# mindspore.nn
import mindspore.nn as nn #  各类网络层都在 nn 里面
from mindspore.nn.metrics import Accuracy #  测试模型用
```

```
from mindspore import Model #  承载网络结构


# os 模块处理数据路径用
import os

# numpy
import numpy as np
```

### 2.2.2 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

代码：

```
def create_dataset(data_path, batch_size=32):
    """
    数据预处理与批量输出的函数

    Args:
        data_path:  数据路径
        batch_size:  批量大小
    """

    #  定义数据集
    data = ds.MnistDataset(data_path)

    #  打乱数据集
    data = data.shuffle(buffer_size=10000)

    #  数据标准化参数
    # MNIST 数据集的  mean = 33.3285，std = 78.5655
    mean, std = 33.3285, 78.5655

    #  定义算子
    nml_op = lambda x : np.float32((x-mean)/std) #  数据标准化，image = (image-mean)/std
    hwc2chw_op = CV.HWC2CHW() #  通道前移（为配适网络，CHW 的格式可最佳发挥昇腾芯片算力）
    type_cast_op = C.TypeCast(mstype.int32) #  原始数据的标签是 unint，计算损失需要 int
```

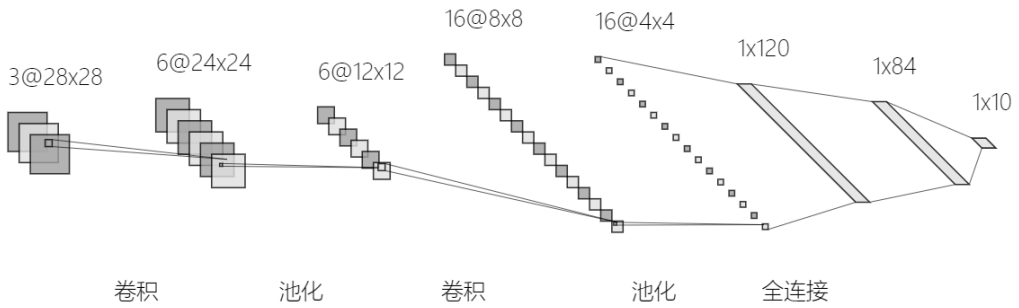
```
    # 算子运算
    data = data.map(operations=type_cast_op, input_columns='label')
    data = data.map(operations=nml_op, input_columns='image')
    data = data.map(operations=hwc2chw_op, input_columns='image')

    # 批处理
    data = data.batch(batch_size)

    # 重复
    data = data.repeat(1)

    return data
```

## 2.2.3 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。

本实验将参考 LeNet 论文，建立以下网络：




- INPUT（输入层）：输入 28*28 的图片。
- C1（卷积层）：选取 6 个 5*5 卷积核(不包含偏置)，得到 6 个特征图，每个特征图的一个边为 28-5+1=24。
- S2（池化层）：池化层是一个下采样层，输出 12*12*6 的特征图。
- C3（卷积层）：选取 16 个大小为 5*5 卷积核，得到特征图大小为 8*8*16。
- S4（池化层）：窗口大小为 2*2，输出 4*4*16 的特征图。
- F5（全连接层）：120 个神经元。
- F6（全连接层）：84 个神经元。
- OUTPUT（输出层）：10 个神经元，10 分类问题。

代码：

```
class LeNet5(nn.Cell):
```

```
# 定义算子
def __init__(self, num_class=10, num_channel=1):
    super(LeNet5, self).__init__()
    # 卷积层
    self.conv1 = nn.Conv2d(num_channel, 6, 5, pad_mode='valid')
    self.conv2 = nn.Conv2d(6, 16, 5, pad_mode='valid')

    # 全连接层
    self.fc1 = nn.Dense(4 * 4 * 16, 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))

    # 激活函数
    self.relu = nn.ReLU()

    # 最大池化层
    self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2)

    # 网络展开
    self.flatten = nn.Flatten()

# 建构网络
def construct(self, x):
    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
```

## 2.2.4 模型训练

步骤 1 载入数据集

代码：

```
train_path = os.path.join('data','train') # 训练集路径
train_data = create_dataset(train_path) # 定义训练数据集

test_path = os.path.join('data','test') # 测试集路径
```

```
test_data = create_dataset(test_path) #  定义测试数据集
```

步骤 2 构建网络

构建网络、选择损失函数、优化器、模型。

代码：

```
#  网络
net = LeNet5()

#  损失函数
net_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')

#  优化器
lr = 0.01
momentum = 0.9
net_opt = nn.Momentum(net.trainable_params(), lr, momentum)

#  模型
model = Model(net, net_loss, net_opt, metrics={'accuracy': Accuracy()})
```

步骤 3 训练模型

代码：

```
model.train(3, train_data) #  训练 3 个 epoch
```

## 2.2.5 模型评估

查看模型在测试集的准确率。

代码：

```
model.eval(test_data) #  测试网络
```

输出：

```
{'accuracy': 0.981}
```

## 2.2.6 效果展示

代码：

```
data_path=os.path.join('data', 'test')

ds_test_demo = create_dataset(test_path, batch_size=1)

for i, dic in enumerate(ds_test_demo.create_dict_iterator()):
    input_img = dic['image']
    output = model.predict(input_img)
    predict = np.argmax(output.asnumpy(),axis=1)[0]
    if i>9:
        break
```

```
    print('True: %s, Predicted: %s'%(dic['label'], predict))
```

输出：

```
True: [2], Predicted: 2
True: [0], Predicted: 0
True: [2], Predicted: 2
True: [9], Predicted: 9
True: [7], Predicted: 7
True: [8], Predicted: 8
True: [8], Predicted: 8
True: [2], Predicted: 2
True: [5], Predicted: 5
True: [4], Predicted: 4
```

## 2.3 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。

## 2.4 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。