

**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 计算机与电子通信

学　　 号

姓 名

同 组 人 员

1. **背景简介/问题描述**

随着深度学习在计算机视觉领域的迅猛发展，图像分类作为基础任务之一，在手写数字识别、目标检测、医学影像分析等诸多应用中发挥了重要作用。MindSpore作为华为推出的深度学习框架，提供了灵活易用的API和高效的计算性能。本实验以经典的MNIST手写数字数据集为例，使用MindSpore框架在本地环境下完成模型训练与推理预测，旨在帮助学习者掌握数据预处理、网络搭建、训练调优及模型部署的基本流程。

1. **算法介绍**

**2.1数据集：**MNIST数据集包含10类手写数字图像，每张图像为28×28像素的灰度图，其中训练集60000张，测试集10000张。

**2.2网络结构：**采用经典的LeNet-5网络，网络由2个卷积层（Conv）、2个池化层（Pool）和3个全连接层（FC）组成，具体结构如下：

Conv1: 输入通道1，输出通道6，卷积核5×5

Pool1: 最大池化，核大小2×2，步长2

Conv2: 输入通道6，输出通道16，卷积核5×5

Pool2: 最大池化，核大小2×2，步长2

FC1: 输入﹡2×5×5=400，输出120

FC2: 输入120，输出84

FC3: 输入84，输出类别数10

**2.3损失函数与优化器：**

损失函数：Softmax交叉熵（SoftmaxCrossEntropyWithLogits），用于衡量模型预测分布与真实标签的差异。

优化器：动量梯度下降（Momentum），学习率lr和动量momentum可调。

**2.4 Modelarts平台介绍**

ModelArts是华为云推出的一款人工智能开发平台，提供了TensorFlow、PyTorch、Caffe、MXNet等多种人工智能框架的全栈式、可持续集成及开发、部署、运维一体化的开发环境，能够快速实现一站式AI开发。同时ModelArts也支持多种AI开发的应用场景，包括图像识别、自然语言处理、机器学习、深度学习等。

**2.5 Tensorflow框架介绍**

TensorFlow是Google开源的一款人工智能框架，它可以用于构建从简单的线性回归模型到复杂的神经网络模型的各种机器学习模型，具有灵活性和可扩展性。TensorFlow包含了一系列工具和库，提供了各种功能，包括图像和语音识别、自然语言处理、语音合成等方面的任务。同时，TensorFlow还提供了多种语言的API，包括Python、C++、Java、Go等，使得开发者可以使用自己熟悉的语言进行开发。

1. **算法实现**

**第一部分：基于Mindspore框架的模型本地训练及预测**

**3.1数据处理模块（create\_dataset）**

|  |
| --- |
| def create\_dataset(data\_path, batch\_size=32, repeat\_size=1,                     num\_parallel\_workers=1):      """      create dataset for train or test      """      # define dataset      mnist\_ds = ds.MnistDataset(data\_path)      resize\_height, resize\_width = 32, 32      rescale = 1.0 / 255.0      shift = 0.0      rescale\_nml = 1 / 0.3081      shift\_nml = -1 \* 0.1307 / 0.3081      # define map operations      resize\_op = CV.Resize((resize\_height, resize\_width), interpolation=Inter.LINEAR)  # Bilinear mode      rescale\_nml\_op = CV.Rescale(rescale\_nml, shift\_nml)      rescale\_op = CV.Rescale(rescale, shift)      hwc2chw\_op = CV.HWC2CHW()      type\_cast\_op = C.TypeCast(mstype.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\_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      buffer\_size = 10000      mnist\_ds = mnist\_ds.shuffle(buffer\_size=buffer\_size)  # 10000 as in LeNet train script      mnist\_ds = mnist\_ds.batch(batch\_size, drop\_remainder=True)      mnist\_ds = mnist\_ds.repeat(repeat\_size)      return mnist\_ds |

**3.2模型定义**

|  |
| --- |
| import mindspore.nn as nn  from mindspore.common.initializer import Normal  # 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 |

**3.3训练与验证**

|  |
| --- |
| **import os**  **import ast**  **import argparse**  **from src.config import mnist\_cfg as cfg**  **from src.dataset import create\_dataset**  **from src.lenet import LeNet5**  **import mindspore.nn as nn**  **from mindspore import context**  **from mindspore.train.callback import (**  **ModelCheckpoint,**  **CheckpointConfig,**  **LossMonitor,**  **TimeMonitor,**  **)**  **from mindspore.train import Model**  **from mindspore.nn.metrics import Accuracy**  **from mindspore.common import set\_seed**  **set\_seed(1)**  **if \_\_name\_\_ == "\_\_main\_\_":**  **parser = argparse.ArgumentParser(description="MindSpore Lenet Example")**  **# 设备设置**  **parser.add\_argument(**  **"--device\_target",**  **type=str,**  **default="CPU",**  **choices=["Ascend", "GPU", "CPU"],**  **help="device where the code will be implemented (default: Ascend)",**  **)**  **parser.add\_argument(**  **"--data\_path",**  **type=str,**  **default="C:/Users/aku05/Downloads/ai\_lab2/lenet/MNIST\_Data",**  **help="path where the dataset is saved",**  **)**  **parser.add\_argument(**  **"--ckpt\_path",**  **type=str,**  **default="C:/Users/aku05/Downloads/ai\_lab2/lenet/ckpt",**  **help="if is test, must provide\**  **path where the trained ckpt file",**  **)**  **parser.add\_argument(**  **"--dataset\_sink\_mode",**  **type=ast.literal\_eval,**  **default=True,**  **help="dataset\_sink\_mode is False or True",**  **)**  **args = parser.parse\_args()**  **context.set\_context(mode=context.GRAPH\_MODE, device\_target=args.device\_target)**  **ds\_train = create\_dataset(os.path.join(args.data\_path, "train"), cfg.batch\_size)**  **network = LeNet5(cfg.num\_classes)**  **# 设定loss函数**  **"""your code here"""**  **loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")**  **# 设定优化器**  **"""your code here"""**  **optimizer = nn.Momentum(**  **network.trainable\_params(), learning\_rate=cfg.lr, momentum=cfg.momentum**  **)**  **time\_cb = TimeMonitor(data\_size=ds\_train.get\_dataset\_size())**  **config\_ck = CheckpointConfig(**  **save\_checkpoint\_steps=cfg.save\_checkpoint\_steps,**  **keep\_checkpoint\_max=cfg.keep\_checkpoint\_max,**  **)**  **ckpoint\_cb = ModelCheckpoint(**  **prefix="checkpoint\_lenet", directory=args.ckpt\_path, config=config\_ck**  **)**  **# 编译形成模型**  **"""your code here"""**  **model = Model(**  **network, loss\_fn=loss, optimizer=optimizer, metrics={"Accuracy": Accuracy()}**  **)**  **print("============== Starting Training ==============")**  **# 训练网络 train.py**  **"""your code here"""**  **model.train(**  **cfg.epoch\_size,**  **ds\_train,**  **callbacks=[time\_cb, ckpoint\_cb, LossMonitor()],**  **dataset\_sink\_mode=args.dataset\_sink\_mode,**  **)** |

**第二部分：基于Modelarts平台和Tensorflow框架的模型训练及部署**

实验结果如下：

|  |
| --- |
|  |

1. **讨论及结论**

**4.1训练过程观察：**

在多个epoch迭代中，损失函数值整体呈下降趋势，表明模型在不断学习。

Accuracy指标逐步提升，一般在第5~10个epoch后趋于稳定。

**4.2实验结果：**

在10个epoch训练后，模型在测试集上的准确率可达约99%以上，证明LeNet在MNIST数据集上的良好表现。

**4.3讨论与改进：**

数据增强：可加入随机裁剪、旋转等增强操作，以提高模型的鲁棒性。

网络结构：尝试更深层或残差网络，如ResNet，以提升复杂任务的性能。

超参数调优：进一步调整学习率、动量和batch\_size等以获得更优表现。

**4.4结论：**

本实验基于MindSpore框架，从数据预处理到模型定义、训练与推理完整地演示了手写数字识别的流程。通过LeNet网络在MNIST数据集上的应用，验证了MindSpore在模型开发和本地训练方面的便利性与高效性，为后续更复杂任务的实现奠定了基础。