**南京信息工程大学 实验（实习）报告**

实验（实习）名称 LeNet5深度学习网络应用 日期 2022.12.14 指导教师 文学志

系 计算机 专业 计科 年级 20 班次 3 姓名 赵骁汉 学号 202083290030

1. 实验目的

本章实验的主要目的是掌握计算机视觉中物体识别相关基础知识点，了解物体识别中的图像识别与视频识别，同时掌握深度学习相关基础知识，尤其是卷积神经网络。掌握不同相关的物体识别中比较成熟的物体识别网络的设计原理，熟悉使用LeNet5深度学习网络进行物体识别。

1. 实验任务

基于cifar10数据集，使用MindSpore和LeNet5深度学习网络，实现图像分类。

（详见物体识别实验手册.docx）

1. 实验步骤
2. 创建华为云notebook
3. 导入实验模块
4. 数据集展示与数据从初始化
5. 构建网络模型
6. 模型训练与测试
7. 模型优化与重新训练
8. 模型测试与可视化
9. 实验结果
10. 创建华为云，超参数定义，下载数据

代码：

import mindspore

# 载入mindspore的默认数据集

import mindspore.dataset as ds

# 常用转化用算子

import mindspore.dataset.transforms.c\_transforms as C

# 图像转化用算子

####\_\_\_\_####

import mindspore.dataset.vision.c\_transforms as CV

from mindspore.common import dtype as mstype

# mindspore的tensor

from mindspore import Tensor

# 各类网络层都在nn里面

import mindspore.nn as nn

# 参数初始化的方式

from mindspore.common.initializer import TruncatedNormal

# 设置mindspore运行的环境

from mindspore import context

# 引入训练时候会使用到回调函数，如checkpoint, lossMoniter

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

# 引入模型

from mindspore.train import Model

# 引入评估模型的包

from mindspore.nn.metrics import Accuracy

# numpy

import numpy as np

# 画图用

import matplotlib.pyplot as plt

####\_\_\_\_####

# 下载数据相关的包

import os

import requests

import zipfile

截图：文本

描述已自动生成

1. 数据预览

代码：

#创建图像标签列表

category\_dict = {0:'airplane',1:'automobile',2:'bird',3:'cat',4:'deer',5:'dog',

6:'frog',7:'horse',8:'ship',9:'truck'}

####\_\_\_\_####

current\_path = os.getcwd()

data\_path = os.path.join(current\_path, 'data/10-verify-bin')

cifar\_ds = ds.Cifar10Dataset(data\_path)

# 设置图像大小

plt.figure(figsize=(8,8))

i = 1

# 打印9张子图

for dic in cifar\_ds.create\_dict\_iterator():

plt.subplot(3,3,i)

####\_\_\_\_####

plt.imshow(dic['image'].asnumpy())

plt.xticks([])

plt.yticks([])

plt.axis('off')

plt.title(category\_dict[dic['label'].asnumpy().sum()])

i +=1

if i > 9 :

break

plt.show()

截图：

图形用户界面, 网站

描述已自动生成

1. 定义网络结构

代码：

def get\_data(datapath):

cifar\_ds = ds.Cifar10Dataset(datapath)

return cifar\_ds

def process\_dataset(cifar\_ds,batch\_size =32,status="train"):

'''

---- 定义算子 ----

'''

# 归一化

rescale = 1.0 / 255.0

# 平移

shift = 0.0

resize\_op = CV.Resize((32, 32))

rescale\_op = CV.Rescale(rescale, shift)

# 对于RGB三通道分别设定mean和std

normalize\_op = CV.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010))

if status == "train":

# 随机裁剪

random\_crop\_op = CV.RandomCrop([32, 32], [4, 4, 4, 4])

# 随机翻转

random\_horizontal\_op = CV.RandomHorizontalFlip()

# 通道变化

channel\_swap\_op = CV.HWC2CHW()

# 类型变化

typecast\_op = C.TypeCast(mstype.int32)

'''

---- 算子运算 ----

'''

cifar\_ds = cifar\_ds.map(input\_columns="label", operations=typecast\_op)

if status == "train":

cifar\_ds = cifar\_ds.map(input\_columns="image", operations=random\_crop\_op)

cifar\_ds = cifar\_ds.map(input\_columns="image", operations=random\_horizontal\_op)

cifar\_ds = cifar\_ds.map(input\_columns="image", operations=resize\_op)

cifar\_ds = cifar\_ds.map(input\_columns="image", operations=rescale\_op)

cifar\_ds = cifar\_ds.map(input\_columns="image", operations=normalize\_op)

cifar\_ds = cifar\_ds.map(input\_columns="image", operations=channel\_swap\_op)

# shuffle

cifar\_ds = cifar\_ds.shuffle(buffer\_size=1000)

# 切分数据集到batch\_size

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

return cifar\_ds

"""LeNet."""

def conv(in\_channels, out\_channels, kernel\_size, stride=1, padding=0):

"""weight initial for conv layer"""

weight = weight\_variable()

return nn.Conv2d(in\_channels, out\_channels,

kernel\_size=kernel\_size, stride=stride, padding=padding,

weight\_init=weight, has\_bias=False, pad\_mode="same")

def fc\_with\_initialize(input\_channels, out\_channels):

"""weight initial for fc layer"""

weight = weight\_variable()

bias = weight\_variable()

return nn.Dense(input\_channels, out\_channels, weight, bias)

def weight\_variable():

"""weight initial"""

return TruncatedNormal(0.02)

class LeNet5(nn.Cell):

"""

Lenet network

Args:

num\_class (int): Num classes. Default: 10.

Returns:

Tensor, output tensor

Examples:

>>> LeNet(num\_class=10)

"""

def \_\_init\_\_(self, num\_class=10, channel=3):

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

self.num\_class = num\_class

self.conv1 = conv(channel, 6, 5)

self.conv2 = conv(6, 16, 5)

self.fc1 = fc\_with\_initialize(16 \* 8 \* 8, 120)

self.fc2 = fc\_with\_initialize(120, 84)

self.fc3 = fc\_with\_initialize(84, self.num\_class)

self.relu = nn.ReLU()

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

self.flatten = nn.Flatten()

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

# 构建网络

network = LeNet5(10)

# 设置模型的设备与图的模式

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

# 使用交叉熵函数作为损失函数

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

# 优化器为Adam

net\_opt = nn.Adam(params=network.trainable\_params(), learning\_rate=0.001)

# 监控每个epoch训练的时间

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

from mindspore.train.callback import Callback

class EvalCallBack(Callback):

def \_\_init\_\_(self, model, eval\_dataset, eval\_per\_epoch, epoch\_per\_eval):

self.model = model

self.eval\_dataset = eval\_dataset

self.eval\_per\_epoch = eval\_per\_epoch

self.epoch\_per\_eval = epoch\_per\_eval

def epoch\_end(self, run\_context):

cb\_param = run\_context.original\_args()

cur\_epoch = cb\_param.cur\_epoch\_num

if cur\_epoch % self.eval\_per\_epoch == 0:

acc = self.model.eval(self.eval\_dataset, dataset\_sink\_mode=False)

self.epoch\_per\_eval["epoch"].append(cur\_epoch)

self.epoch\_per\_eval["acc"].append(acc["Accuracy"])

print(acc)

1. 训练

代码：

# 设置CheckpointConfig，callback函数。save\_checkpoint\_steps=训练总数/batch\_size

config\_ck = CheckpointConfig(save\_checkpoint\_steps=1562,

keep\_checkpoint\_max=10)

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

# 建立可训练模型

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

eval\_per\_epoch = 1

epoch\_per\_eval = {"epoch": [], "acc": []}

eval\_cb = EvalCallBack(model, ds\_train, eval\_per\_epoch, epoch\_per\_eval)

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

model.train(10, ds\_train,callbacks=[ckpoint\_cb, LossMonitor(per\_print\_times=1),eval\_cb],dataset\_sink\_mode=True)

截图：

文本

描述已自动生成

1. 测试

代码：

data\_path = os.path.join(current\_path, 'data/10-verify-bin')

batch\_size=32

status="test"

# 生成测试数据集

cifar\_ds = ds.Cifar10Dataset(data\_path)

ds\_eval = process\_dataset(cifar\_ds,batch\_size=batch\_size,status=status)

res = model.eval(ds\_eval, dataset\_sink\_mode=True)

# 评估测试集

print('test results:',res)

截图：



1. 测试预览

代码：

#创建图像标签列表

category\_dict = {0:'airplane',1:'automobile',2:'bird',3:'cat',4:'deer',5:'dog',

6:'frog',7:'horse',8:'ship',9:'truck'}

cifar\_ds = get\_data('./data/10-verify-bin')

df\_test = process\_dataset(cifar\_ds,batch\_size=1,status='test')

def normalization(data):

\_range = np.max(data) - np.min(data)

return (data - np.min(data)) / \_range

# 设置图像大小

plt.figure(figsize=(10,10))

i = 1

# 打印9张子图

for dic in df\_test:

# 预测单张图片

input\_img = dic[0]

output = model.predict(Tensor(input\_img))

output = nn.Softmax()(output)

# 反馈可能性最大的类别

predicted = np.argmax(output.asnumpy(),axis=1)[0]

# 可视化

plt.subplot(3,3,i)

# 删除batch维度

input\_image = np.squeeze(input\_img.asnumpy(),axis=0).transpose(1,2,0)

# 重新归一化，方便可视化

input\_image = normalization(input\_image)

plt.imshow(input\_image)

plt.xticks([])

plt.yticks([])

plt.axis('off')

plt.title('True label:%s,\n Predicted:%s'%(category\_dict[dic[1].asnumpy().sum()],category\_dict[predicted]))

i +=1

if i > 9 :

break

plt.show()

截图：

日程表

低可信度描述已自动生成

1. 新网络定义：

代码：

class LeNet5\_2(nn.Cell):

"""

Lenet network

Args:

num\_class (int): Num classes. Default: 10.

Returns:

Tensor, output tensor

Examples:

>>> LeNet(num\_class=10)

"""

def \_\_init\_\_(self, num\_class=10, channel=3):

super(LeNet5\_2, self).\_\_init\_\_()

self.num\_class = num\_class

self.conv1\_1 = conv(channel, 8, 3)

self.bn2\_1 = nn.BatchNorm2d(num\_features=8)

self.conv1\_2 = conv(8, 16, 3)

self.bn2\_2 = nn.BatchNorm2d(num\_features=16)

self.conv2\_1 = conv(16, 32, 3)

self.bn2\_3 = nn.BatchNorm2d(num\_features=32)

self.conv2\_2 = conv(32, 64, 3)

self.bn2\_4 = nn.BatchNorm2d(num\_features=64)

self.fc1 = fc\_with\_initialize(64\*8\*8, 120)

self.bn1\_1 = nn.BatchNorm1d(num\_features=120)

self.fc2 = fc\_with\_initialize(120, 84)

self.bn1\_2 = nn.BatchNorm1d(num\_features=84)

self.fc3 = fc\_with\_initialize(84, self.num\_class)

self.relu = nn.ReLU()

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

self.flatten = nn.Flatten()

def construct(self, x):

x = self.conv1\_1(x)

x = self.bn2\_1(x)

x = self.relu(x)

x = self.conv1\_2(x)

x = self.bn2\_2(x)

x = self.relu(x)

x = self.max\_pool2d(x)

x = self.conv2\_1(x)

x = self.bn2\_3(x)

x = self.relu(x)

x = self.conv2\_2(x)

x = self.bn2\_4(x)

x = self.relu(x)

x = self.max\_pool2d(x)

x = self.flatten(x)

x = self.fc1(x)

x = self.bn1\_1(x)

x = self.relu(x)

x = self.fc2(x)

x = self.bn1\_2(x)

x = self.relu(x)

x = self.fc3(x)

return x

1. 训练

代码：

ata\_path = os.path.join(current\_path, 'data/10-batches-bin')

batch\_size=32

status="train"

# 生成训练数据集

cifar\_ds = get\_data(data\_path)

ds\_train = process\_dataset(cifar\_ds,batch\_size =batch\_size, status=status)

network = LeNet5\_2(10)

#network = resnet50(10)

# 返回当前设备

device\_target = mindspore.context.get\_context('device\_target')

# 确定图模型是否下沉到芯片上

dataset\_sink\_mode = True if device\_target in ['Ascend','GPU'] else False

# 设置模型的设备与图的模式

context.set\_context(mode=context.GRAPH\_MODE, device\_target=device\_target)

# 使用交叉熵函数作为损失函数

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

# 优化器为momentum

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

net\_opt = nn.Adam(params=network.trainable\_params(), learning\_rate=0.001)

# 时间监控，反馈每个epoch的运行时间

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

# 设置callback函数。

config\_ck = CheckpointConfig(save\_checkpoint\_steps=1562,

keep\_checkpoint\_max=10)

ckpoint\_cb = ModelCheckpoint(prefix="checkpoint\_lenet\_2\_verified",directory='./results', config=config\_ck)

# 建立可训练模型

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

eval\_per\_epoch = 1

epoch\_per\_eval = {"epoch": [], "acc": []}

eval\_cb = EvalCallBack(model, ds\_train, eval\_per\_epoch, epoch\_per\_eval)

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

model.train(10, ds\_train,callbacks=[ckpoint\_cb, LossMonitor(per\_print\_times=1),eval\_cb],dataset\_sink\_mode=True)

截图：

文本

描述已自动生成

1. 测试：

代码：

data\_path = os.path.join(current\_path, 'data/10-verify-bin')

batch\_size=32

status="test"

# 生成测试数据集

cifar\_ds = ds.Cifar10Dataset(data\_path)

ds\_eval = process\_dataset(cifar\_ds,batch\_size=batch\_size,status=status)

res = model.eval(ds\_eval, dataset\_sink\_mode=True)

# 评估测试集

print('test results:',res)

截图

：

1. 图像预览

代码：

#创建图像标签列表

category\_dict = {0:'airplane',1:'automobile',2:'bird',3:'cat',4:'deer',5:'dog',

6:'frog',7:'horse',8:'ship',9:'truck'}

cifar\_ds = get\_data('./data/10-verify-bin')

df\_test = process\_dataset(cifar\_ds,batch\_size=1,status='test')

def normalization(data):

\_range = np.max(data) - np.min(data)

return (data - np.min(data)) / \_range

# 设置图像大小

plt.figure(figsize=(10,10))

i = 1

# 打印9张子图

for dic in df\_test:

# 预测单张图片

input\_img = dic[0]

output = model.predict(Tensor(input\_img))

output = nn.Softmax()(output)

# 反馈可能性最大的类别

predicted = np.argmax(output.asnumpy(),axis=1)[0]

# 可视化

plt.subplot(3,3,i)

# 删除batch维度

input\_image = np.squeeze(input\_img.asnumpy(),axis=0).transpose(1,2,0)

# 重新归一化，方便可视化

input\_image = normalization(input\_image)

plt.imshow(input\_image)

plt.xticks([])

plt.yticks([])

plt.axis('off')

plt.title('True label:%s,\n Predicted:%s'%(category\_dict[dic[1].asnumpy().sum()],category\_dict[predicted]))

i +=1

if i > 9 :

break

plt.show()

截图：

图形用户界面

中度可信度描述已自动生成

1. 实验总结

本次实验是一个基于开源框架Mindspore的图像识别实验。该实验演示了如何利用开源框架Mindspore完成CIFAR-10图像识别任务。本章对实验做了详尽的剖析，阐明了整个实验功能、结构与流程，详细解释了如何解析数据、如何构建深度学习模型、如何保存模型等内容，并且展示了模型的优化与调参。学员可以在该实验的基础上开发更有针对性的应用实验。