2023-2024 学年春季学期

《智能计算系统》(08696037)

实验报告

姓　　名	Zhou
学　　号	211212xx
实验名称	基于 ACL 搭建 VGG16 实现图像分类应用
日　　期	2024 年 5 月 4 日

实验 (70 分)	目标 1	目标 2	目标 3	得分
	20	20	30	70
报告 (30 分)	代码 (10 分)	结果 (10 分)	格式 (10 分)	得分
	10	10	10	30
批阅人		**批阅日期**		**总得分**
Yan		2024.5.4		100

上海大学 计算机工程与科学学院

目录

一　实验环境

表 1: 实验环境

实验设备	Huawei Ascend 910(32GB)
操作系统	Euler OS 2.8.3
开发语言	Python 3.7
框架	Mindspore 1.10.0

二　实验 1 基于 ACL 实现图片分类应用

1　实验目的

- 将 ONNX 模型转为适配 Ascend910 推理的 OM 模型
- 定义数据预处理模块
- 定义推理模块
- 读取给定图片并进行推理，输出各项类别置信度

2　实验步骤

2.1　导入 Python 库 & 模块并配置运行信息

模型转换模块实现 ONNX 模型转为适配 Ascend910 推理的 OM 模型，使之可以高效的运行在 Ascend 硬件上。

代码 1: 导入库并配置运行信息

```
export install_path=/usr/local/Ascend/ascend-toolkit/latest
export PATH=/usr/local/python3.7.5/bin:${install_path}/atc/ccec_compiler/bin:${
    install_path}/atc/bin:$PATH
export PYTHONPATH=${install_path}/atc/python/site-packages:$PYTHONPATH
export LD_LIBRARY_PATH=${install_path}/atc/lib64:${install_path}/acllib/lib64:
    $LD_LIBRARY_PATH
export ASCEND_OPP_PATH=${install_path}/opp
export ASCEND_AICPU_PATH=/usr/local/Ascend/ascend-toolkit/latest
/usr/local/Ascend/ascend-toolkit/latest/atc/bin/atc --framework=5 --model="./
    model/vgg16.onnx" \
      --output="model/vgg16" --input_format=NCHW \
      --input_shape="image:1,3,224,224" \
      --log=debug \
      --soc_version=Ascend910
```

–framework：原始框架类型 5 表示 onnx

–model：vgg16 网络的模型文件（*.onnx）的路径

–output：vgg16.om 模型文件名

–input_format：输入格式为（批次、通道、长度、宽度）

–input_shape：输入的 shape，“image:” 代表 onnx 输入节点的名称

–soc_version：昇腾 AI 处理器的版本。昇腾 910 AI 处理器，此处配置为 Ascend910
–precision_mode：force_fp16 表示对该模型进行 float16 量化处理

2.2 数据预处理

对数据图片进行预处理

代码 2: 图片预处理

```
import struct
import time
import acl
import numpy as np

def resize_image(image, target_size):
    h, w = image.shape[:2]
    th, tw = target_size

    # 获取等比缩放后的尺寸
    scale = min(th / h, tw / w)
    oh, ow = round(h * scale), round(w * scale)

    # 缩放图片，opencv缩放传入尺寸为（宽，高），这里采用线性差值算法
    image = cv2.resize(image, (ow, oh), interpolation=cv2.INTER_LINEAR).astype(
        np.uint8)

    # 将剩余部分进行填充
    new_image = np.ones((th, tw, 3), dtype=np.uint8) * 114
    new_image[:oh, :ow, :] = image
    return new_image

def process_image(img_path):
    # 读取图片，OpenCV读图后格式是BGR格式，需要转为RGB格式
    image = cv2.imread(img_path, cv2.IMREAD_COLOR)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # 将图片等比resize至(224x224)
    image = resize_image(image, (224, 224))
    image = np.array(image, dtype=np.float32)

    # 将图片标准化
    image -= [125.307, 122.961, 113.8575]
    image /= [51.5865, 50.847, 51.255]

    # (h,w,c) -> (c,h,w)
    image = image.transpose((2, 0, 1))
    return image

```

```
    # 设置全局变量
    CLASSES = ("daisy", "roses", "sunflowers", "tulips")
    ACL_MEM_MALLOC_HUGE_FIRST = 0
    ACL_MEMCPY_HOST_TO_DEVICE = 1
    ACL_MEMCPY_DEVICE_TO_HOST = 2
    NPY_BYTE = 1
    model_path="./model/vgg16.om"
    img_path="./data/sunflowers_demo.jpg"
    device_id = 0
    context = None
    model_id = None
    model_desc = None
    load_input_dataset, load_output_dataset = None, None
    input_data, output_data = None, None
```

2.3 资源初始化并加载模型

代码 3: 资源初始化

```
# pyACL初始化
ret = acl.init()

# 运行管理资源申请
# 指定运算的Device。
device_id = 0
ret = acl.rt.set_device(device_id)
# 显式创建一个Context，用于管理Stream对象。
context, ret = acl.rt.create_context(device_id)
```

代码 4: 模型加载

```
# 加载离线模型文件，返回标识模型的ID。
model_id, ret = acl.mdl.load_from_file(model_path)

# 根据加载成功的模型的ID，获取该模型的描述信息。
model_desc = acl.mdl.create_desc()
ret = acl.mdl.get_desc(model_desc, model_id)
```

2.4 构建模型输入

代码 5: 准备输入数据集

```
# 1.准备模型推理的输入数据集。
# 创建aclmdlDataset类型的数据，描述模型推理的输入。
load_input_dataset = acl.mdl.create_dataset()
# 获取模型输入的数量。
input_size = acl.mdl.get_num_inputs(model_desc)
```

```
input_data = []
# 循环为每个输入申请内存，并将每个输入添加到aclmdlDataset类型的数据中。
for i in range(input_size):
    buffer_size = acl.mdl.get_input_size_by_index(model_desc, i)
    # 申请输入内存。
    buffer, ret = acl.rt.malloc(buffer_size, ACL_MEM_MALLOC_HUGE_FIRST)
    data = acl.create_data_buffer(buffer, buffer_size)
    _, ret = acl.mdl.add_dataset_buffer(load_input_dataset, data)
    input_data.append({"buffer": buffer, "size": buffer_size})

# 2.准备模型推理的输出数据集。
# 创建aclmdlDataset类型的数据，描述模型推理的输出。
load_output_dataset = acl.mdl.create_dataset()
# 获取模型输出的数量。
output_size = acl.mdl.get_num_outputs(model_desc)
output_data = []
# 循环为每个输出申请内存，并将每个输出添加到aclmdlDataset类型的数据中。
for i in range(output_size):
    buffer_size = acl.mdl.get_output_size_by_index(model_desc, i)
    # 申请输出内存。
    buffer, ret = acl.rt.malloc(buffer_size, ACL_MEM_MALLOC_HUGE_FIRST)
    data = acl.create_data_buffer(buffer, buffer_size)
    _, ret = acl.mdl.add_dataset_buffer(load_output_dataset, data)
    output_data.append({"buffer": buffer, "size": buffer_size})
```

2.5 模型推理与后处理

完成模型转换、数据预处理、和推理模块的实现之后，可以加载模型进行推理

代码 6: 读取图片、预处理、传输至 Device

```
# 1.读取并预处理图片
img = process_image(img_path)
# 2.准备模型推理的输入数据，运行模式默认为运行模式为ACL_HOST，当前实例代码中模型只有一个输入。
bytes_data = img.tobytes()
np_ptr = acl.util.bytes_to_ptr(bytes_data)
# 将图片数据从Host传输到Device。
ret = acl.rt.memcpy(input_data[0]["buffer"], input_data[0]["size"], np_ptr,
                    input_data[0]["size"], ACL_MEMCPY_HOST_TO_DEVICE)
```

代码 7: 模型推理

```
# 3.执行模型推理。
# model_id表示模型ID，在模型加载成功后，会返回标识模型的ID。
start_time = time.time()
ret = acl.mdl.execute(model_id, load_input_dataset, load_output_dataset)
end_time = time.time()
```

```
print('inference cost time: {:.1f}ms\n'.format((end_time-start_time)*1000))
```

代码 8: 后处理

```
# 4.处理模型推理的输出数据，输出置信度的类别编号。
inference_result = []
for i, item in enumerate(output_data):
    buffer_host, ret = acl.rt.malloc_host(output_data[i]["size"])
    # 将推理输出数据从Device传输到Host。
    ret = acl.rt.memcpy(buffer_host, output_data[i]["size"], output_data[i]["
        buffer"],output_data[i]["size"], ACL_MEMCPY_DEVICE_TO_HOST)

    bytes_out = acl.util.ptr_to_bytes(buffer_host, output_data[i]["size"])
    data = np.frombuffer(bytes_out, dtype=np.byte)
    inference_result.append(data)

tuple_st = struct.unpack("4f", bytearray(inference_result[0]))
vals = np.array(tuple_st).flatten()
top_k = vals.argsort()[-1:-6:-1]
print("\n======== inference results: =============")
for i, j in enumerate(top_k):
    print("top %d: class:[%s]: probability:[%f]" % (i, CLASSES[j], vals[j]))
```

```
======== inference results: =============
top 0: class:[sunflowers]: probability:[0.997559]
top 1: class:[tulips]: probability:[0.001055]
top 2: class:[daisy]: probability:[0.000991]
top 3: class:[roses]: probability:[0.000452]
```

图 1: 单一图片预测结果

2.6 模型卸载

代码 9: 模型卸载

```
# 卸载模型。
ret = acl.mdl.unload(model_id)
# 释放模型描述信息。
if model_desc:
    ret = acl.mdl.destroy_desc(model_desc)
    model_desc = None
# 释放Context。
if context:
    ret = acl.rt.destroy_context(context)
    context = None
```

代码 10: 释放内存

```
# 释放输出资源，包括数据结构和内存。
while output_data:
    item = output_data.pop()
    ret = acl.rt.free(item["buffer"])
output_number = acl.mdl.get_dataset_num_buffers(load_output_dataset)
for i in range(output_number):
    data_buf = acl.mdl.get_dataset_buffer(load_output_dataset, i)
    if data_buf:
        ret = acl.destroy_data_buffer(data_buf)
ret = acl.mdl.destroy_dataset(load_output_dataset)
```

代码 11: 去初始化

```
# 释放Device。
ret = acl.rt.reset_device(device_id)
# pyACL去初始化。
ret = acl.finalize()
```

3 实验结果分析

程序运行结果如下：

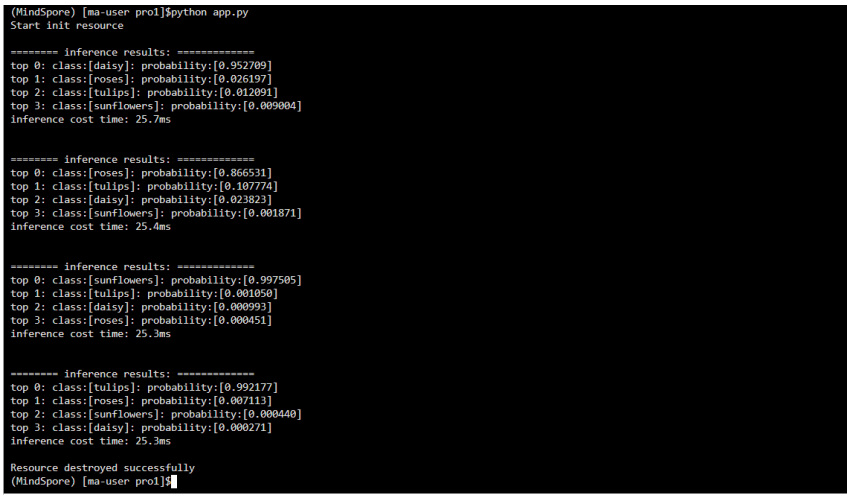

图 2: app.py 运行结果

完成实验后路径中包含的文件信息：

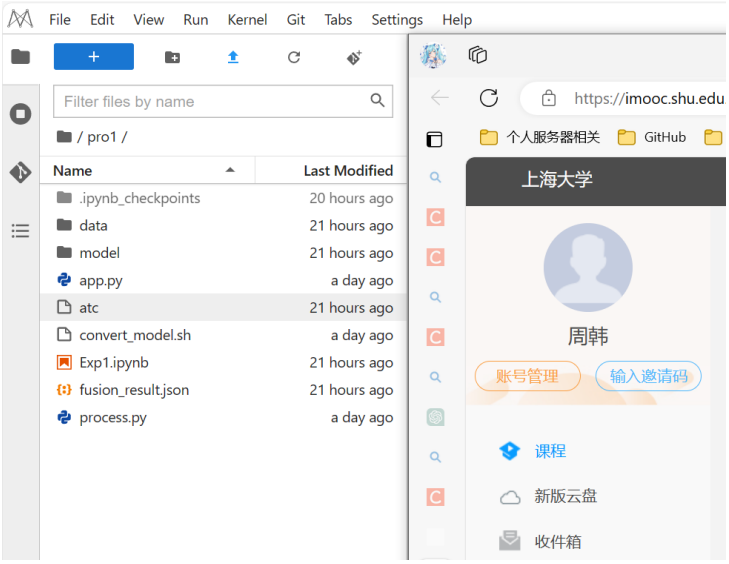

图 3: 实验一文件路径

三　实验 2 基于 MindSpore 框架实现 VGG17

1　实验目的

- 使用 MindSpore 深度学习框架，利用其成熟的算子库搭建 VGG17 神经网络模型，使用花卉数据集（雏菊、蒲公英、玫瑰、向日葵、郁金香）在 Ascend910 加速卡上进行训练和验证。
- 熟悉和使用 MindSpore 框架和 ModelArts，熟悉 MindSpore 常见 API 的使用方法，熟悉 ModelArts 一站式模型训练和部署平台。
- 基于 MindSpore 框架构建 VGG17 网络。利用花卉数据集上完成模型训练（训练平台：ModelArts，可采用昇腾 910 芯片进行训练）。模型训练完成后，对模型进行保存。
- 基于昇腾 910 推理芯片作为计算平台，利用 MindSpore 框架导入训练好的模型，并在花卉测试数据集对构建的模型进行推理验证，输出推理性能以及测试集正确率。

2　实验步骤

2.1　数据加载

代码 12: 数据加载

```
def vgg_create_dataset(data_home, image_size, batch_size, rank_id=0, rank_size
    =1, training=True):
    #加载路径
    """Data operations."""
    if training:
        data_dir = os.path.join(data_home, "train")
    else:
        data_dir = os.path.join(data_home, "test")
        print("data_dir",data_dir)
    data_set = de.ImageFolderDataset(data_dir,
                                     class_indexing={'daisy':0,'dandelion':1,'
                                         roses':2,'sunflowers':3,'tulips':4},
```

```
                                    shuffle=False, num_shards=rank_size,
                                        shard_id=rank_id)

    #数据增强的方法，上述提高的四种方法
    transform_img = vision.RandomCropDecodeResize([224,224], scale=(0.08, 1.0),
                                                  ratio=(0.75, 1.333))  # 改变尺寸

    changeswap_op = vision.HWC2CHW()
    type_cast_op = C.TypeCast(mstype.float32)
    random_horizontal_op = vision.RandomHorizontalFlip()
    #normalize_op =  vision.Normalize((0.4465, 0.4822, 0.4914), (0.2010, 0.1994,
         0.2023))

    #map操作将指定函数操作于数据集的指定列数据
    data_set = data_set.map(input_columns="image", operations=transform_img)
    data_set = data_set.map(input_columns="image", operations=type_cast_op)
    data_set = data_set.map(input_columns="image", operations=
        random_horizontal_op)
    data_set = data_set.map(input_columns="image", operations=changeswap_op)

    # shuffle来进行数据集的混洗
    data_set = data_set.shuffle(buffer_size=data_set.get_dataset_size())

    # 连续 batch_size 条数据合并为一个批处理数据
    data_set = data_set.batch(batch_size=batch_size, drop_remainder=True)
    return data_set
```

本实验顺承上次实验，采用花卉数据集，利用 mindspore.dataset 类下的 ImageFolderDataset 加载图片数据，同一个文件夹中的所有图片将被分配相同的 label。

同时使用以下几种的数据增强操作，

Random Crop：对输入图像进行在随机位置的裁剪。

RandomHorizontalFlip：对输入图像进行随机水平翻转。

HWC2CHW：将输入图像的 shape 从 <H, W, C> 转换为 <C, H, W>。如果输入图像的 shape 为 <H, W>，图像将保持不变。

Resize：对输入图像进行缩放。

2.2 搭建 VGG17 网络架构

代码 13: VGG17 网络结构

```
import mindspore.nn as nn

class Vgg(nn.Cell):
    """
    VGG网络定义.

    参数:
```

```
        num_classes (int): Class numbers. Default: 5.
        phase (int): 指定是训练/评估阶段

    返回值:
        Tensor, infer output tensor.

    example:
        self.layer1_conv1 = nn.Conv2d(in_channels=3, out_channels=64,
            kernel_size=3,weight_init='XavierUniform')
        self.layer1_bn1 = nn.BatchNorm2d(num_features=64)
        self.layer1_relu1 = nn.LeakyReLU()

    """
    def __init__(self, num_classes=5, args=None, phase="train"):
        super(Vgg, self).__init__()
        dropout_ratio = 0.5
        if not args.has_dropout or phase == "test":
            dropout_ratio = 1.0

        self.layer1_conv1 = nn.Conv2d(in_channels=3, out_channels=64,
            kernel_size=3,weight_init='XavierUniform')
        self.layer1_bn1 = nn.BatchNorm2d(num_features=64)
        self.layer1_relu1 = nn.ReLU()
        self.layer1_conv2 = nn.Conv2d(in_channels=64, out_channels=64,
            kernel_size=3,weight_init='XavierUniform')
        self.layer1_bn2 = nn.BatchNorm2d(num_features=64)
        self.layer1_relu2 = nn.ReLU()
        self.layer1_maxpool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.layer2_conv1 = nn.Conv2d(in_channels=64, out_channels=128,
            kernel_size=3,weight_init='XavierUniform')
        self.layer2_bn1 = nn.BatchNorm2d(num_features=128)
        self.layer2_relu1 = nn.ReLU()
        self.layer2_conv2 = nn.Conv2d(in_channels=128, out_channels=128,
            kernel_size=3,weight_init='XavierUniform')
        self.layer2_bn2 = nn.BatchNorm2d(num_features=128)
        self.layer2_relu2 = nn.ReLU()
        self.layer2_maxpool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.layer3_conv1 = nn.Conv2d(in_channels=128, out_channels=256,
            kernel_size=3,weight_init='XavierUniform')
        self.layer3_bn1 = nn.BatchNorm2d(num_features=256)
        self.layer3_relu1 = nn.ReLU()
        self.layer3_conv2 = nn.Conv2d(in_channels=256, out_channels=256,
            kernel_size=3,weight_init='XavierUniform')
        self.layer3_bn2 = nn.BatchNorm2d(num_features=256)
        self.layer3_relu2 = nn.ReLU()
```

```
48          self.layer3_conv3 = nn.Conv2d(in_channels=256, out_channels=256,
                kernel_size=3,weight_init='XavierUniform')
49          self.layer3_bn3 = nn.BatchNorm2d(num_features=256)
50          self.layer3_relu3 = nn.ReLU()
51          self.layer3_maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
52
53          self.layer4_conv1 = nn.Conv2d(in_channels=256, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
54          self.layer4_bn1 = nn.BatchNorm2d(num_features=512)
55          self.layer4_relu1 = nn.ReLU()
56          self.layer4_conv2 = nn.Conv2d(in_channels=512, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
57          self.layer4_bn2 = nn.BatchNorm2d(num_features=512)
58          self.layer4_relu2 = nn.ReLU()
59          self.layer4_conv3 = nn.Conv2d(in_channels=512, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
60          self.layer4_bn3 = nn.BatchNorm2d(num_features=512)
61          self.layer4_relu3 = nn.ReLU()
62          self.layer4_maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
63
64          self.layer5_conv1 = nn.Conv2d(in_channels=512, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
65          self.layer5_bn1 = nn.BatchNorm2d(num_features=512)
66          self.layer5_relu1 = nn.ReLU()
67          self.layer5_conv2 = nn.Conv2d(in_channels=512, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
68          self.layer5_bn2 = nn.BatchNorm2d(num_features=512)
69          self.layer5_relu2 = nn.ReLU()
70          self.layer5_conv3 = nn.Conv2d(in_channels=512, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
71          self.layer5_bn3 = nn.BatchNorm2d(num_features=512)
72          self.layer5_relu3 = nn.ReLU()
73          self.layer5_conv4 = nn.Conv2d(in_channels=512, out_channels=512,
                kernel_size=3,weight_init='XavierUniform')
74          self.layer5_bn4 = nn.BatchNorm2d(num_features=512)
75          self.layer5_relu4 = nn.ReLU()
76          self.layer5_maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
77          self.flatten = nn.Flatten()
78
79          self.fullyconnect1 = nn.Dense(512 * 7 * 7, 4096)
80          self.relu_1 = nn.ReLU()
81          self.dropout_1 = nn.Dropout(dropout_ratio)
82
83          self.fullyconnect2 = nn.Dense(4096, 4096)
84          self.relu_2 = nn.ReLU()
85          self.dropout_1 = nn.Dropout(dropout_ratio)
86
```

```
        self.fullyconnect3 = nn.Dense(4096, num_classes)


    def construct(self, x):
        x  =  self.layer1_conv1(x)
        x  =  self.layer1_bn1(x)
        x  =  self.layer1_relu1(x)
        x  =  self.layer1_conv2(x)
        x  =  self.layer1_bn2(x)
        x  =  self.layer1_relu2(x)
        x  =  self.layer1_maxpool(x)

        x  =  self.layer2_conv1(x)
        x  =  self.layer2_bn1(x)
        x  =  self.layer2_relu1(x)
        x  =  self.layer2_conv2(x)
        x  =  self.layer2_bn2(x)
        x  =  self.layer2_relu2(x)
        x  =  self.layer2_maxpool(x)

        x  =  self.layer3_conv1(x)
        x  =  self.layer3_bn1(x)
        x  =  self.layer3_relu1(x)
        x  =  self.layer3_conv2(x)
        x  =  self.layer3_bn2(x)
        x  =  self.layer3_relu2(x)
        x  =  self.layer3_conv3(x)
        x  =  self.layer3_bn3(x)
        x  =  self.layer3_relu3(x)
        x  =  self.layer3_maxpool(x)

        x  =  self.layer4_conv1(x)
        x  =  self.layer4_bn1(x)
        x  =  self.layer4_relu1(x)
        x  =  self.layer4_conv2(x)
        x  =  self.layer4_bn2(x)
        x  =  self.layer4_relu2(x)
        x  =  self.layer4_conv3(x)
        x  =  self.layer4_bn3(x)
        x  =  self.layer4_relu3(x)
        x  =  self.layer4_maxpool(x)

        x  =  self.layer5_conv1(x)
        x  =  self.layer5_bn1(x)
        x  =  self.layer5_relu1(x)
        x  =  self.layer5_conv2(x)
        x  =  self.layer5_bn2(x)
```

```
        x  =  self.layer5_relu2(x)
        x  =  self.layer5_conv3(x)
        x  =  self.layer5_bn3(x)
        x  =  self.layer5_relu3(x)
        x  =  self.layer5_conv4(x)
        x  =  self.layer5_bn4(x)
        x  =  self.layer5_relu4(x)
        x  =  self.layer5_maxpool(x)

        x = self.flatten(x)
        x = self.fullyconnect1(x)
        x = self.relu_1(x)
        x = self.dropout_1(x)
        x = self.fullyconnect2(x)
        x = self.relu_2(x)
        x = self.dropout_1(x)
        x = self.fullyconnect3(x)

        return x
```

代码 14: 实现模型训练函数

```
# Copyright 2020 Huawei Technologies Co., Ltd
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ============================================================================
"""
##################train vgg17 example on flowerphotos########################
"""
import datetime
import os
import time
import random
import numpy
import mindspore
import mindspore.nn as nn
from mindspore import Tensor
```

```
from mindspore import context
from mindspore.communication.management import init, get_rank, get_group_size
from mindspore.nn import Momentum
from mindspore.nn import Accuracy
from mindspore.train import ModelCheckpoint, CheckpointConfig, LossMonitor,
    TimeMonitor,Callback
from mindspore.train import Model
from mindspore import ParallelMode
from mindspore import load_param_into_net, load_checkpoint
from mindspore.amp import FixedLossScaleManager
from mindspore import set_seed
from src.dataset import vgg_create_dataset
from src.dataset import classification_dataset

from src.crossentropy import CrossEntropy
from src.warmup_step_lr import warmup_step_lr
from src.warmup_cosine_annealing_lr import warmup_cosine_annealing_lr
from src.warmup_step_lr import lr_steps
from src.utils.logging import get_logger
from src.utils.util import get_param_groups
from src.vgg import vgg17

import sys
# Remove the '-f' flag and the following string if present in the command line
    arguments
removed_args = []
if '-f' in sys.argv:
    index = sys.argv.index('-f')
    removed_args.append(sys.argv.pop(index))  # Remove '-f' and store it
    if index < len(sys.argv):  # Check if there is an argument after '-f'
        removed_args.append(sys.argv.pop(index))  # Remove the argument after '-
            f' and store it

from model_utils.moxing_adapter import config
from model_utils.moxing_adapter import moxing_wrapper
from model_utils.device_adapter import get_device_id, get_rank_id,
    get_device_num

import sys
f = open('train.log', 'w')
sys.stdout = f
sys.stderr = f

#modelarts预处理部分
def modelarts_pre_process():
    sum=1
    '''modelarts pre process function.'''
```

```
69      def unzip(zip_file, save_dir):
70          import zipfile
71          s_time = time.time()
72          if not os.path.exists(os.path.join(save_dir, config.
                modelarts_dataset_unzip_name)):
73              zip_isexist = zipfile.is_zipfile(zip_file)
74              if zip_isexist:
75                  fz = zipfile.ZipFile(zip_file, 'r')
76                  data_num = len(fz.namelist())
77                  print("Extract Start...")
78                  print("unzip file num: {}".format(data_num))
79                  data_print = int(data_num / 100) if data_num > 100 else 1
80                  i = 0
81                  for file in fz.namelist():
82                      if i % data_print == 0:
83                          print("unzip percent: {}%".format(int(i * 100 / data_num
                                )), flush=True)
84                      i += 1
85                      fz.extract(file, save_dir)
86                  print("cost time: {}min:{}s.".format(int((time.time() - s_time)
                        / 60),
87                                                        int(int(time.time() -
                                                            s_time) % 60)))
88                  print("Extract Done.")
89              else:
90                  print("This is not zip.")
91          else:
92              print("Zip has been extracted.")
93
94      if config.need_modelarts_dataset_unzip:
95          zip_file_1 = os.path.join(config.data_path, config.
                modelarts_dataset_unzip_name + ".zip")
96          save_dir_1 = os.path.join(config.data_path)
97
98          sync_lock = "/tmp/unzip_sync.lock"
99
100         # Each server contains 8 devices as most.
101         if config.device_target == "GPU":
102             init()
103             device_id = get_rank()
104             device_num = get_group_size()
105         elif config.device_target == "Ascend":
106             device_id = get_device_id()
107             device_num = get_device_num()
108         else:
109             raise ValueError("Not support device_target.")
110
```

```
        if device_id % min(device_num, 8) == 0 and not os.path.exists(sync_lock)
            :
            print("Zip file path: ", zip_file_1)
            print("Unzip file save dir: ", save_dir_1)
            unzip(zip_file_1, save_dir_1)
            print("===Finish extract data synchronization===")
            try:
                os.mknod(sync_lock)
            except IOError:
                pass

        while True:
            if os.path.exists(sync_lock):
                break
            time.sleep(1)

        print("Device: {}, Finish sync unzip data from {} to {}.".format(
            device_id, zip_file_1, save_dir_1))

    config.ckpt_path = os.path.join(config.output_path, config.ckpt_path)
#训练部分
@moxing_wrapper(pre_process=modelarts_pre_process)
def run_train():
    '''run train'''
    config.lr_epochs = list(map(int, config.lr_epochs.split(',')))
    config.image_size = list(map(int, config.image_size.split(',')))
    config.per_batch_size = config.batch_size

    _enable_graph_kernel = (config.device_target == "GPU")
    context.set_context(mode=context.PYNATIVE_MODE,
                        enable_graph_kernel=_enable_graph_kernel, device_target=
                            config.device_target)
    config.rank = get_rank_id()
    config.device_id = get_device_id()
    config.group_size = get_device_num()

    if config.is_distributed:
        if config.device_target == "Ascend":
            init()
            context.set_context(device_id=config.device_id)
        elif config.device_target == "GPU":
            if not config.enable_modelarts:
                init()
            else:
                if not config.need_modelarts_dataset_unzip:
                    init()

```

```
        device_num = config.group_size
        context.reset_auto_parallel_context()
        context.set_auto_parallel_context(device_num=device_num, parallel_mode=
            ParallelMode.DATA_PARALLEL,
                                          gradients_mean=True,
                                              all_reduce_fusion_config=[15, 18])

    else:
        if config.device_target == "Ascend":
            if context.get_context('device_id')!=config.device_id:
                context.set_context(device_id=config.device_id)


    # select for master rank save ckpt or all rank save, compatible for model
        parallel
    config.rank_save_ckpt_flag = 0
    if config.is_save_on_master:
        if config.rank == 0:
            config.rank_save_ckpt_flag = 1
    else:
        config.rank_save_ckpt_flag = 1


    # logger
    config.outputs_dir = os.path.join(config.ckpt_path,
                                      datetime.datetime.now().strftime('%Y-%m-%
                                          d_time_%H_%M_%S'))
    config.logger = get_logger(config.outputs_dir, config.rank)

    if config.dataset == "flower_photos":
        dataset = vgg_create_dataset(config.data_dir, config.image_size, config.
            per_batch_size,
                                     config.rank, config.group_size)
        eval_dataset = vgg_create_dataset(config.data_dir, config.image_size,
            config.per_batch_size,
                                     config.rank, config.group_size)

    batch_num = dataset.get_dataset_size()
    config.steps_per_epoch = dataset.get_dataset_size()
    config.logger.save_args(config)

    # network
    config.logger.important_info('start create network')

    # 构建网络
    network = vgg17(config.num_classes, config)
    network.set_train(True)
```

```
196
197     # 是否有预训练权重文件
198     if config.pre_trained:
199         load_param_into_net(network, load_checkpoint(config.pre_trained))
200
201     # 学习率
202     if config.lr_scheduler == 'exponential':
203         lr = warmup_step_lr(config.lr,
204                             config.lr_epochs,
205                             config.steps_per_epoch,
206                             config.warmup_epochs,
207                             config.max_epoch,
208                             gamma=config.lr_gamma,
209                             )
210     elif config.lr_scheduler == 'cosine_annealing':
211         lr = warmup_cosine_annealing_lr(config.lr,
212                                         config.steps_per_epoch,
213                                         config.warmup_epochs,
214                                         config.max_epoch,
215                                         config.T_max,
216                                         config.eta_min)
217     elif config.lr_scheduler == 'step':
218         lr = lr_steps(0, lr_init=config.lr_init, lr_max=config.lr_max,
            warmup_epochs=config.warmup_epochs,
219                       total_epochs=config.max_epoch, steps_per_epoch=batch_num)
220     else:
221         raise NotImplementedError(config.lr_scheduler)
222
223     # 优化器
224     opt = Momentum(params=get_param_groups(network),
225                    learning_rate=Tensor(lr),
226                    momentum=config.momentum,
227                    weight_decay=config.weight_decay,
228                    loss_scale=config.loss_scale)
229
230
231     loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
232     model = Model(network, loss_fn=loss, optimizer=opt, metrics={"Accuracy":
        Accuracy()},
233                   amp_level="O2", keep_batchnorm_fp32=False,
                          loss_scale_manager=None)
234
235     # 定义回调函数
236     time_cb = TimeMonitor(data_size=batch_num)
237     loss_cb = LossMonitor()
238     #epoch_per_eval = {"epoch": [], "acc": []}
239     #eval_cb = EvalCallBack(model, eval_dataset, 1, epoch_per_eval)  #每个epoch
```

```
        都评估一下
    callbacks = [time_cb, loss_cb]
    if config.rank_save_ckpt_flag:
        ckpt_config = CheckpointConfig(save_checkpoint_steps=config.
            ckpt_interval * config.steps_per_epoch,
                                       keep_checkpoint_max=config.
                                           keep_checkpoint_max)
        save_ckpt_path = os.path.join(config.outputs_dir, 'ckpt_' + str(config.
            rank) + '/')
        print(save_ckpt_path)
        ckpt_cb = ModelCheckpoint(config=ckpt_config,
                                  directory=save_ckpt_path,
                                  prefix='{}'.format(config.rank))
        callbacks.append(ckpt_cb)

    #进行模型训练
    model.train(config.max_epoch, dataset, callbacks=callbacks)

sys.argv.extend(removed_args)
```

```
epoch: 25 step: 31, loss is 0.8279935121536255
epoch: 25 step: 32, loss is 0.854680597782135
epoch: 25 step: 33, loss is 0.8719025254249573
epoch: 25 step: 34, loss is 1.0060001611709595
epoch: 25 step: 35, loss is 0.8856016397476196
epoch: 25 step: 36, loss is 0.8880851864814758
epoch: 25 step: 37, loss is 0.7869280576705933
epoch: 25 step: 38, loss is 0.8051539659500122
epoch: 25 step: 39, loss is 0.7888650894165039
epoch: 25 step: 40, loss is 0.8117711544036865
epoch: 25 step: 41, loss is 0.8676080703735352
epoch: 25 step: 42, loss is 0.8156512379646301
epoch: 25 step: 43, loss is 0.8273199796676636
epoch: 25 step: 44, loss is 0.9930800795555115
epoch: 25 step: 45, loss is 1.0492265224456787
Train epoch time: 9184.325 ms, per step time: 204.096 ms
```

图 4: 训练结果图片

2.3 模型训练

代码 15: 模型训练

```
config.config_path="flowerphotos_config.yaml" # 此yaml文件中配置了VGG16的训练参
    数和评估参数。可自行尝试调整lr，batch_size，max_epoch等参数，提高模型精度。
config.dataset="flower_photos"
config.is_distributed=0
config.data_dir="./data"
config.device_target="Ascend"
config.lr_epochs='30,60,90,120'
config.image_size="224,224"
```

```

run_train()
```

2.4 模型评估

代码 16: 模型评估函数

```
# Copyright 2020 Huawei Technologies Co., Ltd
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ============================================================================
"""Eval"""
import os
import time
import datetime
import random
import mindspore
import glob
import numpy as np
import mindspore.nn as nn

from mindspore import Tensor, context
from mindspore.communication import init, get_rank, get_group_size
from mindspore.train import Model
from mindspore import load_checkpoint, load_param_into_net
import mindspore.ops as P
from mindspore import dtype as mstype

from src.utils.logging import get_logger
from src.vgg import vgg17
from src.dataset import vgg_create_dataset
from src.dataset import classification_dataset

from model_utils.moxing_adapter import config
from model_utils.moxing_adapter import moxing_wrapper
from model_utils.device_adapter import get_device_id, get_rank_id,
    get_device_num
```

```
40
41 import sys
42 f = open('eval.log', 'w')
43 sys.stdout = f
44 sys.stderr = f
45
46 class ParameterReduce(nn.Cell):
47     """ParameterReduce"""
48     def __init__(self):
49         super(ParameterReduce, self).__init__()
50         self.cast = P.Cast()
51         self.reduce = P.AllReduce()
52
53     def construct(self, x):
54         one = self.cast(P.scalar_to_tensor(1.0), mstype.float32)[0]
55         out = x * one
56         ret = self.reduce(out)
57         return ret
58
59
60 def get_top5_acc(top5_arg, gt_class):
61     sub_count = 0
62     for top5, gt in zip(top5_arg, gt_class):
63         if gt in top5:
64             sub_count += 1
65     return sub_count
66
67
68 def modelarts_pre_process():
69     '''modelarts pre process function.'''
70     def unzip(zip_file, save_dir):
71         import zipfile
72         s_time = time.time()
73         if not os.path.exists(os.path.join(save_dir, config.
               modelarts_dataset_unzip_name)):
74             zip_isexist = zipfile.is_zipfile(zip_file)
75             if zip_isexist:
76                 fz = zipfile.ZipFile(zip_file, 'r')
77                 data_num = len(fz.namelist())
78                 print("Extract Start...")
79                 print("unzip file num: {}".format(data_num))
80                 data_print = int(data_num / 100) if data_num > 100 else 1
81                 i = 0
82                 for file in fz.namelist():
83                     if i % data_print == 0:
84                         print("unzip percent: {}%".format(int(i * 100 / data_num
                               )), flush=True)
```

```
                    i += 1
                    fz.extract(file, save_dir)
                print("cost time: {}min:{}s.".format(int((time.time() - s_time)
                    / 60),int(int(time.time() - s_time) % 60)))
                print("Extract Done.")
            else:
                print("This is not zip.")
        else:
            print("Zip has been extracted.")

    if config.need_modelarts_dataset_unzip:
        zip_file_1 = os.path.join(config.data_path, config.
            modelarts_dataset_unzip_name + ".zip")
        save_dir_1 = os.path.join(config.data_path)

        sync_lock = "/tmp/unzip_sync.lock"

        # Each server contains 8 devices as most.
        if config.device_target == "GPU":
            init()
            device_id = get_rank()
            device_num = get_group_size()
        elif config.device_target == "Ascend":
            device_id = get_device_id()
            device_num = get_device_num()
        else:
            raise ValueError("Not support device_target.")

        # Each server contains 8 devices as most.
        if device_id % min(device_num, 8) == 0 and not os.path.exists(sync_lock)
            :
            print("Zip file path: ", zip_file_1)
            print("Unzip file save dir: ", save_dir_1)
            unzip(zip_file_1, save_dir_1)
            print("===Finish extract data synchronization===")
            try:
                os.mknod(sync_lock)
            except IOError:
                pass

        while True:
            if os.path.exists(sync_lock):
                break
            time.sleep(1)

        print("Device: {}, Finish sync unzip data from {} to {}.".format(
            device_id, zip_file_1, save_dir_1))
```

```
128
129     config.log_path = os.path.join(config.output_path, config.log_path)
130
131
132 @moxing_wrapper(pre_process=modelarts_pre_process)
133 def run_eval():
134     """run eval"""
135     config.per_batch_size = config.batch_size
136     config.image_size = list(map(int, config.image_size.split(',')))
137     config.rank = get_rank_id()
138     config.group_size = get_device_num()
139
140
141     _enable_graph_kernel = config.device_target == "GPU"
142     context.set_context(mode=context.GRAPH_MODE, enable_graph_kernel=False,
143                         device_target=config.device_target, save_graphs=False)
144     if os.getenv('DEVICE_ID', "not_set").isdigit() and config.device_target == "
            Ascend":
145         if context.get_context('device_id')!=int(os.getenv('DEVICE_ID')):
146             context.set_context(device_id=int(os.getenv('DEVICE_ID')))
147
148     config.outputs_dir = os.path.join(config.log_path,
149                                       datetime.datetime.now().strftime('%Y-%m-%
                                              d_time_%H_%M_%S'))
150
151     config.logger = get_logger(config.outputs_dir, config.rank)
152     config.logger.save_args(config)
153
154     if config.dataset == "flower_photos":
155         net = vgg17(num_classes=config.num_classes, args=config,phase="test")
156         loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
157         model = Model(net, loss_fn=loss, metrics={'acc'})
158         param_dict = load_checkpoint(config.pre_trained)
159         load_param_into_net(net, param_dict)
160         net.set_train(False)
161         dataset = vgg_create_dataset(config.data_dir, config.image_size, config.
                per_batch_size, training=False)
162         res = model.eval(dataset)
163         print("result: ", res)
```

代码 17: 进行模型评估

```
1 config.pre_trained="./outputs_flowers/2024-05-04_time_00_31_28/ckpt_0/0-25_45.
      ckpt"
2 config.dataset="flower_photos"
3 config.image_size="224,224"
4 config.device_target="Ascend"
```

```
config.data_dir="./data"

run_eval()
```

使用测试数据集，使用 mindspore.Model.eval 接口进行评估，eval 接口参数如下：

valid_dataset：评估模型的数据集。

callbacks：评估过程中需要执行的回调对象或回调对象列表。

dataset_sink_mode：数据是否直接下沉至处理器进行处理。

model.eval 会返回一个字典，里面是传入 metrics 的指标和结果。

```
2024-05-04 00:38:23,777:INFO:--> dataset_name: flower_photos
2024-05-04 00:38:23,777:INFO:--> result_path: ./preprocess_Result/
2024-05-04 00:38:23,777:INFO:--> ckpt_file:
2024-05-04 00:38:23,777:INFO:--> file_name: vgg17
2024-05-04 00:38:23,777:INFO:--> file_format: MINDIR
2024-05-04 00:38:23,777:INFO:--> config_path: flowerphotos_config.yaml
2024-05-04 00:38:23,777:INFO:--> rank: 0
2024-05-04 00:38:23,777:INFO:--> device_id: 0
2024-05-04 00:38:23,778:INFO:--> group_size: 1
2024-05-04 00:38:23,778:INFO:--> rank_save_ckpt_flag: 1
2024-05-04 00:38:23,778:INFO:--> outputs_dir: ./outputs_flowers/2024-05-04_time_00_38_23
2024-05-04 00:38:23,778:INFO:--> logger: <LOGGER mindversion (NOTSET)>
2024-05-04 00:38:23,778:INFO:--> steps_per_epoch: 45
2024-05-04 00:38:23,778:INFO:
./data False
data_dir ./data/test
result:  {'acc': 0.6519886363636364}
```

图 5: 验证结果图片

2.5 模型导出

代码 18: 进行模型导出

```
if __name__ == '__main__':
    config.config_path = '../flowerphotos_config.yaml'
    config.ckpt_file = '../outputs_flowers/2024-05-04_time_00_31_28/ckpt_0/0-25
        _45.ckpt'
    config.file_name = 'flowervgg17'
    config.device_target = 'Ascend'
    config.file_format = 'MINDIR'
    run_export()
```

```
(MindSpore) [ma-user code]$python export.py
{'device_target': 'device where the code will be implemented.', 'dataset': 'flower_photos', 'data_dir': 'data dir',
 'pre_trained': 'model_path, local pretrained model to load', 'lr_gamma': 'decrease lr by a factor of exponential l
r_scheduler', 'eta_min': 'eta_min in cosine_annealing scheduler', 'T_max': 'T-max in cosine_annealing scheduler', '
log_interval': 'logging interval', 'ckpt_path': 'checkpoint save location', 'ckpt_interval': 'ckpt_interval', 'is_s
ave_on_master': 'save ckpt on master or all rank', 'is_distributed': 'if multi device', 'per_batch_size': 'batch si
ze for per npu', 'graph_ckpt': 'graph ckpt or feed ckpt', 'log_path': 'path to save log', 'result_dir': 'result fil
es path.', 'label_dir': 'image file path.', 'dataset_name': 'flower_photos', 'result_path': 'result path', 'ckpt_fi
le': 'vgg17 ckpt file.', 'file_name': 'vgg17 output file name.', 'file_format': "file format, choices in ['AIR', 'O
NNX', 'MINDIR']"}
(MindSpore) [ma-user code]$
```

图 6: 运行导出程序输出结果

3 实验结果分析

完成实验后路径中包含的文件信息：

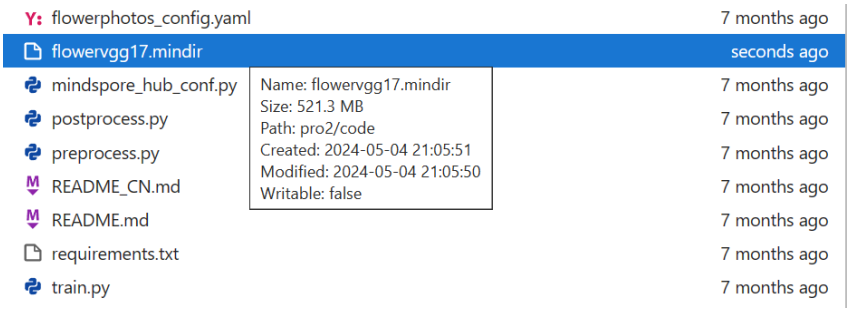

图 7: 导出的模型

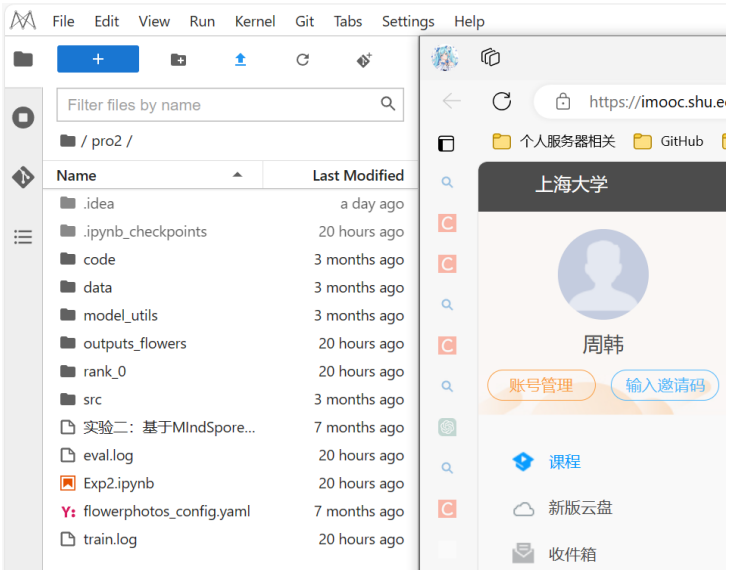

图 8: 实验二文件路径

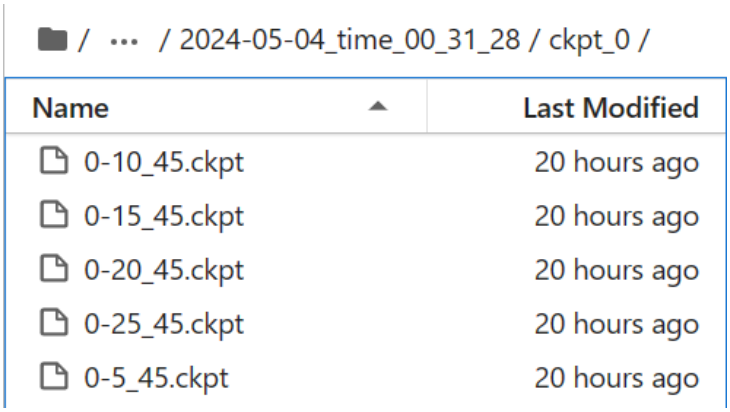

图 9: 保存点

4 思考题

4.1 不同训练速度

通过调整 run_train 函数中 Model 函数的参数 amp_level 和 keep_batchnorm_fp32，来设置不同的精度进行训练，主要包括：单精度（FP32）、半精度（FP16）以及混合精度。训练完成后，生成不同的 train.log 文件，整合 log 文件中的内容得到下表所示的结果（以第 25 训练周期时间为例）：

表 2: 使用不同精度训练

amp_level	keep_batchnorm_fp32	训练类型	Train epoch time	per step time
O0	True	仅使用 FP32	15035.235 ms	323.580 ms
O1	False	部分 FP16 混合	9754.128 ms	211.113 ms
O2	False	除了 BN 层外，使用 FP16 训练	8893.034 ms	176.987 ms
O3	False	使用 FP16 训练	8556.495 ms	180.243 ms

使用 FP16 和 FP32 混合精度训练，可以减少训练时所需要的时间和需要的内存的大小。FP16 占用的内存只需要原来的一半，可以让训练的时候开启更大的 batchsize。在多卡训练的时候可以大幅减少卡与卡之间的通信量，加快数据流通。

4.2 提升模型训练速度方法

1. **优化使用硬件**：使用高性能处理单元：使用 GPU 或者 TPU，GPU 和 TPU 可以显著加快大矩阵计算。优化内存和存储：确保足够的 RAM 和快速的数据存储（如 SSD）可以减少数据加载的时间。

2. **改善数据预处理和传输**：优化数据预处理：在 CPU 上并行处理数据，确保不让 GPU 等待数据使用高效的数据格式：使用 RecordIO(MXNet) 等格式，加快数据的存储和读取。

3. **使用更快的优化器**：高效的优化器算法：使用 Adam 优化器可以让模型更快收敛，减少训练周期。调整超参：适当调整学习率、使用预热 epoches 加快模型收敛。

4. **分布式训练策略**：模型并行：当单个模型太大而不能在一个 GPU 上运行时，可以将模型的不同部分放在不同的设备上。使用高效的通信策略：如 NVIDIA 的 NCCL 库，优化 GPU 间的通信。

四 感想与体会

通过以上两个实验，我学习到了如何使用 Ascend 的计算卡进行深度学习，在国产化的潮流下，了解并使用国产的计算卡和相关的软件具有深刻的意义。将经典的深度学习模型移植到 ACL(Ascend Computing Language)，并搭建 VGG16 模型进行分类任务，让我了解并学习到除 Pytorch 和 Tensorflow 以外的深度学习模型。并在第二个实验使用 Mindspore 框架实现 VGG17，并使用 Ascend 910 卡进行训练，最后进行推理。完整学习了 Ascend 卡和 Mindspore 环境的使用，为我以后继续使用 Mindspore 环境打下了坚实的基础。