Mindspore花卉图像分类

实验目的

本实验为基于卷积神经网络实现的花卉识别实验，与传统图像分类方法不同，卷积神经网络无需人工提取特征，可以根据输入图像，自动学习包含丰富语义信息的特征，得到更为全面的花卉图像特征描述，可以很好地表达图像的不同类别信息。

本实验在华为云ModelArt平台实现，notebook所采用的镜像为

`tensorflow1.15-mindspore1.7.0-cann5.1.0-euler2.8-aarch64`

整个实验的体系结构可以划分为三部分，分别为模型训练、模型保存和模型推理。具体的实验步骤如下所示

- 导入实验环境；
- 数据集获取与预处理；
- 构建CNN图像识别模型；
- 图像分类模型验证；

1.导入相应的模块并定义变量与超参数

```
#easydict模块用于以属性的方式访问字典的值
from easydict import EasyDict as edict
#glob模块主要用于查找符合特定规则的文件路径名，类似使用windows下的文件搜索
import glob
#os模块主要用于处理文件和目录
import os

import numpy as np
import matplotlib.pyplot as plt

import mindspore
#导入mindspore框架数据集
import mindspore.dataset as ds
#vision.c_transforms模块是处理图像增强的高性能模块，用于数据增强图像数据改进训练模型。
import mindspore.dataset.vision.c_transforms as CV
#c_transforms模块提供常用操作，包括OneHotOp和TypeCast
import mindspore.dataset.transforms.c_transforms as C
from mindspore.common import dtype as mstype
```

```
from mindspore import context
#导入模块用于初始化截断正态分布
from mindspore.common.initializer import TruncatedNormal
from mindspore import nn
from mindspore.train import Model
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMoni
from mindspore.train.serialization import load_checkpoint, load_param_into_net
from mindspore import Tensor
# 设置MindSpore的执行模式和设备
context.set_context(mode=context.GRAPH_MODE, device_target="Ascend")

#定义变量
cfg = edict({
    'data_path': 'flower_photos',
    'data_size':3670,
    'image_width': 100,  # 图片宽度
    'image_height': 100,  # 图片高度
    'batch_size': 32,
    'channel': 3,  # 图片通道数
    'num_class':5,  # 分类类别
    'weight_decay': 0.01,
    'lr':0.0001,  # 学习率
    'dropout_ratio': 0.5,
    'epoch_size': 250,  # 训练次数
    'sigma':0.01,

    'save_checkpoint_steps': 1,  # 多少步保存一次模型
    'keep_checkpoint_max': 1,  # 最多保存多少个模型
    'output_directory': './',  # 保存模型路径
    'output_prefix': "checkpoint_classification"  # 保存模型文件名字
})
```

2.数据集获取与预处理

该数据集是开源数据集，总共包括5种花的类型：分别是daisy（雏菊，633张），dandelion（蒲公英，898张），roses（玫瑰，641张），sunflowers（向日葵，699张），tulips（郁金香，799张），保存在5个文件夹当中，总共3670张，大小大概在230M左右。为了在模型部署上线之后进行测试，数据集在这里分成了flower_train和flower_test两部分。

数据读取并处理流程如下：

- MindSpore的mindspore.dataset提供了ImageFolderDatasetV2函数，可以直接读取文件夹图片数据并映射文件夹名字为其标签(label)。这里我们使用ImageFolderDatasetV2函数读取'daisy','dandelion','roses','sunflowers','tulips'数据。并将这五类标签映射为：{'daisy':0,'dandelion':1,'roses':2,'sunflowers':3,'tulips':4}

- 使用RandomCropDecodeResize、HWC2CHW、TypeCast、shuffle进行数据预处理

步骤一：获取数据集

```
# 解压数据集，只需要第一次运行时解压，第二次无需再解压
!wget https://ascend-professional-construction-dataset.obs.myhuaweicloud.com/dee
!unzip flower_photos.zip
```

步骤二：数据预处理

```
#读取图像的源数据集。
de_dataset = ds.ImageFolderDataset(cfg.data_path,
                                   class_indexing={'daisy':0,'dandelion':1,'rose
#解码前将输入图像裁剪成任意大小和宽高比。
transform_img = CV.RandomCropDecodeResize([cfg.image_width,cfg.image_height], sc
#转换输入图像；形状（H，W，C）为形状（C，H，W）。
hwc2chw_op = CV.HWC2CHW()
#转换为给定MindSpore数据类型的Tensor操作。
type_cast_op = C.TypeCast(mstype.float32)
#将操作中的每个操作应用到此数据集。
de_dataset = de_dataset.map(input_columns="image", num_parallel_workers=8, opera
de_dataset = de_dataset.map(input_columns="image", operations=hwc2chw_op, num_pa
de_dataset = de_dataset.map(input_columns="image", operations=type_cast_op, num_
de_dataset = de_dataset.shuffle(buffer_size=cfg.data_size)
```

步骤三：划分训练集与测试集

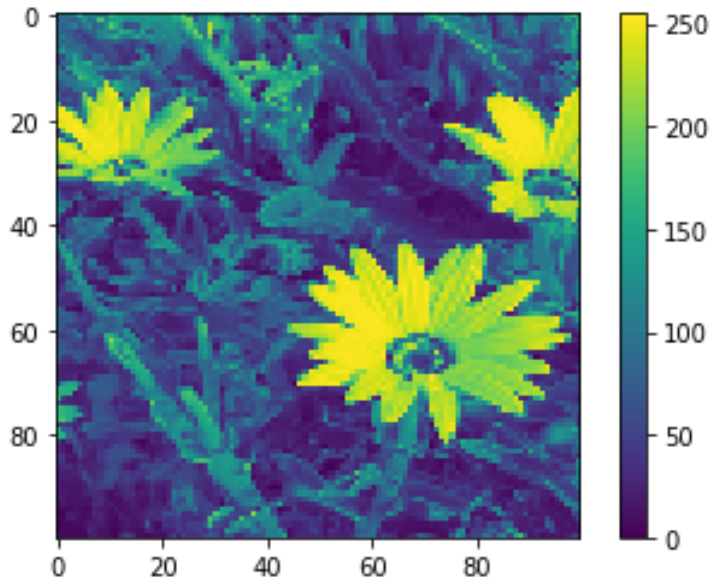
- 按照8:2的比列将数据划分为训练数据集和测试数据集
- 对训练数据和测试数据分批次（batch）

```
#划分训练集测试集
(de_train,de_test)=de_dataset.split([0.8,0.2])
#设置每个批处理的行数
#drop_remainder确定是否删除最后一个可能不完整的批（default=False）。
#如果为True，并且如果可用于生成最后一个批的batch_size行小于batch_size行，则这些行将被删
de_train=de_train.batch(cfg.batch_size, drop_remainder=True)
#重复此数据集计数次数。
de_test=de_test.batch(cfg.batch_size, drop_remainder=True)
print('训练数据集数量：',de_train.get_dataset_size()*cfg.batch_size)#get_dataset_s
print('测试数据集数量：',de_test.get_dataset_size()*cfg.batch_size)

data_next=de_dataset.create_dict_iterator(output_numpy=True).__next__()
```

```
print('通道数/图像长/宽：', data_next['image'].shape)
print('一张图像的标签样式：', data_next['label'])  # 一共5类，用0-4的数字表达类别。

plt.figure()
plt.imshow(data_next['image'][0,...])
plt.colorbar()
plt.grid(False)
plt.show()
```

随机采样进行单个图像及其参数的输出结果如下

```
训练数据集数量： 2912
测试数据集数量： 704
通道数/图像长/宽： (3, 100, 100)
一张图像的标签样式： 0
```



3.构建CNN图像识别模型

花卉图像数据集准备完成，接下来我们就需要构建训练模型，本实验采用的是CNN神经网络算法。

步骤一：定义图像识别模型模型参考资料

```
# 定义CNN图像识别网络
class Identification_Net(nn.Cell):
    def __init__(self, num_class=5,channel=3,dropout_ratio=0.5,trun_sigma=0.01):
        super(Identification_Net, self).__init__()
        self.num_class = num_class
        self.channel = channel
        self.dropout_ratio = dropout_ratio
        #设置卷积层
        self.conv1 = nn.Conv2d(self.channel, 32,
                               kernel_size=5, stride=1, padding=0,
```

```
                                has_bias=True, pad_mode="same",
                                weight_init=TruncatedNormal(sigma=trun_sigma),bia
        #设置ReLU激活函数
        self.relu = nn.ReLU()
        #设置最大池化层
        self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2,pad_mode="valid")
        self.conv2 = nn.Conv2d(32, 64,
                                kernel_size=5, stride=1, padding=0,
                                has_bias=True, pad_mode="same",
                                weight_init=TruncatedNormal(sigma=trun_sigma),bia
        self.conv3 = nn.Conv2d(64, 128,
                                kernel_size=3, stride=1, padding=0,
                                has_bias=True, pad_mode="same",
                                weight_init=TruncatedNormal(sigma=trun_sigma),bia
        self.conv4 = nn.Conv2d(128, 128,
                                kernel_size=3, stride=1, padding=0,
                                has_bias=True, pad_mode="same",
                                weight_init=TruncatedNormal(sigma=trun_sigma), bi
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(6*6*128, 1024,weight_init =TruncatedNormal(sigma=tru
        self.dropout = nn.Dropout(self.dropout_ratio)
        self.fc2 = nn.Dense(1024, 512, weight_init=TruncatedNormal(sigma=trun_si
        self.fc3 = nn.Dense(512, self.num_class, weight_init=TruncatedNormal(sig
    #构建模型
    def construct(self, x):
        x = self.conv1(x)
        #print(x.shape)
        x = self.relu(x)
        x = self.max_pool2d(x)
        x = self.conv2(x)
        x = self.relu(x)
        x = self.max_pool2d(x)
        x = self.conv3(x)
        x = self.max_pool2d(x)
        x = self.conv4(x)
        x = self.max_pool2d(x)
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.relu(x)
        #print(x.shape)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc3(x)
        return x
```

步骤二：模型训练、测试、预测

```
net=Identification_Net(num_class=cfg.num_class, channel=cfg.channel, dropout_rat
#计算softmax交叉熵。
net_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
#opt
fc_weight_params = list(filter(lambda x: 'fc' in x.name and 'weight' in x.name,
other_params=list(filter(lambda x: 'fc' not in x.name or 'weight' not in x.name,
group_params = [{'params': fc_weight_params, 'weight_decay': cfg.weight_decay},
                {'params': other_params},
                {'order_params': net.trainable_params()}]
#设置Adam优化器
net_opt = nn.Adam(group_params, learning_rate=cfg.lr, weight_decay=0.0)
#net_opt = nn.Adam(params=net.trainable_params(), learning_rate=cfg.lr, weight_d

model = Model(net, loss_fn=net_loss, optimizer=net_opt, metrics={"acc"})
loss_cb = LossMonitor(per_print_times=de_train.get_dataset_size()*10)
config_ck = CheckpointConfig(save_checkpoint_steps=cfg.save_checkpoint_steps,
                             keep_checkpoint_max=cfg.keep_checkpoint_max)
ckpoint_cb = ModelCheckpoint(prefix=cfg.output_prefix, directory=cfg.output_dire
print("============== Starting Training ==============")
model.train(cfg.epoch_size, de_train, callbacks=[loss_cb, ckpoint_cb], dataset_s

# 使用测试集评估模型，打印总体准确率
metric = model.eval(de_test)
print(metric)
```

输出训练结果如下，训练模型保存在当前目录下的`checkpoint_classification<>.ckpt`文件中

```
epoch: 300 step: 91, loss is 0.3231726884841919
epoch: 310 step: 91, loss is 0.4372059106826782
epoch: 320 step: 91, loss is 0.20163103938102722
epoch: 330 step: 91, loss is 0.2892054319381714
epoch: 340 step: 91, loss is 0.35935747623443604
epoch: 350 step: 91, loss is 0.2740333676338196
epoch: 360 step: 91, loss is 0.3626466393470764
epoch: 370 step: 91, loss is 0.09469565749168396
epoch: 380 step: 91, loss is 0.21999232470989227
epoch: 390 step: 91, loss is 0.35333311557769775
epoch: 400 step: 91, loss is 0.2996513843536377
{'acc': 0.9446022727272727}
```

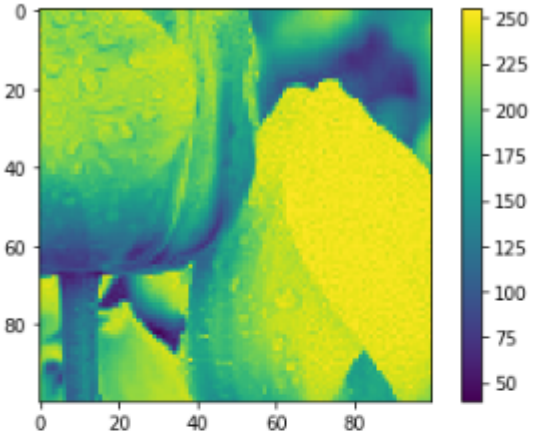
4.图像分类模型验证

- 验证之前训练出来的模型的性能。
- 掌握利图像识别模型验证方法。

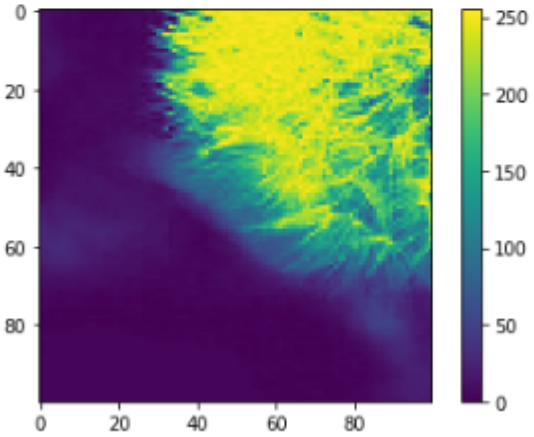
```
#加载模型
import os
CKPT = os.path.join(cfg.output_directory,cfg.output_prefix+'-'+str(cfg.epoch_siz
net = Identification_Net(num_class=cfg.num_class, channel=cfg.channel, dropout_r
load_checkpoint(CKPT, net=net)
model = Model(net)

# 预测
class_names = {0:'daisy',1:'dandelion',2:'roses',3:'sunflowers',4:'tulips'}

for i in range(5):
    data_next=de_dataset.create_dict_iterator(output_numpy=True).__next__()
    print('第'+str(i+1)+'张图像的真实标签为: ', data_next['label'])  # 一共5类，用0-
    plt.figure()
    plt.imshow(data_next['image'][0,...])
    plt.colorbar()
    plt.grid(False)
    plt.show()

    test = Tensor(data_next['image'], mindspore.float32)
    test=test.reshape(1,3,100,100)
    predictions = model.predict(test)
    predictions = predictions.asnumpy()

    #显示预测结果
    p_np = predictions
    pre_label = np.argmax(p_np)
    print('第' + str(i+1) + '个sample预测标签为: ',pre_label,'输出结果为: ', class_
```

验证结果如下:

第1张图像的真实标签为： 4



第1个sample预测标签为： 4 输出结果为： tulips
第2张图像的真实标签为： 1



第2个sample预测标签为： 1 输出结果为： dandelion

参考资料