**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**

|  |
| --- |
| JSON #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, LossMonitor, TimeMonitor 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

|  |
| --- |
| JSON # 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/deep-learning/flower\_photos.zip  !unzip flower\_photos.zip |

步骤二：数据预处理

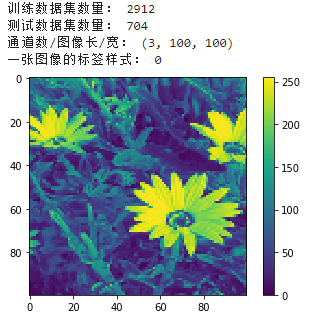
|  |
| --- |
| JSON #读取图像的源数据集。 de\_dataset = ds.ImageFolderDataset(cfg.data\_path,  class\_indexing={'daisy':0,'dandelion':1,'roses':2,'sunflowers':3,'tulips':4}) #解码前将输入图像裁剪成任意大小和宽高比。 transform\_img = CV.RandomCropDecodeResize([cfg.image\_width,cfg.image\_height], scale=(0.08, 1.0), ratio=(0.75, 1.333)) #改变尺寸 #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, operations=transform\_img) de\_dataset = de\_dataset.map(input\_columns="image", operations=hwc2chw\_op, num\_parallel\_workers=8) de\_dataset = de\_dataset.map(input\_columns="image", operations=type\_cast\_op, num\_parallel\_workers=8) 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）

|  |
| --- |
| JSON #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，则这些行将被删除，并且不会传播到子节点。 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\_size()获取批处理的大小。 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



**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。

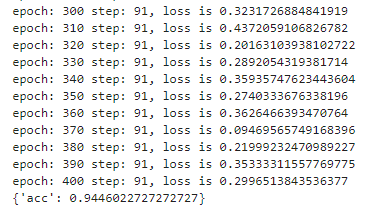
步骤一：定义图像识别模型模型参考资料

|  |
| --- |
| JSON # 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): # 一共分五类，图片通道数是3  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),bias\_init='zeros')  #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),bias\_init='zeros')  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),bias\_init='zeros')  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), bias\_init='zeros')  self.flatten = nn.Flatten()  self.fc1 = nn.Dense(6\*6\*128, 1024,weight\_init =TruncatedNormal(sigma=trun\_sigma),bias\_init = 0.1)  self.dropout = nn.Dropout(self.dropout\_ratio)  self.fc2 = nn.Dense(1024, 512, weight\_init=TruncatedNormal(sigma=trun\_sigma), bias\_init=0.1)  self.fc3 = nn.Dense(512, self.num\_class, weight\_init=TruncatedNormal(sigma=trun\_sigma), bias\_init=0.1)  #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

|  |
| --- |
| JSON net=Identification\_Net(num\_class=cfg.num\_class, channel=cfg.channel, dropout\_ratio=cfg.dropout\_ratio) #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, net.trainable\_params())) other\_params=list(filter(lambda x: 'fc' not in x.name or 'weight' not in x.name, net.trainable\_params())) 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\_decay=0.1)   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\_directory, config=config\_ck) print("============== Starting Training ==============") model.train(cfg.epoch\_size, de\_train, callbacks=[loss\_cb, ckpoint\_cb], dataset\_sink\_mode=True)   # 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

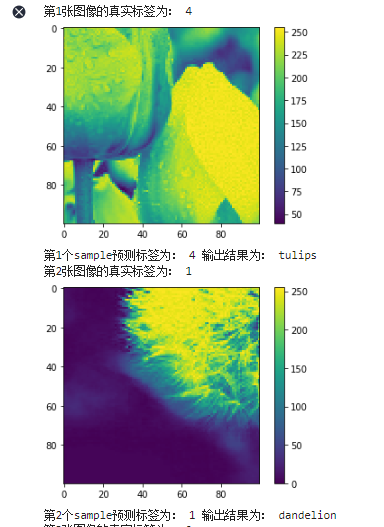


**4.图像分类模型验证**

* 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。

|  |
| --- |
| JSON #加载模型 import os CKPT = os.path.join(cfg.output\_directory,cfg.output\_prefix+'-'+str(cfg.epoch\_size)+'\_'+str(de\_train.get\_dataset\_size())+'.ckpt') net = Identification\_Net(num\_class=cfg.num\_class, channel=cfg.channel, dropout\_ratio=cfg.dropout\_ratio) 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0-4的数字表达类别。  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\_names[pre\_label]) |

验证结果如下：



[参考资料](https://www.hiascend.com/edu/courses)