学科实践实验报告

实验题目： **鲜花识别**

专　　业： **计算机科学与技术**

姓　　名： **吴雨娟**

学　　号： **22920192204097**

实验日期： 2021.5.9

一、 实验目的

鲜花种类繁多，不同种类的鲜花可能十分相似，难以对它们进行准确分类。本实验通过设计基于鲜花图像的分类方法，实现对鲜花的准确分类。通过鲜花识别案例，学会掌握如何用飞桨动态图搭建一个经典的卷积神经网络。

二、实验内容

根据课上所学内容，在 VGGNet 类中补全代码，构造 VGG 网络，保证程序跑通，并进行调优，使准确率尽可能高。

三、实验步骤以及结果

1.根据实验要求，先完成 VGG 网络的构造；

```
#定义CNN网络
class MyVGG(fluid.dygraph.Layer):
    '''
    VGG网络
    '''

    def __init__(self):
        super(MyVGG,self).__init__()
        self.convpool01=ConvPool(
            3,64,3,2,2,2,act="relu")
        self.convpool02=ConvPool(
            64,128,3,2,2,2,act="relu")
        self.convpool03=ConvPool(
            128,256,3,2,2,3,act="relu")
        self.convpool04=ConvPool(
            256,512,3,2,2,3,act="relu")
        self.convpool05=ConvPool(
            512,512,3,2,2,3,act="relu")
        self.pool_5_shape=512*7*7
        self.fc01=fluid.dygraph.Linear(self.pool_5_shape,4096,act="relu")
        self.fc02=fluid.dygraph.Linear(4096,4096,act="relu")
        self.fc03=fluid.dygraph.Linear(4096,train_parameters['class_dim'],act="softmax")
```

```
    def forward(self,inputs,label=None):
        # forward 定义执行实际运行时网络的执行逻辑
        out=self.convpool01(inputs)
        out=self.convpool02(out)
        out=self.convpool03(out)
        out=self.convpool04(out)
        out=self.convpool05(out)
        out=fluid.layers.reshape(out,shape=[-1,512*7*7])
        out=self.fc01(out)
        out=self.fc02(out)
        out=self.fc03(out)
        if label is not None:
            acc=fluid.layers.accuracy(input=out,label=label)
            return out,acc
        else:
            return out
```

结果：完成 VGG 网络构造后，尝试点击运行，发现解压有错误。

2. 在参数配置中修改原始数据集路径；

```
train_parameters = {
    "input_size": [3, 224, 224],                            #输入图片的shape
    "class_dim": -1,                                        #分类数
    "src_path":"data/data84753/flowers.zip",        #原始数据集路径
    "target_path":"/home/aistudio/data/dataset",        #要解压的路径
    "train_list_path": "./train_data.txt",              #train_data.txt路径
    "eval_list_path": "./val_data.txt",                  #eval_data.txt路径
    "test_list_path": "./test_data.txt",                   #test_data.txt路径
    "label_dict":{},                                        #标签字典
    "readme_path": "/home/aistudio/data/readme.json",    #readme.json路径
    "num_epochs": 65,                                       #训练轮数
    "train_batch_size": 64,                                 #批次的大小
    "learning_strategy": {                                  #优化函数相关的配置
        "lr": 0.001                                         #超参数学习率
    }
}
```

结果：发现还是会报错，继续排查。

3. 修改 get_data_list 函数里的保存文件夹名称，在路径后面加上’/flowers’；

```
def get_data_list(target_path,train_list_path,eval_list_path):
    '''
    生成数据列表
    '''
    #存放所有类别的信息
    class_detail = []
    #获取所有类别保存的文件夹名称
    data_list_path=target_path+'/flowers'
    class_dirs = os.listdir(data_list_path)
```

结果：经过调试，可以正常解压。

4.注释掉无关代码；

```
#with open('/home/aistudio/data/dataset/train/Almandine/rsi_2_almandine_21.jpg',
#        lines = [line.strip() for line in f]
```

结果：这块代码会导致程序无法继续往下运行，经检查确定可以删去。

5.修改模型评估代码块的模型名称，修改成’MyVGG’；

```
#模型评估
with fluid.dygraph.guard():
    accs = []
    model_dict, _ = fluid.load_dygraph('MyVGG')
    model = MyVGG()
```

结果：可以开始训练模型。

6.在训练模型的过程中，发现有的图片找不到，会报错。对 data_reader 函数的代码进行修正，用 try 方法尝试打开图片；

```python
def data_reader(file_list):
    '''
    自定义data_reader
    '''
    def reader():
        with open(file_list, 'r') as f:
            lines = [line.strip() for line in f]
            for line in lines:
                img_path, lab = line.strip().split('\t')
                try:
                    img = Image.open(img_path)
                    if img.mode != 'RGB':
                        img = img.convert('RGB')
                    img = img.resize((224, 224), Image.BILINEAR)
                    img = np.array(img).astype('float32')
                    img = img.transpose((2, 0, 1))  # HWC to CHW
                    img = img/255                   # 像素值归一化
                    yield img, int(lab)
                except(OSError,NameError):
                    print('')
    return reader
```

结果：解决了图片无法打开的问题。

7. 做了以上的修改后，程序已经可以跑通，但是按照初始参数跑出来的准确率并不高，不到 30%，所以要进行调优。首先进行数据增广，注意增广函数里打开图片也要用 try 的方式。

```
def data_enhance(target_path):
    target_path=target_path+'/flowers'
    i= 0
    dirs=os.listdir(target_path)
    # print(dirs)
    if '__MACOSX' in dirs:
        dirs.remove('__MACOSX')
    for sdir in dirs:
        rootdir=target_path+'/'+sdir
        # print(rootdir)
        # continue
        for parent, dirnames, filenames in os.walk(rootdir):
            #print('parent is :'+parent)
            for filename in filenames:
                if filename=='.DS_Store':
                    continue
                i=i+1
                currentPath=os.path.join(parent, filename)
                # print(currentPath)
                # continue
                if 'rsi' not in currentPath:
                    try:

                        img=Image.open(currentPath)
                        #if img.mode == "P":
                        img = img.convert('RGB')
```

```
                        random_factor=np.random.randint(0,31)/10. # 随机因子
                        color_image = ImageEnhance.Color(img).enhance(random_factor) # 调整图像的饱和度
                        random_factor = np.random.randint(10, 21)/ 10.  # 随机因子
                        brightness_image = ImageEnhance.Brightness(color_image).enhance(random_factor)# 调整图像的亮度
#
                        random_factor = np.random.randint(10,21)/10.  # 随机因子
                        contrast_image = ImageEnhance.Contrast(brightness_image).enhance(random_factor)# 调整图像对比度
                        random_factor = np.random.randint(0, 31) / 10. # 随机因子
                        sharpness_image = ImageEnhance.Sharpness(contrast_image).enhance(random_factor)
                        out = sharpness_image
                        #out=out.convert('RGB')
                        newname = rootdir + "/rsi_2_" + filename
                        if not os.path.exists(newname):
                            #print(str(i)+'fulll name :'+newname)
                            out.save(newname)
                    except(OSError,NameError):
                        print('')
print('ok')
```

在参数初始化操作中加入数据增广操作；

```
#生成数据列表
get_data_list(target_path,train_list_path,eval_list_path)
data_enhance(target_path)
```

结果：运行所得准确率与数据增广前相差不多。

8.数据增广后准确率提升不大，可能与参数有关。根据经验先把批次的大小改成 64;

```
"train_batch_size": 64,                                    #批次的大小
```

结果：运行后的准确率达到了 90%以上。

9.继续修改参数，把训练轮数调大；

```
"num_epochs": 65,                                          #训练轮数
```

结果：准确率逐渐提高，训练轮数为 65 时，准确率达到 0.99937993。

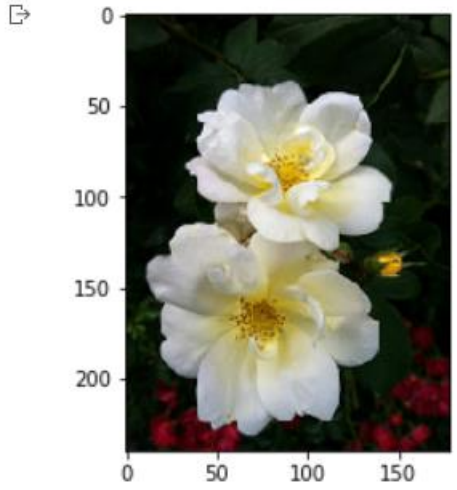
```
[14]
    model.eval() #训练模式
    for batch_id,data in enumerate(train_reader()):#测试
        images = np.array([x[0] for x in data]).astype('
        labels = np.array([x[1] for x in data]).astype('
        labels = labels[:, np.newaxis]
        image=fluid.dygraph.to_variable(images)
        label=fluid.dygraph.to_variable(labels)
        predict=model(image)
        acc=fluid.layers.accuracy(predict,label)
        accs.append(acc.numpy()[0])
    avg_acc = np.mean(accs)
    print(avg_acc)
```

```
0.99937993
```

10.得到足够高的准确率后，发现模型预测的代码块报错，需要进行修改。把模型名称修改为’MyVGG’,并修改预测图片的路径。最后还要修改真实标签的格式，因为图片名称不含有类别，所以真实标签要从类别文件夹名称中寻找。

```
模型预测
'''
with fluid.dygraph.guard():
    model_dict, _ = fluid.load_dygraph('MyVGG')
    model = MyVGG()
    model.load_dict(model_dict) #加载模型参数
    model.eval() #训练模式

    #展示预测图片
    infer_path='data/dataset/flowers/rose/10090824183_d02c613f10_m.jpg'
    img = Image.open(infer_path)
    plt.imshow(img)          #根据数组绘制图像
    plt.show()               #显示图像

    #对预测图片进行预处理
    infer_imgs = []
    infer_imgs.append(load_image(infer_path))
    infer_imgs = np.array(infer_imgs)

    for i in range(len(infer_imgs)):
        data = infer_imgs[i]
        dy_x_data = np.array(data).astype('float32')
        dy_x_data=dy_x_data[np.newaxis,:, : ,:]
        img = fluid.dygraph.to_variable(dy_x_data)
        out = model(img)
        lab = np.argmax(out.numpy())  #argmax():返回最大数的索引

        print("第{}个样本,被预测为：{},真实标签为：{}".format(i+1,label_dic[str(lab)],infer_path.split('/')[-2]))

print("结束")
```

结果：可以正常显示预测图片和真实标签。




```
第1个样本,被预测为：rose,真实标签为：rose
结束
```

四、实验结果与分析

经过以上的 debug 和调参过程，成功完成了鲜花识别的任务，且准确率高达 0.99937993，接近 100%。

在得到这个结果之前，进行了不少参数调整的过程。主要考虑批次大小和训练轮数这两个参数的调整，批次大小 32 的效果没有 64 好，在批次大小为 64 的基础上，训练轮数基本上是越高越好，但没有尝试超过 100 的轮次，因为训练时间可能会很长。在训练轮次为 65 的时候就得到了这个结果，所以认为没有再增加轮次的必要了。

经过本次实验，发现实验结果与批次大小和训练轮次的关系较大，通过调节这两个参数，可以得到较好的实验结果。

五、实验总结

通过本次实验，学会了用飞桨动态图搭建一个经典的卷积神经网络，也学会了一些基本的数据增广和调参技巧。在不断 debug 的过程中，对深度学习代码框架有了更深的认识，对一些代码更加熟悉，也学会了如何去改进它或者根据实际需要去调整它，为以后更多的神经网络应用打下了坚实的基础。