实验 2 图片数据整理

实验难度：一般

实验摘要：在项目 1 中，爬取到了许多的流浪狗图片，但是这些图片辨识程度不高，在本案例中进一步的将这些数据进行处理，把图片按图片大小分类，找出一些高清的图片来提供给意向人群领养，另外一些模糊、不清晰、信息缺失的图片可以回复发帖人索要清晰的信息。

为了划分出不同大小图片，将分别创建两个文件目录用于存放不同大小的图片。

通过调用 os 库操作文件， shutil 库移动文件，将杂乱的照片整理好统一的格式，并按大小放入不同的文件夹。

根据图片大小进行筛选并归类，对于图片大小大于 100KB 存放于清晰文件夹内，小于则放于不清晰文件夹里。

实验建议：了解 os 库操作文件， shutil 库移动文件的相关知识

实验目标：能够通过 Python 语句完成图片数据的整理任务

1、图片数据整理

-

1.1、生成文件列表

创建.py 文件，如文件名为 image_patch.py（文件名可自行设置），并导入 os，shutil 库，代码如下：

```
import os
```

```
import shutil
```

查看当前工作目录，使用 os 库的 getcwd()方法，代码如下:

```
# 显示当前工作目录
```

```
print(os.getcwd())
```

假定上述案例中所下载的图片是存放在"/home/data/downloads "目录下，为了便于文件操作把当前工作目录变更为"/home/data/downloads "，这里使用 os 库的 chdir()方法可以实现，如下。

```
ori_path = '/home/data/downloads'
```

```
os.chdir(ori_path)
```

上述案例中获取文件大多是 jpg 图像格式，也有非图像格式，为了正确读入图片，将所有文件名以 jpg 结尾的文件存入列表如 current_dir_list，便于后续文件读写操作。

```
#找到当前路径下的所有.jpg 文件，查到.jpg 文件打印出来
```

```
current_dir_list = [x for x in os.listdir(os.getcwd()) if x.endswith(".jpg")]
```

```
print(type(current_dir_list),len(current_dir_list))
```

执行命令后，将生成一个列表，长度为 33。.

这里使用列表推导式，如：

```
[x for x in os.listdir(ori_path) if x.endswith(".jpg")]
```

可以理解为

```
x = []
```

```
for i in os.listdir():
```

```
  if i.endswith('.jpg'):
```

```
    x.append(i)
```

也就是遍历 ori_path 路径下的文件，如果文件名是以".jpg"结尾，就追加到名为 x 列表里。

下一步

1、图片数据整理

- 1.1、生成文件列表
- 1.2、创建目录
- 取两个目录名，分别为 good 和 bad，并依此新建一个列表，如下：
- ```
  # 通过创建列表，为后续创建不同的文件夹使用
  ```
- ```
  files_list = ["good", "bad"]
  ```
-
- 为便于区分，在当前工作目录中创建分类的文件夹，并将分类后的图片按条件依次放入，目标路径名如下：
-

```
destination_path = './'
```

-
-

通过上面两步的操作，确定了目录名和目标路径，然后使用 os.mkdir()创建目录，其中 destination_path 是目标路径，分别创建"./good"，"./bad"文件夹，代码如下：

- `for files_list_str in files_list:`
- `    os.mkdir(destination_path + files_list_str)`
-
-
-

1.3、文件分类

-

首先遍历文件列表，这里使用 enumerate 函数遍历文件列表，该函数返回文件列表内元素之外，还返回元素对应的索引。然后在通过 os.path.getsize 方法获取文件大小，作为筛选图片是否清晰的一个条件，文件名作为函数参数，例如

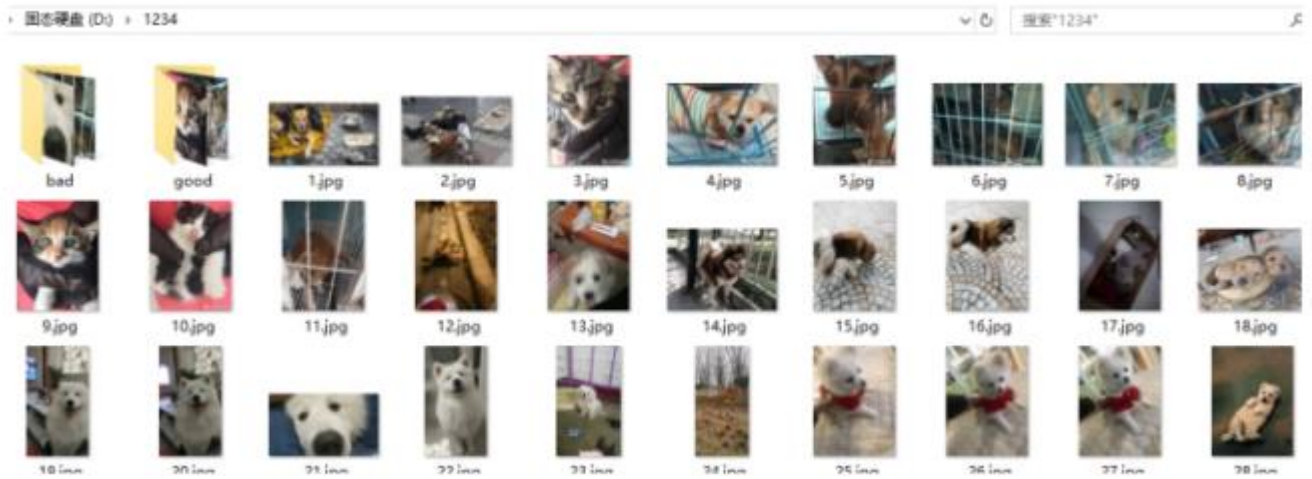
- `for index,file_name in enumerate(current_dir_list):`
- `    size = os.path.getsize(file_name)`
- `    print(size)`

-
- 将图片的大小与 102400 字节进行比较，若是大于 102400 则认为是清晰图片，放入"good"文件夹里，若是小于则认为是不清晰，放入"bad"文件夹里。
- 使用 shutil.move 函数方法完成文件移动，传入函数第一个参数 file_name 为文件名，第二个参数是 destination_path + files_list[0]为文件移动的目标路径，最后将图片移动进度记录在 log.txt 文件内，每移动一张图像的记录占文本一行，待所有图像完成移动后在文本末尾追加“处理完成！”内容，参考代码如下所示：
- `# 图像按大小分类进入文件夹`
- `if size > 102400:`
- `    shutil.move(file_name, destination_path + files_list[0])`
- `else:`
- `    shutil.move(file_name, destination_path + files_list[1])`
- `with open('log.txt','a') as log:`
- `    log.writelines(['总共:', str(len(current_dir_list)), '张，剩余:', str(len(current_dir_list) - index - 1), '张\n'])`

-
```
    if index == len(current_dir_list) - 1:
```
-
```
        log.write('done!')
```
-
```
    else:
```
-
```
        pass
```
-
- 通过查看日志文件来确认图像文件移动是否完成，以二进制方式打开 log.txt，将文件光标定位在文件末尾，向左偏移 6 个字节，并打印其内容，注意要打印输出内容的编码方式，可以'utf-8'或‘gbk’编码，参考代码如下所示。
-
```
f = open('log.txt','rb')
```
-
```
f.seek(-6,2)
```
-
```
content = f.read().decode('utf-8')
```
-
```
print(content)
```
-
```
f.close()
```
-

- 上述代码执行过后会打印输出：done!

- 查看文件夹，可见图片已经成功自动分类。

- 

-

- 下一步

-