实验 2 完成增广程序

实验难度：一般

实验摘要：根据上一个任务完成的增广处理的函数，可以对当前目录下的图像完成增广处理，并用不同的命名来标识处理过程。把增广后的数据保存在新的文件夹下。

实验建议：熟悉相关增广处理函数。

实验目标：能够使用增广处理函数完成对应的增广程序。

1、完成增广程序

- 1.1、准备文件夹
- 需要处理的图像放在了/data/hotdog 和/data/not-hotdog 目录下。
- 首先生成目标文件夹 aug_data,以及对应的文件目录。因为需要创建子目录，这里使用 os.makedirs 完成。
- `#整理所有数据`
- `ROOT_PATH = './data/'`
- `DST_PATH = './aug_data/'`
- `if not os.path.exists(DST_PATH):`
- `  hotdog_path = os.path.join(DST_PATH, 'hotdog')`
- `  not_path = os.path.join(DST_PATH, 'not-hotdog')`
- `  os.makedirs(hotdog_path)`

- ```
  os.makedirs(not_path)
  ```
-
-
- 1.2、遍历每个图像文件

  获得子文件夹名称，判断如果是目录，则进一步获得其中的文件名列表。通过 enumerate 函数，可以获得文件名列表的索引和名称。

  ```
  subfolders = os.listdir(ROOT_PATH)
  for subfolder in subfolders:
    #判断是否是子文件夹，如果不是，则进入下一个循环
    dir_path = os.path.join(ROOT_PATH, subfolder)
    if os.path.isfile(dir_path):
      continue
    #搜集子文件夹下图像文件名称(含后缀名)
    file_names = os.listdir(dir_path)
    #获得文件名和序号
    for cnt, ff in enumerate(file_names):
  ```

- 1.3、完成图像剪裁和尺寸变换

- 首先使用 opencv 读取的图像，如果读取失败，说明图像文件不符合要求跳过。然后调用函数完成随机裁剪图像，统一尺寸。重新命名文件名，使用类似 hotdog_000.jpg 的方式重新命名。注意这里要把序号填充到 3 位，需要用到字符串 format 格式。

-

- ```
  print("随机剪裁第{}幅图像".format(cnt))
  ```

```
    # 获取图像文件的全路径
    path_read_filename =  os.path.join(ROOT_PATH, subfolder, ff)
    print(path_read_filename)

    #读取图像文件，如果失败，则进入下一个循环
    img = cv2.imread(path_read_filename)
    if img is None:
        print("Faild to read image:",path_read_filename)
        continue

    # 调用函数，随机裁剪图像，并统一尺寸
    width = 224
    height = 224
```

```
    resized_file = rdnsize(img,width,height)
```

```
    # 命名为类似 hotdog_000.jpg
```

```
    filename = "{}_{:0>3d}.jpg".format(subfolder, cnt)
```

```
    resized_filename = os.path.join(DST_PATH, subfolder, filename)
```

```
    cv2.imwrite(resized_filename,resized_file)
```

下一步

1.4、完成其他增广操作

针对统一尺寸后的图像，进一步进行增广操作。可以先完成随机翻转，并通过类似 flipped_hotdog_000.jpg 等重新命名。

```
    #随机翻转文件
    flipped_file = rdnflip(resized_file)
```

```
    #命名为类似 flipped_hotdog_000.jpg
    filename = "flipped_{}_{:0>3d}.jpg".format(subfolder, cnt)
    flipped_filename = os.path.join(DST_PATH, subfolder, filename)
    cv2.imwrite(flipped_filename,flipped_file)
```

完成直方图均衡化，用类似 equalized_hotdog_000.jpg 进行重命名。

```
#进行直方图均衡化
    equalized_file = equalize(resized_file)
    #命名为类似 equalized_hotdog_000.jpg
    filename = "equalized_{}_{:0>3d}.jpg".format(subfolder, cnt)
    equalized_filename = os.path.join(DST_PATH, subfolder, filename)
    cv2.imwrite(equalized_filename,equalized_file)
```

增加高斯噪声，用类似 noisy_hotdog_000.jpg 进行重命名。

```
    #增加高斯噪声
    noisy_file = addnoise(resized_file)
    #命名为类似 noisy_hotdog_000.jpg
    filename = "noisy_{}_{:0>3d}.jpg".format(subfolder, cnt)
    noisy_filename = os.path.join(DST_PATH, subfolder, filename)
    cv2.imwrite(noisy_filename,noisy_file)
```

最后随机增加亮度，用类似 noisy_hotdog_000.jpg 重命名。

```
    #增加随机亮度
    bright_file = rdnbright(resized_file)
    #命名为类似 bright_hotdog_000.jpg
    filename = "bright_{}_{:0>3d}.jpg".format(subfolder, cnt)
    bright_filename = os.path.join(DST_PATH, subfolder, filename)
    cv2.imwrite(bright_filename,bright_file)
```

可以看到在 data_aug 目录下，同样有 hotdog 和 not-hotdog 两个子目录，每个目录下都有经过增广后的图像文件。

- 2、完整代码

-
-

2.1、完整代码

-

可以整理完整代码，参考如下

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

```
# -*- coding: utf-8 -*-
import cv2
import os
import random
import numpy as np
```

```
def rdnsize(img,width,height):
    #获取图像尺寸
    h,w,d = img.shape
    #对比裁剪目标图像的尺寸大小
    if h < height or w < width:
        #小图像直接变换尺寸
        result = cv2.resize(img, (width,height))
    else:
        #大图像在 x，y 轴上随机获得裁剪的坐标
        y = random.randint(0, h - height)
        x = random.randint(0, w - width)
        #裁剪到符合大小的图片
        result = img[y:y+height, x:x+width ,:]
    return result
```

```
def rdnflip(img):
    #随机获取翻转类型 1:水平翻转  0：垂直翻转  -1:双向翻转
    flipcode = random.randint(-1,1)
    #对图像进行翻转
    result = cv2.flip(img,flipcode)
    return result
```

```
def equalize(img):
    #颜色通道分离
    (b,g,r) = cv2.split(img)
    #分别进行直方图均衡化
    b_equalize =cv2.equalizeHist(b)
```

```
    g_equalize =cv2.equalizeHist(g)
    r_equalize =cv2.equalizeHist(r)
    #颜色通道合并
    result = cv2.merge((b_equalize,g_equalize,r_equalize))
    return result


def rdnbright(img,max):
    #获取亮度的随机数
    rdn = random.randint(0,max)
    #生成灰度图
    (h,w) = img.shape[:2]
    bright = np.ones((h,w),dtype=np.uint8)* rdn
    #将灰度图像转成彩色图
    bright_bgr = cv2.cvtColor(bright, cv2.COLOR_GRAY2BGR)
```

```
    #和原始图像混合，完成图像加亮度
    result = cv2.add(img, bright_bgr)
    return result


def addnoise(img,mean, sigma):
    #获取图像的行和高
    (h,w) = img.shape[:2]
    
    #生成一个同样大小的噪声图像
    noise = np.zeros((h,w),dtype=np.uint8)
    #用均值为 0， 标准差为 15
    cv2.randn(noise, mean, sigma)

```

```
    #将噪声图像转成彩色图
    noise_bgr = cv2.cvtColor(noise, cv2.COLOR_GRAY2BGR)
    #和原始图像混合，完成图像加噪声
    result = cv2.add(img, noise_bgr)


    return result


#整理所有数据
ROOT_PATH = './data/'
DST_PATH = './aug_data/'
if not os.path.exists(DST_PATH):
    hotdog_path = os.path.join(DST_PATH, 'hotdog')
    not_path = os.path.join(DST_PATH, 'not-hotdog')
```

```
    os.makedirs(hotdog_path)
    os.makedirs(not_path)


subfolders = os.listdir(ROOT_PATH)
for subfolder in subfolders:
    #判断是否是子文件夹，如果不是，则进入下一个循环
    dir_path = os.path.join(ROOT_PATH, subfolder)
    if os.path.isfile(dir_path):
        continue
    #搜集子文件夹下图像文件名称(含后缀名)
    file_names = os.listdir(dir_path)


    #获得文件名和序号

```

```
    for cnt, ff in enumerate(file_names):
```

```

```

```
        print("随机剪裁第{}幅图像".format(cnt))
```

```
        # 获取图像文件的全路径
```

```
        path_read_filename =  os.path.join(ROOT_PATH, subfolder, ff) #ff 是包含图像后缀名的
```

```
        print(path_read_filename)
```

```
        #读取图像文件，如果失败，则进入下一个循环
```

```
        img = cv2.imread(path_read_filename)
```

```
        if img is None:
```

```
            print("Faild to read image:",path_read_filename)
```

```
            continue
```

```

```

```
        # 调用函数，随机裁剪图像，并统一尺寸
```

```
        width = 224
```

```
        height = 224
        resized_file = rdnsize(img,width,height)



        # 命名为类似 hotdog_000.jpg
        filename = "{}_{:0>3d}.jpg".format(subfolder, cnt)
        resized_filename = os.path.join(DST_PATH, subfolder, filename)
        cv2.imwrite(resized_filename,resized_file)


        #随机翻转文件
        flipped_file = rdnflip(resized_file)
        #命名为类似 flipped_hotdog_000.jpg
        filename = "flipped_{}_{:0>3d}.jpg".format(subfolder, cnt)
```

```
        flipped_filename = os.path.join(DST_PATH, subfolder, filename)
        cv2.imwrite(flipped_filename,flipped_file)
        #进行直方图均衡化
        equalized_file = equalize(resized_file)
        #命名为类似 equalized_hotdog_000.jpg
        filename = "equalized_{}_{:0>3d}.jpg".format(subfolder, cnt)
        equalized_filename = os.path.join(DST_PATH, subfolder, filenam
e)
        cv2.imwrite(equalized_filename,equalized_file)
        #增加高斯噪声
        noisy_file = addnoise(resized_file, 0, 15)
        #命名为类似 noisy_hotdog_000.jpg
        filename = "noisy_{}_{:0>3d}.jpg".format(subfolder, cnt)
```

```
        noisy_filename = os.path.join(DST_PATH, subfolder, filename)
```

```
        cv2.imwrite(noisy_filename,noisy_file)
```

```

```

```
        #增加随机亮度
```

```
        bright_file = rdnbright(resized_file,30)
```

```
        #命名为类似 bright_hotdog_000.jpg
```

```
        filename = "bright_{}_{:0>3d}.jpg".format(subfolder, cnt)
```

```
        bright_filename = os.path.join(DST_PATH, subfolder, filename)
```

```
        cv2.imwrite(bright_filename,bright_file)
```

```

```

```

```

```

```

```

下一步
```