实验 1 编写各种图像增广函数

实验时长：4 小时

实验难度：一般

实验摘要：首先，统一图像尺寸到 224*224，对于比这个尺寸更大的图像，采取随机剪裁的方式，截取其中一块作为图像。对于比这个尺寸小的图像，就直接尺寸变换成该大小。这样，对大图进行剪裁时，每次可能图像的不同区域，可以根据实际情况增广更多。

然后对统一尺寸的图像，通过随机翻转图像，增加一倍。再随机增加一定的亮度，增加一倍的图像数量。最后对图像进行直方图均衡，改变颜色的方式完成一次增广。最后，再增加高斯噪声的方式，完成一次增广。最终获取原来 5 倍数量的图像。

实验建议：了解关于 os、opencv、random 库的相关函数。

实验目标：能够使用相关库函数完成各种图像增广函数的编写。

1、编写各种图像增广函数

- 1.1、读取图像
- 原始图像放到 data，以及子目录 hotdog、not-hotdog 下，可以先读取其中一张。首先，导入处理图像的 os 和 opencv 库，随机操作的 random 库
- `import cv2`
- `import os`
- `import random`
- `import matplotlib.pyplot as plt`
-
-

随机读入一张

- `img = cv2.imread("./data/hotdog/1000.png")`
- `if img is None:`
- `    print("Faild to read image!")`
-
-

显示一下图片

-
-
-
- `img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`
- `plt.imshow(img2)`
- `plt.show()`
-

1.2、完成图像剪裁

为了便于后续使用，把每个功能用一个函数实现。这里定义 rdnsize 函数，输入参数分别是图像、剪裁的宽和高。

rdnsize(img,width,height)

函数中，首先需要获取图像的宽高。如果比目标尺寸小的情况小，直接使用 resize 函数进行大小变换。如果比目标尺寸大，在 x 轴和 y 轴上随机取得一个起始坐标，保证裁剪的范围在图像之内。然后通过 numpy 切片的方式直接获取剪裁后的图像并返回。

```
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
```

可以尝试一下，这个函数的效果，读取一张图像，处理后用 matplotlib 显示。

```
img = cv2.imread("./data/hotdog/1051.png")
result = rdnsize(img, 224,224)
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
result2 = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
plt.subplot(121);plt.imshow(img2)
plt.subplot(122);plt.imshow(result2)
plt.show()
```

可以看到确实截取了原图的一部分。

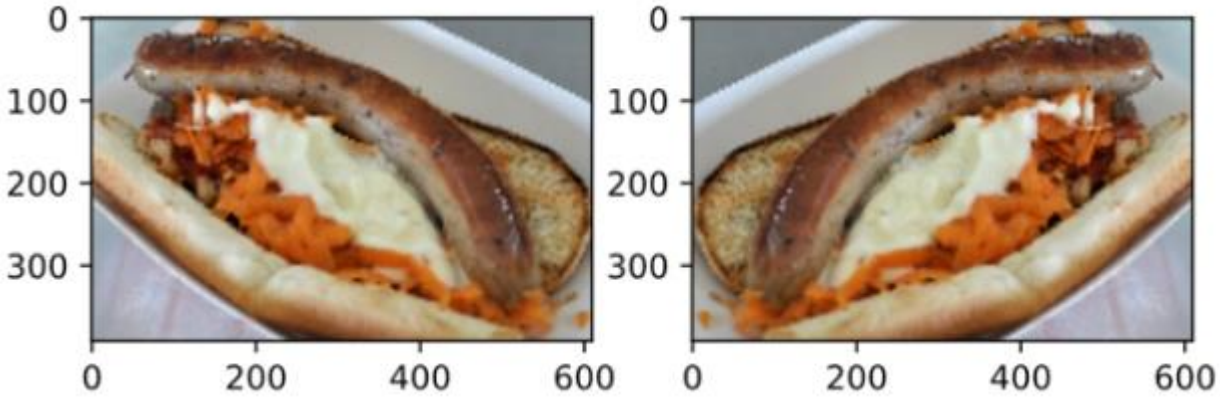
- 1.3、完成图像翻转
- 这里定义 rdnflip 函数，传入参数是图像。
- rdnflip(img)
-
- 函数中，随机获取-1，0，1 之中的一个翻转参数，调用 cv2.flip 完成随机翻转，并将翻转后的图像返回。
- `def rdnflip(img):`
- `  #随机获取翻转类型 1:水平翻转  0：垂直翻转  -1:双向翻转`
- `  flipcode = random.randint(-1,1)`
-

```
  #对图像进行翻转
```

- ```
  result = cv2.flip(img,flipcode)
  ```
- ```
  return result
  ```
-
- 同样，尝试一下函数效果

- ```
  img = cv2.imread("./data/hotdog/1051.png")
  ```
- ```
  result = rdnflip(img)
  ```
- ```
  img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  ```
- ```
  result2 = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
  ```
- ```
  plt.subplot(121);plt.imshow(img2)
  ```
- ```
  plt.subplot(122);plt.imshow(result2)
  ```
- ```
  plt.show()
  ```
-
- 可以查看效果，进行了翻转。

-
  ```



-

-

下一步

-
-
-

1.4、完成直方图均衡化

-

这里定义 equalize 函数，传入参数是图像。

-

equalize(img)

-
-

函数中，把图像的三个颜色通道进行分离，再分别进行直方图均衡化处理，最后将均衡化的颜色通道合并，将图像返回。

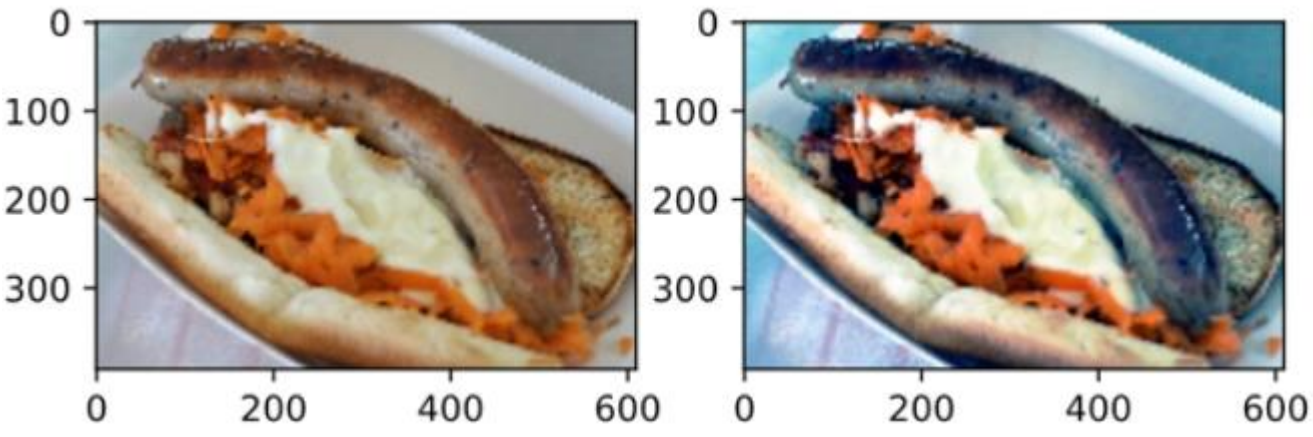
-

```
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
-
- 同样，尝试一下效果
-
-
-
-
-
-
-
-
-
-
-
-

- `img = cv2.imread("./data/hotdog/1051.png")`
- `result = equalize(img)`
- `img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`
- `result2 = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)`
- `plt.subplot(121);plt.imshow(img2)`
- `plt.subplot(122);plt.imshow(result2)`
- `plt.show()`
-
-
- 查看效果，颜色有较明显偏差
- 

-

下一步

1.5、增加随机亮度

这里定义 rdnbright 函数，输入图像，以及最大的亮度值。

rdnbright(img,max)

函数中，先获取亮度变换的随机数，然后生成和原图一样大小的灰度图，亮度设置为这个随机数。然后通过 cv2.add 和原图相加，获取增强了亮度的图像，并返回。

```
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，完成图像加噪声
  result = cv2.add(img, bright_bgr)
  return result
```

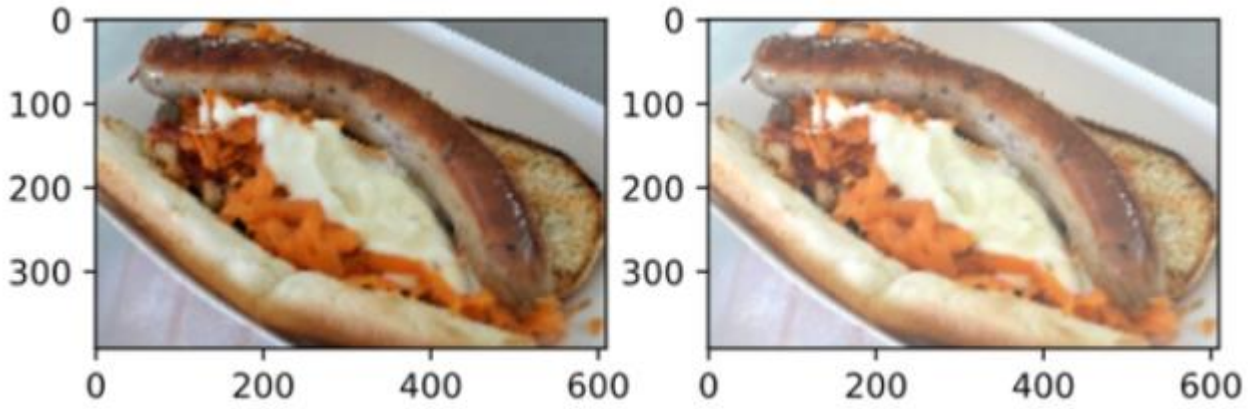
查看函数效果

```
img = cv2.imread("./data/hotdog/1051.png")
result =  rdnbright(img, 50)
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
result2 = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
plt.subplot(121);plt.imshow(img2)
```

`plt.subplot(122);plt.imshow(result2)`

`plt.show()`

得到效果对比图，确实提升了亮度



-

- 1.6、增加高斯噪声

- 这里定义 rdnbright 函数，输入图像，以及最大的亮度值。

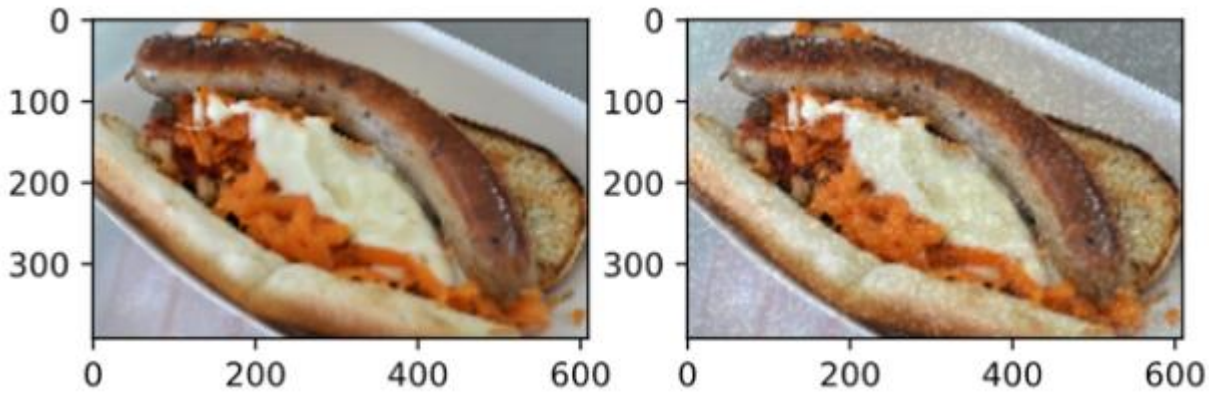
- addnoise(img,mean, sigma)

-
- 函数中，先获取亮度变换的随机数，然后生成和原图一样大小的灰度图，亮度设置为这个随机数。然后通过 cv2.add 和原图相加，获取增强了亮度的图像，并返回。

-
- `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 mean， 标准差为 sigma
- 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
- 尝试一下函数的效果
- img = cv2.imread("./data/hotdog/1051.png")

- `result =  rdnbright(img, 50)`
- `img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`
- `result2 = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)`
- `plt.subplot(121);plt.imshow(img2)`
- `plt.subplot(122);plt.imshow(result2)`
- `plt.show()`
-
- 可以看到图片增加了噪点
- 

-
-

下一步

-