图片数据增强之PIL格式

本项目针对PIL格式的图片提供了不同的数据增强方法，方便使用PIL读取图片后的数据增强操作。

概述

在计算机视觉任务中，图片的数据增强已经成为流程中一种必须的阶段。在其它条件相同的情况下，数据增强往往可以大幅提高最终的效果。 对图片的读取通常采用两种方式，PIL方式和OpenCV方式。由于两种方式读取后的数据增强方式有很多不同，故这里分两个项目来进行讲解。

本篇讲解PIL方式，对于OpenCV方式请参考另一个项目：

图片数据增强之Numpy格式

部分增强方法借鉴了一些官方项目中的方法：

基于YoloV3目标检测模型

基于PaddlePaddle的视频分类-TSN（动态图版）

用PaddlePaddle实现图像分类-ResNet（动态图版）

PIL 方式

PIL的全称是Python Image Library，是基于Python的图像处理工具。如果使用python 3 以上的版本，可以通过

```
pip install Pillow
```

来进行安装。

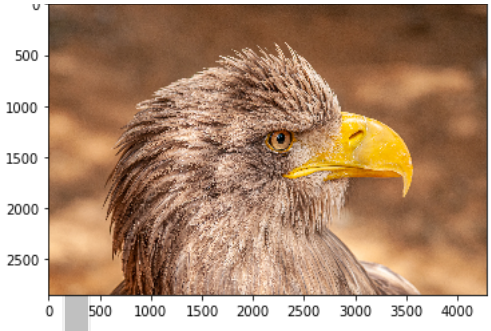
导入PIL及其他相关类

In [1]
```
from PIL import Image, ImageEnhance
import matplotlib.pyplot as plt
import numpy as np
import random
import math
```

读入图片

In [2]
```
img = Image.open('animal.jpeg')
```

显示图片

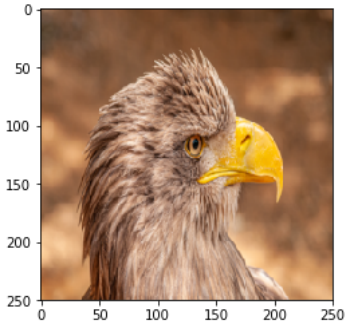


<Figure size 432x288 with 1 Axes>

1. 图片缩放

In [4]
```
def resize_img(img, target_size):
    img = img.resize((target_size, target_size), Image.BILINEAR)
    return img
```

In [5]
```
img_resize = resize_img(img, 250)
plt.imshow(img_resize)
plt.show(img_resize)
```

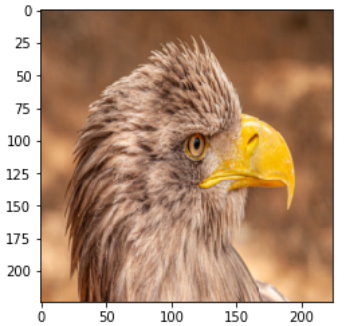


<Figure size 432x288 with 1 Axes>

2. 图片中央剪切

In [6]
```
def center_crop_img(img, target_size):
    w, h = img.size
    tw, th = target_size, target_size
    assert (w >= target_size) and (h >= target_size), \
            "image width({}) and height({}) should be larger than crop size".format(w, h,
    x1 = int(round((w - tw) / 2.))
    y1 = int(round((h - th) / 2.))
    # crop()四个参数分别是：(左上角点的x坐标，左上角点的y坐标，右下角点的x坐标，右下角点的
    img = img.crop((x1, y1, x1 + tw, y1 + th))
    return img
```

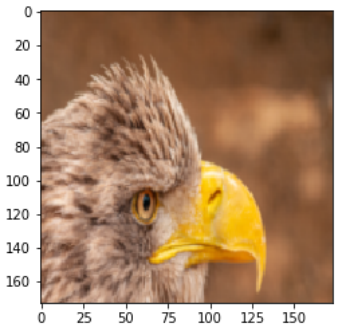
In [7]
```
img_center_crop = center_crop_img(img_resize, 224)
plt.imshow(img_center_crop)
plt.show(img_center_crop)
```



<Figure size 432x288 with 1 Axes>

In [12]
```
def random_crop_img(img, target_size, scale=[0.08, 1.0], ratio=[3. / 4., 4. / 3.]):
    aspect_ratio = math.sqrt(np.random.uniform(*ratio))
    w = 1. * aspect_ratio
    h = 1. / aspect_ratio

    bound = min((float(img.size[0]) / img.size[1]) / (w**2),
                (float(img.size[1]) / img.size[0]) / (h**2))
    scale_max = min(scale[1], bound)
    scale_min = min(scale[0], bound)

    target_area = img.size[0] * img.size[1] * np.random.uniform(scale_min,
                                                                scale_max)
    target_size = math.sqrt(target_area)
    w = int(target_size * w)
    h = int(target_size * h)

    i = np.random.randint(0, img.size[0] - w + 1)
    j = np.random.randint(0, img.size[1] - h + 1)

    img = img.crop((i, j, i + w, j + h))
    img = img.resize((int(target_size), int(target_size)), Image.BILINEAR)
    return img
```

In [13]
```
img_random_crop = random_crop_img(img_resize, 224)
plt.imshow(img_random_crop)
plt.show(img_random_crop)
```

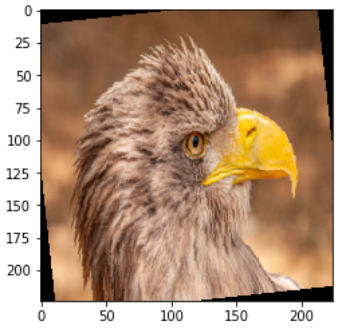


<Figure size 432x288 with 1 Axes>

4. 随机旋转

In [14]
```
def rotate_image(img):
    # 将图片随机旋转-14到15之间的某一个角度
    angle = np.random.randint(-14, 15)
    img = img.rotate(angle)
    return img
```

In [15]
```
img_rotate = rotate_image(img_center_crop)
plt.imshow(img_rotate)
plt.show(img_rotate)
```

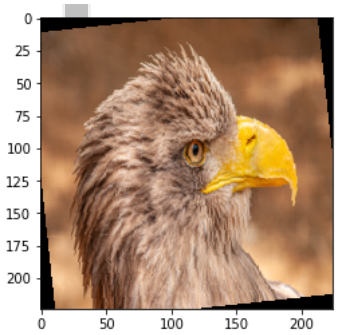


<Figure size 432x288 with 1 Axes>

5. 随机左右翻转

```
    v = random.random()
    if v < 0.5:
        img = img.transpose(Image.FLIP_LEFT_RIGHT)
    return img
```

In [17]
```
img_flip = flip_image(img_rotate)
plt.imshow(img_flip)
plt.show(img_flip)
```

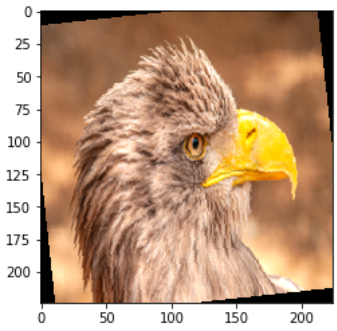


```
<Figure size 432x288 with 1 Axes>
```

6. 亮度调整

In [18]
```
def bright_image(img):
    # 随机调整亮度（调亮或暗）
    v = random.random()
    if v < 0.5:
        brightness_delta = 0.225
        delta = np.random.uniform(-brightness_delta, brightness_delta) + 1
        # delta值为0表示黑色图片，值为1表示原始图片
        img = ImageEnhance.Brightness(img).enhance(delta)
    return img
```

In [19]
```
img_bright = bright_image(img_flip)
plt.imshow(img_bright)
plt.show(img_bright)
```

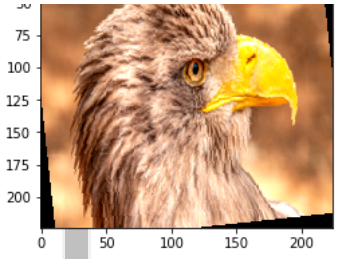


```
<Figure size 432x288 with 1 Axes>
```

7. 对比度调整

In [20]
```
def contrast_image(img):
    # 随机调整对比度
    v = random.random()
    if v < 0.5:
        contrast_delta = 0.5
        delta = np.random.uniform(-contrast_delta, contrast_delta) + 1
        # delta值为0表示灰度图片，值为1表示原始图片
        img = ImageEnhance.Contrast(img).enhance(delta)
    return img
```

In [21]
```
img_contrast = contrast_image(img_bright)
plt.imshow(img_contrast)
plt.show(img_contrast)
```

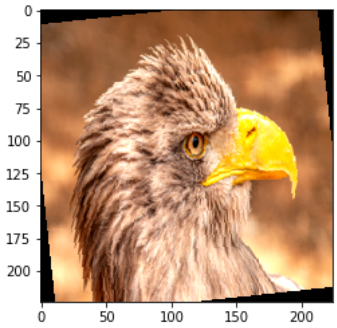


<Figure size 432x288 with 1 Axes>

8. 饱和度调整

In [22]
```
def saturation_image(img):
    # 随机调整颜色饱和度
    v = random.random()
    if v < 0.5:
        saturation_delta = 0.5
        delta = np.random.uniform(-saturation_delta, saturation_delta) + 1
        # delta值为0表示黑白图片，值为1表示原始图片
        img = ImageEnhance.Color(img).enhance(delta)
    return img
```

In [23]
```
img_saturation = saturation_image(img_contrast)
plt.imshow(img_saturation)
plt.show(img_saturation)
```

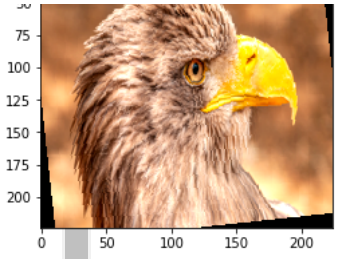


<Figure size 432x288 with 1 Axes>

9. 色度调整

In [24]
```
def hue_image(img):
    # 随机调整颜色色度
    v = random.random()
    if v < 0.5:
        hue_delta = 18
        delta = np.random.uniform(-hue_delta, hue_delta)
        img_hsv = np.array(img.convert('HSV'))
        img_hsv[:, :, 0] = img_hsv[:, :, 0] + delta
        img = Image.fromarray(img_hsv, mode='HSV').convert('RGB')
    return img
```

In [26]
```
img_hue = hue_image(img_saturation)
plt.imshow(img_hue)
plt.show(img_hue)
```



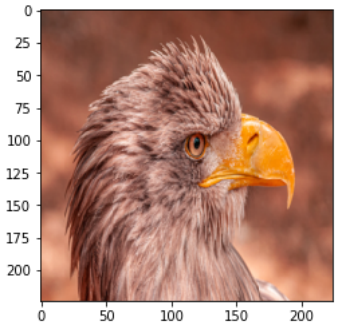
<Figure size 432x288 with 1 Axes>

10. 各种调整的组合

有时，可以将以上各种数据增强方法进行不同顺序的排列组合，这将产生新的数据增强方法。

In [27]
```
def distort_image(img):
    # 随机数据增强
    v = random.random()
    # 顺序可以自己随意调整
    if v < 0.35:
        img = bright_image(img)
        img = contrast_image(img)
        img = saturation_image(img)
        img = hue_image(img)
    elif v < 0.7:
        img = bright_image(img)
        img = saturation_image(img)
        img = hue_image(img)
        img = contrast_image(img)
    return img
```

In [28]
```
img_distort = distort_image(img_center_crop)
plt.imshow(img_distort)
plt.show(img_distort)
```



<Figure size 432x288 with 1 Axes>

结论

以上是基于PIL对图片做的各种数据增强操作。

需要注意的是以上各种操作后，图片的类型仍然是PIL.Image.Image，根据需要，可以用np.array方法将其转为numpy数组类型。