Python\_API\_Guides\_Image

目录

[编码和解码 1](#_Toc497217264)

[resize 2](#_Toc497217265)

[裁剪 3](#_Toc497217266)

[翻转，旋转和移位 4](#_Toc497217267)

[颜色空间转换 4](#_Toc497217268)

[亮度，对比度，色相和饱和度 5](#_Toc497217269)

[使用边框 6](#_Toc497217270)

[去噪 7](#_Toc497217271)

# 编码和解码

| **操作** | **描述** |
| --- | --- |
| tf.image. decode\_gif(contents,name=None) | 将GIF编码图像的第一帧解码为uint8张量。 |
| tf.image. decode\_jpeg(contents, channels=0, ratio=1, fancy\_upscaling=True, try\_recover\_truncated=False, acceptable\_fraction=1, dct\_method='', name=None) | 将JPEG编码的图像解码为uint8张量。 |
| tf.image. encode\_jpeg(image, format='', quality=95, progressive=False, optimize\_size=False, chroma\_downsampling=True, density\_unit='in', x\_density=300, y\_density=300, xmp\_metadata='', name=None) | JPEG图像编码。  image是一个3-D uint8形状的Tensor [height, width, channels]。  输出：JPEG编码图像。 |
| tf.image. decode\_png(contents, channels=0, dtype=tf.uint8, name=None) | 将PNG编码的图像解码为uint8或uint16张量。 |
| tf.image. encode\_png(image, compression=-1, name=None) | PNG编码图像。 |
| tf.image. decode\_image(contents, channels=None, name=None) | 检测图像是BMP，GIF，JPEG还是PNG，并执行适当的操作将输入字节string转换Tensor为类型uint8。 |

# resize

| **操作** | **描述** |
| --- | --- |
| tf. .image. resize\_images(images, size, method=ResizeMethod.BILINEAR, align\_corners=False) | 用method方法resize image到size |
| tf. .image. resize\_area(images, size, align\_corners=False, name=None) | resize images到size使用区域插值。 |
| tf. .image. resize\_bicubic(images, size, align\_corners=False,name=None) | resize images到size使用双三次插值。 |
| tf. .image. resize\_bilinear(images, size, align\_corners=False, name=None) | resize images到size使用双线性插值。 |
| tf. .image. resize\_nearest\_neighbor(images, size, align\_corners=False, name=None) | resize images到size使用最近邻插值。 |

# 裁剪

| **操作** | **描述** |
| --- | --- |
| tf.image. resize\_image\_with\_crop\_or\_pad(image, target\_height, target\_width) | 通过中央裁剪图像或使用零均匀填充图像，将图像调整为目标宽度和高度。 |
| tf.image. central\_crop(image,central\_fraction) | 裁剪图像的中央区域。fraction--float (0, 1] |
| tf.image. pad\_to\_bounding\_box(image, offset\_height,offset\_width, target\_height,target\_width) | 增加了offset\_height在顶部的零的行，offset\_width零列位于左侧，然后垫在底部和右侧用零，直到它具有的尺寸的图像target\_height，target\_width。 |
| tf.image. crop\_to\_bounding\_box(image, offset\_height,offset\_width, target\_height,target\_width) | 这个操作切割出一个矩形部分image。返回图像的左上角位于offset\_height, offset\_width，右下角位于 offset\_height + target\_height, offset\_width + target\_width。 |
| tf.image. extract\_glimpse(input, size, offsets, centered=True, normalized=True, uniform\_noise=True, name=None) | TODO |
| tf.image. crop\_and\_resize(image, boxes, box\_ind, crop\_size, method='bilinear', extrapolation\_value=0, name=None) | TODO |

# 翻转，旋转和移位

| **操作** | **描述** |
| --- | --- |
| tf.image. flip\_up\_down(image) | 垂直翻转图像（倒置）。 |
| tf.image. random\_flip\_up\_down(image, seed=None) | 随意(0.5)垂直翻转图像（倒置）。 |
| tf.image. flip\_left\_right(image) | 水平翻转图像（从左到右）。 |
| tf.image. random\_flip\_left\_right(image, seed=None) | 随机(0.5)地水平翻转图像（从左到右）。 |
| tf.image. transpose\_image(image) | 通过交换第一维和第二维来转置图像。 |
| tf.image. rot90(image, k=1, name=None) | 将图像逆时针旋转90度。 |

# 颜色空间转换

| **操作** | **描述** |
| --- | --- |
| tf.image. rgb\_to\_grayscale(images, name=None) | 将一个或多个图像从RGB转换为灰度。 |
| tf.image. grayscale\_to\_rgb(images, name=None) | 将一个或多个图像从灰度转换为RGB。 |
| tf.image. hsv\_to\_rgb(images, name=None) | 将一个或多个图像从HSV转换为RGB。 |
| tf.image.rgb\_to\_hsv (images, name=None) | 将一个或多个图像从RGB转换为HSV。 |
| tf.image.convert\_image\_dtype(image, dtype, saturate=False, name=None) | 转换image为dtype，如果需要，缩放其值。 |

# 亮度，对比度，色相和饱和度

| **操作** | **描述** |
| --- | --- |
| tf.image. adjust\_brightness(image, delta) | 调整RGB或灰度图像的亮度。delta--[0,1) |
| tf.image. random\_brightness(image, max\_delta, seed=None) | 以随机因素调整图像的亮度。[-max\_delta, max\_delta) |
| tf.image. adjust\_contrast(images, contrast\_factor) | 调整RGB或灰度图像的对比度。每个图像的每个通道独立调整对比度。(x - mean) \* contrast\_factor + mean |
| tf.image. random\_contrast(image, lower, upper, seed=None) | 通过随机因素调整图像的对比度。[lower, upper] |
| tf.image. adjust\_hue(image,delta, name=None) | 通过将RGB图像转换为HSV并旋转色调通道（H）来调整图像色调 delta--[-1, 1]。然后将图像转换回RGB。 |
| tf.image. random\_hue(image, max\_delta, seed=None) | 通过随机因素调整RGB图像的色调。[-max\_delta, max\_delta]。  max\_delta--[0, 0.5]。 |
| tf.image. adjust\_gamma(image, gamma=1, gain=1) | 函数在将每个像素缩放到0到1的范围后，按照等式Out = In \*\* gamma像素地变换输入图像。 |
| tf.image. adjust\_saturation(image, saturation\_factor,name=None) | 通过将RGB图像转换为HSV并将饱和度（S）通道乘以saturation\_factor和clipping来调整图像饱和度，然后转回RGB。 |
| tf.image. random\_saturation(image, lower, upper, seed=None) | 通过随机因素调整RGB图像的饱和度。[lower, upper] |
| tf.image.per\_image\_standardization  (image) | 线性变换image到零均值单位方差 |

# 使用边框

| **操作** | **描述** |
| --- | --- |
| tf.image.draw\_bounding\_boxes(images, boxes, name=None) | 在一批图像上绘制边框。 |
| tf.image. non\_max\_suppression(boxes, scores, max\_output\_size, iou\_threshold=0.5,name=None) | 返回input最大值的索引index |
| tf.image. sample\_distorted\_bounding\_box(  image\_size, bounding\_boxes,  seed=None, seed2=None,  min\_object\_covered=None,  aspect\_ratio\_range=None,  area\_range=None,  max\_attempts=None, use\_image\_if\_no\_bounding\_boxes=None,  name=None) | 随机截取图像信息 |

**去噪**

| **操作** | **描述** |
| --- | --- |
| tf.image.total\_variation(images, name=None) | 返回输入图像中相邻像素值的绝对差的总和 |