scipy.misc.imresize改变图像的大小

原创

一只tobey 于 2018-09-04 22:59:03 发布
分类专栏： python

scipy.misc.imresize(arr, size, interp='bilinear', mode=None)

resize an image.改变图像大小并且隐藏归一化到0-255区间的操作

参数：

arr: ndarray, the array of image to be resized

size: 只对原图的第0和第1维度的size做改变，其他维度的size保持不变。

1)int, percentage（百分比） of current size. 整数的时候是原图像的百分比,例如7,80,160分别是原图像的百分之七，百分之八十，百分之一百六十的大小

2)float, fraction（分数） of current size. 浮点数的时候是原图像的分数，例如0.7，1.5分别是原图像的0.7倍和1.5倍

3)tupe, size of the output image (height,width).元组的时候是直接定义输出图像的长和宽，与原图的大小无关

interp: str, optional. interpolation（插值） to use for re-sizing(**'nearest', 'lanczos', 'bilinear', 'bicubic', 'cubic'**)。是一个可选的参数，是调整图像大小所用的插值的方法，分别有最近邻差值，Lanczos采样放缩插值，双线性插值，双三次插值，三次插值

mode: str, optional.

以下讨论都是基于size=1.0或者size=100也就是说不改变原图像的大小的情况下。如果size不是和原图大小一样，那么先进行mode变化，再利用插值方法对size进行变化。

(1) 当arr是二维的时候arr=[h,w]，可以选择'P','L',None：mode='L'和mode=None的结果是一样的，都是直接将原图归一化到0-255范围内(归一化的方法在下面)，结果shape=[h,w]；mode='P'的时候是将mode='L'的结果是将只有一个图层的二维图像变为3维图像，具体做法是三个图层是一样的，结果shape=[h,w,3]。

(2)当arr是三维的时候arr.shape=[h,w,c],必须满足一个条件，就是至少一个维度是3！！！，mode可以选择'RGB', None实际上这两个选择结果是一样的；1)当c=3时，结果shape=[h,w,3], 2)当c不等于3同时只有一个3的时候，shape的变化如下[h,3,c]------[h,c,3]，[3,w,c]-----[w,c,3]，3)当c不等于3同时h=w=3的时候，shape变化如下[3,3,c]-------[3,c,3]. 注意这里面的2)和3)因为维度改变了，会在下面讨论

(3)当arr是四维的时候：mode='RGBA'或者None，略（日后补充）

返回值： imresize: ndarray, the resized array of image

归一化的方法：

imresize的进行大小改变之前要对原来的数据归一化成0-255区间的大小，对数据的shape不进行更改。在原来的数据中找到最大值max和最小值min，求得极值m，归一化操作y=（x-min）/m，这样就将x归一化到0-1区间，最后归一化0-255区间y*255.

举例：

```
import scipy.misc as smi
import numpy as np

a=np.arange(24).reshape(3,8) # 2-D ndarray,all values are oderly in 0-23
a1=smi.resize(a,size=1.0,mode=None) # all values are casted to 'uint32'in 0-255 range
a2=smi.resize(a,size=1.0,mode='L') # a1=a2,shape=a.shape
a3=smi.resize(a,size=1.0,mode='P') # a3.shape=(3,8,3)

b=np.arange(36).reshape(1,36) # 从0到35总共36个数
b0=smi.resize(b,size=1.0)
b1=b.reshape(2,3,6) # 3-D ndarray,shape=[h,3,c]
b2=smi.resize(b1,size=1.0,mode='RGB') # mode=None也是一样的
b3=b.reshape(3,2,6) # shape=[3,w,c]
b4=smi.resize(b3,size=1.0,mode='RGB')
b5=b.reshape(3,3,4) # shape=[3,3,c]
b6=smi.resize(b5,size=1.0) # mode=None默认
```

二维的结果如下

```
a=
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23]])
a1=                                                  # max=23,min=0,m=23
array([[  0,  11,  22,  33,  44,  55,  67,  78],     # (x-min)*255/m
       [ 89, 100, 111, 122, 133, 144, 155, 166],     # eg:78=(7-0)*255/23 四舍五入
       [177, 188, 200, 211, 222, 233, 244, 255]], dtype=uint8)
a2=
array([[  0,  11,  22,  33,  44,  55,  67,  78],      # 和a1相同
       [ 89, 100, 111, 122, 133, 144, 155, 166],
       [177, 188, 200, 211, 222, 233, 244, 255]], dtype=uint8)
a3=
array([[[  0,   0,   0],              # shape=[h,w,3],三层一模一样
        [ 11,  11,  11],
        [ 22,  22,  22],
        [ 33,  33,  33],
        [ 44,  44,  44],
        [ 55,  55,  55],
        [ 67,  67,  67],
        [ 78,  78,  78]],
       [[ 89,  89,  89],
        [100, 100, 100],
        [111, 111, 111],
        [122, 122, 122],
        [133, 133, 133],
        [144, 144, 144],
        [155, 155, 155],
        [166, 166, 166]],
       [[177, 177, 177],
        [188, 188, 188],
        [200, 200, 200],
        [211, 211, 211],
        [222, 222, 222],
        [233, 233, 233],
        [244, 244, 244],
        [255, 255, 255]]], dtype=uint8)
```

三维的结果如下

```
b=
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35]])
b0=      # 这样就可以看到归一化的值的一一对应了
array([[  0,   7,  15,  22,  29,  36,  44,  51,  58,  66,  73,  80,  87,
         95, 102, 109, 117, 124, 131, 138, 146, 153, 160, 168, 175, 182,
        189, 197, 204, 211, 219, 226, 233, 240, 248, 255]], dtype=uint8)
b1=
array([[[ 0,  1,  2,  3,  4,  5],       # b1.shape=[h,3,c]=[2,3,6]
        [ 6,  7,  8,  9, 10, 11],
        [12, 13, 14, 15, 16, 17]],
       [[18, 19, 20, 21, 22, 23],
        [24, 25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34, 35]]])
b2=                                     # b2.shape=[h,c,3]=[2,6,3]
array([[[  0,  44,  87],
```

```
        [  7,  51,  95],
        [ 15,  58, 102],
        [ 22,  66, 109],
        [ 29,  73, 117],
        [ 36,  80, 124]],
       [[131, 175, 219],
        [138, 182, 226],
        [146, 189, 233],
        [153, 197, 240],
        [160, 204, 248],
        [168, 211, 255]]], dtype=uint8)
b3=                                  # b3.shape=[3,w,c]=[3,2,6]
array([[[ 0,  1,  2,  3,  4,  5],
        [ 6,  7,  8,  9, 10, 11]],
       [[12, 13, 14, 15, 16, 17],
        [18, 19, 20, 21, 22, 23]],
       [[24, 25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34, 35]]])
b4=                                   # b4.shape=[w,c,3]=[2,6,3]
array([[[  0,  87, 175],
        [  7,  95, 182],
        [ 15, 102, 189],
        [ 22, 109, 197],
        [ 29, 117, 204],
        [ 36, 124, 211]],
       [[ 44, 131, 219],
        [ 51, 138, 226],
        [ 58, 146, 233],
        [ 66, 153, 240],
        [ 73, 160, 248],
        [ 80, 168, 255]]], dtype=uint8)
b5=                                   # b5.shape=[3,3,c]=[3,3,4]
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],
       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]],
       [[24, 25, 26, 27],
        [28, 29, 30, 31],
        [32, 33, 34, 35]]])
b6=                                    # b6.shape=[3,c,3]=[3,4,3]
array([[[  0,  87, 175],
        [  7,  95, 182],
        [ 15, 102, 189],
        [ 22, 109, 197]],
       [[ 29, 117, 204],
        [ 36, 124, 211],
        [ 44, 131, 219],
        [ 51, 138, 226]],
       [[ 58, 146, 233],
        [ 66, 153, 240],
        [ 73, 160, 248],
        [ 80, 168, 255]]], dtype=uint8)
```

那么问题来了，如果要对特征图而不是图像操作，特征图的[h,w,c]里面的维度没有任何的要求，可能一个3也没有。现在要求resize之后为[h',w',c]也就是说c不变，只要求原图的h和w进行变化，那该怎么办呢？该用哪个函数呢?如果有谁知道的话，请留言相告，谢谢！