In [0]:

```
import numpy as np
def rle2mask(rle):
  # If rle is empty or null
  if(len(rle)<1):
    return np.zeros((128,800) ,dtype=np.uint8)

  height = 256
  width = 1600

  # Defining the length of mask. This will be 1d array and later will be
reshaped to 2d.
  mask = np.zeros(height*width ).astype(np.uint8)
  # We will have an array that wil contain rle
  array = np.asarray([int(x) for x in rle.split()])
  start = array[0::2]-1 # this willl contain the start of run length
  length = array[1::2] # this will contain the length of each rle.
  '''
    pixels = np.array((0, 1, 1, 1, 1, 0, 0, 0, 1))

    # Concatenating a zero at the start and end of the array is to
    # make sure that the first changing is always from 0 to 1
    pixels = np.concatenate([[0], pixels, [0]])
    print('pixels:', pixels)

    # the array except the first element
    print('pixels[1:]:', pixels[1:])
    # the array except the last element
    print('pixels[:-1]:', pixels[:-1])

    # runs include indices to wherever 0s change to 1s or 1s change to 0
s
    print('where condition:', pixels[1:] != pixels[:-1])
    runs = np.where(pixels[1:] != pixels[:-1])
    print('runs:', runs)

    # the purpose of adding 1 here is to make sure that the indices poin
t to
    # the very first 1s or 0s of the 1s or 0s, this is needed because
    # np.where gets the indices of elements before changing
    runs = runs[0] + 1
    print('runs = runs[0] + 1:', runs)

    # runs[1::2] --> runs[start:stop:step], thus 2 here is the step
    # thus runs[1::2] includes the indices of the changing from 1 to 0
    print('runs[1::2]:', runs[1::2])

    # runs[::2] includes the indices for the changing from 0 to 1
    print('runs[::2]:', runs[::2])

    # the length of 1s
    print('runs[1::2]-runs[::2]:', runs[1::2] - runs[::2])

    # replace runs[1::2] with the lengths of consecutive 1s
    runs[1::2] -= runs[::2]

    print('return:', ' '.join(str(x) for x in runs))

    Output:
```

```
    pixels: [0 0 1 1 1 1 0 0 0 1 0]
    pixels[1:]: [0 1 1 1 1 0 0 0 1 0]
    pixels[:-1]: [0 0 1 1 1 1 0 0 0 1]
    where condition: [False  True False False False  True False False  T
rue  True]
    runs: (array([1, 5, 8, 9]),)
    runs = runs[0] + 1: [ 2  6  9 10]
    runs[1::2]: [ 6 10]
    runs[::2]: [2 9]
    runs[1::2]-runs[::2]: [4 1]
    return: 2 4 9 1

  '''
  #  now we will chane the value of each pixel in the rle to 1.
  for i,start in enumerate(start):
    mask[int(start):int(start+length[i])] = 1

    '''
    width=4, height=3

    s = [1,2,3,4,5,6,7,8,9,10,11,12]

    s.reshape(4,3) :
    [[ 1  2  3]
     [ 4  5  6]
     [ 7  8  9]
     [10 11 12]]

    s.reshape(4,3).T :
    [[ 1  4  7 10]
     [ 2  5  8 11]
     [ 3  6  9 12]]
    '''

  # now we will return the mask by first reshaping it and then rotating
 by 90 degrees and the vertically flipping it upside down.
  #return np.flipud(np.rot90(mask.reshape(width, height), k=1)) # Here k
=1 means we will rotate only once.
  return mask.reshape( (height,width), order='F' )[::2,::2]
```

In [0]:

```
    def mask2rle(img):
        '''
        img: numpy array, 1 - mask, 0 - background
        Returns run length as string formated
        '''
        #print(img.shape)
        pixels= img.T.flatten()
        pixels = np.concatenate([[0], pixels, [0]])
        runs = np.where(pixels[1:] != pixels[:-1])[0] + 1
        runs[1::2] -= runs[::2]
        return ' '.join(str(x) for x in runs)
```

In [ ]:

```
    '''
    mask_rle = ' '.join(str(x) for x in runs)
    s = mask_rle.split()
    print('s:', s)

    print('s[0:][::2]:', s[0:][::2])
    assert(s[0:][::2] == s[::2])

    print('s[1:][::2]:', s[1:][::2])
    assert(s[1:][::2] == s[1::2])

    starts = [np.asarray(x, dtype=int) for x in (s[0:][::2], s[1:][::
2])]
    print('starts:', starts)

    rle_decode(mask_rle, (1, 9))


    output:
    s: ['2', '4', '9', '1']
    s[0:][::2]: ['2', '9']
    s[1:][::2]: ['4', '1']
    starts: [array([2, 9]), array([4, 1])]
    array([[0, 1, 1, 1, 1, 0, 0, 0, 1]], dtype=uint8)
    '''
```

Data Augmentation