```
op1:
 tensor([[[[-1., -1., -1.], [-1., -2., -2.], [-1., -2., -2.]]]])
tensor([[[[-1.,  0.], [ 0., -1.]]]])
op6:
 tensor([[[[-2., -2.], [-2., -2.]]]])
 tensor([[[[-1.,  0.], [ 0., -1.]]]])
```

提示 本节特意花了很大篇幅来解释卷积的操作细节，表明这部分内容非常重要，是卷积神经网络的重点。只有将卷积操作的细节理解透彻，才能在实际编程过程中独立调试和解决程序错误。

另外需要注意的是，通过卷积函数可以实现窄卷积和同卷积，但不能实现全卷积。在 PyTorch 中有单独的反卷积函数可以实现全卷积，该函数在自然语言处理中很少使用，这里不再详细介绍。

5.11.5 了解池化接口

在 PyTorch 中，按照池化处理的维度，又将最大池化和平均池化各分成了 3 个处理函数。

1. 最大池化函数

- torch.nn.functional.max_pool1d()：实现按照 1 个维度进行的最大池化操作。常用于处理序列数据。
- torch.nn.functional.max_pool2d()：实现按照 2 个维度进行的最大池化操作。常用于处理二维图片。
- torch.nn.functional.max_pool3d()：实现按照 3 个维度进行的最大池化操作。常用于处理三维图片。

2. 平均池化函数

- torch.nn.functional.avg_pool1d()：实现按照 1 个维度进行的平均池化操作。常用于处理序列数据。
- torch.nn.functional.avg_pool2d()：实现按照 2 个维度进行的平均池化操作。常用于处理二维图片。
- torch.nn.functional.avg_pool3d()：实现按照 3 个维度进行的平均池化操作。常用于处理三维图片。

3. 池化函数的定义

从最大池化函数和平均池化函数的函数名可以看出，二者的形式几乎完全相同，只不过实现的具体细节不同。这里以基于 2 个维度的最大池化函数为例，详细介绍其具体的用法。