深度学习-语义分割篇

作者：神秘的wz

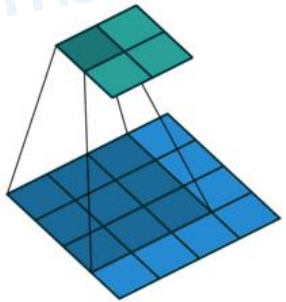
转置卷积

Transposed Convolution

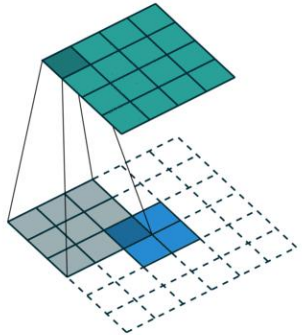
(fractionally-strided convolution、deconvolution)

作用：upsampling

- 转置卷积不是卷积的逆运算
- 转置卷积也是卷积



padding=0, strides=1 (conv)



padding=0, strides=1 (transposed conv)

A guide to convolution arithmetic for deep learning (https://arxiv.org/abs/1603.07285)

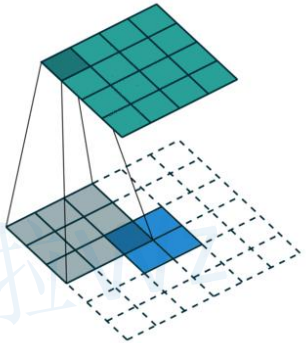
转置卷积

转置卷积运算步骤：

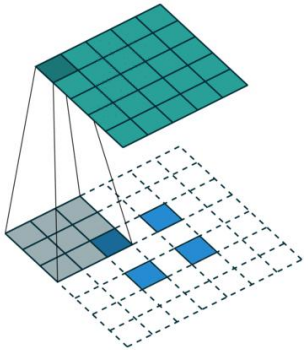
- 在输入特征图元素间填充s-1行、列0
- 在输入特征图四周填充k-p-1行、列0
- 将卷积核参数上下、左右翻转
- 做正常卷积运算（填充0，步距1）

$$H_{out} = (H_{in} - 1) \times stride[0] - 2 \times padding[0] + kernel\_size[0]$$

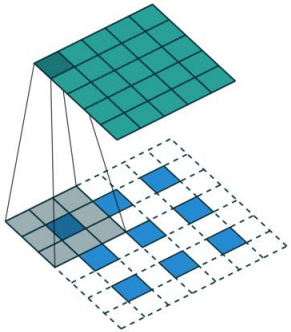
$$W_{out} = (W_{in} - 1) \times stride[1] - 2 \times padding[1] + kernel\_size[1]$$



s=1, p=0, k=3



s=2, p=0, k=3



s=2, p=1, k=3

https://github.com/vdumoulin/conv_arithmetic

转置卷积

忽略偏执bias

0	1	1
0	1	0
1	0	1

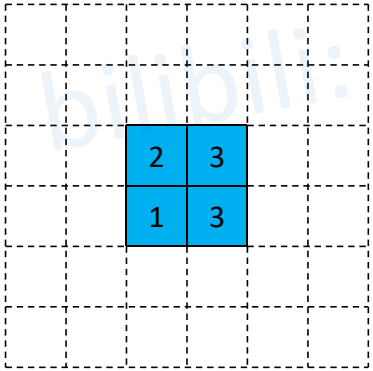
卷积kernel

上下、左右翻转

1	0	1
0	1	0
1	1	0

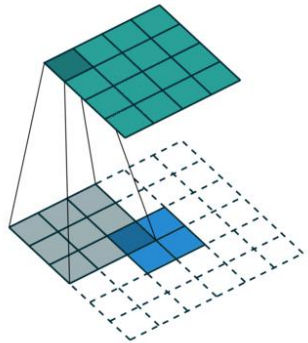
卷积操作

0	2	5	3
0	3	7	3
2	4	5	3
1	3	1	3



输入feature map

输出feature map



s=1, p=0, k=3

转置卷积

torch.nn.ConvTranspose2d参数：

Parameters

- **in_channels** (*int*) – Number of channels in the input image
- **out_channels** (*int*) – Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) – Size of the convolving kernel
- **stride** (*int or tuple, optional*) – Stride of the convolution. Default: 1
- **padding** (*int or tuple, optional*) – `dilation * (kernel_size - 1) - padding` zero-padding will be added to both sides of each dimension in the input. Default: 0
- **output_padding** (*int or tuple, optional*) – Additional size added to one side of each dimension in the output shape. Default: 0
- **groups** (*int, optional*) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) – If `True`, adds a learnable bias to the output. Default: `True`
- **dilation** (*int or tuple, optional*) – Spacing between kernel elements. Default: 1

$$H_{out} = (H_{in} - 1) \times stride[0] - 2 \times padding[0] + dilation[0] \times (kernel\_size[0] - 1) + output\_padding[0] + 1$$

$$W_{out} = (W_{in} - 1) \times stride[1] - 2 \times padding[1] + dilation[1] \times (kernel\_size[1] - 1) + output\_padding[1] + 1$$

转置卷积

普通卷积计算（忽略偏执bias）

1	0	1	0
0	0	1	1
1	0	0	1
0	1	0	0

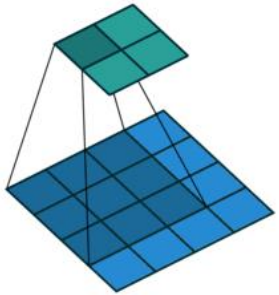
输入feature map

0	1	1
0	1	0
1	0	1

卷积kernel

2	3
1	3

输出feature map

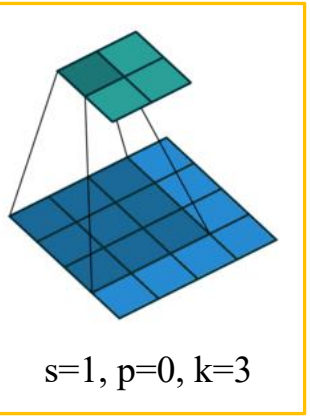


s=1, p=0, k=3

转置卷积

普通卷积计算（忽略偏执bias）

0	1	1
0	1	0
1	0	1



s=1, p=0, k=3

1	0	1	0
0	0	1	1
1	0	0	1
0	1	0	0

输入feature map

0	1	1	0
0	1	0	0
1	0	1	0
0	0	0	0

0	0	1	1
0	0	1	0
0	1	0	1
0	0	0	0

0	0	0	0
0	1	1	0
0	1	0	0
1	0	1	0

0	0	0	0
0	0	1	1
0	0	1	0
0	1	0	1

卷积kernel等效矩阵

2	3
1	3

输出feature map

参考 https://blog.csdn.net/tsyccnh/article/details/87357447

转置卷积

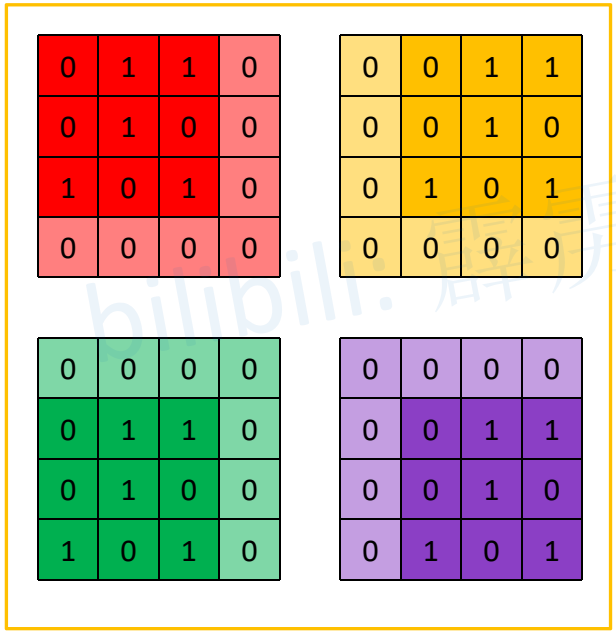
1	0	1	0
0	0	1	1
1	0	0	1
0	1	0	0

1	0	1	0	0	0	1	1	1	0	0	1	0	1	0	0

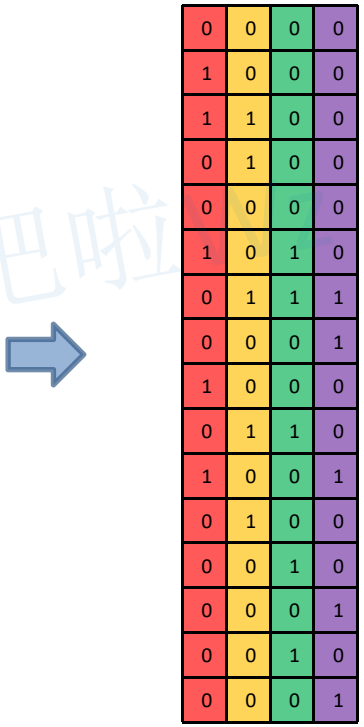
I: 1x16

输入feature map

转置卷积



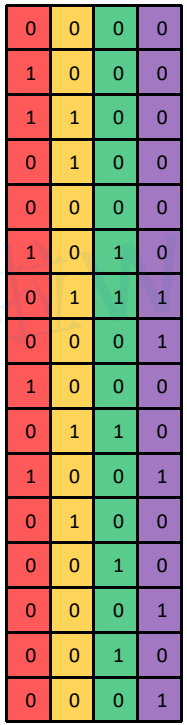
卷积kernel等效矩阵



C: 16x4

转置卷积

I: 1x16



C: 16x4

O: 1x4

$$I^{1\times16}\ C^{16\times4} = O^{1\times4}$$

C: 16x4

转置卷积

1	0	1	0	0	0	1	1	1	0	0	1	0	1	0	0

$\boldsymbol{I}$: 1x16

$\neq$ (不是逆运算)

2	3	1	3

$\boldsymbol{O}$: 1x4

0	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0
0	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0
0	0	0	0	0	1	1	0	0	1	0	0	1	0	1	0
0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	1

$\boldsymbol{C^T}$: 4x16

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

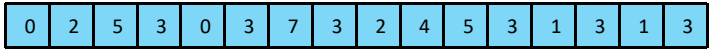
$\boldsymbol{P}$: 1x16

0	2	5	3
0	3	7	3
2	4	5	3
1	3	1	3

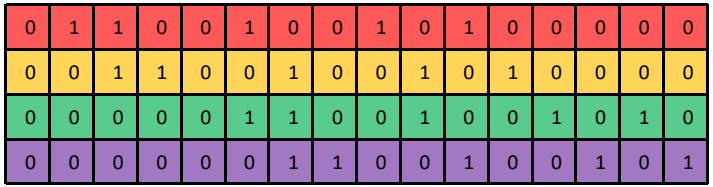
$$\boldsymbol{O}^{1\times 4}\,\boldsymbol{C}^{T} = \boldsymbol{P}^{1\times 16}$$

转置卷积

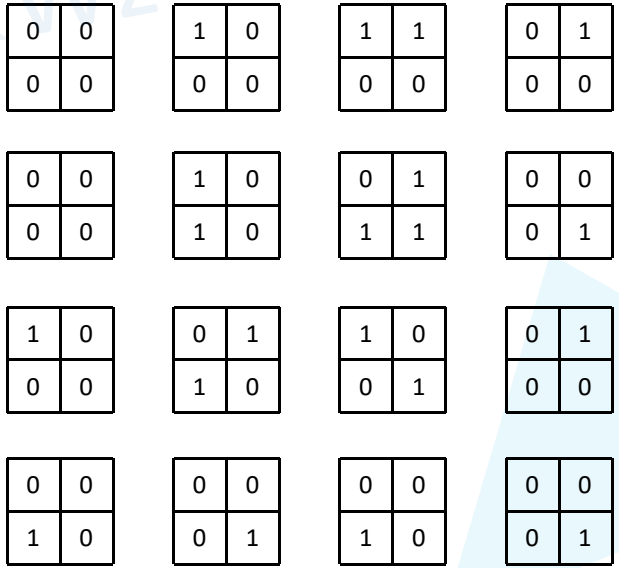
O: 1x4



P: 1x16



***C**^T*: 4x16



转置卷积

2	3
1	3

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

P: 1x16

0	0
0	0

1	0
0	0

1	1
0	0

0	1
0	0

0	0
0	0

1	0
1	0

0	1
1	1

0	0
0	1

1	0
0	0

0	1
1	0

1	0
0	1

0	1
0	0

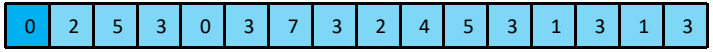
0	0
1	0

0	0
0	1

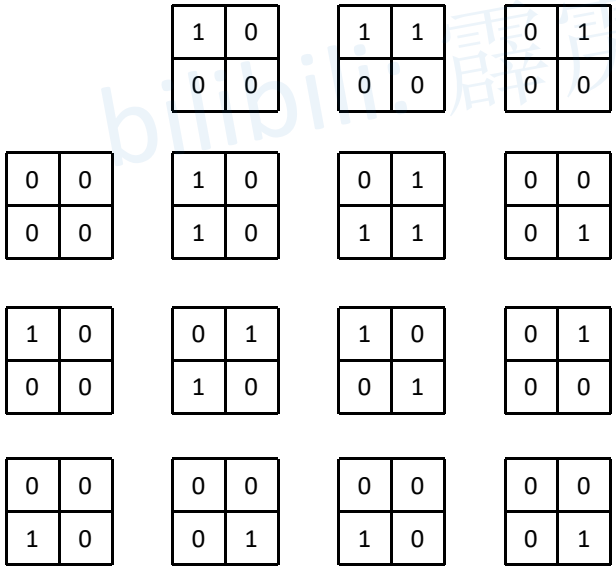
0	0
1	0

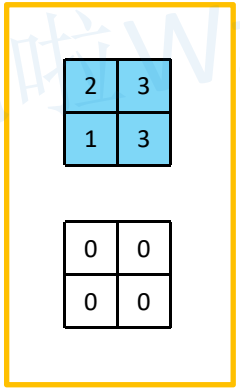
0	0
0	1

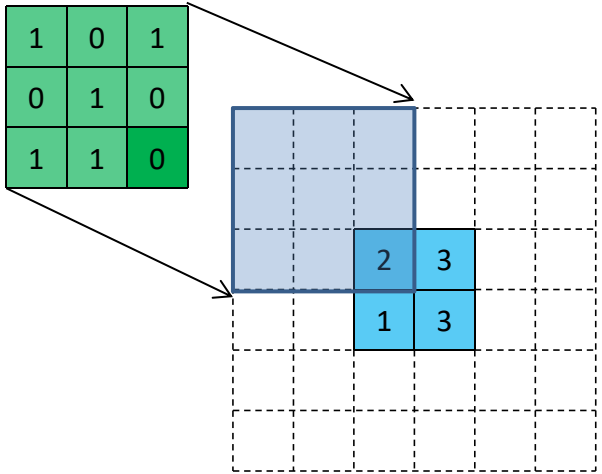
转置卷积



P: 1x16







转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

P: 1x16

0	0
0	0

1	1
0	0

0	1
0	0

0	0
0	0

1	0
1	0

0	1
1	1

0	0
0	1

1	0
0	0

0	1
1	0

1	0
0	1

0	1
0	0

0	0
1	0

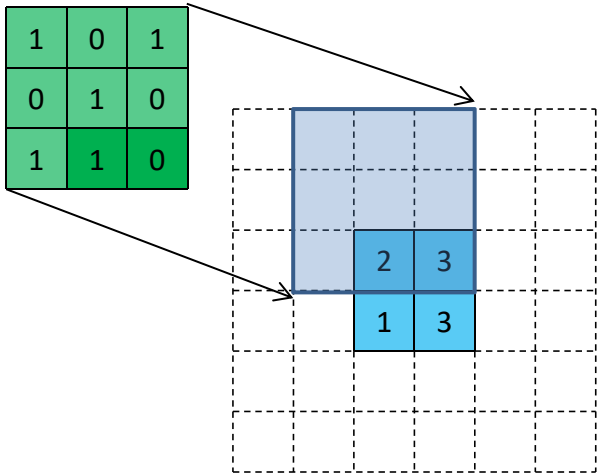
0	0
0	1

0	0
1	0

0	0
0	1

2	3
1	3

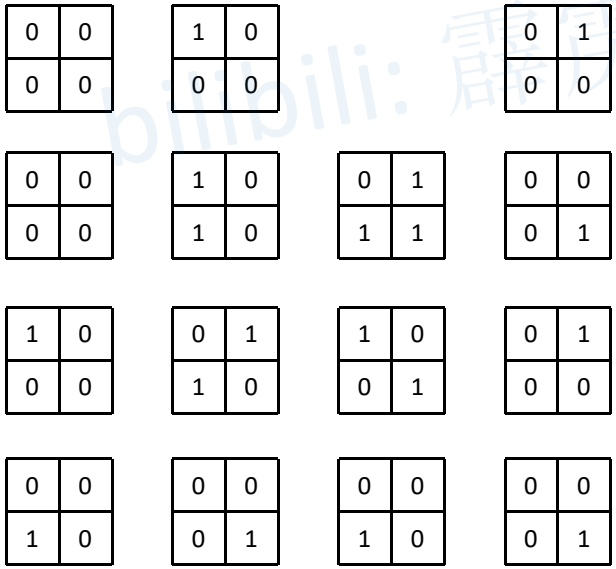
1	0
0	0

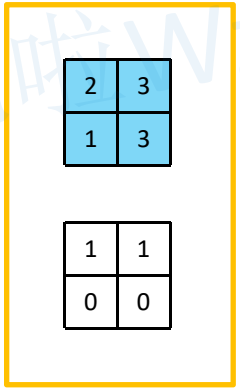


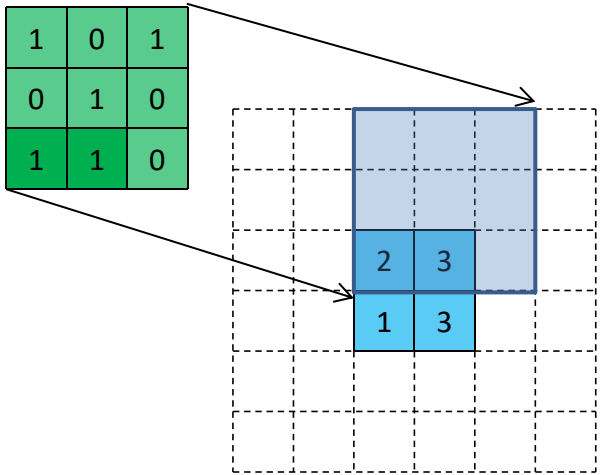
转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

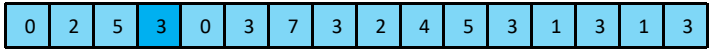
P: 1x16



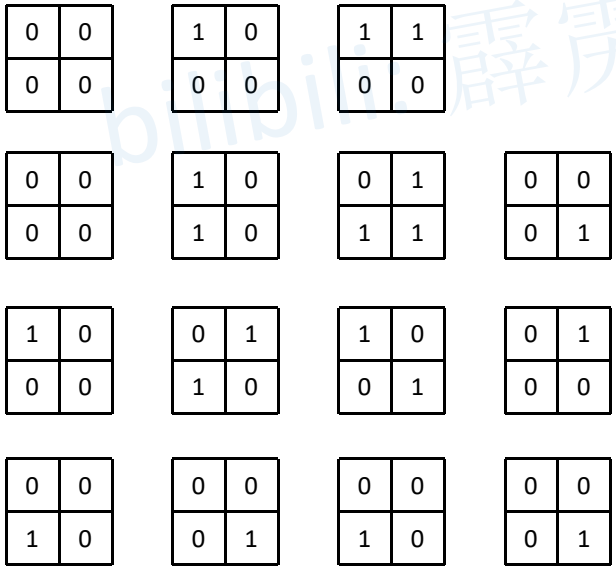


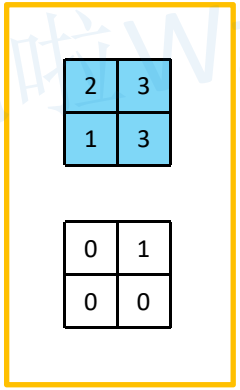


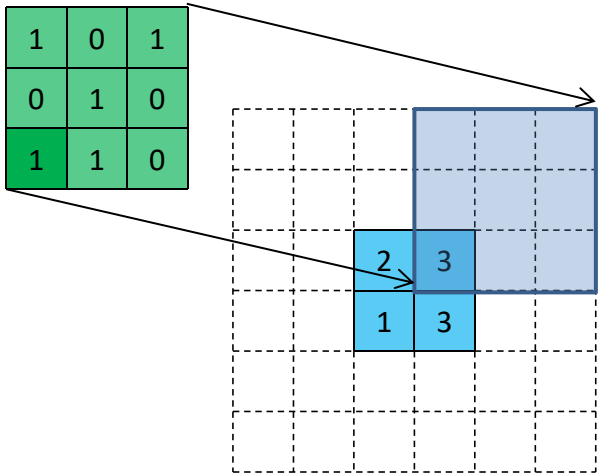
转置卷积



P: 1x16



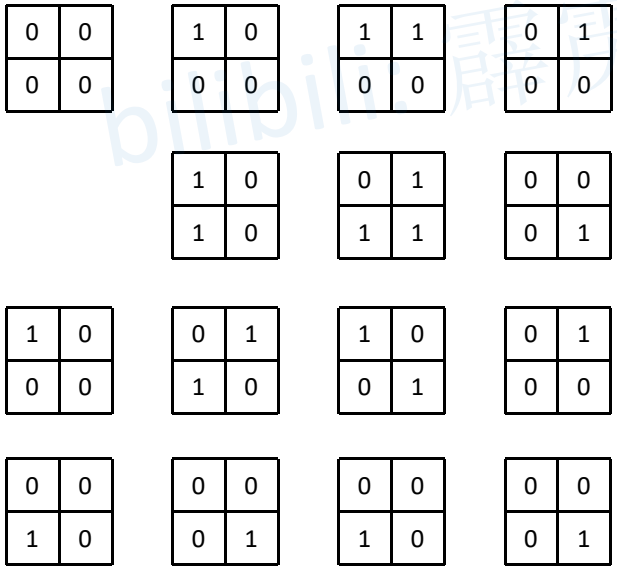


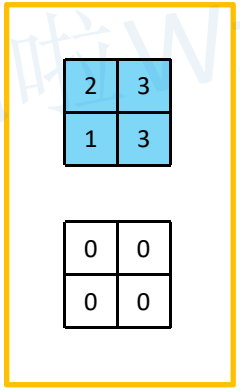


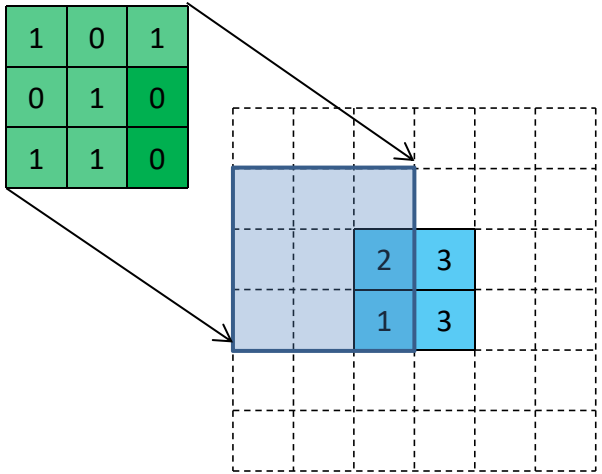
转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

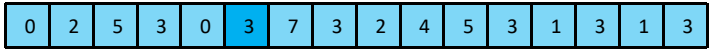
P: 1x16



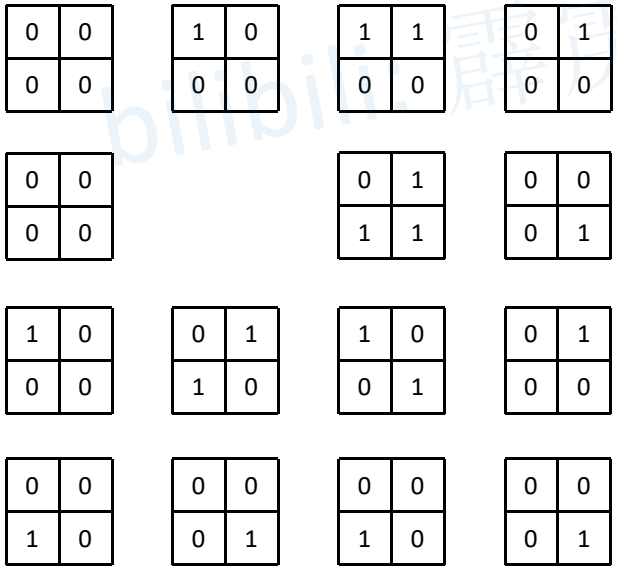


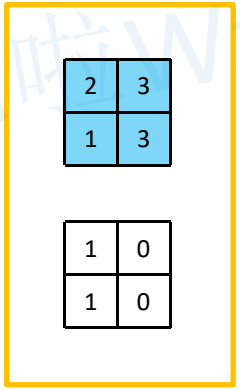


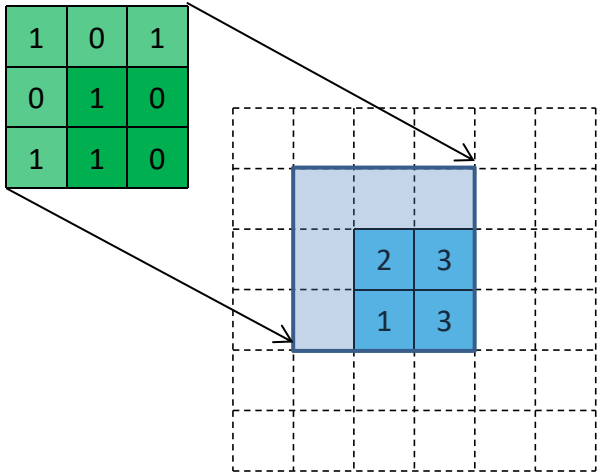
转置卷积


0 2 5 3 0 3 7 3 2 4 5 3 1 3 1 3

P: 1x16


0 0
0 0
1 0
0 0
1 1
0 0
0 1
0 0
0 0
0 0
0 1
1 1
0 0
0 1
1 0
0 0
1 0
0 1
0 1
1 0
0 1
0 0
0 0
1 0
0 1
0 0
1 0
0 0
0 1

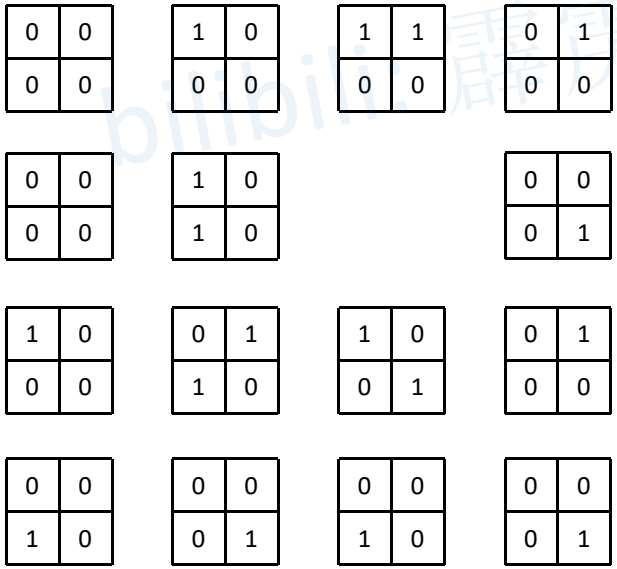

2 3
1 3
1 0
1 0

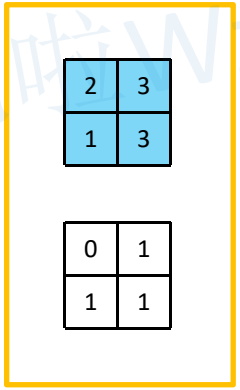

1 0 1
0 1 0
1 1 0
2 3
1 3

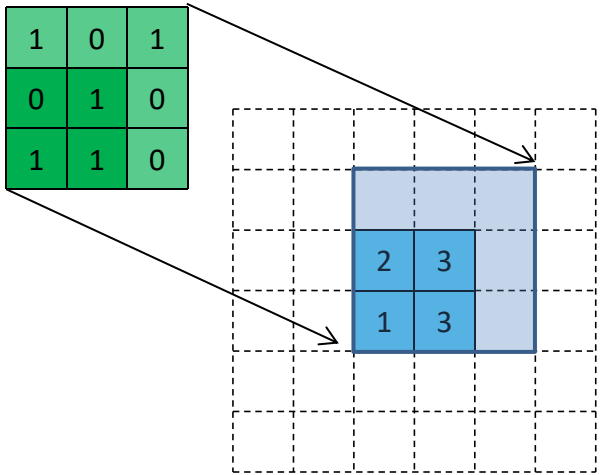
转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

P: 1x16

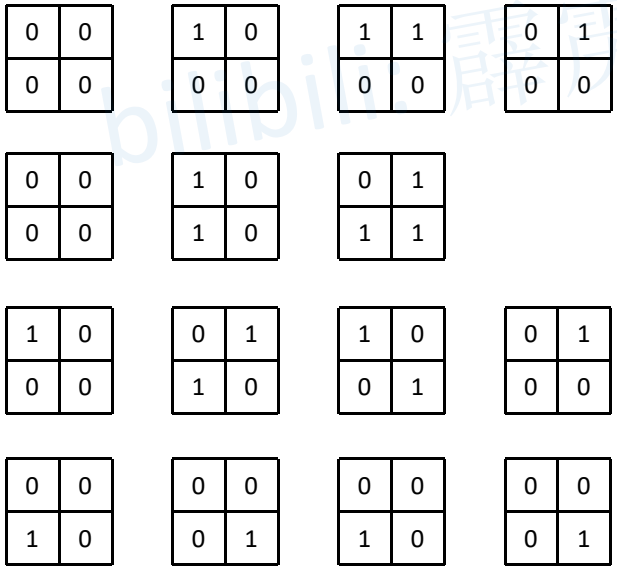


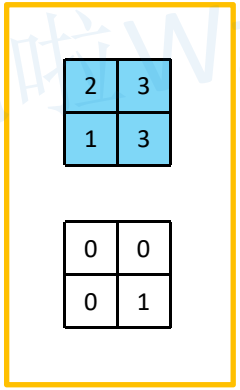


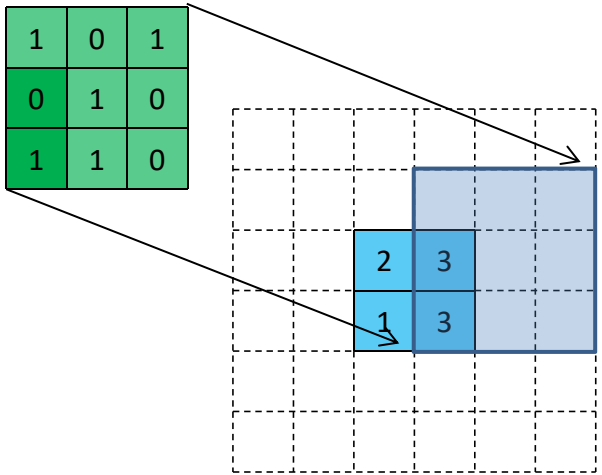


转置卷积

P: 1x16



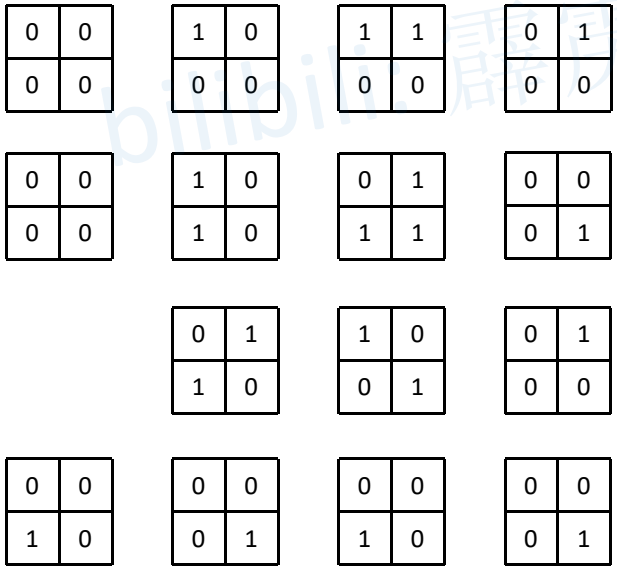


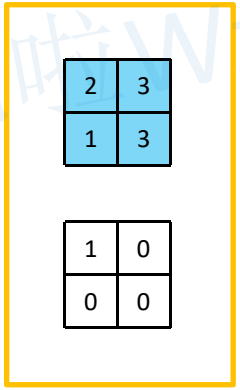


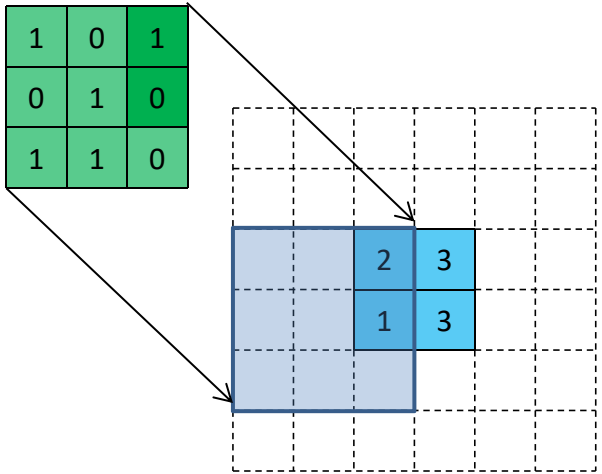
转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

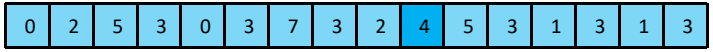
P: 1x16



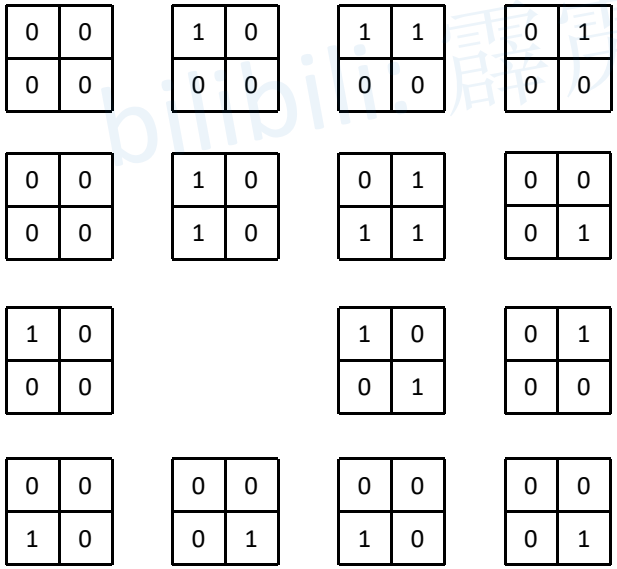


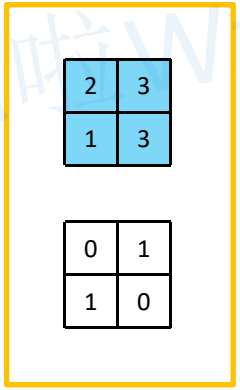


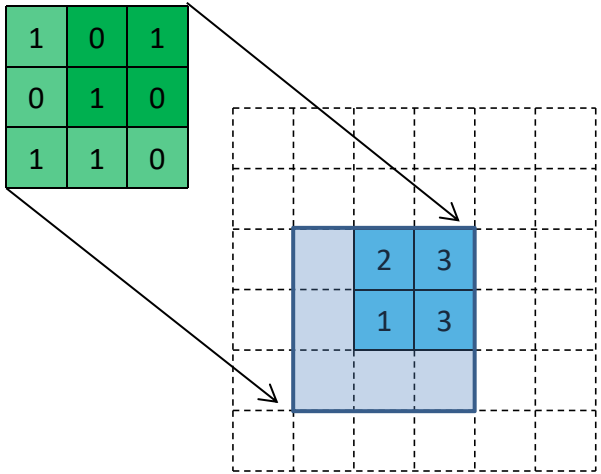
转置卷积


0 2 5 3 0 3 7 3 2 4 5 3 1 3 1 3

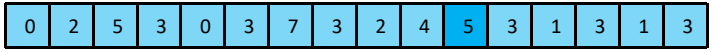
P: 1x16


0 0
0 0
1 0
0 0
1 1
0 0
0 1
0 0
0 0
0 0
1 0
1 0
0 1
1 1
0 0
0 1
1 0
0 0
1 0
0 1
0 0
0 1
0 0
0 1
0 0
1 0
0 0
0 1
1 0
0 1

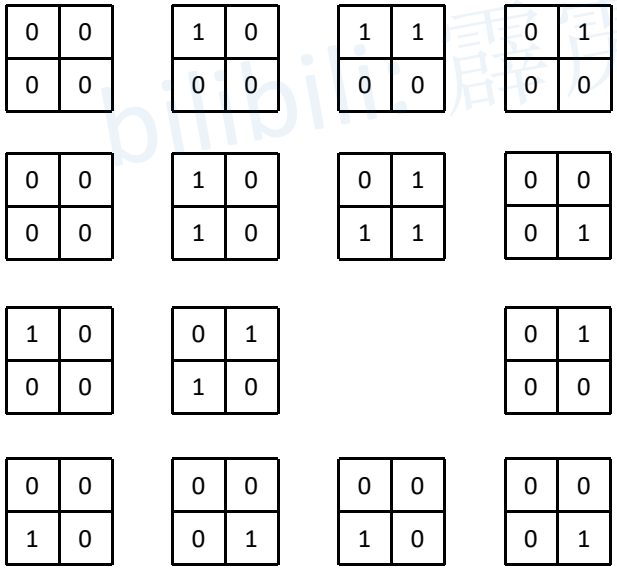

2 3
1 3
0 1
1 0

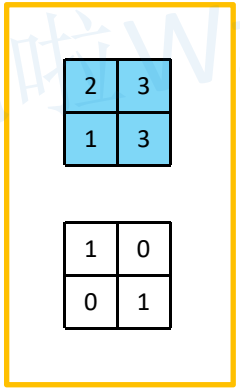

1 0 1
0 1 0
1 1 0
2 3
1 3

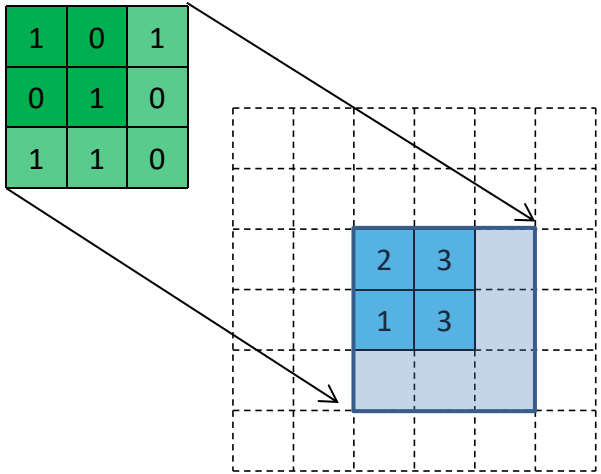
转置卷积


0 2 5 3 0 3 7 3 2 4 5 3 1 3 1 3

P: 1x16


0 0
0 0
1 0
0 0
1 1
0 0
0 1
0 0
0 0
0 0
1 0
1 0
0 1
1 1
0 0
0 1
1 0
0 0
0 1
1 0
0 1
0 1
0 0
0 0
1 0
0 0
0 1
0 0
1 0
0 0
0 1


2 3
1 3
1 0
0 1


1 0 1
0 1 0
1 1 0
2 3
1 3

转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

P: 1x16

0	0
0	0

1	0
0	0

1	1
0	0

0	1
0	0

0	0
0	0

1	0
1	0

0	1
1	1

0	0
0	1

1	0
0	0

0	1
1	0

1	0
0	1

0	0
1	0

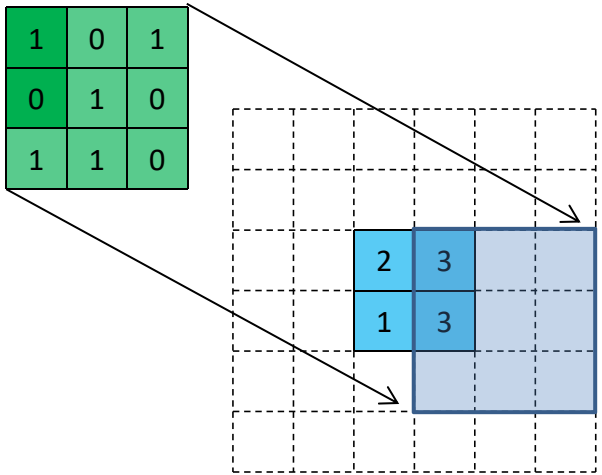
0	0
0	1

0	0
1	0

0	0
0	1

2	3
1	3

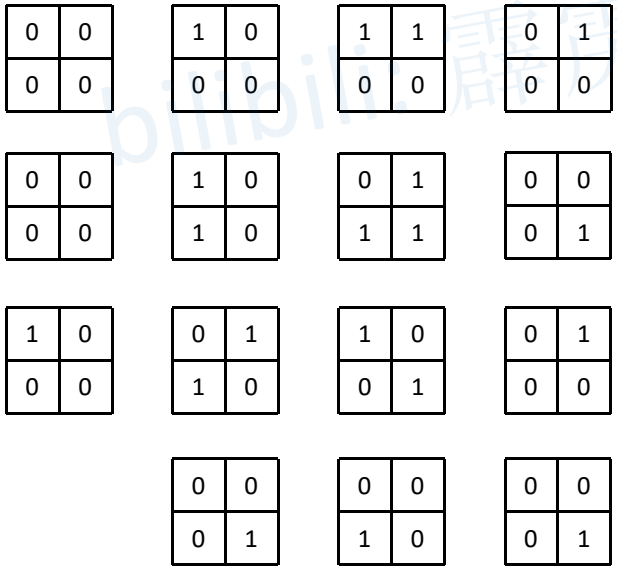
0	1
0	0

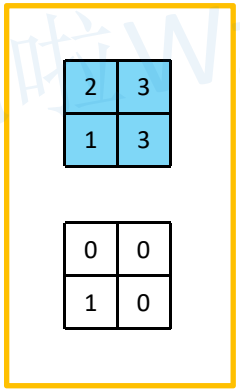


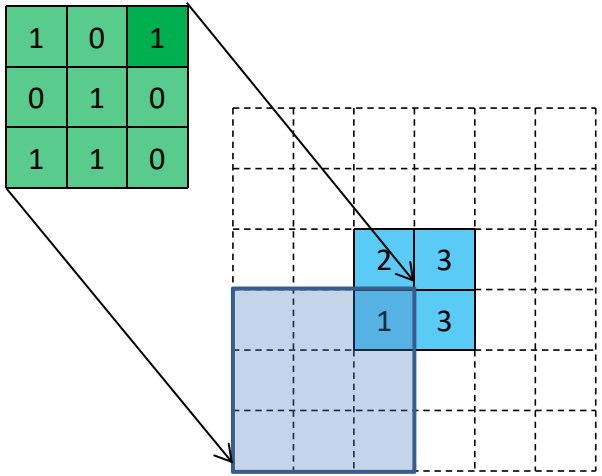
转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

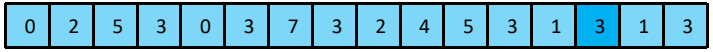
P: 1x16



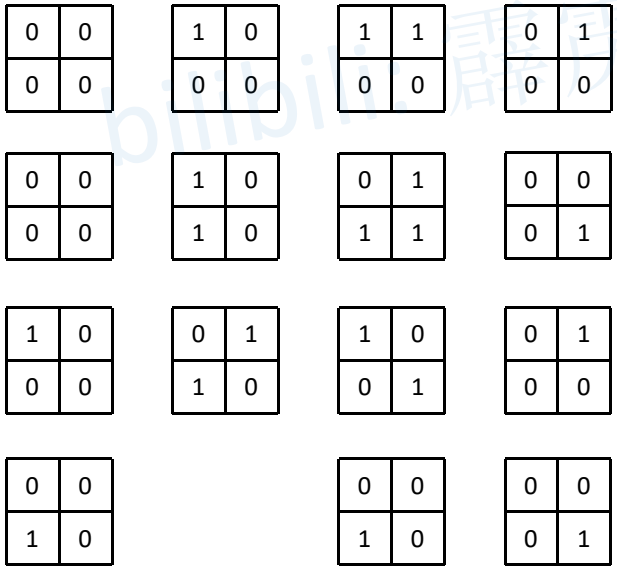


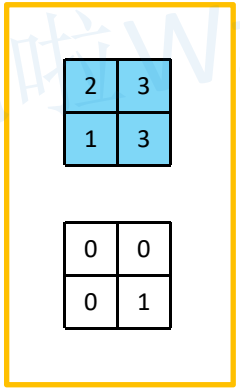


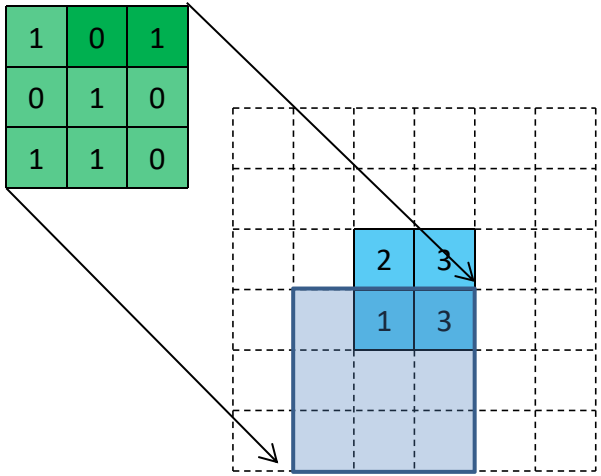
转置卷积


0 2 5 3 0 3 7 3 2 4 5 3 1 3 1 3

P: 1x16


0 0
0 0
1 0
0 0
1 1
0 0
0 1
0 0
0 0
0 0
1 0
1 0
0 1
1 1
0 0
0 1
1 0
0 0
0 1
1 0
0 1
1 0
0 0
0 1
0 0
0 0
1 0
0 0
1 0
0 0
0 1

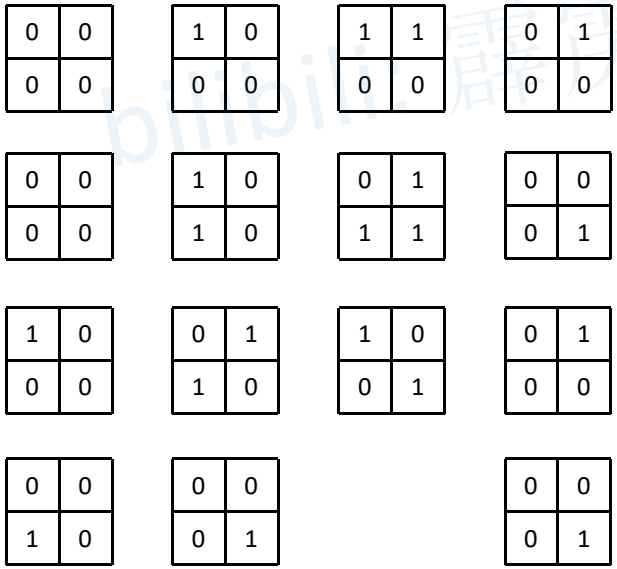

2 3
1 3
0 0
0 1

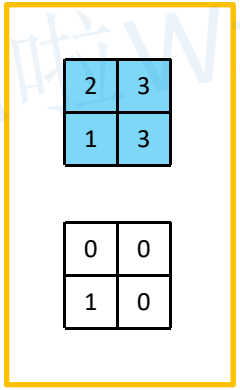

1 0 1
0 1 0
1 1 0
2 3
1 3

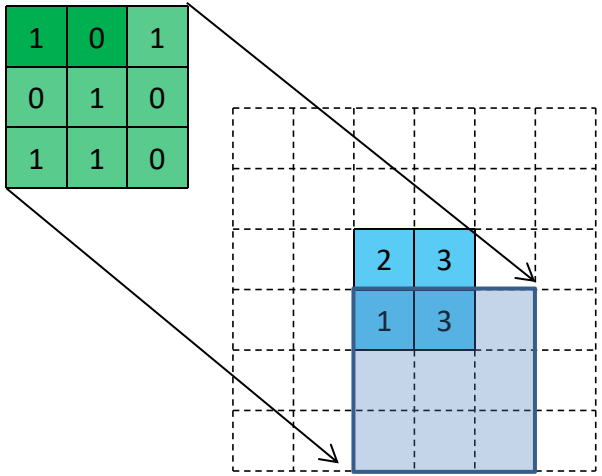
转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

P: 1x16







转置卷积

0	2	5	3	0	3	7	3	2	4	5	3	1	3	1	3

P: 1x16

0	0
0	0

1	0
0	0

1	1
0	0

0	1
0	0

0	0
0	0

1	0
1	0

0	1
1	1

0	0
0	1

1	0
0	0

0	1
1	0

1	0
0	1

0	1
0	0

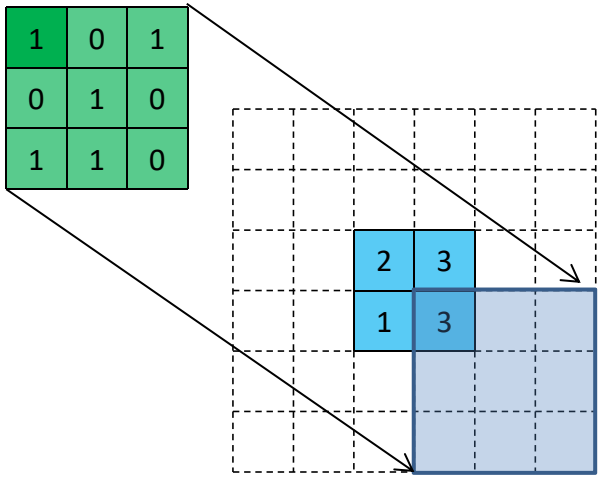
0	0
1	0

0	0
0	1

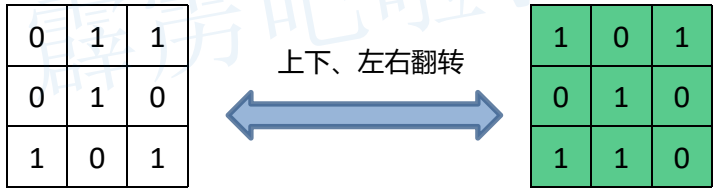
0	0
1	0

2	3
1	3

0	0
0	1



转置卷积

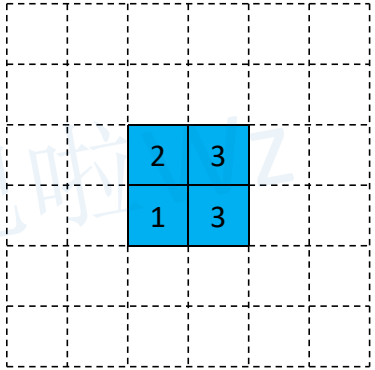


卷积kernel

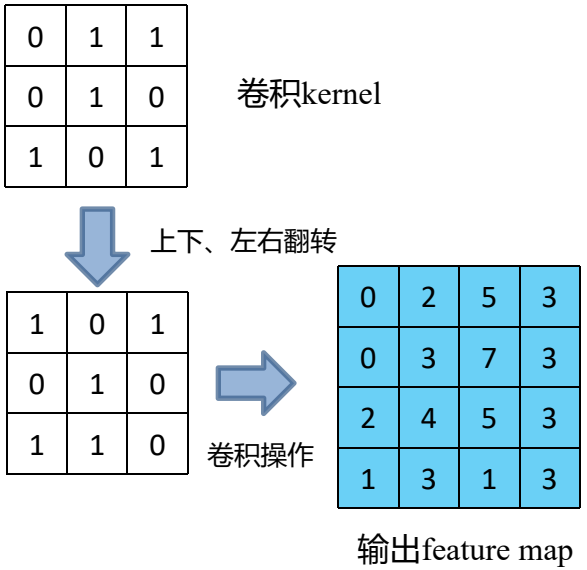
转置卷积

转置卷积运算步骤：

- 在输入特征图元素间填充s-1行、列0
- 在输入特征图四周填充k-p-1行、列0
- 将卷积核参数上下、左右翻转
- 做正常卷积运算（填充0，步距1）



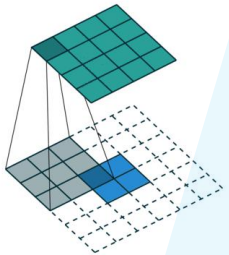
输入feature map



$$H_{out} = (H_{in} - 1) \times stride[0] - 2 \times padding[0] + kernel\_size[0]$$

$$W_{out} = (W_{in} - 1) \times stride[1] - 2 \times padding[1] + kernel\_size[1]$$

https://github.com/vdumoulin/conv_arithmetic



s=1, p=0, k=3

沟通方式

1.github

https://github.com/WZMIAOMIAO/deep-learning-for-image-processing

2.bilibili

https://space.bilibili.com/18161609/channel/index

3.CSDN

https://blog.csdn.net/qq_37541097/article/details/103482003