Transposed Convolution

Monday, May 30, 2022 6:50 PM

https://d2l.ai/chapter_computer-vision/transposed-conv.html
https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md
https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html

It is not a deconvolution operator. I.e., it is not the inverse of convolution.

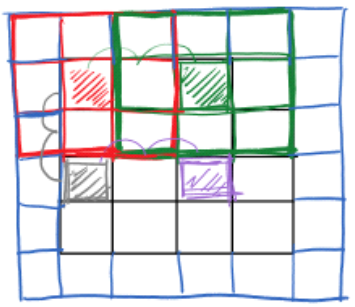
A layer that can increase (upsample) the spatial dimension

- Pixel-wise prediction problems such as semantic segmentation, super resolution

Examples

Consider a convolution operator: k3s2p1
Input size = 4×4
Output size $= \left\lfloor \frac{4+2\times1-3}{2} \right\rfloor + 1 = 2 => 2 \times 2$

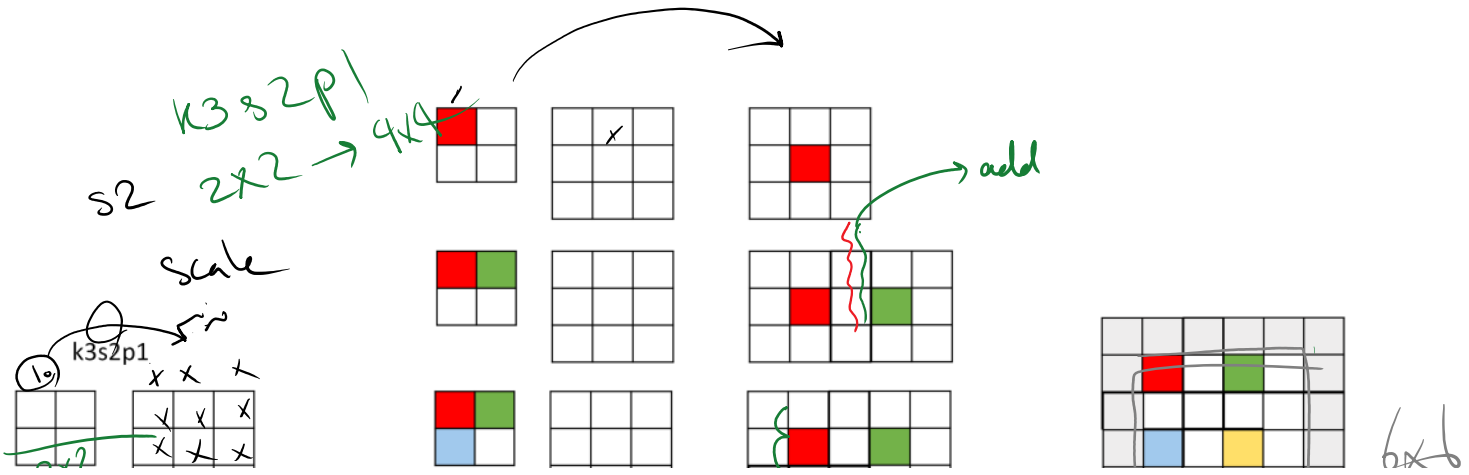
$s = \frac{\#\ moves\ in\ the\ input\ FM}{\#\ moves\ in\ the\ output\ FM}$

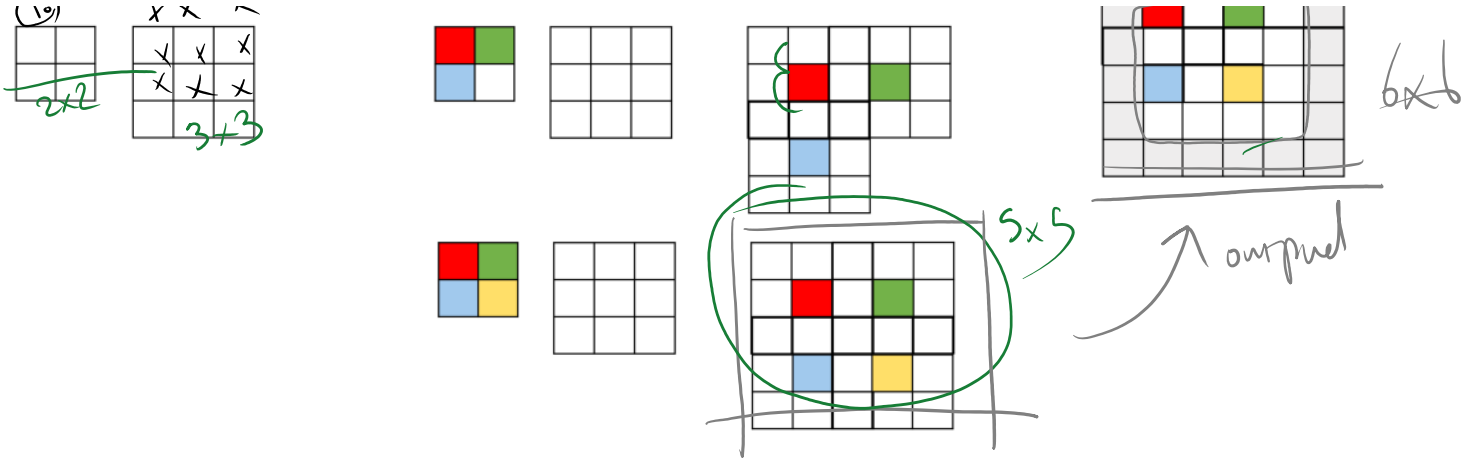


We can denote its corresponding transposed convolution as k3s2p1

But, it is in fact a fractionally strided convolution:

$s = \frac{\#\ moves\ in\ the\ input\ FM}{\#\ moves\ in\ the\ output\ FM} = 0.5 \Rightarrow s' = \frac{2}{1}$





jupyter transposed_convolution Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Help

Run Code

In [21]:
```
import torch
import torch.nn as nn
```

In [22]:
```
inp = torch.randn(1, 3, 2, 2)
```

Pad the output:

In [23]:
```
k = (3, 3)   # Size of the convolving kernel
s = (2, 2)   # Stride for the cross-correlation
op = (1, 1)  # The additional size added to one side of each dimension in the output
convT = nn.ConvTranspose2d(3, 1, kernel_size=k, stride=s, output_padding=op)
```

In [24]:
```
out = convT(inp)
```

In [25]:
```
out.shape
```

Out[25]:
```
torch.Size([1, 1, 6, 6])
```

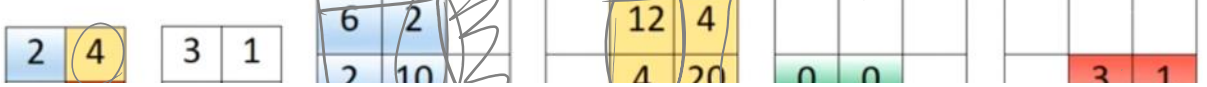
Transposed Convolutions

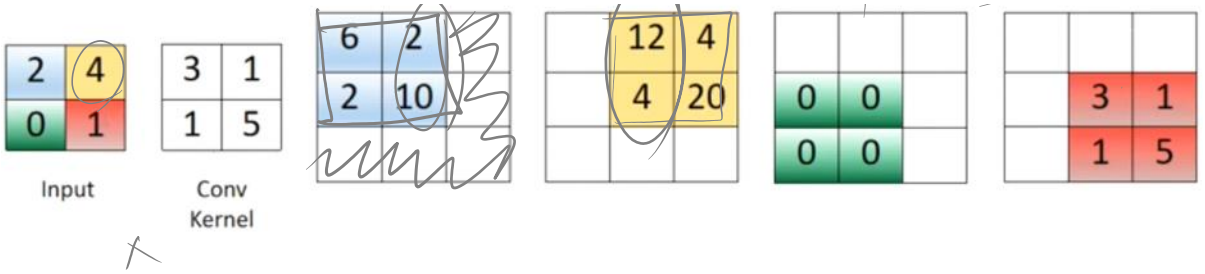
2x2 convolution, stride of 1 and a pad of 0

4 * 3 = 12
2 * 1 = 2
12 + 2 = 14

6	14	4
2	17	21
0	1	5

Output




2 4
0 1
Input
3 1
1 5
Conv
Kernel
6 2
2 10
12 4
4 20
0 0
0 0
3 1
1 5