8, Transposed Convolutions

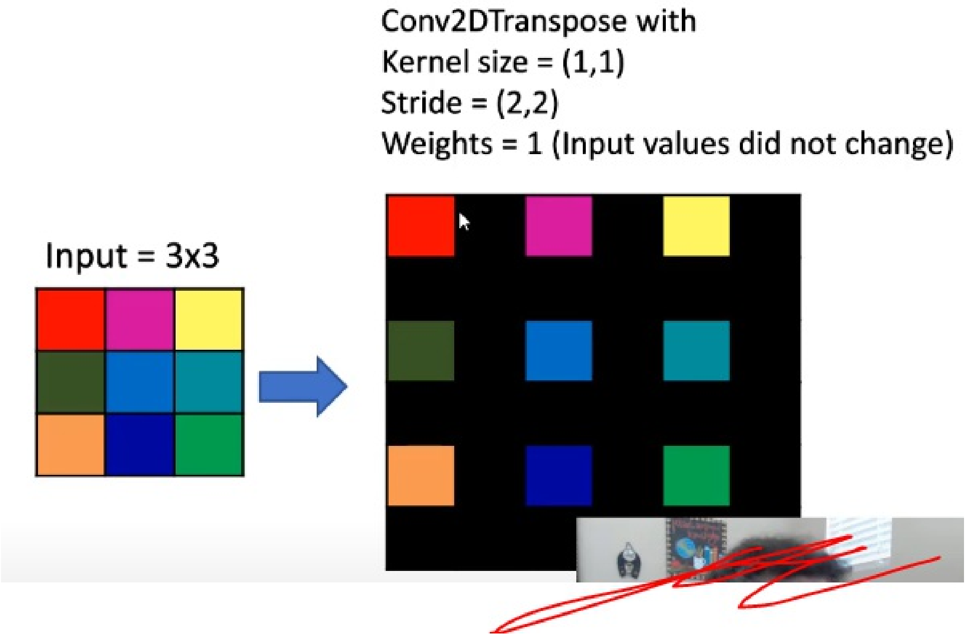
□ Outline of this video

□Notes

▼ from DigitalSreeni

- 2D convolutions: `nn.Conv2d` requires the number of input and output channels, as well as the kernel size.

  ▼ 2D transposed convolutions (aka deconvolutions): `nn.ConvTranspose2d` also requires the number of input and output channels, as well as the kernel size >>similar to UpSampling()



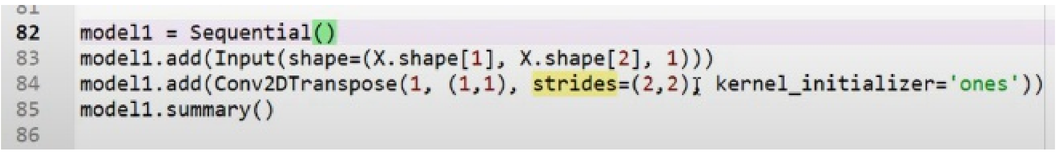
Conv2DTranspose with
Kernel size = (1,1)
Stride = (2,2)
Weights = 1 (Input values did not change)

Input = 3x3

`stride = (2,2)` >> means, doubling the size 3×33 \times 33 × 3 to 6×66 \times 66 × 6

▼ need to define `weights`! here `weights = 1` (so it is not random weight)

In Keras, `kernel_initializer =' ones'` >> assign weights = 1



```
model1 = Sequential()
model1.add(Input(shape=(X.shape[1], X.shape[2], 1)))
model1.add(Conv2DTranspose(1, (1,1), strides=(2,2), kernel_initializer='ones'))
model1.summary()
```

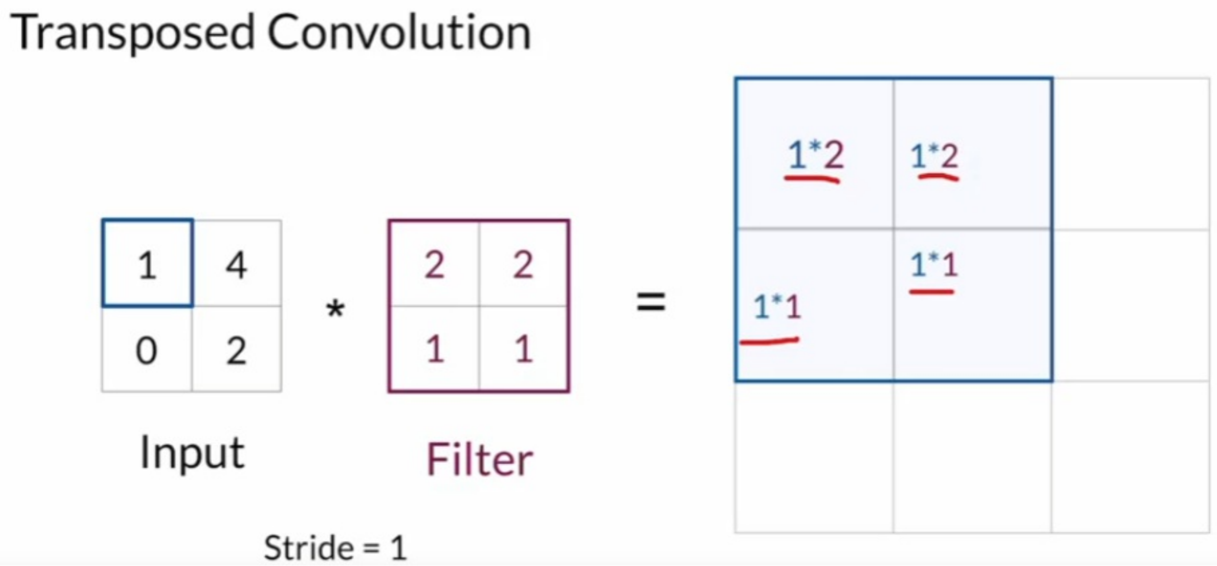
what is convolution? In convolution, u have a kernel, and you multiply that kernel to image

▼ what is transposed convolution?

> A transposed convolution learns a filter to upsample.

▼ How to perform Transposed Convolution?

  ▼ step 1



Transposed Convolution

Input

1	4
0	2

*

Filter

2	2
1	1

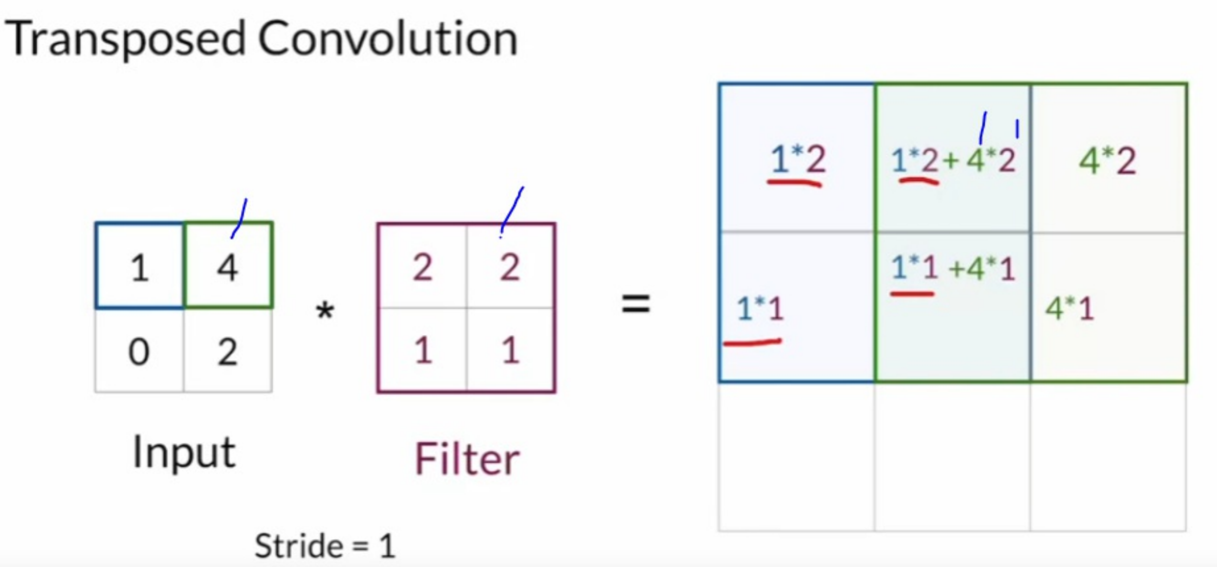
=

1*2	1*2	
1*1	1*1	

Stride = 1

> You start by taking the top-left value from your input and getting its product with every value in the two-by-two filter. Then you save this value in the top two-by-two left corner of your output. Here are the multiplied values from that operation.

▼ step 2



Transposed Convolution

Input

1	4
0	2

*

Filter

2	2
1	1

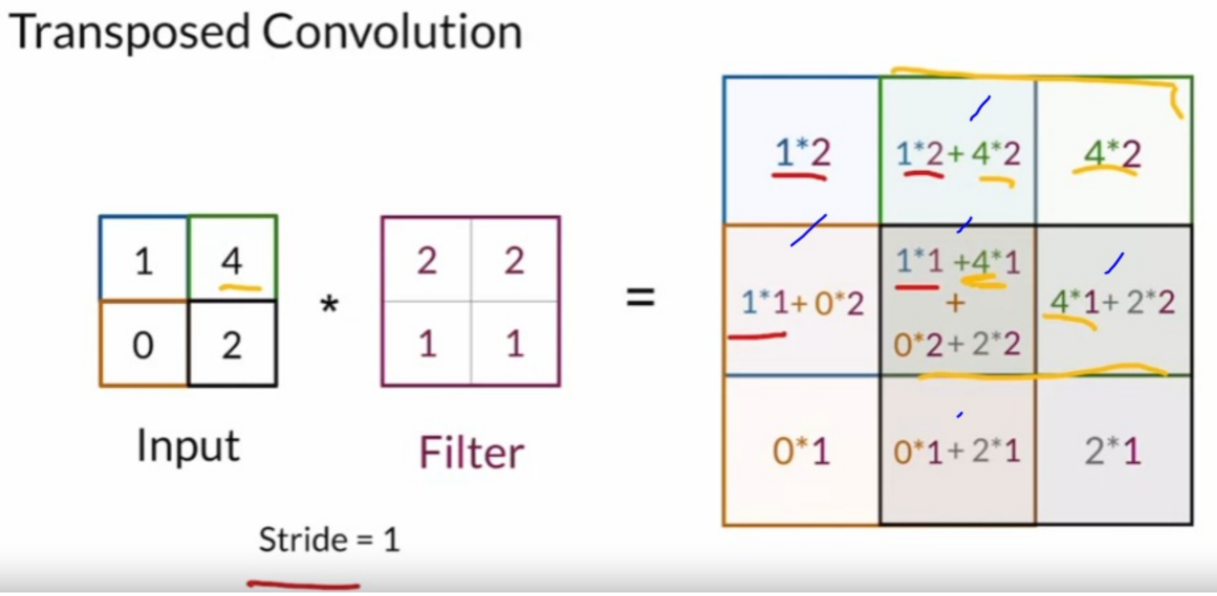
=

1*2	1*2+4*2	4*2
1*1	1*1 +4*1	4*1

Stride = 1

> Next, you shift the filter by your stride of one and you repeat the same process on this next pixel and now it's at the top right corner of your input, as well as these four pixels out here in your output. You keep the products there and where there is overlap, you add it to the previous product.

▼ the rest



Transposed Convolution

Input

1	4
0	2

*

Filter

2	2
1	1

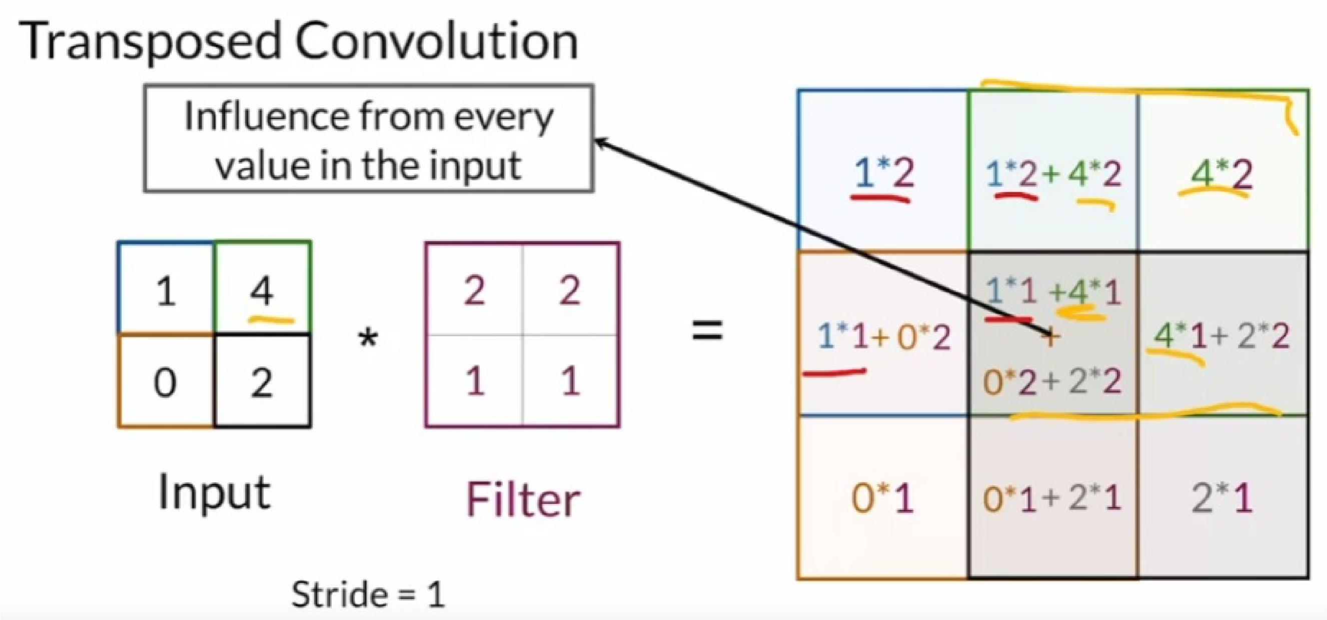
=

1*2	1*2+4*2	4*2
1*1+0*2	1*1 +4*1 + 0*2+2*2	4*1+2*2
0*1	0*1+2*1	2*1

Stride = 1

> After that, you move to the bottom left corner of your input and take the product with the filter, send the result, and so on and so forth until you've covered your entire input. With this computation, you can see that some of the values in the output are influenced much more heavily by the values from the input.

▼ we can see a problem here could happen in Transposed Convolution (checkerboard problem)



Transposed Convolution

Influence from every value in the input

Input

1	4
0	2

*

Filter

2	2
1	1

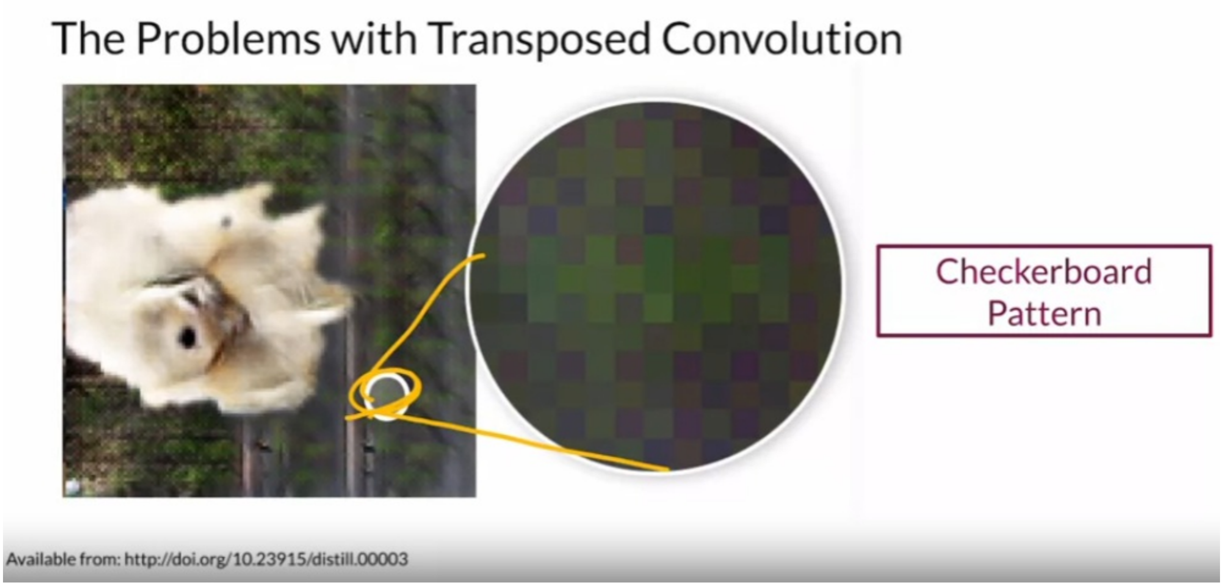
=

1*2	1*2+4*2	4*2
1*1+0*2	1*1 +4*1 + 0*2+2*2	4*1+2*2
0*1	0*1+2*1	2*1

Stride = 1

> For instance, the center pixel in the output is influenced by all the values in the input, while the corners are influenced by just one value.

▼ example



The Problems with Transposed Convolution

Available from: http://doi.org/10.23915/distill.00003

▼ Main difference between transposed convolution?

> Upsampling infers pixels using a predefined method, while transposed convolution learns a filter.

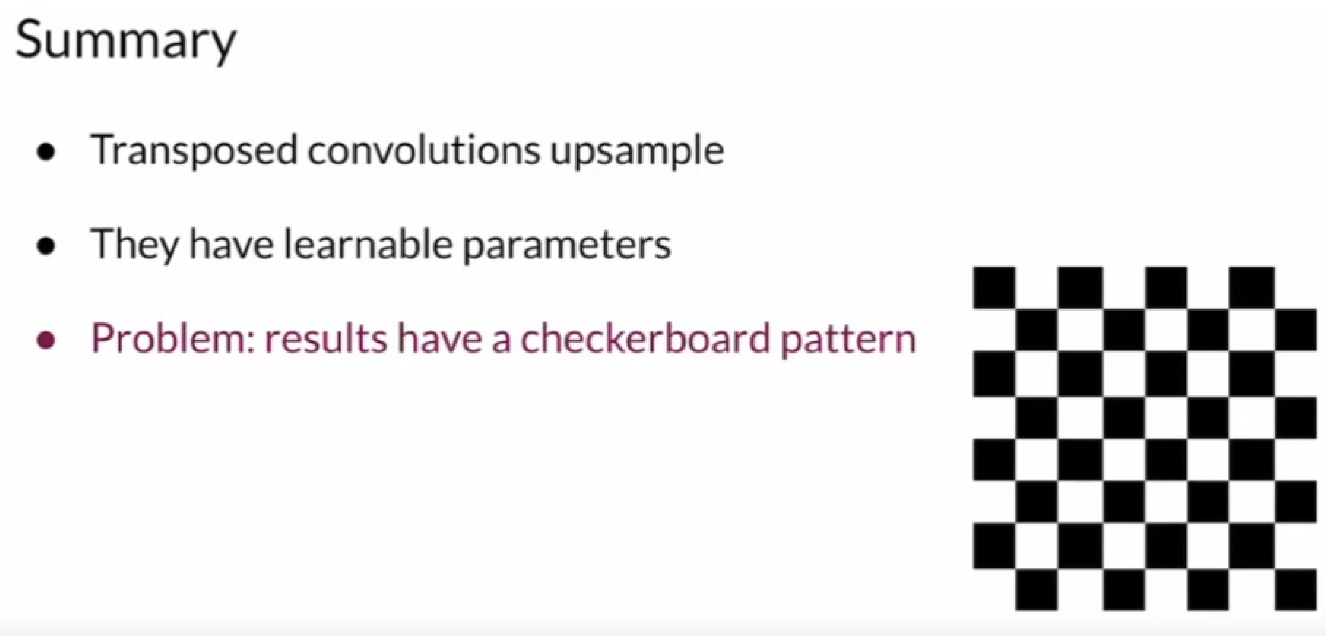
>>the filter is learned while the process of training, while Upsampling (i.e Nearest Neighbour) has no learned parameter

Vocabs

QOTD

Summary

▼ summary



Summary

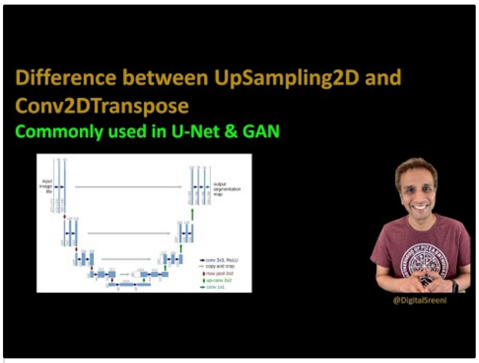
- Transposed convolutions upsample
- They have learnable parameters
- Problem: results have a checkerboard pattern

URL

218 - Difference between UpSampling2D and Conv2DTranspose used in U-Net and GAN

Difference between UpSampling2D and Conv2DTransposeThese are the two common types of layers that can be used to increase the dimensions of arrays.UpSampling2...

https://www.youtube.com/watch?v=fMwti6zFcYY&t=949s


Difference between UpSampling2D and Conv2DTranspose
Commonly used in U-Net & GAN
@DigitalSreeni