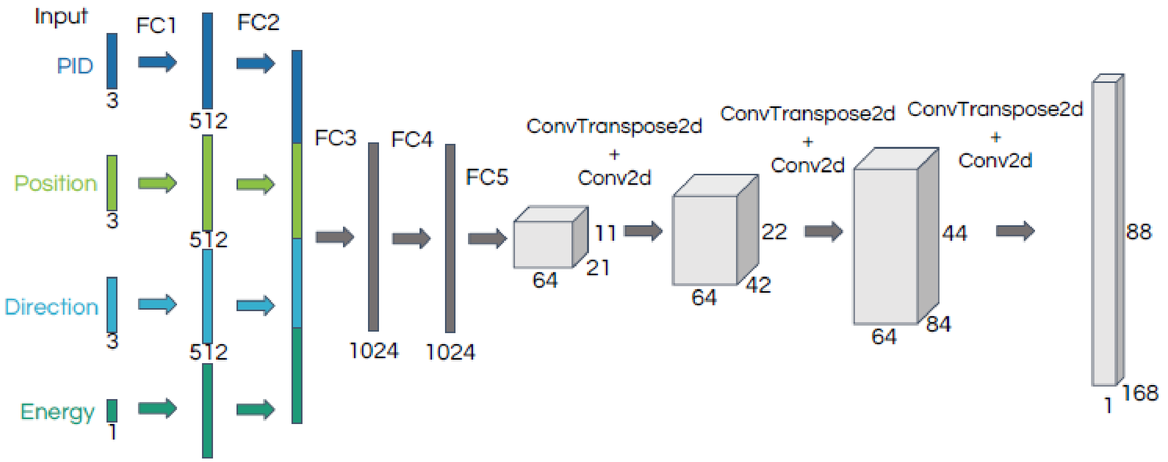
Conv2d and ConvTransposed2d

Chiaki Yanagisawa

Water Cherenkov with Deep Learning Zoom meeting

2/19/2021

Our Network




```
self._upconvs = torch.nn.Sequential(
        torch.nn.ConvTranspose2d(64, 64, 4, 2),  torch.nn.ReLU(),
                                                                        # 24 x 44
        torch.nn.Conv2d(64, 64, 3), torch.nn.ReLU(),       # 22 x 42

        torch.nn.ConvTranspose2d(64, 32, 4, 2), torch.nn.ReLU(),
                                                                        # 46 x 86
        torch.nn.Conv2d(32, 32, 3),  torch.nn.ReLU(),      # 44 x 84

        torch.nn.ConvTranspose2d(32, 32, 4, 2), torch.nn.ReLU(),
                                                                        # 90 x 170
        torch.nn.Conv2d(32, 3, 3)                                # 88 x 168
)
```

Convolution

Conv2d

class torch.nn.Conv2d(*in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros'*)

Input : $(N, C_{in}, H_{in}, W_{in})$ N = batch size, C_{in} = input channels, H_{in} = input height, W_{in} = input height
Output: $(N, C_{out}, H_{out}, W_{out})$ N = batch size, C_{out} = output channels, H_{out} = output height, W_{out} = output height

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

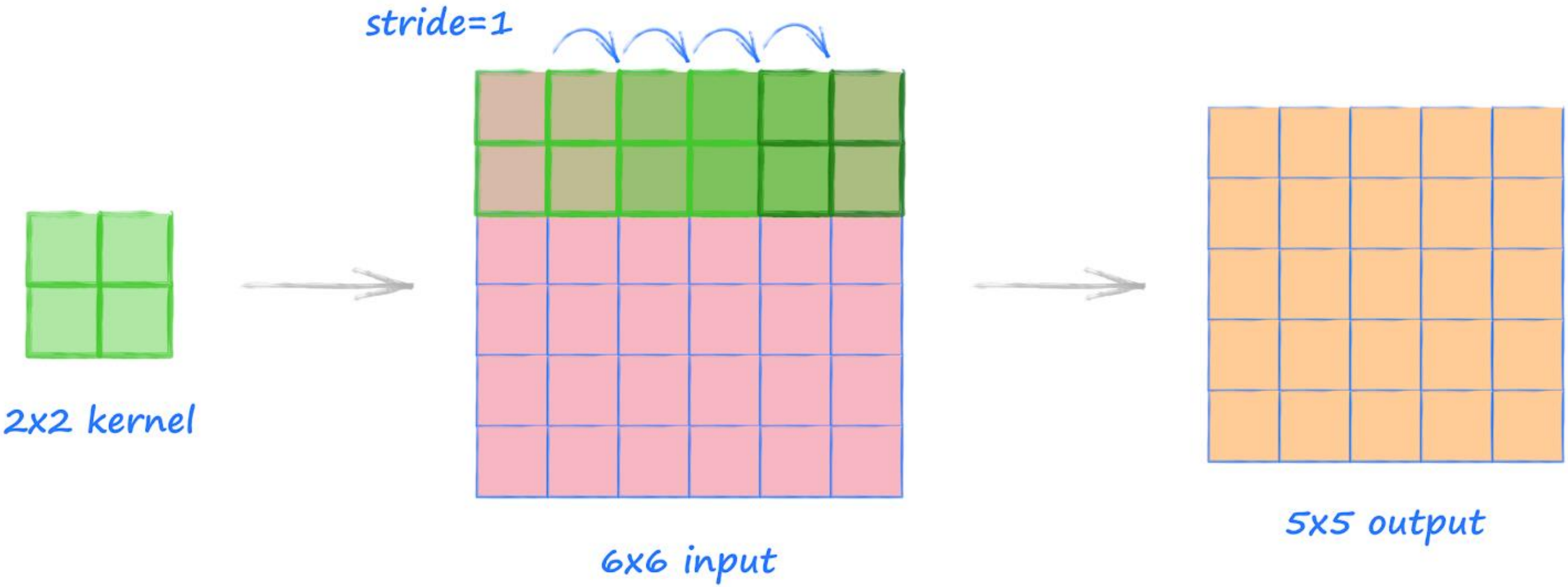
$$W_{out} = \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1$$

Convolution

Example 1: Convolution With Stride 1, No Padding

`nn.Conv2d(in_channels, out_channels, kernel_size=2, stride=1)`

In this first simple example we apply a **2 by 2** kernel to an input of size **6 by 6**, with stride **1**.

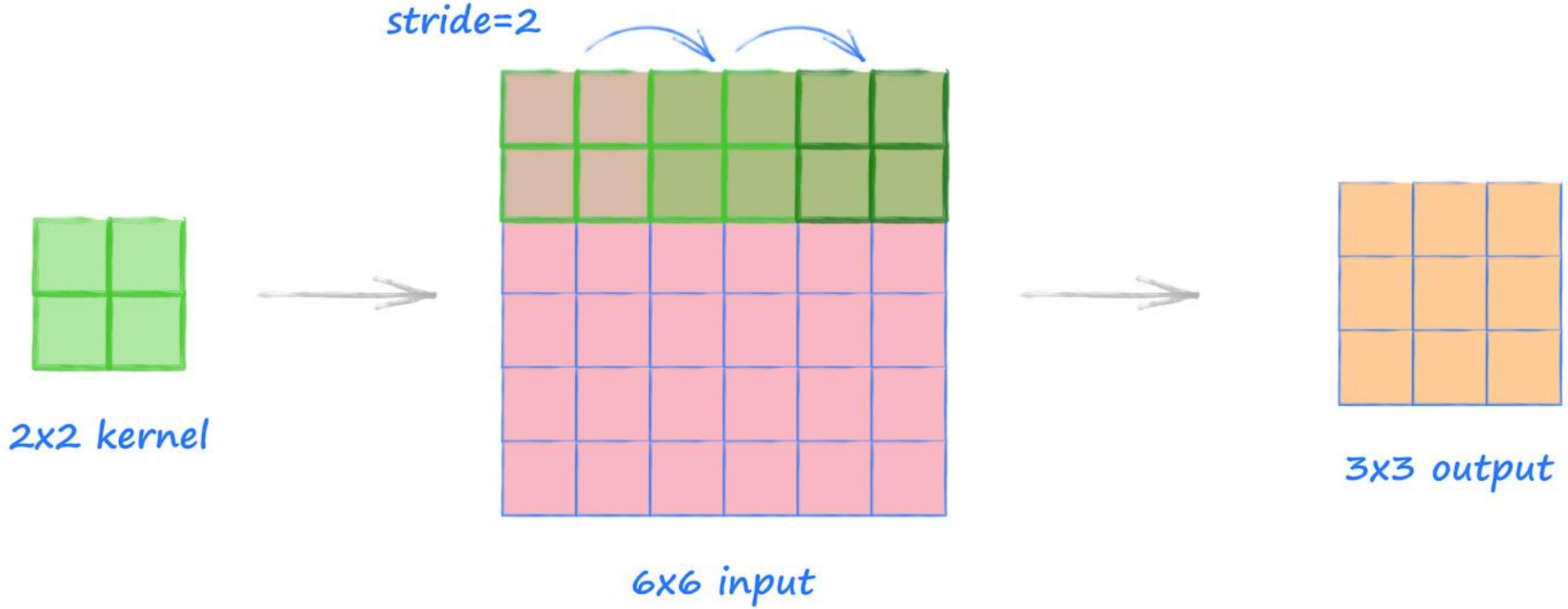


Convolution

Example 2: Convolution With Stride 2, No Padding

```
nn.Conv2d(in_channels, out_channels, kernel_size=2, stride=2)
```

This second example is the same as the previous one, but we now have a stride of **2**.

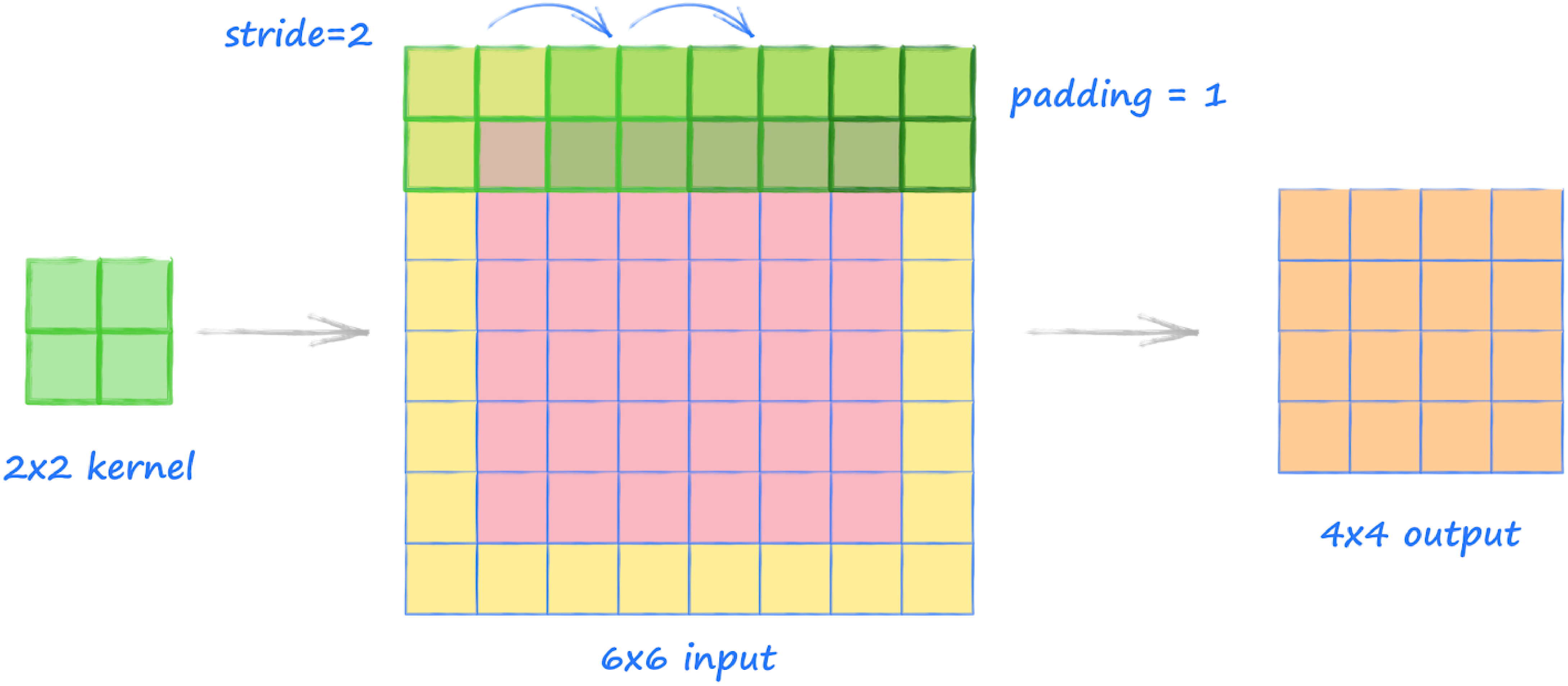


Convolution

Example 3: Convolution With Stride 2, With Padding

`nn.Conv2d(in_channels, out_channels, kernel_size=2, stride=2, padding=1)`

This third example is the same as the previous one, but this time we use a padding of **1**.



Transposed Convolution

ConvTransposed2d

class torch.nn.ConvTranspose2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, output_padding: Union[T, Tuple[T, T]] = 0, groups: int = 1, bias: bool = True, dilation: int = 1, padding_mode: str = 'zeros')

Input : $(N, C_{in}, H_{in}, W_{in})$ N = batch size, C_{in} = input channels, H_{in} = input height, W_{in} = input height
Output: $(N, C_{out}, H_{out}, W_{out})$ N = batch size, C_{out} = output channels, H_{out} = output height, W_{out} = output height

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

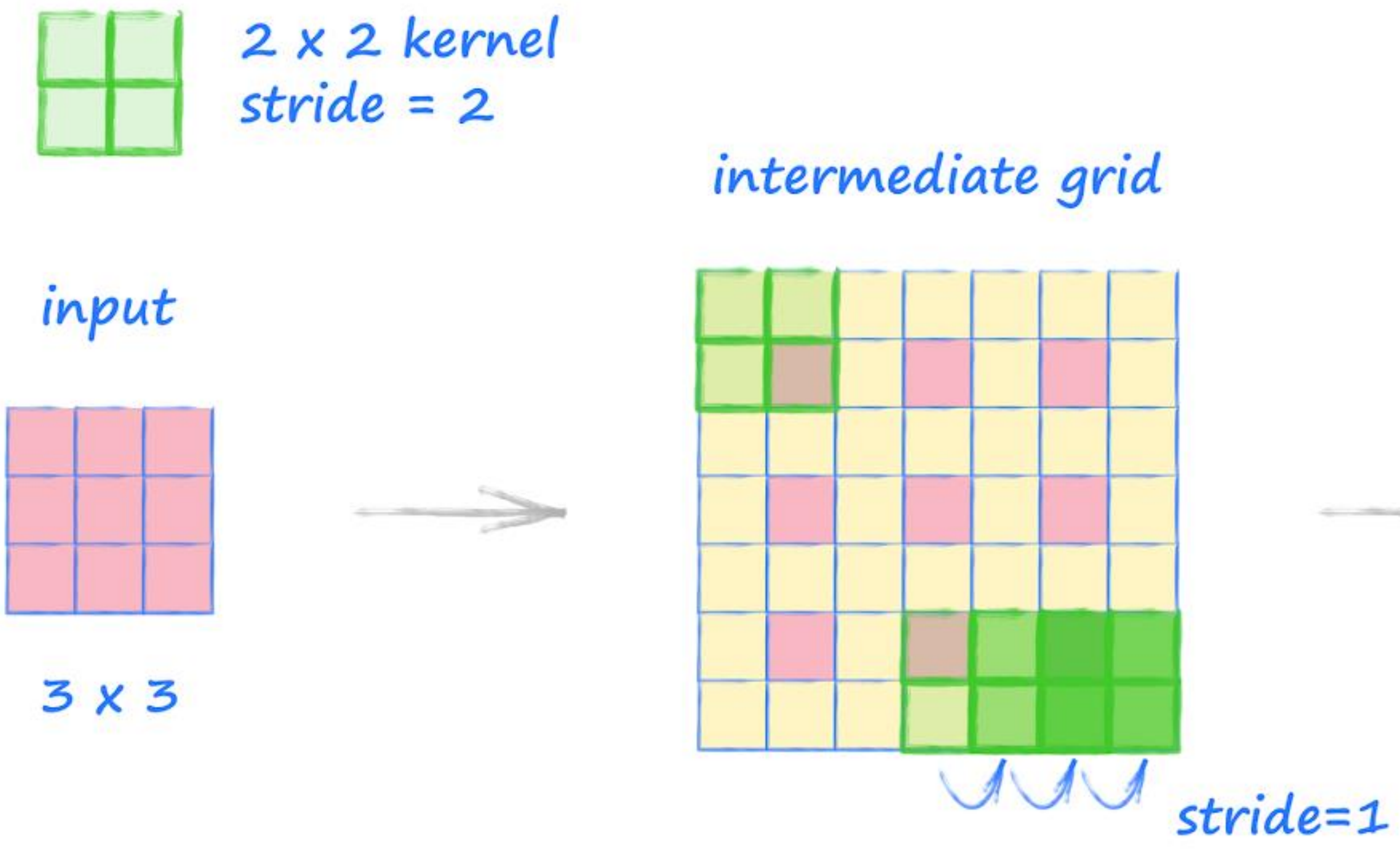
$$W_{out} = (W_{in} - 1) \times \text{stride}[1] - 2 \times \text{padding}[1] + \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) + \text{output\_padding}[1] + 1$$

Transposed Convolution

Example 5: Transpose Convolution With Stride 2, No Padding

`nn.ConvTranspose2d(in_channels, out_channels, kernel_size=2, stride=2, padding=0)`

The transpose convolution is commonly used to expand a tensor to a larger tensor. This is the opposite of a normal convolution which is used to reduce a tensor to a smaller tensor.

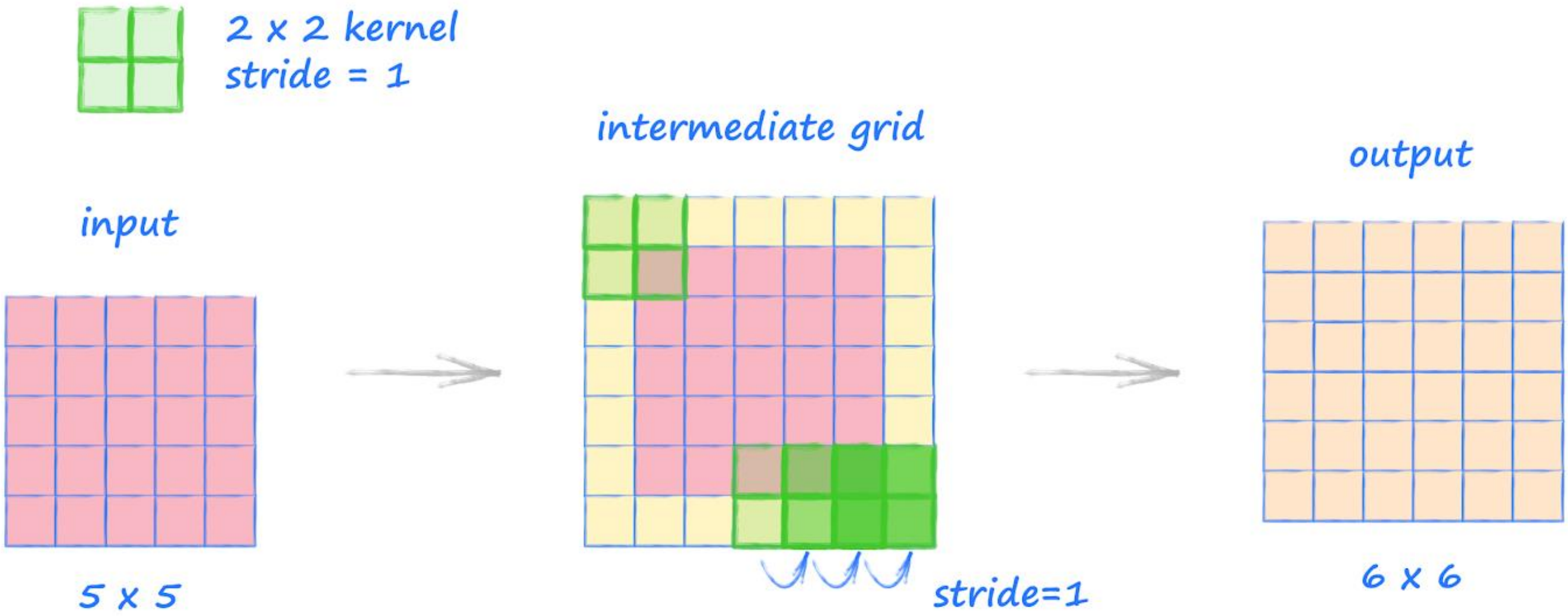


Transposed Convolution

Example 6: Transpose Convolution With Stride 1, No Padding

`nn.ConvTranspose2d(in_channels, out_channels, kernel_size=2, stride=1, padding=0)`

In the previous example we used a stride of **2** because it is easier to see how it is used in the process. In this example we use a stride of **1**.

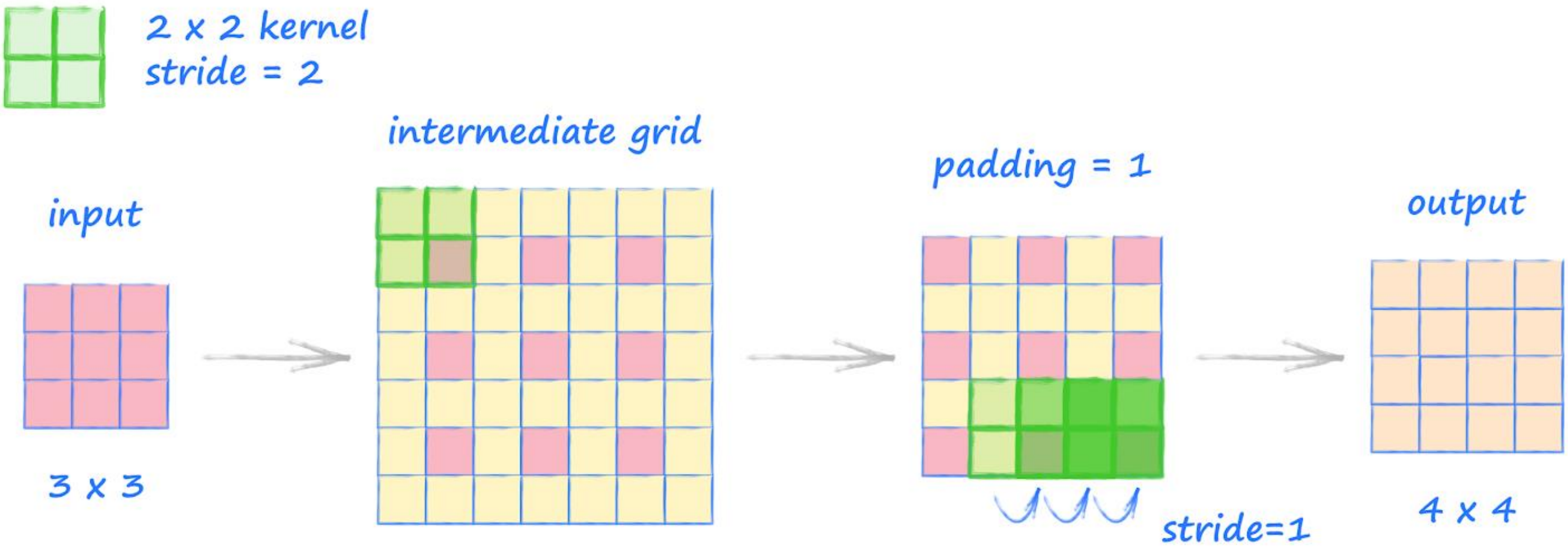


Transposed Convolution

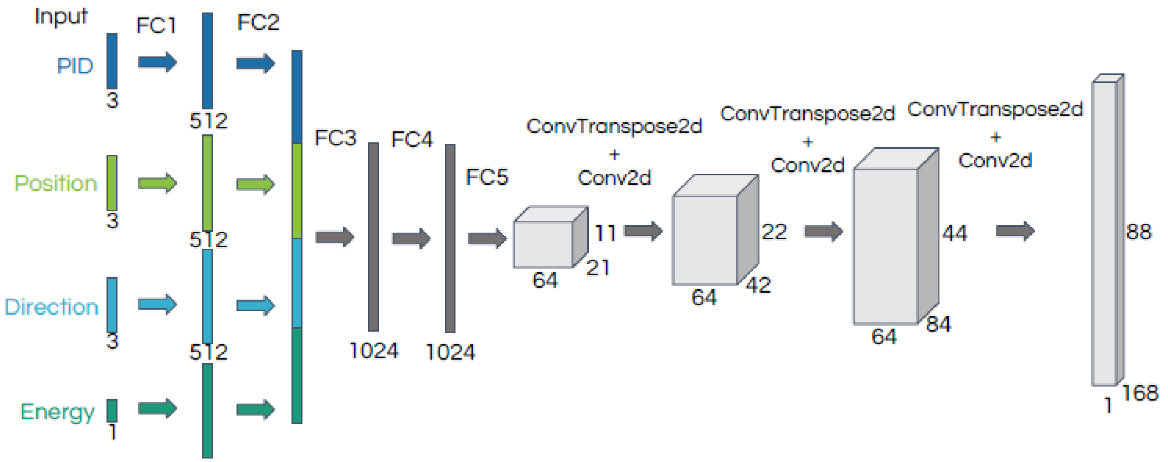
Example 7: Transpose Convolution With Stride 2, With Padding

`nn.ConvTranspose2d(in_channels, out_channels, kernel_size=2, stride=2, padding=1)`

In this transpose convolution example we introduce padding. Unlike the normal convolution where padding is used to expand the image, here it is used to reduce it.



Our Network




```
self._upconvs = torch.nn.Sequential(
        torch.nn.ConvTranspose2d(64, 64, 4, 2),  torch.nn.ReLU(),
                                                        # 24 x 44
        torch.nn.Conv2d(64, 64, 3), torch.nn.ReLU(),    # 22 x 42

        torch.nn.ConvTranspose2d(64, 32, 4, 2), torch.nn.ReLU(),
                                                        # 46 x 86
        torch.nn.Conv2d(32, 32, 3),  torch.nn.ReLU(),   # 44 x 84

        torch.nn.ConvTranspose2d(32, 32, 4, 2), torch.nn.ReLU(),
                                                        # 90 x 170
        torch.nn.Conv2d(32, 3, 3)                       # 88 x 168
    )
```

Upconv Layers of Our Network

First layer

Input ch = 64, output ch = 64, H_{in} = 11, W_{in} = 21, kernel size = 4, stride = 2, dilation = 1, padding = 0, output padding = 0

ConvTranse2d(64, 64, 4, 2) : $H_{out} = (11 – 1) \times 2 – 2 \times 0 + 1 \times (4 – 1) + 0 + 1 = 24$

$W_{out} = (21 – 1) \times 2 – 2 \times 0 + 1 \times (4 – 1) + 0 + 1 = 44$

Input ch = 64, output ch = 64, H_{in} = 24, W_{in} = 44, kernel size = 3, stride = 1, dilation = 1, padding = 0, output padding = 0

Conv2d(64, 64, 3) : $H_{out} = [24 – 2 \times 0 - 1 \times (3 – 1) – 1]/1 + 1 = 22$

$W_{out} = [44 – 2 \times 0 - 1 \times (3 – 1) – 1]/1 + 1 = 42$

Upconv Layers of Our Network

Second layer

Input ch = 64, output ch = 32, H_{in} = 22, W_{in} = 42, kernel size = 4, stride = 2, dilation = 1, padding = 0, output padding = 0

ConvTranse2d(64, 32, 4, 2) : $H_{out} = (22 – 1) \times 2 – 2 \times 0 + 1 \times (4 – 1) + 0 + 1 = 46$

$W_{out} = (42 – 1) \times 2 – 2 \times 0 + 1 \times (4 – 1) + 0 + 1 = 86$

Input ch = 32, output ch = 32, H_{in} = 46, W_{in} = 86, kernel size = 3, stride = 1, dilation = 1, padding = 0, output padding = 0

Conv2d(32, 32, 3) : $H_{out} = [46 – 2 \times 0 - 1 \times (3 – 1) – 1]/1 + 1 = 44$

$W_{out} = [86 – 2 \times 0 - 1 \times (3 – 1) – 1]/1 + 1 = 84$

Upconv Layers of Our Network

Third layer

Input ch = 32, output ch = 32, H_{in} = 44, W_{in} = 84, kernel size = 4, stride = 2, dilation = 1, padding = 0, output padding = 0

ConvTranse2d(64, 32, 4, 2) : $H_{out} = (44 – 1) \times 2 – 2 \times 0 + 1 \times (4 – 1) + 0 + 1 = 90$

$W_{out} = (84 – 1) \times 2 – 2 \times 0 + 1 \times (4 – 1) + 0 + 1 = 170$

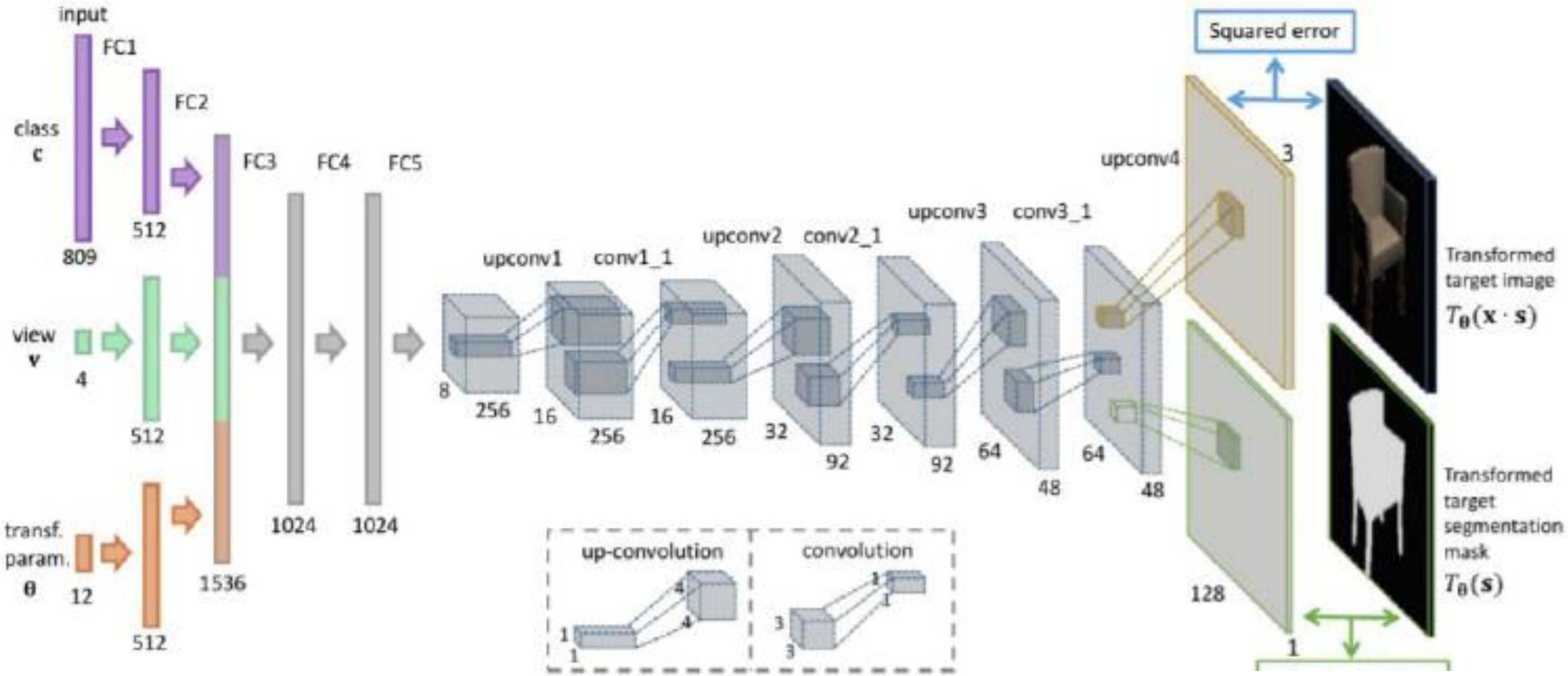
Input ch = 32, output ch = 1, H_{in} = 90, W_{in} = 170, kernel size = 3, stride = 1, dilation = 1, padding = 0, output padding = 0

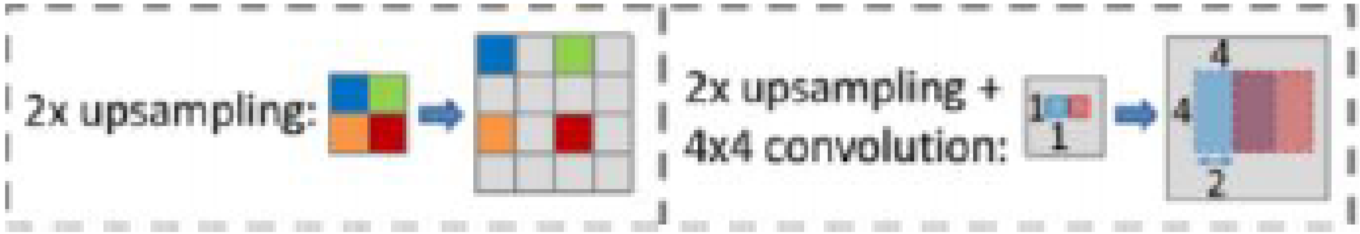
Conv2d(32, 1, 3) : $H_{out} = [90 – 2 \times 0 - 1 \times (3 – 1) – 1]/1 + 1 = 88$

$W_{out} = [170 – 2 \times 0 - 1 \times (3 – 1) – 1]/1 + 1 = 168$

Note: 88 x 168 = 64 x 11 x 21= 14784

Upconv Layers of Original Paper





Bed of nails upsampling

"Bed of Nails"

1	2
3	4

Input: 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

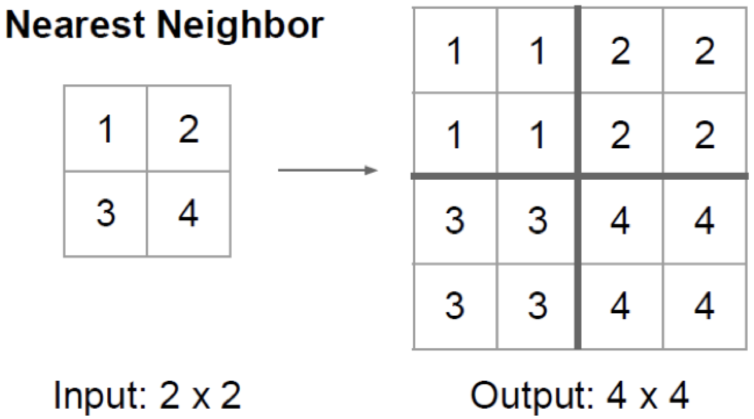
Upsampling

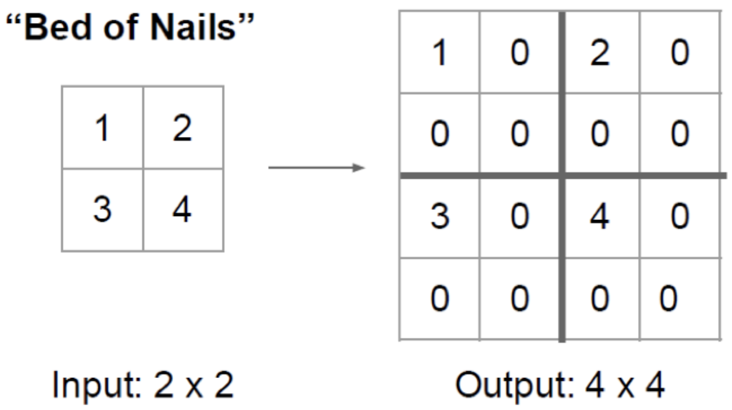
Nearest-Neighbor

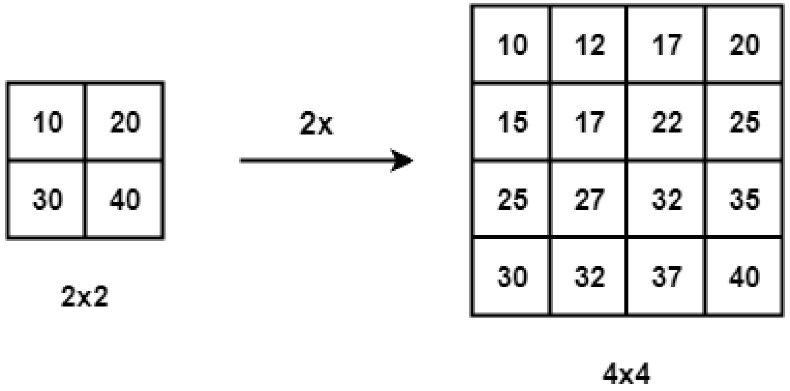
Bed of Nails

Interpolation

Bilinear,….





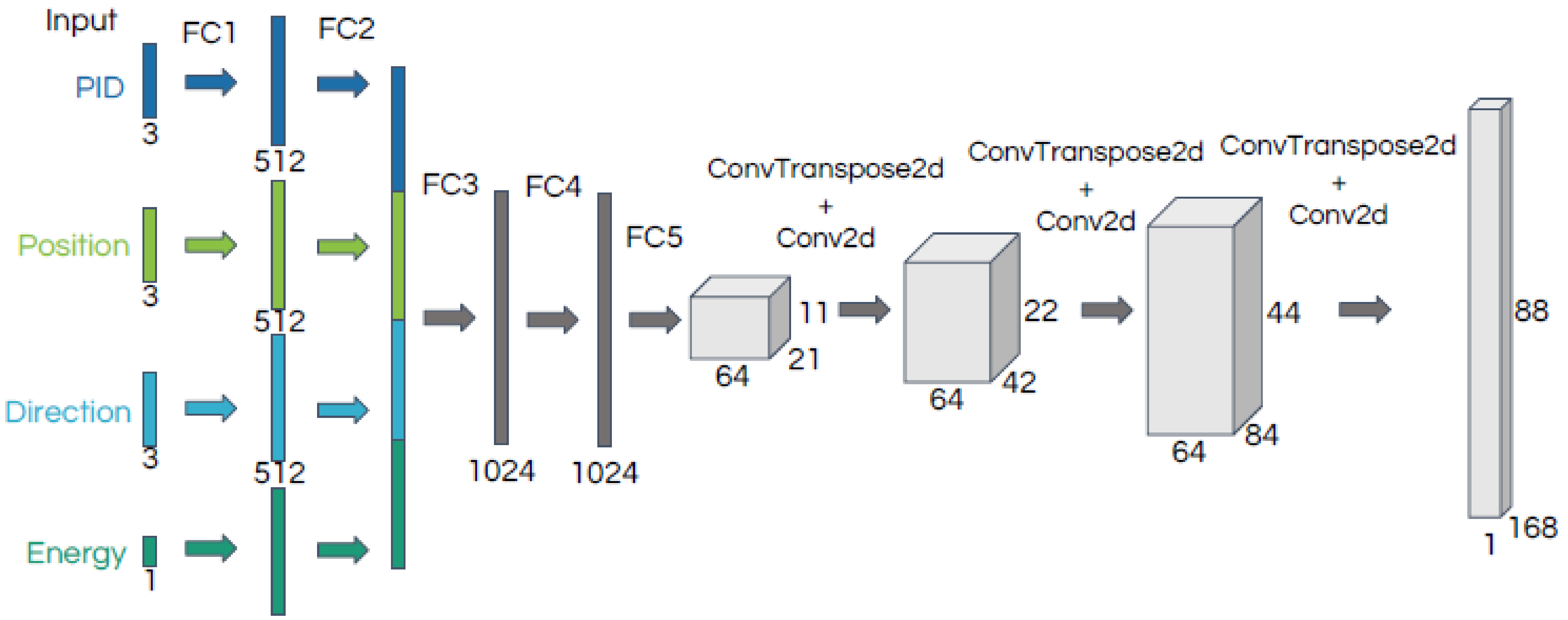


- The transposed convolution is the cause of the checkerboard artifacts in generated images.
- Some recommend an upsampling operation (i.e. interpolation method) followed by a convolution that preserves the image size to reduce such effects.
- The weights in the transposed convolution are learnable, while the upsampling operation is not learnable.

References

- https://arxiv.org/pdf/1603.07285.pdf
- Slides by Karan Yang on 9/13/2019 (I will post it at CERN) at this meeting
- https://distill.pub/2016/deconv-checkerboard/
- https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d
- https://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html
- https://medium.com/jun-devpblog/dl-12-unsampling-unpooling-and-transpose-convolution-831dc53687ce

Our Network



Input
FC1
FC2
PID
3
512
Position
3
512
Direction
3
512
Energy
1
FC3
FC4
FC5
1024
1024
ConvTranspose2d
+
Conv2d
11
21
64
ConvTranspose2d
+
Conv2d
22
42
64
ConvTranspose2d
+
Conv2d
44
84
64
88
168
1