How to calculate the output shape of convolution, deconvolution and pooling layers in CNNs for 2D input?

Formulas

Convolution $\rightarrow$ $O = \dfrac{F - K + 2P}{S} + 1$

Deconvolution $\rightarrow$ $O = S \times (F - 1) + K - 2P$

Pooling $\rightarrow$ $O = \dfrac{F - K}{S} + 1$

Here, O = Output shape, F = Input shape, K = Kernel size, S = Stride, P = Padding

Convolution

```
In [1]: import torch
        from torch import nn

In [2]: inp1 = torch.randn(32, 3, 64, 64) # BCHW
        inp2 = torch.randn(32, 3, 64, 128) # BCHW
        inp1.shape, inp2.shape

Out[2]: (torch.Size([32, 3, 64, 64]), torch.Size([32, 3, 64, 128]))

In [3]: CONVOLUTION = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=4, stride=2, padding=1)

In [4]: out1 = CONVOLUTION(inp1)
        out2 = CONVOLUTION(inp2)
        out1.shape, out2.shape

Out[4]: (torch.Size([32, 16, 32, 32]), torch.Size([32, 16, 32, 64]))
```

$$O = \frac{F - K + 2P}{S} + 1$$

Channels will be equal to the number of output channels indicated in the convolution operation.

For input 1, $O = \dfrac{64 - 4 + 2}{2} + 1 = 32$ $F = F_H = F_W$

For input 2, $O_H = \dfrac{64 - 4 + 2}{2} + 1 = 32$ $O_W = \dfrac{128 - 4 + 2}{2} + 1 = 64$ $F_H \neq F_W$

Deconvolution

$$O = S \times (F - 1) + K - 2P$$

```
In [1]: import torch
        from torch import nn

In [2]: inp1 = torch.randn(32, 3, 64, 64) # BCHW
        inp2 = torch.randn(32, 3, 64, 128) # BCHW
        inp1.shape, inp2.shape

Out[2]: (torch.Size([32, 3, 64, 64]), torch.Size([32, 3, 64, 128]))

In [3]: DECONVOLUTION = nn.ConvTranspose2d(in_channels=3, out_channels=16, kernel_size=4, stride=2, padding=1)

In [4]: out1 = DECONVOLUTION(inp1)
        out2 = DECONVOLUTION(inp2)
        out1.shape, out2.shape

Out[4]: (torch.Size([32, 16, 128, 128]), torch.Size([32, 16, 128, 256]))
```

Channels will be equal to the number of output channels indicated in the deconvolution operation.

For input 1, $O = 2 \times (64 - 1) + 4 - 2 = 128$ $F = F_H = F_W$

For input 2, $O_H = 2 \times (64 - 1) + 4 - 2 = 128$

$O_W = 2 \times (128 - 1) + 4 - 2 = 256$ $F_H \neq F_W$

Pooling

```
In [1]: import torch
        from torch import nn

In [2]: inp1 = torch.randn(32, 3, 64, 64) # BCHW
        inp2 = torch.randn(32, 3, 64, 128) # BCHW
        inp1.shape, inp2.shape

Out[2]: (torch.Size([32, 3, 64, 64]), torch.Size([32, 3, 64, 128]))

In [3]: POOLING = nn.MaxPool2d(kernel_size=2, stride=2)

In [4]: out1 = POOLING(inp1)
        out2 = POOLING(inp2)
        out1.shape, out2.shape

Out[4]: (torch.Size([32, 3, 32, 32]), torch.Size([32, 3, 32, 64]))
```

$$O = \frac{F - K}{S} + 1$$

Pooling does not deal with the channels. Output tensor will have the same number of channels as the input tensor.

For input 1, $O = \frac{64 - 2}{2} + 1 = 32$ $\quad F = F_H = F_W$

For input 2, $O_H = \frac{64 - 2}{2} + 1 = 32$ $\quad O_W = \frac{128 - 2}{2} + 1 = 64$ $\quad F_H \neq F_W$

THE END