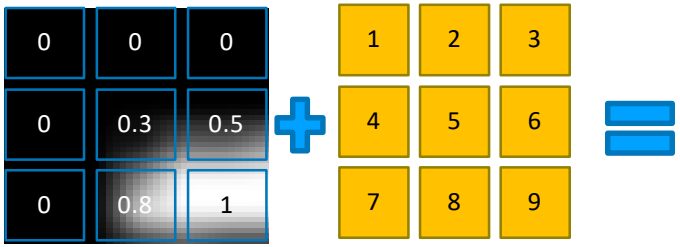
Pytorch nn.Conv2d

CLASS `torch.nn.Conv2d(`*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True*`)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$



ex) 입력 채널 1 /출력채널 1 / 커널 크기3x3

conv = nn.Conv2d(1,1,3)

- input type : torch.Tensor
- input shape : (N x C x H x W)

(batch_size, channel, height, width)

Output Volume Caculations

예제 1)
input image size : 227 x 227
filter size = 11x11
stride = 4
padding = 0
output image size = ? = 55

예제 2)
input image size : 64 x 64
filter size = 7x7
stride = 2
padding = 0
output image size = ? 29

예제 3)
input image size : 32 x 32
filter size = 5x5
stride = 1
padding = 2
output image size = ? 32

예제 4)
input image size : 32 x 64
filter size = 5x5
stride = 1
padding = 0
output image size = ? ⇒28x60

예제 5)
input image size : 64 x 32
filter size = 3x3
stride = 1
padding = 1
output image size = ? 64 x 32

$$Output\ size = \frac{input\ size - filter\ size + (2 * padding)}{Stride} + 1$$

Pytorch MaxPool2d

MaxPool2d

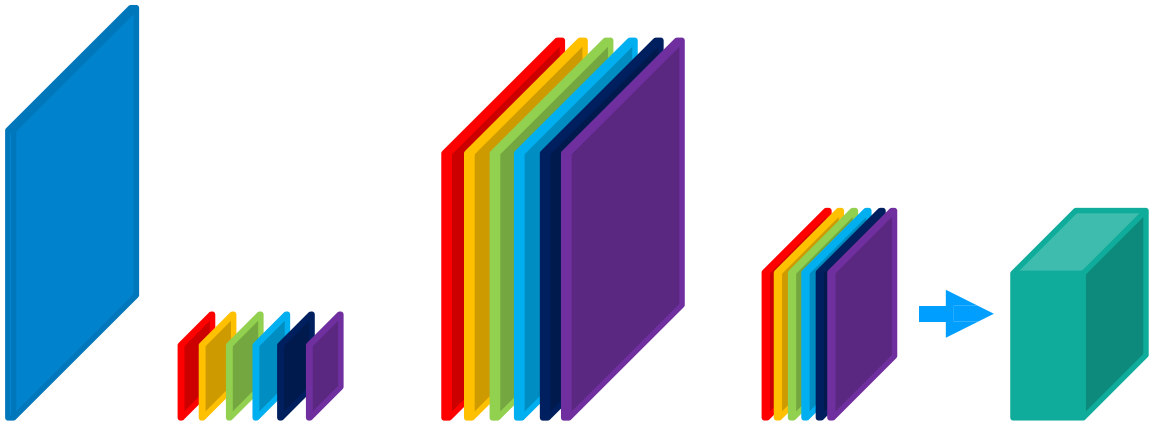
CLASS `torch.nn.MaxPool2d`(*kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False*) [SOURCE]

Applies a 2D max pooling over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C, H, W), output (N, C, H_{out}, W_{out}) and `kernel_size` (kH, kW) can be precisely described as:

$$out(N_i, C_j, h, w) = \max_{m=0,\ldots,kH-1} \max_{n=0,\ldots,kW-1} \text{input}(N_i, C_j, \text{stride}[0] \times h + m, \text{stride}[1] \times w + n)$$

CNN Implementation



1x 28x28
Input

5x 5x5
filter

5x 24x24
conv1

5x 12x12
pool1

5x 12x12
output

```
input = torch.Tensor(1,1,28,28)
conv1=nn.Conv2d(1,5,5)
pool  =  nn.MaxPool2d(2)
out = conv1(input)
out2 =pool(out)
out.size()
out2.size()
```