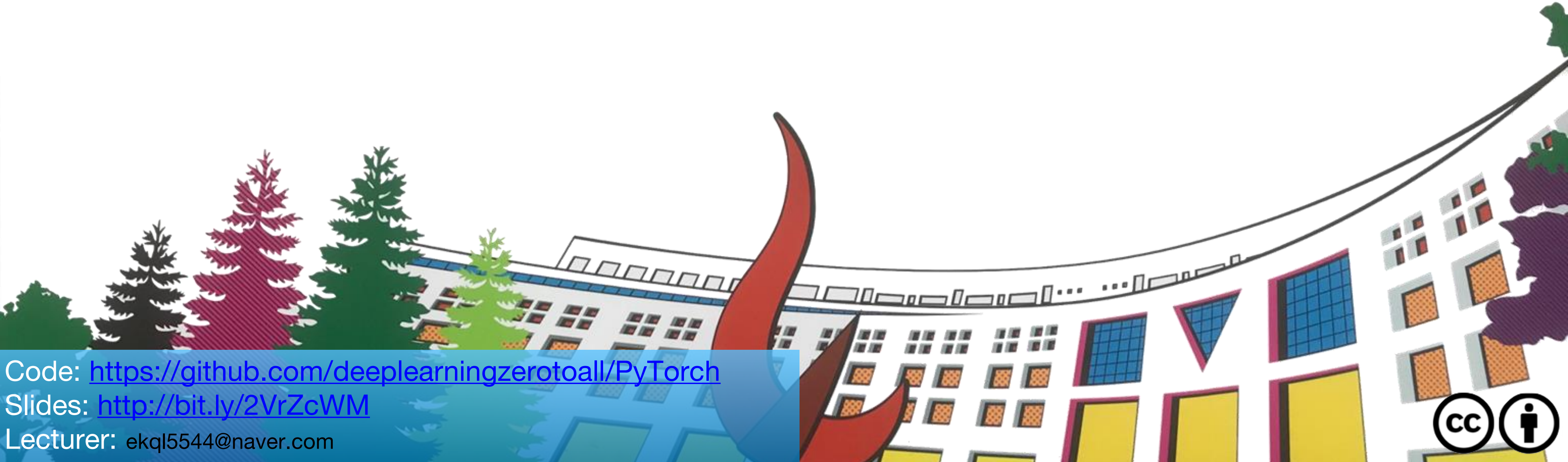
ML/DL for Everyone Season2

with PYTORCH

10-1 Convolution



Code: https://github.com/deeplearningzerotoall/PyTorch
Slides: http://bit.ly/2VrZcWM
Lecturer: ekql5544@naver.com

오늘의 목차

- Convolution?
- Neuron과 Convolution
- Pooling
- 다음시간에는?
- One more Thing!

Convolution?

- 이미지 위에서 stride 값 만큼 filter(kernel)을 이동시키면서 겹쳐지는 부분의 각 원소의 값을 곱해서 모두 더한 값을 출력으로 하는 연산

1	2	3	0	1
0	1	5	1	0
1	0	2	2	1
1	1	2	0	0
1	0	1	1	1

input

1	0	1
0	1	0
1	0	1

filter

8	9	8
8	5	9
6	5	5

output

Convolution?

이미지 위에서 stride 값 만큼 filter(kernel)을 이동시키면서 겹쳐지는 부분의 각 원소의 값을 곱해서 모두 더한 값을 출력으로 하는 연산

$$\begin{bmatrix}1&2&3\\0&1&5\\1&0&2\end{bmatrix}\quad\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}\quad\begin{bmatrix}8\end{bmatrix}$$

$$(1x1) + (2x0) + (3x1) +$$
$$(0x0) + (1x1) + (5x0) +$$
$$(1x1) + (0x0) + (2x1) = 8$$

1	2	3	0	1
0	1	5	1	0
1	0	2	2	1
1	1	2	0	0
1	0	1	1	1

input

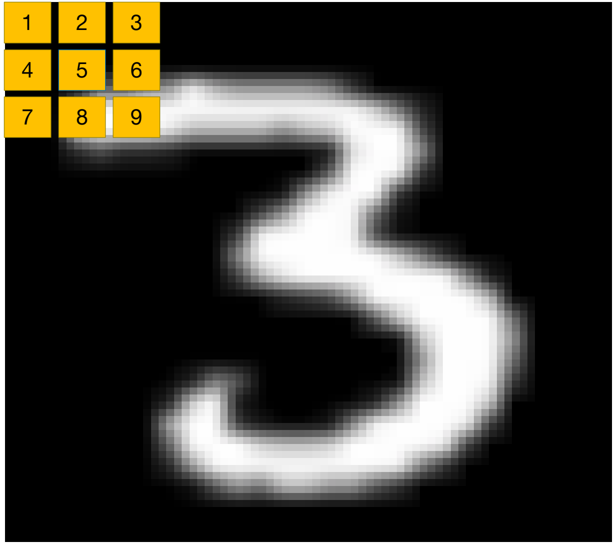
1	0	1
0	1	0
1	0	1

filter

8	9	8
8	5	9
6	5	5

output

계산해보기



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.3 & 0.5 \\ 0 & 0.8 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = 19.9$$

(0 * 1) + (0 * 2) + (0 * 3) +
(0 * 4) + (0.3 * 5) + (0.5 * 6) +
(0 * 7) + (0.8 * 8) + (1 * 9) = 19.9

Stride and Padding

- stride : filter를 한번에 얼마나 이동 할 것인가
- padding : zero-padding

0	0	0	0	0	0	0
0	1	2	3	0	1	0
0	0	1	5	1	0	0
0	1	0	2	2	1	0
0	1	1	2	0	0	0
0	1	0	1	1	1	0
0	0	0	0	0	0	0

input + padding

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:

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

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.3 & 0.5 \\ 0 & 0.8 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = ???$$

ex) 입력 채널 1 /출력채널 1 / 커널 크기3x3

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

입력의 형태

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

  (batch_size, channel, height, width)

직접 해볼까요?

Convolution의 output 크기

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

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

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

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

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

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

Neuron과 Convolution

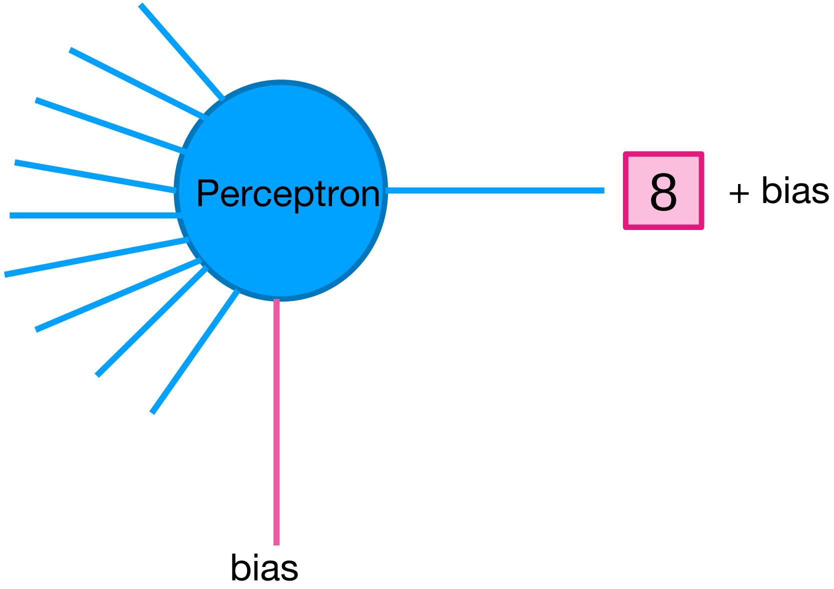
- Perceptron과 Convolution

Convolution filter(=kernel)

1	0	1
0	1	0
1	0	1

1	2	3
0	1	5
1	0	2

1	2	3	0	1
0	1	5	1	0
1	0	2	2	1
1	1	2	0	0
1	0	1	1	1

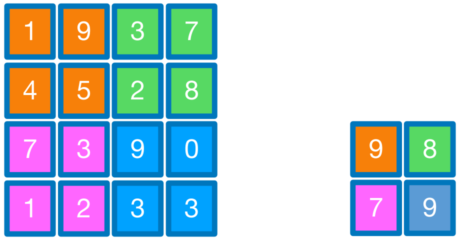


직접 해볼까요?

- nn.Conv2d에 입력
- filter size 변경 (size = 1x1, 3x3, 5x5)
- bias
- stride
- padding

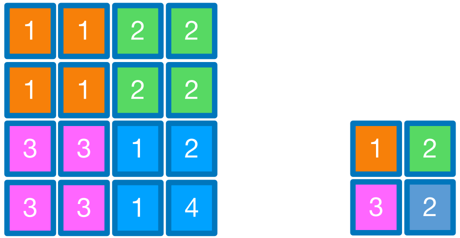
Pooling

- Max Pooling



max pooling

- Average Pooling



Average pooling

MaxPool2d

MaxPool2d

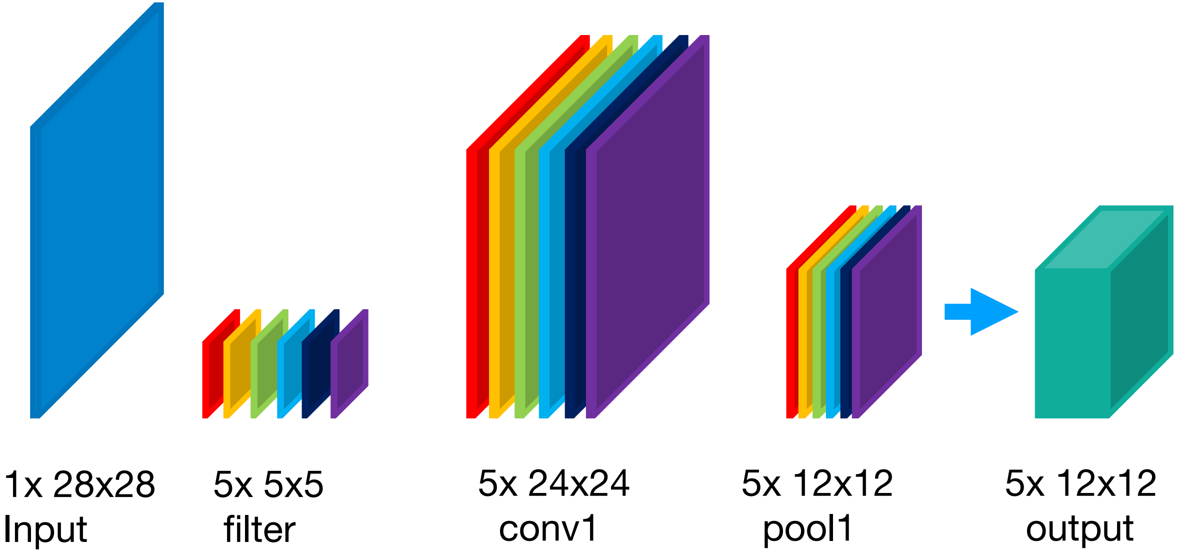
CLASS `torch.nn.MaxPool2d`(*kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False*) [SOURCE]

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




```
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()
```

직접 해볼까요?

What's Next?

- 오늘은 Convolution 연산과 Pooling 연산을 배우고 직접 활용해봤습니다.

- 다음시간에는 MNIST dataset에 CNN을 적용해보도록 하겠습니다.

One More Thing!

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.

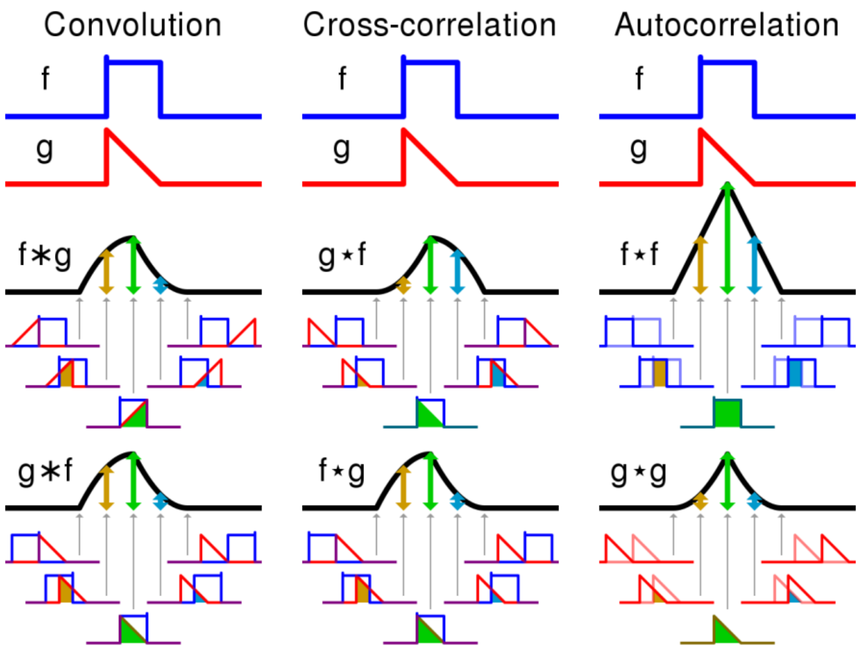
In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

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

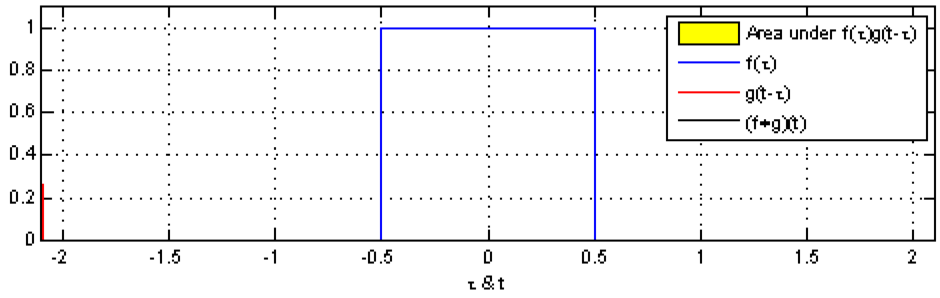
where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

- `stride` controls the stride for the cross-correlation, a single number or a tuple.
- `padding` controls the amount of implicit zero-paddings on both sides for `padding` number of points for each dimension.
- `dilation` controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but this link has a nice visualization of what `dilation` does.
- `groups` controls the connections between inputs and outputs. `in_channels` and `out_channels` must both be divisible by `groups`. For example,
    - At groups=1, all inputs are convolved to all outputs.
    - At groups=2, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels, and producing half the output channels, and both subsequently concatenated.
    - At groups= `in_channels`, each input channel is convolved with its own set of filters, of size: $\left\lfloor \frac{C_{\text{out}}}{C_{\text{in}}} \right\rfloor$.

What is Convolution?



$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau.$$



Cross-correlation

- Cross-correlation과 Convolution의 차이를 간단히 말하면...?

- 뒤집고 계산하면 => (Convolution)
- 안 뒤집고 계산하면 => (Cross-Correlation)

- 안 뒤집고 계산해서 Cross-Correlation!