Mathematics for Artificial Intelligence

9강: CNN 첫걸음

임성빈 UNIST 인공지능대학원 & 산업공학과
Learning Intelligent Machine Lab

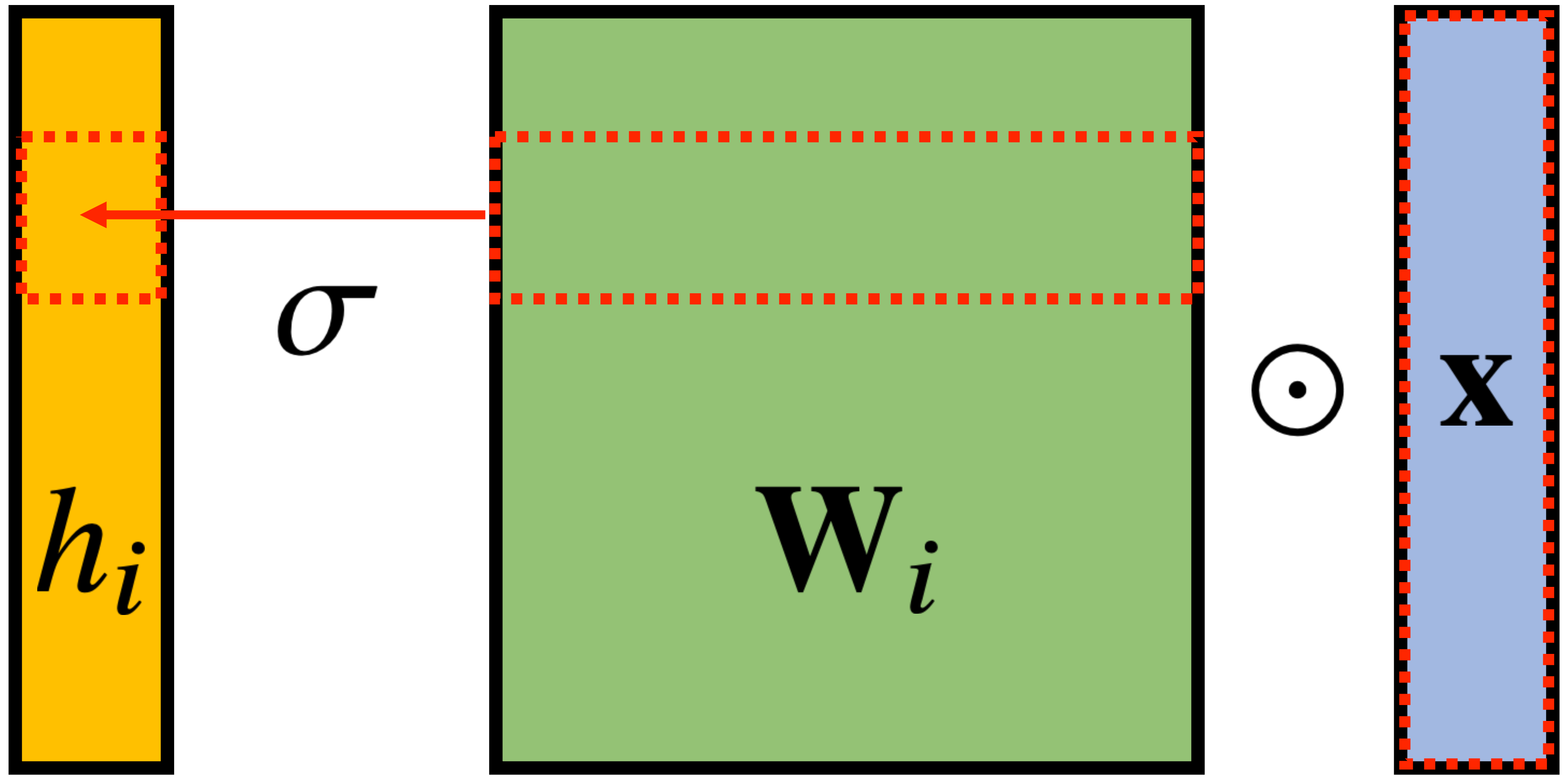
UNIST LIM Lab
Learning Intelligent Machine

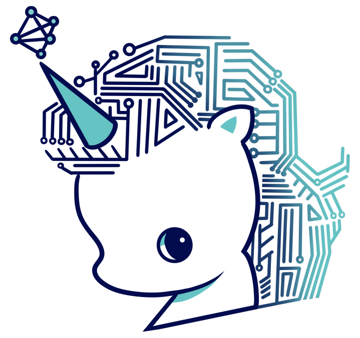
Convolution 연산 이해하기

- 지금까지 배운 다층신경망(MLP)은 각 뉴런들이 선형모델과 활성함수로 모두 연결된 (fully connected) 구조였습니다

$$h_i = \sigma\left(\sum_{j=1}^{p} W_{ij} x_j\right)$$

활성함수 가중치 행렬





각 성분 h_i 에 대응하는 가중치 행 $\mathbf{W}_i$ 이 필요하다

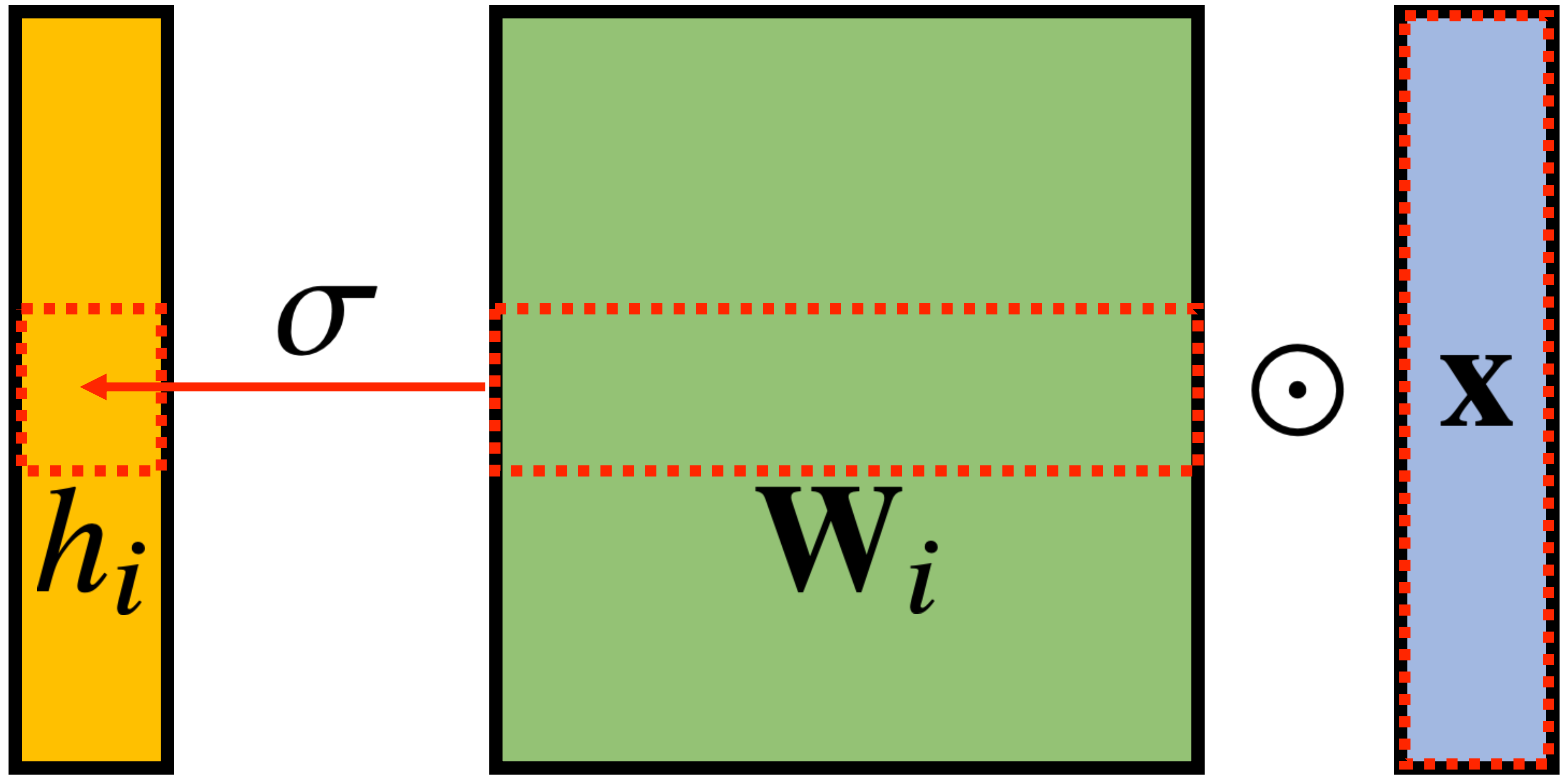
(출처: https://priorprobability.com)

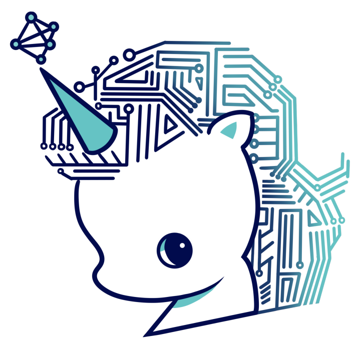
Convolution 연산 이해하기

- 지금까지 배운 다층신경망(MLP)은 각 뉴런들이 선형모델과 활성함수로 모두 연결된 (fully connected) 구조였습니다

$$h_i = \sigma\left(\sum_{j=1}^{p} W_{ij} x_j\right)$$

활성함수 가중치 행렬





만일 i 가 바뀌면 사용되는 가중치도 바뀝니다

(출처: https://priorprobability.com)

Convolution 연산 이해하기

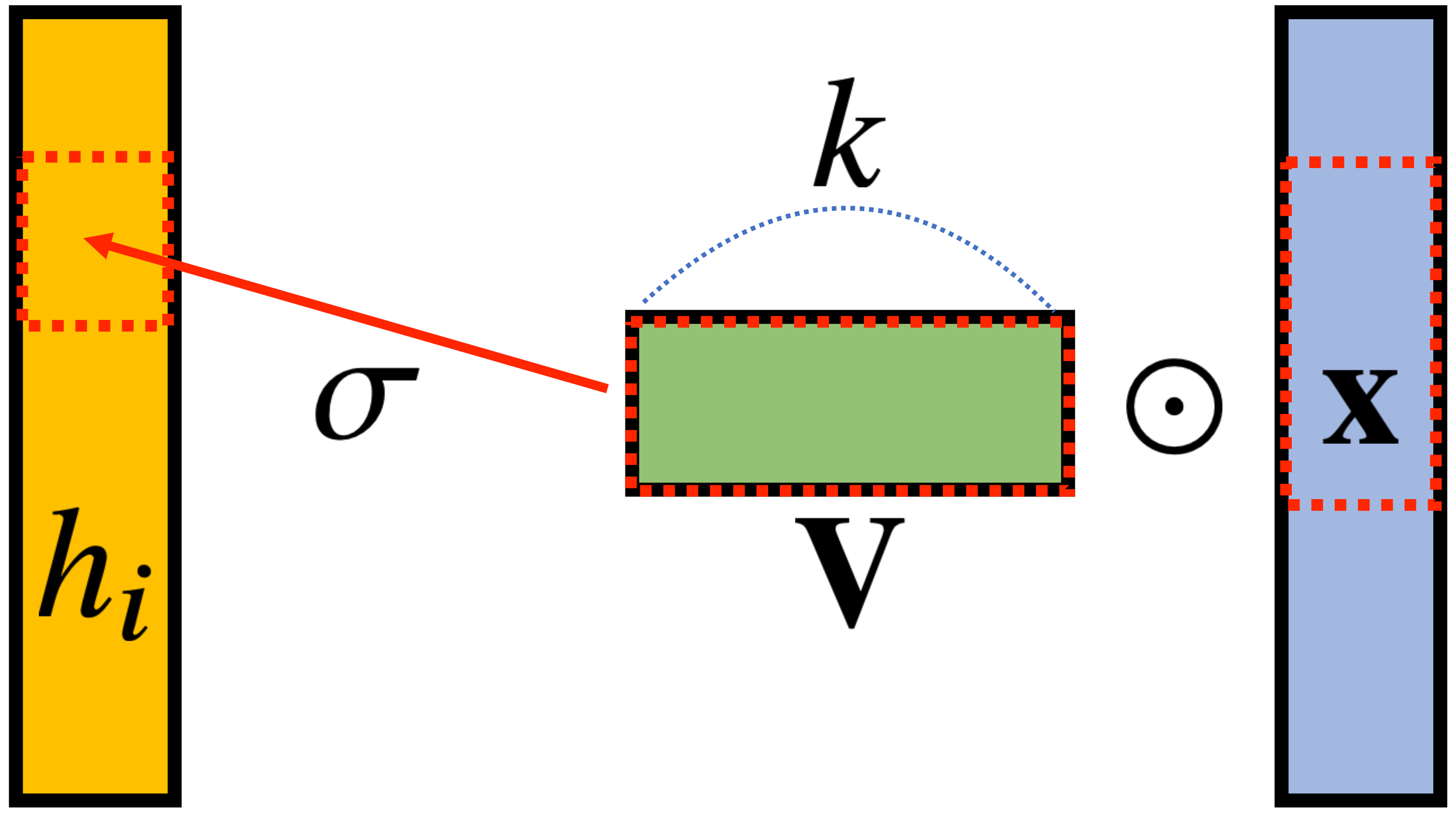
- Convolution 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

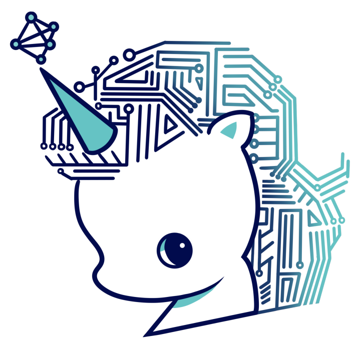
커널 사이즈

$$h_i = \sigma\left(\sum_{j=1}^{k} V_j x_{i+j-1}\right)$$

활성함수

가중치 행렬





모든 i 에 대해 적용되는 커널은 $\mathbf{V}$ 로 같고 커널의 사이즈만큼 $\mathbf{x}$ 상에서 이동하면서 적용합니다

(출처: https://priorprobability.com)

Convolution 연산 이해하기

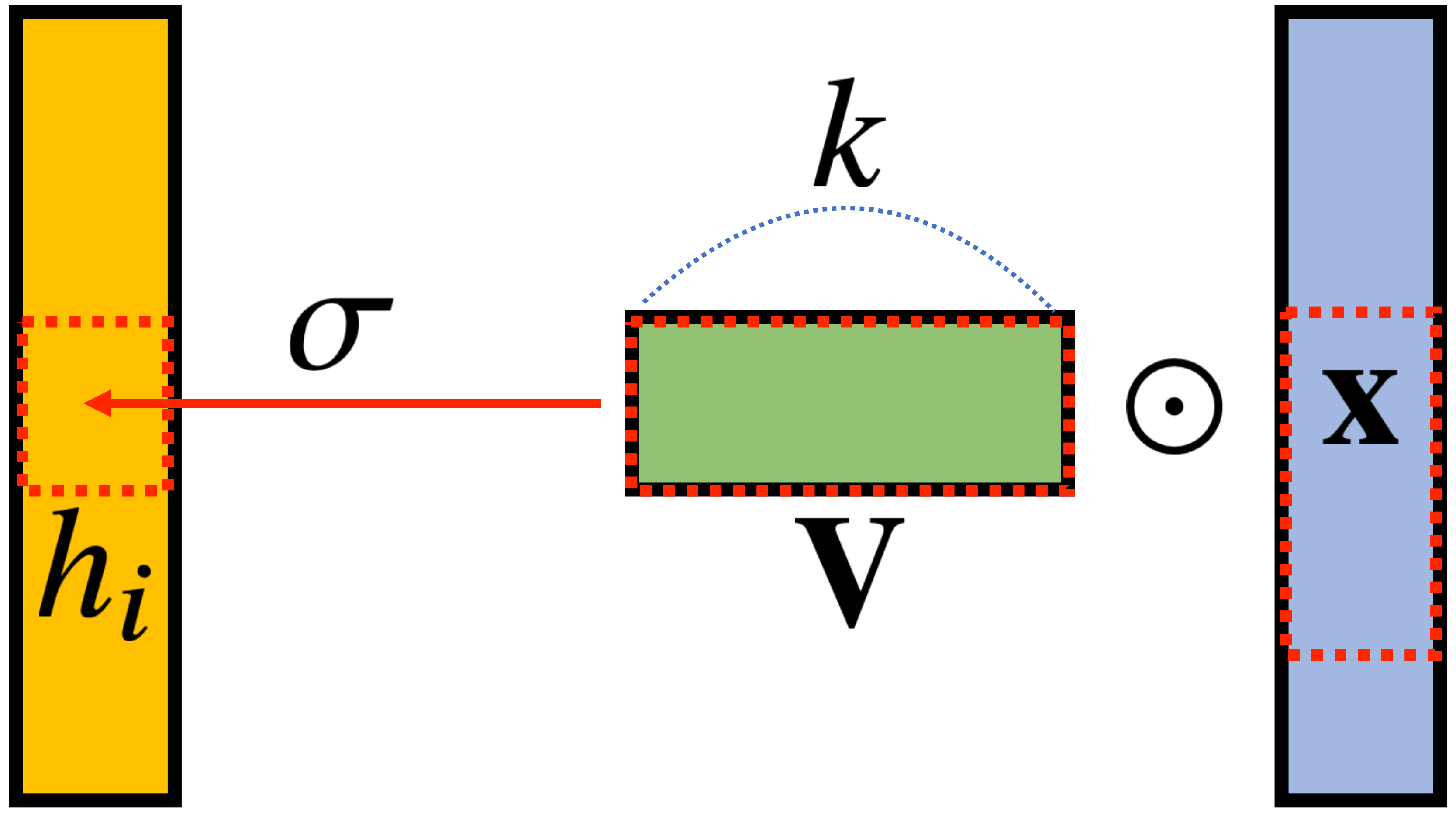
- Convolution 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

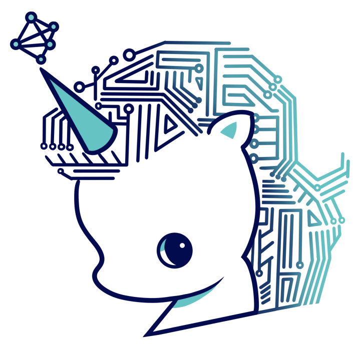
커널 사이즈

$$h_i = \sigma\left(\sum_{j=1}^{k} V_j x_{i+j-1}\right)$$

활성함수

가중치 행렬





활성화 함수를 제외한 Convolution 연산도 선형변환에 속합니다

(출처: https://priorprobability.com)

Convolution 연산 이해하기

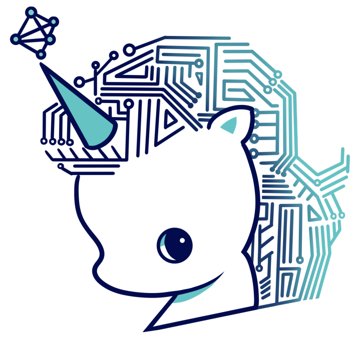
- Convolution 연산의 수학적인 의미는 신호(signal)를 커널을 이용해 국소적으로 증폭 또는 감소시켜서 정보를 추출 또는 필터링하는 것입니다

continuous

$$[f * g](x) = \int_{\mathbb{R}^d} f(z)g(x-z)\mathrm{d}z = \int_{\mathbb{R}^d} f(x-z)g(z)\mathrm{d}z = [g * f](x)$$

discrete

$$[f * g](i) = \sum_{a \in \mathbb{Z}^d} f(a)g(i-a) = \sum_{a \in \mathbb{Z}^d} f(i-a)g(a) = [g * f](i)$$



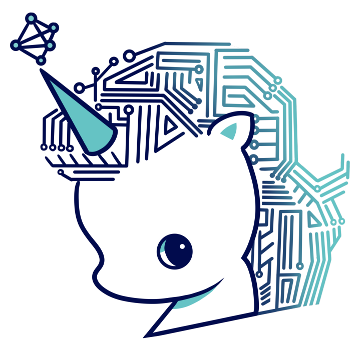
Convolution 을 수식으로만 이해하는 것은 매우 어렵습니다

Convolution 연산 이해하기

- Convolution 연산의 수학적인 의미는 신호(signal)를 커널을 이용해 국소적으로 증폭 또는 감소시켜서 정보를 추출 또는 필터링하는 것입니다

continuous $$[f * g](x) = \int_{\mathbb{R}^d} f(z)g(x+z)dz = \int_{\mathbb{R}^d} f(x+z)g(z)dz = [g * f](x)$$

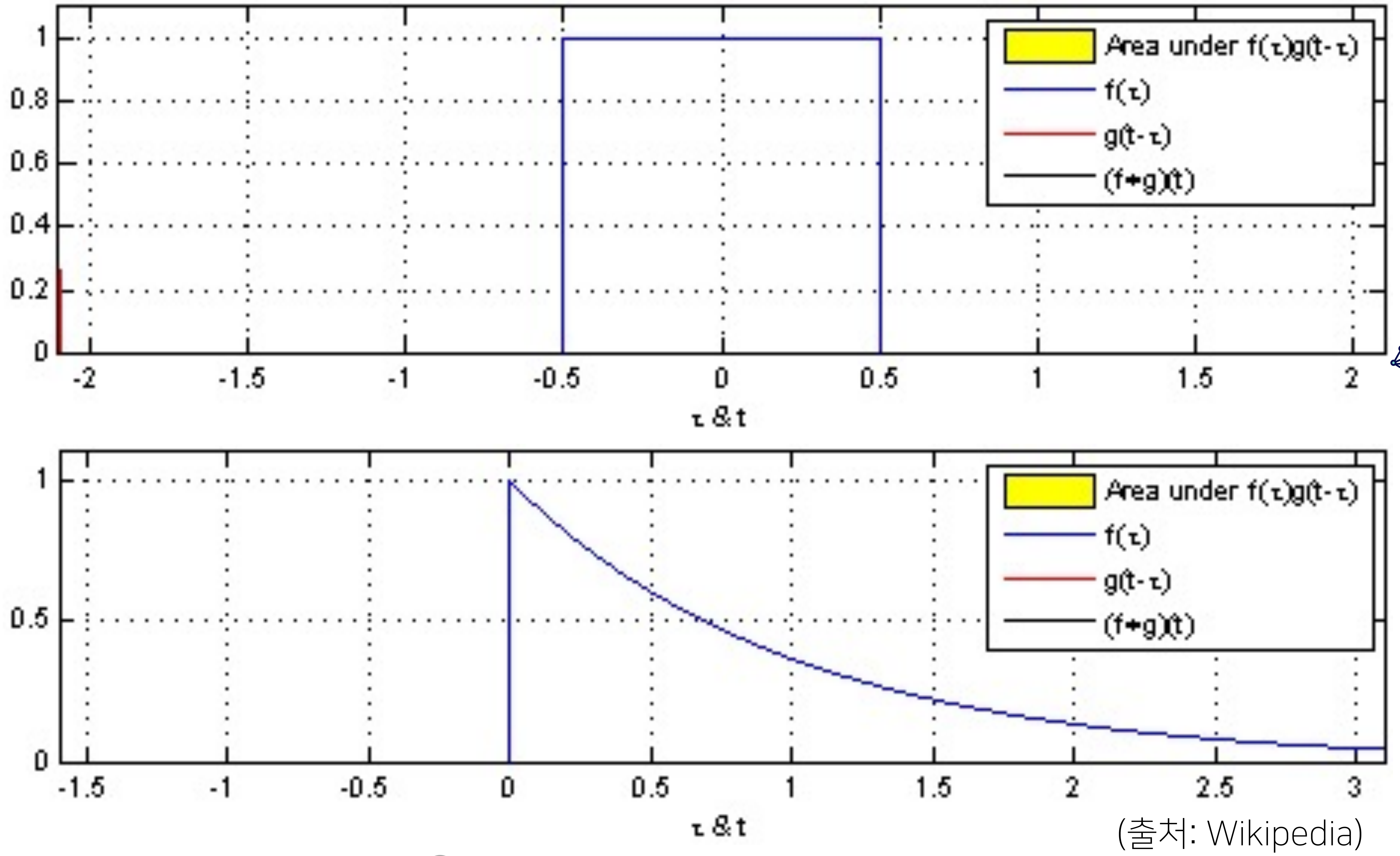
discrete $$[f * g](i) = \sum_{a \in \mathbb{Z}^d} f(a)g(i+a) = \sum_{a \in \mathbb{Z}^d} f(i+a)g(a) = [g * f](i)$$



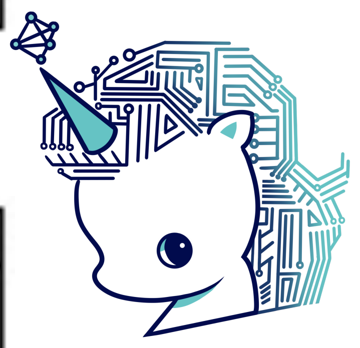
CNN 에서 사용하는 연산은 사실 convolution 이 아니고 cross-correlation 이라 부릅니다

Convolution 연산 이해하기

- 커널은 정의역 내에서 움직여도 변하지 않고(translation invariant) 주어진 신호에 국소적(local)으로 적용합니다



(출처: Wikipedia)



파란색이 신호
빨간색이 커널
검은색이 결과입니다

영상처리에서 Convolution

0	0	0
0	1	0
0	0	0

custom

The **custom** kernel is whatever you make it.

For more, have a look at Gimp's excellent documentation on using Image kernel's. You can also apply your own custom filters in Photoshop by going to Filter -> Other -> Custom...

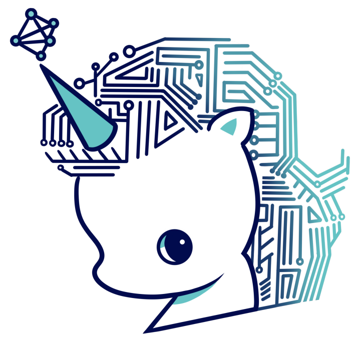
(출처: http://setosa.io/ev/image-kernels/)

다양한 차원에서의 Convolution

- Convolution 연산은 1차원뿐만 아니라 다양한 차원에서 계산 가능합니다

1D-conv

$$[f * g](i) = \sum_{p=1}^{d} f(p)g(i+p)$$



데이터의 성격에 따라
사용하는 커널이 달라집니다

2D-conv

$$[f * g](i, j) = \sum_{p,q} f(p, q)g(i+p, j+q)$$

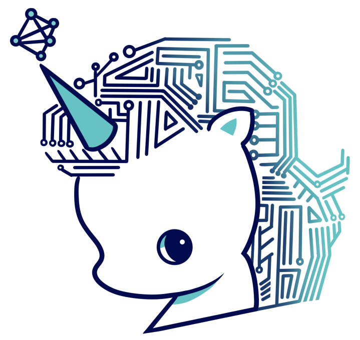
3D-conv

$$[f * g](i, j, k) = \sum_{p,q,r} f(p, q, r)g(i+p, j+q, k+r)$$

다양한 차원에서의 Convolution

- Convolution 연산은 1차원뿐만 아니라 다양한 차원에서 계산 가능합니다

1D-conv
$$[f * g](i) = \sum_{p=1}^{d} \boxed{f(p)} g(i+p)$$



i, j, k 가 바뀌어도 커널 f 의 값은 바뀌지 않습니다

2D-conv
$$[f * g](i, j) = \sum_{p,q} \boxed{f(p, q)} g(i+p, j+q)$$

3D-conv
$$[f * g](i, j, k) = \sum_{p,q,r} \boxed{f(p, q, r)} g(i+p, j+q, k+r)$$

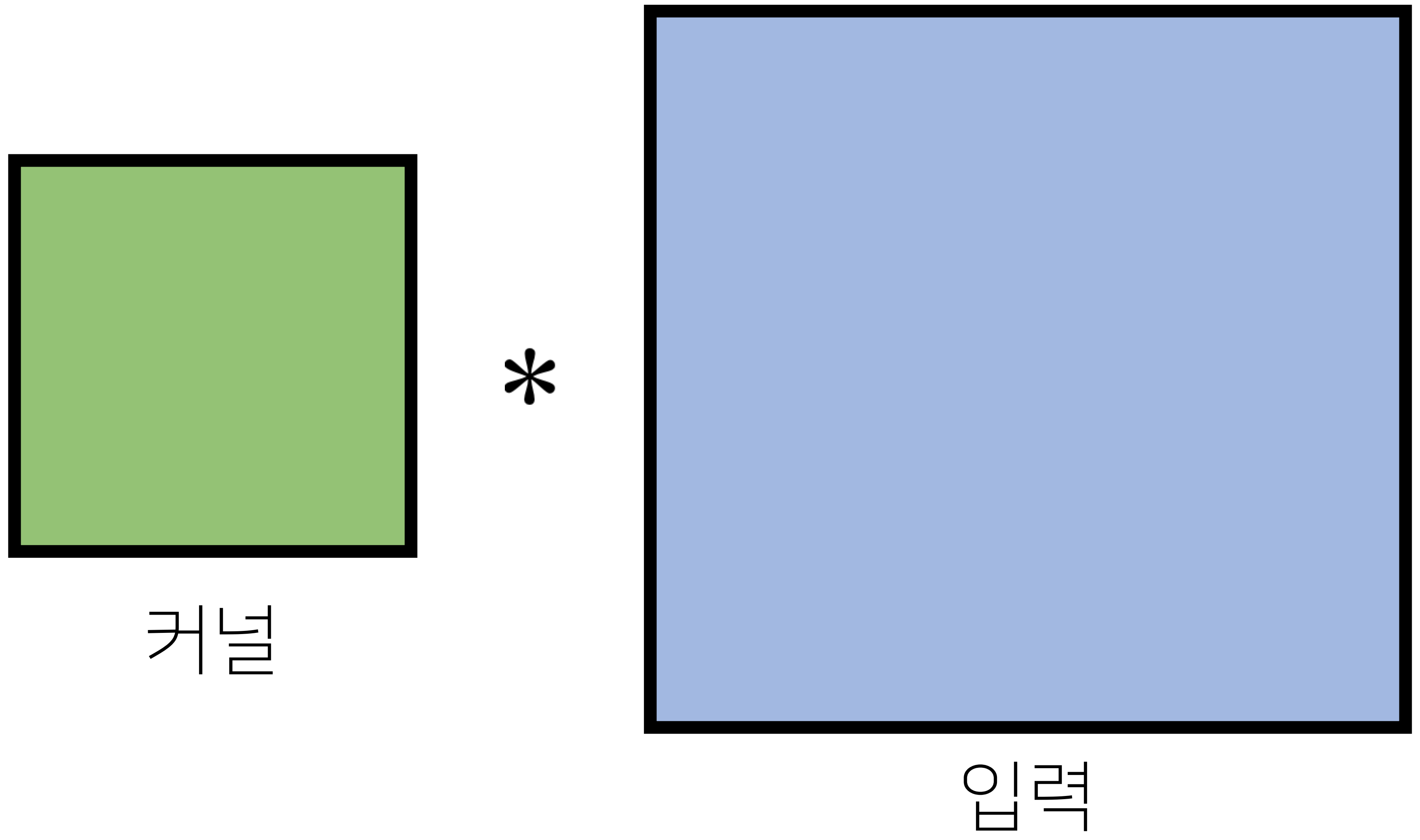
2차원 Convolution 연산 이해하기

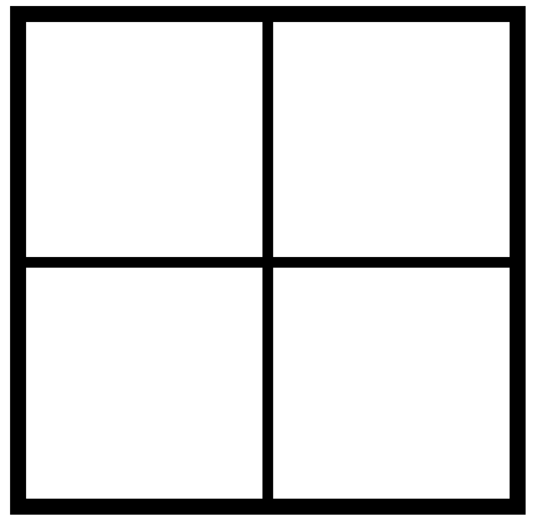
- 2D-Conv 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

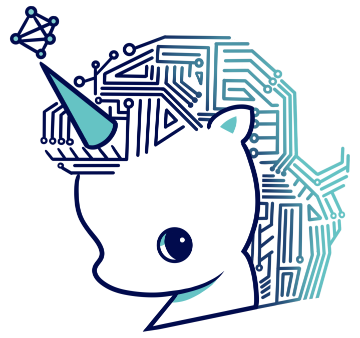
커널

$$[f * g](i, j) = \sum_{p,q} f(p, q) g(i + p, j + q)$$

입력







$$0 \times 0 + 1 \times 1 + 2 \times 3 + 3 \times 4 = 19$$

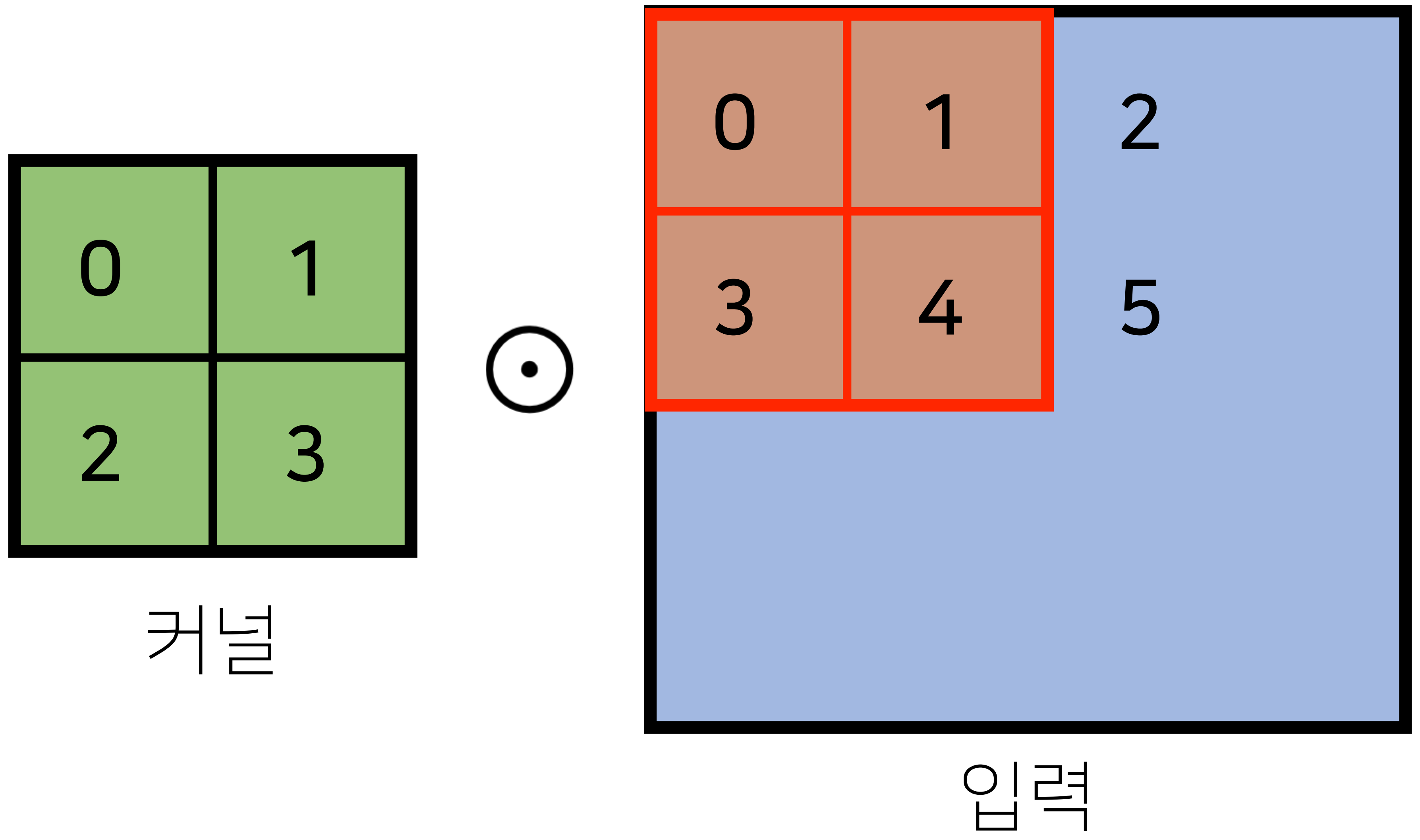
2차원 Convolution 연산 이해하기

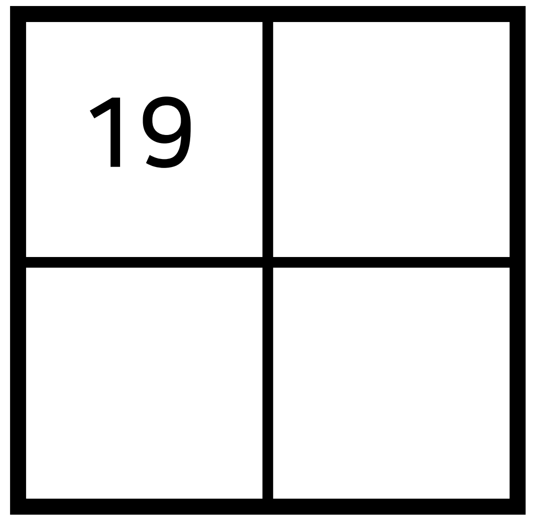
- 2D-Conv 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

커널

$$[f * g](i, j) = \sum_{p,q} f(p, q) g(i + p, j + q)$$

입력





$$0 \times 0 + 1 \times 1 + 2 \times 3 + 3 \times 4 = 19$$

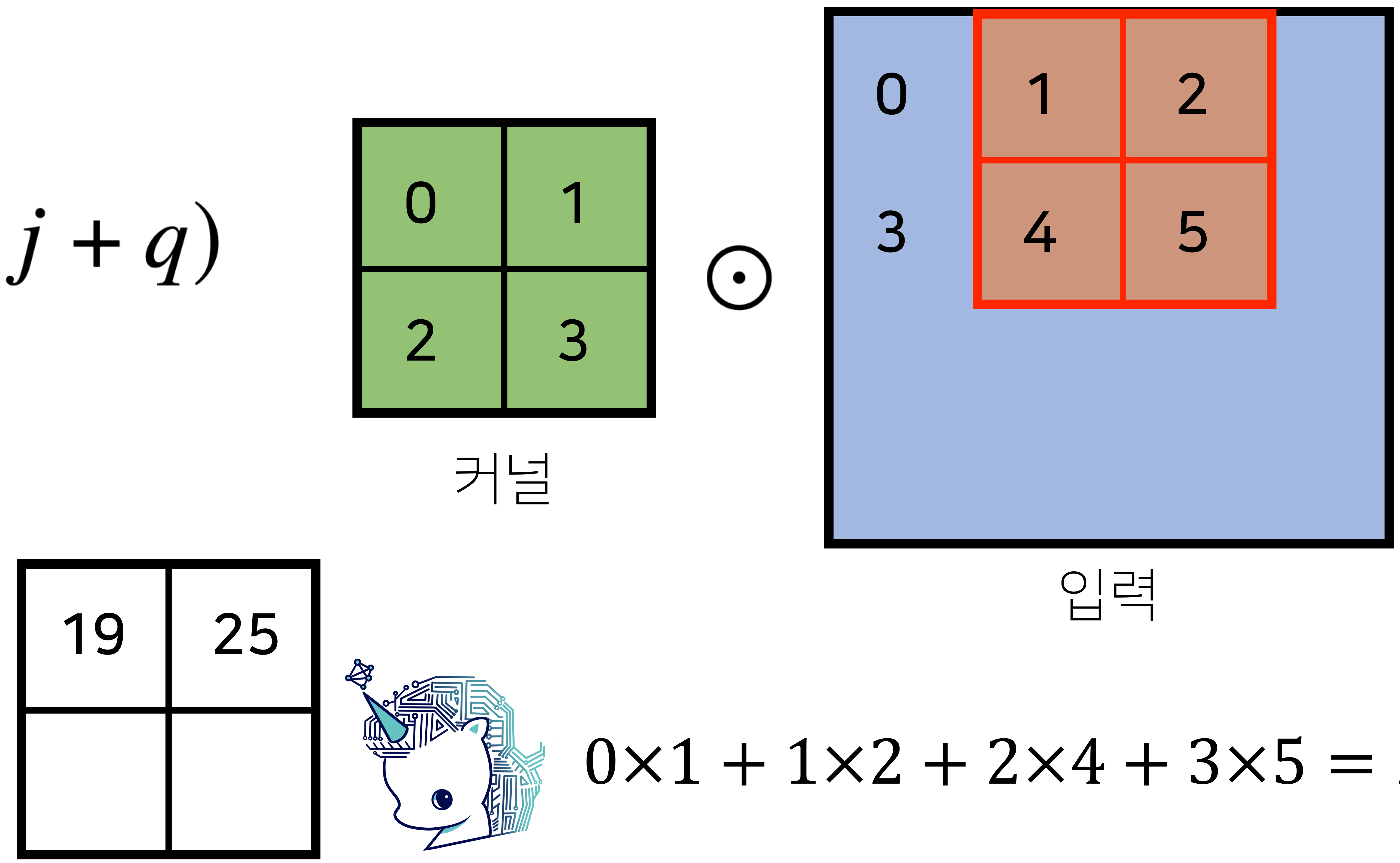
2차원 Convolution 연산 이해하기

- 2D-Conv 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

커널

$$[f * g](i, j) = \sum_{p,q} f(p, q)g(i + p, j + q)$$

입력

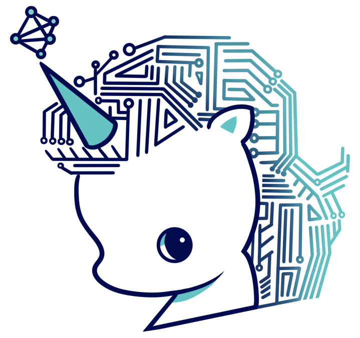


$$0\times1 + 1\times2 + 2\times4 + 3\times5 = 25$$

2차원 Convolution 연산 이해하기

- 2D-Conv 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다
- 입력 크기를 (H, W), 커널 크기를 (K_H, K_W), 출력 크기를 (O_H, O_W) 라 하면 출력 크기는 다음과 같이 계산합니다

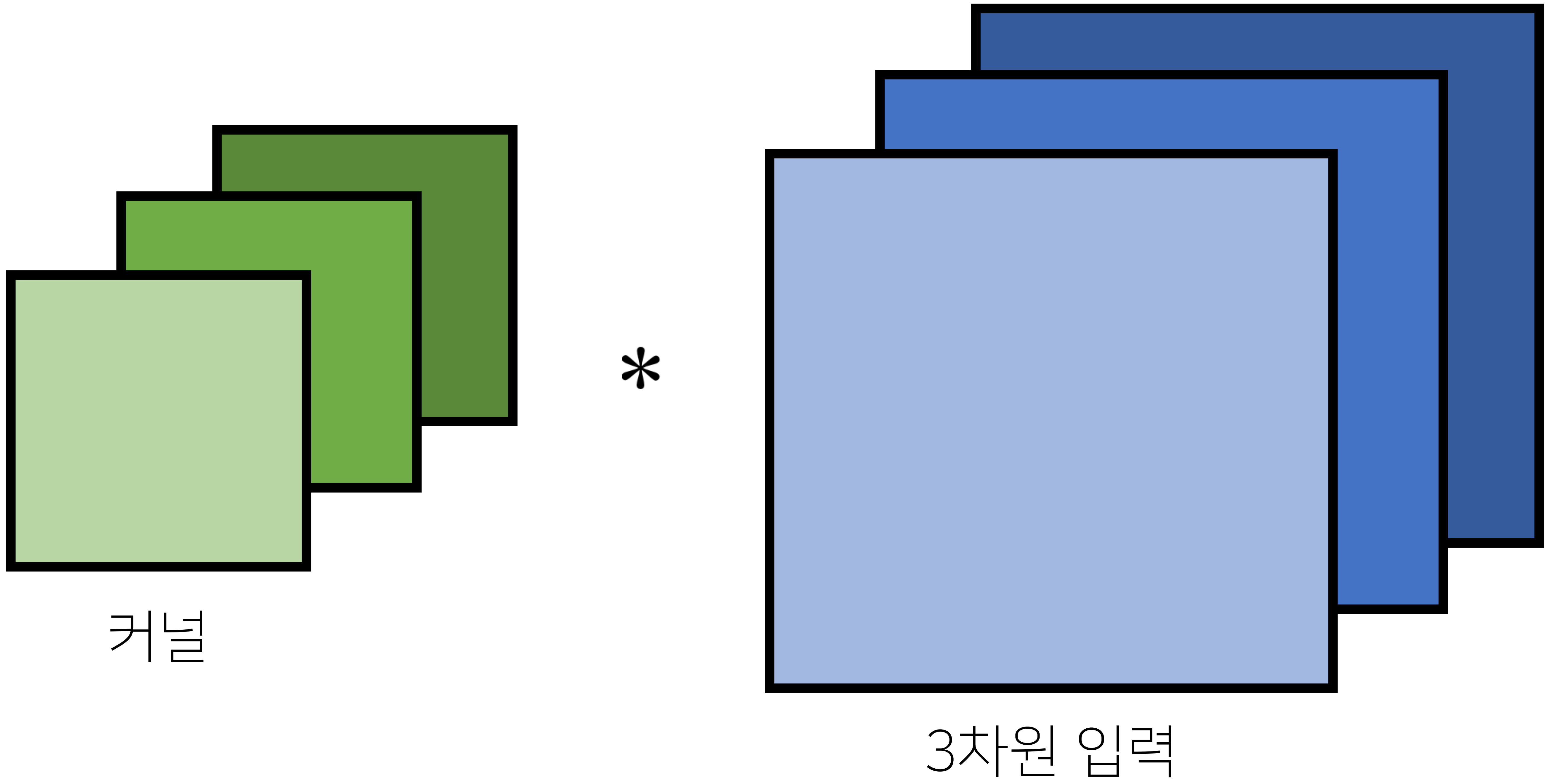
$$O_H = H - K_H + 1$$
$$O_W = W - K_W + 1$$

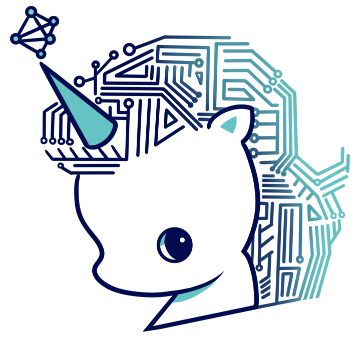


가령 28x28 입력을 3x3 커널로
2D-Conv 연산을 하면 26x26 이 된다

2차원 Convolution 연산 이해하기

- 채널이 여러개인 2차원 입력의 경우 2차원 Convolution 을 채널 개수만큼 적용한다고 생각하면 됩니다





3차원부터는 행렬이 아닌 텐서라 부릅니다

2차원 Convolution 연산 이해하기

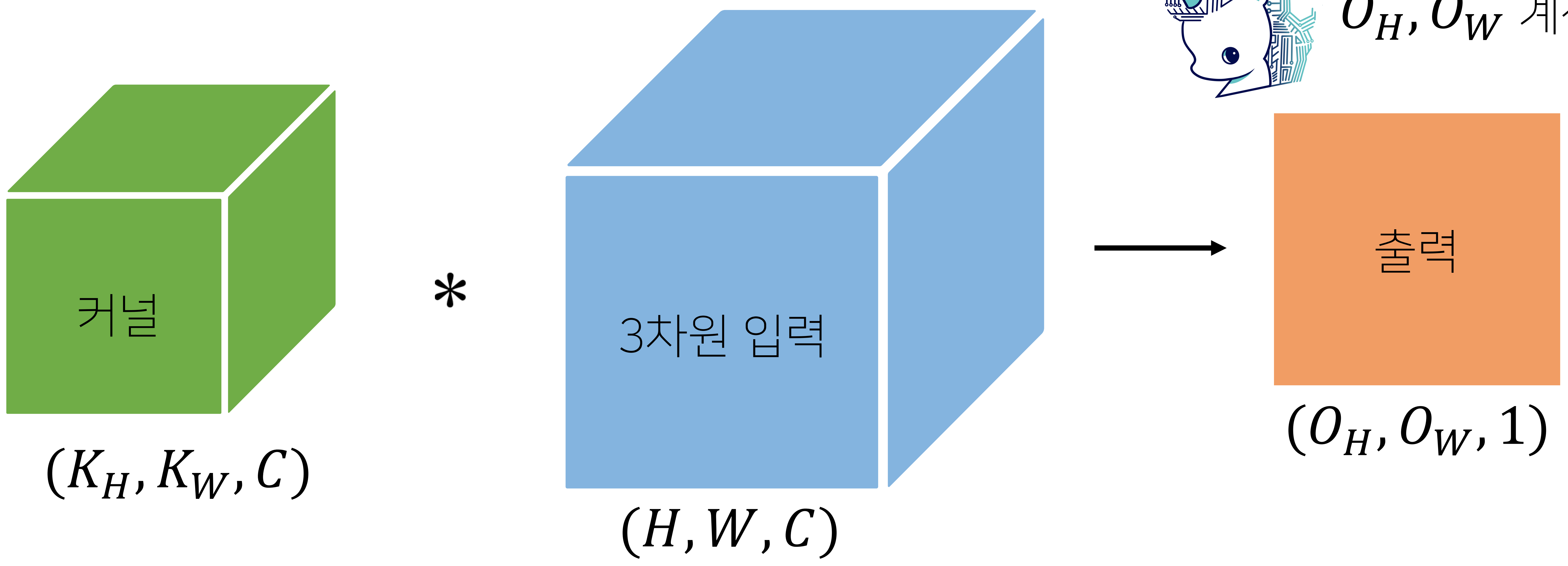
- 채널이 여러개인 2차원 입력의 경우 2차원 Convolution 을 채널 개수만큼 적용한다고 생각하면 됩니다

채널이 여러개인 경우 커널의 채널 수와 입력의 채널수가 같아야 합니다

2차원 Convolution 연산 이해하기

- 채널이 여러개인 2차원 입력의 경우 2차원 Convolution 을 채널 개수만큼 적용한다고 생각하면 됩니다. 텐서를 직육면체 블록으로 이해하면 좀 더 이해하기 쉽습니다

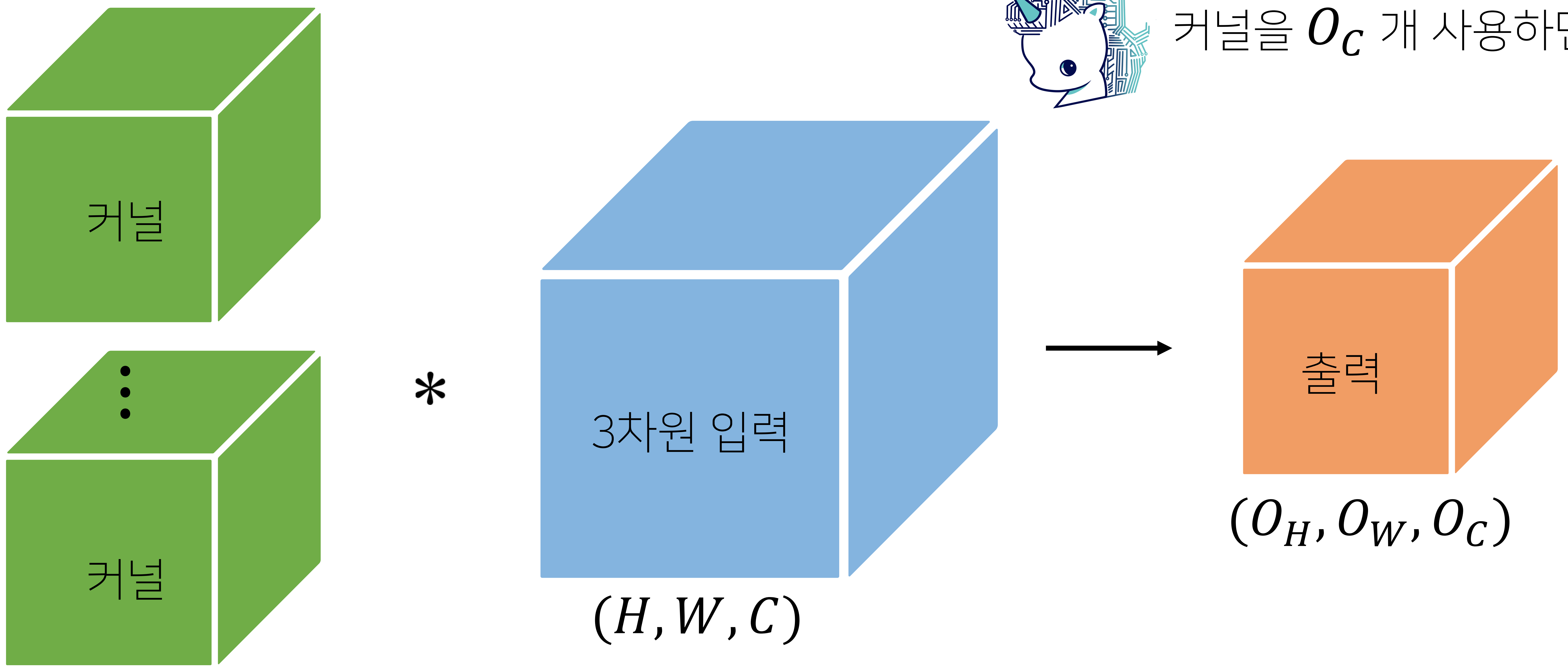
O_H, O_W 계산법은 앞 슬라이드 참조



2차원 Convolution 연산 이해하기

- 채널이 여러개인 2차원 입력의 경우 2차원 Convolution 을 채널 개수만큼 적용한다고 생각하면 됩니다. 텐서를 직육면체 블록으로 이해하면 좀 더 이해하기 쉽습니다

커널을 O_C 개 사용하면 출력도 텐서가 된다

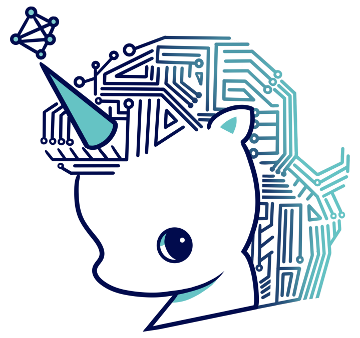


(K_H, K_W, C) ⋮ (K_H, K_W, C) $*$ (H, W, C) $\longrightarrow$ (O_H, O_W, O_C)

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

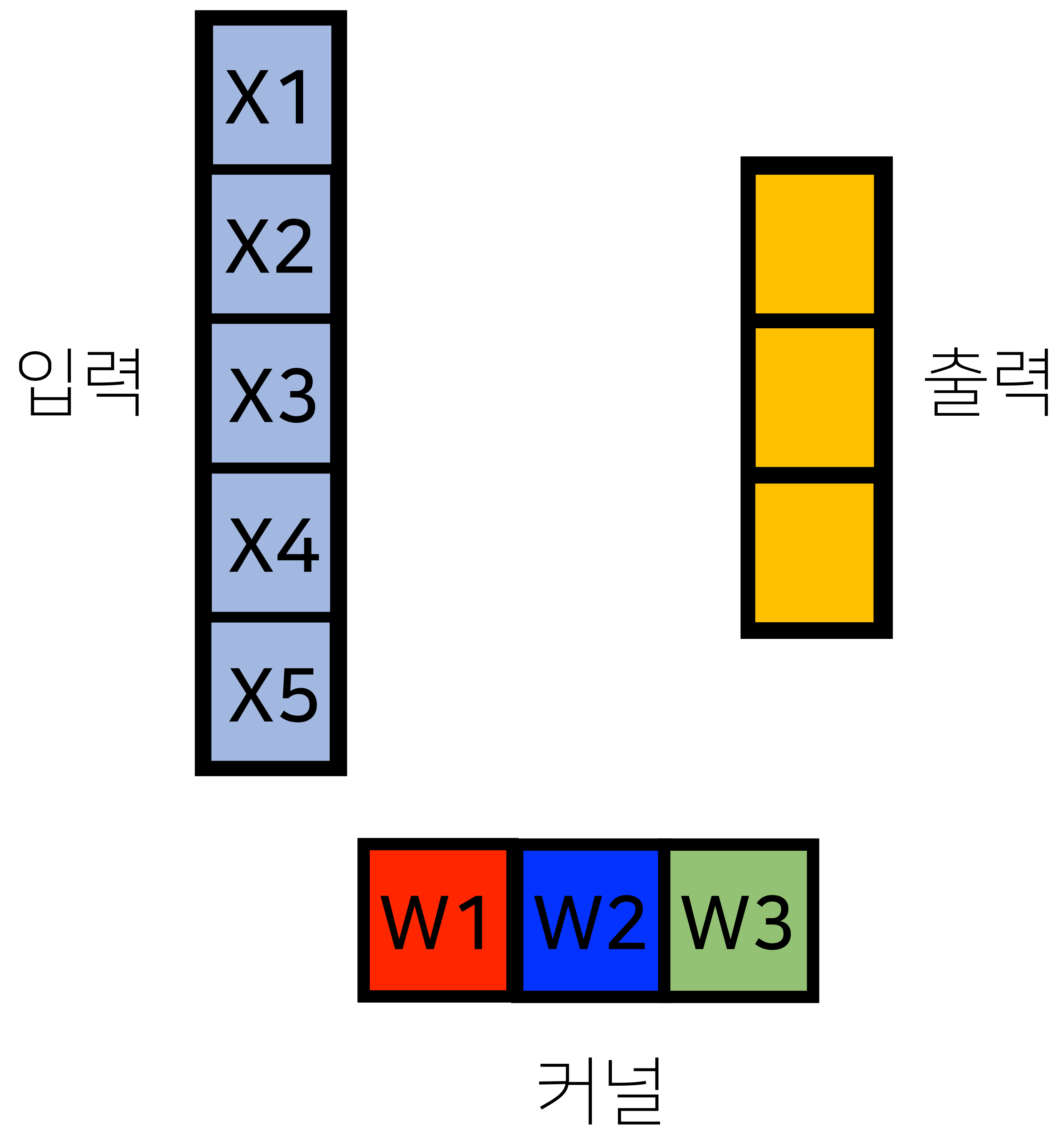
$$
\begin{aligned}
\frac{\partial}{\partial x}[f * g](x) &= \frac{\partial}{\partial x}\int_{\mathbb{R}^d} f(y)g(x-y)\mathrm{d}y \\
&= \int_{\mathbb{R}^d} f(y)\frac{\partial g}{\partial x}(x-y)\mathrm{d}y \\
&= [f * g'](x)
\end{aligned}
$$

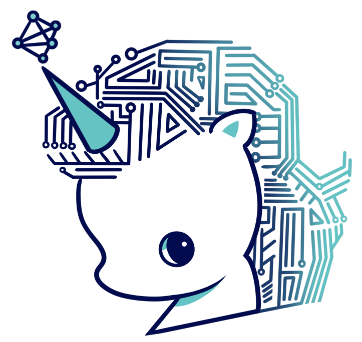


Discrete 일 때도 마찬가지로 성립한다

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다



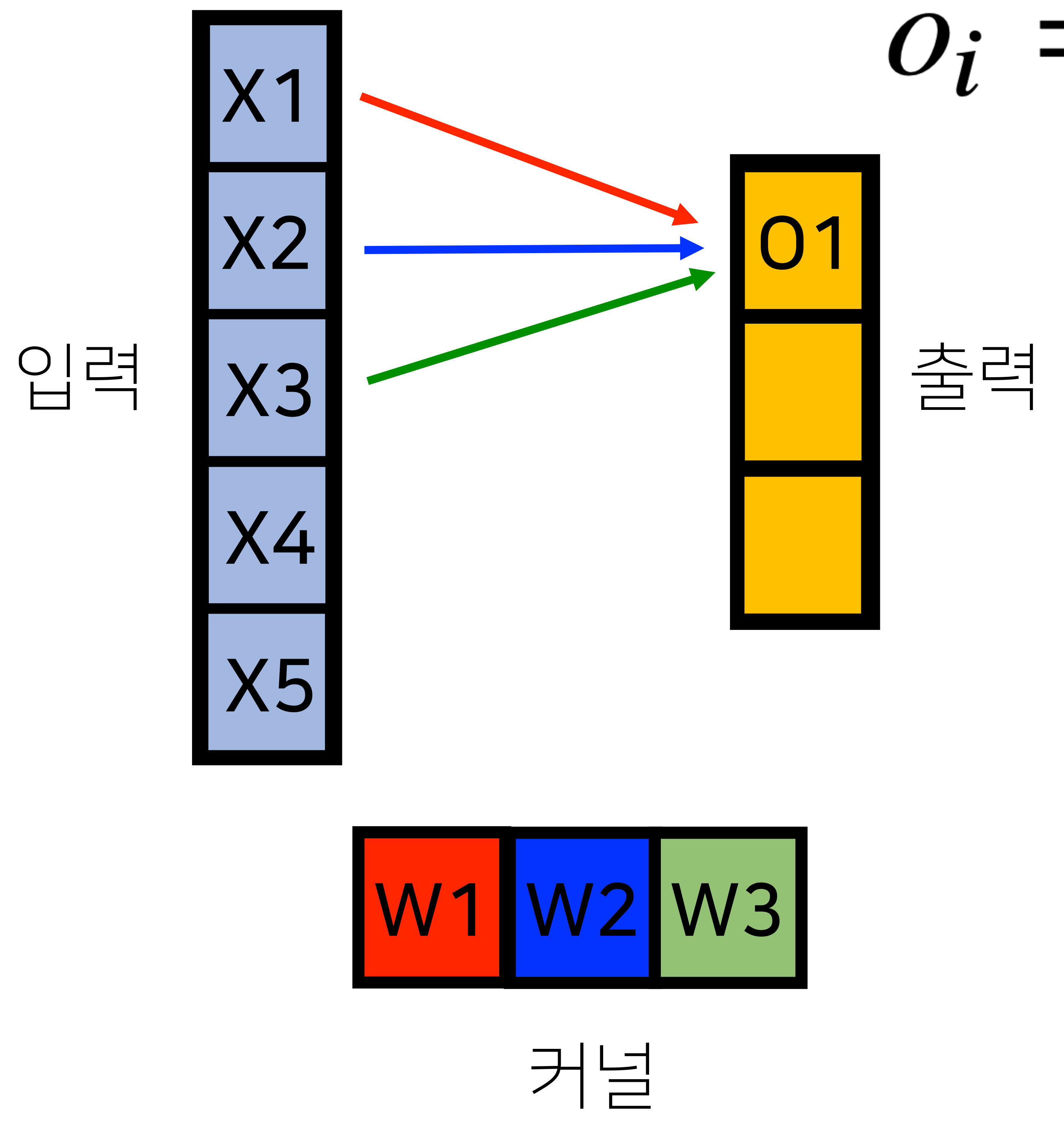


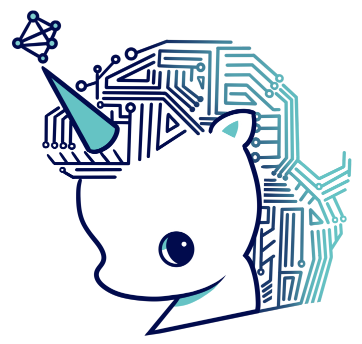
그림으로 이해하면 훨씬 쉽습니다

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

$$o_i = \sum_j w_j x_{i+j-1}$$



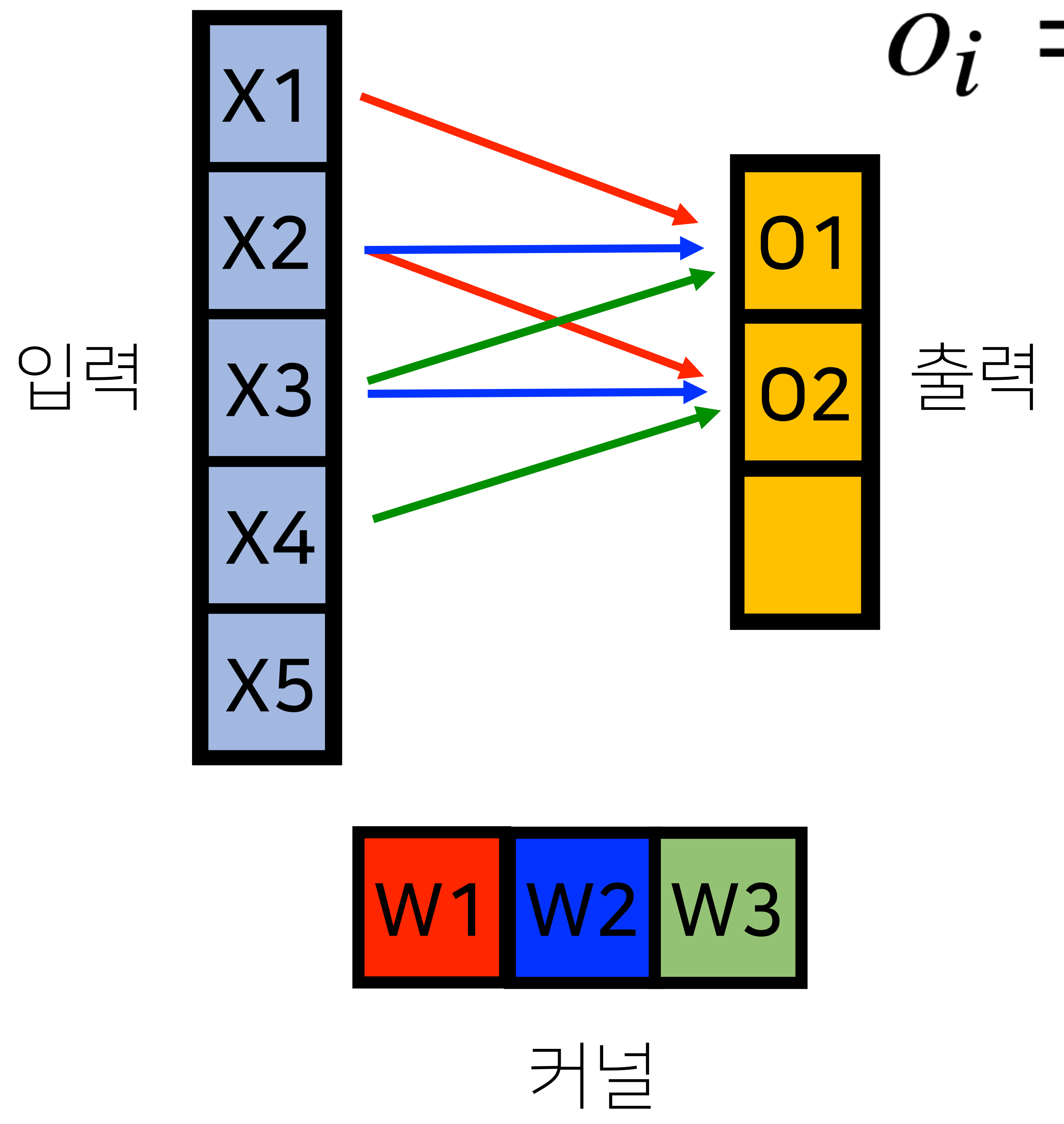


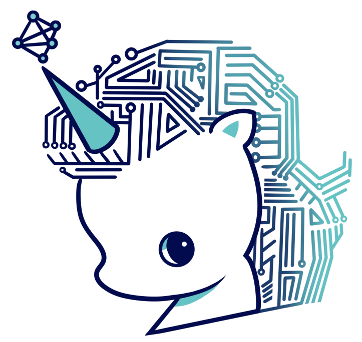
그림으로 이해하면 훨씬 쉽습니다

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

$$o_i = \sum_j w_j x_{i+j-1}$$



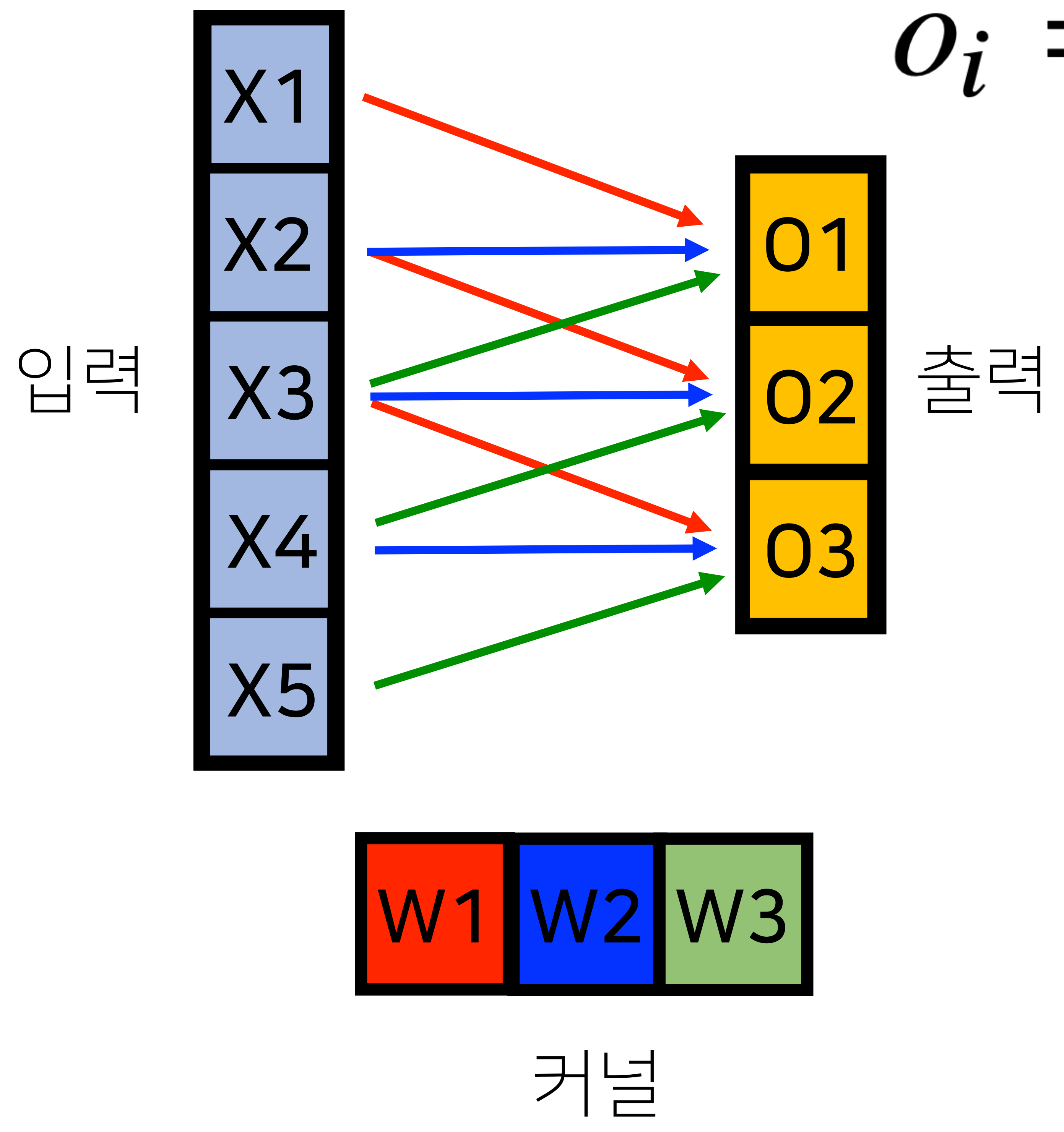


그림으로 이해하면 훨씬 쉽습니다

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

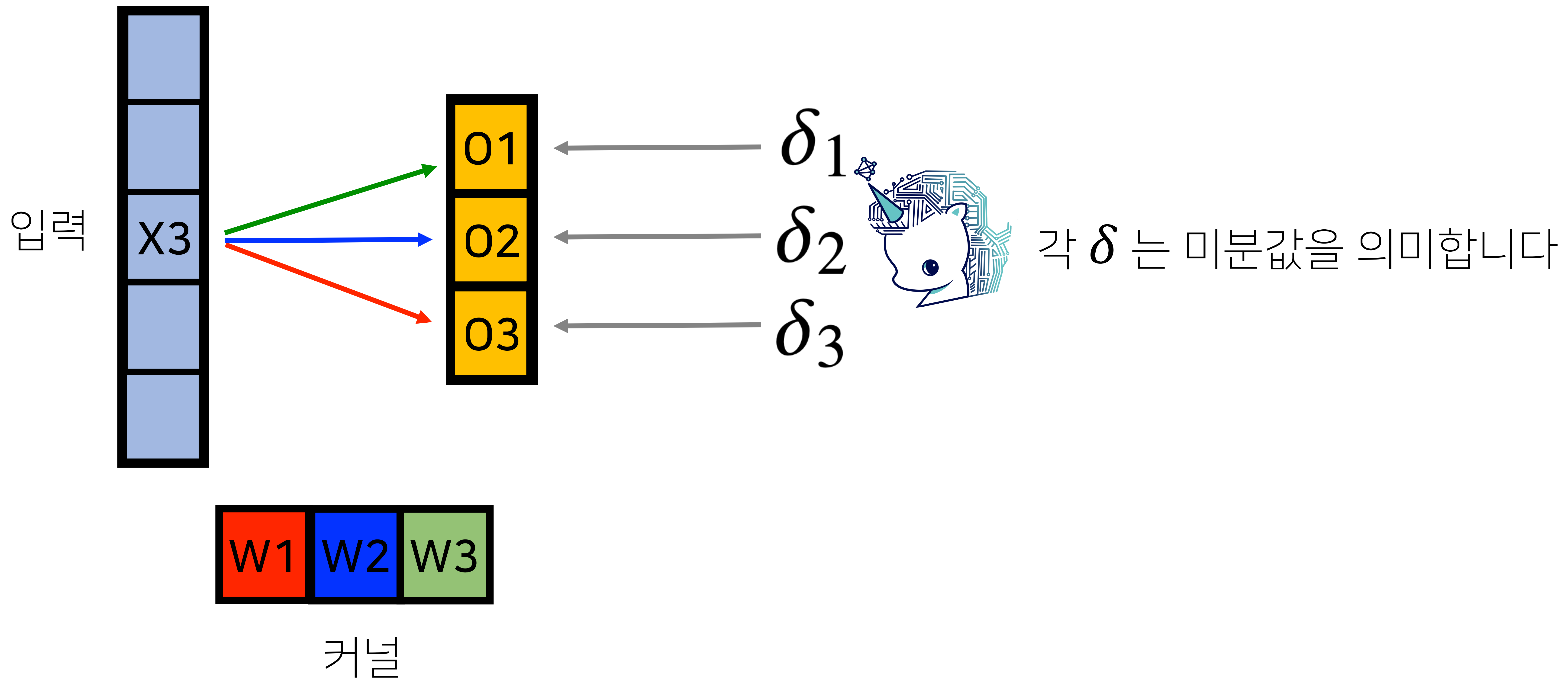
$$o_i = \sum_j w_j x_{i+j-1}$$



그림으로 이해하면 훨씬 쉽습니다

Convolution 연산의 역전파 이해하기

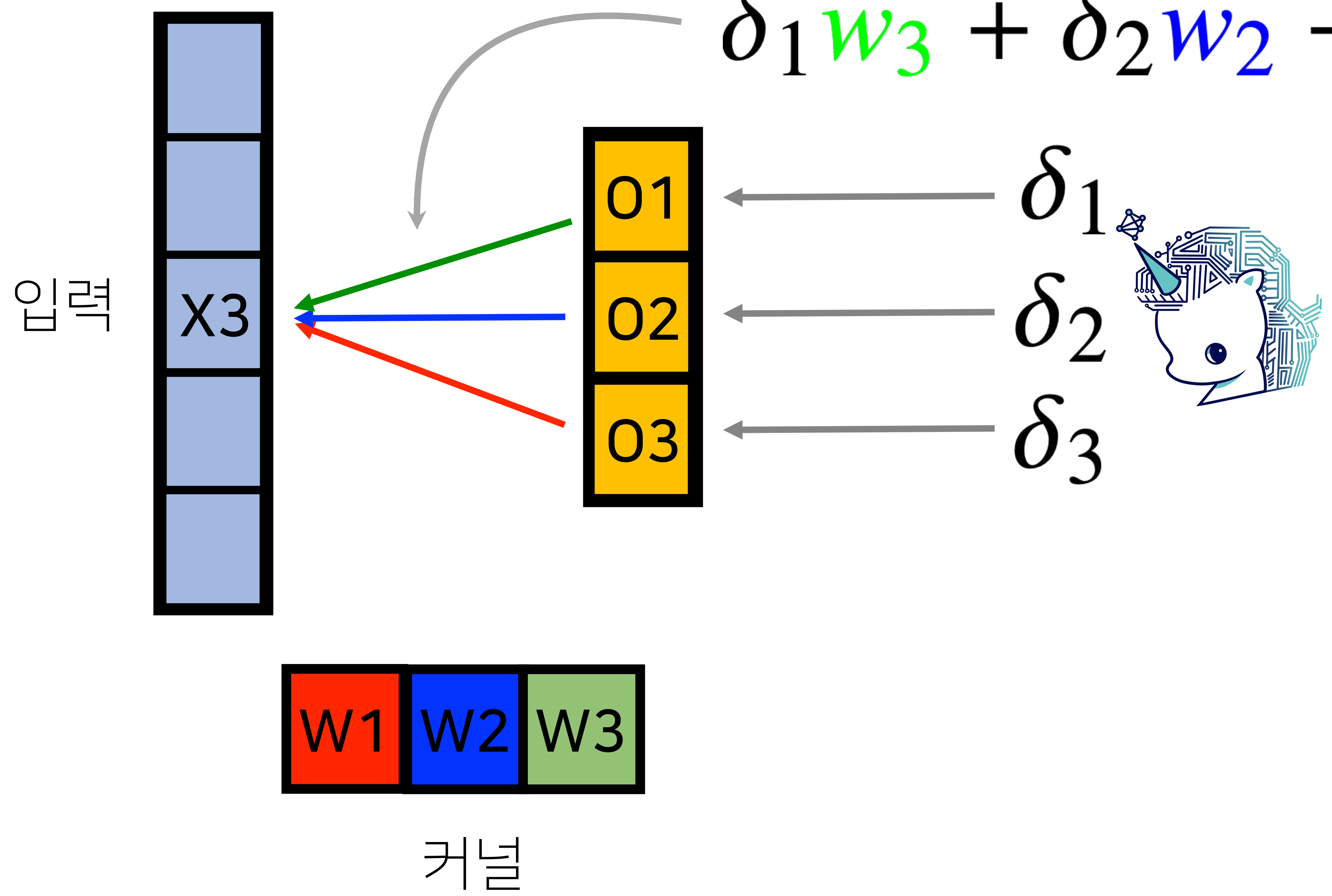
- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다



Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

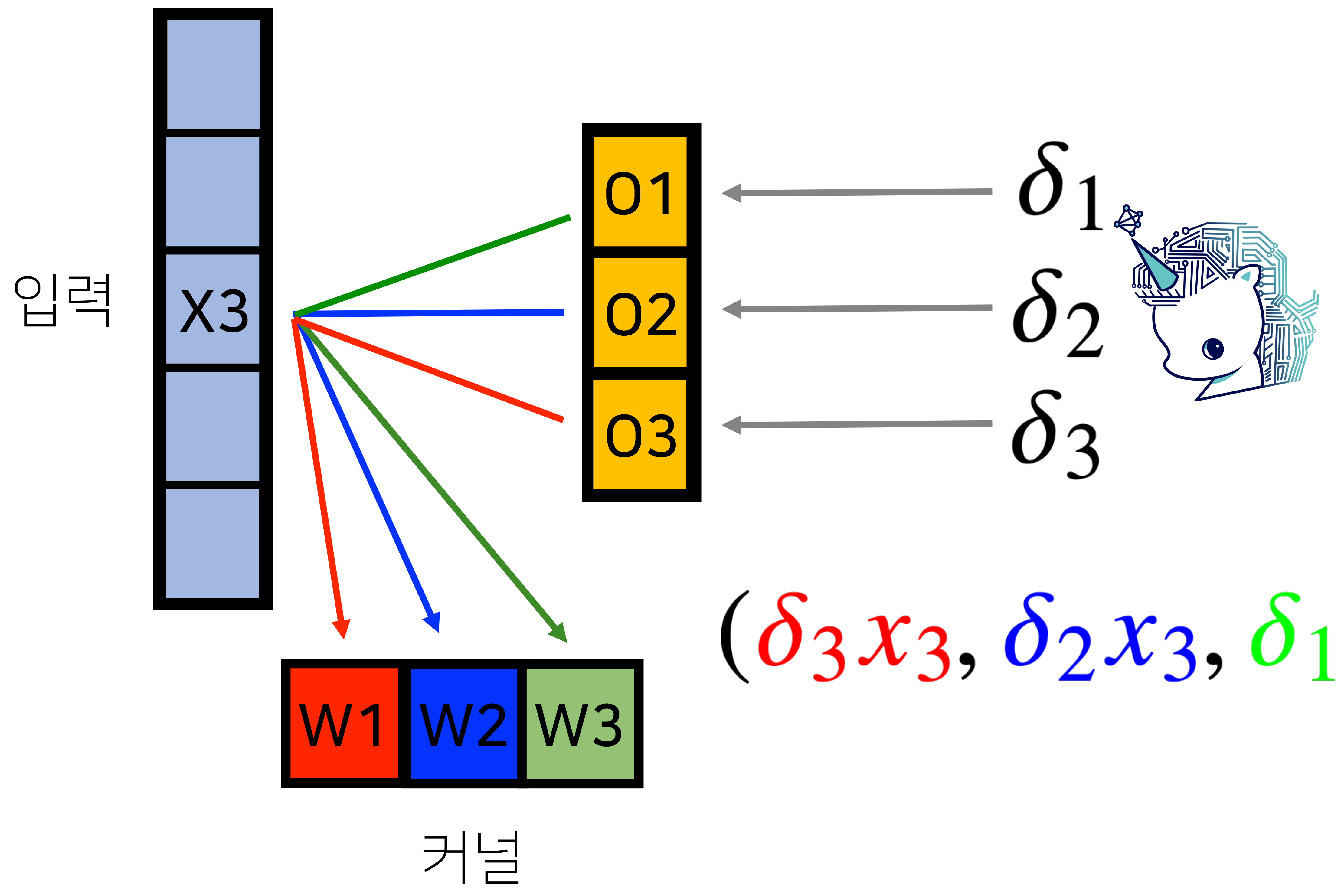
$$\delta_1 w_3 + \delta_2 w_2 + \delta_3 w_1$$



역전파 단계에서 다시 커널을 통해 그레디언트가 전달된다

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

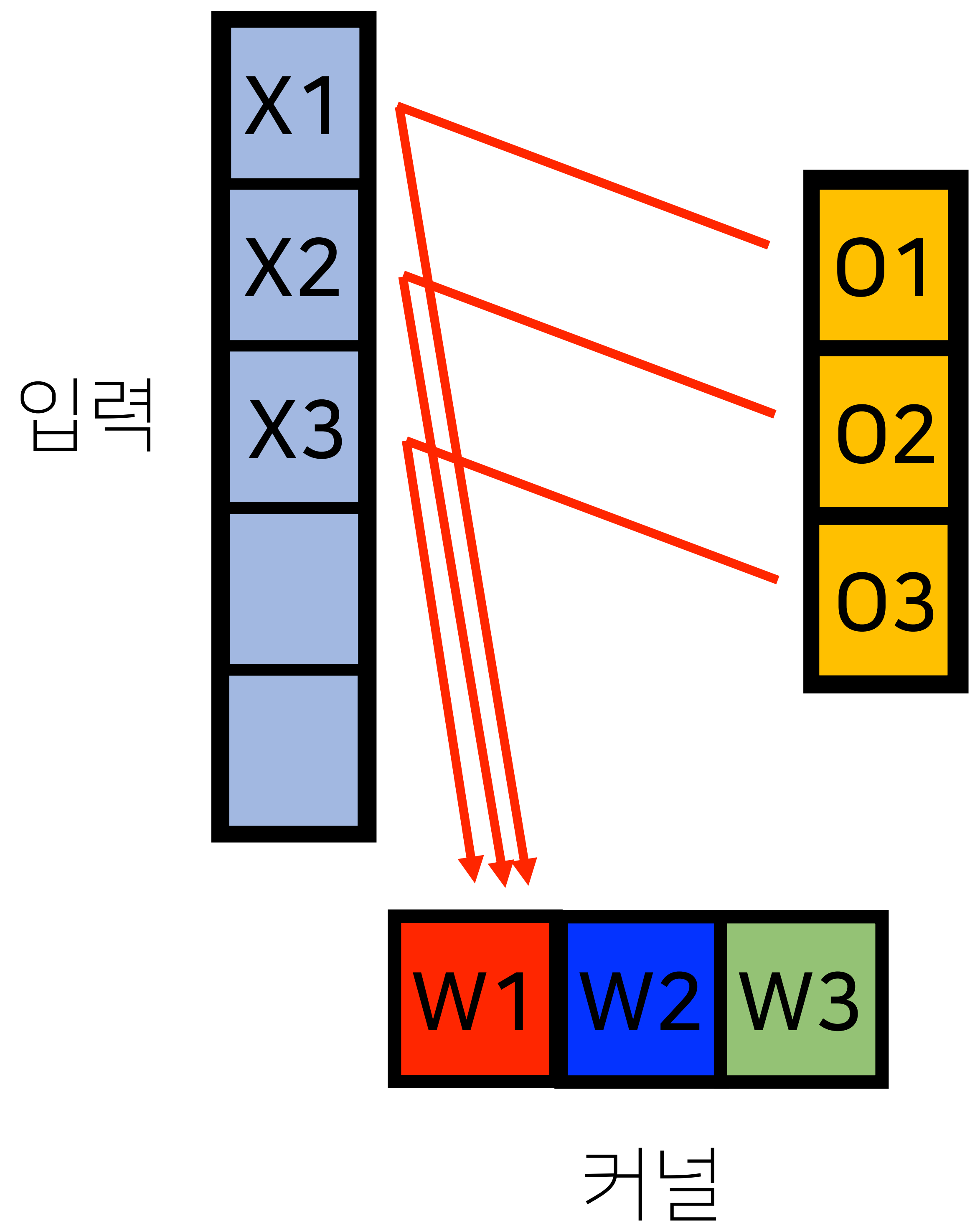


커널에는 δ 에 입력값 x_3 을 곱해서 전달한다

$$(\delta_3 x_3, \delta_2 x_3, \delta_1 x_3)$$

Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다



δ_1, δ_2, δ_3

$$\frac{\partial \mathcal{L}}{\partial w_i} = \sum_j \delta_j x_{i+j-1}$$

각 커널에 들어오는 모든 그레디언트를 더하면 결국 convolution 연산과 같다

$$\frac{\partial \mathcal{L}}{\partial w_1} = \delta_1 x_1 + \delta_2 x_2 + \delta_3 x_3$$

THE END

다음 시간에 보아요!