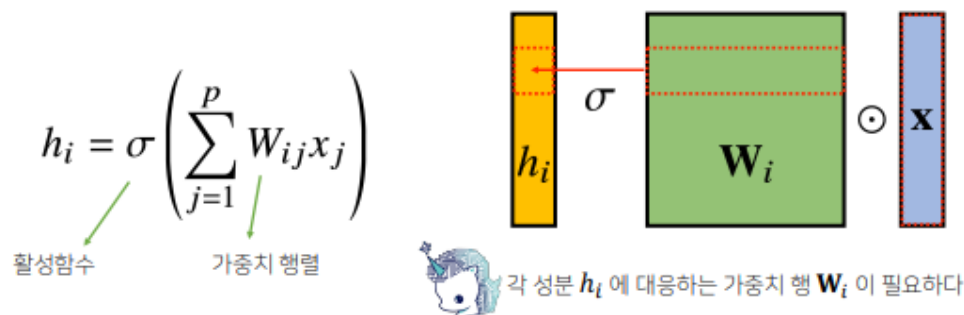
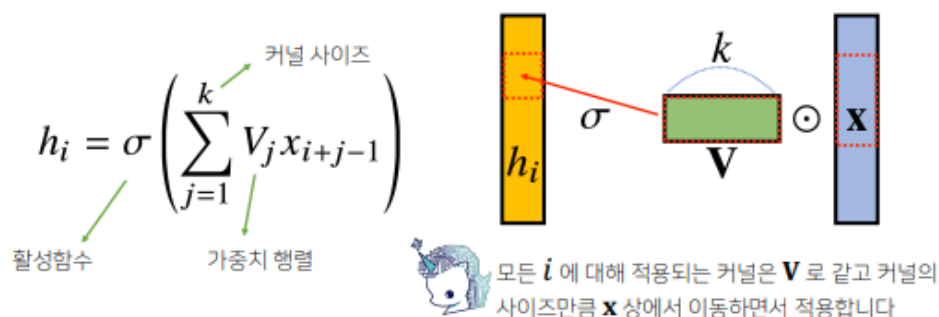


# 1. Convolution 연산 이해하기

다층신경망(MLP)은 각 뉴런들이 선형모델과 활성화함수로 모두 연결된(fully connected) 구조였다.



Convolution 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직이면서 선형모델과 합성함수가 적용되는 구조이다.



Convolution 연산의 수학적 의미는 신호(signal)를 커널을 이용해 국소적으로 증폭 또는 감소시켜 정보를 추출 또는 필터링하는 것이다. 정의역이 연속인 공간에서 적분을 사용하거나 공간이 이상 공간이면 적분을 할 수가 없으므로 급수로 정의할 수 있다.

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

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

Convolution 을 수식으로만 이해하는 것은 매우 어렵습니다

사실 convolution neural network에서 사용하는 연산은 뺄셈이 아니라 더하기를 사용한 cross-correlation을 사용하므로 엄밀히 말해서 cross-correlation 연산이라 보는게 맞다. 하지만 전체 공간에서 더하기냐 뺄셈이냐는 중요하지 않기 때문에 convolution 이나 cross-correlation 이나 똑같이 성립한다. 하지만 컴퓨터에서 더하기냐 뺄셈이냐는 큰 차이가 있다. 하지만 관례적으로 convolution 이라고 부른다.

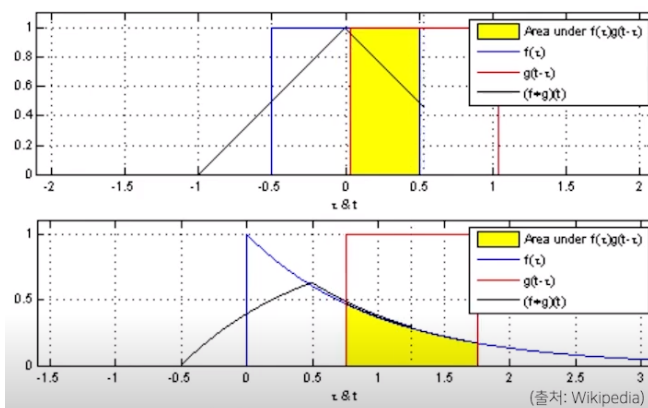
continuous  $[f * g](x) = \int_{\mathbb{R}^d} f(z)g(x \oplus z)dz = \int_{\mathbb{R}^d} f(x \oplus z)g(z)dz = [g * f](x)$

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



CNN에서 사용하는 연산은 사실 convolution 이 아니고 cross-correlation 이라 부릅니다

커널은 정의역 내에서 움직여도 변하지 않고(translation invariant) 주어진 신호에 국소적(local)으로 적용한다. 아래를 보면 빨간색 커널을 움직이면서 파란색 신호에 적용될 때 노란색에 해당하는 것이 국소적으로 적용되는 연산이며 검은색이 그에 대한 결과이다. 즉, 파란색 함수를 검은색 함수로 확장시키거나 추출하거나 감소시키는 역할을 convolution이 수행하는 것이다.



파란색이 신호  
빨간색이 커널  
검은색이 결과입니다

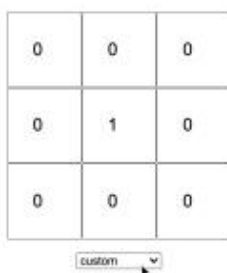
© NAVER Connect Foundation

(출처: Wikipedia)

8

## 2. 영상처리에서 Convolution

아래 페이지에서 kernel의 종류를 바꾸면 이미지의 형태가 계속 바뀐다.




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

For more, have a look at Gimp's excellent documentation on using [image kernels](http://setosa.io/ev/image-kernels/). You can also apply your own custom filters in Photoshop by going to Filter -> Other -> Custom...

### 3. 다양한 차원에서의 Convolution

Convolution 연산은 1차원뿐만 아니라 다양한 차원에서 계산할 수 있다. 1차원에서는 한 변수에서만 움직이는 것이라면 2차원은 두 개의 변수에서 커널을 움직이는 것이다. 3차원은 세 개의 변수에서 커널을 움직이면 된다.

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

  $i, j, k$  가 바뀌어도 커널  $r$  의 값은 바뀌지 않습니다

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

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

### 4. 2차원 Convolution 연산

2D-Conv 연산은 커널(kernel)을 입력벡터 상에서 움직이면서 선형모델과 합성함수가 적용되는 구조이다.

$$[f * g](i, j) = \sum_{p, q} \overset{\text{커널}}{f(p, q)} \underset{\text{입력}}{g(i+p, j+q)}$$

0	1
2	3


커널

⊙

0	1	2
3	4	5

입력

19	


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

커널을 움직여도 커널의 값은 바뀌지 않고 입력에 해당하는 값만 바뀐다.

$$[f * g](i, j) = \sum_{p, q} \overset{\text{커널}}{f(p, q)} \underset{\text{입력}}{g(i+p, j+q)}$$

0	1
2	3


커널

⊙

0	1	2
3	4	5

입력

19	25


 $0 \times 1 + 1 \times 2 + 2 \times 4 + 3 \times 5 = 25$

입력 크기(H, W), 커널 크기를 (K\_h, K\_w), 출력 크기를(O\_h, O\_w)라 하면 출력 크기는 다음과 같이 계산한다.

$$O_H = H - K_H + 1$$

$$O_W = W - K_W + 1$$

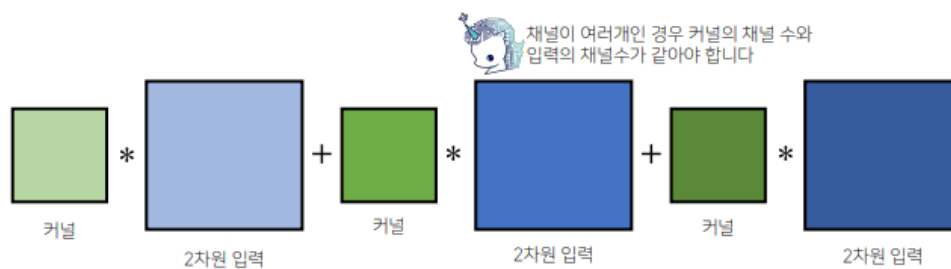


가령 28x28 입력을 3x3 커널로  
2D-Conv 연산을 하면 26x26 이 된다

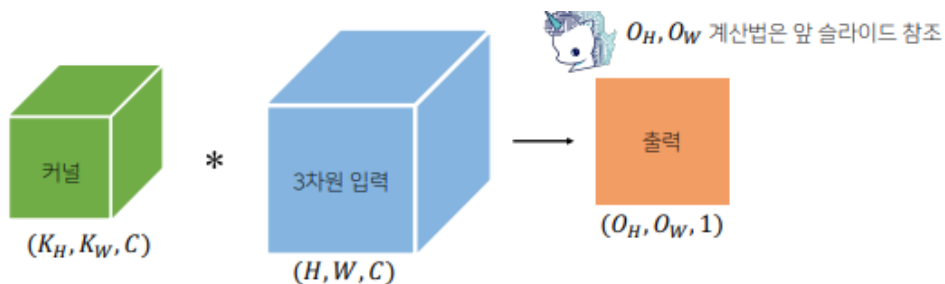
채널이 여러개인 2차원 입력의 경우 2차원 Convolution을 채널 개수만큼 적용하면 된다.



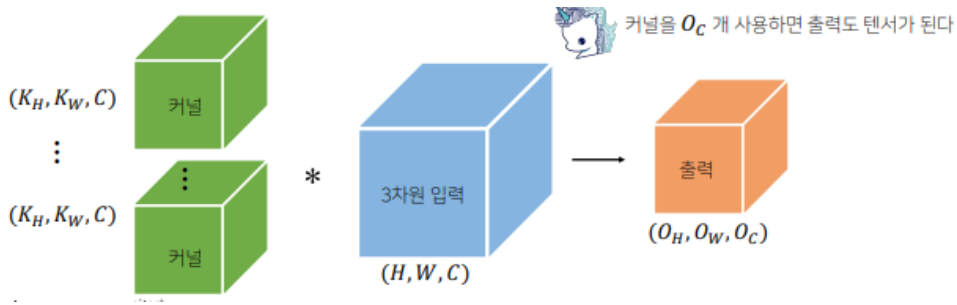
각각의 채널 개수만큼 2차원 입력을 분리한 상태에서 각 커널을 각각의 2차원 입력에 convolution 연산을 적용하고 그 결과를 더해준다. 따라서 커널의 개수는 채널의 개수와 같아야 한다.



텐서를 직육면체 블록으로 이해하면 좀더 이해하기 쉽다.



만약 출력이 여러 개의 채널을 가지게 하고 싶다면 커널의 개수를 늘려주면 된다.



## 5. Convolution 연산의 역전파

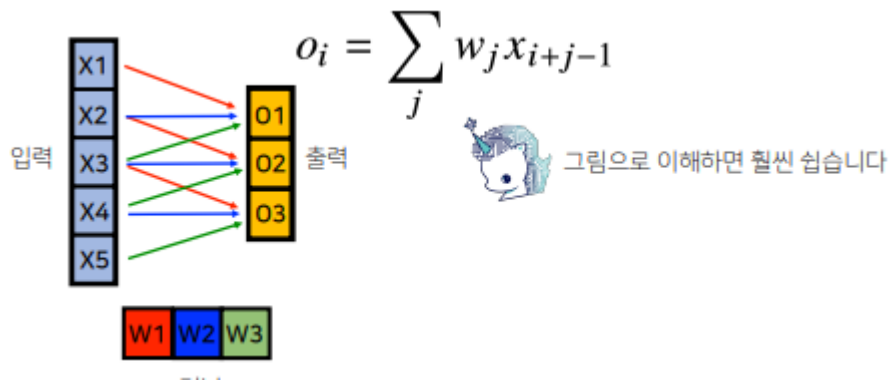
Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 Convolution 연산이 나오게 된다. 아래 수식을 보면 Convolution 연산에 미분을 해도 똑같이 Convolution 연산을 하면 된다.

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

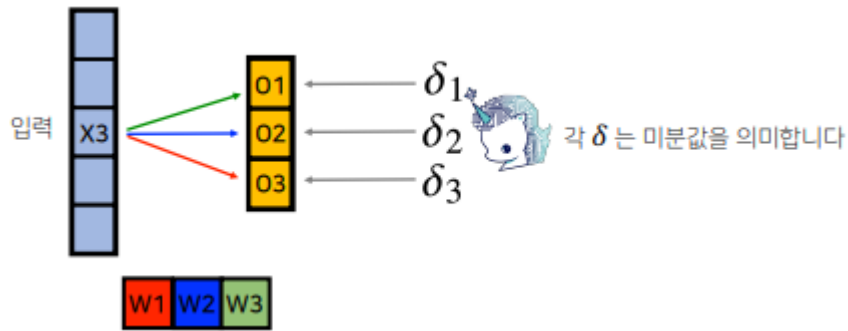


Discrete 일 때도 마찬가지로 성립한다

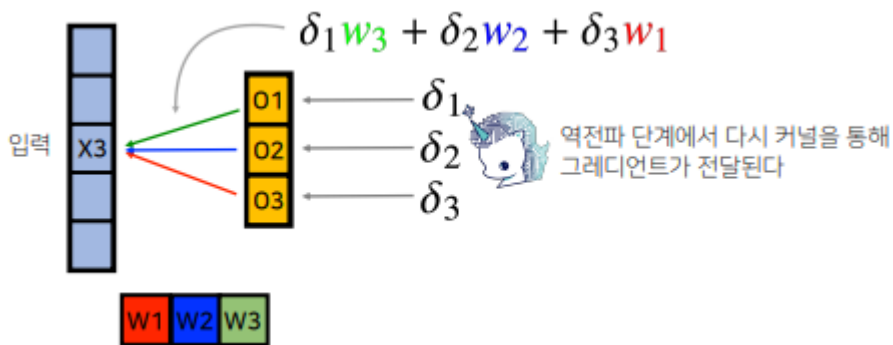
아래는 Convolution 연산 과정을 그림으로 표현한 것이다.



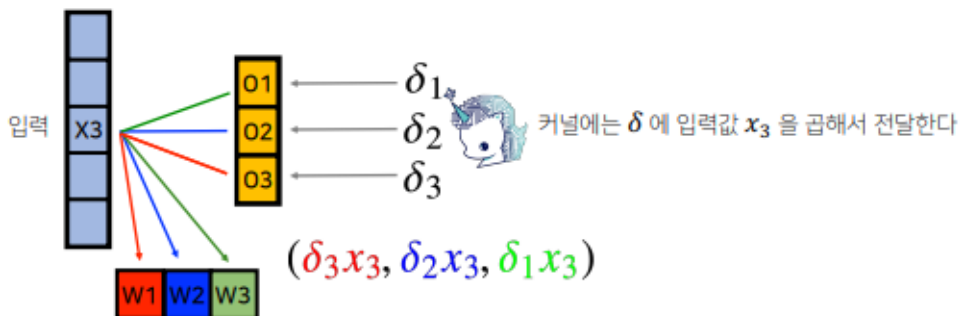
이번에는 역전파 과정을 생각해보자. 출력 벡터에는 미분값이 들어있다 하자.



이전에  $x_3$ 에서  $o_1$ 에는  $w_3$ ,  $o_2$ 에는  $w_2$ ,  $o_3$ 에는  $w_1$ 을 적용했다. 해당 가중치와 출력값을 일치시켜 연산을 진행한다.



각각의 커널들에는 어떻게 그래디언트가 전달될까?  $o_3$ 가  $x_3$ 에  $w_1$ 을 통해 그래디언트를 전달했으므로  $x_3$ 를  $\delta_3$ 와 곱해서  $w_1$ 에 전달한다. 다른 값들도 마찬가지다.



다른 입력들에도 위 과정이 똑같이 진행되어 아래와 같이 각각의 델타들이 그래디언트를 통해 전달된다. 이것을 통해  $x_1, x_2, x_3$ 가  $w_1$ 에 해당하는 그래디언트를 전달하게 된다.

