모두의연구소

지난시간 Review

모두의연구소
박은수 Research Director

지난시간 돌아보기 …

모두의연구소

- #lec_deeplearning
- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

Loss를 최소화하는 w와 b를 찾아라

내리막을 찾는 방법

모두의연구소

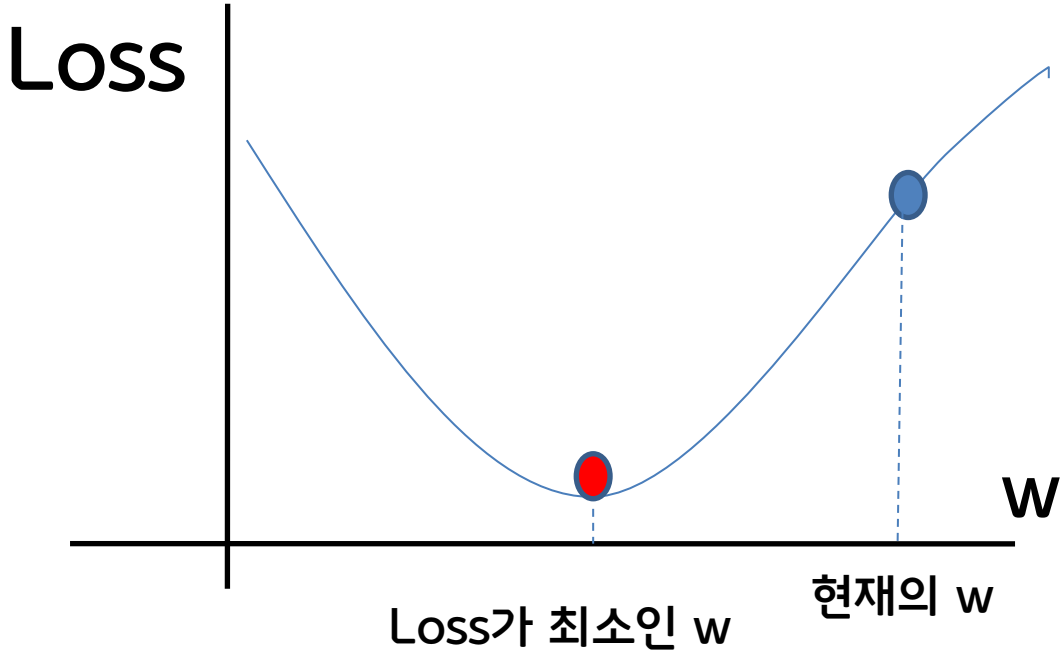
내리막 ?? == 기울기 ??

기울기를 따라 내려 가보자

Gradient Descent

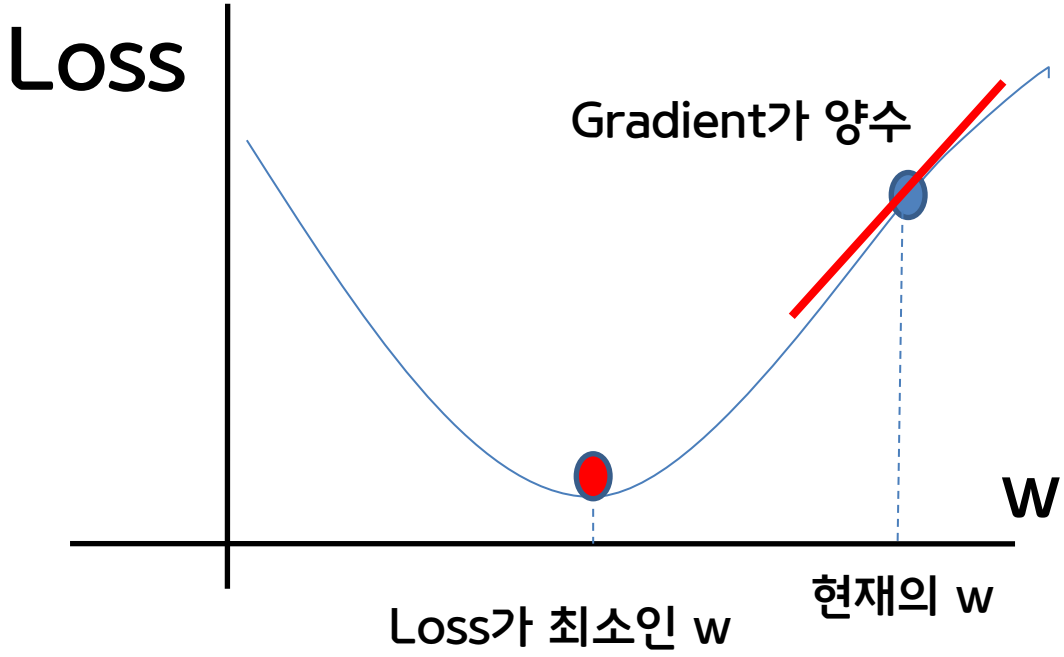
Gradient Descent

모두의연구소



$$w = w - \eta \frac{\partial L}{\partial w}$$

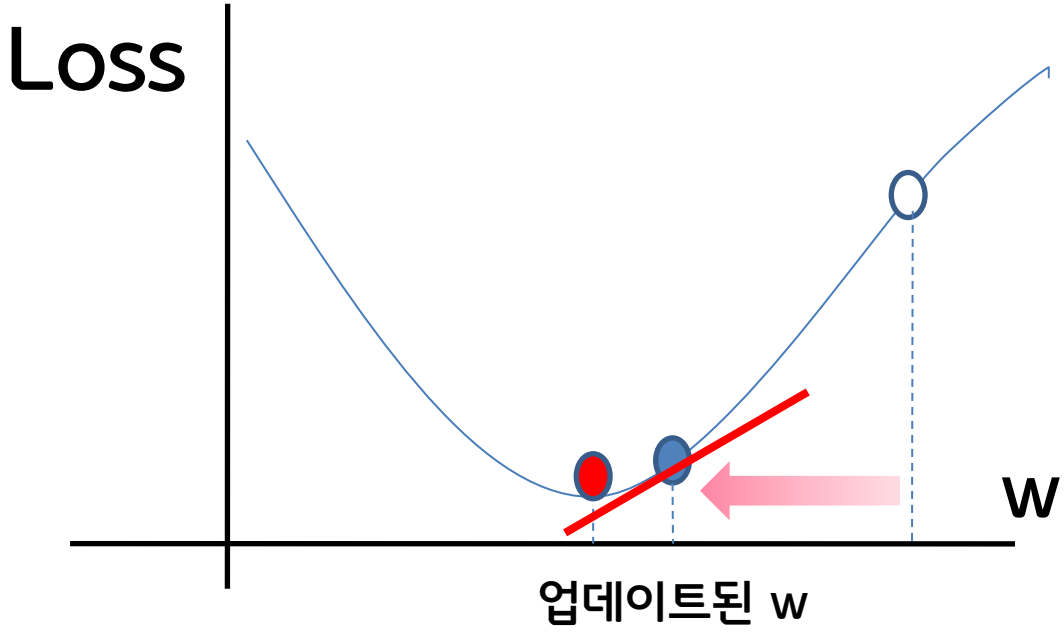
Gradient Descent



$$w = w - \eta \frac{\partial L}{\partial w}$$

η : learning rate

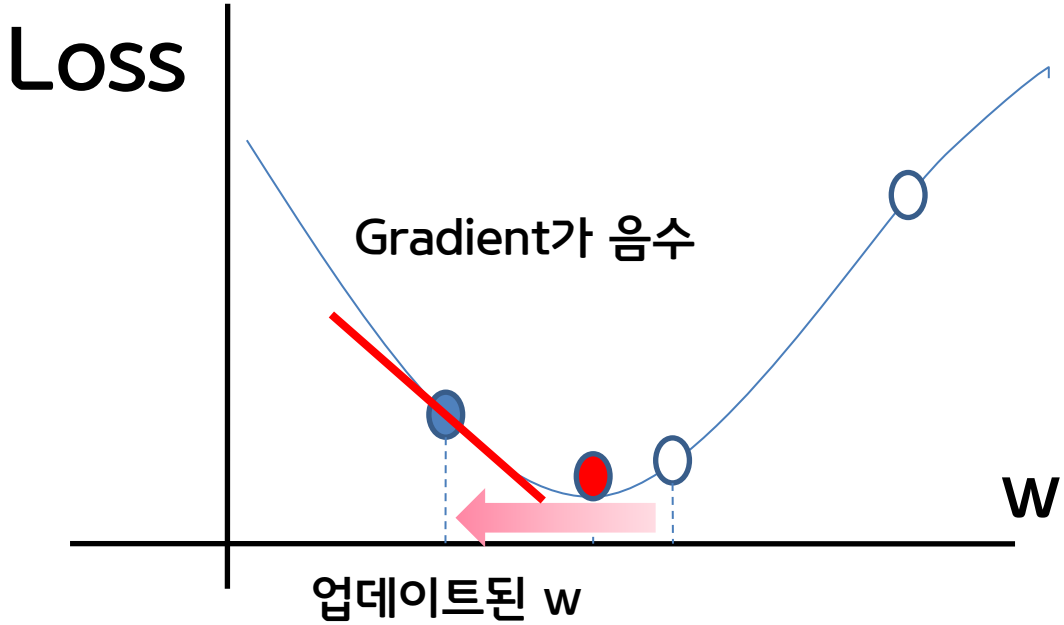
Gradient Descent



$$w = w - \eta \frac{\partial L}{\partial w}$$

η : learning rate

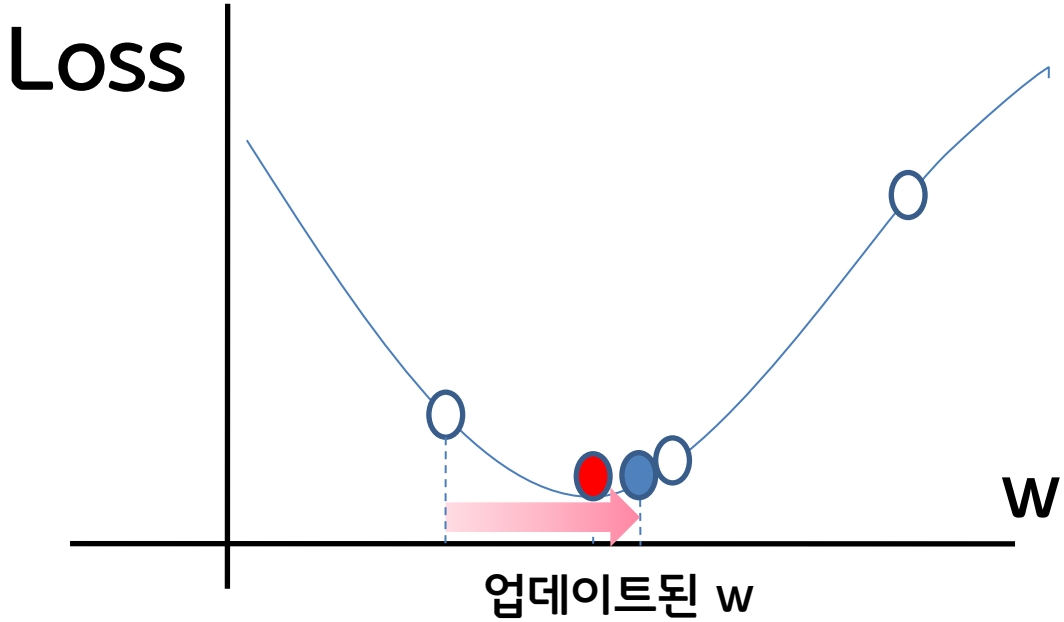
Gradient Descent



$$w = w - \eta \frac{\partial L}{\partial w}$$

η : learning rate

Gradient Descent



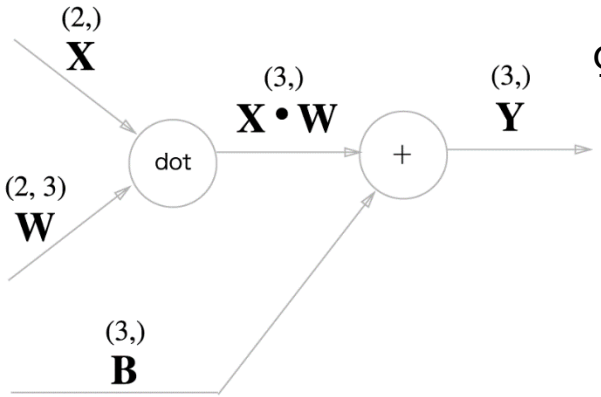
$$w = w - \eta \frac{\partial L}{\partial w}$$

Loss 함수에 대한 w의 음의 Gradient 찾아서 연속적으 업데이트해 주면 되는군요

그런데 어떻게 Gradient를 찾죠?

Computational Graph

- Affine 계층



역전파

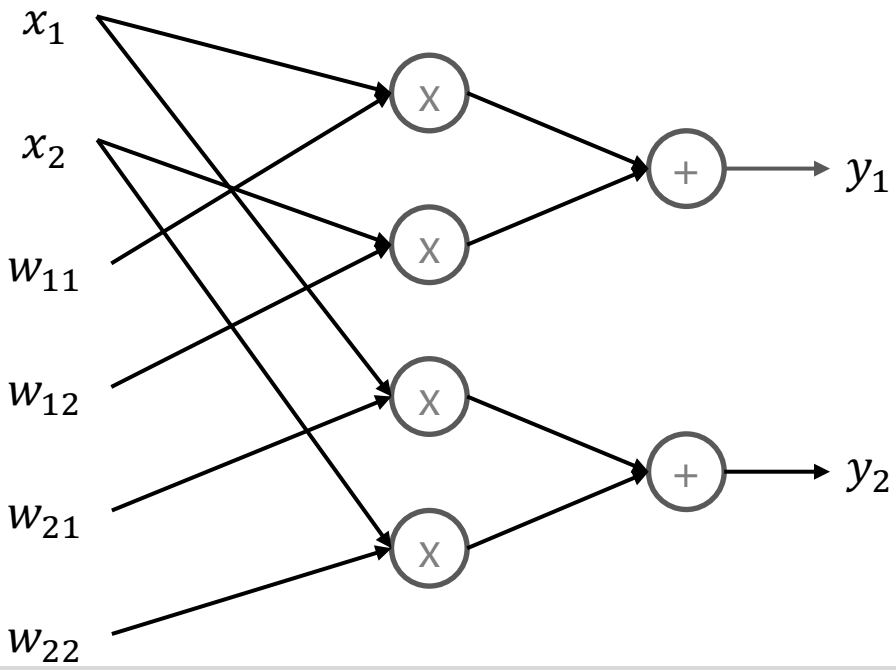
$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

Computational Graph

- Affine 계층

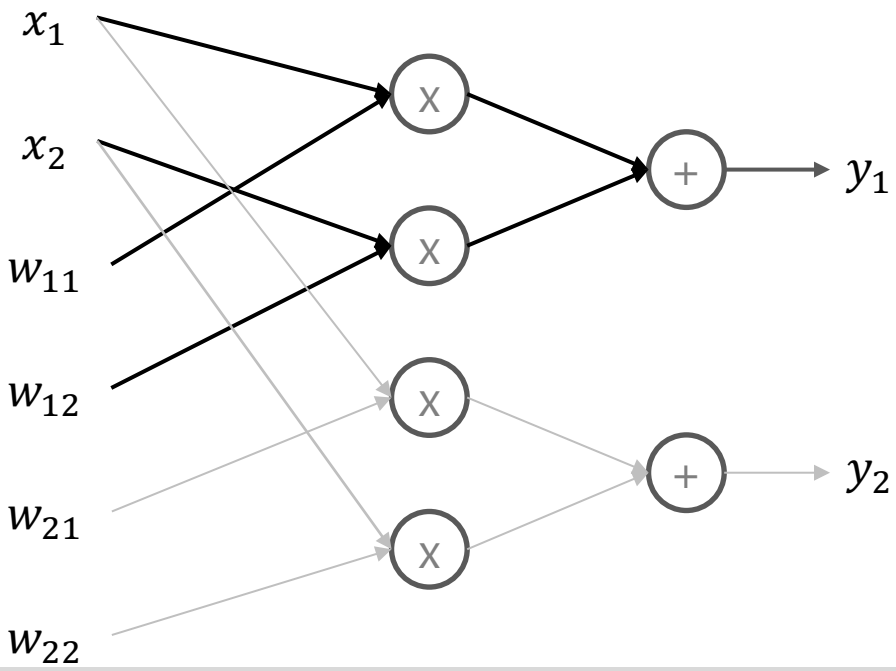
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

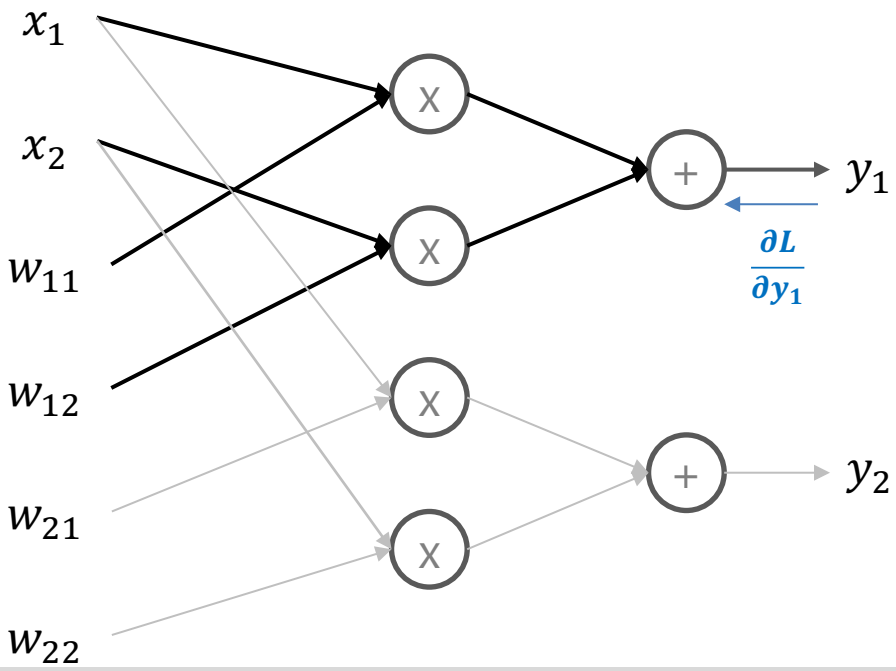
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

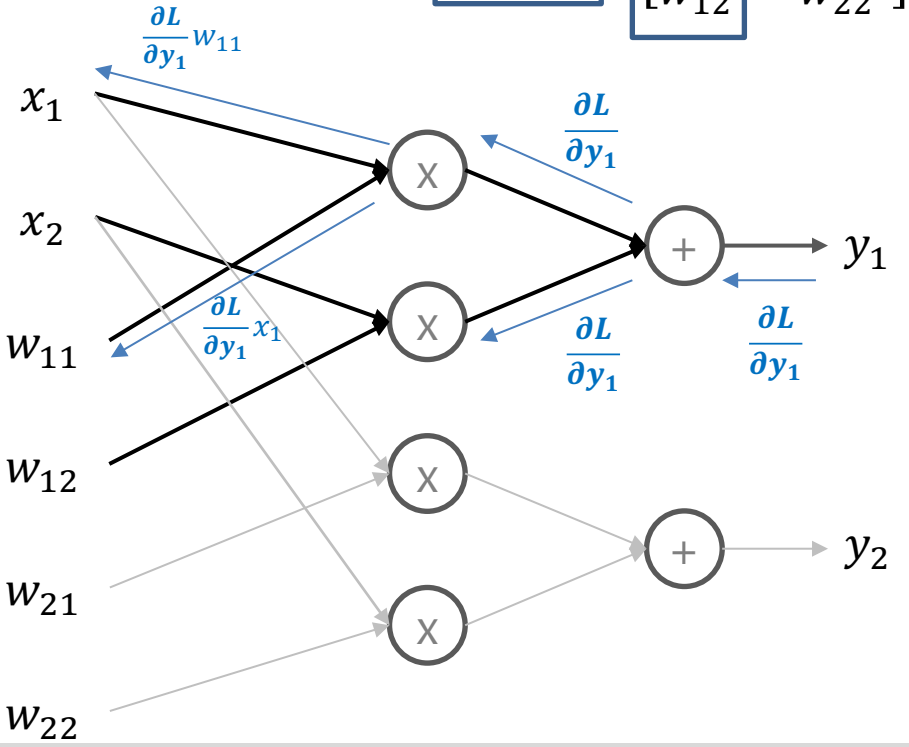
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

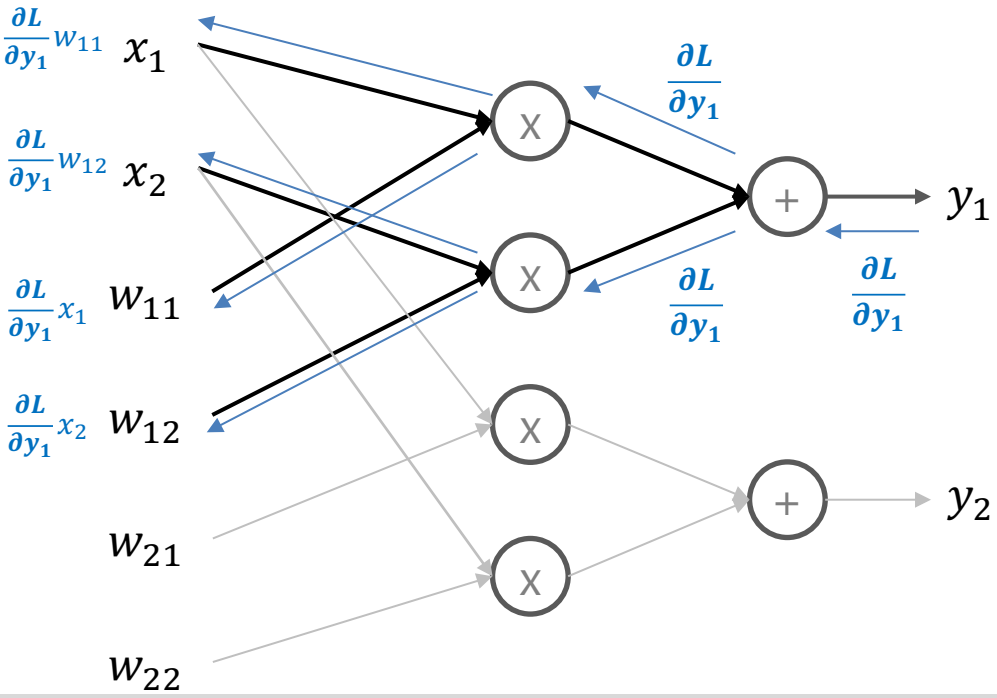
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

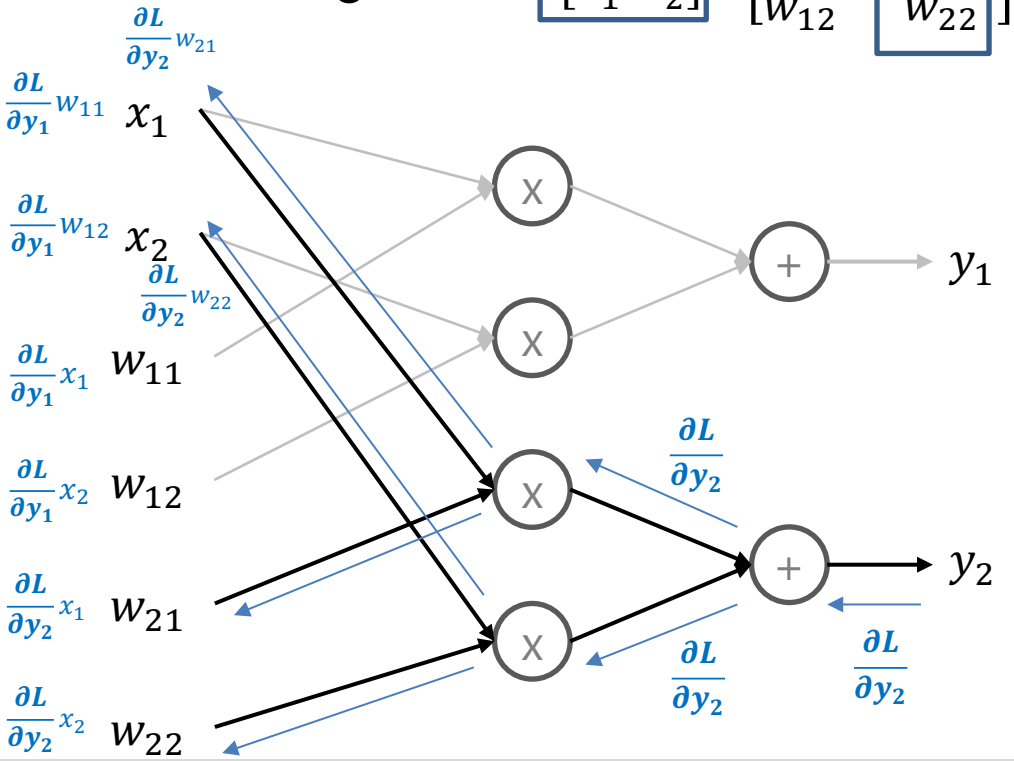
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

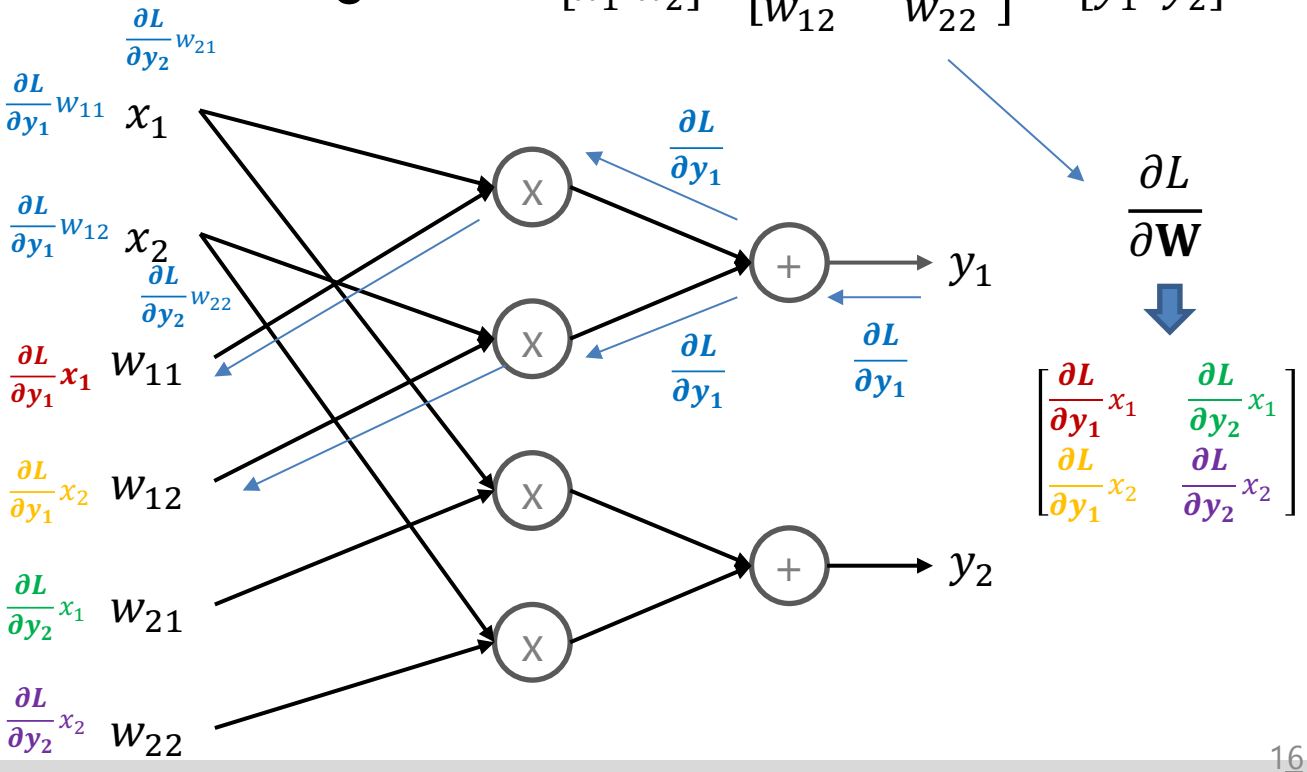
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

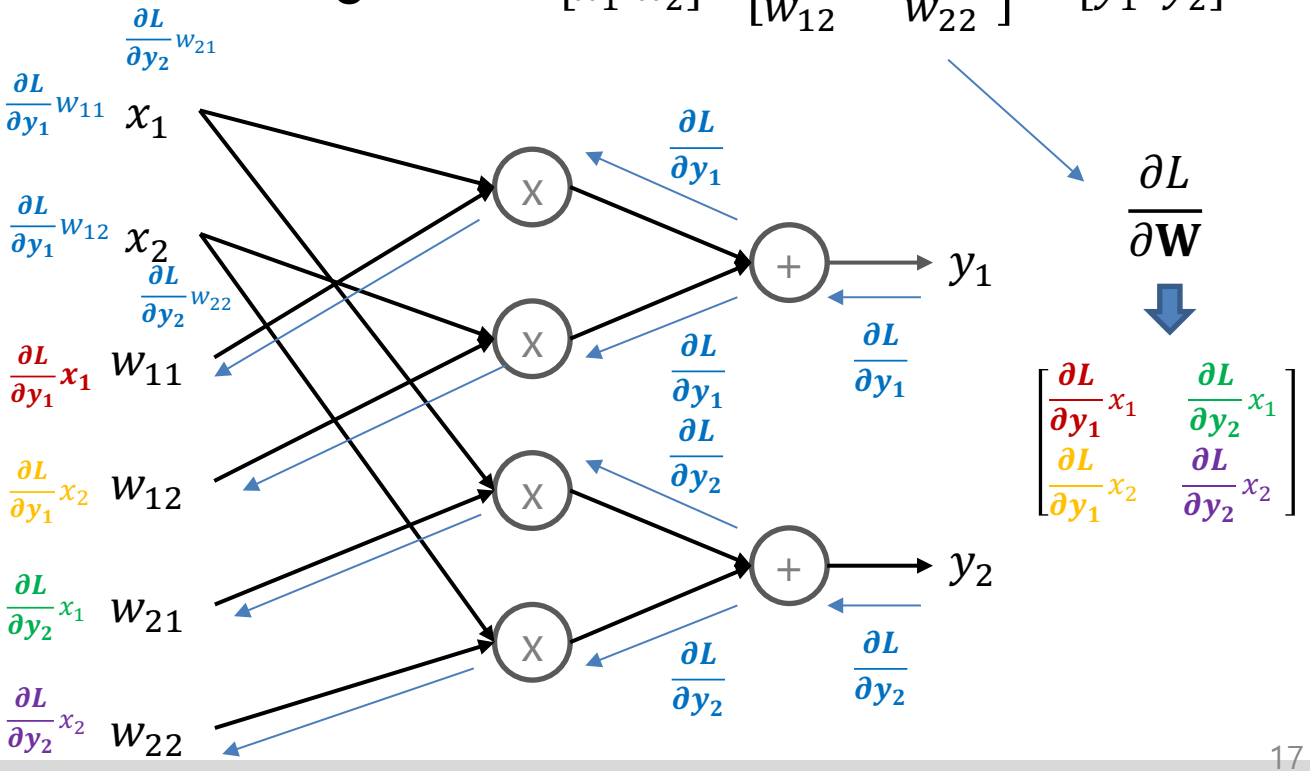
$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \;\; y_2]$$



Computational Graph

- Affine 계층

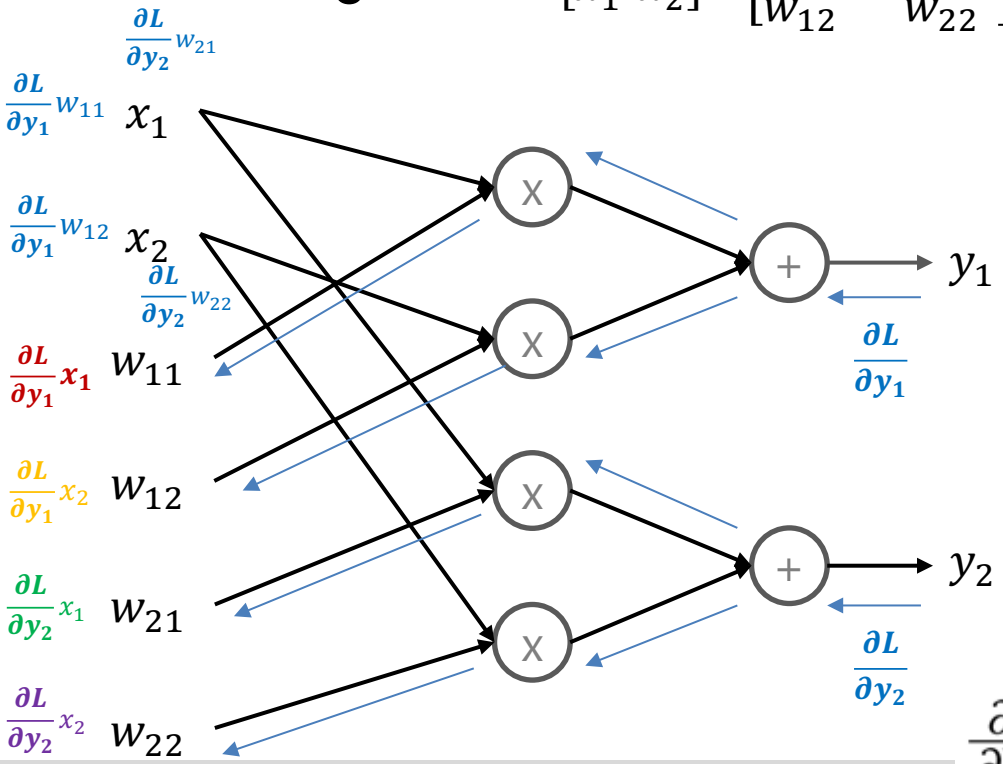
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \;\; y_2]$$



$$\frac{\partial L}{\partial \mathbf{W}}$$

$$\begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 & \frac{\partial L}{\partial y_2} x_1 \\ \frac{\partial L}{\partial y_1} x_2 & \frac{\partial L}{\partial y_2} x_2 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} \end{bmatrix}$$

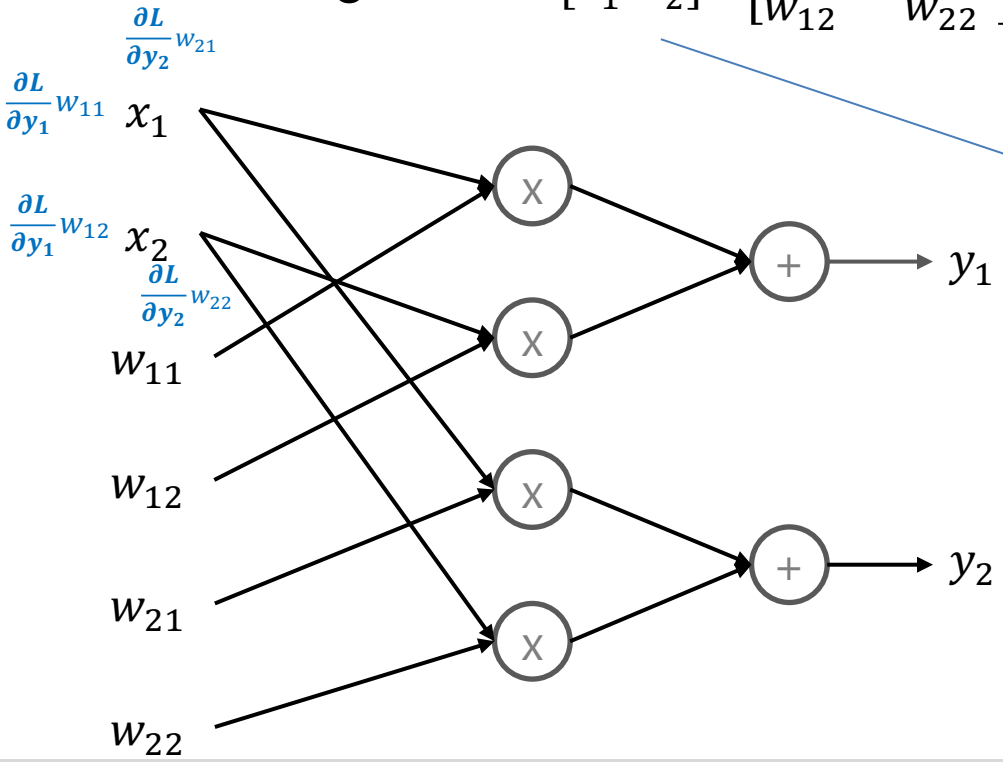
자코비안

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

Computational Graph

- Affine 계층

$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



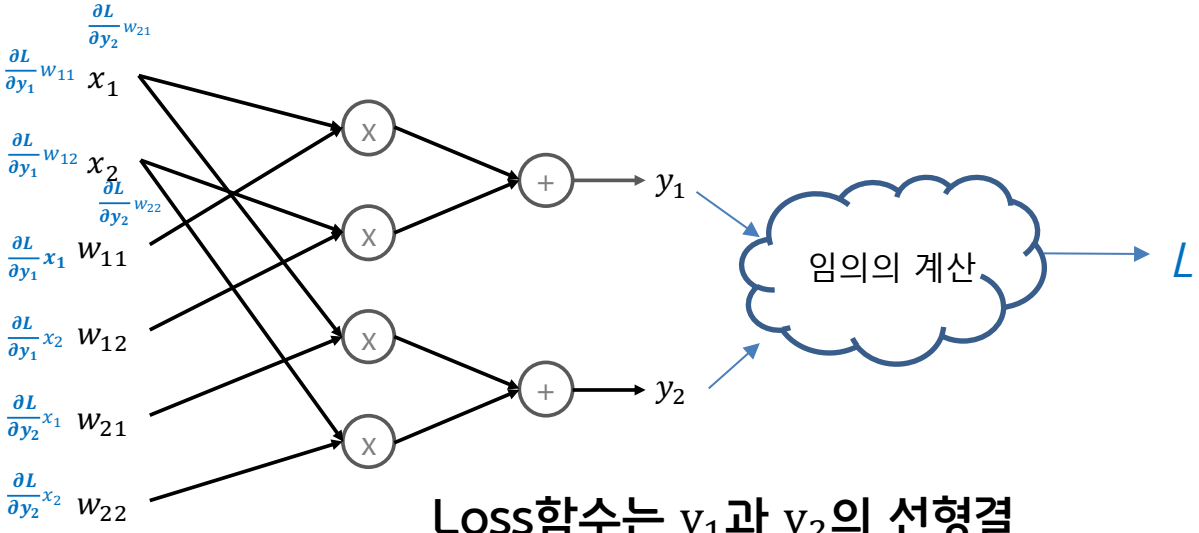
$$\frac{\partial L}{\partial \mathbf{X}}$$

2개씩 나왔어요
ㅠㅠ

Computational Graph

- Affine 계층

$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



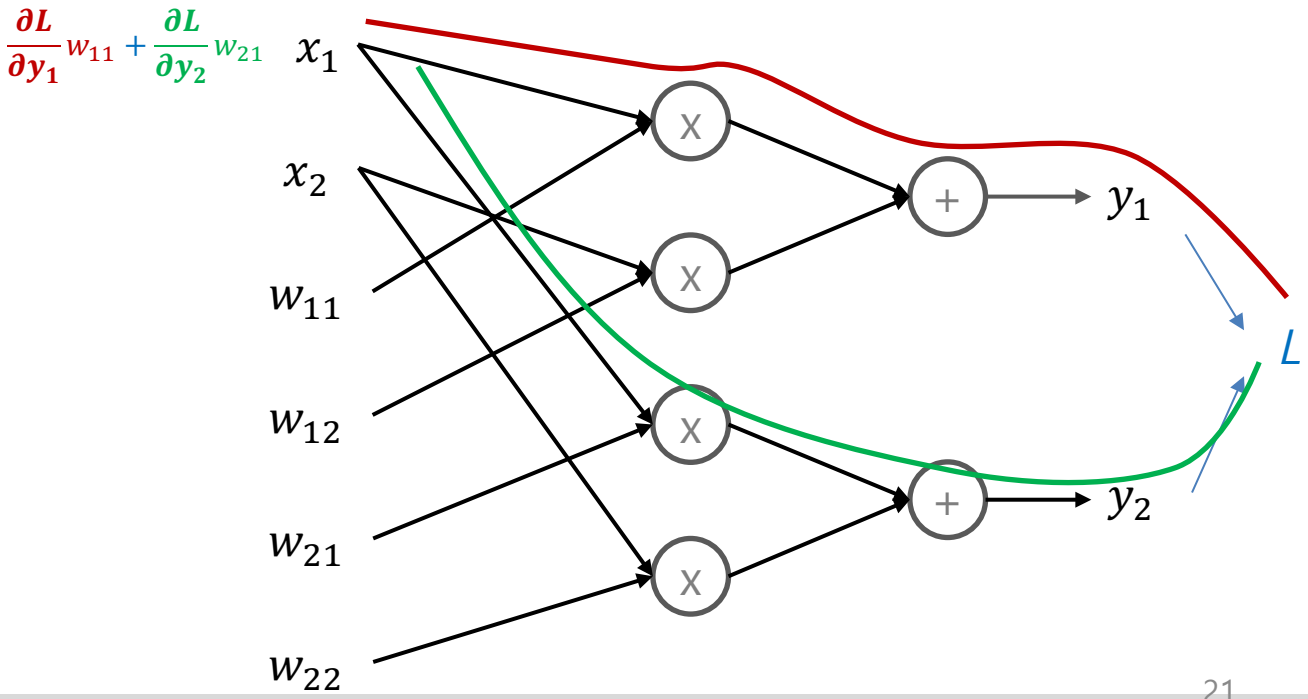
Loss함수는 y_1과 y_2의 선형결 합일 겁니다

예) $L = ay_1 + by_2$

Computational Graph

- Affine 계층

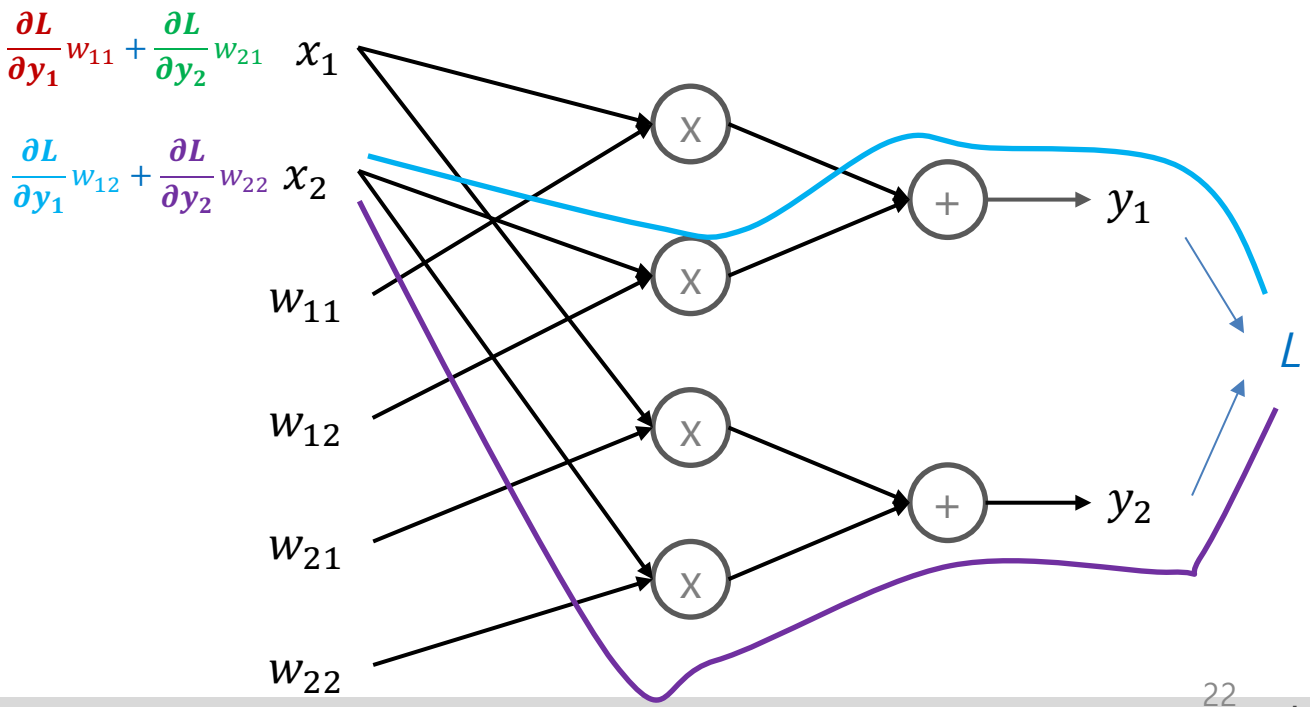
$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

- Affine 계층

$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$



Computational Graph

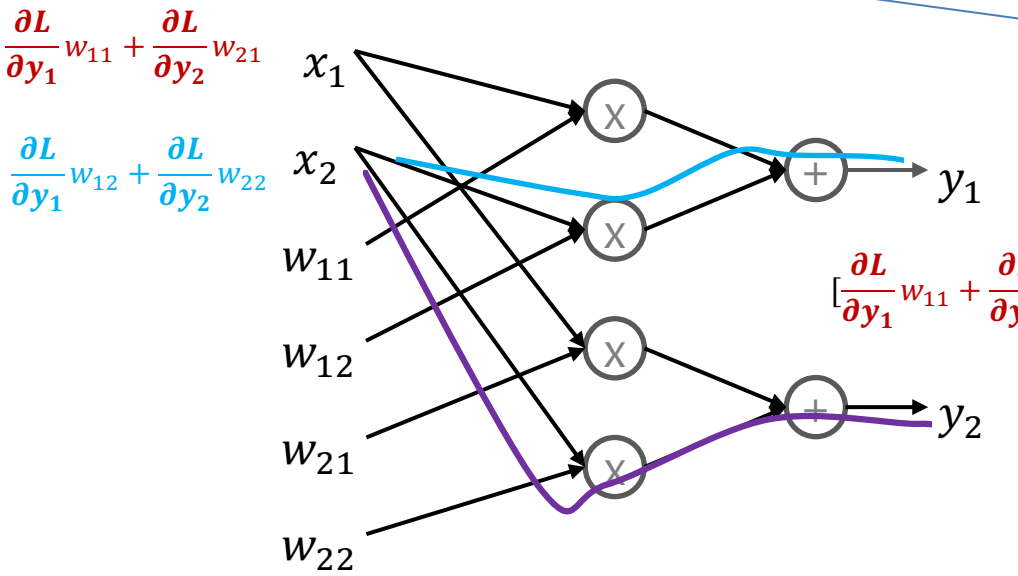
모두의연구소

- Affine 계층

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \;\; y_2]$$

$$\frac{\partial L}{\partial \mathbf{X}}$$

$$\left[\frac{\partial L}{\partial y_1} w_{11} + \frac{\partial L}{\partial y_2} w_{21} \quad \frac{\partial L}{\partial y_1} w_{12} + \frac{\partial L}{\partial y_2} w_{22}\right]$$



Computational Graph

- Affine 계층

$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = [y_1 \ y_2]$$

$$\frac{\partial L}{\partial \mathbf{X}} = [\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y_1}} w_{11} + \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y_2}} w_{21} \quad \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y_1}} w_{12} + \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y_2}} w_{22} \ ]$$

⇓

자코비안 $\left[\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y_1}} \quad \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y_2}}\right] \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$

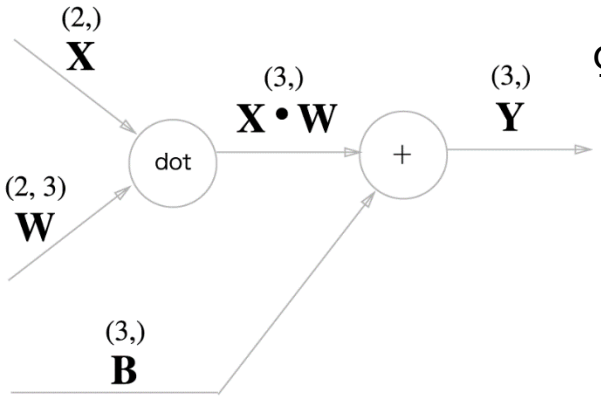
⇓

이제 다시 예제로 돌아와 봅시다

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

Computational Graph

- Affine 계층



역전파

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

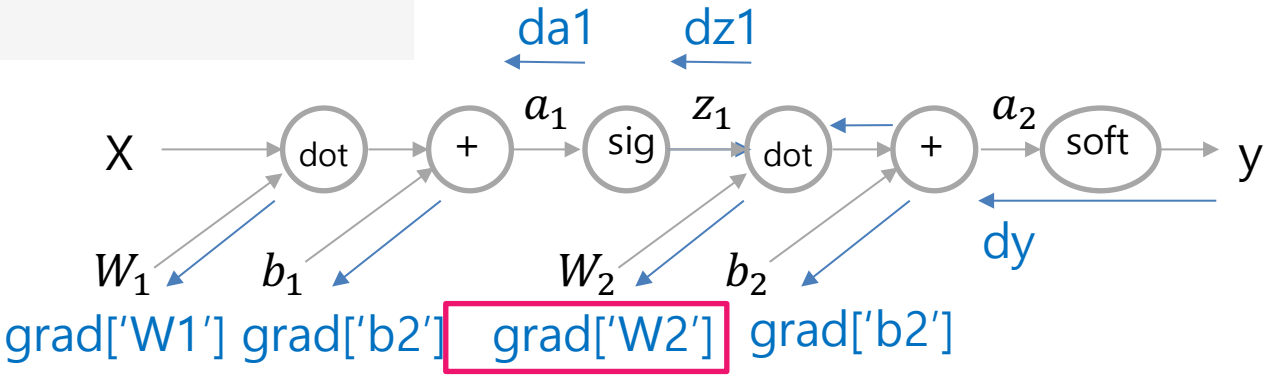
$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

숙제 : 구자현 님 (1등)

```
# backward
dy = (y - t) / batch_num
###################################
######### Write your codes #######
###################################
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis = 0)

dz1 = np.dot(dy, W2.T)
da1 = dz1 * (1 - z1) * z1

grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(da1, axis = 0)

return grads
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

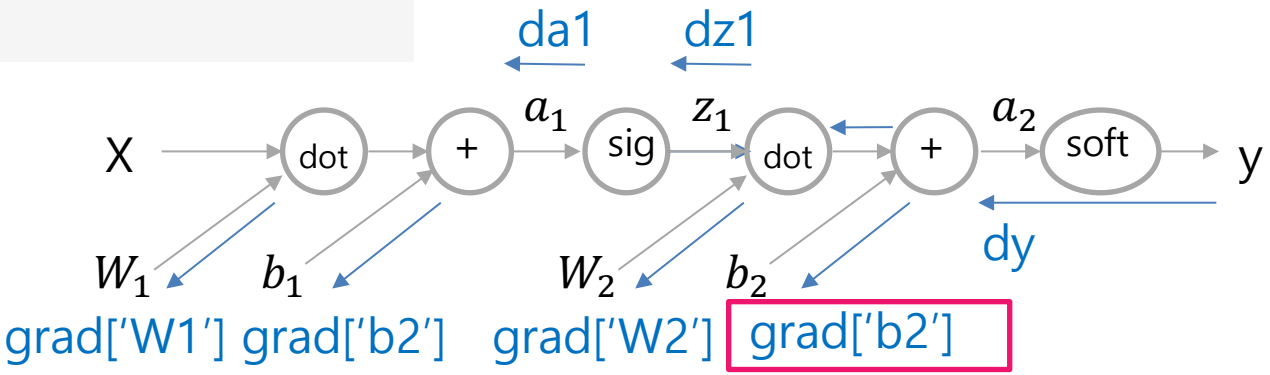


숙제 : 구자현 님 (1등)

```
# backward
dy = (y - t) / batch_num
##################################
######### Write your codes #######
##################################
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis = 0)

dz1 = np.dot(dy, W2.T)
da1 = dz1 * (1 - z1) * z1

grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(da1, axis = 0)

return grads
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

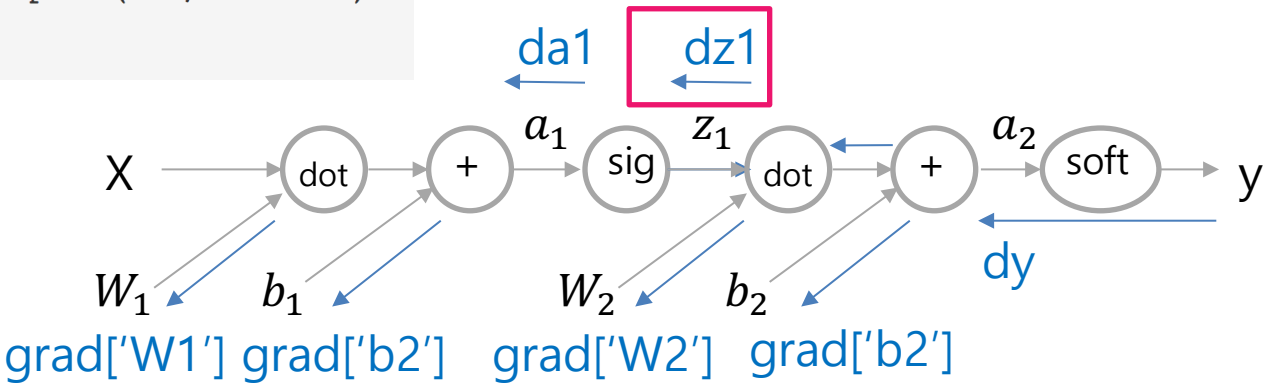


숙제 : 구자현 님 (1등)

```
# backward
dy = (y - t) / batch_num
#################################
######### Write your codes ######
#################################
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis = 0)

dz1 = np.dot(dy, W2.T)
da1 = dz1 * (1 - z1) * z1

grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(da1, axis = 0)

return grads
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$



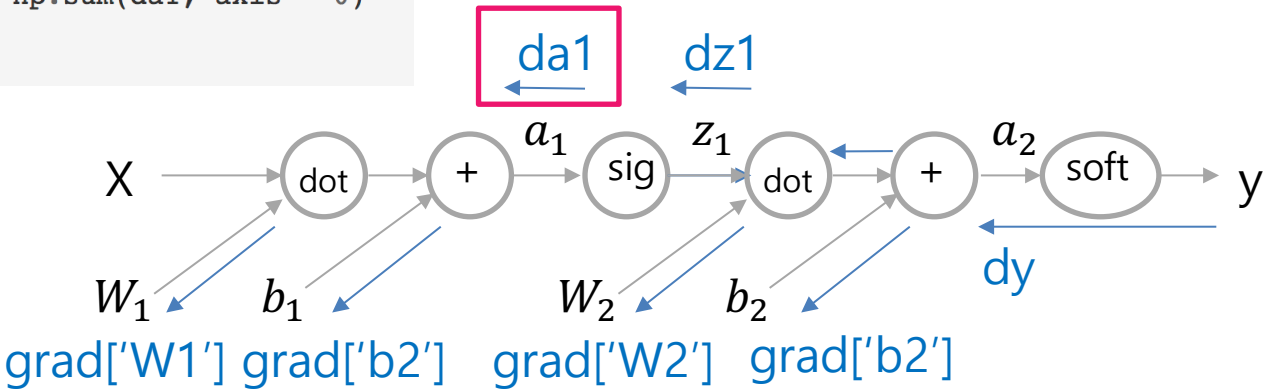
숙제 : 구자현 님 (1등)

모두의연구소

```
# backward
dy = (y - t) / batch_num
###################################
######### Write your codes #######
###################################
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis = 0)

dz1 = np.dot(dy, W2.T)
da1 = dz1 * (1 - z1) * z1

grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(da1, axis = 0)

return grads
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

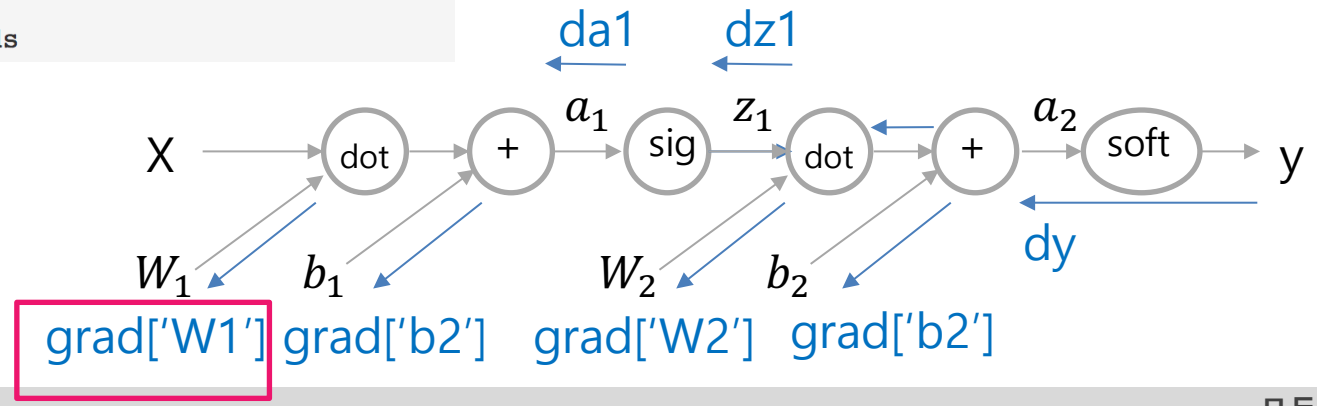


숙제 : 구자현 님 (1등)

```
# backward
dy = (y - t) / batch_num
###################################
######### Write your codes #######
###################################
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis = 0)

dz1 = np.dot(dy, W2.T)
da1 = dz1 * (1 - z1) * z1

grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(da1, axis = 0)

return grads
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

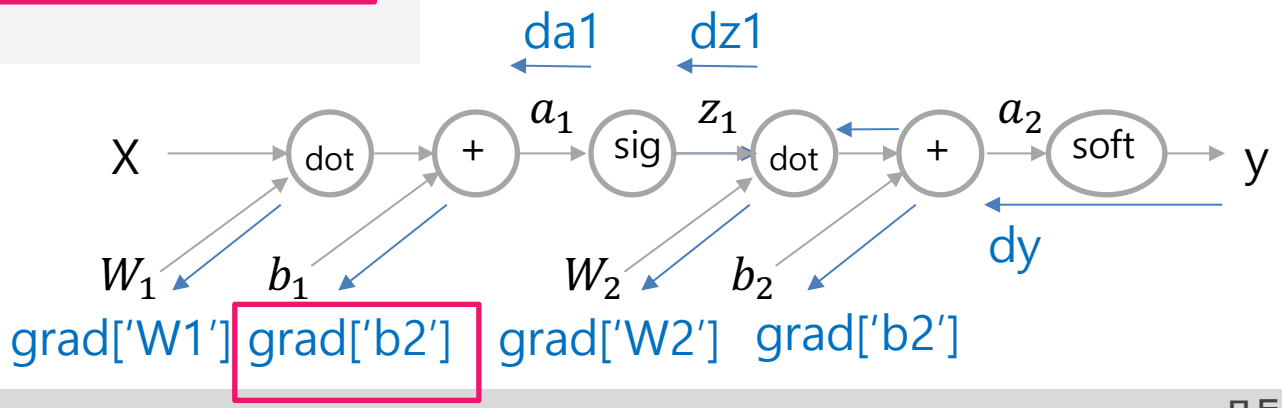


숙제 : 구자현 님 (1등)

```
# backward
dy = (y - t) / batch_num
###################################
######### Write your codes #######
###################################
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis = 0)

dz1 = np.dot(dy, W2.T)
da1 = dz1 * (1 - z1) * z1

grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(da1, axis = 0)

return grads
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$



```
# forward
a1 = np.dot(x, W1) + b1
z1 = sigmoid(a1)
a2 = np.dot(z1, W2) + b2
z2 = relu(a2)
a3 = np.dot(z2, W3) + b3
y = softmax(a3)

# backward
dy = (y - t) / batch_num

#################################
######## Write your codes #######
#################################

grads['W3'] = z2.T @ dy
grads['b3'] = np.sum(dy, axis=0)

dz2 = dy @ W3.T
da2_mask = (a2<=0) #mask for filte
da2 = dz2
da2[da2_mask] = 0 #filter the elem

grads['W2'] = z1.T @ da2
grads['b2'] = np.sum(da2, axis=0)

dz1 = da2 @ W2.T
da1 = dz1 * z1 * (1 - z1)
grads['W1'] = x.T @ da1
grads['b1'] = np.sum(da1, axis=0)


return grads
```

숙제 : 이동현 님

모두의연구소

- ReLU 적용
- @ Operator (Only Python 3.5? >)
- 3 layers
- Initialization Test

```
train acc, test acc | 99.500, 97.3000
cost: 0.0066
```

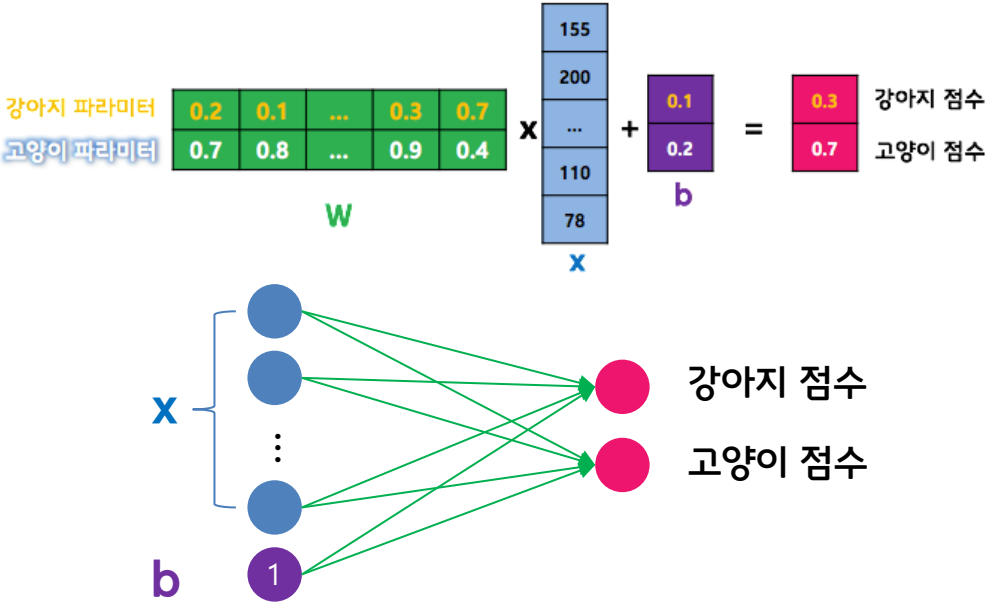
모두의연구소

Neural Networks

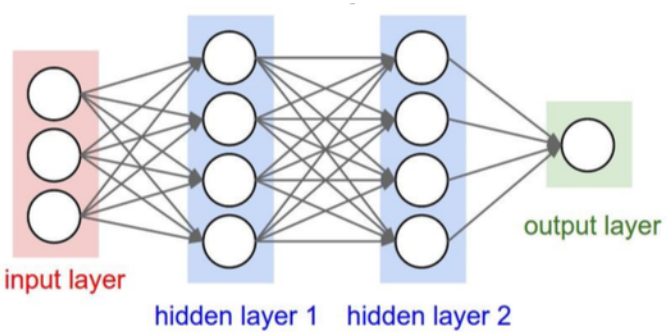
모두의연구소
박은수 Research Director

Neural Networks

모두의연구소



Before



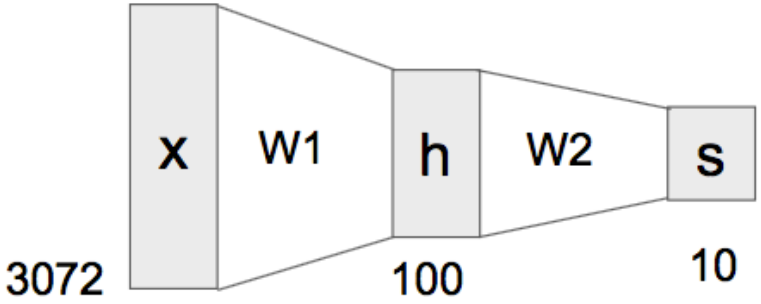
레이어를 쌓아서 더 깊게

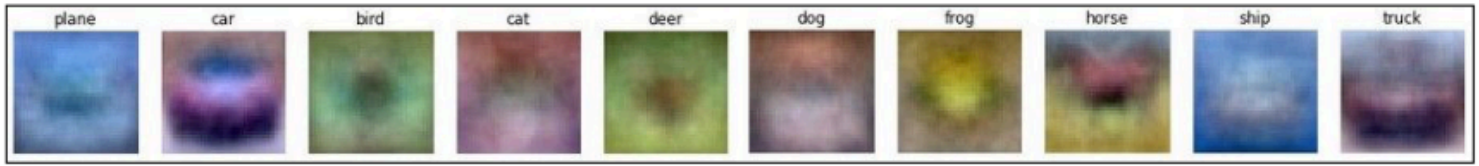
Now

Neural Networks

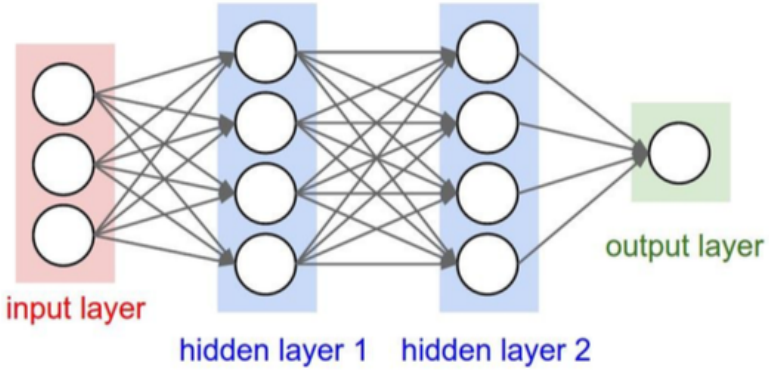
모두의연구소

- Cifar-10 (영상크기 : 32x32x3, 10개의 클래스) 데이터에서의 첫번째 레이어의 시각화





Neural Networks



$$output\ layer = W_2(W_1\mathrm{x}) = W_2W_1\mathrm{x} = \mathrm{Wx}$$

레이어를 하나 쌓는 것과 같음 → 비선형 Activation function이 필요

Neural Networks

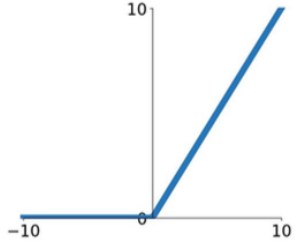
모두의연구소

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

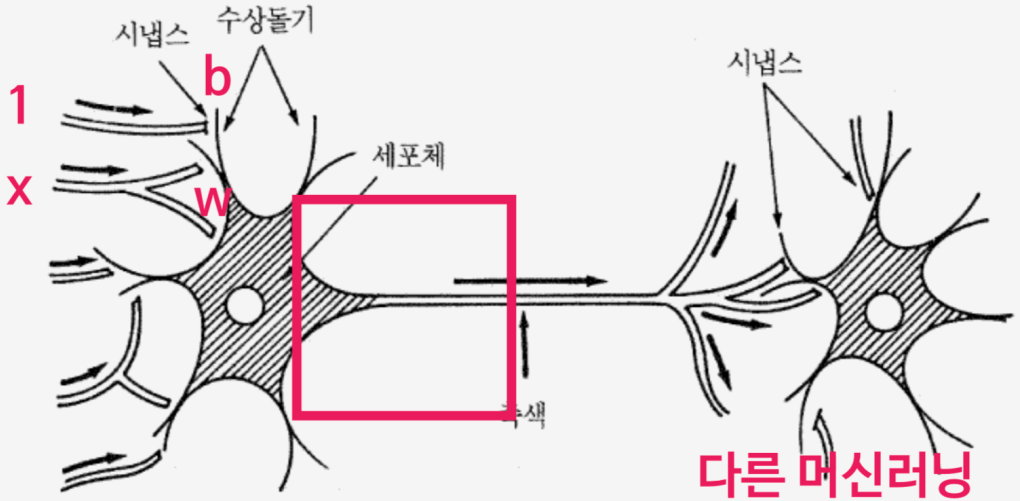
or 3-layer Neural Network

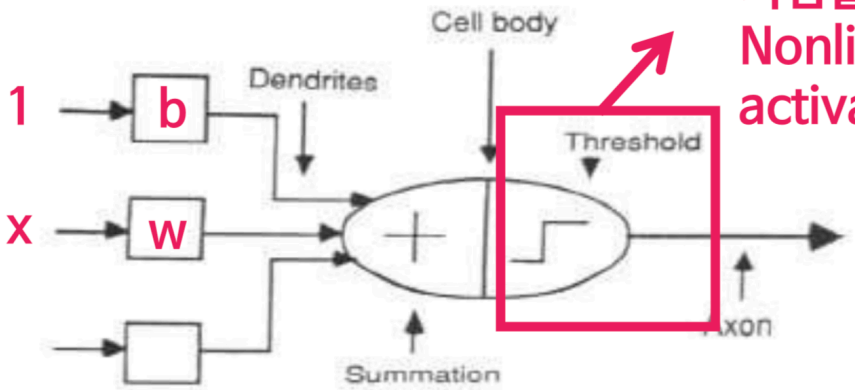
$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$



ReLU
(Rectified Linear Unit)

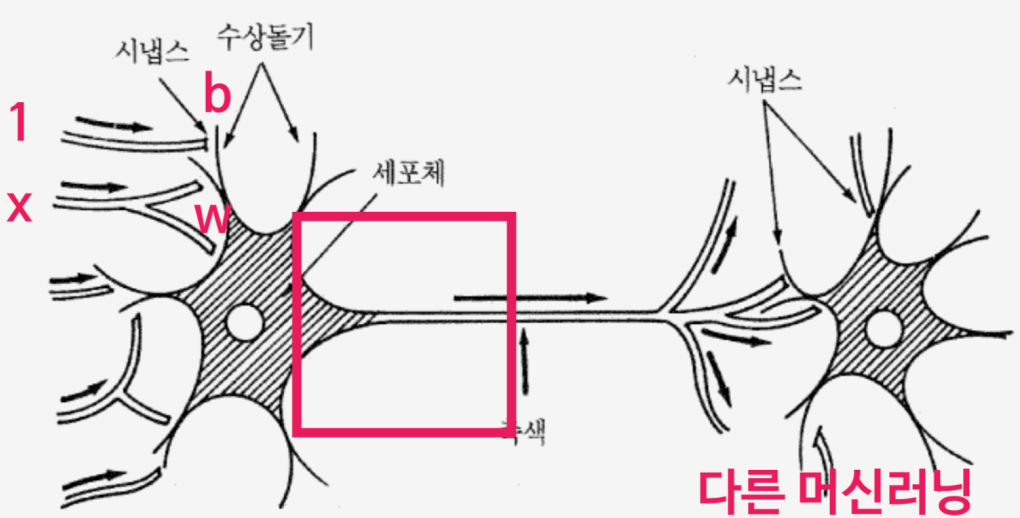
뉴런과 인공뉴런

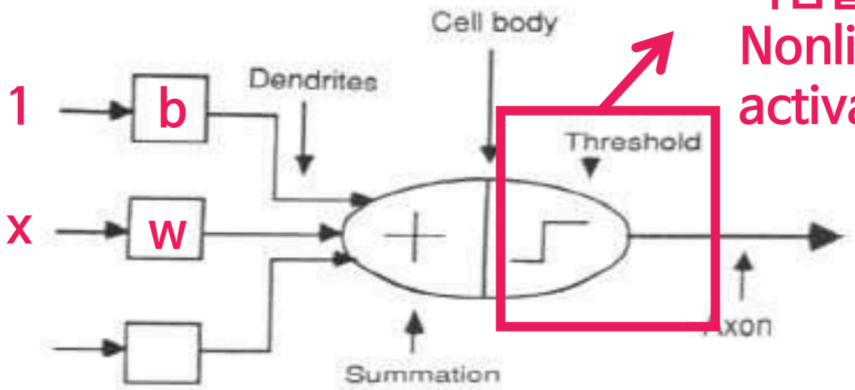




다른 머신러닝 기법들과의 차이점 1: Nonlinear(복잡한) activation function

뉴런과 인공뉴런





다른 머신러닝 기법들과의 차이점 1: Nonlinear(복잡한) activation function

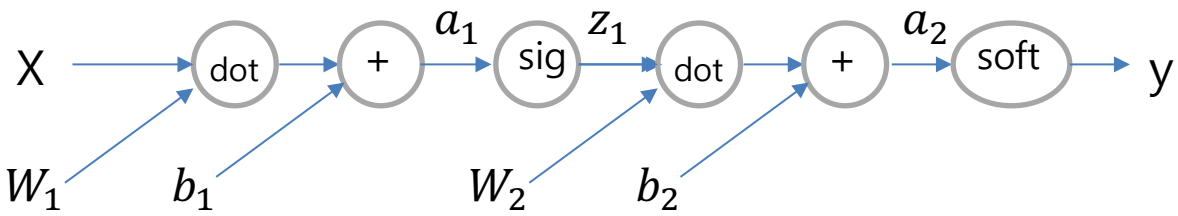
완전히 단순화 시킨 것이니 주의하세요

2레이어 네트워크의 구현 예

모두의연구소

- **Forward**
  - Activation function : Sigmoid
  - Loss function : Softmax loss

```
# forward
a1 = np.dot(x, W1) + b1
z1 = sigmoid(a1)
a2 = np.dot(z1, W2) + b2
y = softmax(a2)
```



2레이어 네트워크의 구현 예

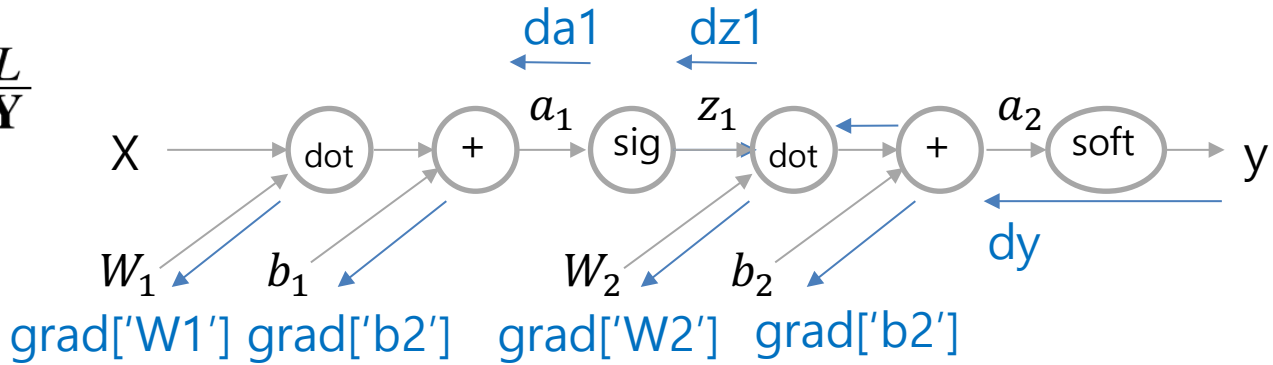
모두의연구소

- **Backward**
  - Activation function : Sigmoid
  - Loss function : Softmax loss

```
# backward
dy = (y - t) / batch_num
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis=0)

dz1 = np.dot(dy, W2.T)
da1 = z1*(1-z1)*dz1
grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(dz1, axis=0)
```

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

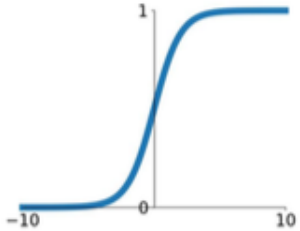


Activation function

- **ReLU vs. Sigmoid**

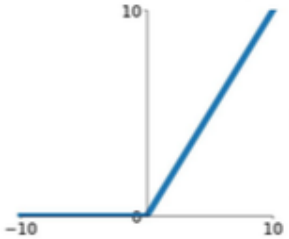
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



ReLU

$$\max(0, x)$$



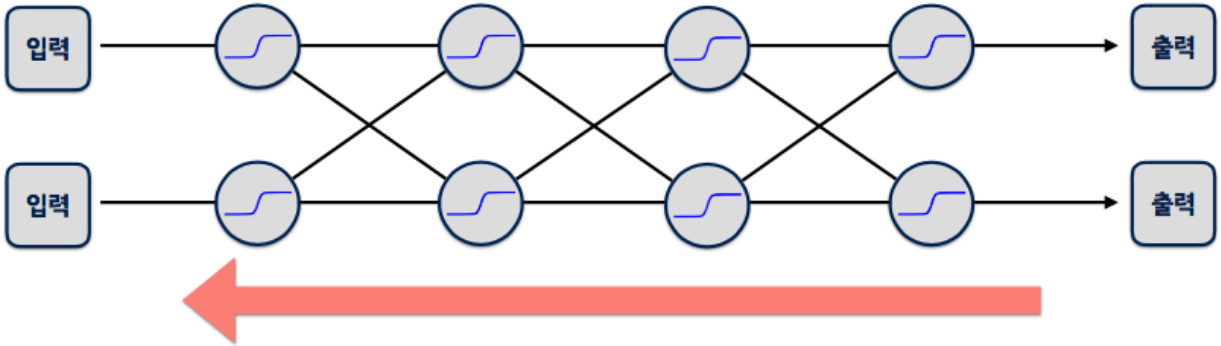
뉴럴넷의 학습방법 Back propagation

(사실 별거 없고 그냥 "뒤로 전달")

모두의연구소

뭐를 전달하는가?

현재 내가 틀린정도를 '미분(기울기)' 한 거



미분하고, 곱하고, 더하고를 역방향으로 반복하며 업데이트한다.

근데 문제는?

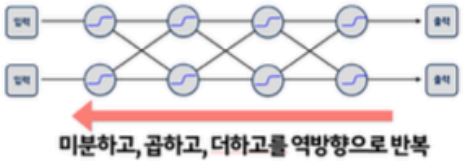
모두의연구소

우리가 activation 함수로 sigmoid  를 썼다는 것

여기의 미분(기울기)는 뭐라도 있다. 다행

```
# backward
dy = (y - t) / batch_num
grads['W2'] = np.dot(z1.T, dy)
grads['b2'] = np.sum(dy, axis=0)

dz1 = np.dot(dy, W2.T)
da1 = z1*(1-z1)*dz1
grads['W1'] = np.dot(x.T, da1)
grads['b1'] = np.sum(dz1, axis=0)
```

근데 여기는 기울기 0.. 이런거 중간에 곱하면 **뭔가 뒤로 전달할게 없다?!**

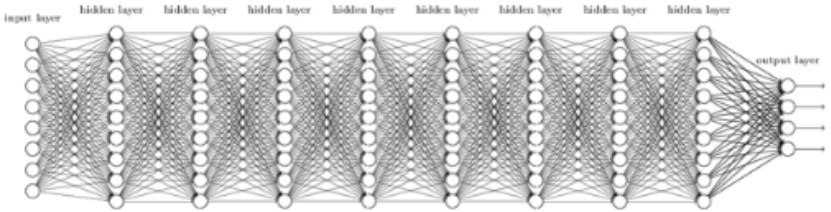


그런 상황에서 이걸 반복하면??????

Vanishing gradient 현상 : 레이어가 깊을 수록 업데이트가 사라져간다. 그래서 fitting이 잘 안됨(underfitting)

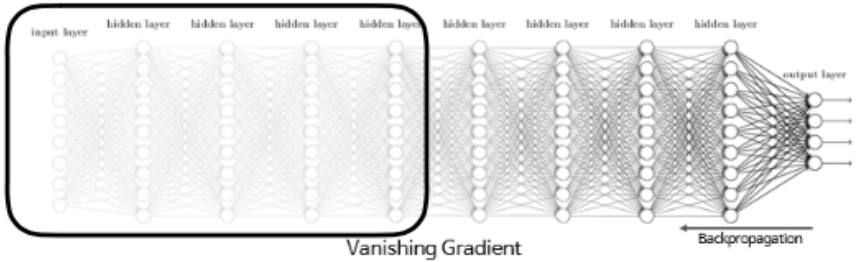
모두의연구소

끝줄학생 ← 줄.. 줄.. 줄.. 줄좀.. 줄좀맞추자 → 교장샘



Deep Neural Network

학습이 잘 안됨

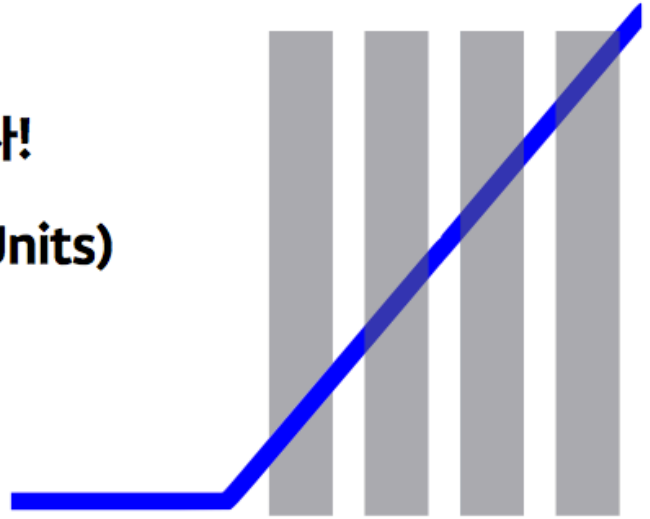


Vanishing Gradient

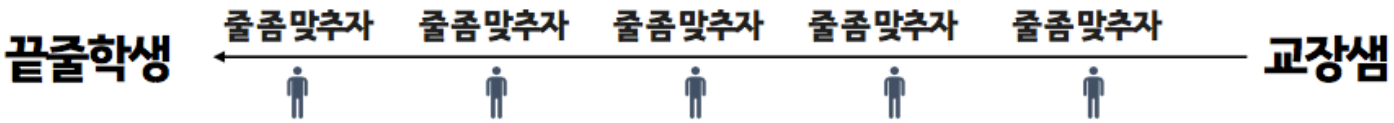
Reference : "자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다", 발표자료, 하용호

사그라드는 sigmoid대신
죽지않는 activation func을 쓰자!

→ **ReLU** (Rectified Linear Units)



이녀석은 양의 구간에서 전부 미분 값(1)이 있다!



끝 줄 학생까지 이야기가 전달이 잘 되고 위치를 고친다!

모두의연구소

Convolutional Neural Networks

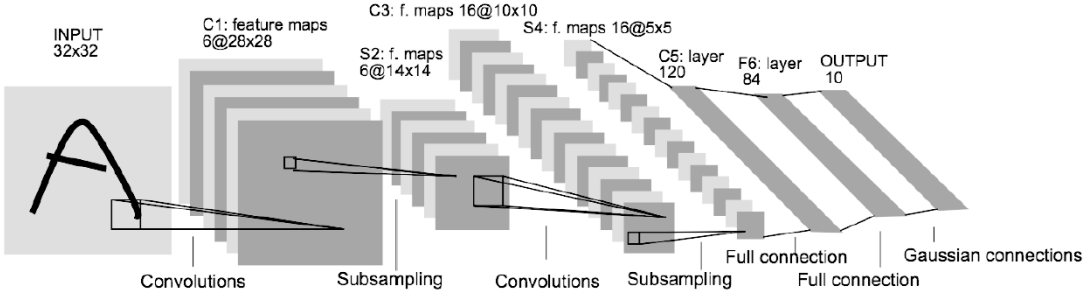
모두의연구소
박은수 책임연구원

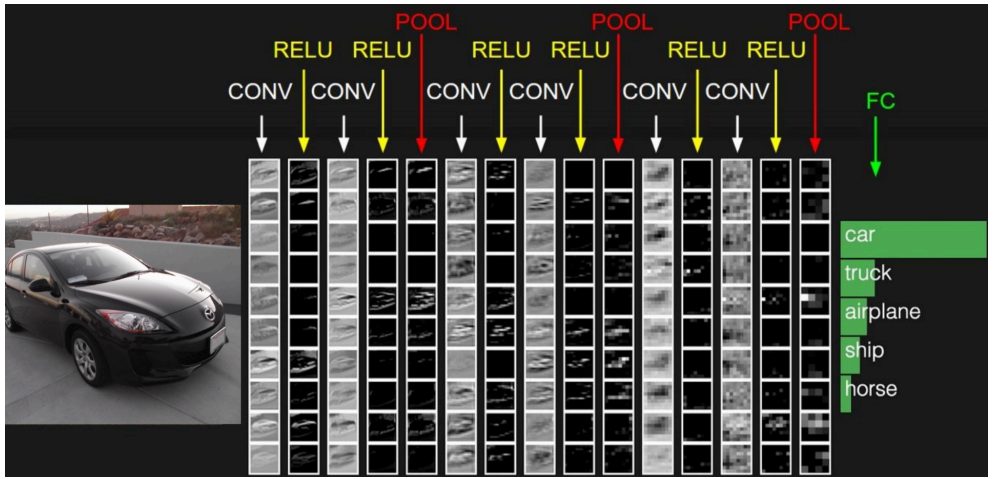
모두의연구소

큰 그림

큰 그림 : 구조

- LeNet (1998년)





전형적인 구조

CONV POOL FC

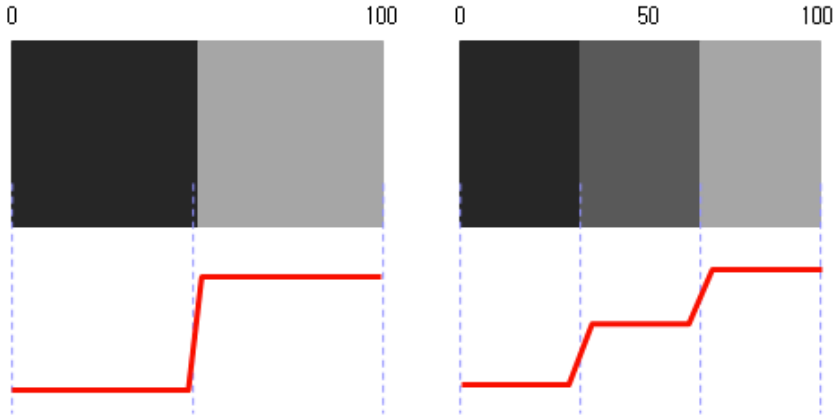
큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다

-1	0	1
-1	0	1
-1	0	1

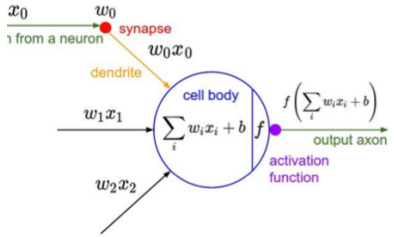
수직 에지를 찾는
컨볼루션 필터



큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0

100

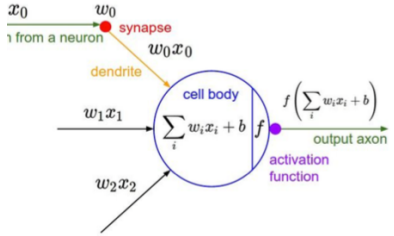
-1	0	1
-1	0	1
-1	0	1

0			

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

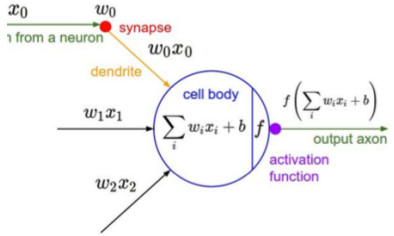
	-1	0	1		
	-1	0	1		
	-1	0	1		

0	300		

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

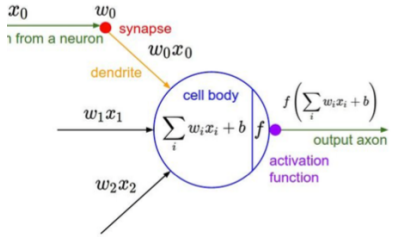
			-1	0	1
			-1	0	1
			-1	0	1

0	300	300	0

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

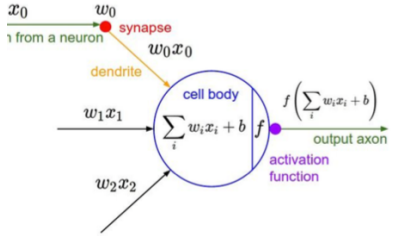
-1	0	1			
-1	0	1			
-1	0	1			

0	300	300	0
0			

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

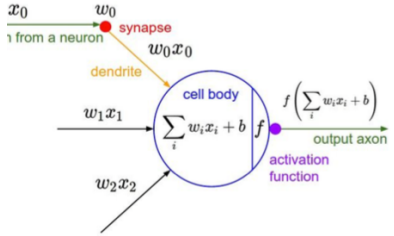
	-1	0	1		
	-1	0	1		
	-1	0	1		

0	300	300	0
0	300		

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0

100

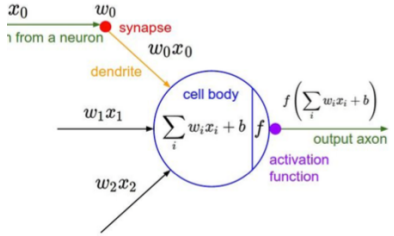
		-1	0	1	
		-1	0	1	
		-1	0	1	

0	300	300	0
0	300	300	

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

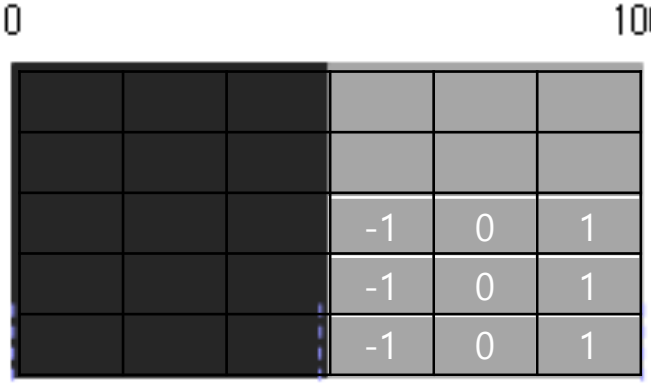
			-1	0	1
			-1	0	1
			-1	0	1

0	300	300	0
0	300	300	0
0	300	300	0

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



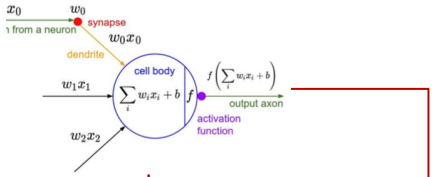
0	300	300	0
0	300	300	0
0	300	300	0

수직 에지 검출

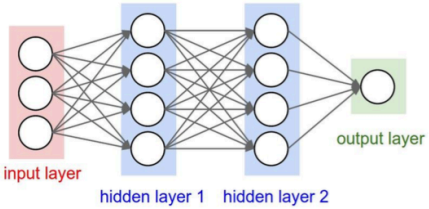
다양한 필터로 다양한
특징을 추출할 수 있다

컨볼루셔널 뉴럴넷 (Convolutional Neural Network)

- 필터를 학습하는 구조다

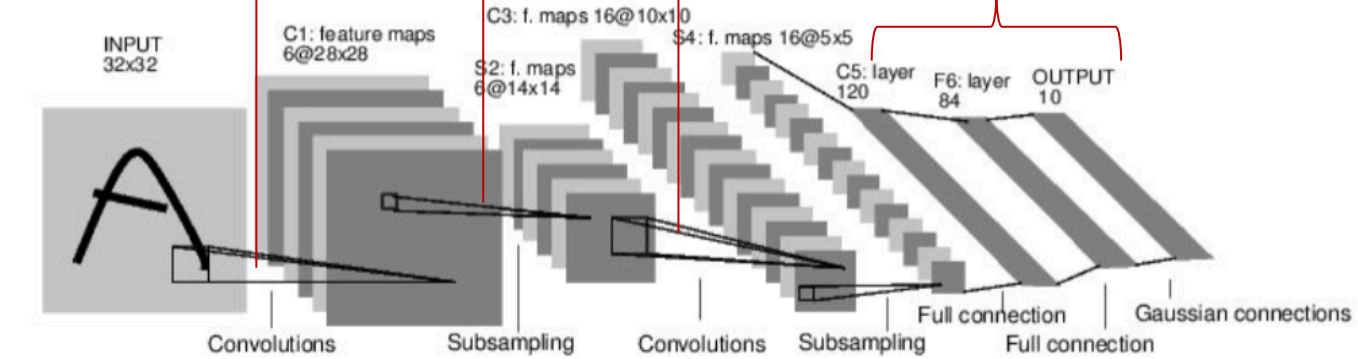


필터가 학습됨



똑같은 구조

[LeCun et al., 1998]

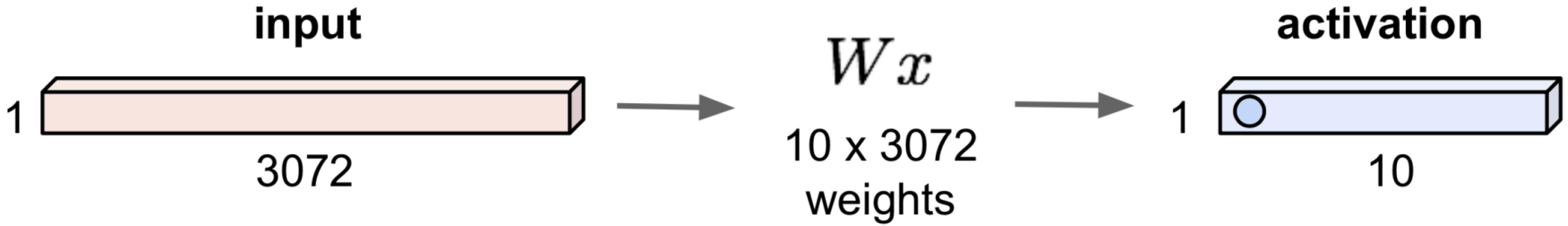


모두의연구소

이제부터
자세히 살펴봅시다

Fully Connected Layer

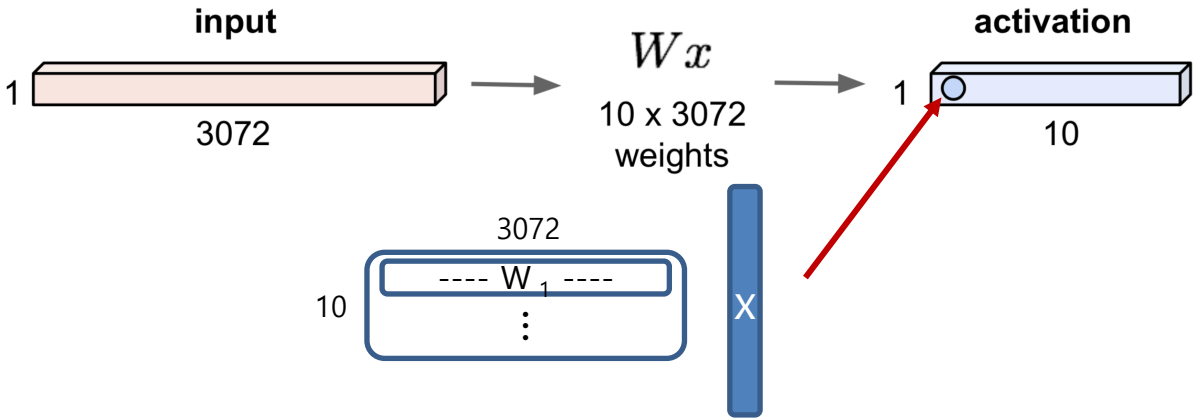
32x32x3 image -> stretch to 3072 x 1



지금까지 살펴봤펴봤던 것 입니다

Fully Connected Layer

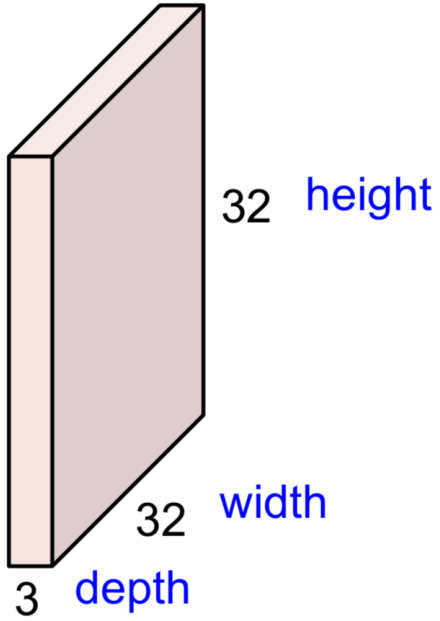
32x32x3 image -> stretch to 3072 x 1



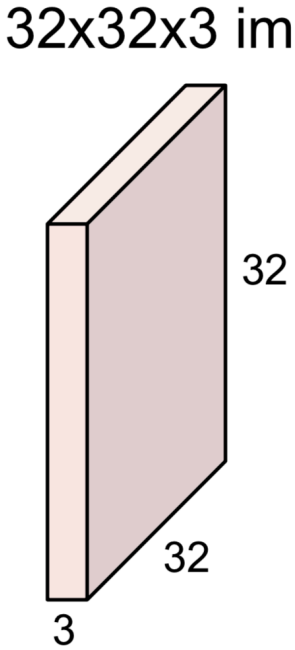
1) 전체 영상 각 픽셀에 W_1의 weighted sum으로 계산한 score
2) Activation 적용

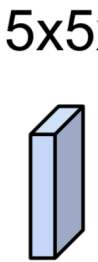
Convolution Layer

32x32x3 image -> preserve spatial structure



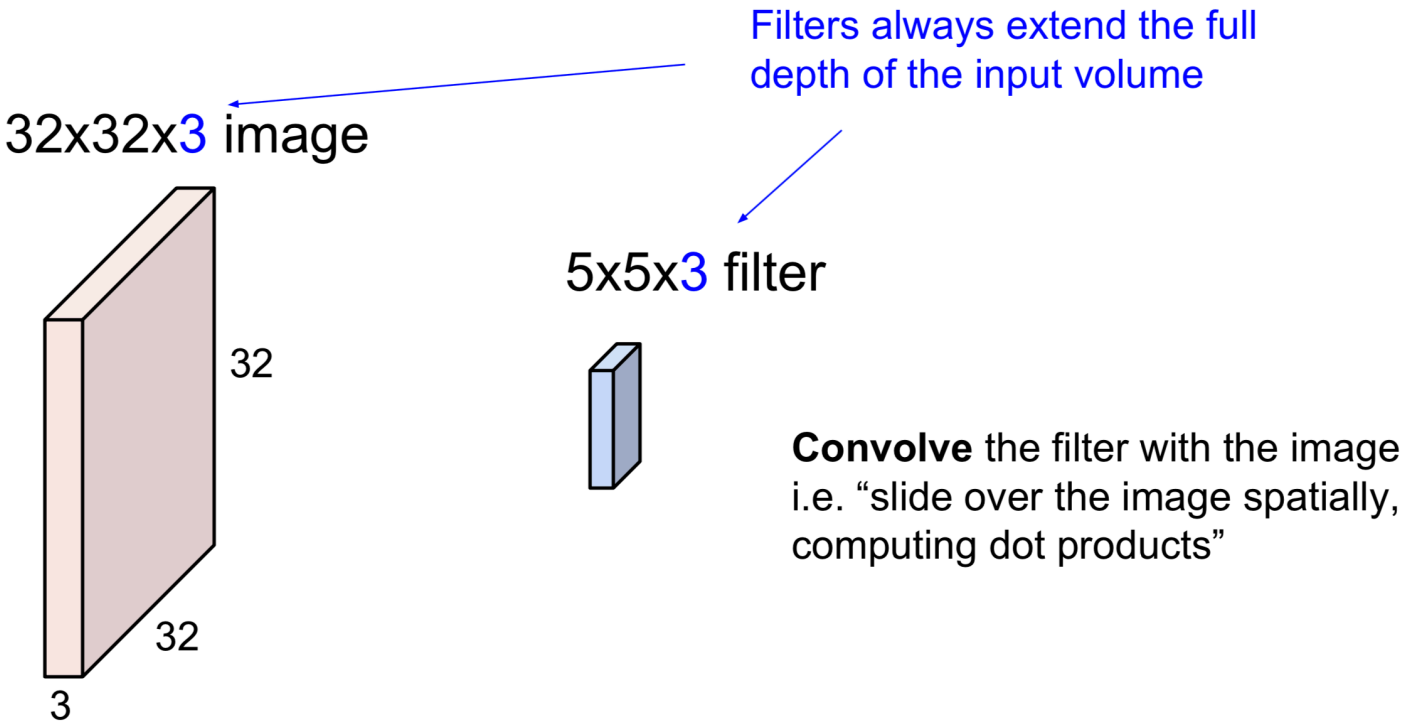
Convolution Layer



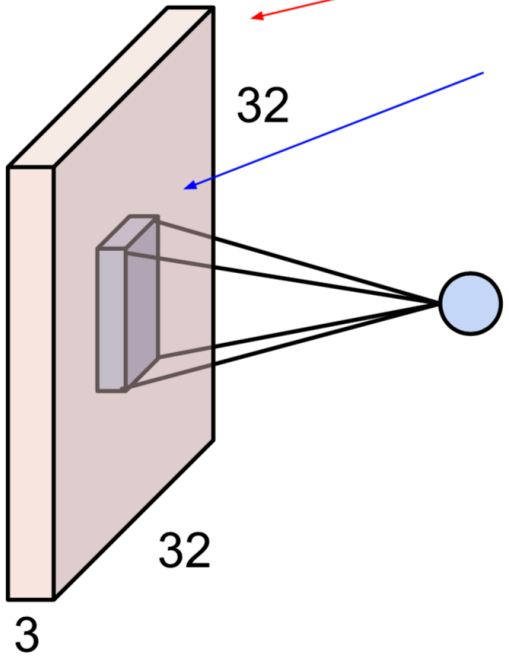


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer



Convolution Layer

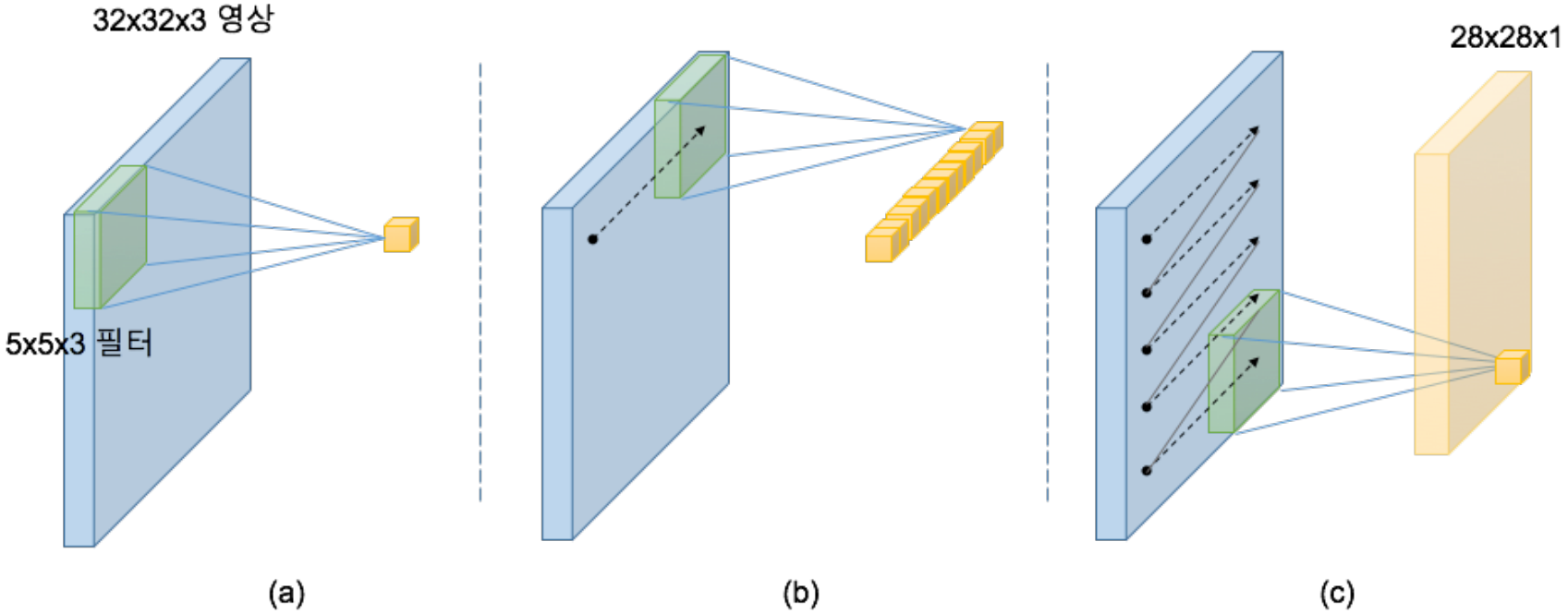


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

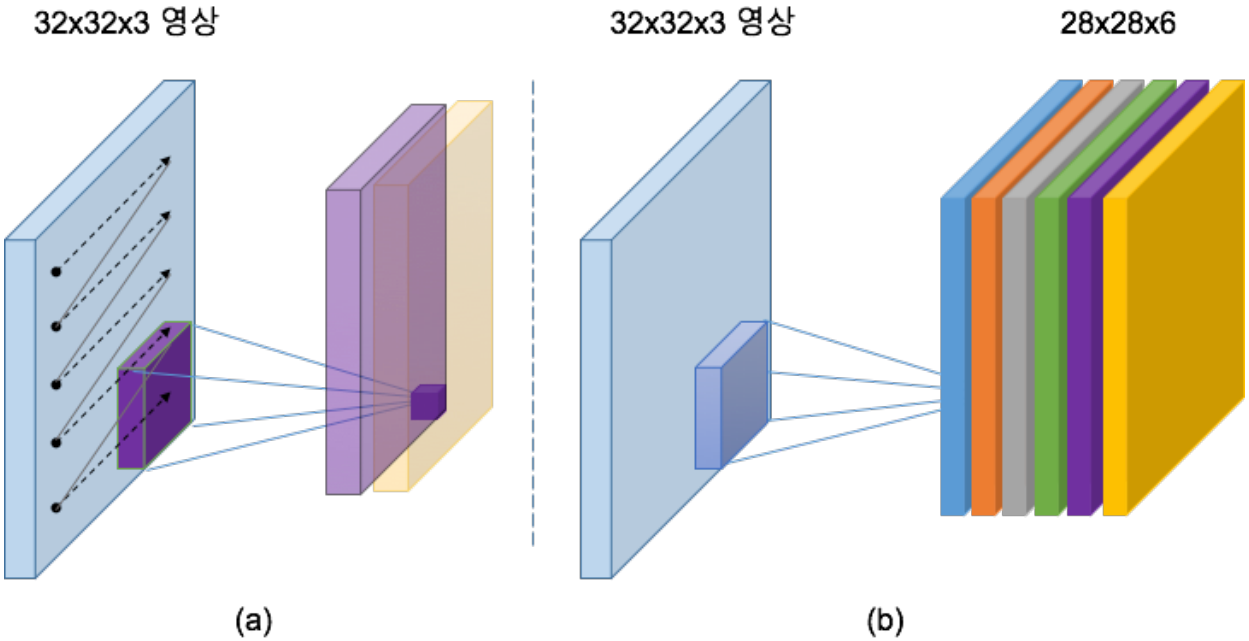
모두의연구소



Convolution Layer

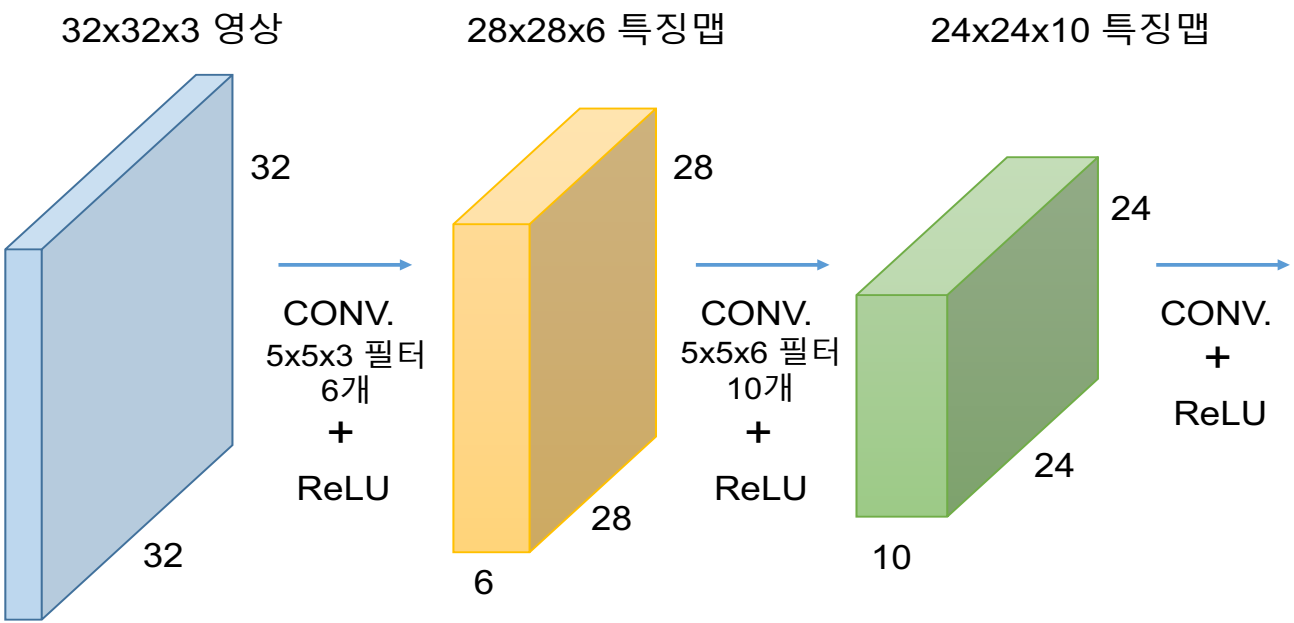
모두의연구소

똑같은 크기의 필터 6개를 더 만들어 봅시다



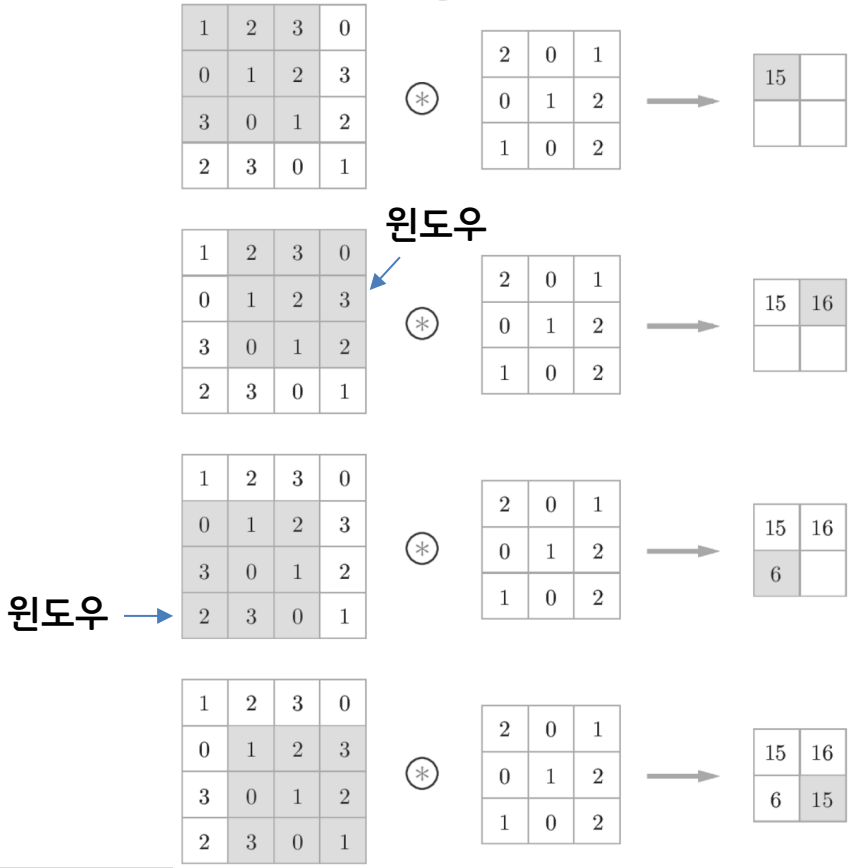
Convolution Layer

컨볼루션 네트워크는 활성화 함수를 포함한 컨볼루션 레이어의 연결 입니다



Convolution Layer

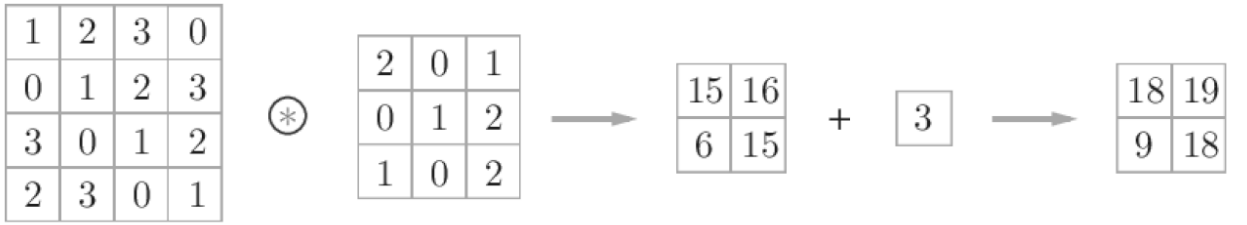
- 컨볼루션 연산
  - 1) 윈도우를 일정 간격으로 이동해가며 입력 데이터에 적용
  - 2) 입력과 필터에 대응하는 원소끼리 곱한 후 그 총합을 함 (단일 곱셈-누산 (fused multiply-add, FMA)
  - Bias 파라미터 존재함



Convolution Layer

모두의연구소

- **컨볼루션 연산**

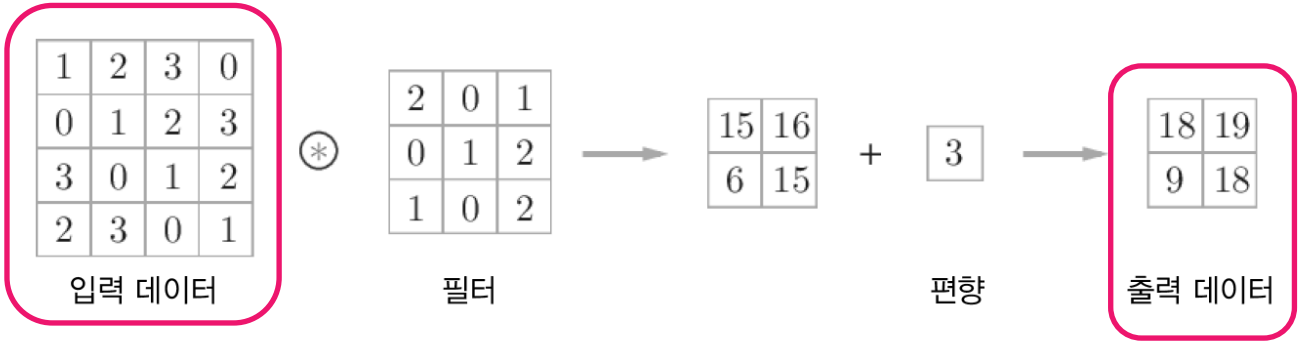


Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolution Layer

모두의연구소

- **컨볼루션 연산**



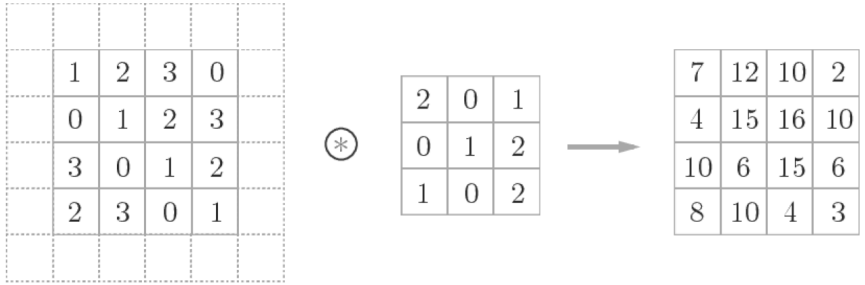
크기가 줄어드는 군요

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolution Layer

모두의연구소

- 패딩 (padding)
  - 컨볼루션 연산의 패딩 처리 : 입력 데이터 주위에 0을 채운다 (패딩은 점선으로 표시했으며 그 안의 값 '0'은 생략함)

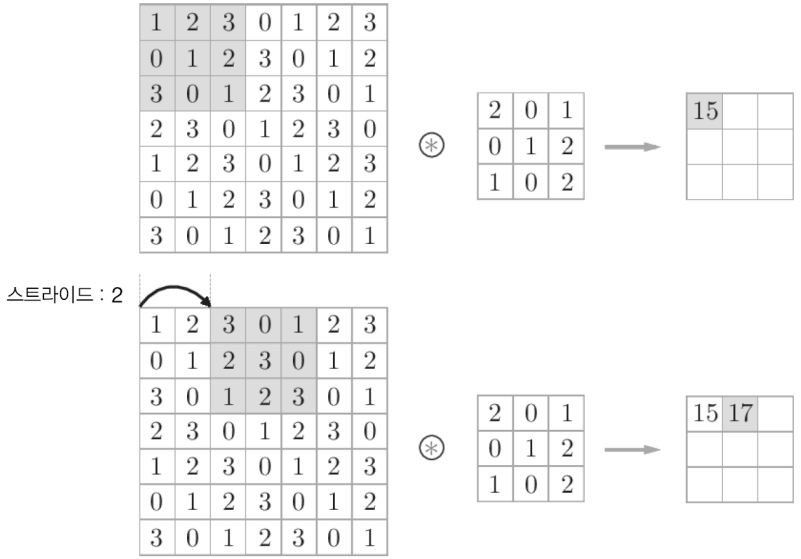


  - 패딩은 출력(특징맵(feature map))의 크기를 유지 시키고자 할때 주로 사용함

Convolution Layer

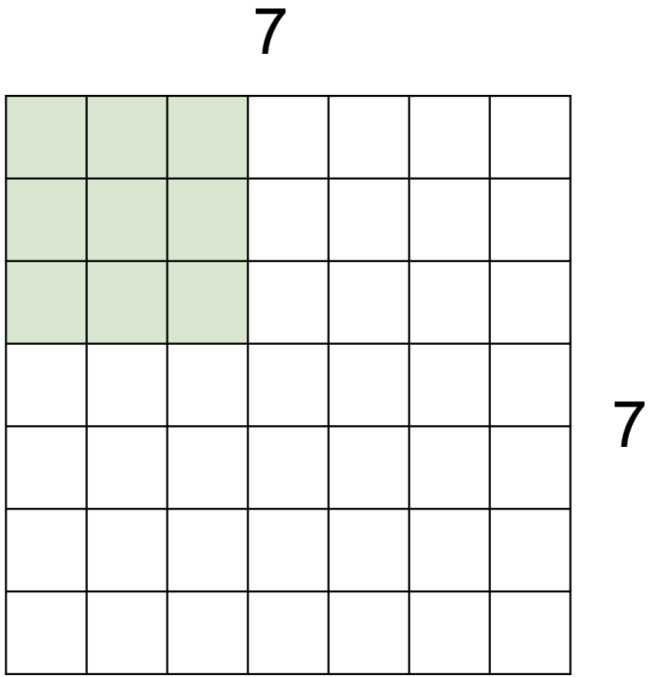
- 스트라이드 (stride)
  - 필터 적용하는 위치의 간격
    - 스트라이드를 2로 하면 필터를 적용하는 윈도우가 두 칸씩 이동함

- 스트라이드를 2로 하니 출력이 3x3이 됨



Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

특징맵의 크기변화

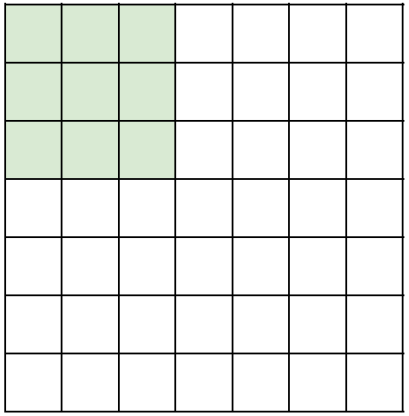


7x7 input (spatially)
assume 3x3 filter

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

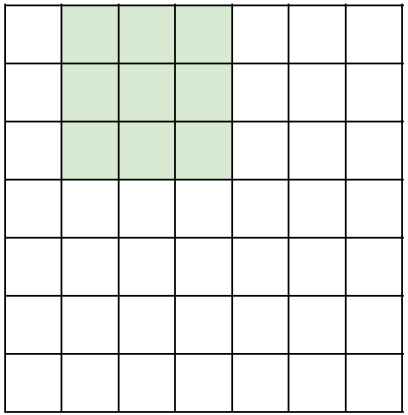


7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



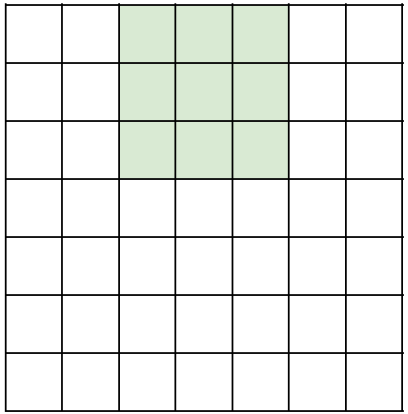
7x7 input
assume 3x3 connectivity, stride 1

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



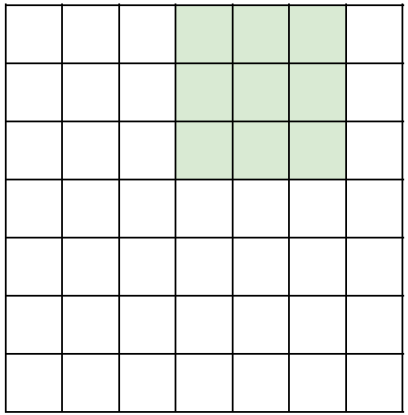
7x7 input
assume 3x3 connectivity, stride 1

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



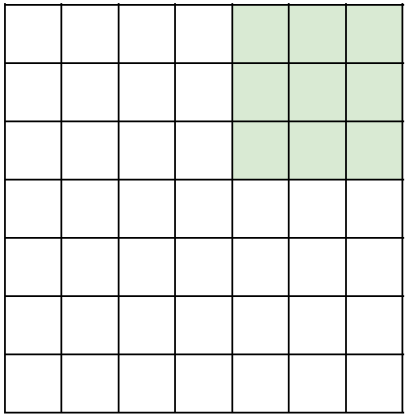
7x7 input
assume 3x3 connectivity, stride 1

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

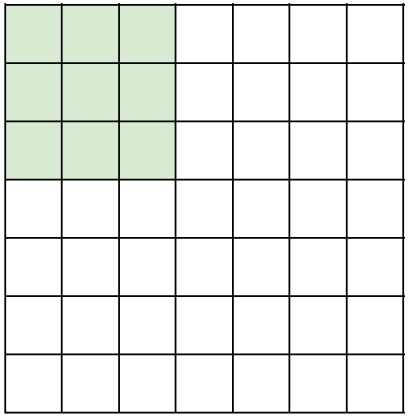


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

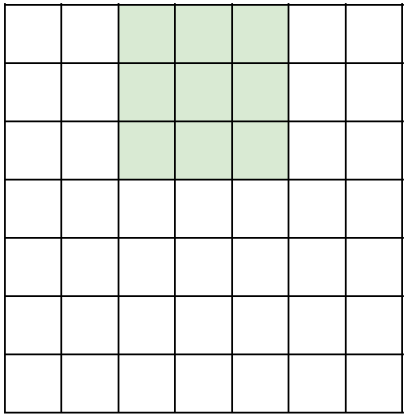
what about stride 2?

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

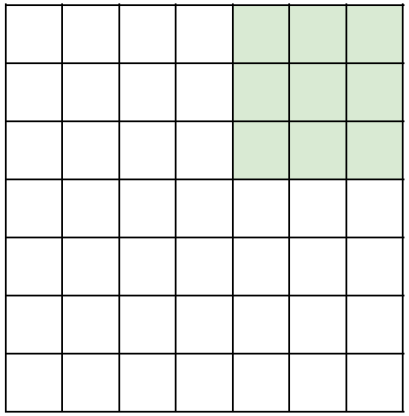
what about stride 2?

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

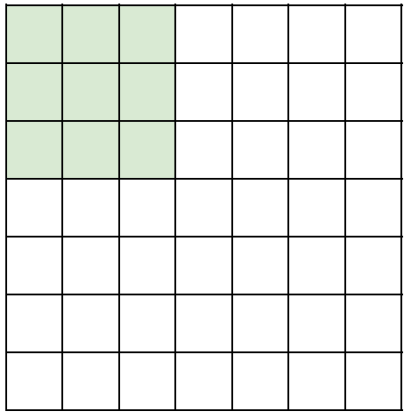
what about stride 2?
=> **3x3 output**

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



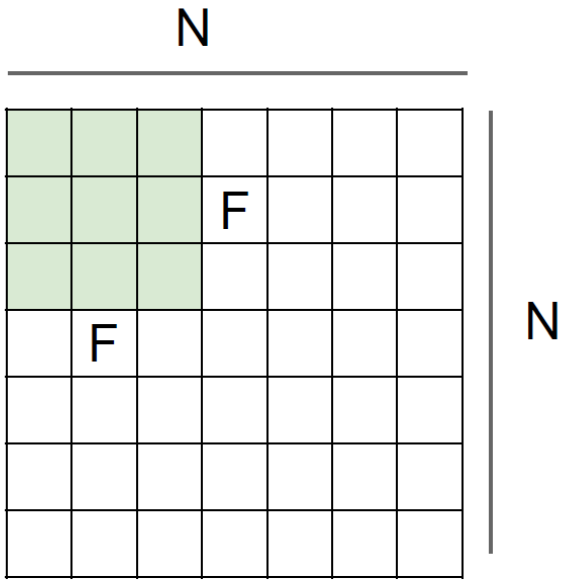
7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?
=> **3x3 output**

what about stride 3? Cannot.

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화



Output size:
(N - F) / stride + 1

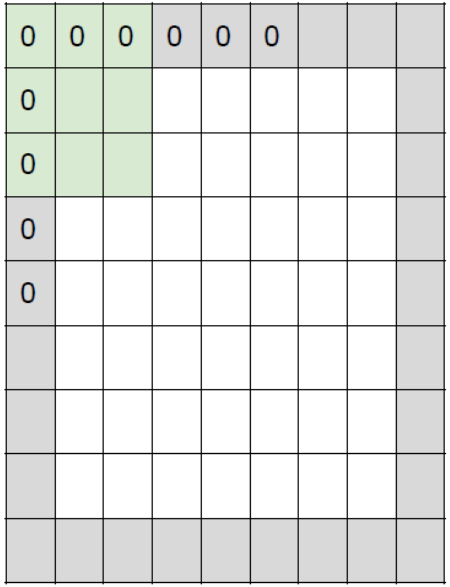
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = … :\

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

In practice: Common to zero pad the border



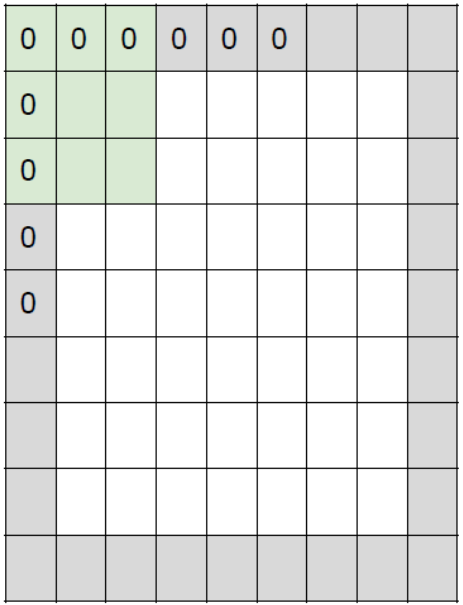
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

특징맵의 크기변화

모두의연구소

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

- 입력크기 : (*H*, *W*)
- 필터크기 : (*FH*, *FW*)
- 출력크기 : (*OH*, *OW*)
- 패딩 : *P*
- 스트라이드 : *S*

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{\mathrm{W} + 2P - FW}{S} + 1$$

특징맵의 크기변화

모두의연구소

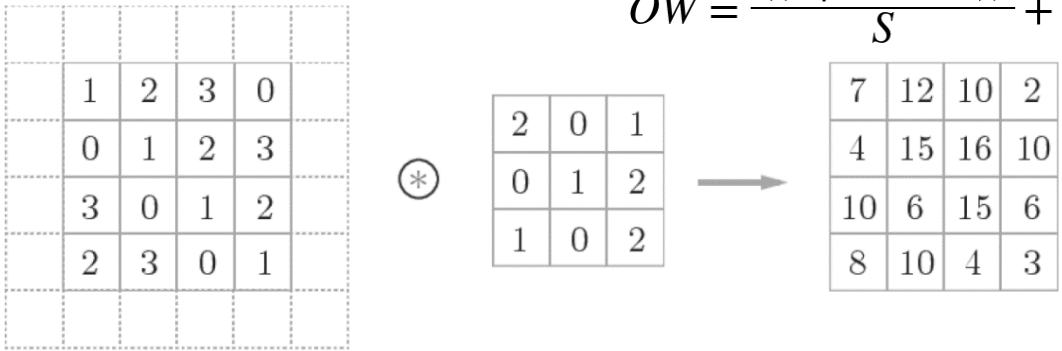
- **출력크기 계산해 보기**

입력 : (4,4), 패딩 : 1 , 스트라이드 : 1, 필터: (3,3)

$$OH = \frac{4 + 2 \cdot 1 - 3}{1} + 1 = 4$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$



(4, 4) 입력 데이터(패딩 : 1)

(3, 3) 필터

(4, 4) 출력 데이터

특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

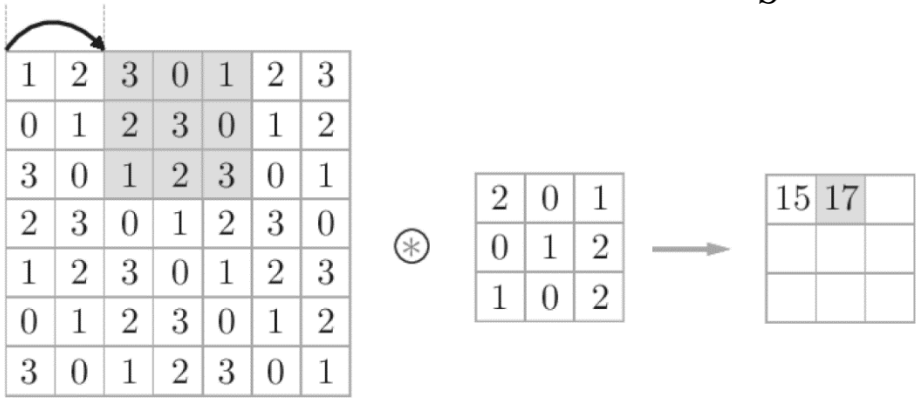
입력 : (7,7), 패딩 : 0 , 스트라이드 : 2, 필터: (3,3)

$$OH = \frac{7 + 2 \cdot 0 - 3}{2} + 1 = 3$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

스트라이드 : 2



특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

입력 : (28, 31), 패딩 : 2 , 스트라이드 : 3, 필터: (5, 5)

$$OH = ?$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = ?$$

$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$

특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

입력 : (28, 31), 패딩 : 2 , 스트라이드 : 3, 필터: (5, 5)

$$OH = \frac{28 + 2 \cdot 2 - 5}{3} + 1 = 10$$

$$OW = \frac{31 + 2 \cdot 2 - 5}{3} + 1 = 11$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

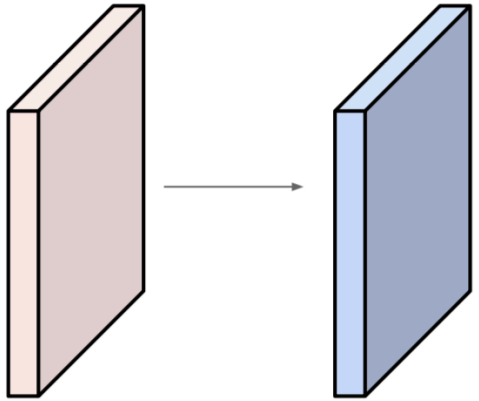
$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$

- ***OH, OW*가 정수가 아니면?**
  - 출력이 안나오는 것임. 오류를 내는 등의 대응 필요
  - 딥러닝 프레임웍은 가까운 정수로 내림 하는 등, 특별히 에러를 내지않고 진행되도록 구현되는 경우가 많음

특징맵의 크기변화

모두의연구소

Examples time:



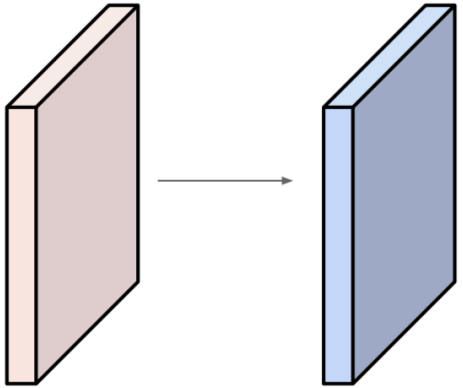
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

특징맵의 크기변화

모두의연구소

Examples time:



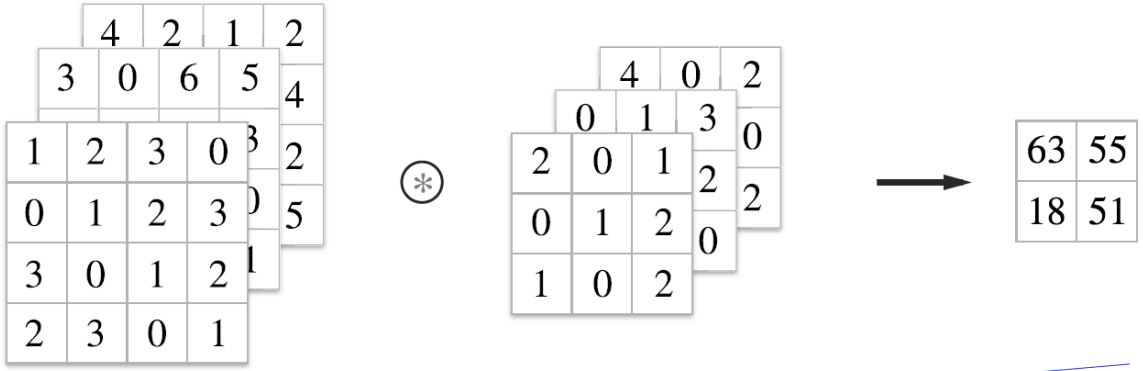
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

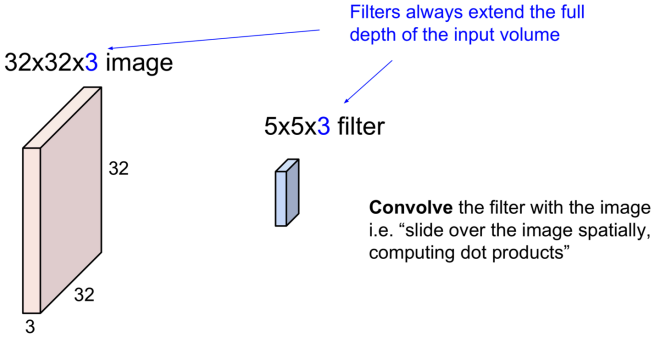
Convolution Layer

모두의연구소

- 필터의 채널 수는 입력되는 특징맵의 채널 수와 일치해야 함을 잊지 마세요

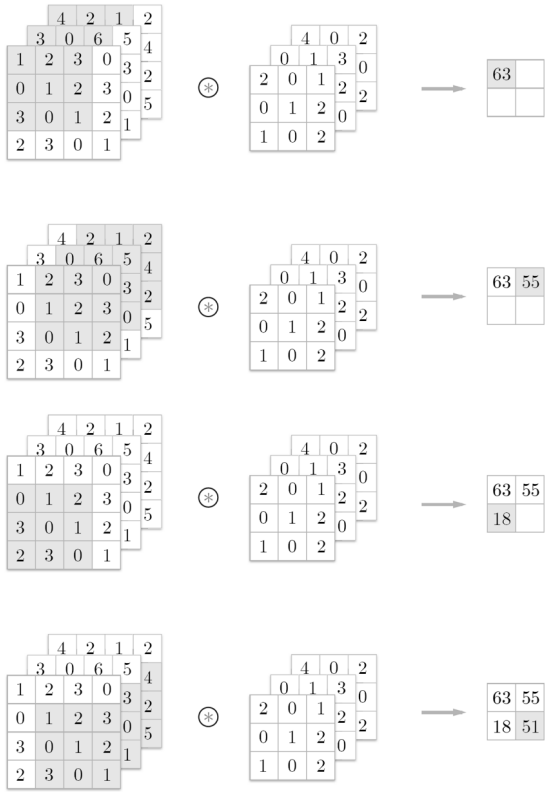


입력 데이터 필터



Convolution Layer

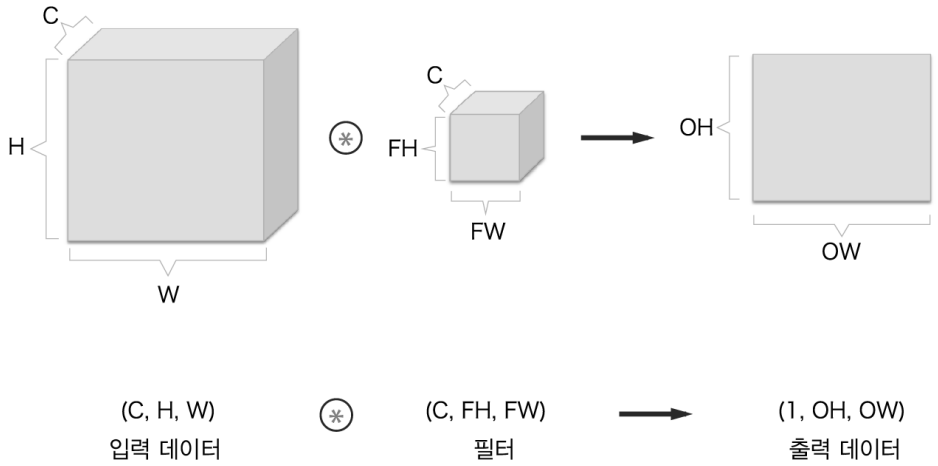
- 3차원 데이터의 컨볼루션
  - 주의할 점은 **입력데이터의 채널 수와 필터의 채널 수가 같아야** 한다는 점
  - 각 필터의 채널크기는 같아야 함



Convolution Layer

모두의연구소

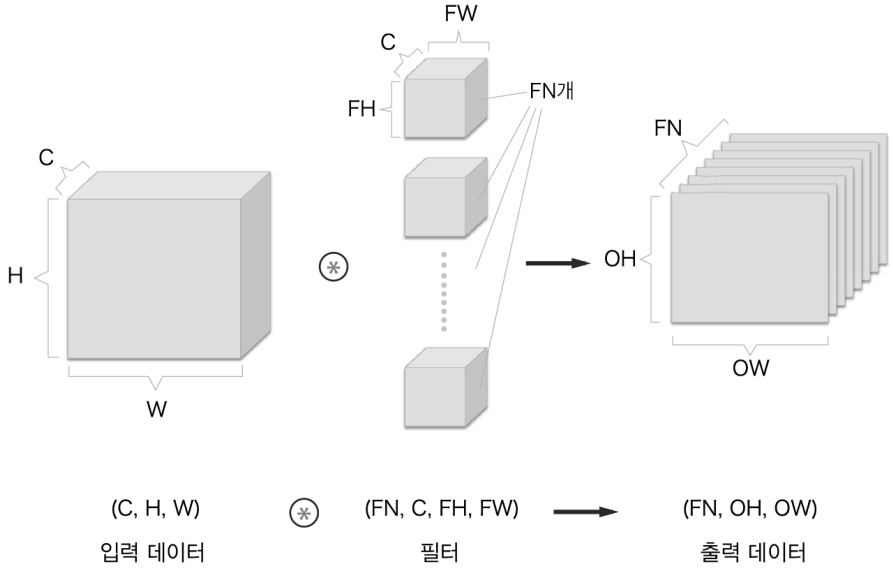
- 블록으로 생각하기



- 출력 데이터는 한장의 특징맵
- 한장이 아니라 여러장, 즉 다수의 채널을 내보내려면?

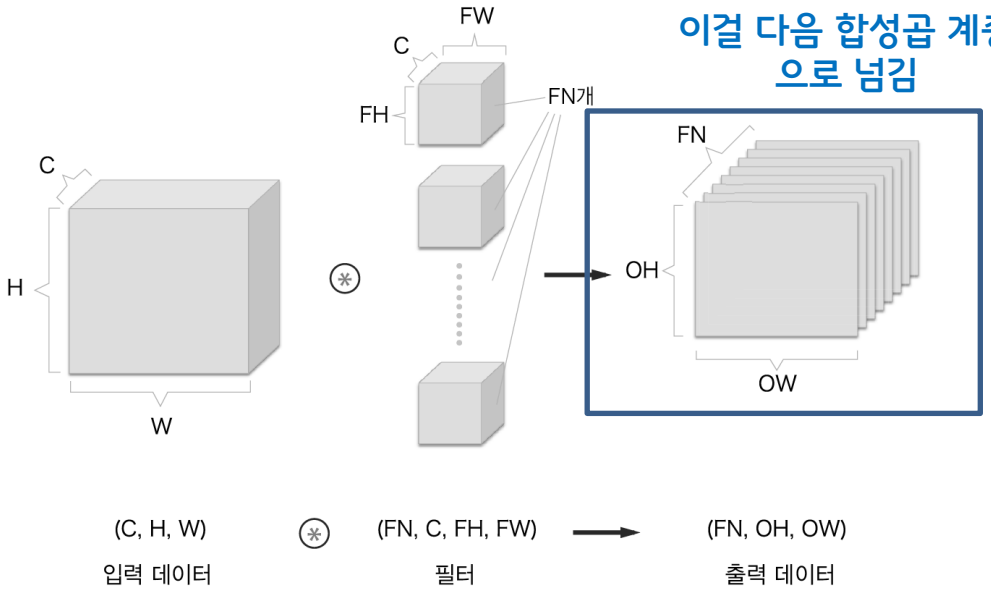
Convolution Layer

- 블록으로 생각하기
  - 한장이 아니라 여러장, 즉 다수의 채널을 내보내려면?
    - 필터를 여러개 사용하면 됨



Convolution Layer

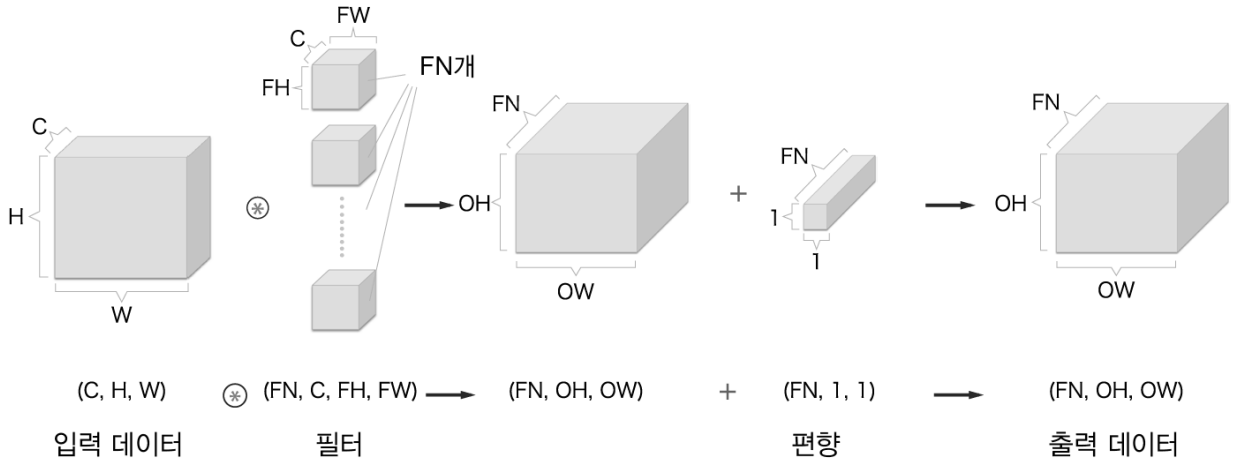
- **블록으로 생각하기**
  - 한장이 아니라 여러장, 즉 다수의 채널을 내보내려면?
    - **필터를 여러개 사용하면 됨**



Convolution Layer

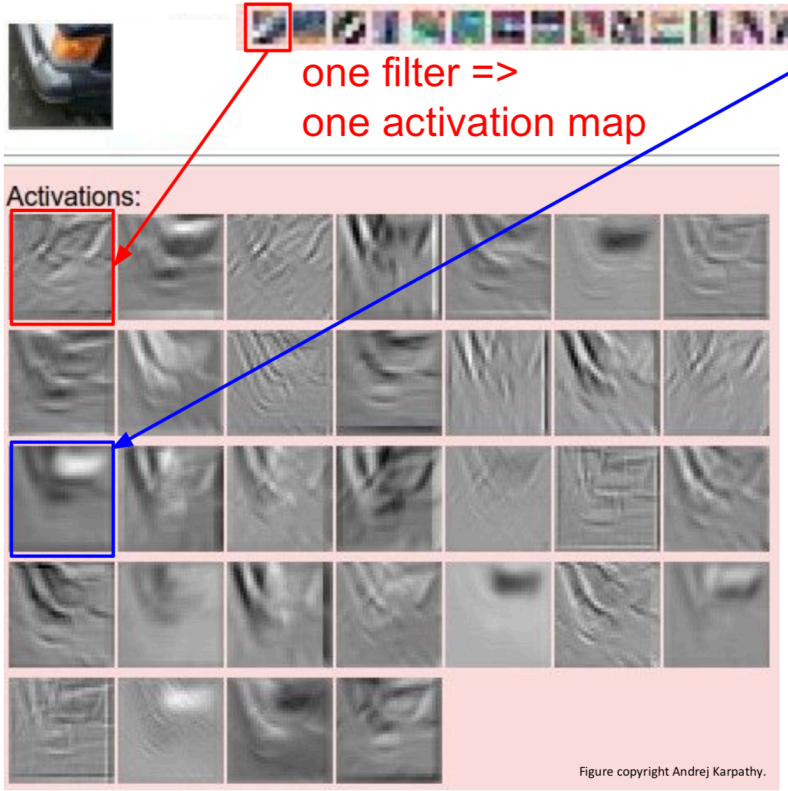
모두의연구소

- 편향 (bias)



- 필터마다 편향이 하나씩 존재

Convolution Layer



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

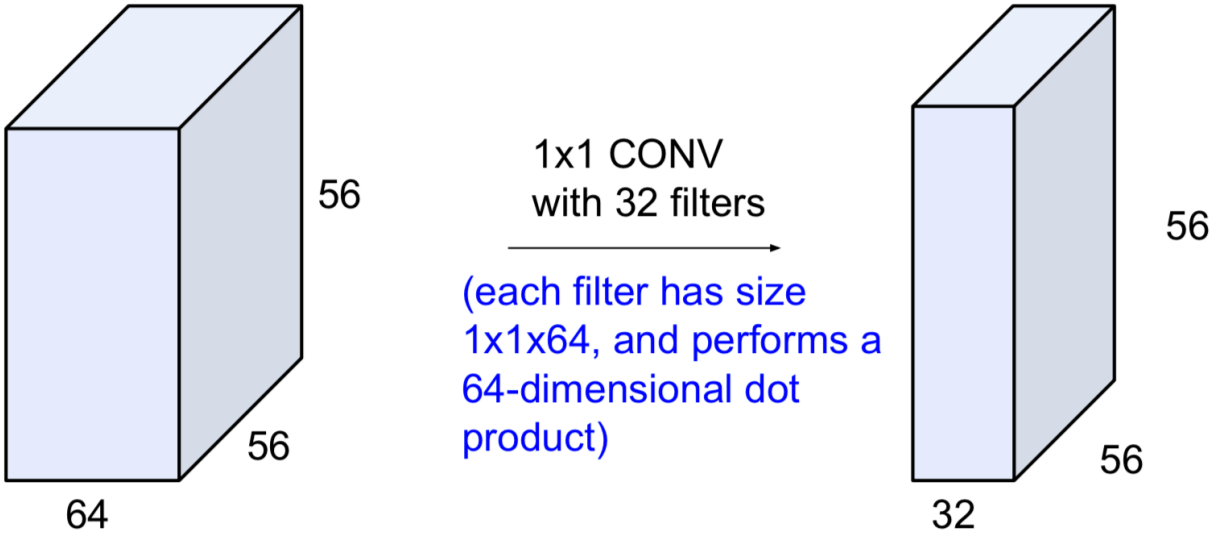
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

Convolution Layer

모두의연구소

(btw, 1x1 convolution layers make perfect sense)



Convolution Layer

모두의연구소

Example: CONV layer in Torch

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

SpatialConvolution

```
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])
```

Applies a 2D convolution over an input image composed of several input planes. The `input` tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane` : The number of expected input planes in the image given into `forward()` .
- `nOutputPlane` : The number of output planes the convolution layer will produce.
- `kW` : The kernel width of the convolution
- `kH` : The kernel height of the convolution
- `dW` : The step of the convolution in the width dimension. Default is `1` .
- `dH` : The step of the convolution in the height dimension. Default is `1` .
- `padW` : The additional zeros added per width to the input planes. Default is `0` , a good number is `(kW-1)/2` .
- `padH` : The additional zeros added per height to the input planes. Default is `padW` , a good number is `(kH-1)/2` .

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

If the input image is a 3D tensor `nInputPlane x height x width` , the output image size will be `nOutputPlane x oheight x owidth` where

```
owidth  = floor((width  + 2*padW - kW) / dW + 1)
oheight = floor((height + 2*padH - kH) / dH + 1)
```

Convolution Layer

모두의연구소

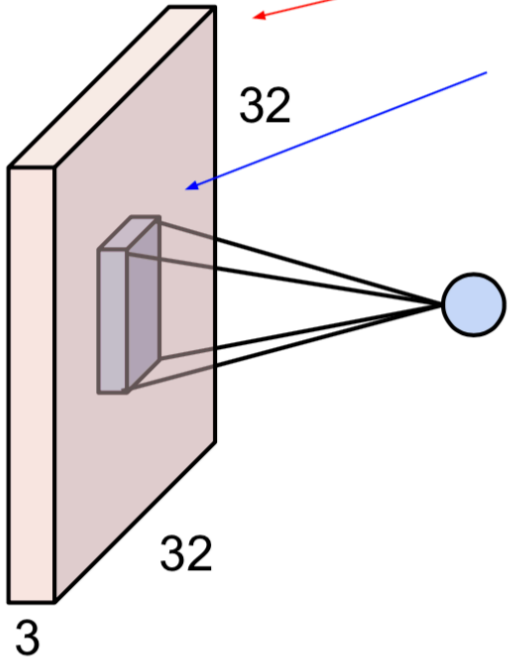
Example: CONV layer in Caffe

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  # learning rate and decay multipliers for the filters
  param { lr_mult: 1 decay_mult: 1 }
  # learning rate and decay multipliers for the biases
  param { lr_mult: 2 decay_mult: 0 }
  convolution_param {
    num_output: 96     # learn 96 filters
    kernel_size: 11    # each filter is 11x11
    stride: 4          # step 4 pixels between each filter application
    weight_filler {
      type: "gaussian" # initialize the filters from a Gaussian
      std: 0.01        # distribution with stdev 0.01 (default mean: 0)
    }
    bias_filler {
      type: "constant" # initialize the biases to zero (0)
      value: 0
    }
  }
}
```

Summary. To summarize, the Conv Layer:

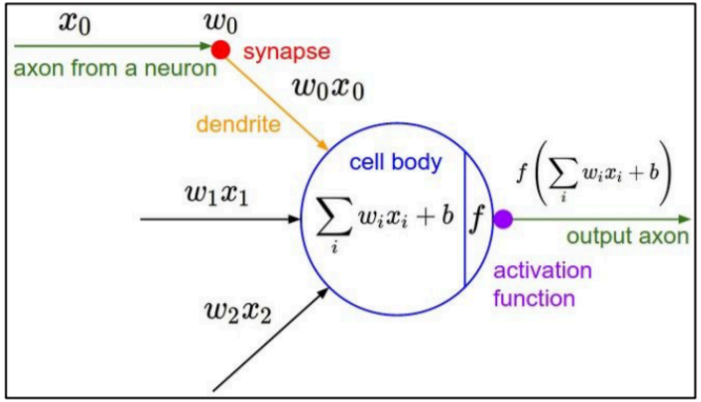
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

The brain/neuron view of CONV Layer



32x32x3 image

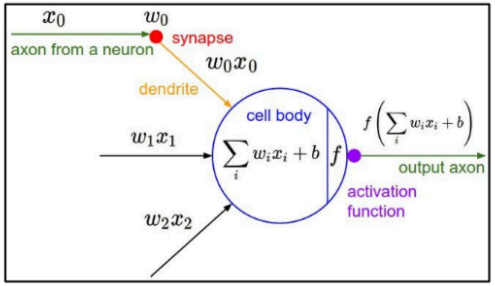
5x5x3 filter

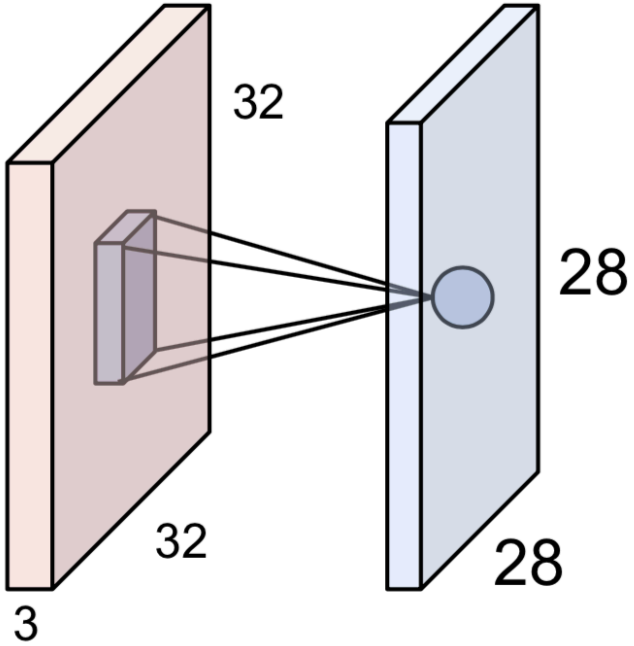


It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

The brain/neuron view of CONV Layer



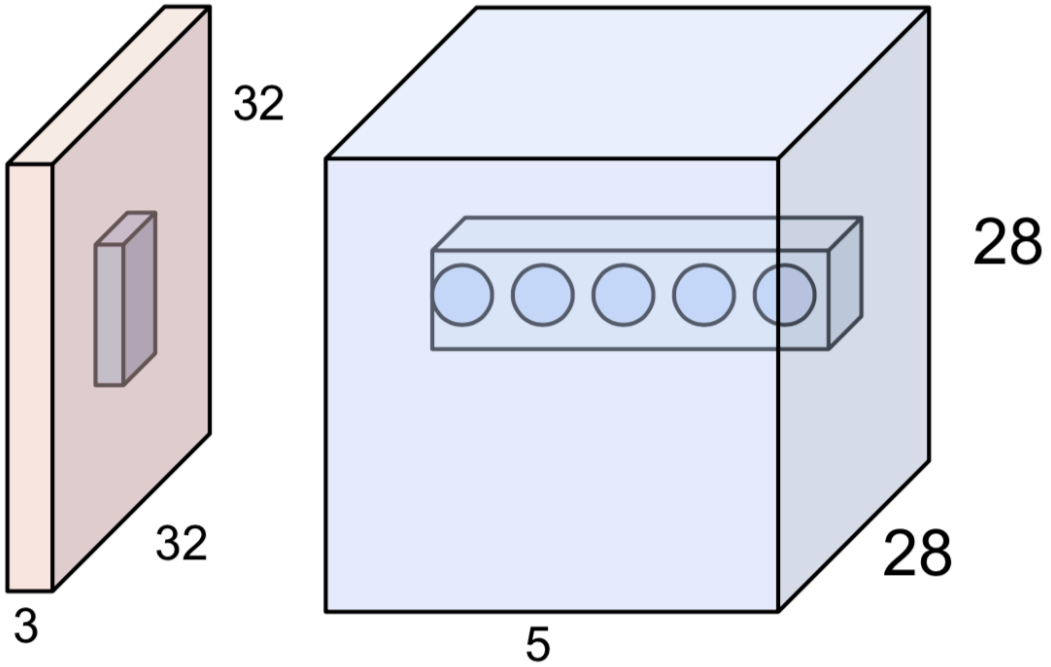


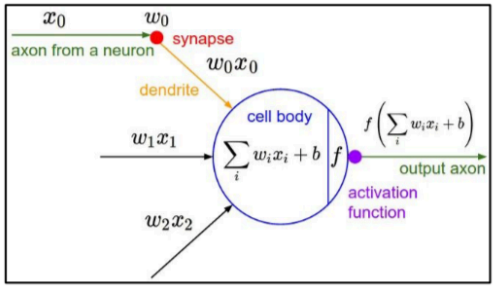
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The brain/neuron view of CONV Layer





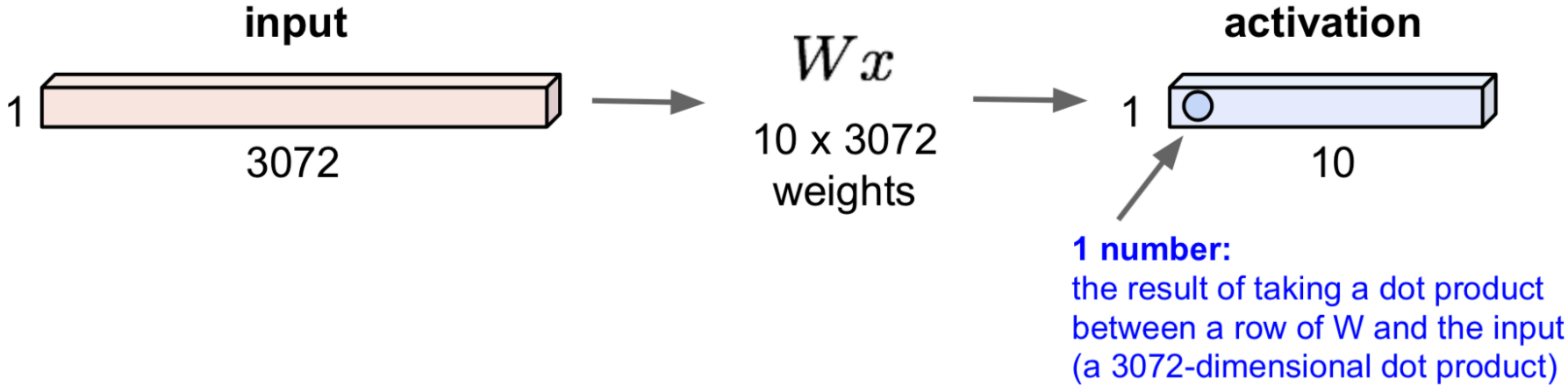
E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

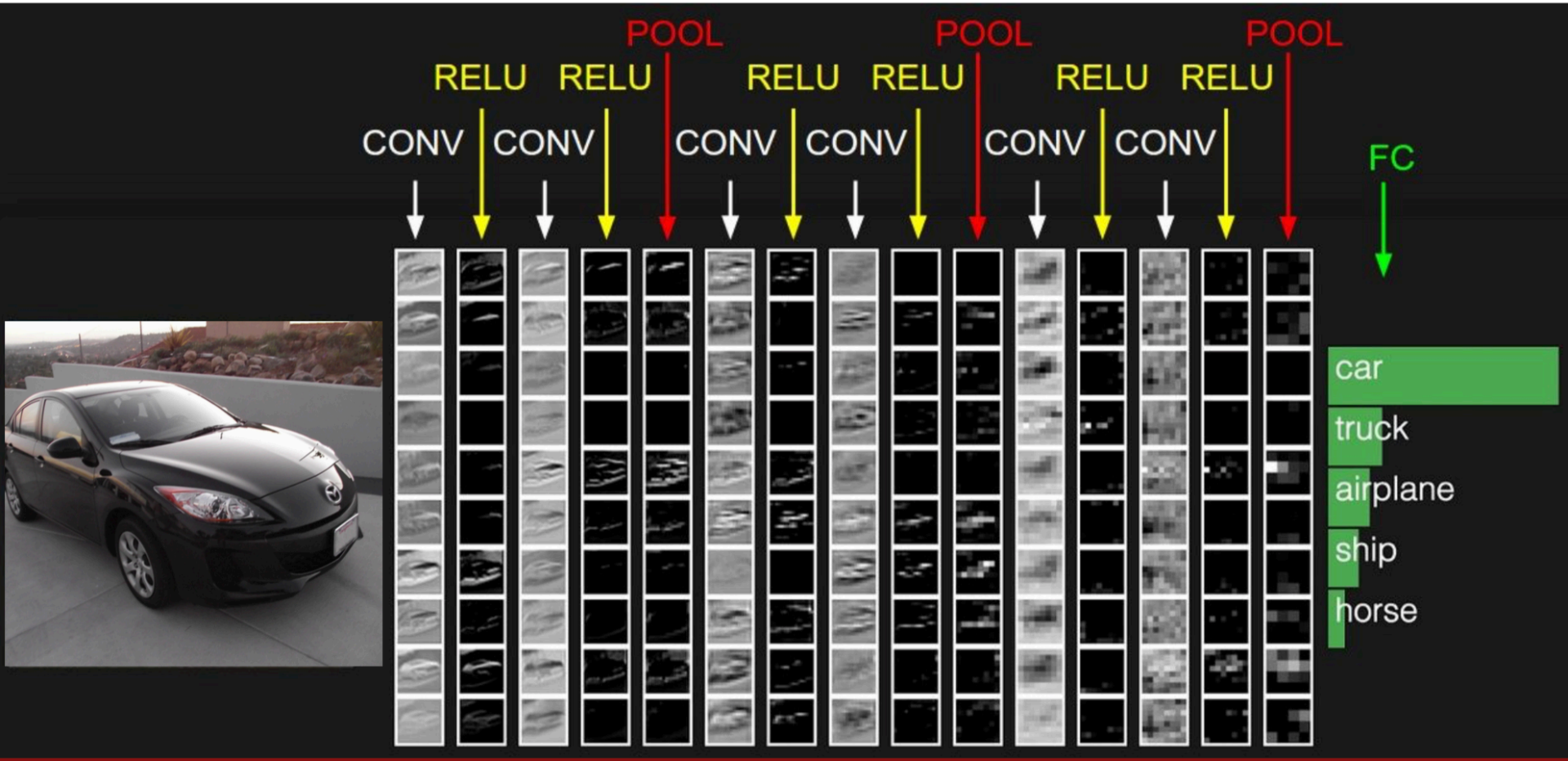
Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume

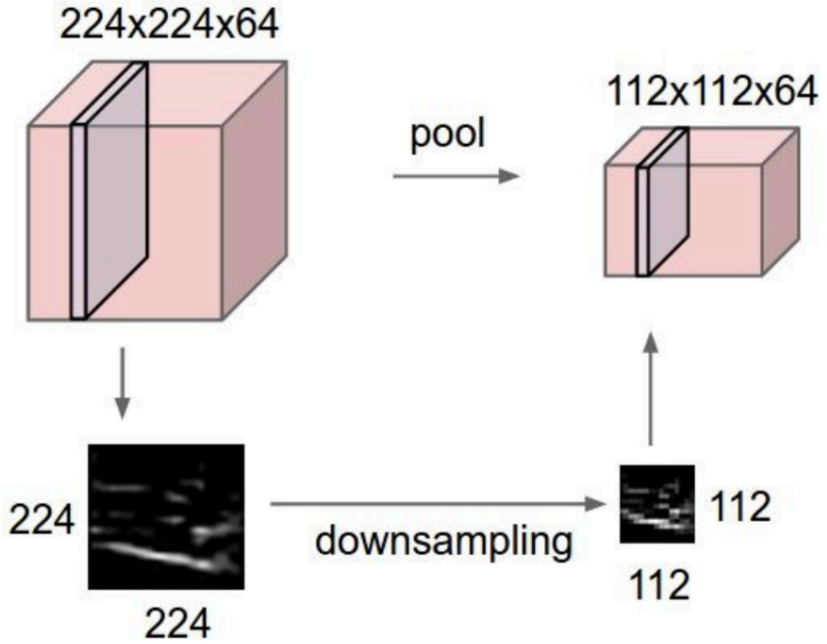


two more layers to go: POOL/FC



Pooling Layer

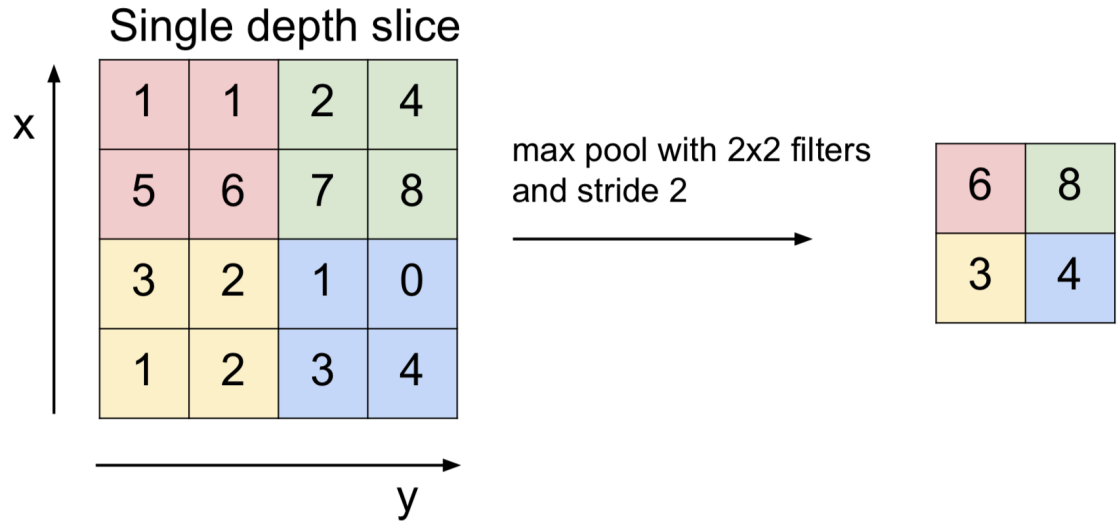
- makes the representations smaller and more manageable
- operates over each activation map independently:



Pooling Layer

모두의연구소

- 세로·가로 방향의 공간을 줄이는 연산
  - 2x2 최대 풀링(max pooling)을 스트라이드 2로

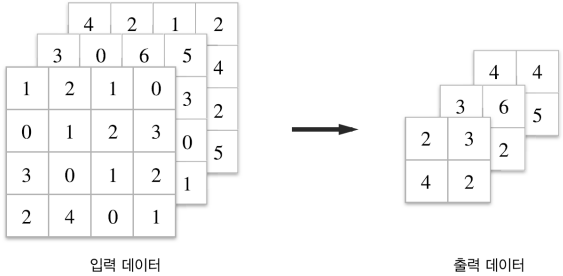


  - 평균 풀링(Average pooling)도 있습니다

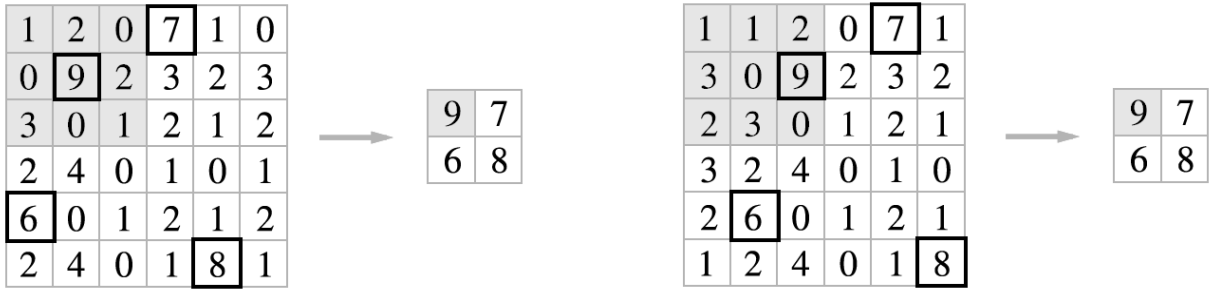
Pooling Layer

모두의연구소

- 풀링 계층의 특징
  - 학습해야 할 매개변수가 없음
  - 채널 수가 변하지 않음
  - 입력의 변화에 영향을 적게 받음 (강건하다)



데이터가 오른쪽으로 1칸씩 이동한 경우



Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Pooling 후 특징맵 크기 변화

모두의연구소

Common settings:

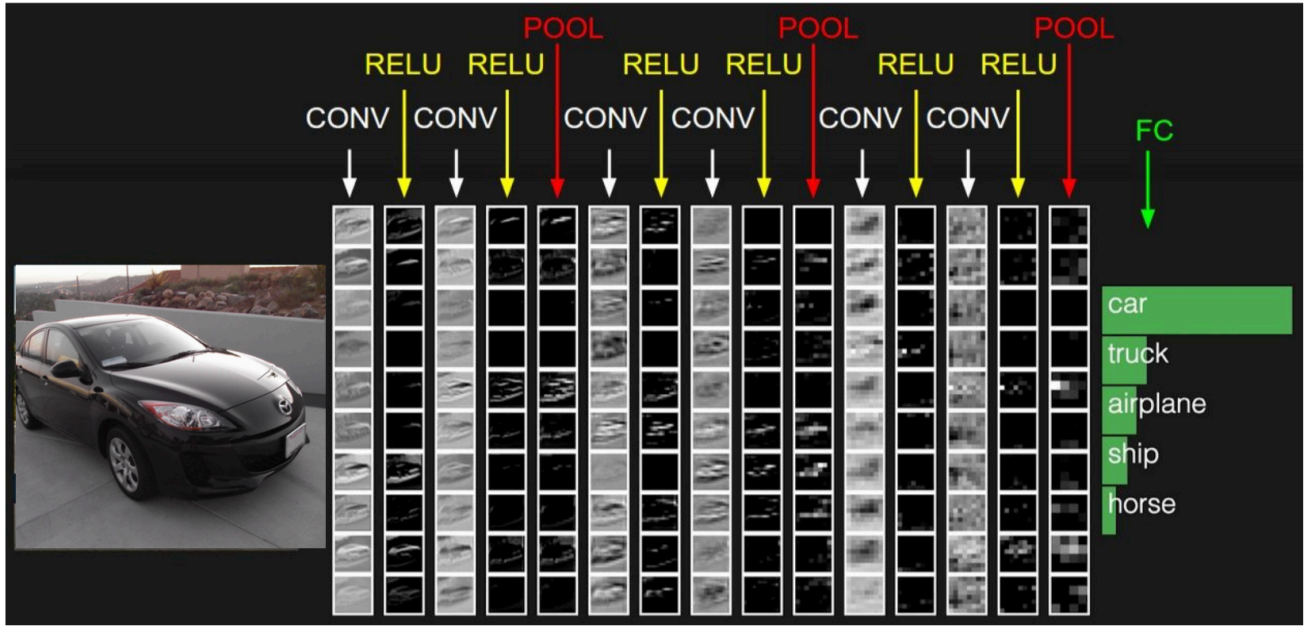
F = 2, S = 2
F = 3, S = 2

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Reference : Stanford University cs231n Lecture note 6

Fully Connected Layer (FC layer)

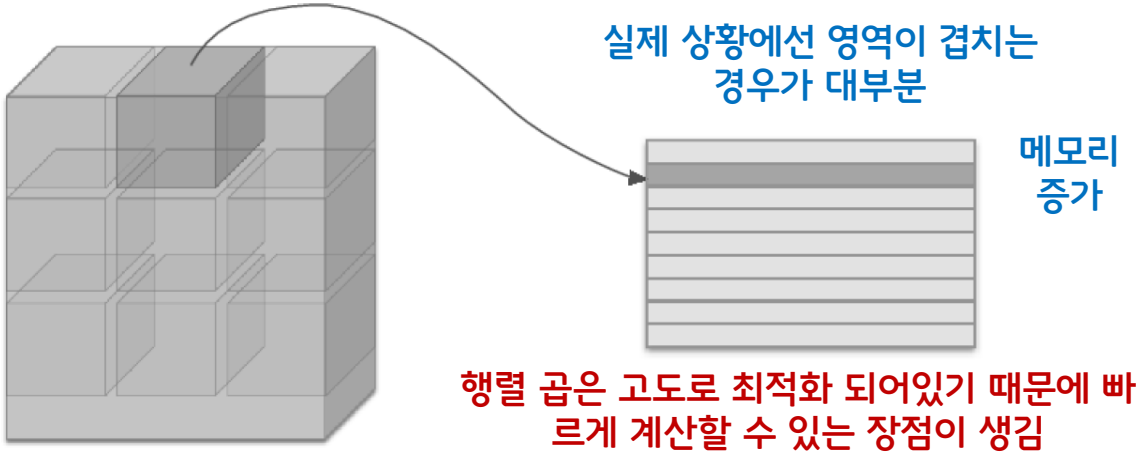
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Convolutional Networks 구현

모두의연구소

- im2col로 데이터 전개하기
  - 입력데이터에서 **필터를 적용하는 영역(3차원 블록)을 한줄로 늘어 놓습니다**
  - 이 전개를 필터를 적용하는 모든 영역에서 수행하는게 im2col입니다



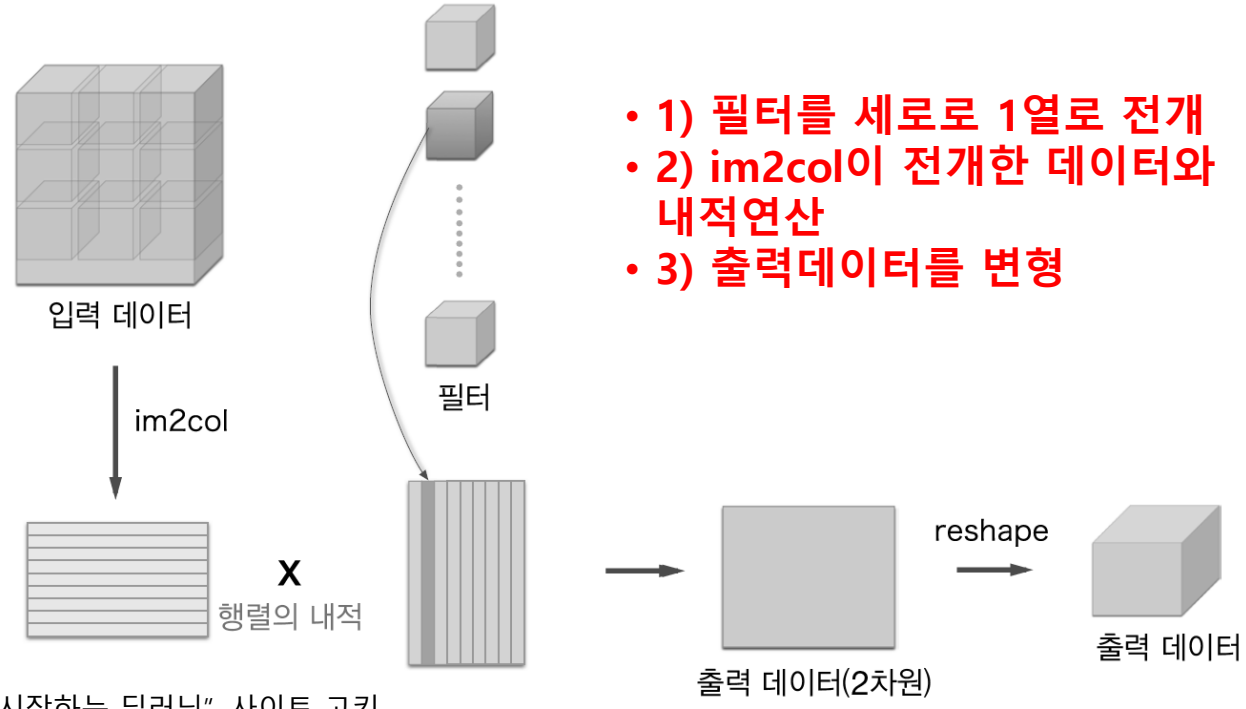
필터 적용 영역을 앞에서부터 순서대로 1줄로 펼친다

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolutional Networks 구현

모두의연구소

- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



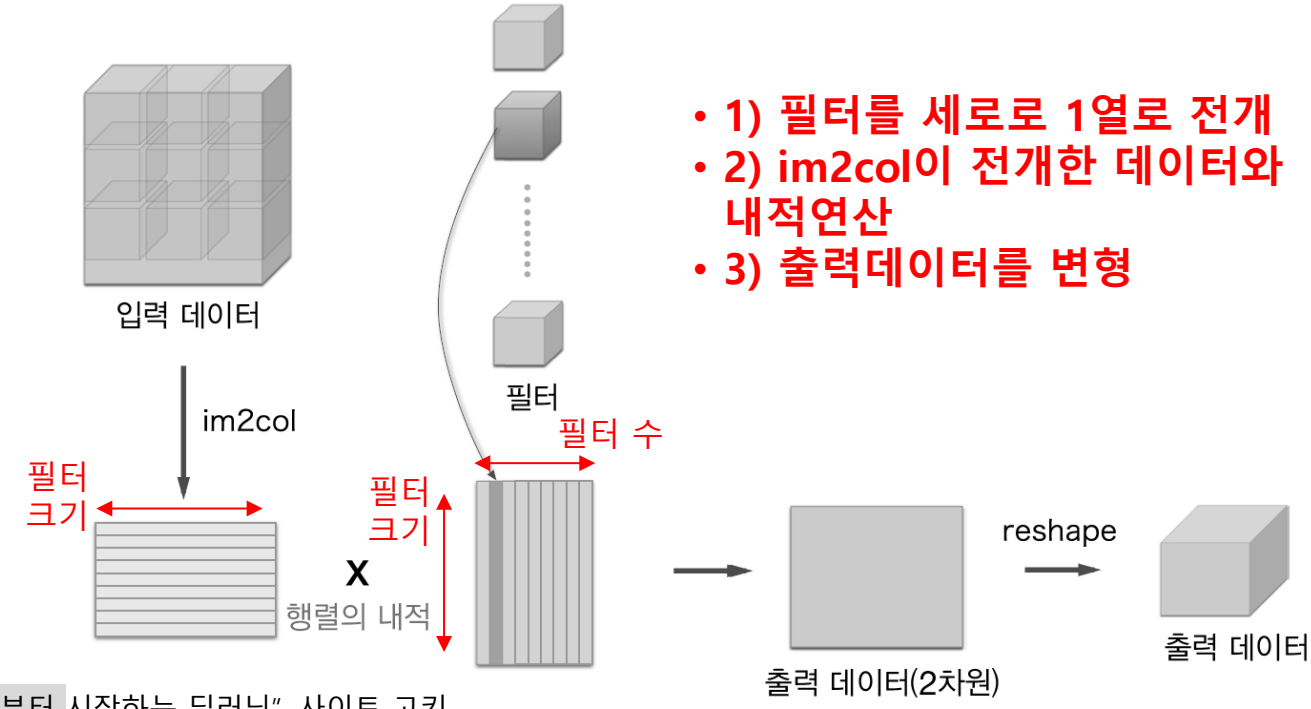
- 1) 필터를 세로로 1열로 전개
- 2) im2col이 전개한 데이터와 내적연산
- 3) 출력데이터를 변형

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolutional Networks 구현

모두의연구소

- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



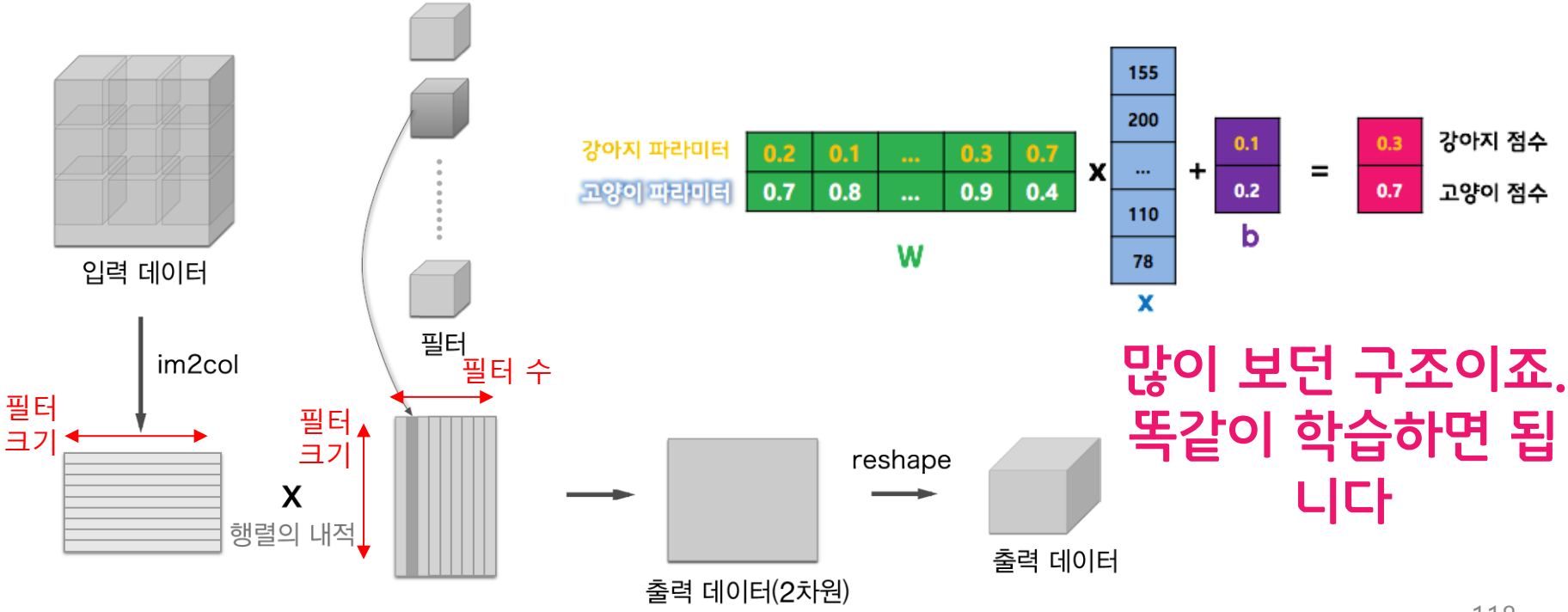
- 1) 필터를 세로로 1열로 전개
- 2) im2col이 전개한 데이터와 내적연산
- 3) 출력데이터를 변형

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolution 레이어 구현하기

모두의연구소

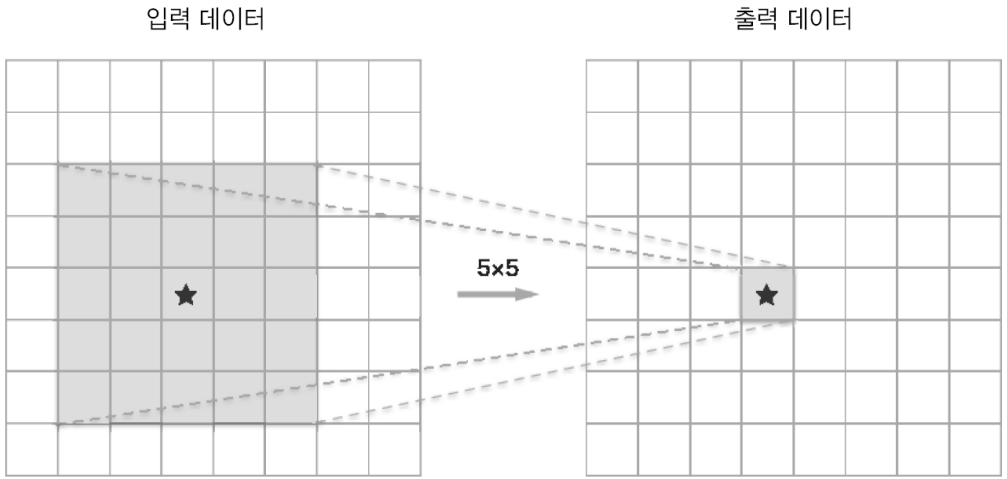
- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



많이 보던 구조이죠. 똑같이 학습하면 됩니다

더 작은 필터

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교



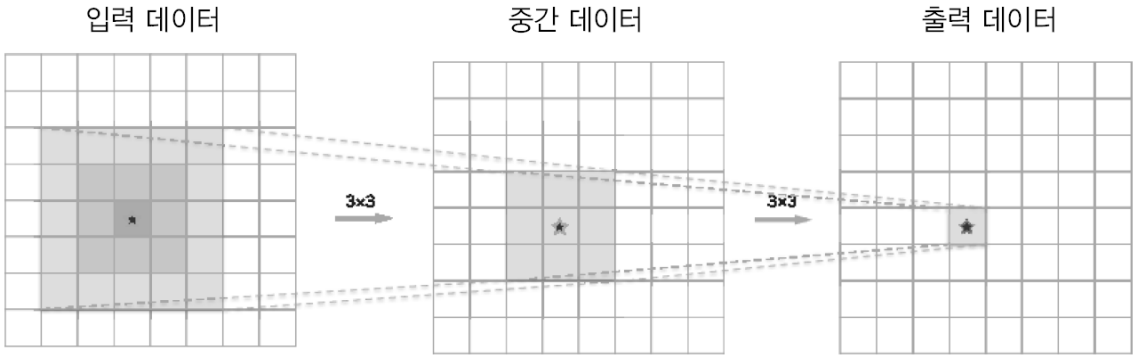
모두의연구소

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

더 작은 필터

모두의연구소

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교



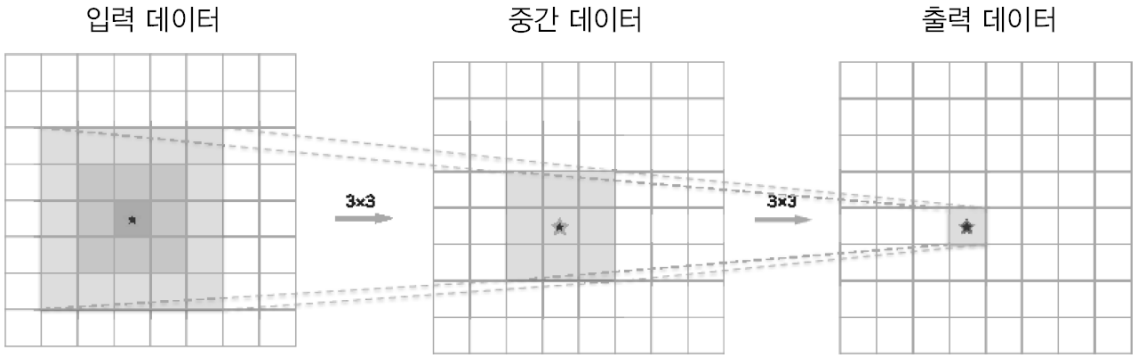
  - 5x5와 같은 크기의 영역을 처리 (receptive field)
  - 층이 깊어지기에 ReLU와 같은 비선형성 추가로 표현력이 개선. 비선형 함수가 겹쳐지면 더 복잡한것도 표현 가능

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

더 작은 필터

모두의연구소

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교

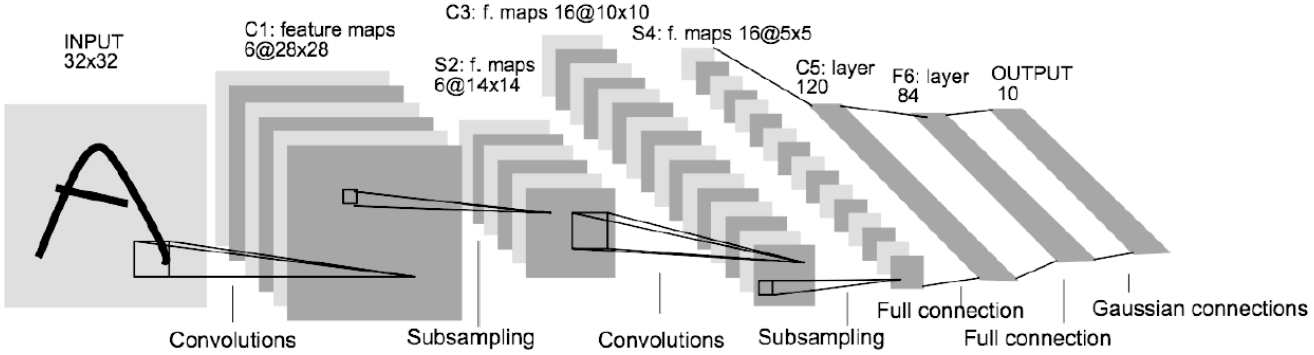


- 매개변수 수 비교
  - 5x5 필터 1개 : 25개
  - 3x3 필터 2개 : (3x3)x2 = 18개
  - 7x7 필터 1개 : 49개
  - 3x3 필터 3개 : (3x3)x3 = 27개

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

LeNet

- LeNet (1998년)



- 현재의 CNN과의 가장 큰 차이 : 시그모이드
  - 현재는 주로 ReLU

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

요약

모두의연구소

- ConvNet은 CONV, POOL, FC 레이어로 구성
- 작은 필터로 더 깊은 네트워크를 구성하는게 트렌드
- FC레이어를 제거하는게 트렌드
  - 추후 영상처리 부분에서 다루게 됨
- 전형적 구조
  - [(CONV-RELU)*N-POOL]*M-(FC-RELU)*K, SOFTMAX
    - N : 보통 ~5, M : 크게, $0 \leq K \leq 2$
- 최근엔 전혀적인 구조 보다 다양한 CNN 모델 변형들이 많아짐

또 다른 숙제

모두의연구소

- cs231n Assignment 1
  - http://cs231n.github.io/assignments2017/assignment1/
- 힌트
  - 내일저녁 업로드 예정

또 다른 숙제

모두의연구소

- 제출방법
  - Q4까지 푸시고 PDF로 인쇄 후 메일로 제출
    - 중간중간 노트북내 Question은 응답 안해도 됨
- 문의 #assign1_cs231n

Q1: k-Nearest Neighbor classifier (20 points)

The IPython Notebook **knn.ipynb** will walk you through implementing the kNN classifier.

Q2: Training a Support Vector Machine (25 points)

The IPython Notebook **svm.ipynb** will walk you through implementing the SVM classifier.

Q3: Implement a Softmax classifier (20 points)

The IPython Notebook **softmax.ipynb** will walk you through implementing the Softmax classifier.

Q4: Two-Layer Neural Network (25 points)

The IPython Notebook **two_layer_net.ipynb** will walk you through the implementation of a two-layer neural network classifier.

박 은 수 Research Director

E-mail : es.park@modulabs.co.kr