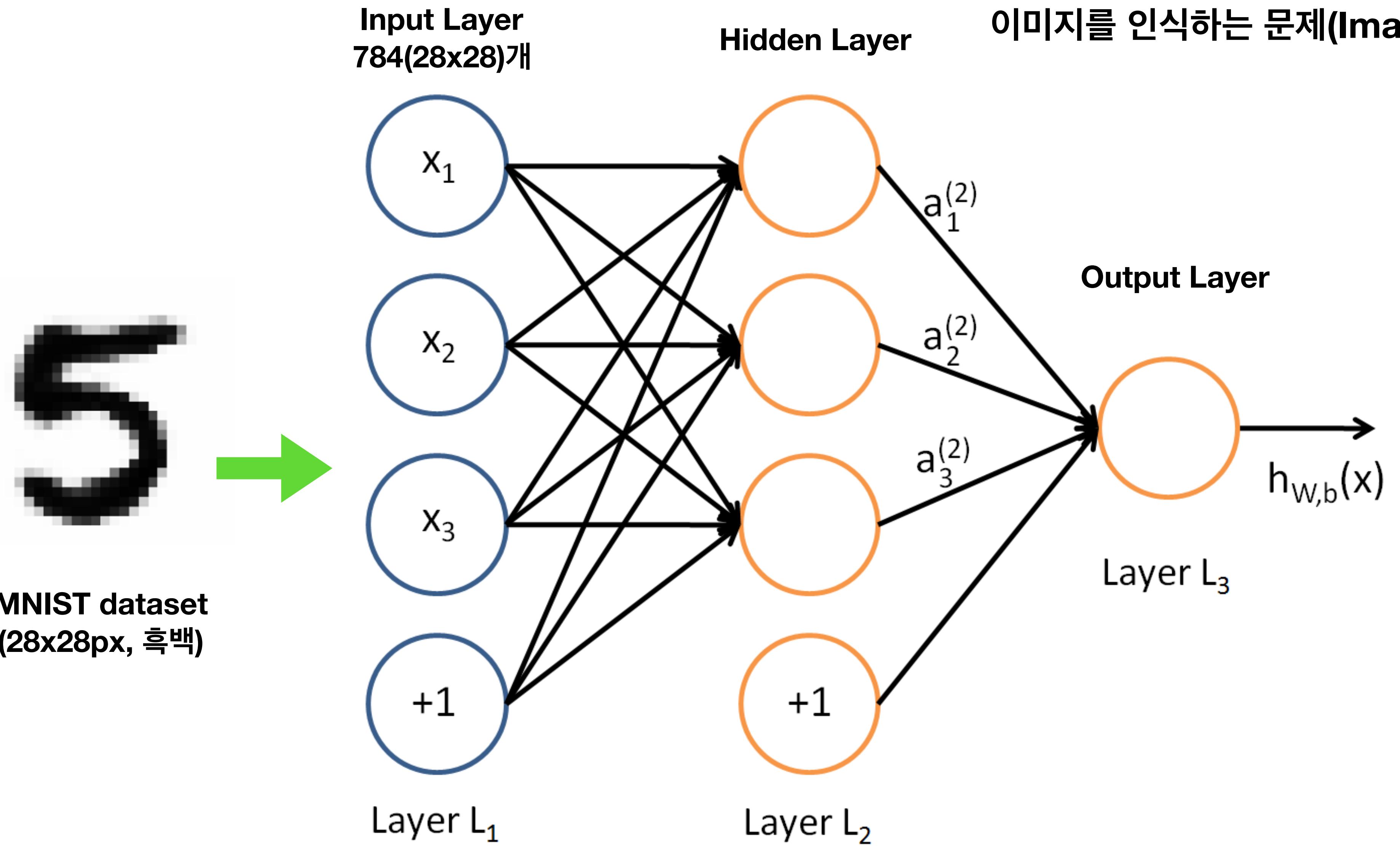


Convolutional Neural Network

Before we learned



이전까지는 Multi-layer Neural Network로
이미지를 인식하는 문제(Image Classification)를 풀었다

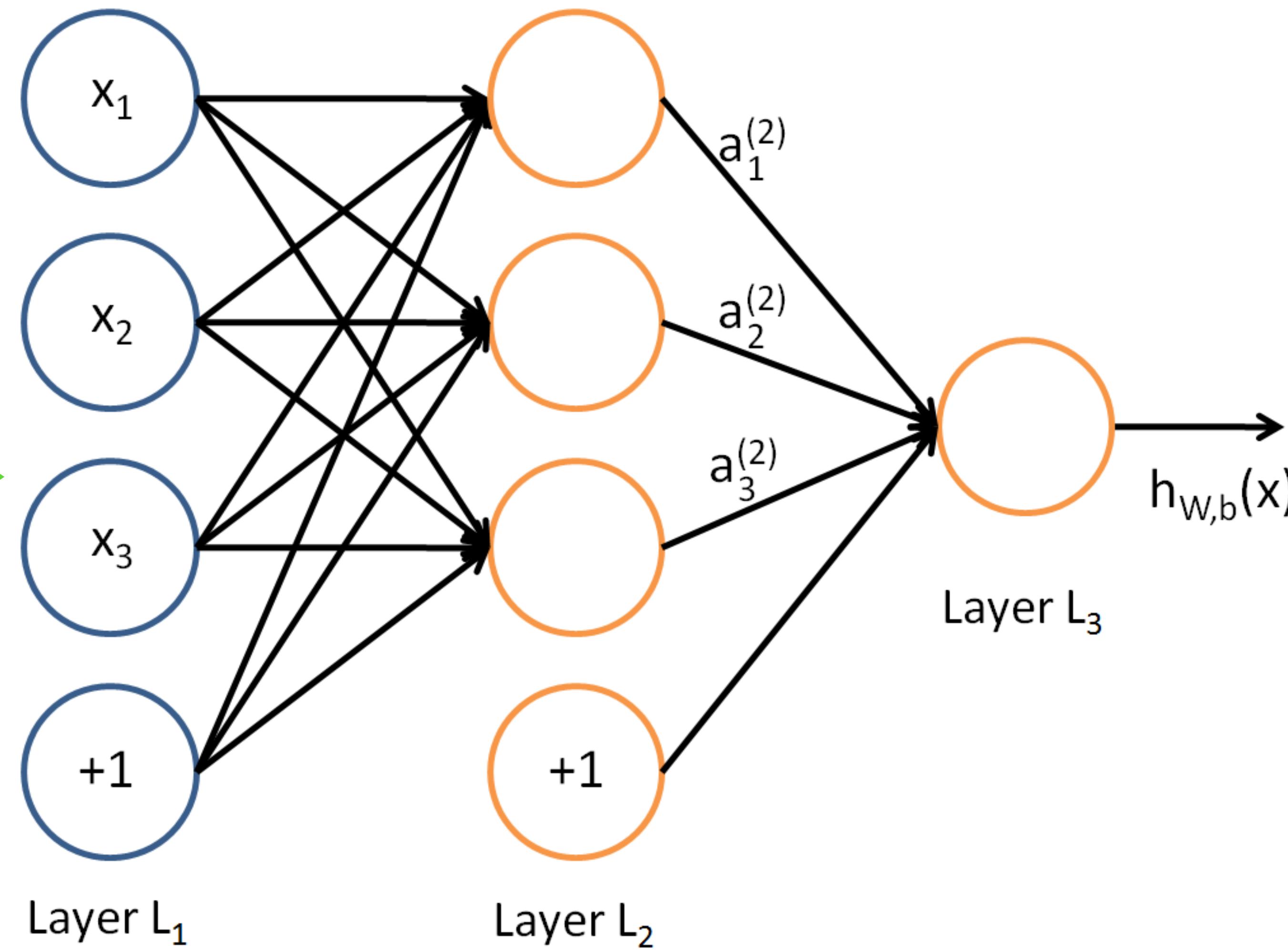
0
1
2
3
4
5
6
7
8
9

Before we learned



고양이 사진
(360x240px, RGB)

Input Layer
259200(360x240x3)개



가령 고양이나 강아지를 인식하거나, 얼굴을 인식하는 문제도 동일하게 풀 수 있다.

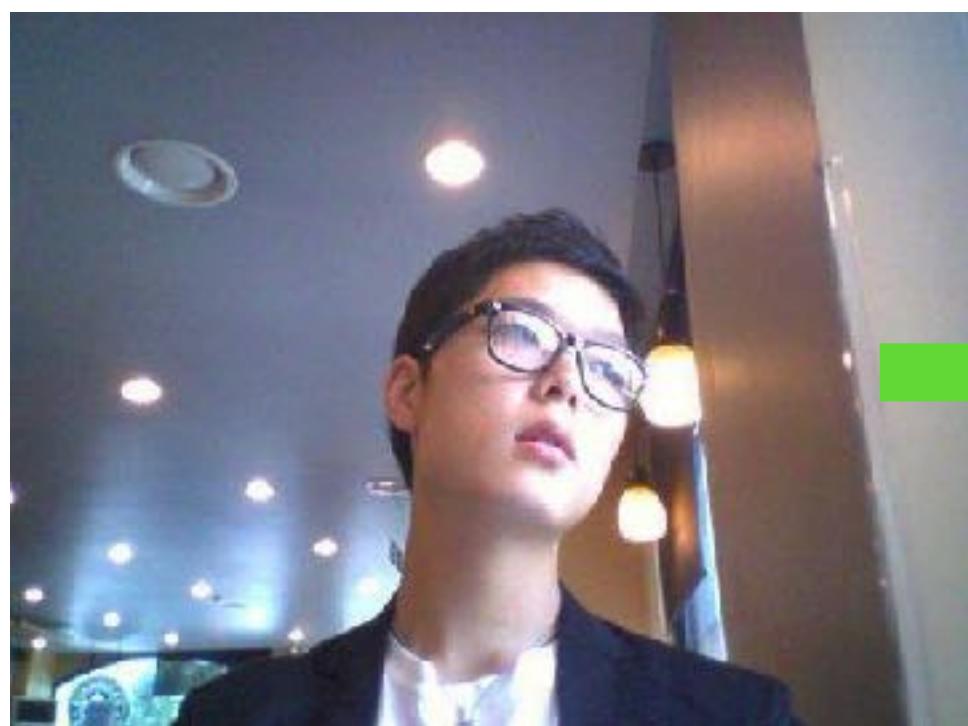
동일한 알고리즘으로 다른 이미지 인식 문제도 풀 수 있다

Cat!

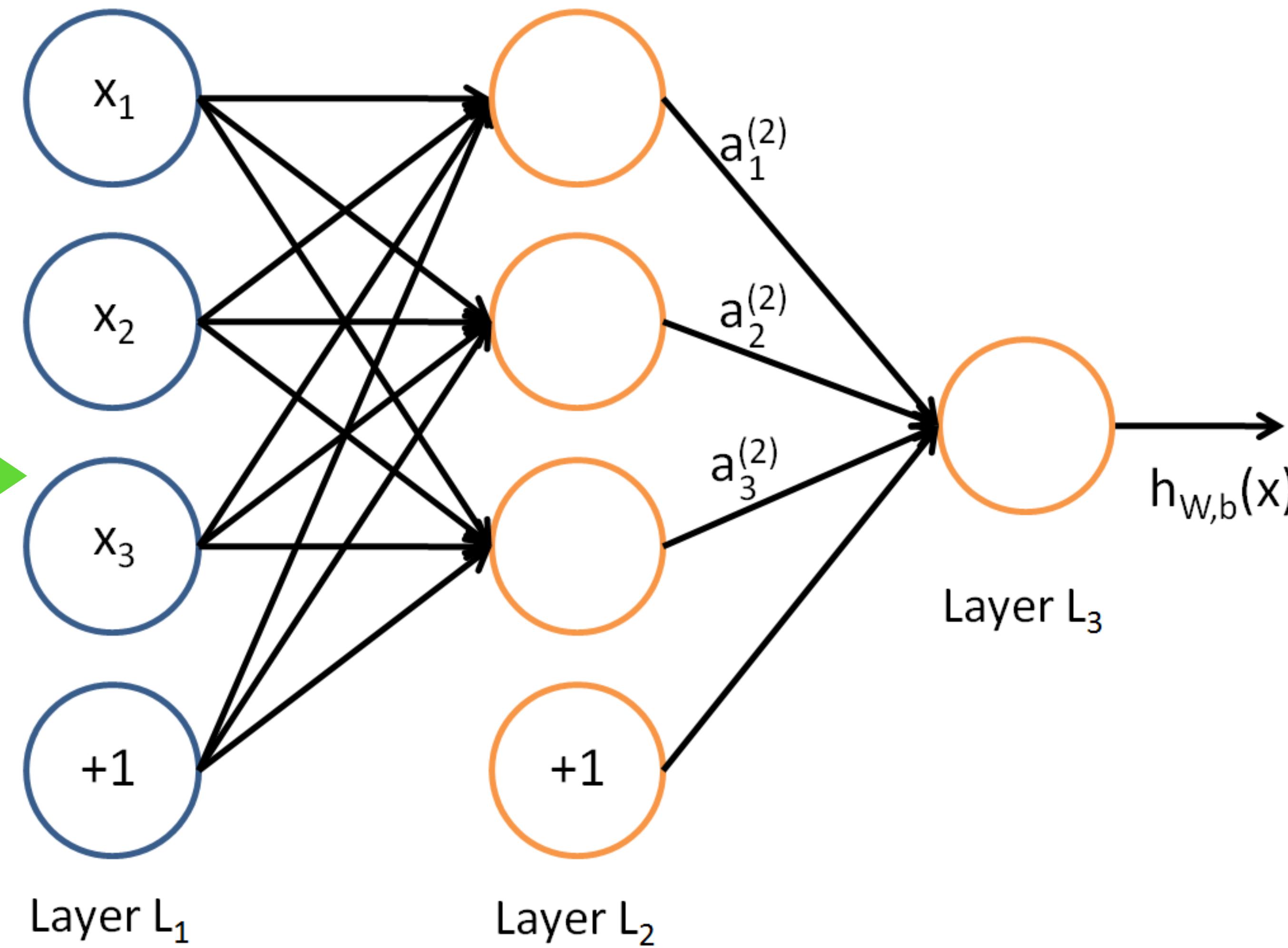
Before we learned

Input Layer
291600(360x270x3)개

동일한 알고리즘으로 다른 이미지 분류(Image Classification) 문제도 풀 수 있다
가령 고양이나 강아지를 인식하거나, 얼굴을 인식하는 문제도 동일하게 풀 수 있다.

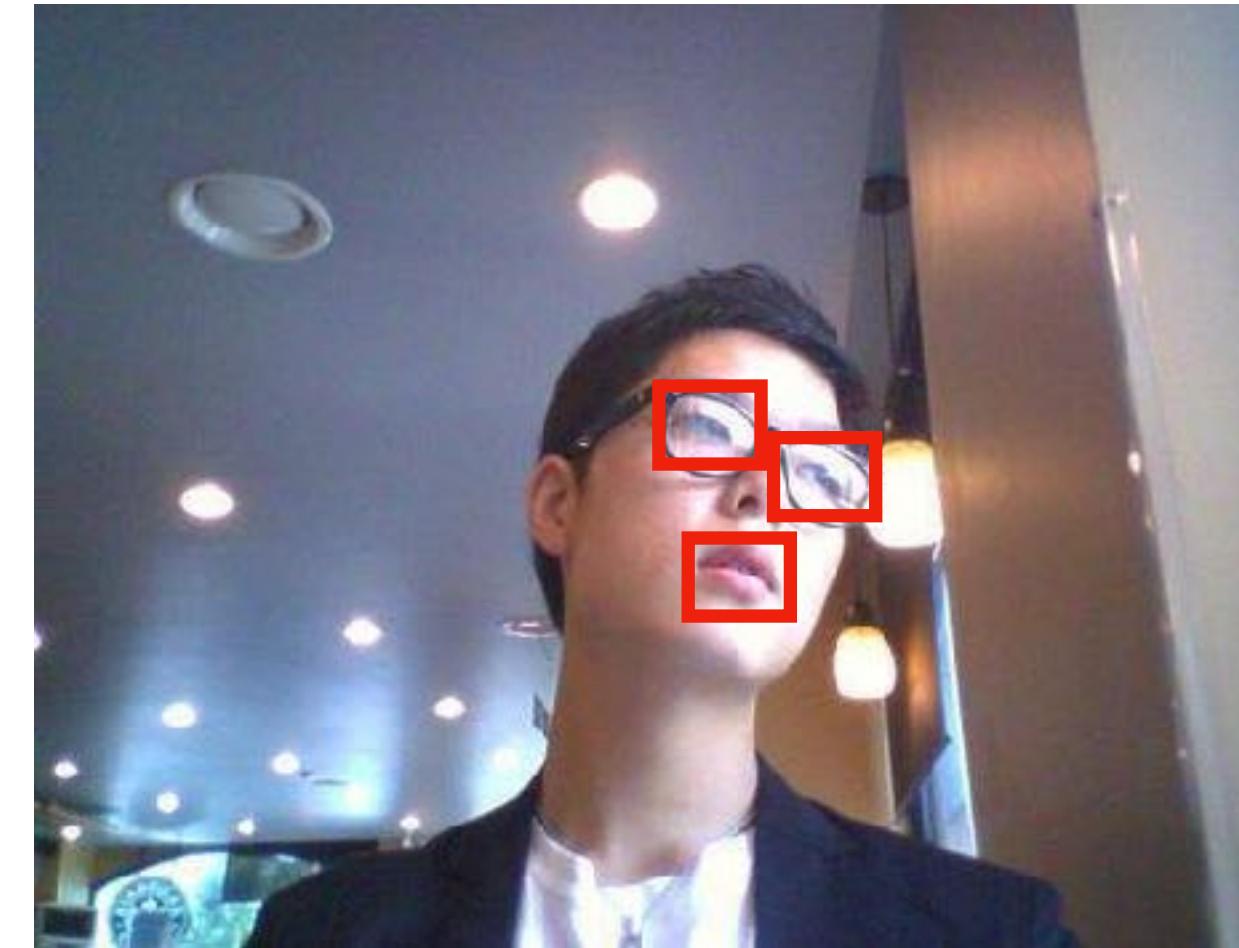


얼굴 사진
(360x270px, RGB)



The problem of Multi-layer Neural Network

모든 픽셀들의 연관관계를 다 계산할 필요가 있을까? 이미지라는건 주변 픽셀들간의 조합을 통해 특징(눈, 코, 입)을 표현하므로(locally-connected), 모든 픽셀간의 연관관계를 파악하는 것은 낭비에 가깝다



이미지는 주변 픽셀들의 조합을 통해
사람을 구분하는데 중요한 요소(눈, 코, 입)를 표현한다.

The problem of Multi-layer Neural Network

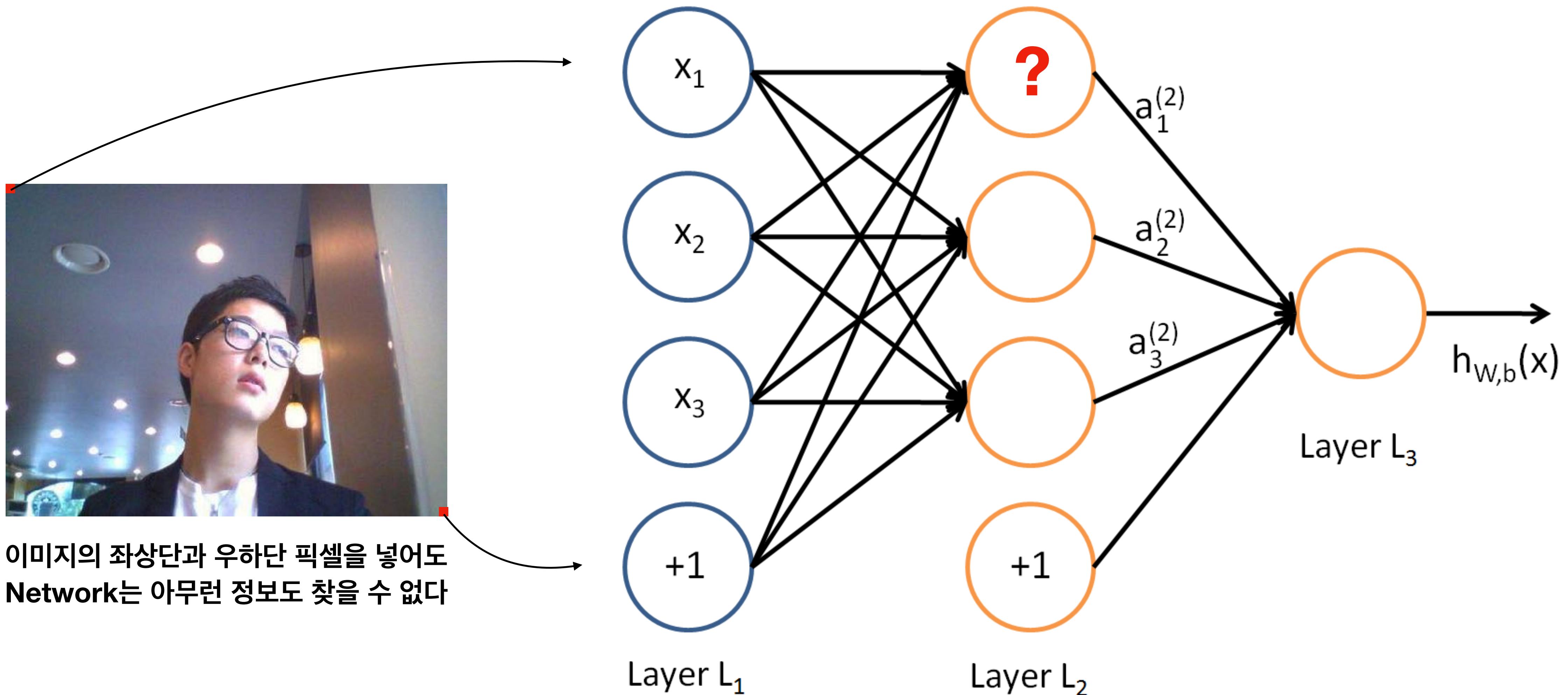
모든 픽셀들의 연관관계를 다 계산할 필요가 있을까? 이미지라는건 주변 픽셀들간의 조합을 통해 특징(눈, 코, 입)을 표현하므로(locally-connected), 모든 픽셀간의 연관관계를 파악하는 것은 낭비에 가깝다



반면 거리가 먼 픽셀들끼리는 연관관계가 없다.
이런 픽셀들의 관계를 유추하는 것은 큰 의미가 없다.

The problem of Multi-layer Neural Network

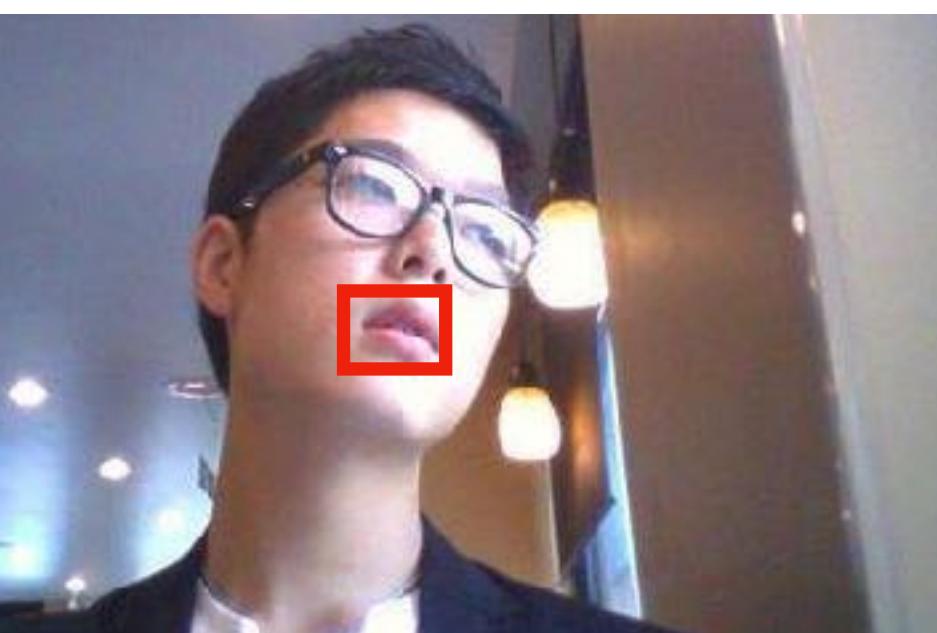
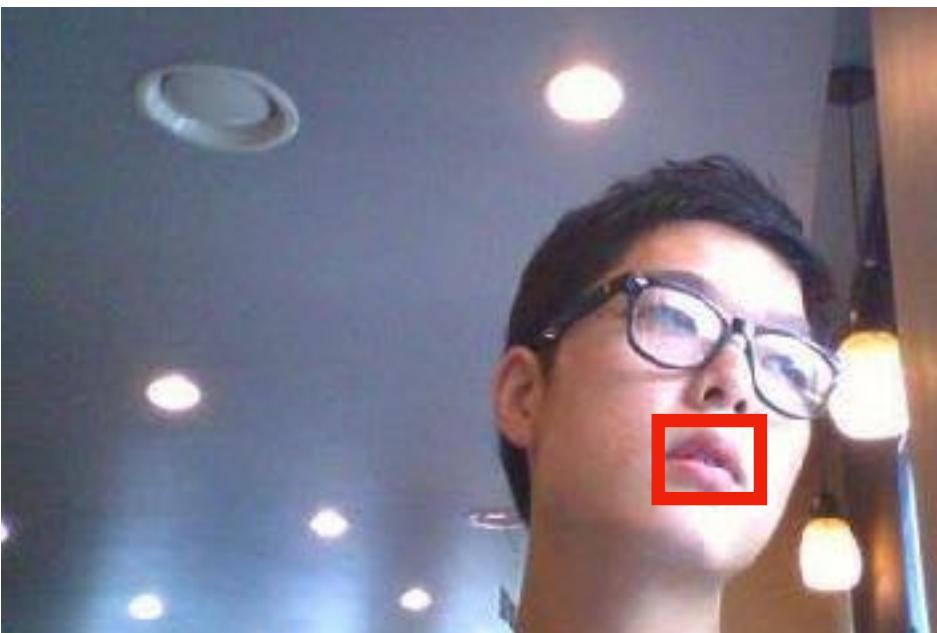
모든 픽셀들의 연관관계를 다 계산할 필요가 있을까? 이미지라는건 주변 픽셀들간의 조합을 통해 특징(눈, 코, 입)을 표현하므로(locally-connected), 모든 픽셀간의 연관관계를 파악하는 것은 낭비에 가깝다



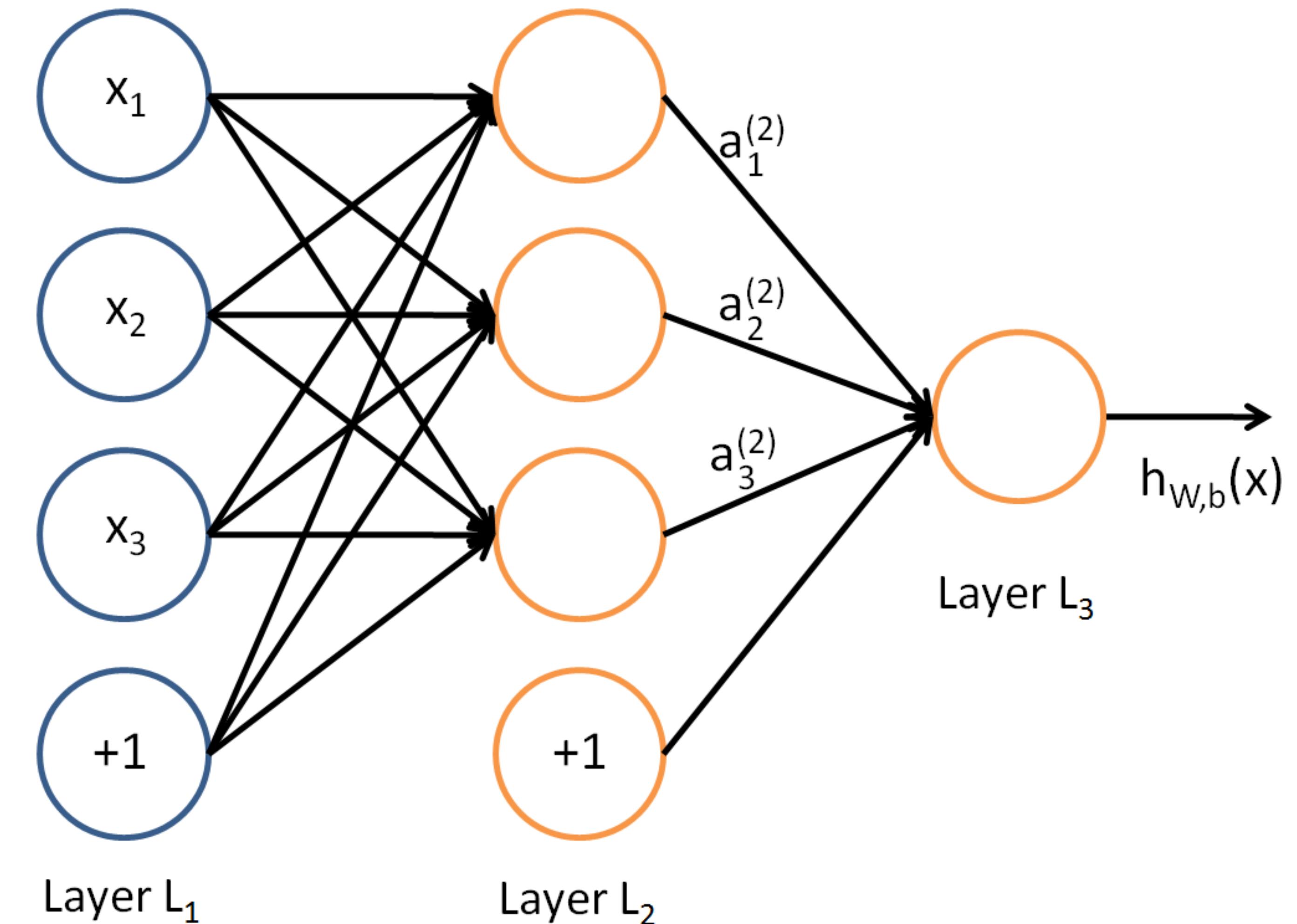
The problem of Multi-layer Neural Network

물체의 특징은(눈, 코, 입)은 사진의 어디에서든지 나타날 수 있다.

하지만 Multi-layer Neural Network는 다른 위치에서 등장하는 동일한 특징을 동일하게 인식하지 못한다.

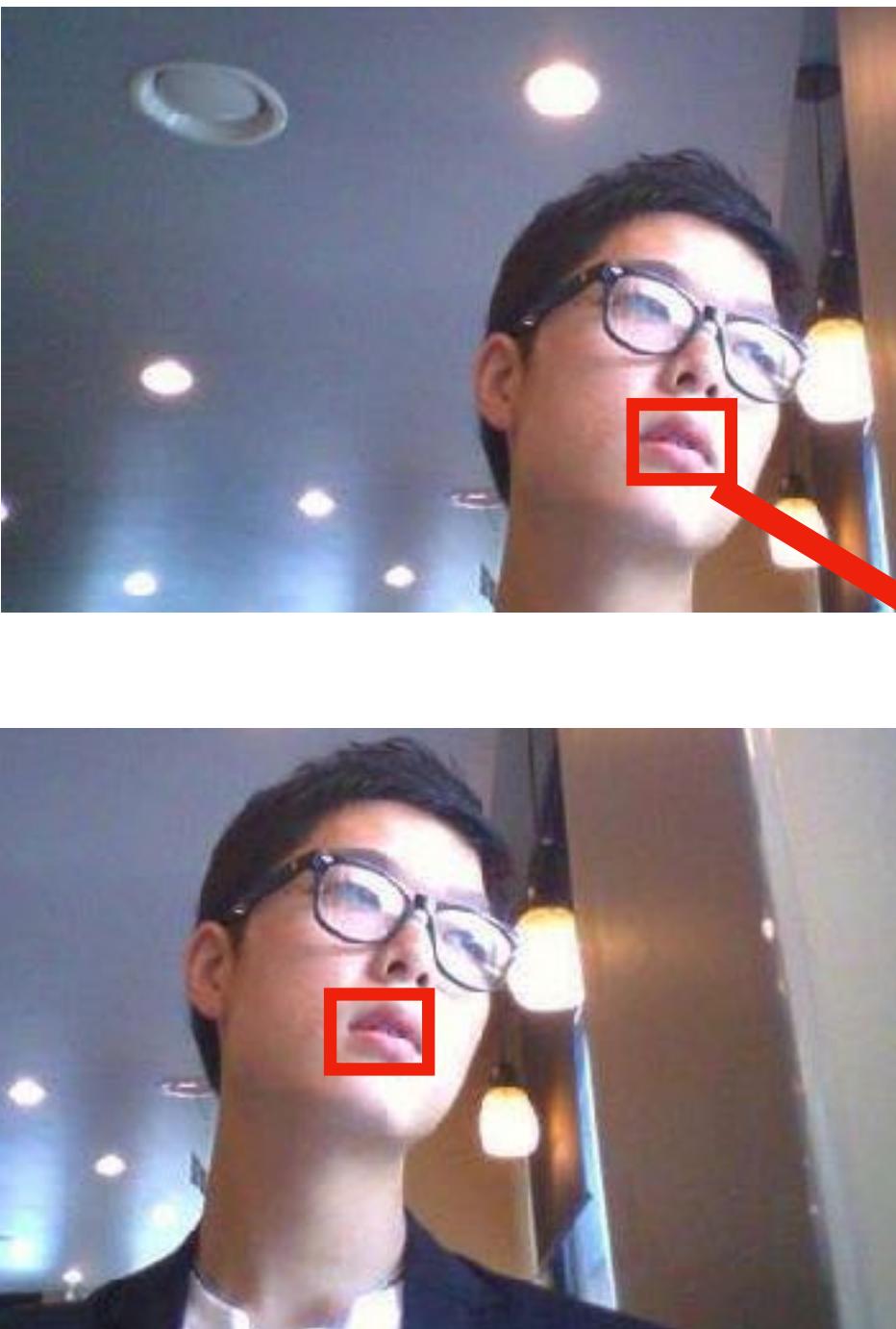


동일한 입이지만 다른 위치에 있으므로
Network는 다른 특징이라고 인식한다
(해당 픽셀을 담당하는 weight가 서로 다르기 때문)

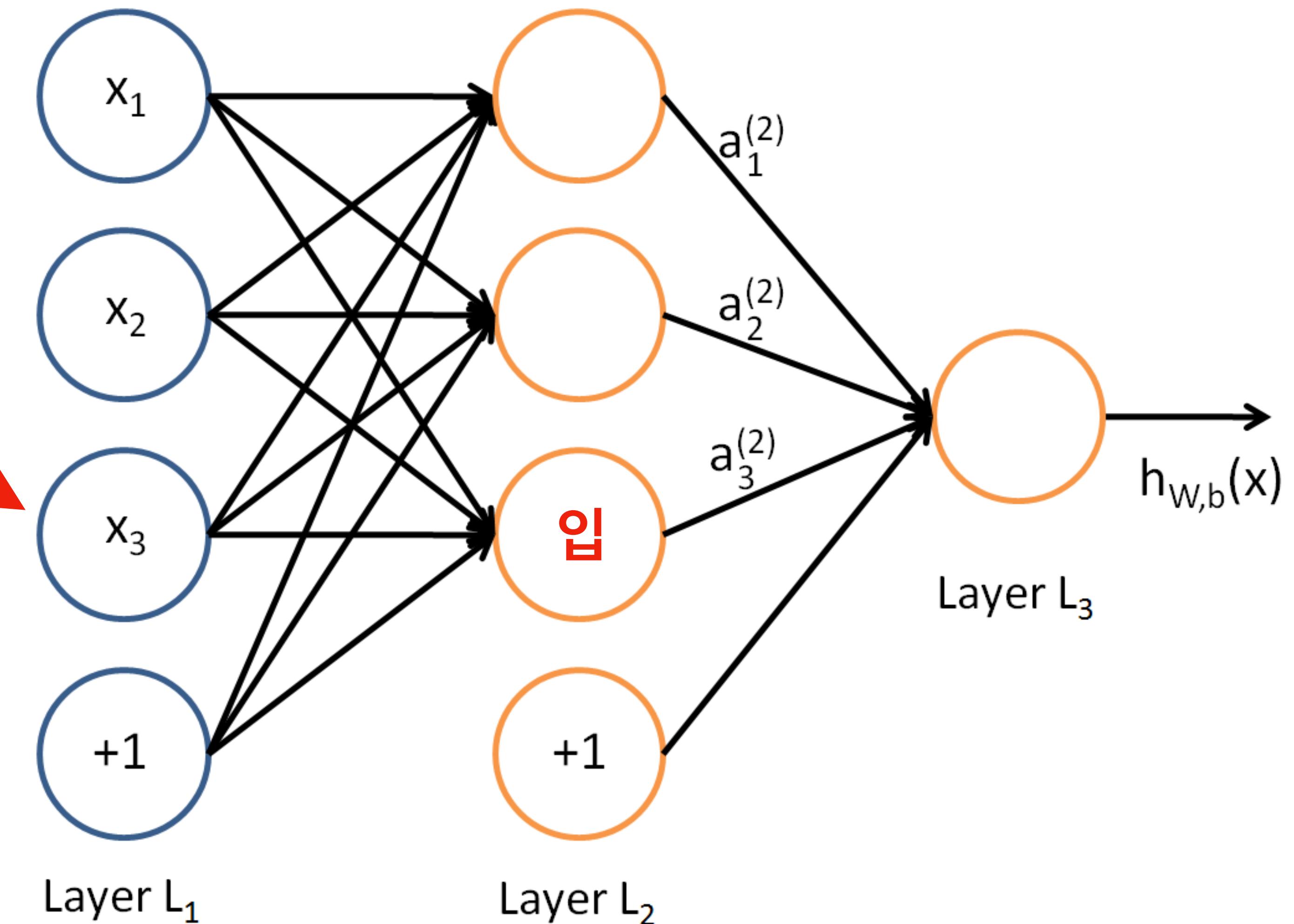


The problem of Multi-layer Neural Network

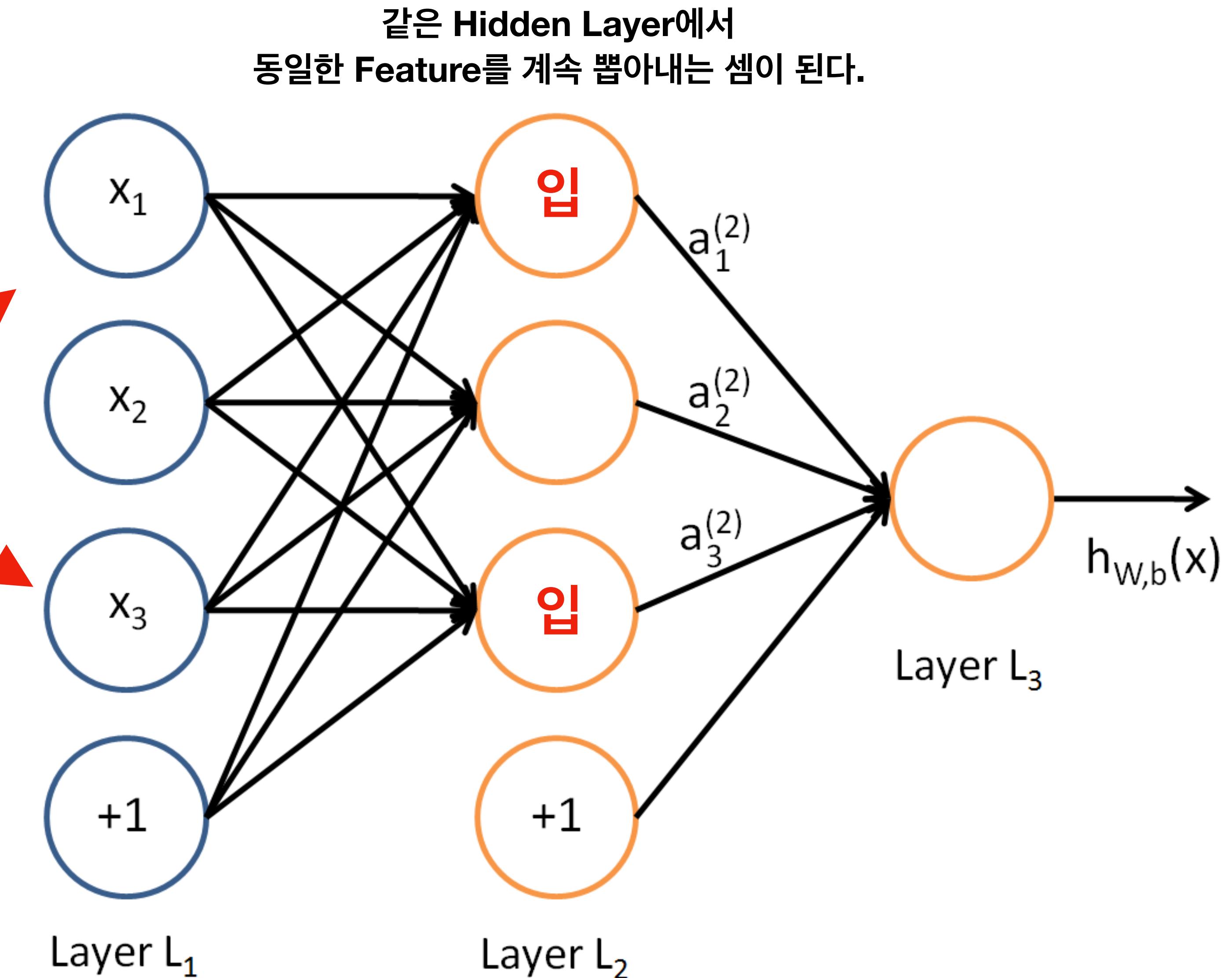
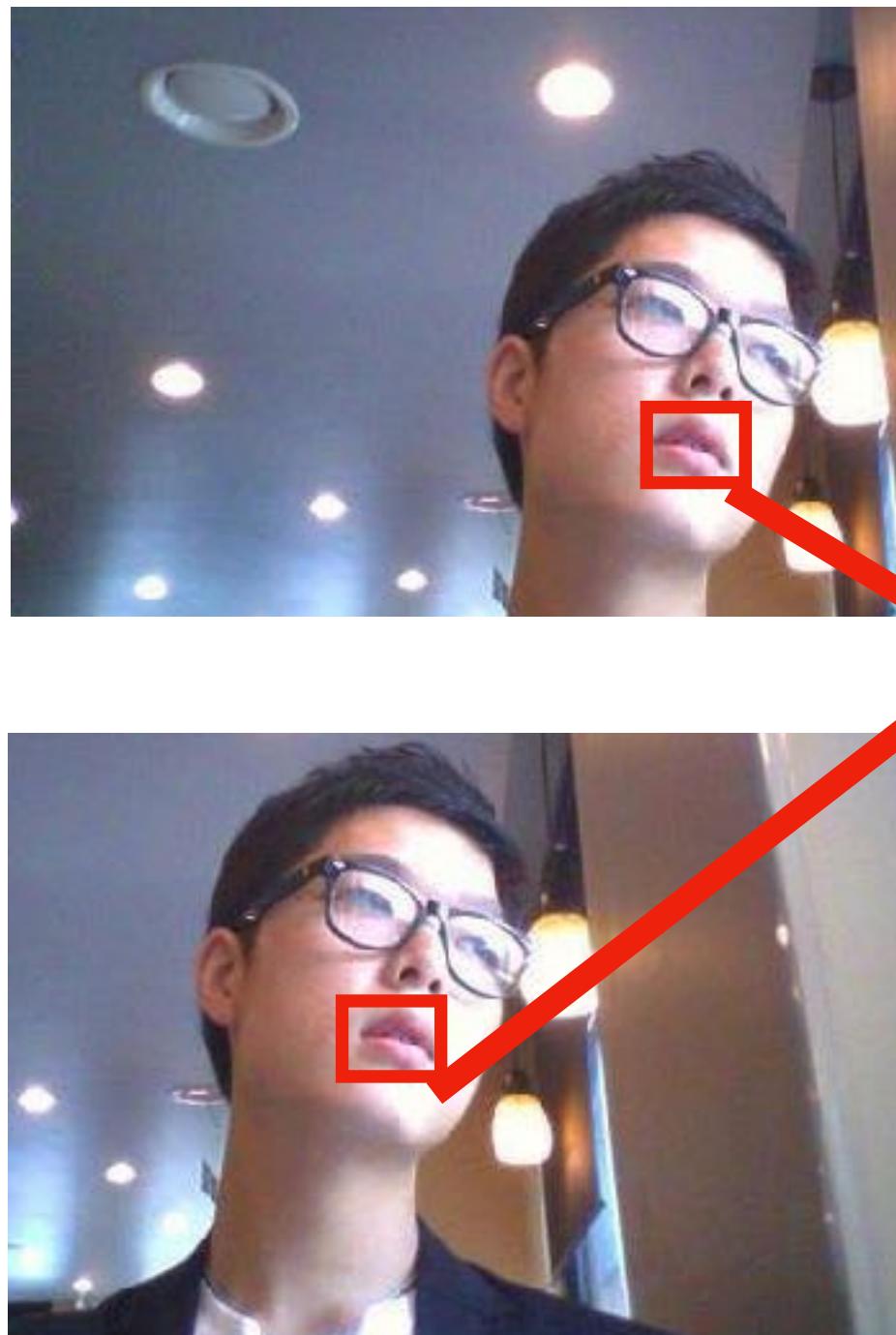
물체의 특징은(눈, 코, 입)은 사진의 어디에서든지 나타날 수 있다.
하지만 Multi-layer Neural Network는 다른 위치에서 등장하는 동일한 특징을 동일하게 인식하지 못한다.



동일한 입이지만 다른 위치에 있으므로
Network는 다른 특징이라고 인식한다
(해당 픽셀을 담당하는 weight가 서로 다르기 때문)



The problem of Multi-layer Neural Network



Solution

weight의 양을 줄이되, 전체 픽셀이 아닌 주변 픽셀만 계산한다.
이렇게 하면 **weight**의 크기를 대폭적으로 줄이면서도 효율적인 연산이 가능하다.

Before



After



Multi-layer Neural Network의 **Hidden Layer**는
사실상 모든 픽셀을 한 번에 연산한다.

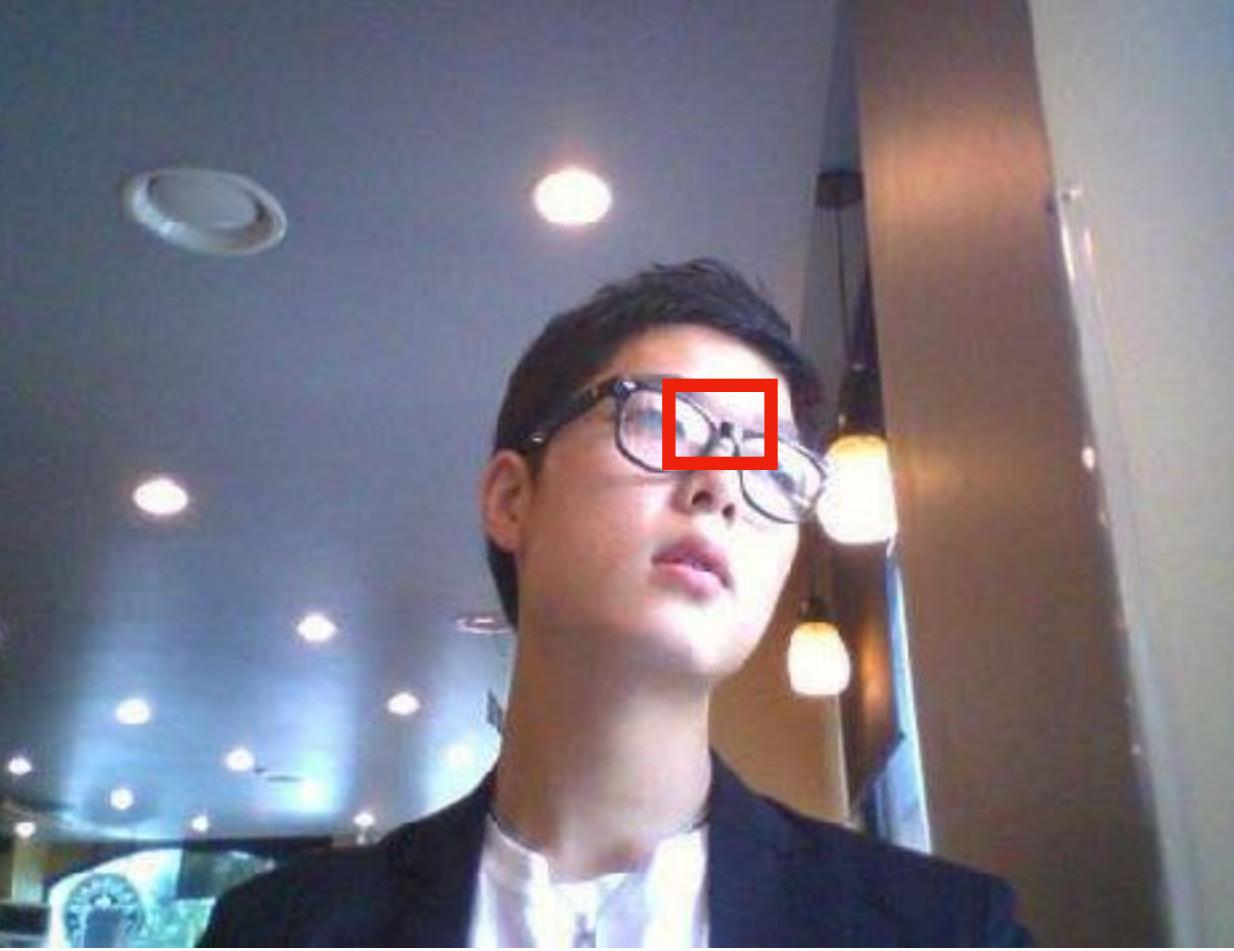
모든 픽셀이 아닌 주변 픽셀만 연산해도
동일하게 입을 인식할 수 있다.

Solution

픽셀 위치마다 같은 Hidden Layer로 계산한다.
이렇게 하면 다른 위치에 있는 같은 특징도 동일하게 인식할 수 있다



Hidden Layer가 눈을 인식한다



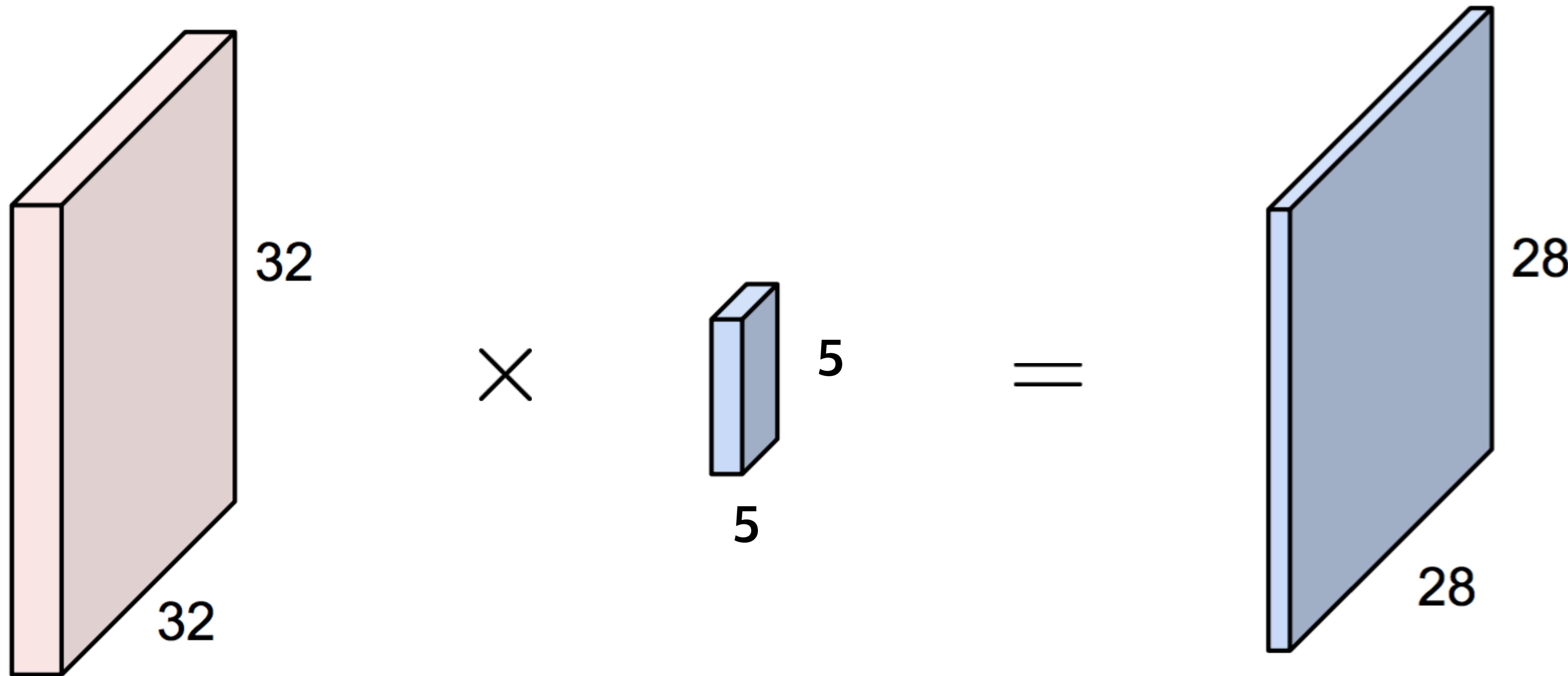
눈을 인식하지 못 한다



Hidden layer가 눈을 인식하며,
이 layer는 처음의 layer와 동일하므로
같은 눈이라고 판단할 수 있다.

Convolutional Neural Network: Overview

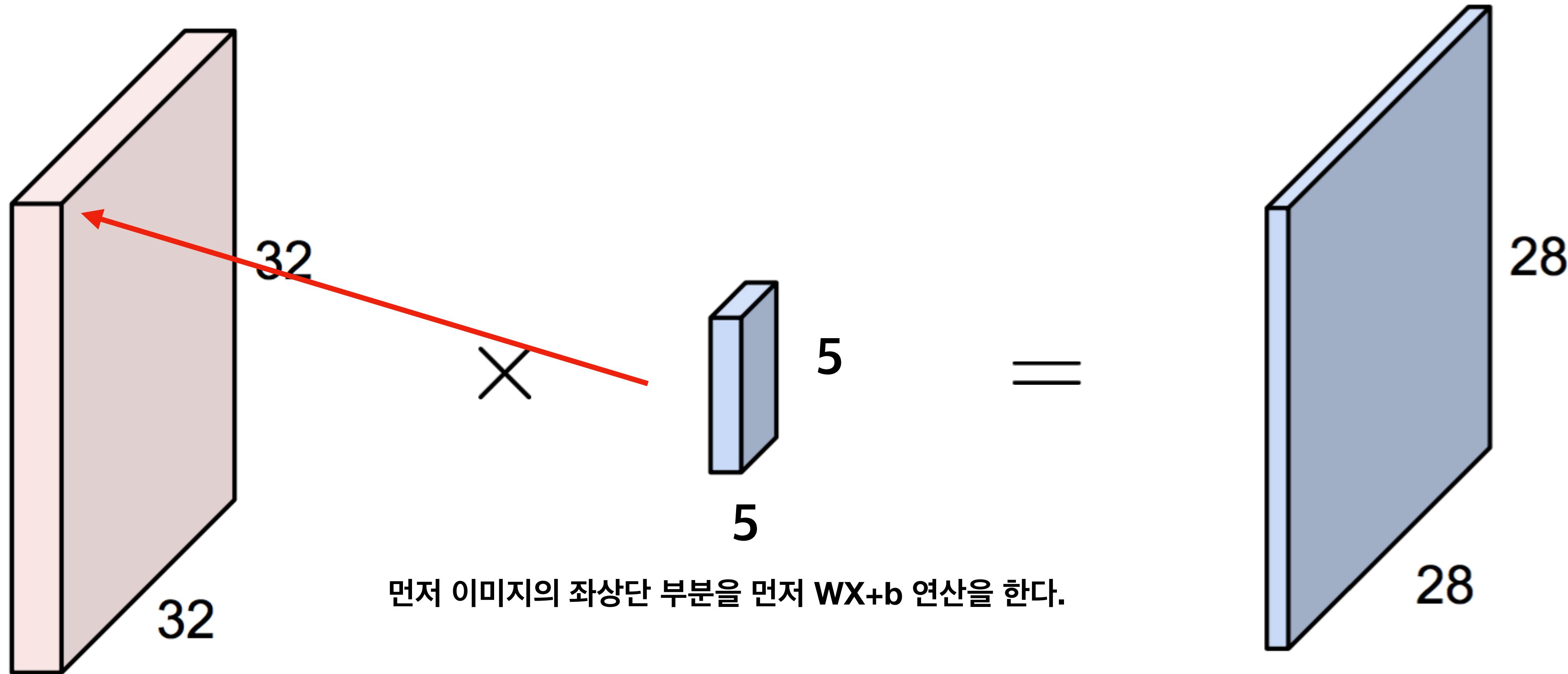
위와 같은 문제를 Convolutional Layer를 도입하면 해결할 수 있다
우리는 이를 Convolutional Neural Network라 부른다



Convolutional Neural Network: Overview

Multi-layer Neural Network와 다르게

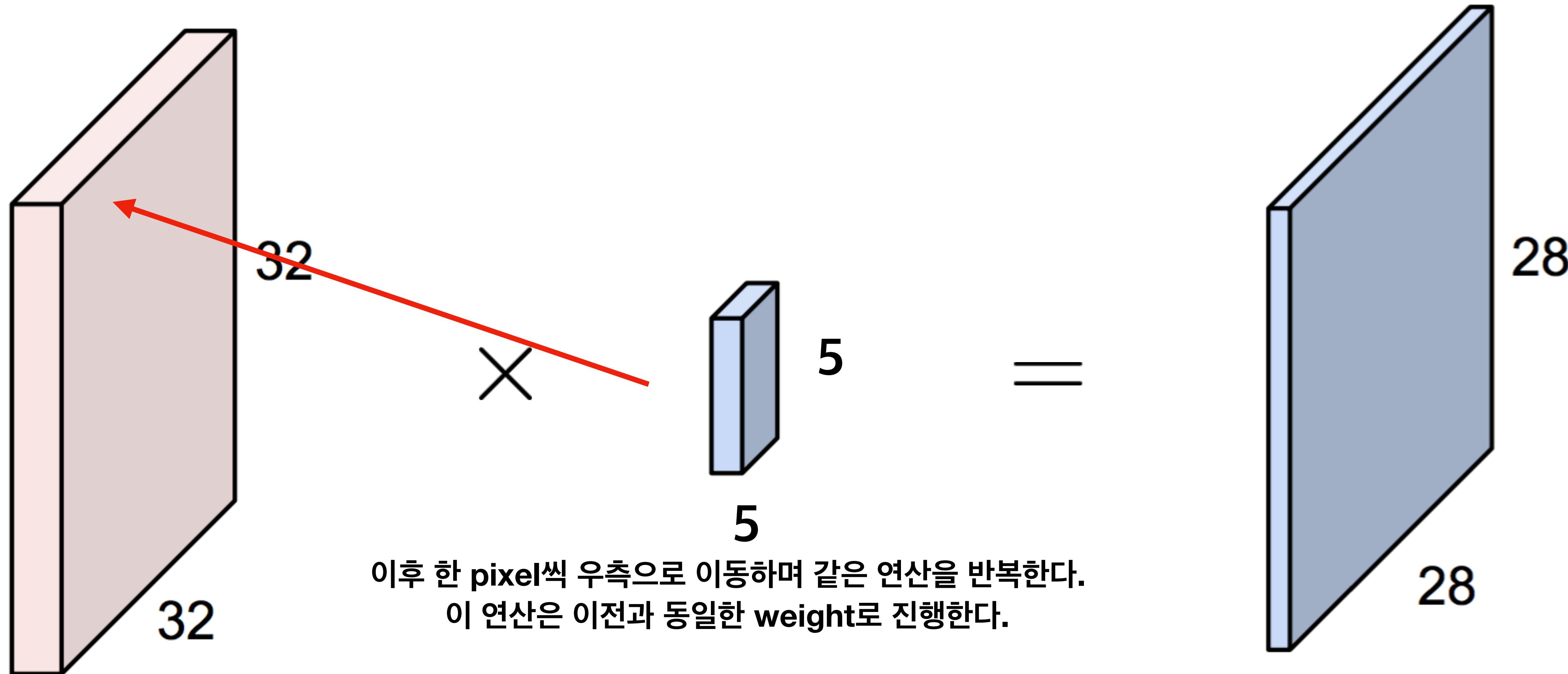
1) 이미지의 전체가 아닌 주변 픽셀만 계산하고, 2) 모든 연산마다 weight를 공유하면 이 문제를 해결할 수 있다.



Convolutional Neural Network: Overview

Multi-layer Neural Network와 다르게

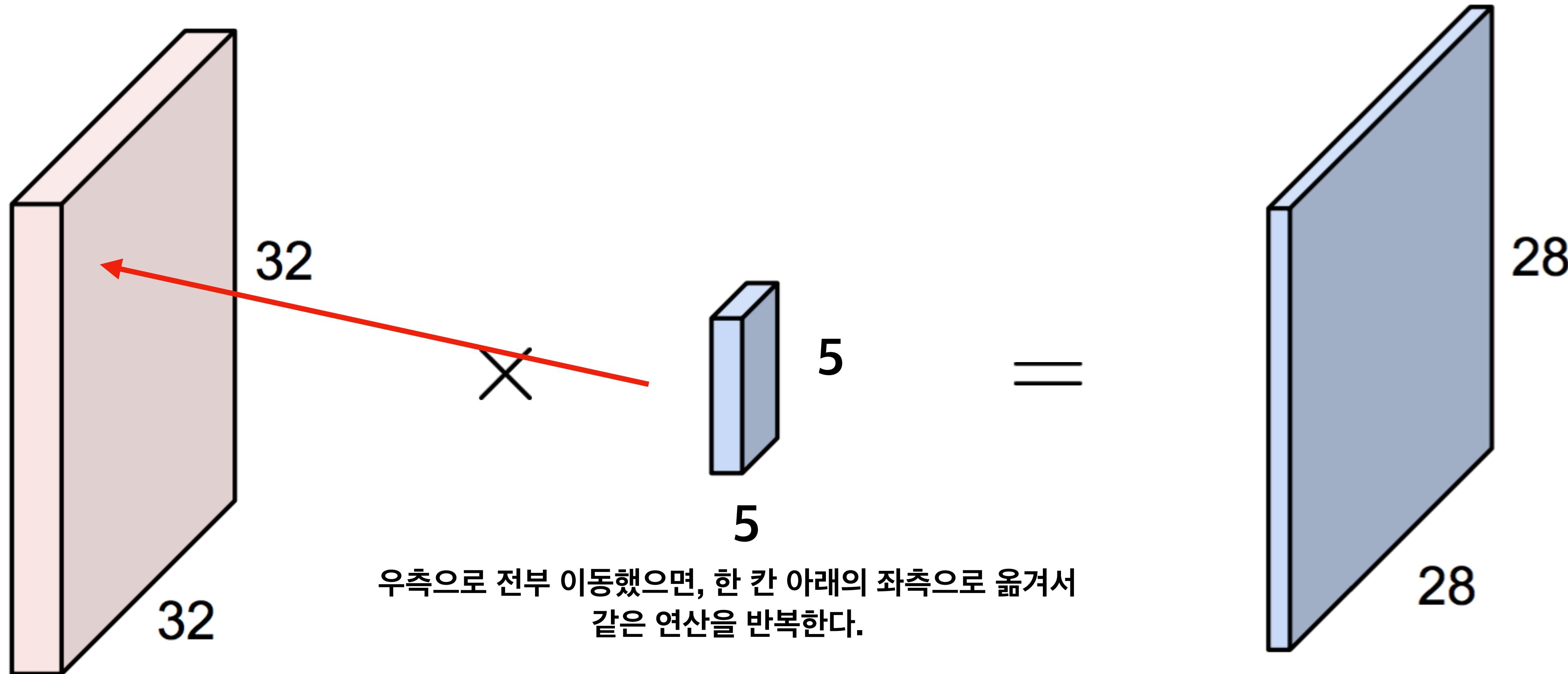
1) 이미지의 전체가 아닌 주변 픽셀만 계산하고, 2) 모든 연산마다 weight를 공유하면 이 문제를 해결할 수 있다.



Convolutional Neural Network: Overview

Multi-layer Neural Network와 다르게

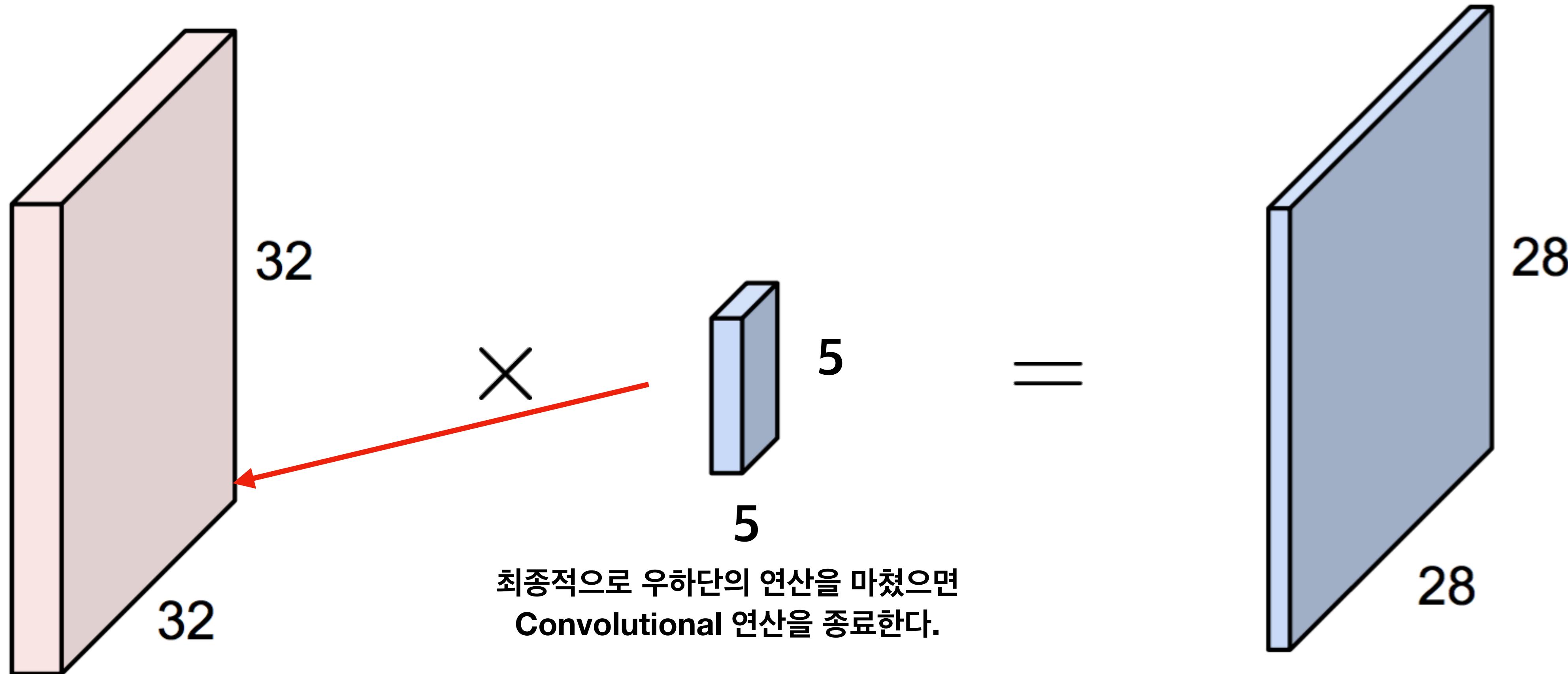
1) 이미지의 전체가 아닌 주변 픽셀만 계산하고, 2) 모든 연산마다 weight를 공유하면 이 문제를 해결할 수 있다.



Convolutional Neural Network: Overview

Multi-layer Neural Network와 다르게

1) 이미지의 전체가 아닌 주변 픽셀만 계산하고, 2) 모든 연산마다 weight를 공유하면 이 문제를 해결할 수 있다.



Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0	40

\times

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

?	?
?	?

Convolutional Layer: Example

Input (3x3)

0	0	80	0.25	40
20	0.5	40	1	0
0	0	40		

×

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

?	?
?	?

$$0 \times 0 + 0.25 \times 80 + 0.5 \times 20 + 1 \times 40 = ?$$

Convolutional Layer: Example

Input (3x3)

0	0	80	0.25	40
20	0.5	40	1	0
0	0	40		

×

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	?
?	?

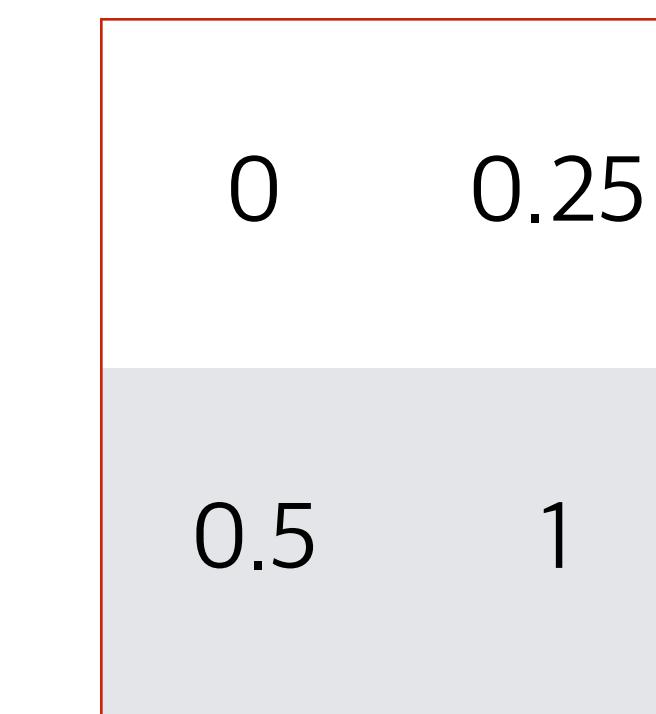
$$0 \times 0 + 0.25 \times 80 + 0.5 \times 20 + 1 \times 40 = 70$$

Convolutional Layer: Example

Input (3x3)



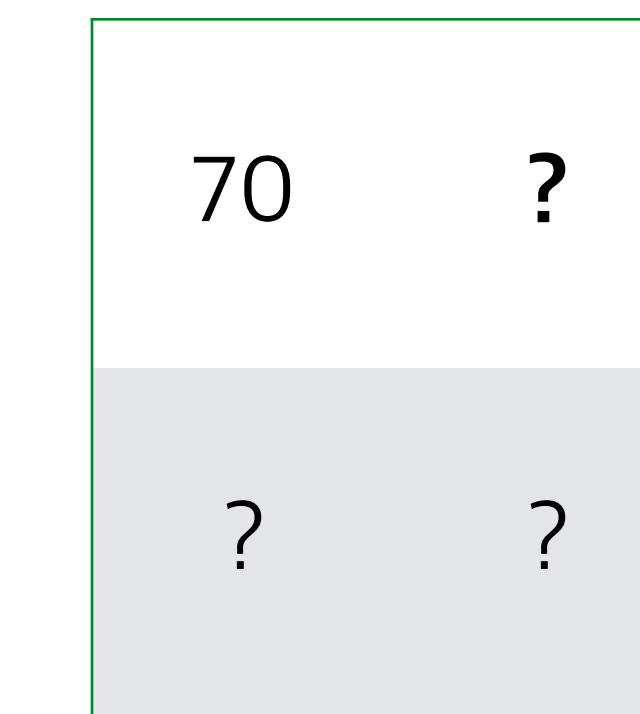
Weight (2x2)



×

=

Output (2x2)



$$0 \times 80 + 0.25 \times 40 + 0.5 \times 40 + 1 \times 0 = ?$$

Convolutional Layer: Example

Input (3x3)

0	80	0	40	0.25
20	40	0.5	0	1
0	0	40		
0				

×

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
?	?

$$0 \times 80 + 0.25 \times 40 + 0.5 \times 40 + 1 \times 0 = 30$$

Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0.25	0

0.5 0 1

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
?	?

$$0 \times 20 + 0.25 \times 40 + 0.5 \times 0 + 1 \times 0 = ?$$

Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0.25	0

A 3x3 input matrix with values: Row 1: 0, 80, 40; Row 2: 20, 40, 0; Row 3: 0, 0.25, 0. A red box highlights the central 2x2 submatrix [0, 0.25; 0.25, 0].

×

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
10	?

$$0 \times 20 + 0.25 \times 40 + 0.5 \times 0 + 1 \times 0 = 10$$

Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0	0.25

×

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
10	?

$$0 \times 40 + 0.25 \times 0 + 0.5 \times 0 + 1 \times 40 = ?$$

Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0	0.25

\times

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
10	40

$$0 \times 40 + 0.25 \times 0 + 0.5 \times 0 + 1 \times 40 = 40$$

Convolutional Layer: Example

Forward Propagation

Input (3x3)

0	80	40
20	40	0
0	0	40

\times

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
10	40

Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0	40

×

Weight (2x2)

0	0.25
0.5	1

=

Output (2x2)

70	30
10	40



Backward Propagation

Convolutional Layer: Example

Input (3x3)

0	80	40
20	40	0
0	0	40

×

Weight (2x2)

1	0.3
0.4	1

=

Output (2x2)

?	?
?	?

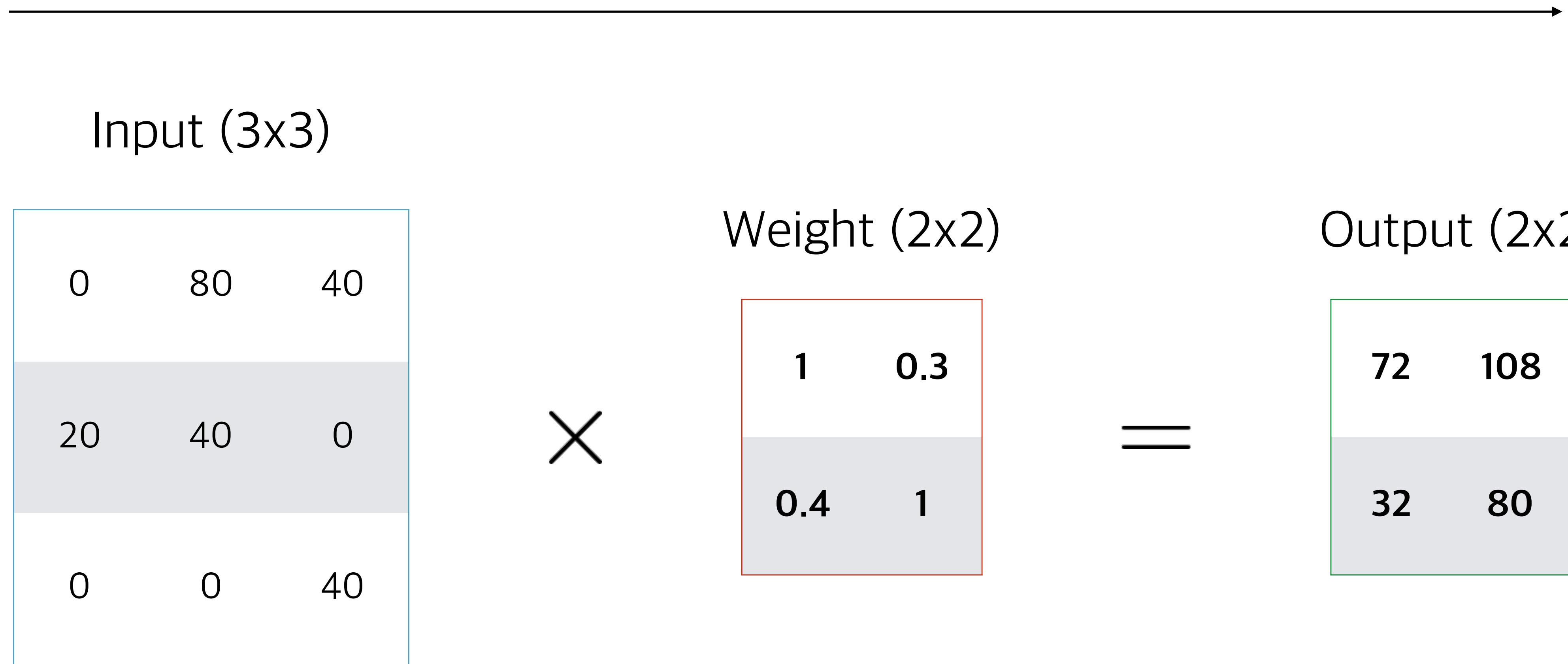
backpropagation을 통해
Layer의 weight를 업데이트한다



Backward Propagation

Convolutional Layer: Example

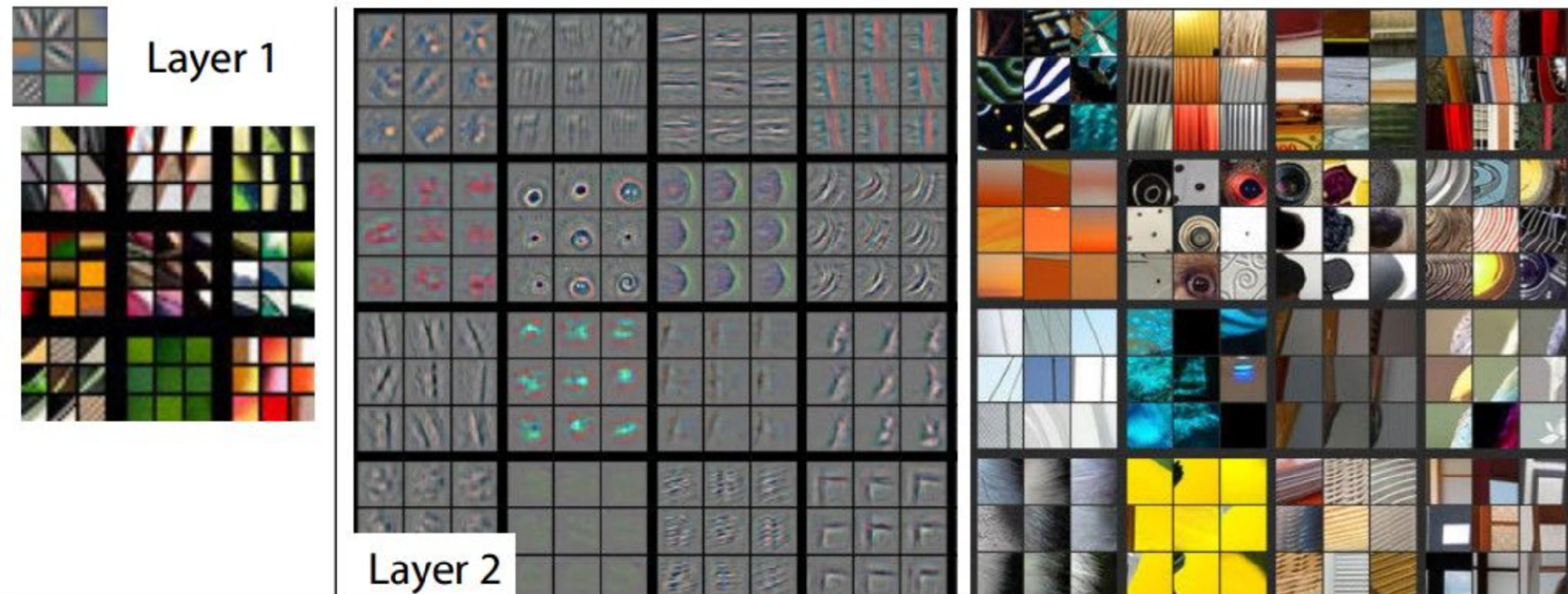
Forward Propagation



업데이트된 weight로 다시 forward/backward propagation을 반복
loss가 수렴해서 더이상 줄어들지 않을 때 까지 이를 반복하면서 weight를 업데이트한다

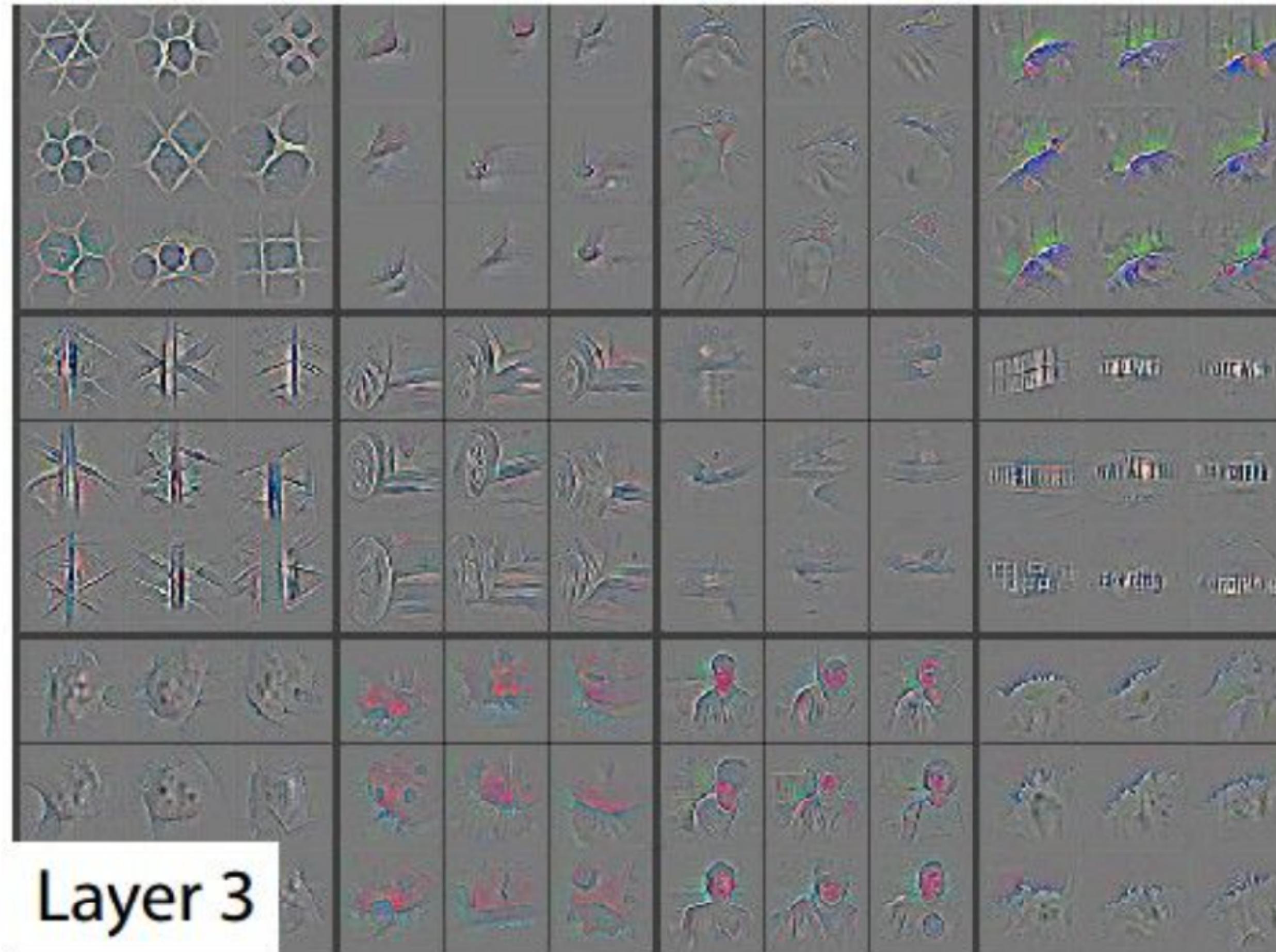
Convolutional Layer: Result

Convolutional Layer를 시각화하면, Layer가 어떤 Feature를 뽑아냈는지 알 수 있다.
앞의 레이어일수록 Linear한 결과를 뽑아낸다.



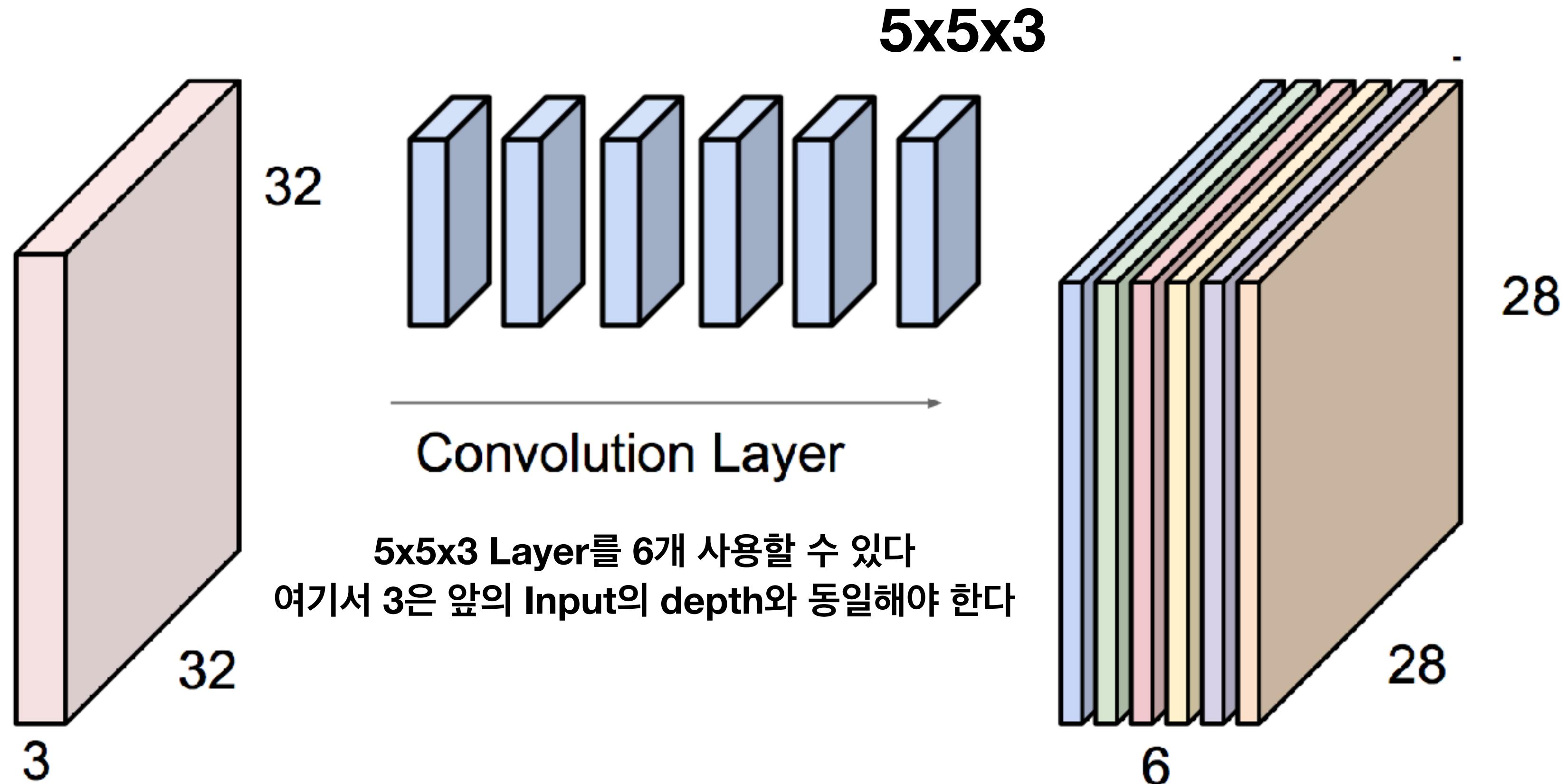
Convolutional Layer: Result

뒤의 레이어일수록 조금 더 구체적인 결과를 뽑아낸다.



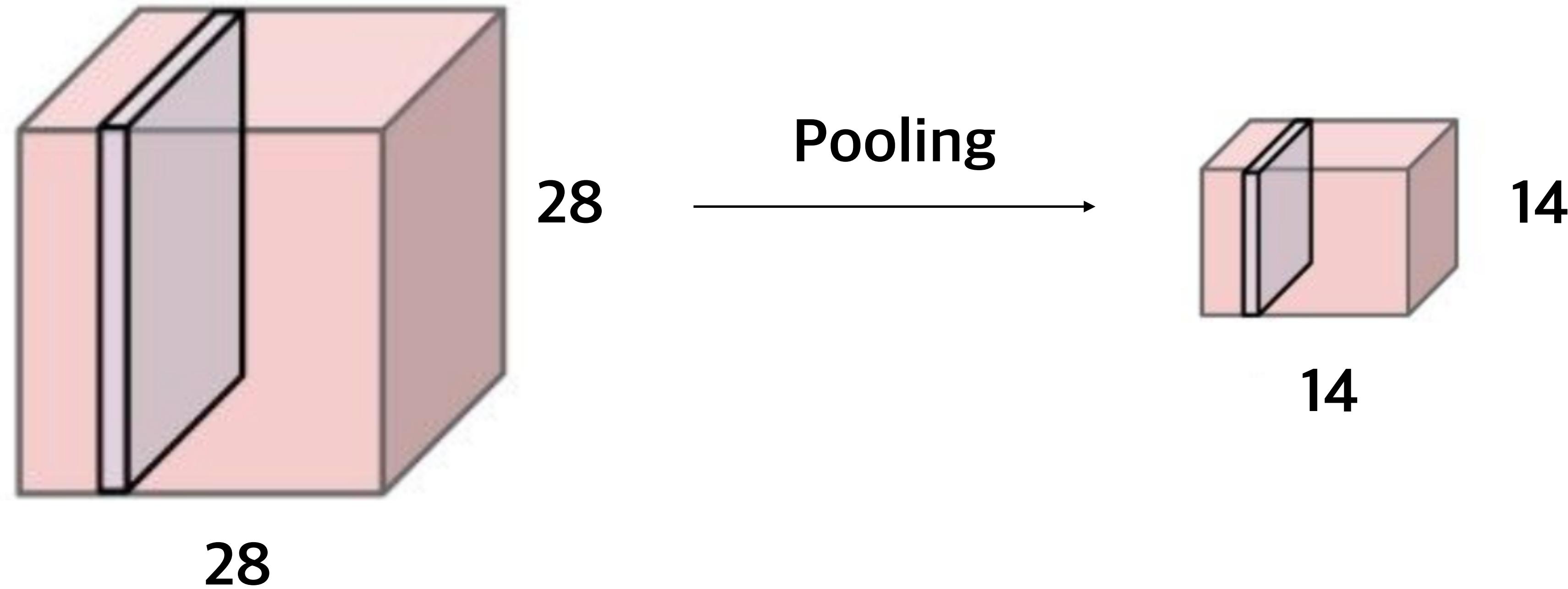
Convolutional Layer w/ Depth

실전에서는 여러개의 Convolutional Layer를 겹쳐서 하나로 사용한다.
이렇게 해야 한 번 연산할 때 많은 레이어를 사용할 수 있기 때문



Max Pooling: Overview

Pooling 범위에 들어온 값 중 가장 높은 값만 남기고 나머지는 버린다.
이렇게 하면 필요 없는 데이터를 빠르게 버릴 수 있고, Convolutional Layer로 하여금 중요한 값을 높게 주도록 만들 수 있다.



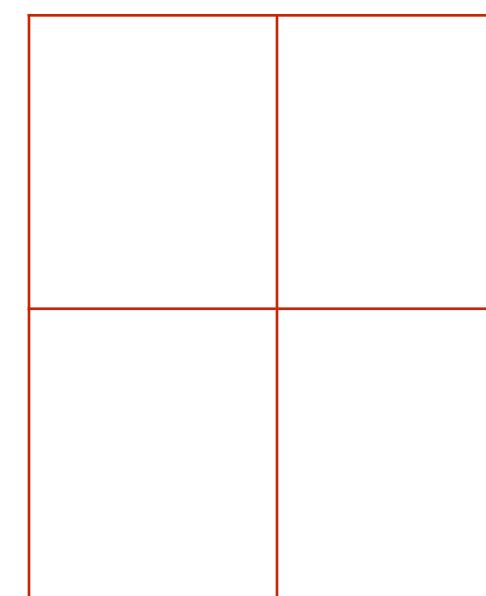
Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

×

Pooling (2x2)



=

Output (2x2)

?	?
?	?

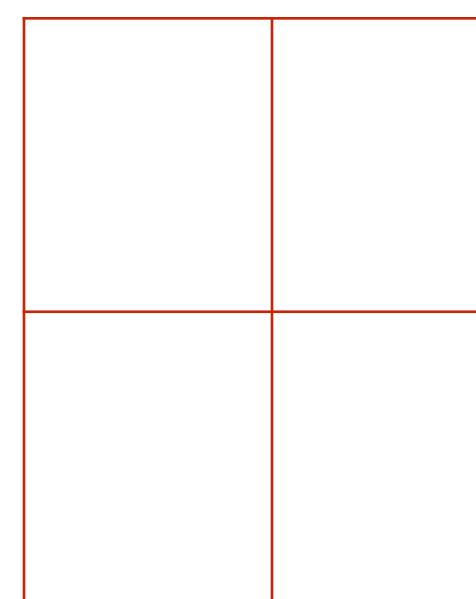
Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

×

Pooling (2x2)



=

Output (2x2)

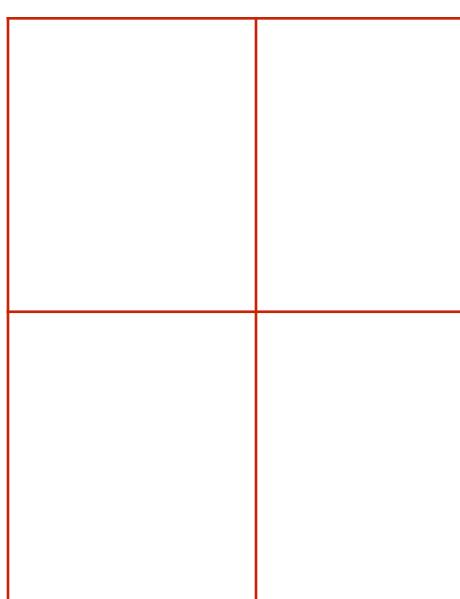
?	?
?	?

Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

Pooling (2x2)



×

Output (2x2)

0.19	?
?	?

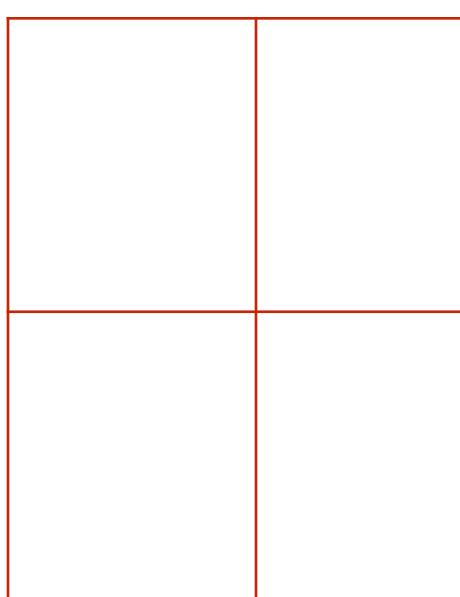
=

Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

Pooling (2x2)



×

Output (2x2)

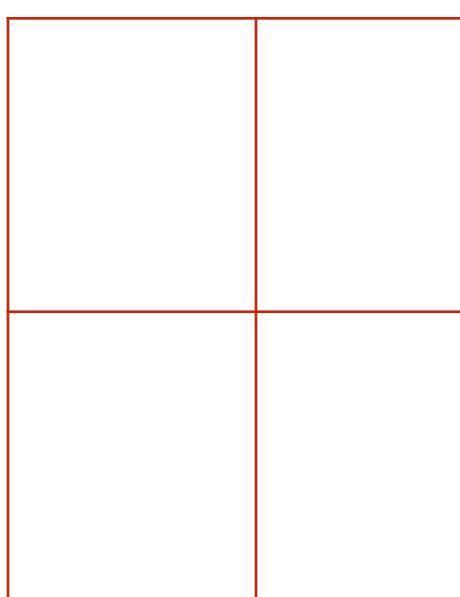
0.19	?
?	?

Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

Pooling (2x2)



×

Output (2x2)

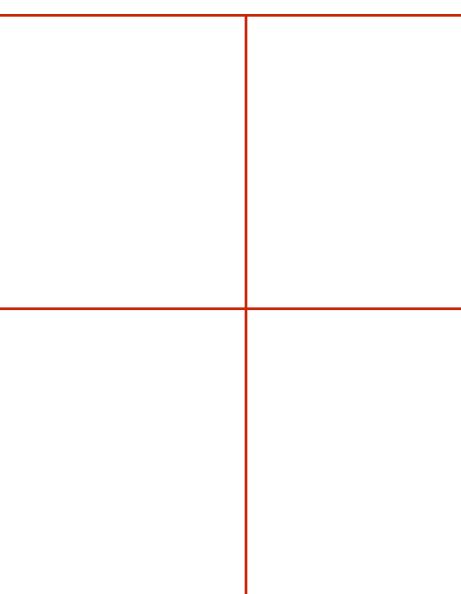
0.19	0.19
?	?

Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

Pooling (2x2)



×

Output (2x2)

0.19	0.19
?	?

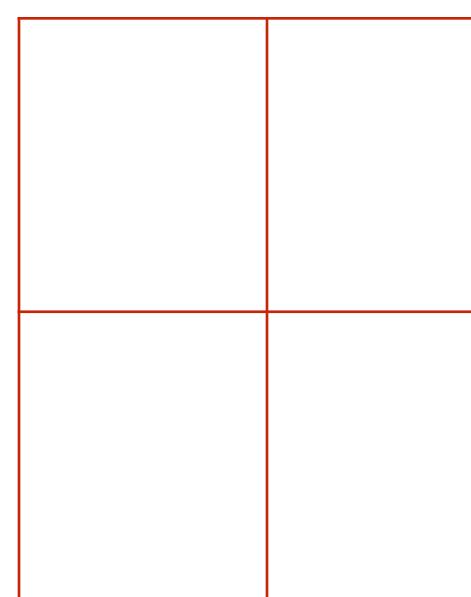
Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

×

Pooling (2x2)



=

Output (2x2)

0.19	0.19
0.75	?

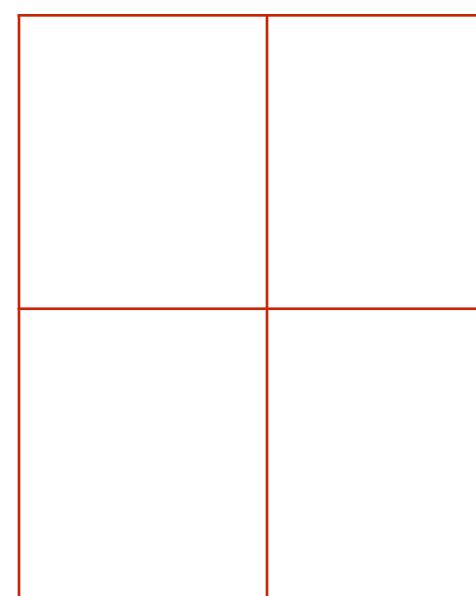
Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

×

Pooling (2x2)



=

Output (2x2)

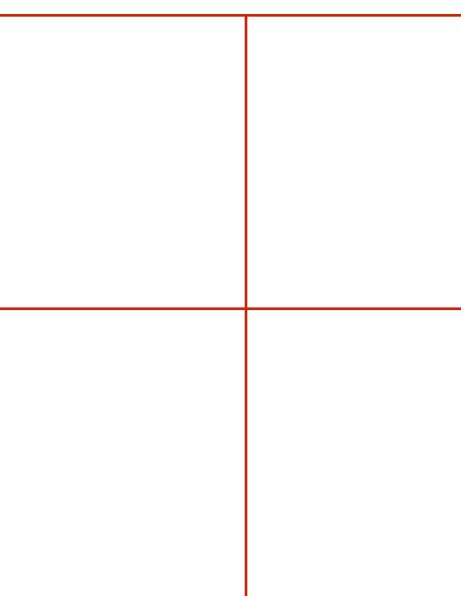
0.19	0.19
0.75	?

Max Pooling: Example

Data (4x4)

0.02	0.19	0.19	0.02
0.02	0.19	0.19	0.02
0.02	0.75	0.02	0.02
0.75	0.02	0.02	0.02

Pooling (2x2)



×

Output (2x2)

0.19	0.19
0.75	0.02

=

Convolutional Neural Network

Convolutional -> Activation(ex: Sigmoid) -> Convolutional -> Activation -> Max Pooling ⋯ -> Multi-layer Neural Network

