인공지능 심화학습 프로그램

Convolution Neural Network

Computer vision

Vision and Learning laboratory
김민지

2023.11.06 ~ 2022.11.09

소개

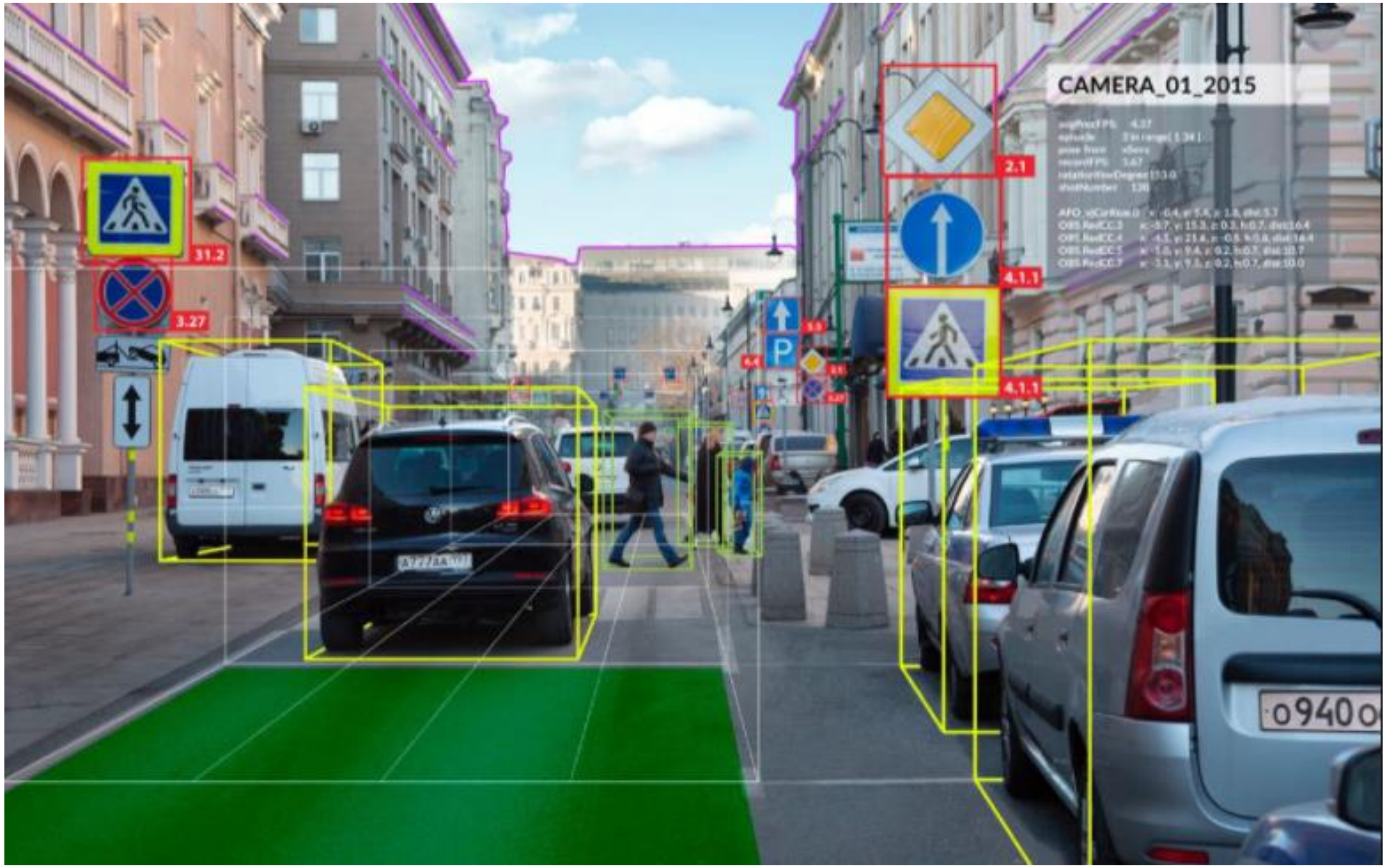
- 출처 : cs231n.stanford.edu 등

Index

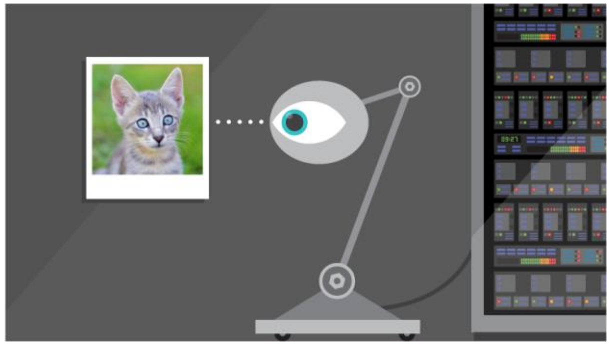
- What is computer vision?
- Let’s know Image *like RGB channel
- Convolution neural network example
- Filter
- Why Convolution neural network?
- Convolution neural network
- Convolution layer
- Stride, padding, pooling
- Receptive field
- ImageNet
- VGG, Resnet, Googlenet
- QnA & Training

What is computer vision?

- 컴퓨터가 보는 시야
- 컴퓨터에 들어갈 이미지, 영상 등을 분석해 의미있는 정보를 도출하고 활용하는 분야



출처 : https://medium.com/@miccowang/computer-vision-the-closet-thing-to-ai-on-our-personal-device-d2ff63994856



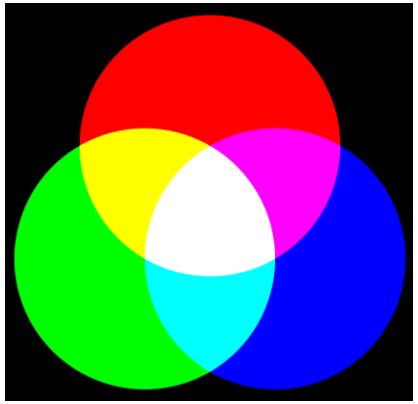
출처 : https://www.annalect.com/7-ways-computer-vision-helps-marketers-see-better-performance/

What is computer vision?

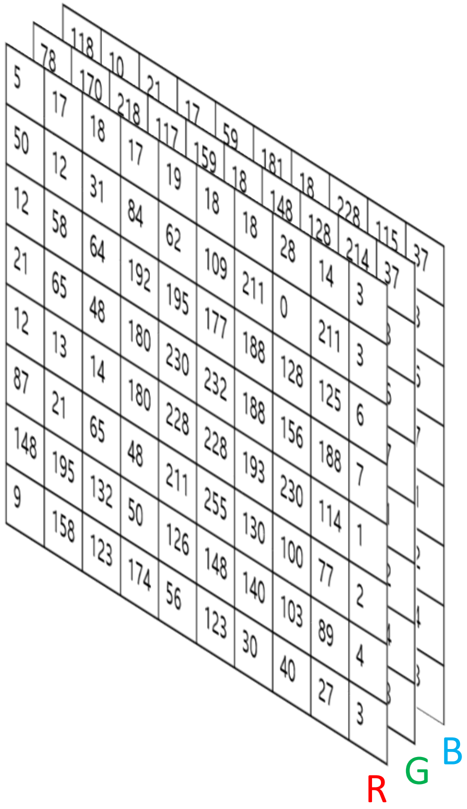
- 컴퓨터가 사물을 보기 위해선 컴퓨터가 알아볼 수 있게 저장해야 함
  - 저장 방식에는 다양한 방법이 있음
    - 일반적인 이미지는 RGB 채널을 이용하여 0~255 사이의 숫자를 집어 넣어서 이미지를 숫자화함
    - 빛의 삼원색인 R, G, B 색을 가산 혼합(섞을수록 밝아짐)하여 색을 표현함

FFFFFF	FFCCFF	FF99FF	FF66FF	FF33FF	FF00FF	66FFFF	66CCFF	6699FF	6666FF	6633FF	6600FF	EEEEEE
FFFFCC	FFCCCC	FF99CC	FF66CC	FF33CC	FF00CC	66FFCC	66CCCC	6699CC	6666CC	6633CC	6600CC	DDDDDD
FFFF99	FFCC99	FF9999	FF6699	FF3399	FF0099	66FF99	66CC99	669999	666699	663399	660099	CCCCCC
FFFF66	FFCC66	FF9966	FF6666	FF3366	FF0066	66FF66	66CC66	669966	666666	663366	660066	BBBBBB
FFFF33	FFCC33	FF9933	FF6633	FF3333	FF0033	66FF33	66CC33	669933	666633	663333	660033	AAAAAA
FFFF00	FFCC00	FF9900	FF6600	FF3300	FF0000	66FF00	66CC00	669900	666600	663300	660000	999999
CCFFFF	CCCCFF	CC99FF	CC66FF	CC33FF	CC00FF	33FFFF	33CCFF	3399FF	3366FF	3333FF	3300FF	888888
CCFFCC	CCCCCC	CC99CC	CC66CC	CC33CC	CC00CC	33FFCC	33CCCC	3399CC	3366CC	333CCC	3300CC	777777
CCFF99	CCC99	CC9999	CC6699	CC3399	CC0099	33FF99	33CC99	339999	336699	333399	330099	666666
CCFF66	CCCC66	CC9966	CC6666	CC3366	CC0066	33FF66	33CC66	339966	336666	333366	330066	555555
CCFF33	CCCC33	CC9933	CC6633	CC3333	CC0033	33FF33	33CC33	339933	336633	333333	330033	444444
CCFF00	CCCC00	CC9900	CC6600	CC3300	CC0303	33FF00	33CC00	339900	336600	333300	330000	333333
99FFFF	99CCFF	9999FF	9966FF	9933FF	9900FF	00FFFF	00CCFF	0099FF	0066FF	0033FF	0000FF	222222
99FFCC	99CCCC	9999CC	9966CC	9933CC	9900CC	00FFCC	00CCCC	0099CC	0066CC	0033CC	0000CC	111111
99FF99	99CC99	999999	996699	993399	990099	00FF99	00CC99	009999	006699	003399	000099	000000
99FF66	99CC66	999966	996666	993366	990066	00FF66	00CC66	009966	006666	003366	000066	FF0000
99FF33	99CC33	999933	996633	993333	990033	00FF33	00CC33	009933	006633	003333	000033	EE0000
99FF00	99CC00	999900	996600	993300	990000	00FF00	00CC00	009900	006600	003300	000002	DD0000

출처 : https://qits.tistory.com/entry/RGB%EC%83%89%EC%83%81%ED%91%9C

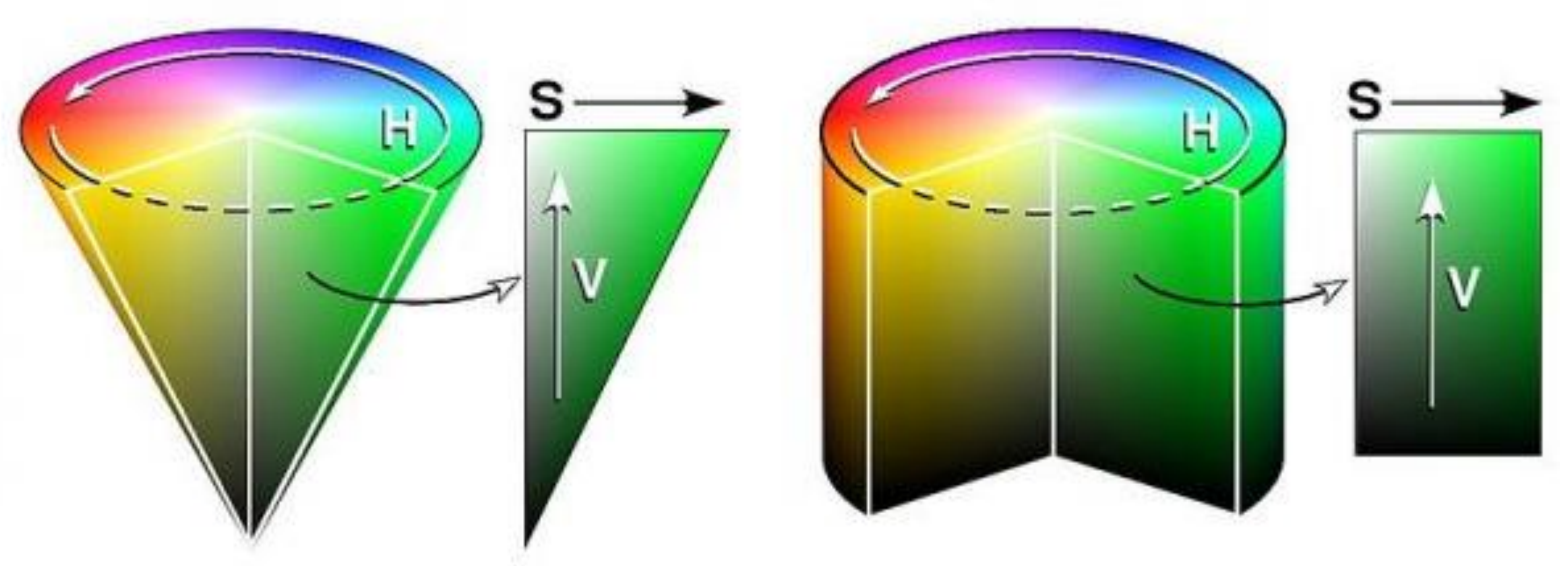


출처 : https://www.charlezz.com/?p=44906



What is computer vision?

- 컴퓨터가 사물을 보기 위해선 컴퓨터가 알아볼 수 있게 저장해야 함
  - 저장 방식에는 다양한 방법이 있음
    - HSV는 색조(Hue), 채도(Saturation), 명도(Value)로 색을 표현하는 방법



출처 : https://www.charlezz.com/?p=44906

Convolution neural network example

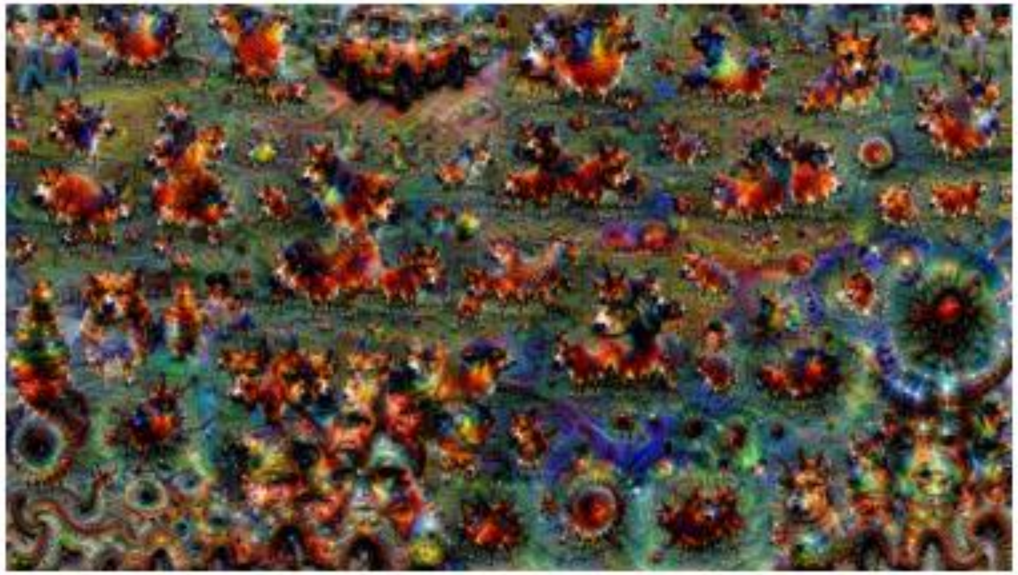
Classification

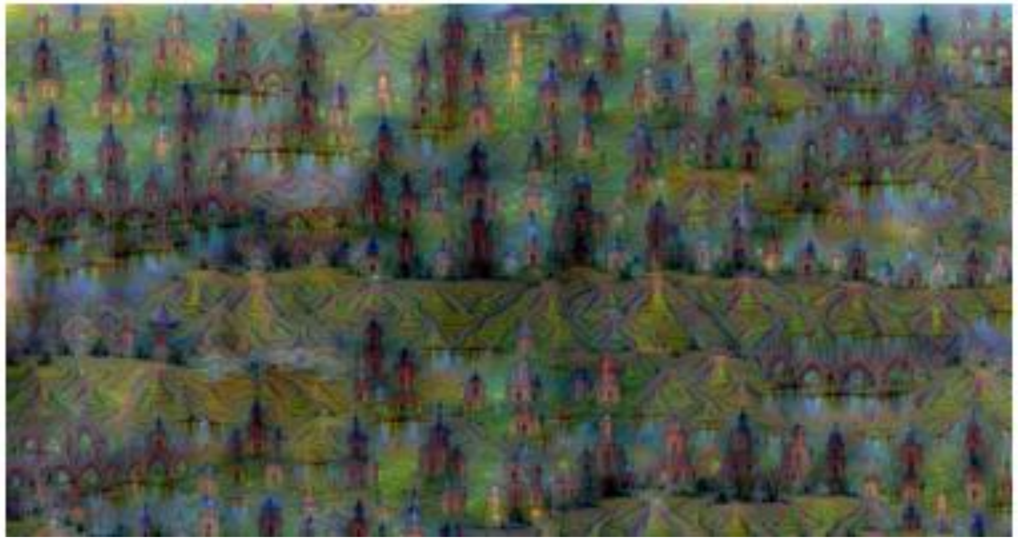
Retrieval

Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Convolution neural network example

Convolution neural network example





Filter?

Original
Bike

Bike blurred horizontally
Filter impulse response

$$\frac{1}{5}\begin{pmatrix} 1 & 1 & [1] & 1 & 1 \end{pmatrix}$$

Bike blurred vertically
Filter impulse response

$$\frac{1}{5}\begin{pmatrix} 1 \\ 1 \\ [1] \\ 1 \\ 1 \end{pmatrix}$$

Bike blurred by convolution
Impulse response „box filter‘

$$\frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & [1] & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Bike sharpened
Filter impulse response

$$\frac{1}{4}\begin{pmatrix} 0 & -1 & 0 \\ -1 & [8] & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Filter?

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

-1, 0, 1

image $f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $h[\cdot,\cdot]$

	0								

h[1, 1]

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

output — filter — image (signal)

Filter?

$g[\cdot,\cdot]$

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

-1, 0, 1

image $f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $h[\cdot,\cdot]$

	0								

h[1, 1]

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

output filter image (signal)

$$h[1,1] = \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[1+k, 1+l]$$

$$\begin{aligned} H[1,1] = {} & f[1+-1, 1+-1] + f[1+-1, 1+0] + f[1+-1, 1+1] \\ & + f[1+0, 1+-1] + f[1+0, 1+0] + f[1+0, 1+1] \\ & + f[1+1, 1+-1] + f[1+1, 1+0] + f[1+1, 1+1] \end{aligned}$$

Filter?

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $h[\cdot,\cdot]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

output   filter   image (signal)

Filter?

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $h[\cdot,\cdot]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\,f[m+k,n+l]$$

output — filter — image (signal)

Filter?

Original
Bike

18	8	9
19	12	7
13	14	30

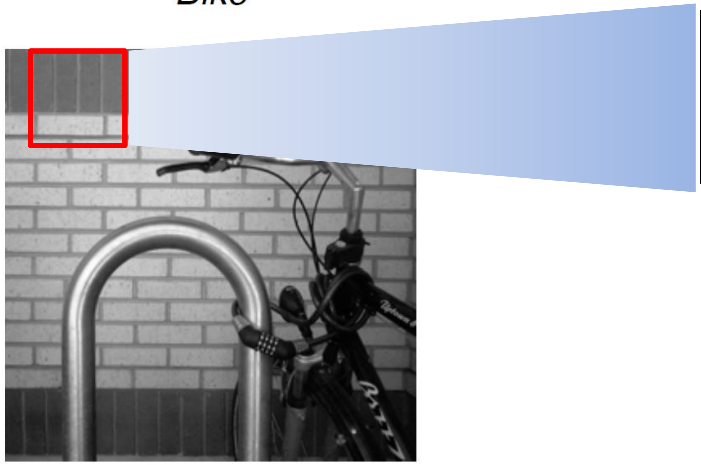
×

0	-1	0
-1	8	-1
0	-1	0

=

0	-8	0
-19	96	-7
0	-14	0

= 48



Original
Bike

8	9	6
12	7	17
14	30	12

×

0	-1	0
-1	8	-1
0	-1	0

=

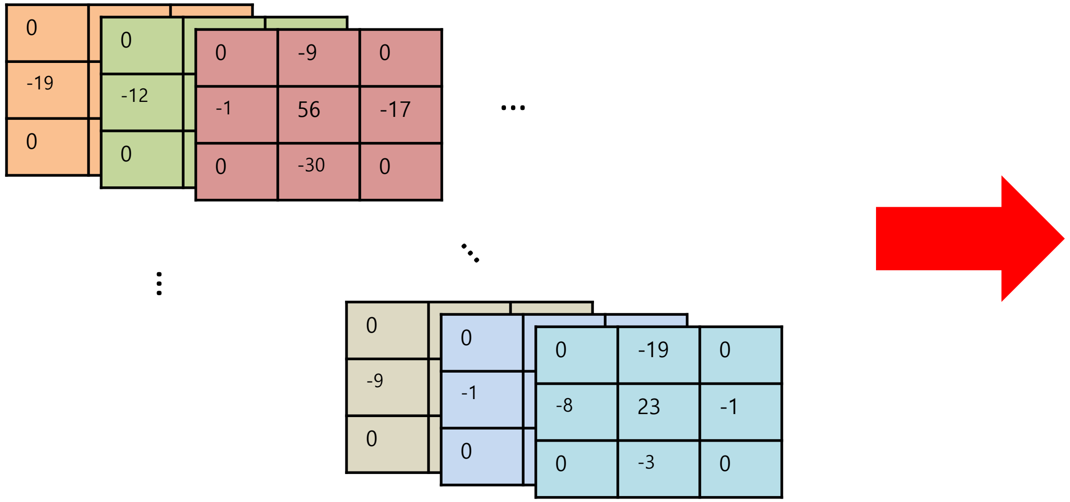
0	-9	0
-12	56	-17
0	-30	0

Bike sharpened
Filter impulse response

$$\frac{1}{4}\begin{pmatrix} 0 & -1 & 0 \\ -1 & [8] & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Filter?

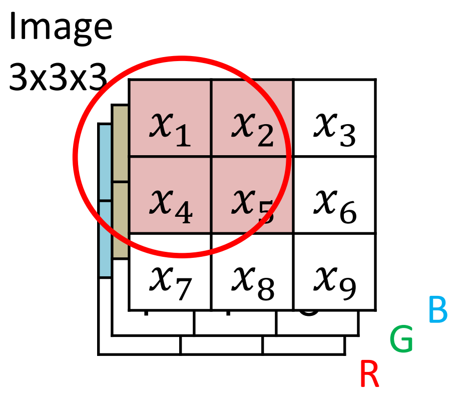
Original
Bike

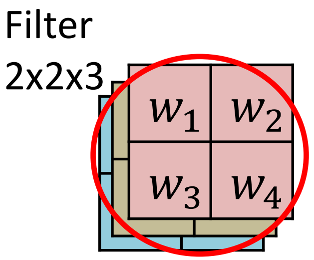


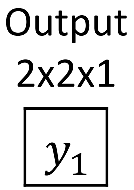
Filter

$$\frac{1}{4}\begin{pmatrix} 0 & -1 & 0 \\ -1 & [8] & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Filter?







$$y_1 = w_1x_1 + w_2x_2 + w_3x_4 + w_4x_5 \\ + w_5x_{10} + w_6x_{11} + w_7x_{13} + w_8x_{14} \\ + w_9x_{19} + w_{10}x_{20} + w_{11}x_{22} + w_{12}x_{23}$$

Filter?

Filter

$$\frac{1}{5}\begin{pmatrix} 1 & 1 & [1] & 1 & 1 \end{pmatrix}$$

Filter

$$\frac{1}{5}\begin{pmatrix} 1 \\ 1 \\ [1] \\ 1 \\ 1 \end{pmatrix}$$

Filter

$$\frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & [1] & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Filter

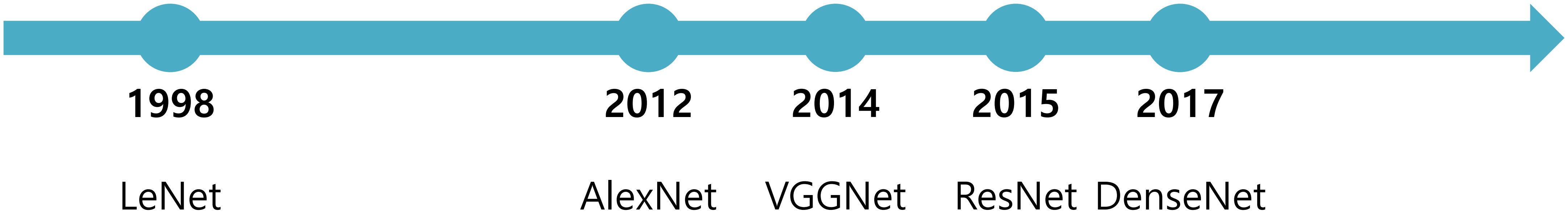
$$\frac{1}{4}\begin{pmatrix} 0 & -1 & 0 \\ -1 & [8] & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

사람이 정한 filter를 이용하면 사진에 특수한 효과를 넣을 수 있다.

그럼 기계가 filter를 자동으로 찾아내고 특수한 filter를 만들 순 없을까?

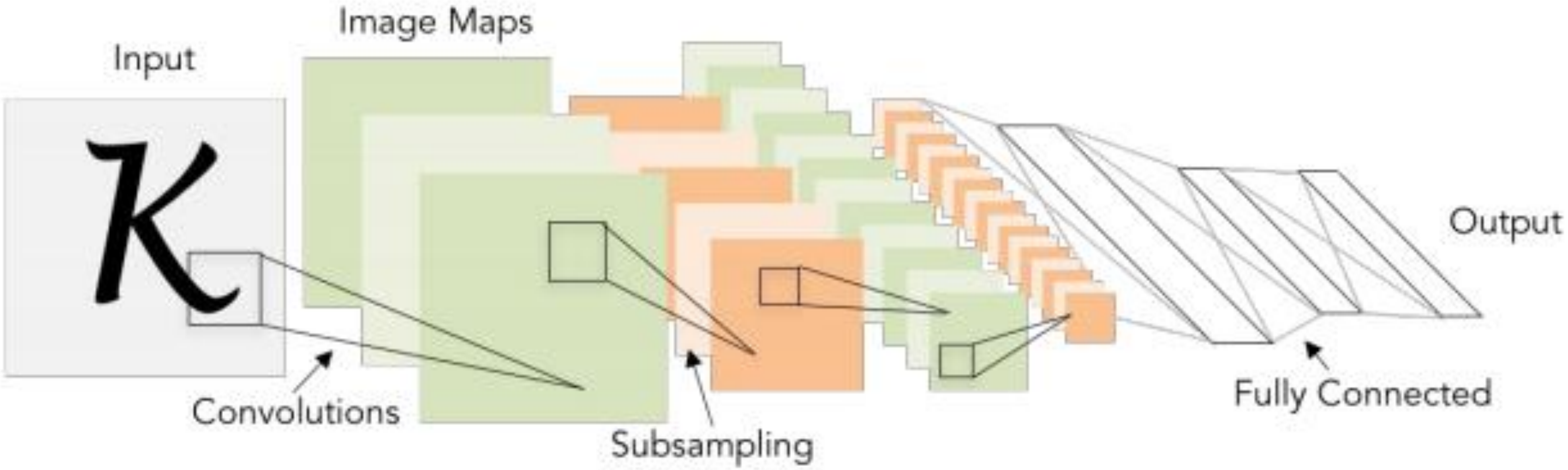
Ex) 눈을 찾는 filter, 모서리를 찾는 filter, 사람이 알지 못하는 특수한 차원의 filter

Convolutional Neural Network



Convolutional Neural Network

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

Convolution

- 합성곱(Convolution)은 두 함수에 적용하여 새로운 함수를 만드는 수학 연산자

1. 배열 하나 선택하여 뒤집기

두 배열 x와 w가 있다고 가정하고 두 배열 중 원소수가 적은 배열 w의 원소 순서를 뒤집어보자.

뒤집은 배열은 wr이라고 표현해보자.

x	2	8	3	7	1	2	0	4	5
w	2	1	5	3					
wr	3	5	1	2					

Convolution

- 합성곱(Convolution)은 두 함수에 적용하여 새로운 함수를 만드는 수학 연산자

2. 합성곱

각 배열 원소끼리 곱한 후 더한다.

x	2	8	3	7	1	2	0	4	5	
wr	3	5	1	2						**63**
		3	5	1	2					**48**
			3	5	1	2				**49**
				3	5	1	2			**28**
					3	5	1	2		**21**
						3	5	1	2	**20**

Convolution

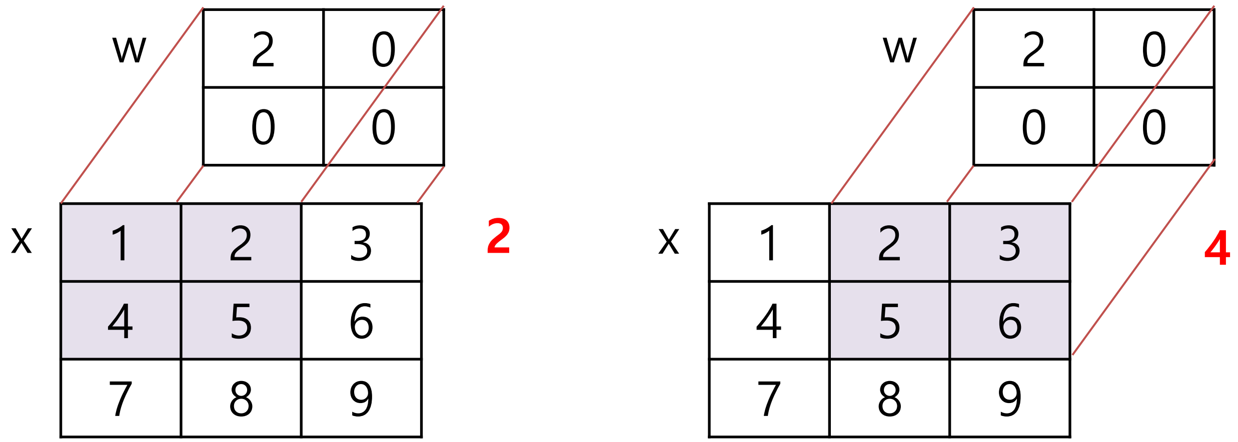
- CNN은 합성곱을 사용하지 않는다!
- 대부분의 딥러닝 패키지는 CNN을 만들 때 합성곱이 아니라 교차 상관을 사용한다.

- 교차상관은 합성곱과 동일한 방법으로 연산이 진행되지만 '배열을 뒤집지 않는' 점이 다르다.

x	2	8	3	7	1	2	0	4	5	
w	2	1	5	3						**48**
		2	1	5	3					**57**
			2	1	5	3				**24**
				2	1	5	3			**25**
					2	1	5	3		**16**
						2	1	5	3	**39**

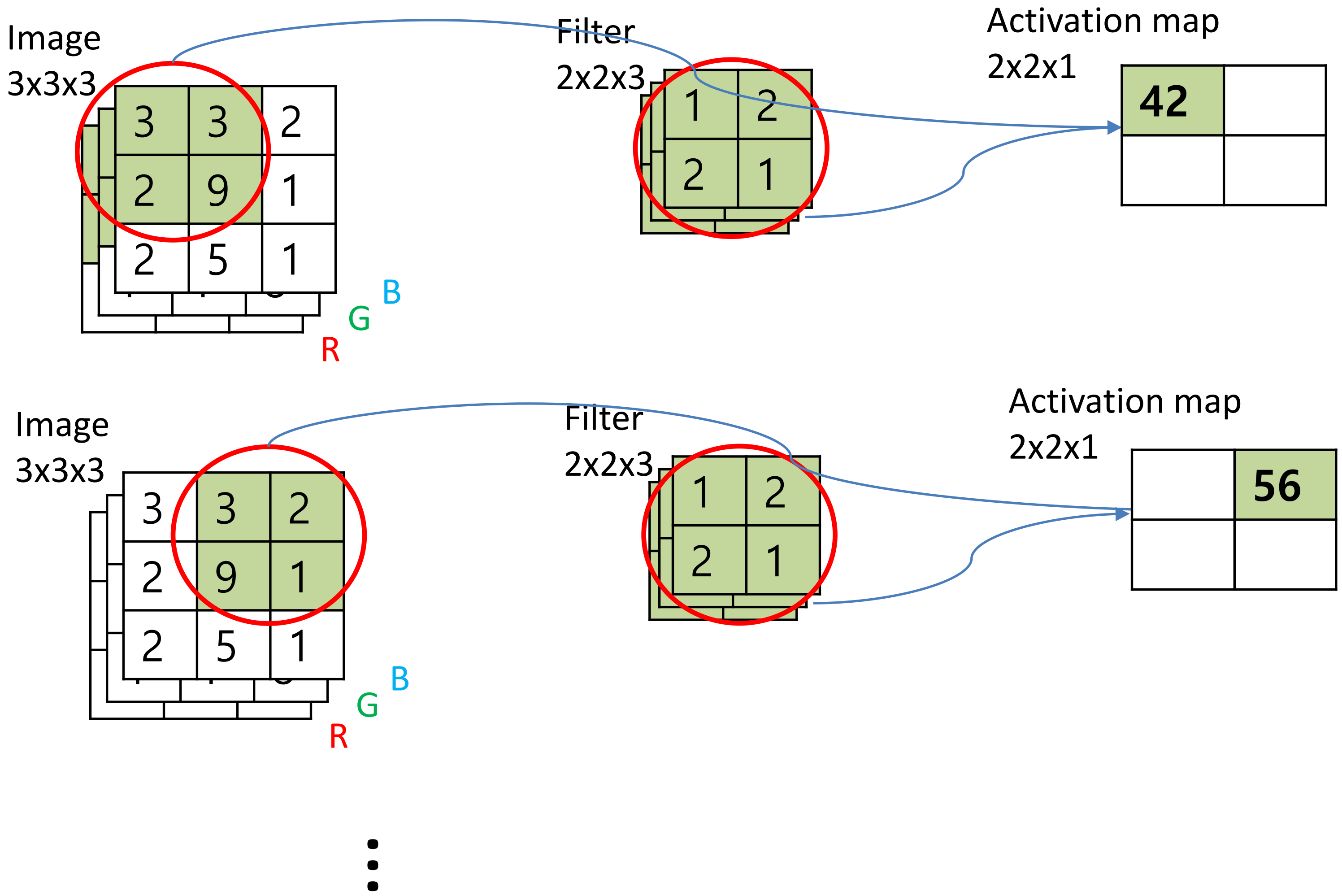
Convolution

- CNN은 대부분 2차원 배열에 대한 합성곱을 사용함.
- 합성곱의 수행 방향은 원본 배열의 왼쪽에서 오른쪽, 위에서 아래로 이동하며 배열 원소끼리 곱한다.



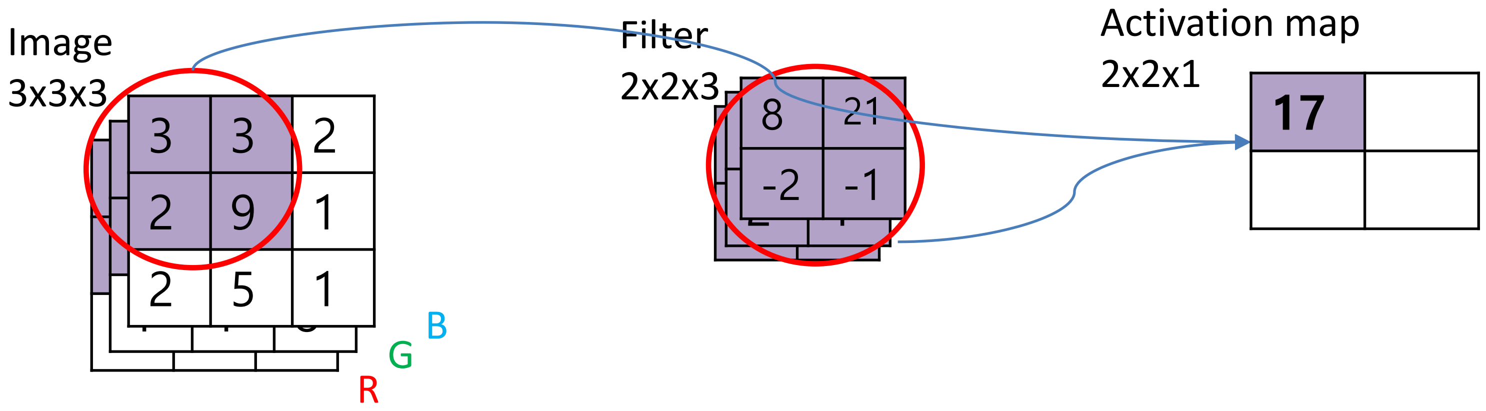
2	4
8	10

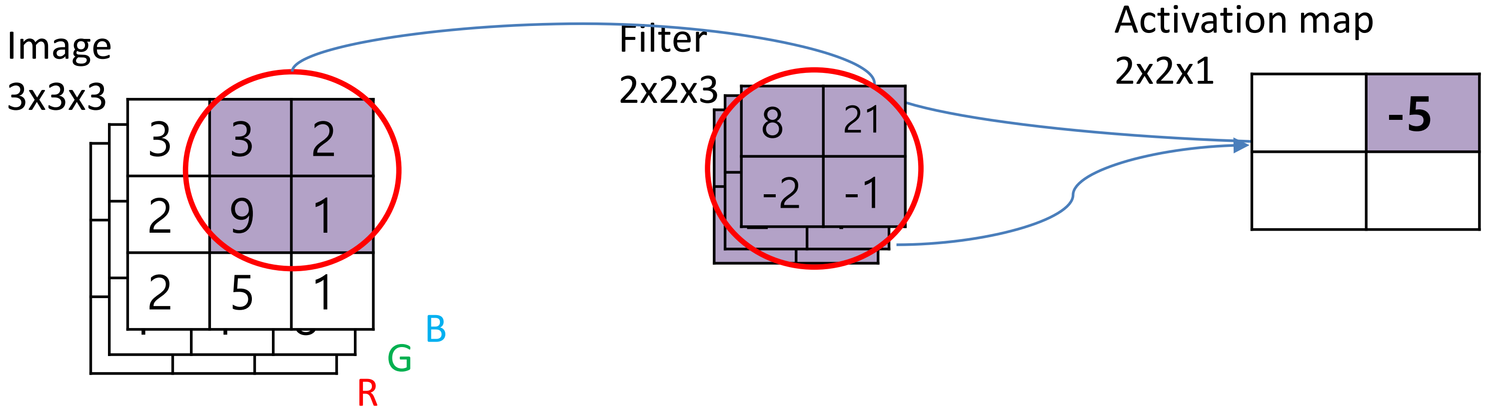
Convolution layer



42	56
17	95

Convolution layer

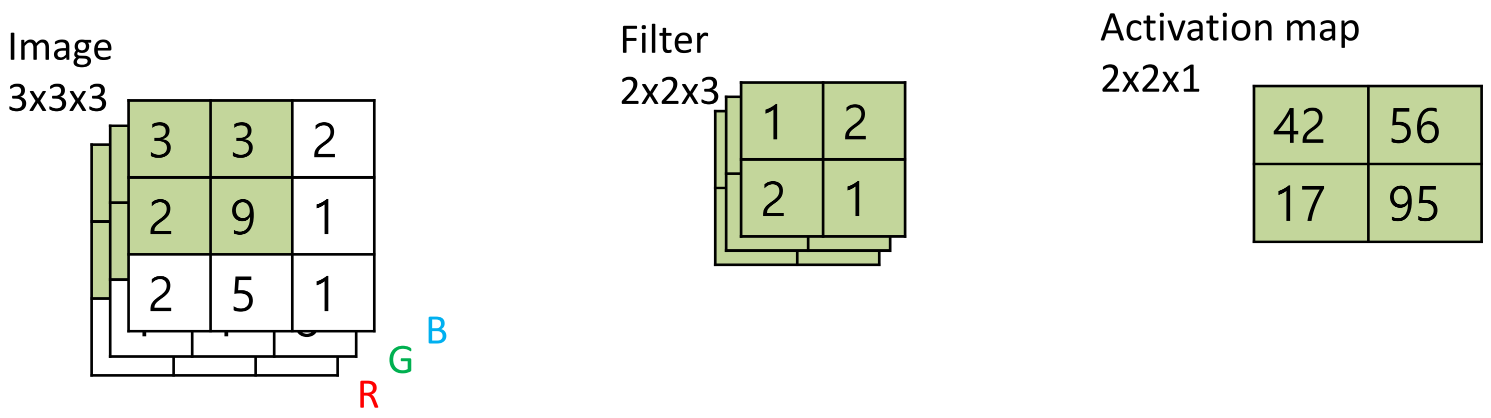


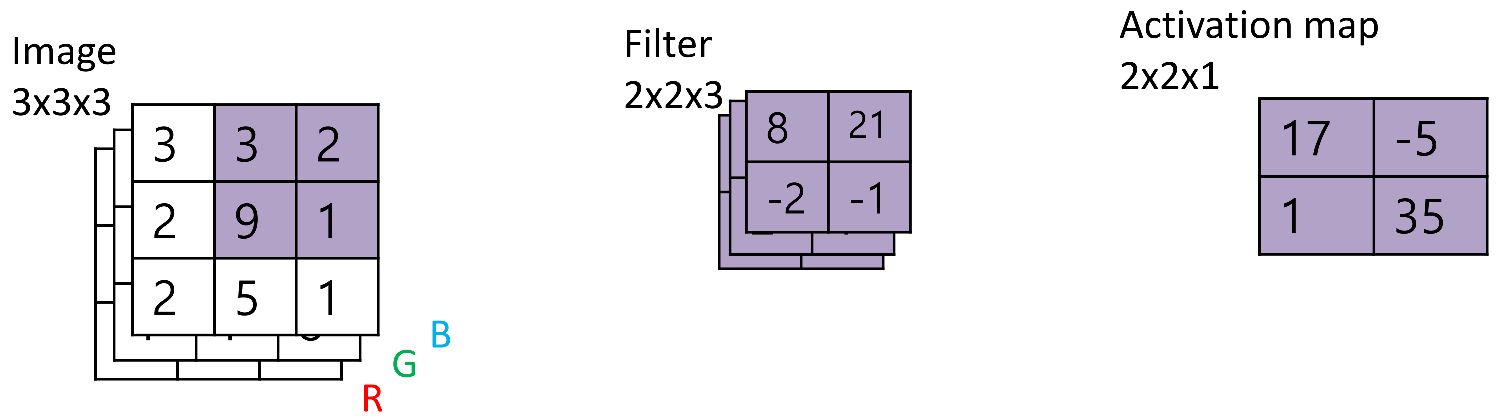


⋮

17	-5
1	35

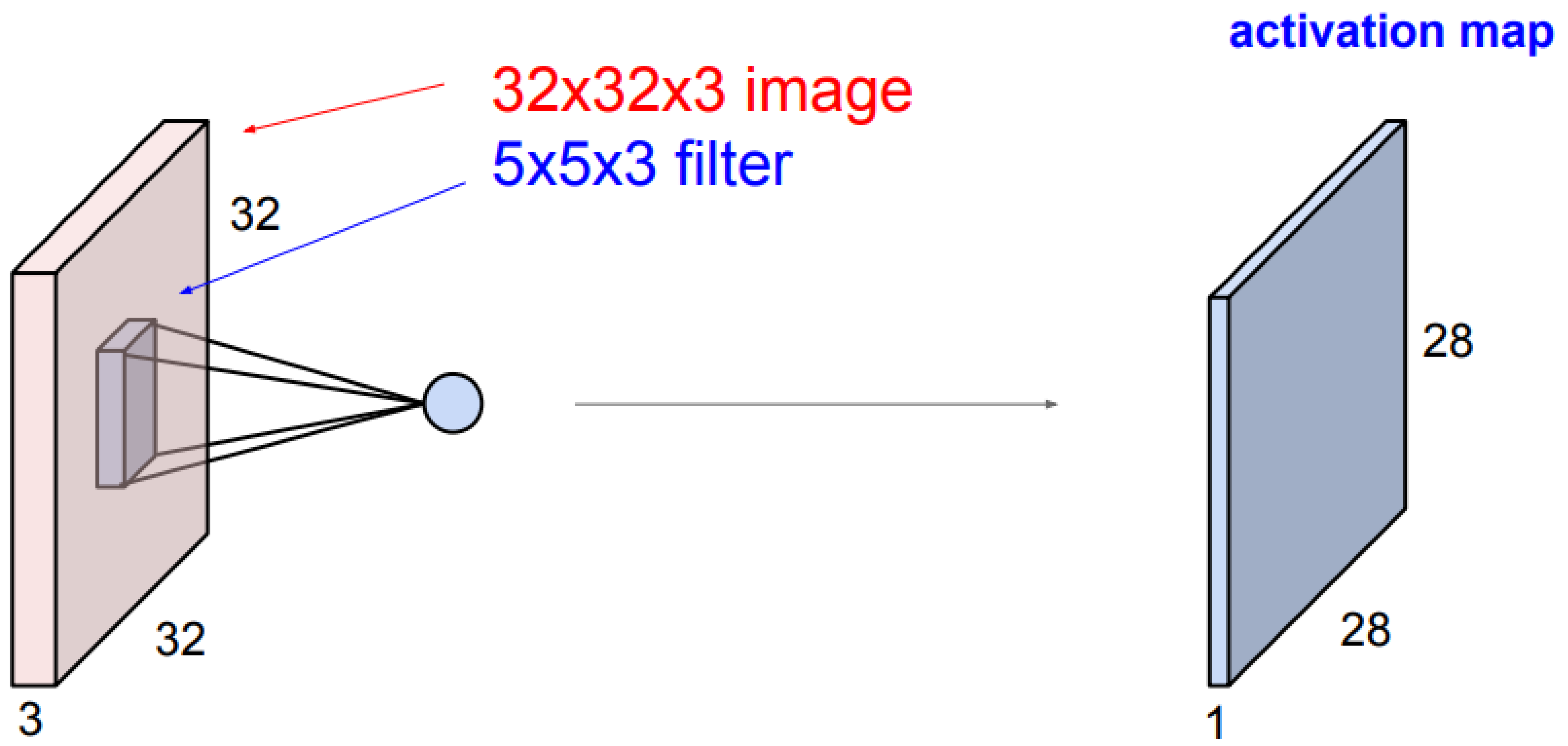
Convolution layer



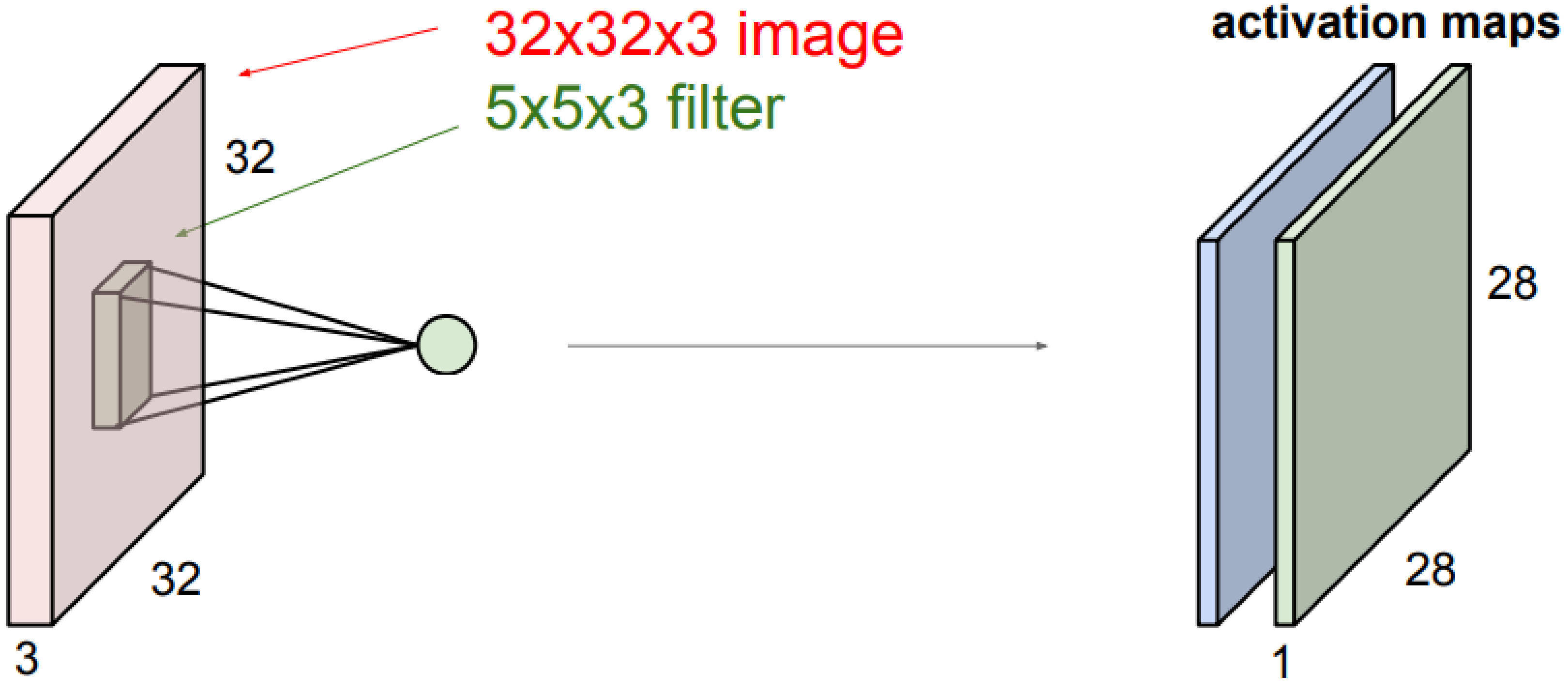


⋮

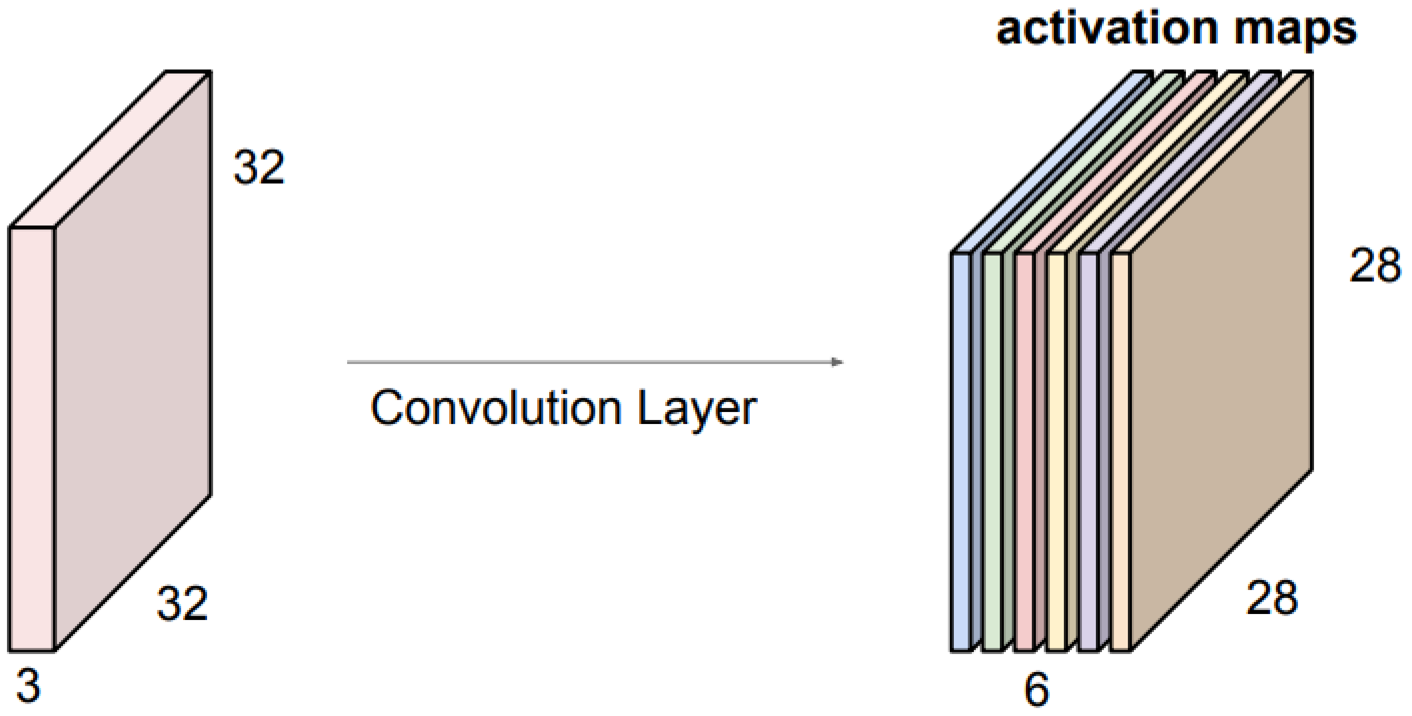
Convolution layer



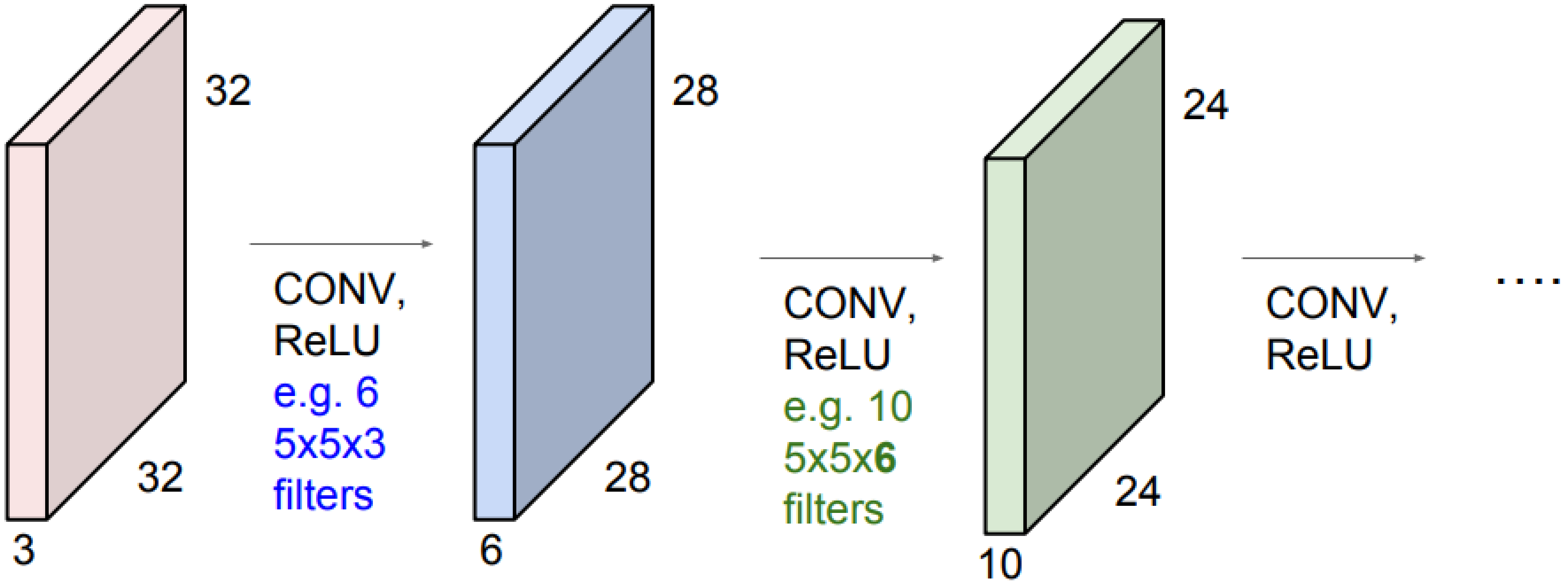
Convolution layer



Convolution layer

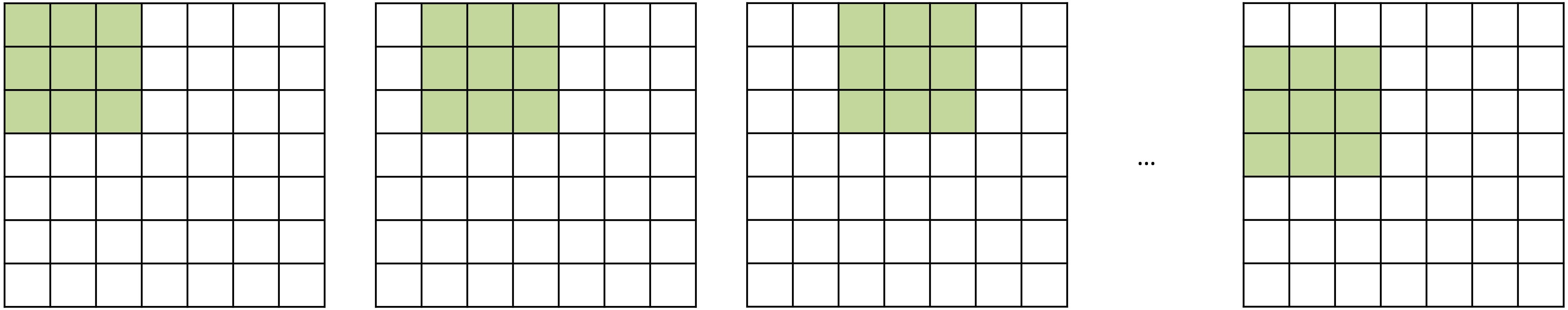


Convolution layer

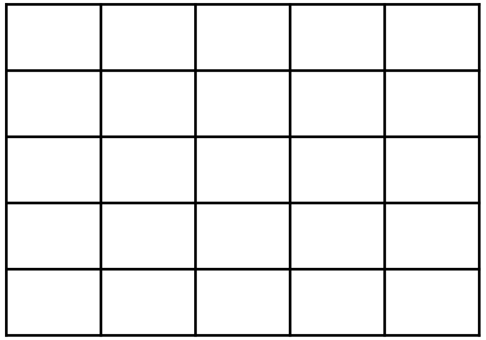


Convolution layer stride

7x7 image / 3x3 filter / stride 1

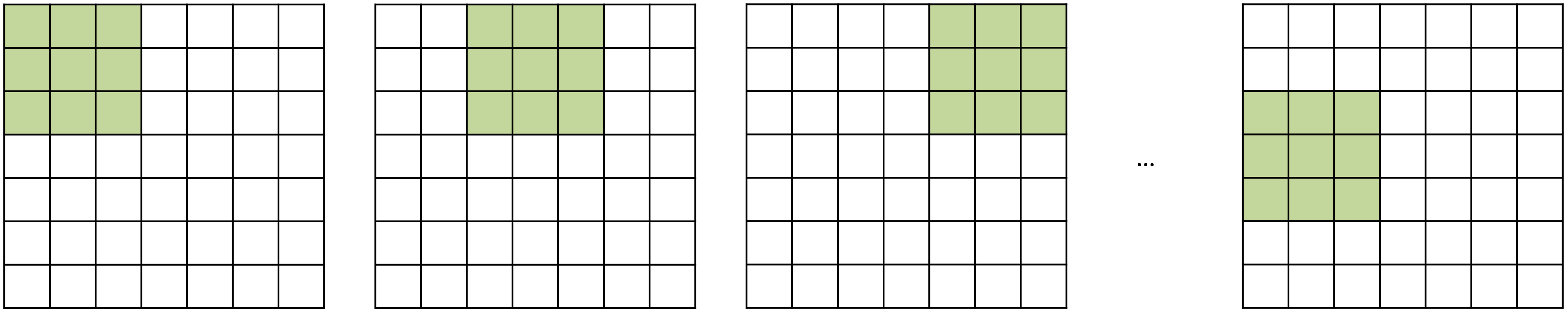


Output activation map
7x7 image / 3x3 filter / stride 1

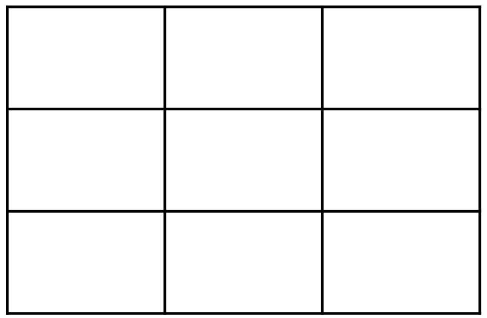


Convolution layer stride

7x7 image / 3x3 filter / stride 2

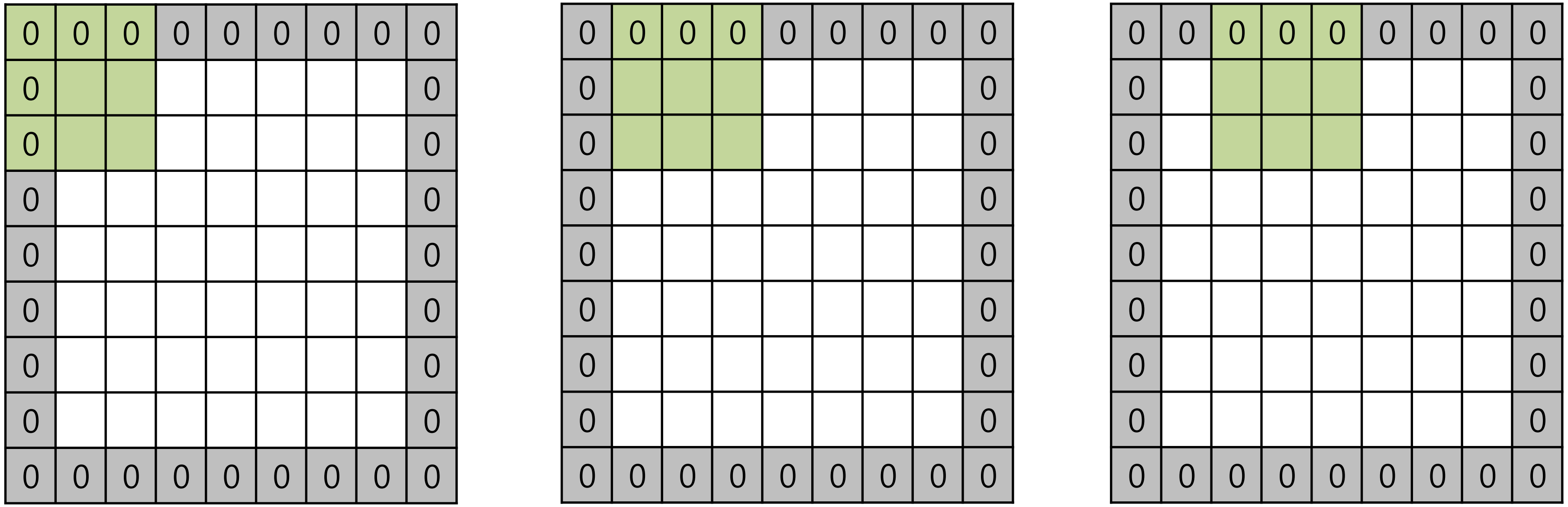


Output activation map
7x7 image / 3x3 filter / stride 2

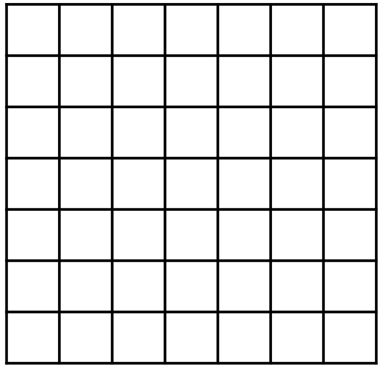


Convolution layer padding

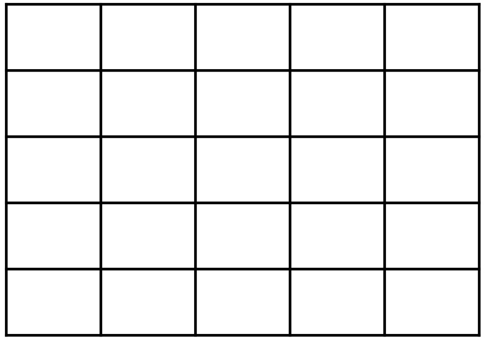
7x7 image / 3x3 filter / stride 1 / padding 1



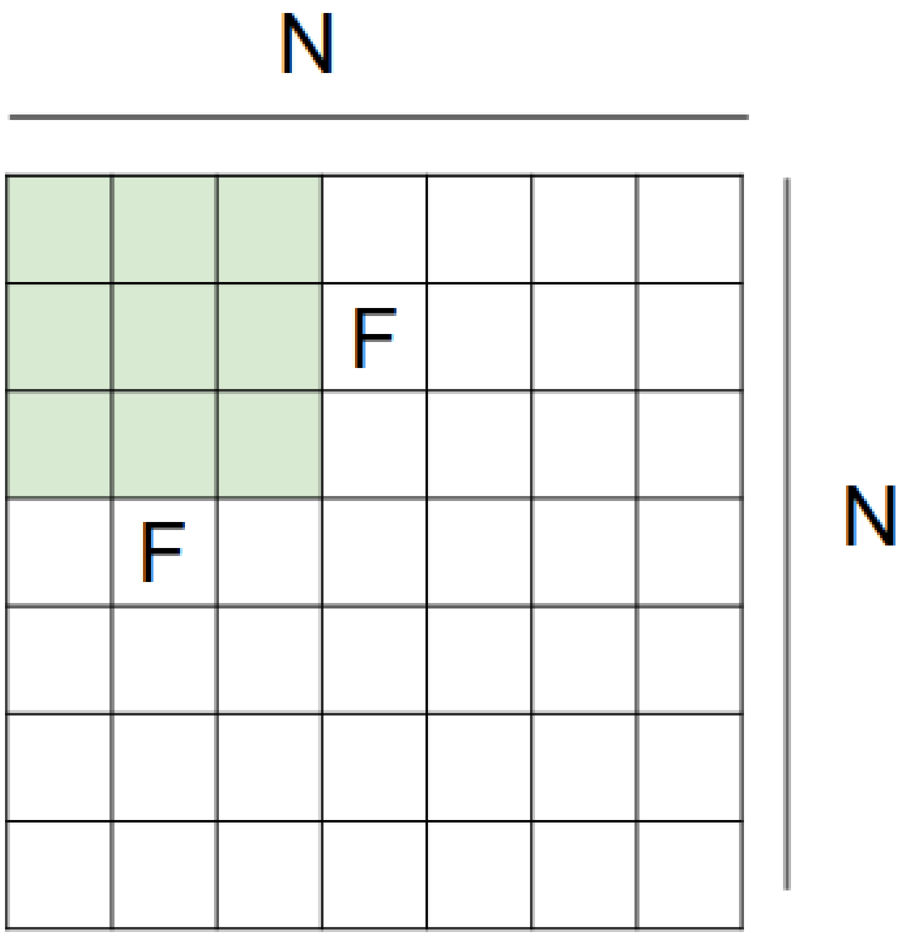
Output activation map
7x7 image / 3x3 filter / stride 1 / padding 1



Output activation map
7x7 image / 3x3 filter / stride 2 / padding 1



Filter size formula



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 = 2.33 :\

Convolution layer pooling

- 풀링이란 특성 맵을 스캔하여 최댓값을 고르거나 최솟값, 평균값을 계산하는 것을 말함.
- CNN에서는 최대 풀링 또는 평균 풀링을 주로 사용함.

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

6

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

8

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

14

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

16

6	8
14	16

Convolution layer pooling

- 풀링이란 특성 맵을 스캔하여 최댓값을 고르거나 최솟값, 평균값을 계산하는 것을 말함.
- CNN에서는 최대 풀링 또는 평균 풀링을 주로 사용함.

1. 최대 풀링 (max pooling)

- 특성 맵 위를 스캔하며 최댓값을 고른다.
- 풀링 영역의 크기는 보통 2x2를 지정함. (2x2 풀링은 특성 맵의 크기를 절반으로 줄인다. 면적은 25%)
- 특성 맵의 크기를 절반으로 줄이면 특성 맵의 한 요소가 더 넓은 영역을 대표하는 효과를 얻음.
- 일반적으로 스트라이드(stride)는 풀링의 한 모서리 크기로 지정함.

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

6

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

8

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

14

x

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

16

6	8
14	16

Convolution layer pooling

- 풀링이란 특성 맵을 스캔하여 최댓값을 고르거나 최솟값, 평균값을 계산하는 것을 말함.
- CNN에서는 최대 풀링 또는 평균 풀링을 주로 사용함.

2. 평균 풀링 (average pooling)

- 특성 맵 위를 스캔하며 평균값을 계산한다.
- 일반적으로 최대 풀링을 더 선호함. (평균 풀링은 특징을 희석시킬 수 있음)
- 평균 풀링은 최대 특성을 뭉갤 수 있기 때문에

X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

3.5

X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

5.5

X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

11.5

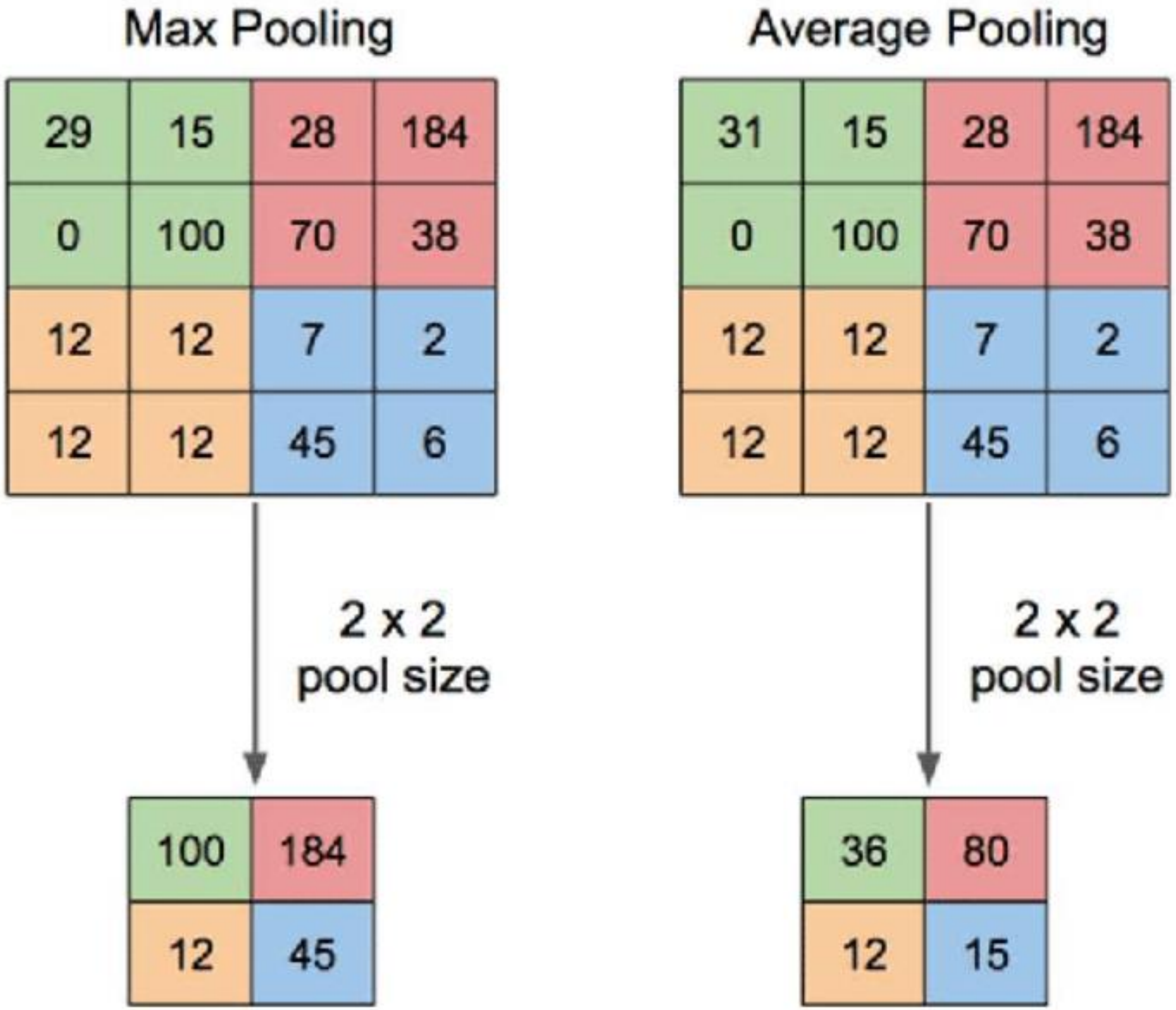
X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

13.5

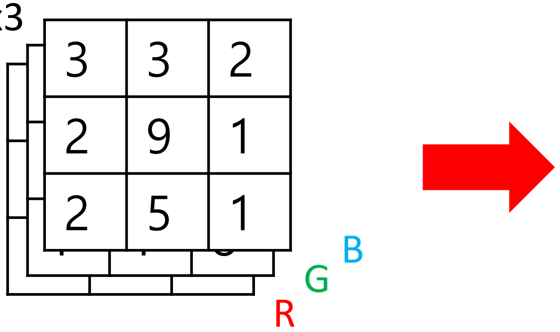
3.5	5.5
11.5	13.5

Convolution layer pooling

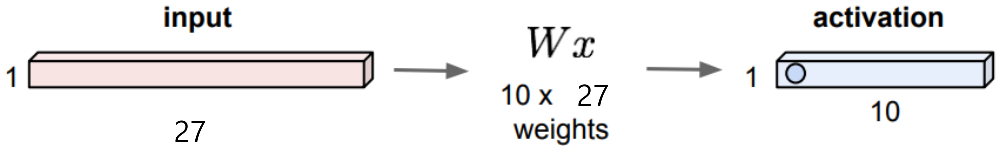


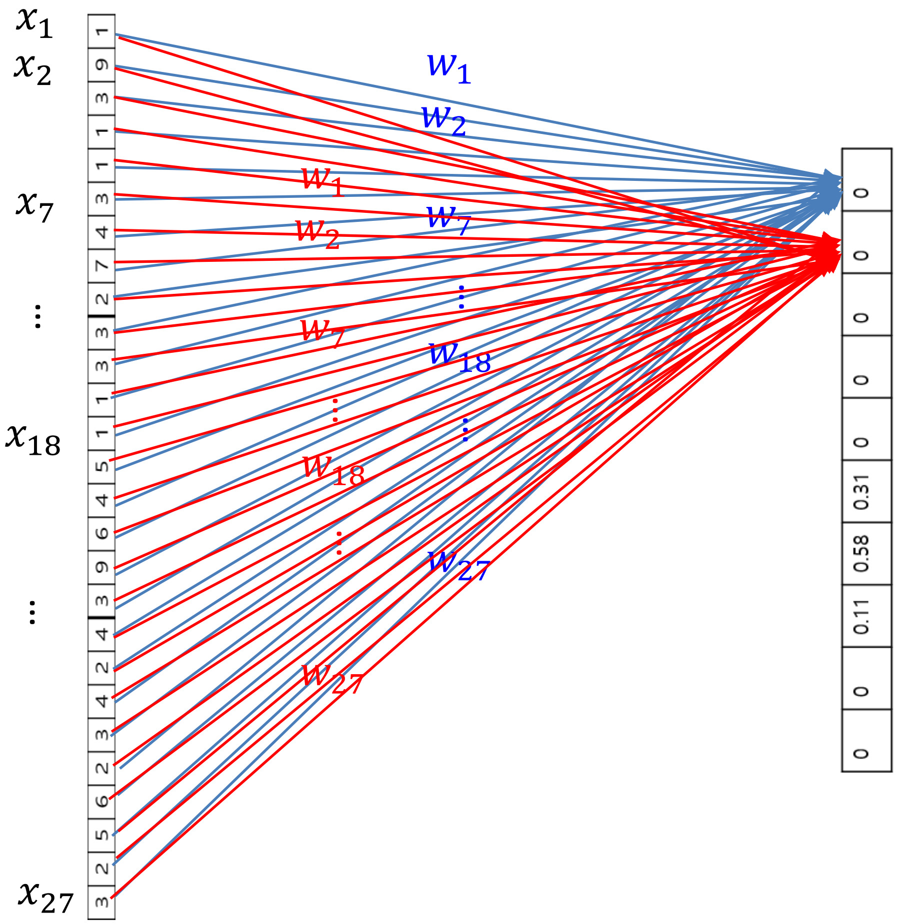
Fully connected layer

Image
3x3x3

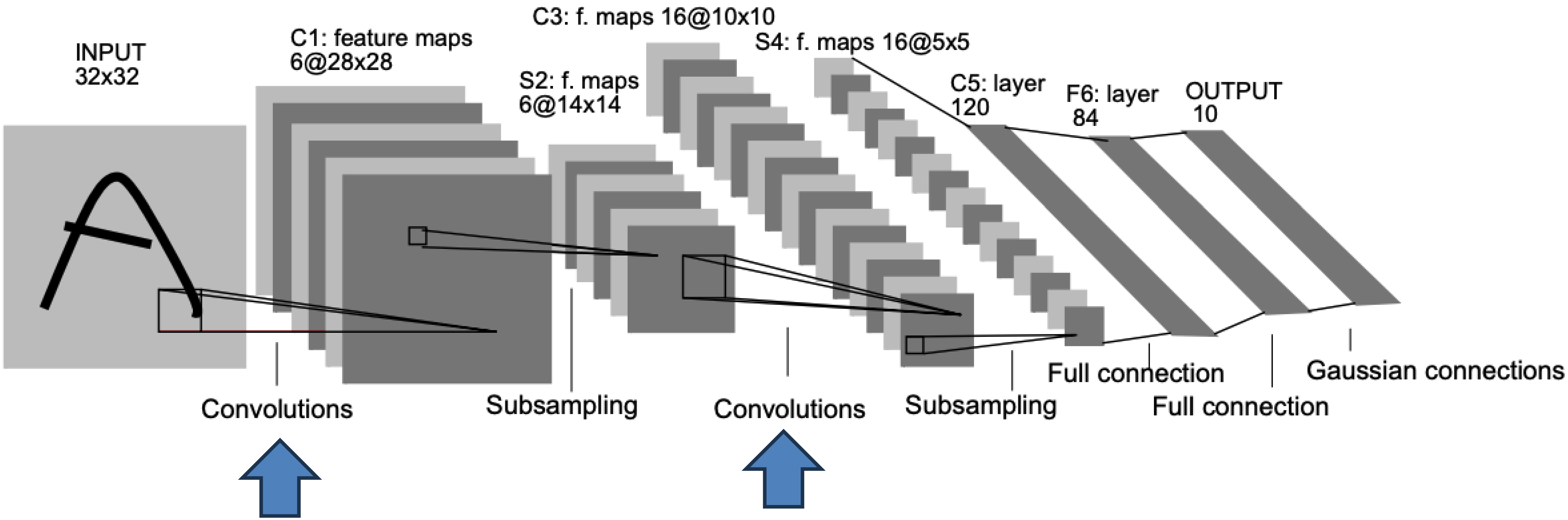


3	2	5	6	2	3	4	2	4	3	9	6	4	5	1	1	3	3	2	7	4	3	1	1	3	9	1





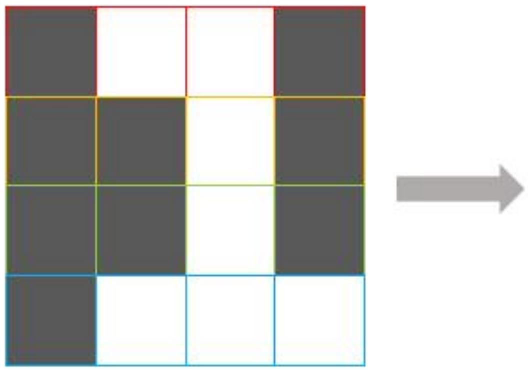
Convolution Neural Network

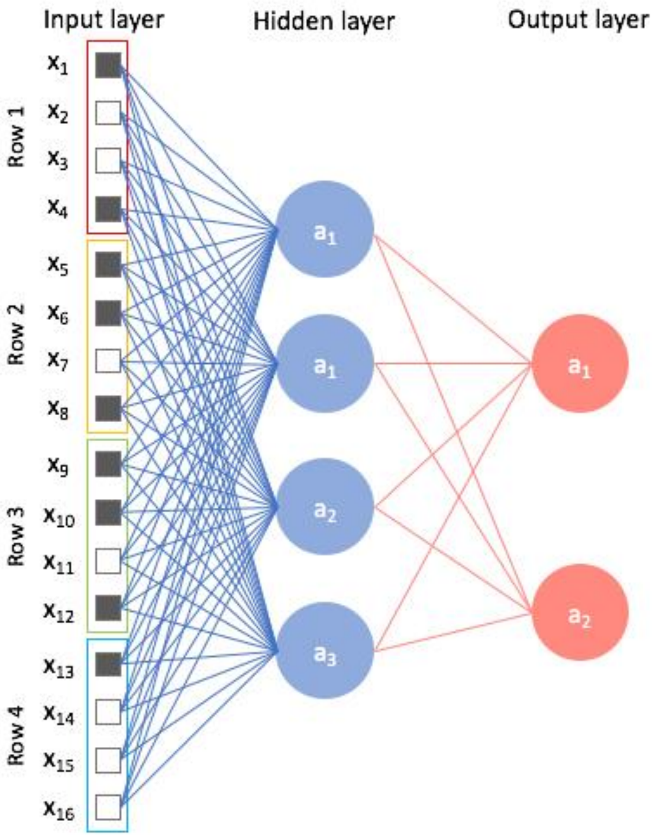


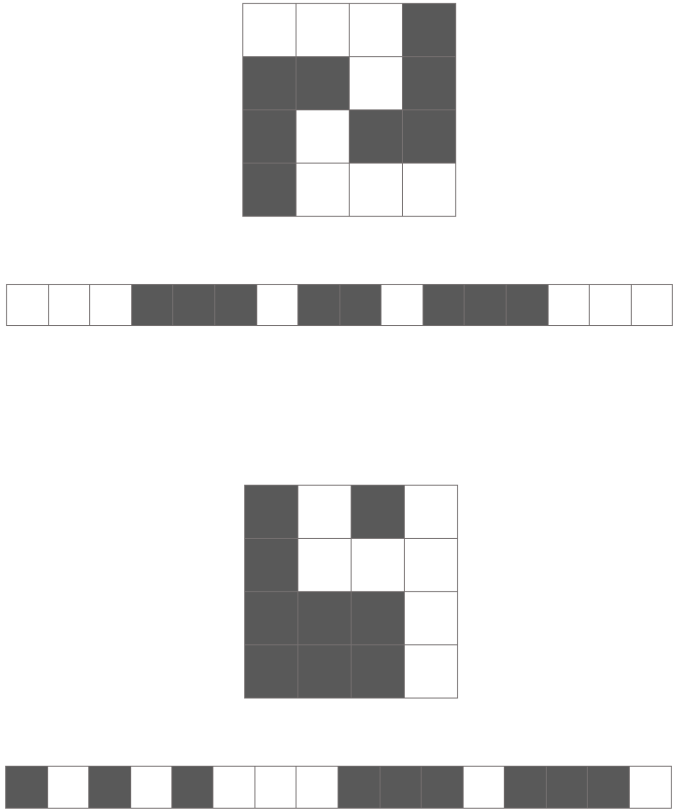
5*5*1 filter

5*5*6 filter

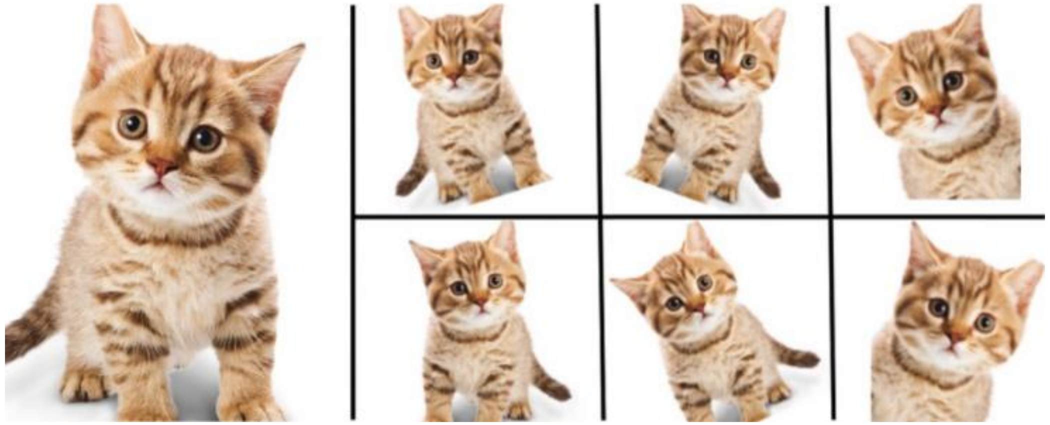
Why Convolution neural network?

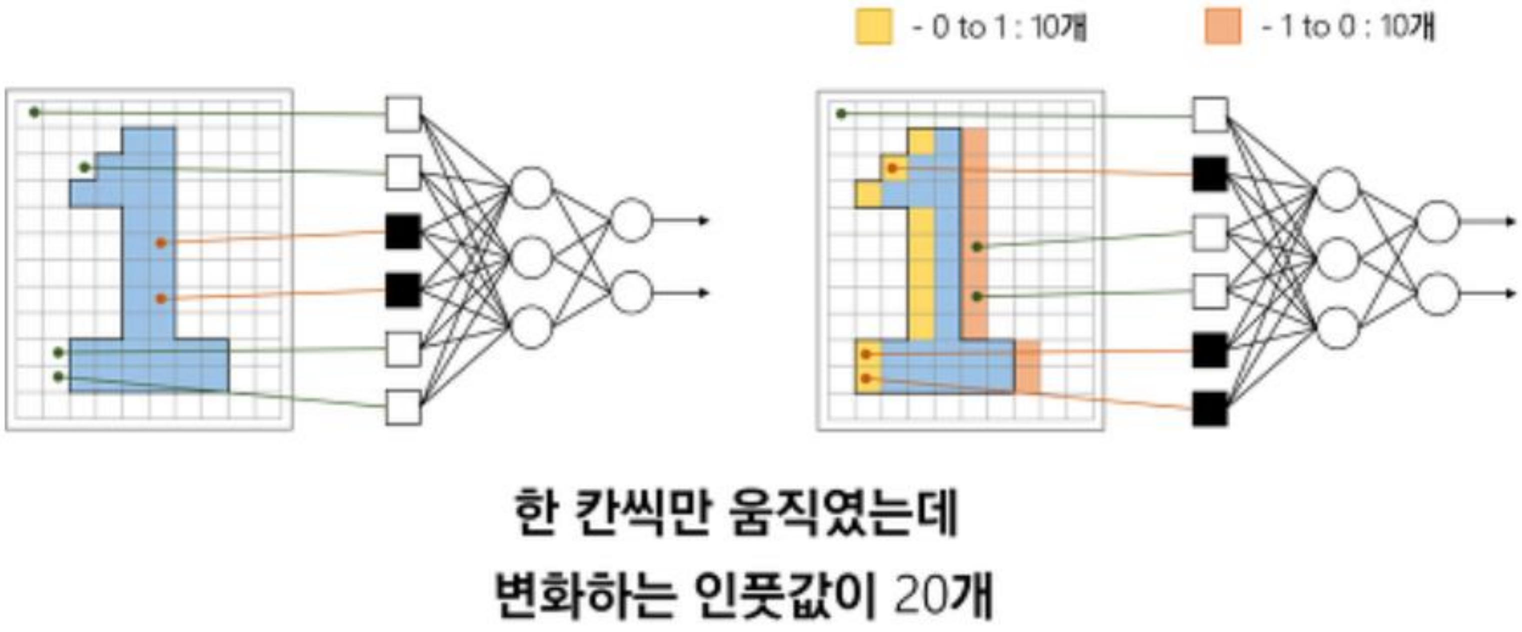




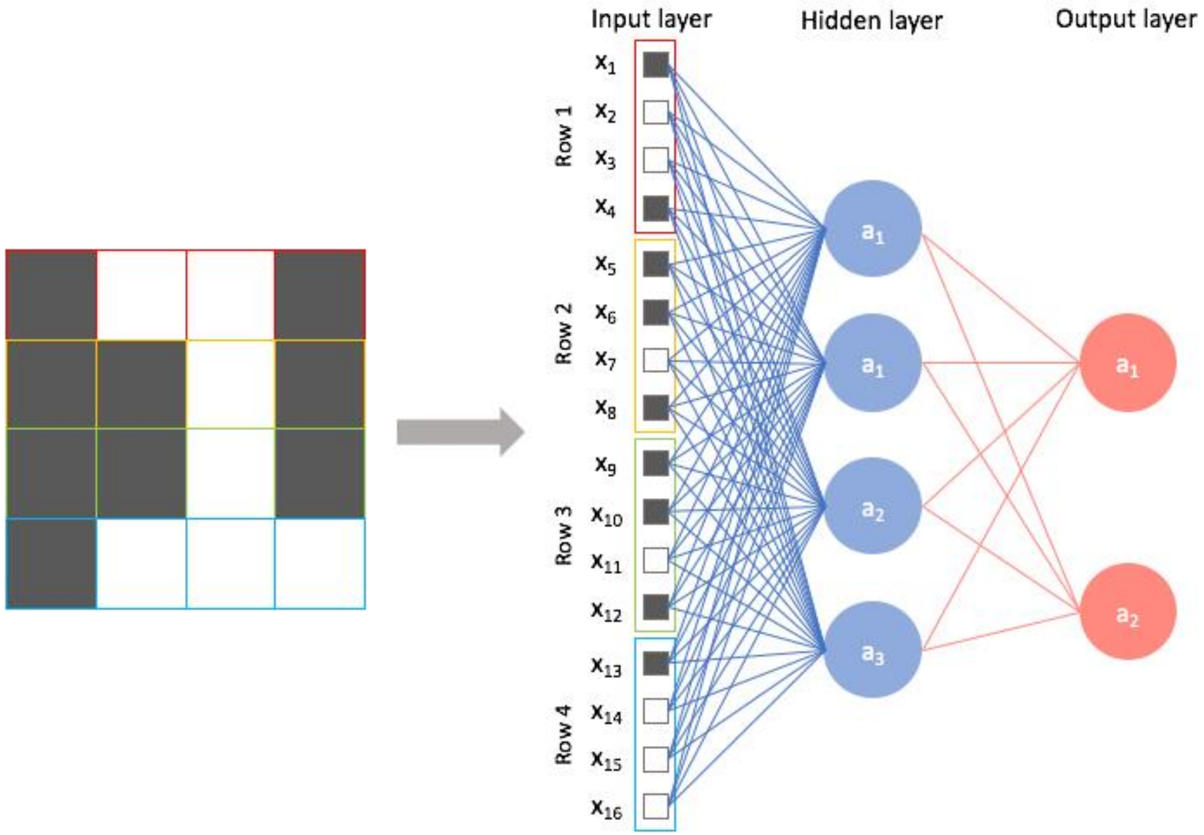


Why Convolution neural network?





Why Convolution neural network?

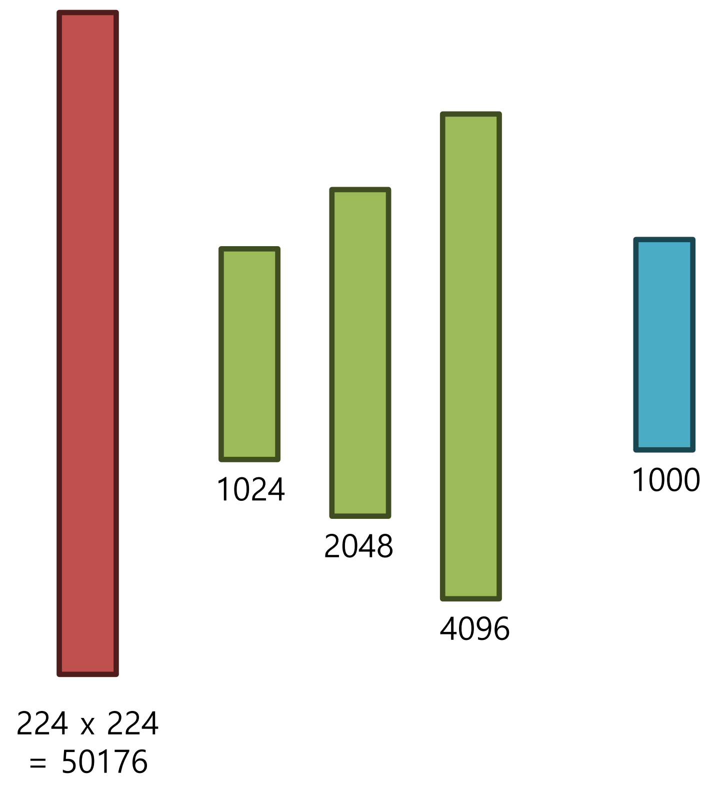


Model이 찾아야 하는 weight의 개수는?

16 (input size) x 4 (hidden node)
+ 4 (hidden node) x 2 (output)

72

Why Convolution neural network?



만약 input size 224 x 224

hidden layer가 3개

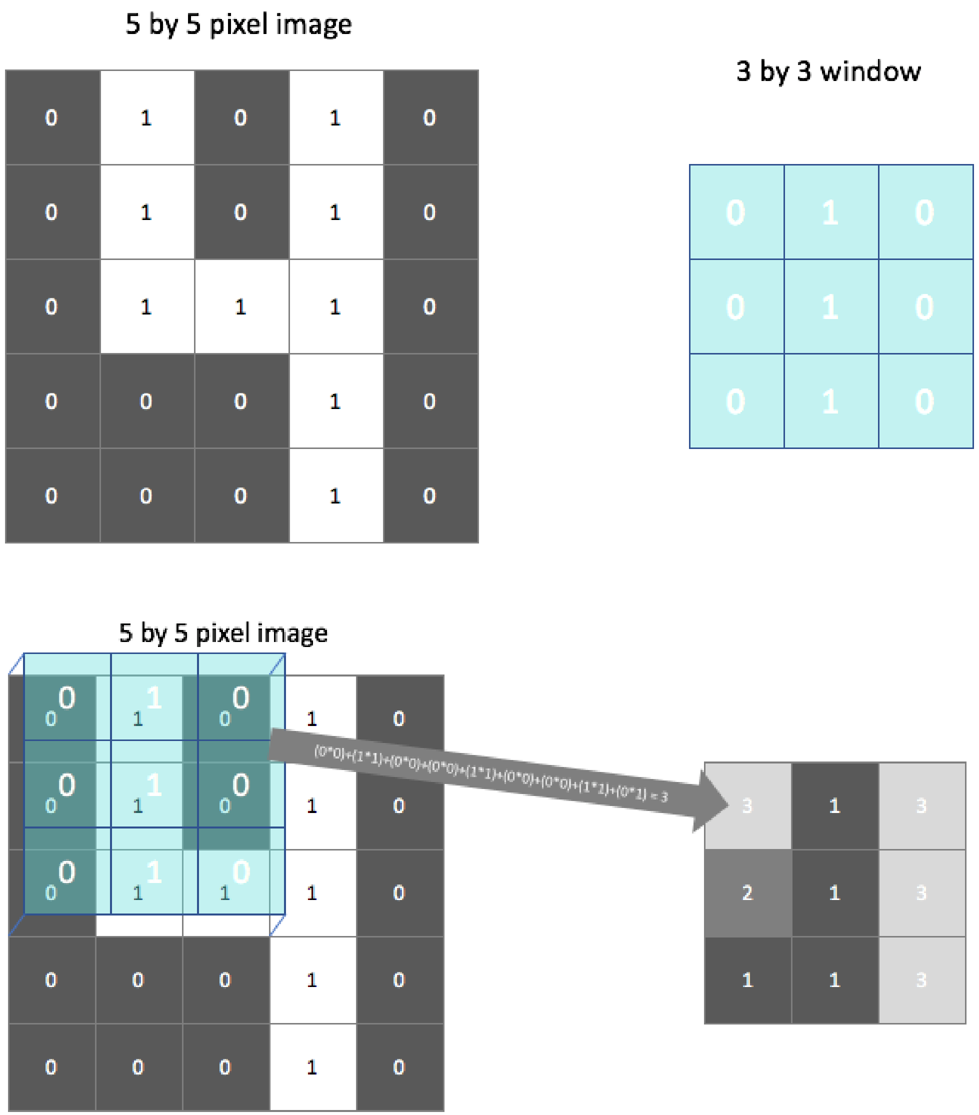
각각 1024, 2048, 4096개 node

Output 1000개의 class 일 경우

필요한 weight 개수는?

50176 x 1024 + 1024 x 2048 + 2048 x 4096 + 4096 x 1000
= 65,961,984개

Why Convolution neural network?



Input size 224x224

Convolution layer 3 kernel size 3x3

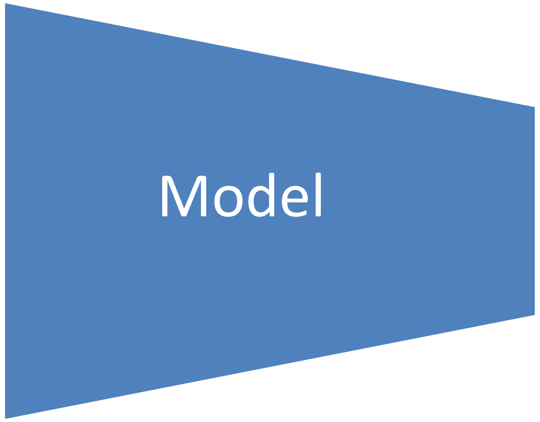
Each channel size 128, 256, 512

Last fully connected layer 1024

Output class 1000

3 x 3 x 128 + 3 x 3 x 128 x 256 + 3 x 3 x 256 x 512 + 1024 x 1000
=9 x 128 + 9 x 128 x 256 + 9 x 256 x 512 + 1024 x 1000
= 1,152 + 294,912 + 1,179,648 + 1,024,000
= 2,499,712

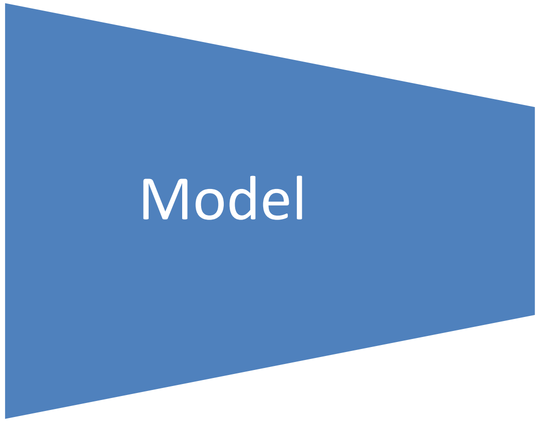
Classification



Cat	0.7
Dog	0.08
Car	0.01
Bird	0.01
Lion	0.2

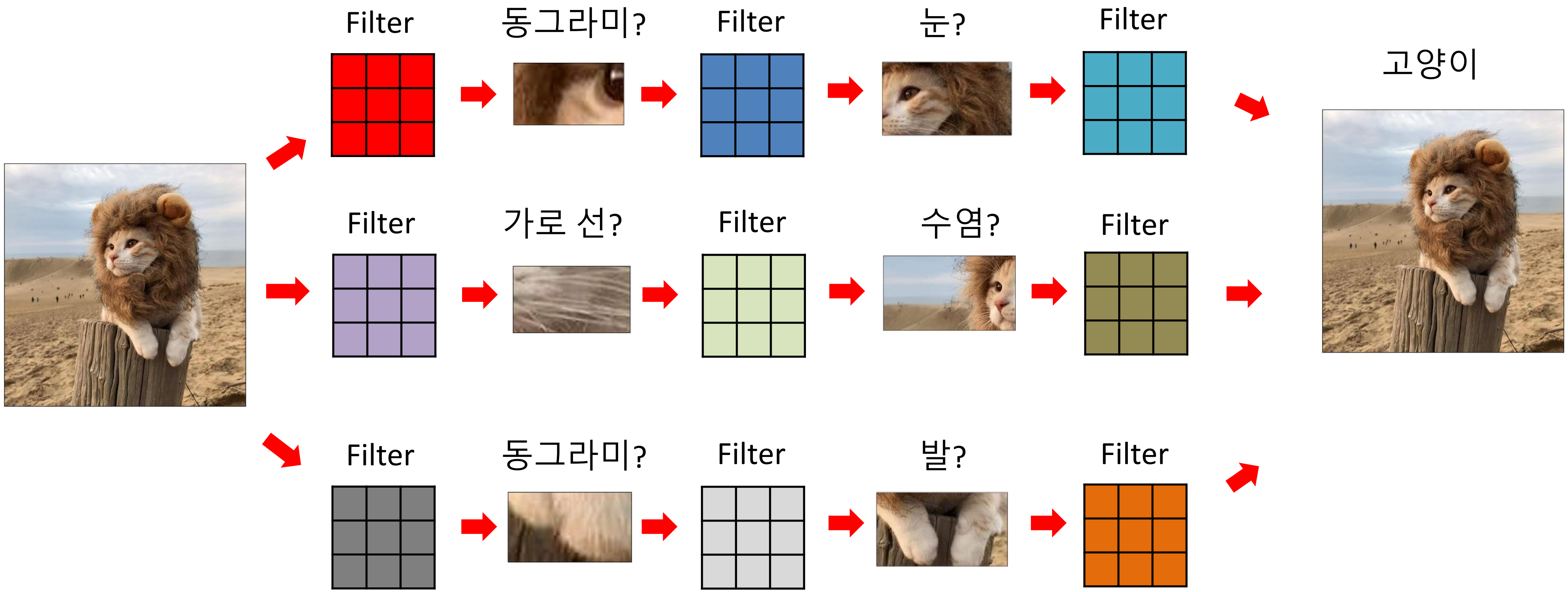
Detection

Segmentation

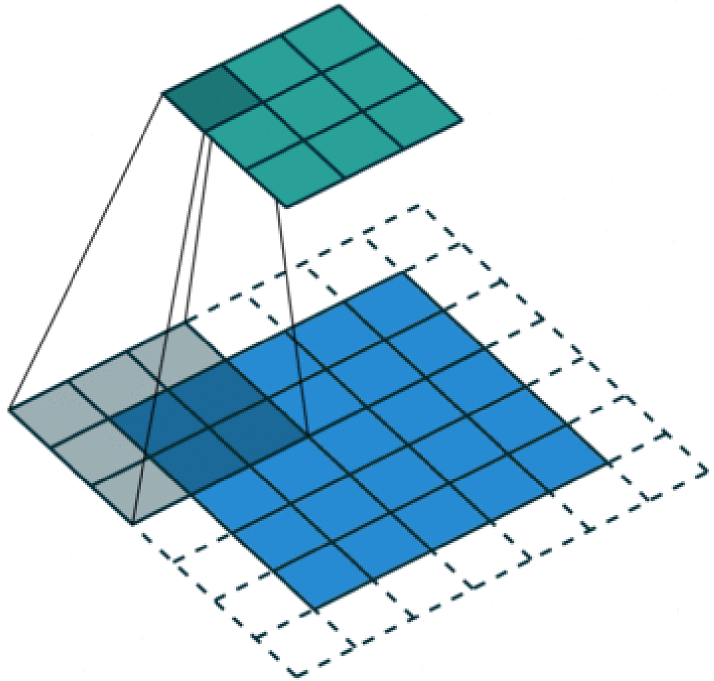


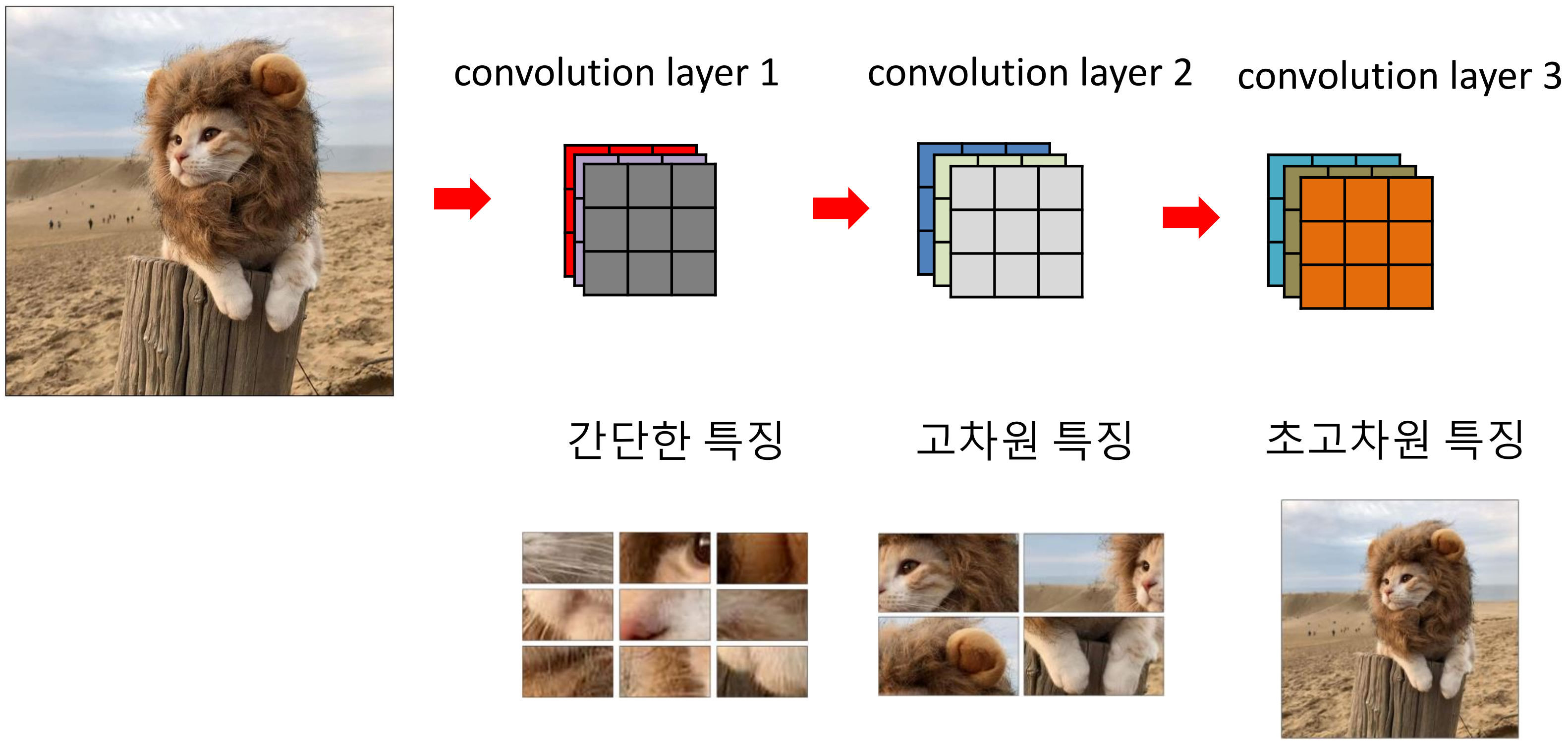
1	0	1	1	0	1
1	0	1	1	0	1
1	1	1	1	1	1
2	1	1	1	1	1
1	1	1	1	0	1
2	1	2	2	0	1
2	2	2	2	0	2
0	0	2	2	0	2

Convolution neural network

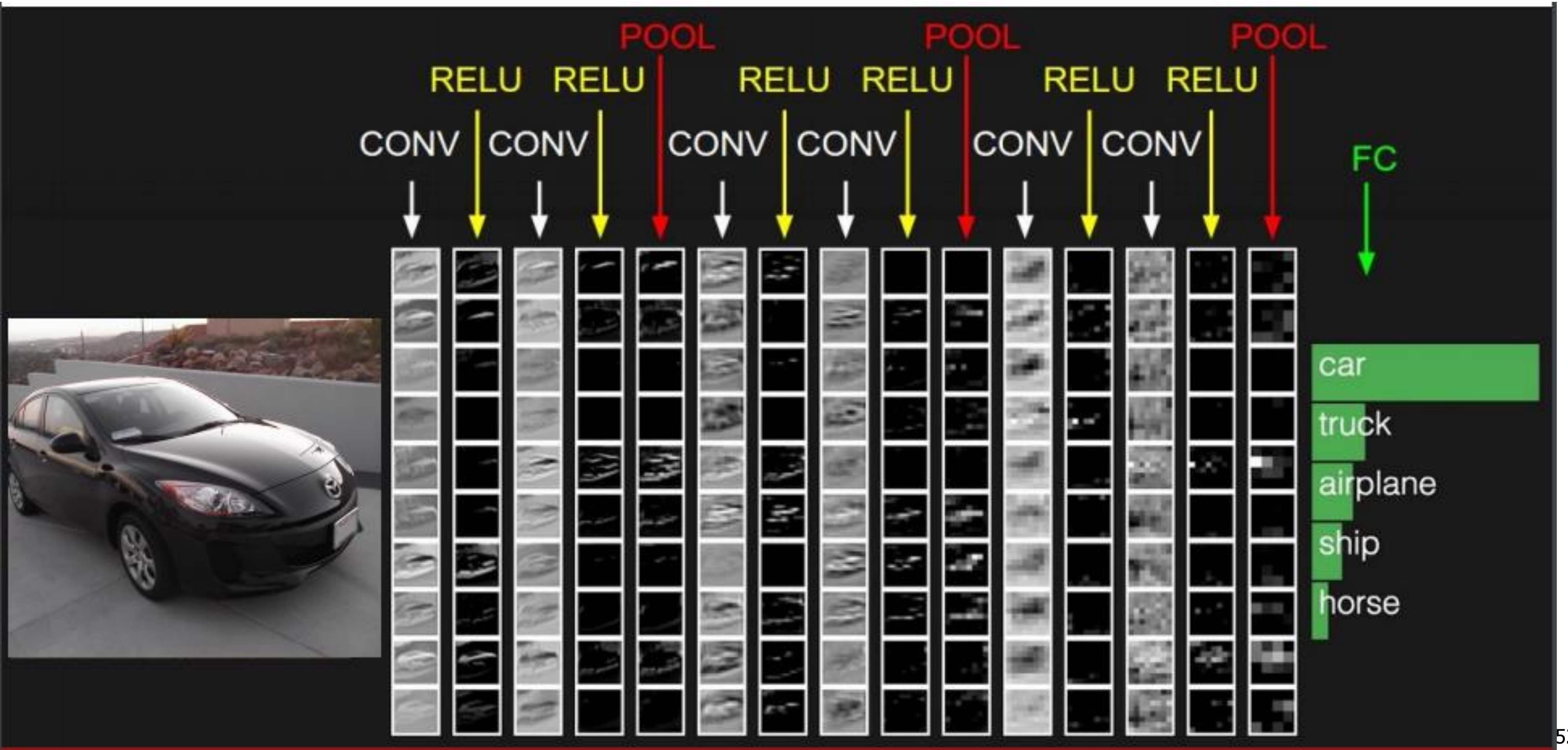


Convolution neural network

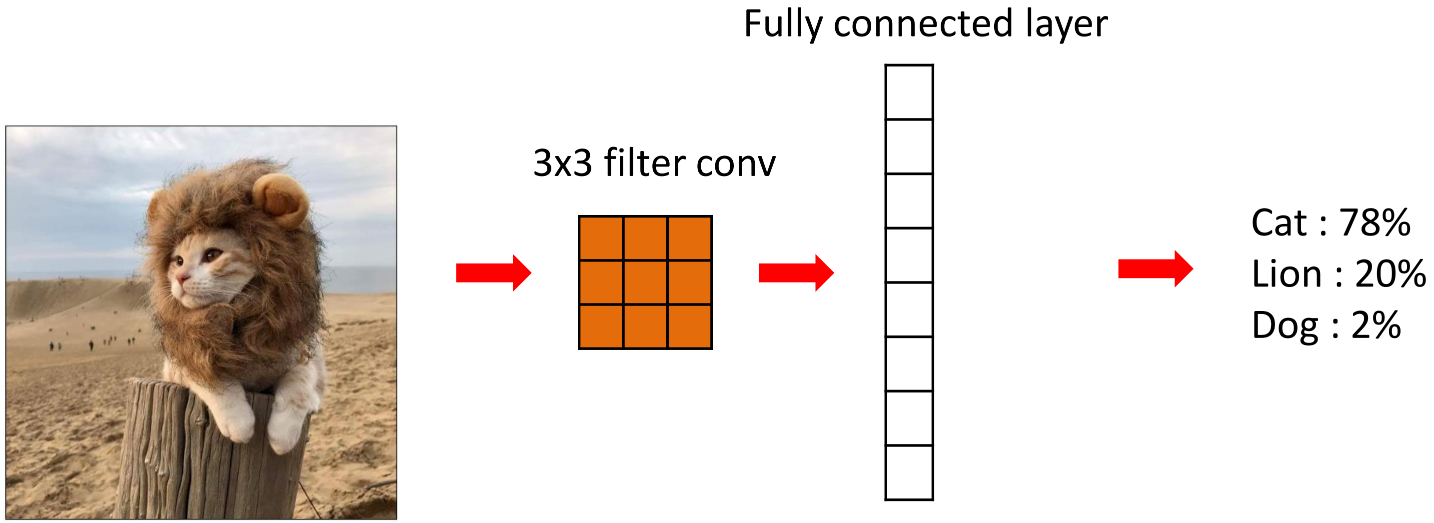




Convolution neural network



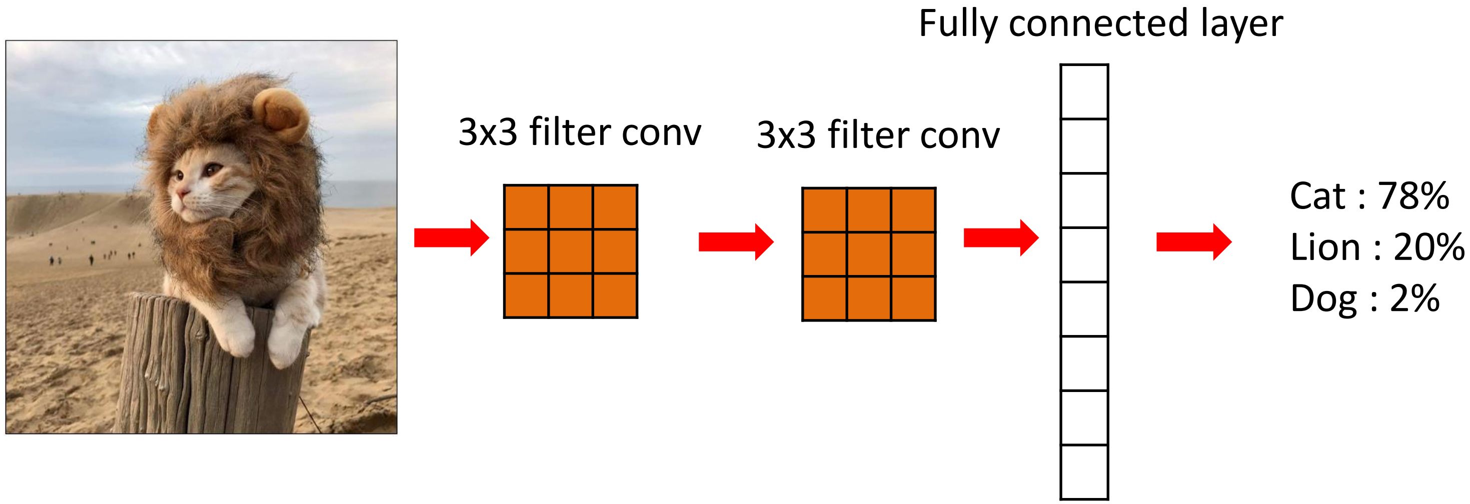
Receptive field



이 경우 model은  만 보고 판단을 한다.

즉 더 넓은 영역의 정보를 취합하여 예측하는게
아니라 3x3 영역까지의 정보만 보고 예측한다.

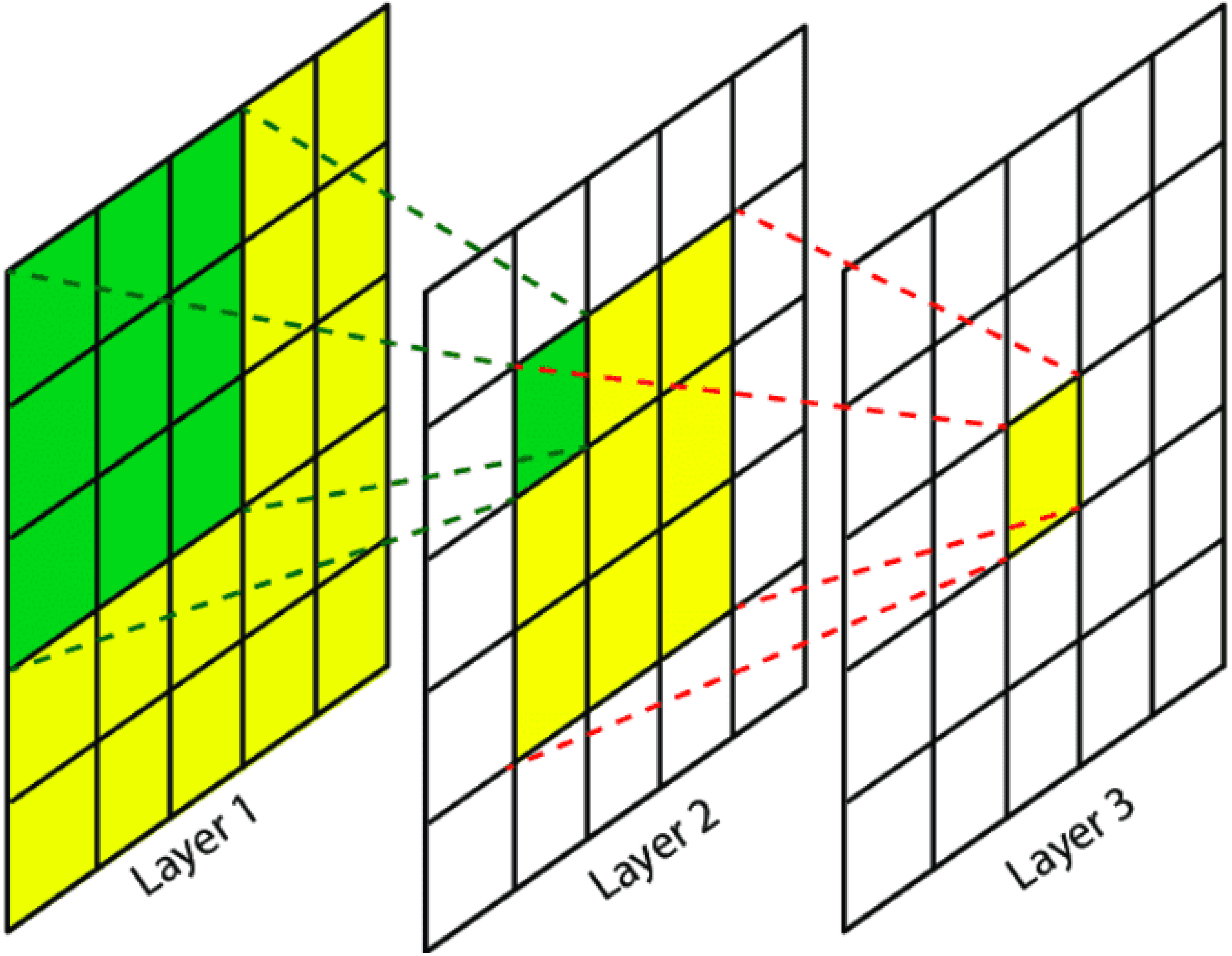
Ex) 귀가 있으면서 발이 있다 -> 정보를 모름
귀와 발은 3x3 이상 떨어져 있기 때문



이 경우 model은 5x5 영역까지의

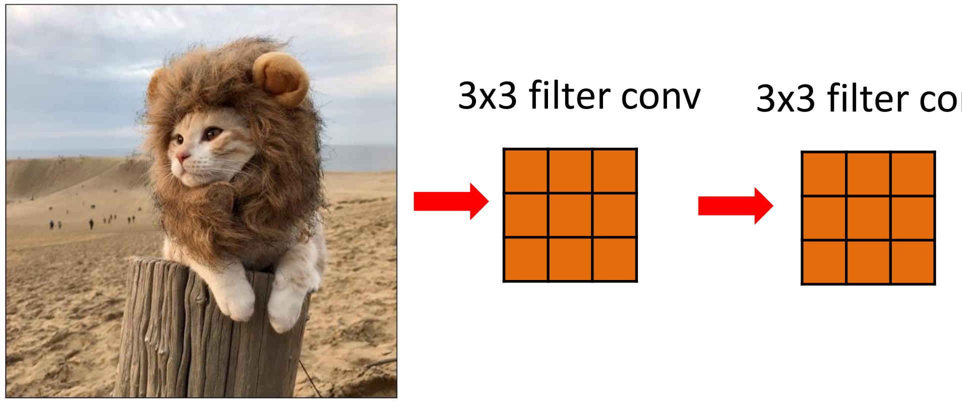
정보를 보고 예측한다. 

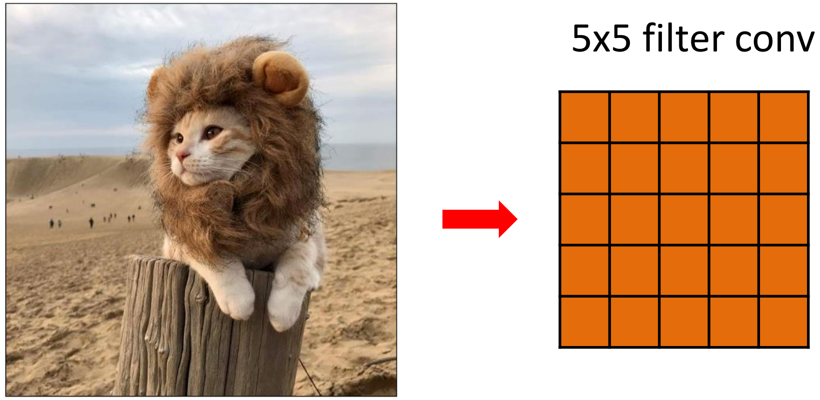
Receptive field



[출처] https://theaisummer.com/receptive-field/

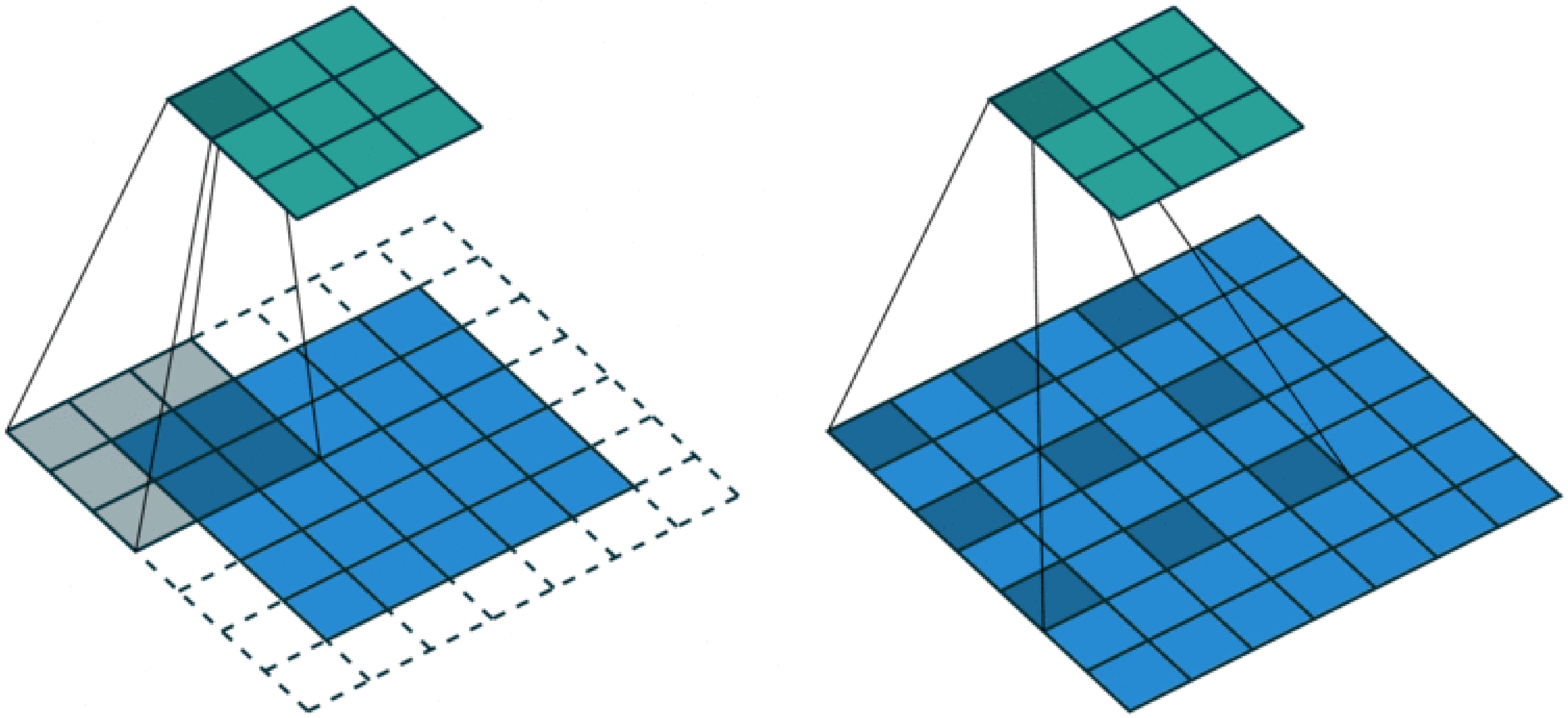
Receptive field





둘의 차이점은?

Dilation convolution



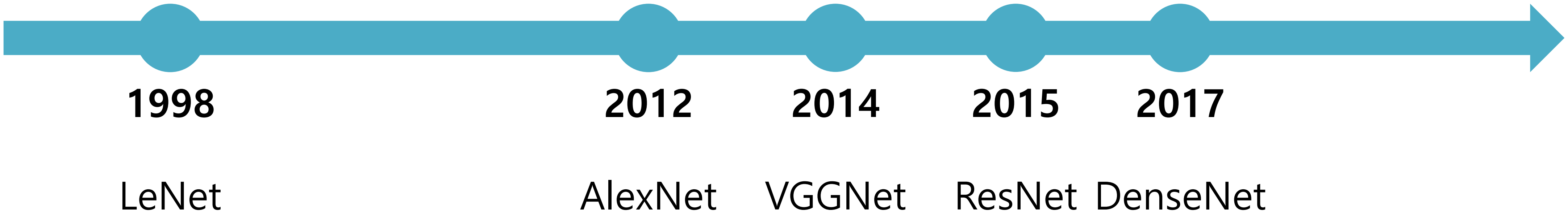
ImageNet

ImageNet은 computer vision 경연을 위해
ILSVRC에서 사용하는 유명한 데이터셋.
2012~2017년 까지 대회를 진행하였으며 현재에도
각종 논문에서 사용하고 있는 가장 유명한 데이터이다.

1000종류와 총 1,281,167장의 데이터가 존재하며
200GB가 넘는다.

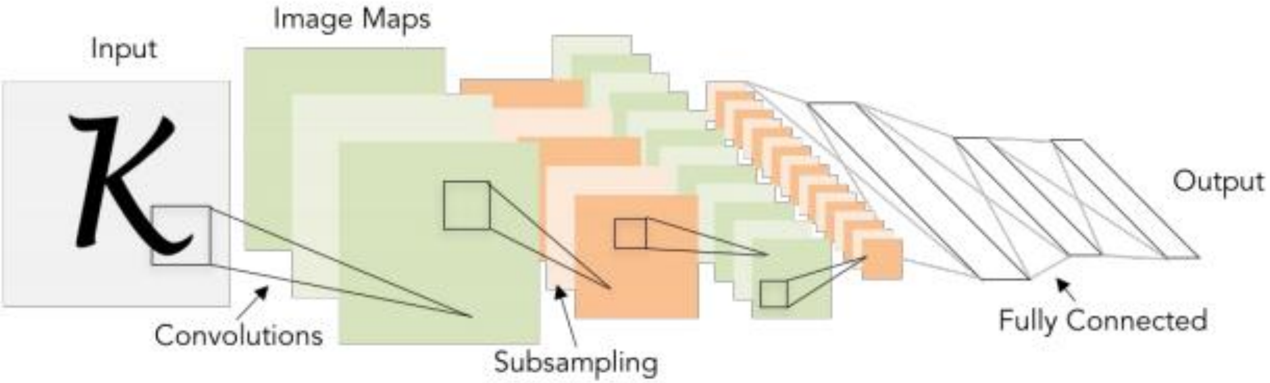
ILSVRC: ImageNet Large Scale Visual Recognition Competition

Convolutional Neural Network



Convolution neural network

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

[Krizhevsky, Sutskever, Hinton, 2012]

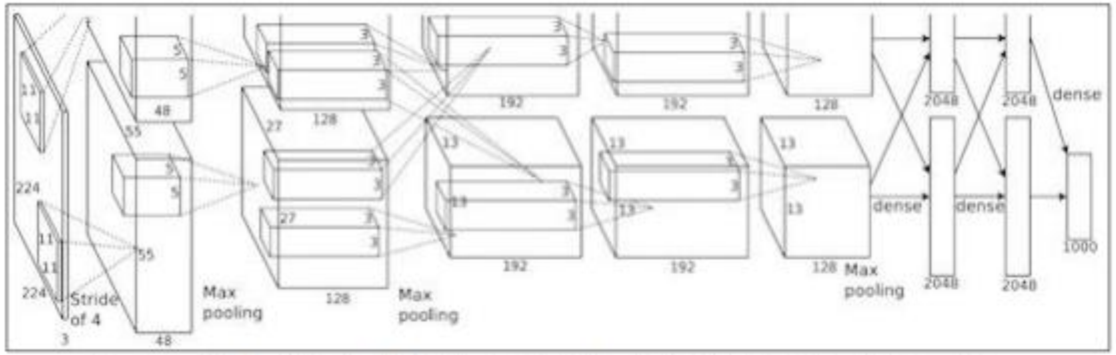
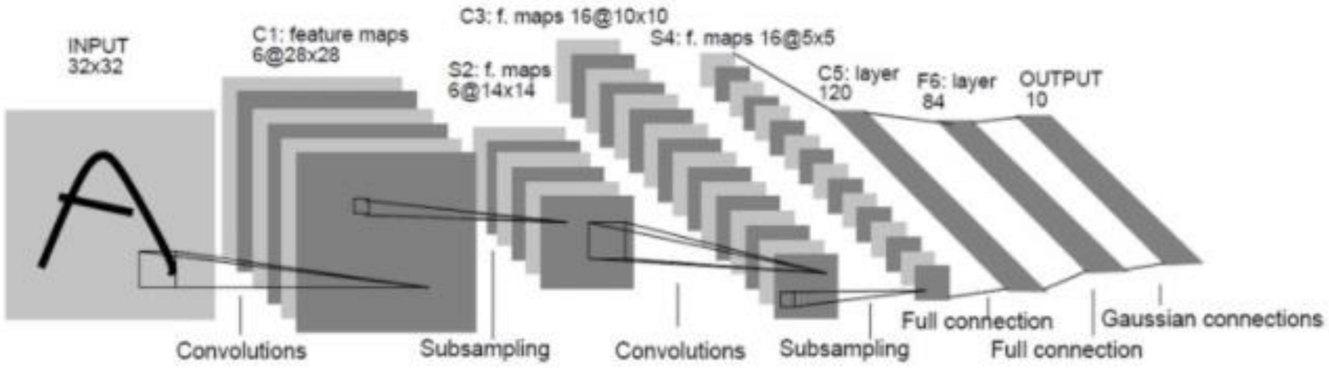


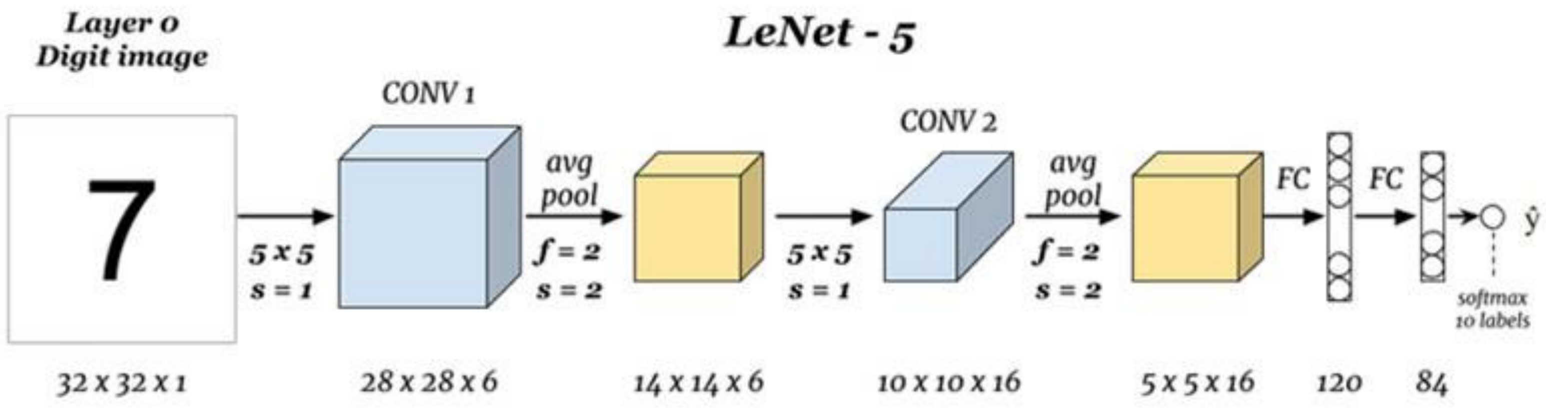
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

LeNet-5

- LeNet은 1998년 초창기 CNN 모델임
- 원래 우편번호와 수표의 필기체들을 인식하기 위한 용도로 사용하기 위해 개발됨

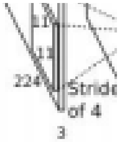




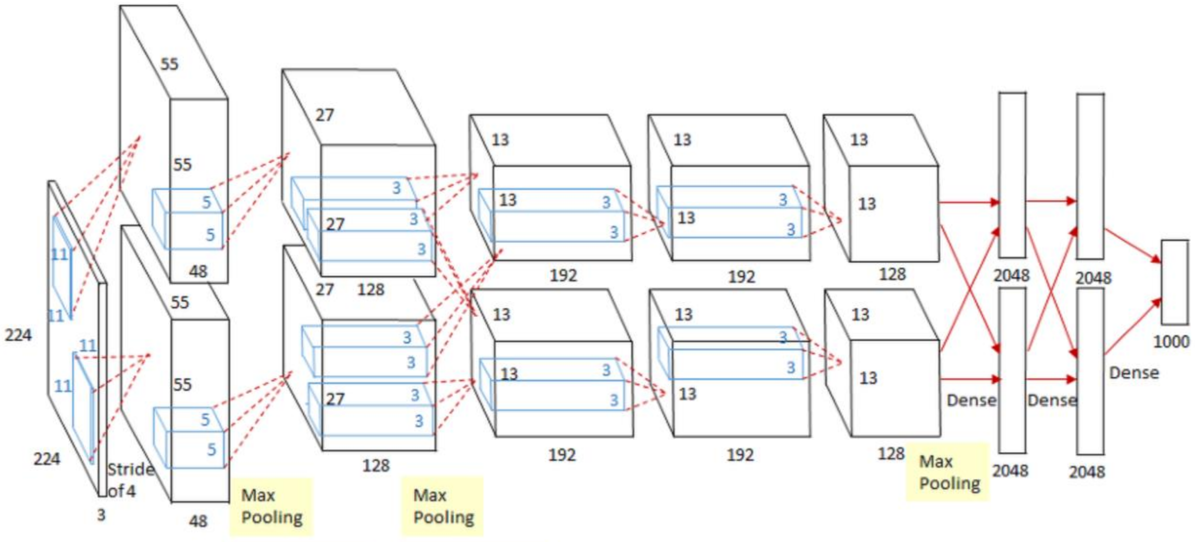
Parameter size ≒ 60,000

AlexNet

- AlexNet은 2012년 ILSVRC에서 우승인 CNN 구조
- LeNet-5 의 구조를 따르지만 컴퓨터 성능이 발전함에 따라 2개의 GPU를 병렬연산을 할 수 있게 모델을 설계함



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



Parameter size ≒ 62,000,000

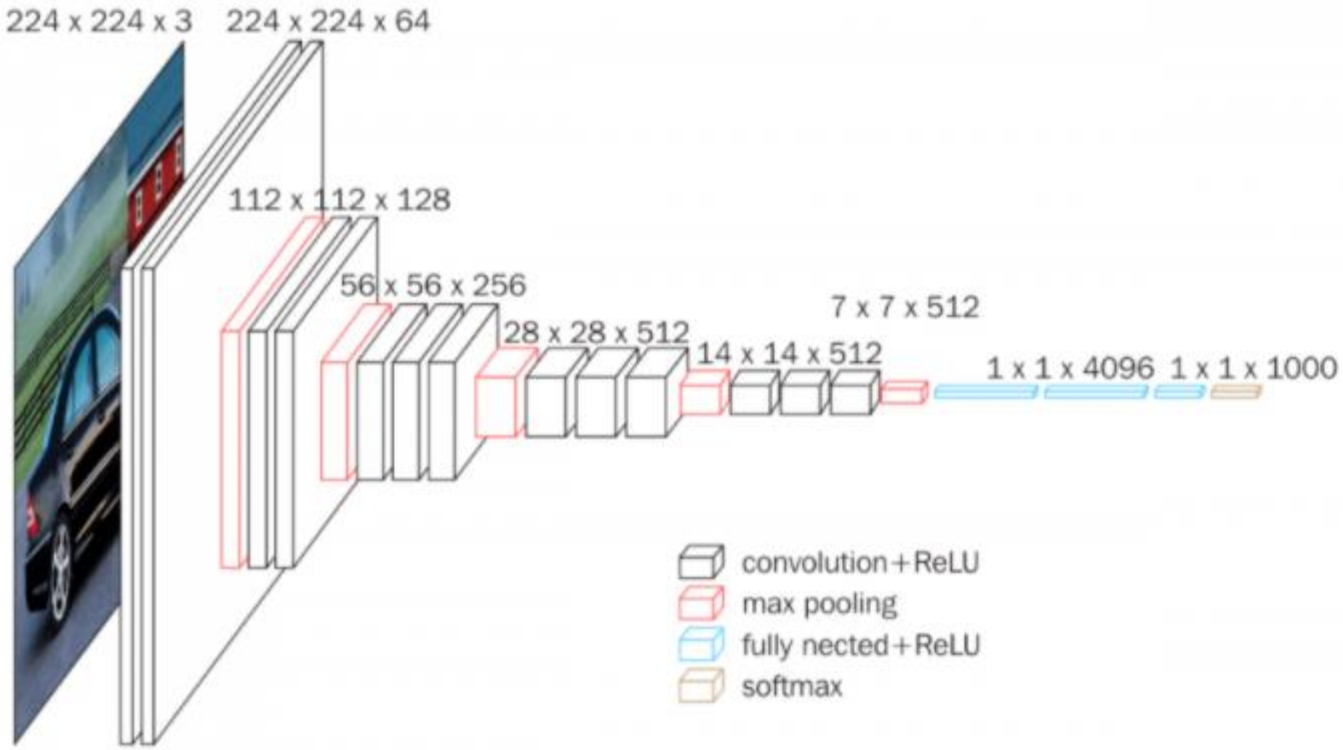
VGG

- VGG는 2014년 ILSVRC에서 2등을 차지한 CNN 구조
- 1등인 GoogleNet보다 구조는 간단하고 성능의 차이가 크지 않고 응용하기 쉬움

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

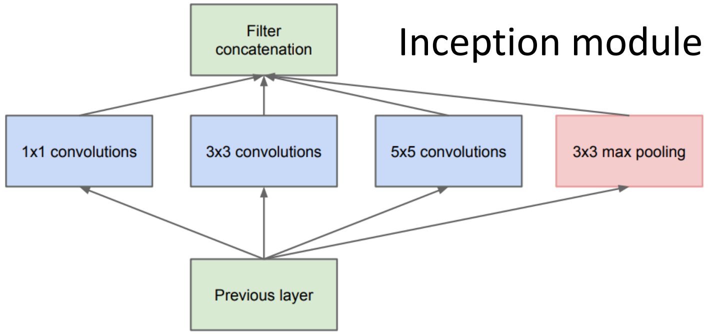
Parameter size ≒ 138,000,000

Ex) VGG16 (D type)

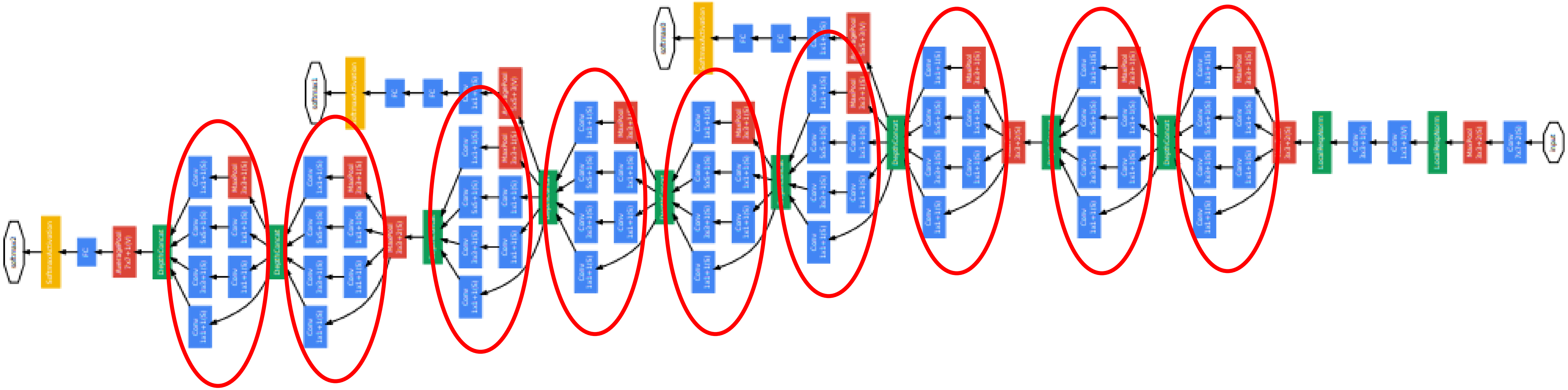


GoogleNet

- GoogleNet은 2014년 ILSVRC에서 우승한 CNN 구조
- Inception module을 이용함

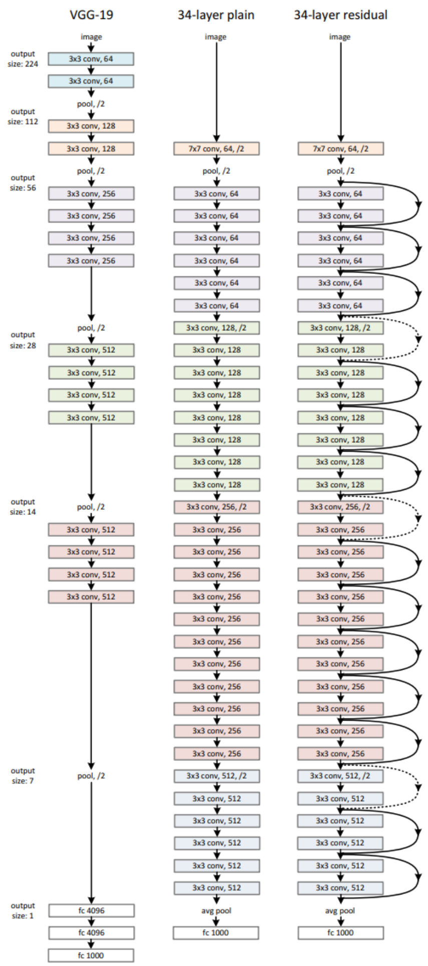


Inception module

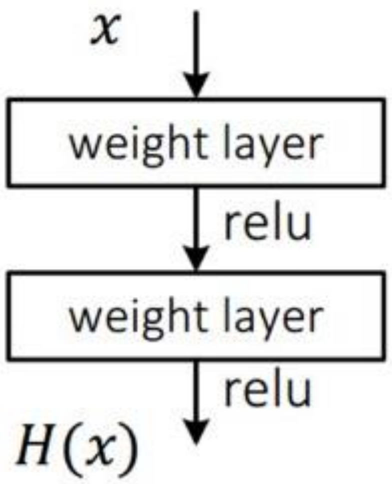


Resnet

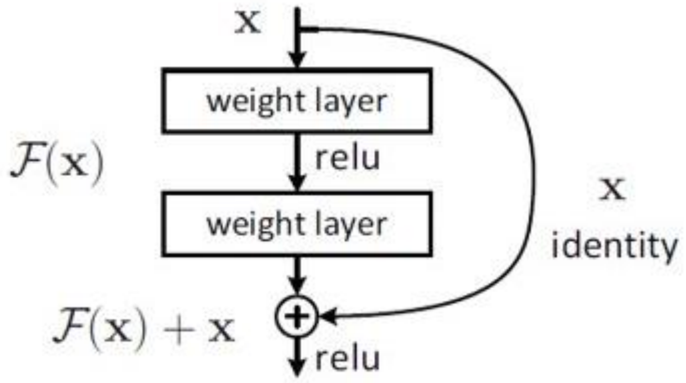
- Resnet은 2015년 ILSVRC에서 우승한 CNN 구조
- 기존 모델들과 다르게 residual 개념을 사용하여 더욱 더 깊은 network를 설계함



일반적인 network의 구조



residual network의 구조



Q & A

- Thank You