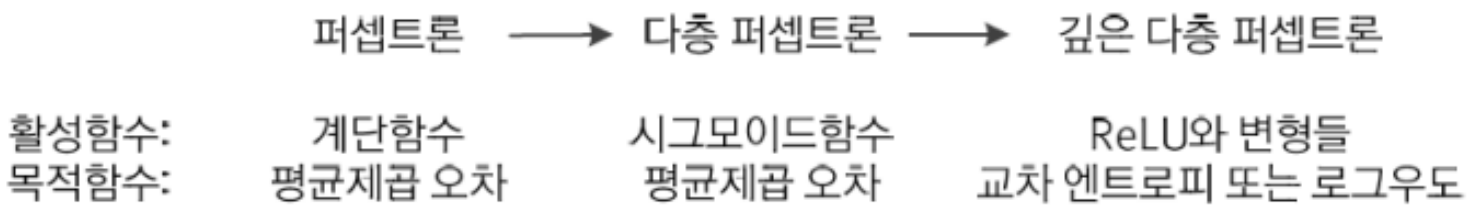
CNN: Convolutional Neural Network

15기 분석 임창건

학습알고리즘의주요개선



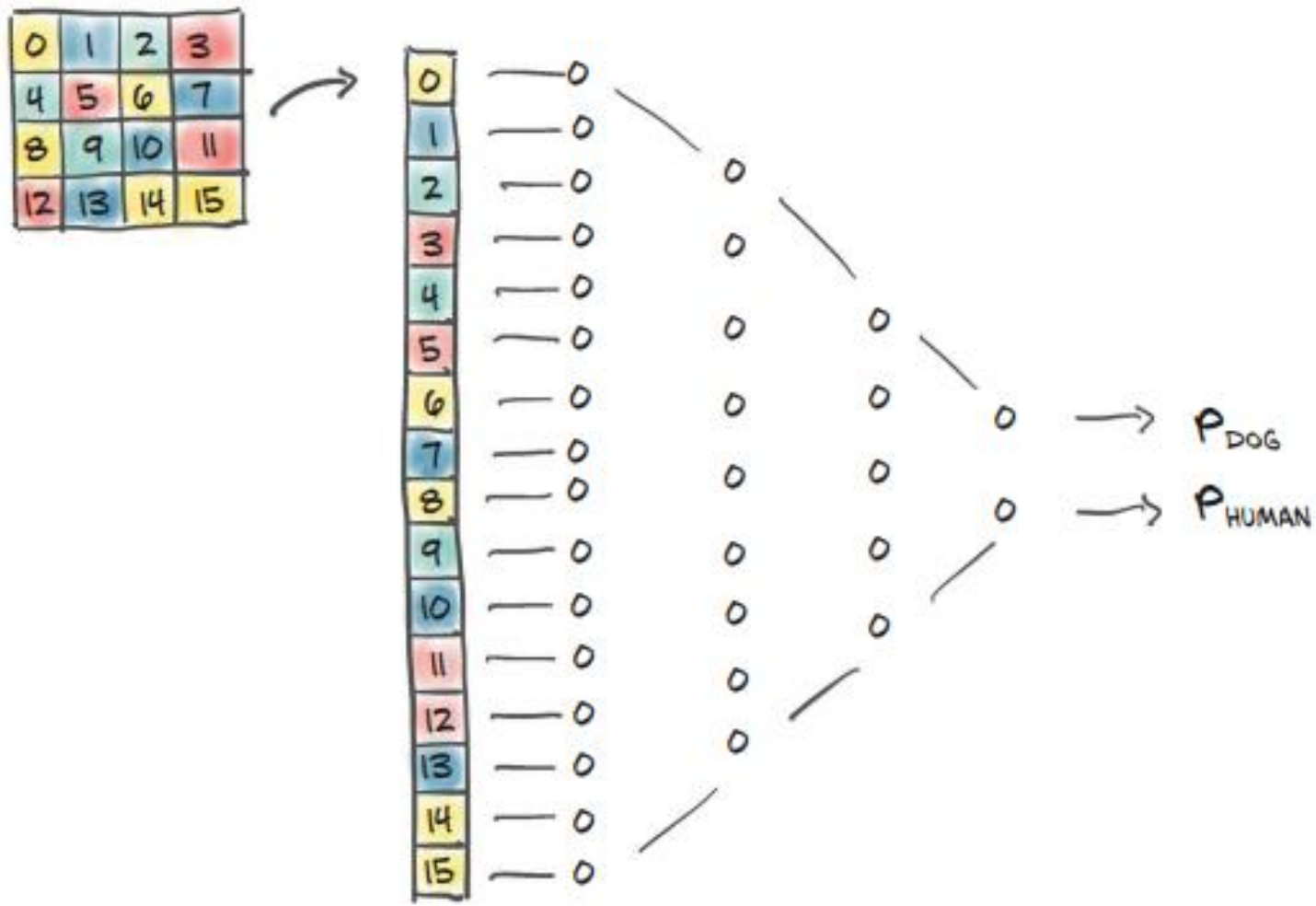
1. **Perceptron → MLP**
   - 선형적 분리만 가능했던 perceptron의 한계 극복하고자 여러 층 쌓음
   - Activation func을 step → sigmoid로 (확률적 요소 넣어 mapping)

2. **MLP → DMLP**
   - Sigmoid의 gradiant vanishing 문제를 ReLU를 사용함으로써 해결

3. **DMLP is good, but…**
   - fully connection에서 발생하는 parameter 수 너무 많음 (= overfitting 발생가능성 높음)

   → 'CNN' 등장

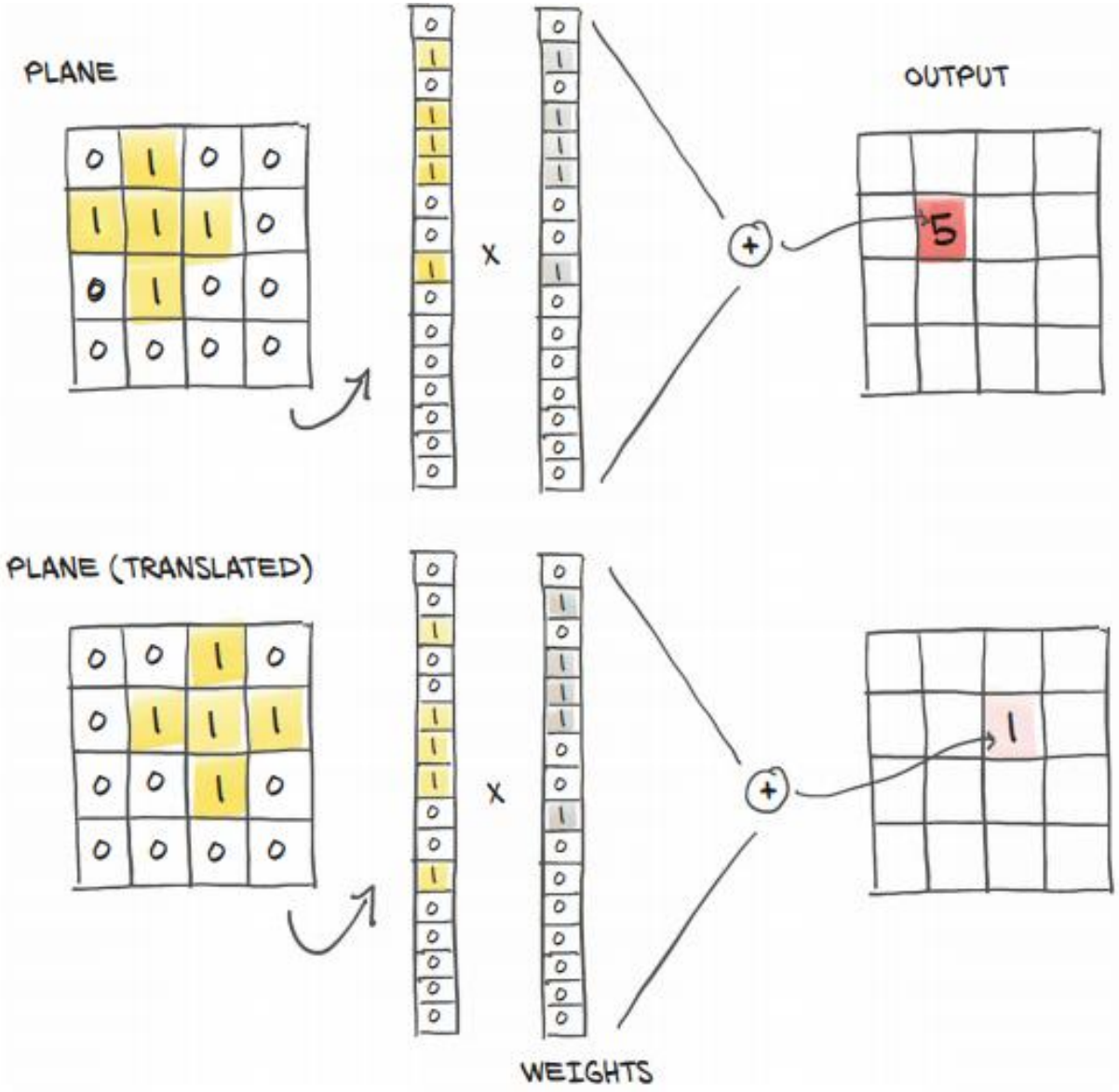
MLP



MLP

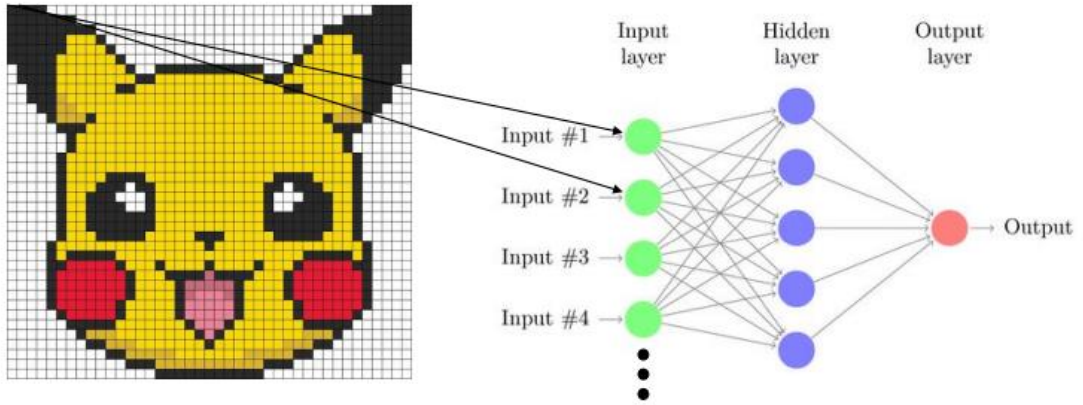
한계점

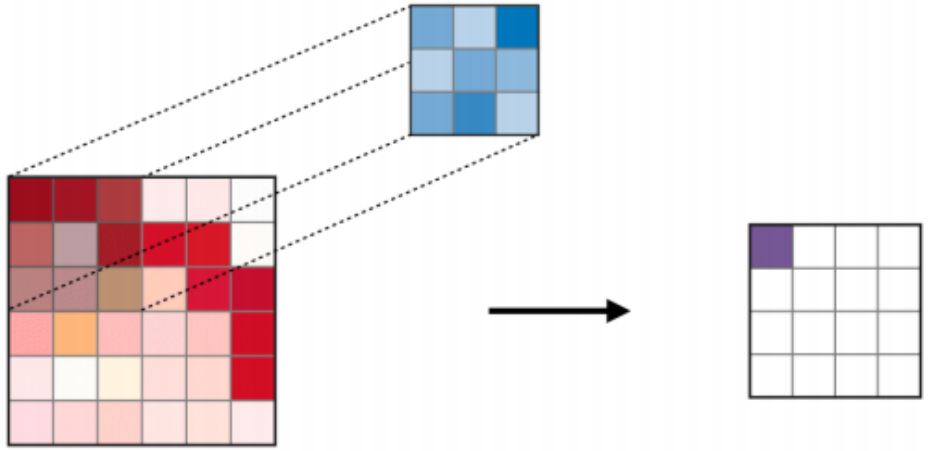
- 물체가 이동시 새롭게 학습이 필요하다(translation invariant)
- 학습시간이 오래 걸린다
- 파라미터가 많다
- 오버피팅 가능성이 높다



DMLP vs CNN

- **DMLP와 CNN의 비교**
  - DMLP
    - fully connected 구조로 높은 복잡도
    - 학습이 매우 느리고 overfitting 우려
    - 공간적 정보 손실
  - CNN
    - 입력 데이터의 locality에 입각, Convolution 연산을 이용한 부분연결(희소연결) 구조, 파라미터 공유로 복잡도 크게 낮춤
    - Convolution 연산은 좋은 특징 추출
    - 공간적 정보 활용
- CNN
  - 격자 구조(영상, 음성 등)를 갖는 데이터에 적합
  - 수용장은receptive field 인간 시각과 유사
  - 가변 크기의 입력 처리 가능

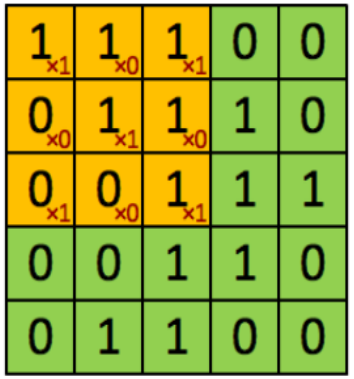




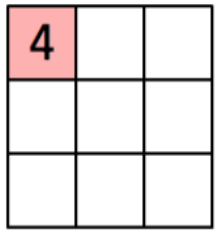
What is Convolution?

1	0	1
0	1	0
1	0	1

Kernel or filter
(3x3)



Image



Convolved
Feature

Input image

x11	x12	x13	x14
x21	x22	x23	x24
x31	x32	x33	x34

Kernel or filter
(2x2)

w11	w12
w21	w22

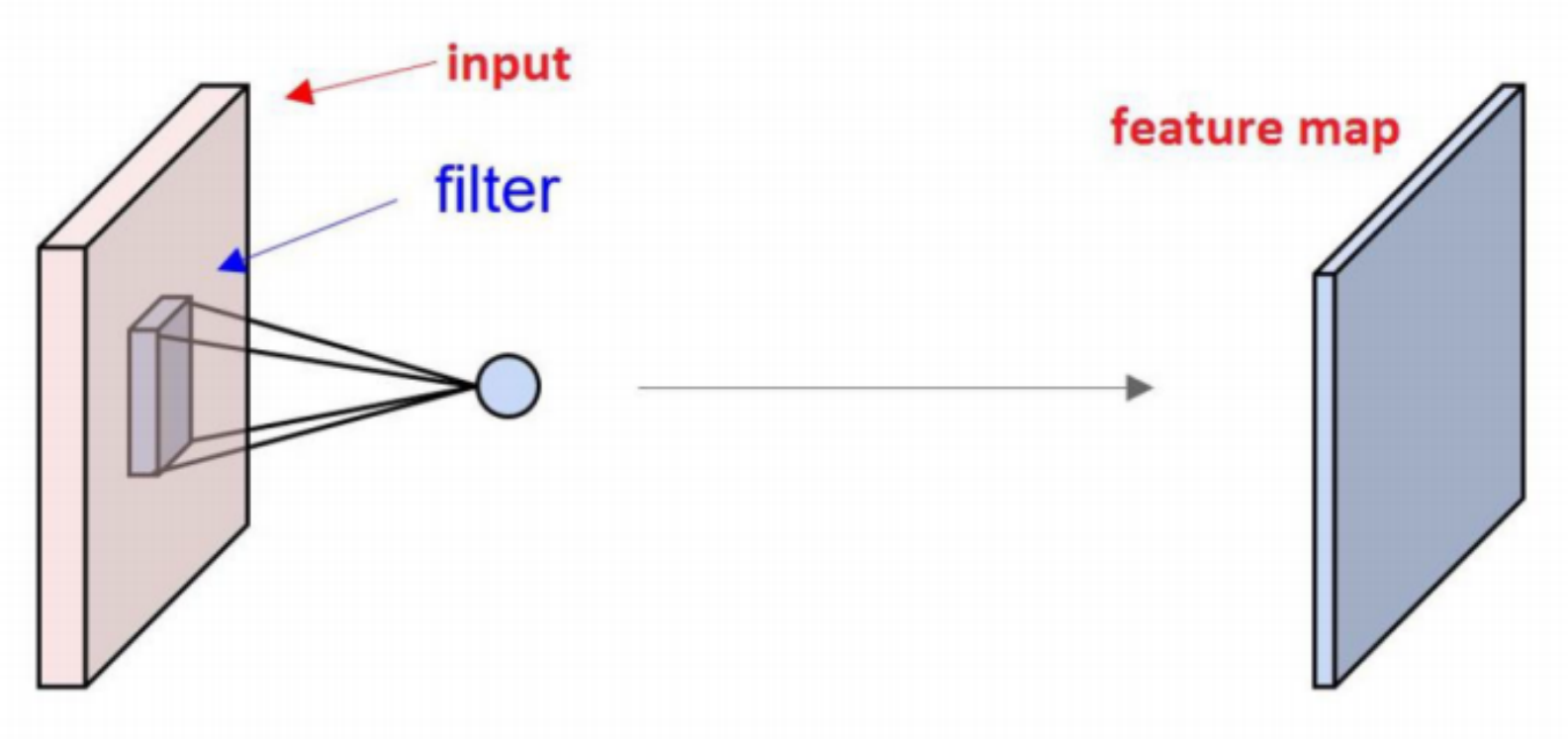
Output

x11w11+x12w12+ x21w21+x22w22	x12w11+x13w12+ x22w21+x23w22	x13w11+x14w12+ x23w21+x24w22
x21w11+x22w12+ x31w21+x32w22	x22w11+x23w12+ x32w21+x33w22	x23w11+x24w12+ x33w21+x34w22

Convolution layer

1.Filter(Kernel)
2.Padding
3.Stride
4.Pooling

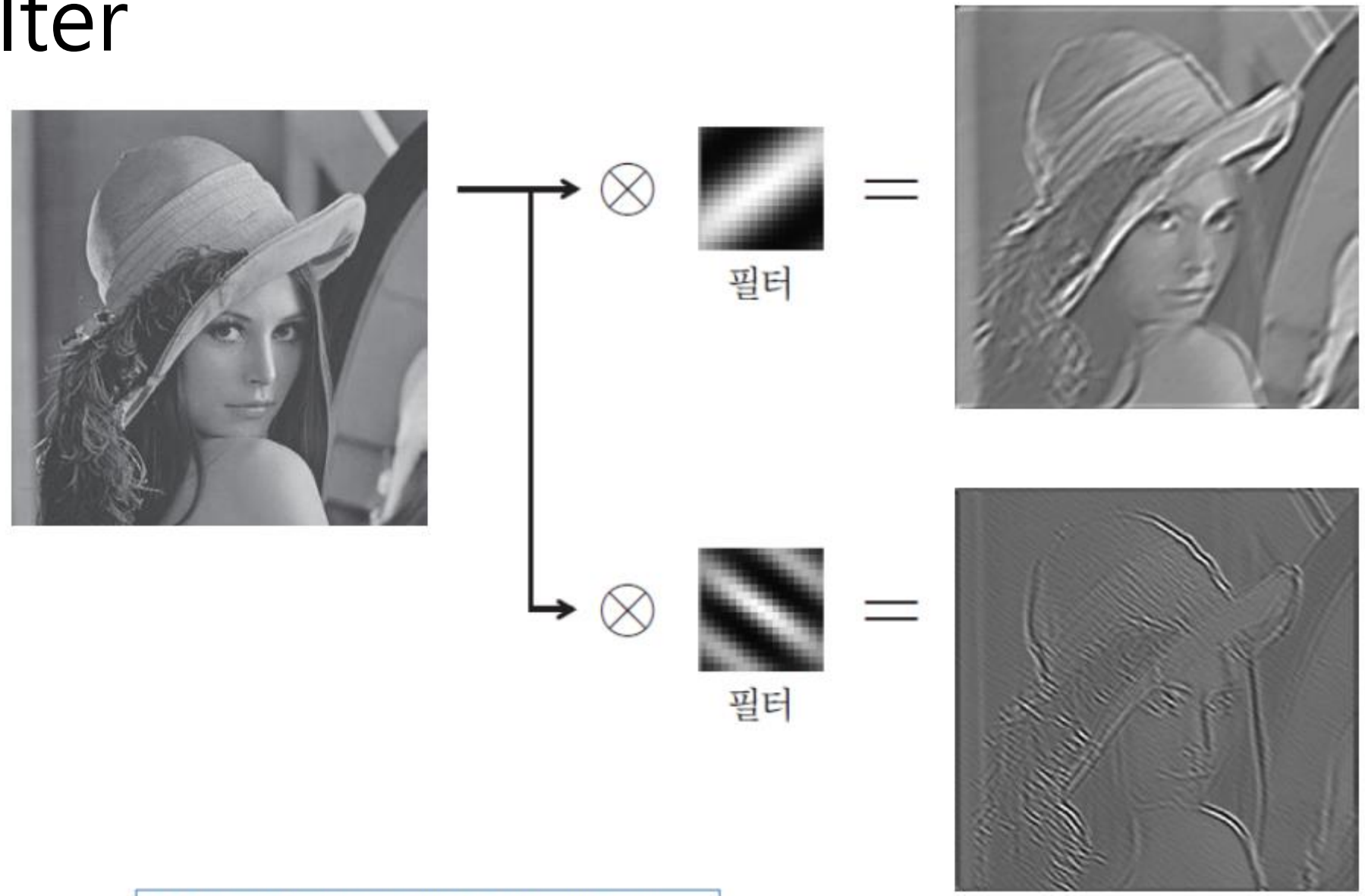
1.Filter



- **하나의 Filter당 하나의 Feature map 생성**
- 하나의 Filter는 input을 **sliding**하면서 **parameter 공유**(= 모든 노드가 동일한 kernel 사용)

  → 모델 복잡도 크게 낮아짐

1.Filter



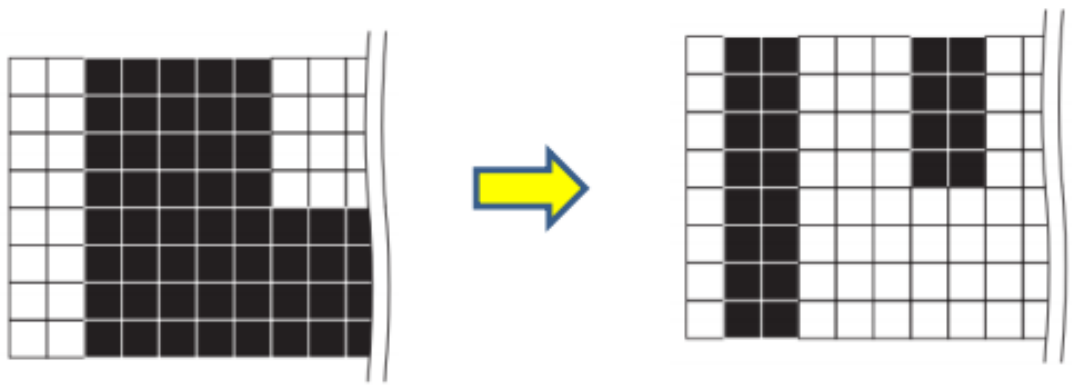
딥 러닝 제대로 시작하기, 출판사 제이펍, 심효섭역

Filters

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

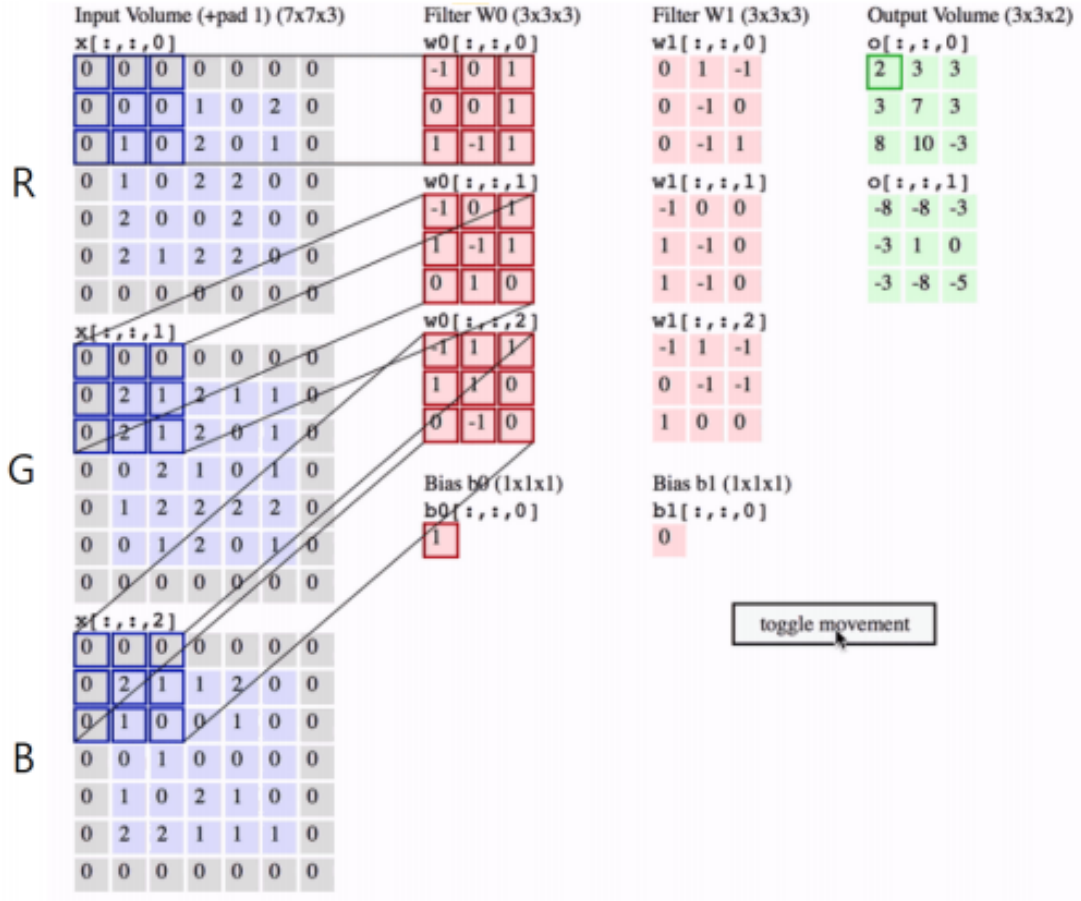
- 평균을 추출하는 filter

-1	0	1
-2	0	2
-3	0	3



- 세로 edge를 추출하는 filter
- 가로로 같은 색이 이어지면 +-상쇄
- 결과가 음수이면 절대값

1.Filter



- **input의 depth에 따라 kernel의 depth도 결정**

  ex1) 7 x 7 x 3(RGB) → 3 x 3 x 3 filter 사용

  ex2) 12 x 12 x 40 → 5 x 5 x 40 filter 사용

- **Feature map의 depth = Filter 개수**

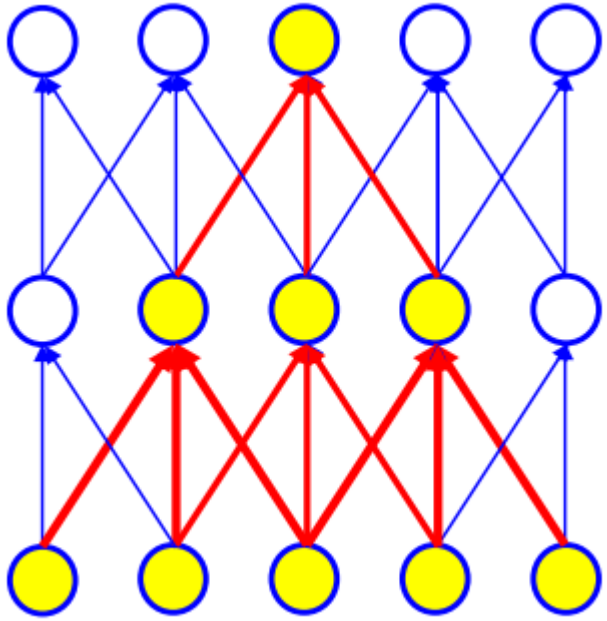
  ex) 3 x 3 x 3 filter 2개 → 3 x 3 x 2 feature map

1.Filter

Q. 그렇다면 CNN은 결국 부분만 보고 판단하는 게 아닌가?

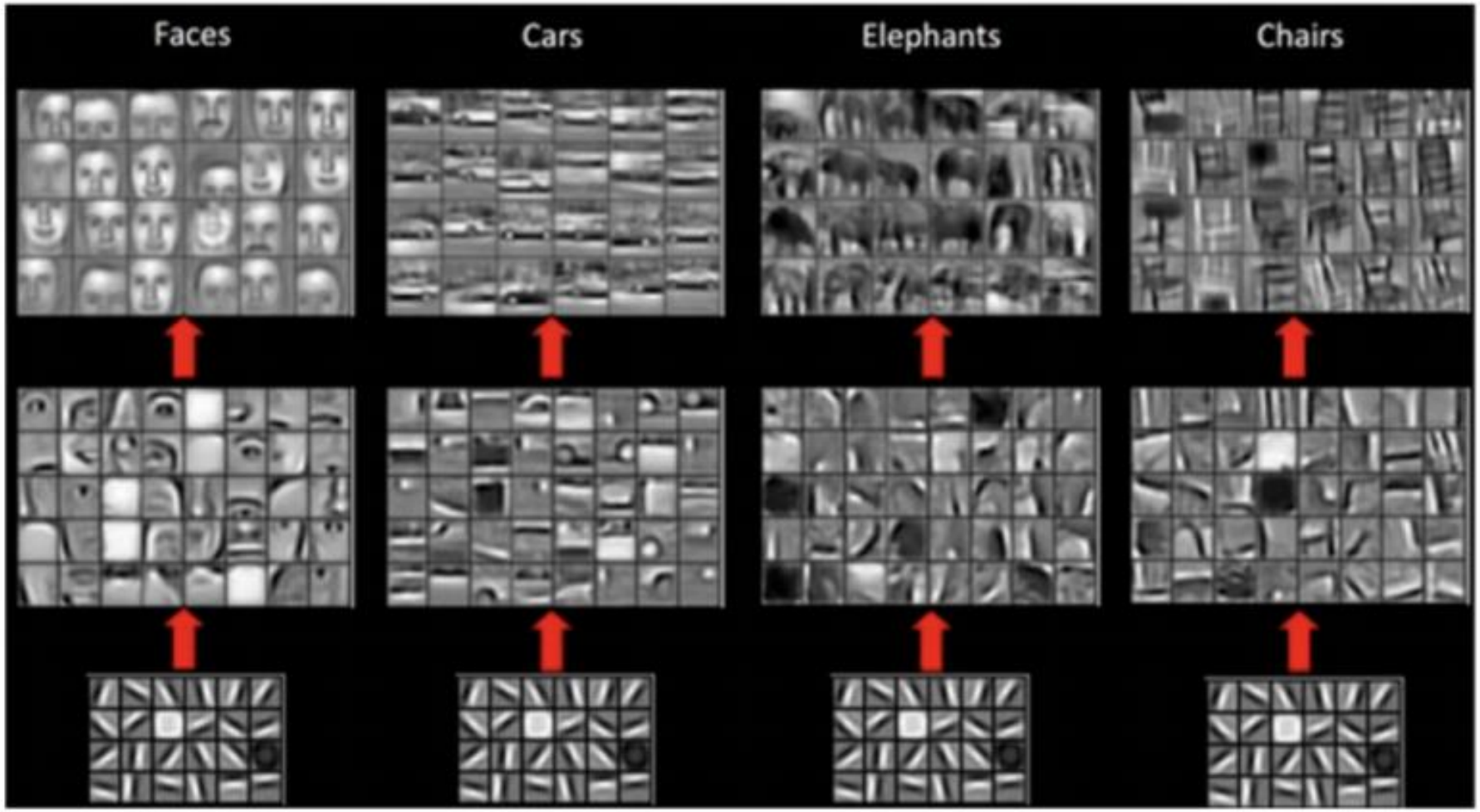
layer가 거듭되면서 조합을 만들기 때문에 층이 깊어질 수록 전체를 보는 것과 같아지는 것!

" CNN은 깊어질수록 powerful하다 "



- **CNN의 목적은 image를 잘 인식할 수 있도록 filter들을 잘 학습시키는 것**
- kernel을 사람이 설계하지 않고, 적절한 값을 학습으로 찾음
- ex) 2차원 영상이 7x7 kernel을 64개 사용한다면,
- 학습을 통해 (7x7+1)x64 = 3200개의 parameter를 찾아내야 함
- kernel 크기는 일반적으로 2x2, 3x3 많이 사용
- DMLP와 마찬가지로 Back Propagation을 통해 kernel을 학습

1.Filter



- 갈수록 **저차원적인 특징**(edge, corner,…) → **고차원적인 특징**(full object) 학습

2.Padding

1. 합성곱 1번 실행시 8X8에서 6X6으로 작아진다.
2. 가장자리 학습이 중앙부분에 비해 덜 학습된다.

77	80	82	78	70	82	82	140
83	78	80	83	82	77	94	151
87	82	81	80	74	75	112	152
87	87	85	77	66	99	151	167
84	79	77	78	76	107	162	160
86	72	70	72	81	151	166	151
78	72	73	73	107	166	170	148
76	76	77	84	147	180	168	142

$\otimes$

0.01	0.08	0.01
0.08	0.62	0.08
0.01	0.08	0.01

$=$

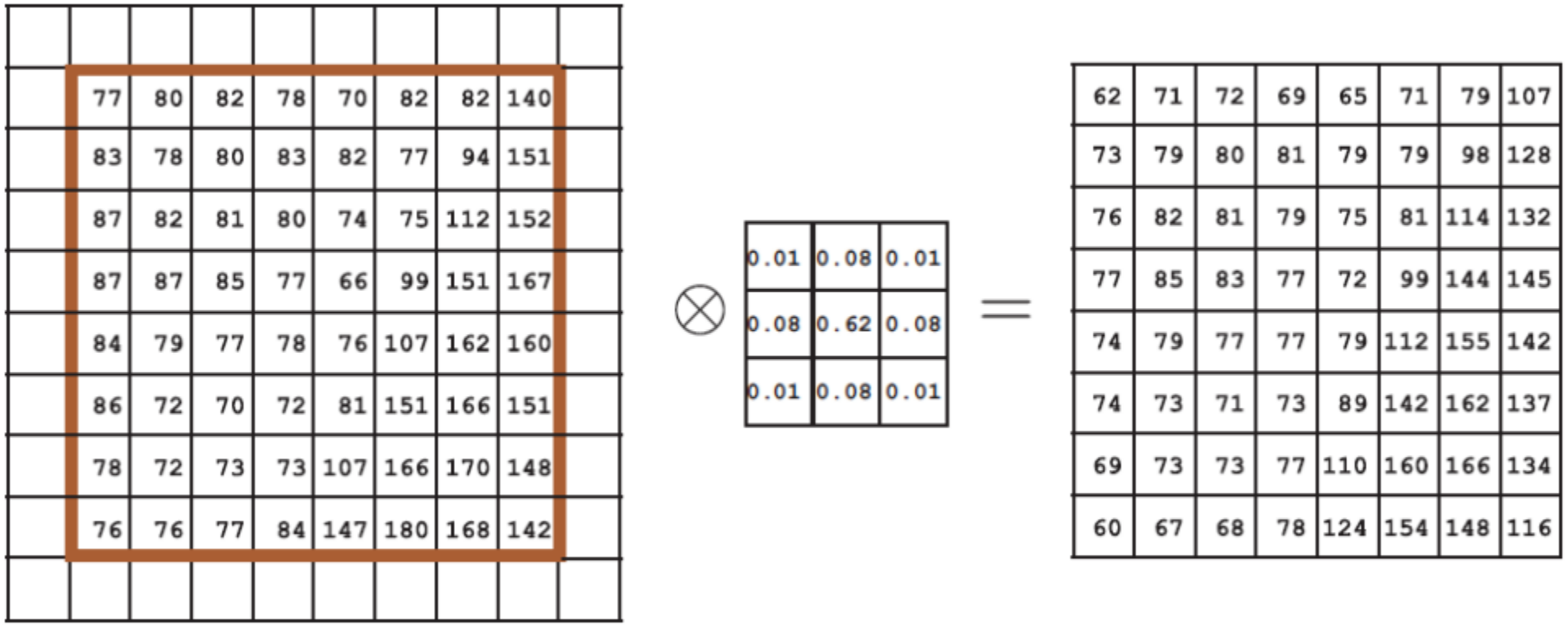
79	80	81	79	79	98
82	81	79	75	81	114
85	83	77	72	99	144
79	77	77	79	112	155
73	71	73	89	142	162
73	73	77	110	160	166

2.Padding

zero-padding 가장 많이 사용

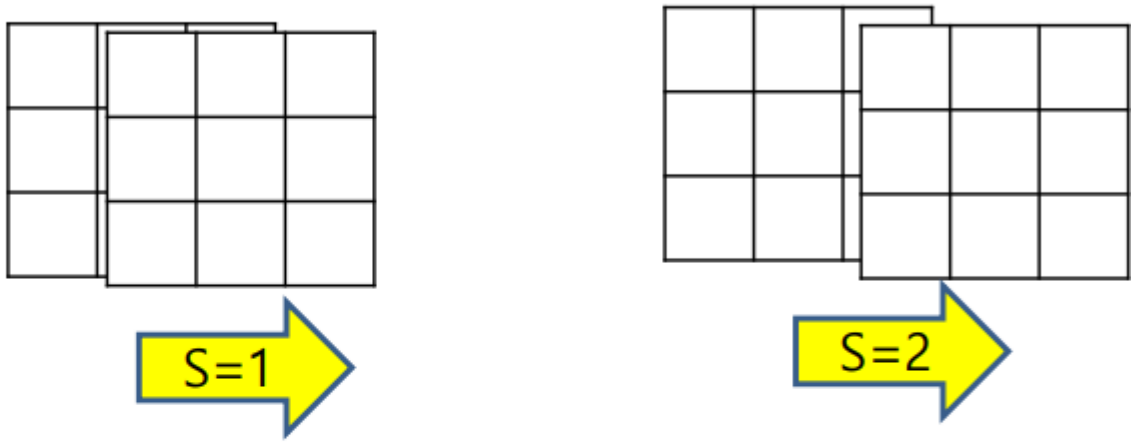
이미지 크기를 유지해야 할 필요가 있는 경우

- 깊이를 깊게 할 때
- Residual Net



3.Stride

Stride : filter 의 적용 위치 간격 s

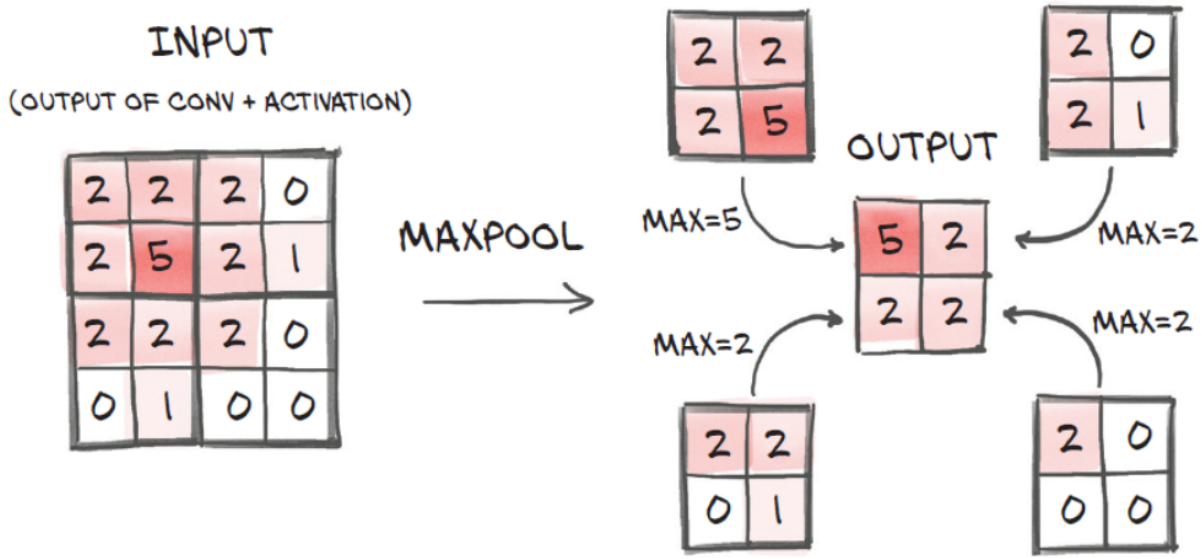


출력 이미지의 크기

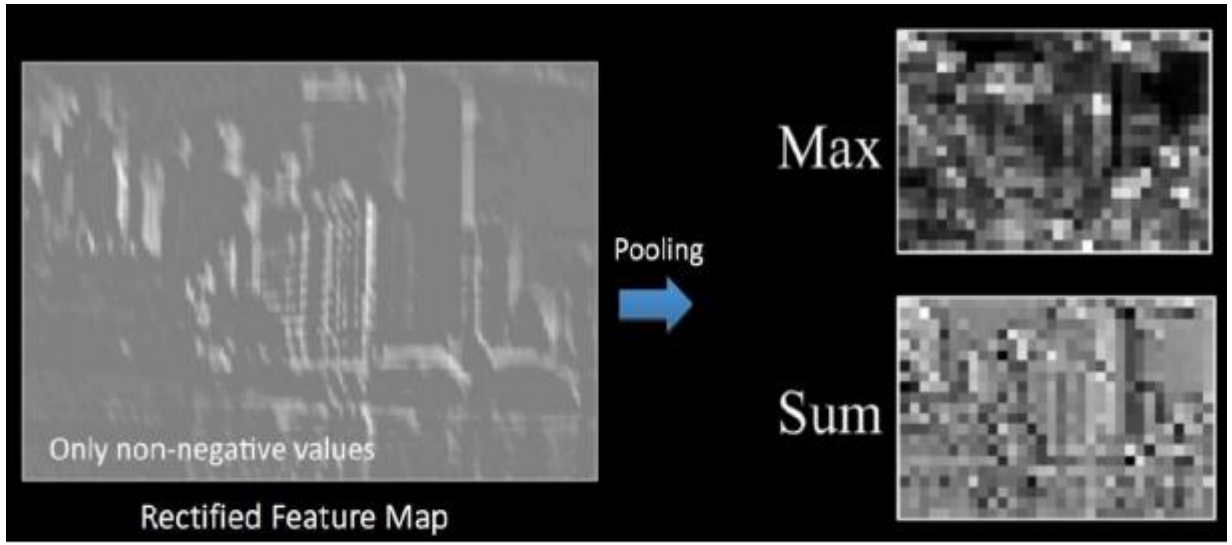
$$O = \frac{W - K}{S} + 1$$

W : image size
K : kernel size

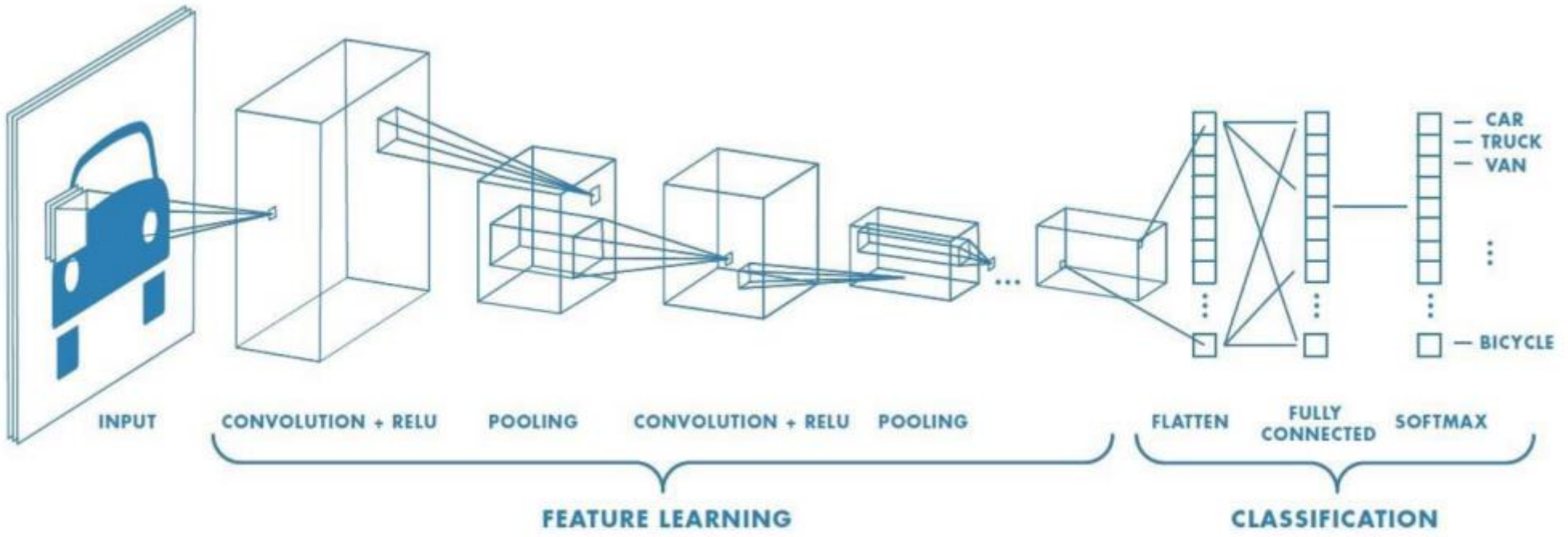
4.Pooling



- 인접한 셀은 비슷한 정보를 갖기 때문에 압축하여 효율성을 높인다(subsampling)
- 주로 max-pooling사용
- 평균값으로 하는 average pooling도 있다
- 각 feature map마다 독립적으로 downsampling
  - 따라서 파리미터가 없고 depth 변화가 없다

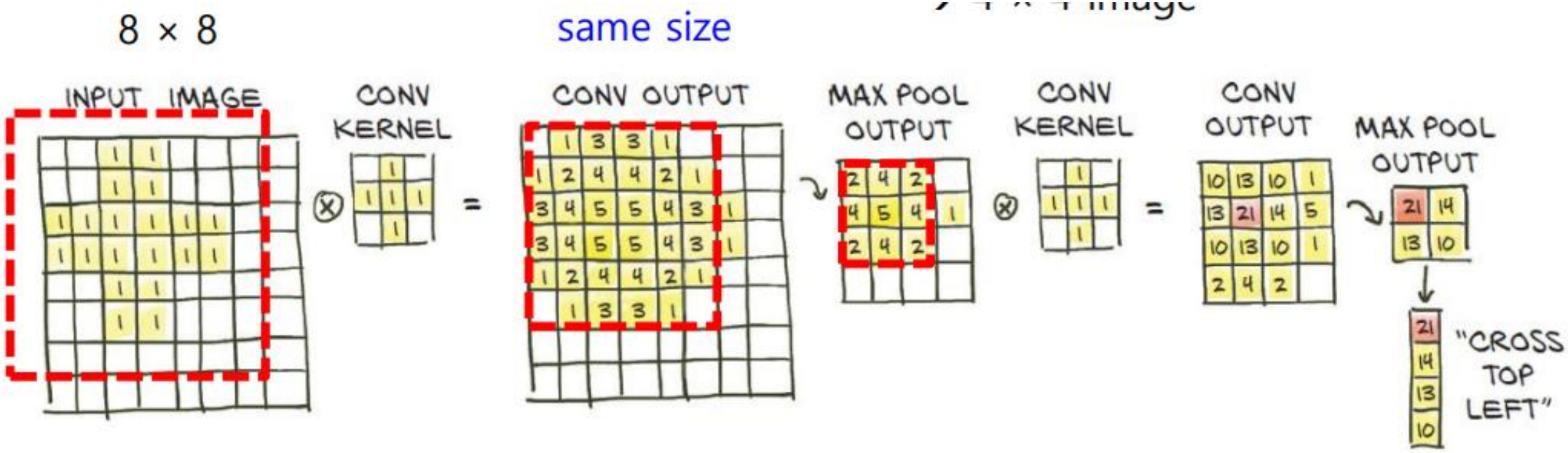


CNN Architecture



- Conv layer에서는 filter 계산 후 activation func(ex. ReLU) 적용
- Conv layer 후에는 선택적으로 pooling layer 적용
- 마지막에 fc layer 쌓아서 target class별로 score 계산 (classification task)

CNN Architecture



low-level features

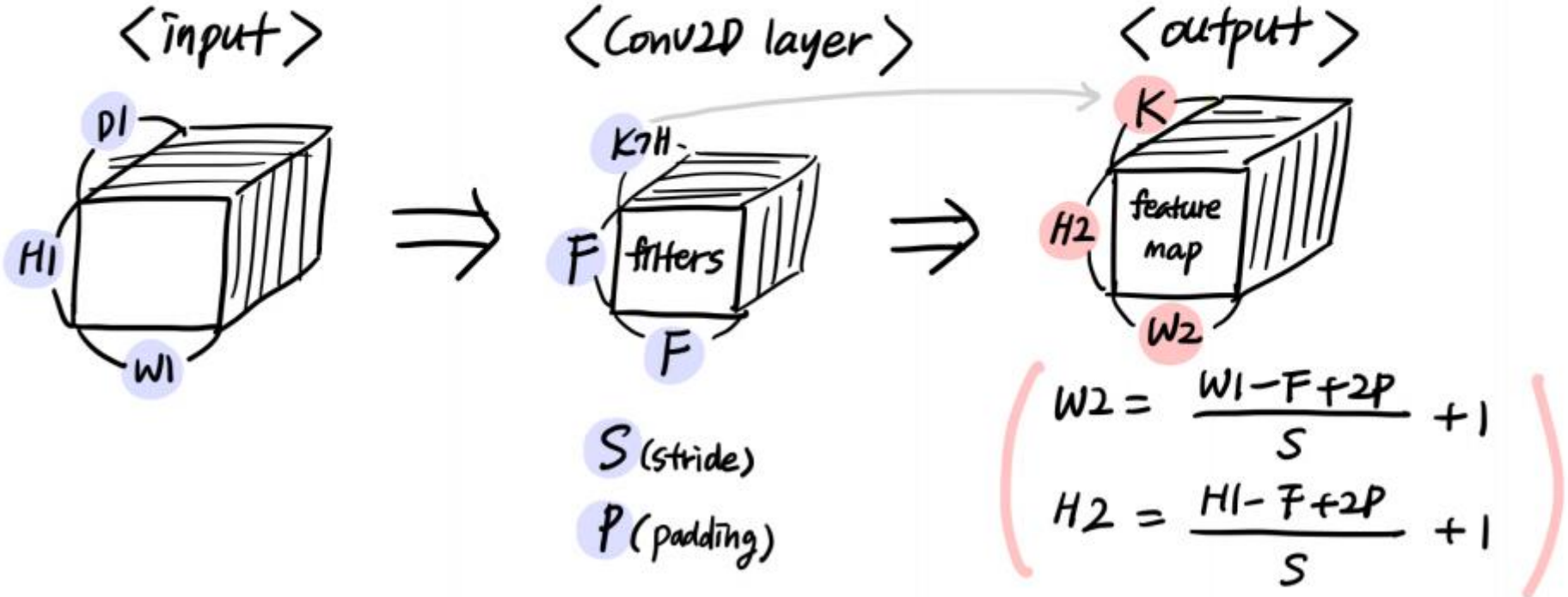
- A given output neuron of the 3 × 3-conv, 2 × 2-max-pool, 3 × 3-conv construction has a ***receptive field*** of 8 × 8.

feature map shape 및 parameter 개수계산법

Output Shape

1) Conv2D layer

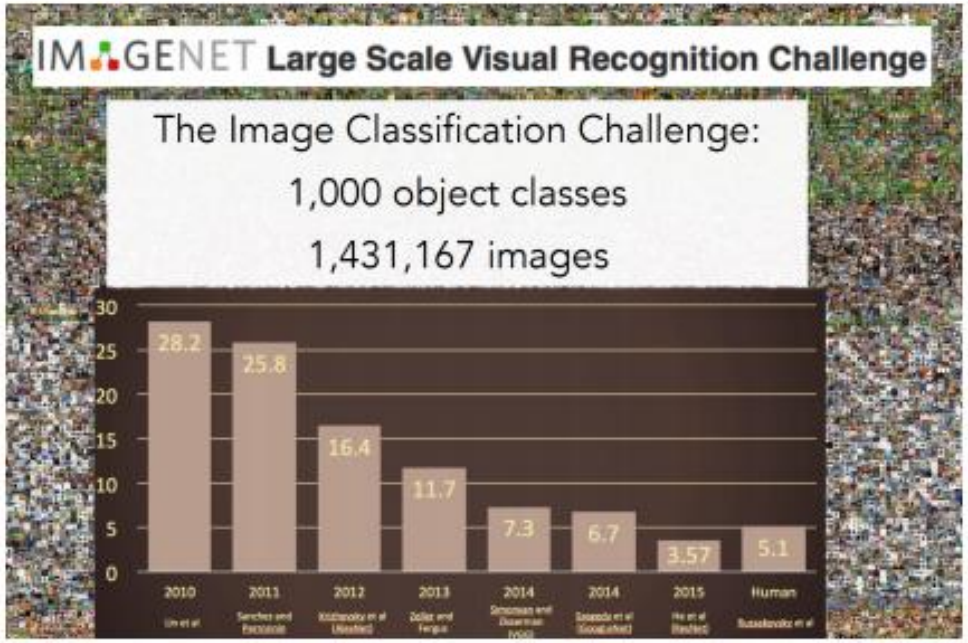
parameter 수
→ kernel 마다 (F*F*D1)개의 weight, 1개의 bias
→ 총 K{(F*F*D1)+1} = K(F*F*D1)+K개

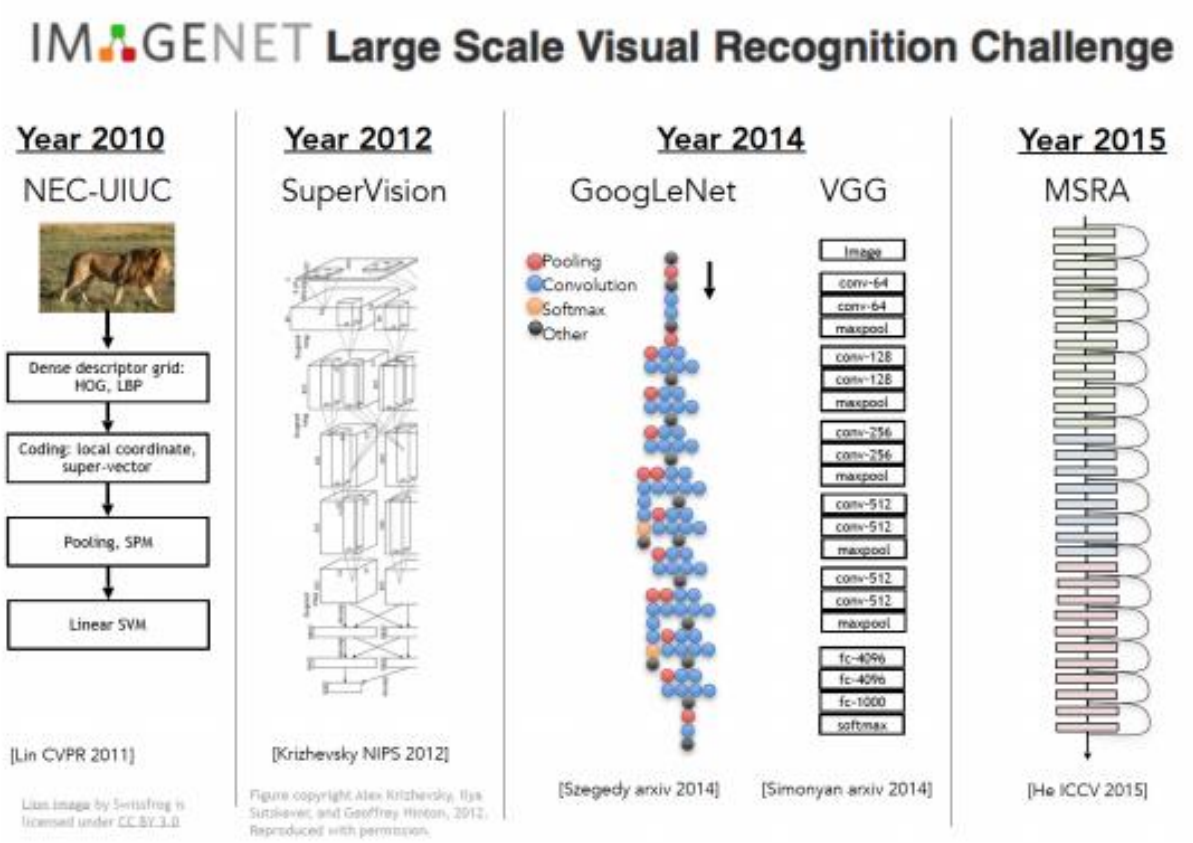


$$W2 = \frac{W1 - F + 2P}{S} + 1$$

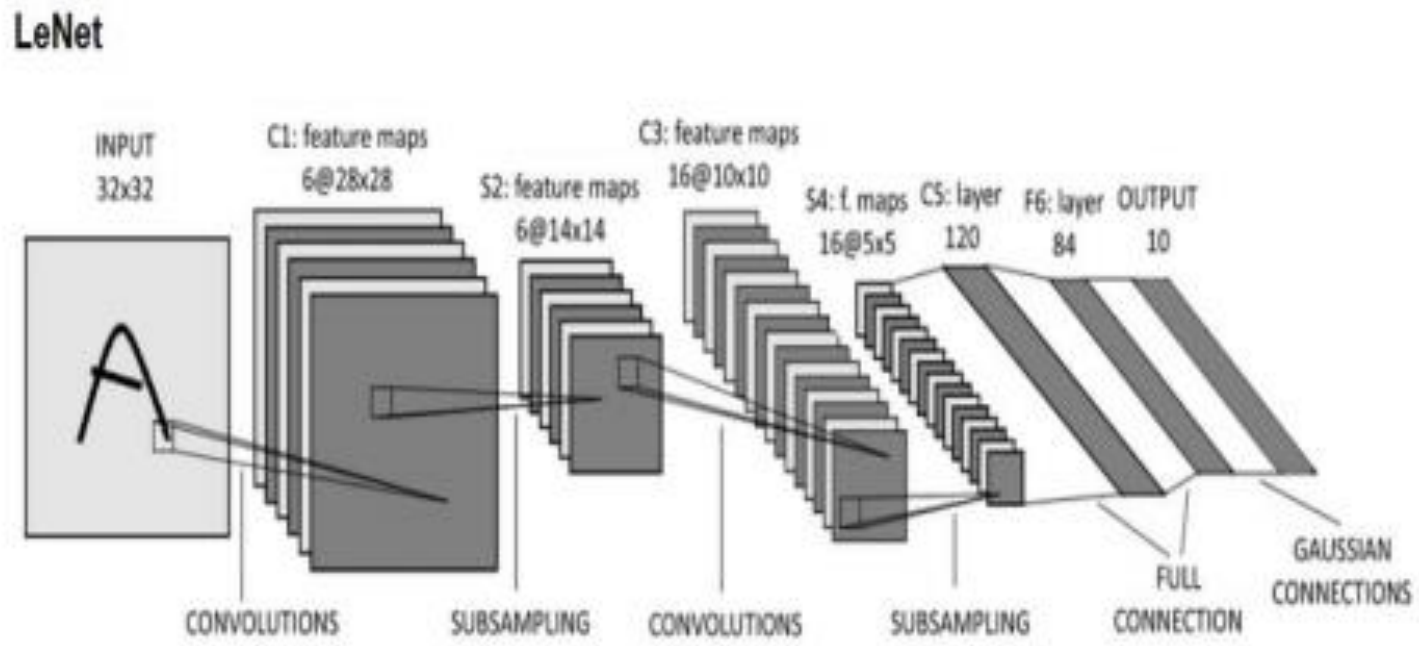
$$H2 = \frac{H1 - F + 2P}{S} + 1$$

대표적인CNN Architecture들



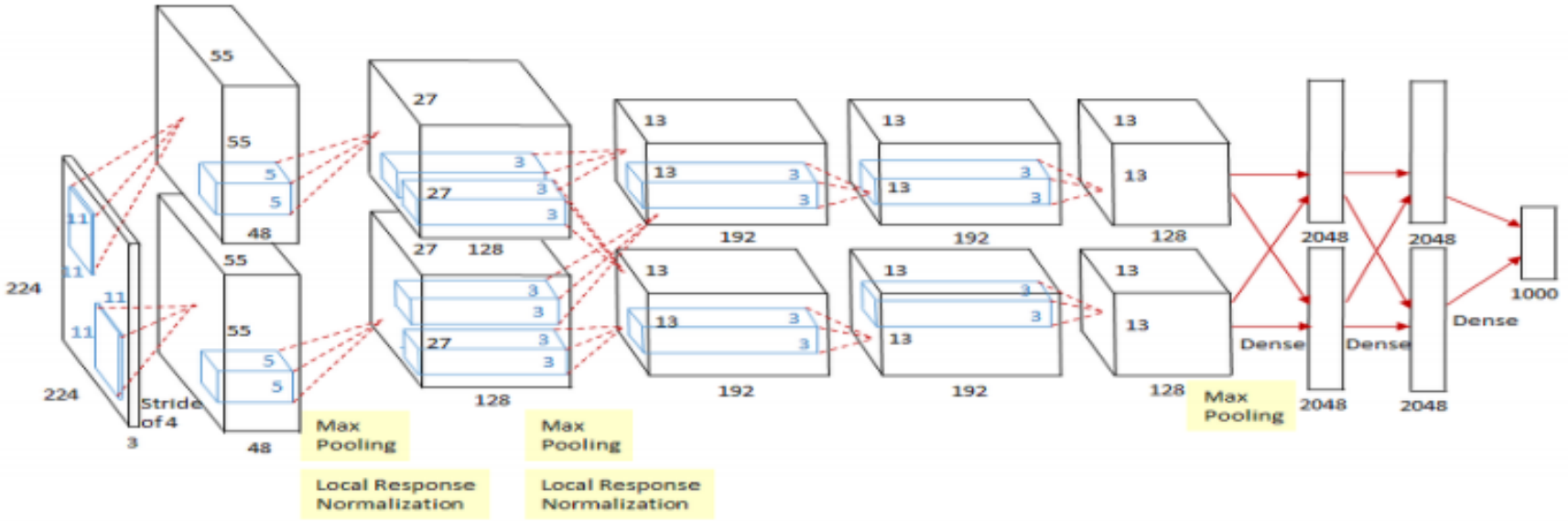


LeNet



- 최초로 CNN구조를 Backpropgation을 사용해 학습
- 활성함수로 Sigmoid
- Average Pooling 사용

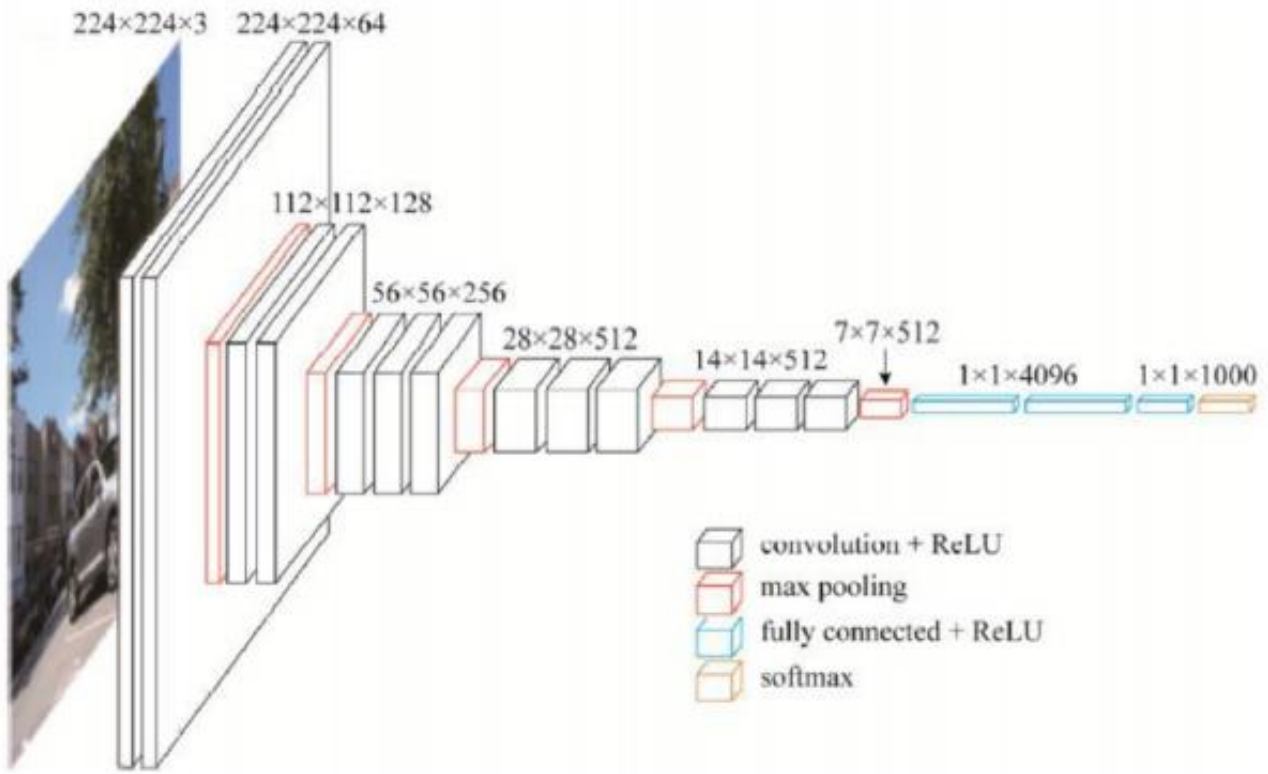
AlexNet



- 활성화 함수로 ReLU 사용
- Max pooling 사용
- GPU 사용

VGGNet

- 핵심 아이디어
  1) **3*3의 작은 kernel 사용**
  2) **신경망을 더욱 깊게 (신경망의 깊이가 성능에 어떤 영향을 미치는지 확인)**
- **VGG-16 (Conv layer 13층 + FC layer 3층)**
  - AlexNet에 비해 3배 깊어짐



VGGNet

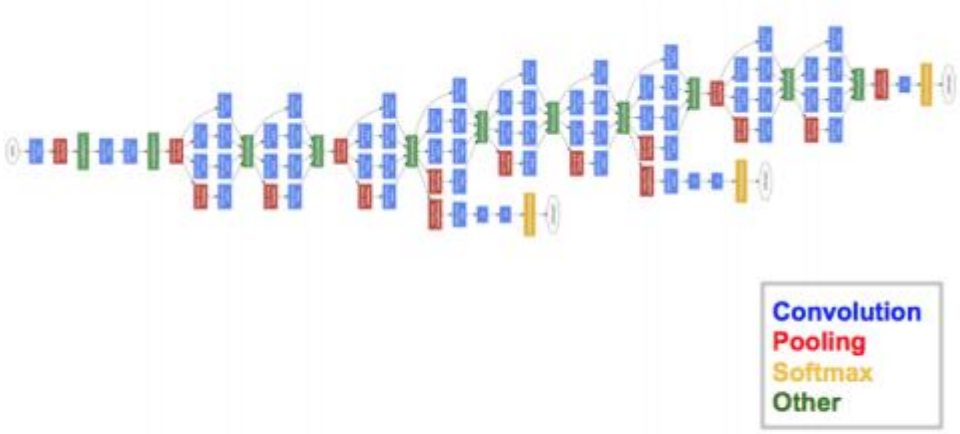
작은 filter을 사용하는 이유
- 큰 크기의 필터는 여러 개의 작은 필터로 분해될 수있다
    1. 더 깊어지는 효과
    2. 파라미터 감소

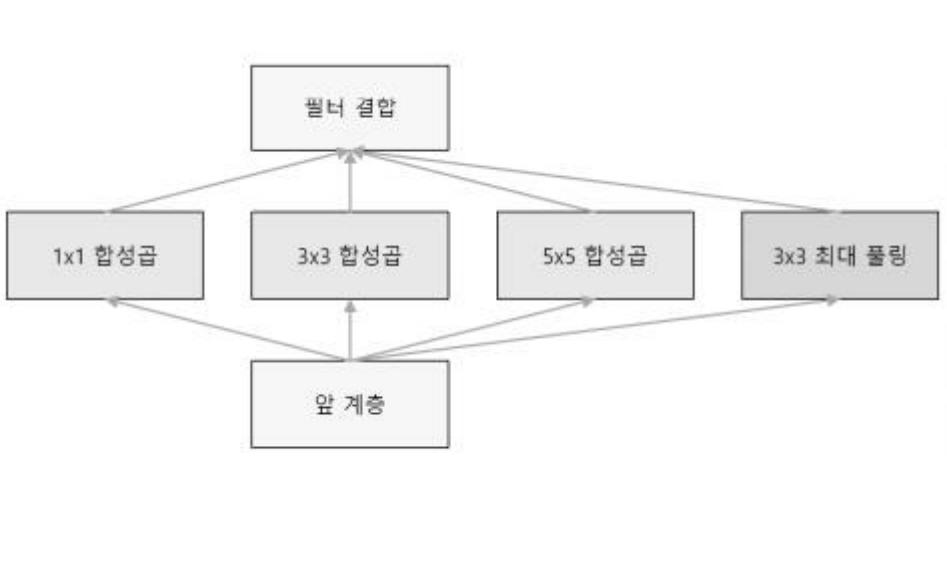
ex) 1개의 7x7 layer 와 3개의 3x3 layer일때 파라미터 비교

$$3\left(3^2C^2\right) = 27C^2 \quad < \quad 7^2C^2 = 49C^2$$

Good! **Bad!**

GoogLeNet



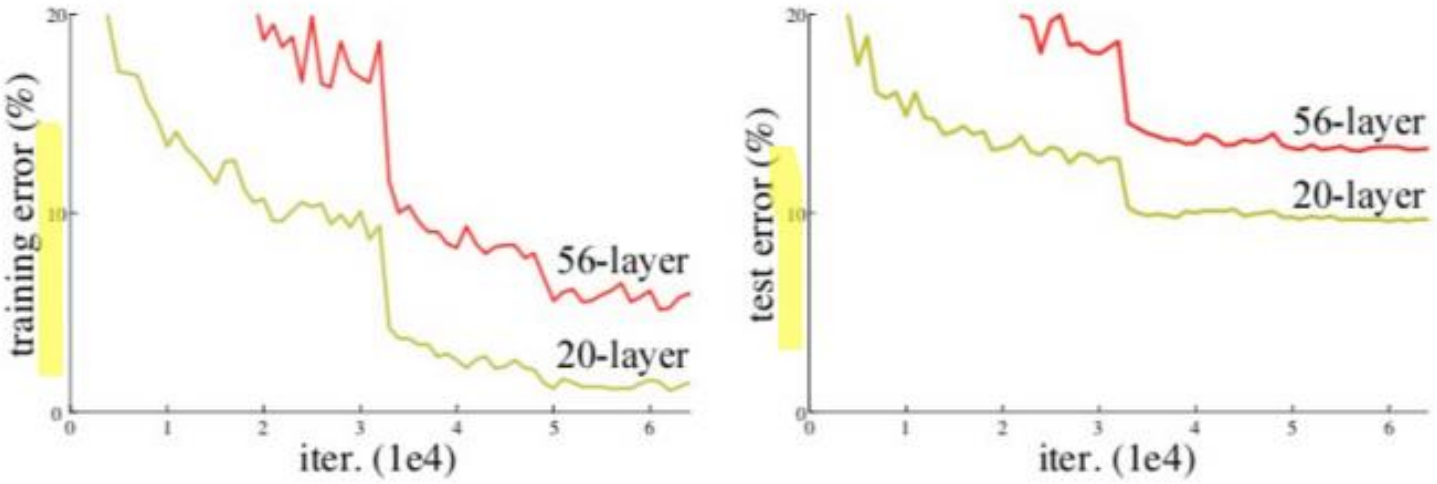


- 하나의 layer에서도 다양한 종류의 filter, pooling을 도입
- Inception Module(Block) 사용
- Fully connected layer 대신 Global Average Pooling Layer 사용

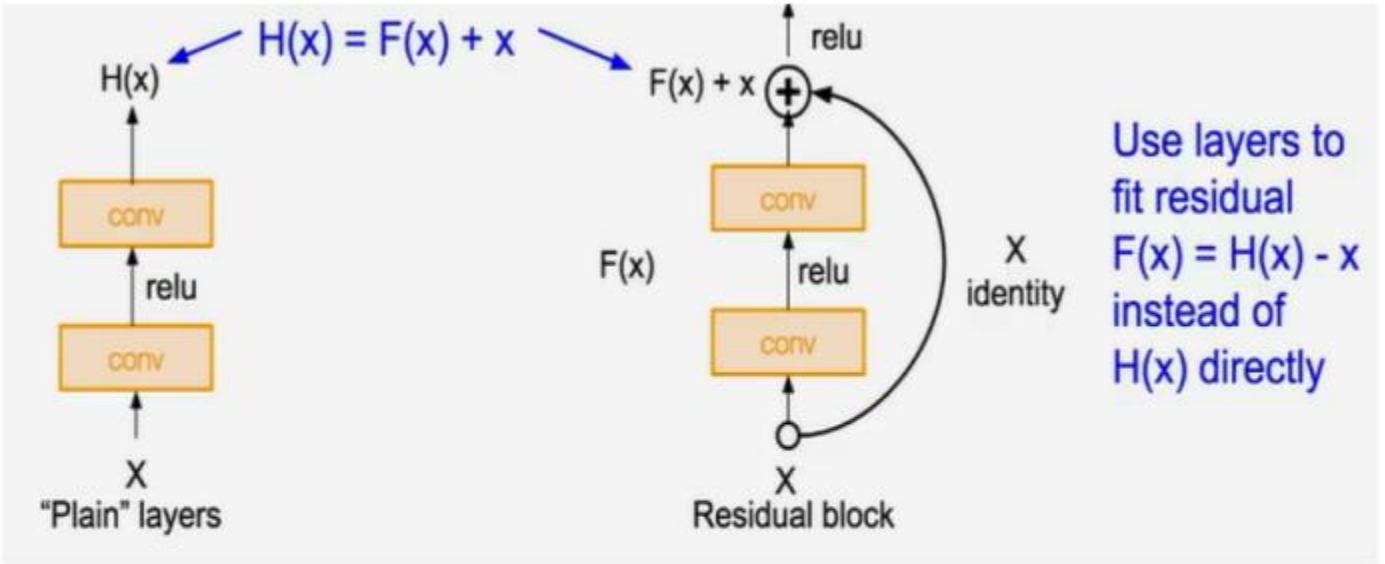
ResNet

이전 모델의 한계

- 모델이 깊어 질수록 성능이 높다 생각했다.
- 하지만 일정 depth 이상 깊어지니 성능 저하가 발생
- “ **Degradation Problem** “
  - network의 depth가 증가함에 따라 accuracy가 saturate되어 degrade가 점점 빨라지며, 이는 overfitting 때문에 발생하는 것이 아님



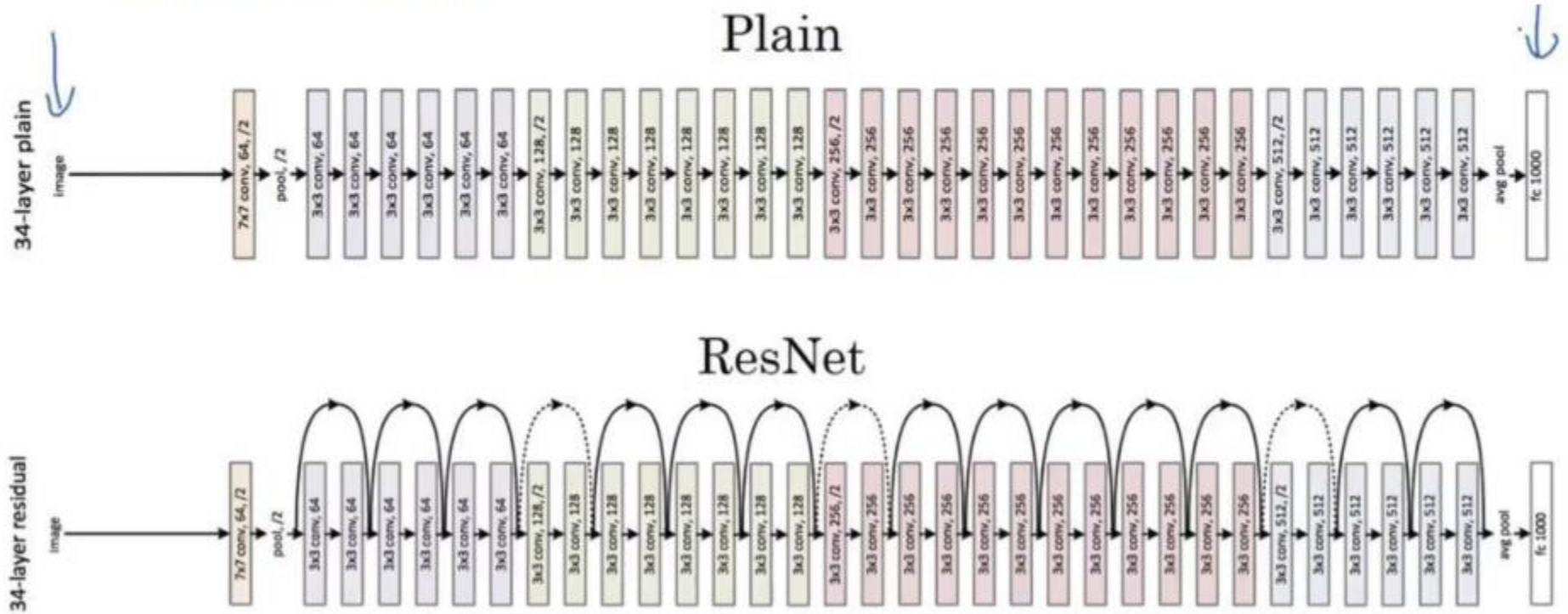
ResNet



- **" Skip Connection ", " Shortcut Connection "**
  - **1개 이상(보통 2개)의 layer를 skip하여 input을 전달하는 방법**
  - 학습이 더 쉬워지는 경향 있음
- ResNet은 이런 **Residual Block**을 사용하여 **성능 저하를 피하면서 layer 수를 대폭 늘림** (최대 1202층)

ResNet

- Residual Learning
- Global Average Pooling 사용 (FC layer 제거)
- Batch Normalization 적용



Applications

Image Classification

This image is CC0 public domain

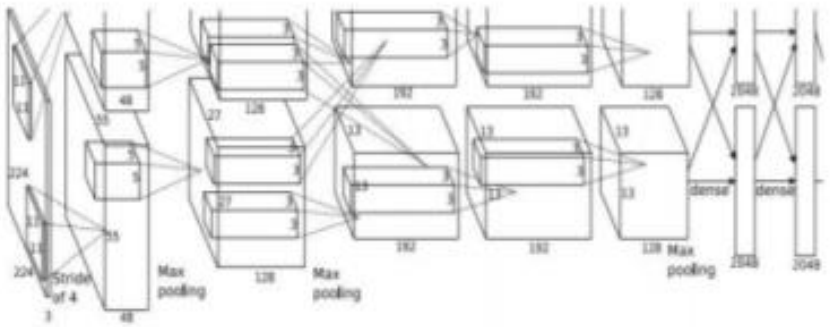


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

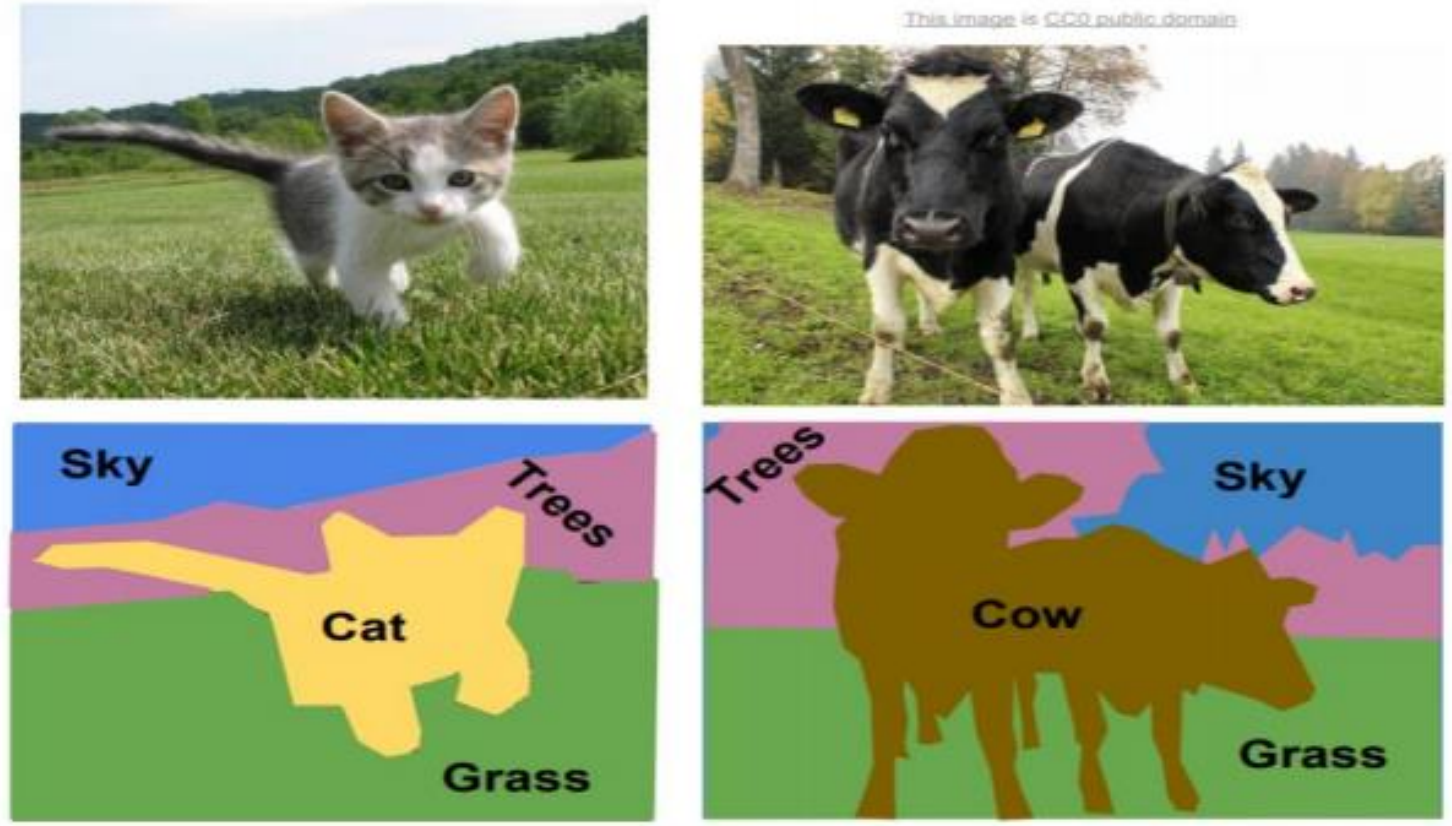
Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Applications

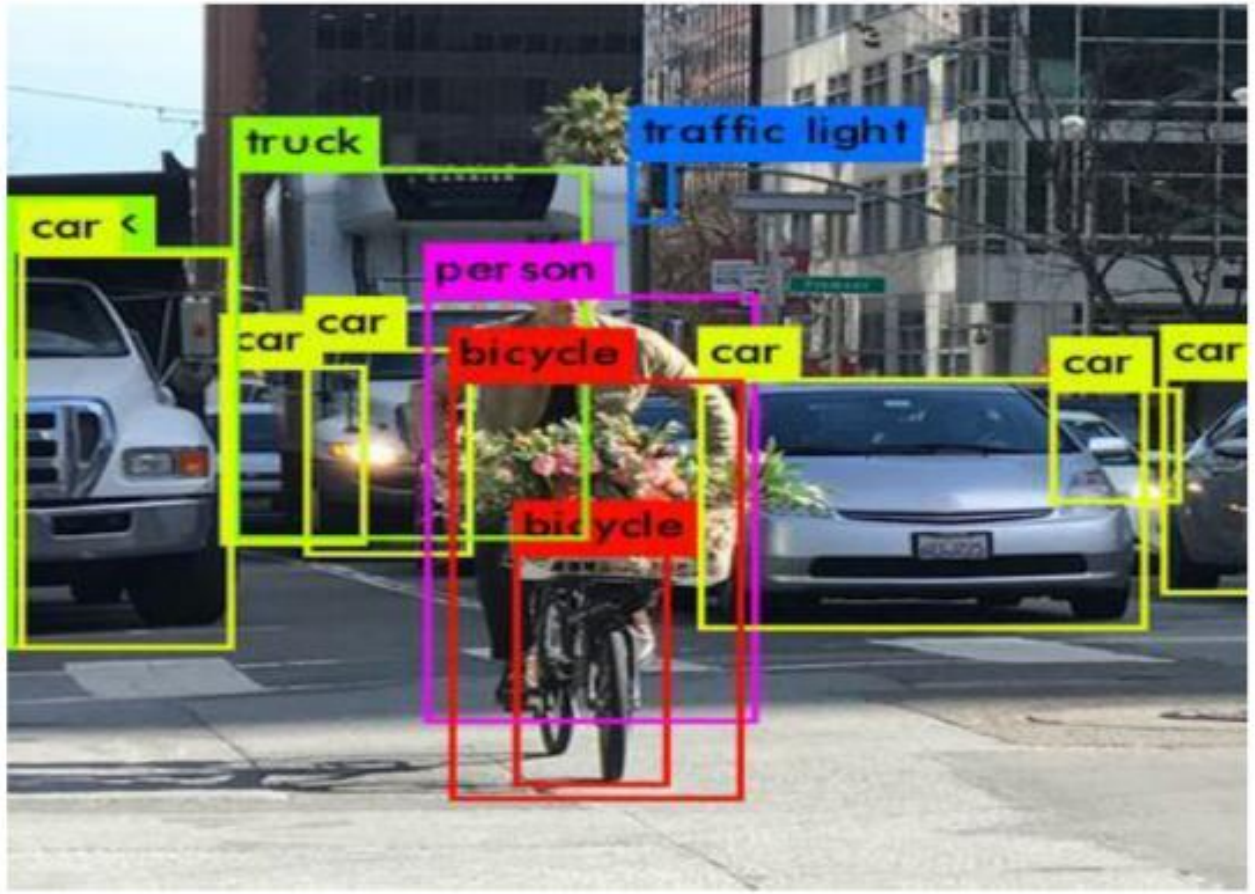
Semantic Segmentation



- 이미지 픽셀별로 분류

Applications

Object Detection



Applications

Image Segmentation

Semantic Segmentation + Object Detection