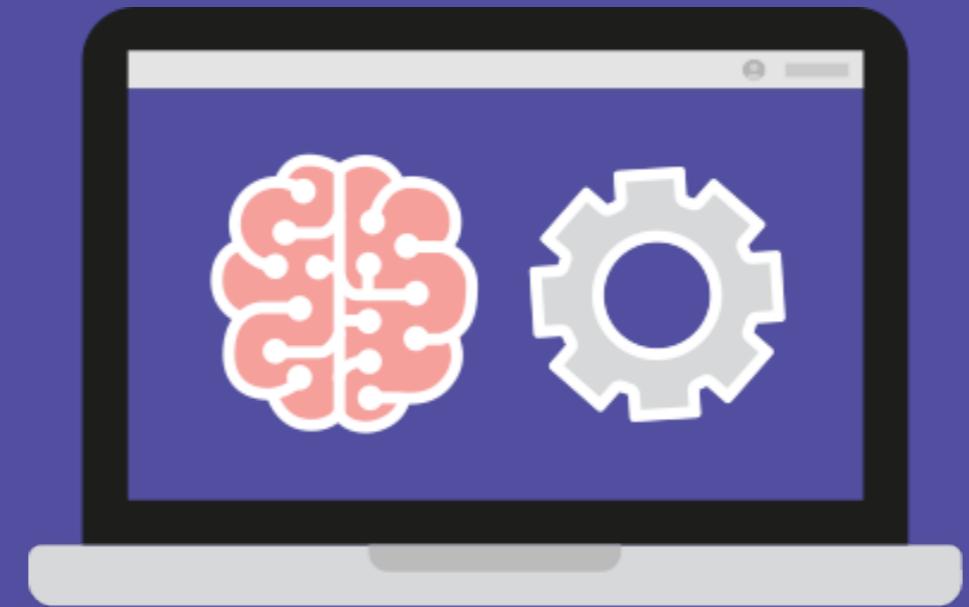


/\* elice \*/

# 양재 AI School 인공지능 캠프

Lecture 17

CNN Architectures



박상수 선생님

# 커리큘럼

- 1  **컨볼루션 신경망의 구조 (CNN Architectures)**

컨볼루션 신경망 구조에 대해 학습 합니다.
- 2  **다양한 컨볼루션 신경망**

다양한 컨볼루션 신경망의 구조를 파악하고  
신경망에 적용된 정확도를 개선하기 위한 방법을 학습합니다.

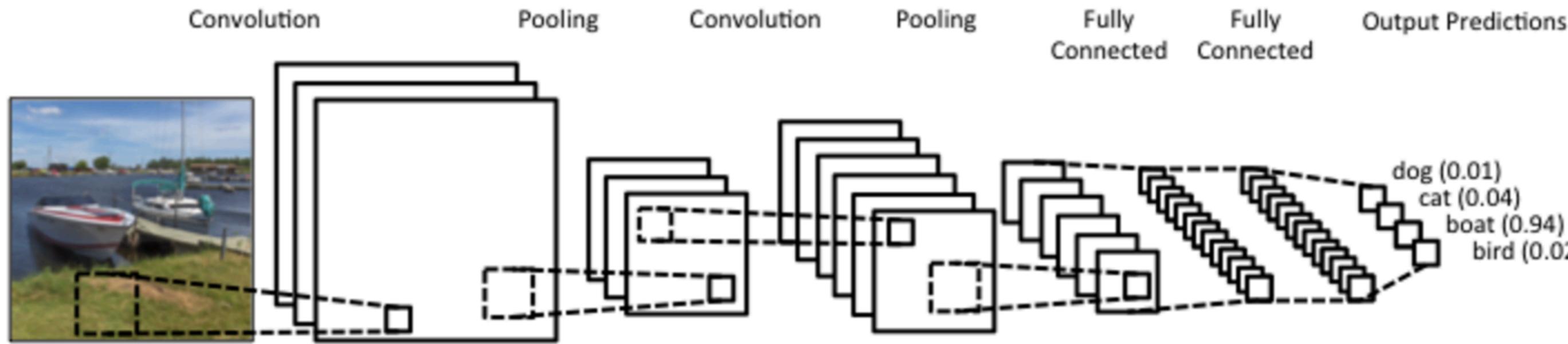
# 목차

1. 컨볼루션 신경망 리뷰
2. 벤치마크 (이미지넷)
3. AlexNet
4. VGGNet
5. GoogLeNet
6. ResNet

컨볼루션 신경망

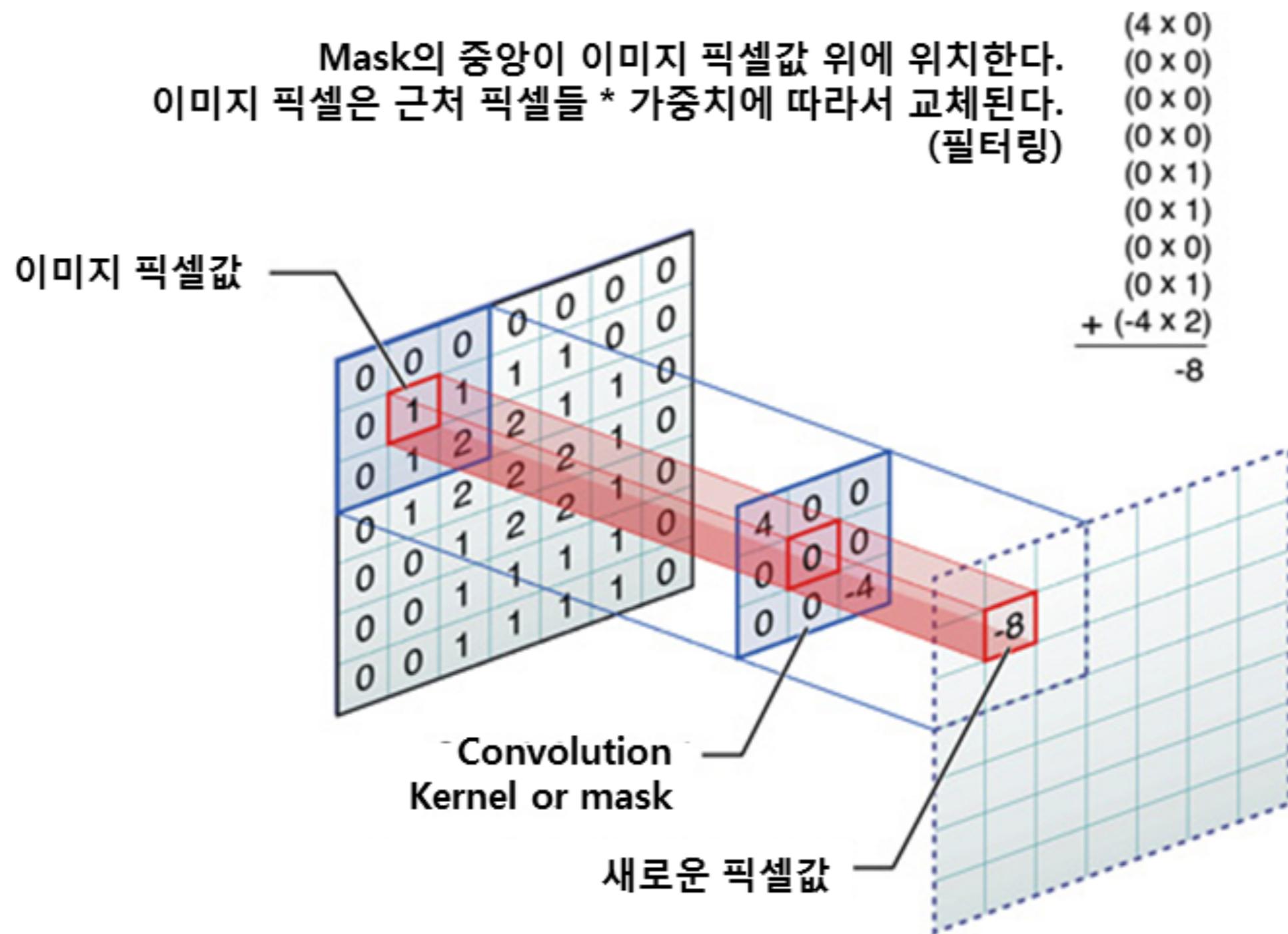
인공신경망 + 컨볼루션

# 컨볼루션 신경망



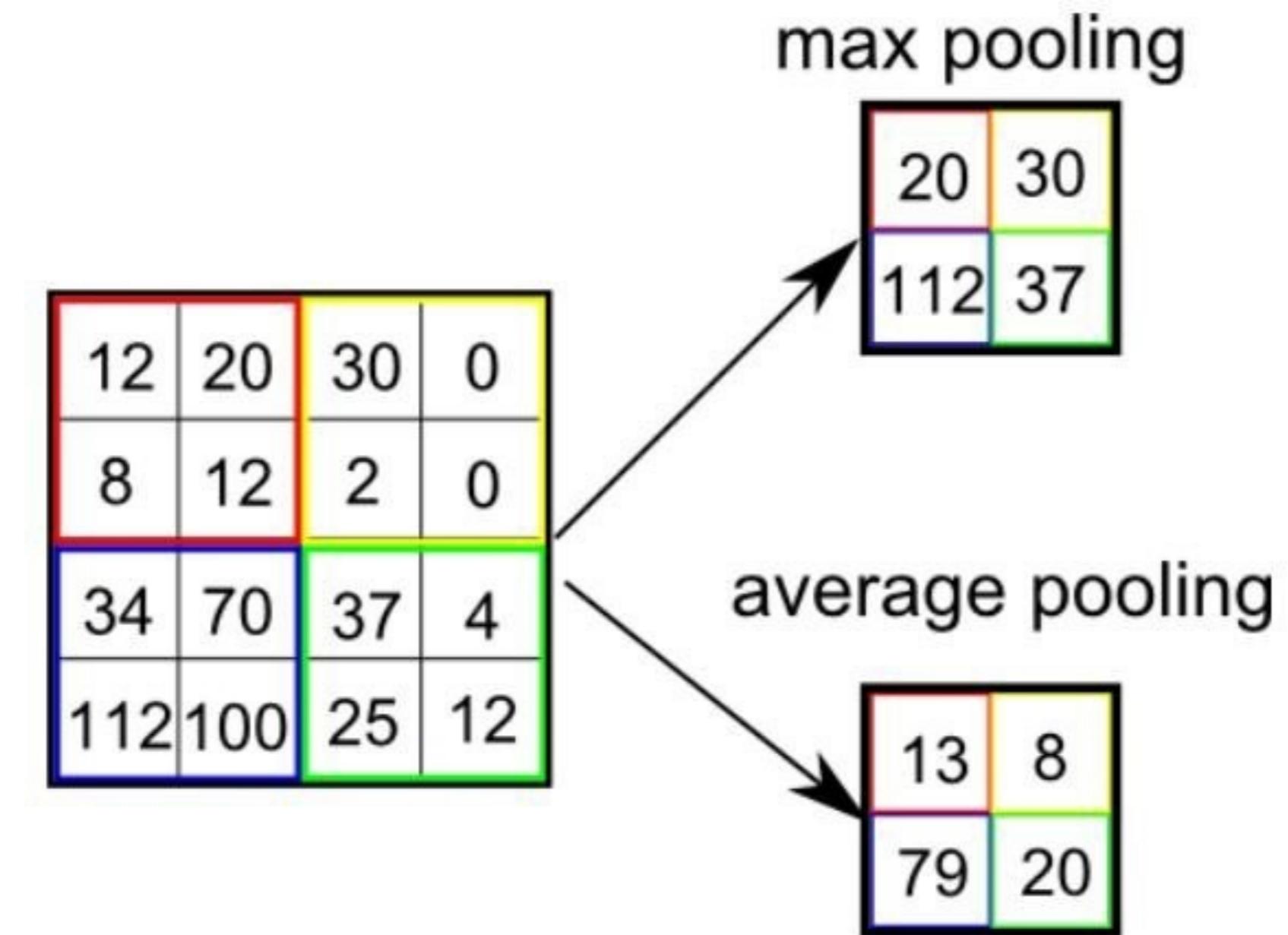
**Convolution, Pooling, Fully Connected Layer**

# Convolution Layer



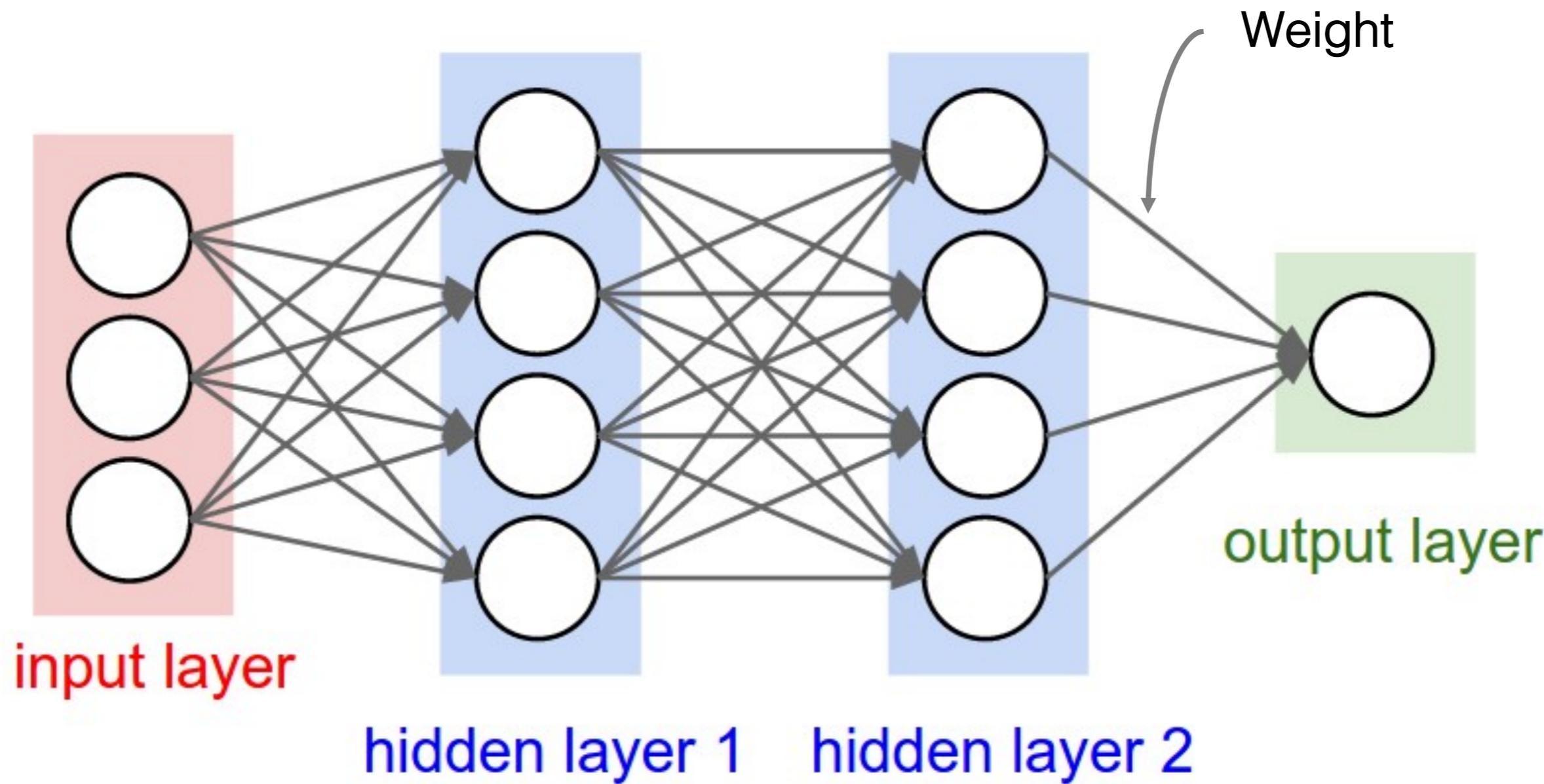
Element-Wise Matrix Multiplication

# Pooling Layer



피쳐맵의 크기를 줄이는 과정

# Fully Connected Layer



특징을 입력으로 하여 분류

# 벤치마크

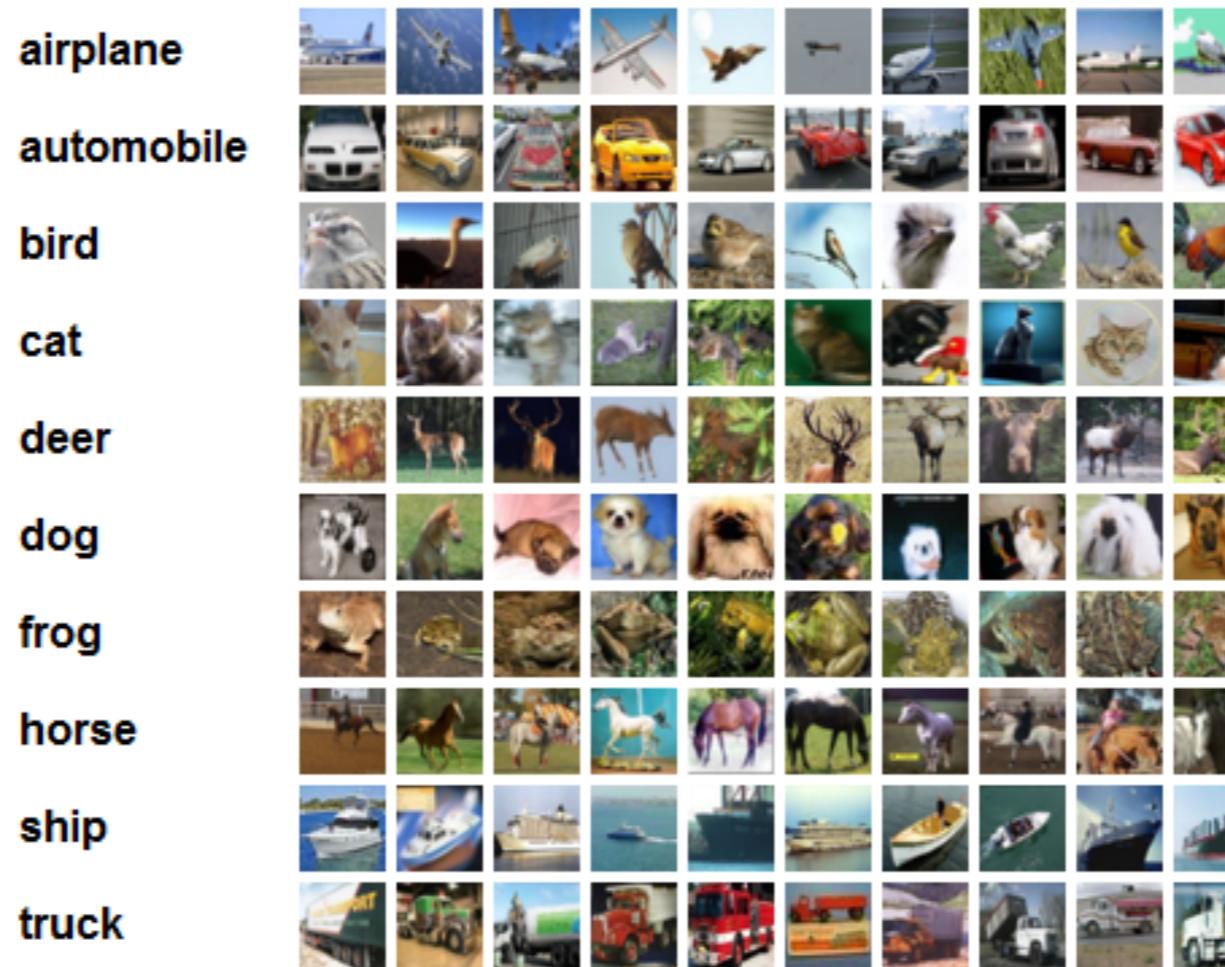
투자의 성과를 평가할 때 “기준이 되는 지표”

# MNIST 벤치마크



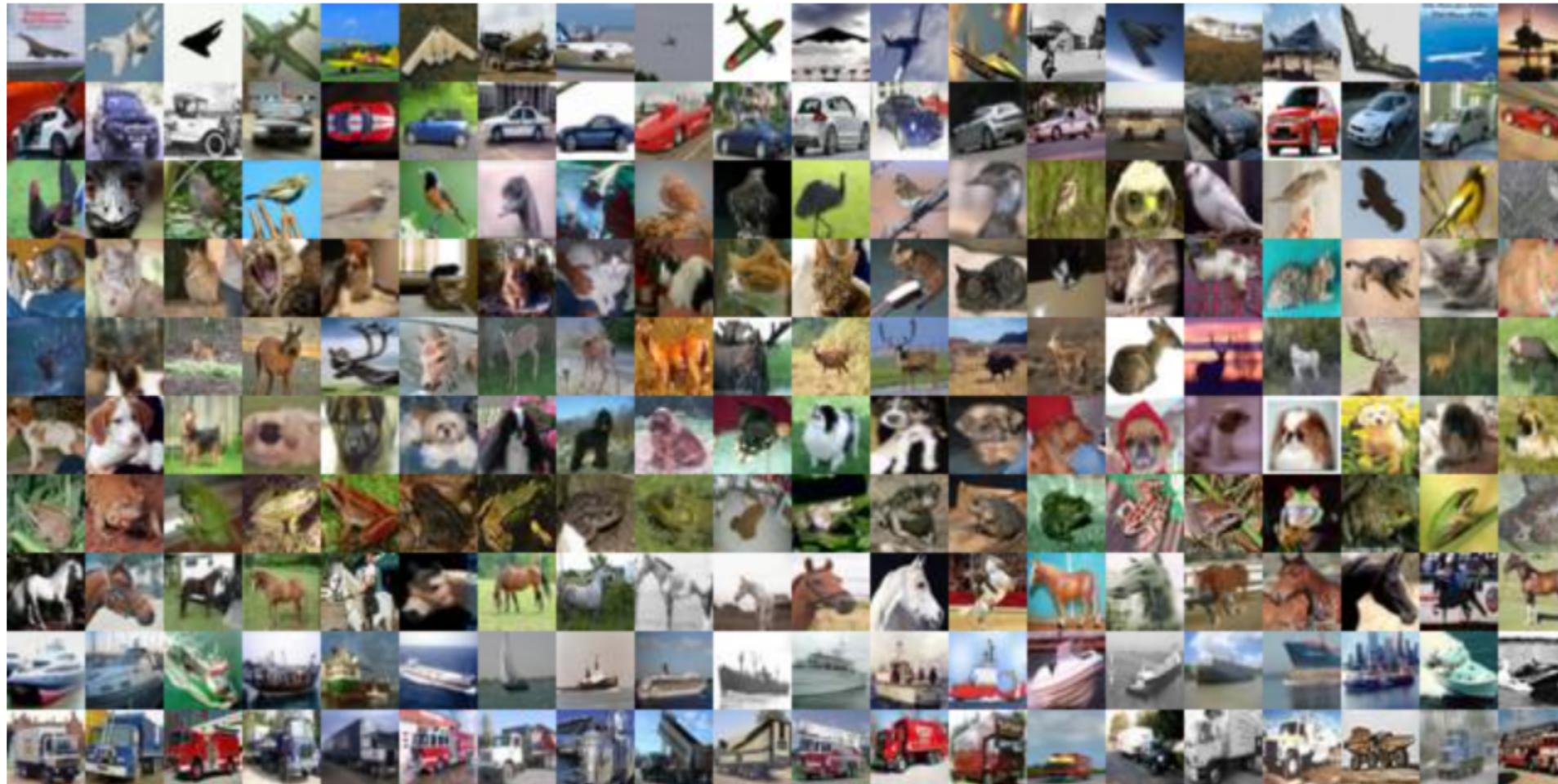
손으로 쓴 숫자들로 이루어진 대형 데이터 베이스  
필기체 숫자 하나는 28\*28 크기의 이미지  
**60,000개의 트레이닝 이미지, 10,000개의 테스트 이미지**

# CIFAR-10/100 벤치마크



10/100개의 사물 사진을 모아 놓은 데이터 베이스  
이미지 한장당 크기는  $32*32*3$  크기의 이미지  
50,000개의 트레이닝 이미지, 10,000개의 테스트 이미지

# ImageNet 벤치마크



ILSVRC 챌린지에서 사용되는 벤치마크

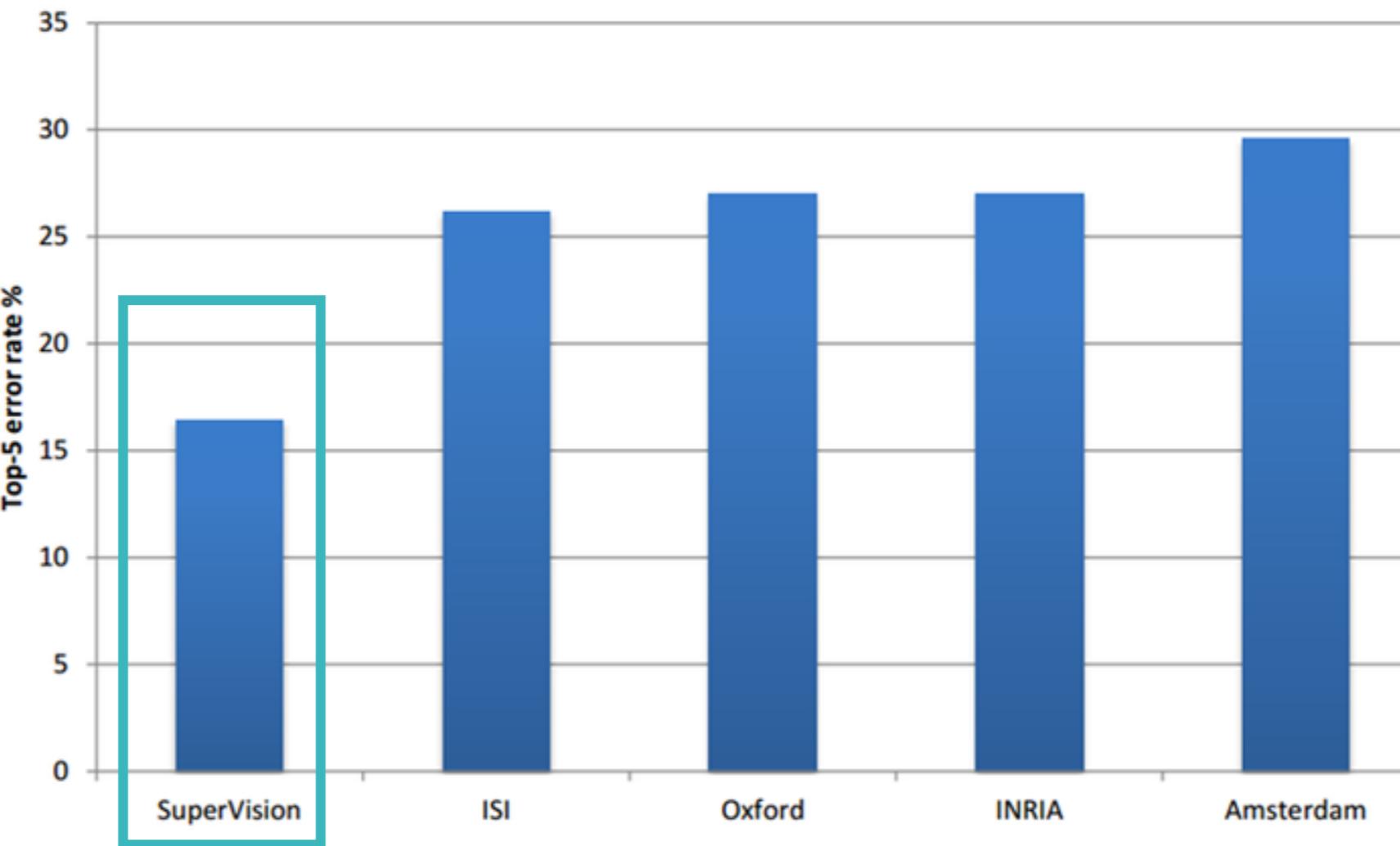
224\*224\*3 크기의 입력 이미지

1,000 종류의 1,281,167개의 데이터 (200GB 이상)

# AlexNet

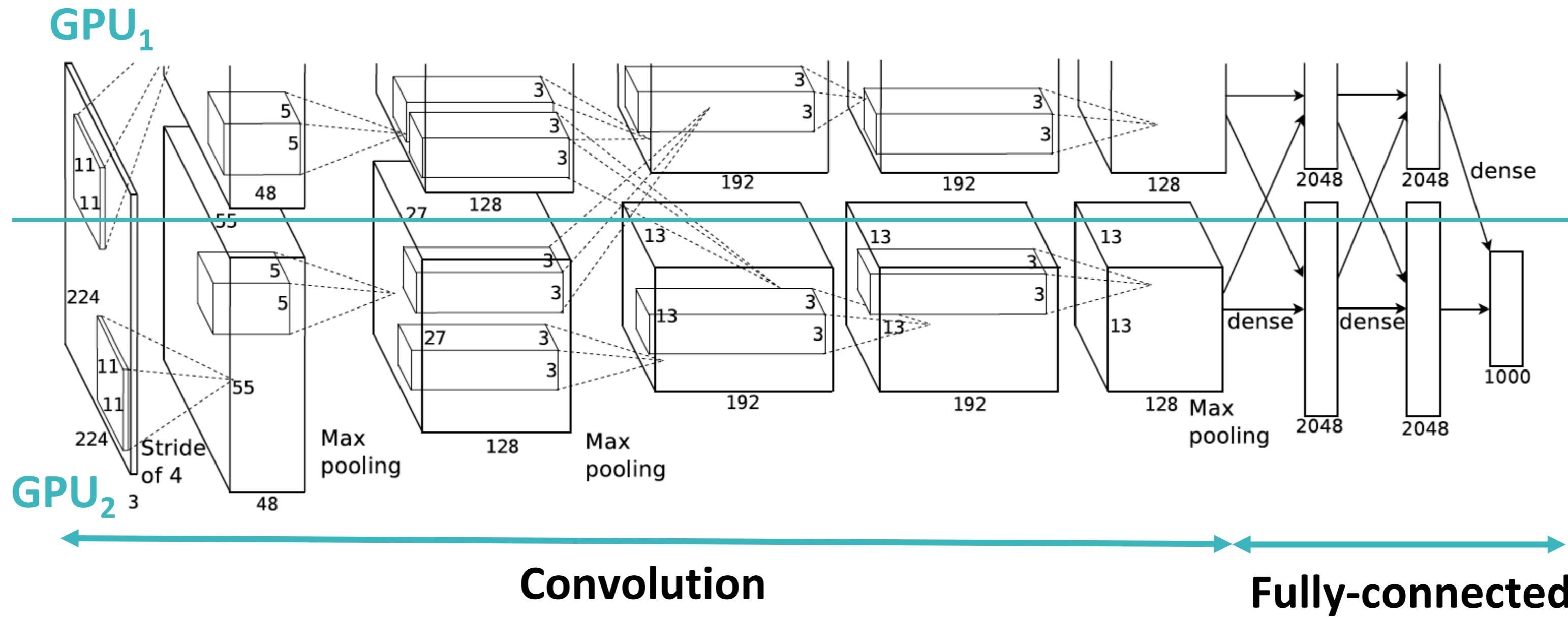
## ILSVRC 2012 우승 모델

# 우수한 정확도



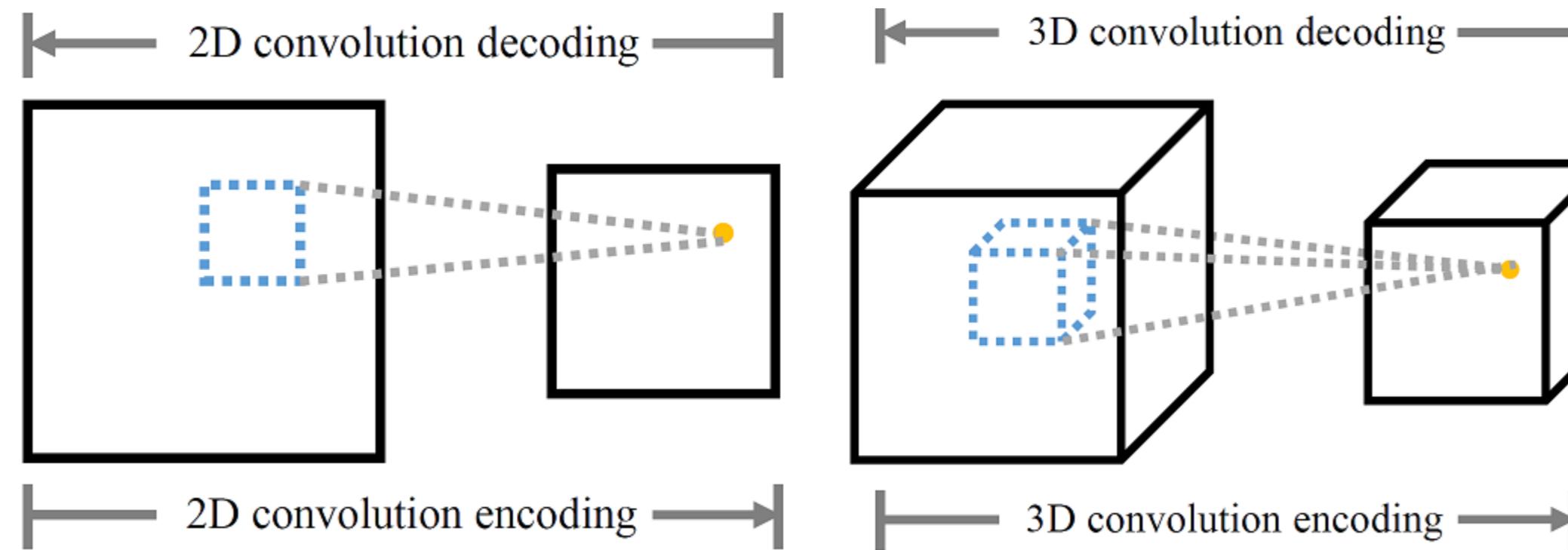
분류기는 주어진 이미지가 어떤 클래스에 속하는지 분류  
{고양이, 강아지, 집, 축구공, 사자} = {0.1, 0.6, 0.05, 0.05, 0.2)  
Top-1 (강아지), Top-5 (상위 5개 클래스)

# 2개의 GPU를 사용한 구조



5개의 Convolution 레이어, 3개의 Fully-connected 레이어로 구성  
2개의 GPU를 사용하여 학습 (GTX580, 3GB)  
GTX580은 메모리 용량 한계로 2대를 사용함

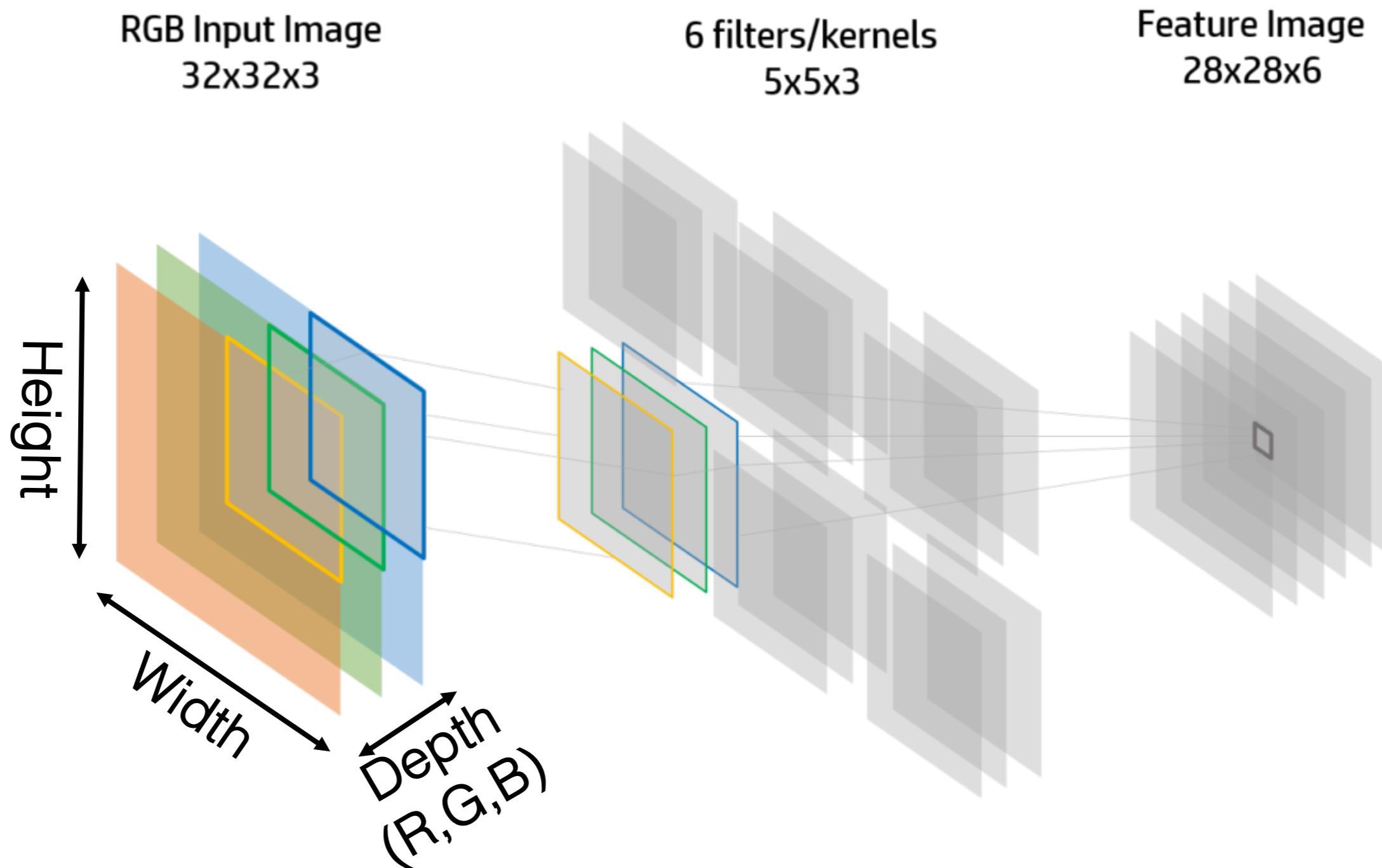
# 3차원 구조의 컨볼루션 레이어



**2D Convolution:** LeNet-5의 입력 이미지 ( $32*32$ )와 커널 ( $5*5$ )

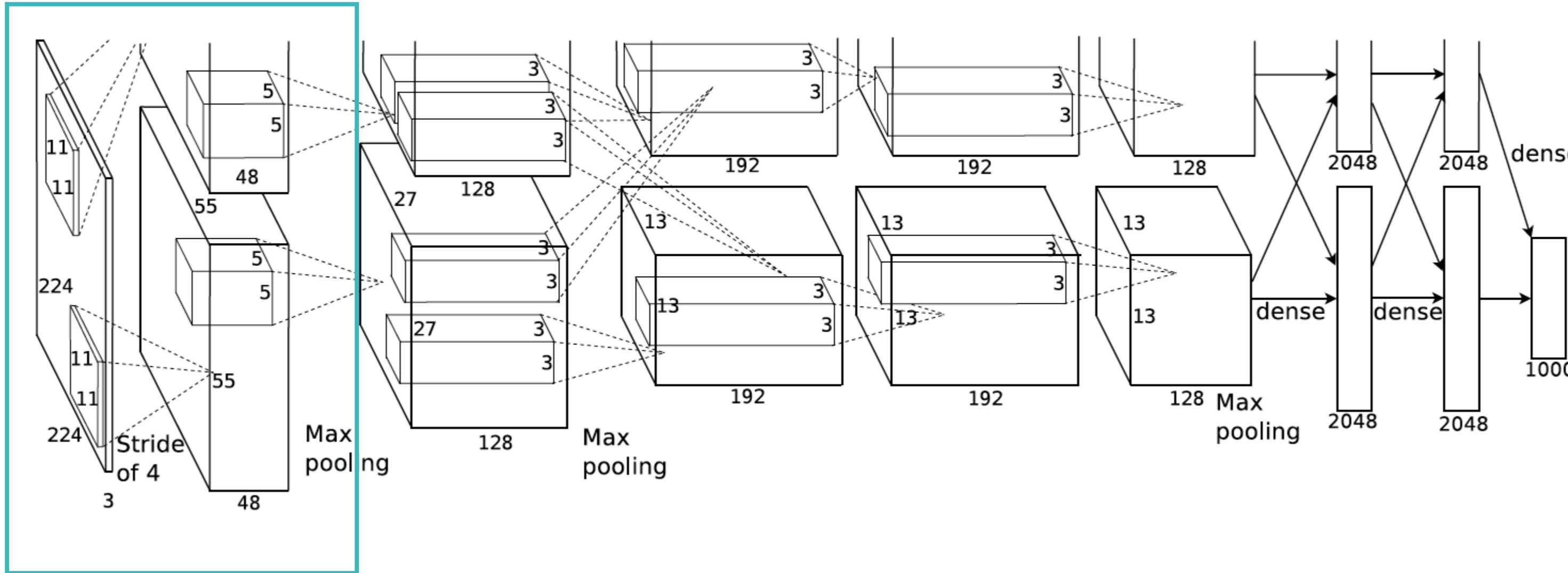
**3D Convolution:** AlexNet에서는 3차원 구조의 이미지/커널 사용

# 3차원 구조의 컨볼루션 레이어



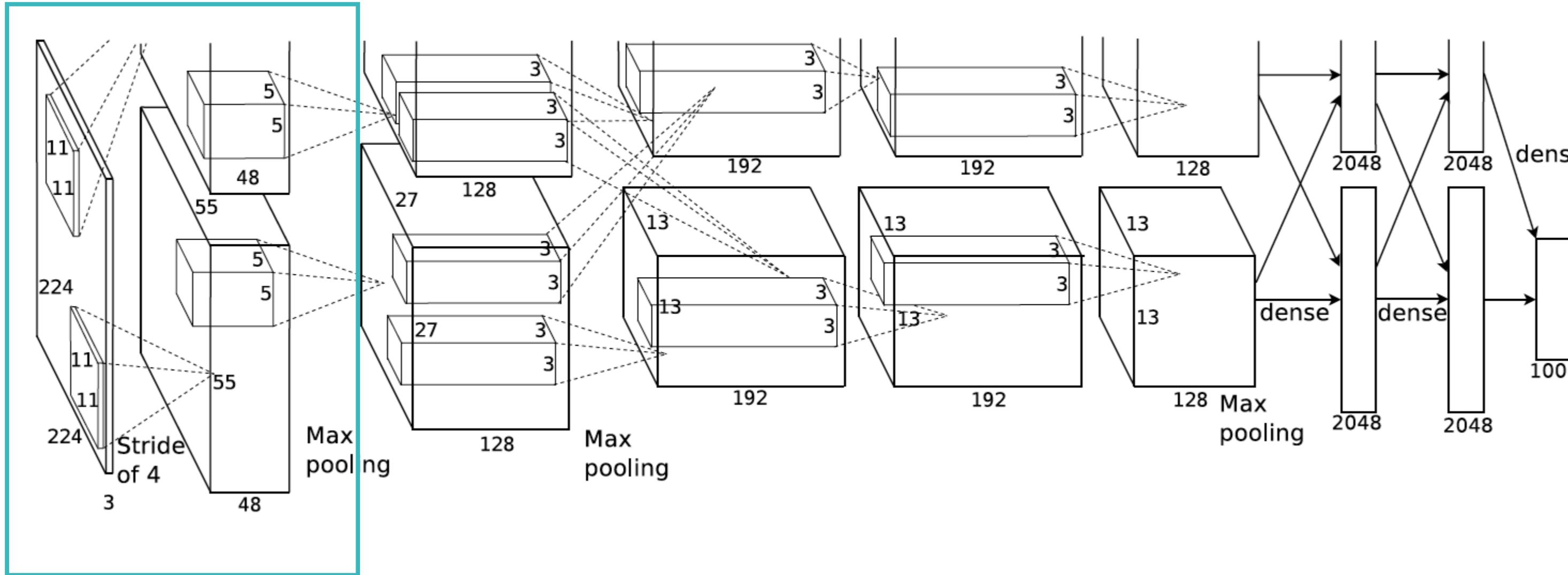
이미지는 **Red, Green, Blue** 3개의 성분을 갖기 때문에  
Depth 성분의 크기는 3

# 컨볼루션 레이어 #1



입력 영상의 크기: 제로 패딩을 통해 얻은 이미지 ( $227*227*3$ )  
컨볼루션 커널 ( $11*11*3*96$ )을 사용하여  $55*55*96$  이미지를 얻음  
(Stride 4)

# 컨볼루션 레이어 파라미터



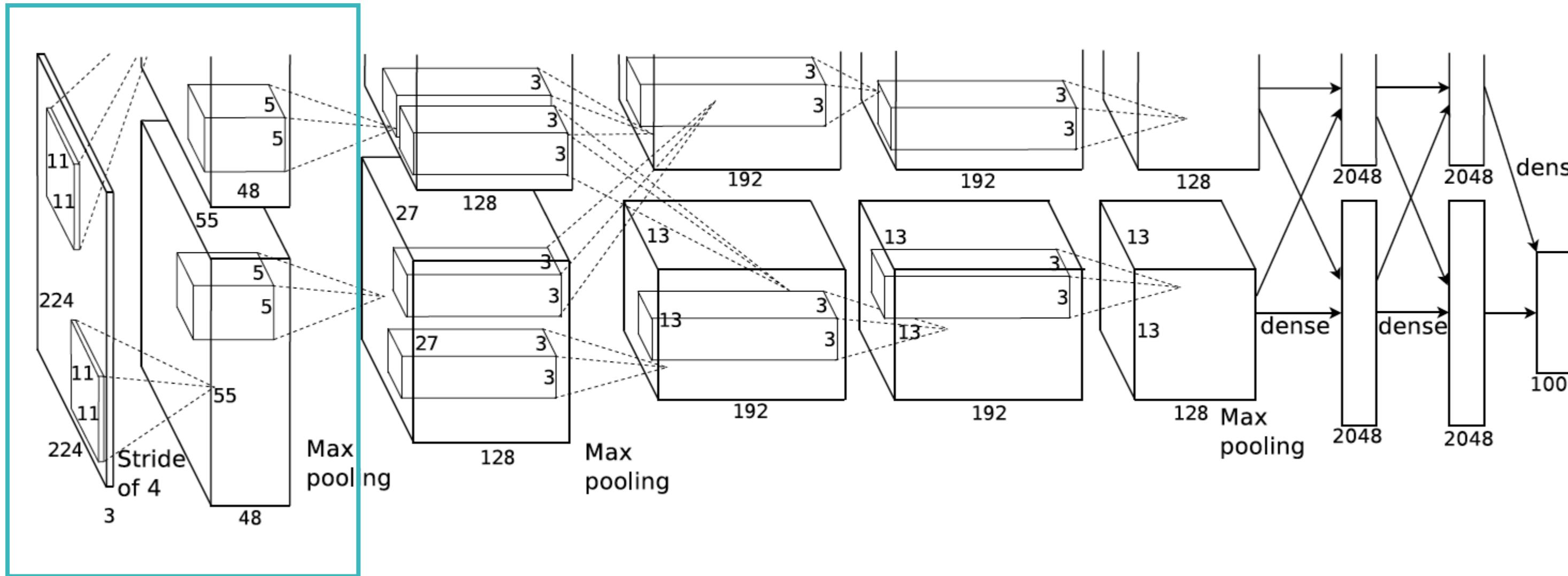
출력 이미지 ( $55 \times 55 \times 96$ ) = 뉴런 ( $55 \times 55 \times 96 = 290,400$ )

컨볼루션 커널은 363 ( $11 \times 11 \times 3$ )개의 weight 및 1개의 bias

96개의 컨볼루션 커널, 총 파라미터의 개수는  $96 \times 364 = 34,944$

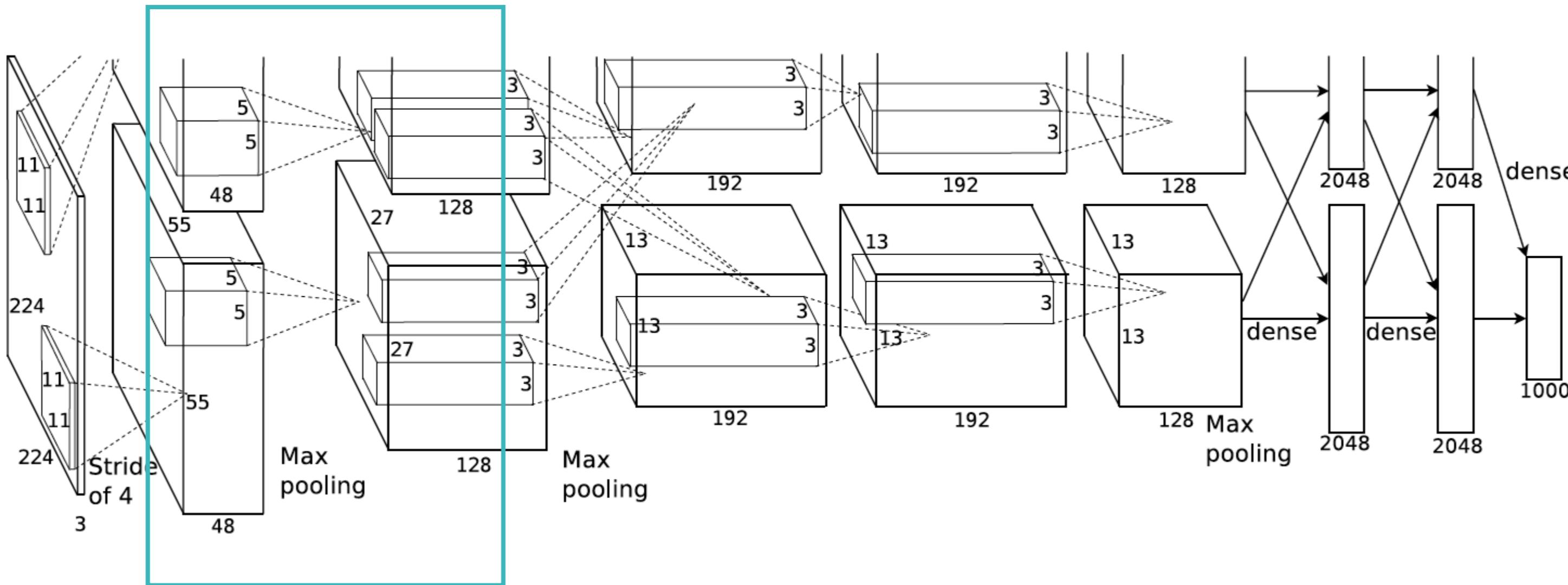
Connection의 개수:  $290,400 \times 364 = 105,750,600$

# 컨볼루션 레이어 파라미터



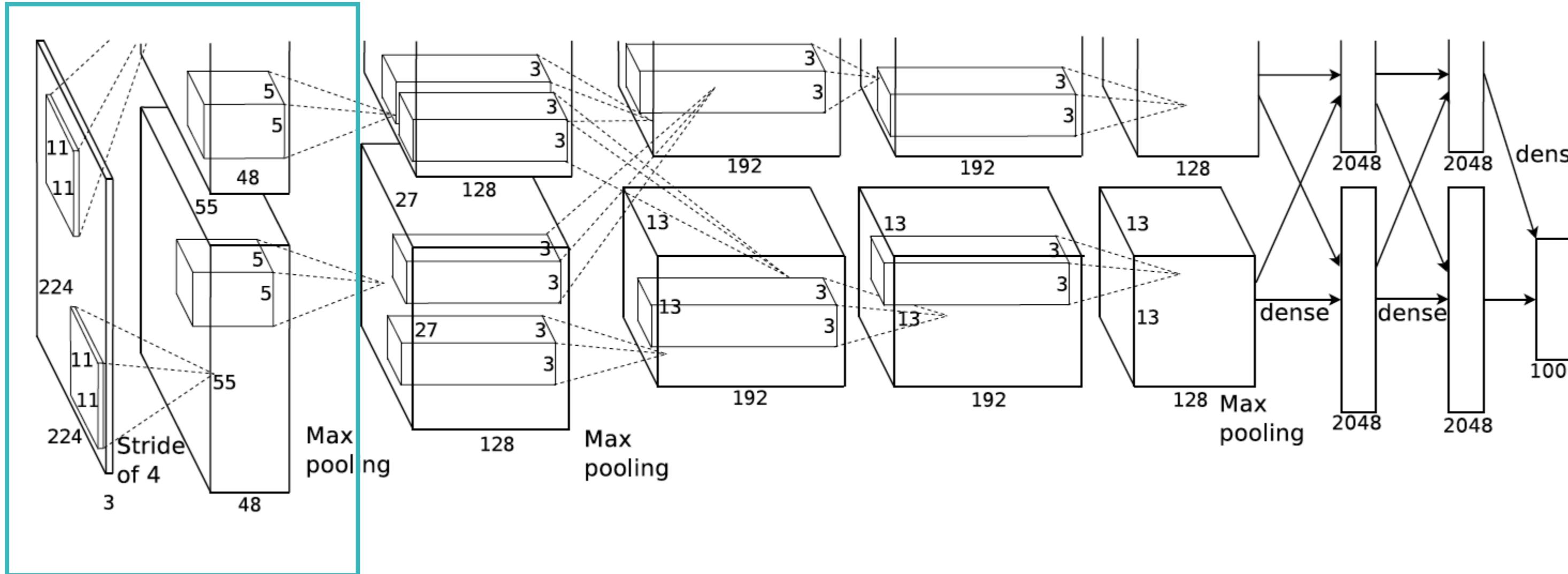
파라미터의 개수: 가중치와 바이어스의 갯수  
연산량: 컨볼루션 레이어를 계산하는데 덧셈과 곱셈의 횟수  
파라미터와 연산량을 줄이는 것이 모델 설계의 핵심 !

# 풀링 레이어 (서브 샘플링 레이어)



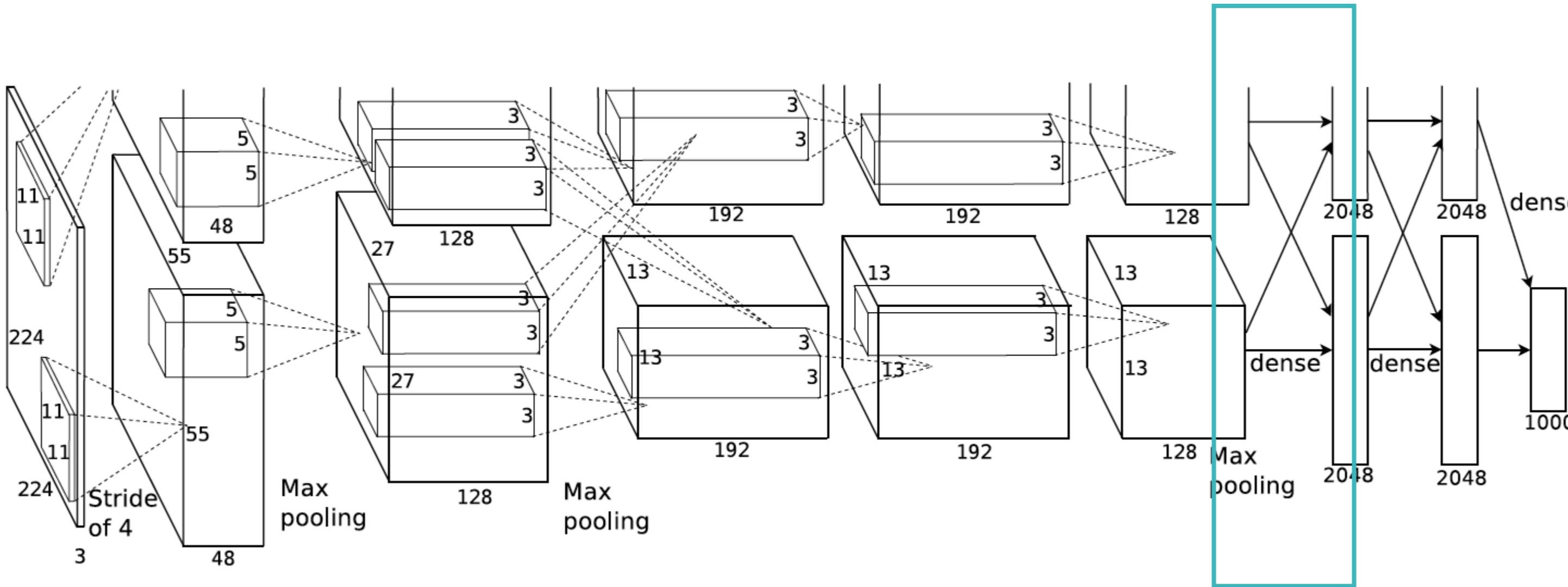
입력 이미지 (55\*55\*96)을 출력 이미지 (27\*27\*96)로 축소

# 컨볼루션 레이어 #2



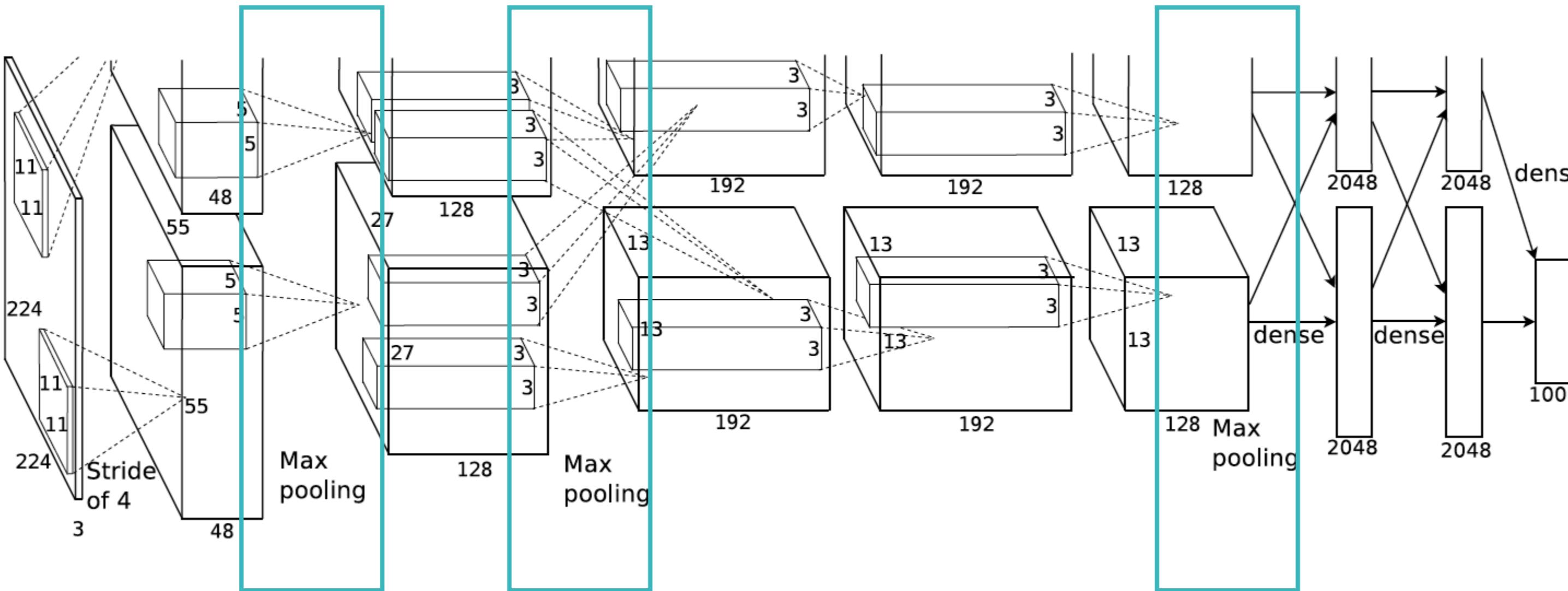
입력 영상의 크기: 풀링을 통해 얻은 이미지 ( $27*27*96$ )  
컨볼루션 커널 ( $5*5*96*256$ )을 사용하여  $27*27*256$  이미지를 얻음  
(Stride 1, Zero-padding 2)

# 풀리 커넥티드 레이어 #1



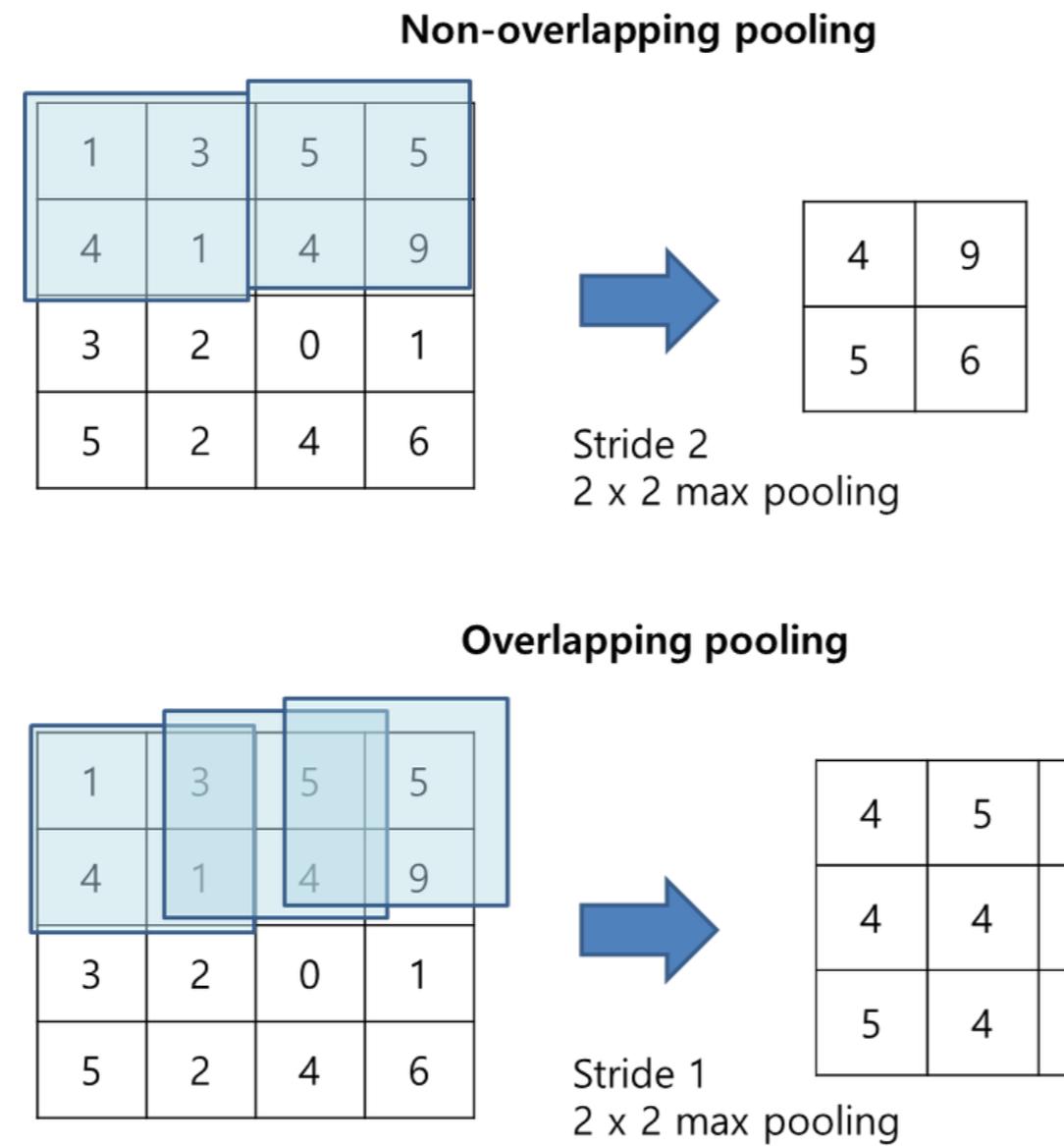
입력 영상의 크기: 컨볼루션을 통해 얻은 이미지 크기 ( $6*6*256$ )  
컨볼루션 커널 ( $6*6*256*4096$ )을 사용하여  $4,096*1$  이미지를 얻음  
4,096개의 성분 (뉴런)으로 구성된 벡터 생성

# AlexNet 성능 개선을 위한 고려 #1



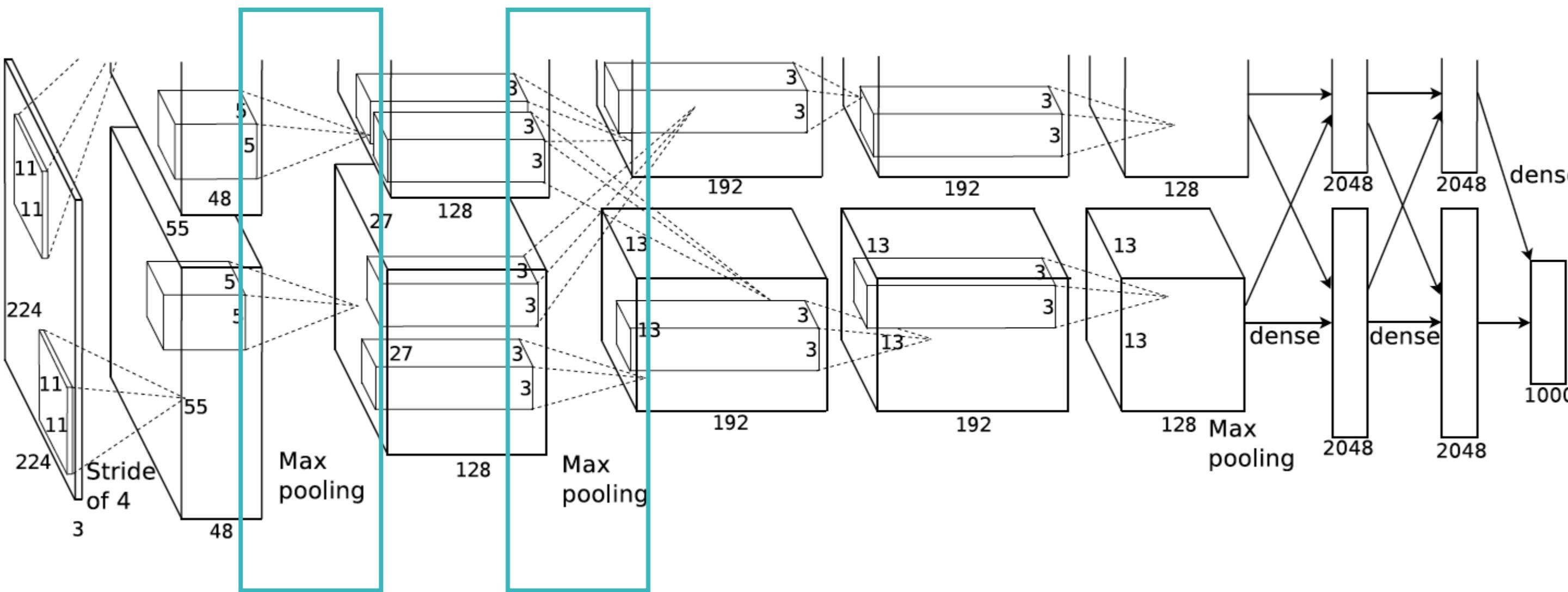
CNN에서 풀링은 입력 이미지의 크기를 줄이는 용도로 사용됨  
Average 또는 Max 풀링 방법을 사용  
AlexNet에서는 3\*3 window, Stride를 2로 하는 방법을 사용  
(Overlapped pooling)

# Overlapped pooling



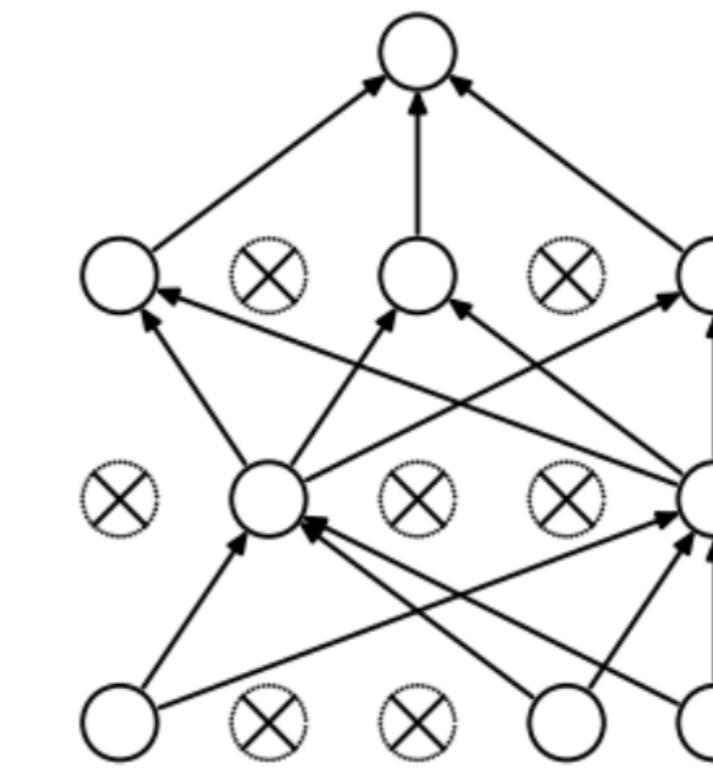
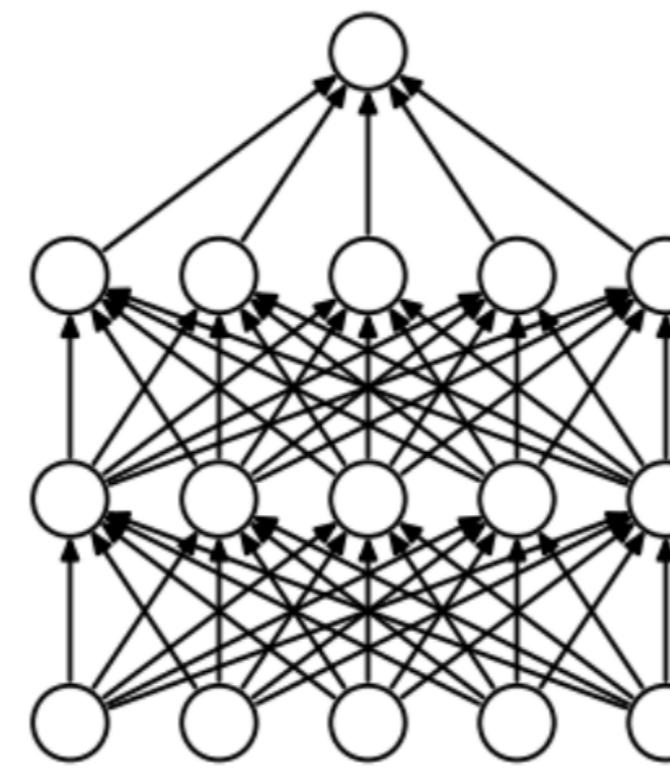
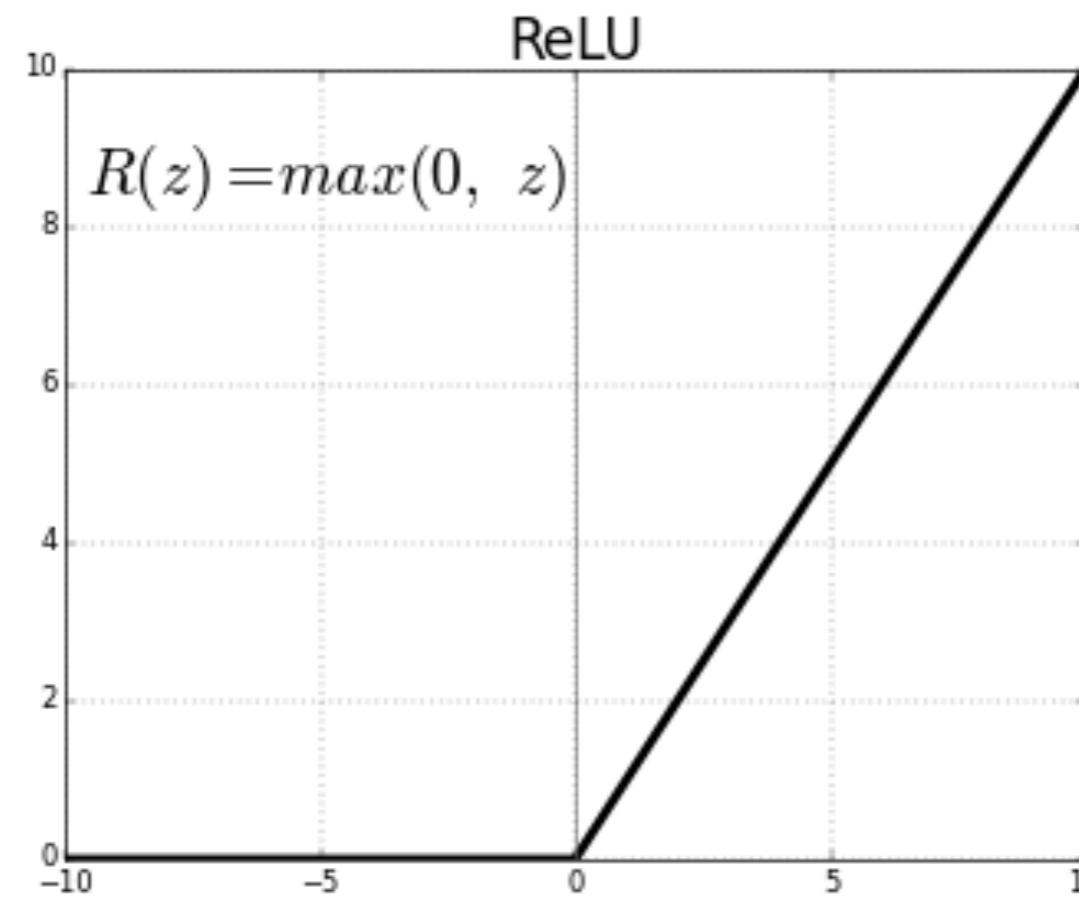
Overlapped pooling을 사용하면 풀링 윈도우가 중첩되면서 지나감  
Overlapping pooling을 사용은 정확도 개선에 유리함

# AlexNet 성능 개선을 위한 고려 #2



신경생물학에는 Lateral inhibition이라는 개념  
(활성화된 뉴런이 주변 이웃 뉴런을 억누르는 현상)  
Local response normalization은 이러한 현상을 모델링한 것  
Generalization 관점에서 더 좋은 정확도를 얻는 것이 가능함  
(Top-1 1.4%, Top-5 1.2% 정확도 개선)

# AlexNet 성능 개선을 위한 고려 #3

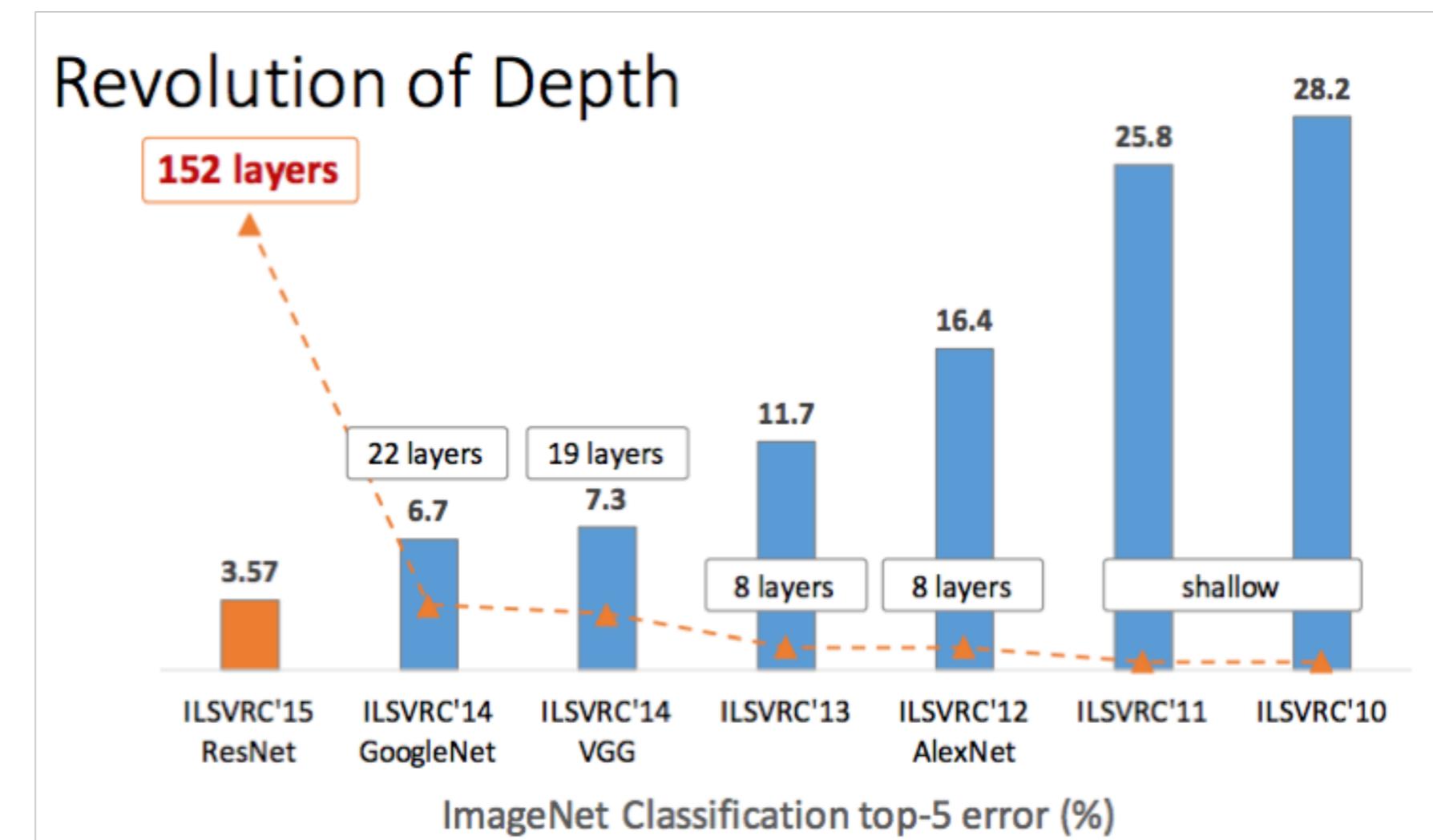


Data argument, ReLU, Dropout의 방법이 정확도 개선에 사용

# VGGNet

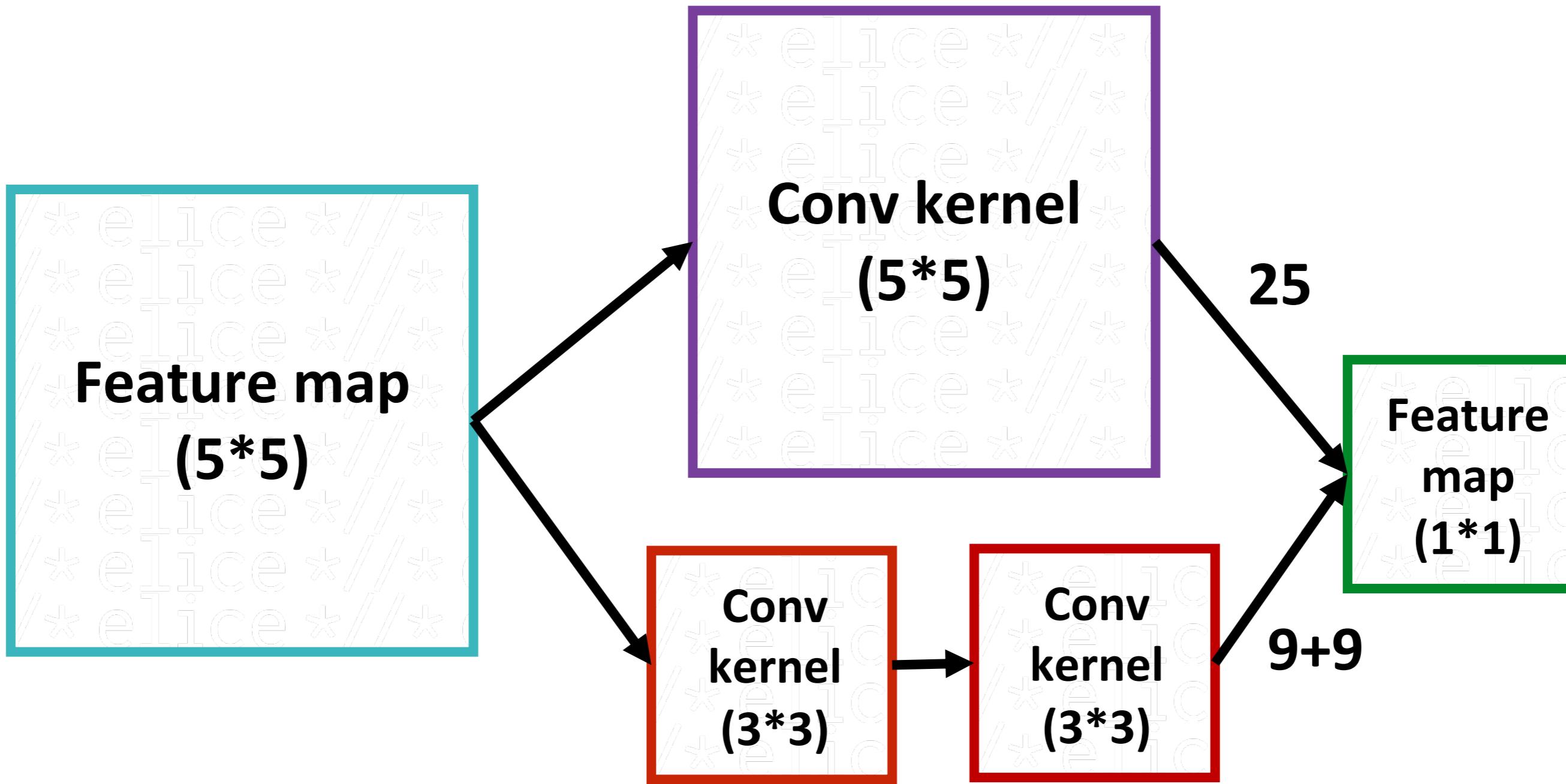
단순하지만 성능이 좋은 신경망

# Deeper and Deeper



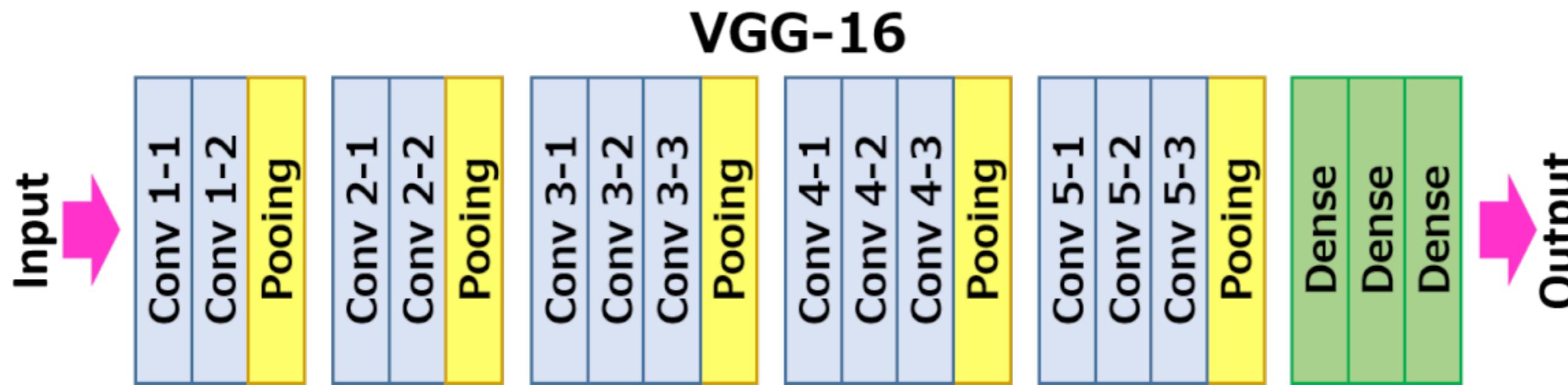
신경망의 깊이는 점점 깊어지는 추세  
망의 깊이가 깊어지는 만큼, 더 복잡한 문제의 해결 가능  
하지만, 파라미터와 연산량의 증가 그리고 Overfitting

# 파라미터의 숫자를 줄이는 방법



큰 필터를 갖는 Conv레이어는 대신에  
여러 개의 작은 Conv 필터를 갖는 레이어를 이용하면  
작은 파라미터와 연산량으로 가능

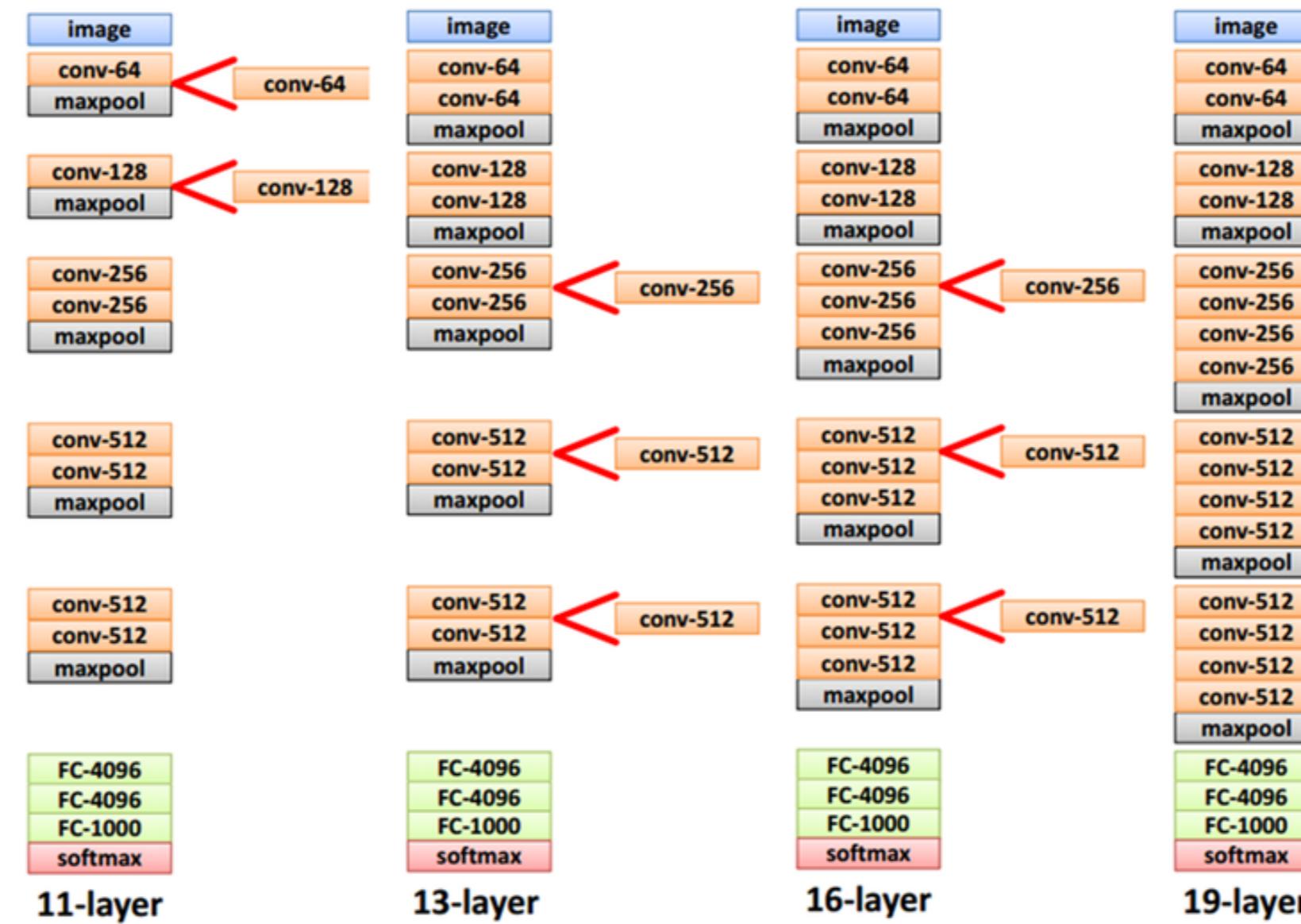
# 파라미터의 숫자를 줄이는 방법



| Model   | Top-5 error | # of parameter |
|---------|-------------|----------------|
| AlexNet | 15.3%       | 60M            |
| VGG-16  | 7.3%        | 138M           |

3\*3 컨볼루션 레이어를 2개 쌓으면 5\*5 컨볼루션 레이어  
3\*3 컨볼루션 레이어를 3개 쌓으면 7\*7 컨볼루션 레이어와 동일  
파라미터의 개수가 작기 때문에, 빠른 학습 속도

# 망의 깊이에 따른 VGGNet

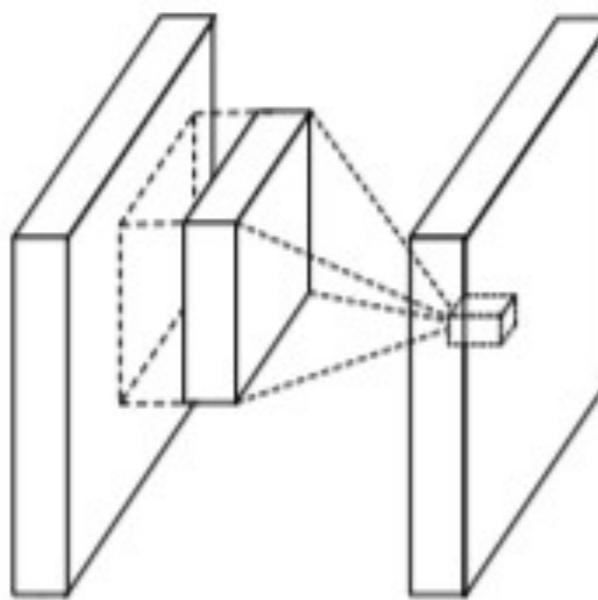


망이 깊어지만 Vanishing gradient 문제로 인해 학습이 어렵지만  
VGGNet에서는 11-layer의 학습 결과를 더 깊은 Layer의 파라미터  
초기화에 사용하여 문제 해결

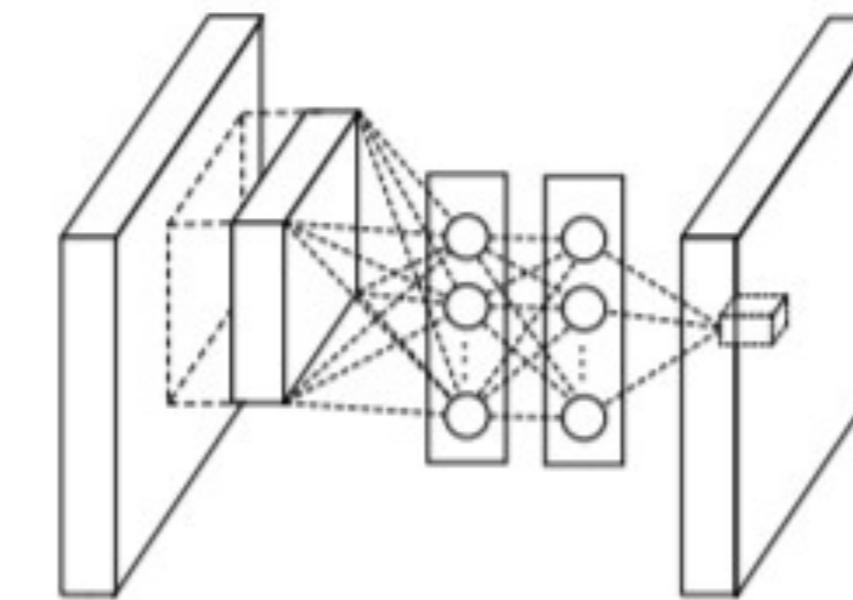
# GoogLeNet

구글이 만든 신경망은 얼마나 좋을까 ?

# Network in Network (NIN)



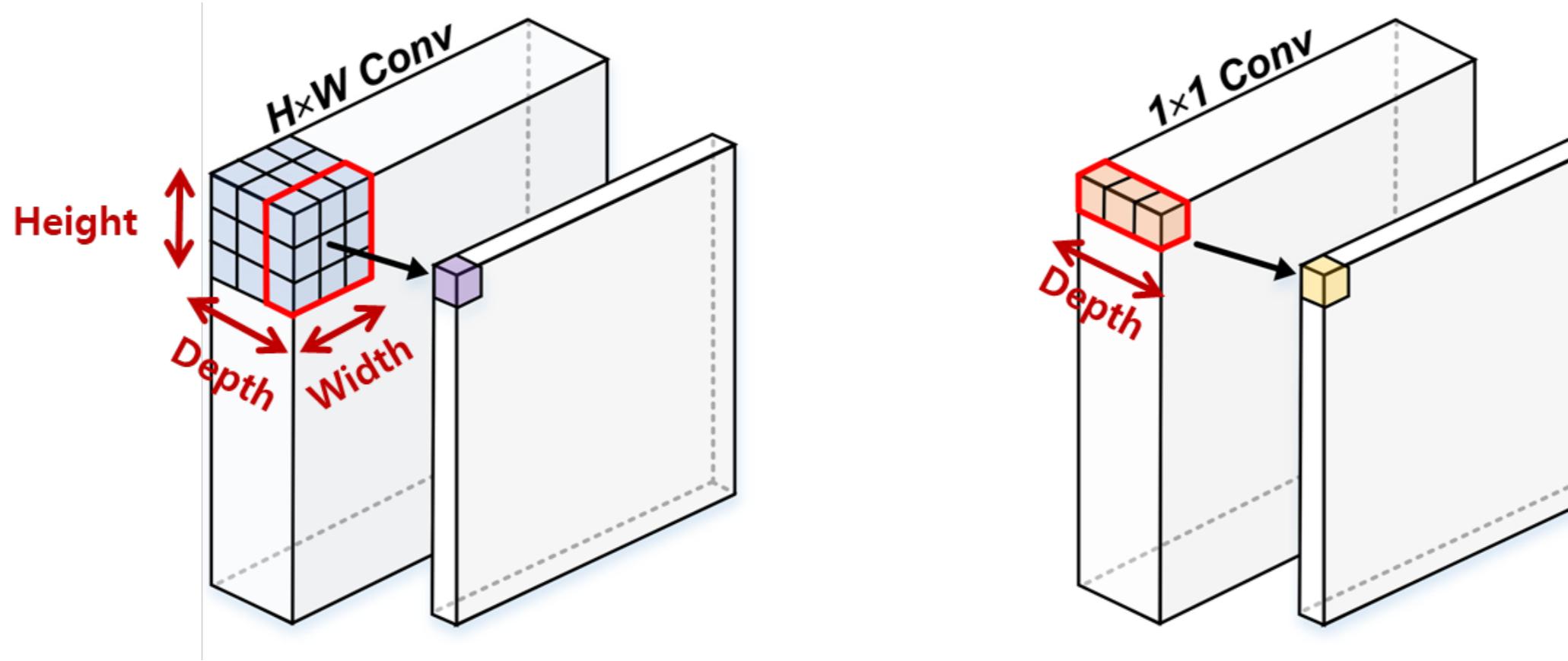
(a) Linear convolution layer



(b) MLPconv layer

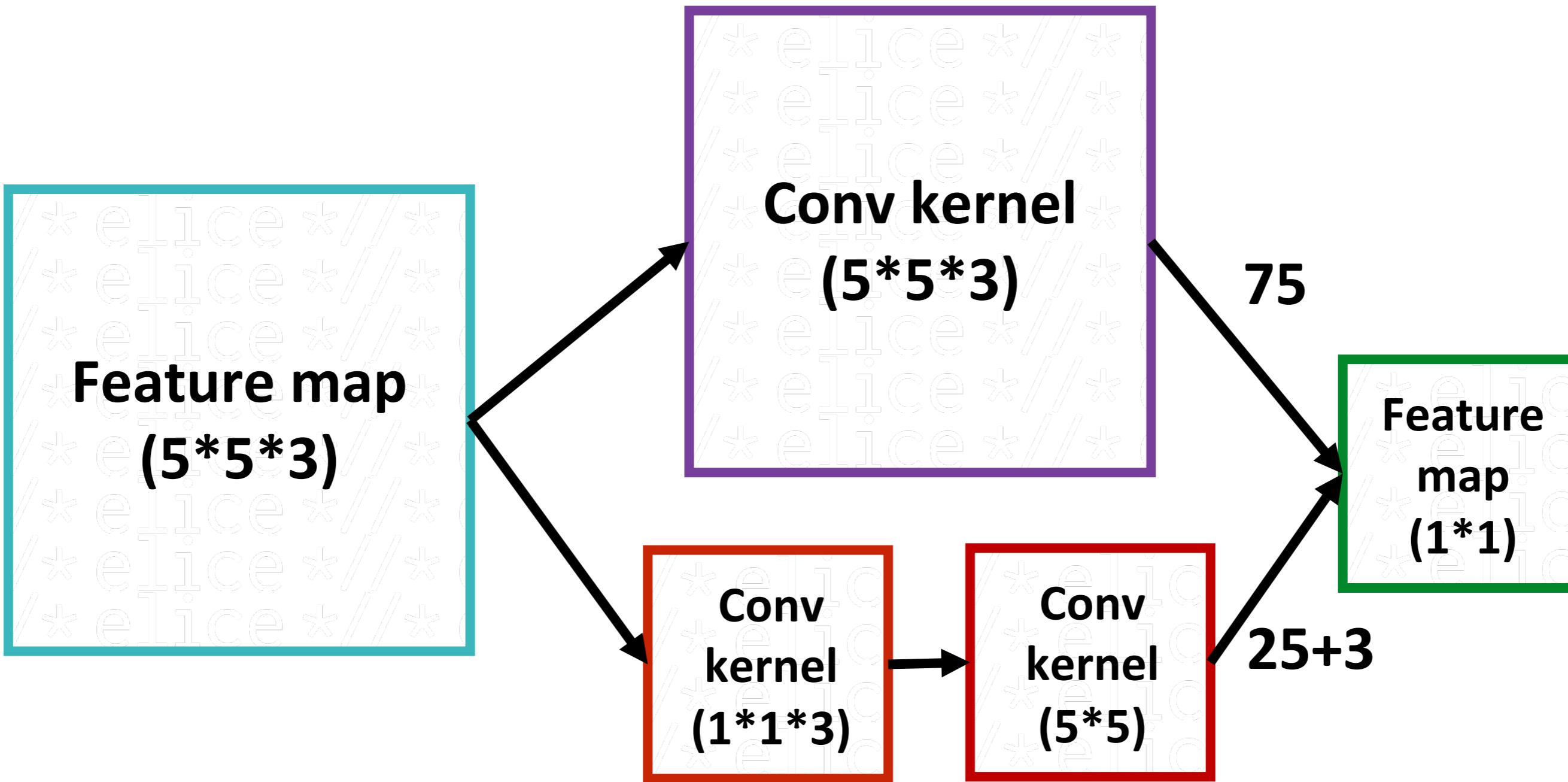
데이터의 분포가 비 선형적인 관계라면, MLP 필요  
MLP를 사용하면 Conv보다 비 선형적인 성질의 사용이 가능해  
특징을 잘 추출할 수 있는 능력이 우수함

# 1\*1 컨볼루션



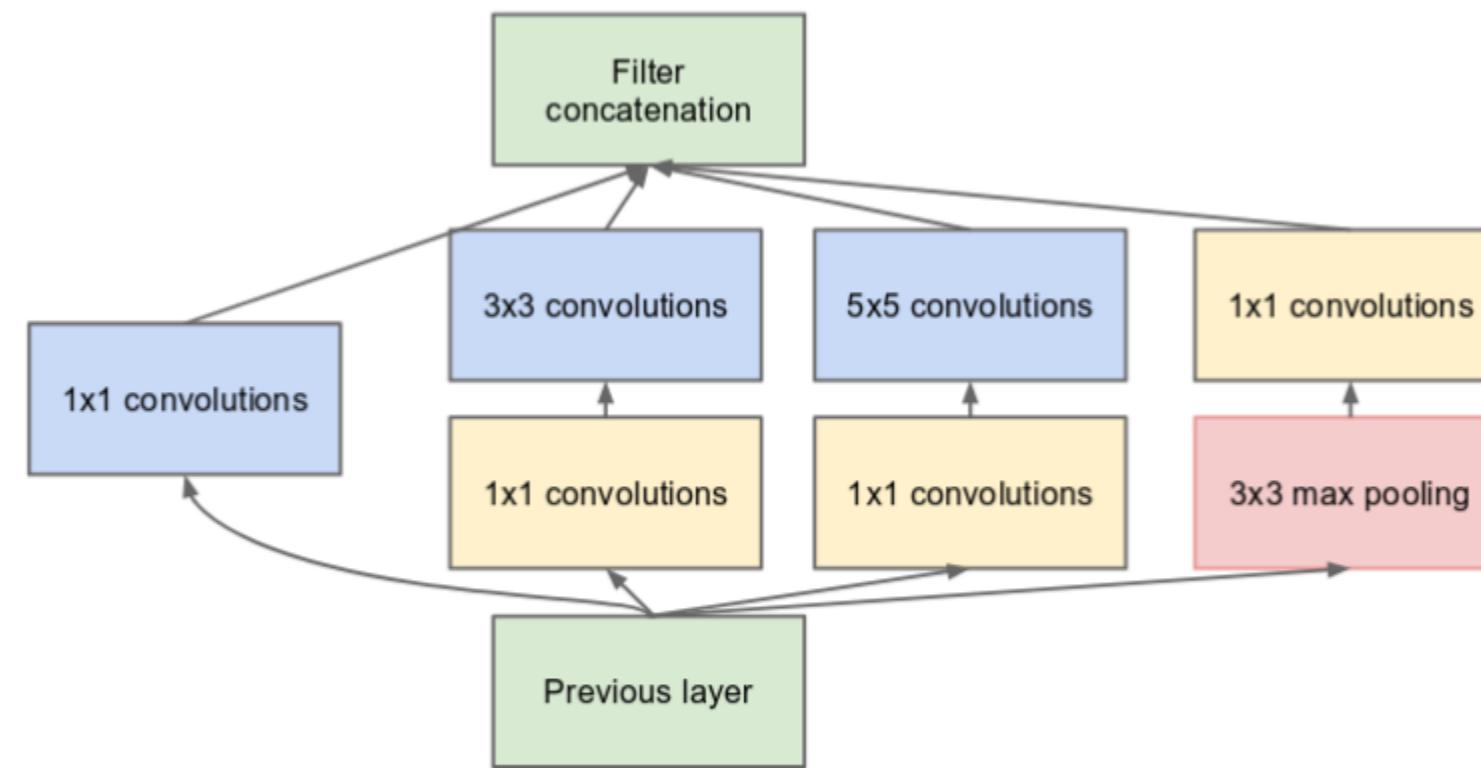
1\*1 컨볼루션은 차원을 줄이는 방법  
여러 개의 특징으로부터 비슷한 성질을 갖는 것들 묶는 기능  
피쳐맵과 연산량을 줄이는데 특화된 컨볼루션

# 1\*1 컨볼루션



보통 방법 ( $5*5*3=75$ ), 1\*1 컨볼루션 ( $3+25=28$ )  
비 선형적인 관계를 표현하는데 1\*1 컨볼루션 필터를 사용하여  
파라미터의 개수를 줄이는 것 가능

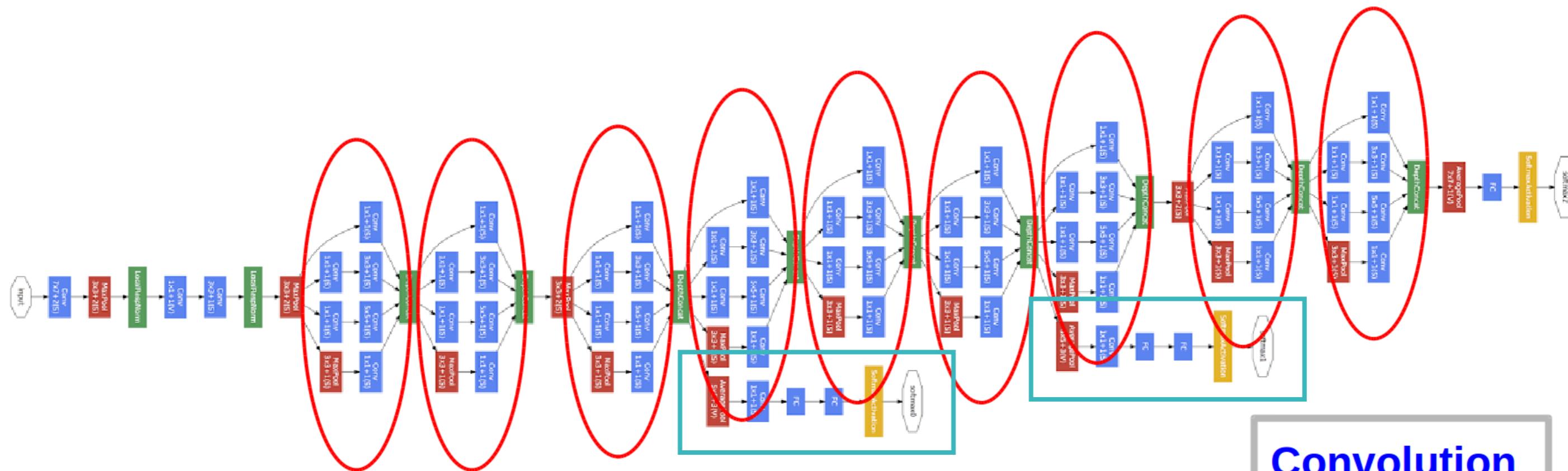
# Inception module



(b) Inception module with dimension reductions

Inception module에서는  $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$  컨볼루션 연산을 각각 수행  
 $1 \times 1$  컨볼루션 (채널 축소),  $N \times N$  컨볼루션 (채널 확대)

# GoogLeNet

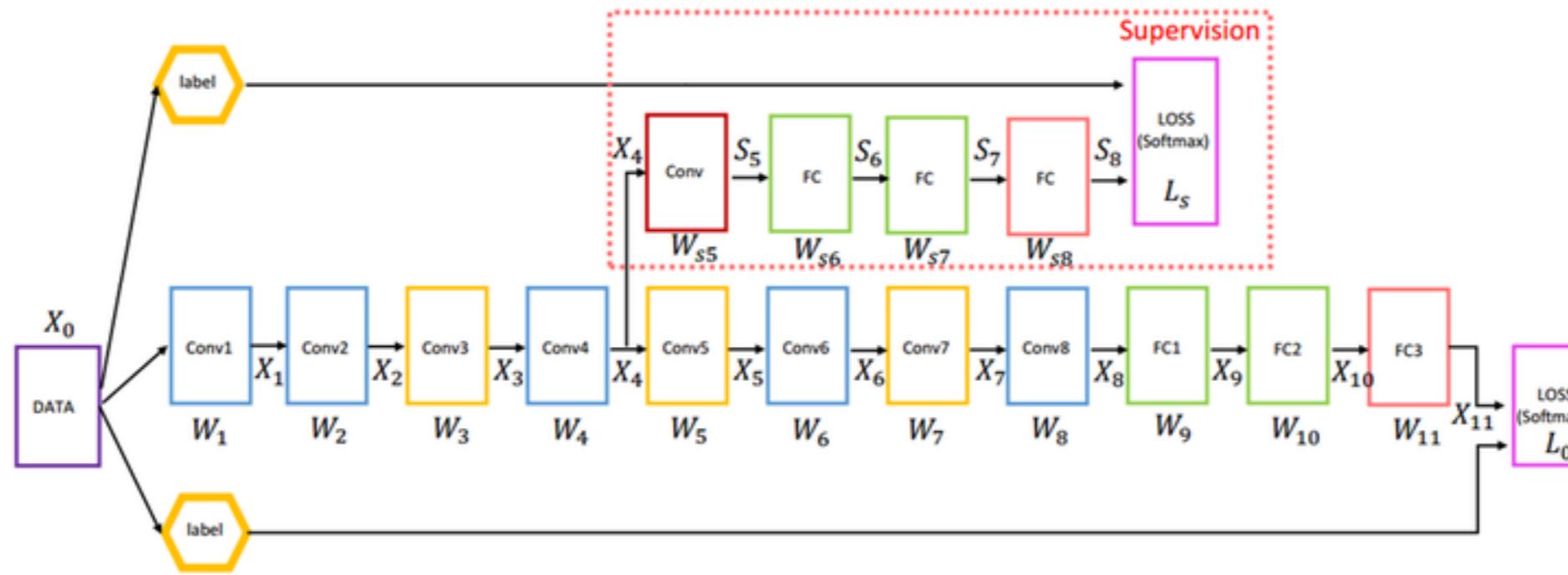


| Model     | Top-5 error | # of parameter |
|-----------|-------------|----------------|
| AlexNet   | 15.3%       | 60M            |
| VGG-16    | 7.3%        | 138M           |
| GoogLeNet | 6.67%       | 4M             |

Convolution  
Pooling  
Softmax  
Concat/Normalize

Inception module과 Auxiliary classifier 사용  
망이 깊어지면 학습이 잘 안되는 현상을 극복하기 위한 방법  
(Gradient가 작아지는 현상)

# Auxiliary classifier



Auxiliary classifier의 Back propagation 결과가 더해지기 때문에  
Gradient descent가 작아지는 문제를 해결하는 것이 가능함

# ResNet

정말 정말 깊이가 깊은 신경망

# 정확도와 망의 깊이

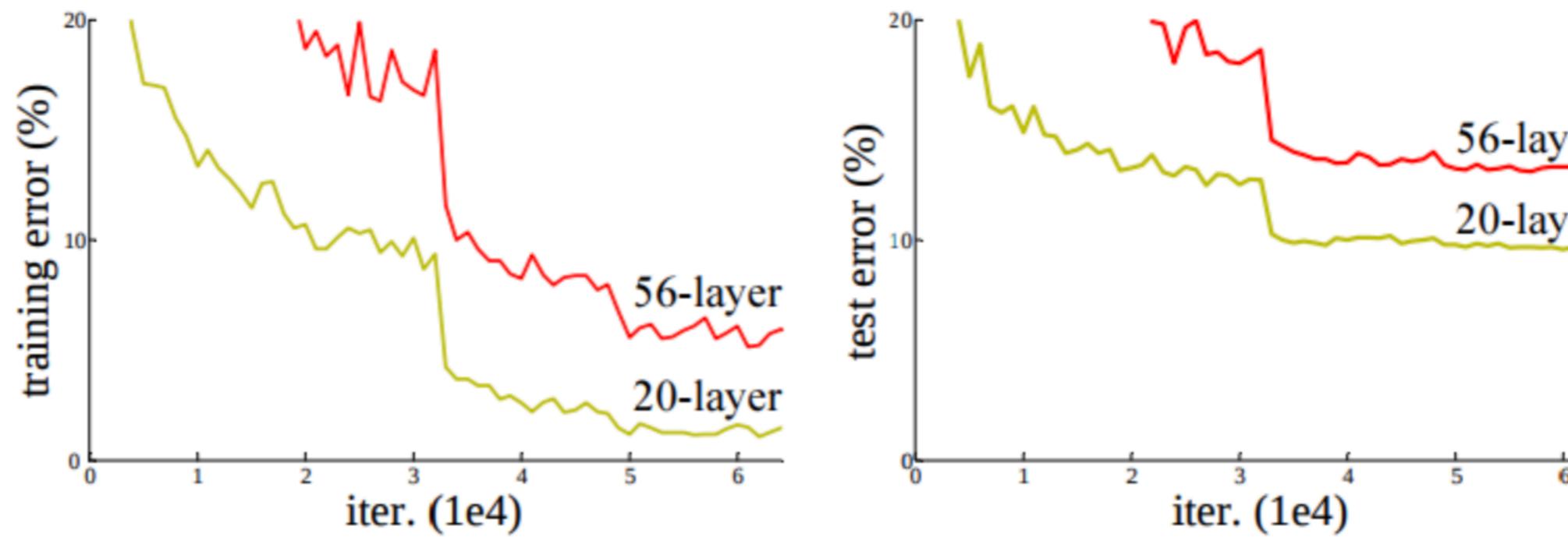
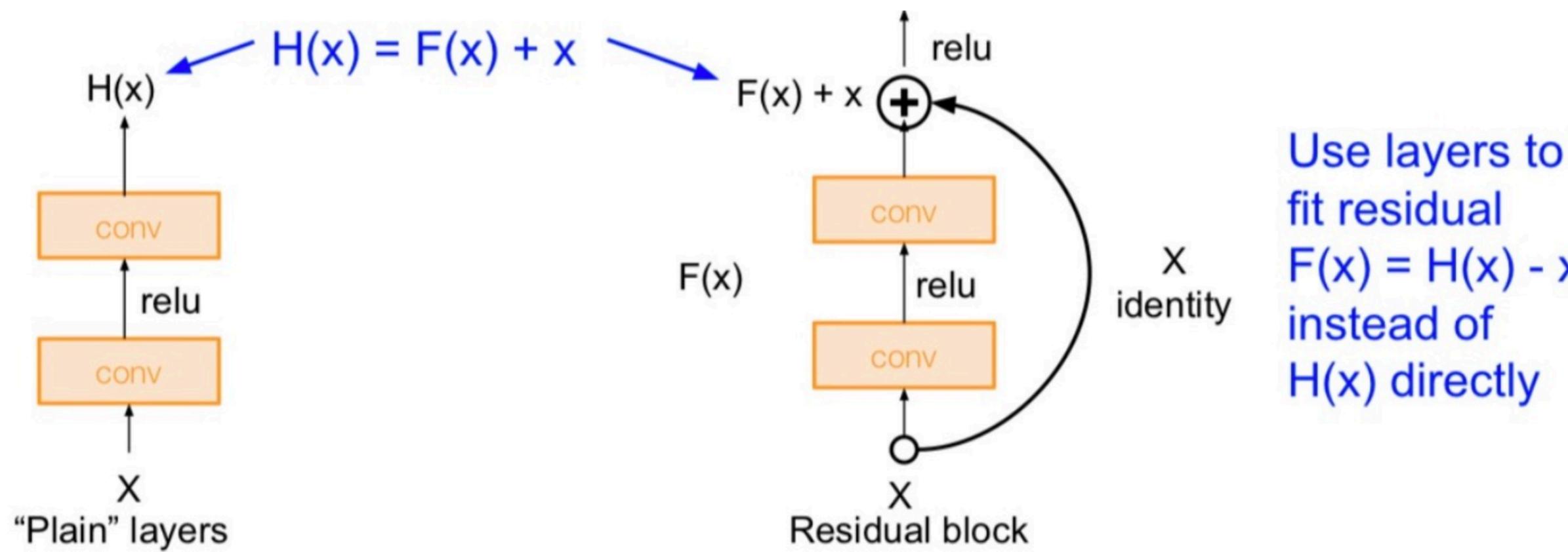


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

망의 깊이를 깊게 만드는 것은 정확도 향상에 큰 도움이 되지만  
깊이가 깊어지면, **Vanishing Gradient** 문제 때문에 정확도 감소

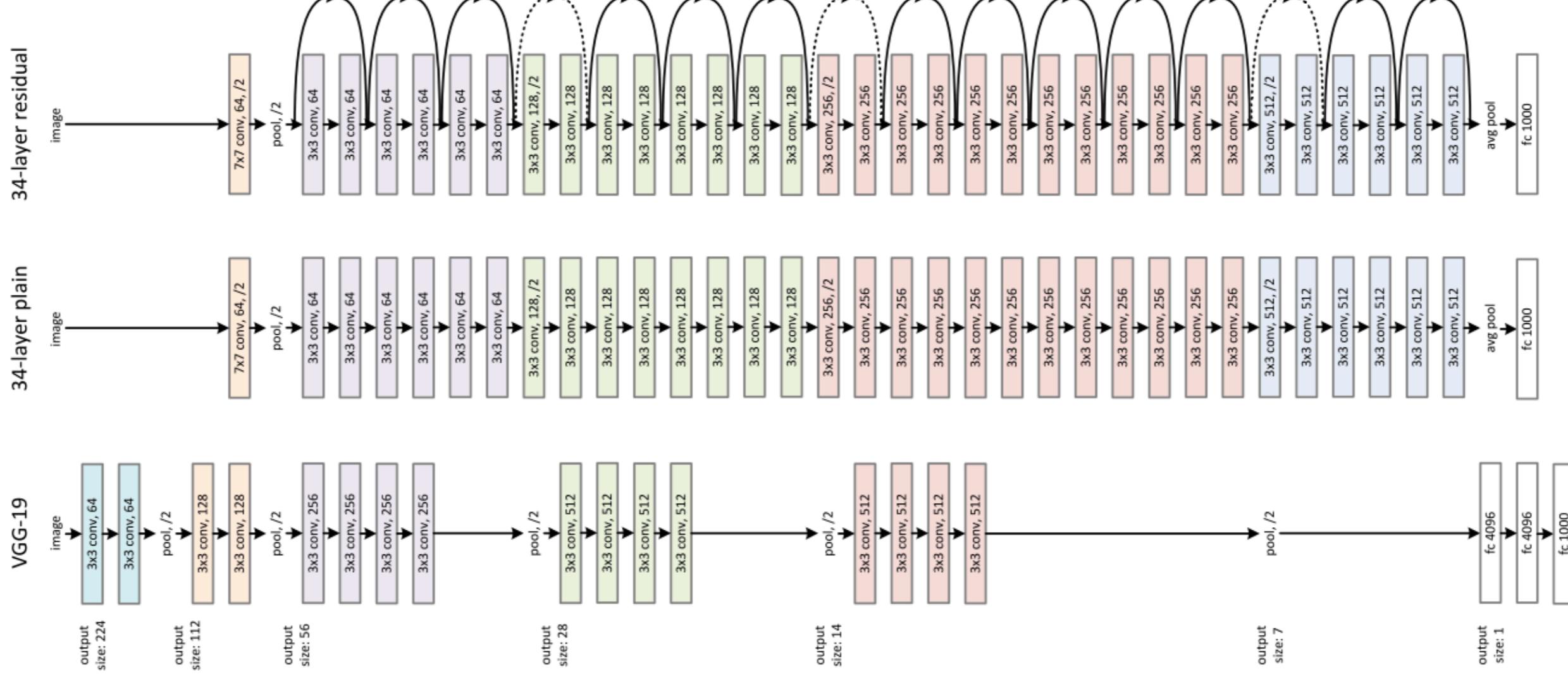
# ResNet 컨셉: Skip Connection



기존 연구: 입력 ( $x_i$ )을 목표값 ( $F(x_i)$ )으로 매핑하는  $H(x)$ 를 찾는 것  
( $H(x_i) - F(x_i)$ 를 최소화 하는 방향으로 학습)

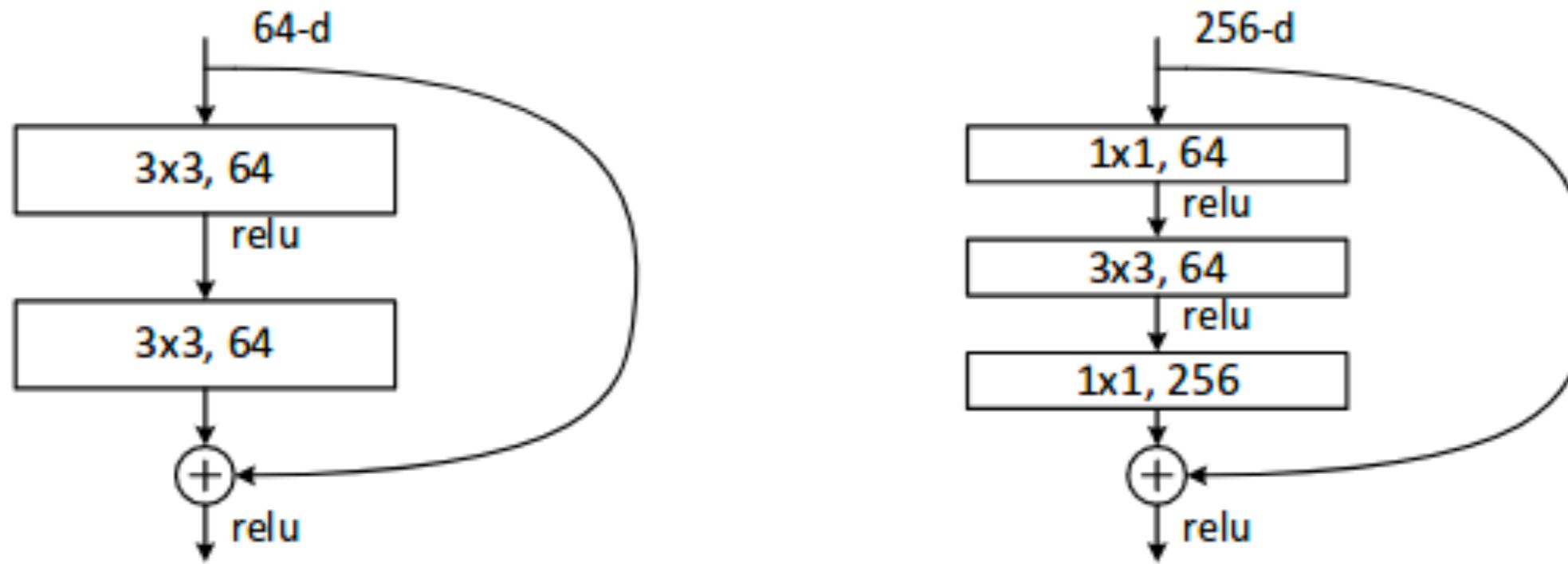
ResNet: 입력과 출력의 차이값 ( $F(x_i) = H(x_i) - x_i$ )를 찾는 것  
( $F(x_i) = 0$  최적의 경우, 학습의 목표가 정해진 상태에서 학습)

# ResNet



**Skip Connection의 사용을 통해 망의 깊이가 깊어도  
안정적으로 학습 가능**

# ResNet: Bottleneck Block



**두 종류의 Residual Block 사용  
더 Deep한 신경망을 만들기 위해 3 Layer 모델 사용**

# ResNet Architecture

| layer name | output size | 18-layer  | 34-layer  | 50-layer  | 101-layer  | 152-layer  |
|------------|-------------|---|---|---|--|--|
| conv1      | 112×112     |   |   | 7×7, 64, stride 2   |  |  |
|            |             |   |   | 3×3 max pool, stride 2  |  |  |
| conv2_x    | 56×56       | $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$   | $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$   | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$    | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$     | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$     |
| conv3_x    | 28×28       | $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$ | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$  | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$   | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$   |
| conv4_x    | 14×14       | $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$ | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$ |
| conv5_x    | 7×7         | $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$  | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$  |
|            | 1×1         |   |   | average pool, 1000-d fc, softmax  |  |  |
| FLOPs      |             | $1.8 \times 10^9$   | $3.6 \times 10^9$   | $3.8 \times 10^9$   | $7.6 \times 10^9$  | $11.3 \times 10^9$   |

| model          | top-1 err.   | top-5 err.  |
|----------------|--------------|-------------|
| VGG-16 [41]    | <b>28.07</b> | 9.33        |
| GoogLeNet [44] | -            | 9.15        |
| PReLU-net [13] | 24.27        | 7.38        |
| plain-34       | 28.54        | 10.02       |
| ResNet-34 A    | 25.03        | 7.76        |
| ResNet-34 B    | 24.52        | 7.46        |
| ResNet-34 C    | 24.19        | 7.40        |
| ResNet-50      | 22.85        | 6.71        |
| ResNet-101     | 21.75        | 6.05        |
| ResNet-152     | <b>21.43</b> | <b>5.71</b> |

5종류의 깊이를 갖는 망으로 실험  
깊어진 망의 깊이만큼 더 좋은 정확도

/\* elice \*/

문의 및 연락처

[academy.elice.io](http://academy.elice.io)

[contact@elice.io](mailto:contact@elice.io)

[facebook.com/elice.io](https://facebook.com/elice.io)

[medium.com/elice](https://medium.com/elice)