CNN
(Convolutional Neural Networks)

까지 요약(모두의 딥러닝)

2021310821 정현일

Contents

- CNN?
- Stride / Padding / 여러개의 filters / Convolutional Layers
- Max Pooling
- CNN case

CNN?

Stride / Padding / 여러개의 filters / Convolutional Layers

Max Pooling

CNN case

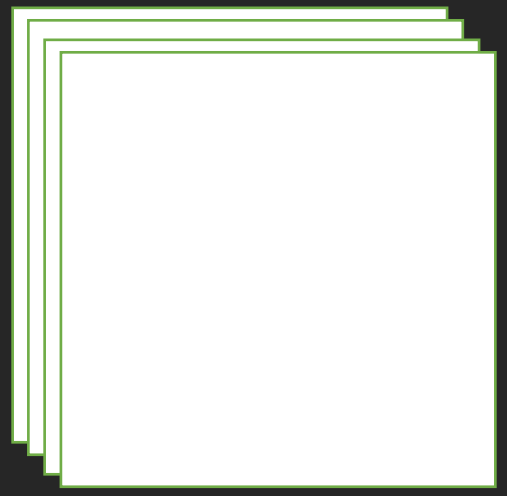
CNN?

이미지를 학습할 때 사용하는
Neural Network

CNN?

Input (Training Data)

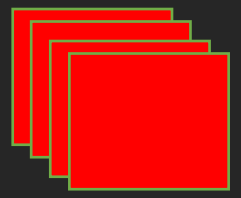
Image (width * height * depth)



CNN?

Filter

(width * height * depth)

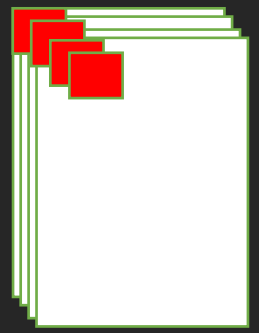


→ 하나의 수

= WX + b

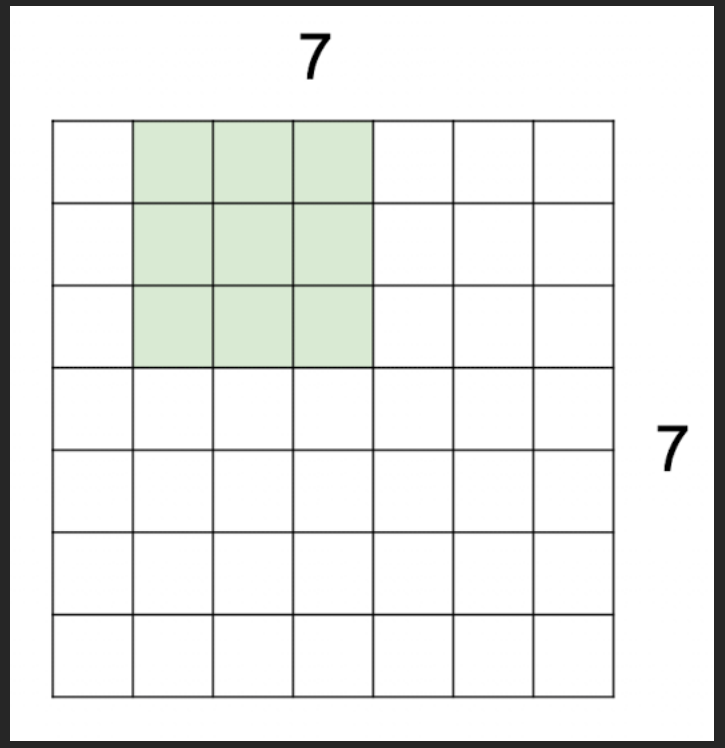
Input (Training Data)

Image (width * height * depth)



Stride / Padding /
여러개의 filters & Convolutional Layers

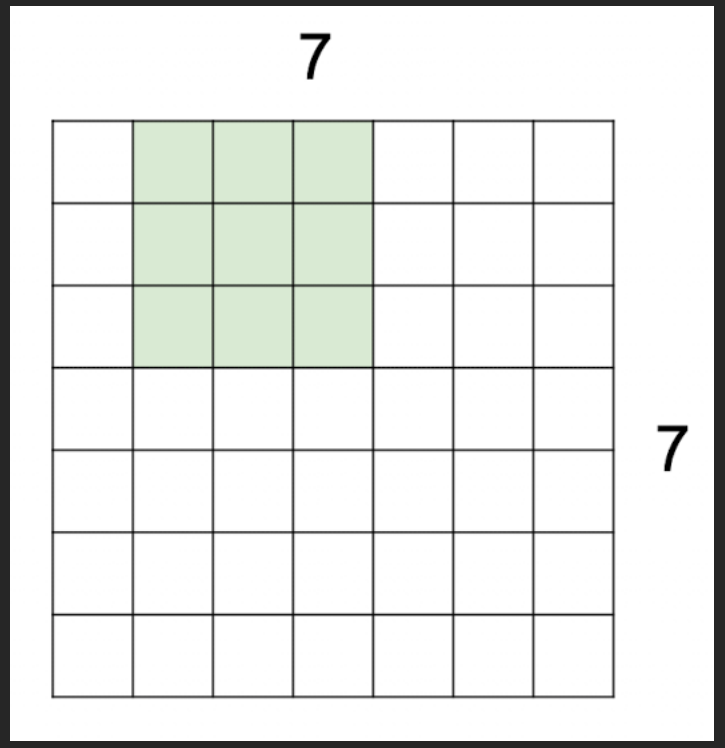
예시



Input : 7 * 7 * depth
filter : 3 * 3 * depth
strade : 1

Stride / Padding / 여러개의 filters & Convolutional Layers

예시

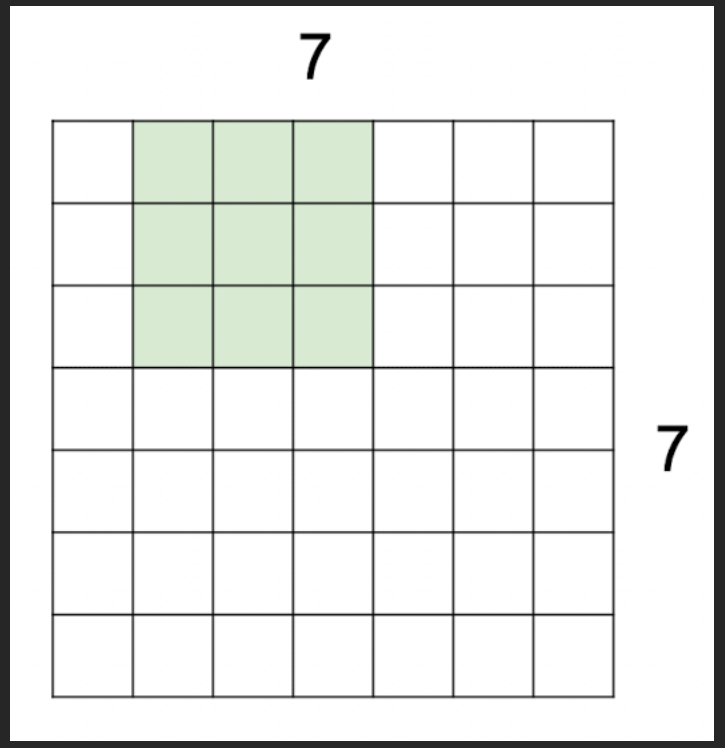


Input : 7 * 7 * depth
filter : 3 * 3 * depth
stride : 1

Output : 5 * 5 * depth

Stride / Padding / 여러개의 filters & Convolutional Layers

예시



Input : N * N * depth
filter : F * F * depth
stride : S

Output : O * O * depth
O = (N - F) / S + 1

Stride / Padding /
여러개의 filters & Convolutional Layers

문제 : 필터를 지나칠 수록 사이즈가 작아짐!

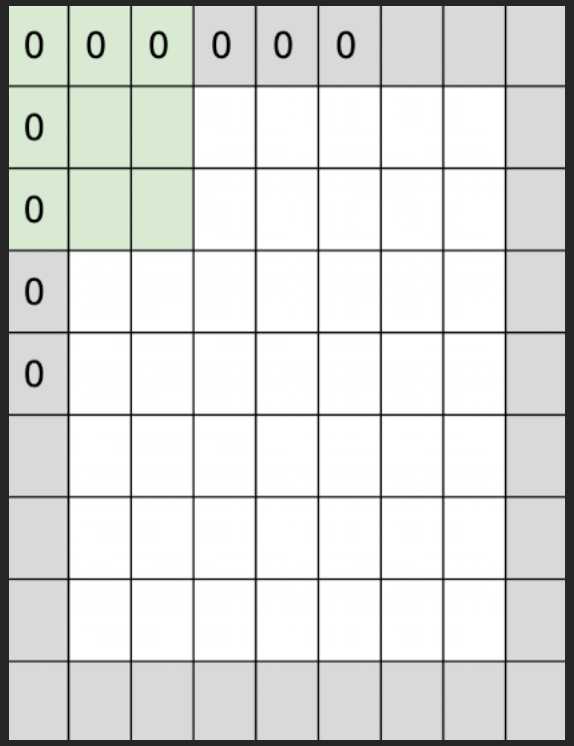
-> 정보의 손실

Stride / Padding / 여러개의 filters & Convolutional Layers

해결방법 : Padding

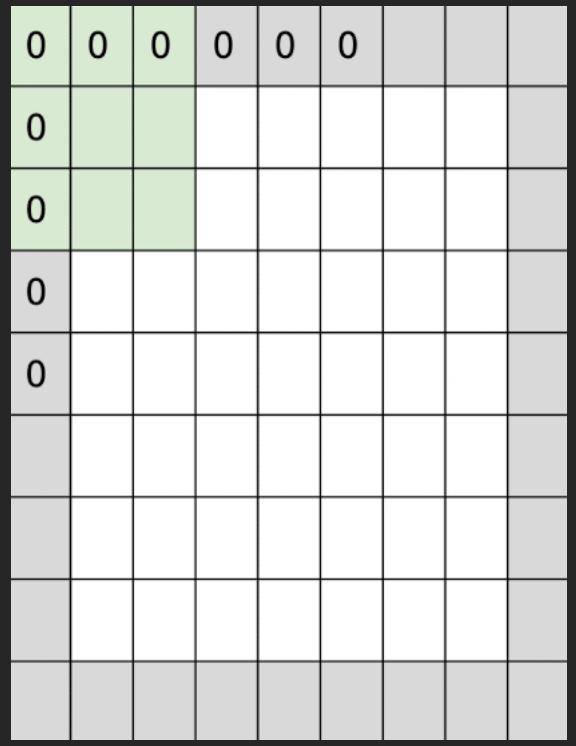
Stride / Padding /
여러개의 filters & Convolutional Layers

해결방법 : Padding



Stride / Padding / 여러개의 filters & Convolutional Layers

Padding



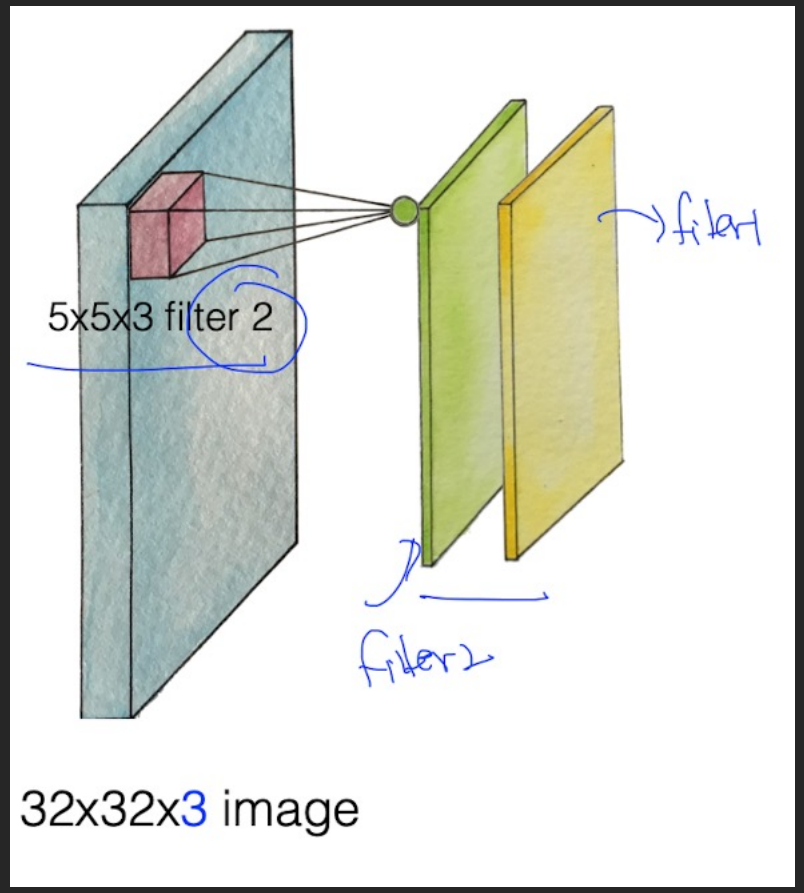
장점

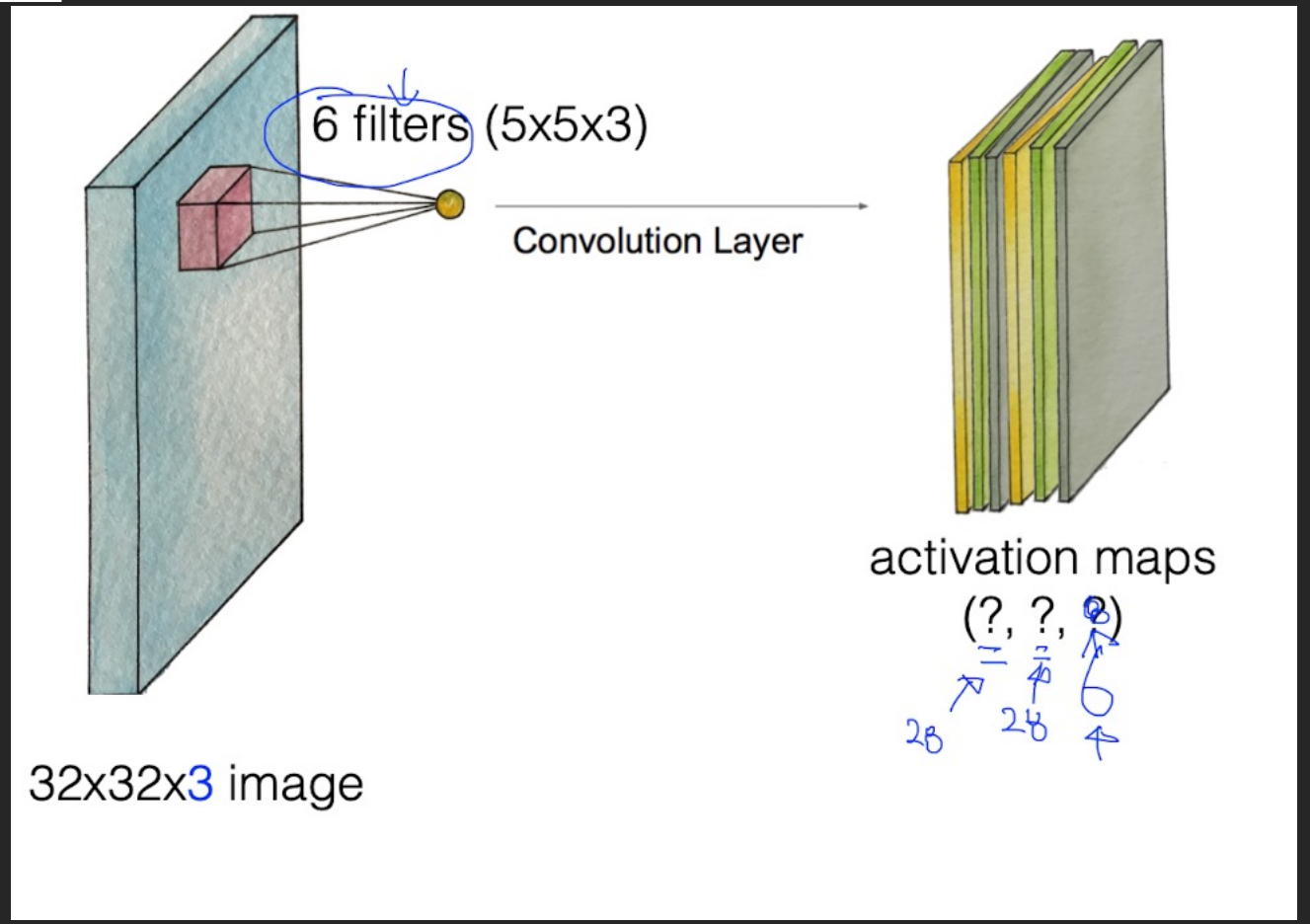
1. 이미지가 작아지는 것을 방지

2. model한테 이미지의 모서리 부분을 학습시키는 기능

Stride / Padding / 여러개의 filters & Convolutional Layers

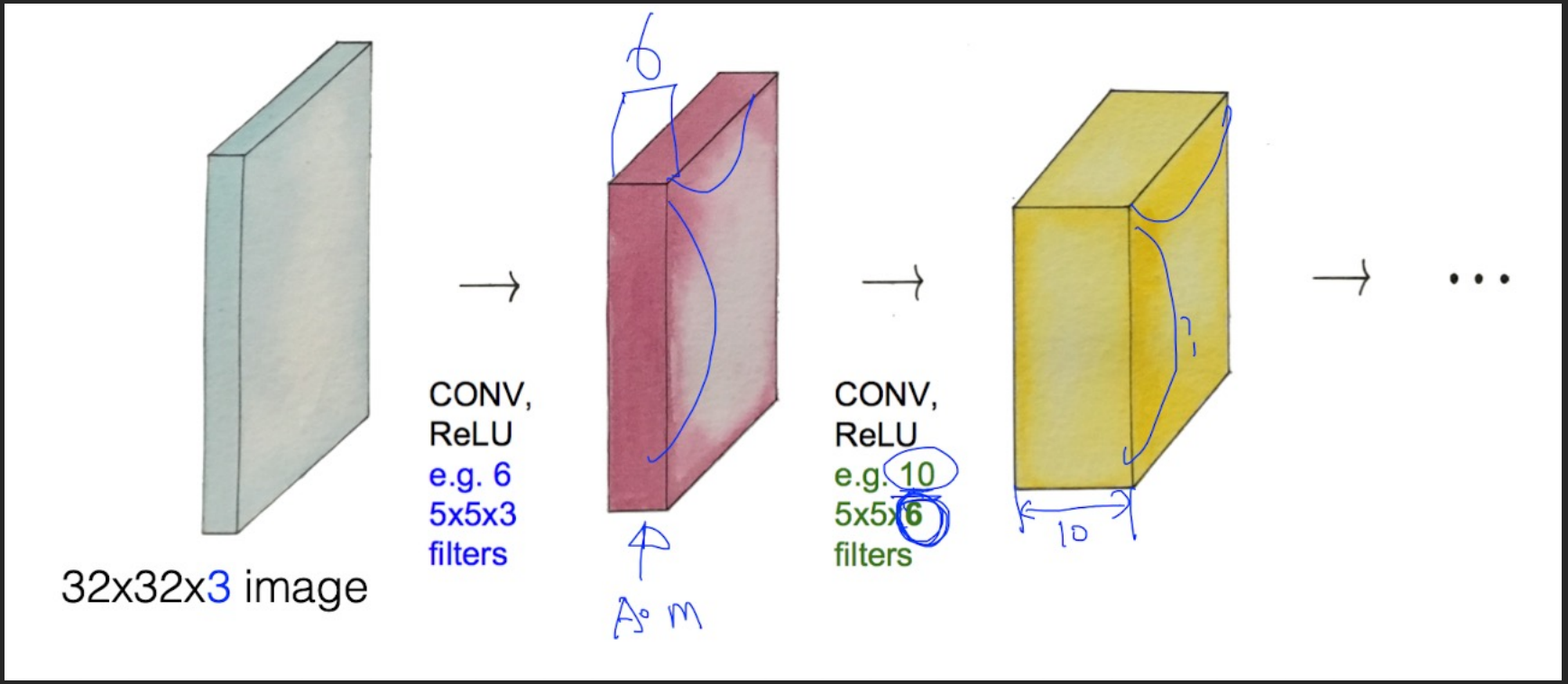
여러개의 filters





Stride / Padding / 여러개의 filters & Convolutional Layers

Convolutional Layers



Max Pooling

Pooling Layer

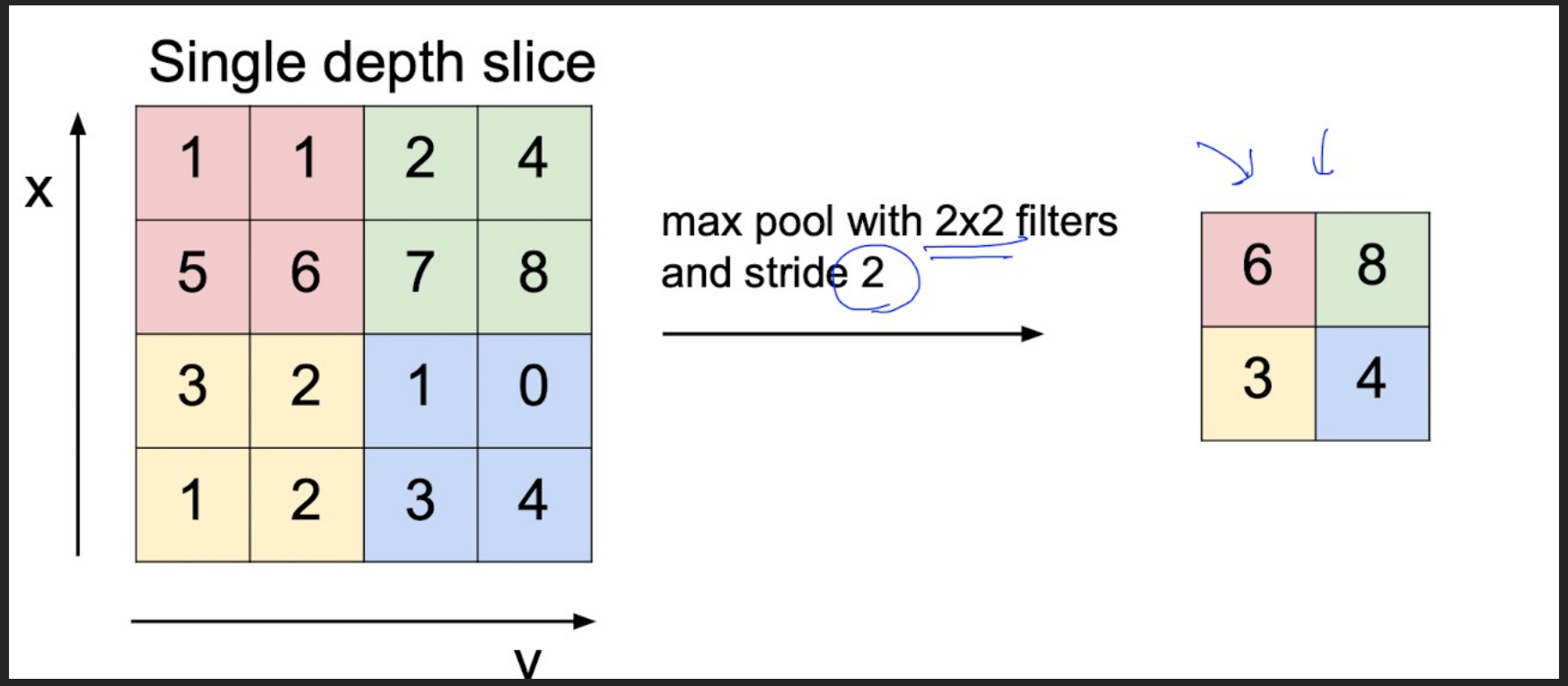
Weight를 곱해서 layer를 만드는 것이 아니라 다른 방식으로 만들어진 Layer

Max Pooling

max pooling은 필터 사이즈 안에서 가장 큰 값을 하나의 수로 취해서 오는 것

Max Pooling

max pooling은 필터 사이즈 안에서 가장 큰 값을 하나의 수로 취해서 오는 것

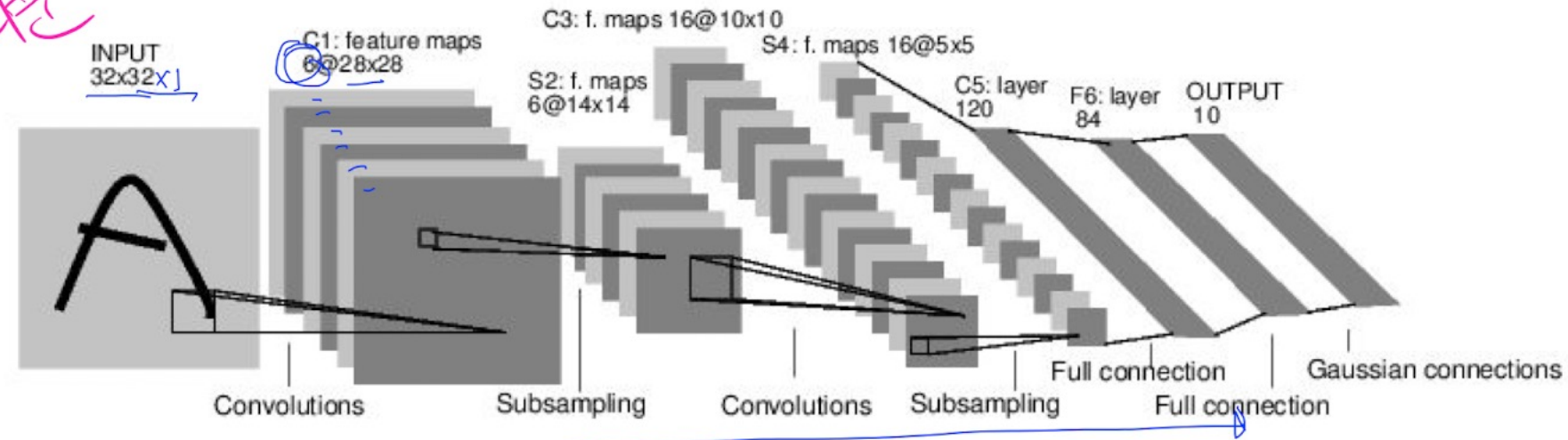


FC Layer

- CNN에서 마지막 Layer
- 일반적인 Neural Network
- FC Layer에서 마지막 Layer는 output (label과 비교해서 loss를 구하는)

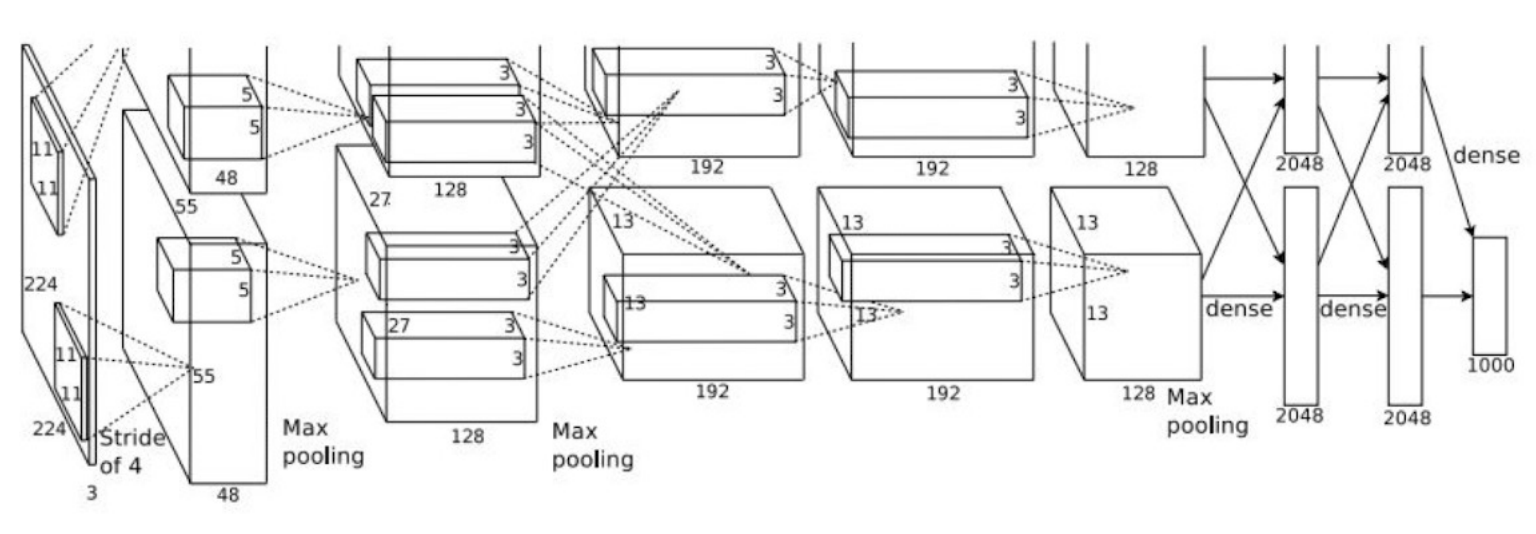
Case1- LeNet-5

[LeCun et al., 1998]



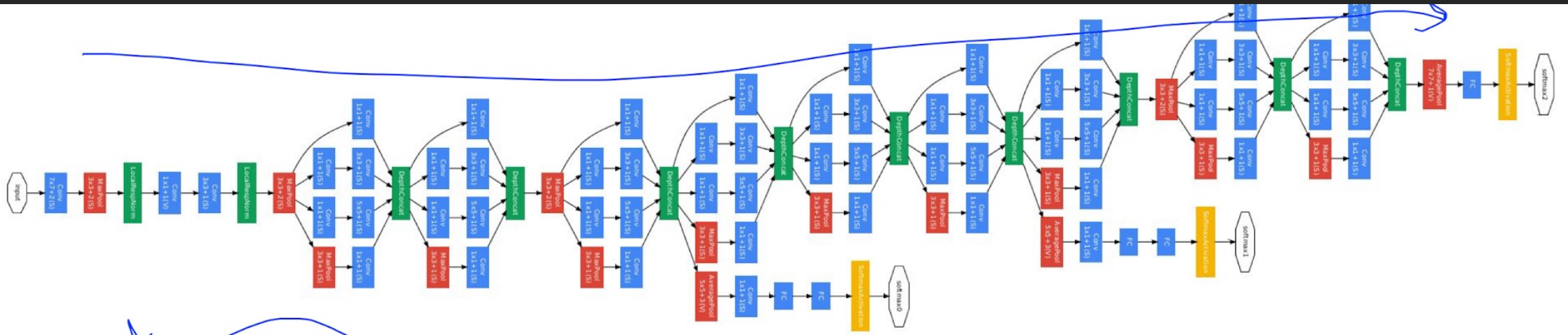
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

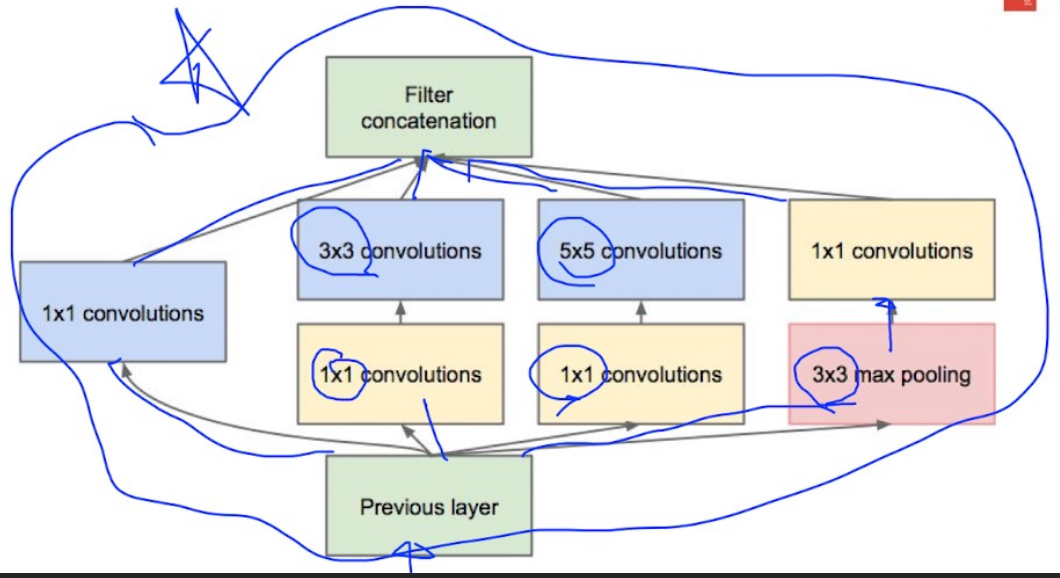
Case2- AlexNet



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)

Case3- GoogleLeNet

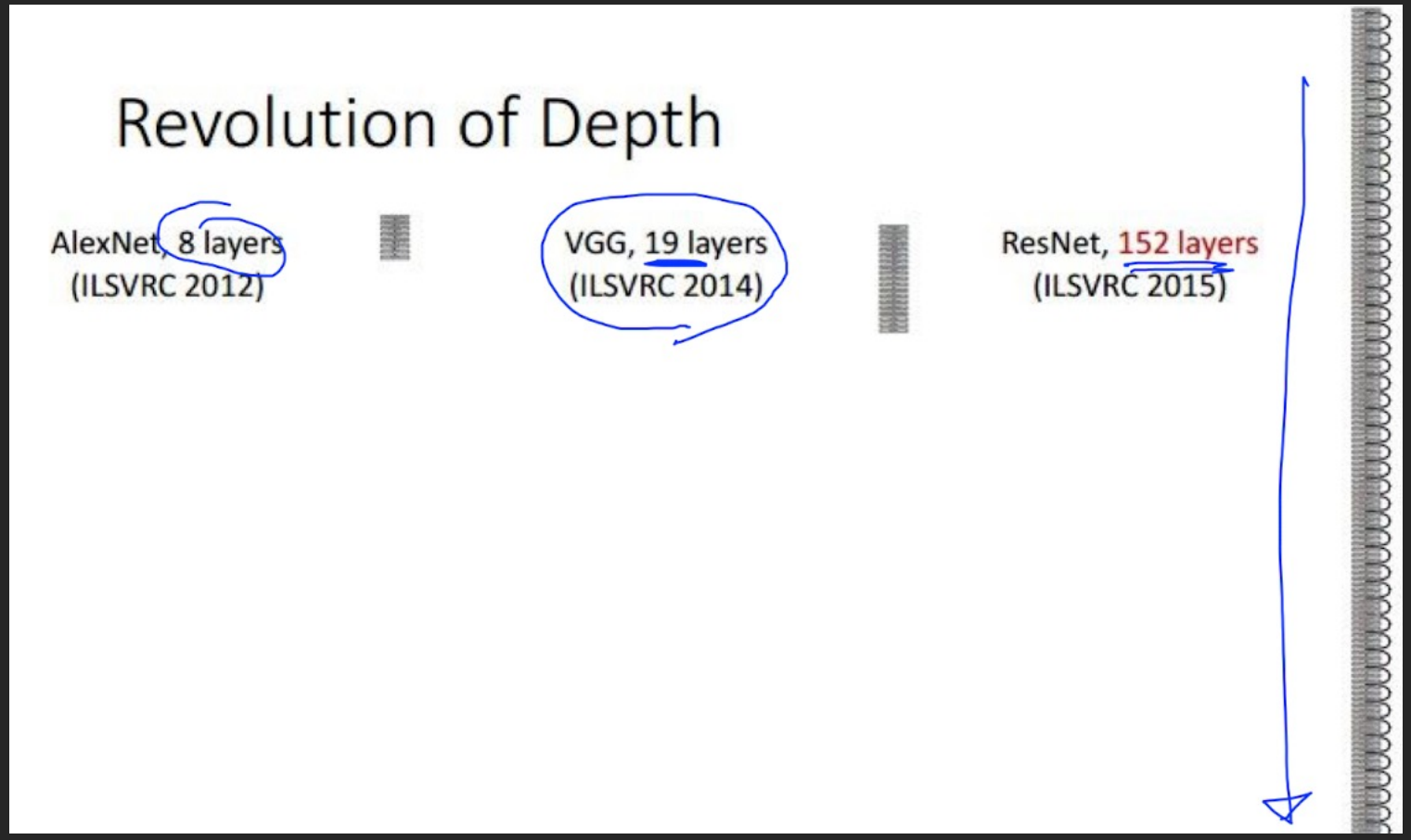




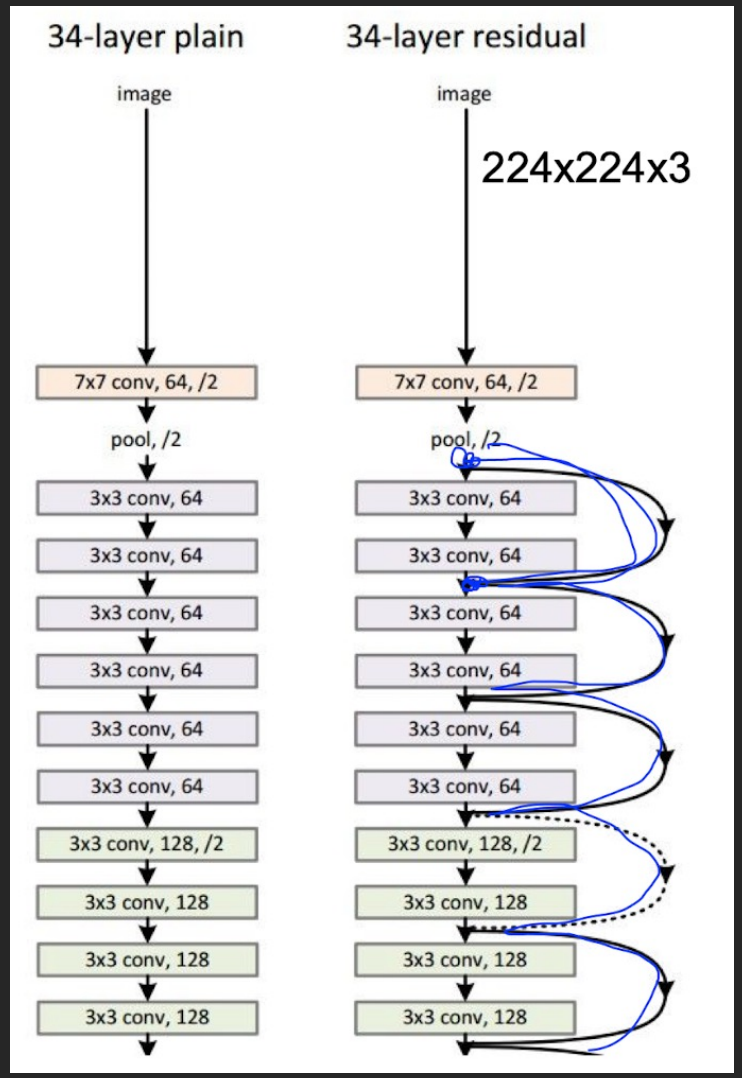
Inception module

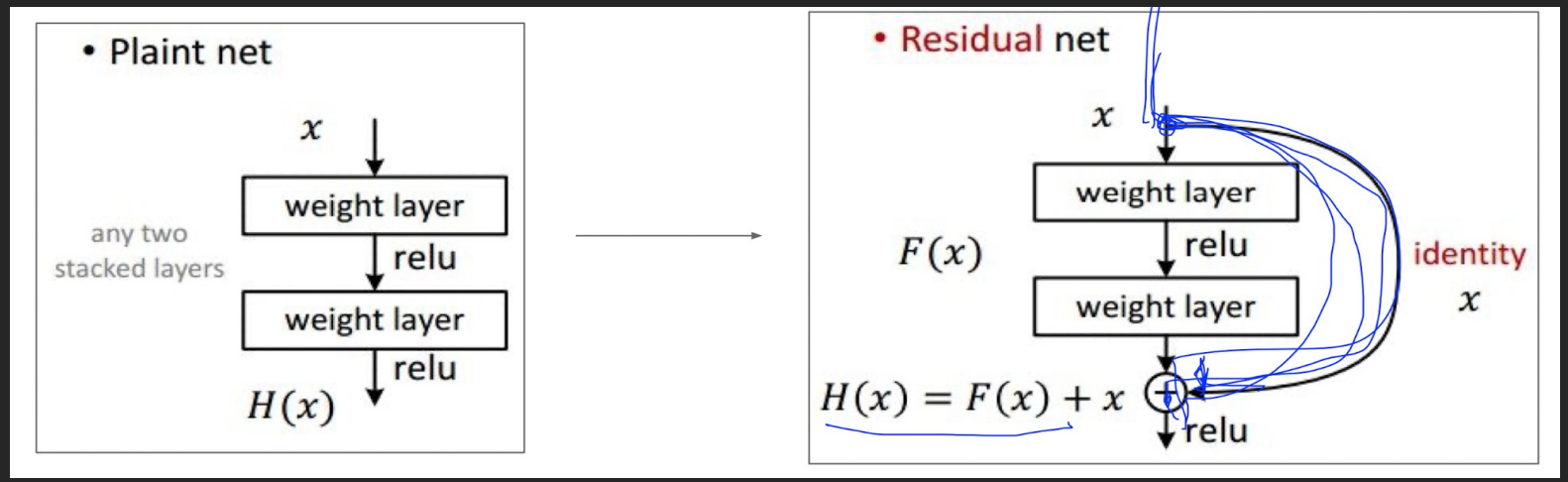
ILSVRC 2014 winner (6.7% top 5 error)

Case4- ResNet



Case4- ResNet





Case4- ResNet