스마트 에너지를 위한
이미지 처리와 CNN 모델

2024년 1월 22일~1월 23일

Convolution Neural Network (CNN)

A bit of history:

Hubel & Wiesel,

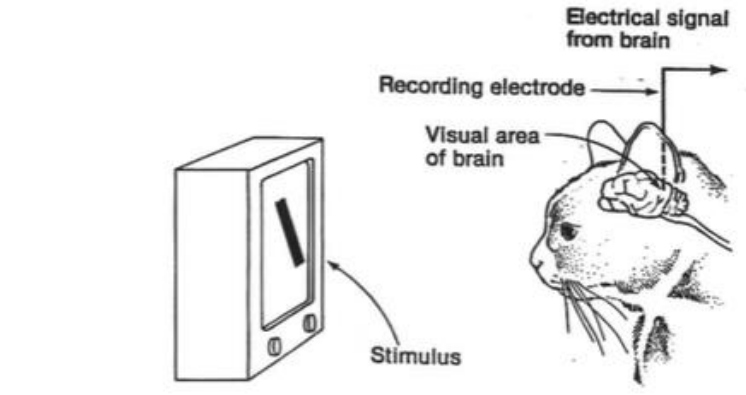
1959

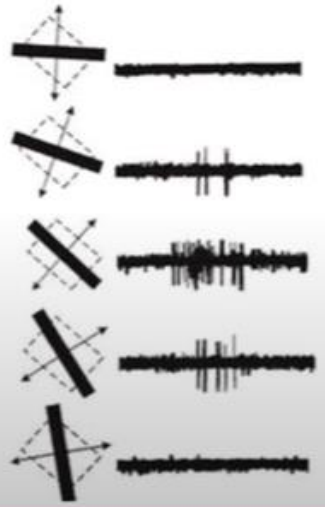
RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

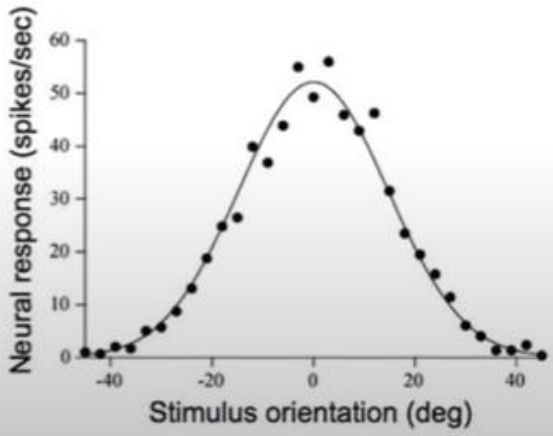
1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

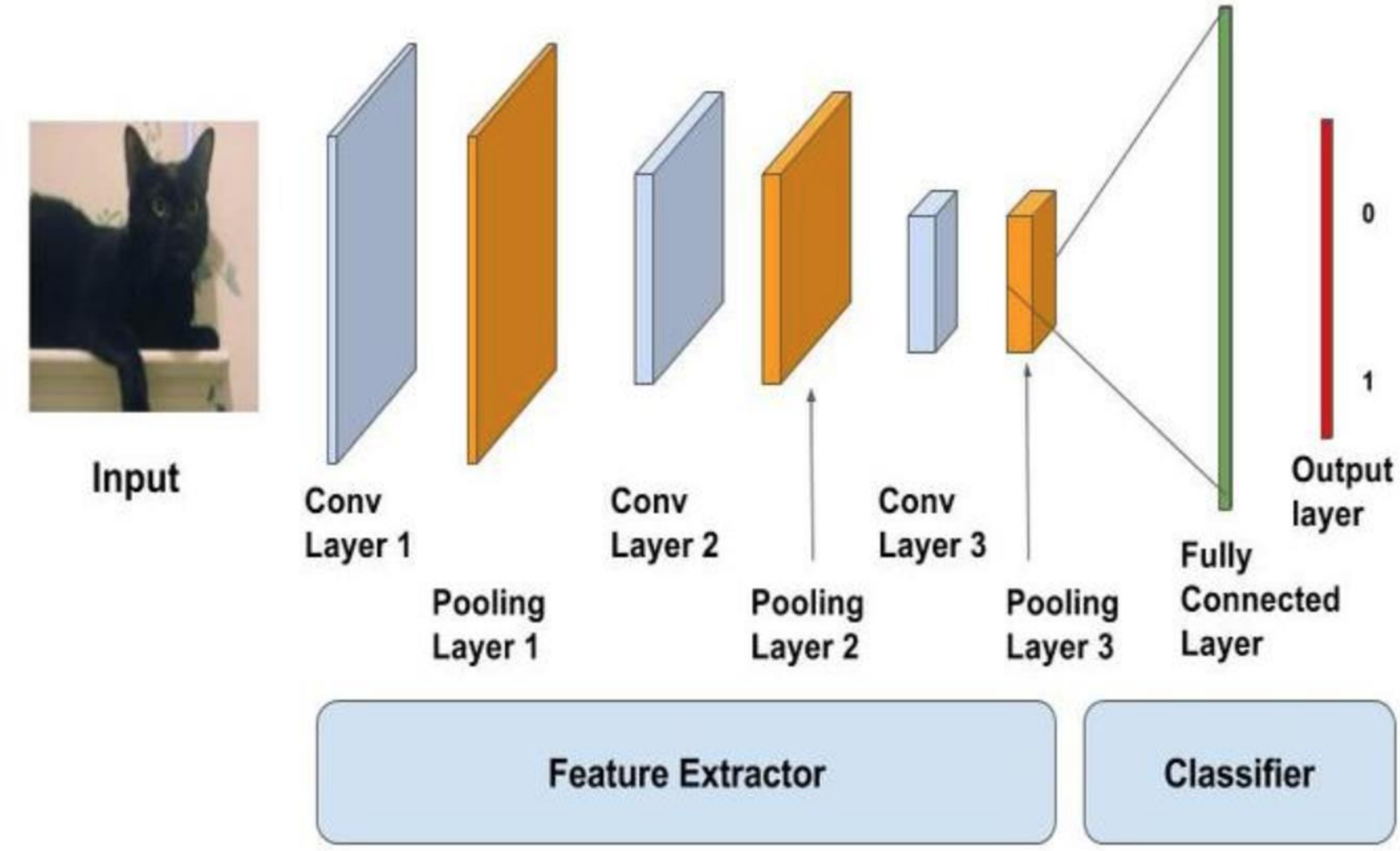
1968...





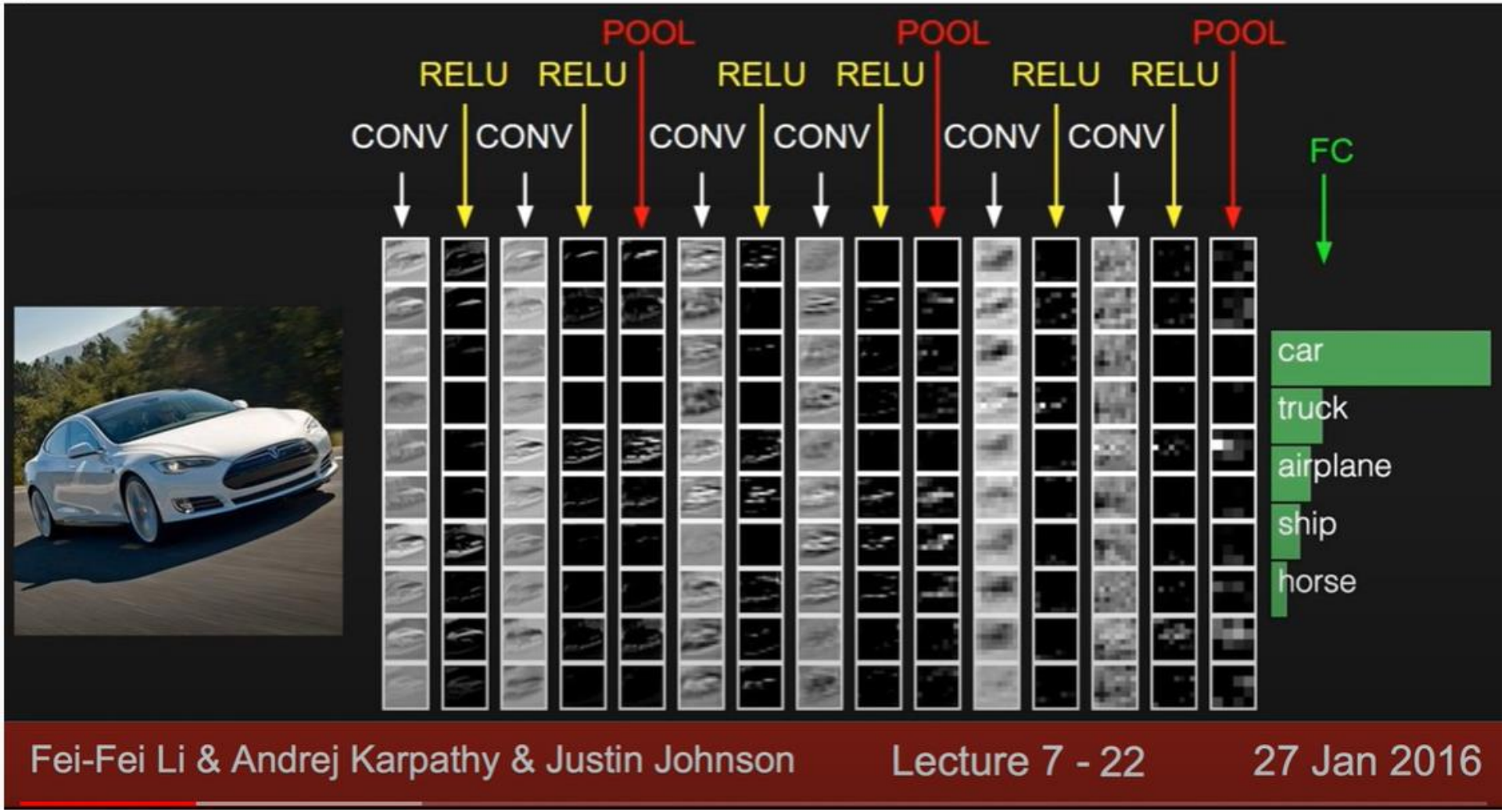


Convolution Neural Network (CNN)

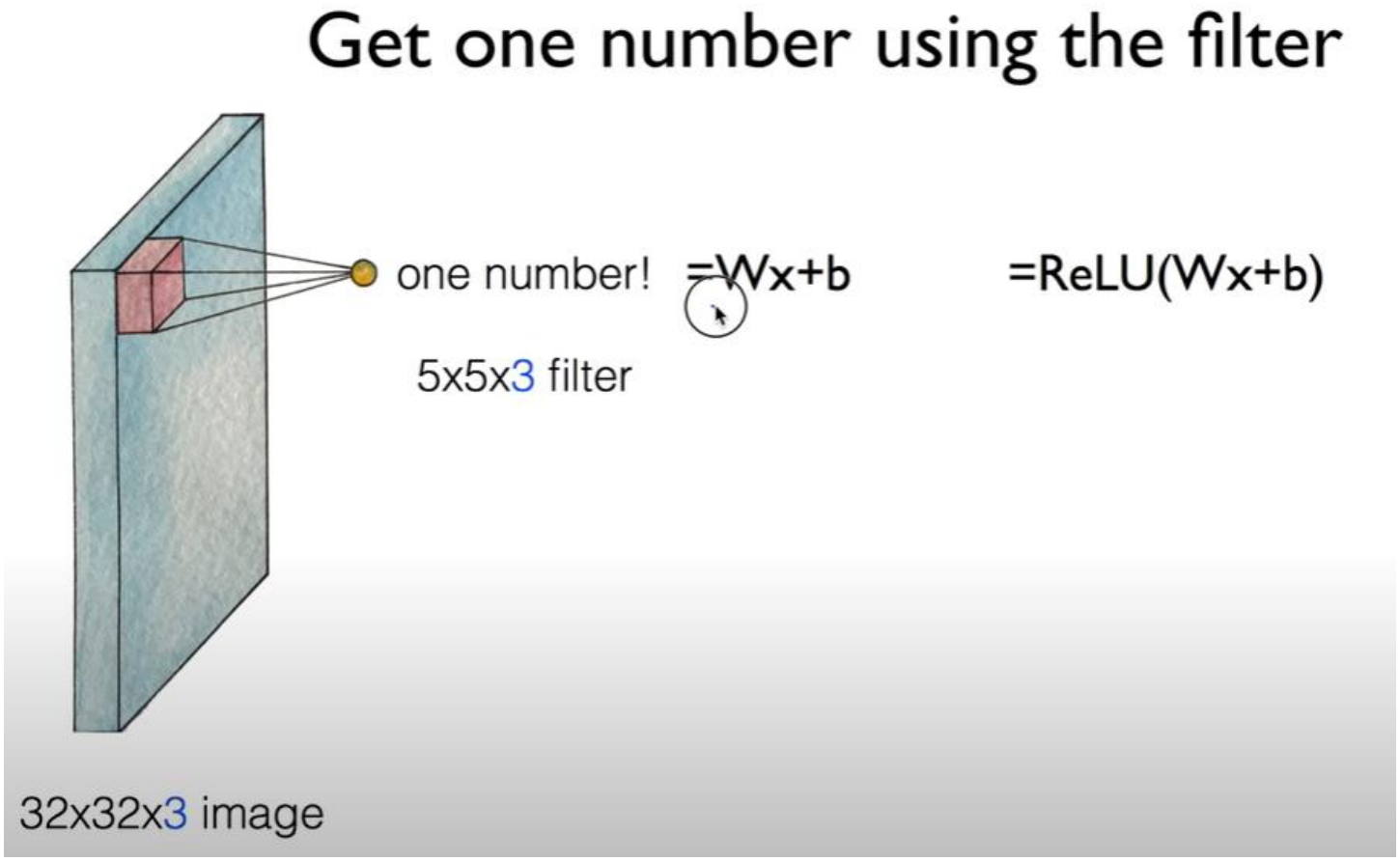


CNN

preview:


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse
Fei-Fei Li & Andrej Karpathy & Justin Johnson
Lecture 7 - 22
27 Jan 2016

CNN 동작

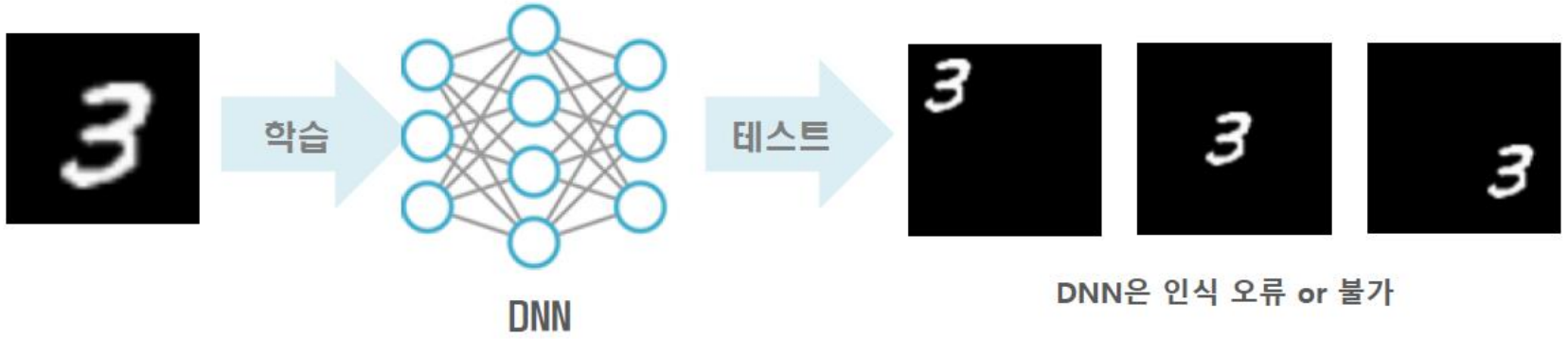

Get one number using the filter
one number!
=Wx+b
=ReLU(Wx+b)
5x5x3 filter
32x32x3 image

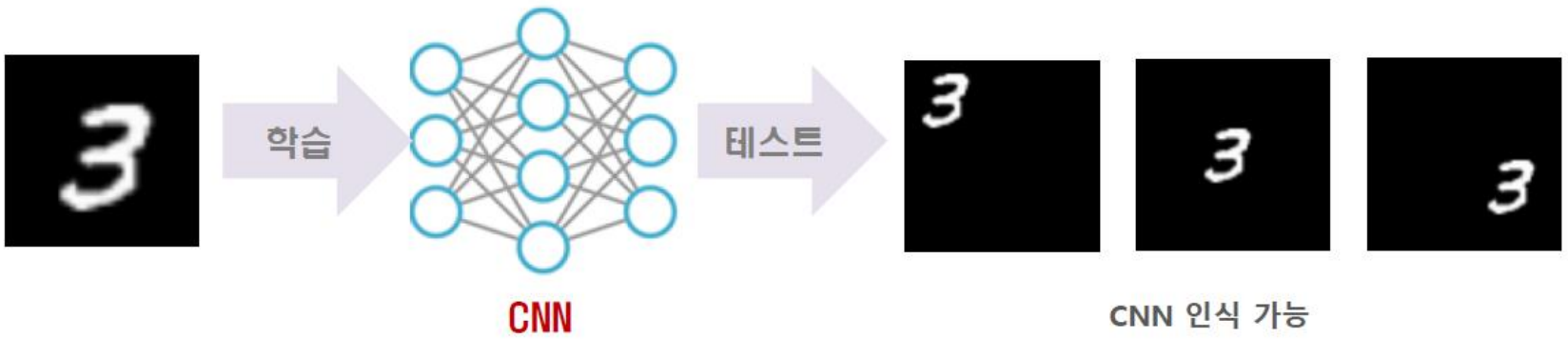
MLP의 한계

- 계층 수와 유닛(뉴런) 수의 곱에 비례하여 가중치 수가 급격히 증가
  - 1000개의 유닛으로 구성된 계층이 2개만 있어도 1백만개의 계수가 필요

- 이미지 처리에 사용할 경우
  - 학습 패턴의 위치에 민감하게 동작
  - 아래 세 개의 5는 패턴이 다르다고 판단한다. MLP로 이러한 숫자 인식을 하려면 숫자의 크기를 비슷하게 맞추어야 한다

MLP의 한계





CNN 특징

- MLP 신경망의 한계를 극복하기 위해서 제안
- 전결합망 구조를 사용하지 않고, 좁은 면적 단위(예를 들면 3x3 픽셀 단위)로 신호를 필터링하고, 그 결과를 다음 계층의 입력으로 사용
- 작은 공단 단위(패치)로 필터링하는 목적
  - 이미지의 특징을 구성하는 어떤 패턴을 공간상 위치에 상관없이 찾아내기 위해서

특성 맵 (Feature map)

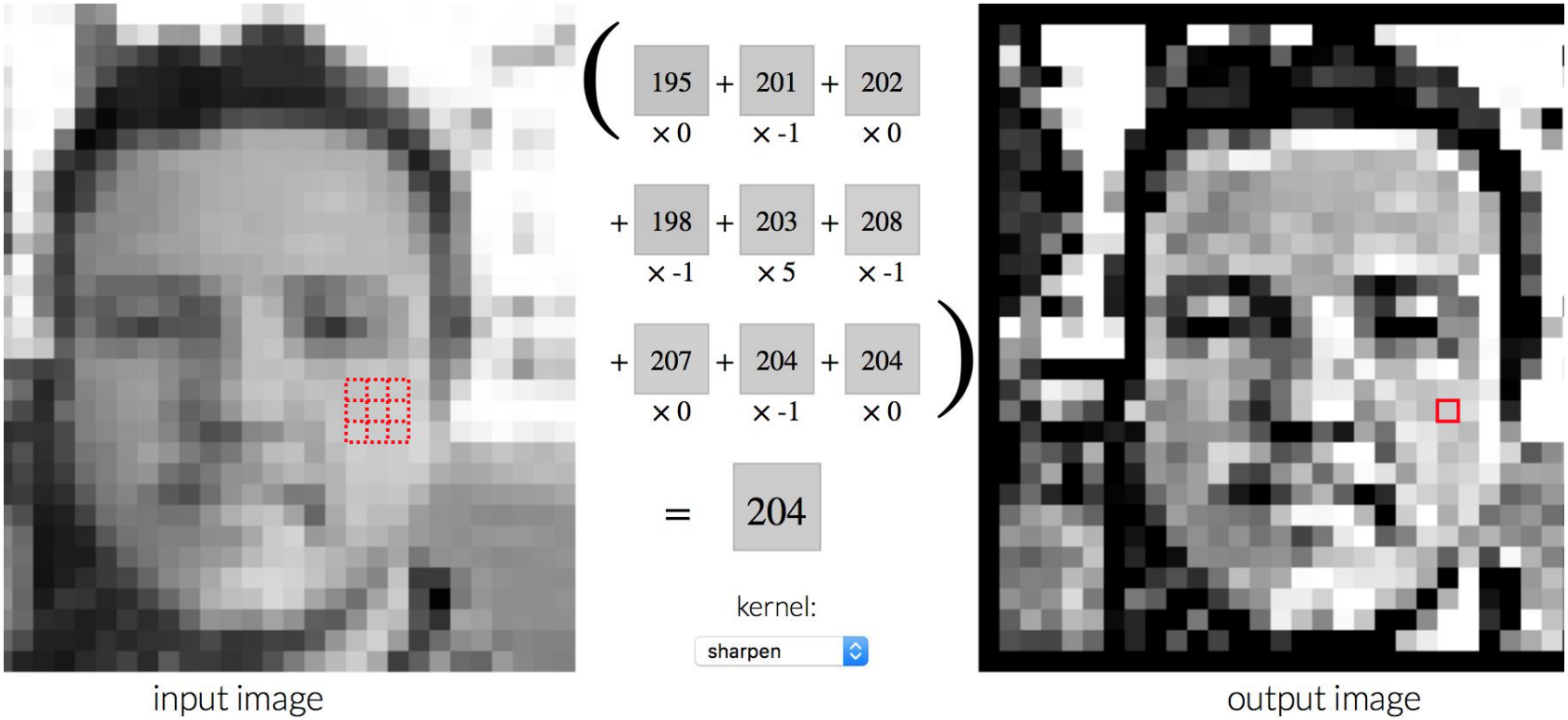
- 특징 패턴
  - 기하학적인 단위 모양(엣지, 대각선, 수평선, 수직선 등)
  - 질감(texture) 등

- 특성맵(feature map)
  - 다음 계층으로 넘겨주는 유효한 값을 활성값(activation)이라고 함
  - 활성값의 전체 집합

- CNN의 동작은, 마치 돋보기로 이미지 전체를 차례대로 스캔하면서 특정 성분을 파악하는 것과 같다

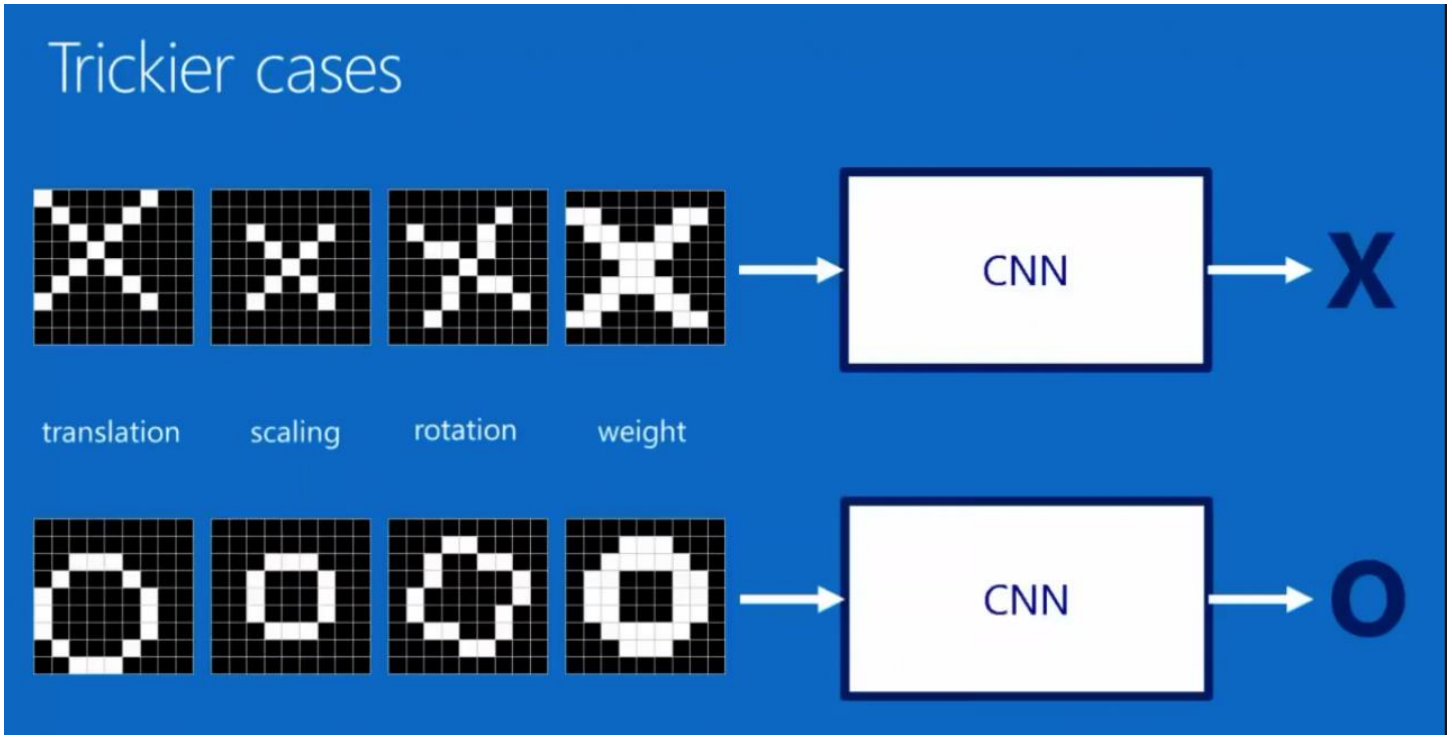
이미지 필터링

- 시각화


195 + 201 + 202
× 0 × -1 × 0
+ 198 + 203 + 208
× -1 × 5 × -1
+ 207 + 204 + 204
× 0 × -1 × 0
= 204
kernel:
sharpen
input image
output image

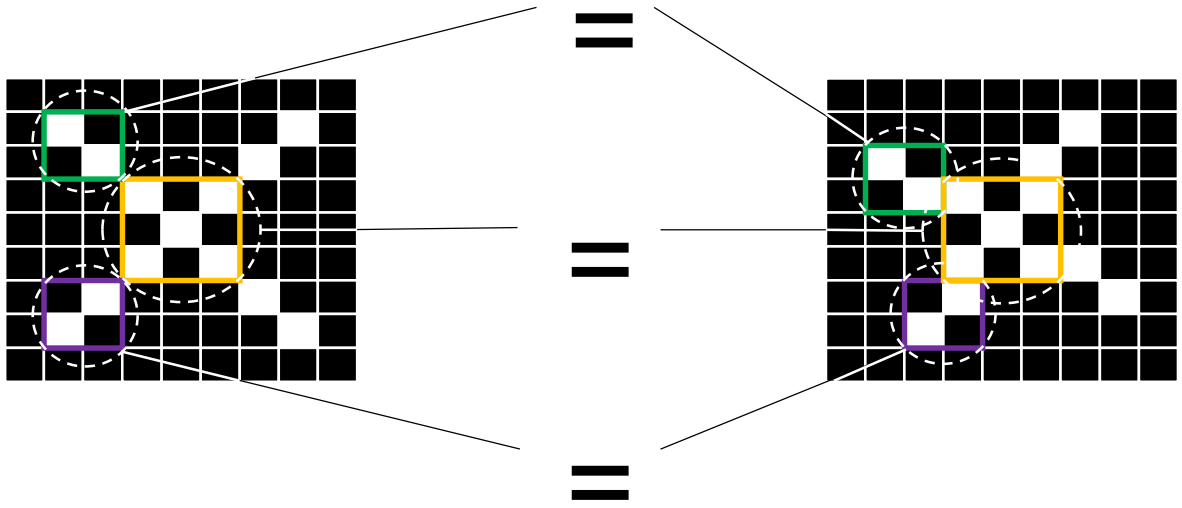
CNN 개념 소개

- 동영상(2:40) (원본(26:30))



동일한 패턴 검출

- 공간상의 위치를 픽셀 단위로 일 대 일 비교하면
  - 이 두 그림에서 흰 점이 일치하는 부분은 거의 없다.
  - 따라서 이미지를 벡터로 환산(즉, 1차원 벡터로 변환)하여 비교하면 서로 다르다고 판단
- 3x3 픽셀 단위로 나누어 관찰하면
  - 절대적인 위치는 다르지만 일치하는 패턴이 여러 곳에 나타나므로
  - 다른 위치에 있는 같은 패턴을 찾아낼 수 있다



위치, 크기, 두께, 각도 등에 invariant

- CNN의 필터는 바로 이러한 패턴의 활성화 성분 크기 즉, 활성값을 찾아 준다.
- 이러한 원리를 이용하여 CNN은 어떤 패턴의 크기, 절대적 위치, 두께, 회전 각도 등이 다르더라도 이를 찾아서 다음 계층으로 전달할 수 있다.

커널 (kernel) 수

- CNN에서는 특성맵을 한가지만 만드는 것이 아니라 수십~수백 가지를 만든다. (처음에는 3원색인 경우 3에서 출발)

- 왜냐하면 찾아내야 할 패턴이 가로성분, 세로성분, 대각선성분, 'X' 형 성분, 색상정보, 엣지 등 여러 가지가 있기 때문

-

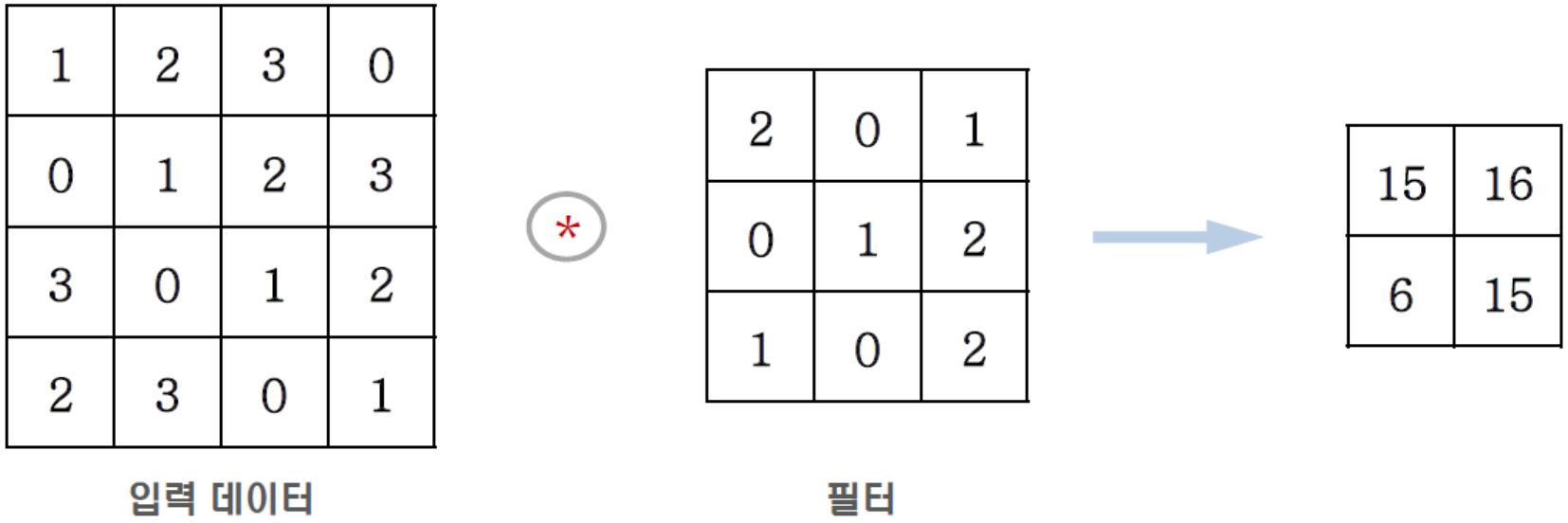
- 어떤 계층의 합성곱 필터의 종류 수가 32라면 이 계층에서 생산되는 특성맵은 32개가 된다.

- 합성곱 필터를 커널(kernel)이라고 부르는데 커널수가 많을수록 다양한 패턴을 찾아낼 수 있다

- 그러나 모델의 복잡도가 너무 커지면 과대적합의 원인
  - 내부 parameter 수가 많아 학습 데이터를 너무 정교하게 모델링하기 때문

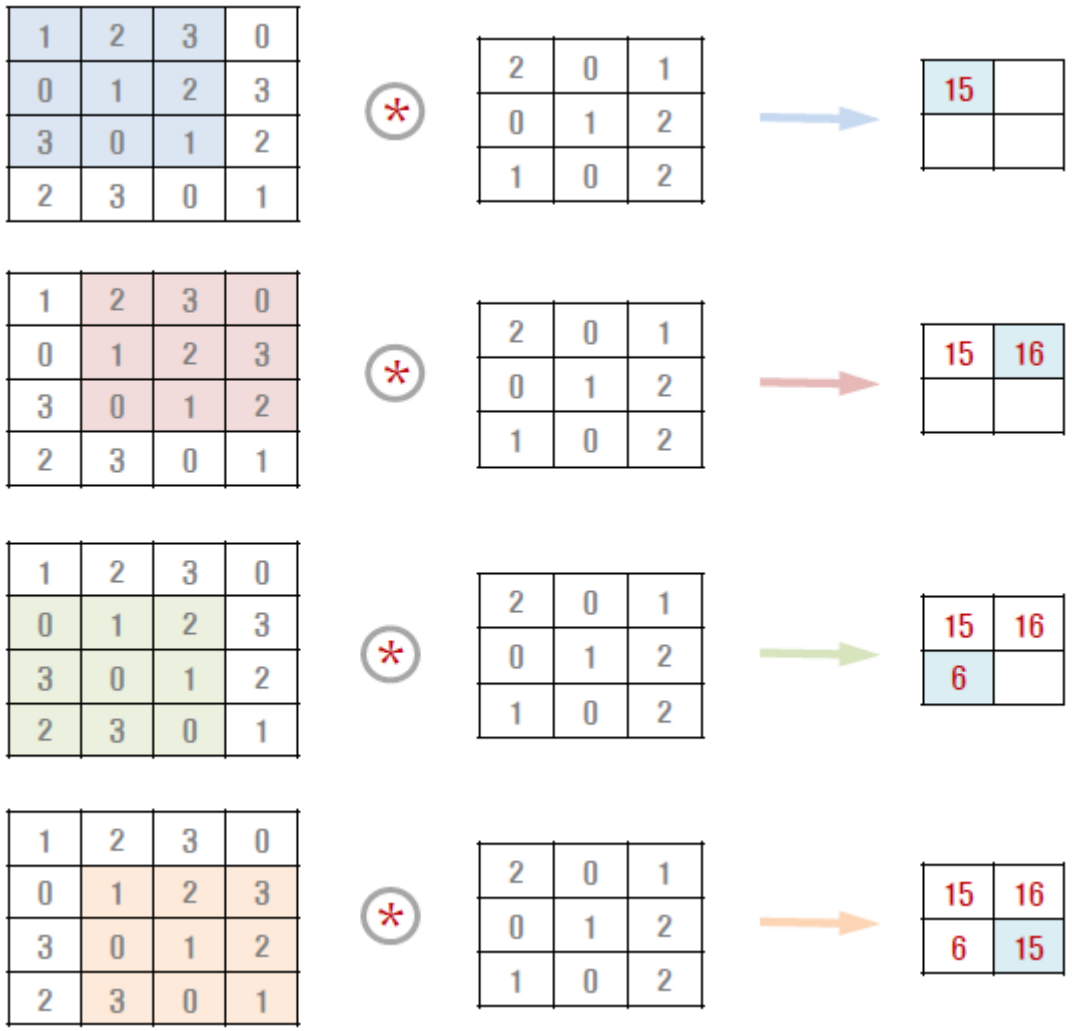
Convolution Filter

입력 데이터(height, width)에 대해 **필터(커널)**을
일정 간격(**Stride**) 만큼 이동해 가며 행렬 곱셈 연산 수행

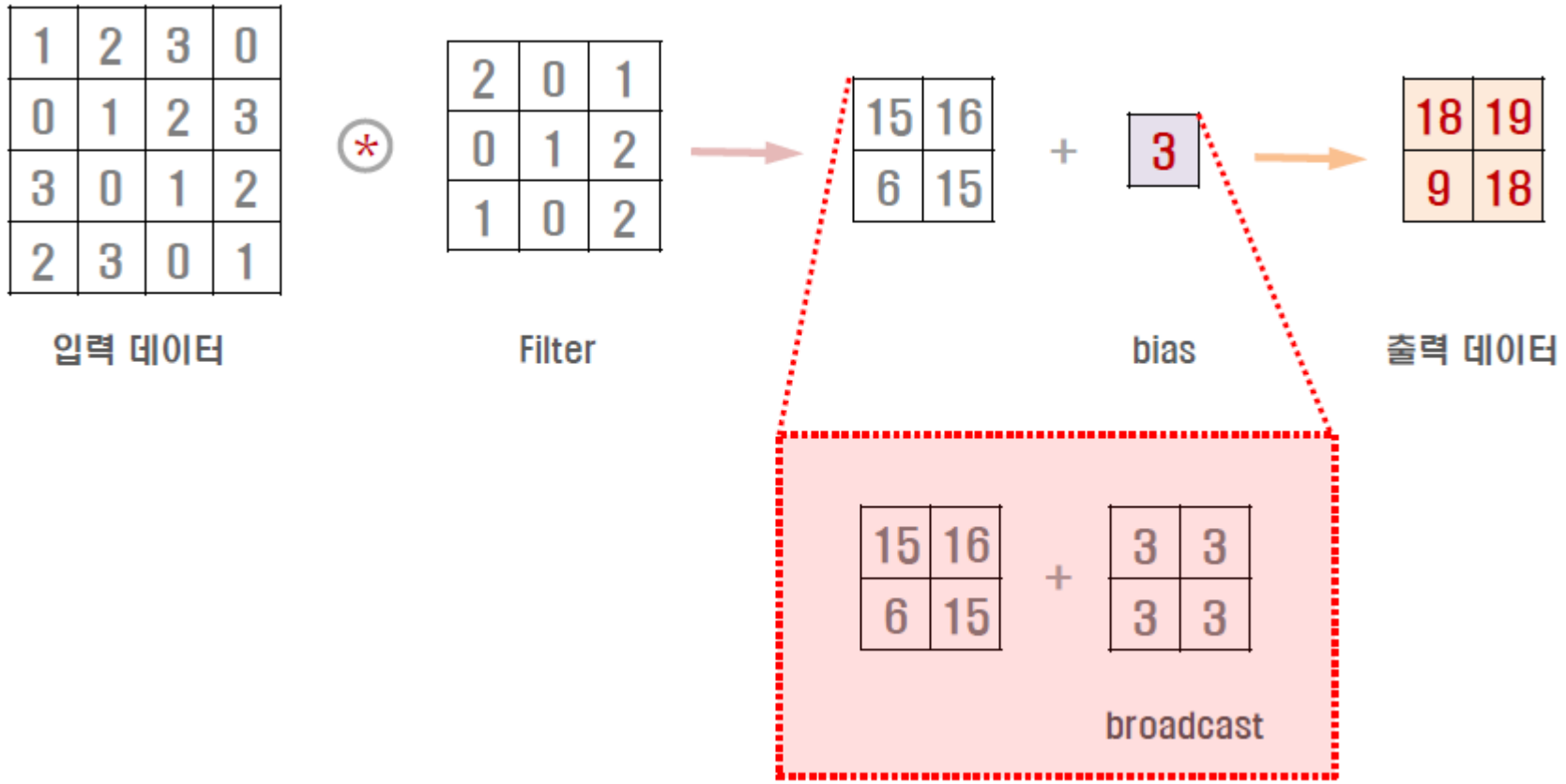


입력 데이터 필터

Convolution Filter 연산



Convolution Filter 연산



패딩 (Padding)

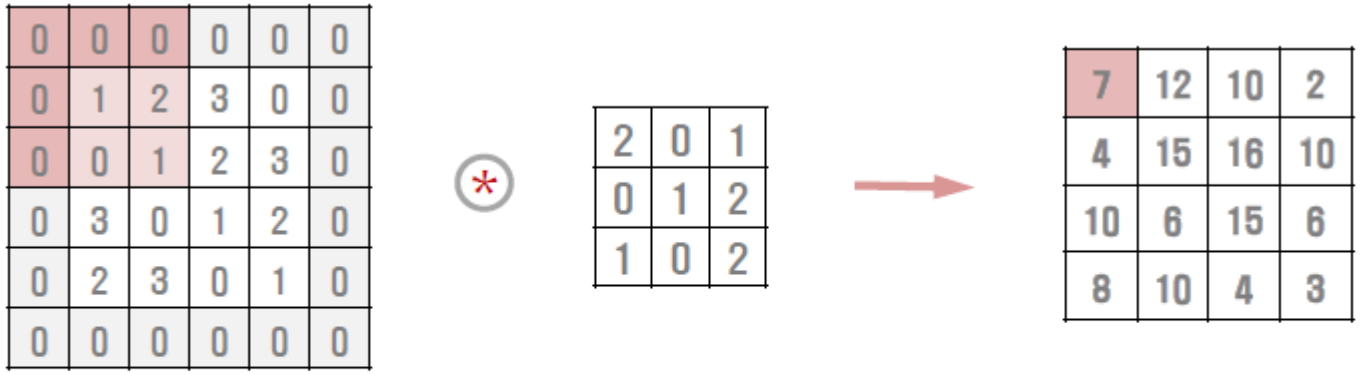
- 필터의 크기는 5x5나 7x7 등 임의의 크기로 정할 수 있다.

- 필터의 크기로 인해 가장자리 부분의 데이터가 부족
  - 입력과 출력의 크기가 달라지게 된다
  - 3x3 윈도우의 패치를 사용하는 가로, 세로 각 2 만큼씩 축소(상하좌우 가장자리에 픽셀이 한 줄씩 부족)

- 이를 보정하기 위해서 입력신호의 가장자리 부분에 보통 0을 미리 채워 넣는 것을 패딩(padding)

- Conv2D 계층 : padding 인자를 사용
  - valid : 패딩을 사용하지 말라는 의미
  - same : 출력의 차원이 입력과 같아지도록 적절한 수의 패딩을 자동으로 입력하라는 의미

패딩 (Padding)



입력 데이터(4, 4)
zero padding(1,1)
추가(6,6)

Filter(3, 3)

출력 데이터(4, 4)

축소 샘플링 (subsampling)

- 합성곱을 수행한 결과 신호를 다음 계층으로 전달할 때, 모든 정보를 전달하지 않고 일부만 샘플링하여 넘겨주는 작업

- 축소 샘플링을 하는 이유
  - 좀 더 가치 있는 정보만을 다음 단계로 넘겨주기 위해서

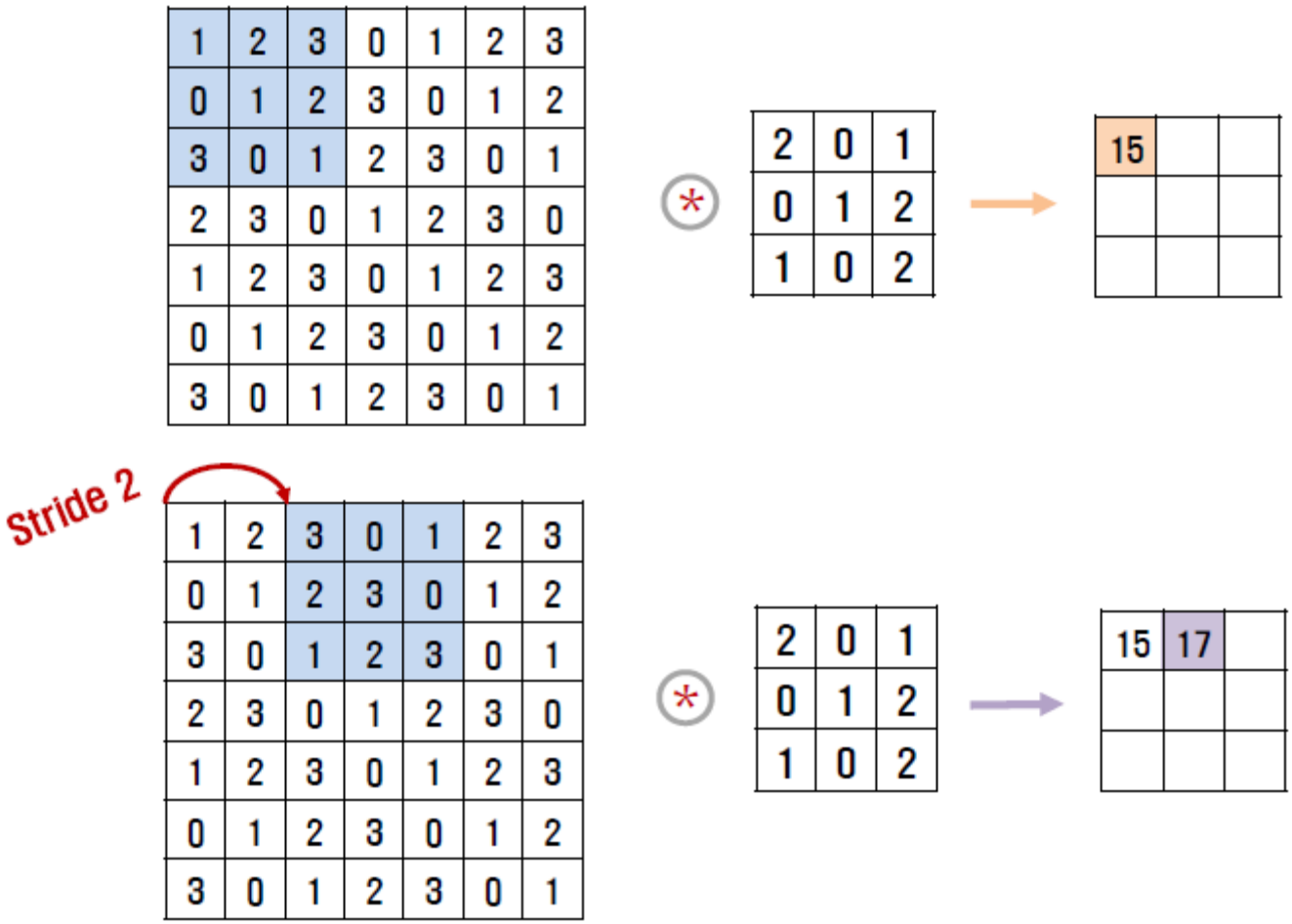
- 머신러닝의 최종 목적은 정보를 결국 줄여나가야 하며 따라서 핵심 정보만 다음 계층으로 전달하는 장치가 필요
  - 커널 수를 늘리면 특성맵의 숫자가 점차 커지게 된다.

- 축소 샘플링
  - 스트라이드(stride)
  - 풀링(pooling)

스트라이드 (Stride)

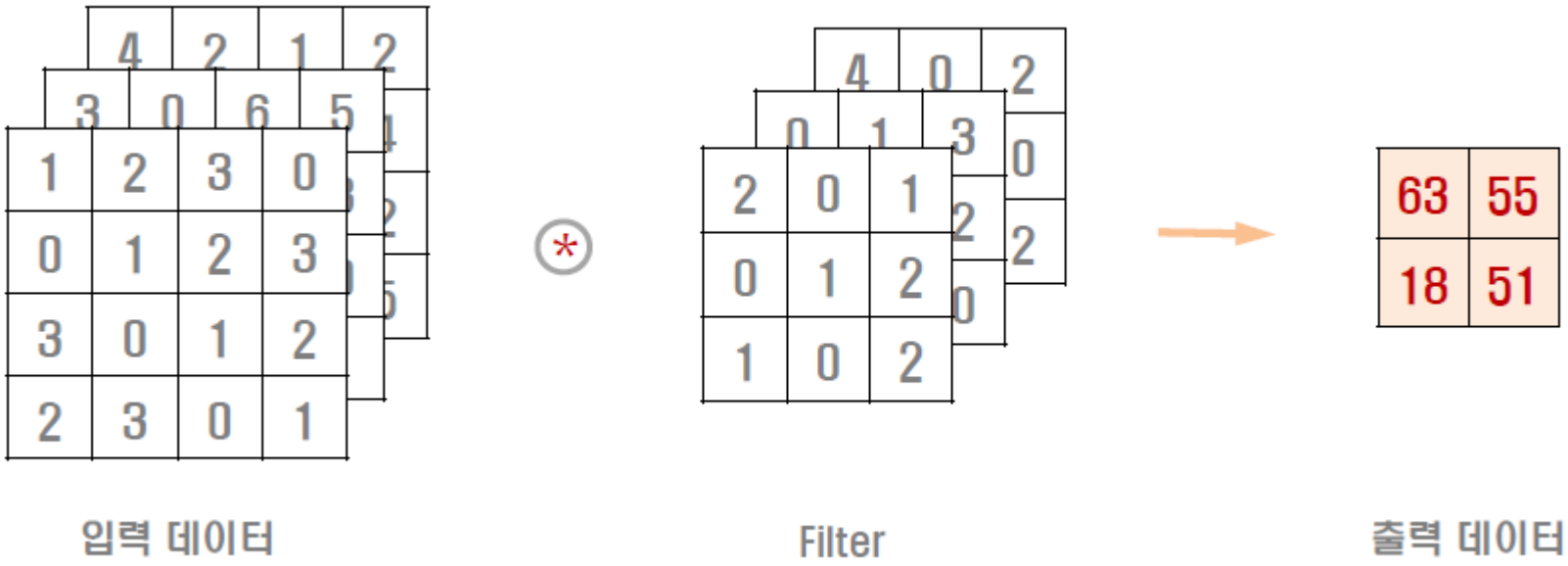
- 합성곱 필터링을 수행할 때 패치를 (예를 들면 3x3 크기)를 한 픽셀씩 옆으로 이동하면서 출력을 얻지 않고, 2 픽셀씩 또는 3 픽셀씩 건너 뛰면서 합성곱을 수행하는 방법
  - 스트라이드 2 : 출력 특성맵의 크기가 1/4로 축소
  - 스트라이드 3 : 출력 특성맵의 크기가 1/9로 축소

스트라이드 (Stride)

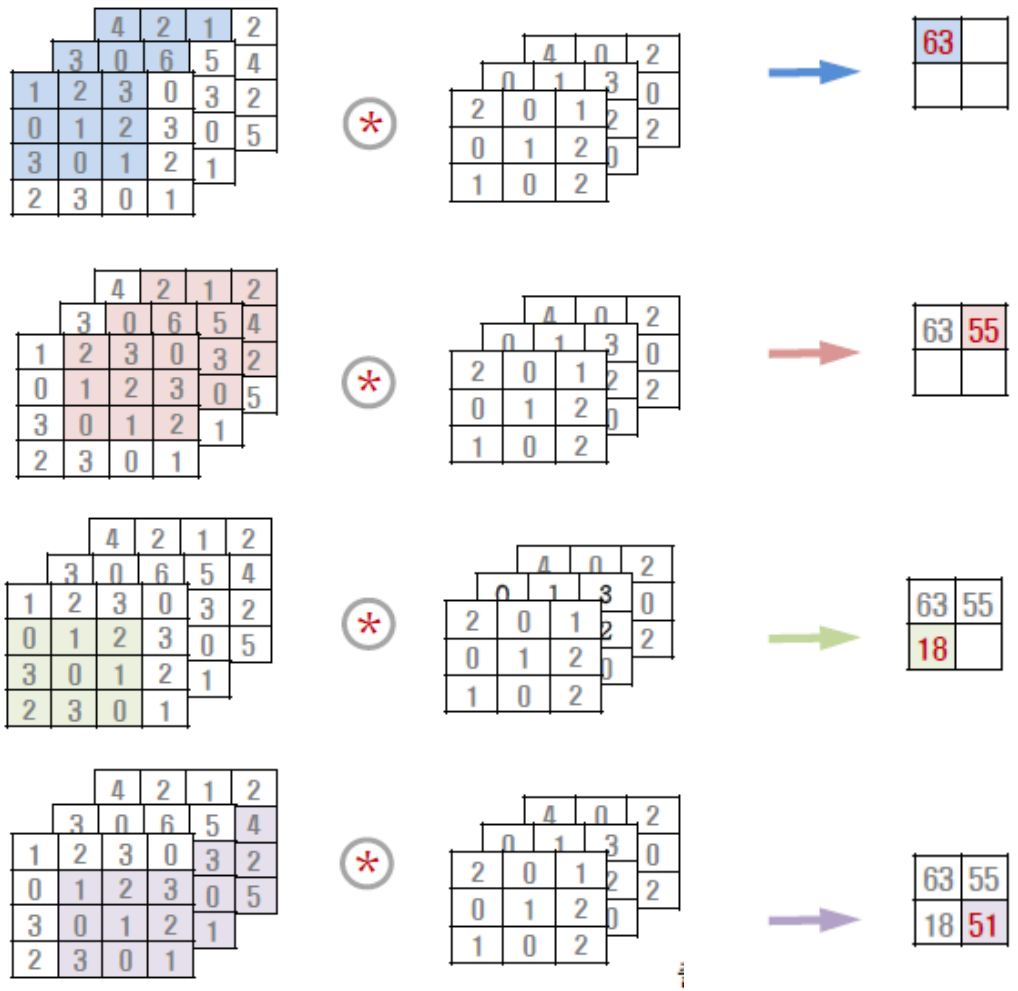


2차원 데이터 합성곱 연산 – 단일 필터

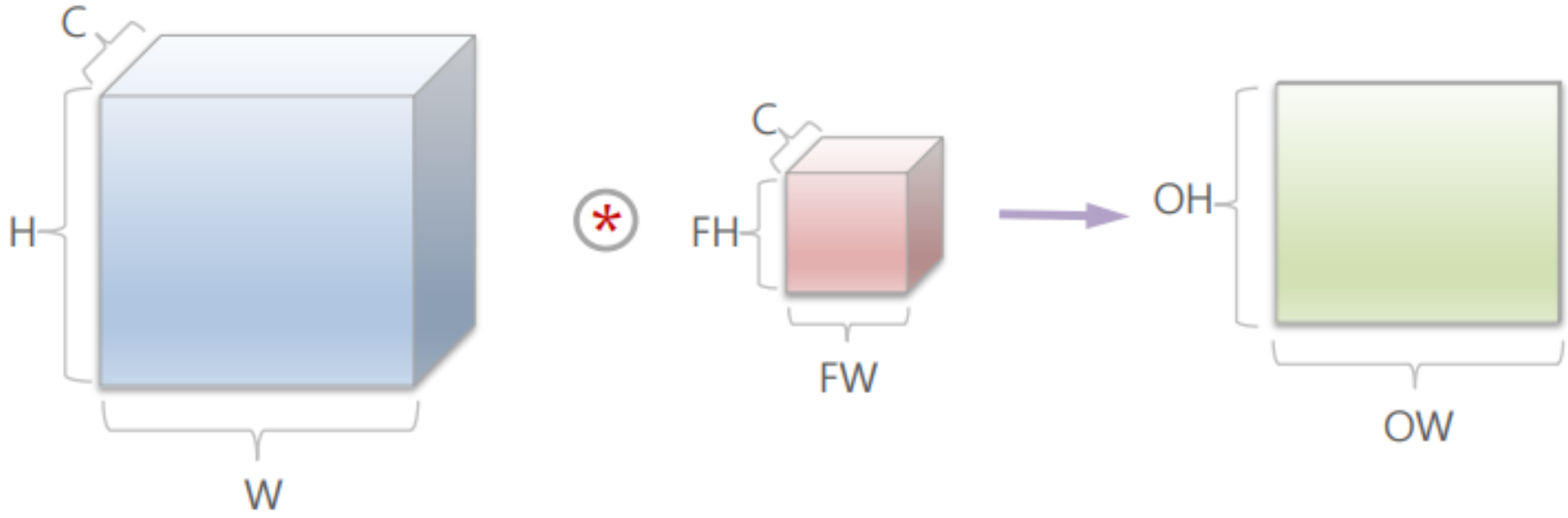
- 입력 데이터의 채널 수와 필터의 채널 수 일치
- 모든 채널의 필터가 같은 크기



2차원 데이터 합성곱 연산 – 단일 필터



3차원 데이터 합성곱 연산 – 단일 필터

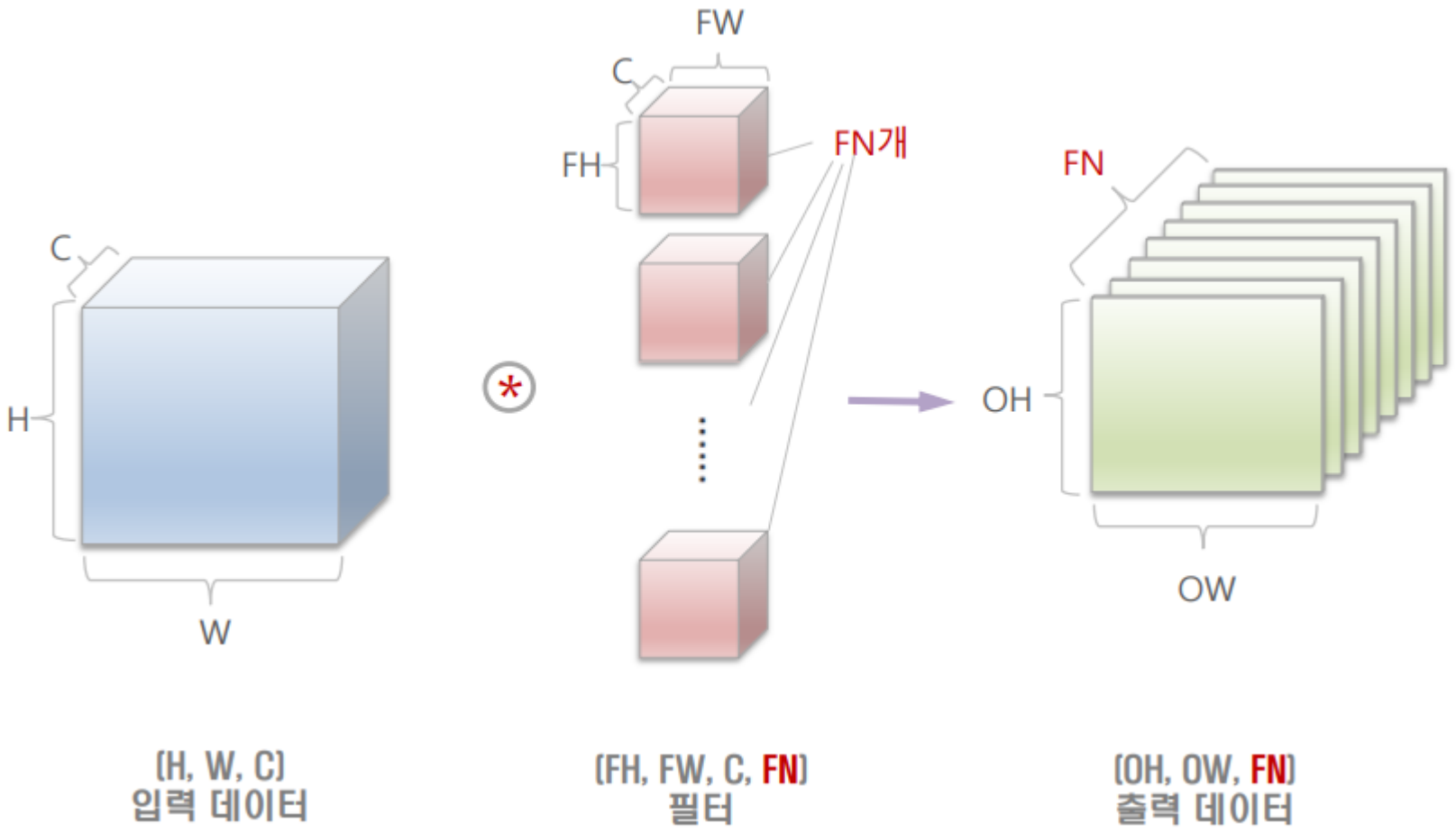


(H, W, C)
입력 데이터

(FH, FW, C)
필터

(OH, OW, 1)
출력 데이터

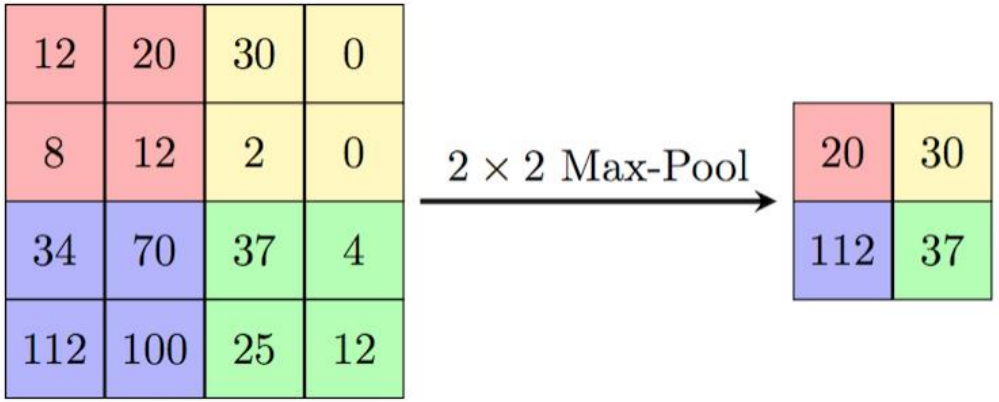
3차원 데이터 합성곱 연산 – 다중 필터



풀링 (Pooling)

- 축소 샘플링은 주로 풀링 작업을 통해 수행

- CNN에서 합성곱 수행 결과를 다음 계층으로 모두 넘기지 않고, 일정 범위 내에서 (예를 들면 2x2 픽셀 범위) 가장 큰 값을 하나만 선택하여 넘기는 방법을 사용



풀링 (Pooling)

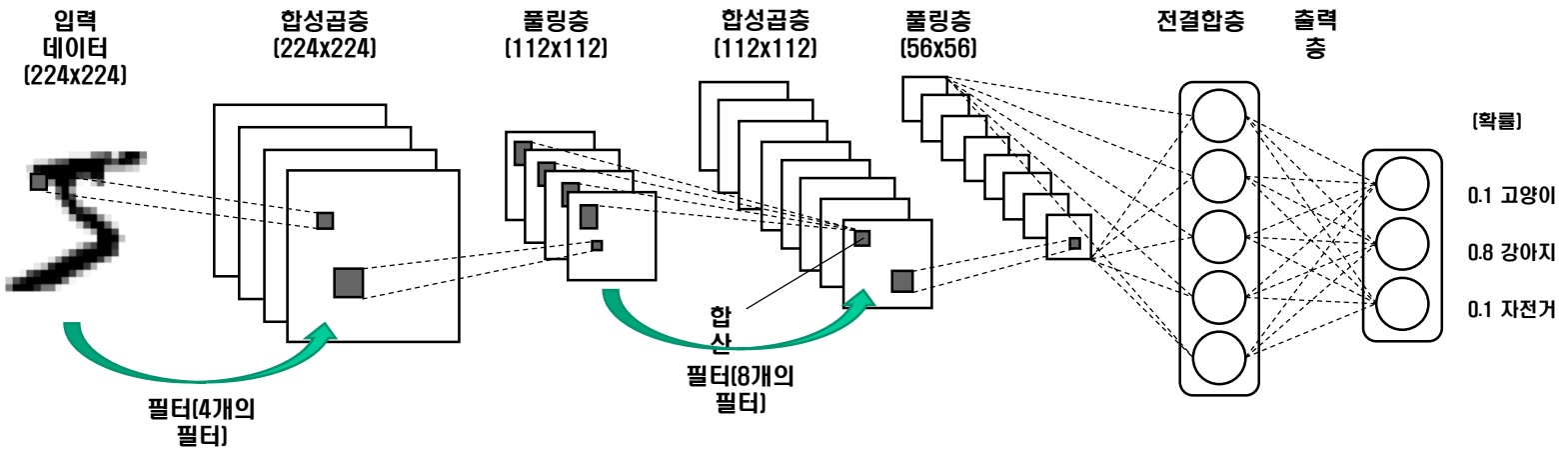
- 최대 풀링(max pooling)
  - 작은 지역 공간의 대표 정보만 남기고, 나머지 신호들을 제거하는 효과
  - 특정한 패턴이 공간상의 어느 위치에 있든 이 활성값이 다음 단계로 넘어가면서 좌우로 조금씩 움직일 수 있는 여지를 준다
  - 이러한 작업을 여러 단계 거치면 위치에 무관하게 특정 패턴의 유효한 값이 최종 출력단의 원하는 위치로 이동할 수 있게 된다.

- 또한 풀링은 과대적합을 해소하는 데에도 기여

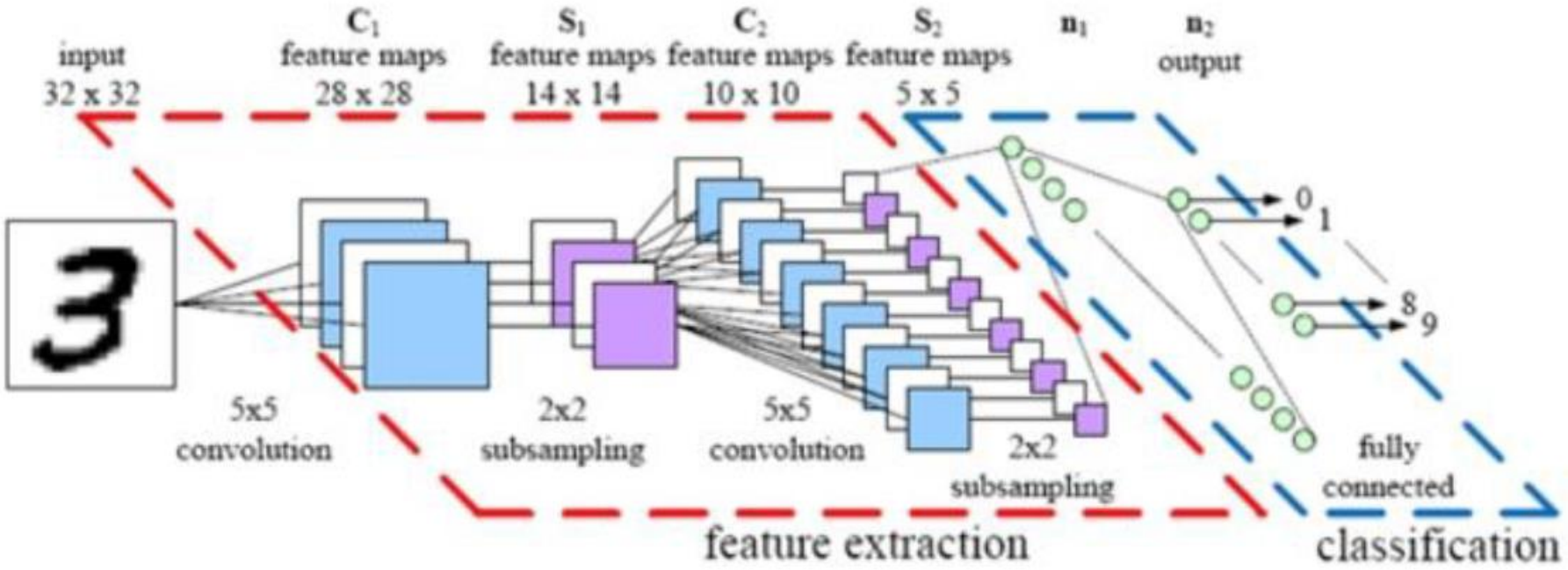
CNN 구성

- CNN은 일반적으로 커널 필터링과 활성화 함수 그리고 최대 풀링을 묶어서 하나의 계층을 형성한다.
  - 참고) MLP에서는 전결합망과 활성화함수를 묶어서 한 계층을 형성
- CNN에서도 다양한 구조의 활성화 함수를 사용하고 필요하면 전결합망을 사용한다.
- 분류를 수행하는 경우 최종 계층에서는 전결합망을 만들고 그 결과에 소프트맥스 함수(또는 시그모이드함수)를 적용

CNN 구성


입력
데이터
(224x224)
합성곱층
(224x224)
풀링층
(112x112)
합성곱층
(112x112)
풀링층
(56x56)
전결합층
출력
층
(확률)
0.1 고양이
0.8 강아지
0.1 자전거
합
산
필터(4개의
필터)
필터(8개의
필터)

CNN 구성


input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

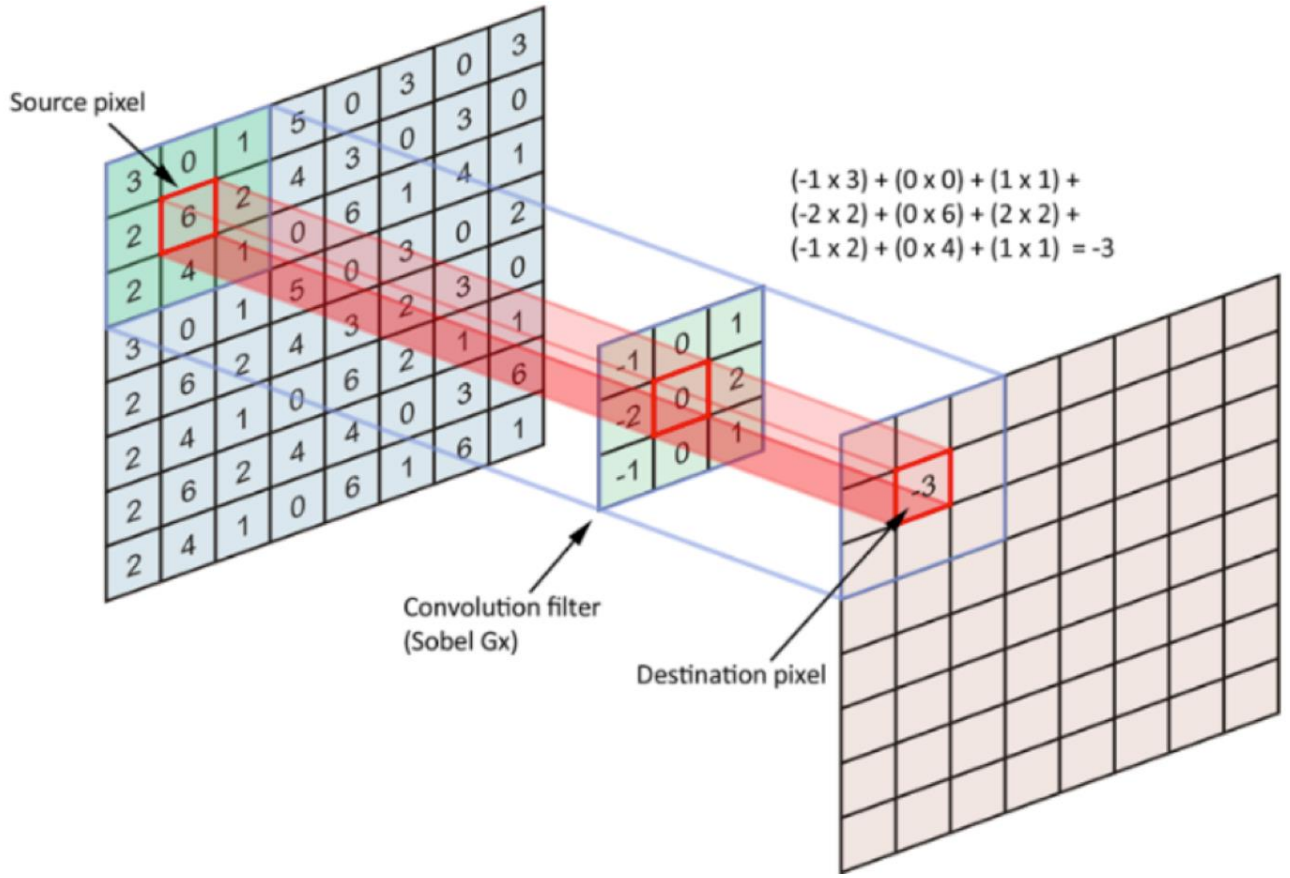
CNN의 특징

- MLP 구조
  - 입력 이미지 전체 픽셀의 특성을 한 번에 학습

- CNN
  - 입력 신호의 지역적인 특성들을 작은 단위로(예, 3x3 픽셀) 나누어 특정한 패턴이 있는지를 학습
  - 패턴으로는 기하학적인 단위 모양(엣지, 대각선, 수평선, 수직선 등)과 질감(texture) 등
  - 이러한 필터링은 스캔하듯이 입력 이미지 전체에 대해서 옆으로 이동하면서 적용
  - 따라서 CNN은 어떤 특정 패턴이 이미지 상에서 나타나는 위치가 변경되어도 이를 다시 찾아낼 수 있다

CNN의 특징

- 합성곱동작
- 다층 필터링

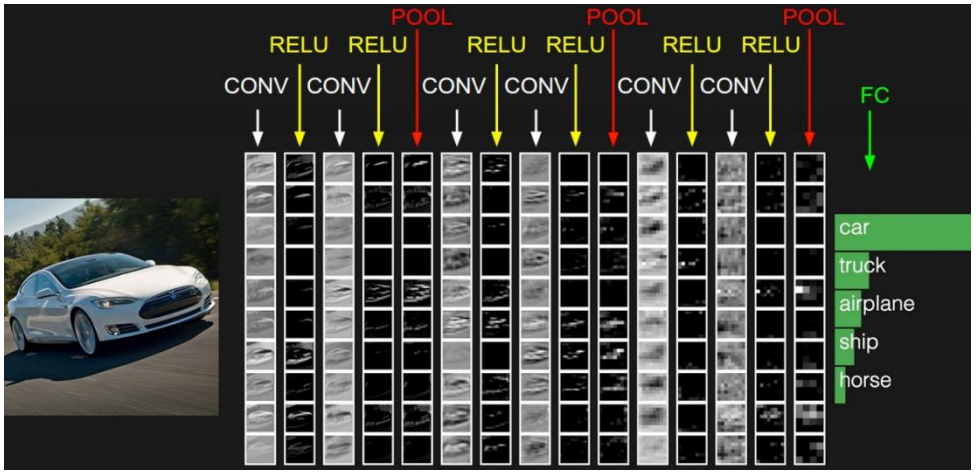


CNN의 특징

- CNN을 여러 계층으로 쌓으면 공간적인 계층구조(hierarchy)를 찾아낼 수 있다. 즉 계층을 올라가면서 점차 추상적인 내용을 파악할 수 있다

- 예를 들어 저층에서는 이미지의 기본적인 패턴을 찾고, 다음 계층에서는 이를 이용한 보다 복잡한 패턴을 찾는다

- 즉, 세밀한 것들이 모여서 큰 그림을 만들 듯이 이미지 전체를 이해하는 데는 이러한 공간적인 구조 파악이 중요한 역할을 한다



특성 맵 (Feature map)

- 평면을 구성하는 2차원 축 데이터와(폭, 높이), 깊이(depth) 축으로 구성 (이를 채널이라 함)

- 입력신호가 RGB 신호로 코딩된 경우, 깊이 축의 차원은 세가지 색을 각각 나타내는 3이 된다.

- 흑백으로 코딩된 경우 (MNIST처럼) 흑백의 그레이 스케일만 나타내면 되므로 깊이는 1이 된다.

- 컨볼류션 동작은 입력 특성맵으로부터 일정 영역을 추출하며 이들 영역들에 대해서 모두 동일한 변환(필터링)을 수행한다. 그 결과로 출력특성맵을 생성한다.

- 이 출력 특성맵도 역시 3D 텐서 구조를 갖는다. 필터링을 할 때 깊이의 크기를 변경할 수 있다. (필터의 개수가 깊이의 차원이 된다.)

특성 맵 (Feature map)

- **여기서** 주의할 것은 출력 특성맵 깊이의 각 채널이 여전히 입력 신호가 가지고 있던 RGB 신호 정보를 유지하는 것이 아니라는 것이다

- 이 채널은 입력 신호를 다양한 필터를 사용하여 필터링한 새로운 신호들의 집합을 나타낸다.

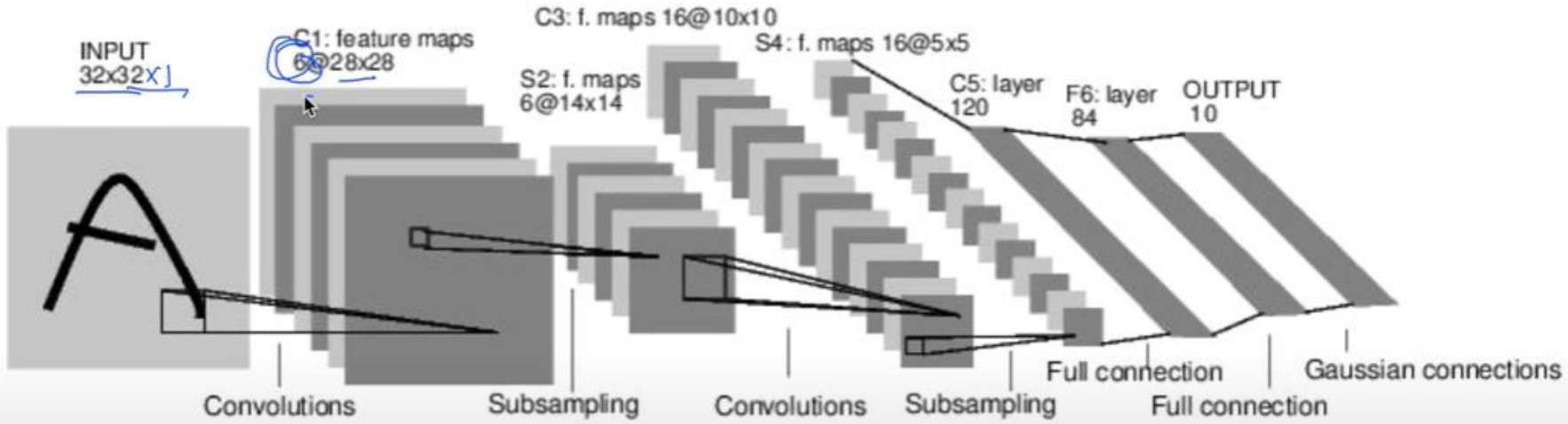
- **출력** 특성맵의 깊이(depth)는 컨볼류션 수행에 사용된 필터의 개수

- 컨볼류션 동작은 이 패치 크기의 윈도우를 옆으로 슬라이딩하면서 필터링 작업을 반복한다. 필터 가중치 매트릭스를 컨볼류션 커널이라고 한다.

- 필터 출력은 1차원 벡터가 되며 크기는 (output_depth,)가 된다. 이러한 1차원 벡터가 공간적으로 배치되어 출력 특성맵을 구성

Case Study: LeNet-5

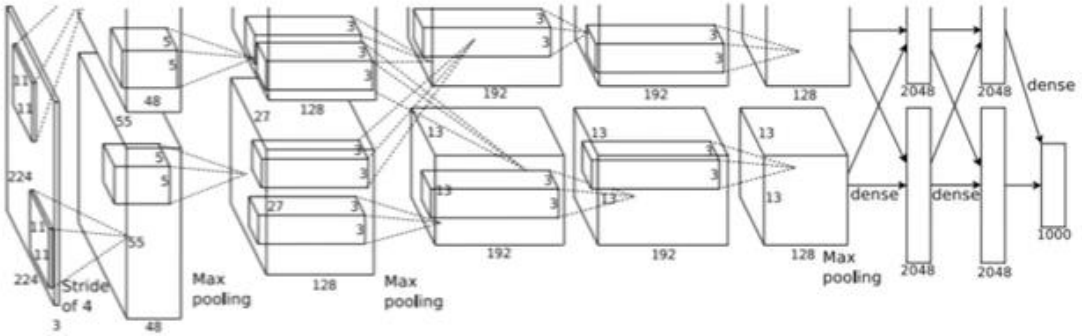
[LeCun et al., 1998]



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]

Case Study: AlexNet

[Krizhevsky et al. 2012]

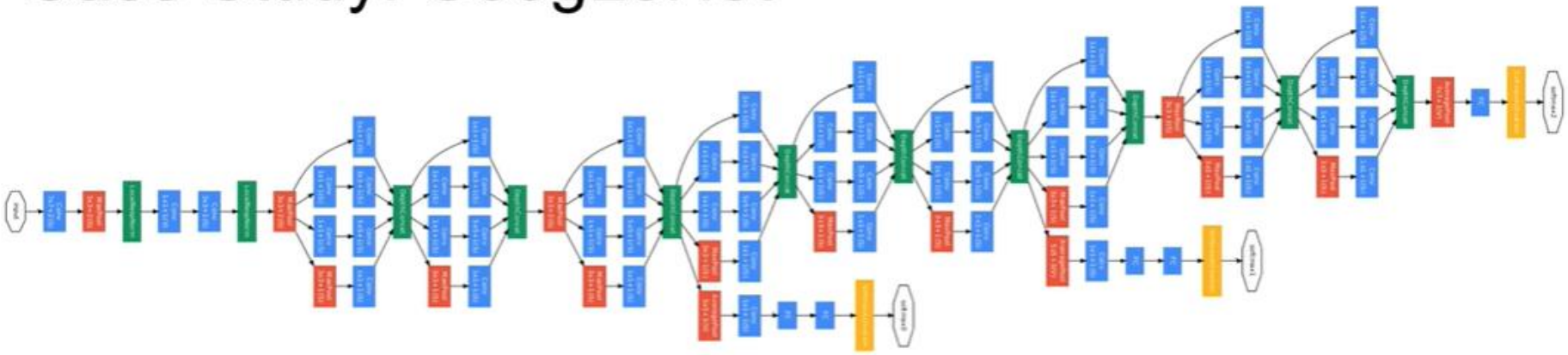


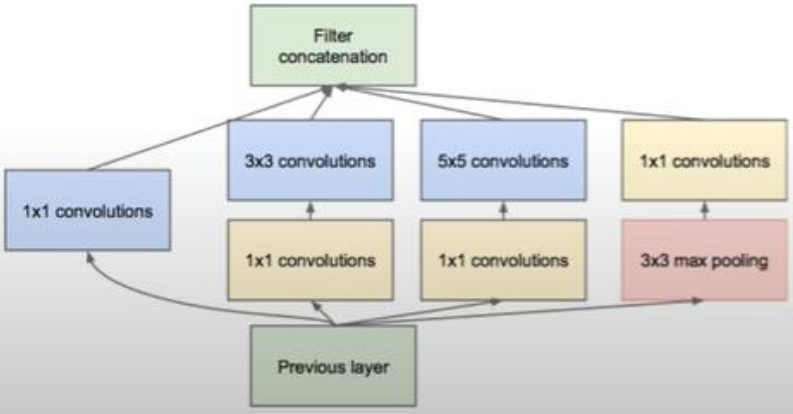
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] 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)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Case Study: GoogLeNet

[Szegedy et al., 2014]



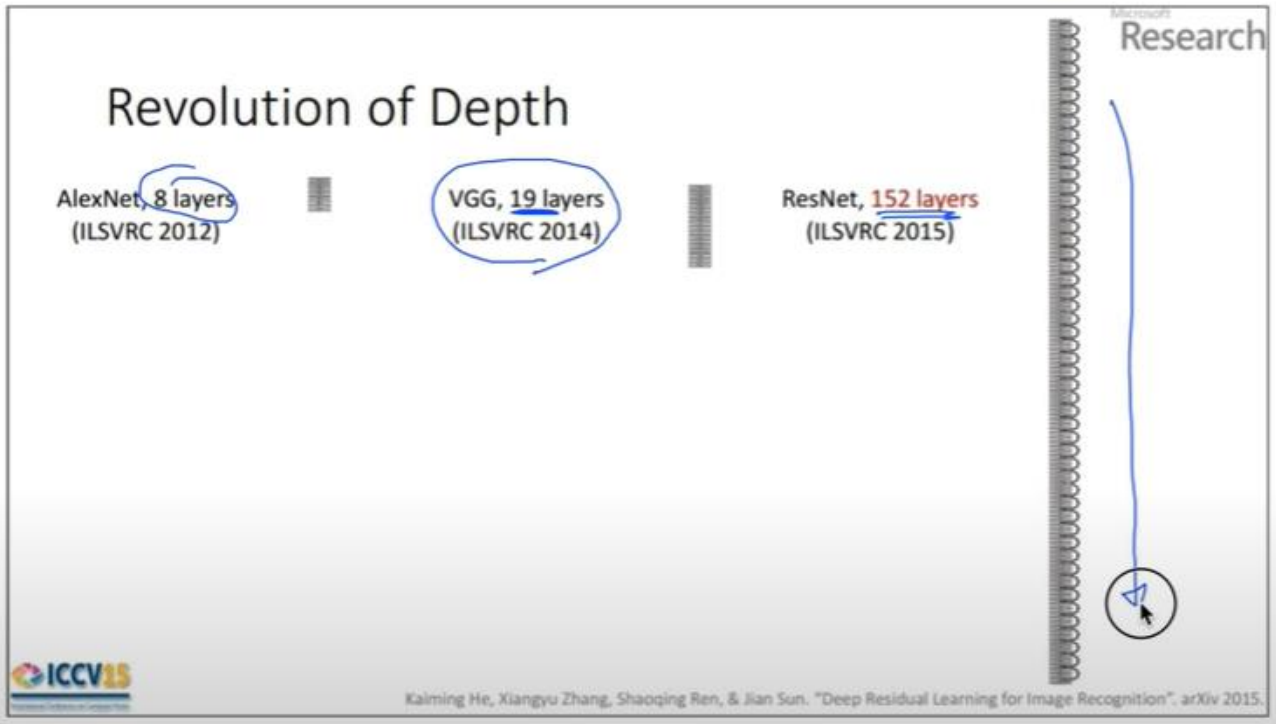


Inception module

ILSVRC 2014 winner (6.7% top 5 error)

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

(slide from Kaiming He's recent presentation)