Deep Learning Seminar

4. Convolutional Neural Network

HA SEUNG HYUN

Contents

일반적으로 NN 이라고 하면 FCN이라고 하는 경우가 많다.
fully connected layer가 모여 FCN을 구성한다.

히든 layer을 하나만 사용하면 FCN중에서도 ANN.
히든 layer 여러개를 사용하면 DNN.
이때 DNN + Convolution 연산을 추가하면 CNN.
이 모든 ANN, DNN, CNN 또한 NN이라고 한다.
즉, NN은 두가지의 중의적인 의미를 갖는다.

Reference: lecture note (Fei-Fei Li)
lecture note (Andrew Ng)
모두를 2위한 머신러닝 (Sung kim)

What is CNN ?

작은 부분을 보고 핵심적인 정보를 추출

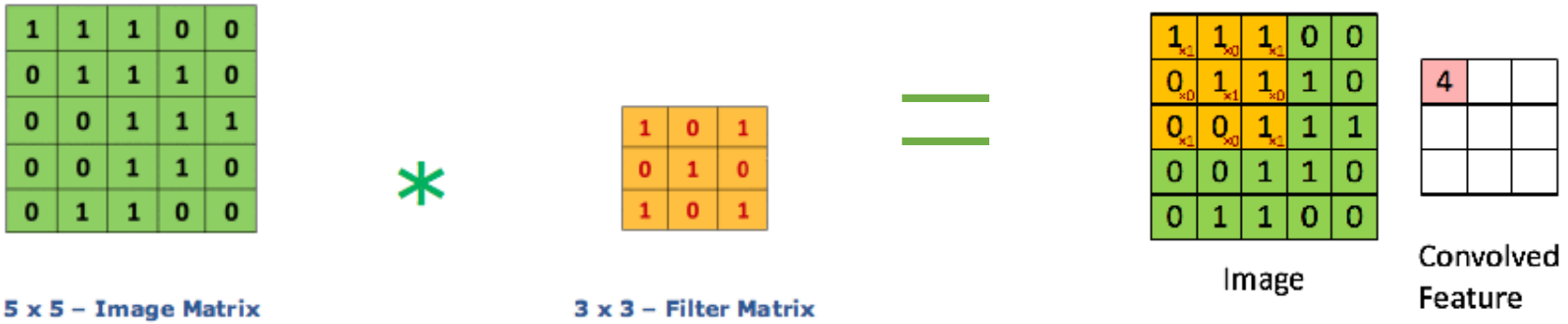
- Convolution Neural Network (CNN)

영상처리 (Computer Vision)에서 널리 사용하는 기계학습 방법론 중 하나.

이미지의 전체가 아닌 여러 개의 **작은 부분들을 보고 (by Convolution Filter)**

핵심적인 정보(Feature-map)를 추출하여 해당 정보를 이용하여 원하는 결과를 예측하는 기계학습 방법론.

filter 크기가 작기 때문에 원래 이미지에서 filter를 거치지 출력 크기가 작아짐.

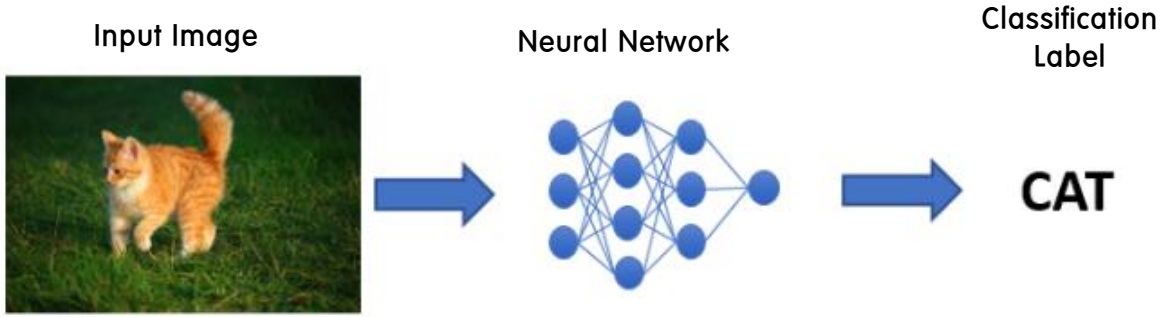


Filter안에 들어가 있는 값이 weight 값

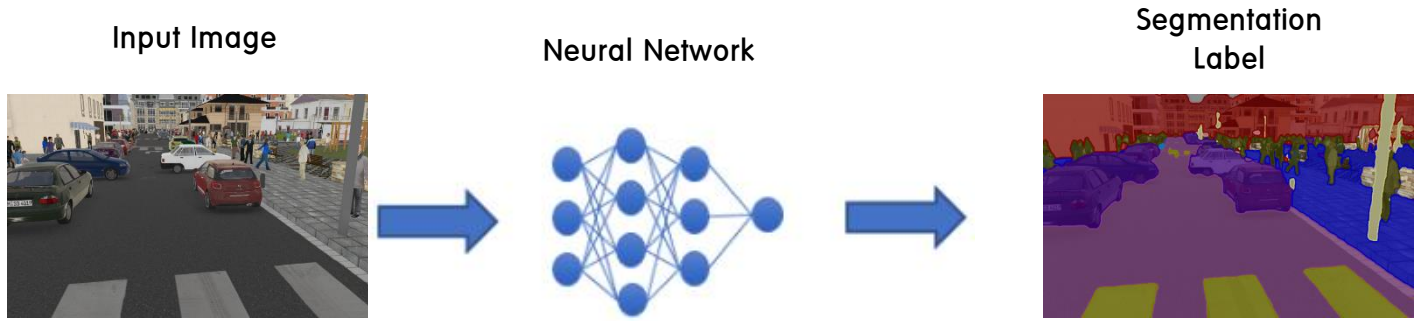
〈 Convolution Filter 〉

What can we do using CNN?

- Classification

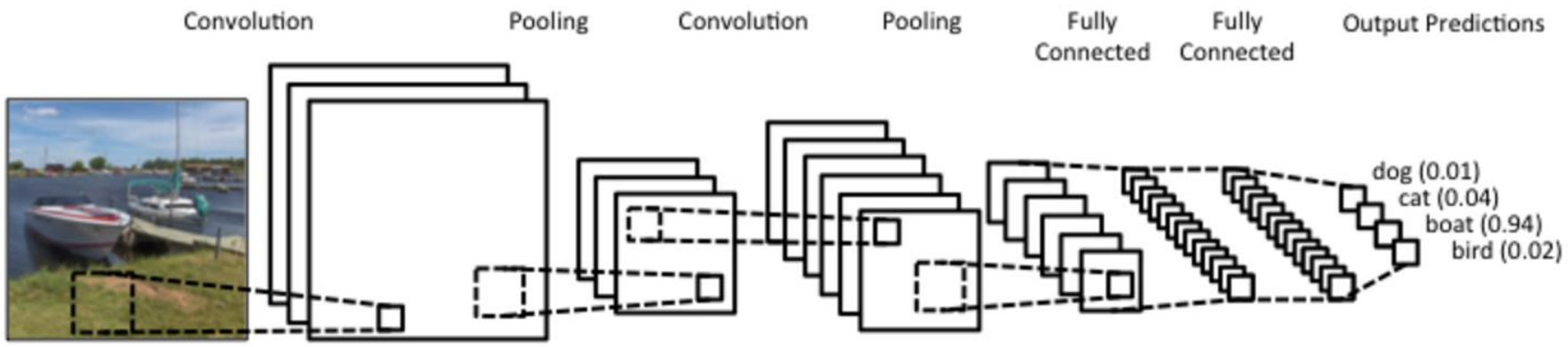


- Segmentation



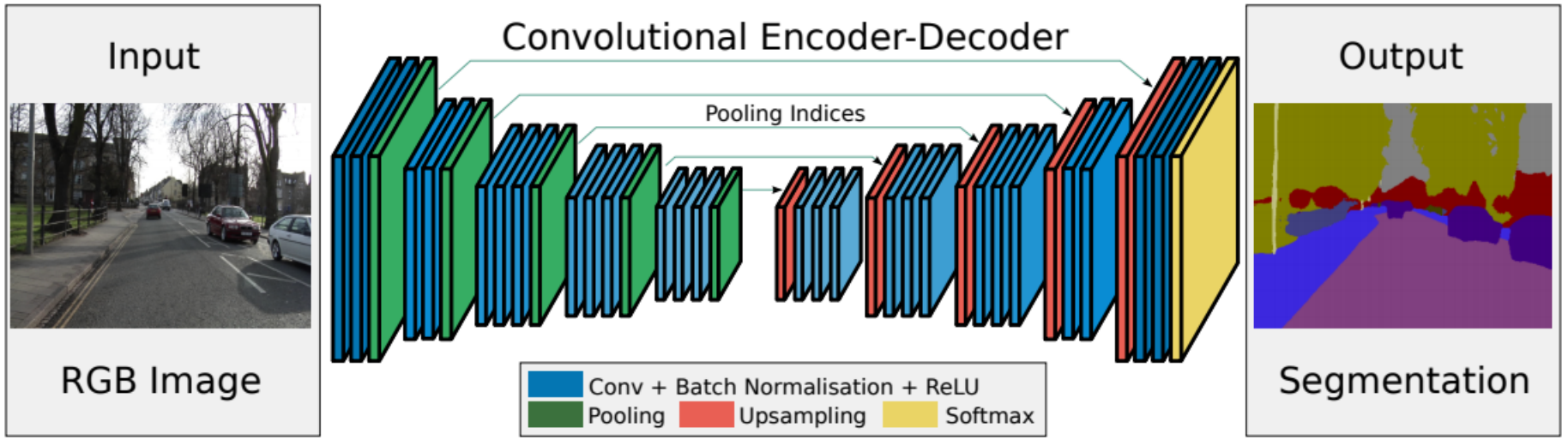
How does CNN work ?

- Classification Model



How does CNN work ?

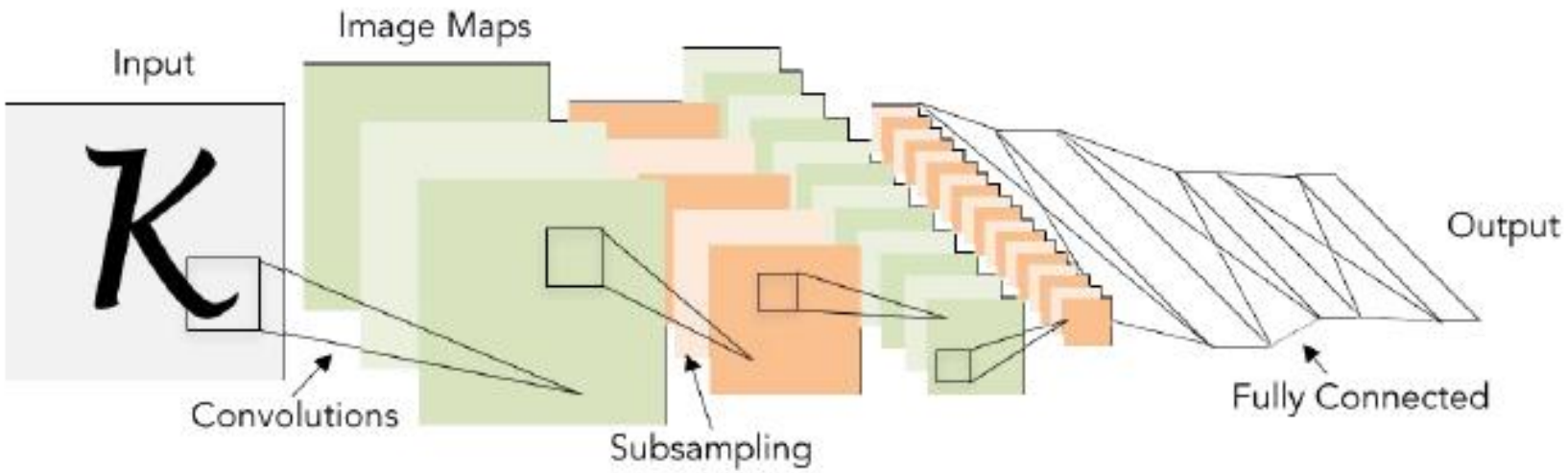
- Segmentation Model



CNN History

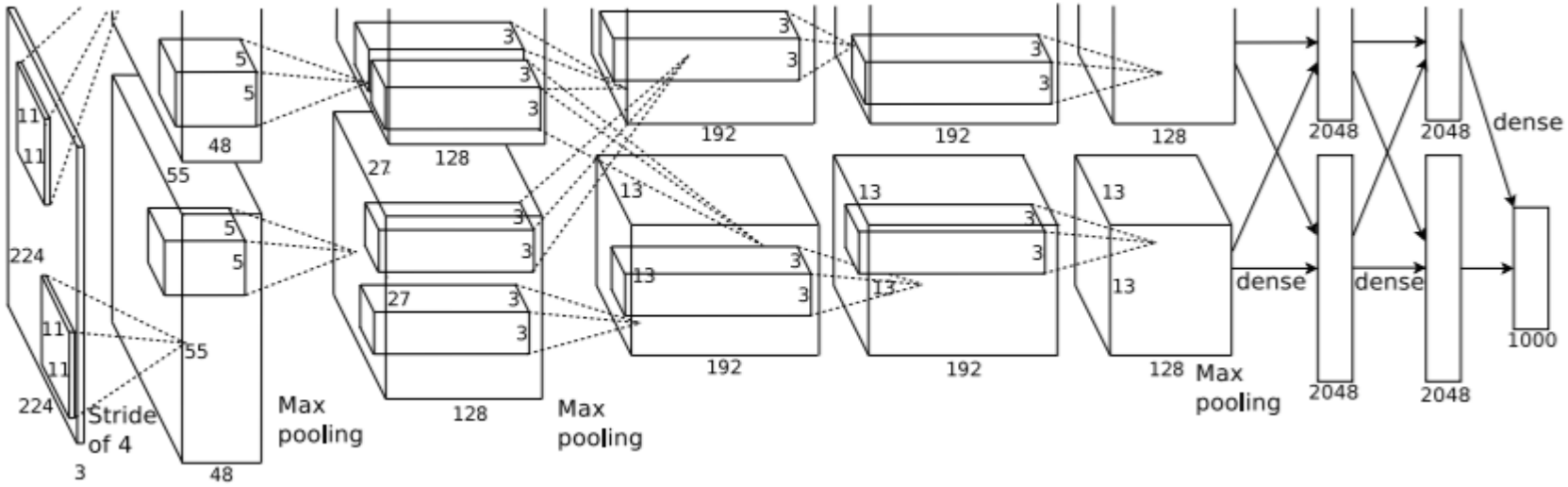
- 1) LeNet (1998) 얀르쿤 - > 제프리 힌틀

컴퓨터 사양↓ (computer power)
주목을 받지 못했음



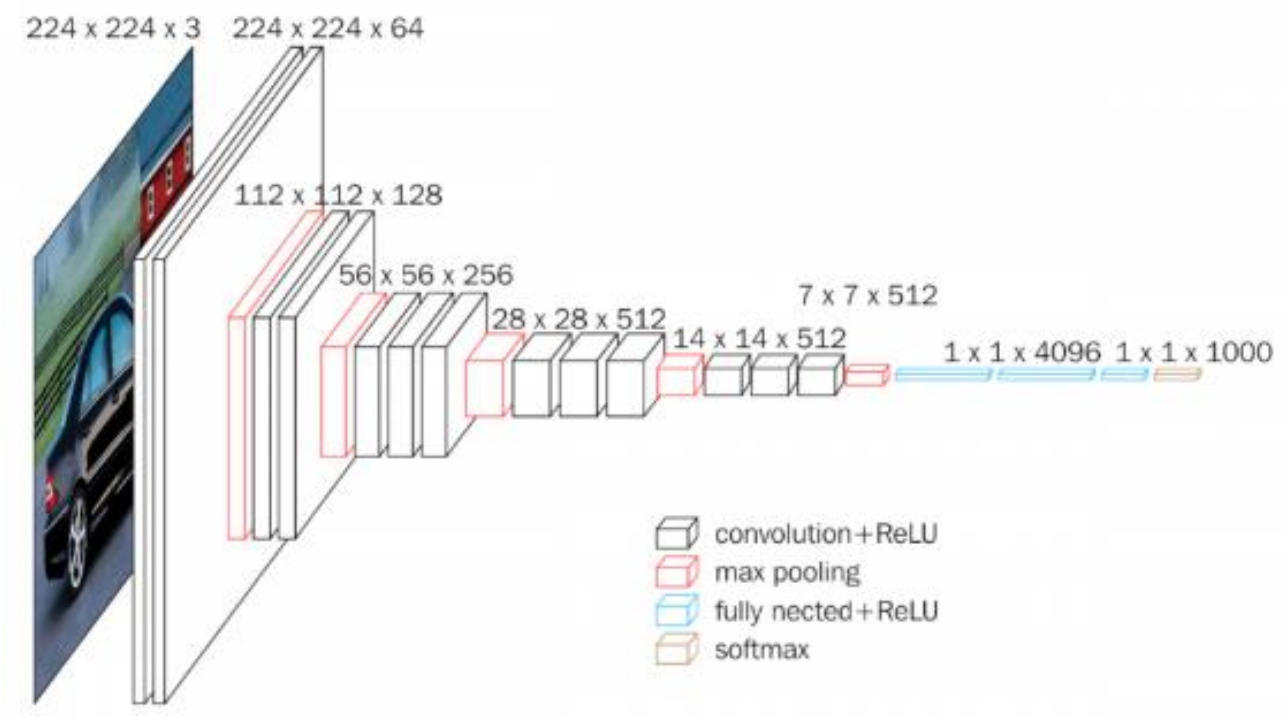
CNN History

- 2) AlexNet (2012)



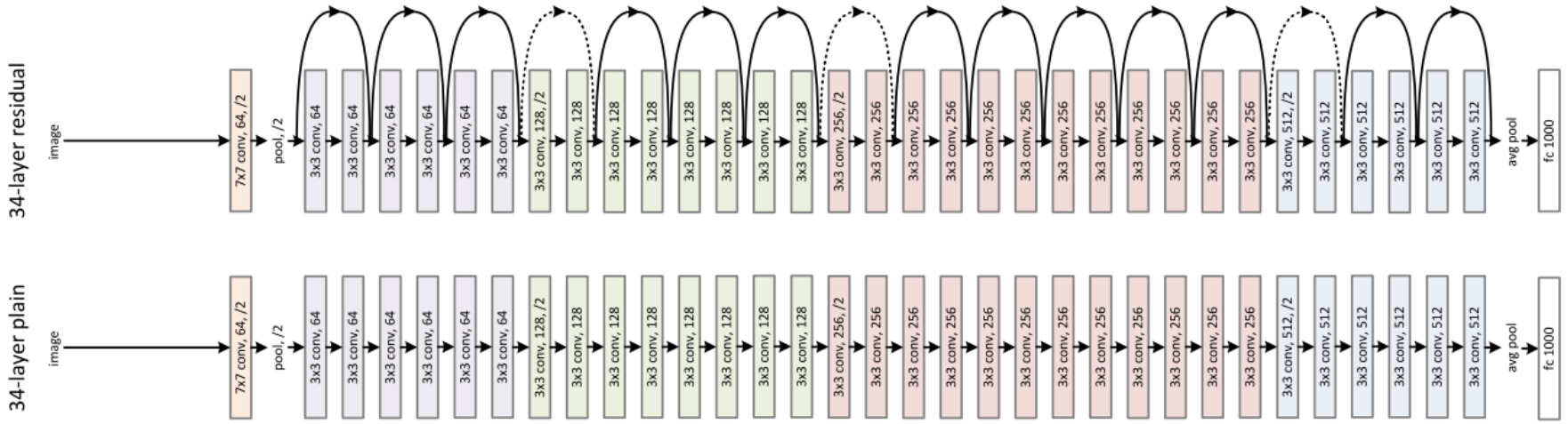
CNN History

- 3) VGG (2014)



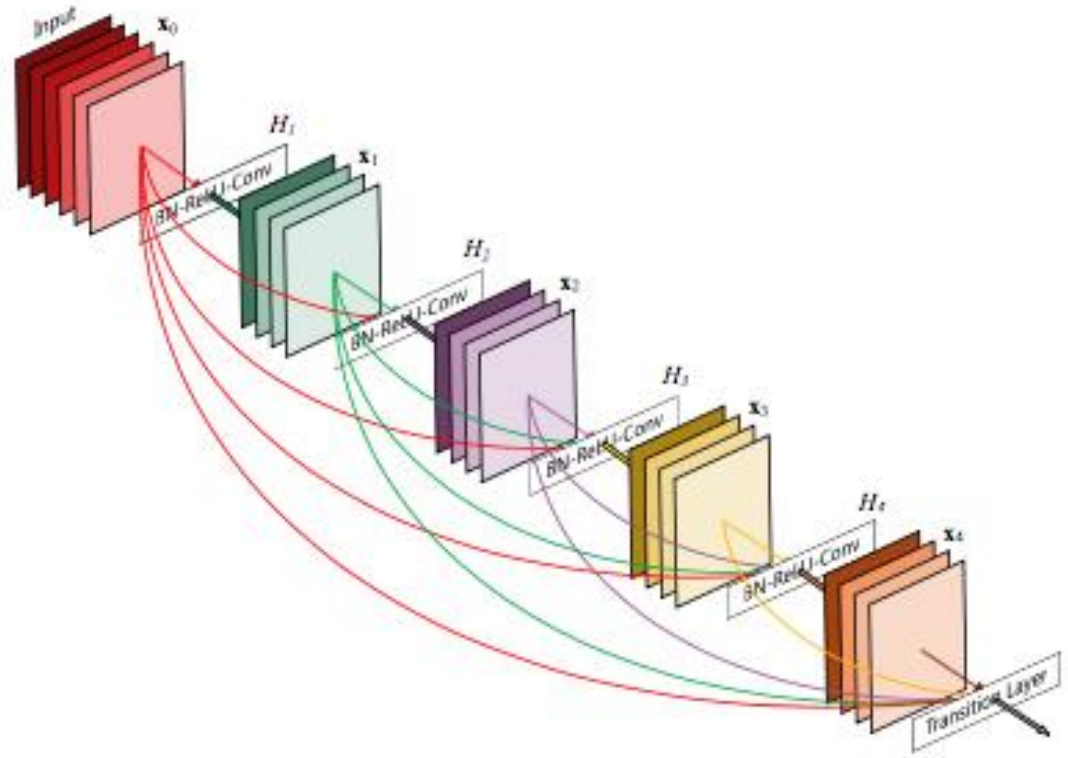
CNN History

- 4) ResNet (2015)



CNN History

- 5) DenseNet (2016)

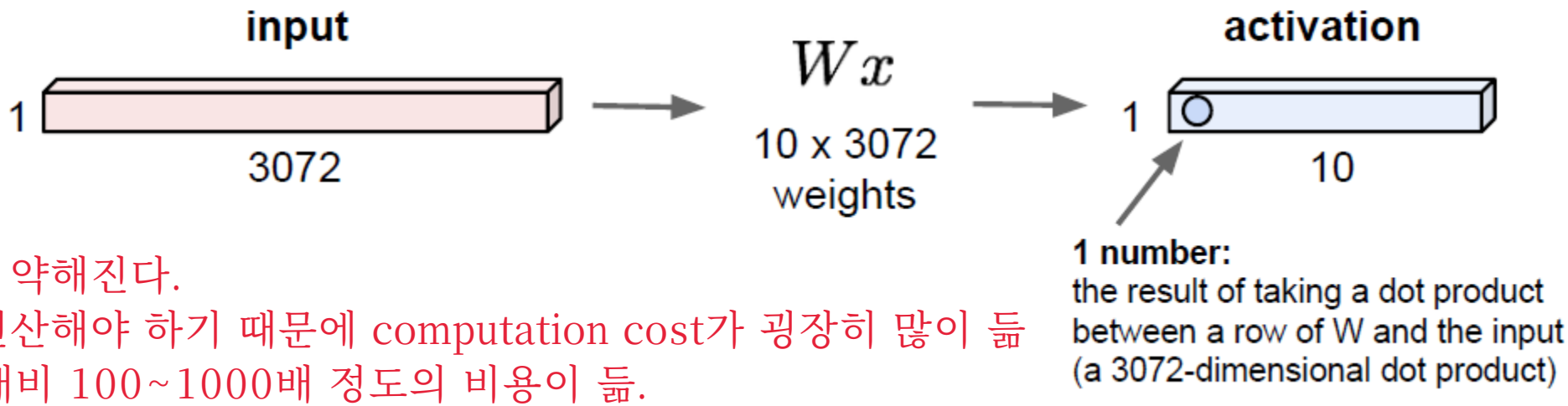


Convolutional Neural Network

- Fully Connection Layer

Fully Connected Network 단점
하나의 픽셀은 2가지 정보를 갖는다.
1) 공간 정보 (x, y 좌표)
2) 색 정보 (R, G, B)
그런데 FCN은 해당 이미지를 한 줄로 펼쳐서 넣는다.
이것은 공간 정보를 잃게 되는 것이다.
(spatial imformation - 여기서 spatial은 공간 정보를 잃어버린다는 의미)

32x32x3 image -> stretch to 3072 x 1



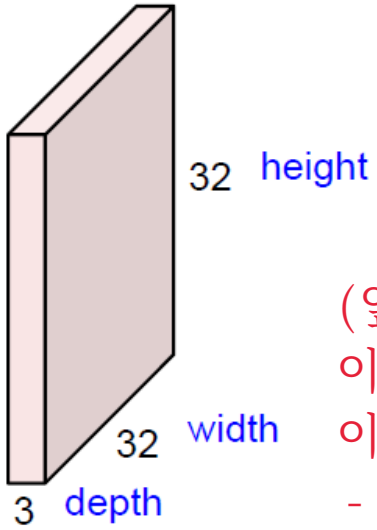
정리.
1. 공간 정보가 약해진다.
2. 한번에 다 연산해야 하기 때문에 computation cost가 굉장히 많이 듦
   - > CNN 대비 100~1000배 정도의 비용이 듦.
   - > 동일한 코드를 FCN으로 돌리면 안돌아감.
   CNN으로 돌리면 바로 돌아감

(뒷장 이어짐)

Convolutional Neural Network

- Convolution Layer

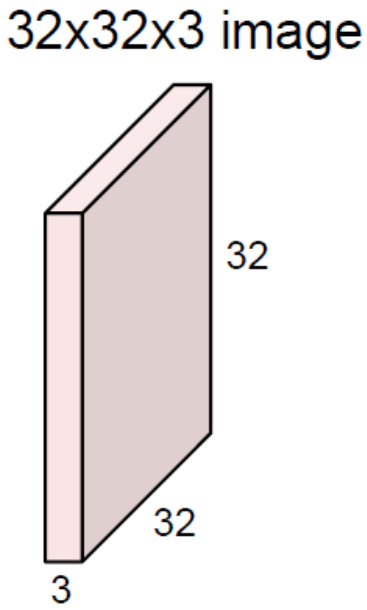
32x32x3 image -> preserve spatial structure 공간 정보를 유지할 수 있다



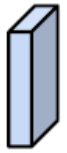
(앞부분과 이어짐)
이것은 즉, 한줄로 펼쳐서 읽으면 안된다는 결론에 도달하고
이미지를 덩어리로 바로 네트워크에 넣어야 한다.
- > 공간 정보를 살리기 위해
- > 이미지를 덩어리로 넣는 것이 가능하게 된 이유는 filter를 사용하기 때문에!!

Convolutional Neural Network

- Convolution Layer



5x5x3 filter

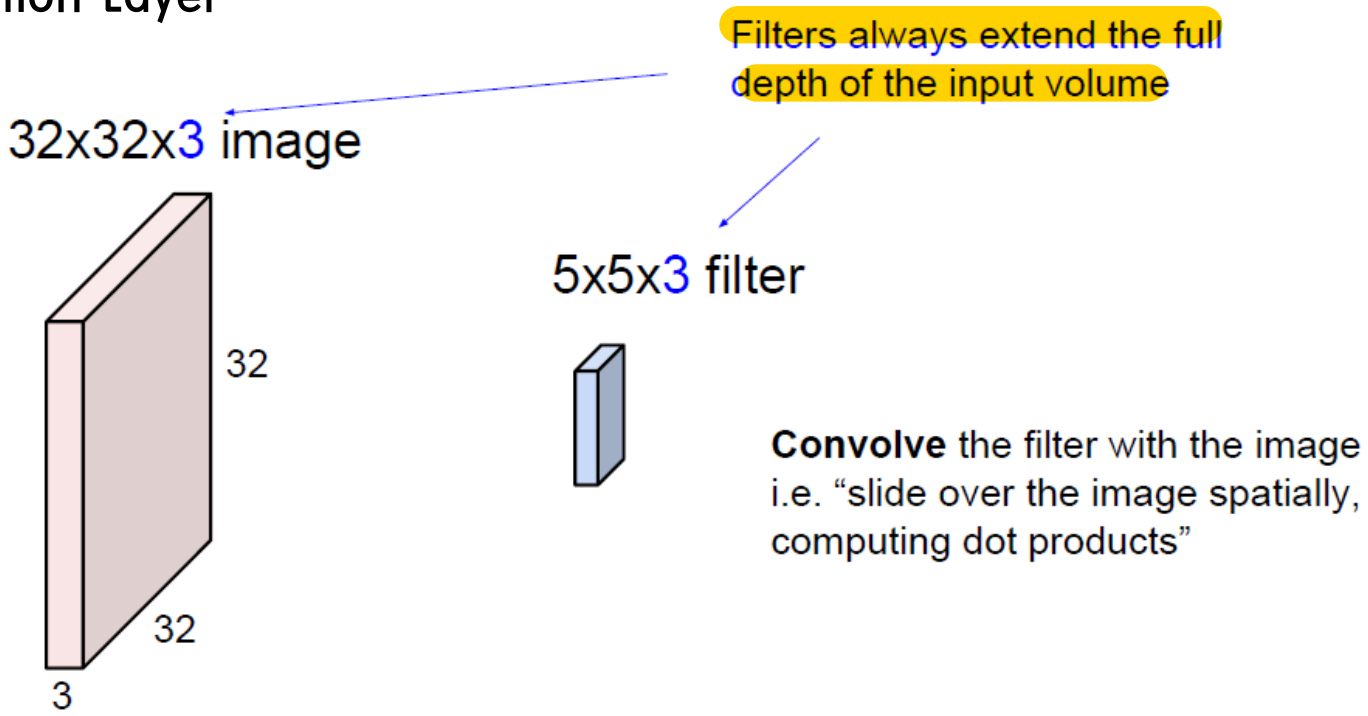


3x3 주로 사용
1x1 특별한 경우에
5x5 논문에 많이 등장

Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

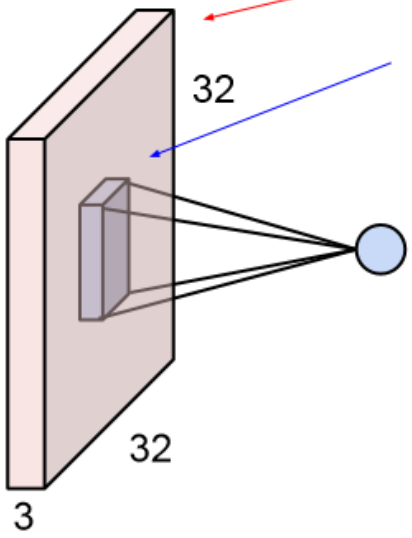
Convolutional Neural Network

- Convolution Layer



Convolutional Neural Network

- Convolution Layer



32x32x3 image

5x5x3 filter w

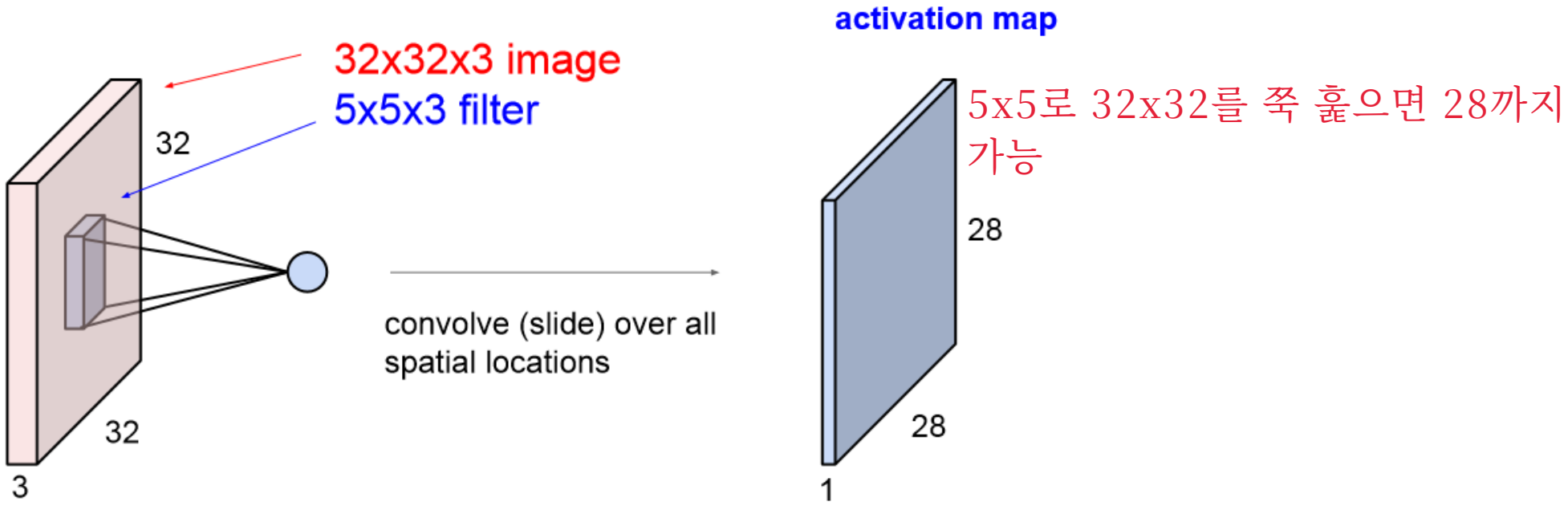
filter가 image를 훑어보기 위해서는
채널이 똑같아야 한다.

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolutional Neural Network

- Convolution Layer

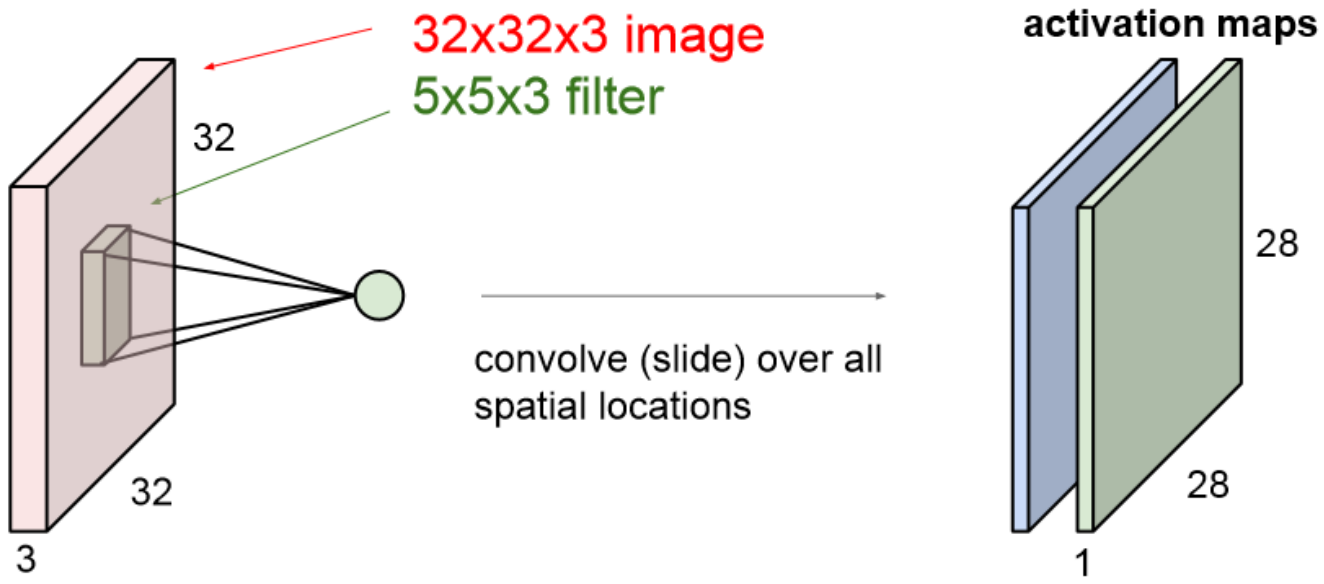


5x5로 32x32를 쭉 훑으면 28까지 가능

image와 filter의 채널이 같이 때문에 한번에 훑어서 1이 나옴.

Convolutional Neural Network

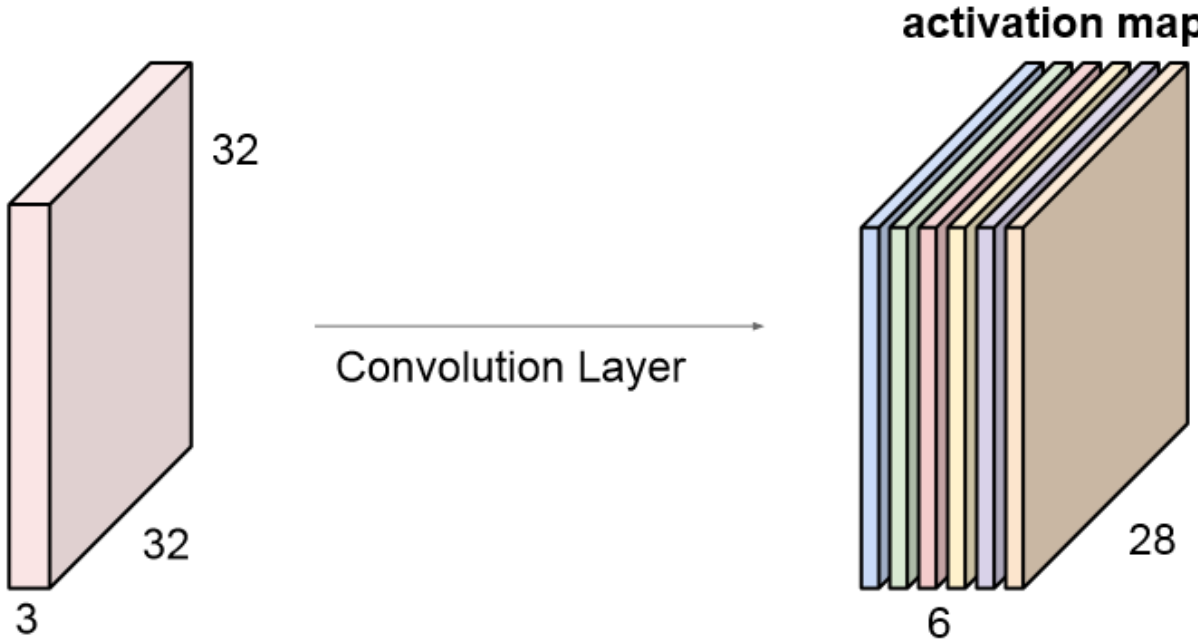
- Convolution Layer



Convolutional Neural Network

여러가지 특징으로 훑는 경우
이러한 결과가 나옴.

- Convolution Layer



중간 출력 값
Feature Map

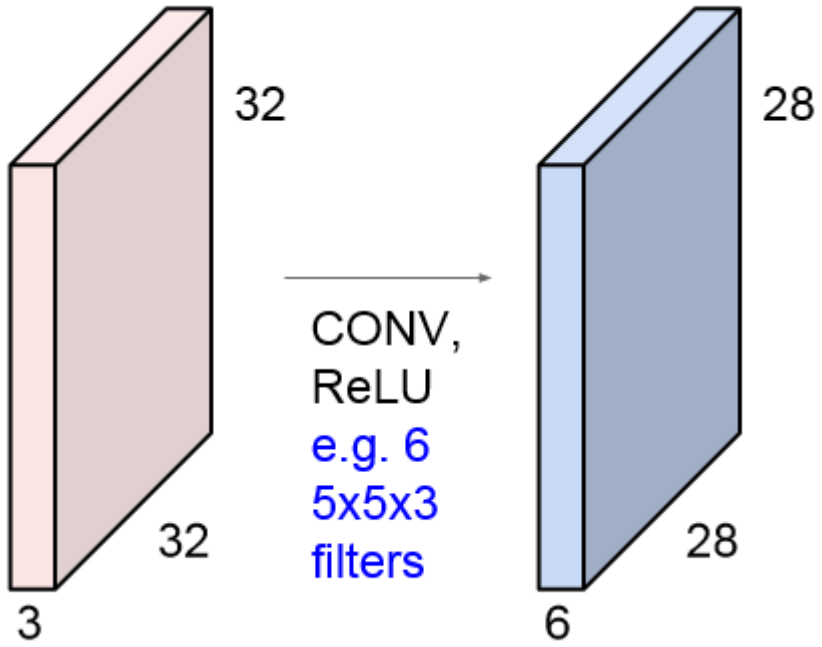
feature map의 채널이 6이라는 것은
6개의 filter가 사용되었다는 것.

We stack these up to get a “new image” of size 28x28x6!

중간 출력 값으로 알 수 있는 정보 : filter는 5x5이고 입력 이미지의 채널이 3이기 때문에 filter의 채널도 3이다.
또한 중간 출력 값의 채널이 6이라는 것은 filter의 갯수가 6개 하는 것!

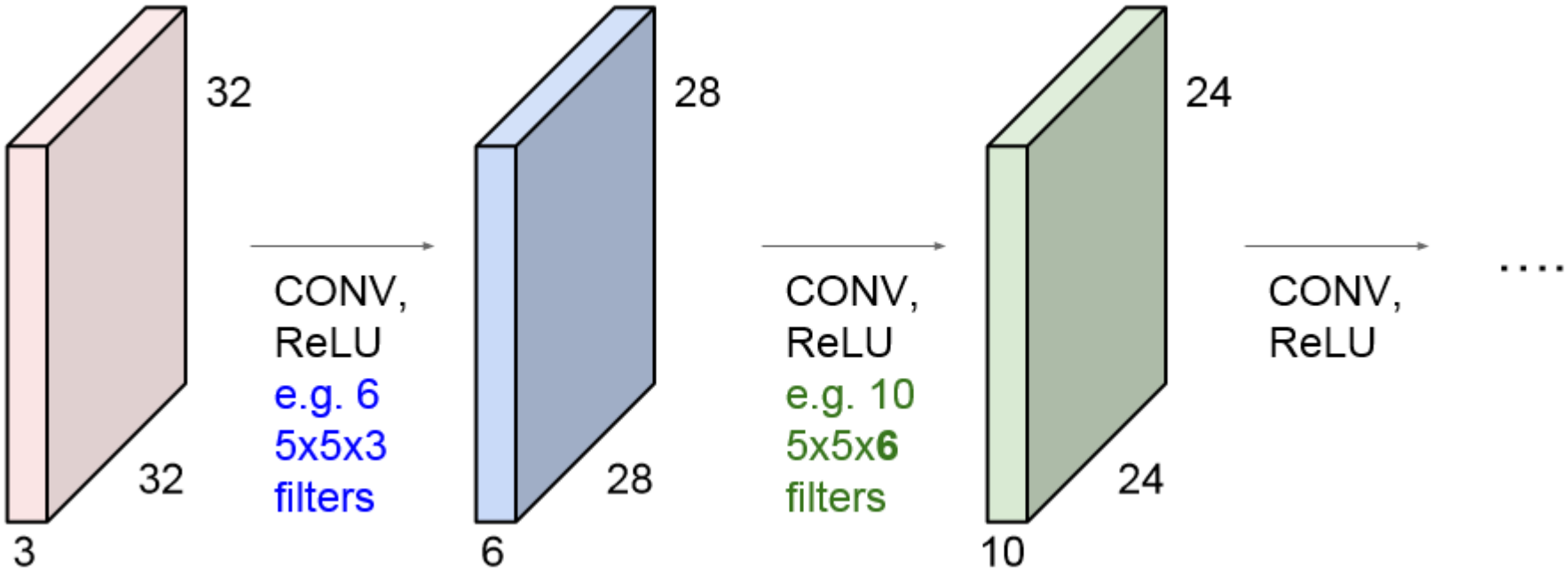
Convolutional Neural Network

- Convolution Layer



Convolutional Neural Network

- Convolution Layer

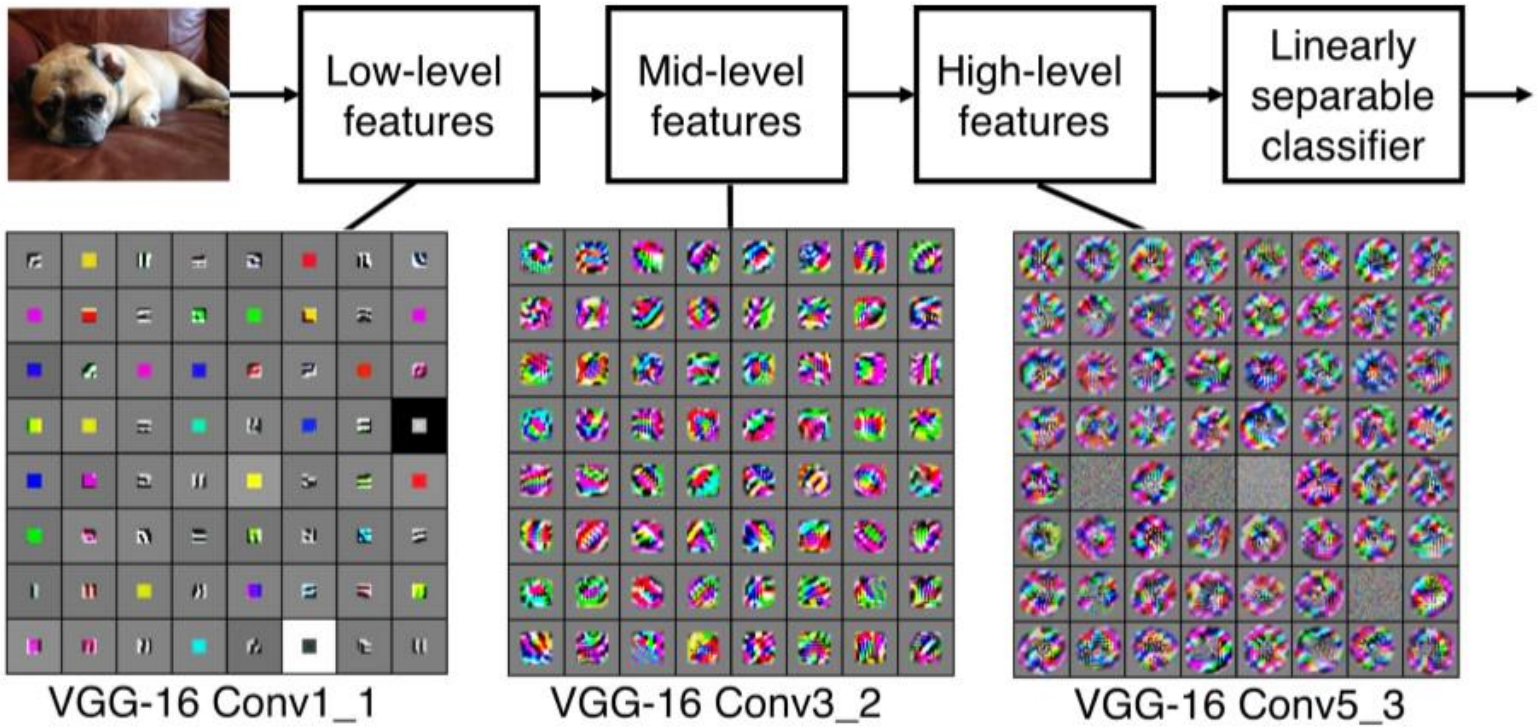


5x5x3 filter 6개 사용

5x5x6 짜리 filter 10개 사용

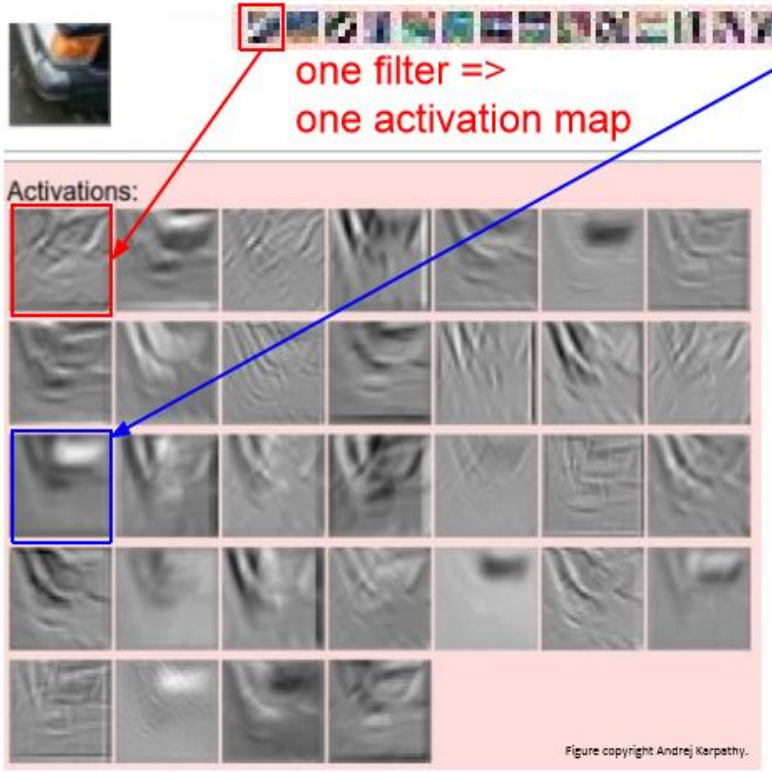
Convolutional Neural Network

- Feature-maps (Layers)



Convolutional Neural Network

- Feature-maps (Filters)



example 5x5 filters
(32 total)

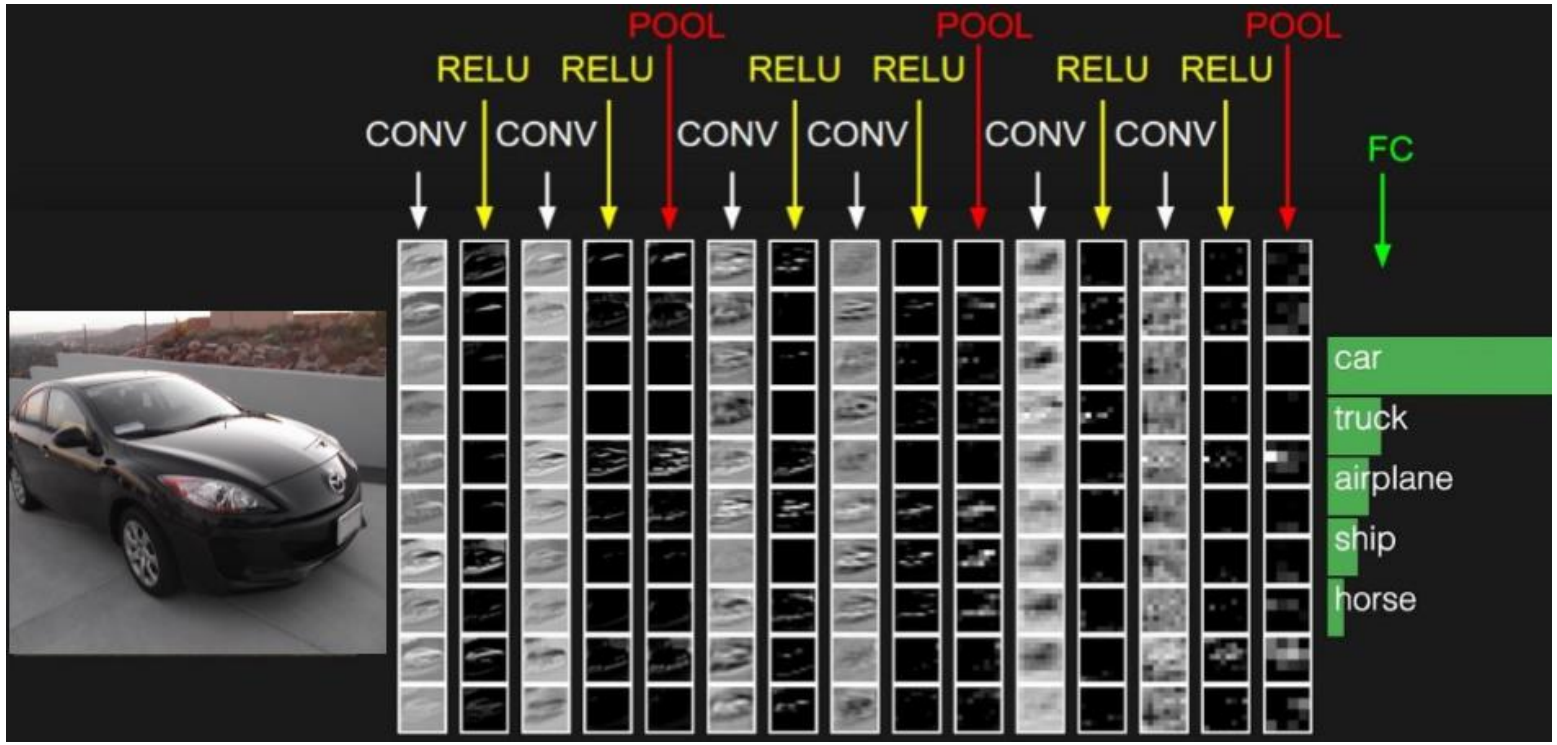
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

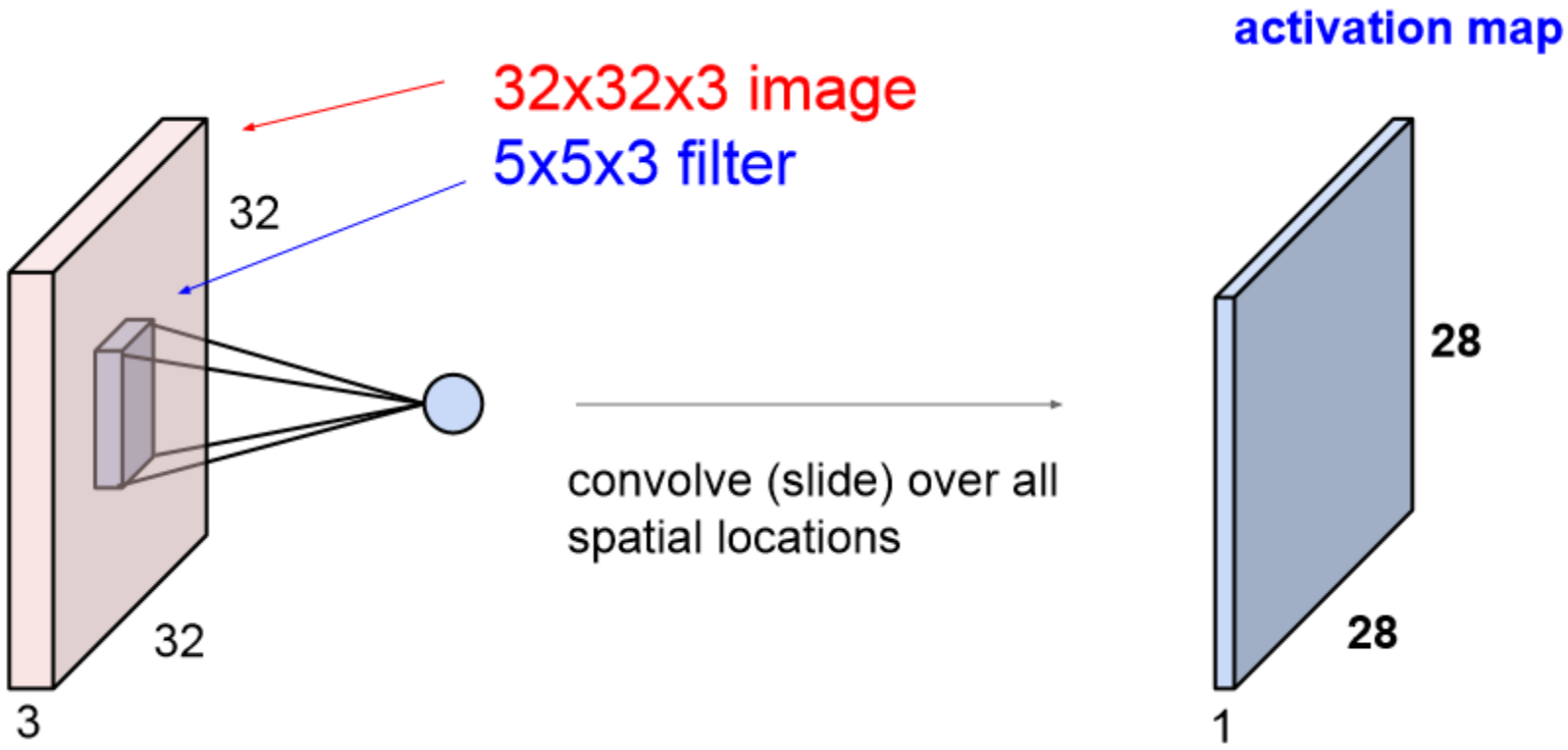
Convolutional Neural Network

- Feature-maps (All)



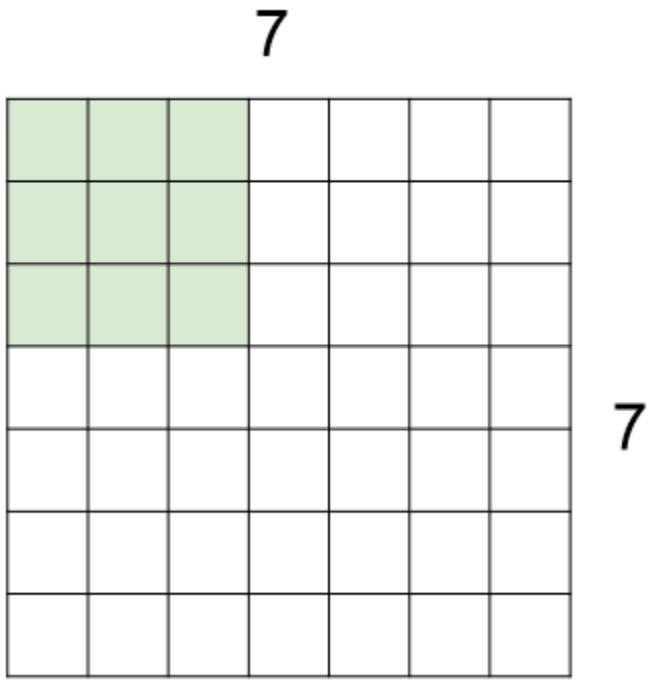
Convolutional Neural Network

- Convolution Layer



Convolutional Neural Network

- Convolution Layer



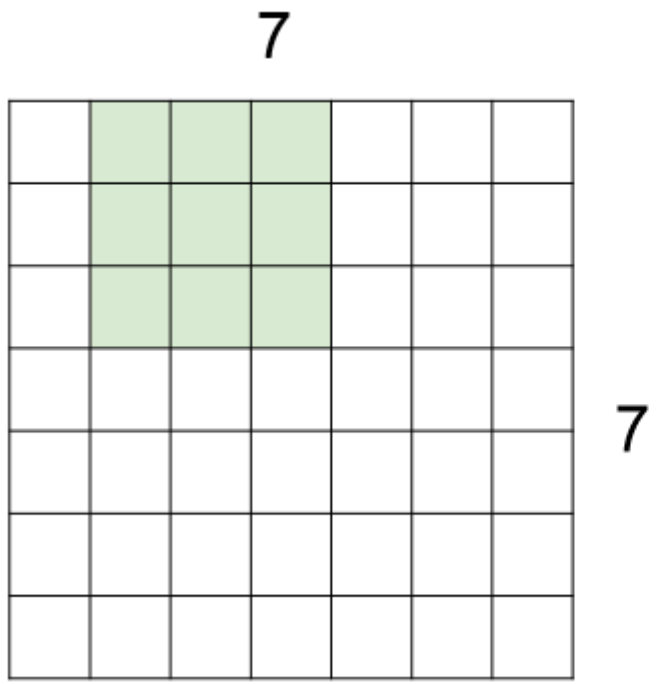
7x7 input (spatially)
assume 3x3 filter

STRIDE = 1

STRIDE : filter가 image에서 몇칸씩 움직이는지

Convolutional Neural Network

- Convolution Layer

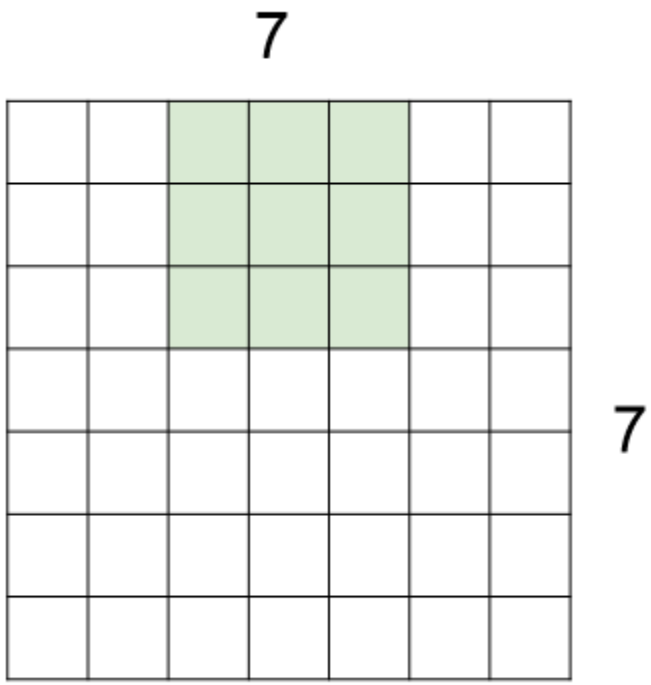


7x7 input (spatially)
assume 3x3 filter

STRIDE = 1

Convolutional Neural Network

- Convolution Layer

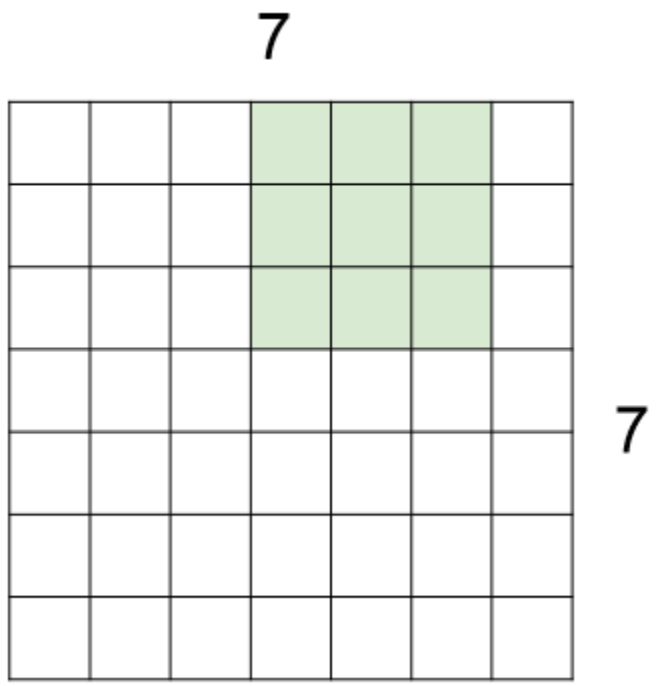


7x7 input (spatially)
assume 3x3 filter

STRIDE = 1

Convolutional Neural Network

- Convolution Layer

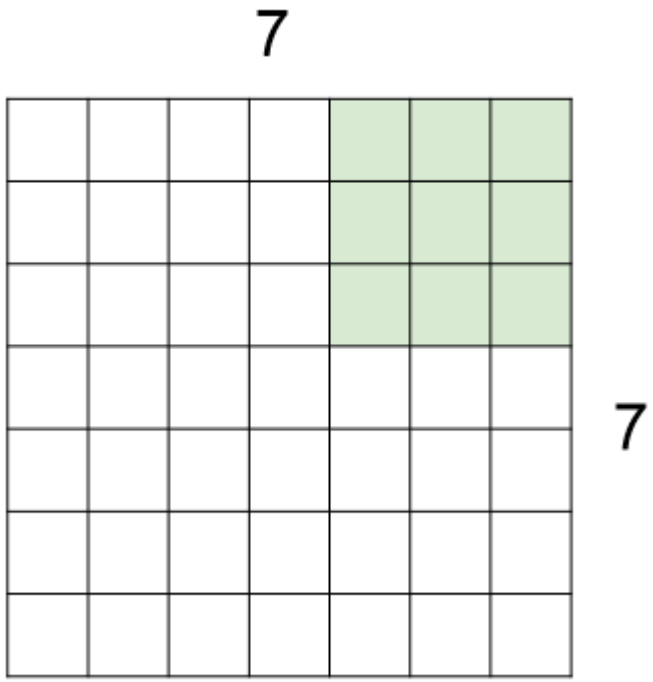


7x7 input (spatially)
assume 3x3 filter

STRIDE = 1

Convolutional Neural Network

- Convolution Layer



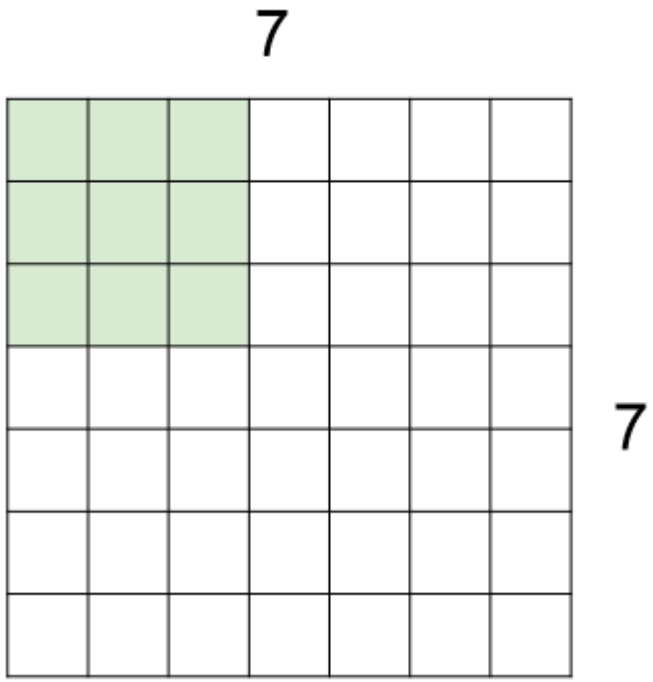
7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

STRIDE = 1

Convolutional Neural Network

- Convolution Layer

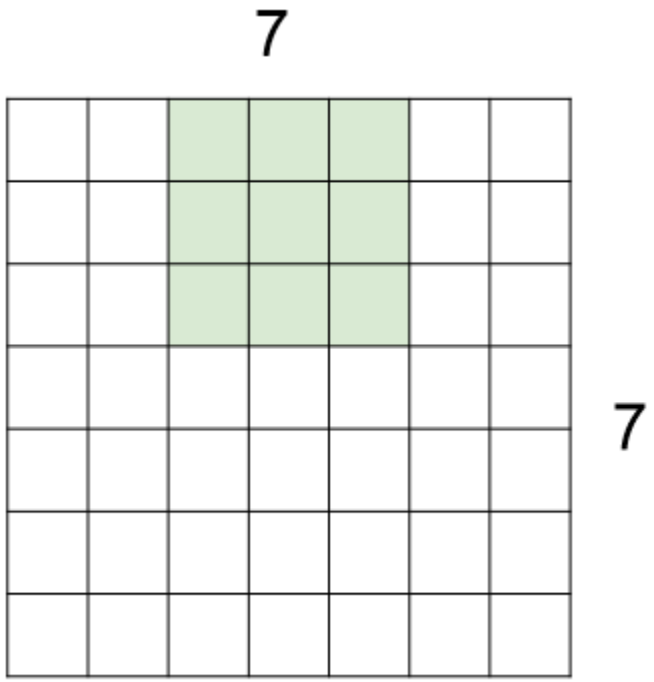


STRIDE = 2

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolutional Neural Network

- Convolution Layer

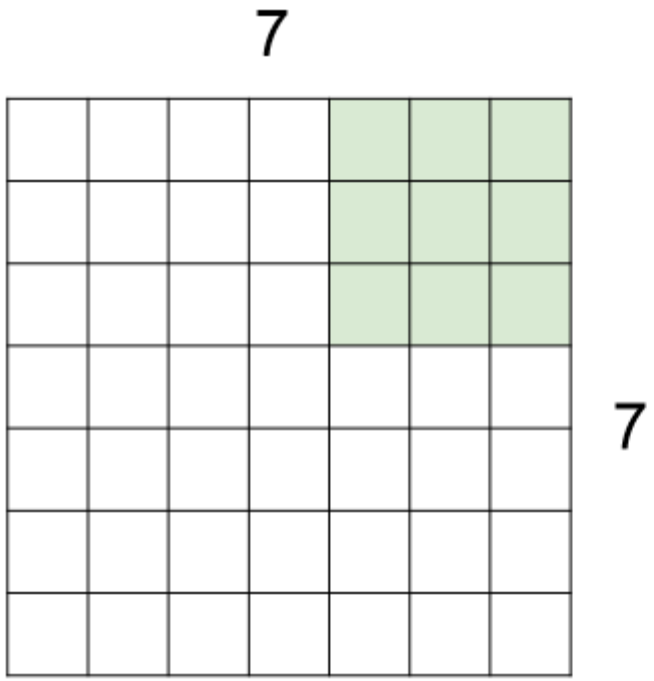


STRIDE = 2

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolutional Neural Network

- Convolution Layer

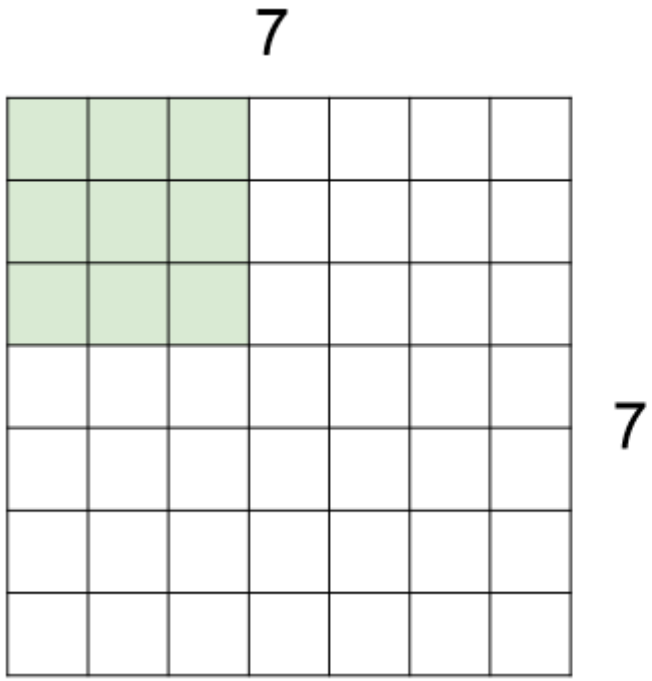


STRIDE = 2

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolutional Neural Network

- Convolution Layer



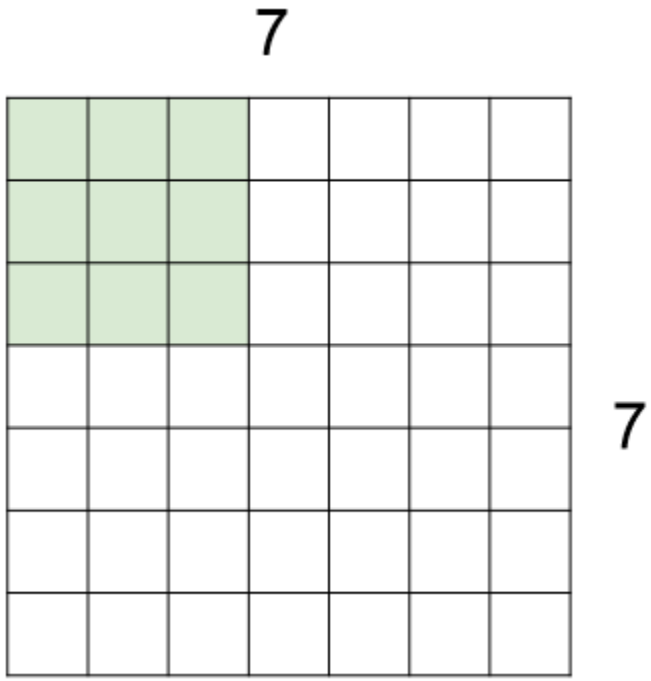
STRIDE = 3

7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Convolutional Neural Network

- Convolution Layer

stride도 잘 지정해야하고 filter size도 아무렇게 지정하면 안된다.



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

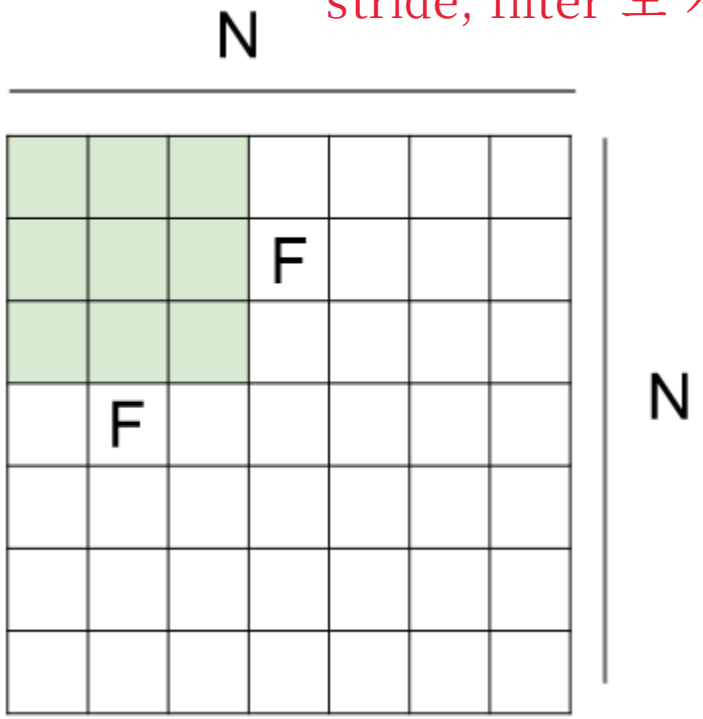
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

stride는 보통 2의 배수로 지정
image size는 최대한 2의 배수로 맞추려고 함.

Convolutional Neural Network

왜 이런 공식이 나왔을까?
입력과 출력 사이즈가 다르게 되면 계속 신경쓰인다...
원본에 비해 출력 사이즈는 filter 때문에 작아지기 때문에
stride, filter 도 계속 생각해야함. 매번 계산하기 불편;;

- Convolution Layer



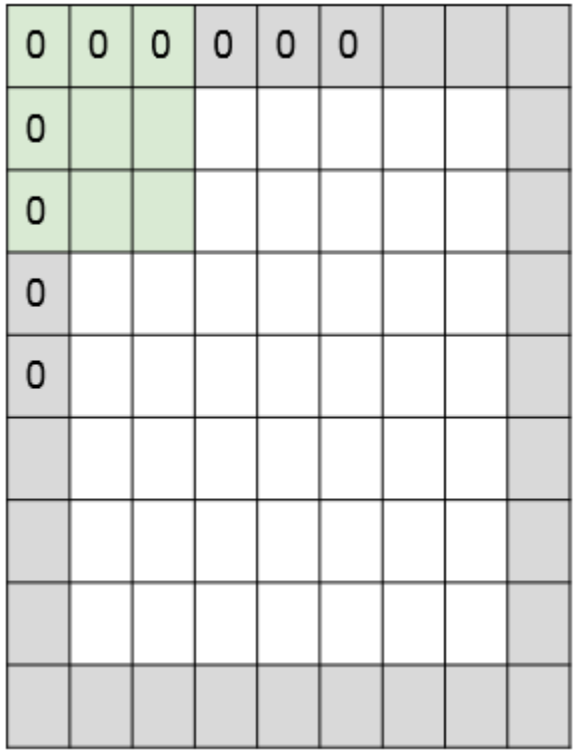
Output size:
(N - F) / stride + 1

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 :\

이를 신경 안쓰고 출력 사이즈도 똑같이 나오고 싶다!! 같은 filter를 계속 사용하고 싶다!!
실제 출력값의 빈 부분에 의미 없는 값을 넣는다(padding기법)

Convolutional Neural Network

- Convolution Layer



(N + padding - F) / stride + 1 = N
filter size에 의해 padding 값이 정해진다.
stride 1 : size 변화가 없을때
stride 2 : 정확하게 사이즈가 절반

e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

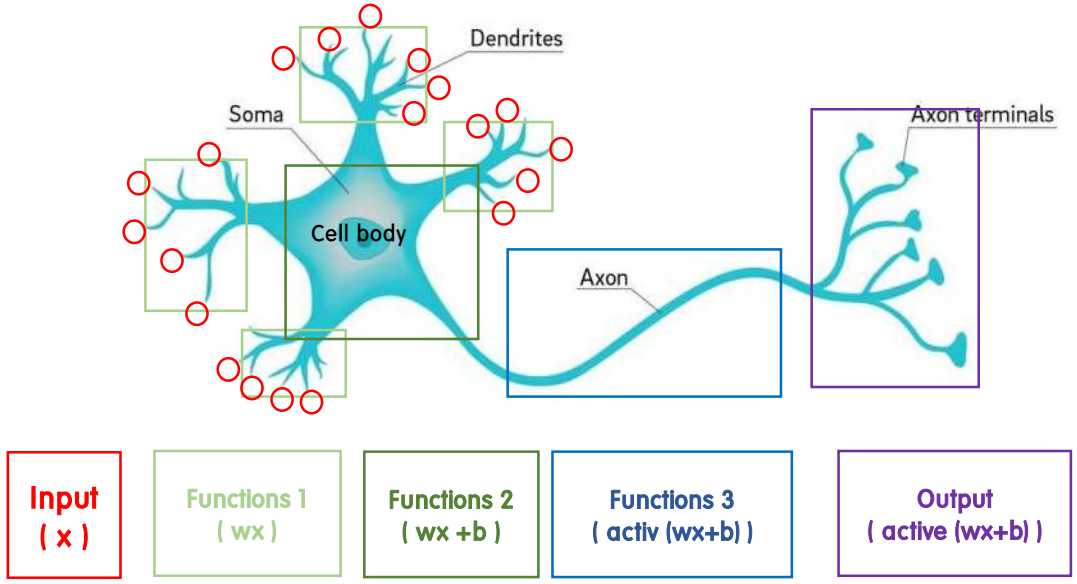
암기!!

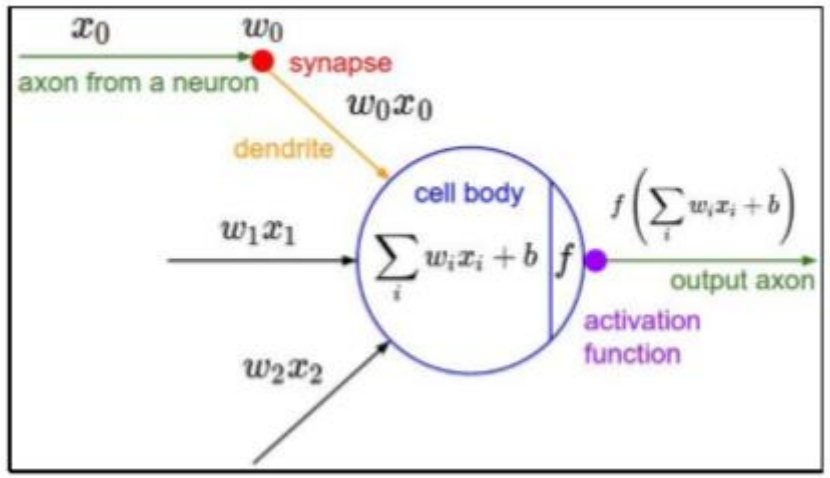
1x1 필터의 경우 padding은 0!!

Output size:
(N - F) / stride + 1

Convolutional Neural Network

- Structure of a neuron

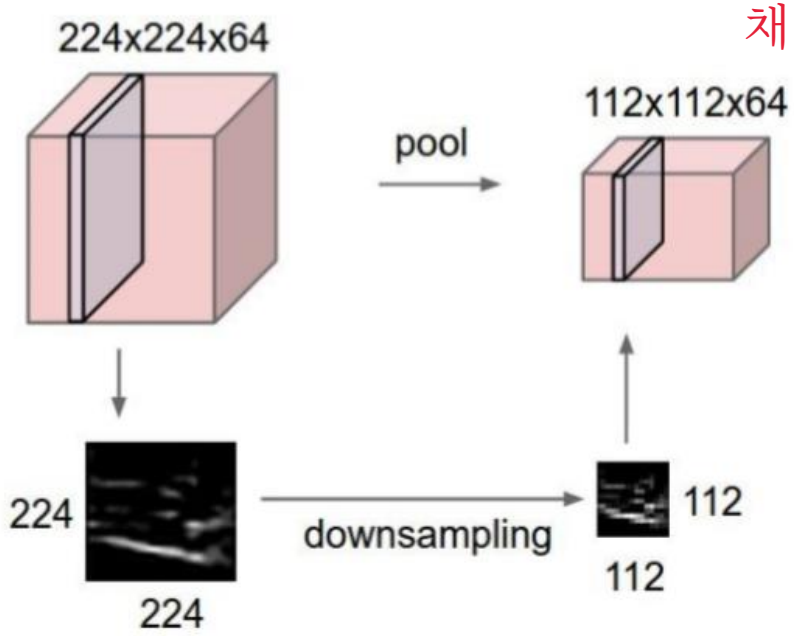




Convolutional Neural Network

- Pooling Layer

  - makes the representations smaller and more manageable
  - operates over each activation map independently:



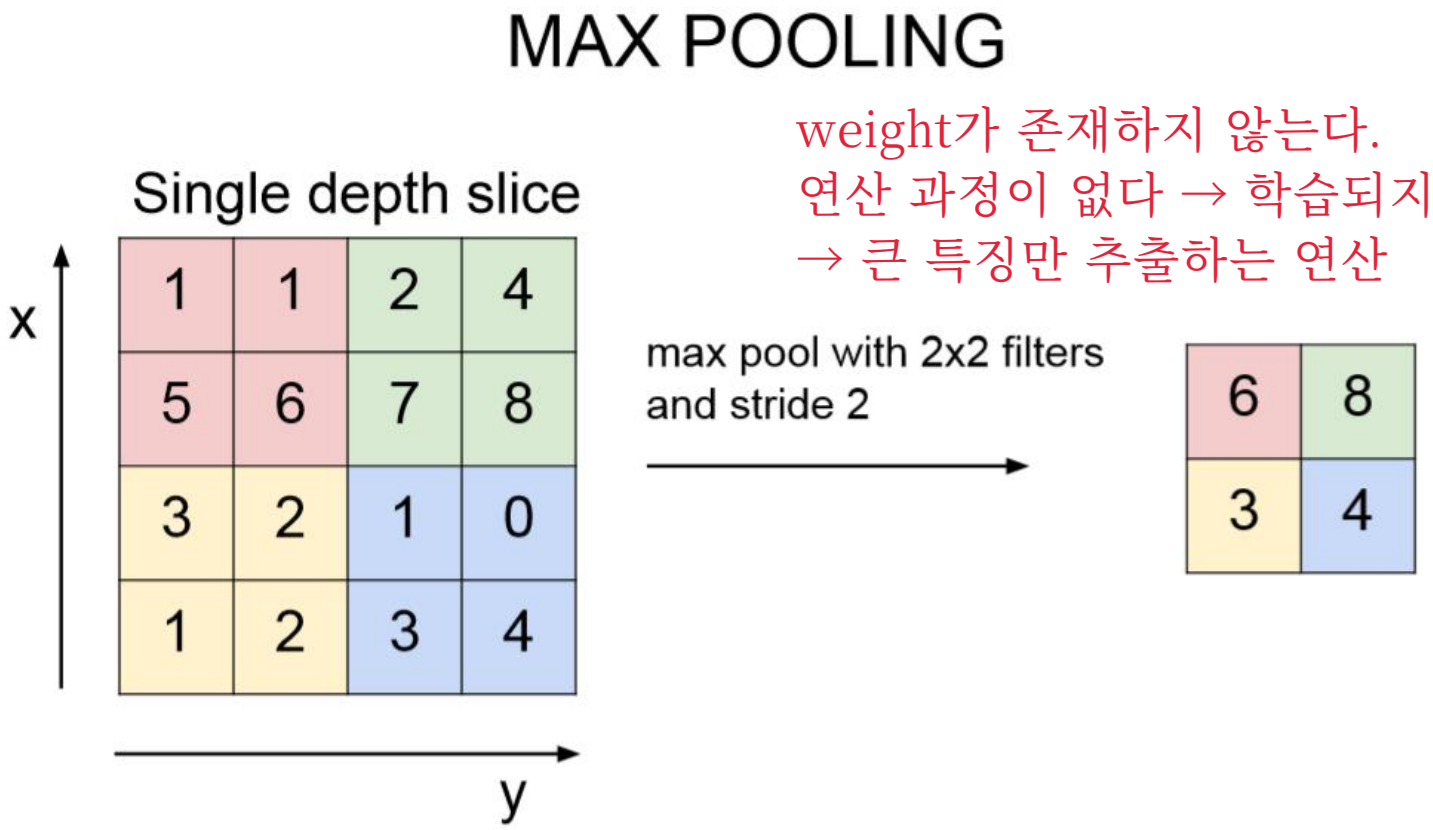
"Max Pooling은 사이즈는 절반으로 줄어들고 채널 수는 변하지 않는다."

사실 max pooling의 채널은 변하지 않기 때문에 채널이 없다고 보면 됨.

Convolutional Neural Network

- Pooling Layer

학습되지 않은 값이기 때문에 최근에는 Max Pooling은 사용하지 않음.
차라리 사이즈 반으로 줄이고 학습까지 진행한 stride 2를 사용함.



weight가 존재하지 않는다.
연산 과정이 없다 → 학습되지 않는다.
→ 큰 특징만 추출하는 연산

Convolutional Neural Network

CNN 연산 2가지
1. Convolution
2. Pooling

- Pooling Layer

  - Max Pooling

  픽셀 값 중에서 제일 큰 값 추출. 큰 값은 중요한 요소를 의미.
  → 중요한 특징들만 추출

  - Average Pooling

  전체적인 형상, 윤곽을 추출할 때 사용

  - ~~Min Pooling~~

Average Pooling은 평균 값으로 추출되는 것이고 나머지 특징은 max pooling과 동일.

Convolutional Neural Network

- Summary

[(Conv-ReLU)*N - Pool] * M - (FC-ReLU) * K - Softmax

이미지 특징 추출 | classification | 예측값

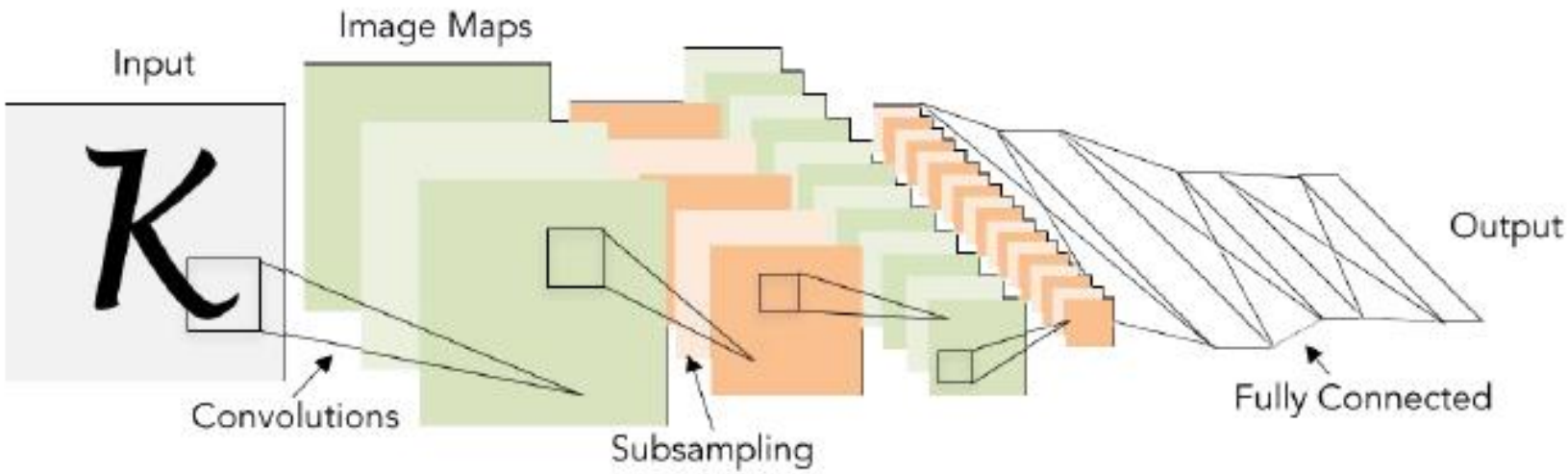
Convolutional Neural Network

- Summary

[(Conv-ReLU)*2 - Pool] * 3 - (FC-ReLU) * 2 - Softmax

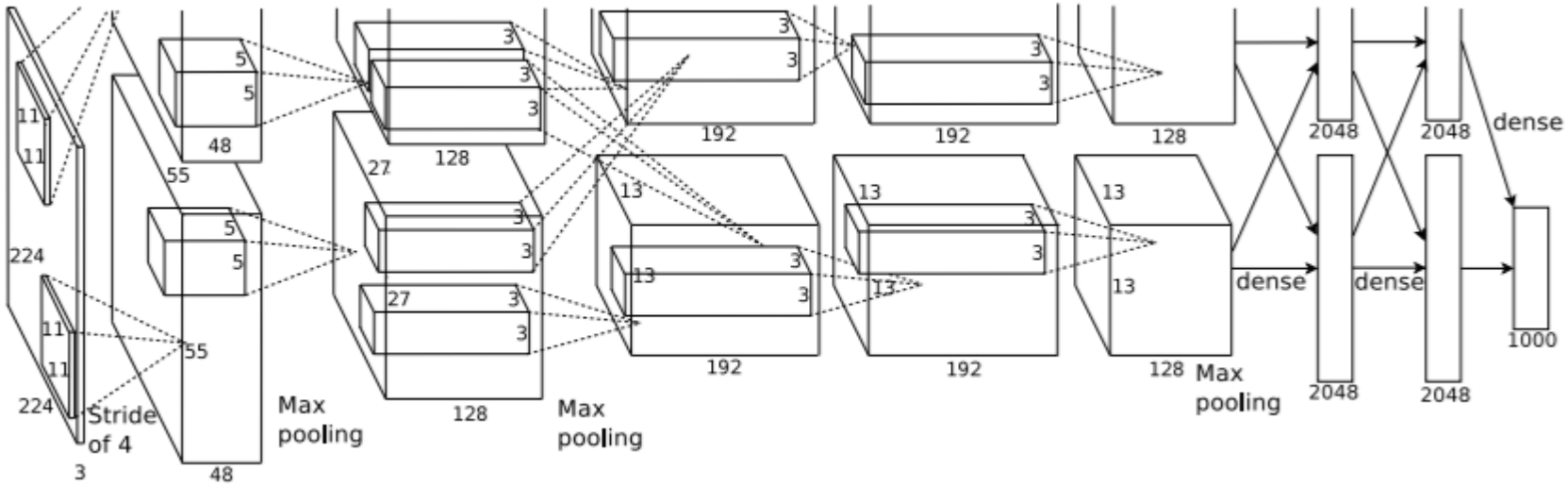
CNN History

- 1) LeNet (1998)



CNN History

- 2) AlexNet (2012)



CNN History

"필터의 갯수는 다음 출력되는 Feature Map의 채널 수와 동일하다"

- 3) VGG (2014)

1. filter 개수 : 64
2. 3x3 의 채널 : 3
3. stride : 1 - > size 변화가 없기 때문에
4. padding : 1

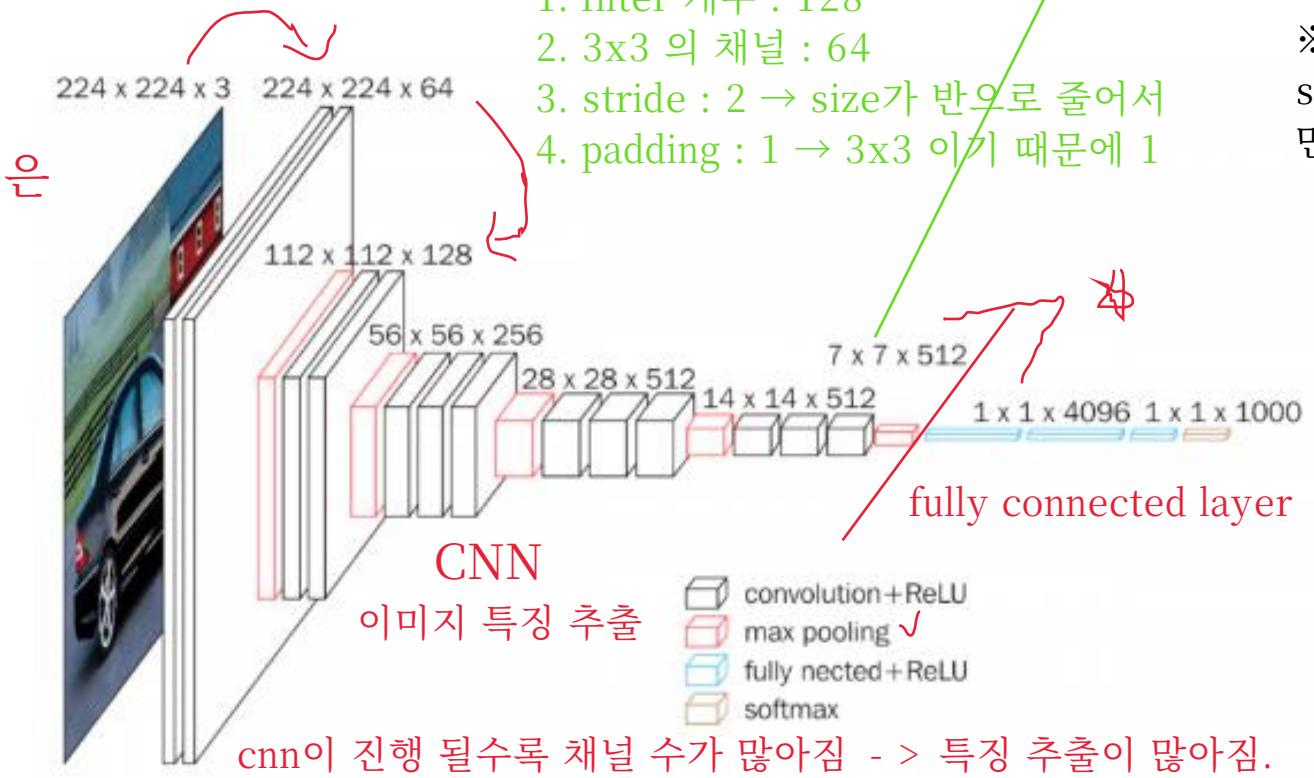
1. filter 개수 : 128
2. 3x3 의 채널 : 64
3. stride : 2 → size가 반으로 줄어서
4. padding : 1 → 3x3 이기 때문에 1

7x7x512 다음에 1x1x4096이 되려면 중간 과정이 하나 생략 되어있다.(그림상 - 논문ㅅㅂ)

7x7x4096 으로 채널만 늘림.
- global max pooling을 사용
- 1x1x4076 : FCL

※ Global max pooling
size랑 상관없이 가장 큰 특징 하나만 추출

원본 이미지의 채널 3은 (R,G,B)



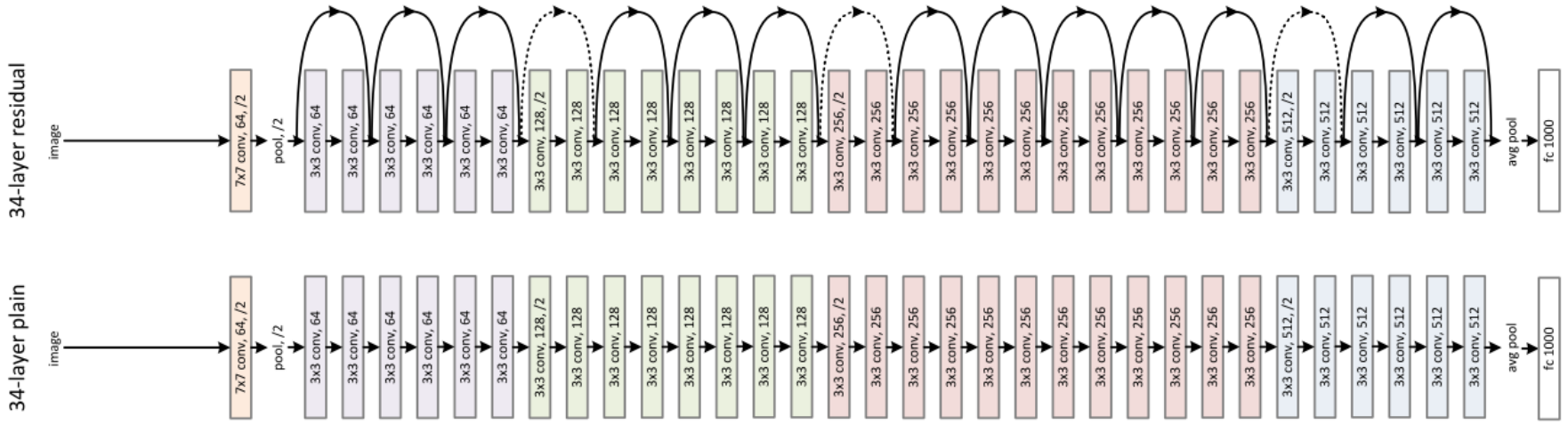
filter size
stride
padding
개념 이해 가능!!
(추가 개념은 요구 되지만)

cnn이 진행 될수록 채널 수가 많아짐 - > 특징 추출이 많아짐.
size가 작아지는 것은 특징을 압축.

max pooling을 배우기 전에 size 변화를 stride 2로 언급했지만 그 당시에는 max pooling을 사용

CNN History

- 4) ResNet (2015)



CNN History

- 5) DenseNet (2016)