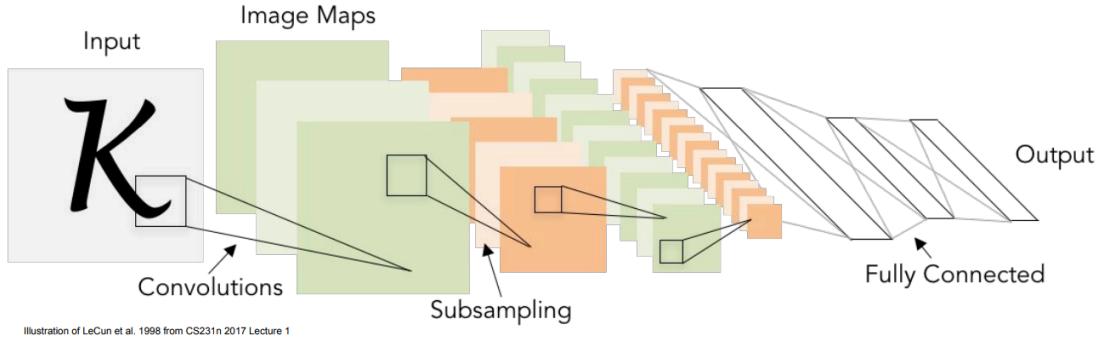
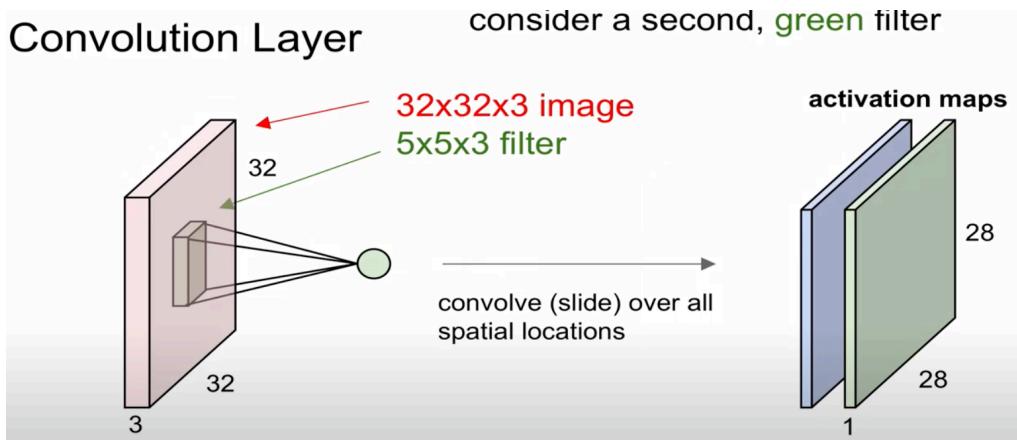


Cnn 7강(2016)

대략적인 Conv 과정 소개

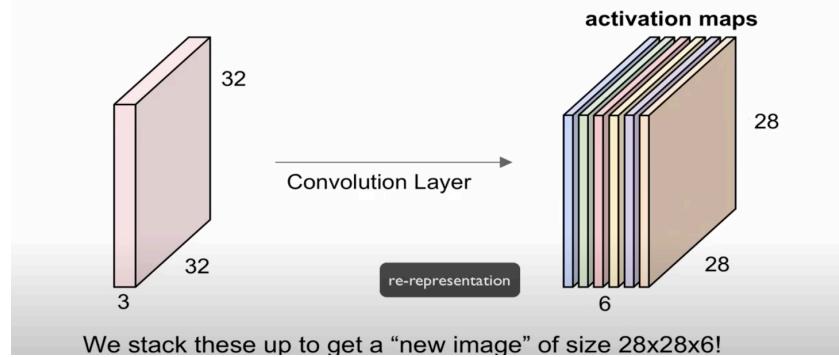


필터가 이미지 내의 모든 공간 위치를 슬라이드 하면서 점곱 연산을 수행한다. cnn의 특징 중 하나는 필터와 이미지의 depth가 동일하다는 것이다. 또한, 하나의 필터는 하나의 activation map을 생성한다.



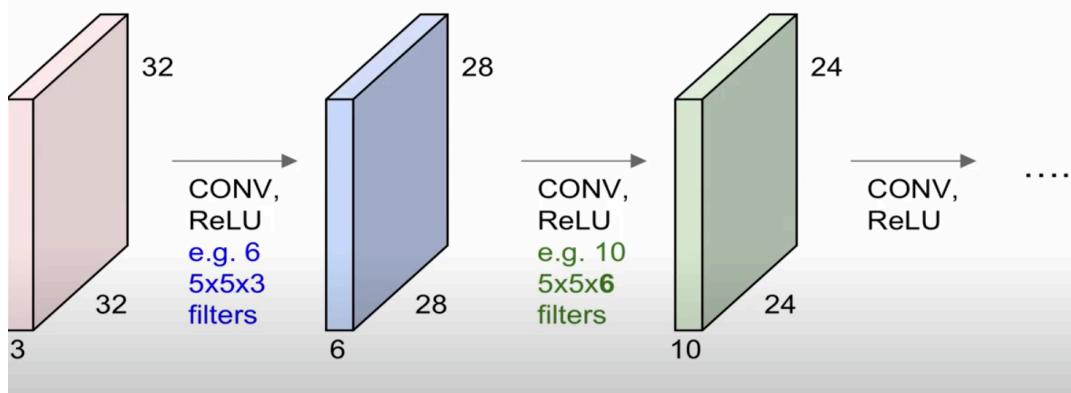
2 번째 필터로 해당 이미지에 컨볼루션을 하면 위와 같이 초록색의 activation map이 추가적으로 생성된다.

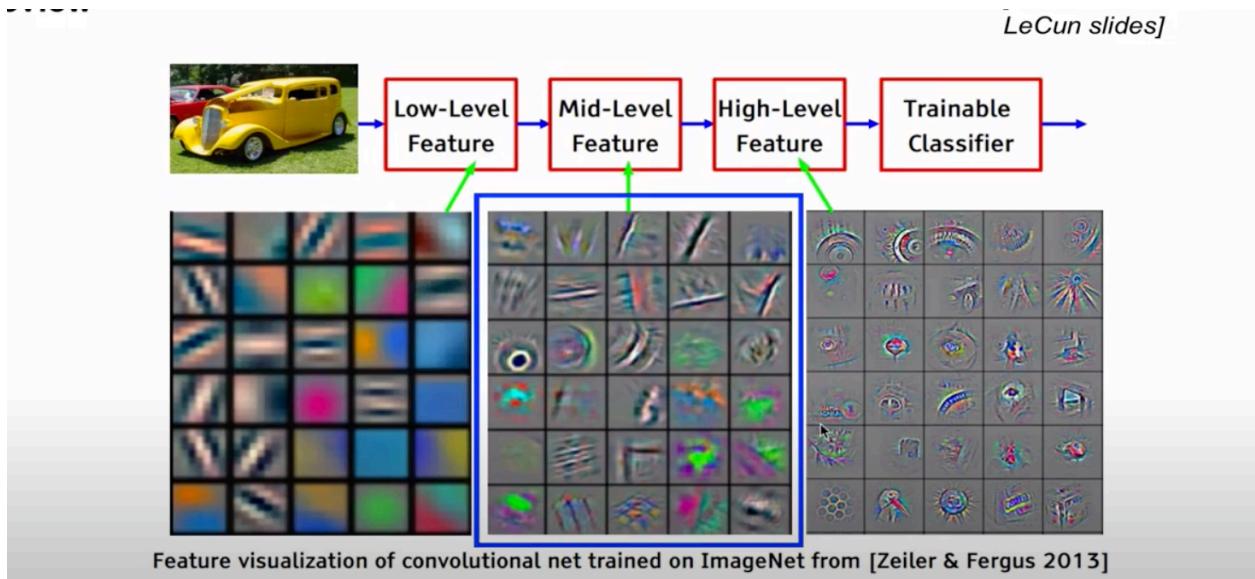
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



위 과정들이 종합되어 $28 \times 28 \times 6$ 형태의 activation map이 생성되었다고 해보자. 즉 이것은 우리는 처음에 $32, 32, 3$ 형태의 입력값을 받았는데 활성화를 통해 $28, 28, 6$ 이라는 활성화 볼륨을 얻게 되는 것이고, 이것은 다음의 conv layer의 입력값으로 들어간다.

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

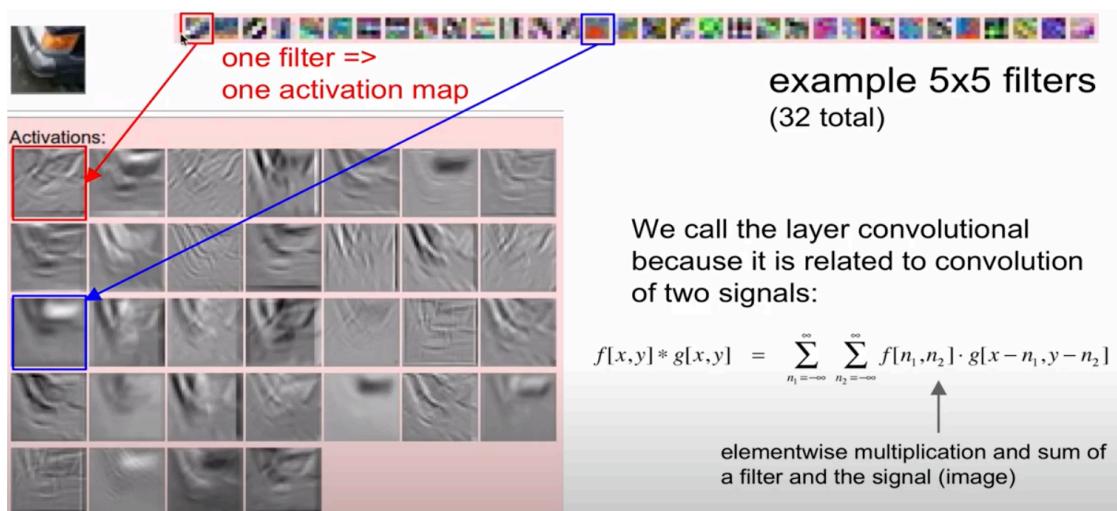




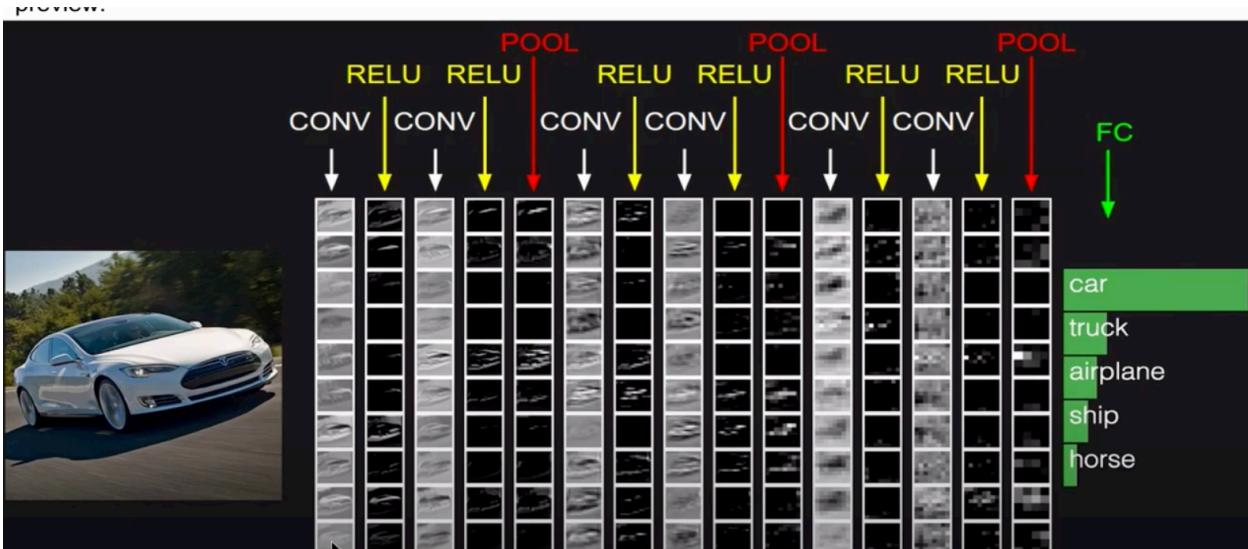
우리가 궁극적으로 업데이트 해야 하는 것은 필터 내의 값들이다.

CNN에서의 첫 번째 필터를 시각화 해보면, edge, blob 등이 보이는데, 이것이 첫 번째 필터에서 볼 수 있는 특징이다. 그리고 중간 단계에 가면, 이전 필터를 통합한 시각화를 볼 수 있고, 마지막의 고 수준 필터가 되면 더 통합된 형태를 볼 수 있다.

실제 사례



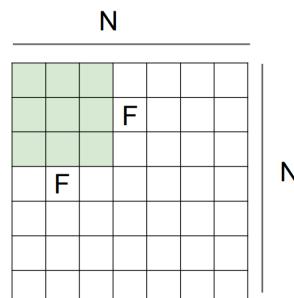
위에 하나의 가로줄은 각각의 필터들을 열거해 놓은 것이다. 위 이미지에 대한 하나의 필터는 하나의 활성화 맵을 만든다. 위 그림은 그 과정을 시각화한 것이다. 참고로 활성화 맵에서 활성화가 높은 곳은 하얀색으로, 낮은 곳은 검은색으로 표현된다.



여러 개의 컨볼루션 레이어들의 수행 과정 전체를 시각화 하면 위와 같다.

어떻게 필터는 입력값 이미지로부터 활성화 맵의 shape을 결정하는가

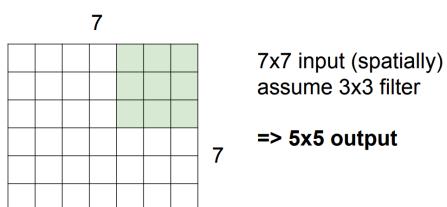
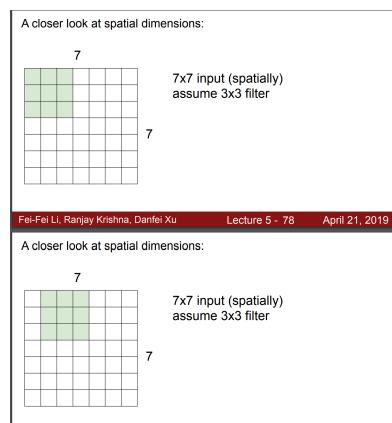
공식:



Output size:
 $(N - F) / \text{stride} + 1$

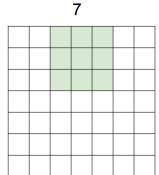
e.g. $N = 7$, $F = 3$:
 stride 1 => $(7 - 3)/1 + 1 = 5$
 stride 2 => $(7 - 3)/2 + 1 = 3$
 stride 3 => $(7 - 3)/3 + 1 = 2.33 : \backslash$

stride = 1



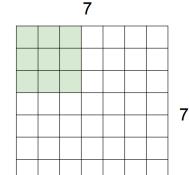
Stride = 2

A closer look at spatial dimensions:



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

stride = 3 (not fit)

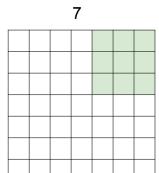


7x7 input (spatially)
assume 3x3 filter
applied with **stride 3?**

Fei-Fei Li, Ranjay Krishna, Danfei Xu

Lecture 5 - 84 April

A closer look at spatial dimensions:

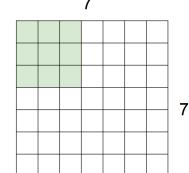


7x7 input (spatially)
assume 3x3 filter
applied with **stride 2**
=> 3x3 output!

Fei-Fei Li, Ranjay Krishna, Danfei Xu

Lecture 5 - 86 April 21

A closer look at spatial dimensions:

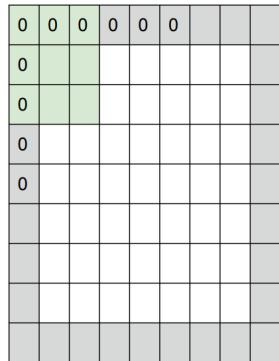


7x7 input (spatially)
assume 3x3 filter
applied with **stride 3?**

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

제로 패딩 (zero padding)

In practice: Common to zero pad the border



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

7x7 output!

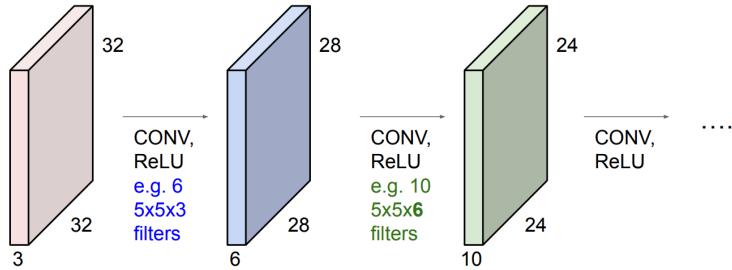
(recall:)
$$(N + 2P - F) / \text{stride} + 1$$

- 제로 패딩 시, 아웃풋의 shape? $(N+2P - F)/\text{stride} + 1$
- 왜 제로 패딩을 하는가? 제로 패딩을 하면 인풋 사이즈와 아웃풋 사이즈를 동일하게 만들어줄 수 있어서 사이즈 보존을 할 수 있다.
- 왜 데이터의 볼륨의 사이즈를 보존하는 것이 중요한가?

컨볼루션 과정을 거칠수록 데이터의 볼륨이 줄어드는데, 그러면 결국 그것의 볼륨은 0이 될 수 있다. 그렇다면 우리는 컨볼루션을 더 진행할 수 없다. 이것을 막기 위해 패딩을 이용해서 사이즈를 보존해준다. 하지만 다른 측면에서 고려할 때 사이즈를 줄이는 것도 중요하다. 이것은 아래서 다시 알아보자.

Remember back to...

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially!
(32 \rightarrow 28 \rightarrow 24 ...). Shrinking too fast is not good, doesn't work well.



위 그림은 컨볼루션 과정을 진행할 때마다 활성화 맵의 사이즈가 줄어드는 것을 보여준다. 반면에,

In practice: Common to zero pad the border

0	0	0	0	0	0				
0									
0									
0									
0									

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. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

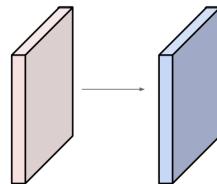
위 그림은 필터의 크기가 3, 5, 7일 때 사이즈 보존을 위해 요구되는 제로 패드의 개수를 보여준다.

* 제로 패딩 예제)

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Output volume size: ?

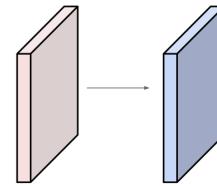
Ans) $(N+2P-F) / \text{stride} + 1 = (32 + 2*2 - 5) / 1 + 1 = 32$. 즉, Output의 2D size는 32×32 . 하지만 필터의 개수가 10 개이기 때문에, 전체 3D size는 $32 \times 32 \times 10$.

** 제로 패딩 예제2)

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

each filter has $5 \times 5 \times 3 + 1 = 76$ params
=> $76 \times 10 = 760$

(+1 for bias)

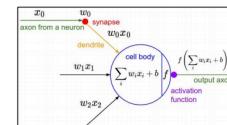
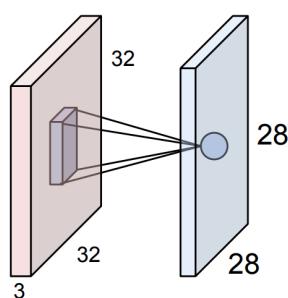
Q. 이 레이어의 파라미터 개수?

$$(\text{필터의 개수})(F^2 * \text{depth} + \text{bias}) = 10(5^2 * 3 + 1) = 760$$

파라미터 공유

동일한 템스 내의 뉴런들은 파라미터 공유를 한다. 하지만 가령 필터가 5 개일 때, 각각의 다른 활성화 맵에서 동일한 위치에 있는 뉴런은 아래의 두 번째 그림처럼 인풋 이미지의 동일한 지점을 향하지만 이 경우는 파라미터 공유는 하지 않는다.

The brain/neuron view of CONV Layer

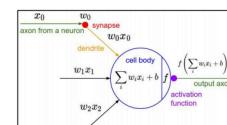
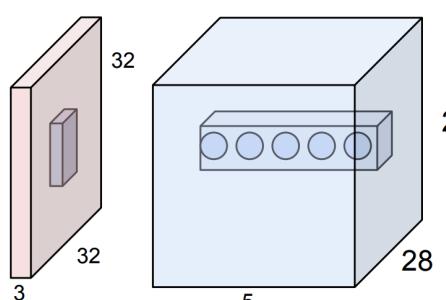


An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The brain/neuron view of CONV Layer



E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

풀링 레이어

데이터의 볼륨을 좀 더 작게, 관리하기 편하게 만들어주는 것. 이것은 각각의 활성화 맵에 대해 독립적으로 작용한다. 따라서 데스는 동일하게 만들지만 사이즈는 줄여준다. 즉, 풀링 레이어는 데이터에 대해 데스는 유지하면서 다운샘플링 시킨다.

Pooling layer

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

