BE530 – Medical Deep Learning

– Convolutional Neural Networks –

Byoung-Dai Lee

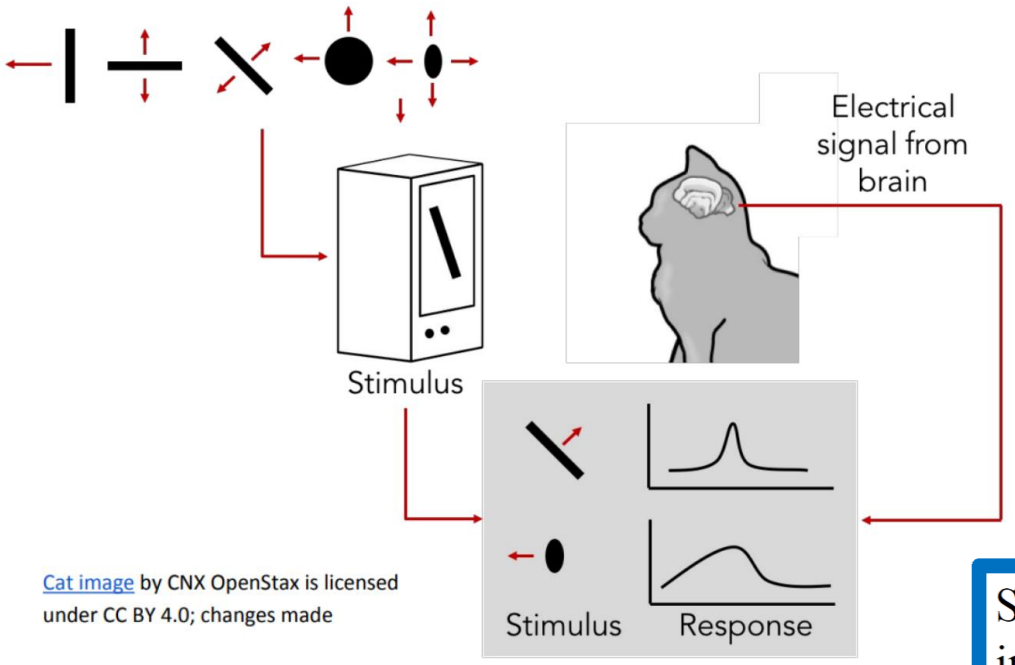
Division of AI Computer Science and Engineering

Kyonggi University

KYONGGI UNIVERSITY

Biological Inspiration

- David H. Hubel & Torsten Wiesel
  - 시각 피질안의 뉴런들이 각자 수용 영역(receptive field)을 담당하여 인지
  - 인지 과정이 계층적으로 일어남

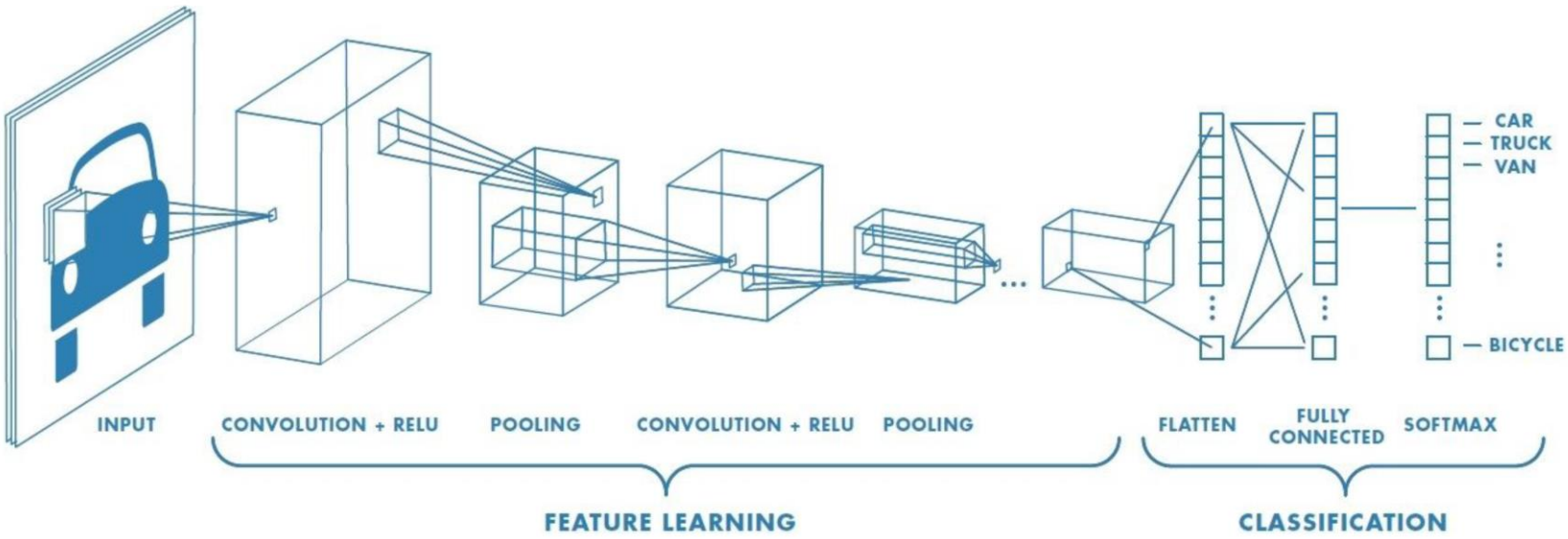


Cat image by CNX OpenStax is licensed under CC BY 4.0; changes made

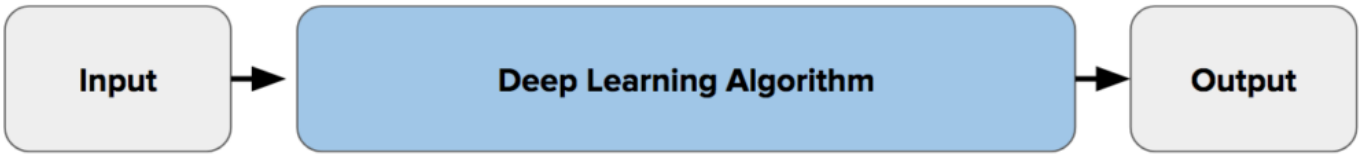
David H. Hubel and Torsten Wiesel

Suggested a **hierarchy** of **feature detectors** in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells.

A Classical CNN Architecture



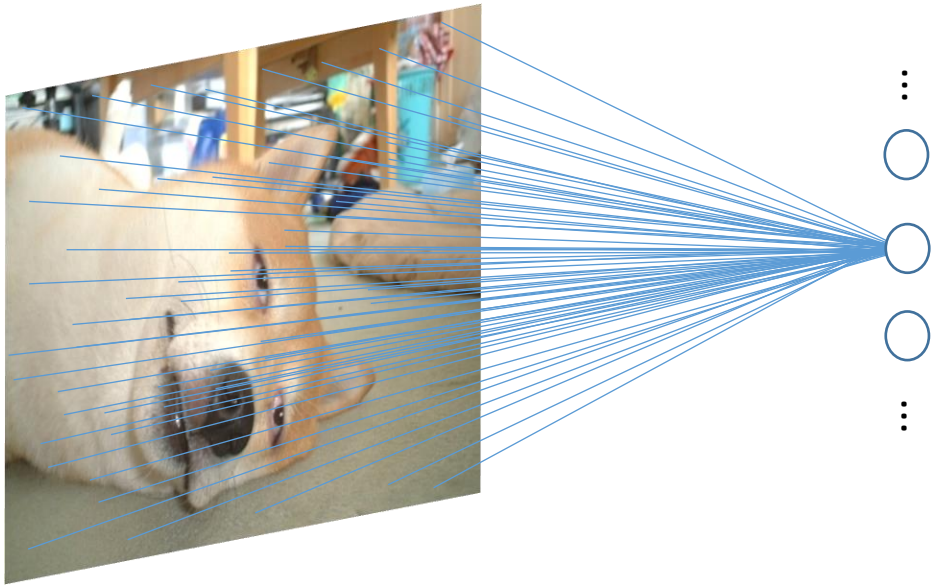
Traditional Machine Learning Flow

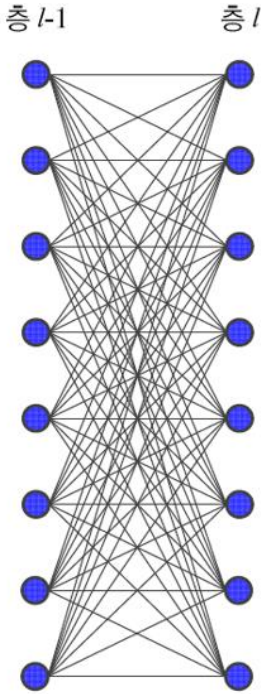


Deep Learning Flow

Deep MLP vs. CNN (1)

- Deep MLP
  - 인접한 계층 간 완전 연결 구조로 높은 복잡도
    - 학습이 느리고 과적합에 빠질 수 있음
  - 입력 크기가 고정





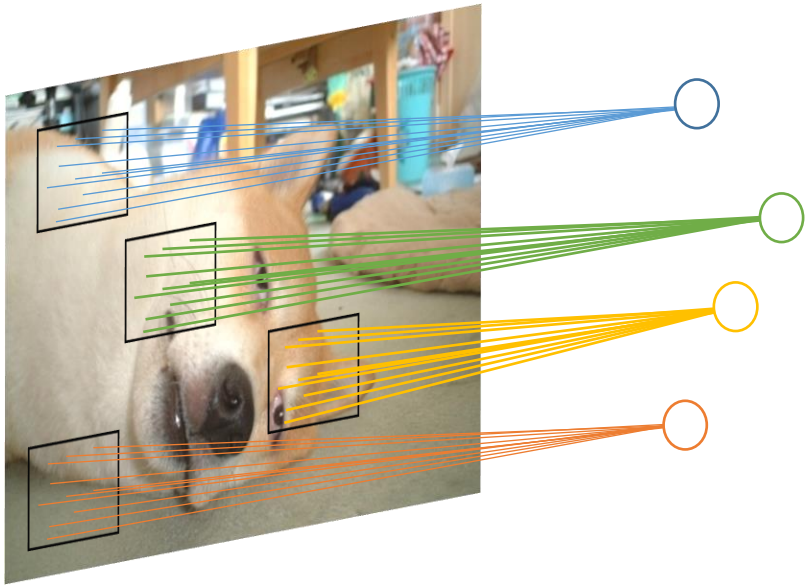
A 200×200 image
40k hidden unit

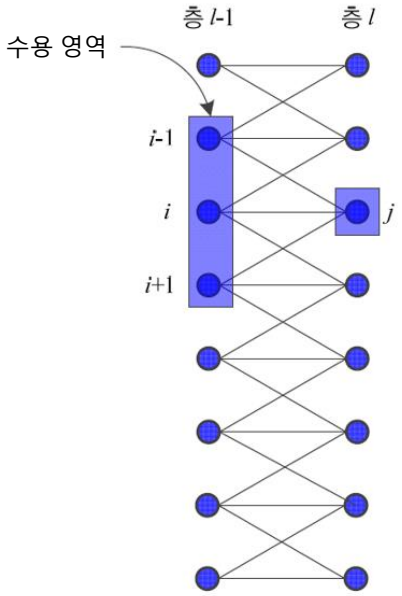
➡

200×200×40k =
1.6B parameters

Deep MLP vs. CNN (2)

- CNN
  - 수용 영역을 대상으로 부분 연결 구조
    - Spatial correction ➔ local
  - 층을 쌓아가면 점진적으로 고차원적 특징을 학습
  - 가변 크기의 입력 처리 가능





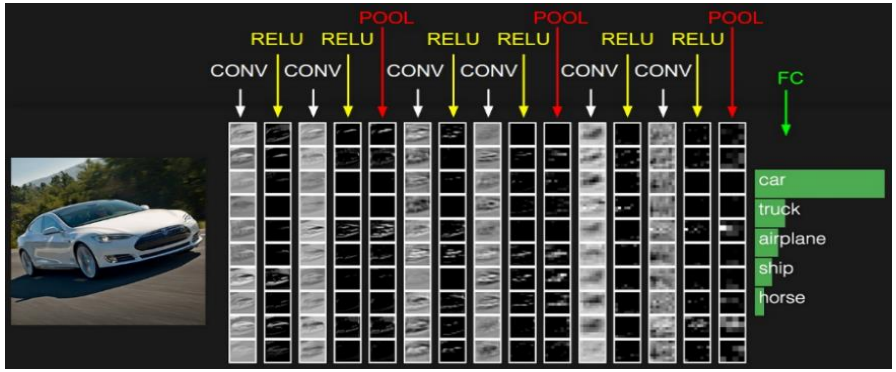
A 200×200 image
40k hidden unit
Filter Size: 10×10

➡

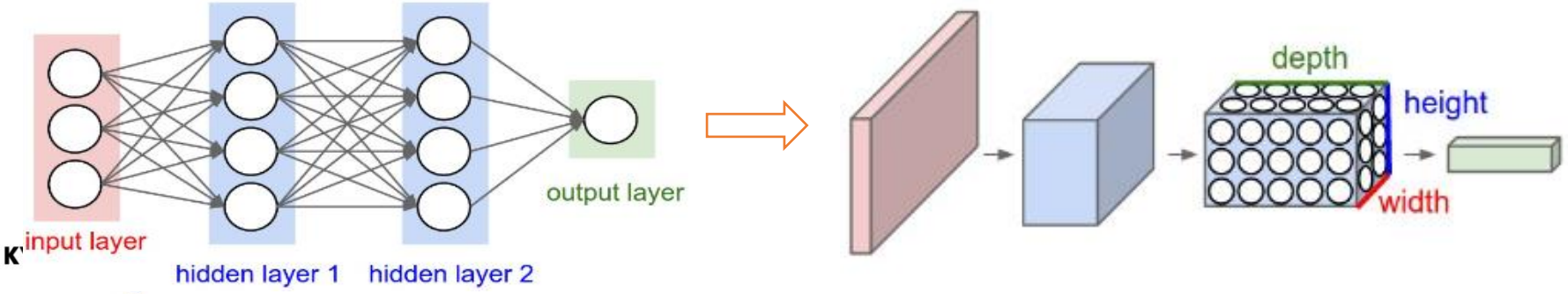
10×10×40k = 4MB parameters
10×10 = 100 parameters

CNN Architecture

- Main layers for CNNs
  - Convolutional layer, ReLU layer, Pooling layer, FC layer



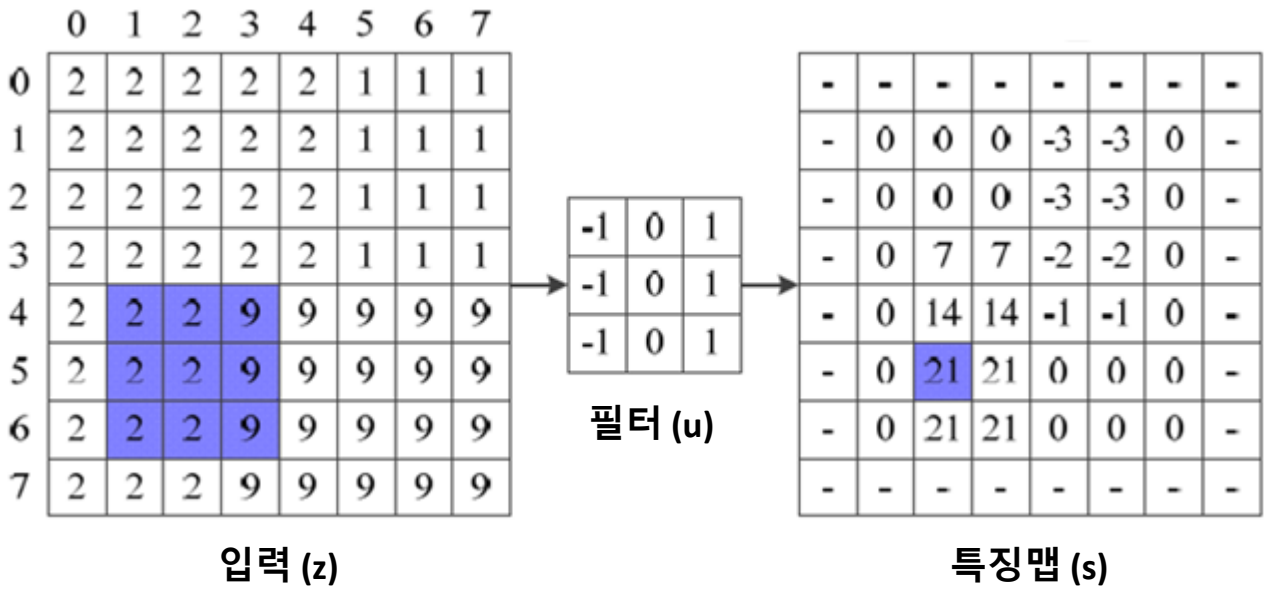
- CNN arranges its neurons in 3-dimensions
  - Every layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations



Convolutional Layer (1)

- 기본 개념

$$s(j,i) = z \circledast u = \sum_{y=-(h-1)/2}^{(h-1)/2} \sum_{x=-(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x)$$

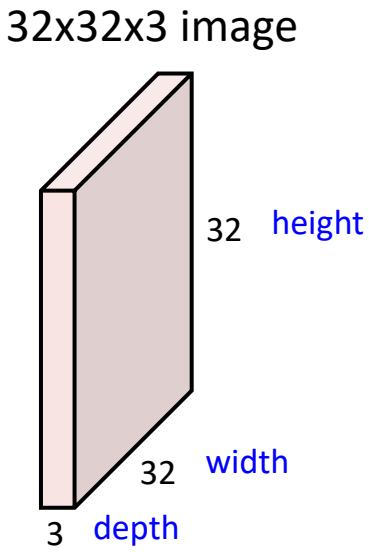


입력 (z)

필터 (u)

특징맵 (s)

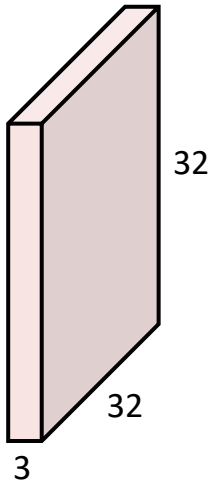
Convolutional Layer (2)



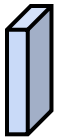
* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (3)

32x32x3 image



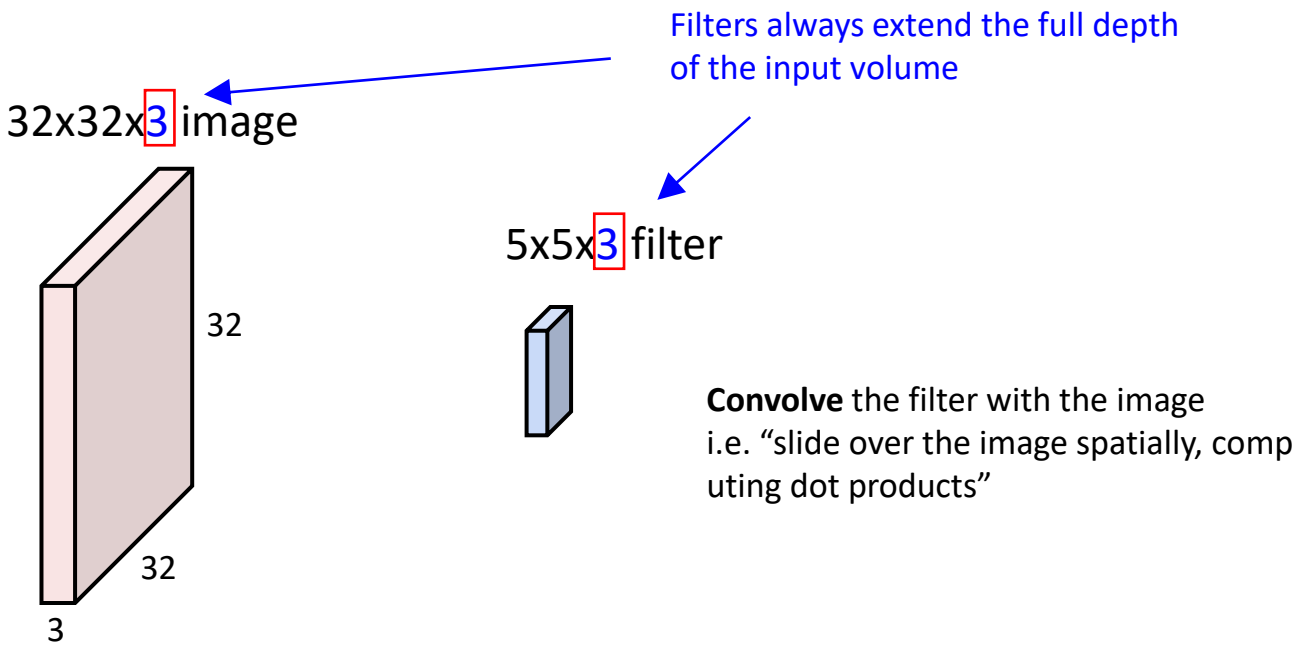
5x5x3 filter



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

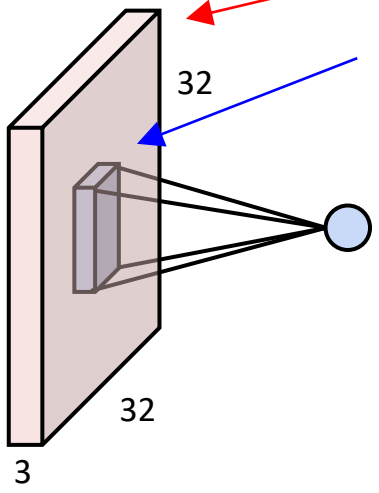
* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (4)



* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (5)

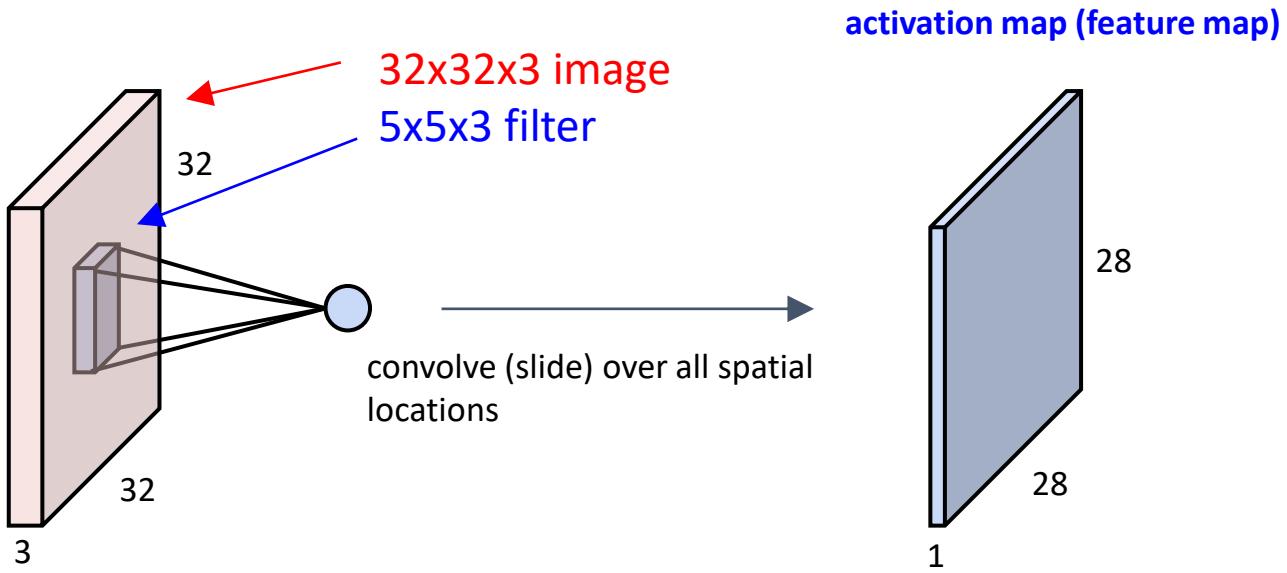


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

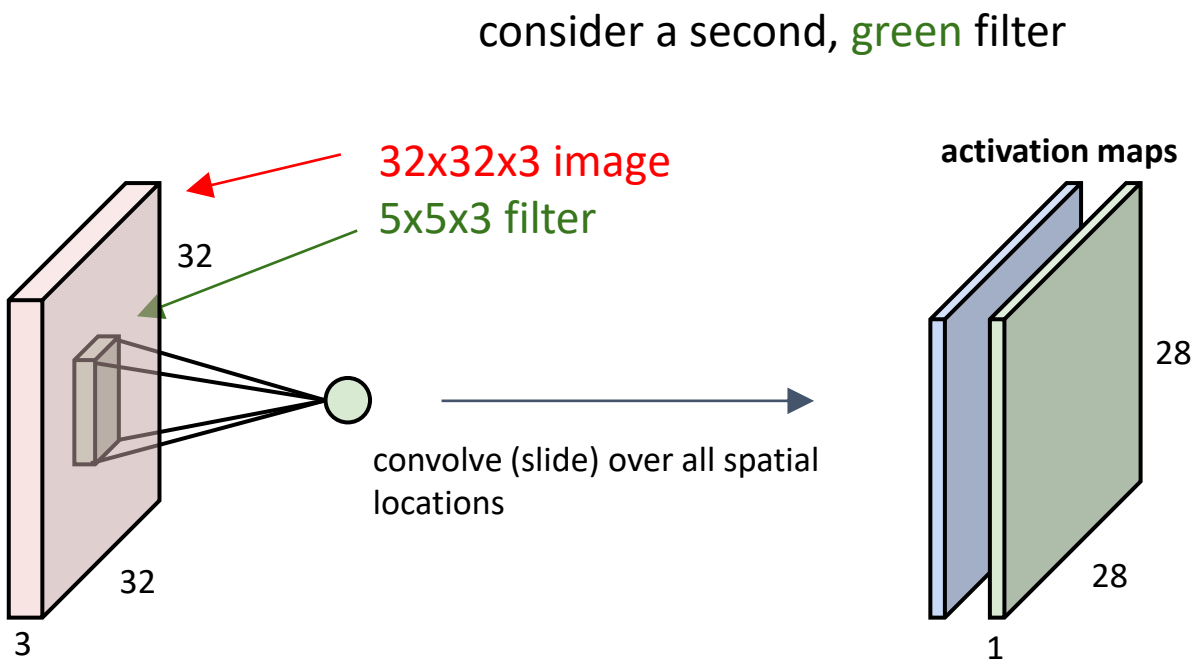
* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (6)



* Source: Stanford Univ. CS231 Lecture Notes

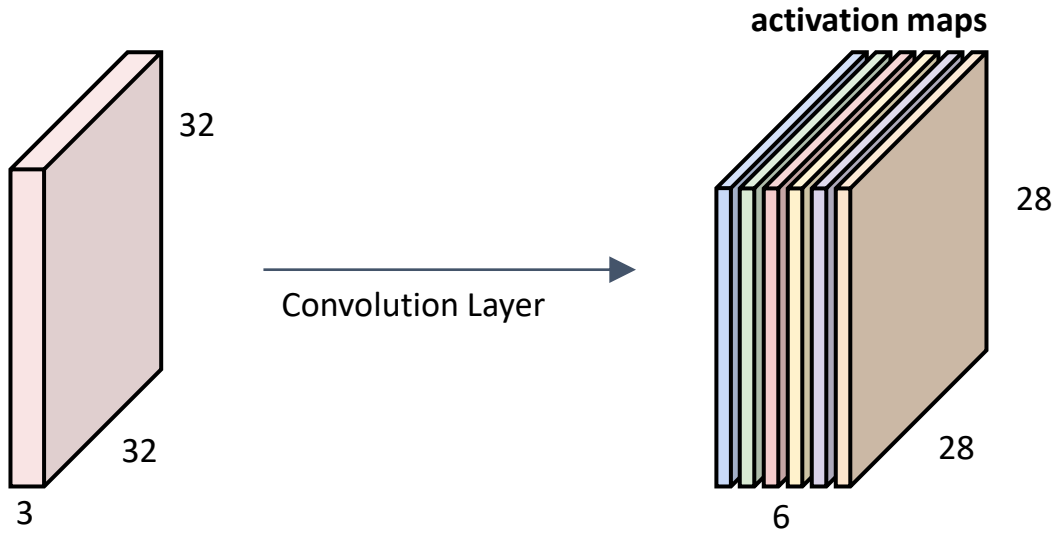
Convolutional Layer (7)



* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (8)

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

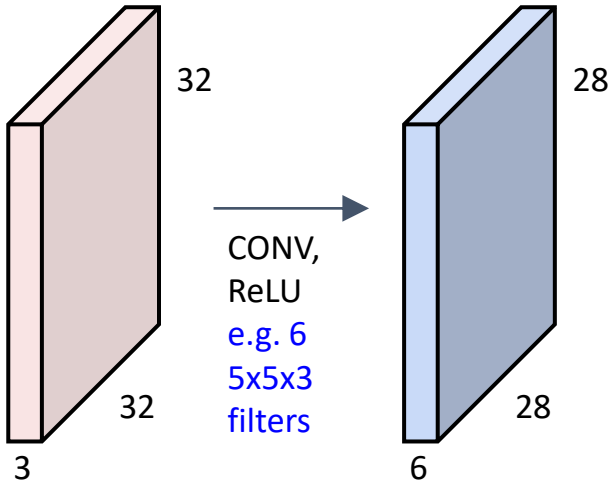


We stack these up to get a "new image" of size 28x28x6!

* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (9)

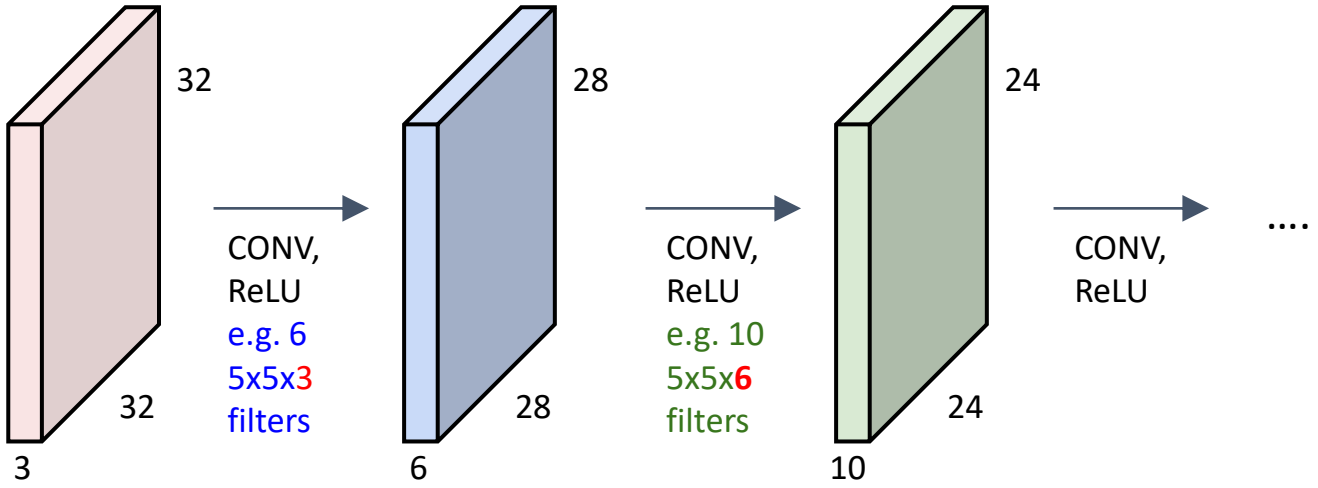
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



* Source: Stanford Univ. CS231 Lecture Notes

Convolutional Layer (10)

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



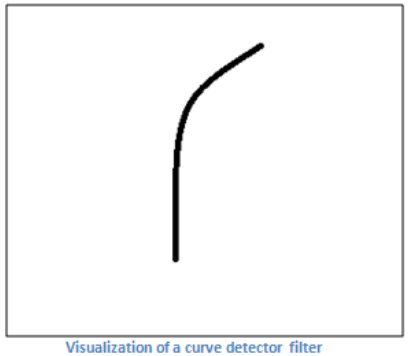
* Source: Stanford Univ. CS231 Lecture Notes

Filters (1)

- Each of filters can be thought of as a feature identifier
  - Curve detector, line detector, ⋯
- Filters are also learnable

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

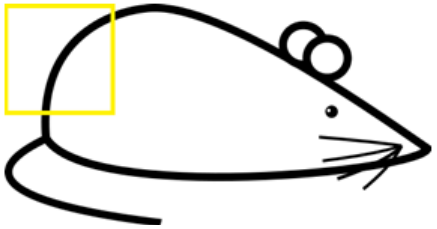
Pixel representation of filter



Visualization of a curve detector filter

Original image

Filters (2)



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

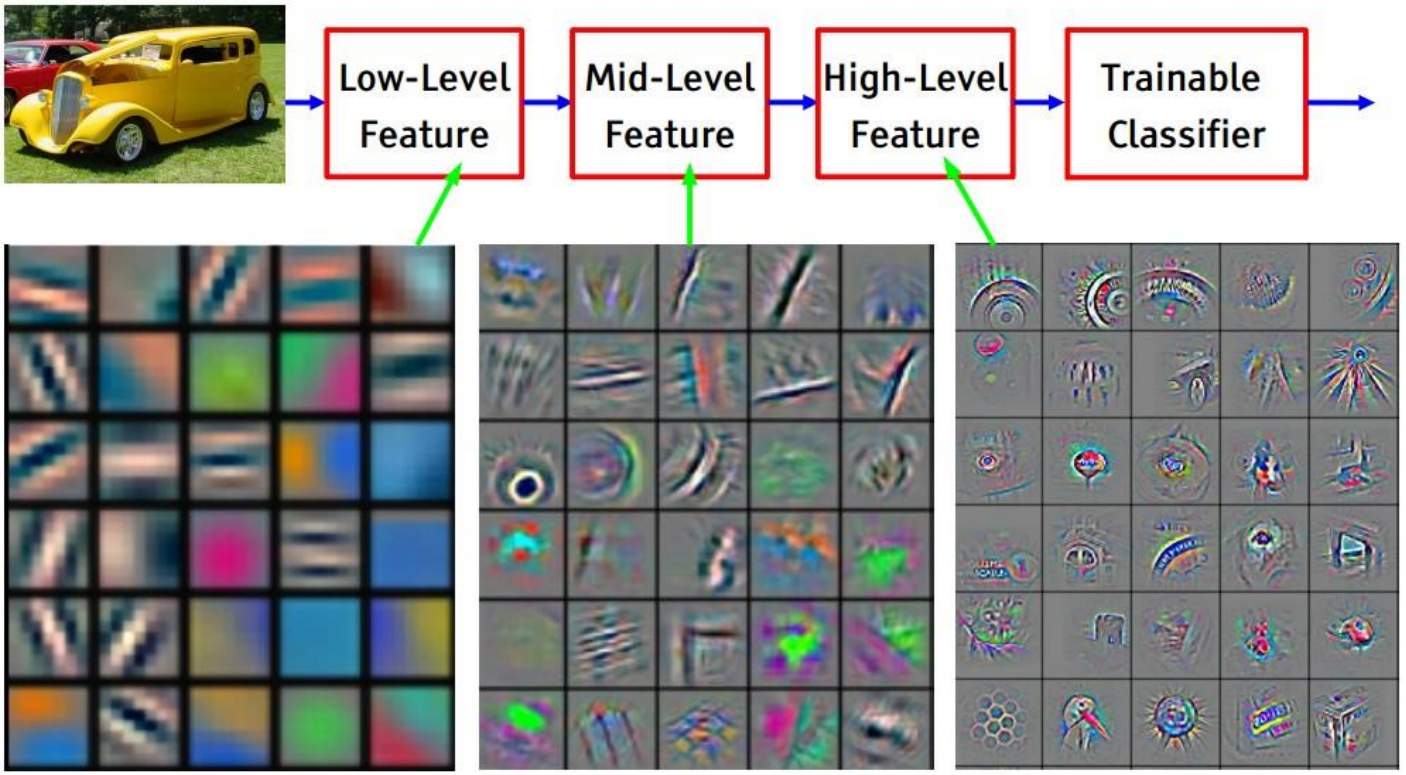
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

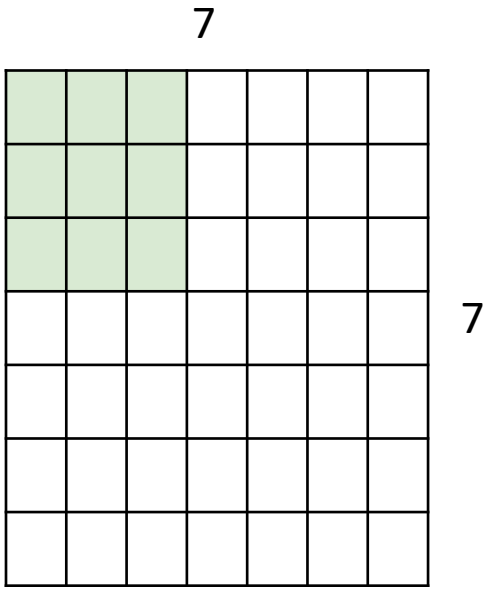
Multiplication and Summation = 0

Filters (3)



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

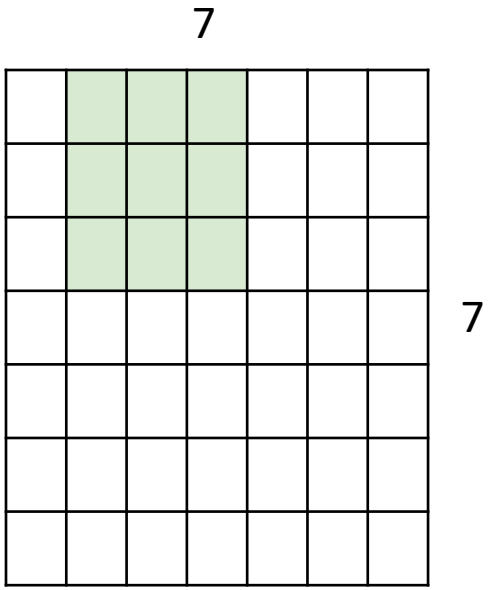
Stride (1)



7x7 input (spatially) assume 3x3 filter

* Source: Stanford Univ. CS231 Lecture Notes

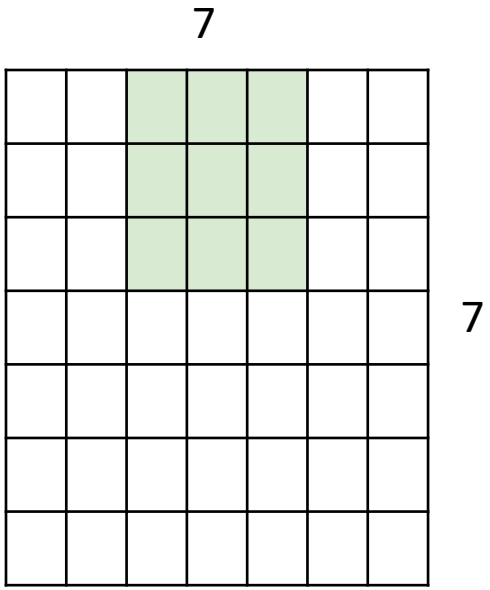
Stride (2)



7x7 input (spatially) assume 3x3 filter

* Source: Stanford Univ. CS231 Lecture Notes

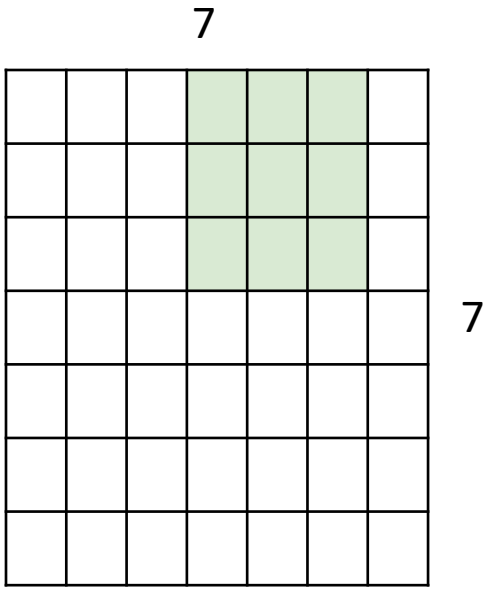
Stride (3)



7x7 input (spatially) assume 3x3 filter

* Source: Stanford Univ. CS231 Lecture Notes

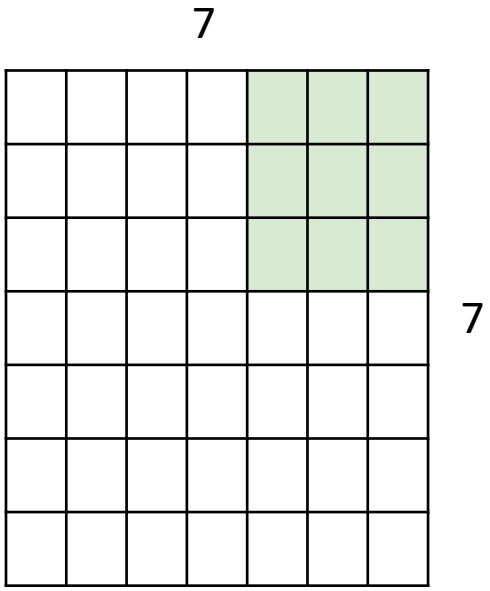
Stride (4)



7x7 input (spatially) assume 3x3 filter

* Source: Stanford Univ. CS231 Lecture Notes

Stride (5)

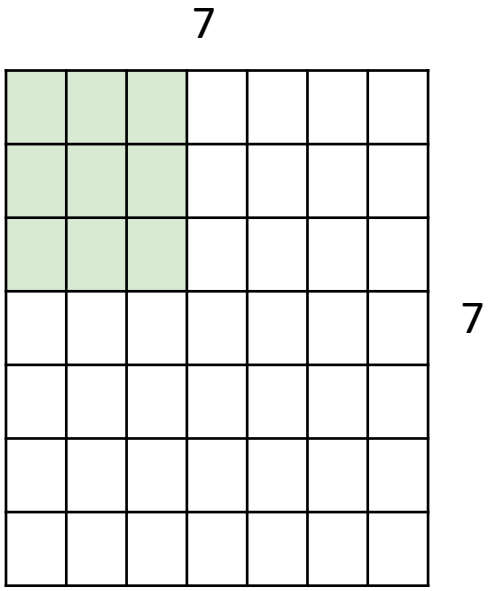


7x7 input (spatially) assume 3x3 filter

=> 5x5 output

* Source: Stanford Univ. CS231 Lecture Notes

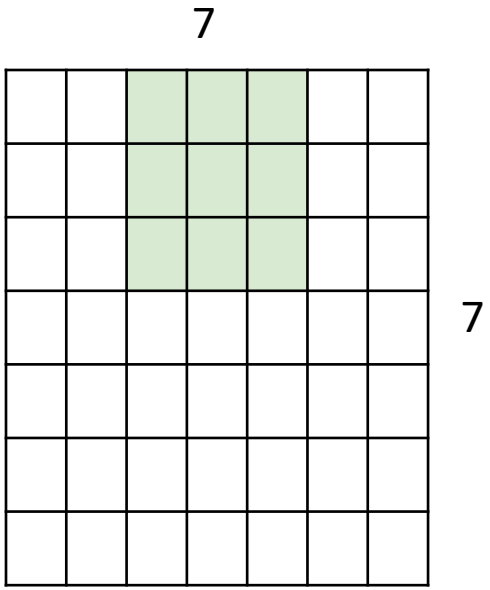
Stride (6)



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

* Source: Stanford Univ. CS231 Lecture Notes

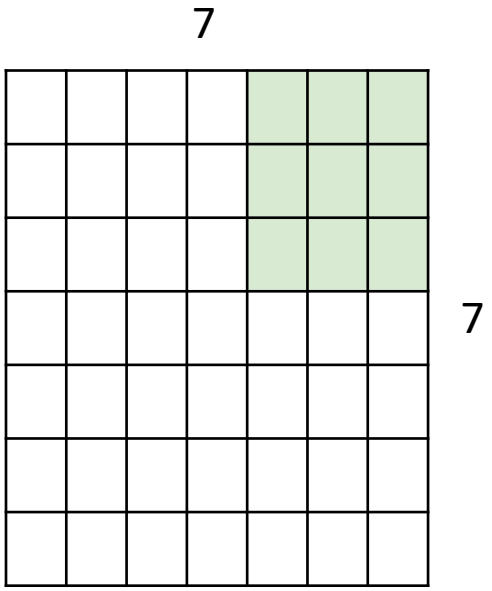
Stride (cont.)



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

* Source: Stanford Univ. CS231 Lecture Notes

Stride (7)



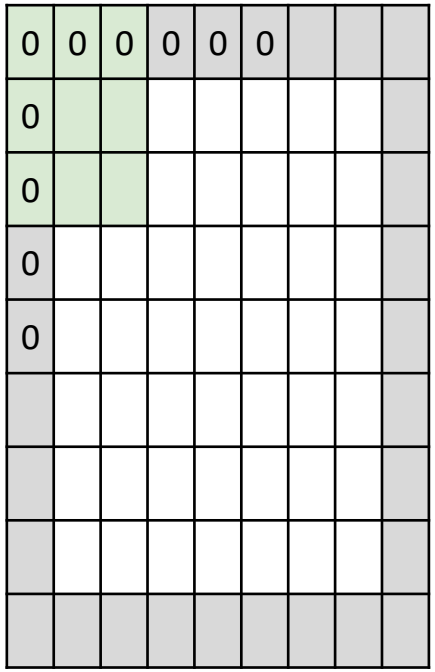
7x7 input (spatially) assume 3x3 filter applied **with stride 2**

=> 3x3 output!

* Source: Stanford Univ. CS231 Lecture Notes

Zero padding (1)

In practice: Common to zero pad the border

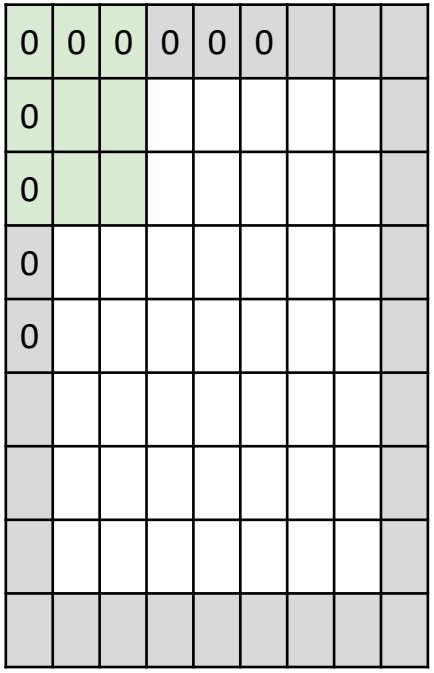


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

* Source: Stanford Univ. CS231 Lecture Notes

Zero padding (2)

In practice: Common to zero pad the border



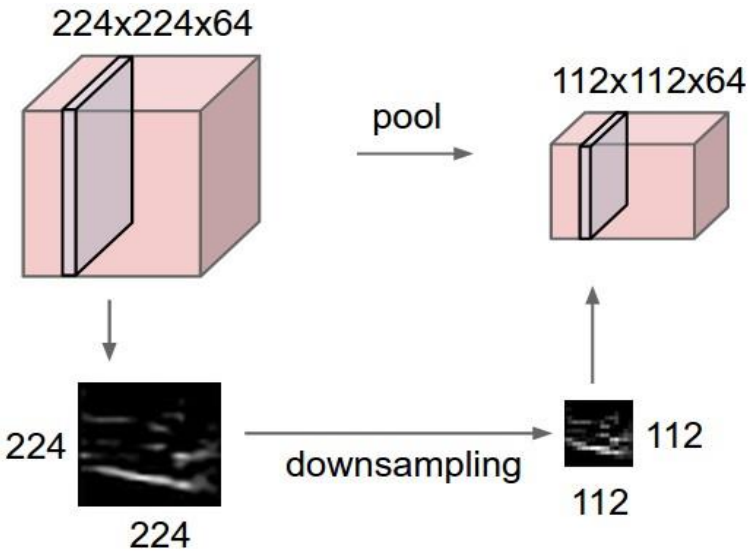
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

* Source: Stanford Univ. CS231 Lecture Notes

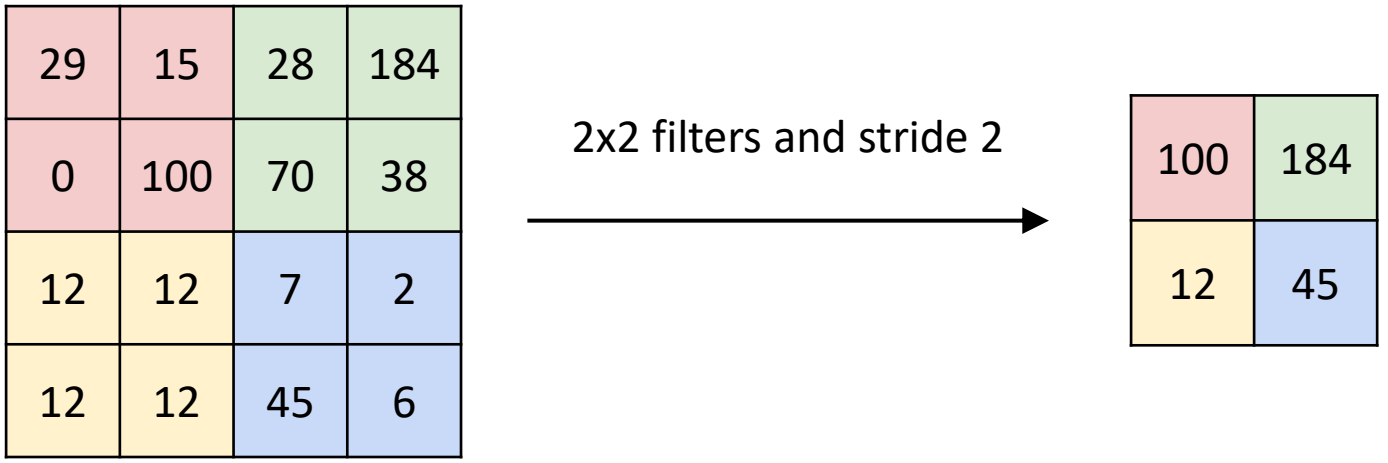
Pooling Layer (1)

- Make the representation smaller and more manageable
- Operates over each activation map independently



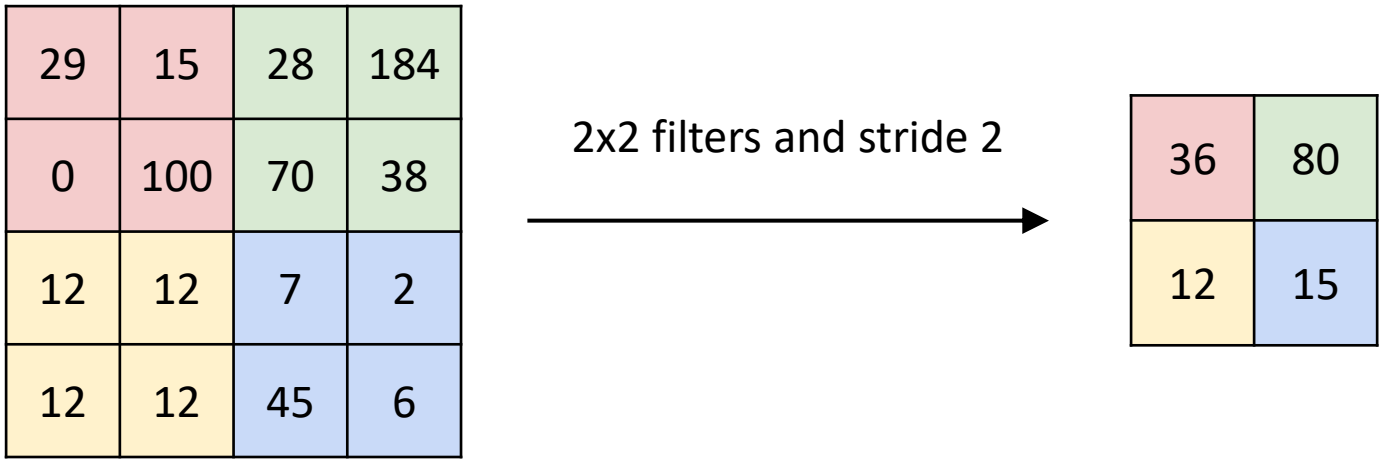
Pooling Layer (2)

- Max Pooling

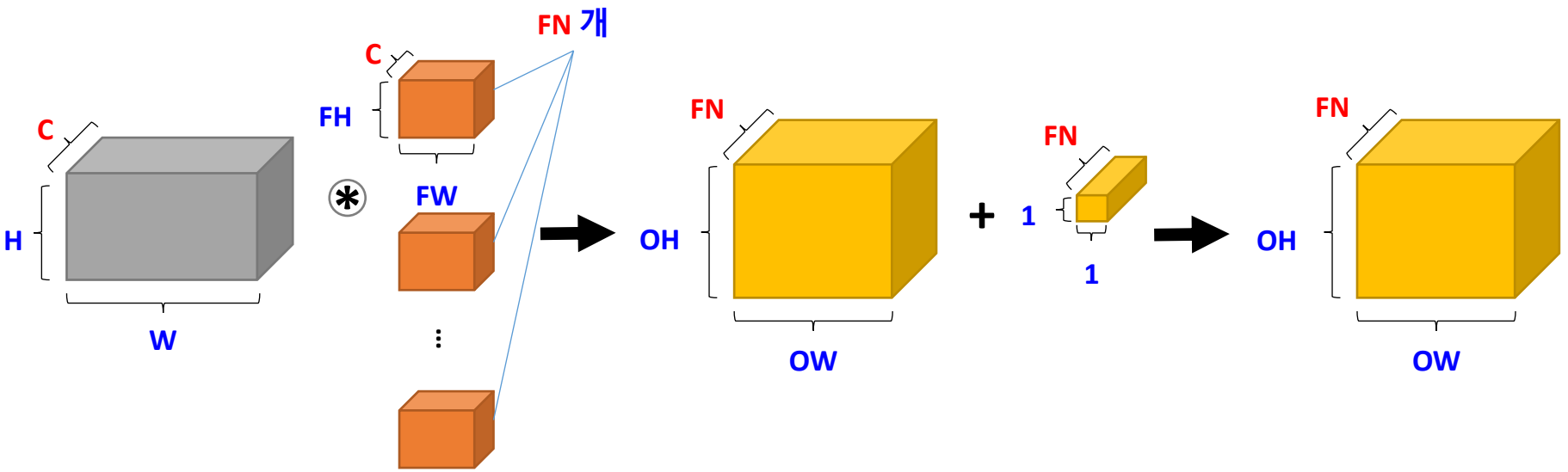


Pooling Layer (3)

- Average Pooling



Spatial Dimensions (1)



(C, H, W) Input ⊛ **(FN, C, FH, FW)** Filters → (FN, OH, OW) + (FN, 1, 1) Bias → **(FN, OH, OW)** Output

Spatial Dimensions (2)

- Input volume of size: W x H x C
- Hyperparameters
  - Number of filters: FN
  - Filter (Receptive Field) size: (FW, FH)
  - Stride: S
  - The amount of zero padding: P
- Output volume of size: OW x OH x FN
  - OW = (W−FW+2P)/S + 1
  - OH = (H−FH+2P)/S + 1

- (FW×FH×C)×FN weights and FN biases are required

Spatial Dimensions (3)

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params
=> 76*10 = **760**

Spatial Dimensions (4)

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Hyperparameters in CNN (1)

① Filter의 개수

- 일반적으로 영상의 크기가 큰 입력단 근처의 layer에는 filter의 개수가 적고, 입력단에서 멀어질수록 filter의 개수가 증가하는 경향
- Convolution layer의 연산 시간
  - $T_c = N_p \times N_f \times T_k$

    N_p:출력 Pixel 수, N_f:전체 feature map의 개수, T_k 각 filter당 연산 시간
- Filter 개수를 정할 때 주로 사용하는 방법은 각 layer에서의 연산 시간량을 비교적 일정하게 유지하여 시스템의 균형을 맞춤
  - 예) Pooling layer를 거치면서 2×2 sub-sampling을 하게 될 경우 convolutional layer를 지날 때 마다 pixel 수가 ¼로 감소 ➔ feature map의 개수를 대략 4배 정도 증가시키면 됨

Hyperparameters in CNN (2)

② Filter의 형태

- 일반적으로 32×32 또는 28×28과 같은 작은 크기의 영상에 대해서는 5×5 filter를 주로 사용하지만 큰 크기의 자연 영상을 처리할 때나 혹은 1단계 필터에는 11×11이나 15×15와 같은 큰 크기의 filter를 사용하기도 함
- 여러 개의 작은 크기의 filter를 중첩해서 사용하는 것이 유리함
  - 예) 1개의 7×7 filter vs. 3×3 filter를 중첩해서 사용
  - 작은 filter 여러 개를 중첩하면 중간 단계에 있는 non-linearity를 활용하여 원하는 특징을 좀 더 돋보이게 할 수 있음
  - 작은 filter를 중첩해서 사용하는 것이 연산량도 적게 소요됨

Hyperparameters in CNN (3)

③ Stride 값

- Stride는 입력 영상이 큰 경우 연산량을 줄이기 위한 목적으로 입력단과 가까운 쪽에만 적용함
- Stride vs. Pooling
  - Stride를 1로 하면 경계가 아닌 모든 입력 영상에 대해 convolution 연산을 수행하고 pooling하면서 값을 선택적으로 고를 수 있으나 Stride를 크게 하면 그런 기회가 사라짐
  - 일반적으로 Stride를 1로 하고 Pooling을 통해 적절한 sub-sampling 과정을 거치는 것이 결과가 좋음
  - 큰 영상에 대해 CNN을 적용하는 경우 연산량을 줄이기 위해 입력 영상을 직접 처리하는 1단계 Convolutional layer에서 stride값을 1이 아닌 값으로 적용하기도 함

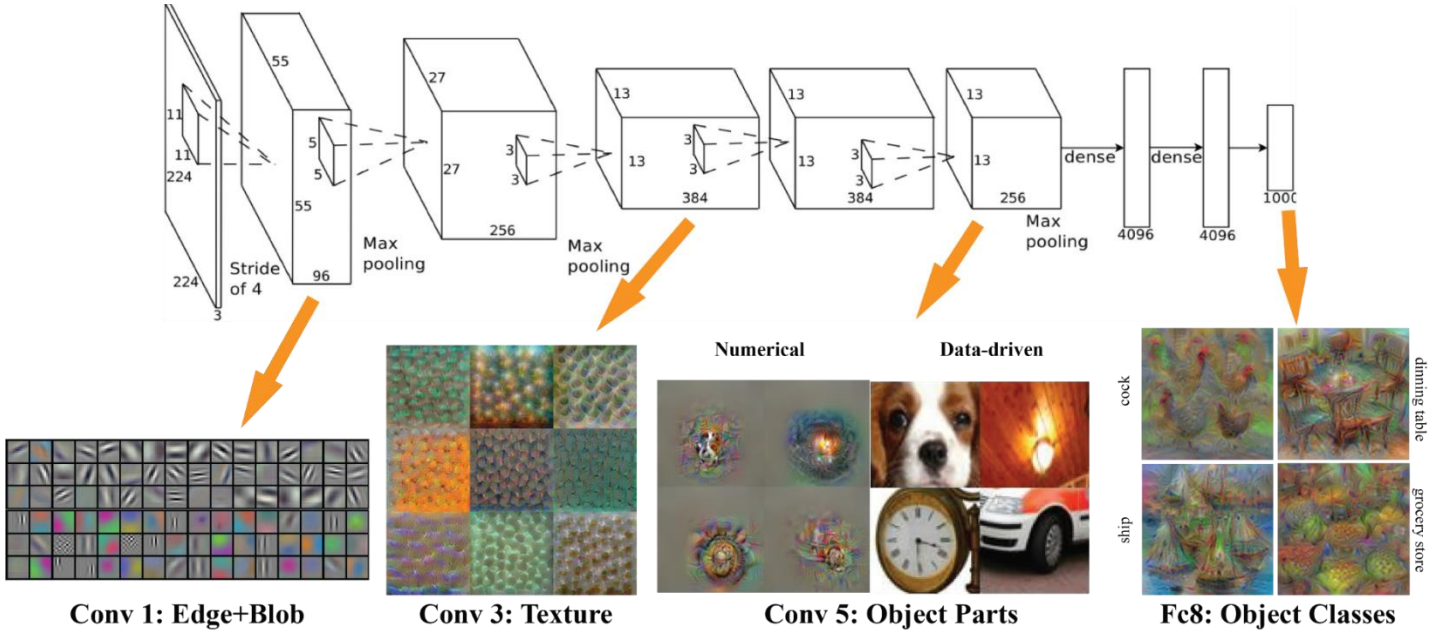
Hyperparameters in CNN (4)

④ Zero-padding 지원 여부

- 일반적으로 convolution 연산을 수행하게 되면 경계 처리 문제로 인해 출력 영상인 feature map의 크기가 입력 영상보다 작아지게 됨
  - Zero-padding을 통해 영상의 크기를 동일하게 유지
- 경계면의 정보까지 살릴 수 있어 zero-padding을 지원하지 않는 경우에 비해 좀 더 좋은 결과를 보임

AlexNet

- DNN 구조
  - Convolutional layer – 5개 (≈ 2M parameters)
  - FC layer – 3개 (≈ 65M parameters)
    - FC layer에 많은 parameters가 존재 ➔ 향후 CNN은 FC layer의 parameters를 줄이는 방향으로 발전



References

- CS231n: Convolutional Neural Networks for Visual Recognition, http://cs231n.stanford.edu/
- A Beginner's Guide to Understanding Convolutional Neural Networks, https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
- Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail, https://iopscience.iop.org/article/10.1088/1742-6596/1201/1/012052
- ImageNet Classification with Deep Convolutional Neural Networks, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- 라온피플 머신러닝 아카데미, https://laonple.blog.me/


ANY
QUESTIONS?