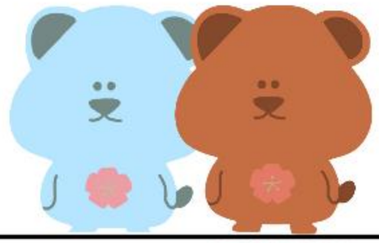
Convolutional Neural Networks

Week5 민소연,안서연

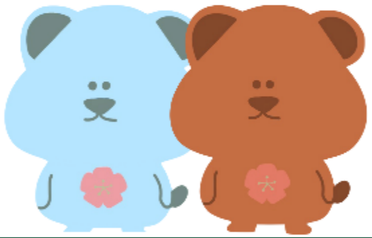
EWHA EURON

Index



EWHA EURON

Overview



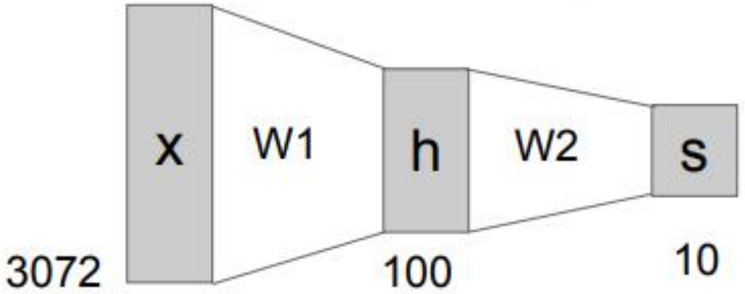
EWHA EURON

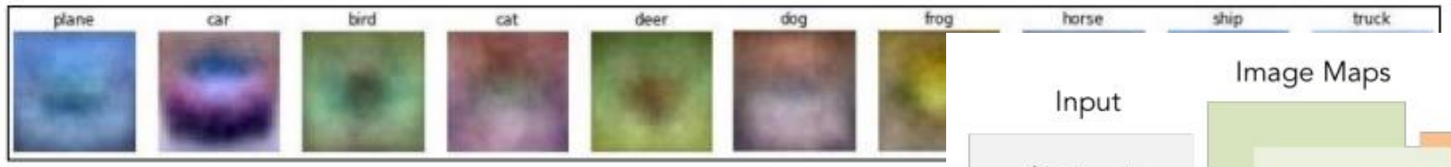
01 In the last lecture…

Last time: Neural Networks

Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$





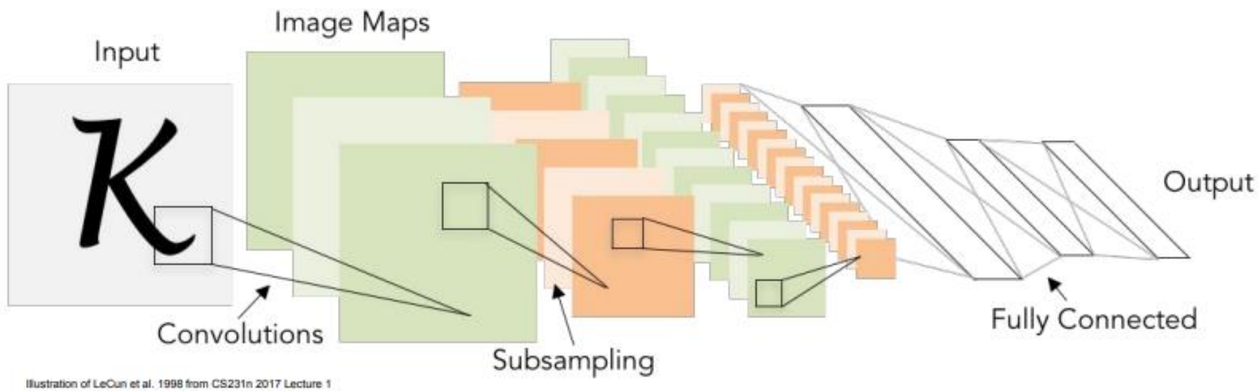
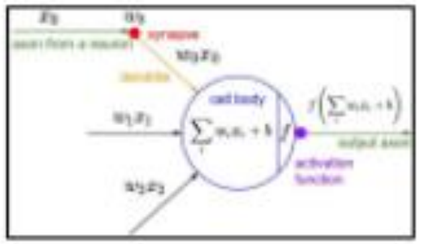


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

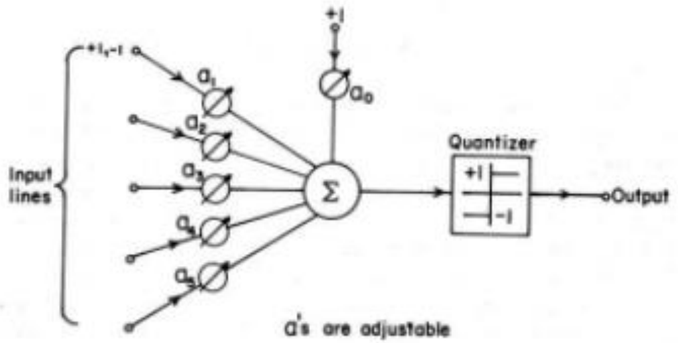
02 History

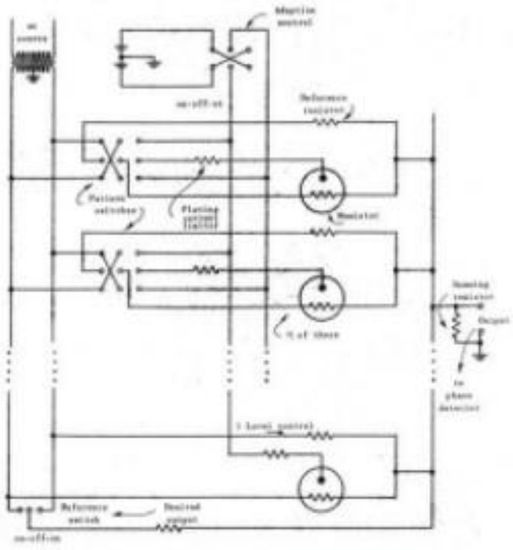
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



update rule:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$





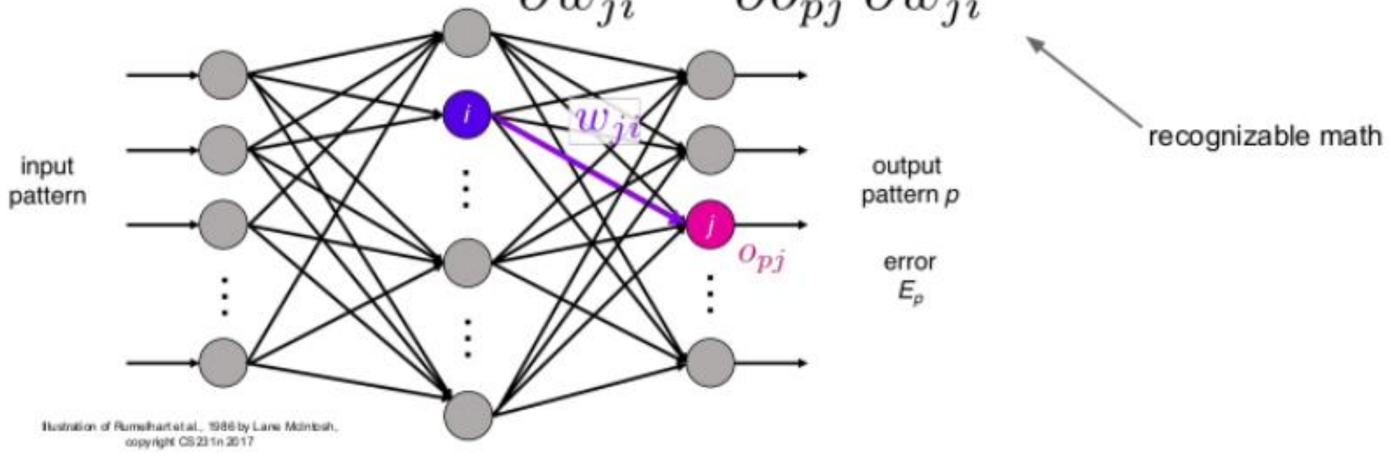
Widrow and Hoff, ~1960: Adaline/Madaline

These figures are reproduced from Widrow 1960, Stanford Electronics Laboratories Technical Report with permission from Stanford University Special Collections.

02 History

A bit of history...

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}$$



Rumelhart et al., 1986: First time back-propagation became popular

제 1 암흑기
Combinational Explosion

제 2 암흑기
컴퓨터의 계산 성능
머신러닝 알고리즘으로의 대체

02 History

A bit of history...

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning

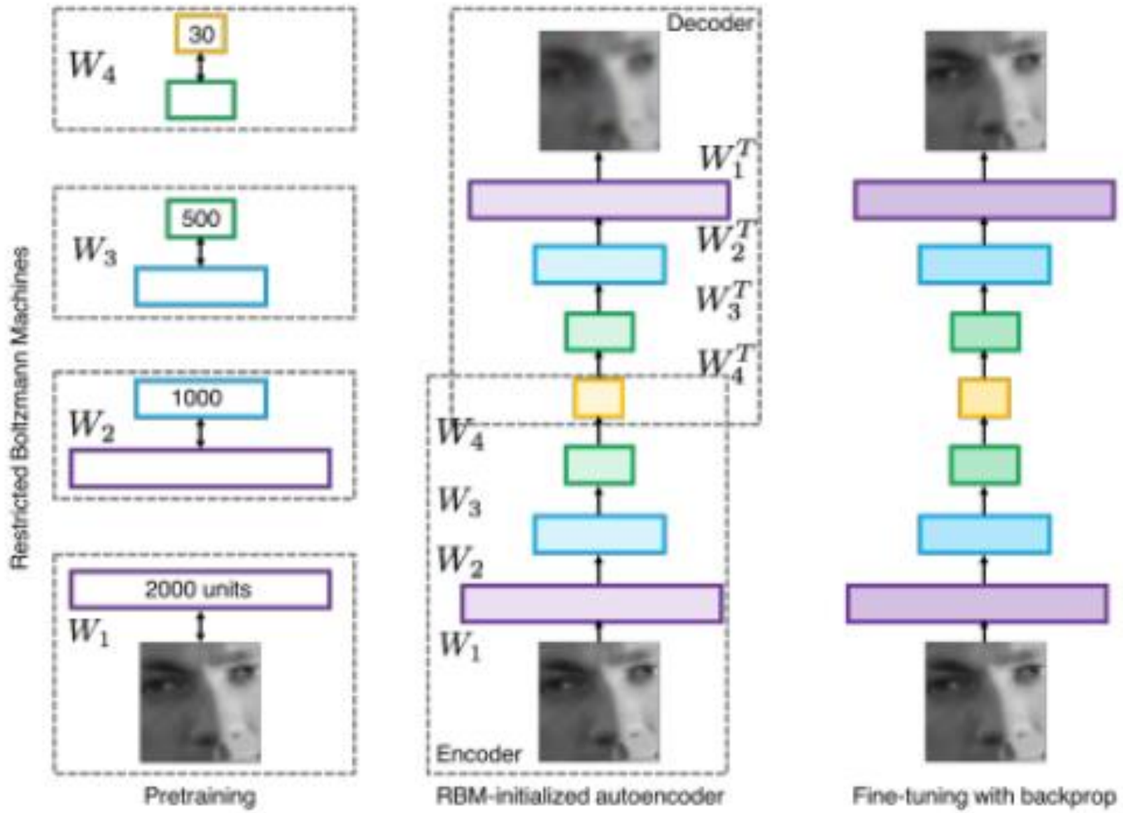


Illustration of Hinton and Salakhutdinov 2006 by Lane Mcintosh, copyright CS231n 2017

02 History

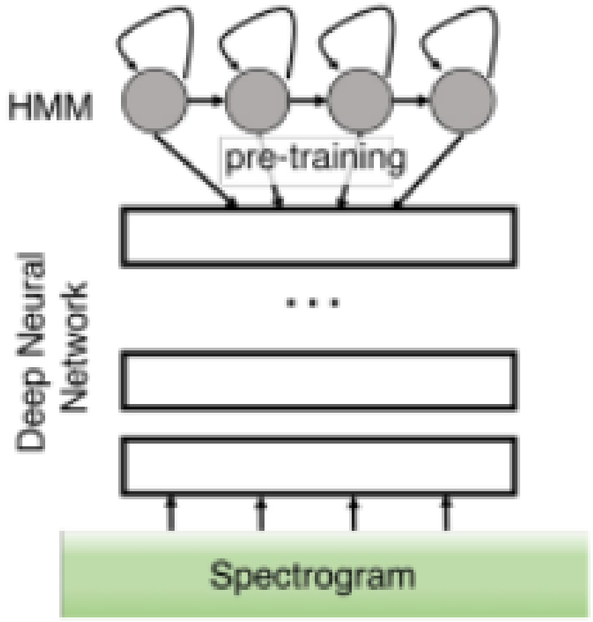
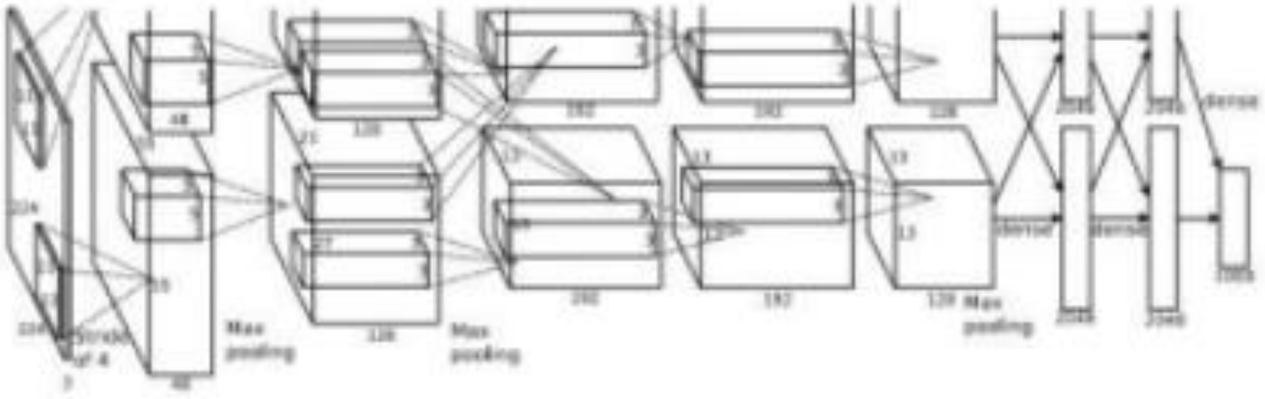
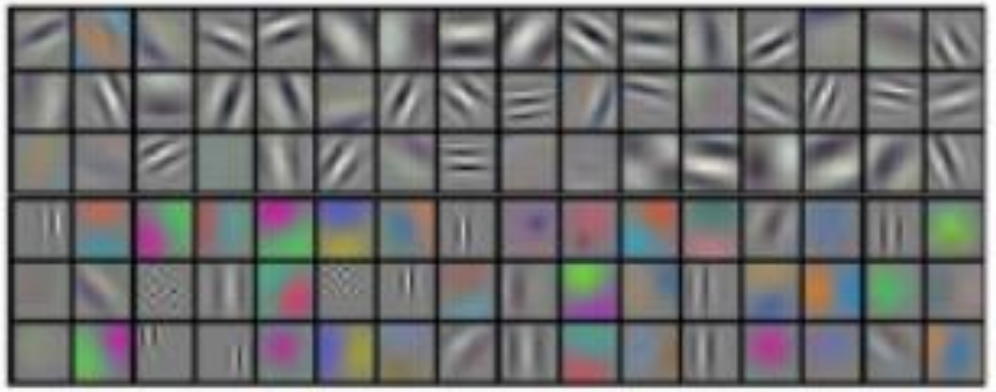


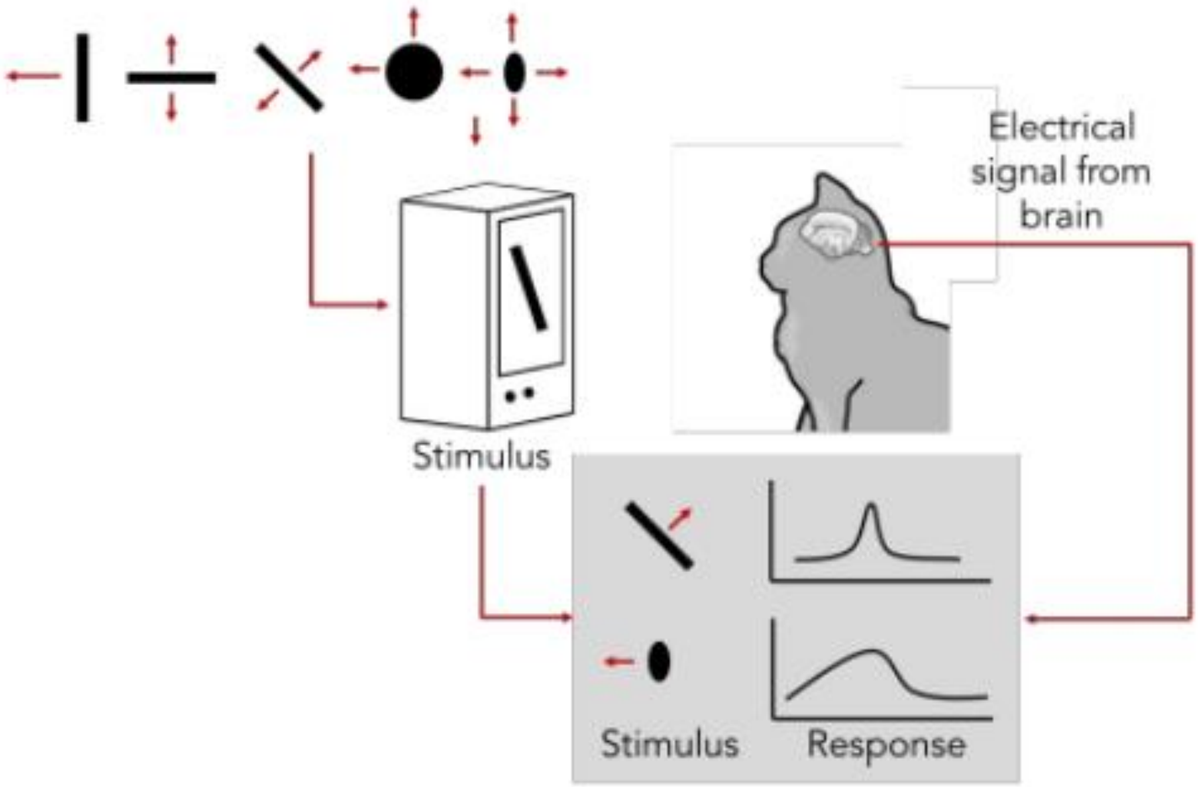
Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017



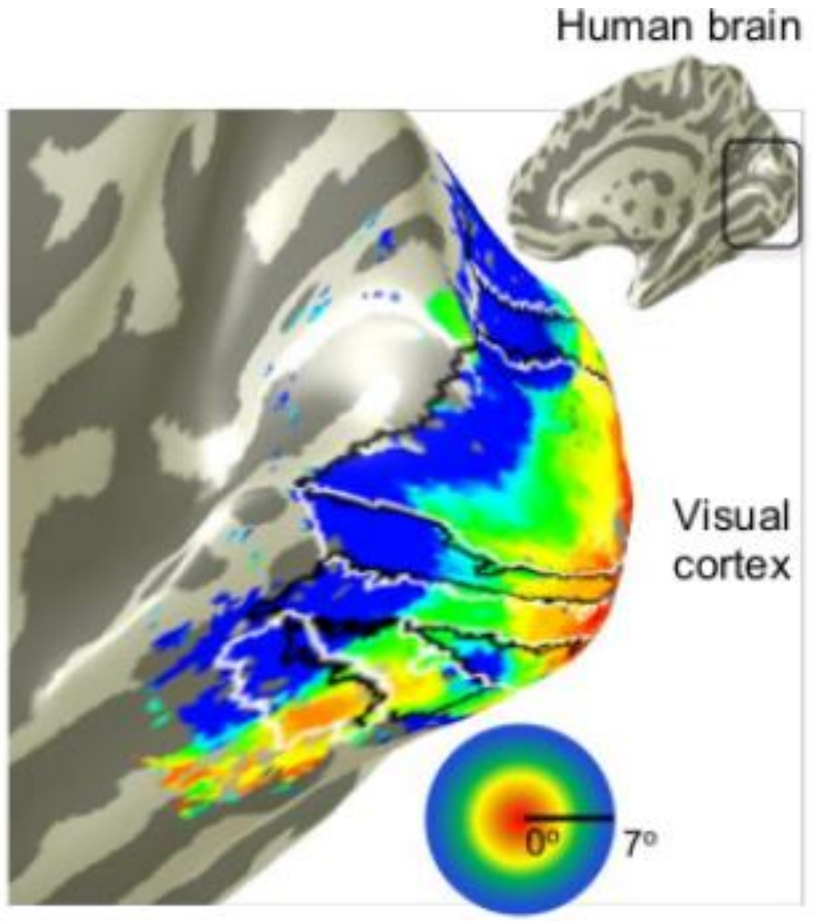


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

02 History



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



Retinotopy images courtesy of Jesse Gomez in the Stanford Vision & Perception Neuroscience Lab.

02 History

Hierarchical organization

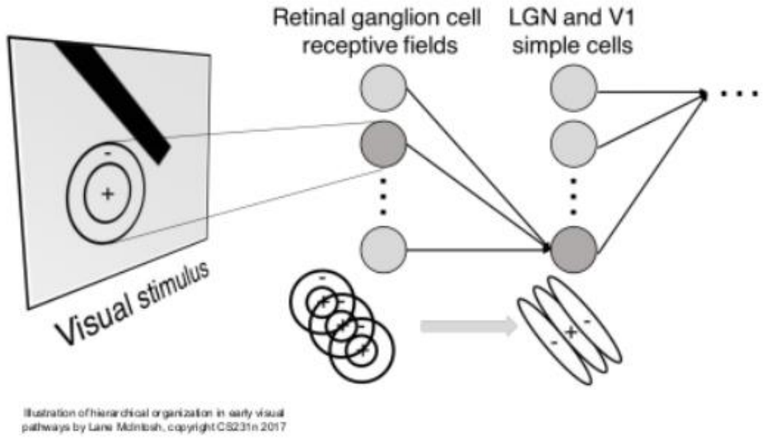
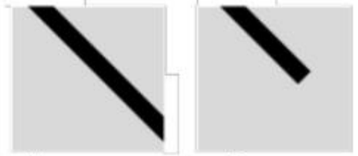


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

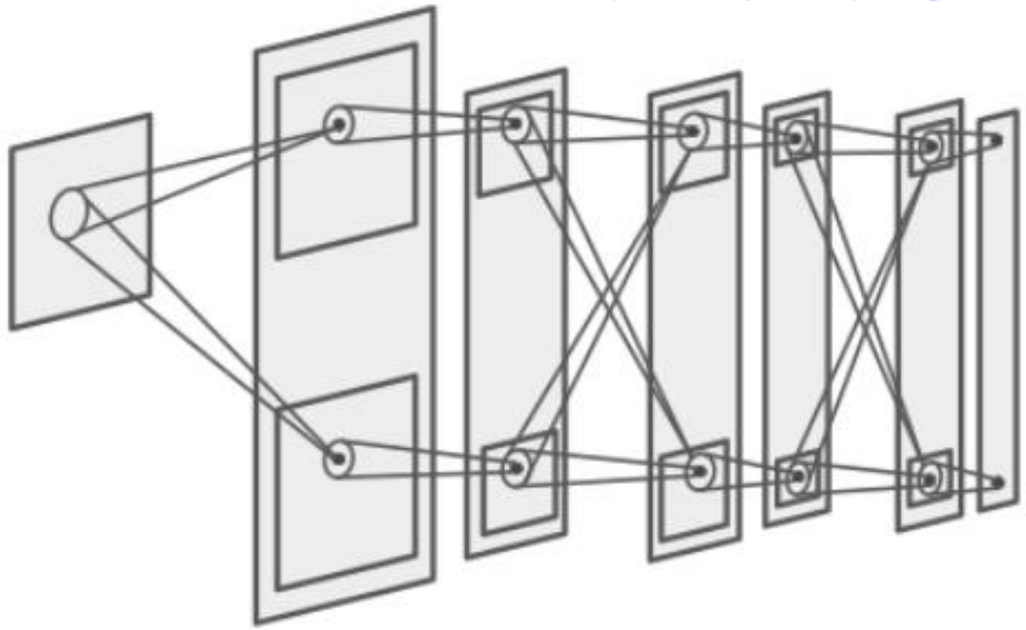
Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point

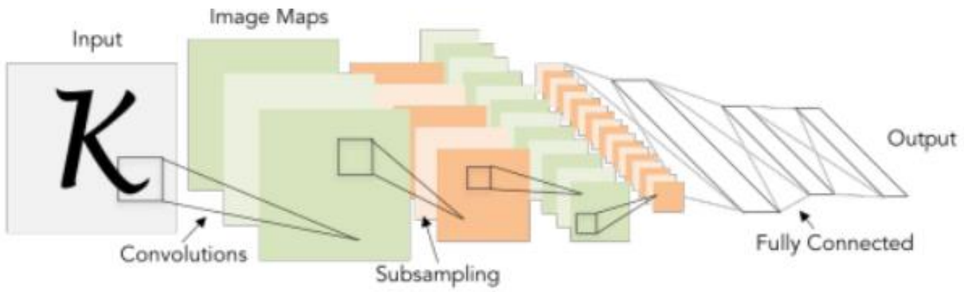


Neocognitron
[Fukushima 1980]

"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling



02 History



LeNet-5

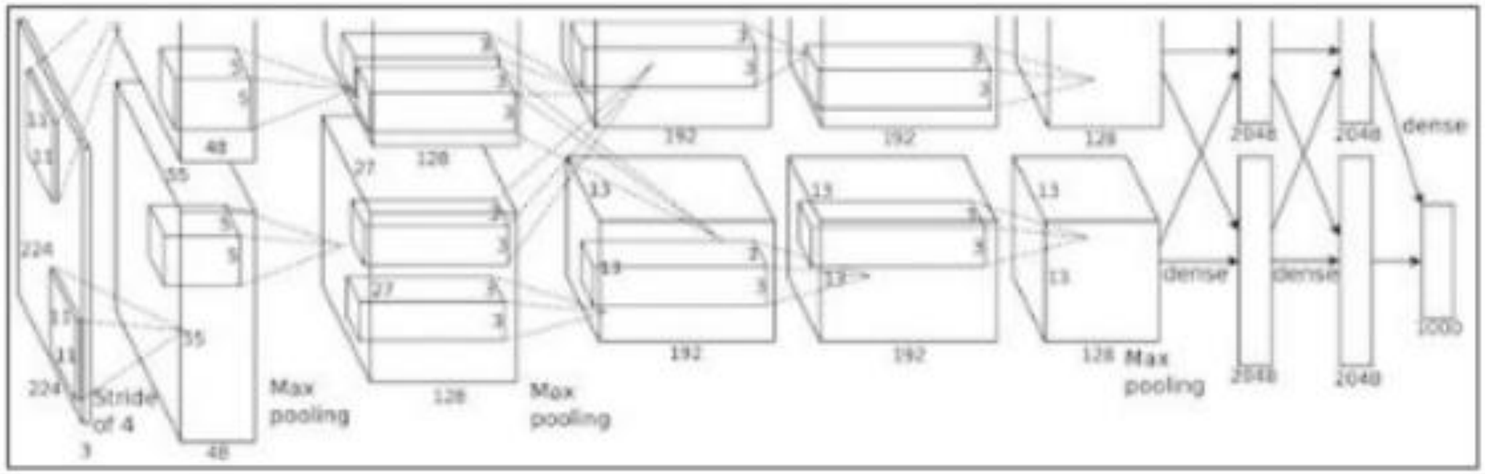
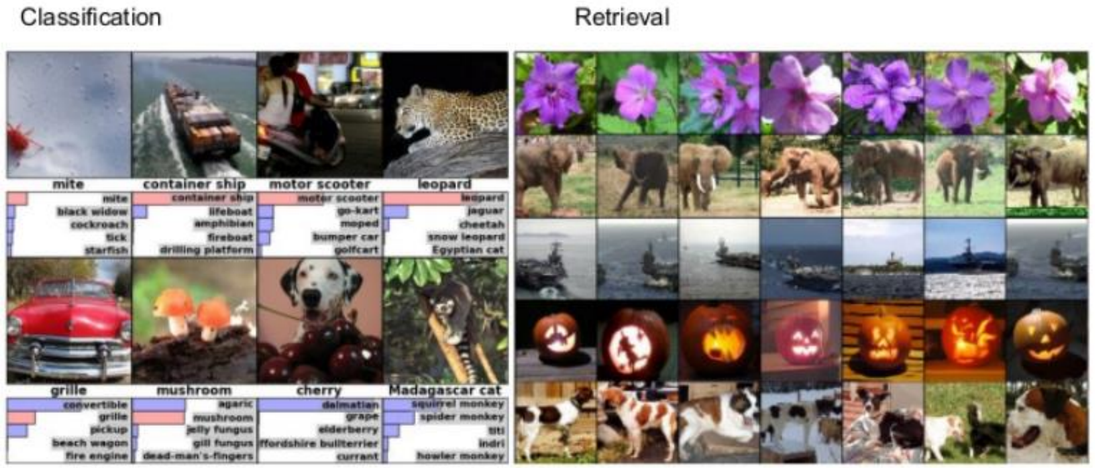


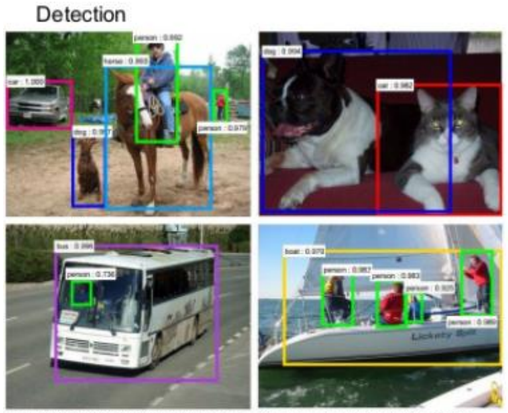
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

02 The Use of CNN



Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.



Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

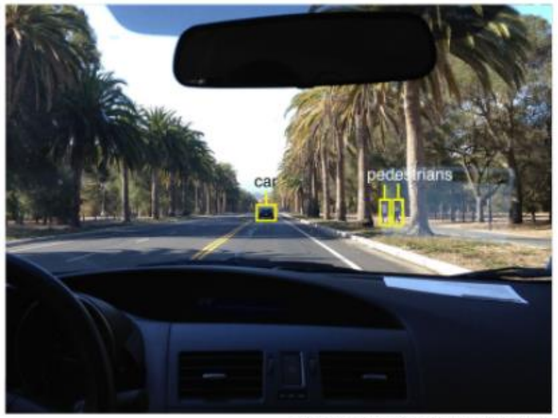


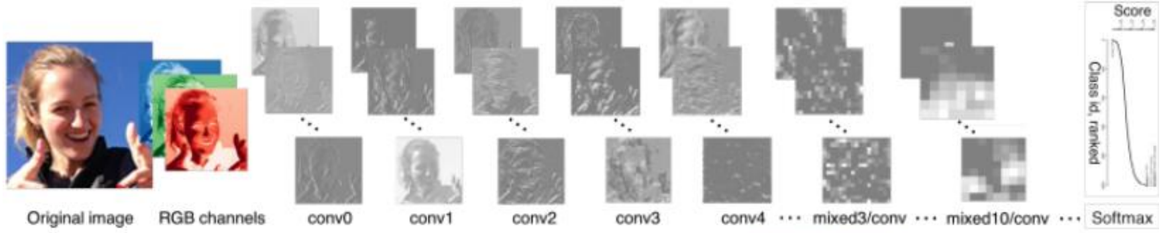
Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

This image by GBPublic_PR is licensed under CC-BY 2.0

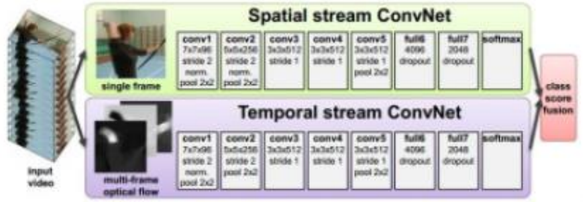
NVIDIA Tesla line
(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.



[Taigman et al. 2014]

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

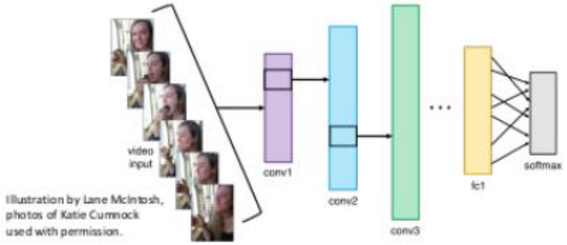
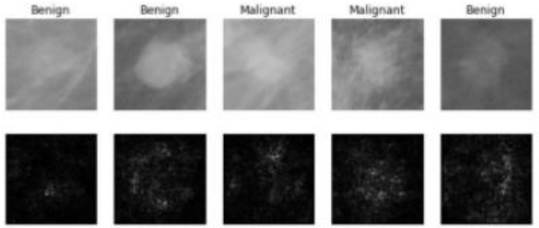


Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

02 The Use of CNN

Fast-forward to today: ConvNets are everywhere



[Levy et al. 2016]

Figure copyright Levy et al. 2016. Reproduced with permission.

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

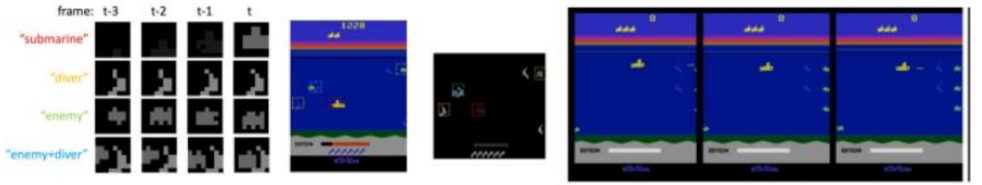
Photos by Lane McIntosh. Copyright CS231n 2017.

Whale recognition, Kaggle Challenge

Mnih and Hinton, 2010

[Toshev, Szegedy 2014]

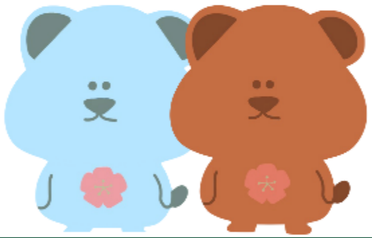
Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.



[Guo et al. 2014]

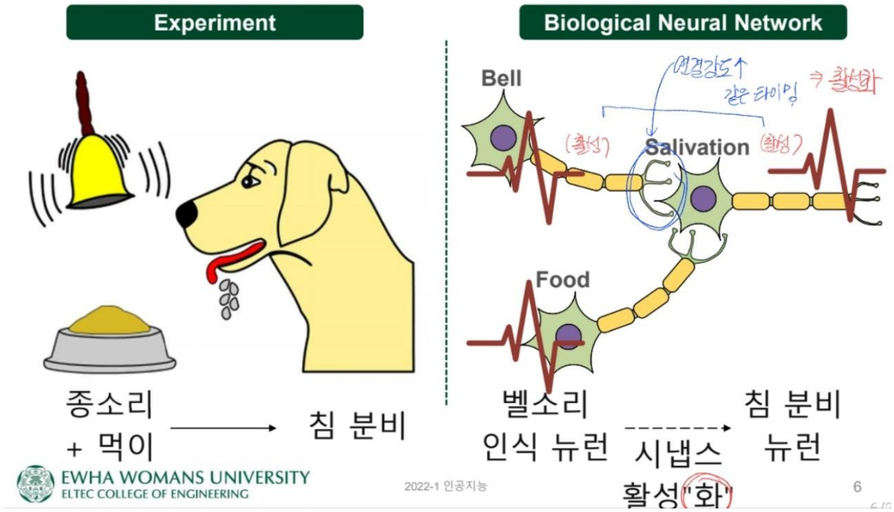
Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

CNN

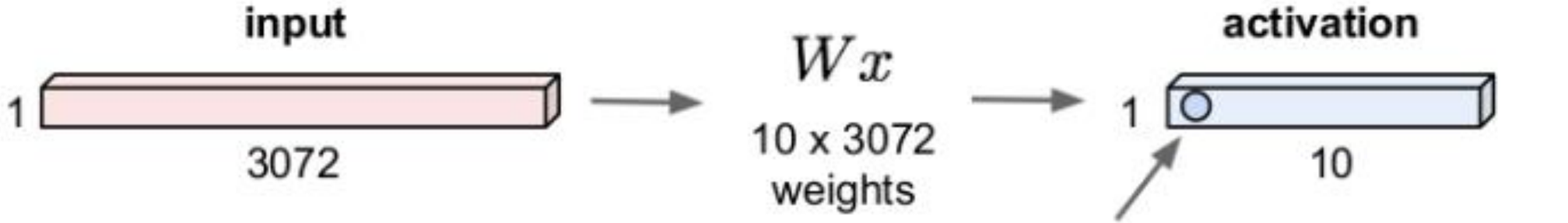


EWHA EURON

01 FC Layer



32x32x3 image -> stretch to 3072 x 1



1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

02 Convolution Layer

1. Convolution

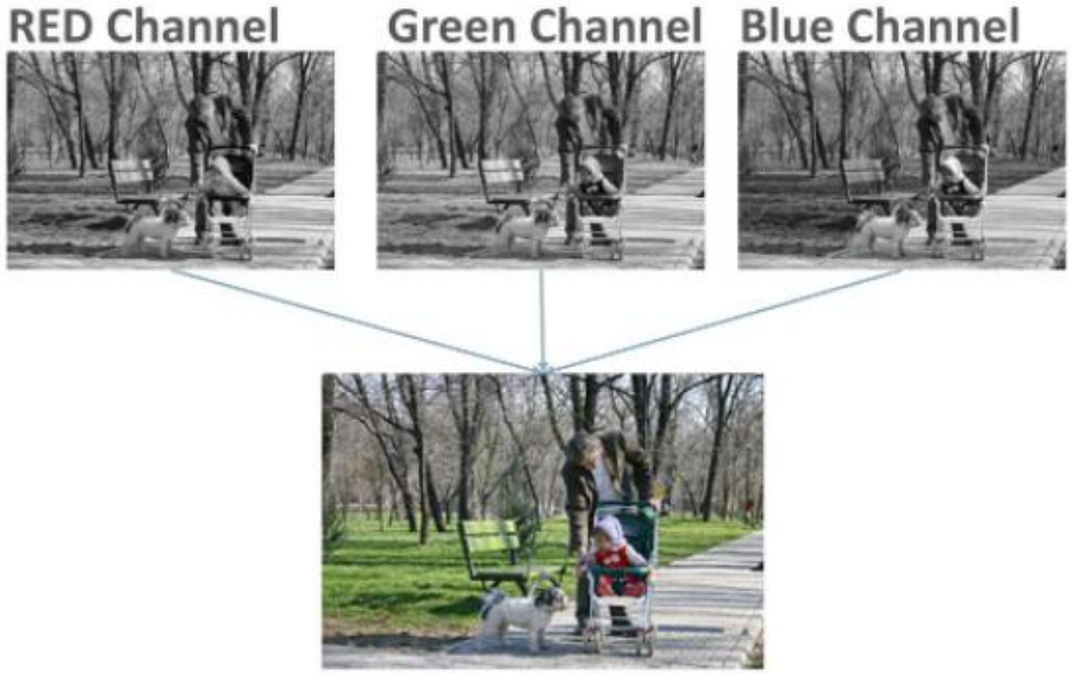
1	1	1	0	0
0	1	1	1	0
0	0	$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$
0	0	$1_{\times 0}$	$1_{\times 1}$	$0_{\times 0}$
0	1	$1_{\times 1}$	$0_{\times 0}$	$0_{\times 1}$

Image

4	3	4
2	4	3
2	3	4

Convolved Feature

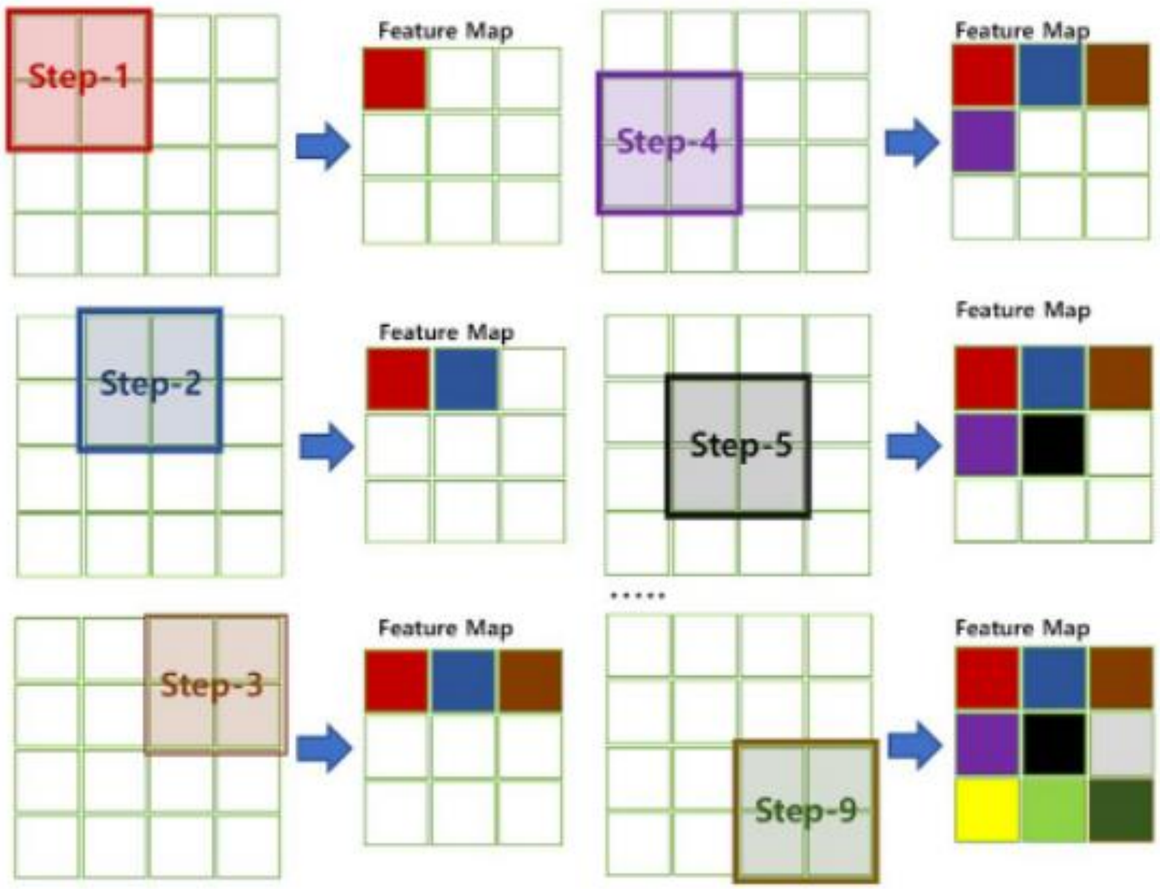
2. Channel



32 * 32 * 3

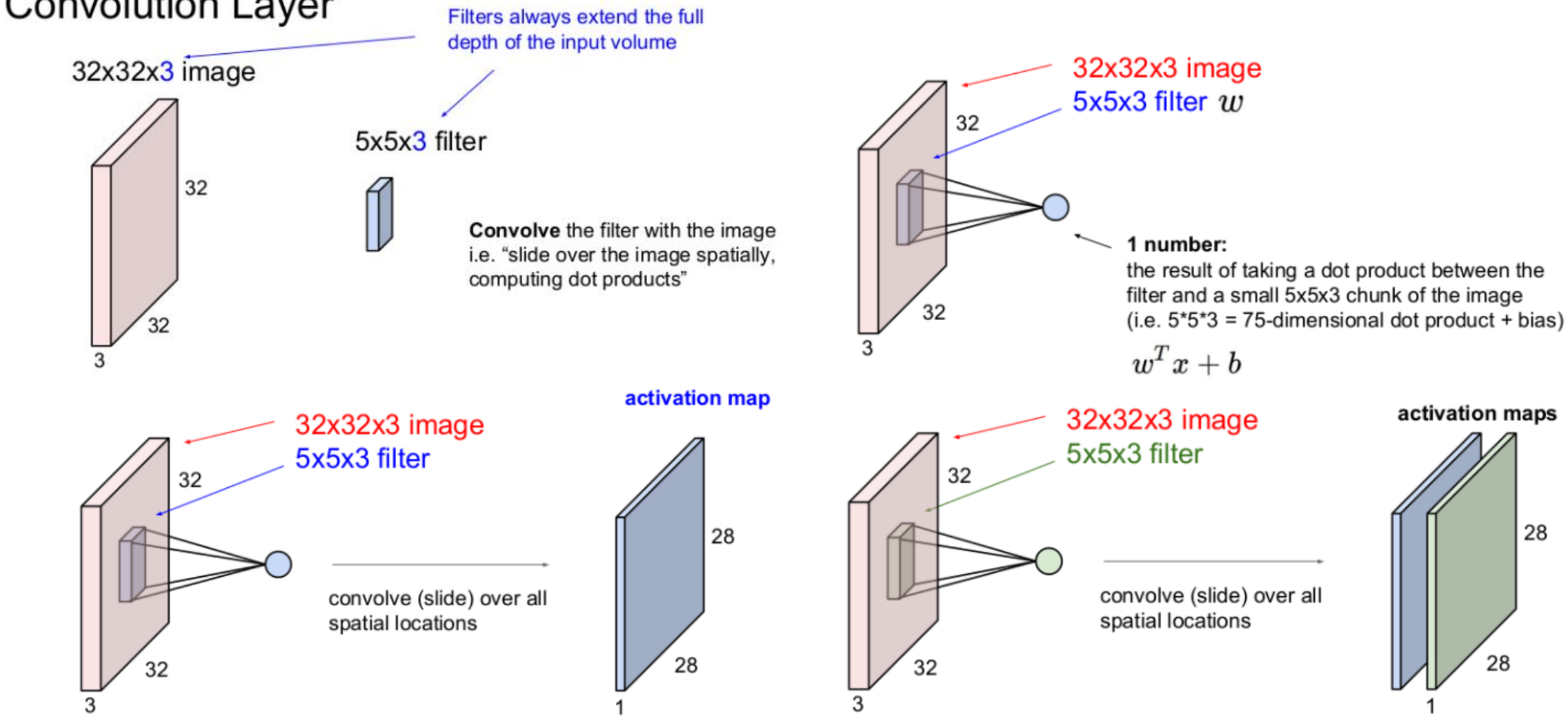
02 Convolution Layer

3. Filter = Kernel

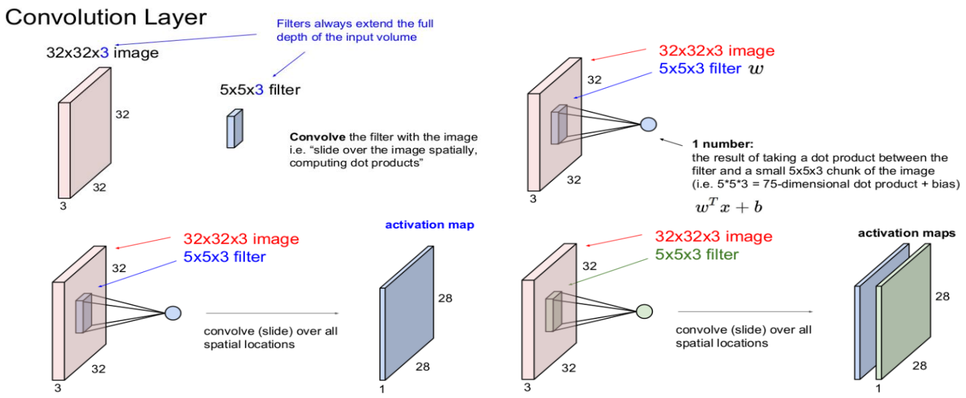


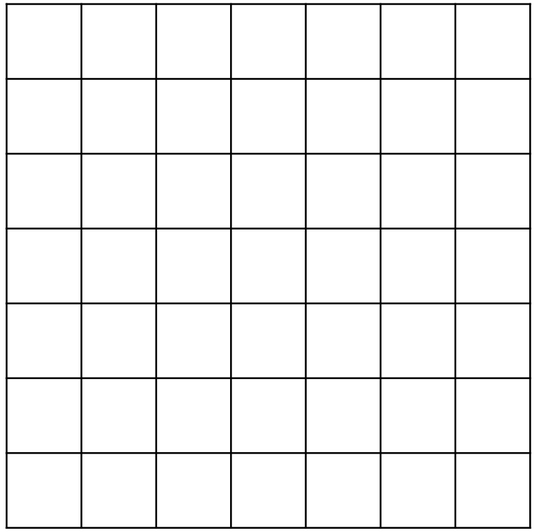
02 Convolution Layer

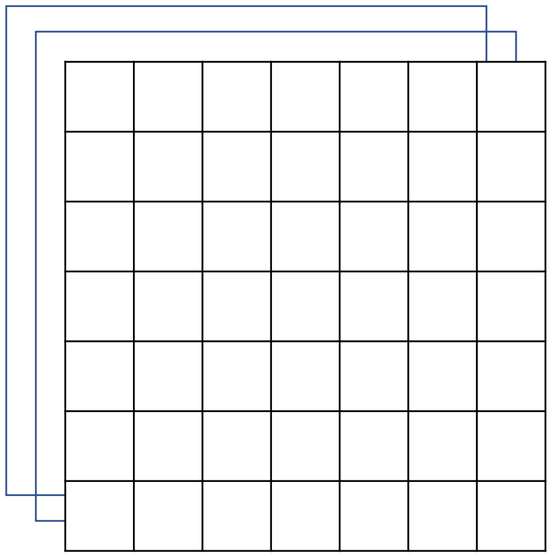
Convolution Layer



02 Convolution Layer

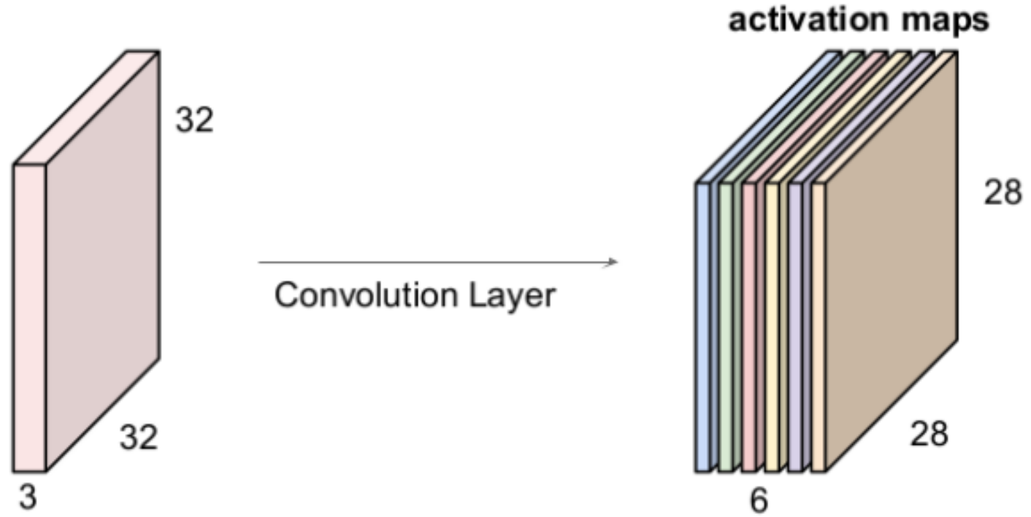






02 Convolution Layer

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

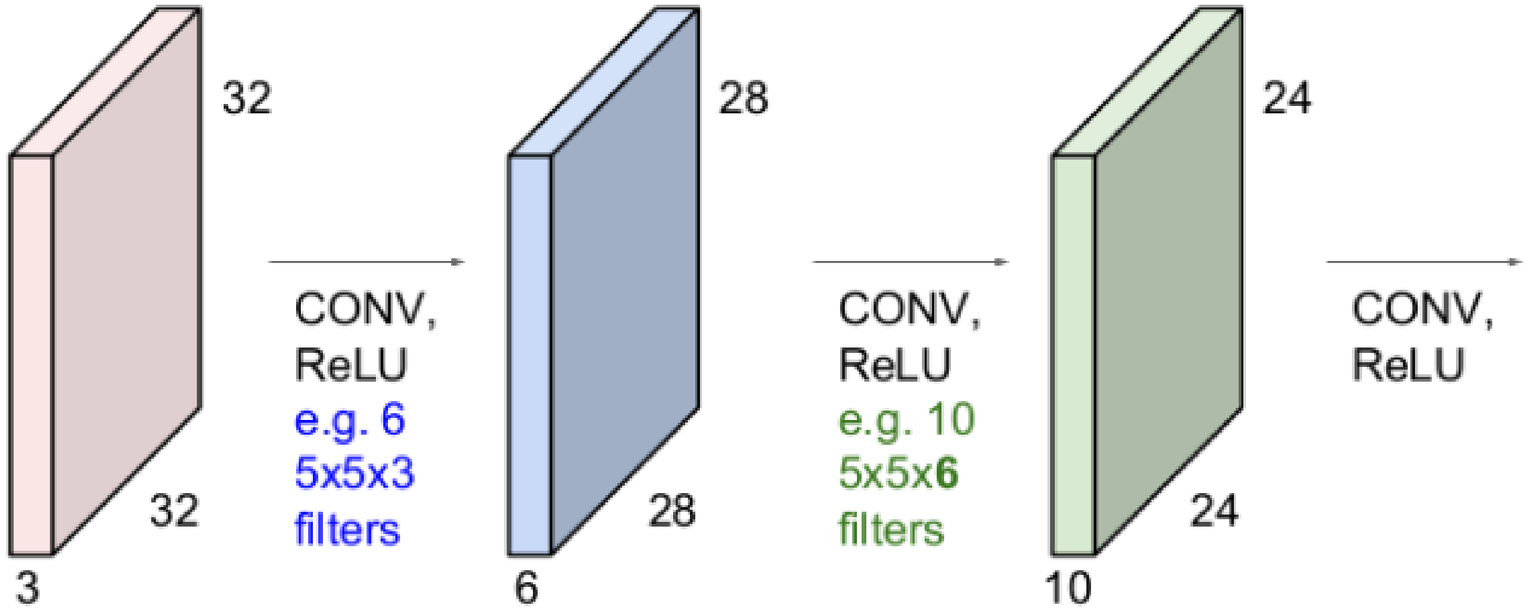


We stack these up to get a "new image" of size 28x28x6!

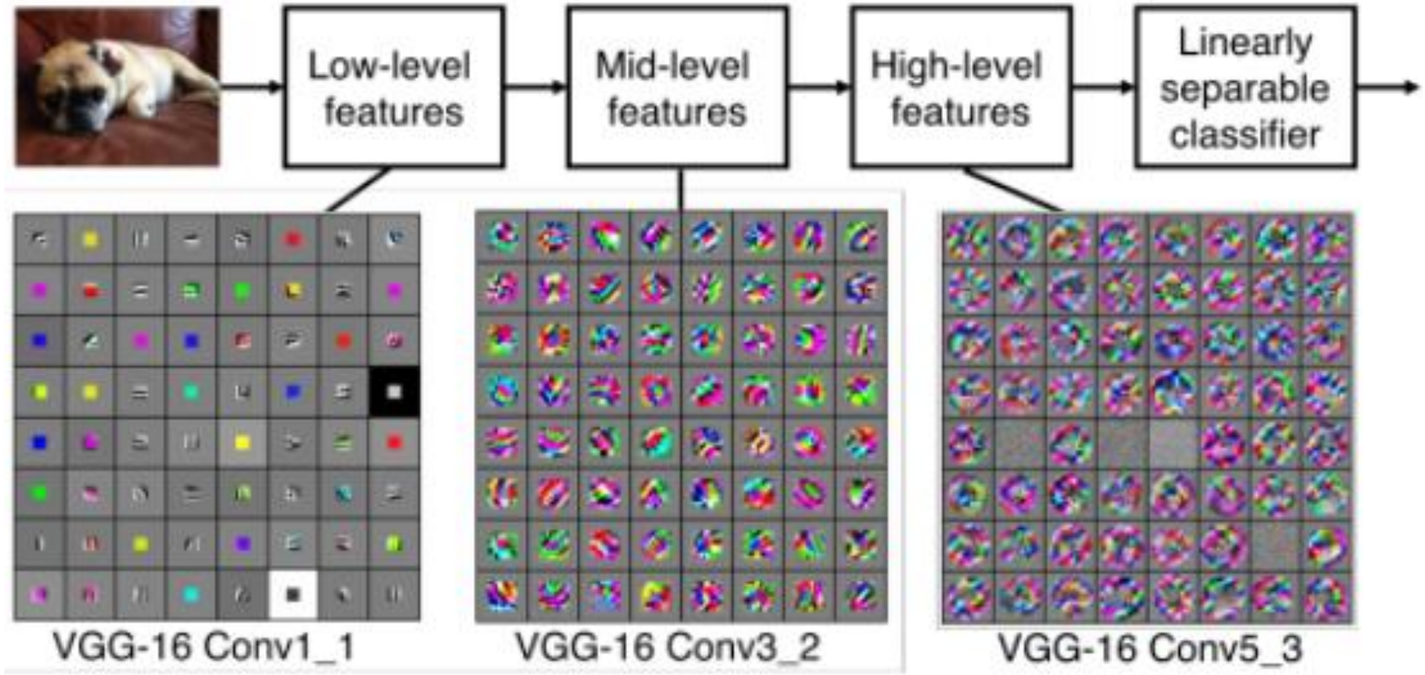
Q.
Image size 64 * 64 * 3
Filter size 3 * 3 * 3 -> 4개, 1칸 sliding
What is the size of output activation map?

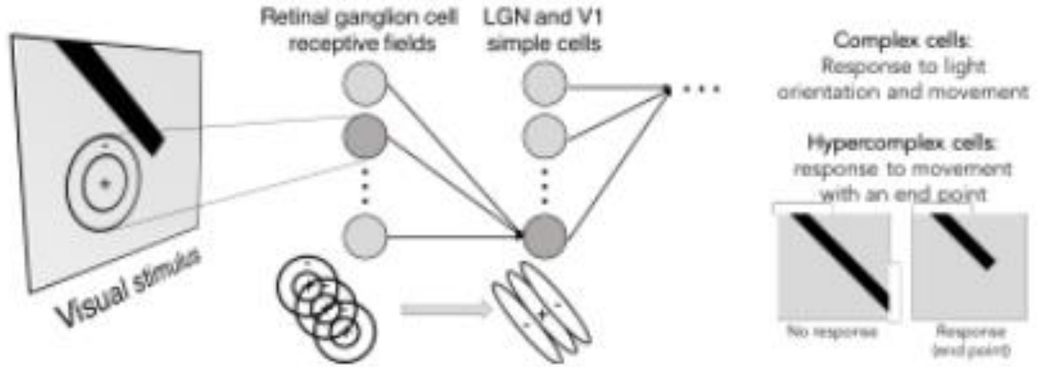
02 Convolution Layer

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

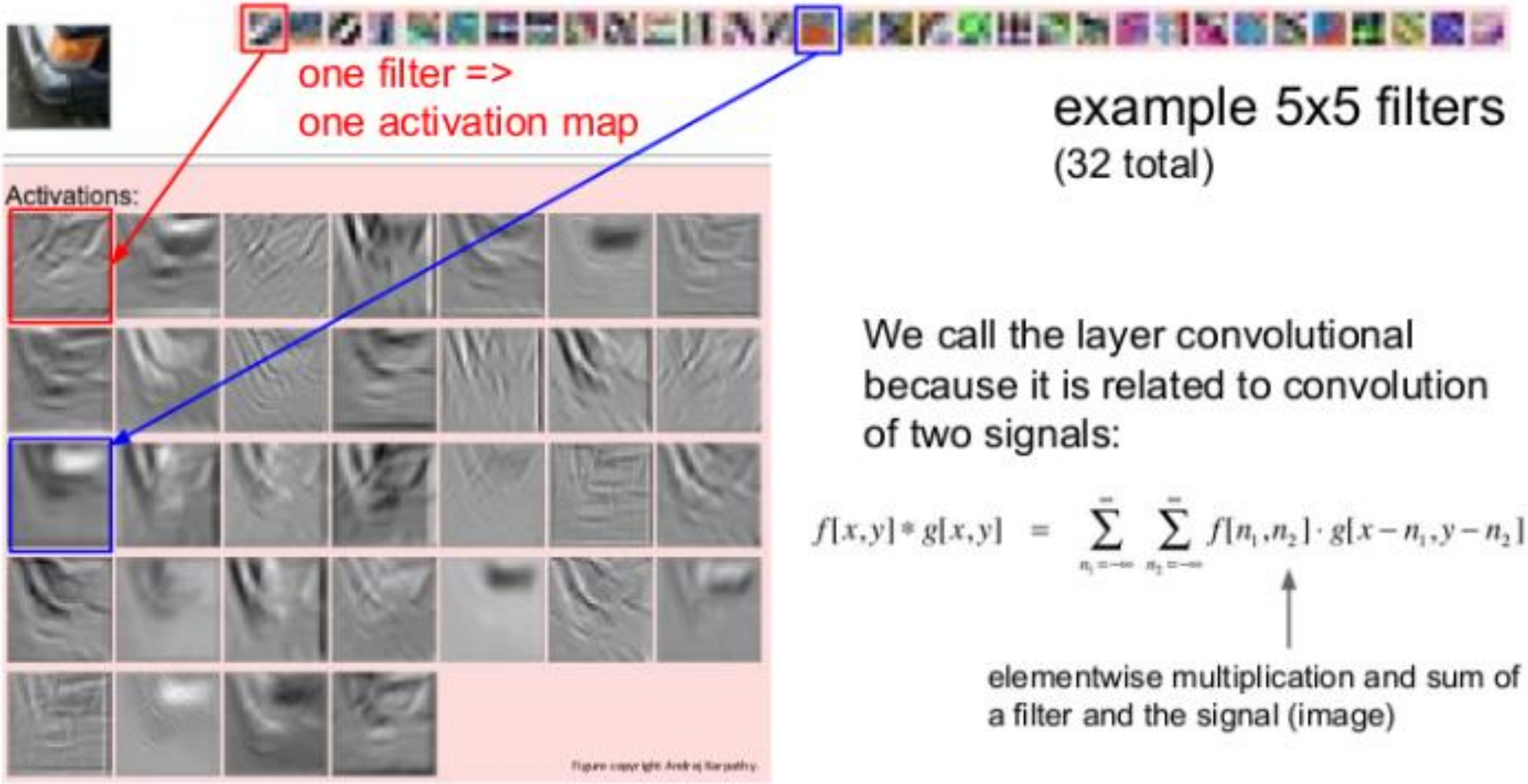


03 Filter


Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3


Retinal ganglion cell receptive fields
LGN and V1 simple cells
Visual stimulus
Complex cells:
Response to light orientation and movement
Hypercomplex cells:
response to movement with an end point
No response
Response (end point)

03 Filter



example 5x5 filters
(32 total)

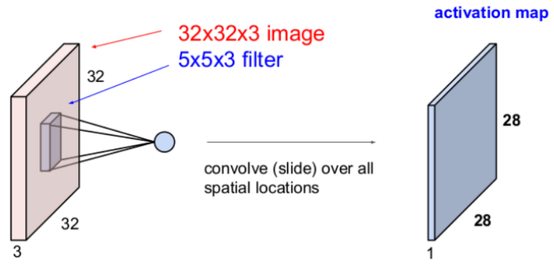
We call the layer convolutional because it is related to convolution of two signals:

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

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

03 Filter

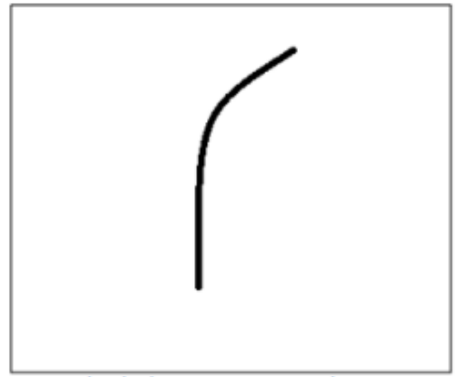
A closer look at spatial dimensions:



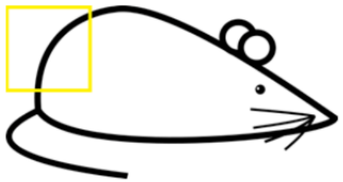
Original image

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

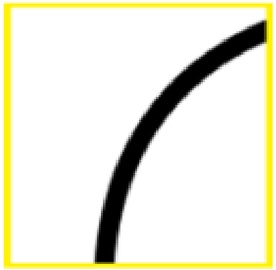
Pixel representation of filter



Visualization of a curve detector filter



Visualization of the filter on the image



Visualization of the receptive field

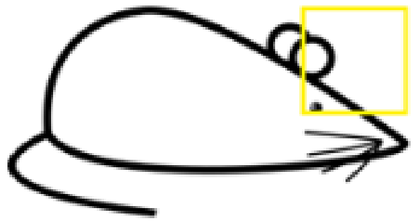
0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

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



Visualization of the filter on the image

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

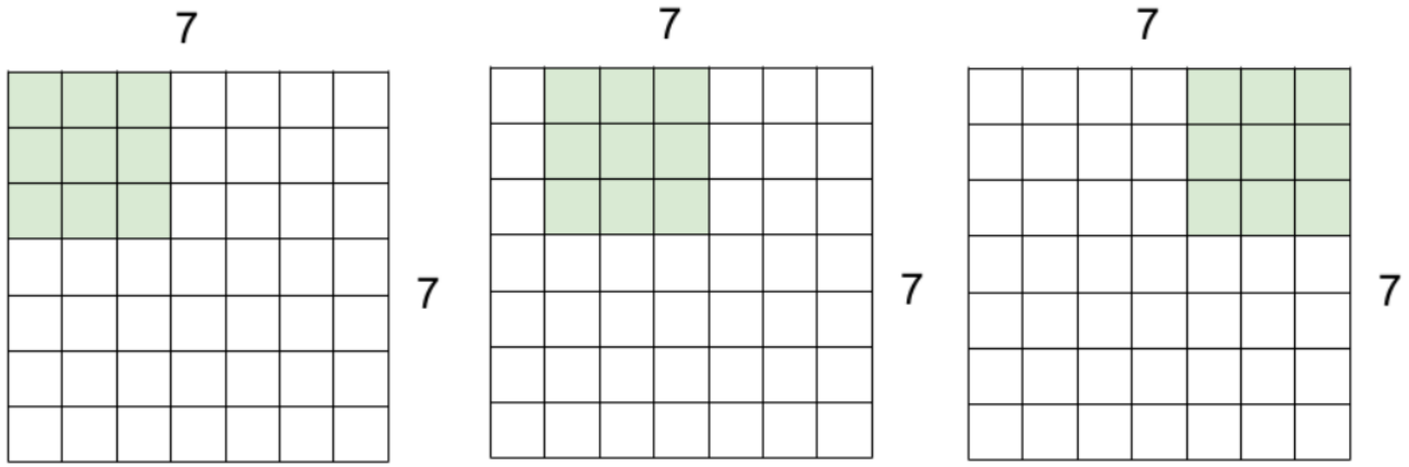
Pixel representation of receptive field

*

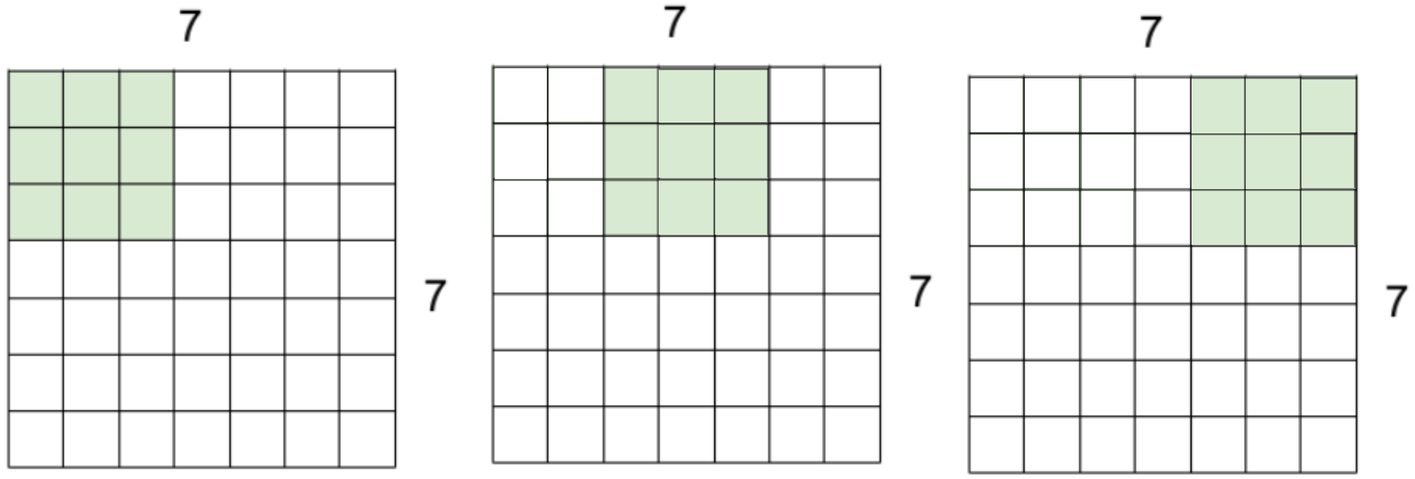
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

03 Filter

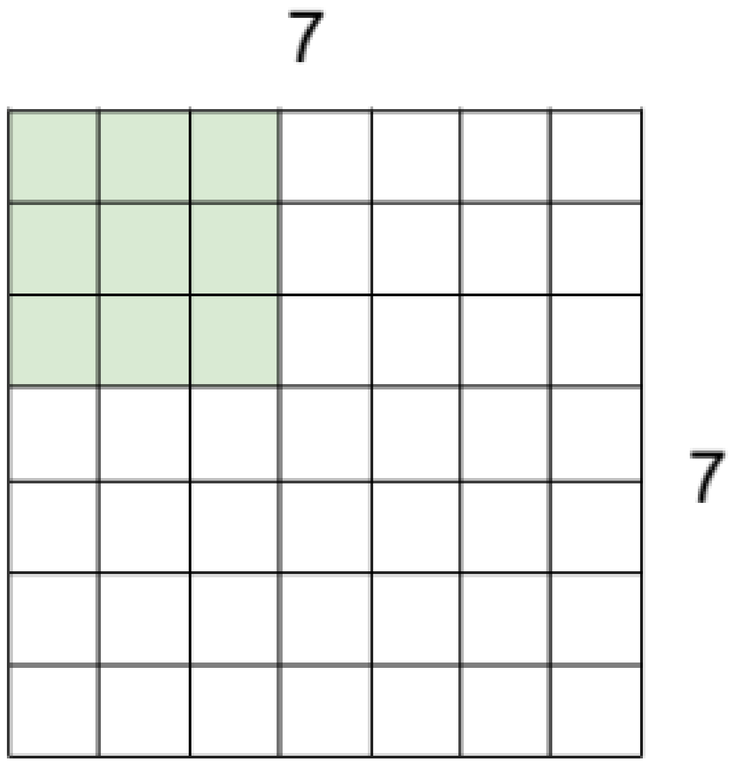


=> 5 x 5 output



=> 3 x 3 output

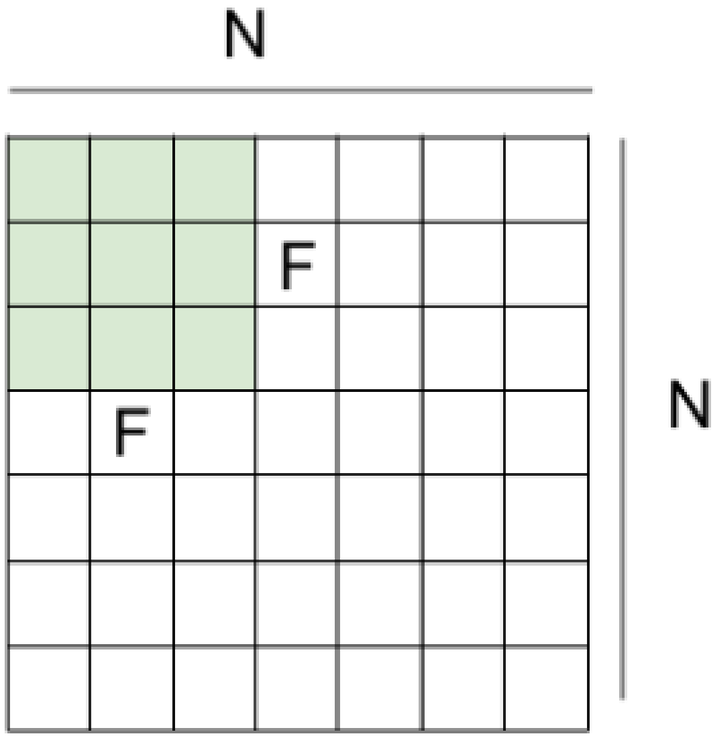
03 Filter



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

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

03 Filter



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

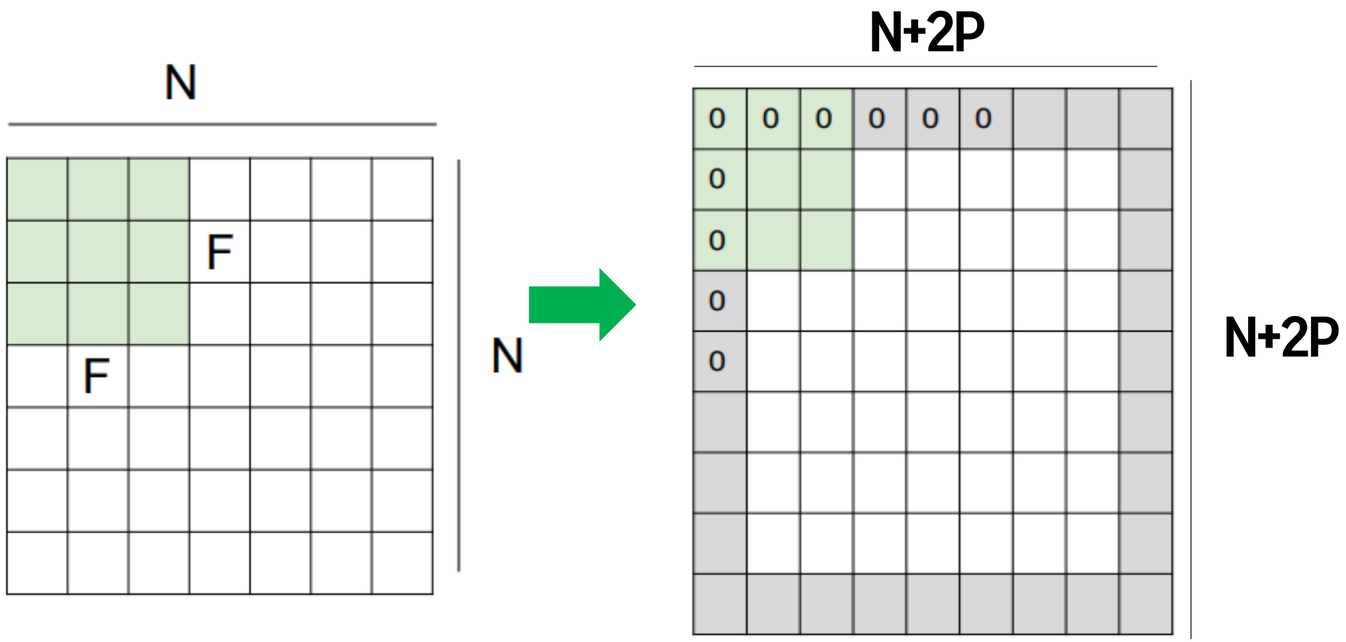
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

04 Padding

- Convolution 연산에서 문제점

- Input에서 corner 부분의 pixel은 필터가 다른 부분보다 덜 적용되어 정보를 잃음
- convolution 연산을 거치면서 output의 크기가 점점 작아짐

➡ image의 테두리에 특정 숫자를 채우는 **Padding** 하기!

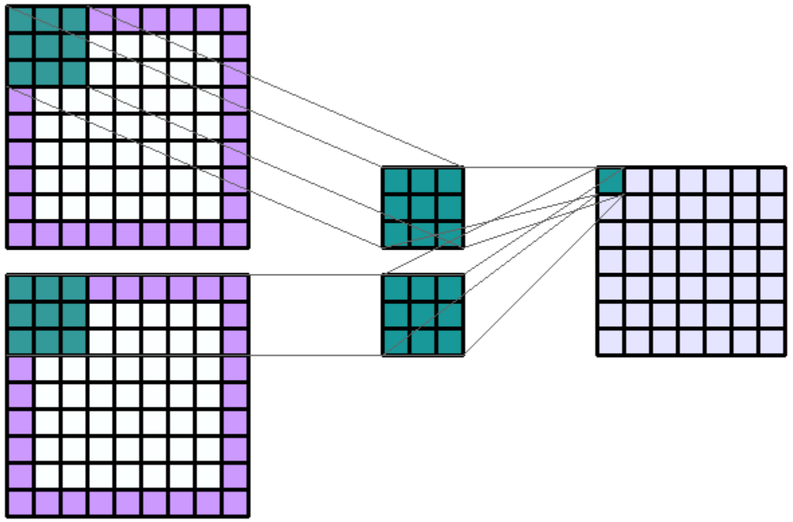


(N - F) / stride + 1

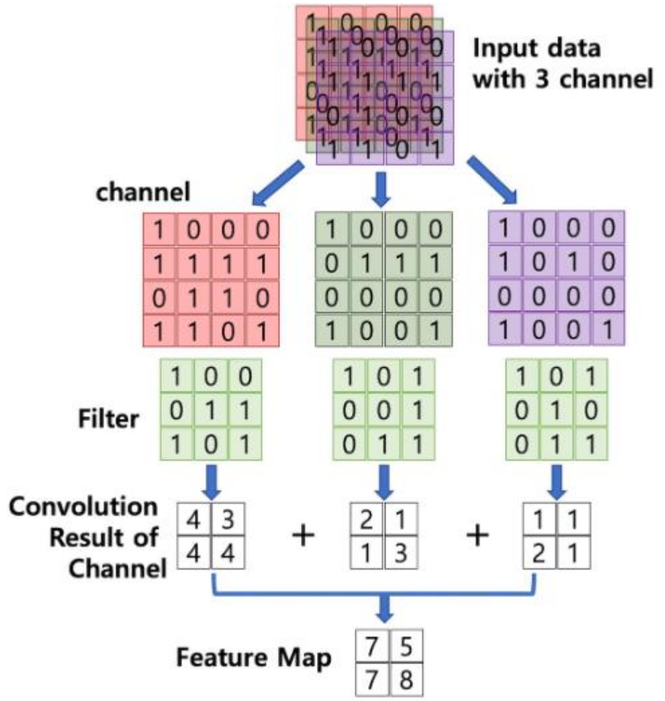
(N-F+2P)/stride +1

04 Padding

Padding: input의 테두리에 특정 숫자로 채우는 방법으로 zero-padding이 흔히 쓰임
그 외 가장자리 값 그대로 복사하는 방법, interpolation 하는 방법이 있음



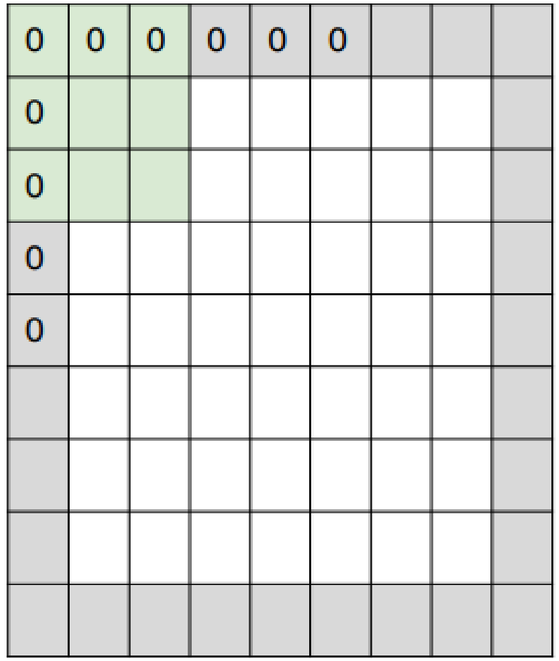
- Input: 7x7x2
- Padding:1x1
- Stride:1
- Filter:3x3 1개
- Output:7x7x1 (input과 같은 크기)
  :(7-3+2)/1+1,Depth는 Filter 갯수



* 하나의 Filter를 여러 채널(depth>1)인 이미지에 적용하면
  여러 개의 결과를 더함 = output depth는 1
* 두개의 Filter를 사용하면 output이 두개 생기므로 output depth는 2

04 Padding - 예제

Input(N): 7x7x1
Filter(F): 3x3 1개
Stride: 1
Padding:1

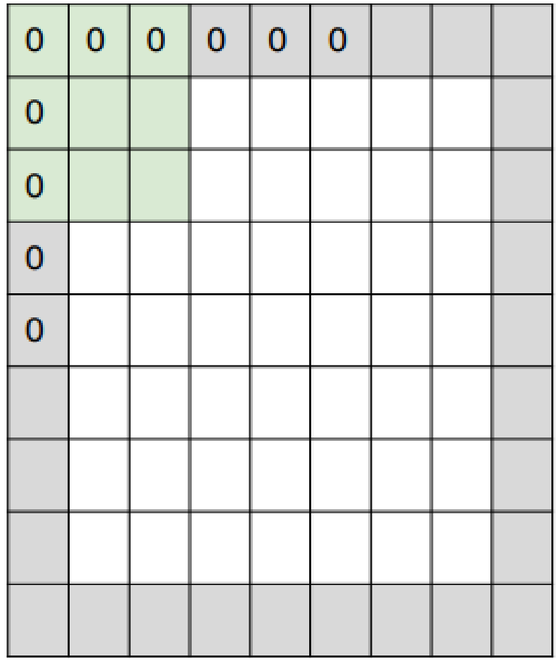


(N-F+2P)/Stride+1

Output Size?

04 Padding - 예제

Input(N): 7x7x1
Filter(F): 3x3 1개
Stride: 1
Padding:1



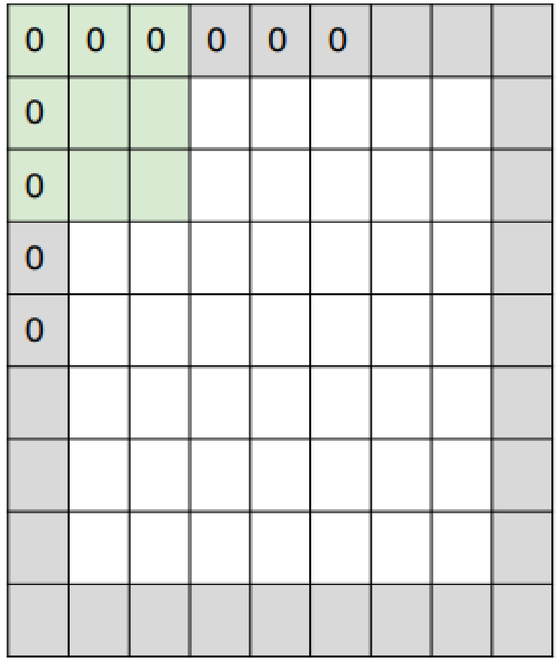
(N-F+2P)/Stride+1

Output Size? (7-3+2x1)/1+1 = 7

Number of parameters?

04 Padding - 예제

Input(N): 7x7x1
Filter(F): 3x3 1개
Stride: 1
Padding:1



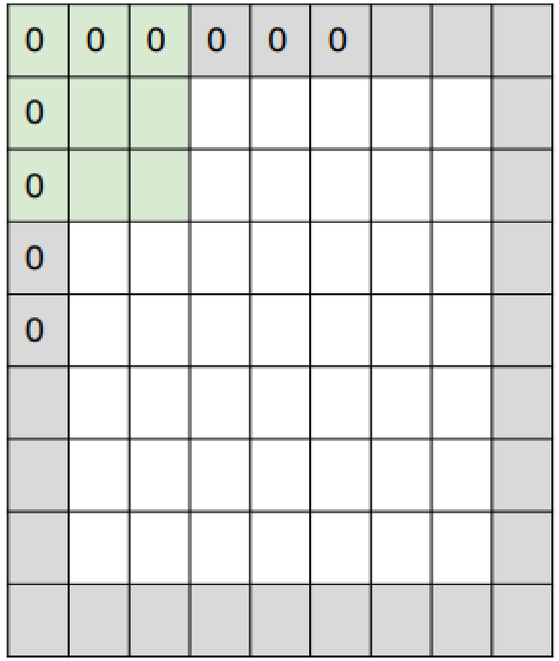
(N-F+2P)/Stride+1

Output Size? (7-3+2x1)/1+1 = 7

Number of parameters?
(Filter폭xFilter높이x채널개수 +1(bias)xFilter개수
(3x3x1+1)x1 =10

04 Padding - 예제

Input(N): 7x7x1
Filter(F): 3x3 1개
Stride: 1
Padding:1



(N-F+2P)/Stride+1

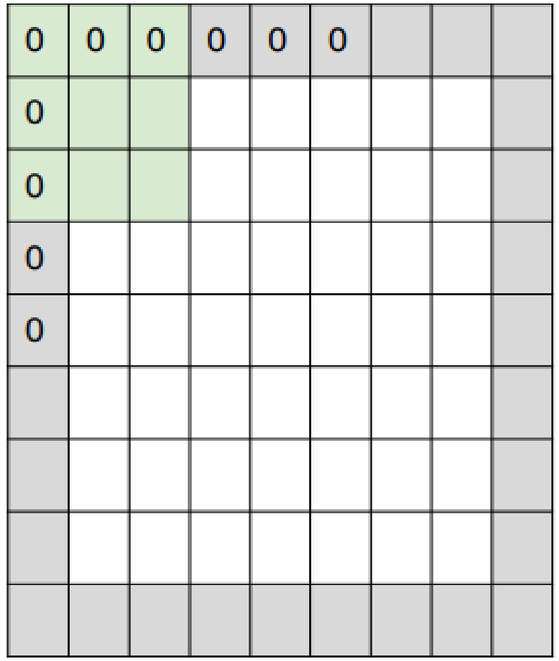
Output Size? (7-3+2x1)/1+1 = 7

Number of parameters?
(Filter폭xFilter높이x채널개수 +1(bias)xFilter개수
(3x3x1+1)x1 =10

input과 size 같게 하려면 padding과 F의 관계는?

04 Padding - 예제

Input(N): 7x7x1
Filter(F): 3x3 1개
Stride: 1
Padding:1



(N-F+2P)/Stride+1

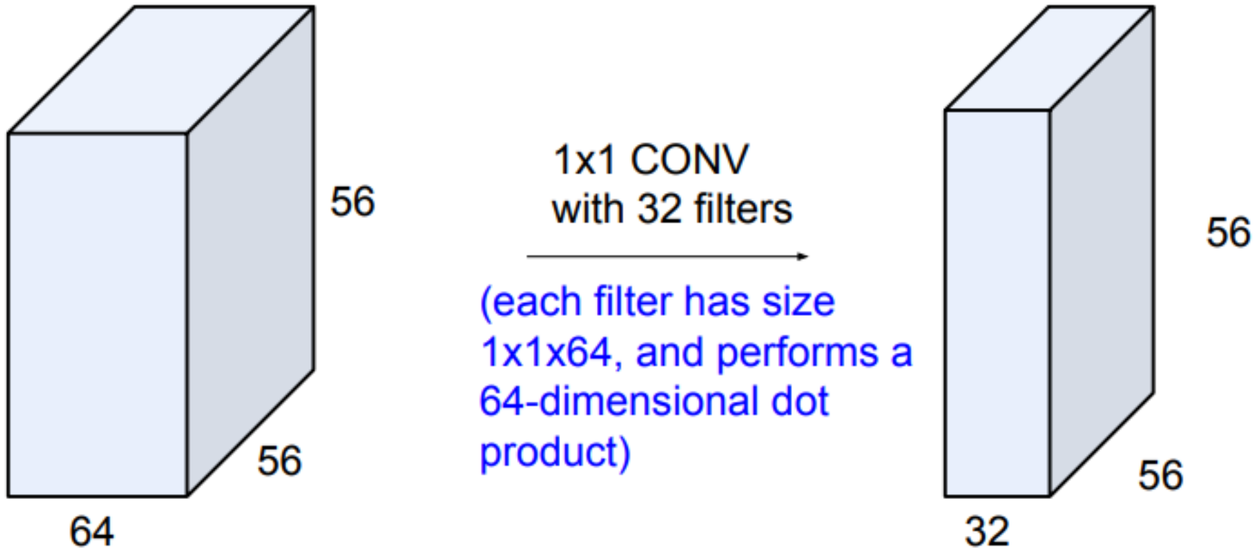
Output Size? (7-3+2x1)/1+1 = 7

Number of parameters?
(Filter폭xFilter높이x채널개수 +1(bias)xFilter개수
(3x3x1+1)x1 =10

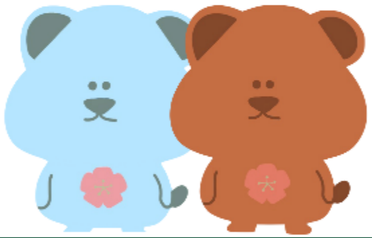
input과 size 같게 하려면 padding과 F의 관계는?
P=(F-1)/2=3

05 1X1 Convolution

1x1 Convolution: 1X1 크기의 convolution Filter를 사용한 convolution layer
* 장점
1) Channel 수 조절: parameter 수가 급격하게 증가하는것을 예방하여 channel 개수 증가 가능
2) 연산량 감소(Efficient)
3) 비선형성(Non-linearity): ReLu Activation을 더 사용할수 있어져 모델의 비선형성 좋아짐

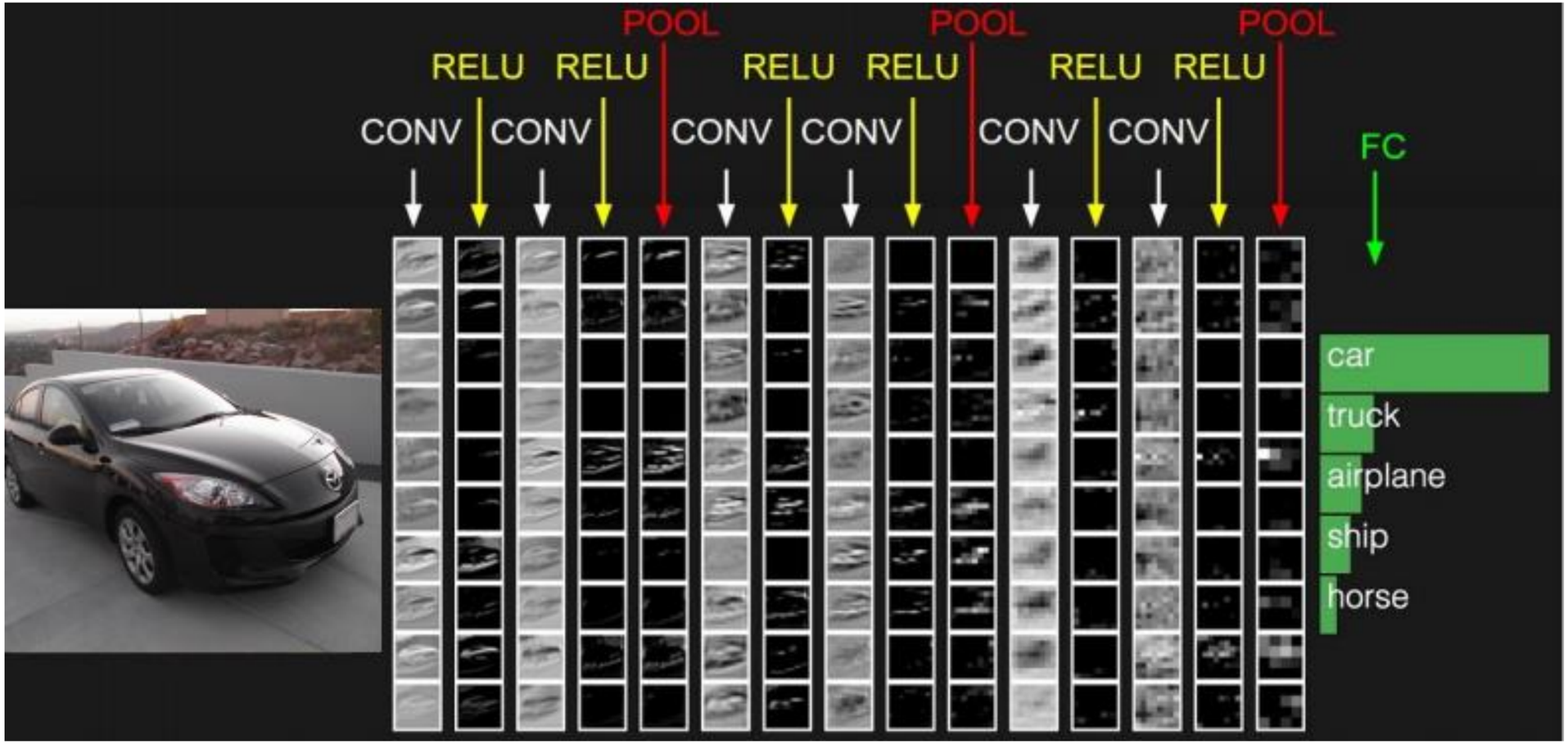


Pooling Layer



EWHA EURON

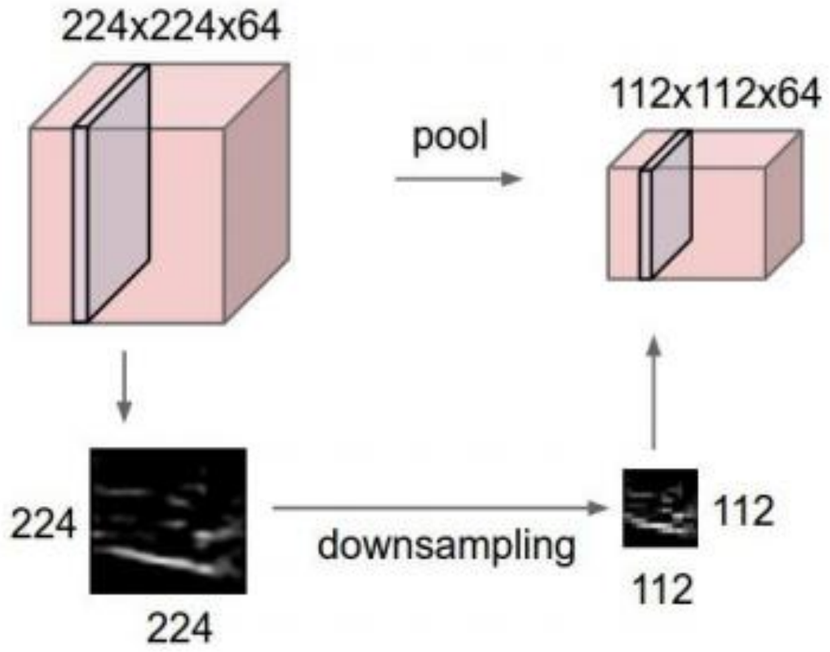
01 Pooling layer



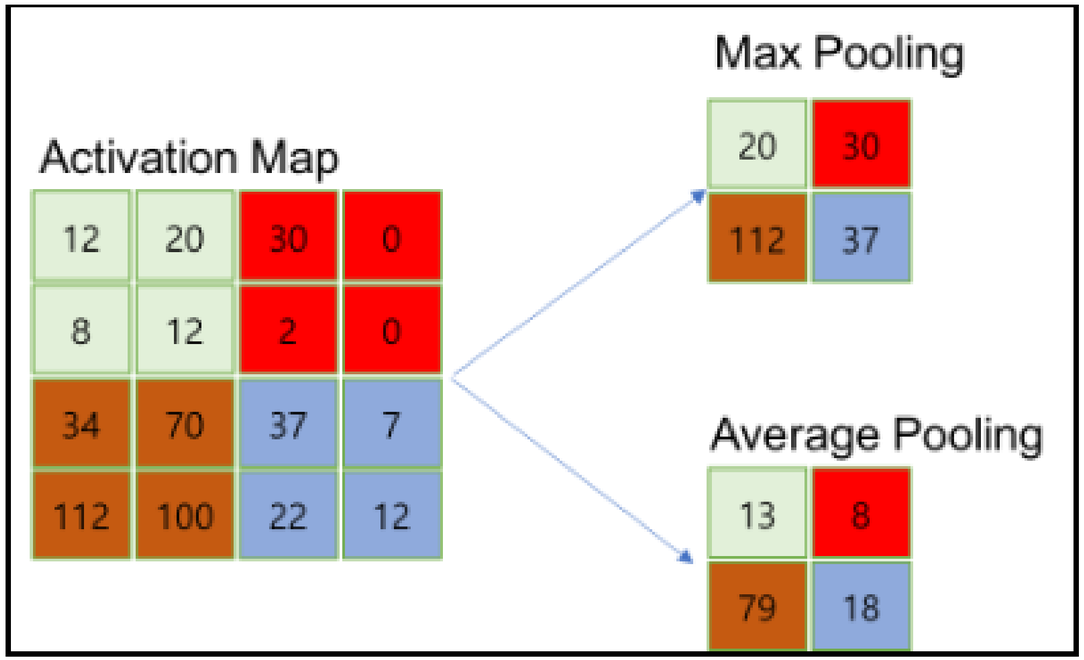
01 Pooling layer

Pooling layer

- Convolution layer의 출력 데이터(activation map)를 입력으로 받아서 출력 데이터를 downsampling을 하거나 특정 데이터를 강조함.
- 데이터가 작아져 파라미터 수를 줄임(overfitting↓)
- Max pooling(많이 사용됨), Average Pooling,Min Pooling 등이 있음



01 Pooling layer



Max Pooling

:각 영역에서 최댓값을 선택

Average Pooling

:각 영역의 평균값을 선택

Filter가 이미지의 어떤 영역에서 얼마나 활성되었는지가 중요하므로 값이 얼마나 클 수 있는지를 나타내는 **max pooling**을 주로 사용함

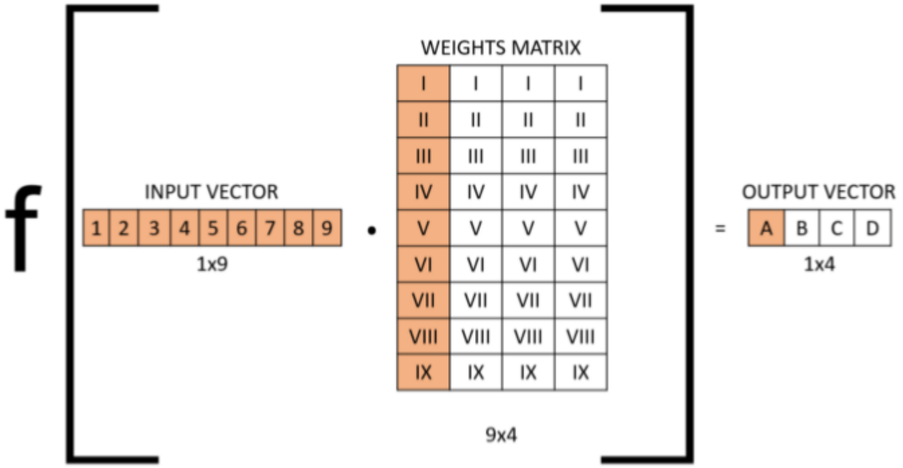
EWHA EURON

02 FC VS Conv layer

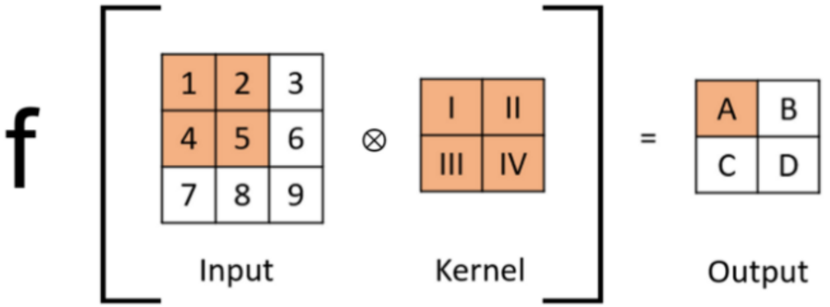
Fully Connected layer는 input image를 Flatten(평탄화)하는 과정을 거침
Convolutional layer는 input image의 원형이 그대로 보존됨

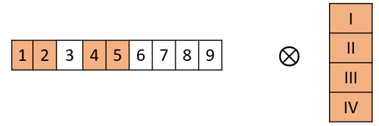
Fully Connected layer 모든 input이 weight와 연결되지만,
Convolutional layer는 Filter 부분과 만나는 일부의 input 부분만이 연결된다.

Fully Connected layer to Convolutional layer
- FC에서 가중치 W를 Conv의 Filter로 변환하기
- 1 forward pass → sliding



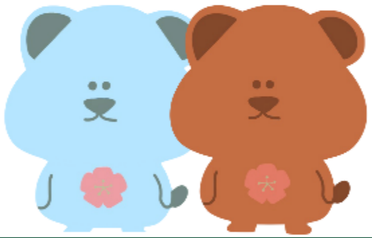
Fully Connected layer





Convolutional layer

ConvNet Algorithms

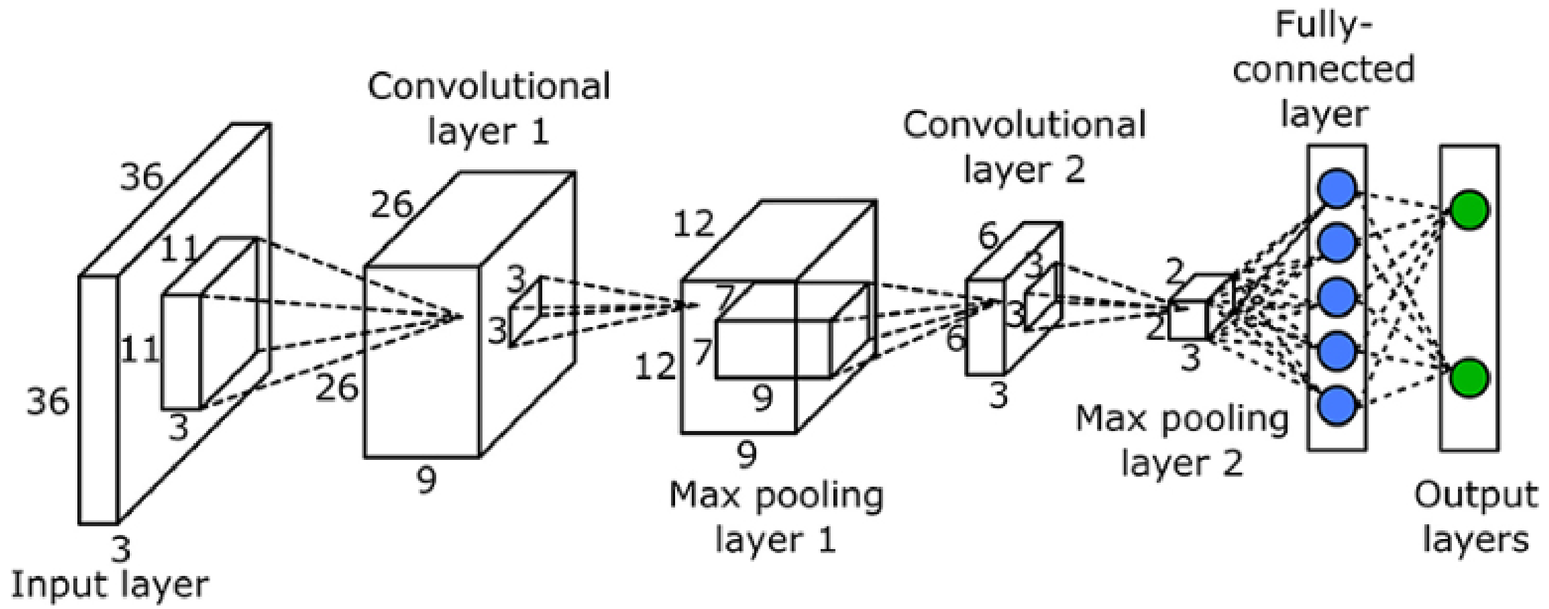


EWHA EURON

01 ConvNet

ConvNet 의 구조

Classification	Image Detection	Semantic Segentation
VGG Net	RCNN	FCN
GoogLeNet	Fast RCNN	DeepLab
ResNet	Faster RCNN	U-Net
MobileNet	YOLO	ReSeg
ShuffleNet	SDD	



1. Input layer – 2. Convolutional layer ~ Max pooling layer – 3. Fully-connected layer~output layers

01 ConvNet

ConvNet 의 layer sizing pattern

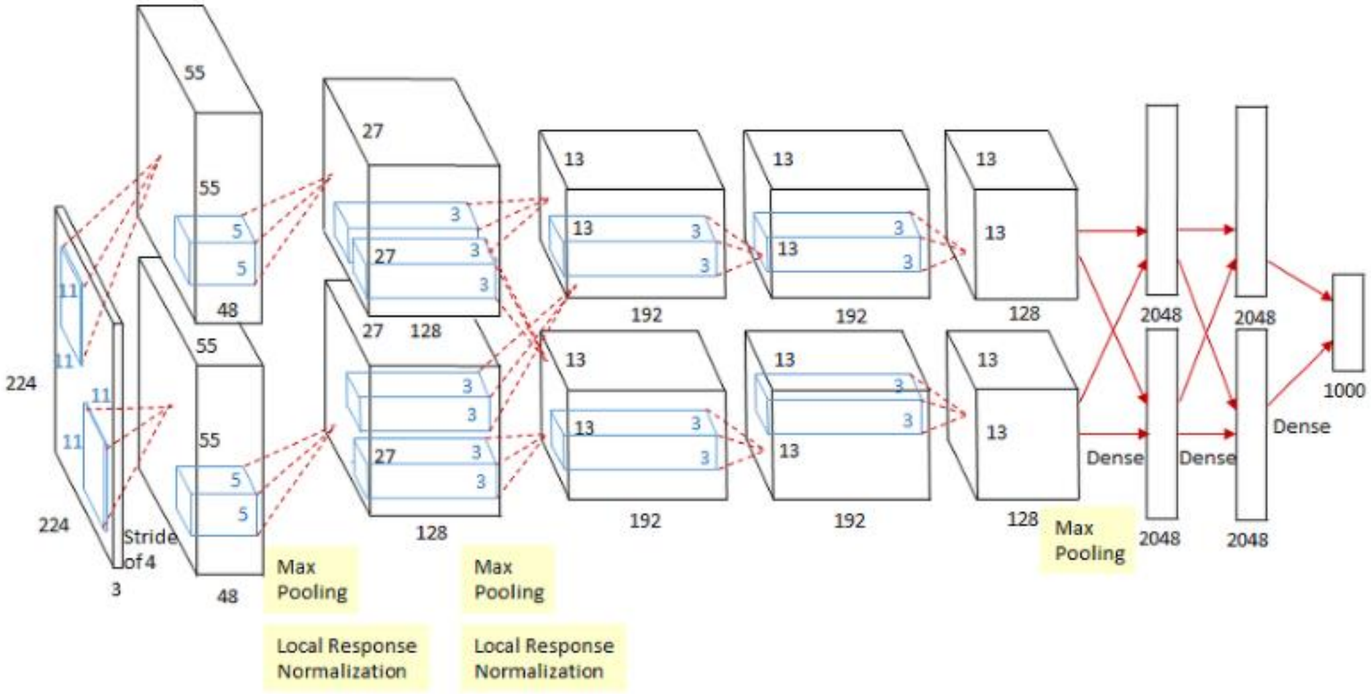
각 layer에서 주로 사용되는 hyperparameter setting
(F:filter P:pad S:stride)

1. Conv layer
- F=3 (최대 5)
- P=1 (p=(F-1)/2),zero padding
- S=1

2. Pooling layer
- F=2
- S=2

02 AlexNet

AlexNet

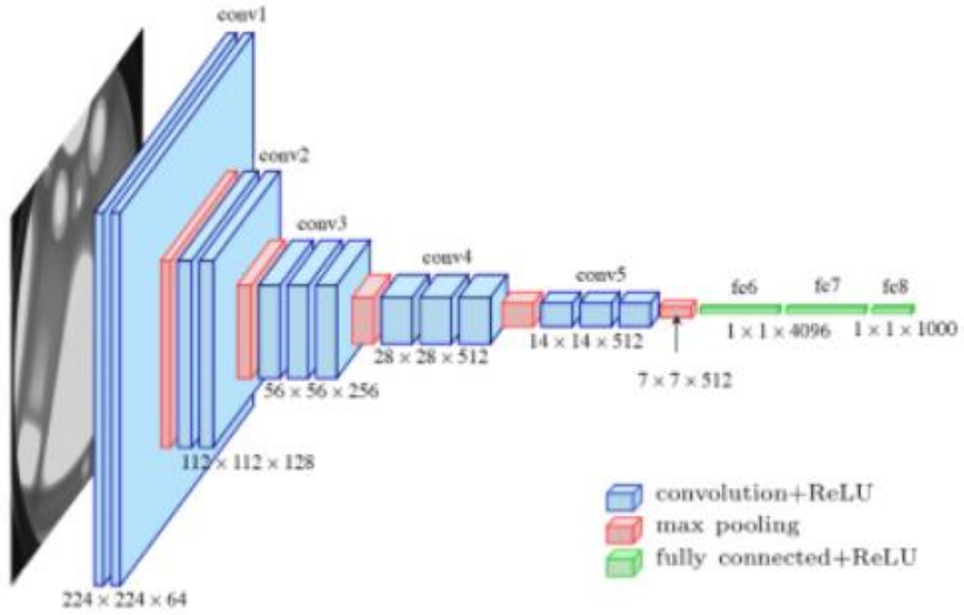


- GPU의 보급으로 대량연산이 가능해져 Alexnet을 시작으로 딥러닝 주목
- 하나 이상의 convolution layer와 pooling layer를 반복하고, 마지막에 Fully connected layer 거쳐 결과 출력
- 활성화 함수로 ReLu 처음 사용

EWHA EURON

VGGNET

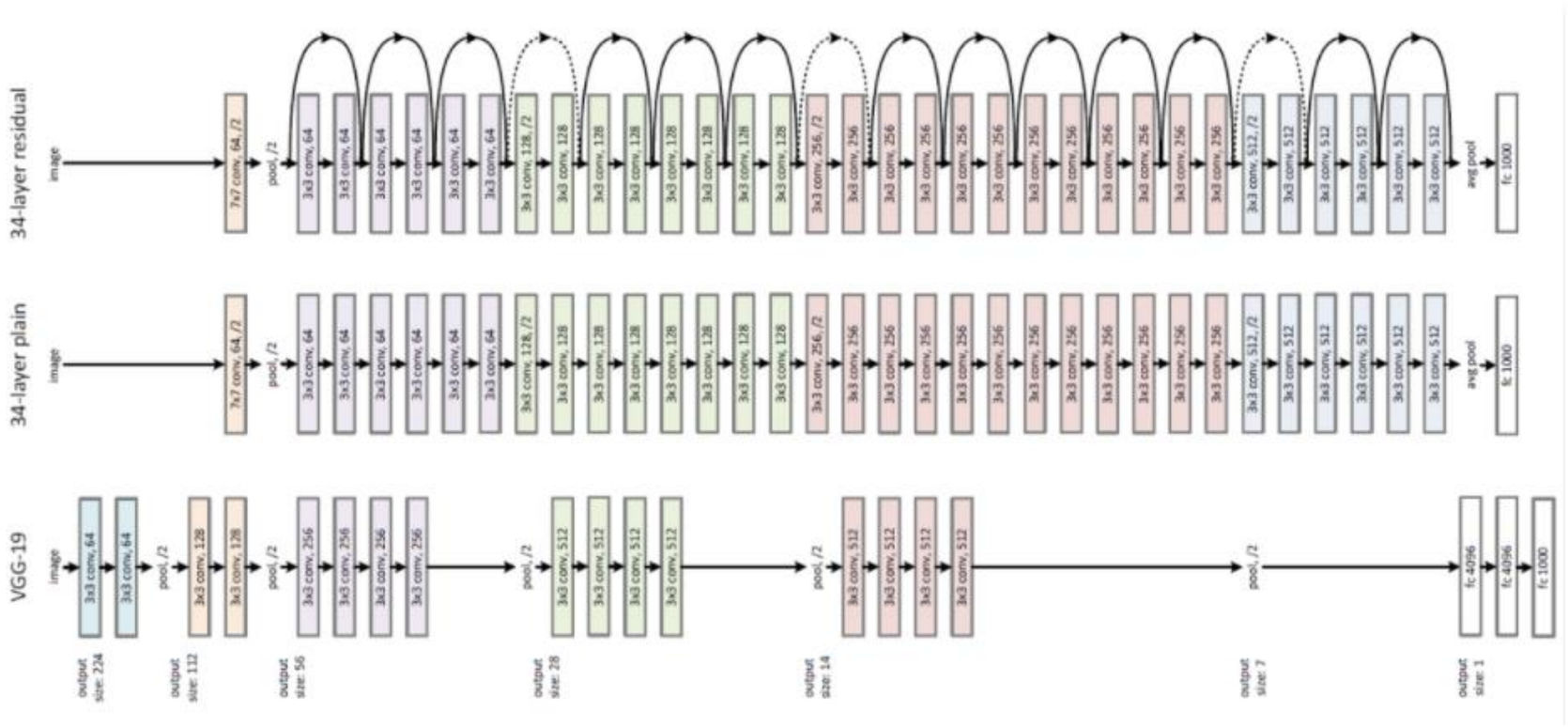
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



VGG16의 구조

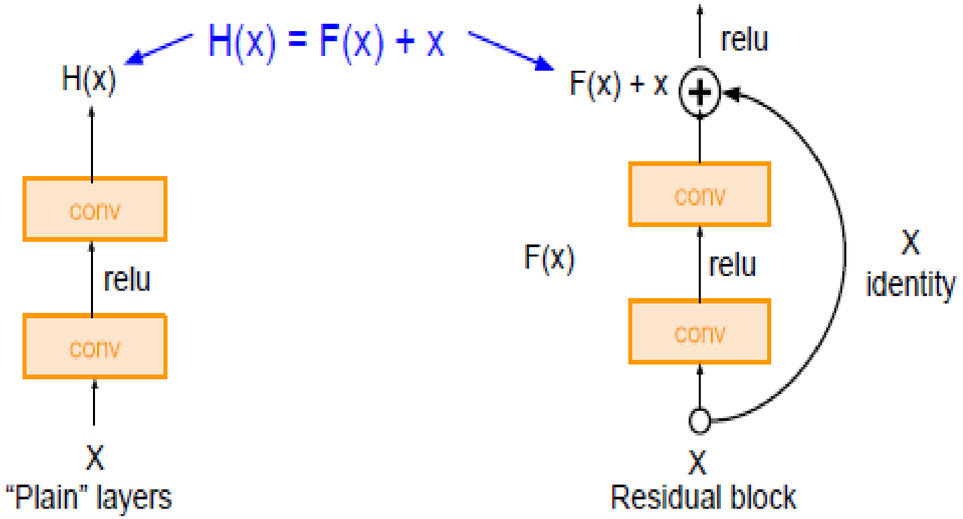
04 RESNET

RESNET



04 RESNET

RESNET



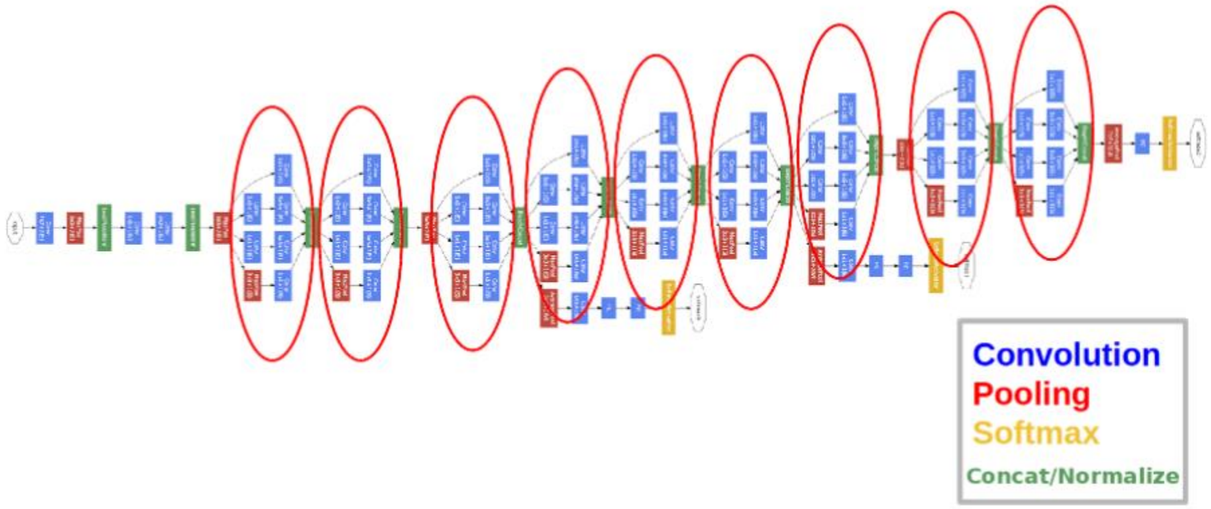
Residual Connection(Skip Connection)

- 기존에 학습한 정보(x)를 보존하고 추가적으로 layer에서 학습한 정보 추가함
→각각의 layer가 학습할 정보량 축소
-오픈북 시험에서 시험자료(x)는 그대로 있고 추가로 필요한 정보를 찾는 방식

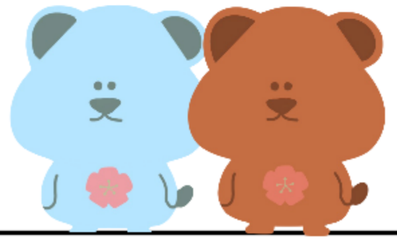
05 GoogLeNet

GoogLeNet(Inceptionet)

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								



THANK YOU



EWHA EURON