

AI and Deep Learning

# CNN, Convolutional Neural Network

Jeju National University  
Yung-Cheol Byun

컨볼루션  
(곱하고 합친 후 이동하는 것)

AI and Deep Learning

# CNN, Convolutional Neural Network

Jeju National University

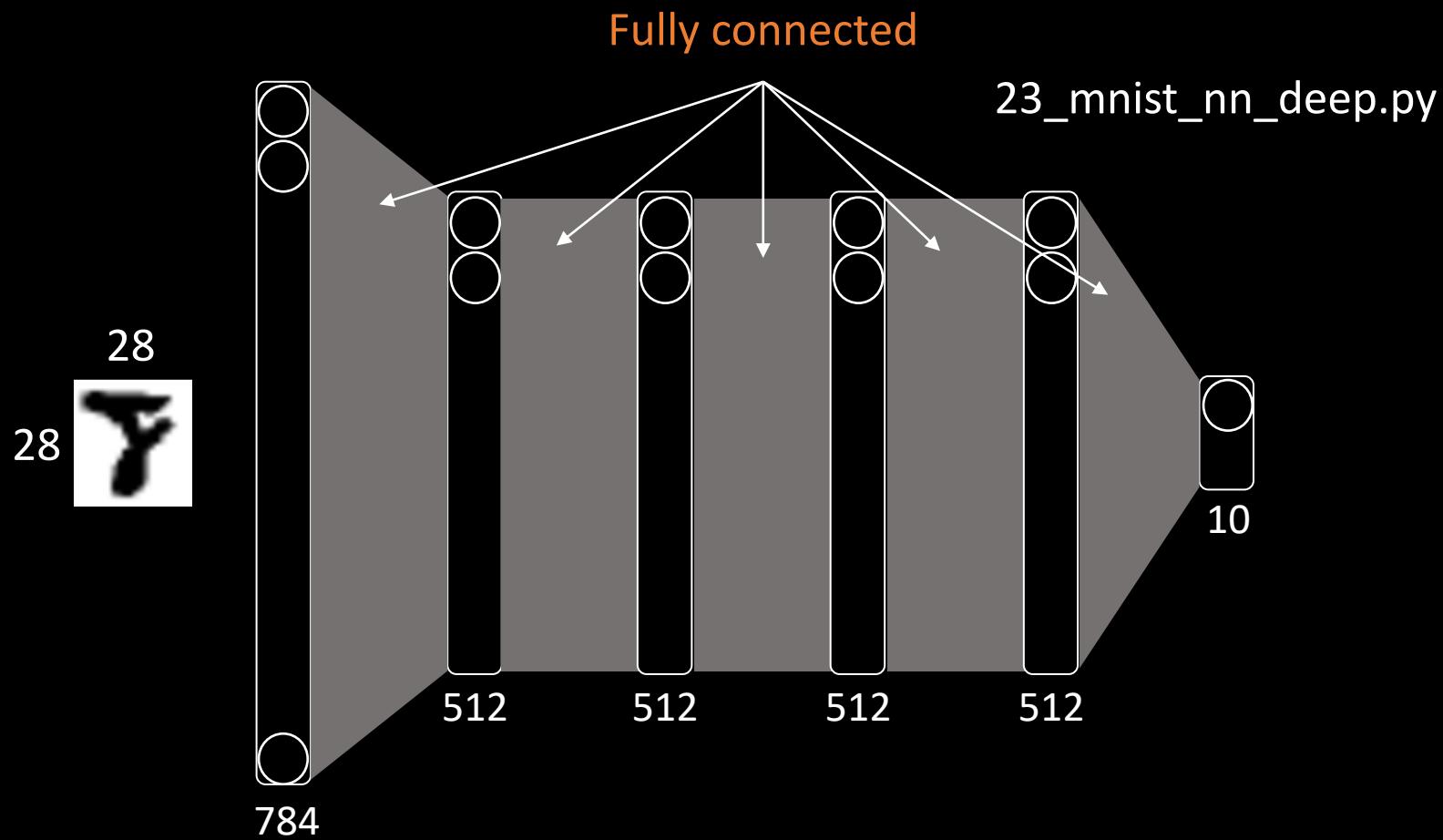
Yung-Cheol Byun

# Agenda

- Artificial Intelligence
- Brain and Neurons
- Learning
- Regression
- Deep Neural Networks
- CNN

Supervised  
Learning

이제까지 배운  
신경망의 단점은?



28

28



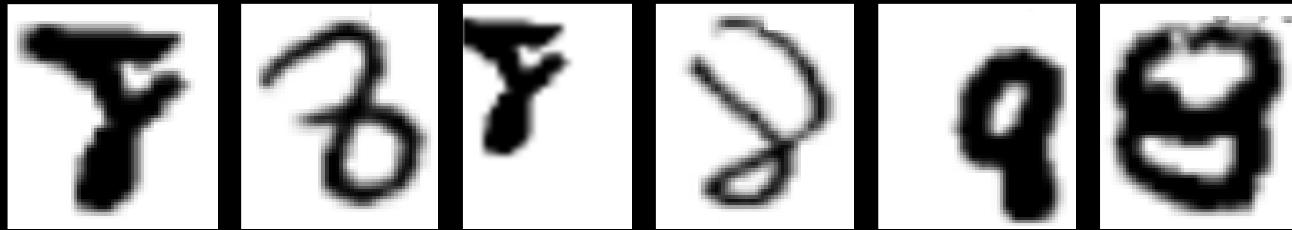
784(28x28) pixels as inputs to NN

Different inputs according to the size, location, skewness, and distortion



이미지는 같은데 이미지 회전, 위치, 확대/축소 등에 따라 픽셀이 달라지고, 따라서 신경망 입력층으로 주어지는 **입력 값이 확연히 달라짐.**

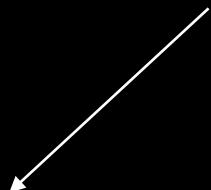




모양, 위치, 크기, 기울기에 따라  
신경망 입력이 달라짐



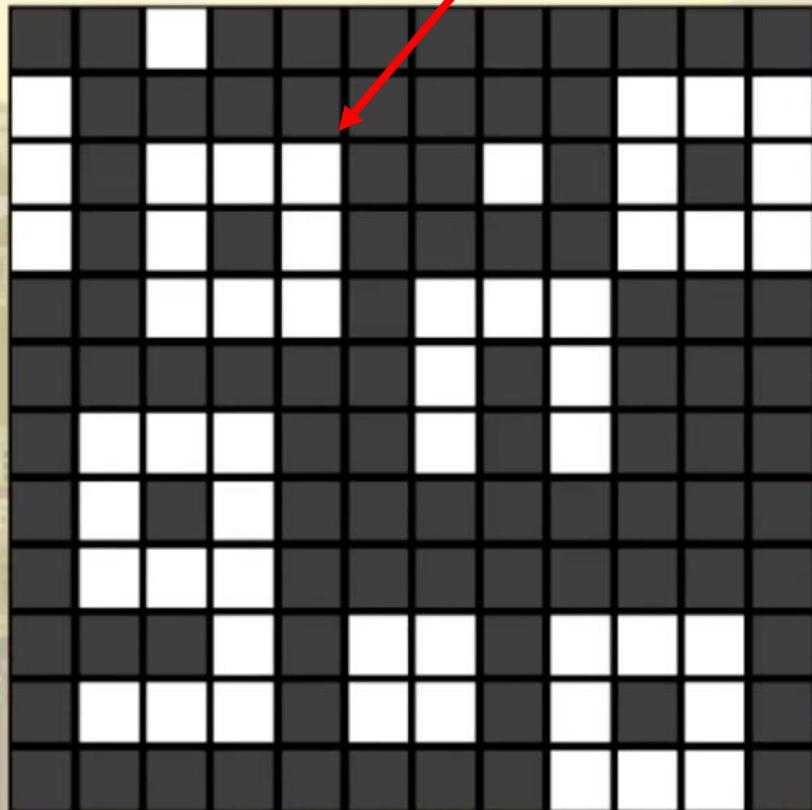
따라서 이미지를 바로 신경망에  
넣지 말고 중요한 정보만  
걸러서(필터링해서) 주자.



거를 때 쓰는 것 : 필터(filter)

# 도넛을 찾아라!

“도넛이 될 조건을 말해보자”

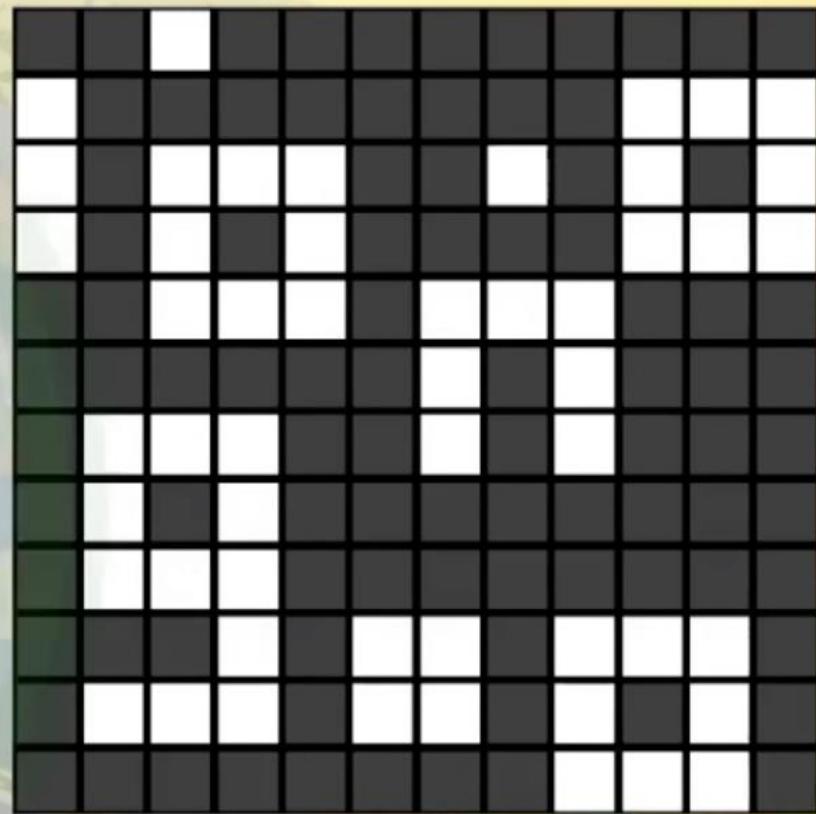


IMAGE



# "DONUT FILTER"

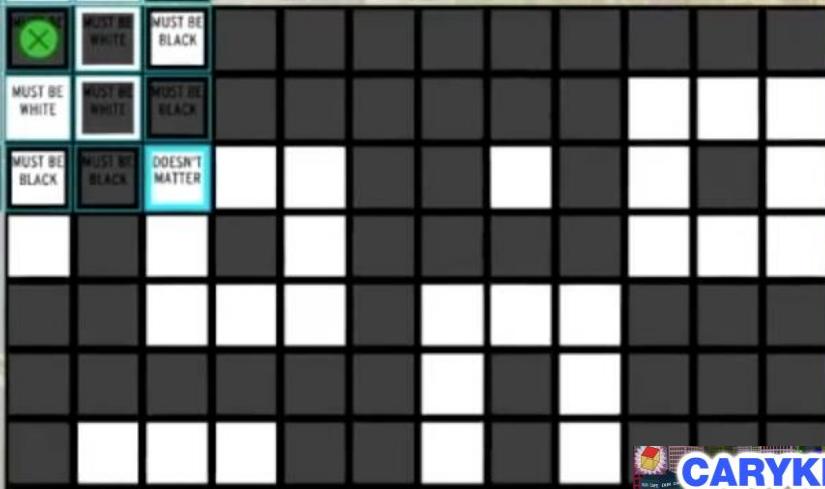
DOESN'T MATTER	MUST BE BLACK	MUST BE BLACK	MUST BE BLACK	DOESN'T MATTER
MUST BE BLACK	MUST BE WHITE	MUST BE WHITE	MUST BE WHITE	MUST BE BLACK
MUST BE BLACK	MUST BE WHITE	MUST BE BLACK	MUST BE WHITE	MUST BE BLACK
MUST BE BLACK	MUST BE WHITE	MUST BE WHITE	MUST BE WHITE	MUST BE BLACK
DOESN'T MATTER	MUST BE BLACK	MUST BE BLACK	MUST BE BLACK	DOESN'T MATTER



"DONUT FILTER"

upper-left pixel

DOESN'T MATTER	MUST BE BLACK	MUST BE BLACK	MUST BE BLACK	DOESN'T MATTER
MUST BE BLACK	MUST BE WHITE	MUST BE WHITE	MUST BE WHITE	MUST BE BLACK
MUST BE BLACK	MUST BE WHITE		MUST BE WHITE	MUST BE BLACK
MUST BE BLACK	MUST BE WHITE	MUST BE WHITE	MUST BE WHITE	MUST BE BLACK
DOESN'T MATTER	MUST BE BLACK	MUST BE BLACK	MUST BE BLACK	DOESN'T MATTER



"우리가 정한 조건을 모두 만족하는가?"

"Is every condition of the filter satisfied?"

DOESN'T MATTER ✓ Yes	MUST BE BLACK ✓ Yes	MUST BE BLACK ✓ Yes	MUST BE BLACK ✓ Yes	DOESN'T MATTER ✓ Yes
MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE WHITE ✗ No	MUST BE WHITE ✗ No	MUST BE BLACK ✓ Yes
MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE BLACK ✗ No
MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE WHITE ✓ Yes	MUST BE WHITE ✗ No	MUST BE BLACK ✓ Yes
DOESN'T MATTER ✓ Yes	MUST BE BLACK ✓ Yes	MUST BE BLACK ✗ No	MUST BE BLACK ✓ Yes	DOESN'T MATTER ✓ Yes

Out-of-bounds pixels are considered black.

"Is every condition of the filter satisfied?"

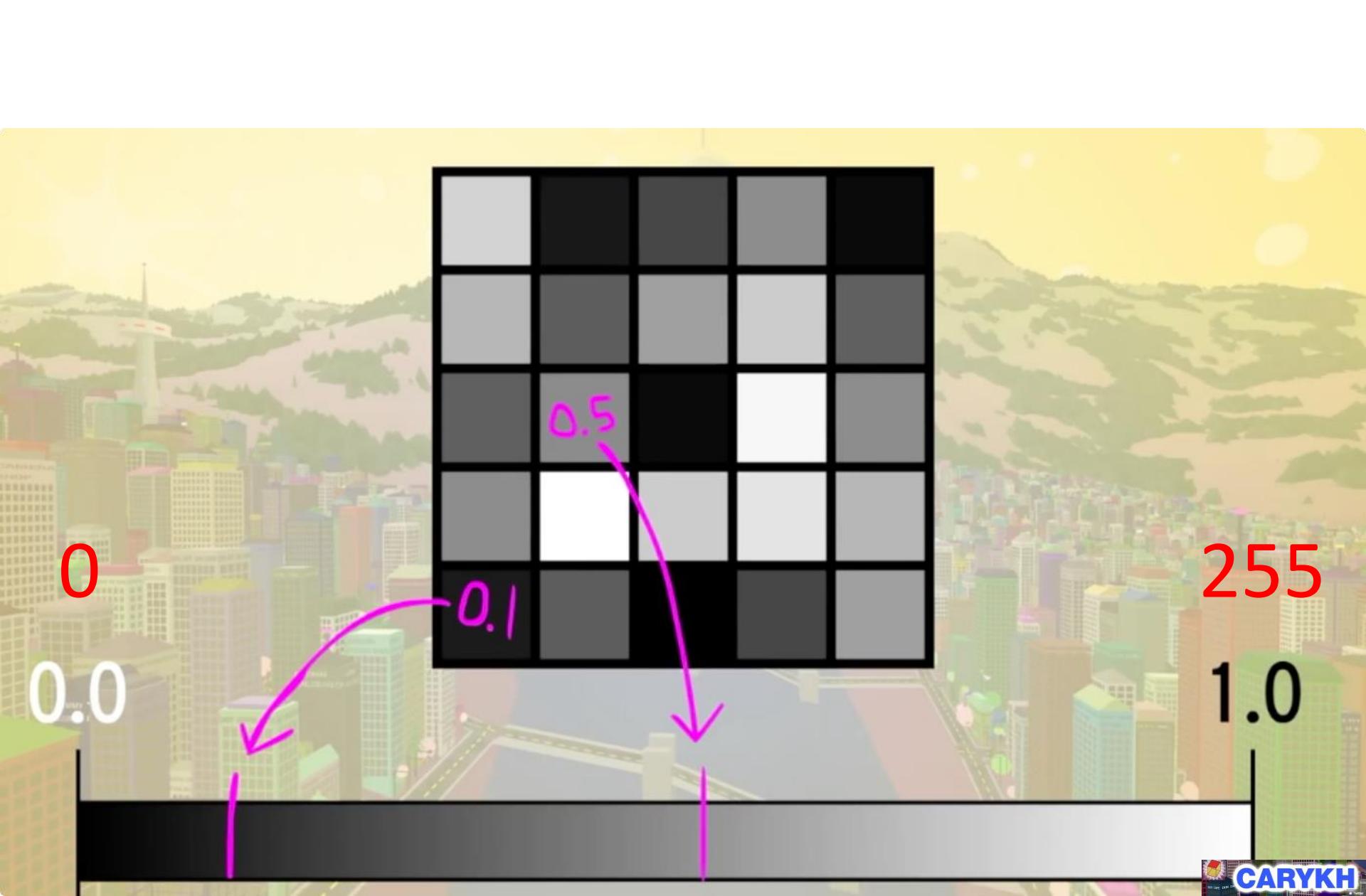
It's not a donut.

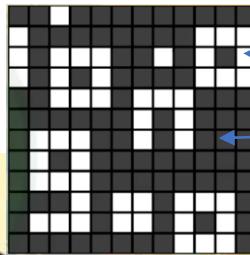
DOESN'T MATTER ✓ Yes	MUST BE BLACK ✓ Yes	MUST BE BLACK ✓ Yes	MUST BE BLACK ✓ Yes	DOESN'T MATTER ✓ Yes
MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE WHITE ✗ No	MUST BE WHITE ✗ No	MUST BE BLACK ✓ Yes
MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE BLACK ✗ No
MUST BE BLACK ✓ Yes	MUST BE WHITE ✗ No	MUST BE WHITE ✓ Yes	MUST BE WHITE ✗ No	MUST BE BLACK ✓ Yes
DOESN'T MATTER ✓ Yes	MUST BE BLACK ✓ Yes	MUST BE BLACK ✗ No	MUST BE BLACK ✓ Yes	DOESN'T MATTER ✓ Yes

바깥 부분은 black으로  
간주하여 확인  
→ 9개 조건 만족하지 않아  
도넛이 아님.

## 블랙 & 화이트 vs. Gray vs. 컬러







# DONUT FILTER

DOESN'T MATTER	MUST BE BLACK	MUST BE BLACK	MUST BE BLACK	DOESN'T MATTER
MUST BE BLACK	MUST BE WHITE	MUST BE WHITE	MUST BE WHITE	MUST BE BLACK
MUST BE BLACK	MUST BE WHITE	MUST BE BLACK	MUST BE WHITE	MUST BE BLACK
MUST BE BLACK	MUST BE WHITE	MUST BE WHITE	MUST BE WHITE	MUST BE BLACK
DOESN'T MATTER	MUST BE BLACK	MUST BE BLACK	MUST BE BLACK	DOESN'T MATTER

# New improved filter

x0.0	x-0.2	x-0.1	x-0.2	x0.0
x-0.2	x0.7	x1.0	x0.7	x-0.2
x-0.1	x1.0	x-2.5	x1.0	x-0.1
x-0.2	x0.7	x1.0	x0.7	x-0.2
x0.0	x-0.1	x-0.1	x-0.1	x0.0

"set of multipliers"

“흰색이 있어야 할 곳에 흰색이 있으면 큰 점수를 더하고”  
“흰색이 있어야 할 곳에 검정색이 오면 큰 점수를 빼고”

"If you want to  
be considered  
donutty..."



x-0.2	x-0.1	x-0.2	x0.0
x0.7	x1.0	x0.7	x-0.2
0.1	x1.0	x-2.5	x1.0
x-0.2	x0.7	x1.0	x-0.1
x0.0	x-0.1	x-0.1	x0.0

...you'd better  
have a high  
value for this  
pixel."

컨볼루션  
(곱하고 합치는 것)

# THE FILTER ↴



0.73	0.86	0.74	0.85	0.99	0.58	0.96	0.25
0.29	0.92	0.25	0.38	0.67	0.16	0.63	0.73
0.90	0.99	0.95	0.06	0.56	0.85	0.23	0.84
0.00	0.45	0.55	0.28	0.33	0.59	0.69	0.01
0.50	0.23	0.03	0.11	0.48	0.23	0.62	0.69
0.85	0.52	0.11	0.95	0.80	0.24	0.03	0.54
0.99	0.07	0.69	0.35	0.05	0.15	0.61	0.80
0.03	0.84	0.79	0.52	0.17	0.66	0.73	0.01

Filter size : 5 X 5

0.00	x-0.20	x-0.10	x-0.20	x0.00					
0.20	x0.70	x1.00	x0.70	x-0.20					
0.10	x1.00	0.73 x-2.50	0.86 x1.00	0.74 x-0.10	0.85	0.99	0.58	0.96	0.25
0.20	x0.70	0.29 x1.00	0.92 x0.70	0.25 x-0.20	0.38	0.67	0.16	0.63	0.73
0.00	x-0.20	0.90 x-0.10	0.99 x-0.20	0.95 x0.00	0.06	0.56	0.85	0.23	0.84
	0.00	0.45	0.55	0.28	0.33	0.59	0.69	0.01	
0.50	0.23	0.03	0.41	0.48	0.23	0.62	0.69		
0.85	0.52	0.11	0.95	0.80	0.24	0.03	0.54		
0.99	0.07	0.69	0.35	0.05	0.15	0.61	0.80		
0.03	0.84	0.79	0.52	0.17	0.66	0.73	0.01		

out-of-bound  
패딩

컨볼루션  
(곱하고 합친 후 이동하는 것)

How donutty is each pixel?



“필터로 이미지에 대해 합성곱(곱한 후 합치는 것) 수행”

Convolve the filter  
with the image

slide over the image spatially, computing dot product

“이미지에서 공간적으로 필터를 이동하면서 dot  
product를 수행”

A filter containing a group of  
synapsis (parameters)

shared by the whole neurons

하나의 필터로  
입력 이미지 전체에 대하여  
컨볼루션 수행하여 결과 만듬

x1.0	x1.5	x1.0	x0.4
x0.7	x1.0	x0.7	x0.6
x-0.4	x-0.4	x-0.4	x0.0

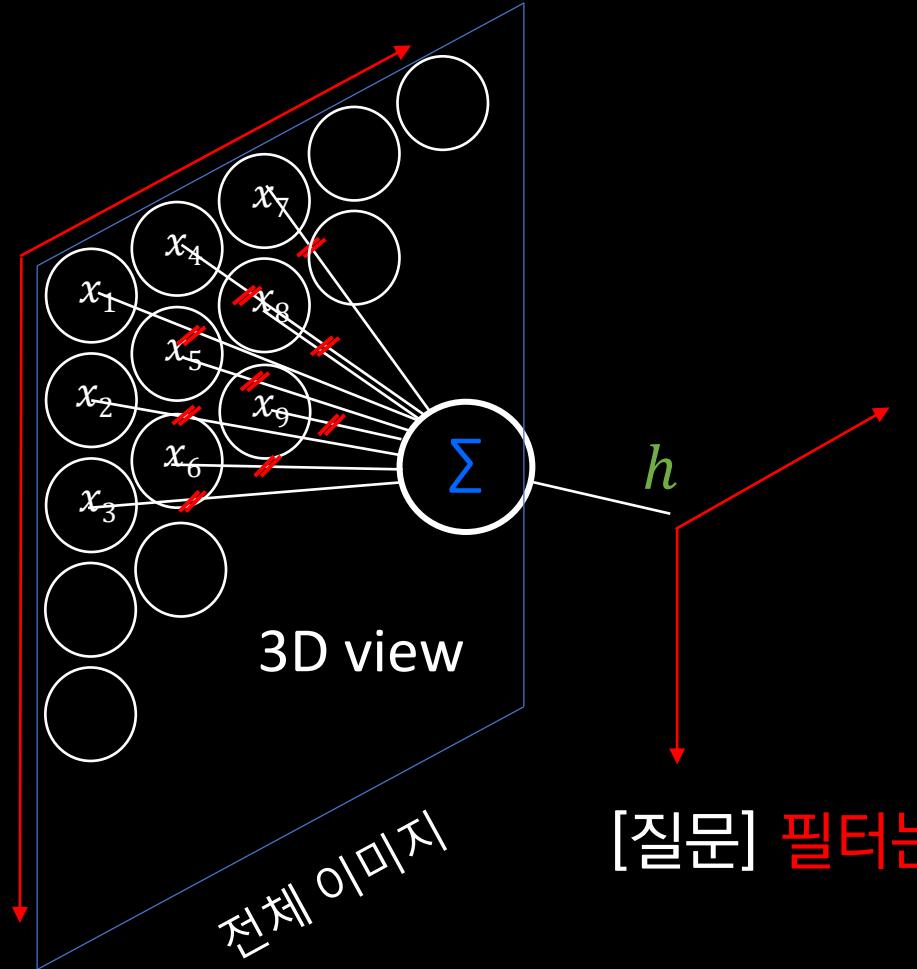
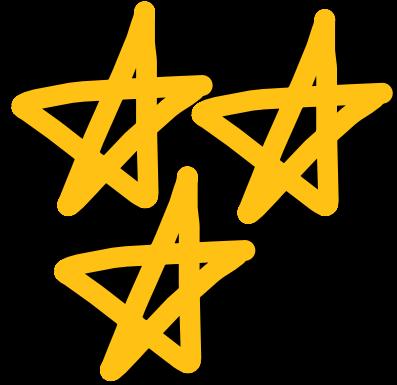
0.73	0.86	0.74	0.85	0.99	0.58	0.96	0.25
0.29	0.92	0.25	0.38	0.67	0.16	0.63	0.73
0.90	0.99	0.95	0.06	0.56	0.85	0.23	0.84
0.00	0.45	0.55	0.28	0.33	0.59	0.59	0.01
0.50	0.23	0.03	0.41	0.49	0.23	0.62	0.69
0.85	0.52	0.11	0.95	0.80	0.24	0.03	0.54
0.99	0.07	0.69	0.35	0.05	0.15	0.61	0.80
0.03	0.84	0.79	0.52	0.17	0.66	0.73	0.01

AVIARY

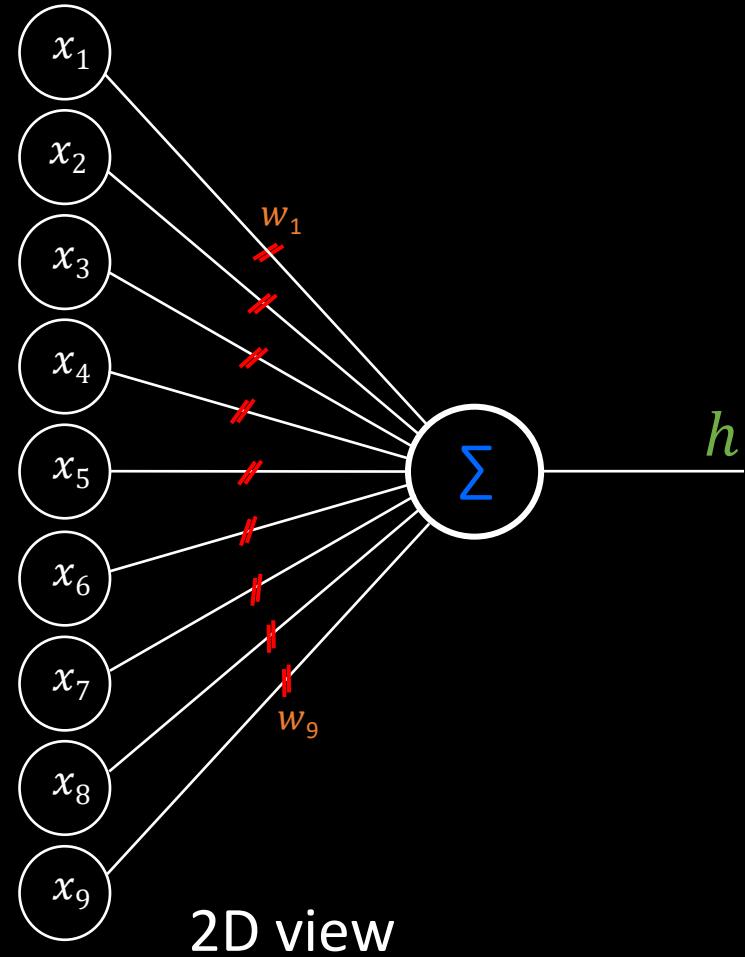
CIRCUIT  
TENNIS COURT



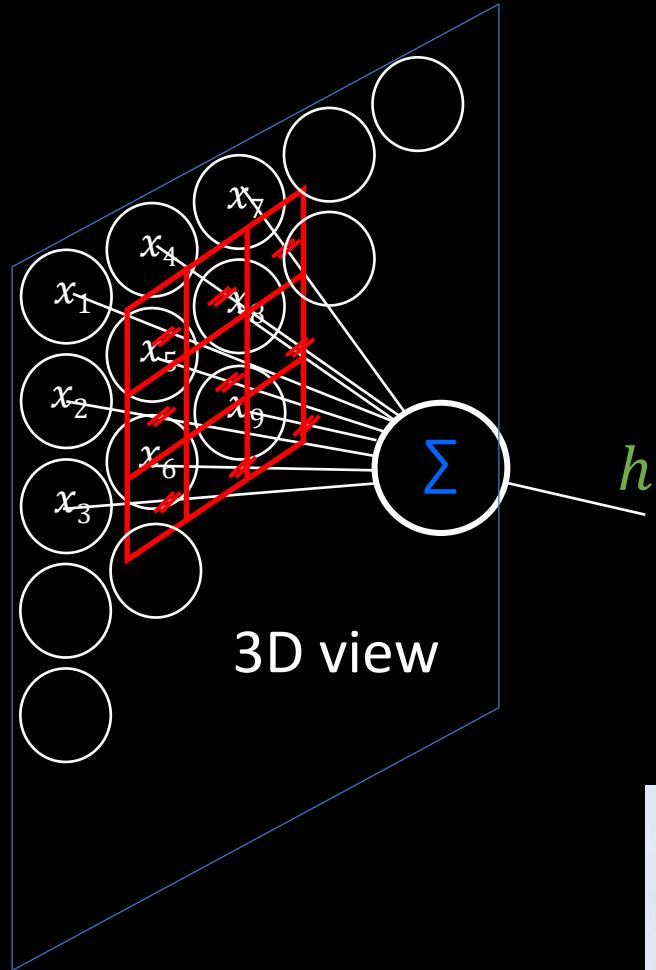
9 connections = 9 synaps (values)



[질문] 필터는 어디에?



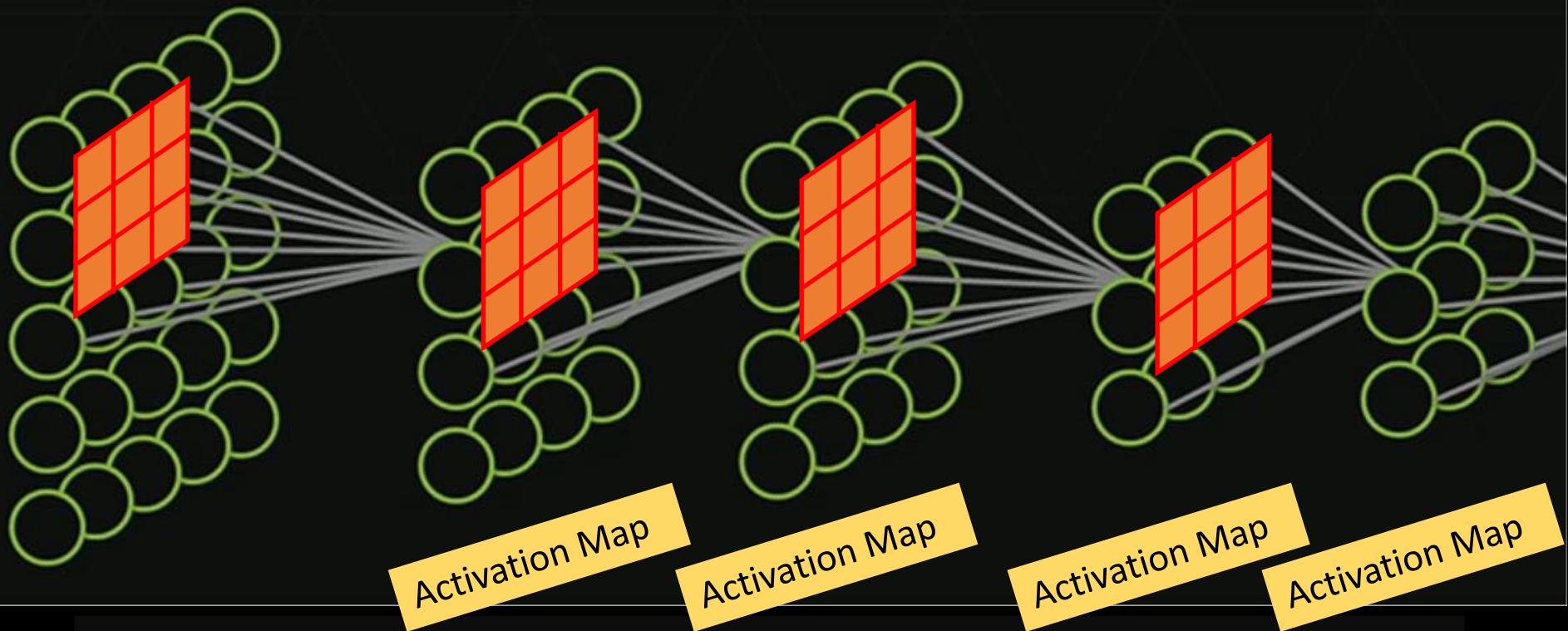
9 connections = 9 synaps (values) -> 필터



`sess.run(train)`

Filter size : 3x3, 1 filter for each convolution,  
then how many parameters to be tuned?

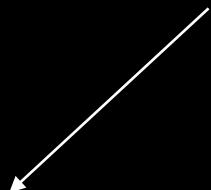
$$(3 * 3) * 4 = 36$$



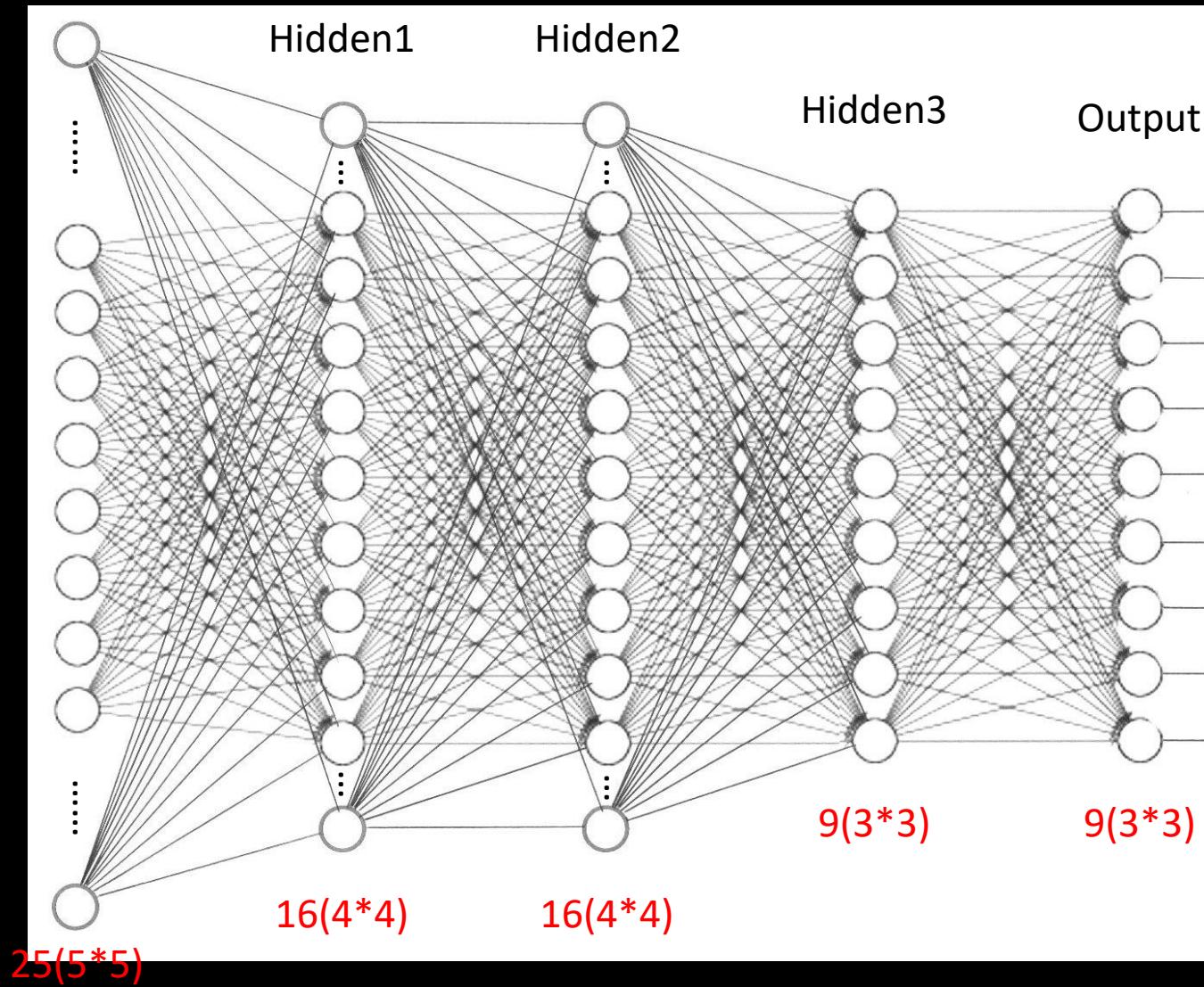
If *fully* connected,  $25*16+16*16+16*9+9*9 = 881$



따라서 이미지를 바로 신경망에  
넣지 말고 중요한 정보만  
걸러서(필터링해서) 주자.

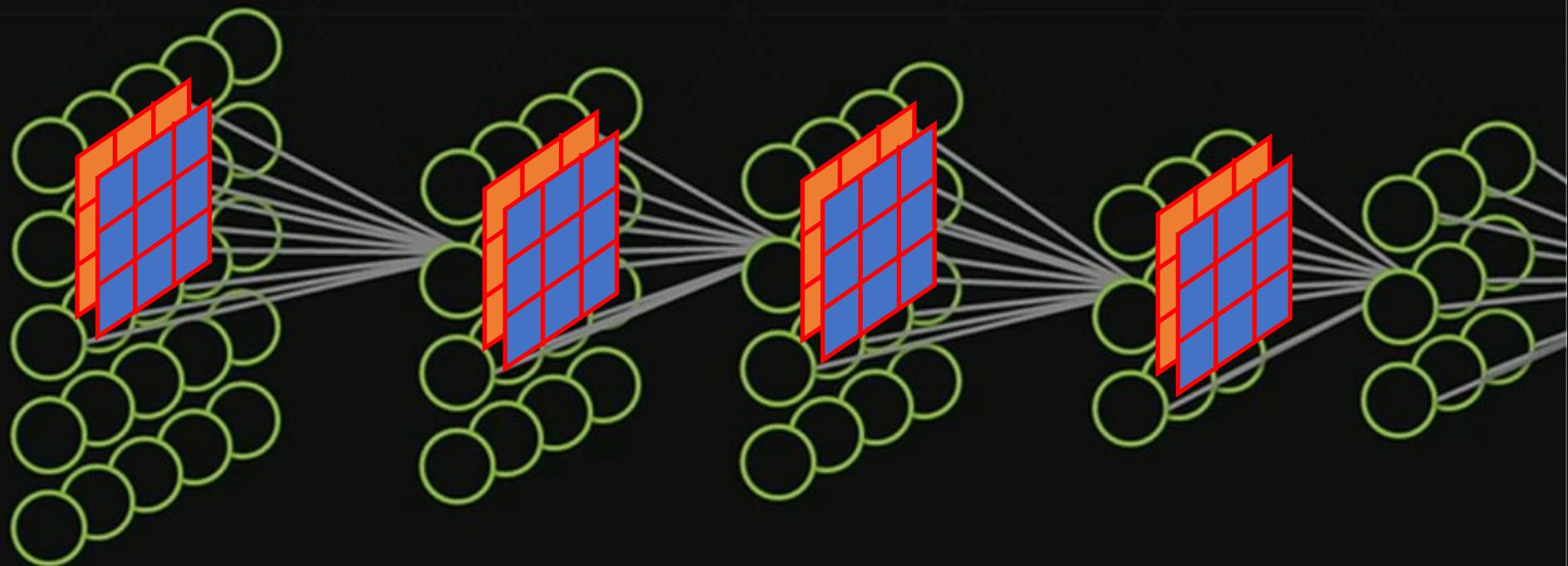


거를 때 쓰는 것 : 필터(filter)



필터가 2개라면

$$(3 * 3 * 2) * 4 = 72$$



out-of-bound,  
패딩

0.0	0.05	0.04	0.06	0.0	0	0	0	0	0	0	0	0
0.0	0.07	0.0	0.07	0.0	0	0	0	0	0	0	0	0
0.0	0	0.73 x1.5	0.86 x1.0	0.74 x0.4	0.85	0.99	0.58	0.96	0.25	0	0	0
0.0	0	0.29 x1.0	0.92 x0.7	0.25 x0.6	0.38	0.67	0.16	0.63	0.73	0	0	0
0.0	0	0.90 x0.4	0.99 x0.4	0.95 x0.0	0.06	0.56	0.85	0.23	0.84	0	0	0
0.0	0	0.00	0.45	0.55	0.28	0.33	0.59	0.69	0.01	0	0	0
0.0	0	0.50	0.23	0.03	0.41	0.48	0.23	0.62	0.69	0	0	0
0.0	0	0.85	0.52	0.11	0.95	0.80	0.24	0.03	0.54	0	0	0
0.0	0	0.99	0.07	0.69	0.35	0.05	0.15	0.61	0.80	0	0	0
0.0	0	0.03	0.84	0.79	0.52	0.17	0.66	0.73	0.01	0	0	0
0.0	0	0	0	0	0	0	0	0	0	0	0	0
0.0	0	0	0	0	0	0	0	0	0	0	0	0

activation map

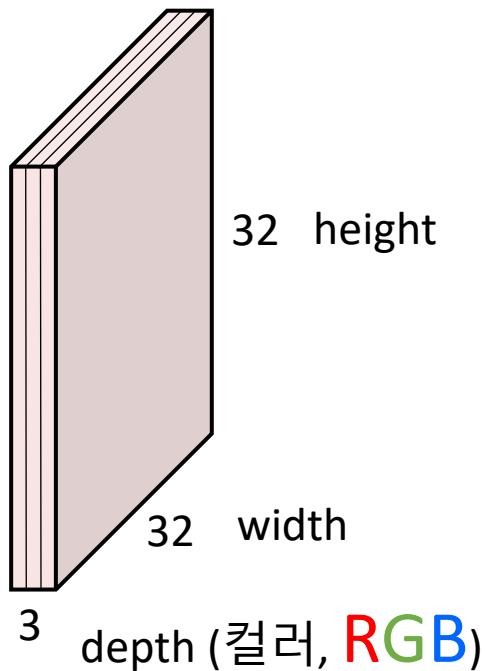
-0.44	0.00	0.34	-0.04	-0.58	1.00	-0.92	1.29
2.66	2.03	3.37	2.19	1.18	3.48	1.71	0.29
-0.60	0.88	-0.06	2.10	0.40	-0.01	2.71	-0.85
2.04	1.74	0.39	1.02	1.17	1.05	0.92	2.25
-0.01	0.90	1.81	1.07	1.33	1.79	0.17	-0.58
-0.34	0.86	1.54	-0.49	-0.34	0.73	2.66	0.63
-0.89	3.64	0.97	2.02	2.38	1.63	0.76	-0.69
1.39	-0.40	-0.14	0.05	0.43	-0.61	-0.07	1.77

# Some questions

- Fully connected? (Yes/No)
- How many connections for each pixels?
- Where are the parameters to be tuned?
- What happens if we have 2 filters?
- What happens if no padding(out-of-bound)?
- Convolution again with the resulting activation map

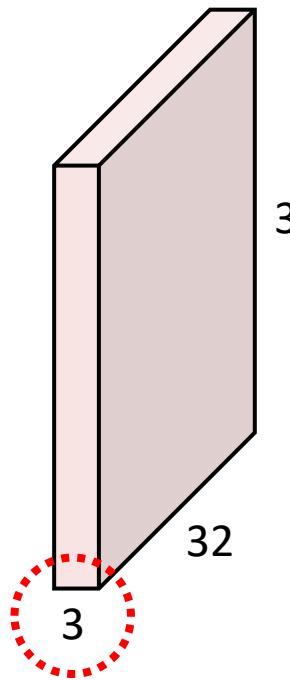
# Convolution Layer

32x32x3 pixels image



# Convolution Layer

32x32x3 image



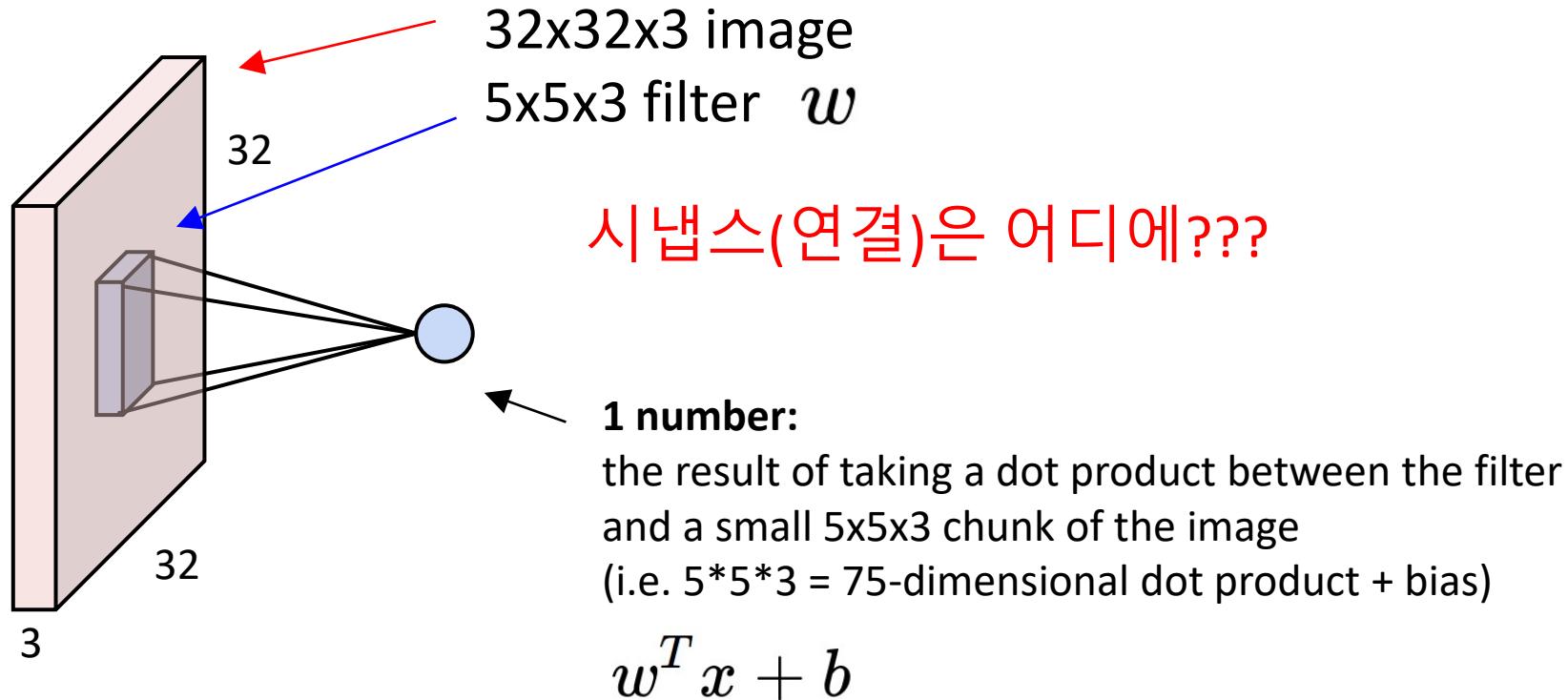
5x5x3 filter



Filters always extend the full depth of the input volume

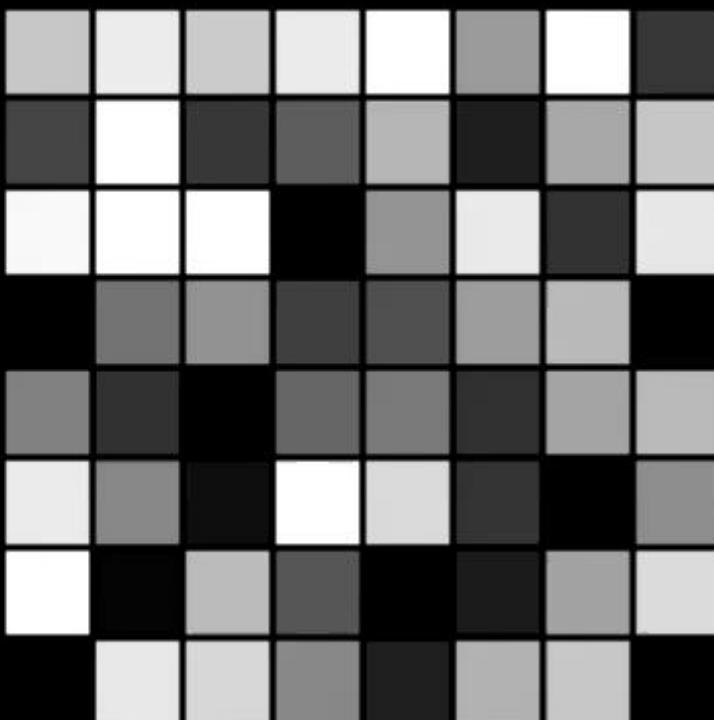
**Convolve** the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”

# Convolution Layer



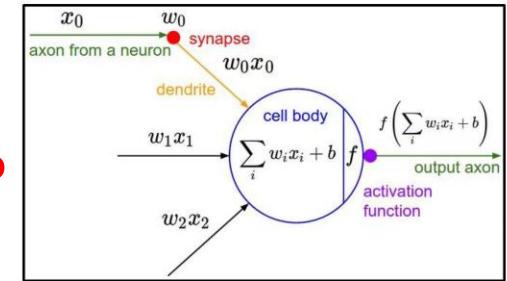
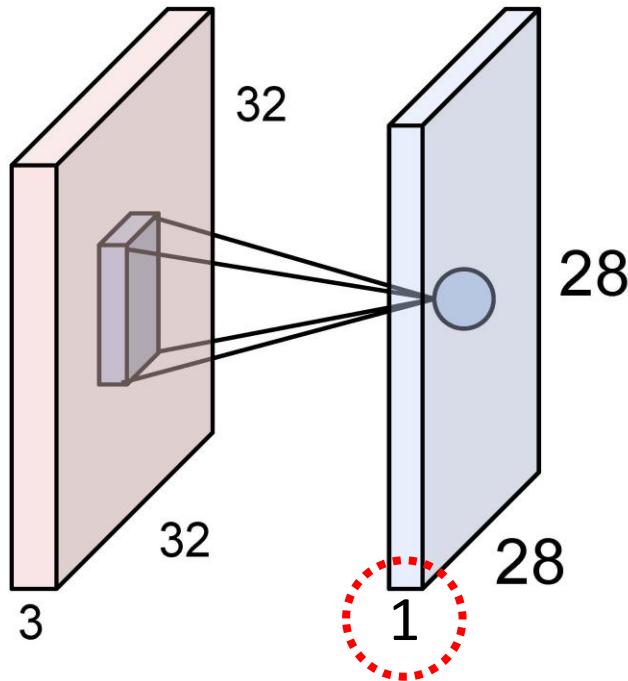
0.00	x-0.20	x-0.10	x-0.20	x0.00					
0.20	x0.70	x1.00	x0.70	x-0.20					
0.10	x1.00	0.73 x-2.50	0.86 x1.00	0.74 x-0.10	0.85	0.99	0.58	0.96	0.25
0.20	x0.70	0.29 x1.00	0.92 x0.70	0.25 x-0.20	0.38	0.67	0.16	0.63	0.73
0.00	x-0.20	0.90 x-0.10	0.99 x-0.20	0.95 x0.00	0.06	0.56	0.85	0.23	0.84
	0.00	0.45	0.55	0.28	0.33	0.59	0.69	0.01	
	0.50	0.23	0.03	0.41	0.48	0.23	0.62	0.69	
	0.85	0.52	0.11	0.95	0.80	0.24	0.03	0.54	
	0.99	0.07	0.69	0.35	0.05	0.15	0.61	0.80	
	0.03	0.84	0.79	0.52	0.17	0.66	0.73	0.01	

How donutty is each pixel?



# The brain/neuron view of CONV Layer

시냅스(연결)은 어디에???

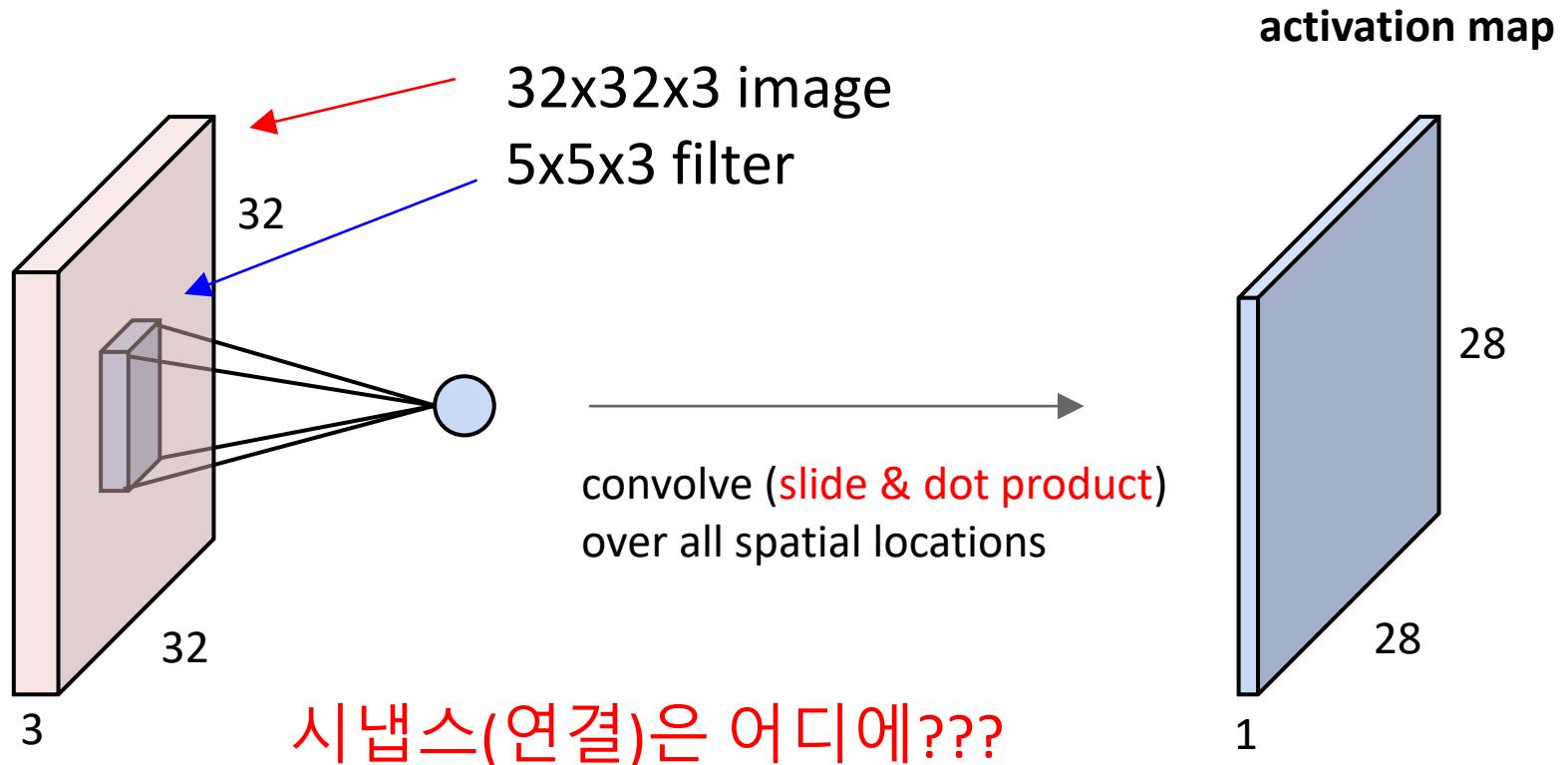


An activation map is a  $28 \times 28$  sheet of neuron outputs:

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

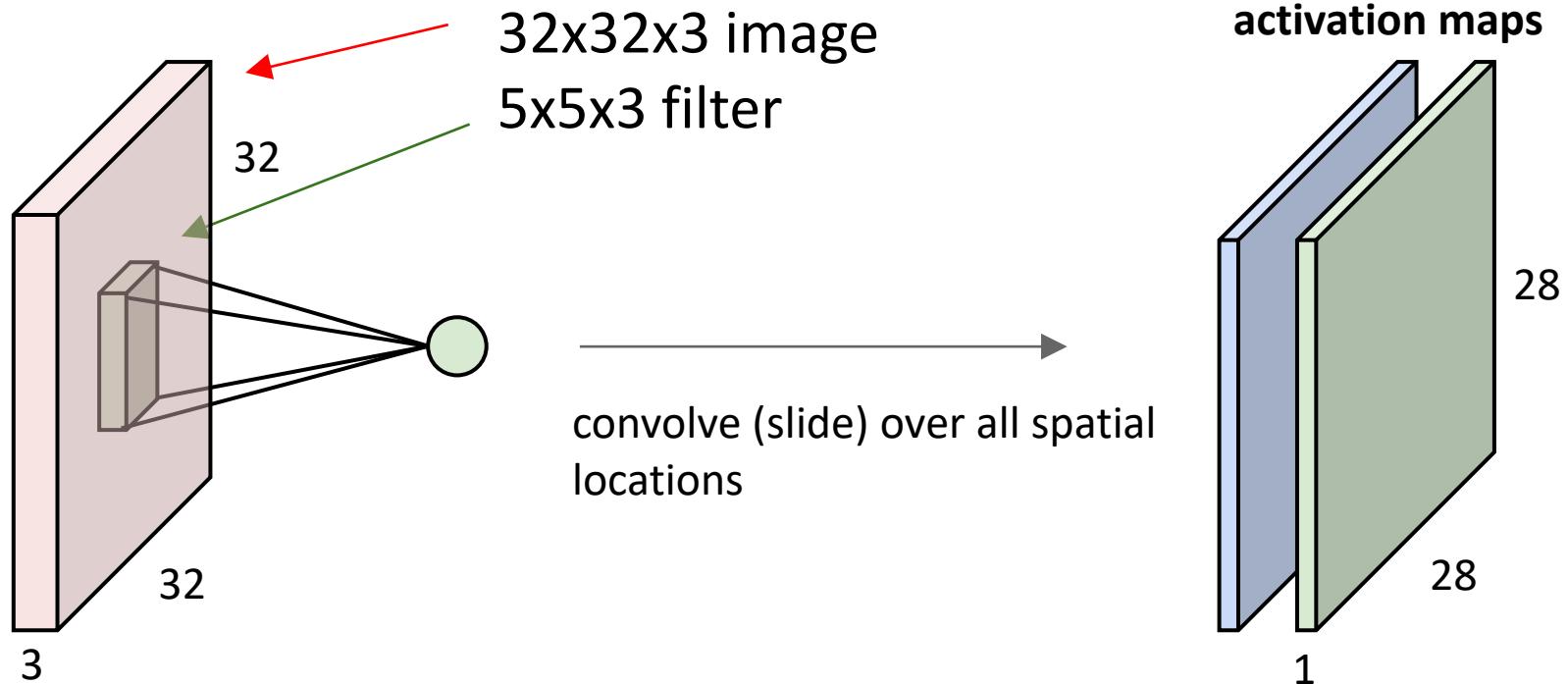
“ $5 \times 5$  filter” -> “ $5 \times 5$  receptive field for each neuron”

# Convolution Layer

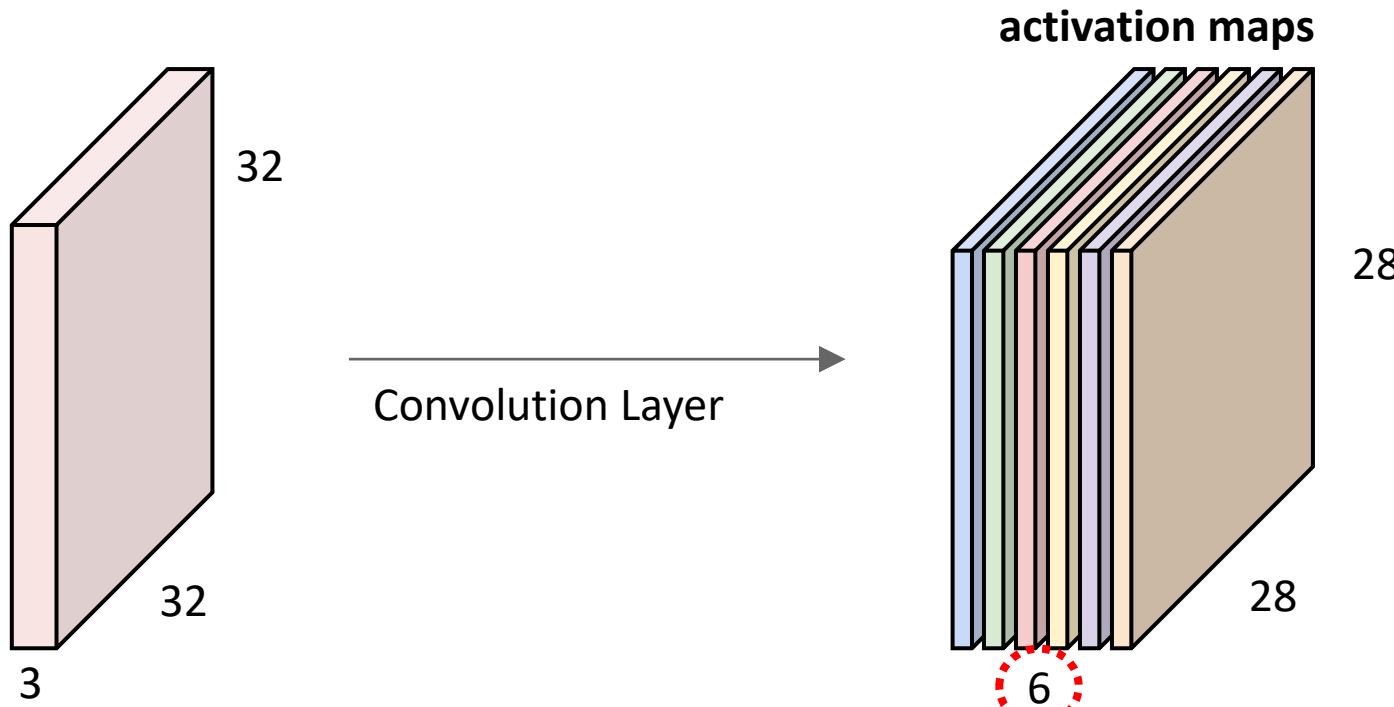


# Convolution Layer

consider a second, green filter



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

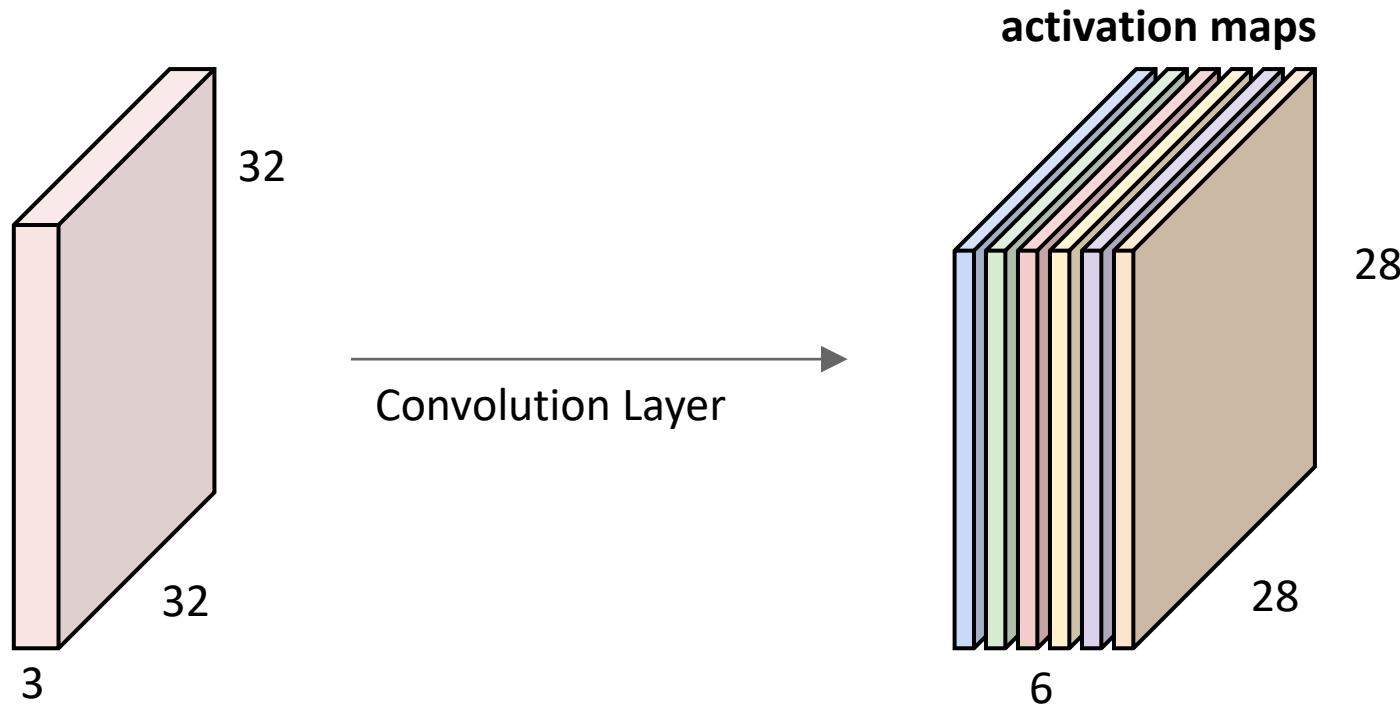


We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters would this be if we used a fully connected layer instead?

A:  $(32*32*3)*(28*28*6) = 14.5M$  parameters, ~14.5M multiplies

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



We processed [32x32x3] volume into [28x28x6] volume.

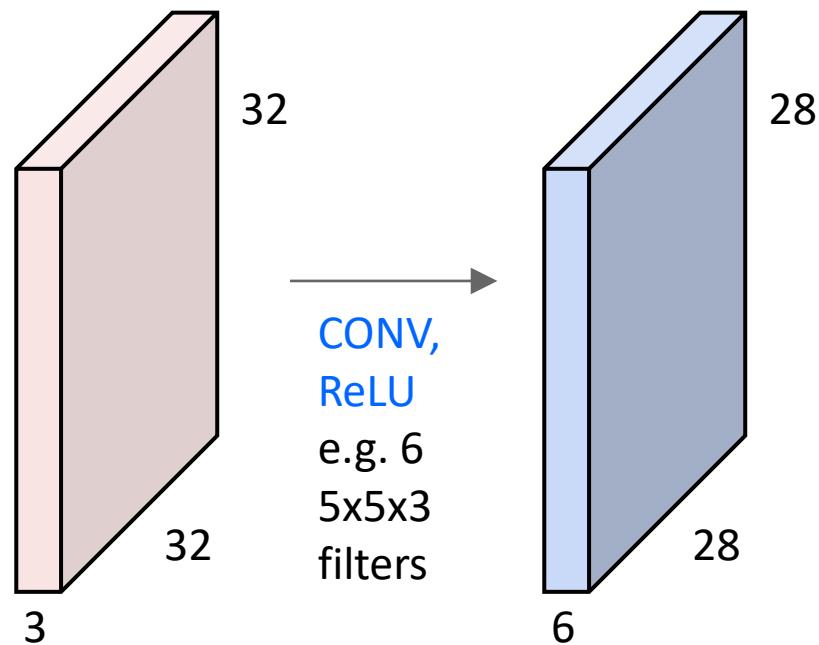
Q: how many parameters are used instead?

A:  $(5*5*3)*6 = 450$  parameters,  $(5*5*3)*(28*28*6) = \sim 350K$  multiplies

1 Filter -> 1 Activation Map

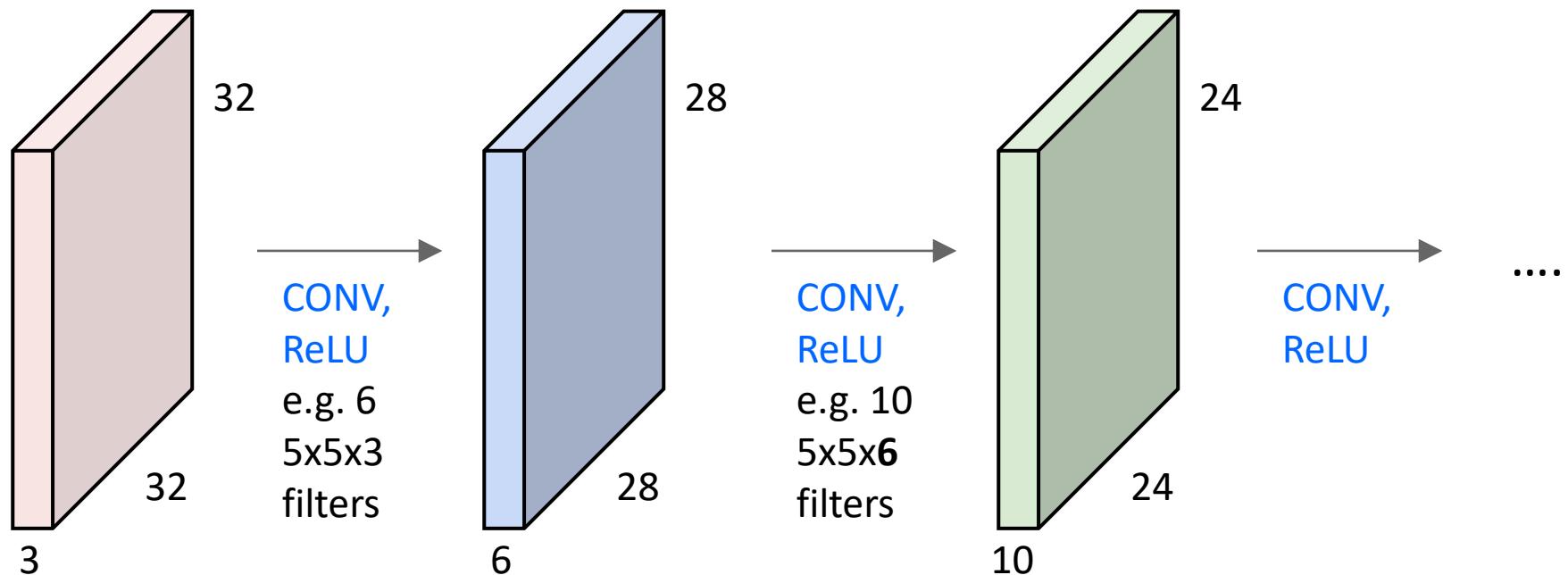
6 Filters -> 6 Activation Maps

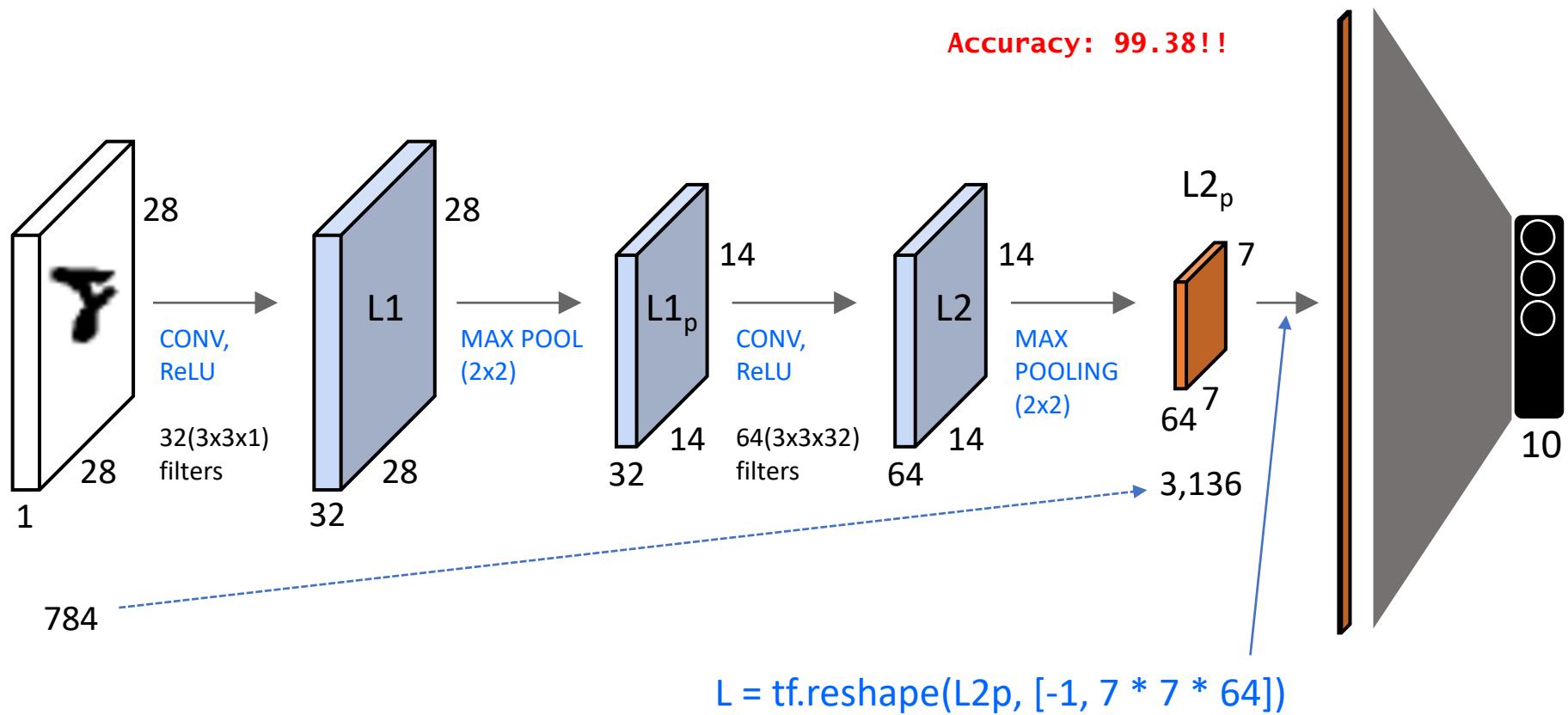
**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions

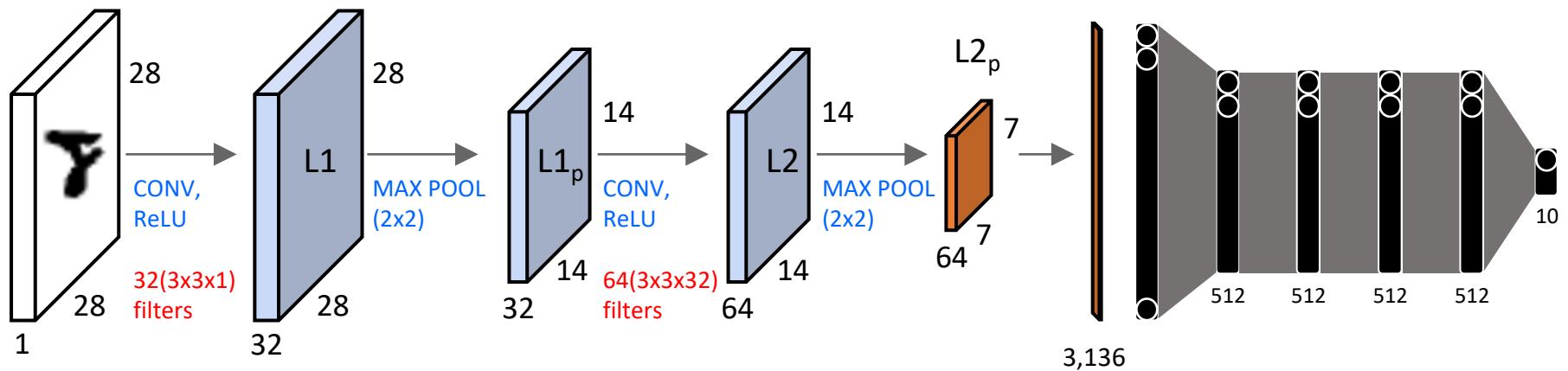


**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

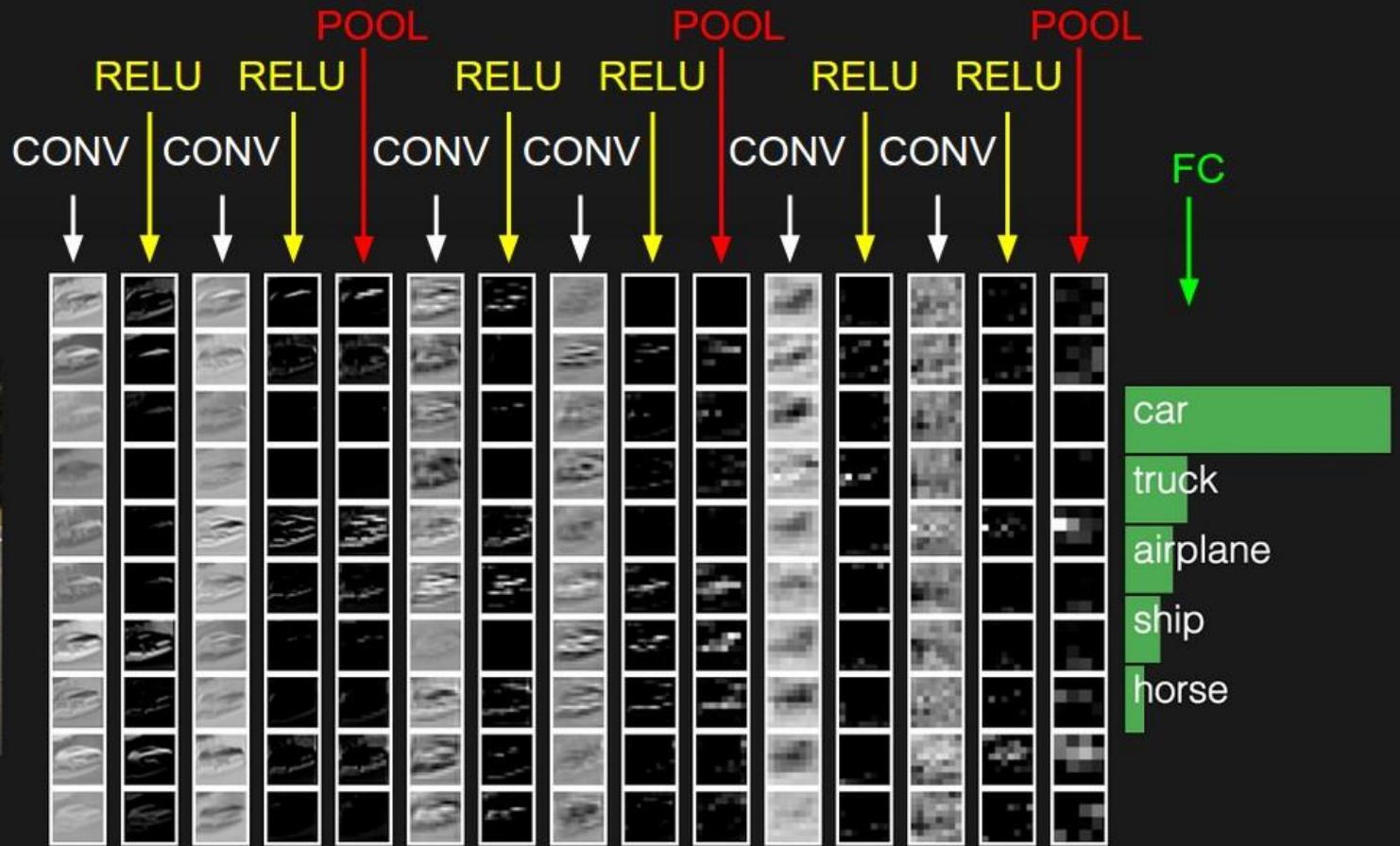
시냅스(연결)은 어디에???





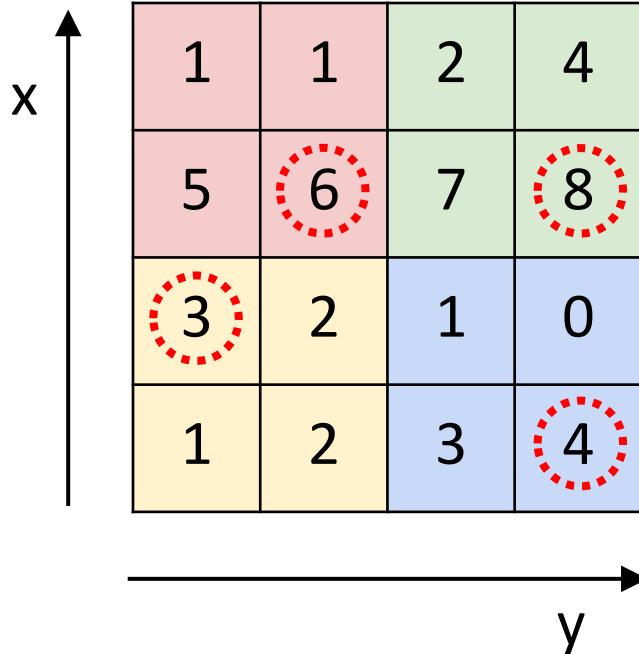


Two more layers to go: POOL/FC



# MAX POOLING

Single depth slice

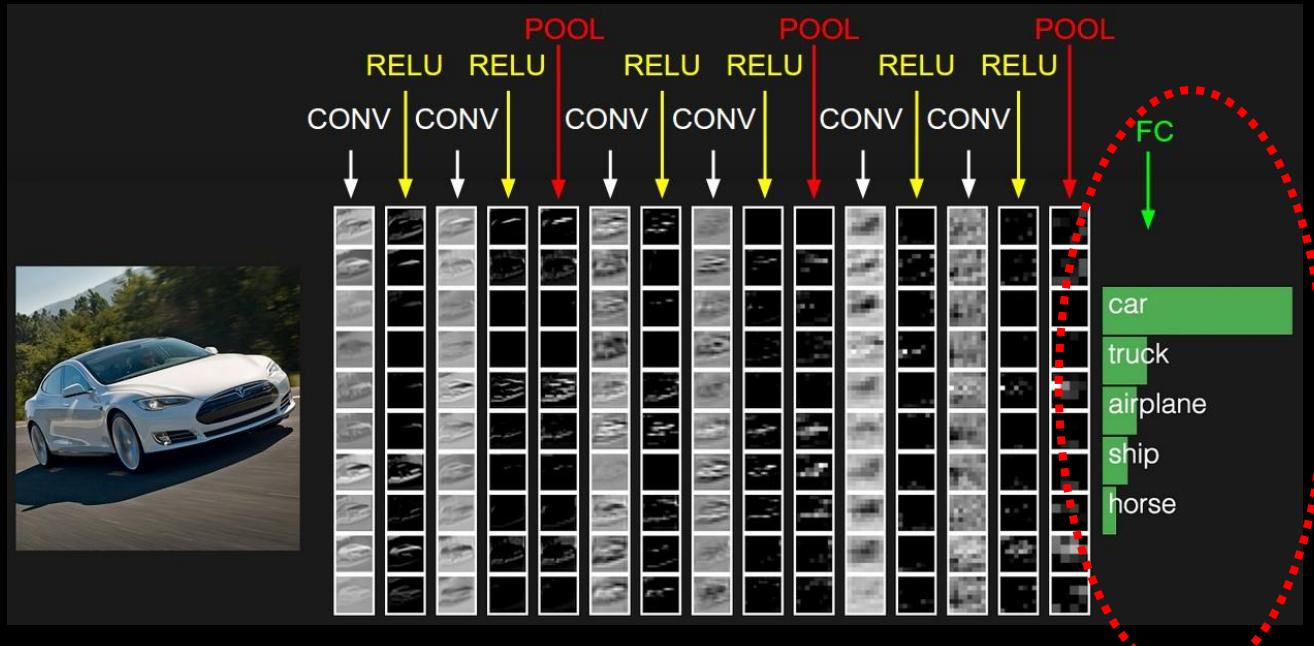


max pool with 2x2 filters  
and stride 2

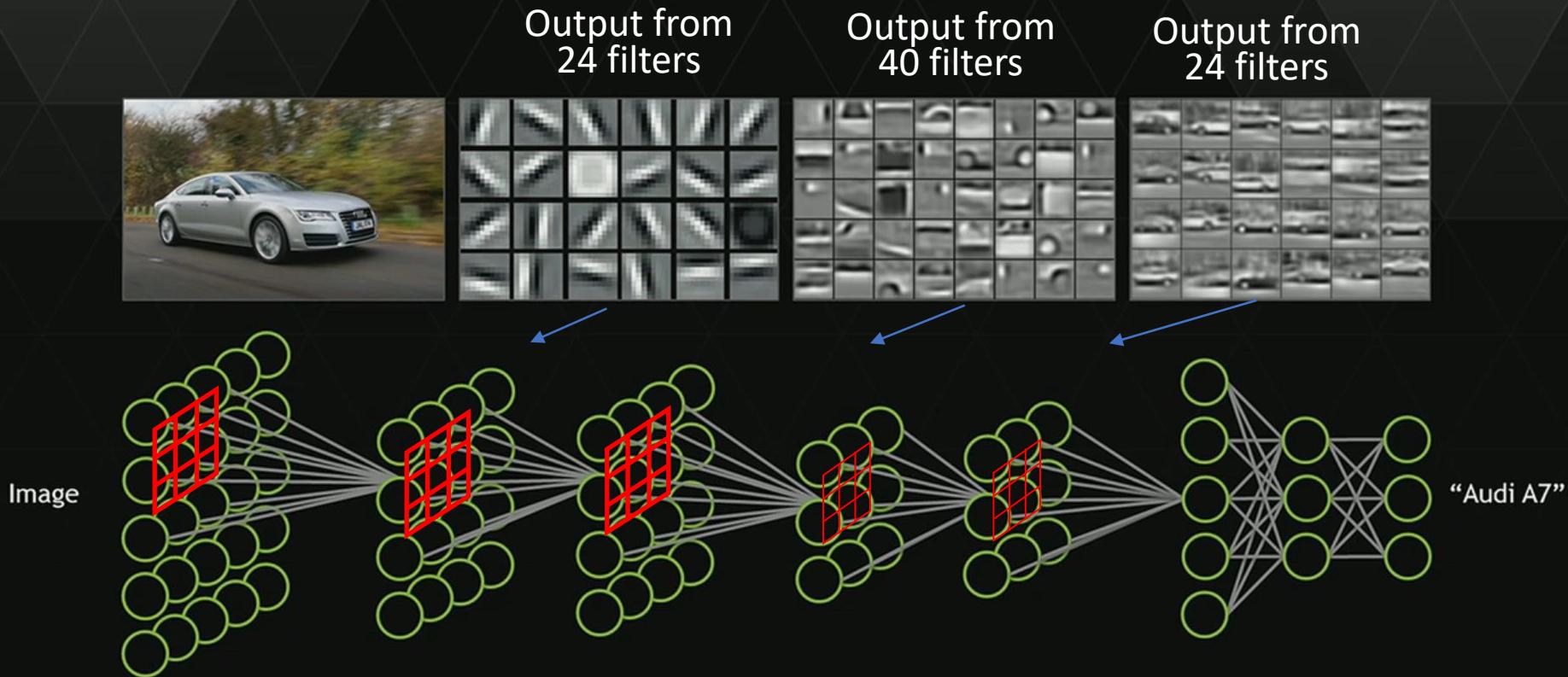
6	8
3	4

# Fully Connected Layer (FC layer)

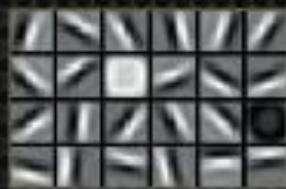
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



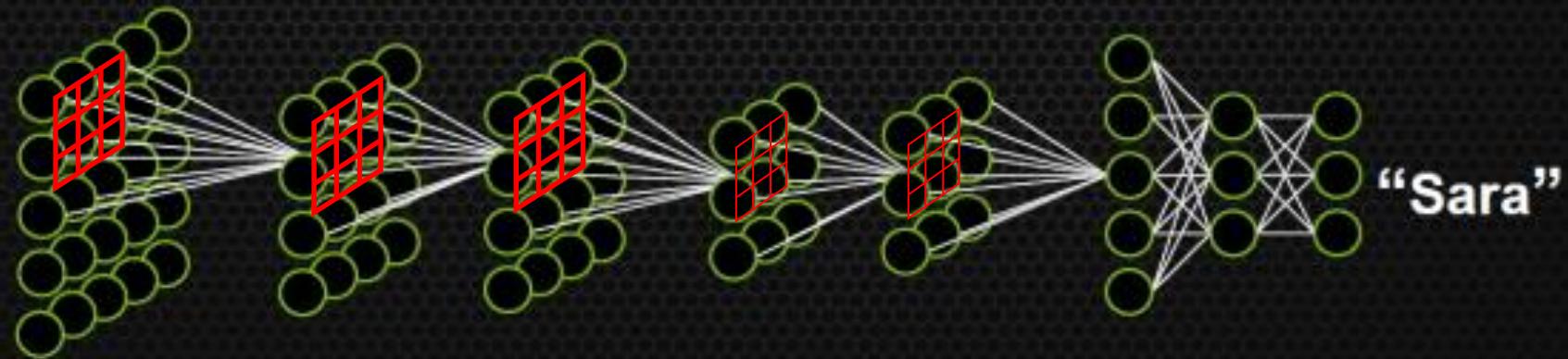
# HOW A DEEP NEURAL NETWORK SEES



*Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011.  
Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.*



Image



# Learning in CNN

- Parameter tuning in filters  
→ filter organizing
- Parameter tuning in fully connected layers

# Theoretical backgrounds of CNN

# A bit of history:

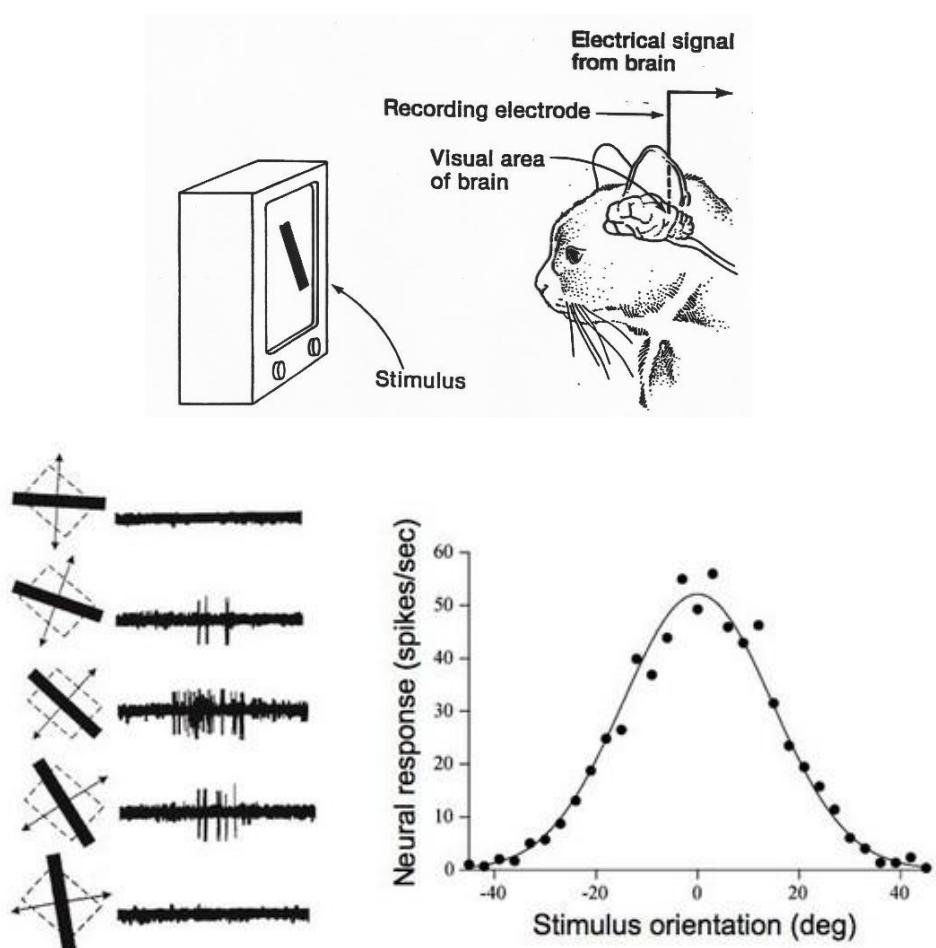
## Hubel & Wiesel, 1959

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

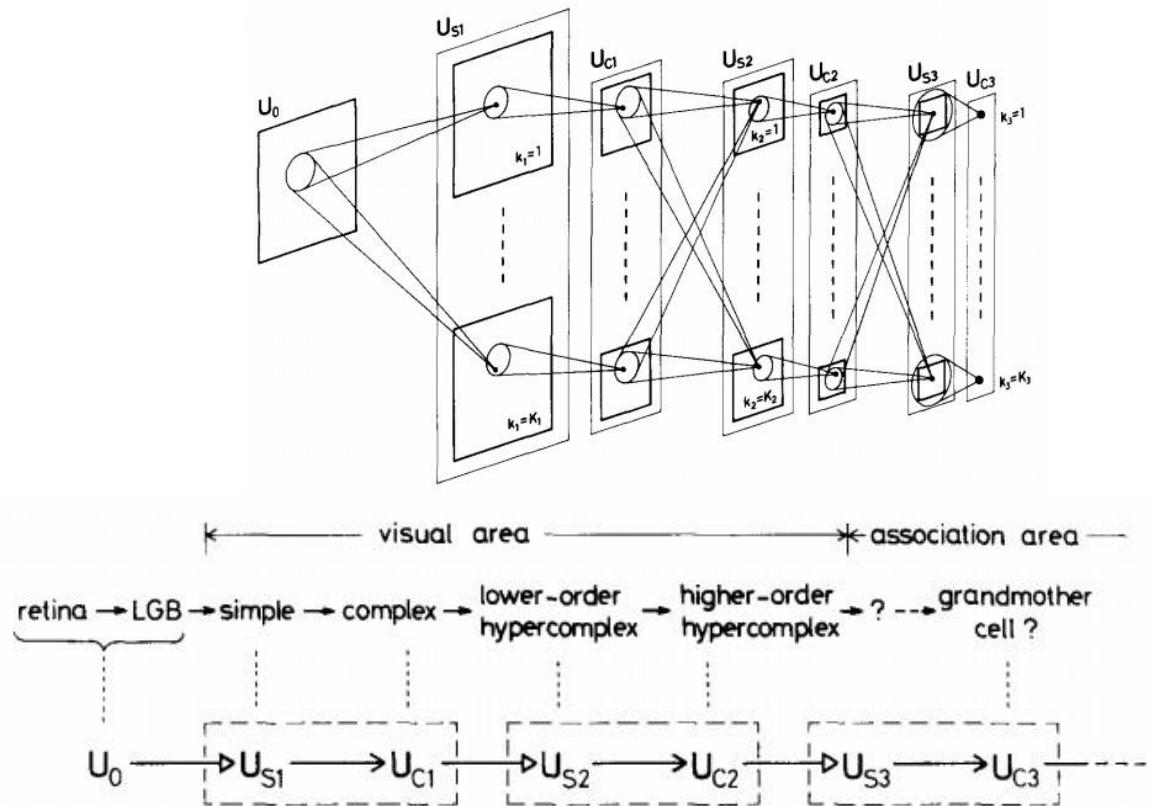
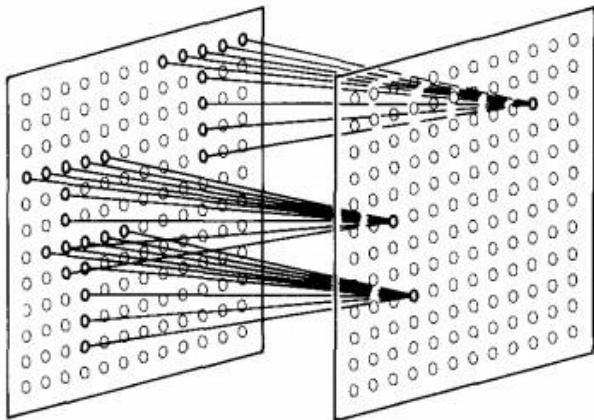
1968...



# A bit of history:

“sandwich” architecture (SCSCSC...)  
simple cells: modifiable parameters  
complex cells: perform pooling

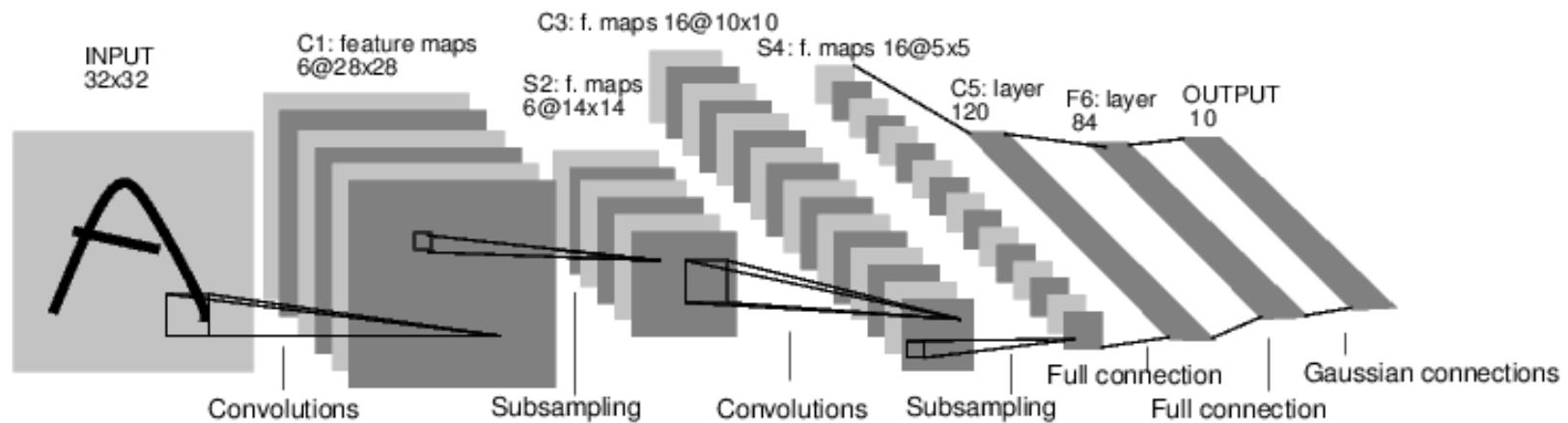
## Neuro-cognitron [Fukushima 1980]



# LeNet-5

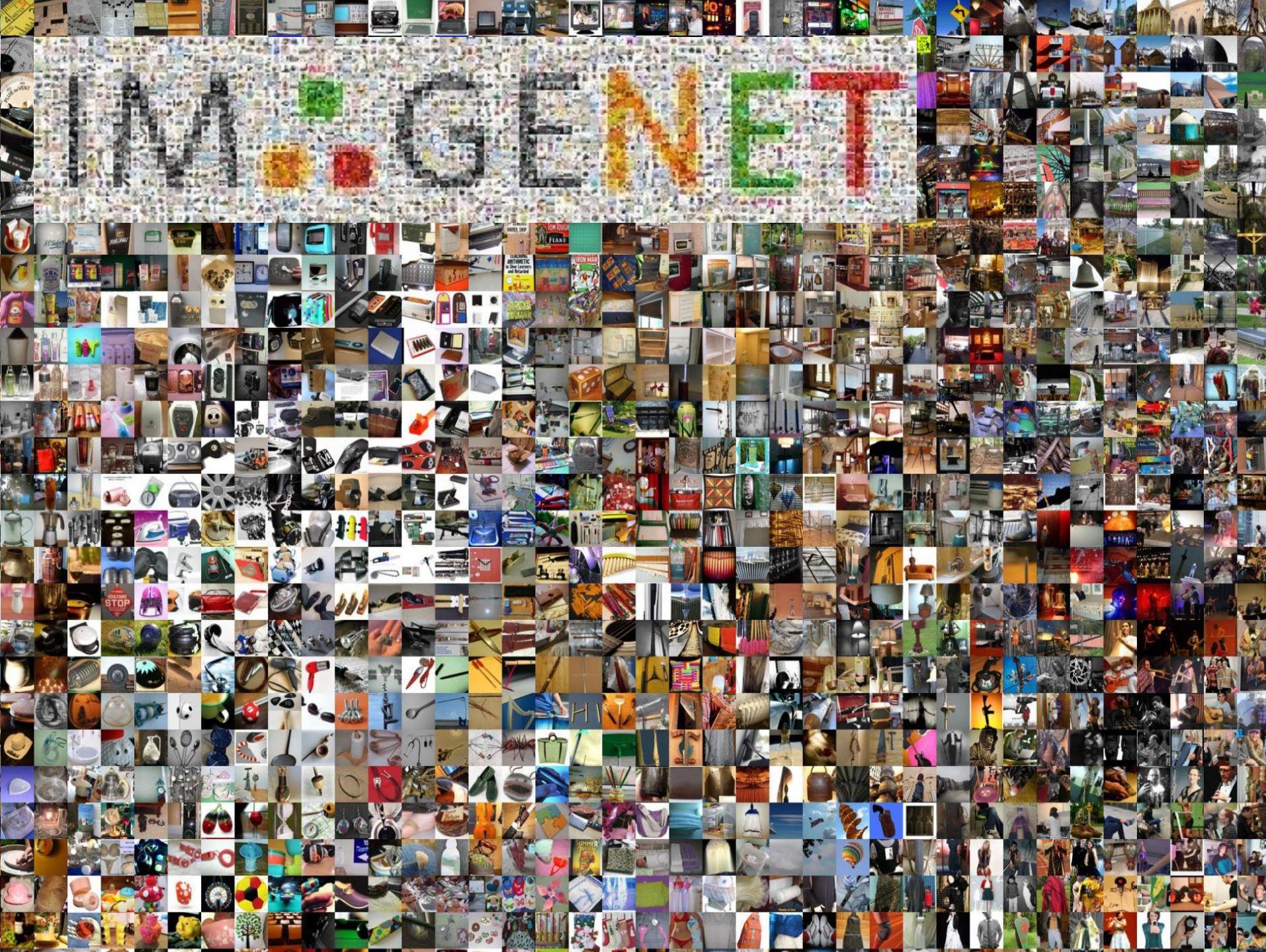
## Gradient-based learning applied to document recognition

[*LeCun, Bottou, Bengio, Haffner 1998*]



Convolution-Pooling-Convolution-Pooling-FC

# Case Studies

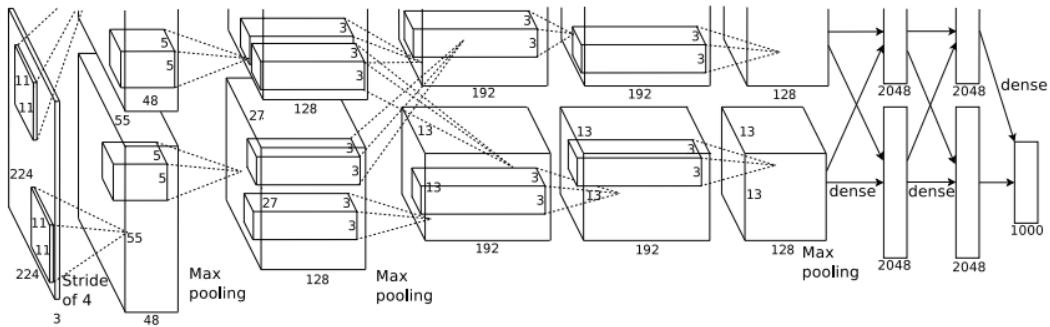


# ImageNet and Competition

- **ImageNet**: categorization of huge amount of images according to WordNet schema
- database for ILSVRC (ImageNet Large Scale Visual Recognition **Competition**)
- Largest Computer Vision Challenge showing state-of-the-art performance on the dataset every year
- firstly collected by Deng et al. in 2009
- And it has been organized by Stanford University Vision Lab (prof. Fei-Fei Li) since 2010.

# 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

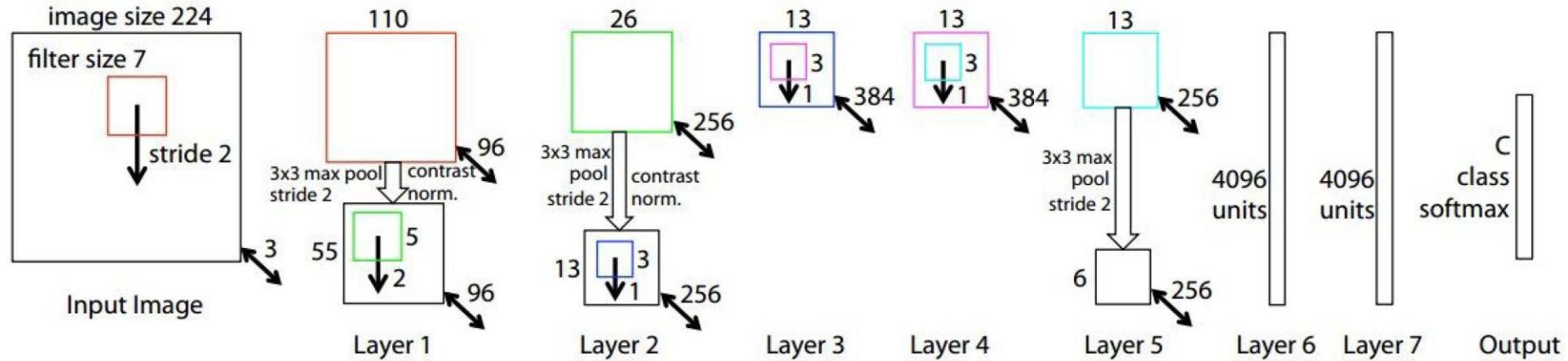
[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

# Case Study: ZFNet

[Zeiler and Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 15.4% -> 14.8%

# Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1  
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013

->

7.3% top 5 error

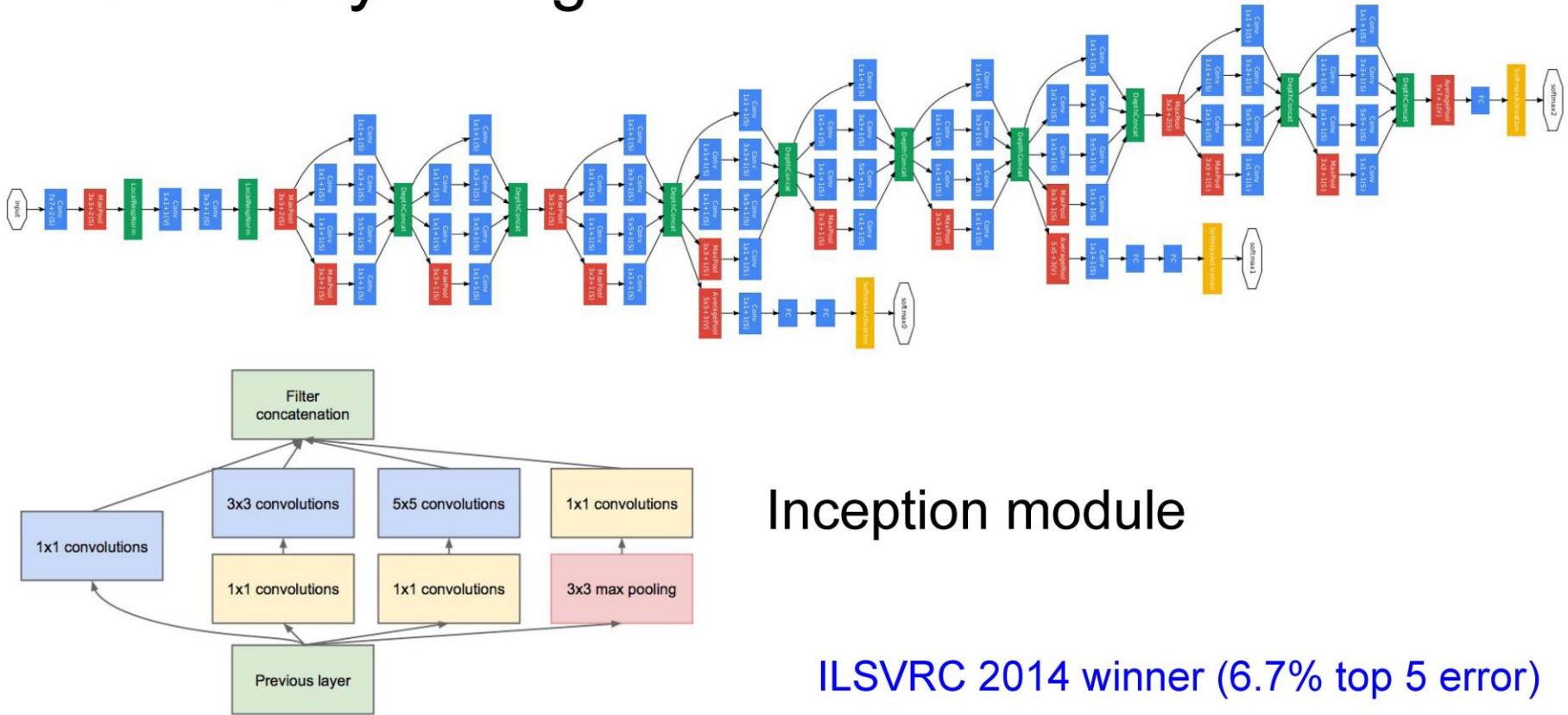
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128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

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

# 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)

Microsoft  
Research

## MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer nets**
  - ImageNet Detection: **16%** better than 2nd
  - ImageNet Localization: **27%** better than 2nd
  - COCO Detection: **11%** better than 2nd
  - COCO Segmentation: **12%** better than 2nd

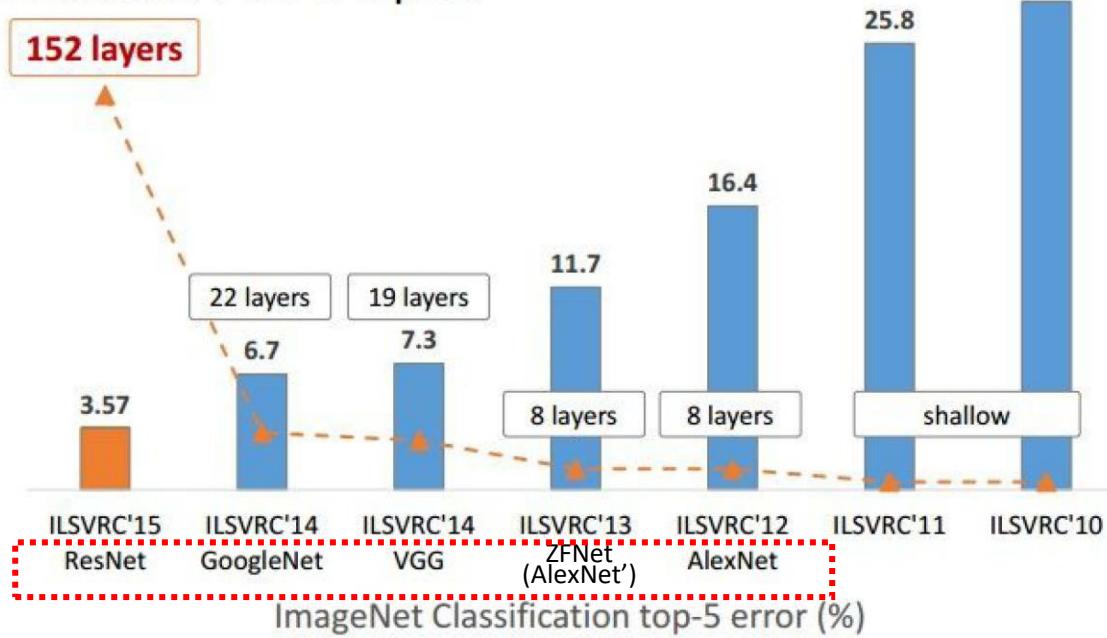
\*improvements are relative numbers

ICCV15  
International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide from Kaiming He's recent presentation <https://www.youtube.com/watch?v=1PGLj-uKT1w>

## Revolution of Depth



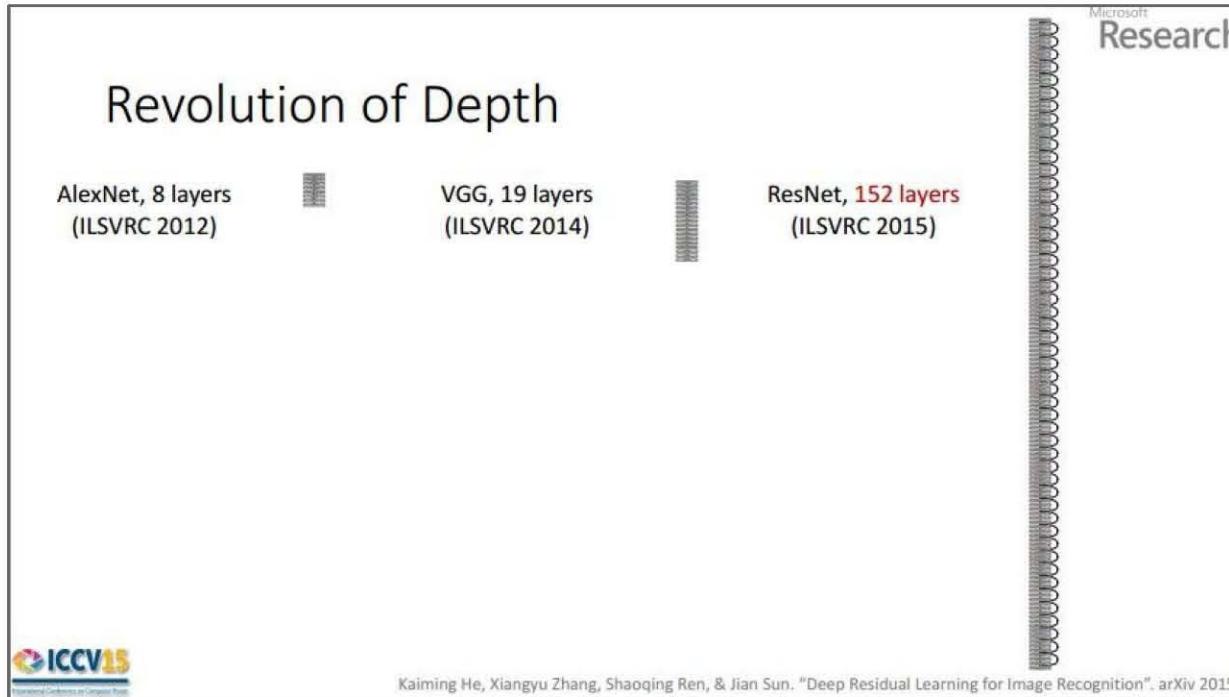
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

(slide from Kaiming He's recent presentation)

# Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

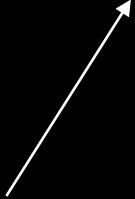


2-3 weeks of training  
on 8 GPU machine

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

(slide from Kaiming He's recent presentation)

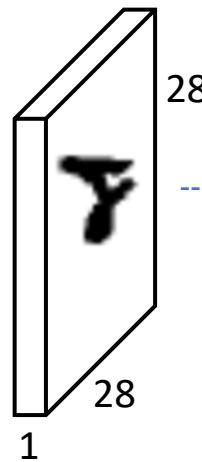
Dot product & shift for feature extraction



# CNN, Convolutional Neural Network

20\_mnist\_softmax.py

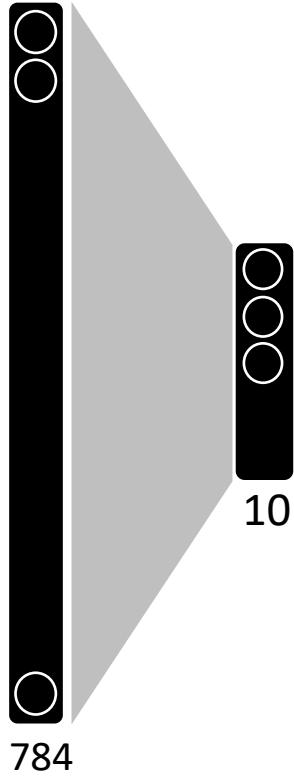
Accuracy: 90.23



28

28

1



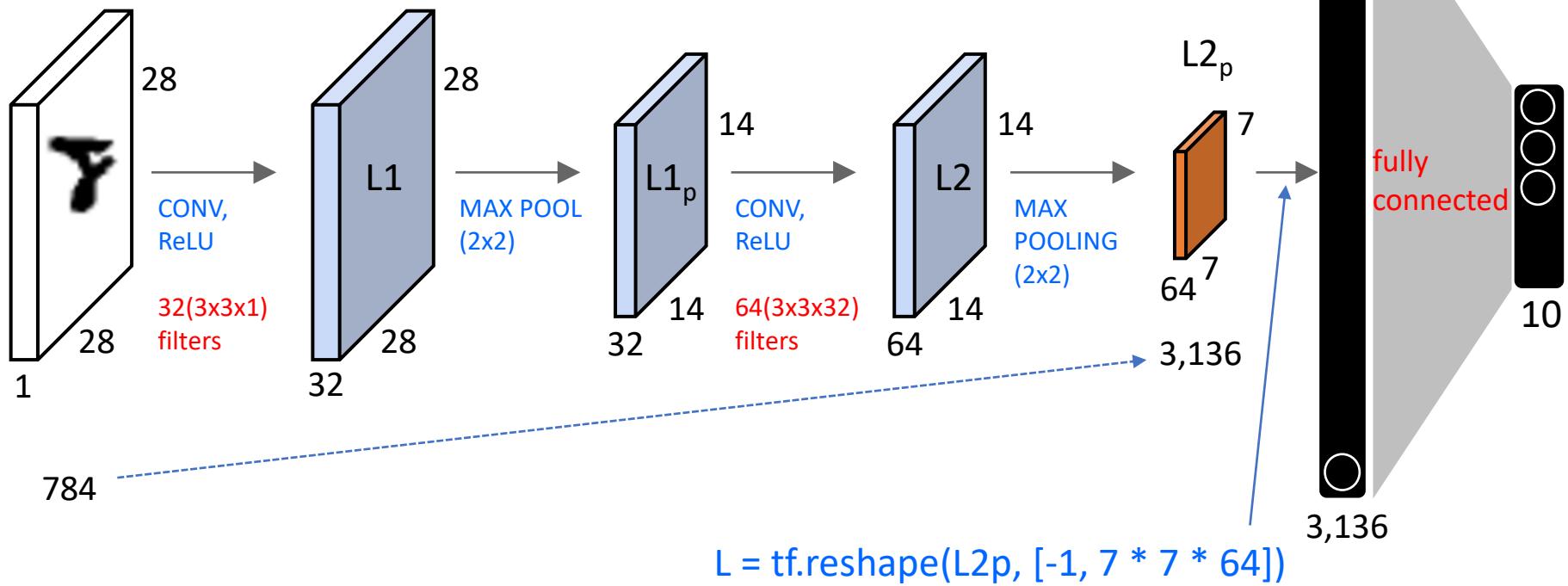
784

10

3

## 26\_mnist\_softmax.py

Accuracy: 98.85



Accuracy: ?????

