



Chapter

1

CNN 개요

1. CNN 소개
2. CNN 기본

학습 목표

- ✓ CNN이 기본적으로 어떤 곳에 쓰일 수 있는지 이해한다
- ✓ Convolution 을 이해하고 레이어를 설계에 익숙해진다

주요 내용

- ✓ CNN의 정의와 필요성
- ✓ Convolution, Pooling Layer 설계 방법
- ✓ Binary, Multinomial Logistic Regression 을 통한 Cross Entropy 설명

강의에 앞서서..

❖ 본 문서는 아래의 자료들을 활용하여 만들어 졌음을 알립니다

❖ 모두를 위한 딥러닝 강좌

- 네이버 Search & Clova AI 부분 리더 김성훈 교수님
- https://www.youtube.com/playlist?list=PLIMkM4tgfjnLSOjrEJN31gZATbcj_MpUm
- <https://www.edwith.org/boostcourse-dl-tensorflow/lecture/43739/>

❖ 스탠포드 대학 CNN 강좌

- Fei-Fei Li & Andrej Karpathy & Justin Johnson
- <http://cs231n.stanford.edu/slides/2020/>

CS231n: Convolutional Neural Networks for Visual Recognition

- This course, Prof. Fei-Fei Li & Justin Johnson & Serena Yeung
- Focusing on applications of deep learning to computer vision

1

1. CNN 소개



Image Classification

Fei-Fei Li, Ranjay Krishna, Danfei Xu

Lecture 2 - 5

April 9, 2020

Image Classification: A core task in Computer Vision



This image by [hiita](#) is
licensed under CC-BY 2.0

(assume given a set of labels)
{dog, cat, truck, plane, ...}



Image Classification

Fei-Fei Li, Ranjay Krishna, Danfei Xu

Lecture 2 - 14

April 9, 2020

Attempts have been made



Find edges



Find corners

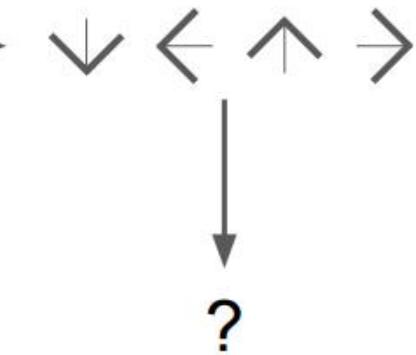


Image Classification

Fei-Fei Li, Ranjay Krishna, Danfei Xu

Lecture 2 - 15

April 9, 2020

Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning algorithms to train a classifier
3. Evaluate the classifier on new images

Example training set

```
def train(images, labels):
    # Machine learning!
    return model

def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```

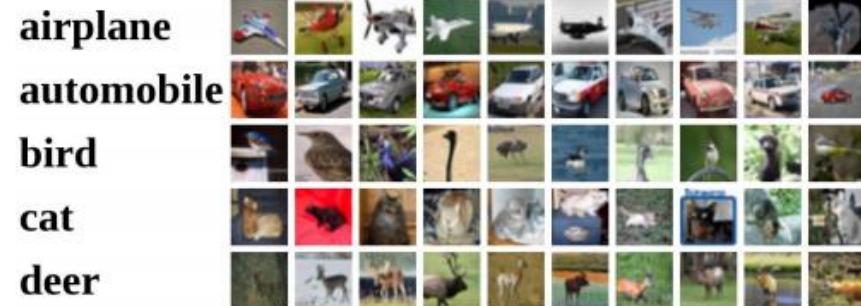
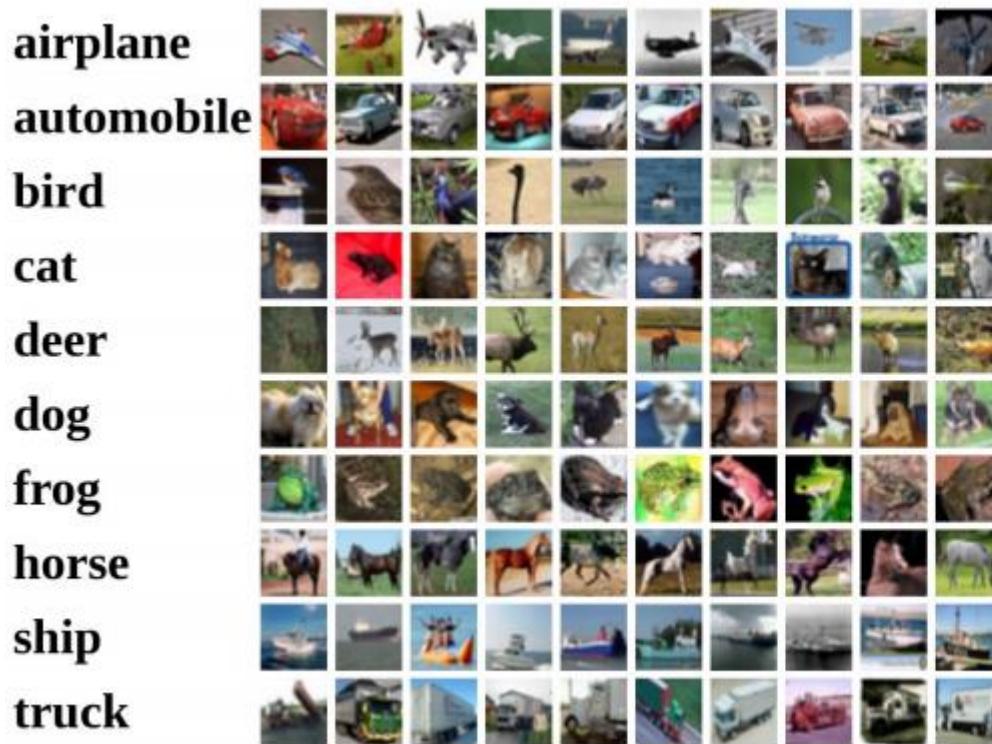


Image Classification

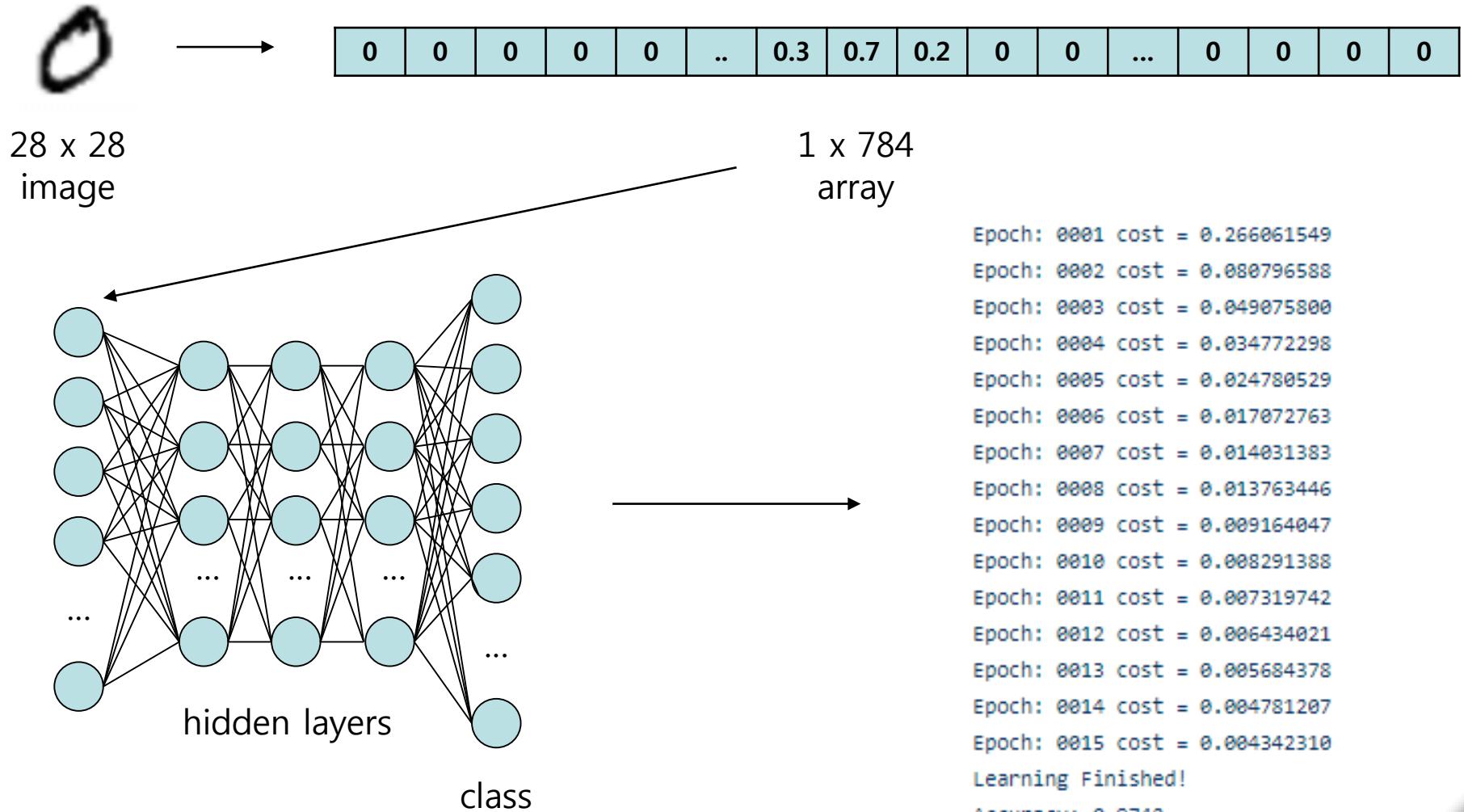
❖ CIFAR-10 dataset



50,000 training images
each image is **32x32x3**

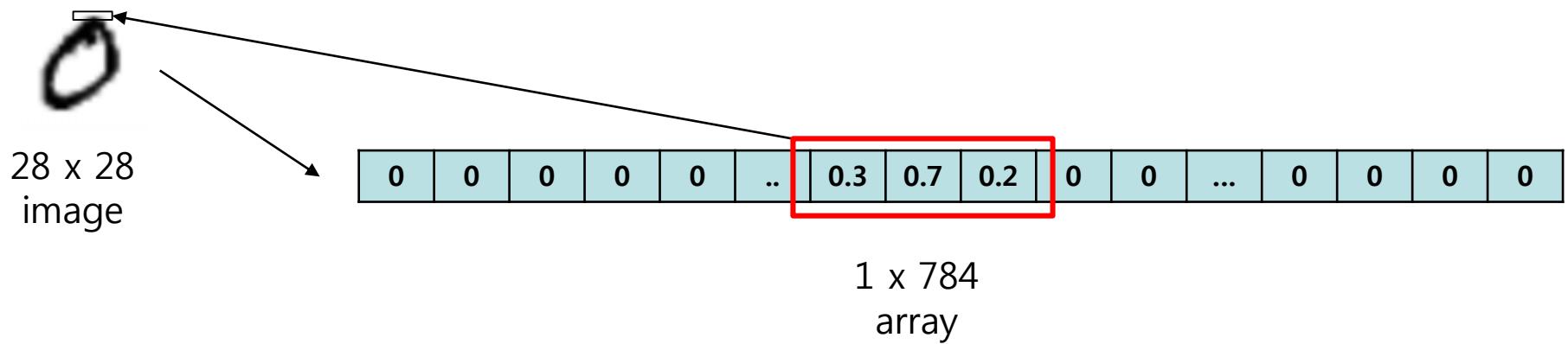
10,000 test images.

MNIST with Neural Network



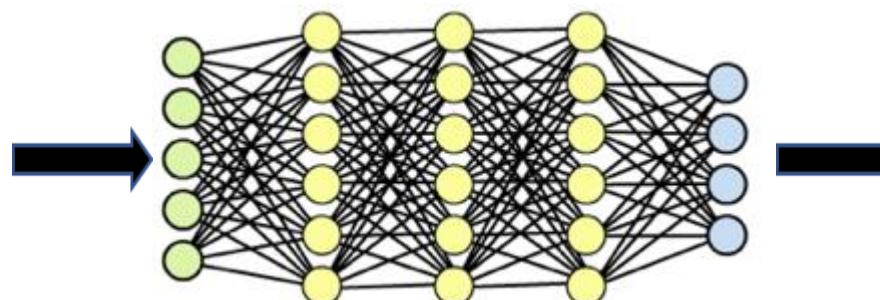
MNIST with Neural Network

- ❖ Neural network은 매우 강력하지만 전체 데이터로는 feature selection이 어렵다

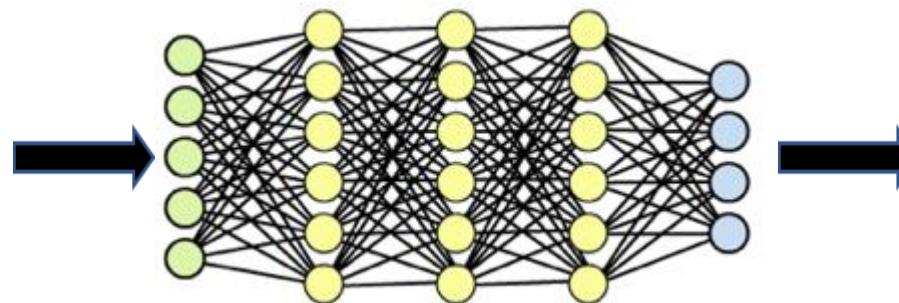


Deep Neural Network

- 데이터의 크기가 클수록, 형태가 복잡할수록 학습이 어려워짐
 - 학습시간(training time), 네트워크 크기(network size), 매개변수의 개수(parameter)

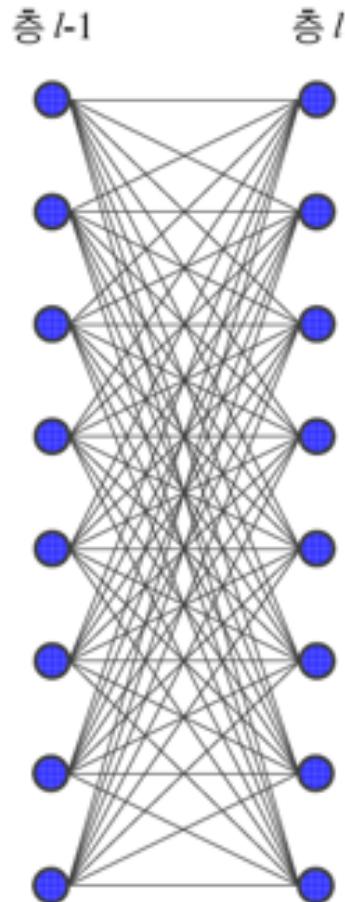


0(고양이)

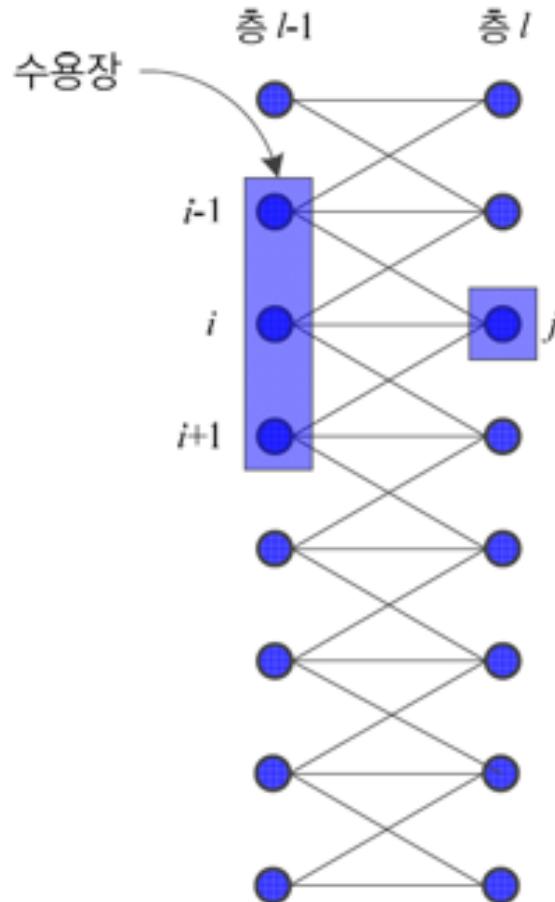


1(강아지)

Deep Neural Network & Convolutional Neural Network



(a)DNN (fully connected)



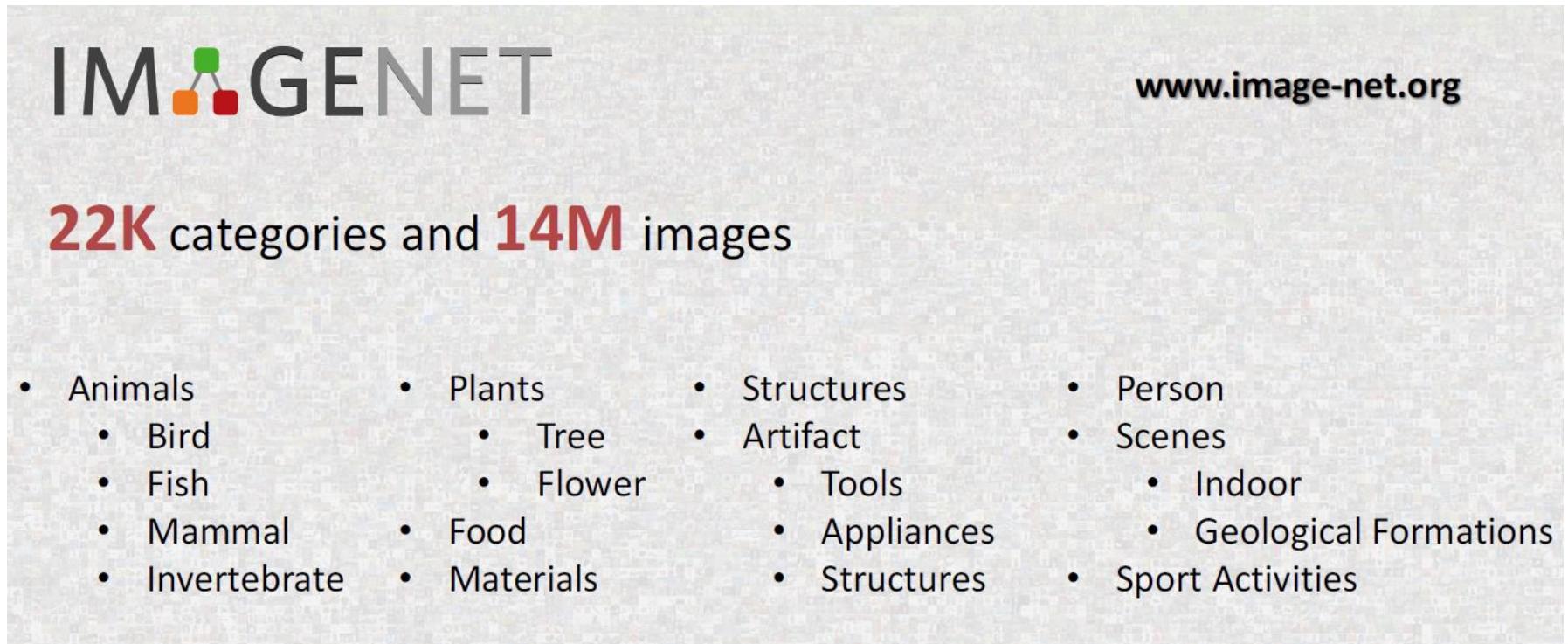
(b)CNN (sparse connected)

- 각 레이어의 입력력 데이터의 형상 유지
- 이미지의 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식
- 복수의 필터로 이미지의 특징 추출 및 학습
- 추출한 이미지의 특징을 모으고 강화하는 Pooling 레이어
- 필터를 공유 파라미터로 사용하기 때문에, 일반 인공 신경망과 비교하여 학습 파라미터가 매우 적음

Visual Object Challenge

❖ IMAGENET

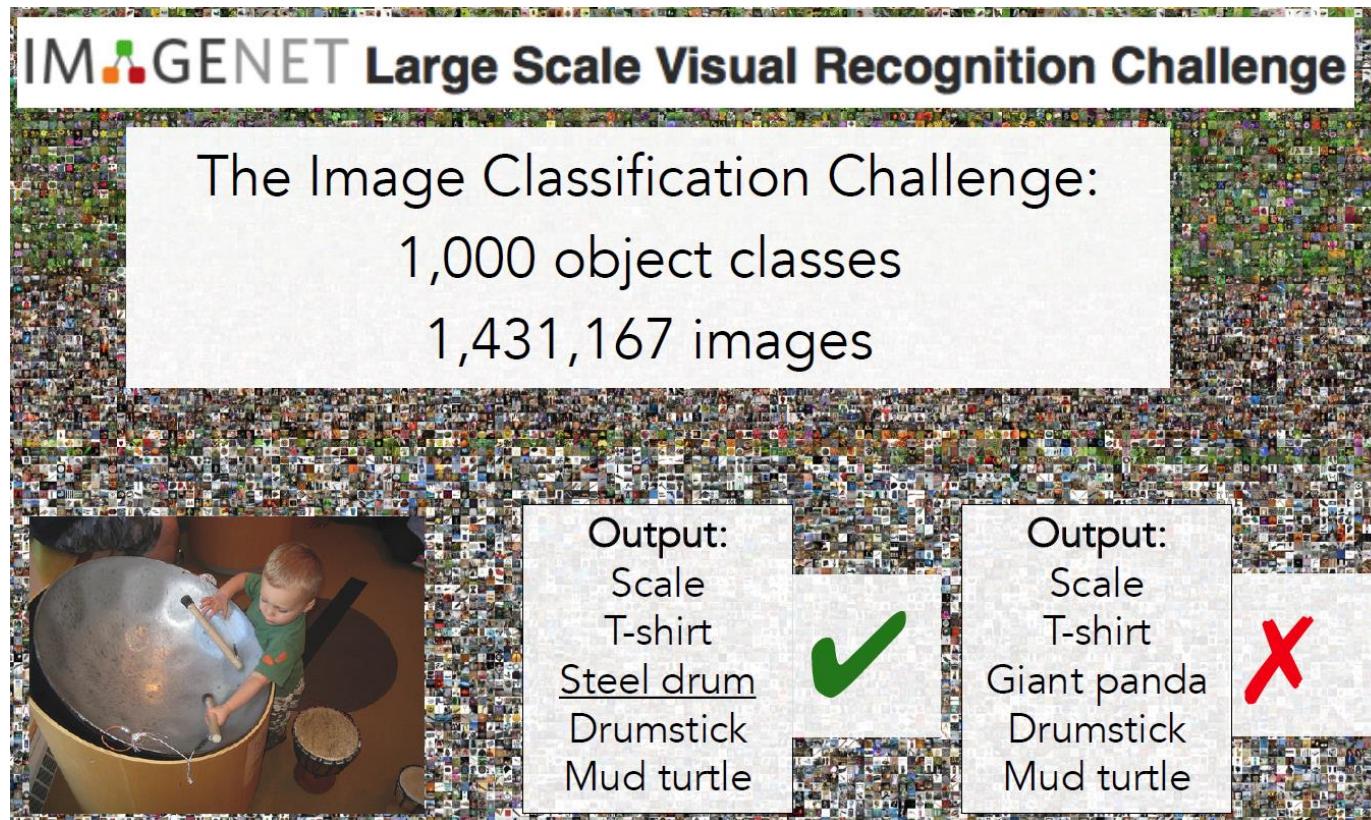
- Visual object recognition 연구를 위한 매우 큰 데이터베이스
- 22000 카테고리에 140만개의 이미지



Visual Object Challenge

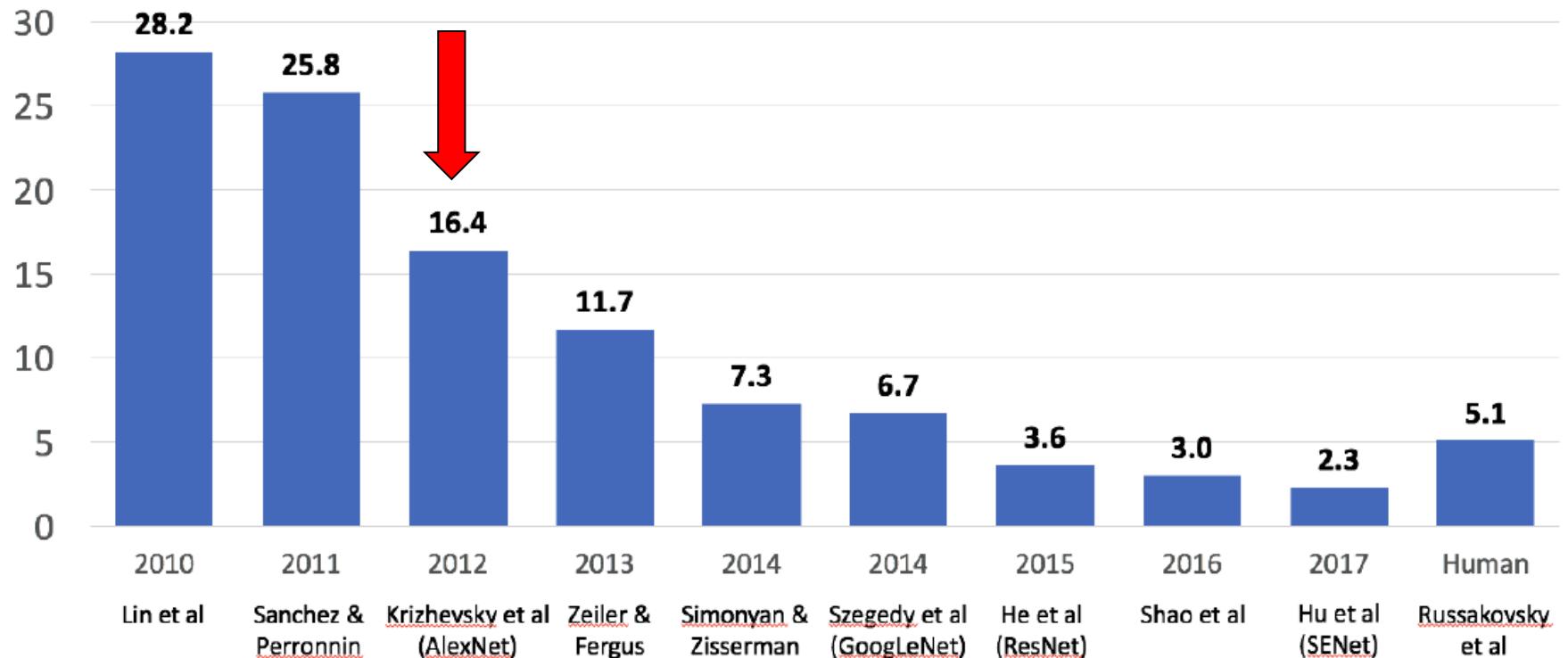
❖ ImageNet Large Scale Visual Recognition Competition(ILSVRC)

- 1000개의 카테고리, 140만개의 이미지로 이미지 인식 대회 개최



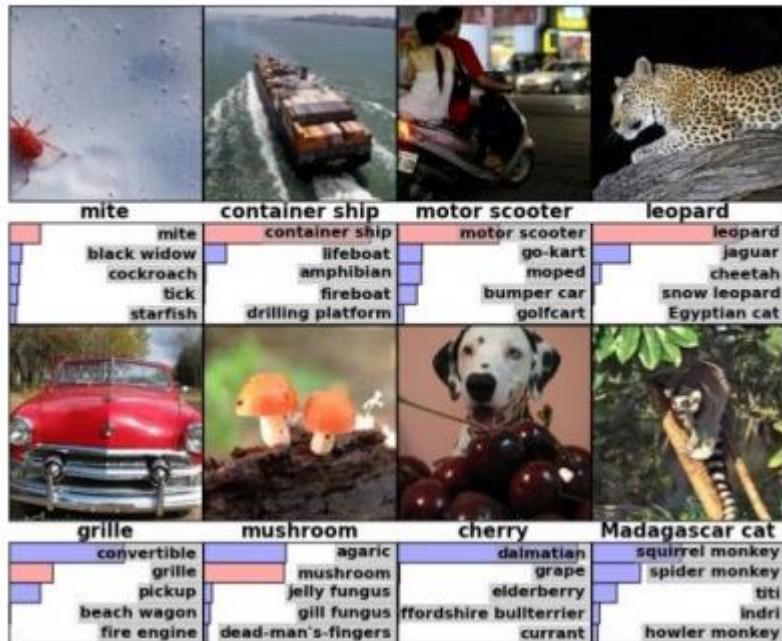
Visual Object Challenge

- 2012년 갑자기 16.4%로 오답률이 낮아짐 -> CNN
- 사람의 오답률이 5.1%인데 2015년에 사람을 뛰어넘음



Visual Recognition Problems

Classification



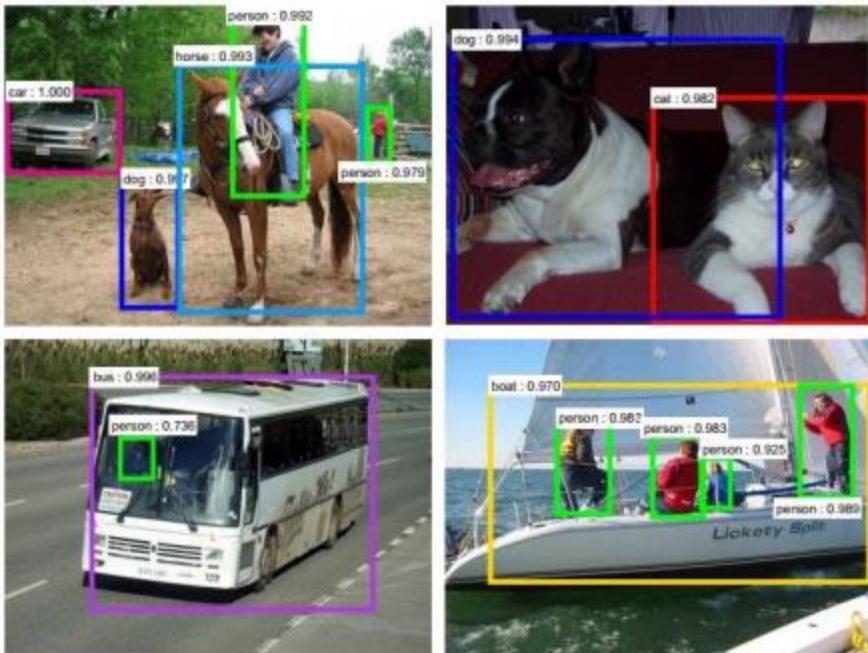
Retrieval



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

Visual Recognition Problems

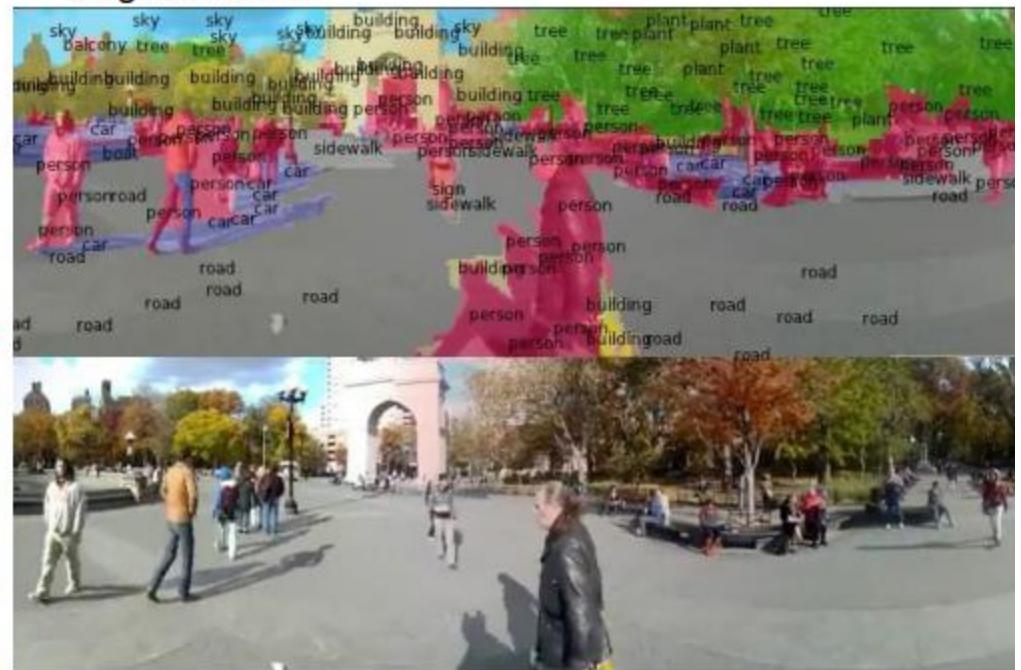
Detection



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

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

Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.

[Farabet et al., 2012]

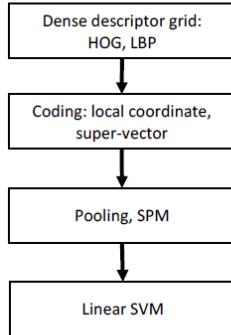
Visual Recognition Problems

- ILSVRC에서 우승한 VGG, ResNet 등의 모델이 Keras에 공개

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

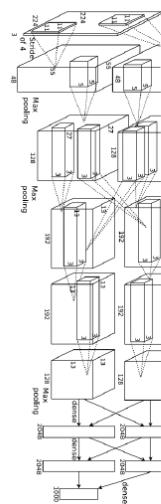


[Lin CVPR 2011]

Lion image by Swissfrog is licensed under CC BY 3.0

Year 2012

SuperVision
AlexNet



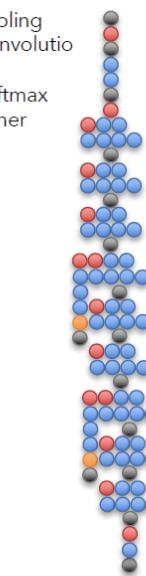
[Krizhevsky NIPS 2012]

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

Year 2014

GoogLeNet
Inception

● Pooling
● Convolution
● n
● Softmax
● Other

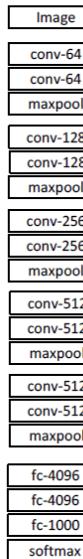


[Szegedy arxiv 2014]

[Simonyan arxiv 2014]

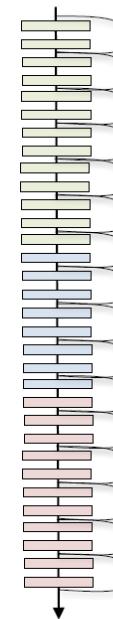
Visual Geometry Group

VGG



Year 2015

MSRA ResNet



[He ICCV 2015]

Microsoft Research

Visual Recognition Problems

Convolutional Neural Networks (CNN) have become an important tool for object recognition

Visual Recognition Problems

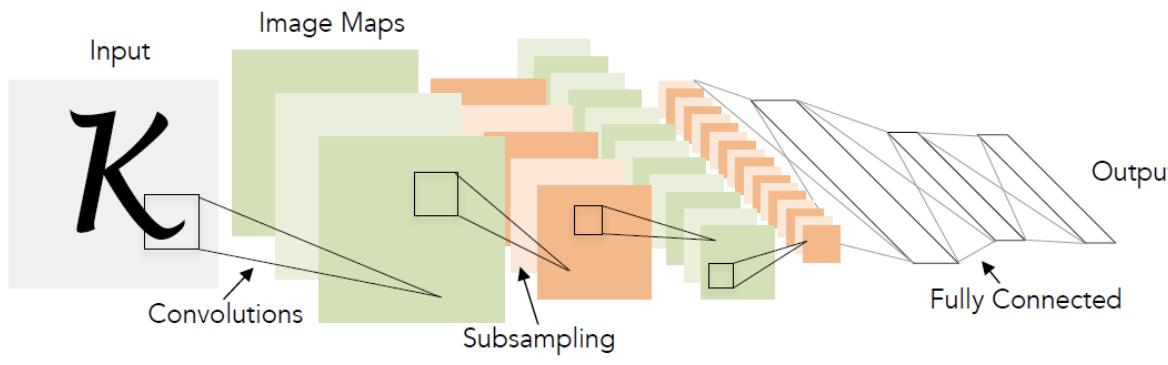
1998

LeCun et al.

LeNet-5

"Gradient-based learning applied to document recognition"

논문에서 CNN을 이용하여 필기체를 성공적으로 인식



of transistors



10^6

of pixels used in training

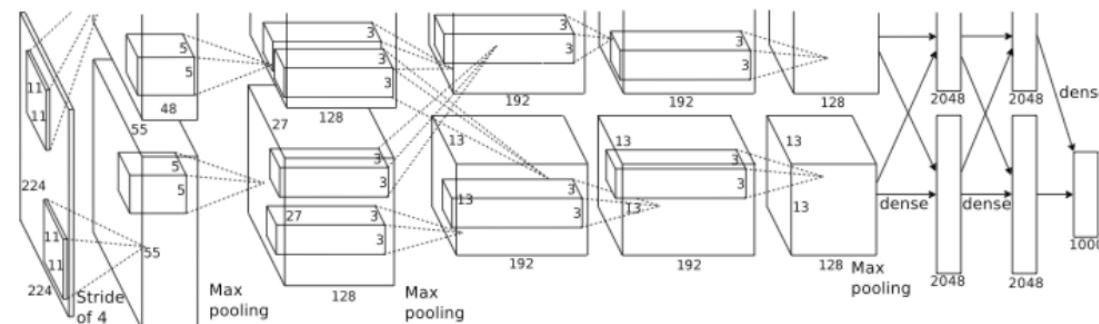
10^7 NIST

2012

Krizhevsky et al.

AlexNet

"ImageNet Classification with Deep Convolutional Neural Networks"



of transistors



10^9

GPUs



of pixels used in training

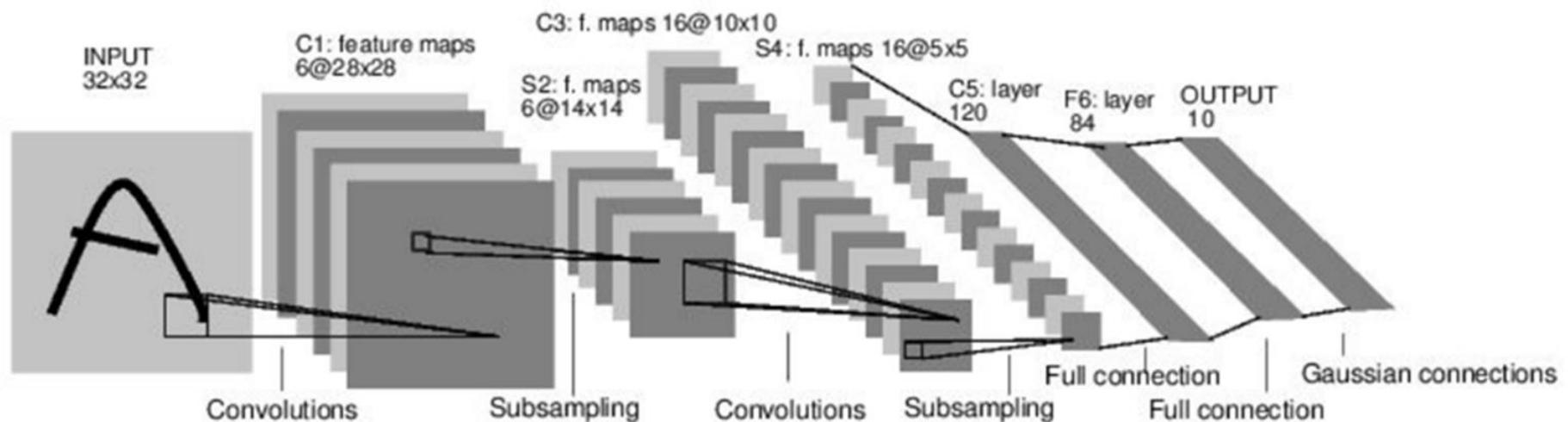
10^{14} IMGENET

Our Goal

Deep Learning 기술 중
이미지 인식에 많이 활용되는
CNN에 대한 이해와 활용능력 습득

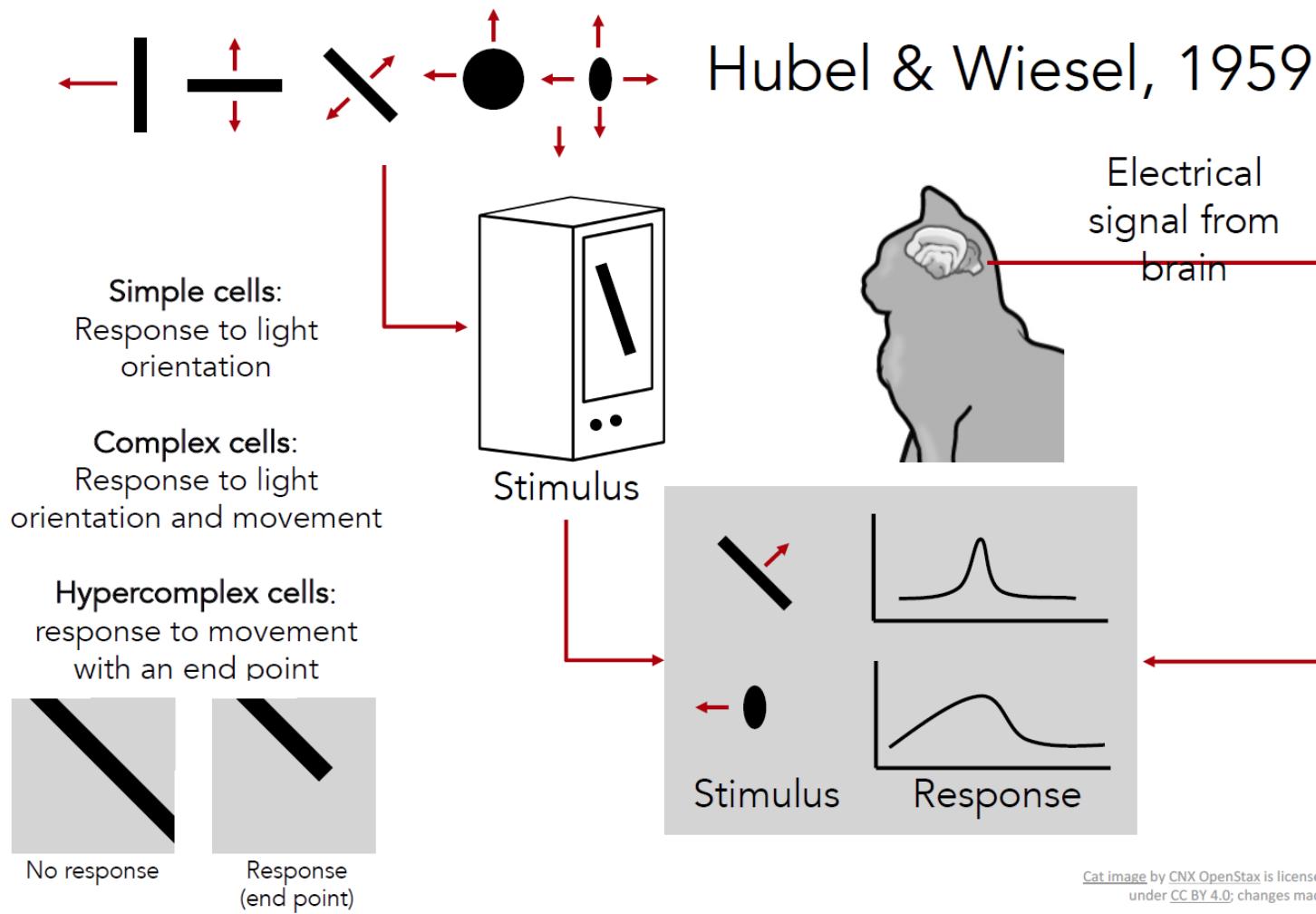
Convolutional Neural Network

- ❖ 이미지의 부분 부분을 샘플링 하여 특징을 추출

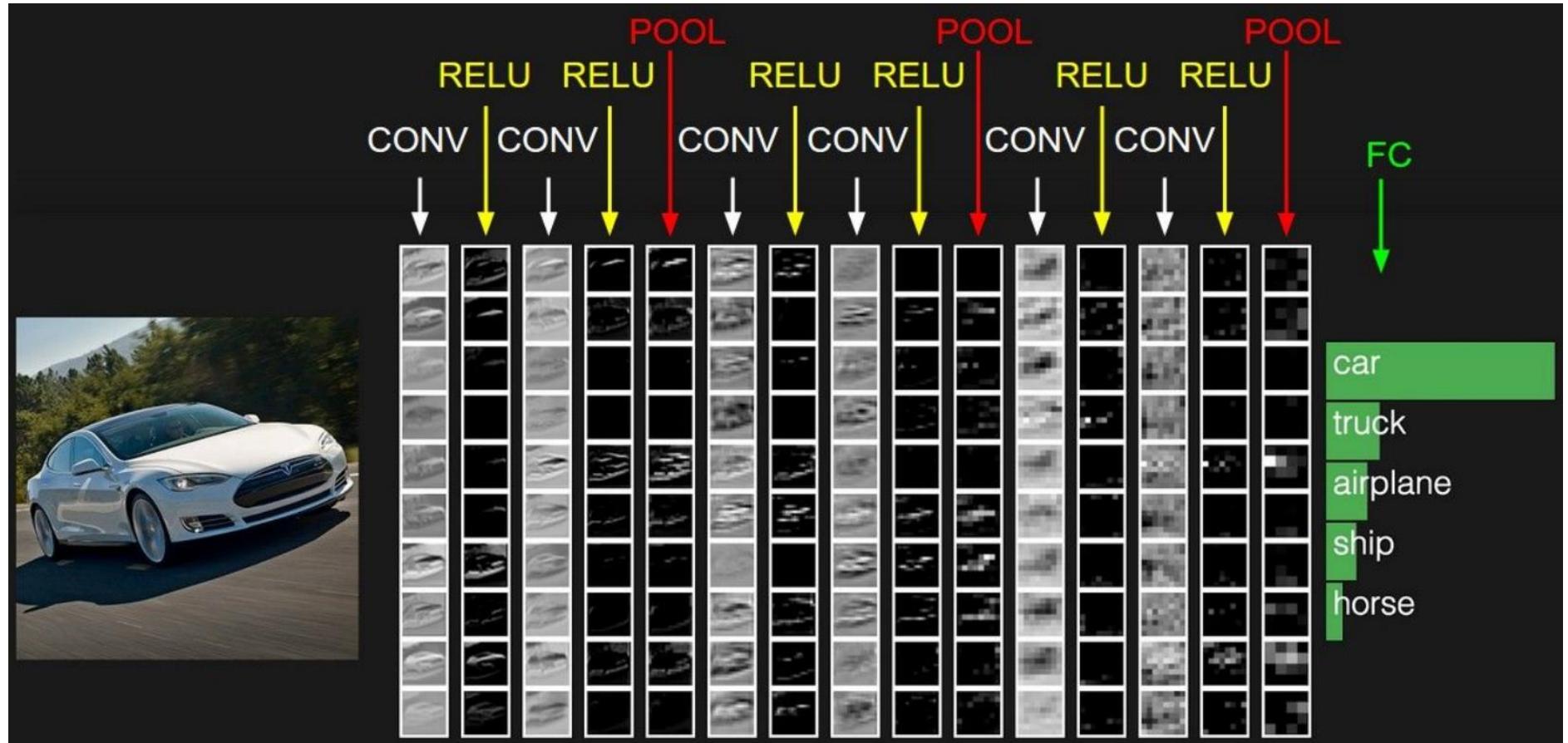


[LeNet-5, LeCun]

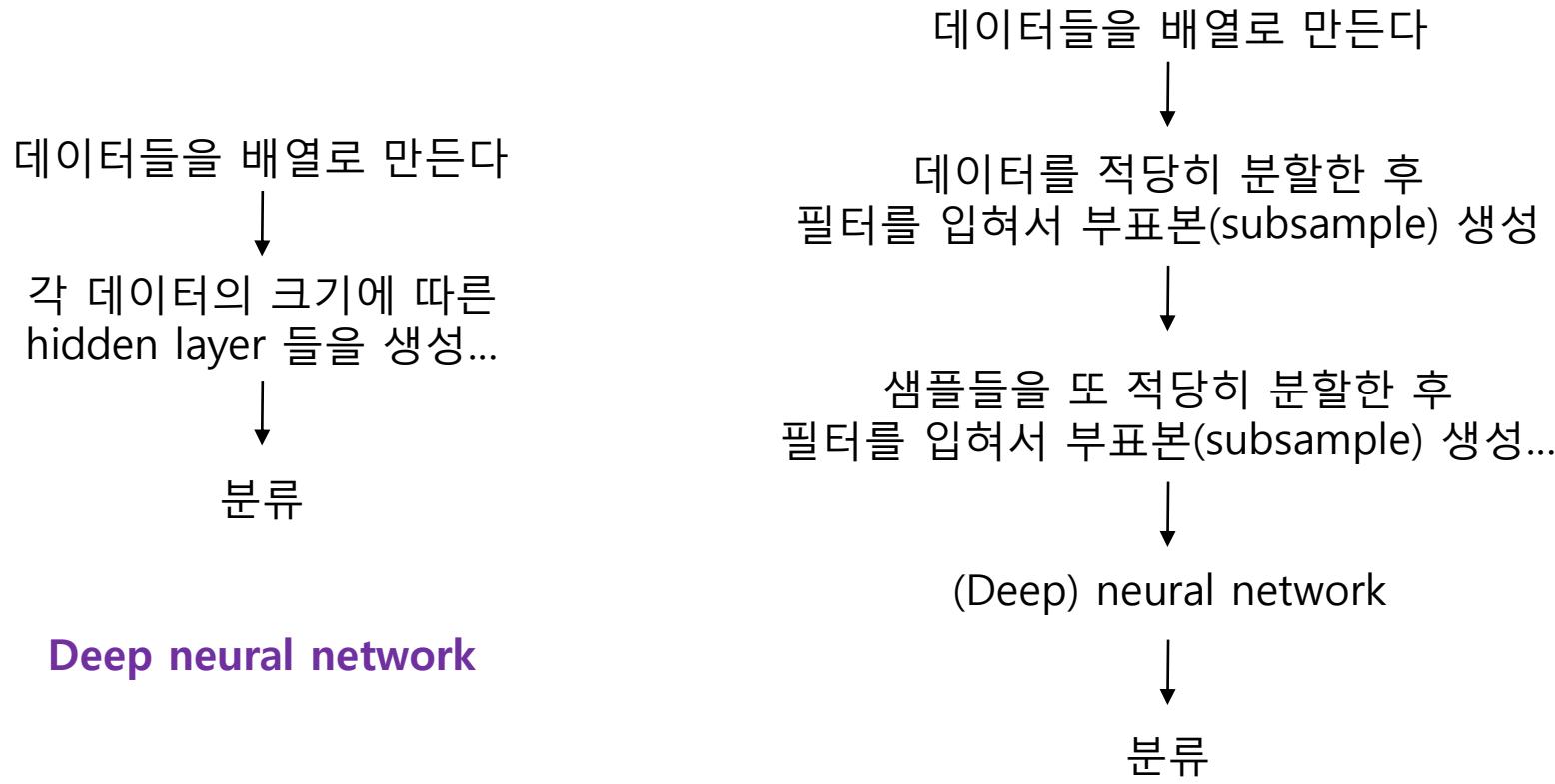
Convolutional Neural Network



Convolutional Neural Network



Convolutional Neural Network

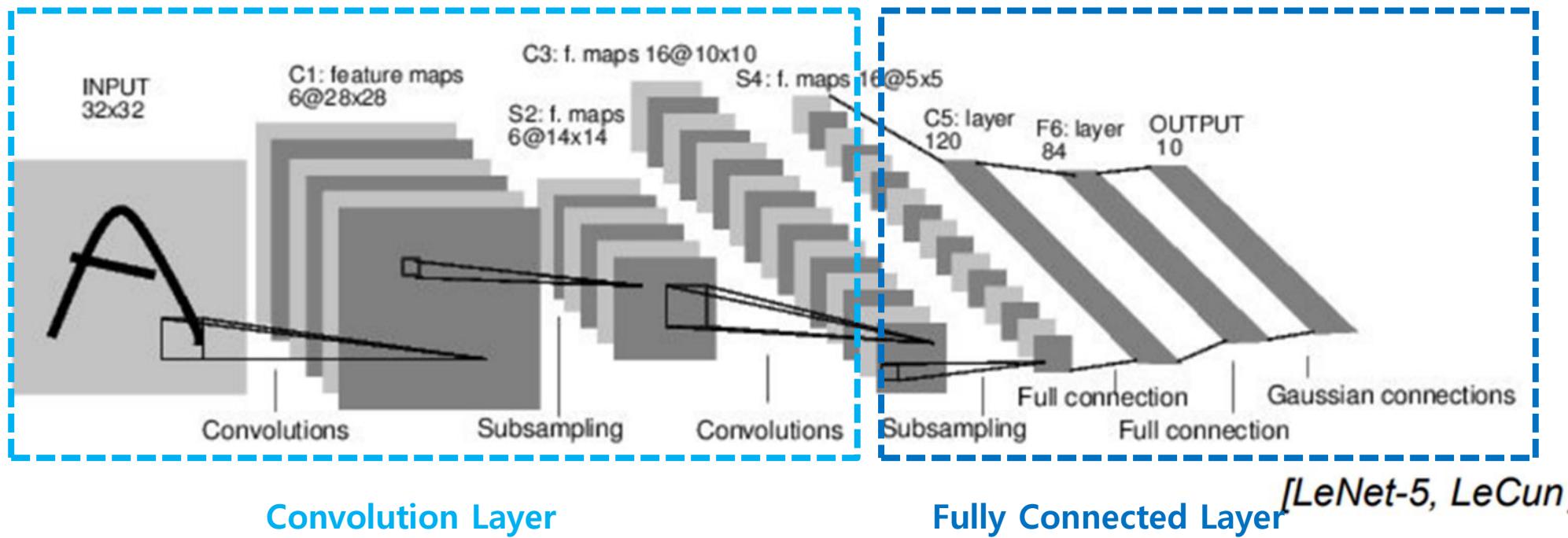


1

2. CNN 기본



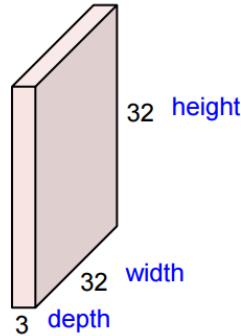
CNN 구조



CNN 구조

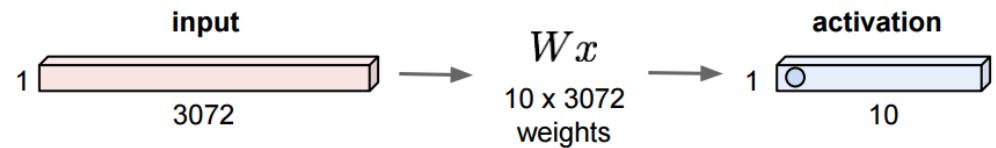
Convolution Layer

32x32x3 image -> preserve spatial structure



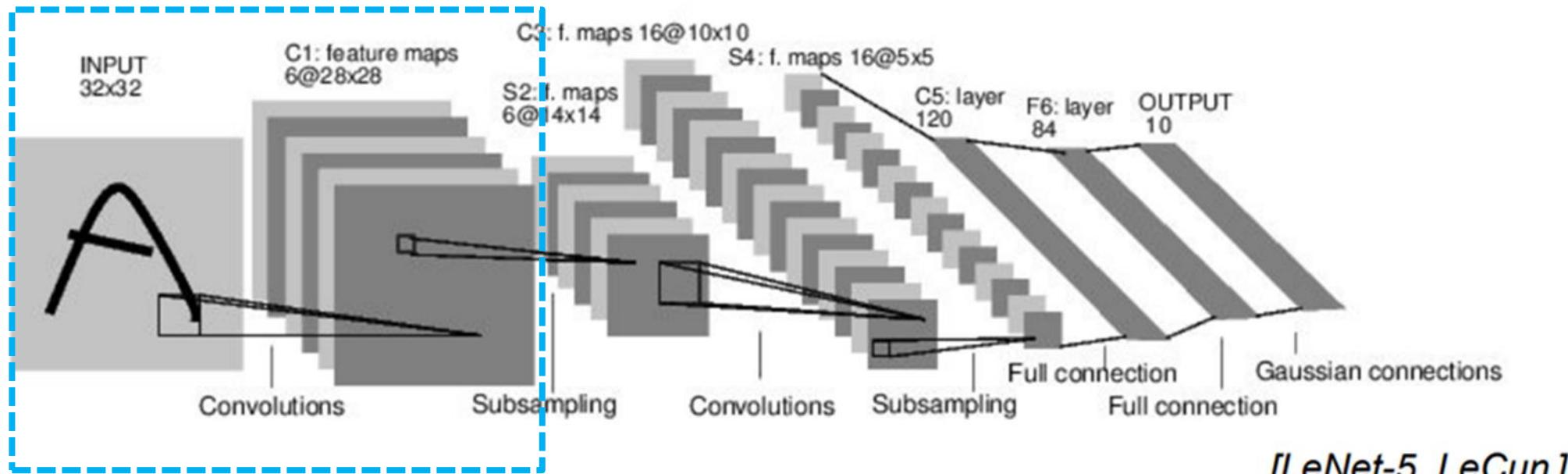
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



Convolution Layer

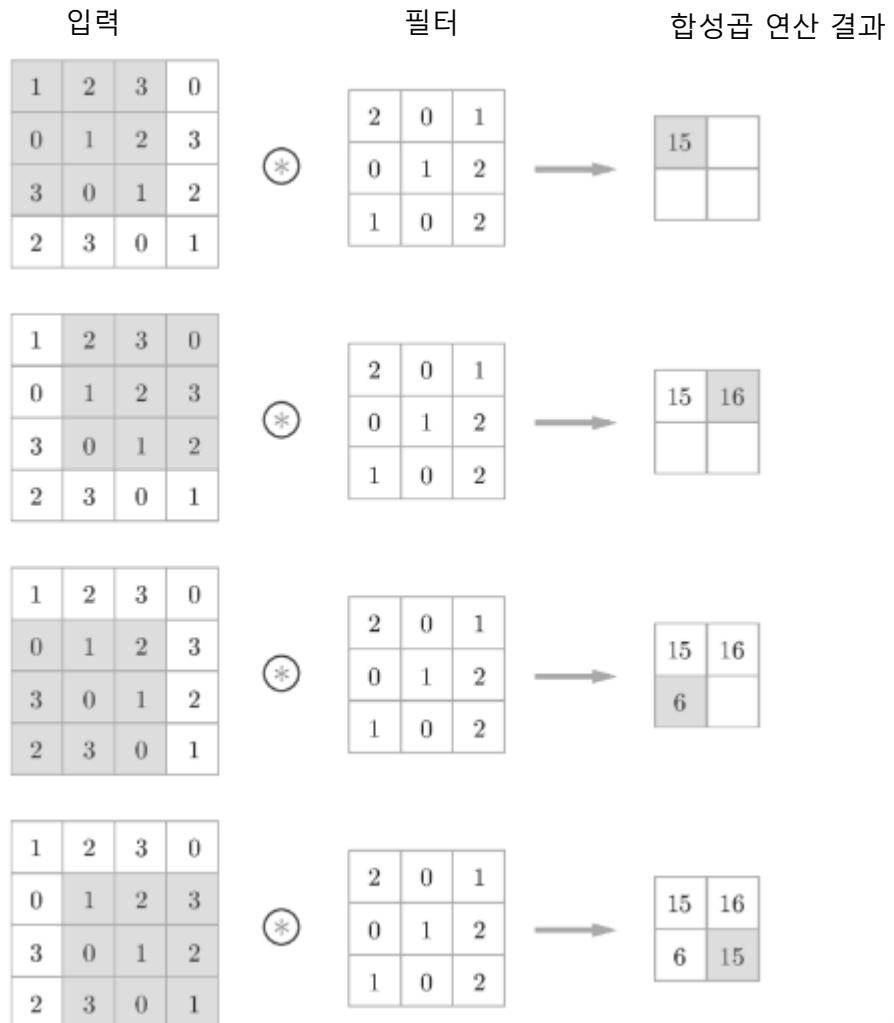
- ❖ Input에서 feature(activation) map을 만드는 과정
- ❖ Convolution(합성곱) 연산으로 처리



Convolution Layer

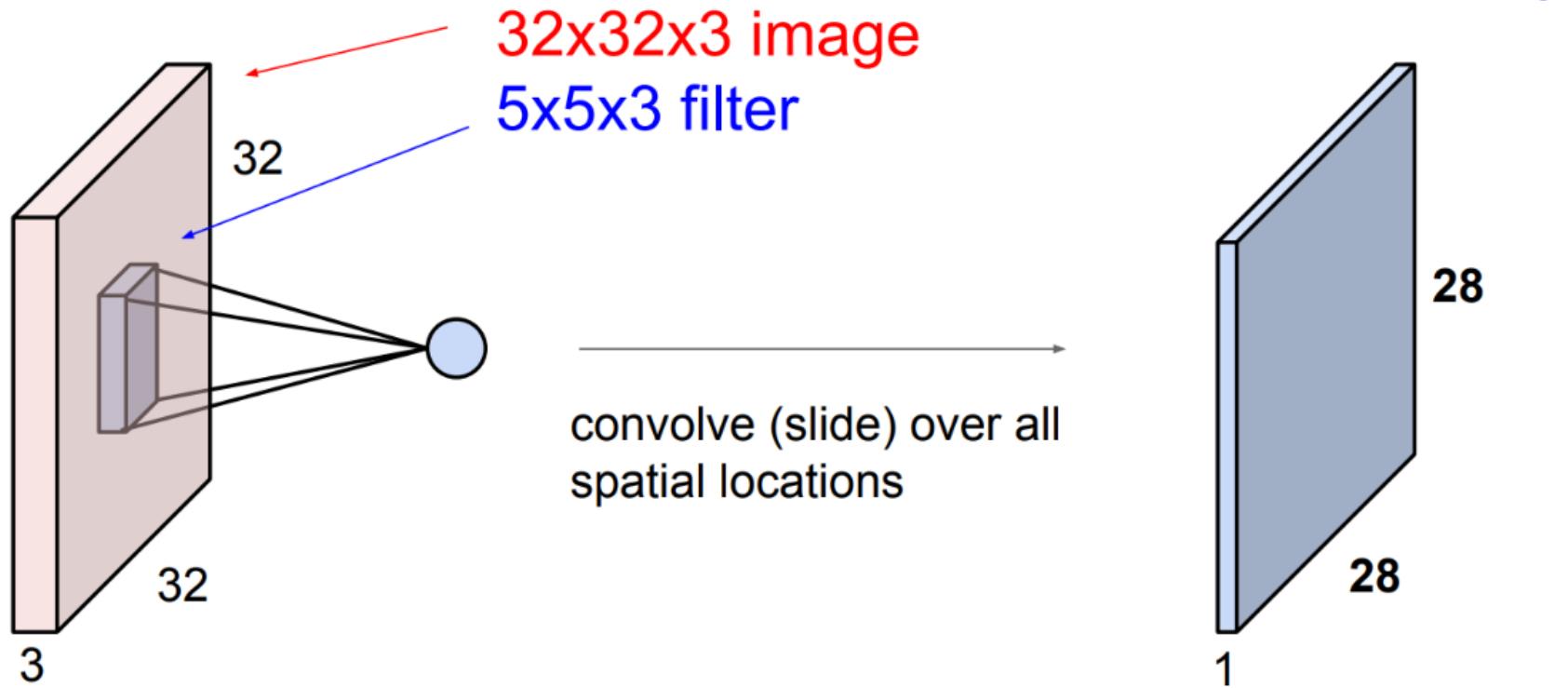
❖ Convolution(합성곱) 연산

- 필터(filter, kernel)의 윈도우를 일정 간격(stride) 이동시키면서 입력에 weighted sum을 적용
- weighted sum : 각 요소의 값을 곱하여 더함

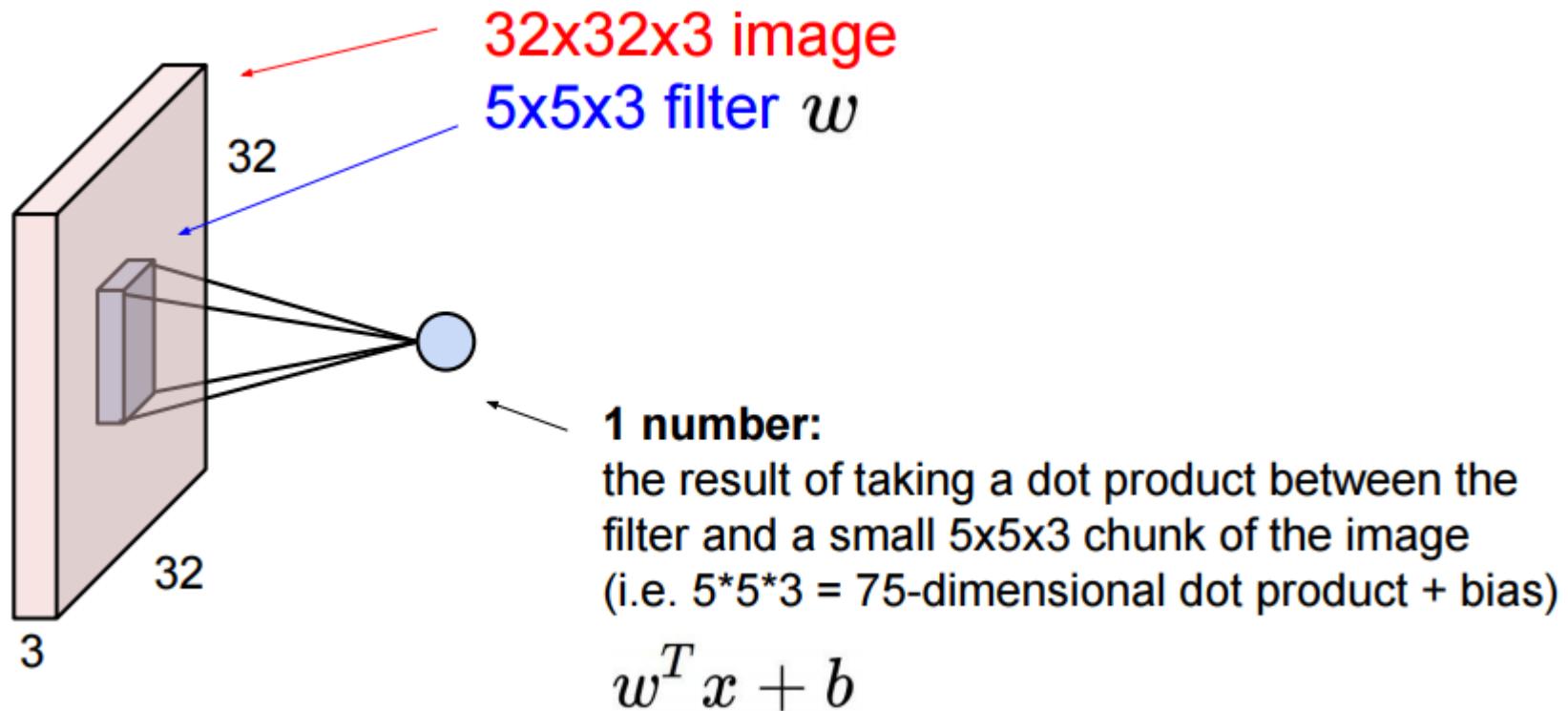


Convolution Layer

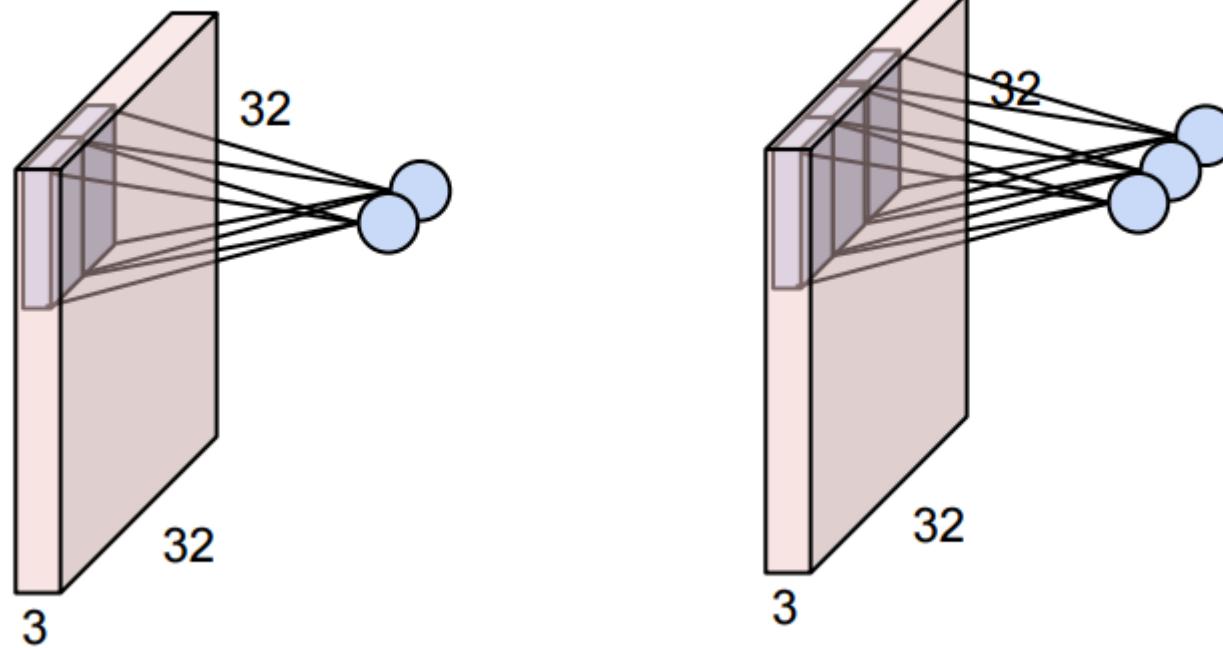
입력 이미지에 필터를 적용하여 작은 영역을 한 점으로 표현



Convolution Layer

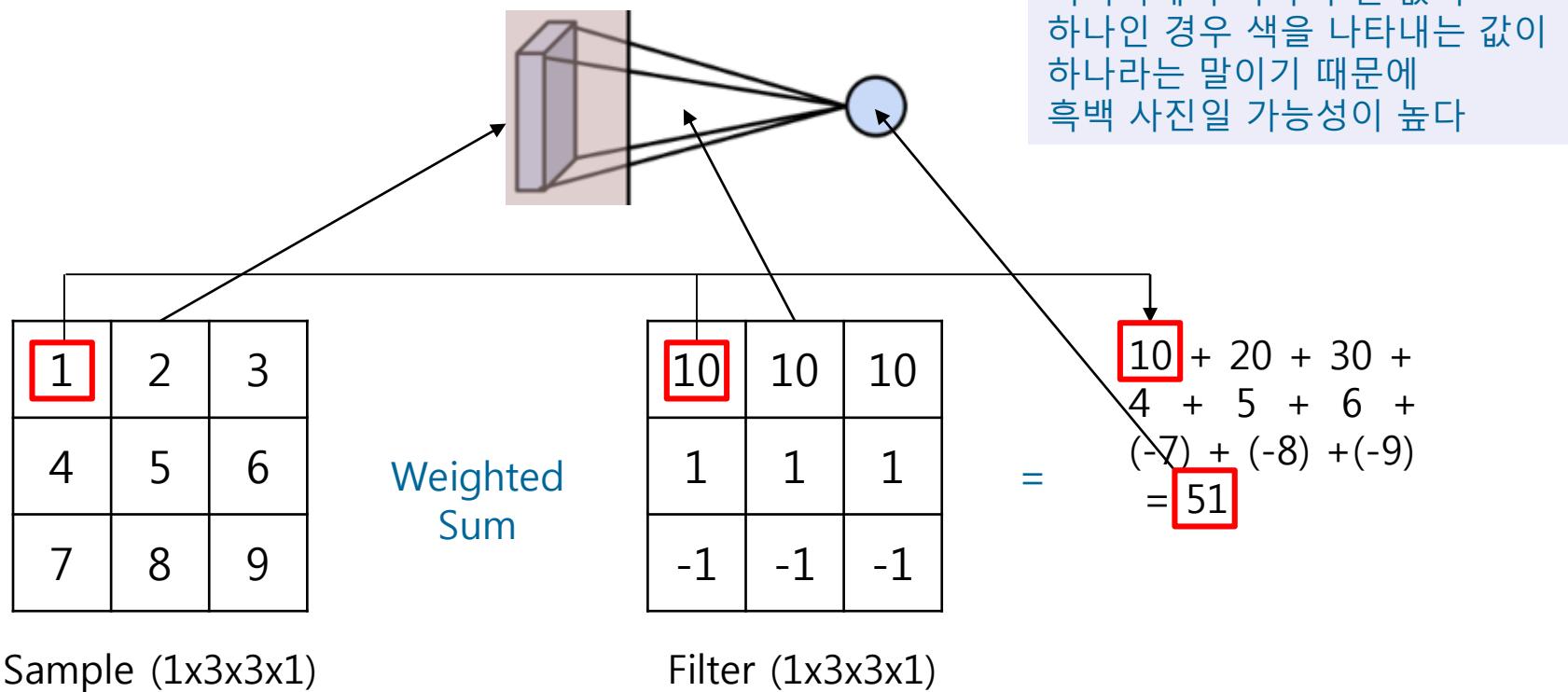


Convolution Layer



Convolution Layer

- ❖ 이미지를 Filter의 크기만큼 조각 내어 weighted sum을 한다



Convolution Layer

❖ 연습 문제

1	5	4	3
8	4	1	4
1	2	3	3
6	5	3	1

Weighted
Sum

Sample(1x4x4x1)

1	1	1	1
-1	-1	-1	-1
2	2	2	2
3	3	3	3

Filter(1x4x4x1)

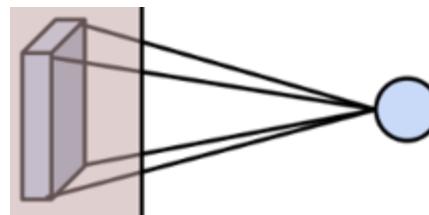
=



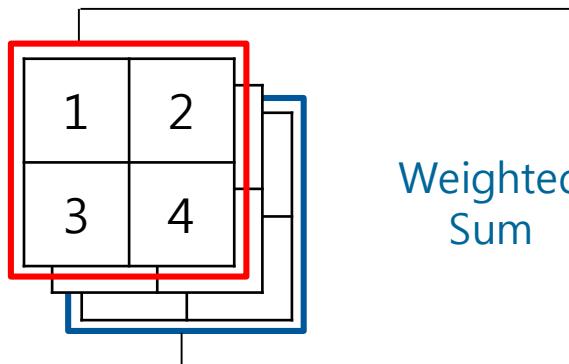
Convolution Layer

❖ Sample의 깊이에 대한 차원과 필터의 깊이에 대한 차원이 동일

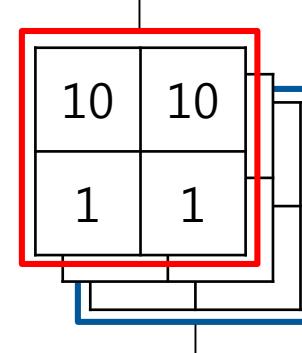
각 깊이마다 합성곱연산을 수행하여



이미지에서 하나의 셀 값이 세 개인 경우 색을 나타내는 값이 세 개이고, 이는 보통 R,G,B 이기 때문에 컬러 사진일 가능성이 높다



Sample(1x2x2x3)



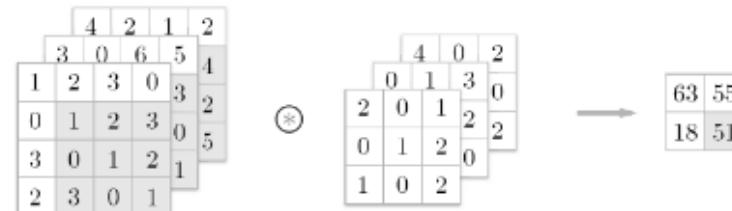
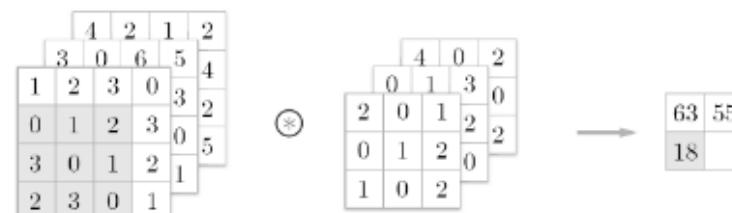
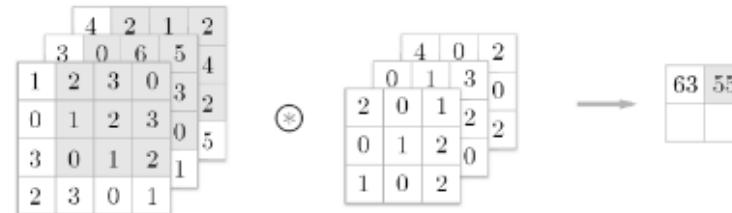
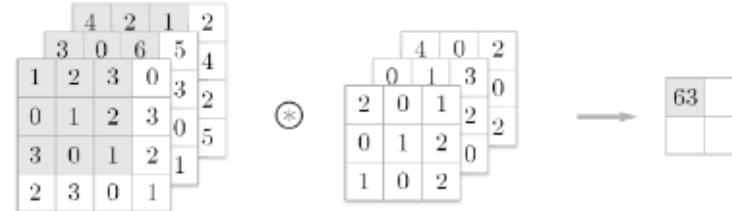
Filter(1x2x2x3)

$$\begin{aligned} & 10 + 20 + 3 + 4 \\ & + (?) + (?) + (?) + (?) \\ & + (?) + (?) + (?) + (?) \\ = & ? \end{aligned}$$

Convolution Layer

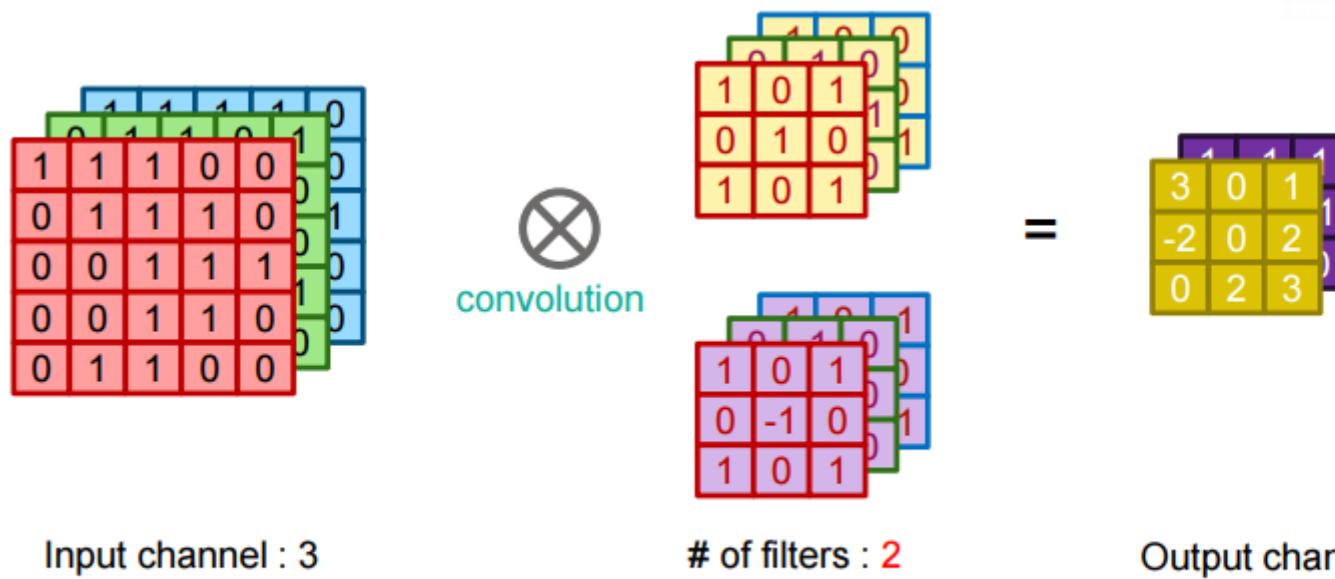
❖ 3차원 데이터의 합성곱

- 입력 데이터와 필터의 합성곱 연산을 깊이마다 수행하고 그 결과를 더해서 출력



Convolution Layer

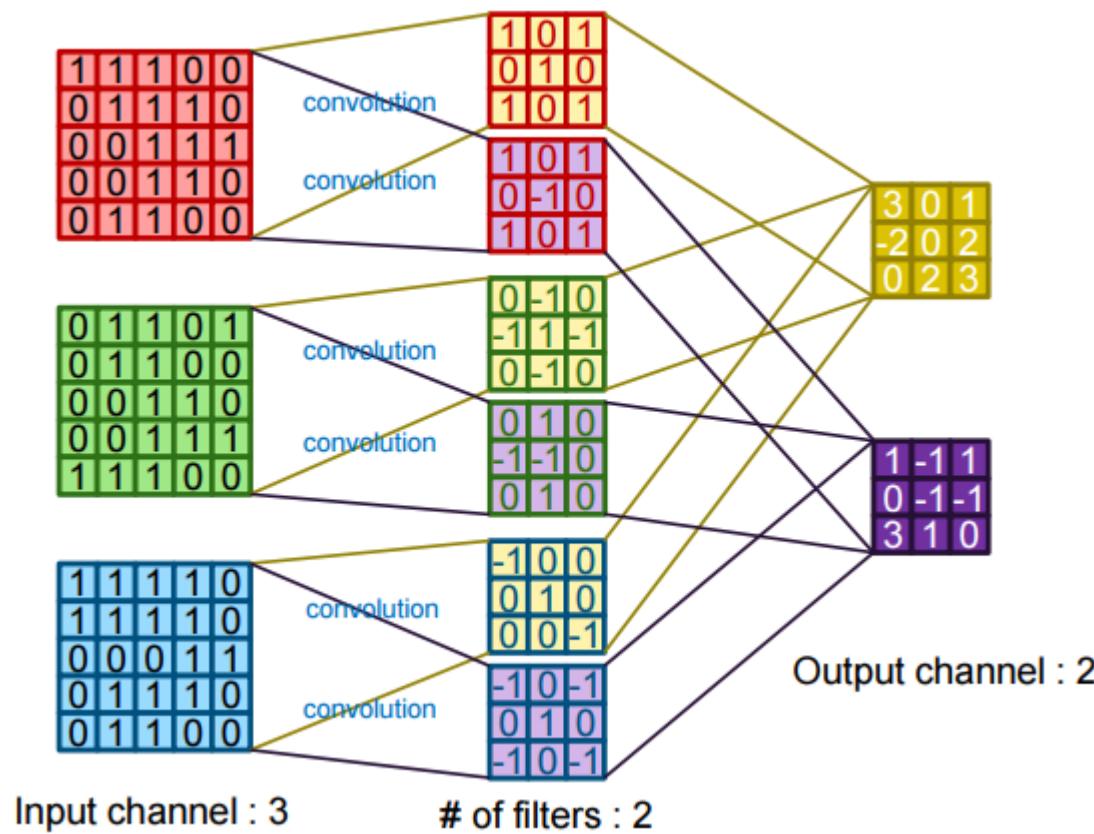
- ❖ Many filters, multi Channel



<https://predictiveprogrammer.com/famous-convolutional-neural-network-architectures-1>

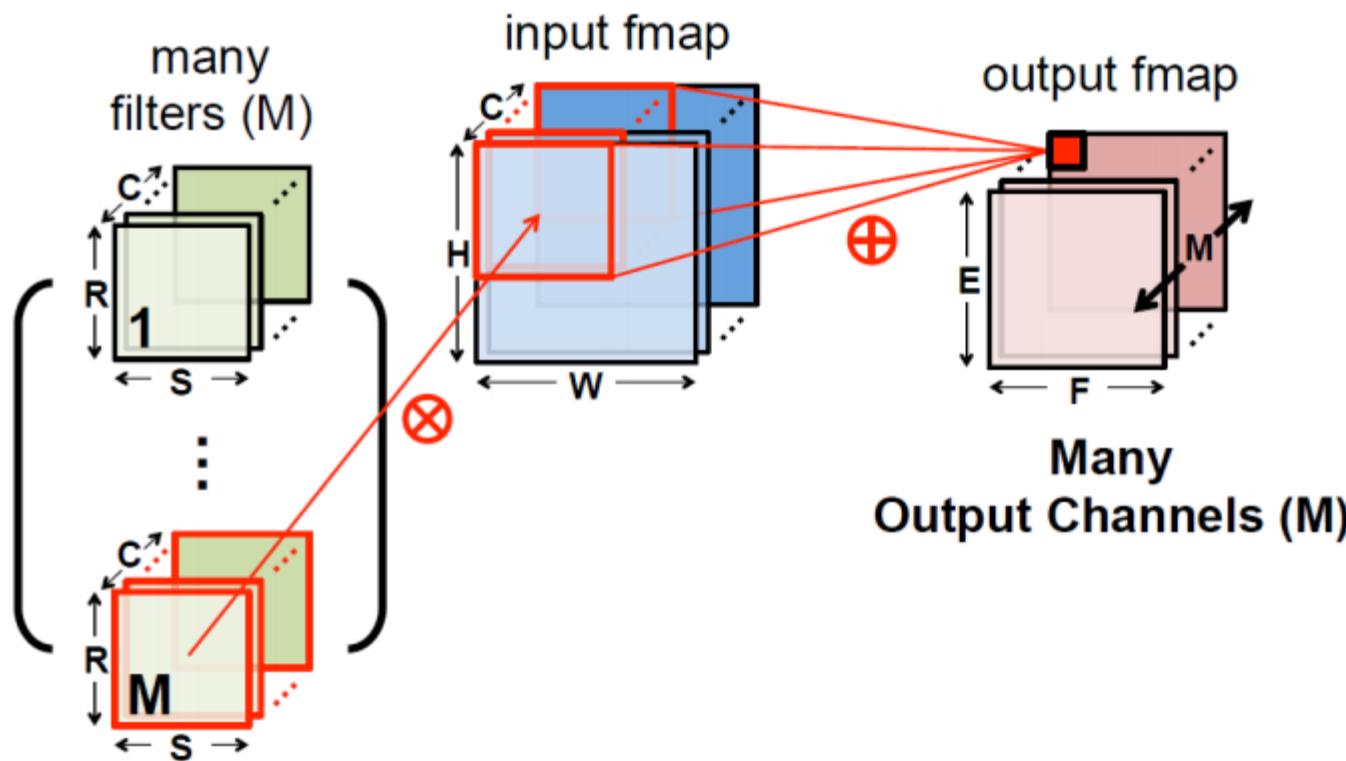
Convolution Layer

- ❖ Many filters, multi Channel



Convolution Layer

❖ Many filters, multi Channel

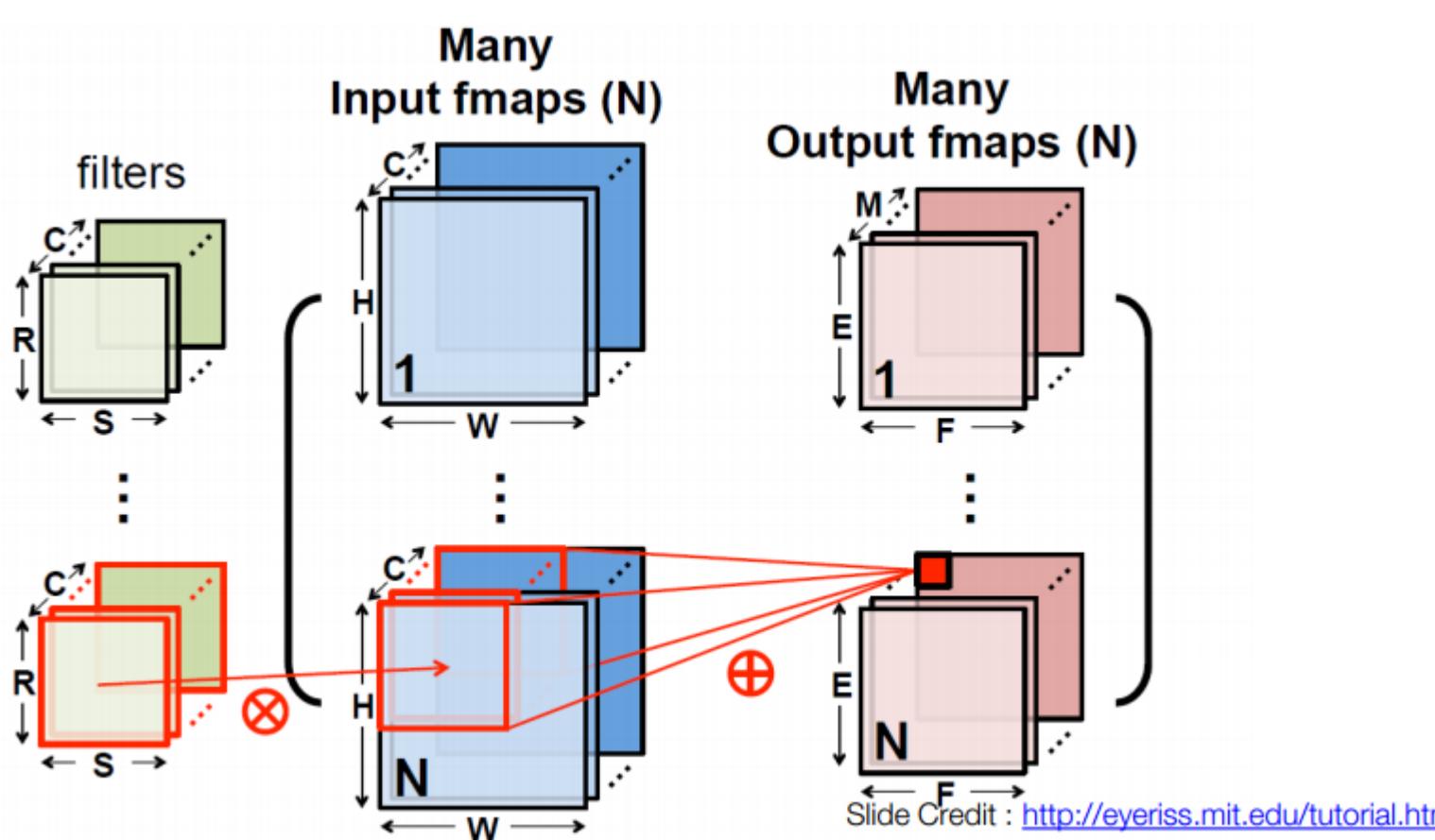


Slide Credit : <http://eyeriss.mit.edu/tutorial.html>

Convolution Layer

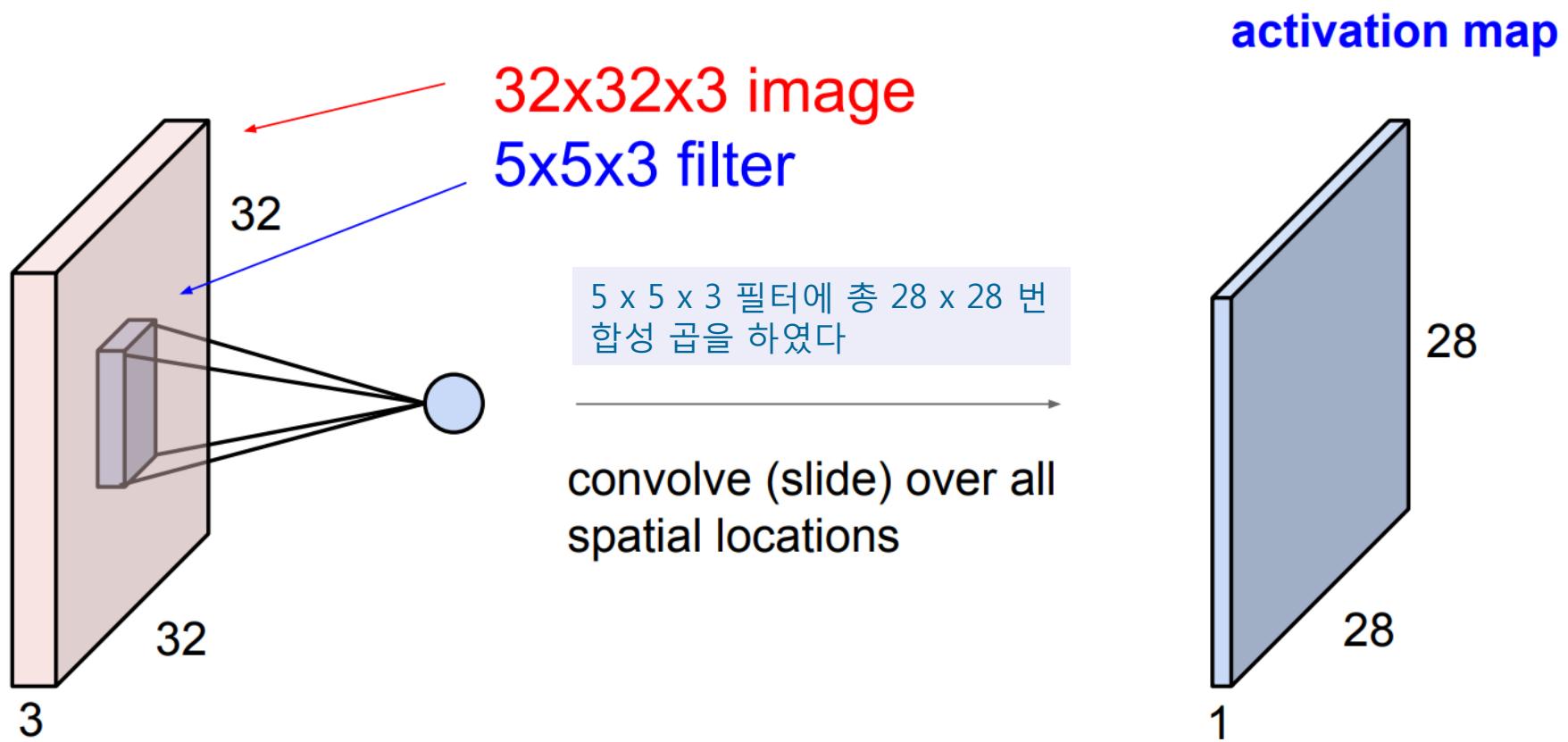
❖ Many input(batch size), Many filters, multi Channel

- Input($\text{inN}, \text{H}, \text{W}, \text{C}$), filter($\text{R}, \text{S}, \text{C}, \text{outN}$), Output($\text{N}, \text{E}, \text{F}, \text{M}$)



Convolution Layer

- ❖ Activation map 이 28×28 이 되려면 어떻게 해야 할까?



Convolution Layer

❖ 쉬운 예

- 7x7 이미지에 3x3 필터를 적용하여 나올 수 있는 Activation map은?

다음 페이지에서 예상해보세요

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

7 x 7 x 1 image

0	0	0
1	1	1
2	2	2

3 x 3 x 1 filter

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

7 x 7 x 1 image

0	0	0
1	1	1
2	2	2

3 x 3 x 1 filter

5 x 5 x 1 activation map

or

3 x 3 x 1 activation map

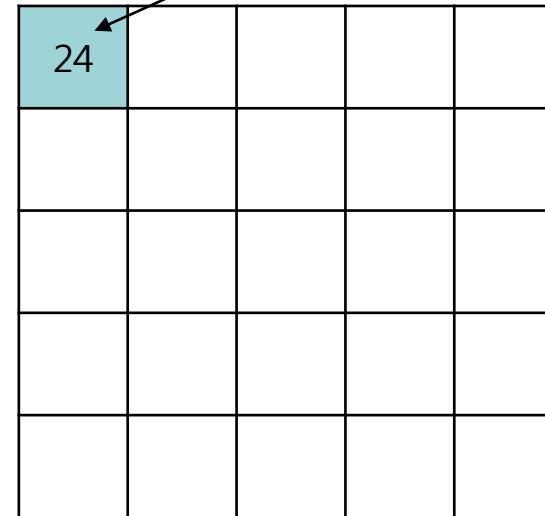
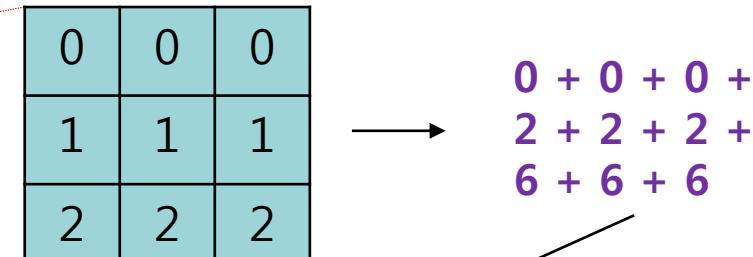
Convolution Layer

❖ Stride

- step당 건너뛰는 크기

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

Stride = 1



Activation map

Convolution Layer

❖ Stride

- step당 건너뛰는 크기

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

Stride = 1

0	0	0
1	1	1
2	2	2

Filter

24	24			

Activation map

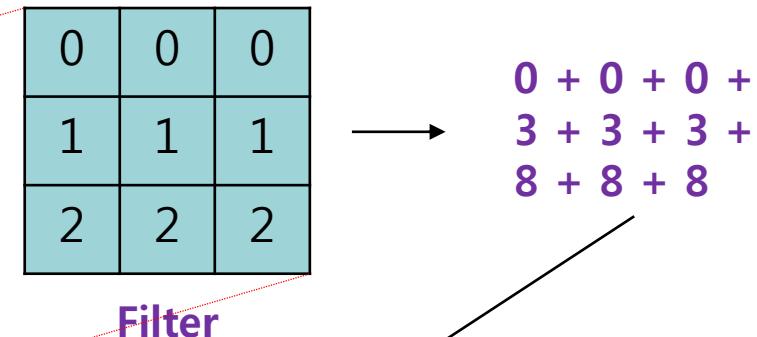
Convolution Layer

❖ Stride

- step당 건너뛰는 크기

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

Stride = 1



24	24	24	24	24
33				

Activation map

Convolution Layer

❖ Stride

- step당 건너뛰는 크기

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

Stride = 2

0	0	0
1	1	1
2	2	2

Filter

24		

$$0 + 0 + 0 + \\ 2 + 2 + 2 + \\ 6 + 6 + 6$$

Activation map

Convolution Layer

❖ Stride

- step당 건너뛰는 크기

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

Stride = 2

0	0	0
1	1	1
2	2	2

Filter

24	24	0

Activation map

$$0 + 0 + 0 + \\ 2 + 2 + 2 + \\ 6 + 6 + 6$$

Convolution Layer

❖ Stride

- step당 건너뛰는 크기

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

Stride = 2

0	0	0
1	1	1
2	2	2

Filter

$$0 + 0 + 0 + \\ 4 + 4 + 4 + \\ 10 + 10 + 10$$

24	24	24
42		

Activation map

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

7 x 7 x 1 image

0	0	0
1	1	1
2	2	2

3 x 3 x 1 filter

24	24	24	24	24
33	33	33	33	33
42	42	42	42	42
51	51	51	51	51
60	60	60	60	60

5 x 5 x 1 activation map

or

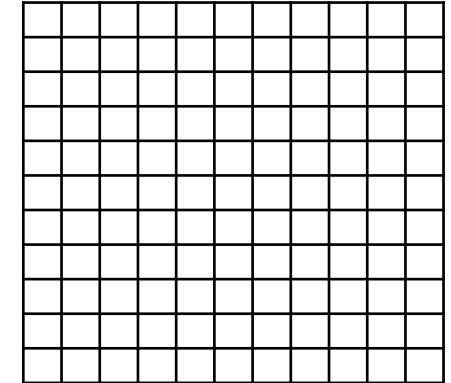
24	24	24
42	42	42
60	60	60

3 x 3 x 1 activation map

Exercise

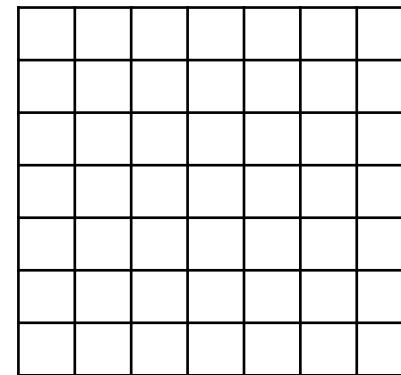
Q. $11 \times 11 \times 1$ Sample, $3 \times 3 \times 1$ Filter

- Stride = 1 일 때 activation map의 크기?
- Stride = 2 일 때 activation map의 크기?



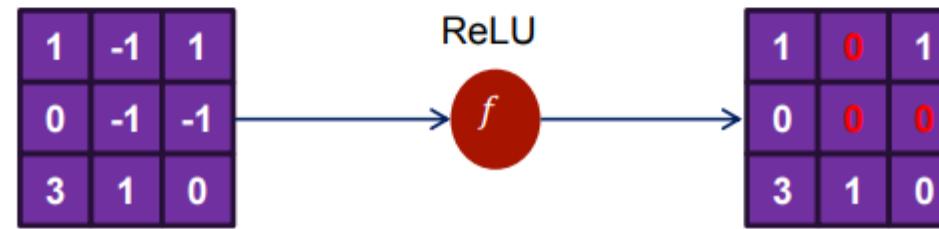
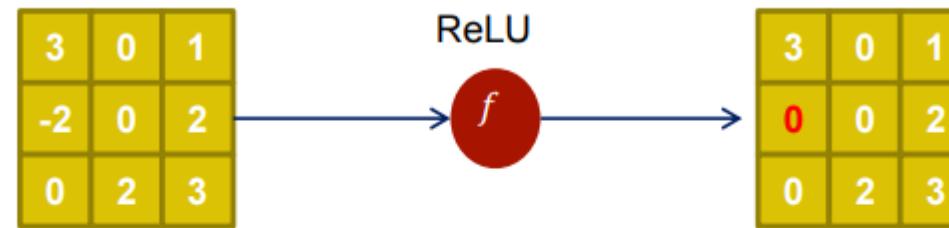
Q. $7 \times 7 \times 3$ Sample에서 $5 \times 5 \times 1$ 의 activation map 생성

- Filter의 크기는?
- Stride의 크기는?

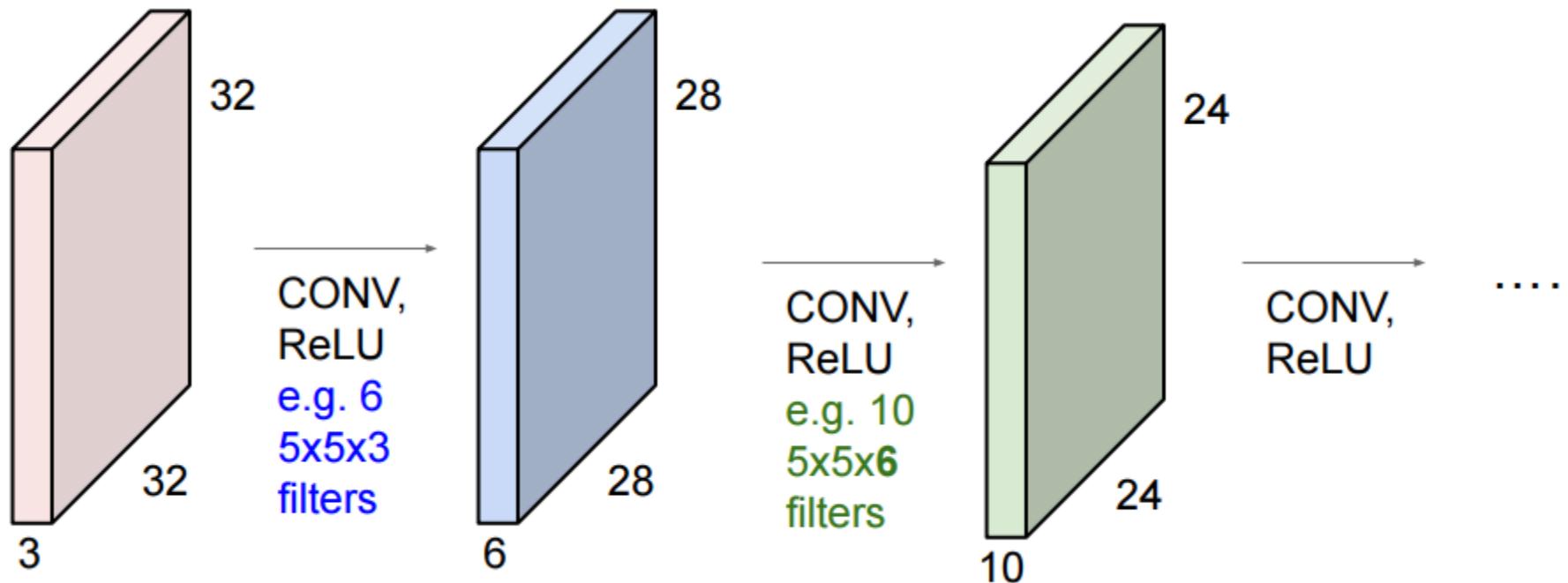


Activation function

- ReLU



Activation function



tf.keras.layers.Conv2D

```
tf.keras.layers.Conv2D(  
    filters, kernel_size, strides=(1, 1), padding='valid',  
    data_format=None, dilation_rate=(1, 1), activation=None,  
    use_bias=True, kernel_initializer='glorot_uniform',  
    bias_initializer='zeros', kernel_regularizer=None,  
    bias_regularizer=None, activity_regularizer=None,  
    kernel_constraint=None, bias_constraint=None, **kwargs  
)
```

filters : 필터개수

kernel_size : 커널 크기

strides=(1, 1) : 폭, 높이로 컨볼루션 보폭 지정

padding='valid' (패딩 안함), **'same'** (stride 가 1일 때, 입력과 출력 크기가 같아지도록)

data_format : 입력 데이터 구조의 순서 (batch_size, height, width, channels)

activation: 활성화함수 지정

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

tf.keras.layers.Conv2D

(출력 channel 개수 == 필터 개수)

kernel dimension : {height, width, in_channel, out_channel}
Ex) {5, 5, 3, 2}



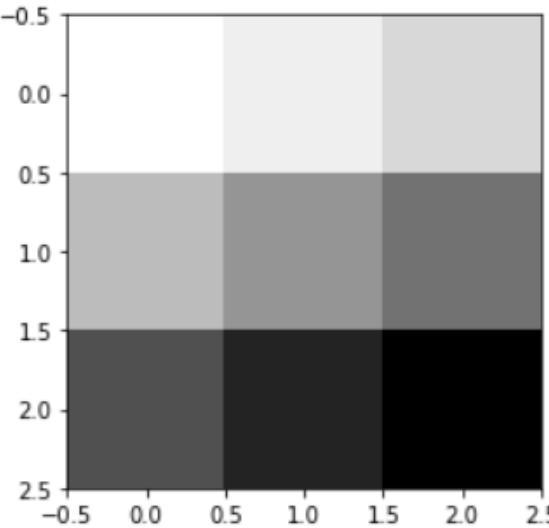
Input

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt

print(tf.__version__)
print(keras.__version__)
```

```
image = tf.constant([[[[1],[2],[3]],
                     [[4],[5],[6]],
                     [[7],[8],[9]]]], dtype=np.float32)
print(image.shape)
plt.imshow(image.numpy().reshape(3,3), cmap='Greys')
plt.show()
```

(1, 3, 3, 1)

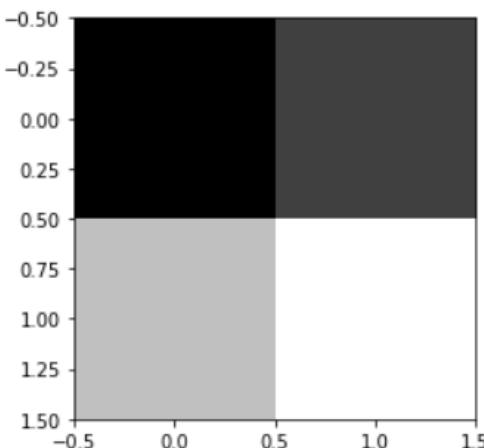


https://github.com/deeplearningzerotoall/TensorFlow/blob/master/tf_2.x/lab-11-0-cnn-basics-keras-eager.ipynb

Weight(filter)

```
print("image.shape", image.shape)
weight = np.array([[[[1.],[1.]],
                   [[1.],[1.]]]])
print("weight.shape", weight.shape)
weight_init = tf.constant_initializer(weight)
conv2d = keras.layers.Conv2D(filters=1, kernel_size=2, padding='VALID',
                             kernel_initializer=weight_init)(image)
print("conv2d.shape", conv2d.shape)
print(conv2d.numpy().reshape(2,2))
plt.imshow(conv2d.numpy().reshape(2,2), cmap='gray')
plt.show()
```

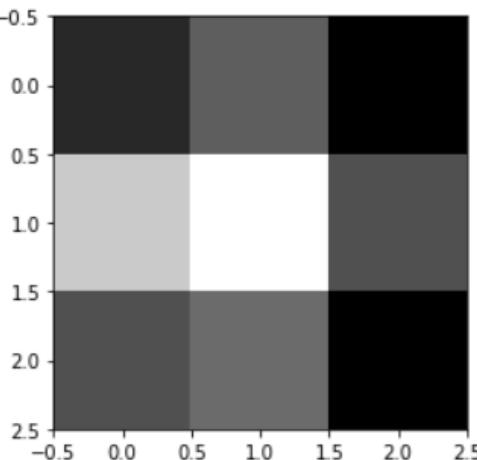
```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d.shape (1, 2, 2, 1)
[[12. 16.]
 [24. 28.]]
```



conv2d

```
print("image.shape", image.shape)
weight = np.array([[[[1.],[1.]],
                   [[1.],[1.]]]])
print("weight.shape", weight.shape)
weight_init = tf.constant_initializer(weight)
conv2d = keras.layers.Conv2D(filters=1, kernel_size=2, padding='SAME',
                            kernel_initializer=weight_init)(image)
print("conv2d.shape", conv2d.shape)
print(conv2d.numpy().reshape(3,3))
plt.imshow(conv2d.numpy().reshape(3,3), cmap='gray')
plt.show()
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d.shape (1, 3, 3, 1)
[[12. 16.  9.]
 [24. 28. 15.]
 [15. 17.  9.]]
```

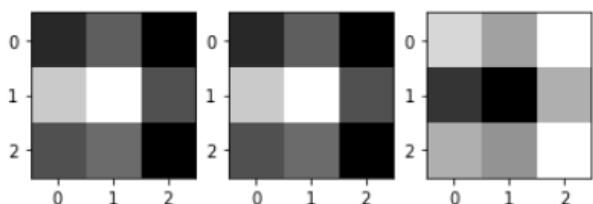


3 filters (2,2,1,3)

```
# print("image:\n", image)
print("image.shape", image.shape)

weight = np.array([[[[1.,10.,-1.]], [[1.,10.,-1.]]],  
                  [[[1.,10.,-1.]], [[1.,10.,-1.]]]])
print("weight.shape", weight.shape)
weight_init = tf.constant_initializer(weight)
conv2d = keras.layers.Conv2D(filters=3, kernel_size=2, padding='SAME',  
                           kernel_initializer=weight_init)(image)
print("conv2d.shape", conv2d.shape)
feature_maps = np.swapaxes(conv2d, 0, 3)
for i, feature_map in enumerate(feature_maps):
    print(feature_map.reshape(3,3))
    plt.subplot(1,3,i+1), plt.imshow(feature_map.reshape(3,3), cmap='gray')
plt.show()
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 3)
conv2d.shape (1, 3, 3, 3)
[[12. 16. 9.]
 [24. 28. 15.]
 [15. 17. 9.]]
[[120. 160. 90.]
 [240. 280. 150.]
 [150. 170. 90.]]
[[-12. -16. -9.]
 [-24. -28. -15.]
 [-15. -17. -9.]]
```



Padding

❖ Padding

- 이미지의 외각에 지정된 값만큼 특정 값으로 채워 넣는 작업

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

원본 $5 \times 5 \times 1$

Padding

❖ Padding

- 만일 외각 두께 1만큼 0으로 모두 채운다면 아래와 같은 형태가 된다

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Padding 후 $7 \times 7 \times 1$

모두 0으로 채우는 것을 zero padding 이라고도 한다

Padding

❖ Padding

- filter = $3 \times 3 \times 1$, stride = 1 을 진행하면 padding 을 하기 전과 크기가 같다

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Padding 후 $7 \times 7 \times 1$



3 x 3 x 1 filter

Stride = 1

4	6	6	6	4
6	9	9	9	6
6	9	9	9	6
6	9	9	9	6
4	6	6	6	4

Activation map
 $5 \times 5 \times 1$

Padding

❖ Padding

- Weighted sum 을 데이터의 drop 없이 모두 수행
- Sample 의 크기가 너무 빨리 줄어드는 것을 방지
- 경계면의 정보를 유지
- 출력크기를 조정할 때 사용

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Padding

- ❖ 입력 크기 (H,W), 필터 크기 (FH, FW), 출력 크기를 (OG, OW), 패딩 P, 스트라이드 S일 때 출력의 크기는 ?

$$OH = (H + 2P - FH) / S + 1$$

$$OW = (W + 2P - FW) / S + 1$$

예>(1) 입력 (4,4) 필터(3,3) 패딩 1, 스트라이드 1

$$OH = (4 + 2*1 - 3) / 1 + 1 = 4$$

$$OW = (4 + 2*1 - 3) / 1 + 1 = 4$$

(2) 입력 (28,32) 필터(5,5) 패딩 2, 스트라이드 2 ??

Padding

❖ Padding 고급

- padding = 'VALID'

원본 $11 \times 11 \times 1$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

하얀색은 쓰이지 않고 버려지는 부분

1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1

$6 \times 6 \times 1$ filter

Activation map
 $(2, 2, 1)$

Stride = 4

Padding

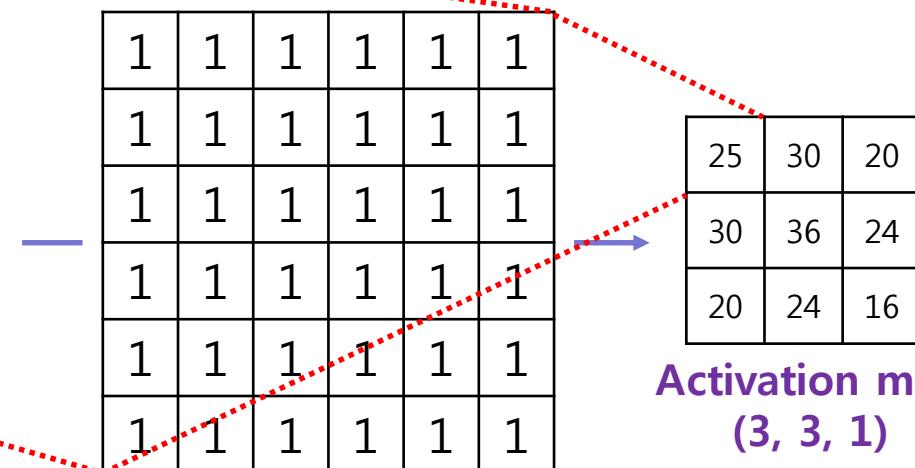
❖ Padding 고급

- padding = 'SAME' -> stride = 4, filter = 6×6 을 완수하려면 3칸이 더 필요

A 11x11 input matrix with padding. The input values are 0 or 1. The input is processed by a 6×6 filter with stride 4. The output is a 3x3 activation map. The input has 3 extra columns of padding (0s) on both sides.

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	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

원본 $11 \times 11 \times 1 \rightarrow 14 \times 14 \times 1$



Stride = 4

Activation map
(3, 3, 1)