Convolutional Neural Network

Hyunjoong Kim

soy.lovit@gmail.com

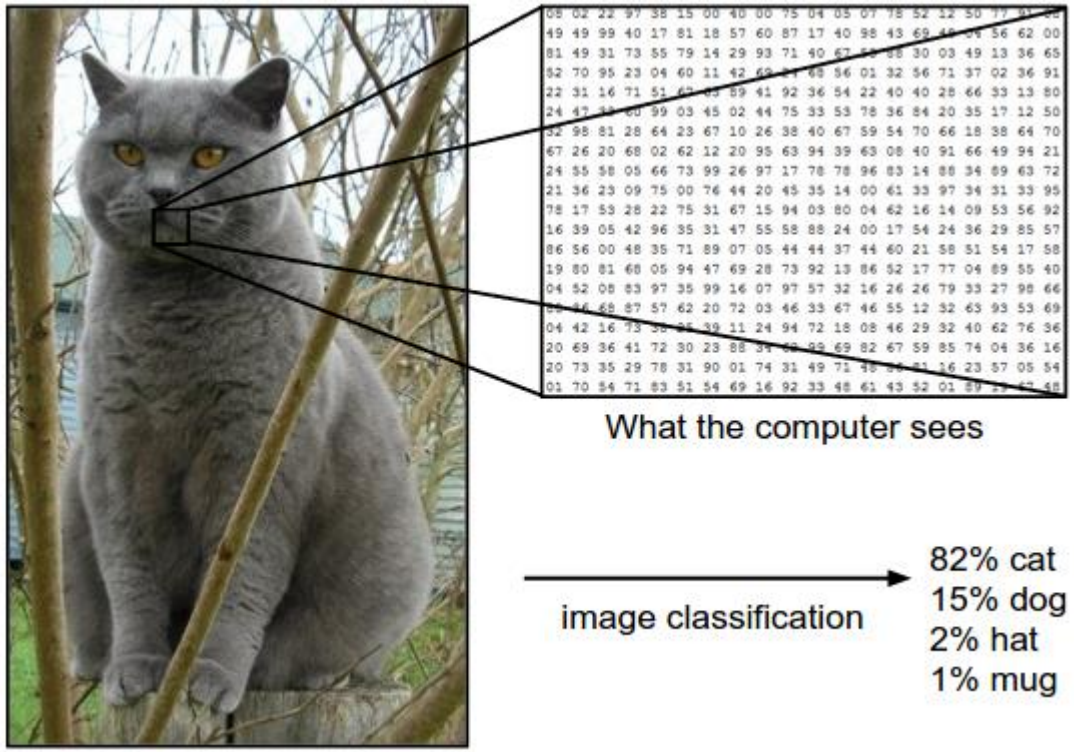
github.com/lovit

Why Convolutional Neural Network ?

- Convolutional Neural Network (CNN) 은 처음 이미지 데이터의 분류를 위하여 제안되었습니다.
  - 이미지 분류 문제의 성능 향상을 위하여 모델이 발전되었으며,
  - 이후 음성과 자연어처리 영역에서 이용되었습니다.
- CNN 의 원리를 이해하기 위해서 이미지 인식을 간단히 살펴봅니다.

Image recognition

- 이미지는 벡터로 될 수 있습니다.



http://cs231n.github.io/classification/

Image recognition

- 이미지는 벡터로 될 수 있습니다.
  - 주로 RBG color 를 이용합니다. 한 점은 3 차원 벡터입니다.

```
from matplotlib.pyplot import imshow
from matplotlib.pyplot import figure
from skimage.io import imread

image = imread('lalaland.jpg')
print(type(image))
print(image.shape)
```

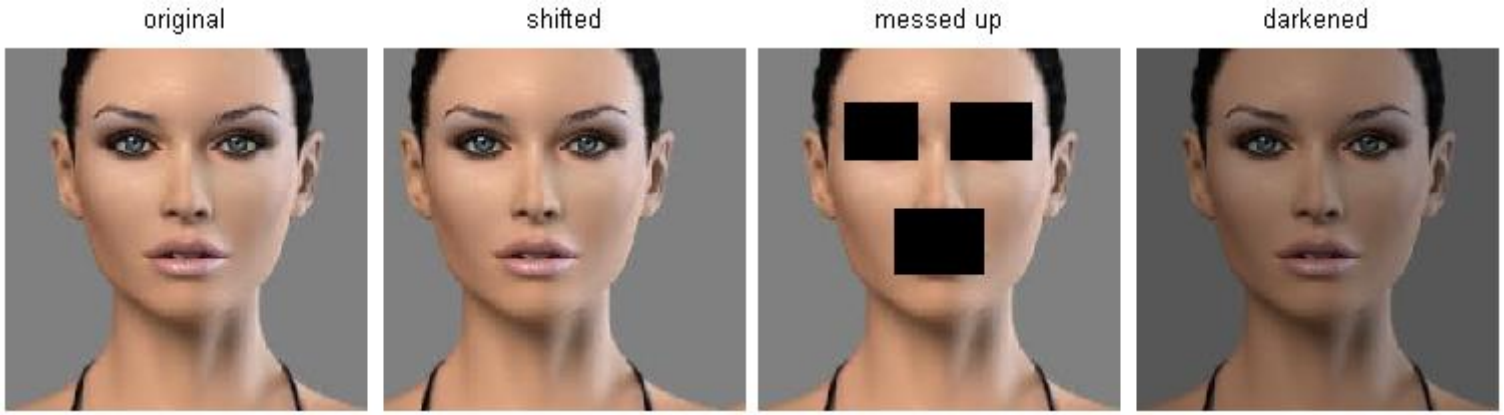
```
<class 'numpy.ndarray'>
(1377, 2000, 3)
```

Image recognition

- 이미지는 벡터로 될 수 있습니다.
  - 주로 RBG color 를 이용합니다. 한 점은 3 차원 벡터입니다.
  - 3 channels 라고 표현합니다.

Image recognition

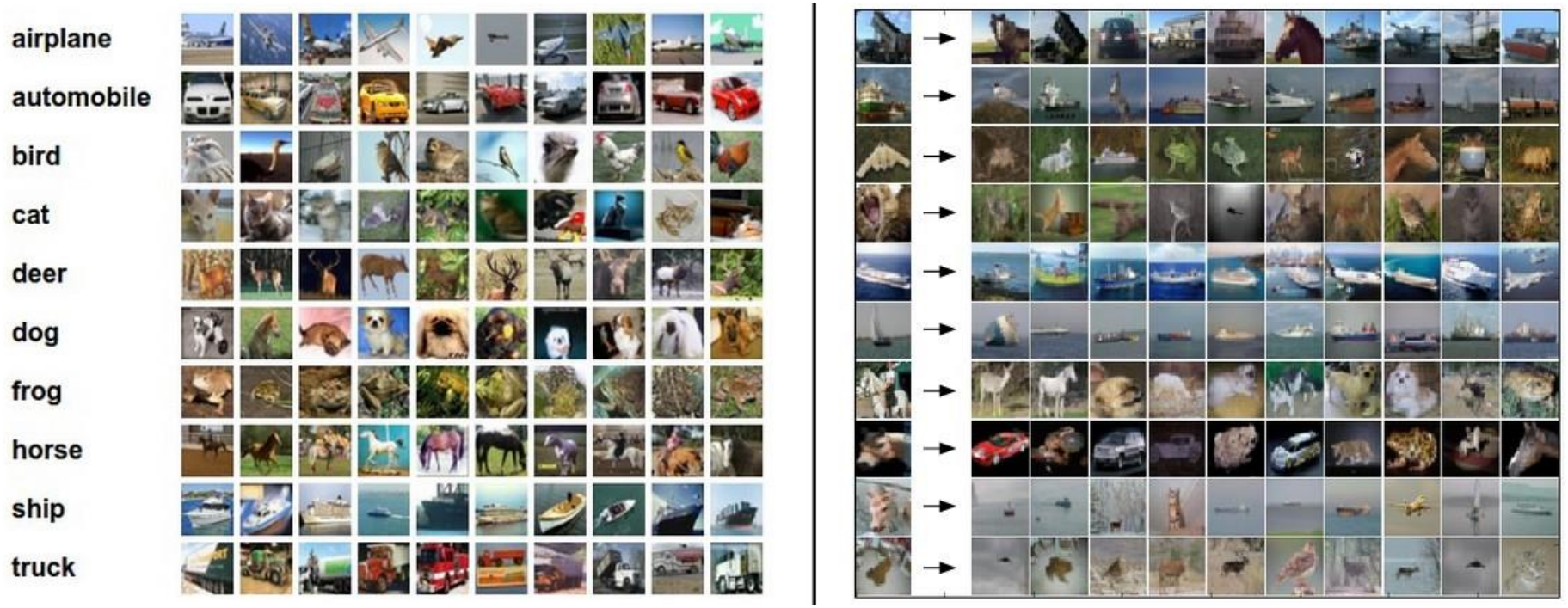
- “이미지 벡터의 벡터 거리”는 이미지의 다름을 잘 설명하지 못합니다.
  - 아래 이미지는 모두 original 과 같은 L2 distance 를 지니는 이미지입니다.



http://cs231n.github.io/classification/

Image recognition

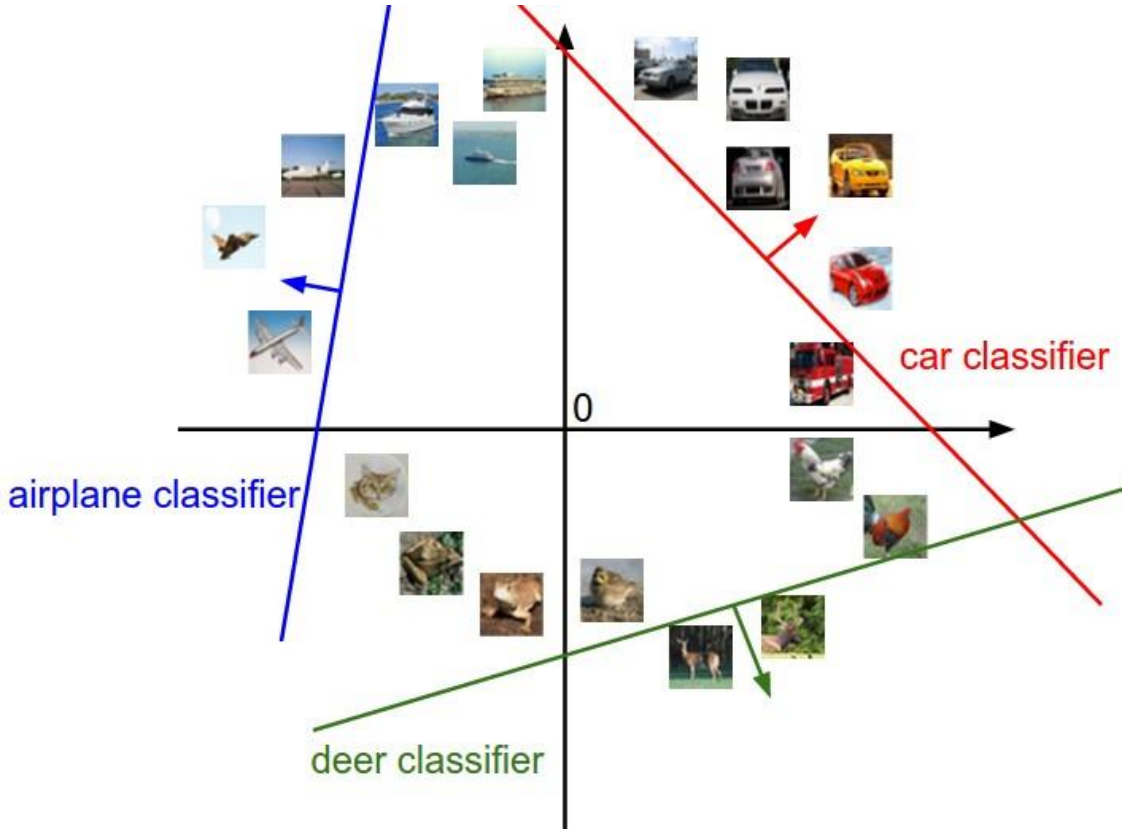
- “이미지 벡터의 벡터 거리”는 이미지의 다름을 잘 설명하지 못합니다.
  - 좌측은 CIFAR 10 데이터에서의 같은 class 이미지이며,
  - 우측은 거리가 가까운 이미지 입니다.



http://cs231n.github.io/classification/

Image recognition

- Linear classifier 이미지 벡터들이 클래스별로 한 방향에 모여 있을 때 잘 작동합니다.



http://cs231n.github.io/linear-classify/

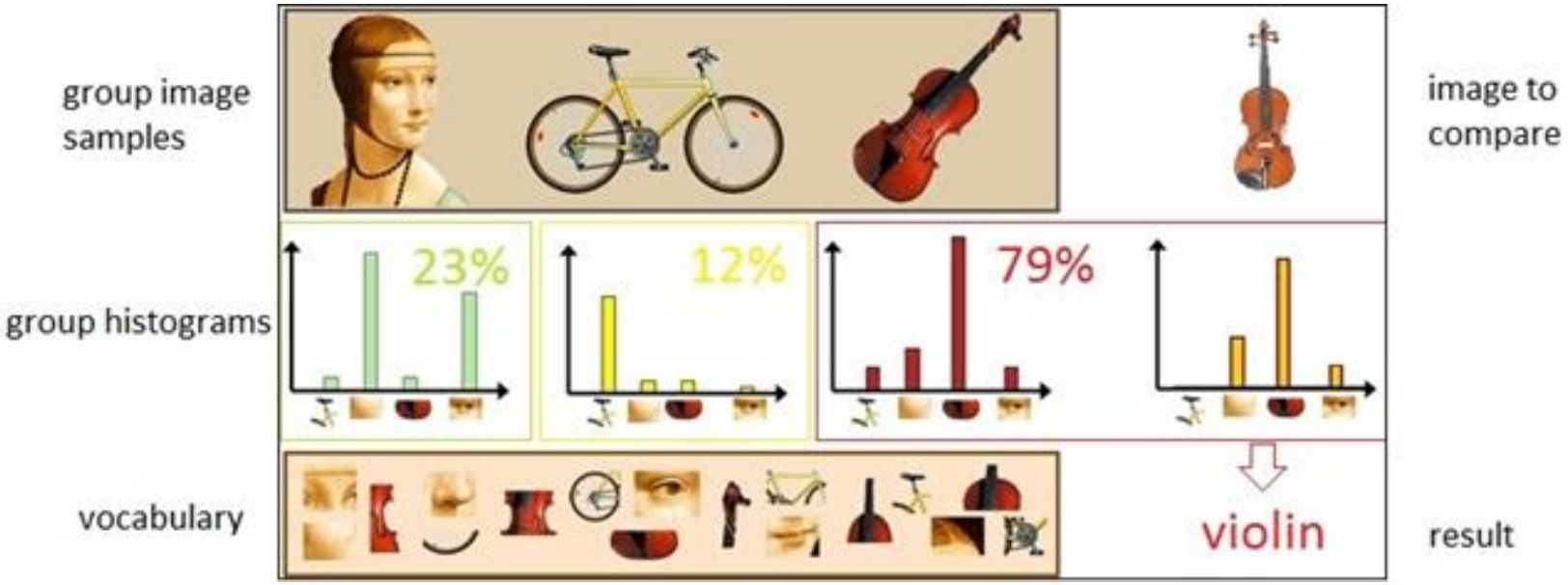
Image recognition

- Linear classifier 의 대표 벡터를 template 으로 해석할 수 있습니다.
  - 각 클래스의 대표벡터는 아래 그림처럼 여러 그림을 겹쳐놓은 듯한 blur 한 이미지입니다.

http://cs231n.github.io/linear-classify/

Image recognition

- Bag of (visual) words model
  - 이미지의 부분 요소를 vocabulary 로 정의,
  - 각 vocabulary 의 matching histogram vector 로 이미지를 표현합니다.



http://vgg.fiit.stuba.sk/2015-02/bag-of-visual-words-in-opencv/

Image recognition

- 각 vocabulary 의 matching 은 이미지를 잘라내어 sub-image 와 vocabulary 간의 유사도를 측정합니다.

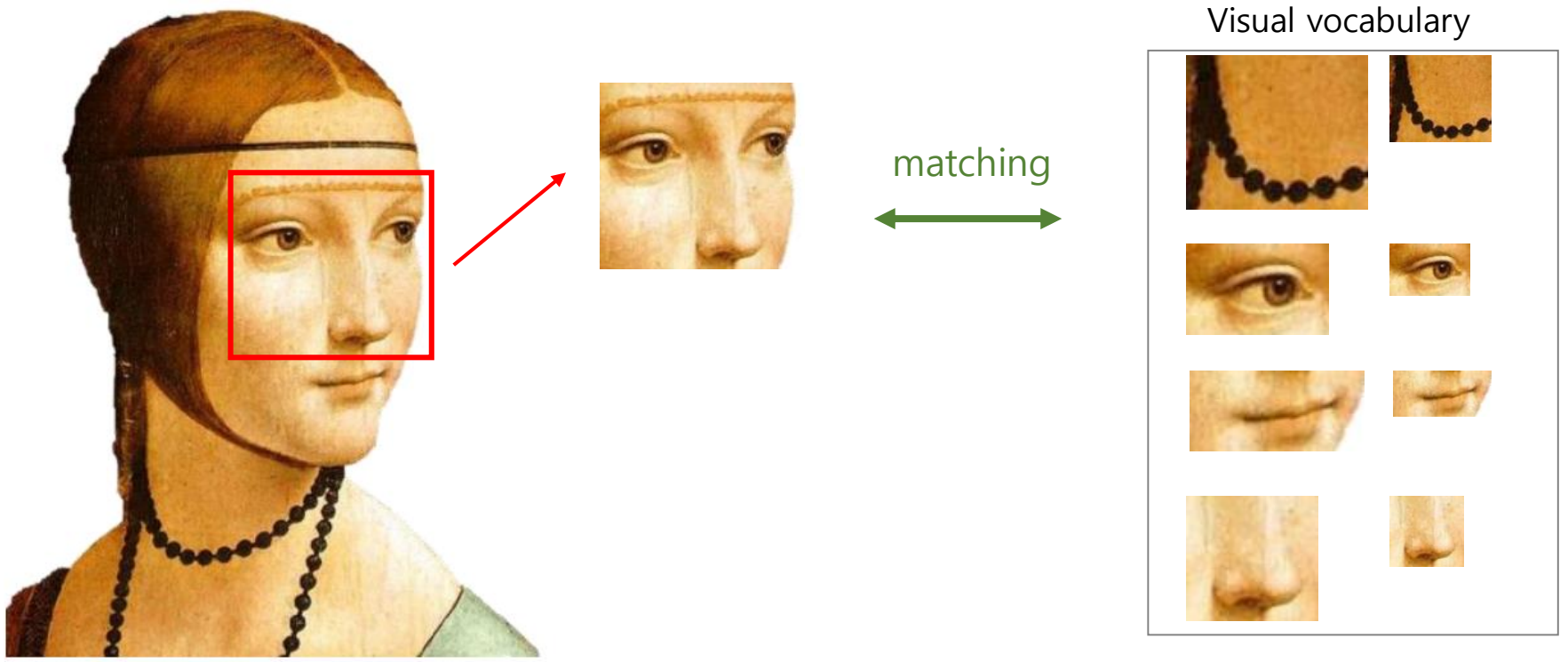


Image recognition

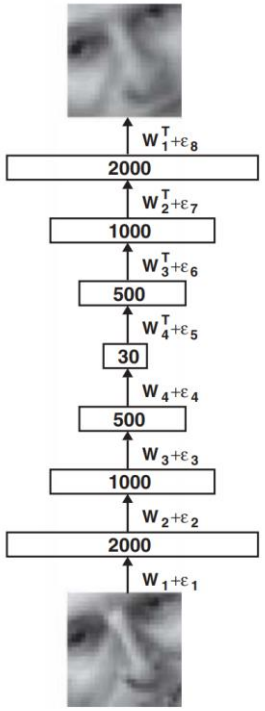
- Bag of (visual) words model
  - Vocabulary 가 잘 만들어져 있어야 합니다.
  - Scale, rotation invariant 하지 않습니다.

Image recognition

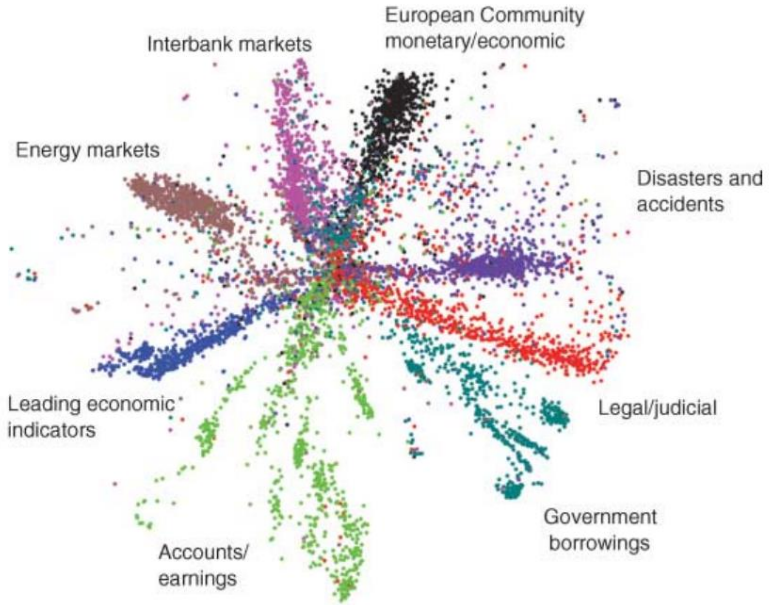
- 이미지 인식을 위해서 이미지 벡터를 그대로 인식하지 않습니다.
  - 이미지는 작은 요소들의 구성으로 이뤄져 있습니다.
  - 한 장의 이미지 전체를 한 번에 처리하면 중요하지 않는 요소들에 영향력이 클 수 있습니다.
- 이미지도 저차원의 semantic space 의 벡터로 표현할 수 있습니다.

Representation in semantic space

- Hinton and Salakhutdinov (2006) 은 neural network 를 이용하여 저차원의 semantic space 로 의미를 압축할 수 있음을 보여줬습니다.



RBM based encoder – decoder



2-dim representation of news documents

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

Convolutional Neural Network

- CNN 은 이미지를 의미를 보존하는 저차원의 벡터로 표현합니다.
  - Convolutions 는 bag of visual words model 처럼 이미지에서 각 visual words 가 존재하는지 살펴봅니다.
  - Subsampling (pooling) 은 의미적 정보를 지닌 저차원의 벡터로 변환합니다.

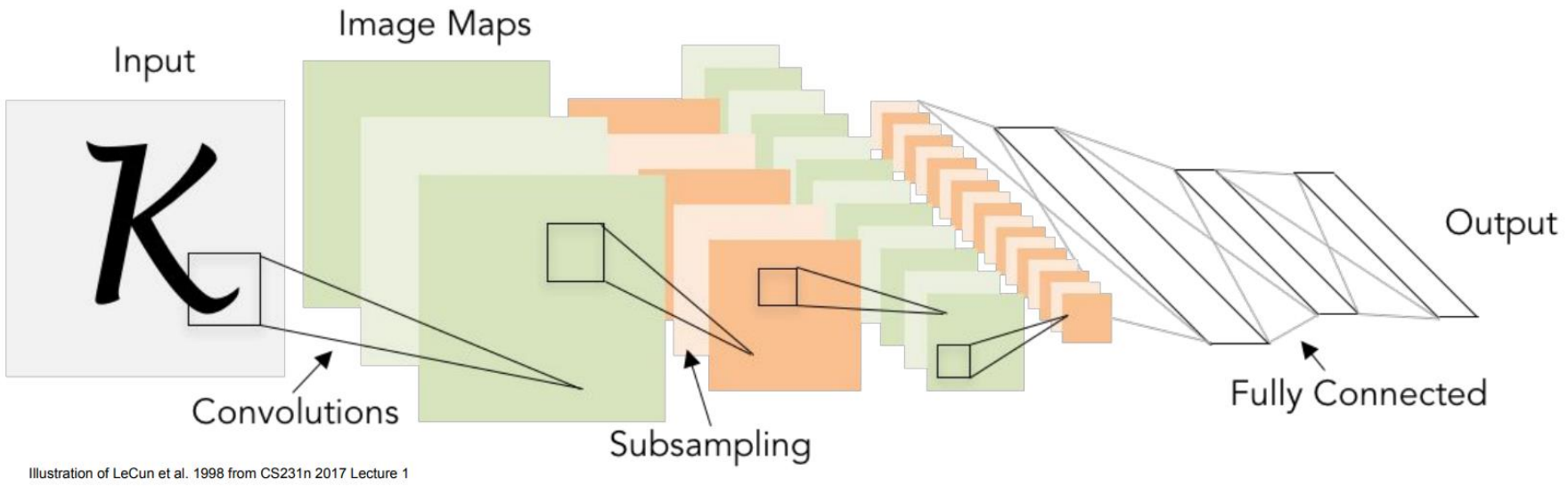
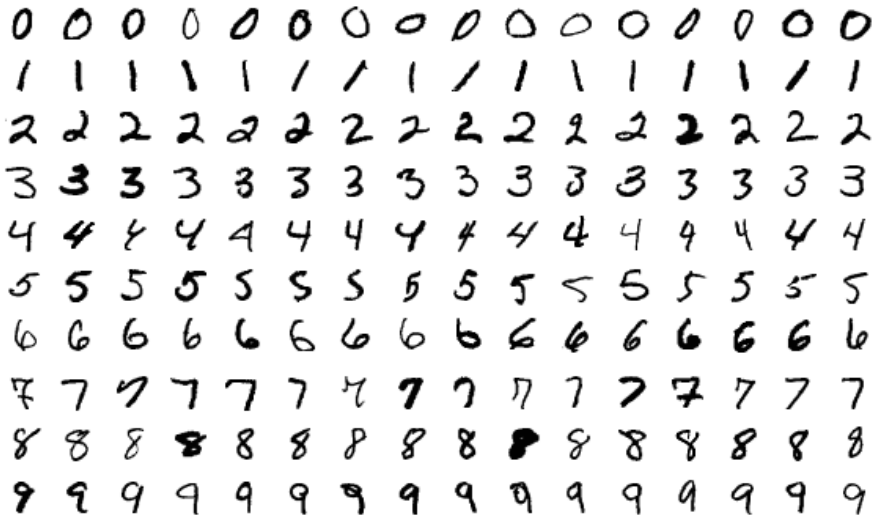


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

CS231 lecture note.

LeNet

- CNN 의 첫번째 성공적인 모델링 사례는 Yann LeCun 의 LeNet 입니다.
  - 손글씨와 프린팅된 숫자를 자동으로 인식하기 위한 neural network 입니다.
  - 0 ~ 9 까지의 10 개의 숫자를 분류합니다.



MNIST dataset

LeNet

- LeNet 은 두 번의 (convolution, pooling) 을 거친 뒤, 2 개의 fully connected layer 를 이용합니다.

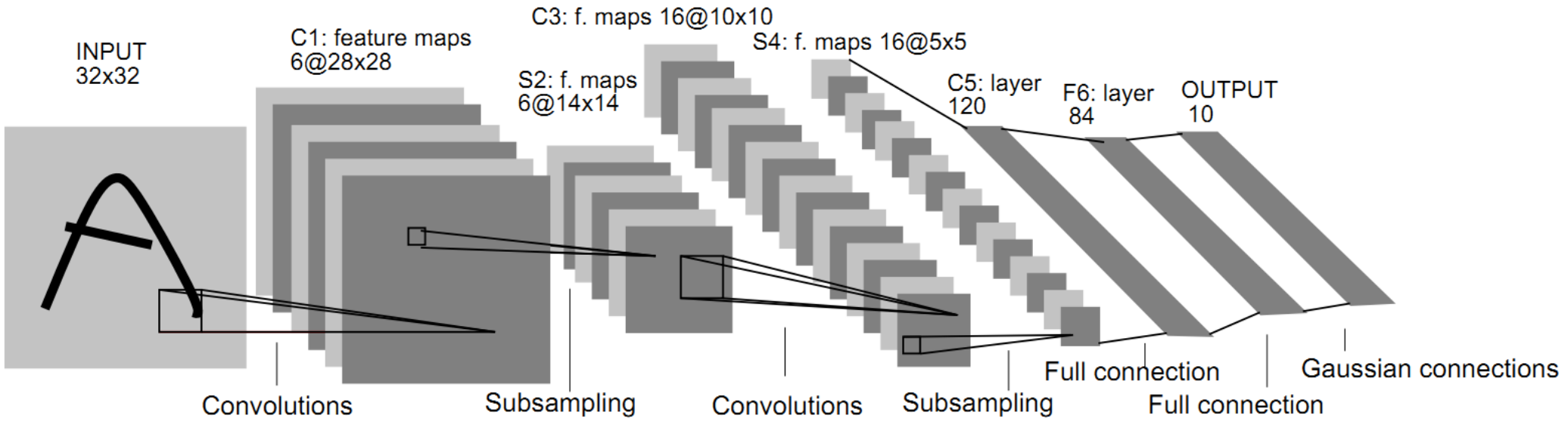
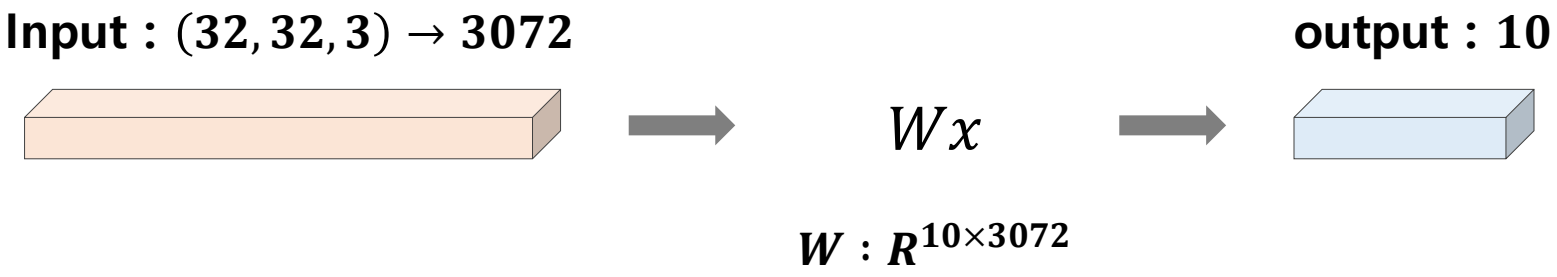


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

https://world4jason.gitbooks.io/research-log/content/deepLearning/CNN/Model%20&%20ImgNet/lenet.html

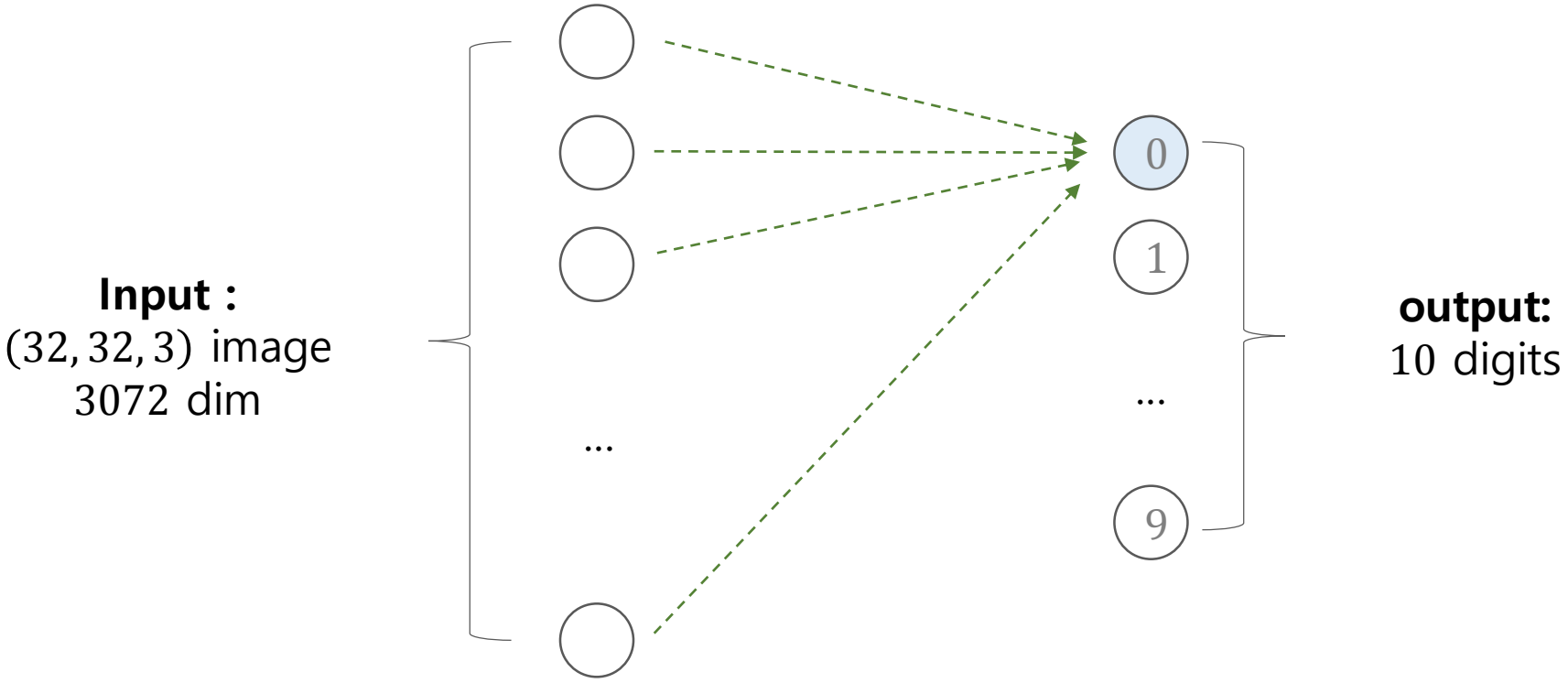
Convolutional Neural Network

- Feed-forward (fully connected) neural network 는 이미지 전체를 한 번에 인식합니다.



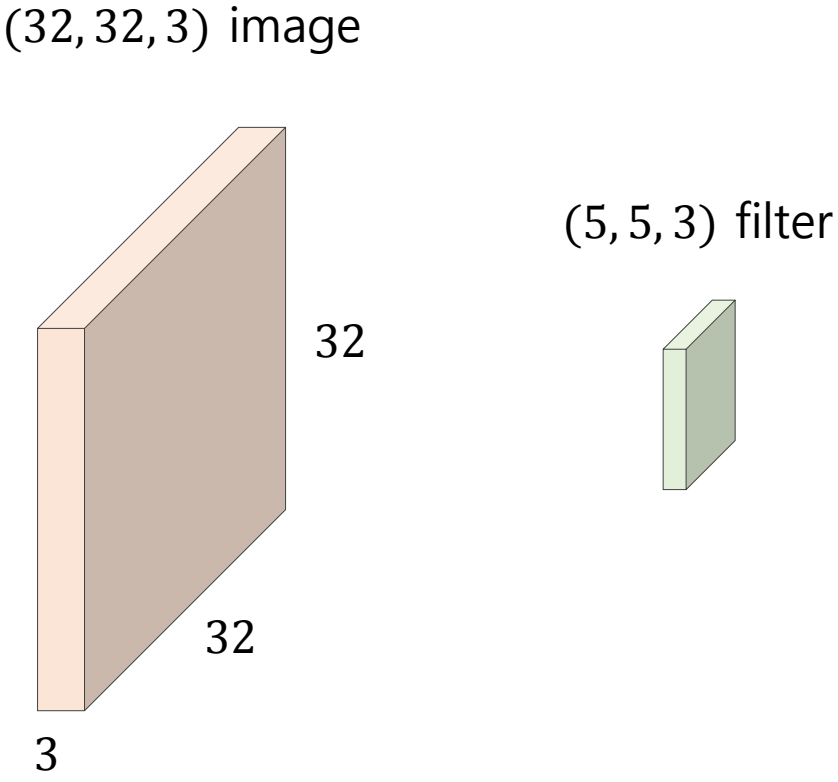
Convolutional Neural Network

- Feed-forward (fully connected) neural network 는 input 이 output (hidden) 에 모두 연결된 형태입니다.



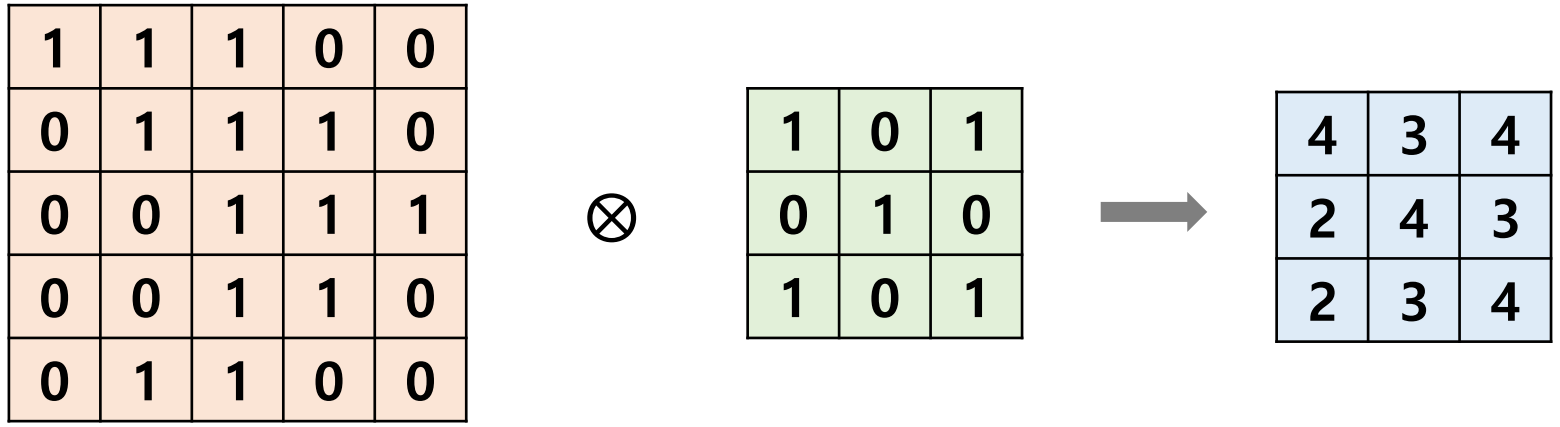
Convolutional Neural Network

- CNN 은 filter 를 이용하여 이미지에 특정한 visual component 가 존재하는지를 확인합니다.



Convolutional Neural Network

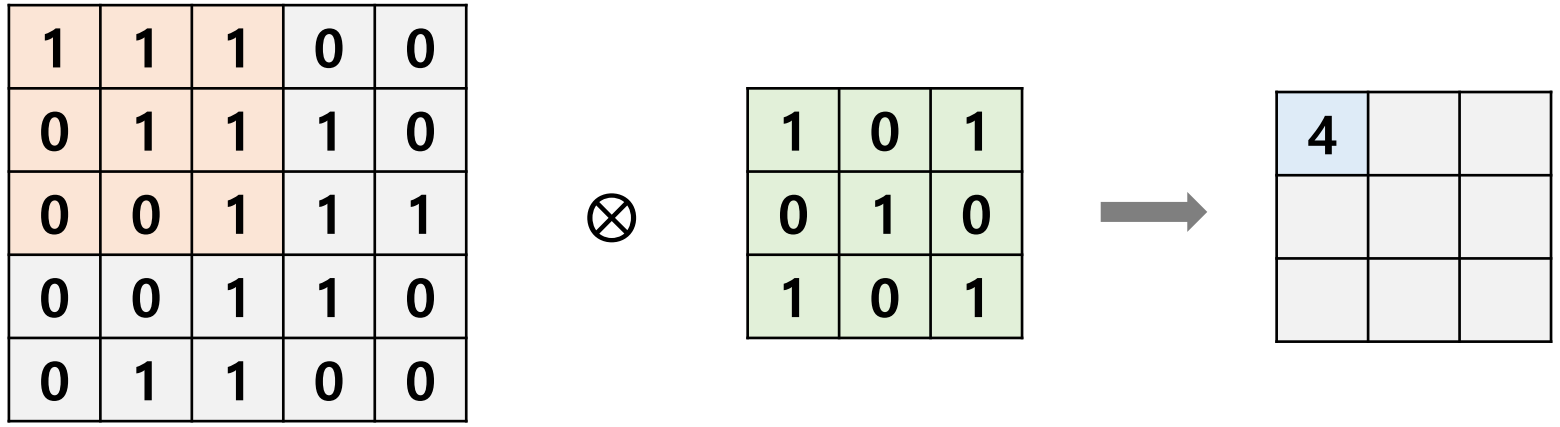
- 필터는 이미지를 슬라이딩하며, element-wise dot product 를 합니다.
  - (5, 5, 1) image 에 (3, 3, 1) filter 가 적용된 예시로 살펴봅니다.
  - Output image 의 크기는 (3, 3, 1) 입니다.



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

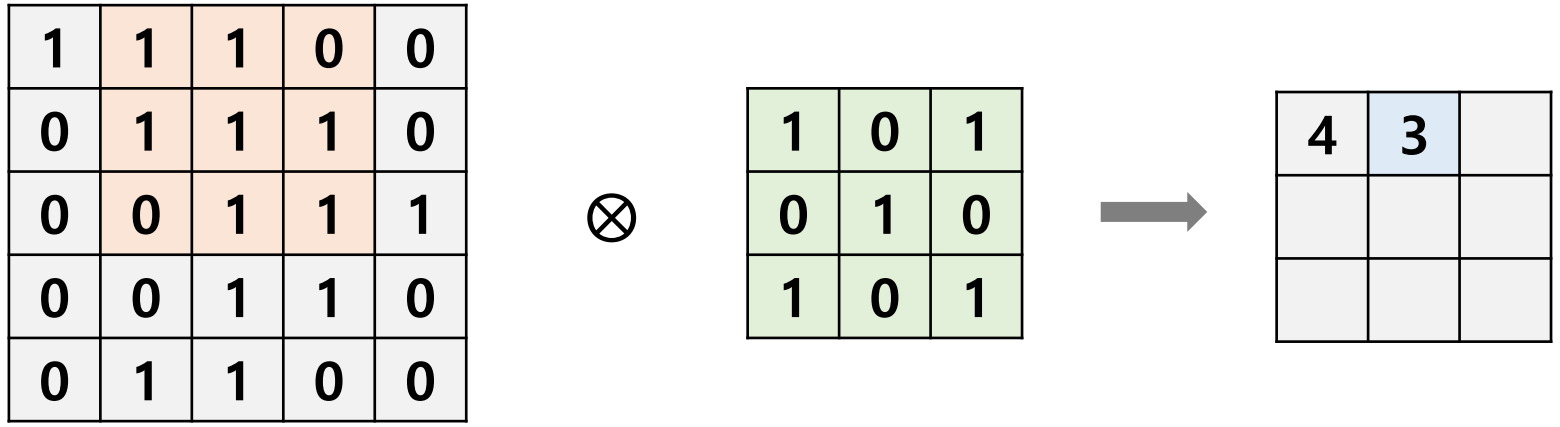
Convolutional Neural Network

- 필터는 이미지를 슬라이딩하며, element-wise dot product 를 합니다.



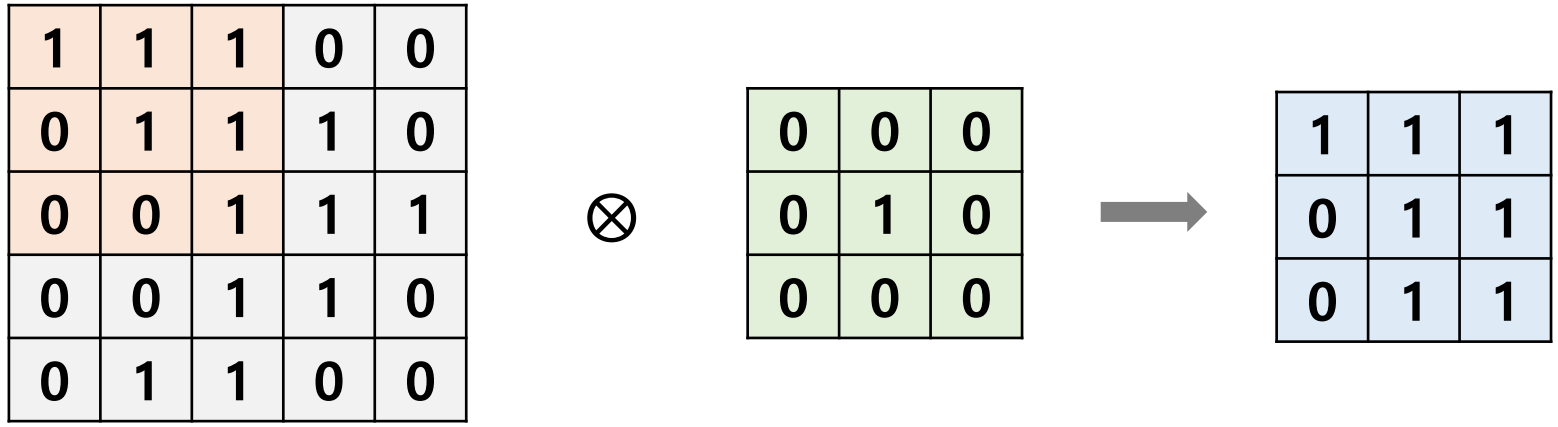
Convolutional Neural Network

- 필터는 이미지를 슬라이딩하며, element-wise dot product 를 합니다.
  - 한 칸 (stride) 씩 움직이며 output image 를 만듭니다.



Convolutional Neural Network

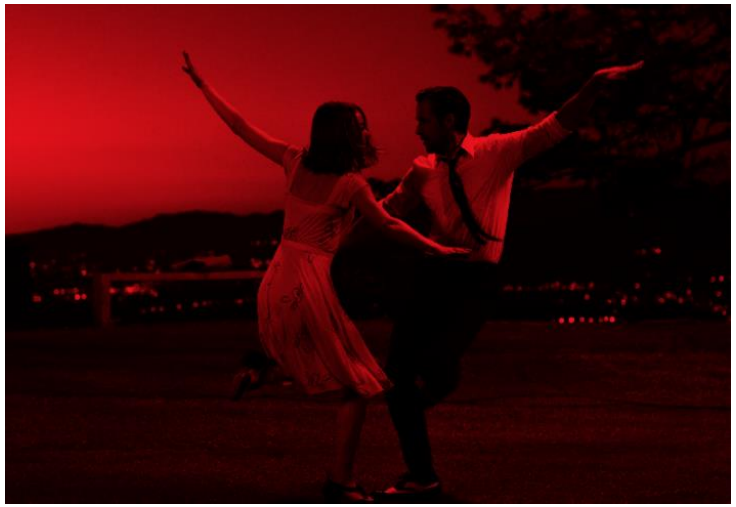
- 다른 필터를 이용하면 output image 는 달라집니다.



Convolutional Neural Network

- 필터는 이미지를 보는 관점의 역할을 합니다.
  - RGB 로 표현된 이미지에서 R 부분만을 취하기 위해서는 [[(1, 0, 0)]] 의 필터를 이미지에 적용합니다.

$(1,1,3)$ 크기의 필터
[[(1, 0, 0)]] 적용



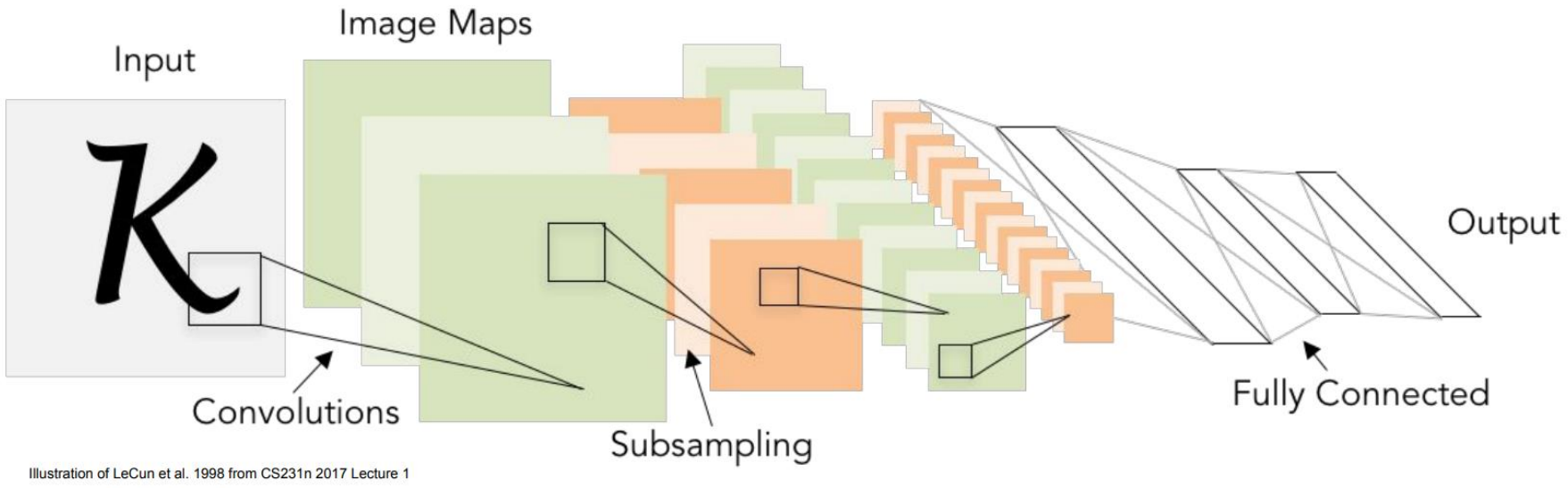
Convolutional Neural Network

- 필터는 이미지를 보는 관점의 역할을 합니다.
  - 이미지의 경계선을 찾을 수도 있습니다.

-1	-1	-1
-1	8	-1
-1	-1	-1

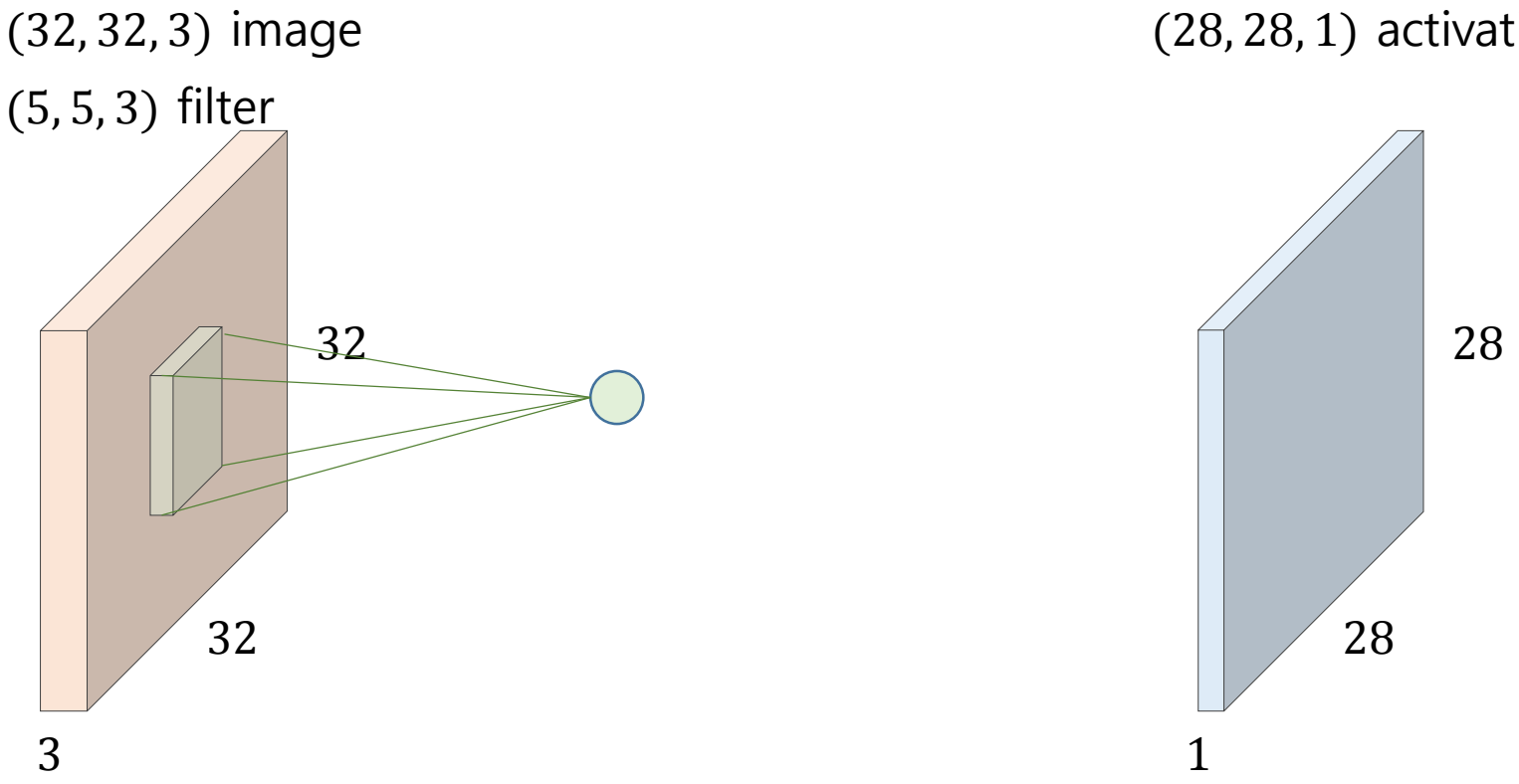
Convolutional Neural Network

- CNN 은 (image, output) pair data 로부터, classification 을 잘 수행할 수 있는 filter 도 학습합니다. Hand-crafted filter 를 이용하지 않아도 됩니다.



Convolutional Neural Network

- Output image 를 activation map 이라 하며, 각 filter 가 activation 되는 지점이 표시된 이미지 지도입니다.

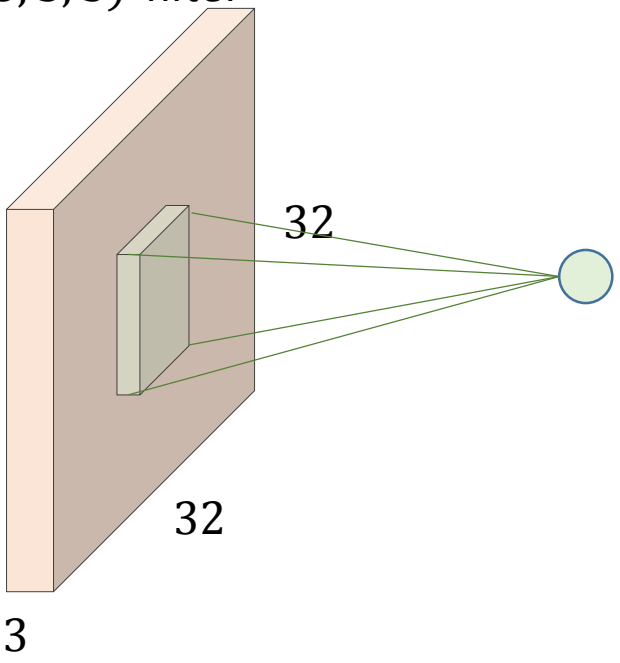


Convolutional Neural Network

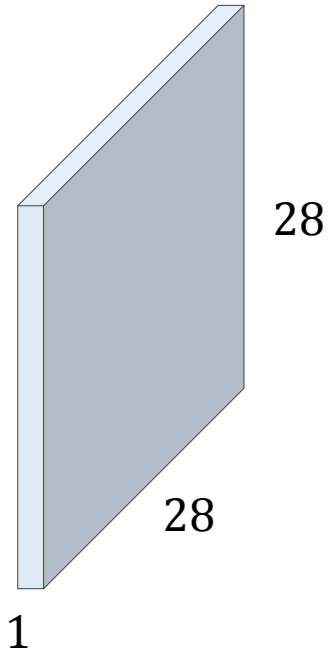
- $(5, 5, 3)$ filter 는 세가지 색 (3 channels) 의 정보를 종합적으로 해석합니다.

$(32, 32, 3)$ image

$(5, 5, 3)$ filter

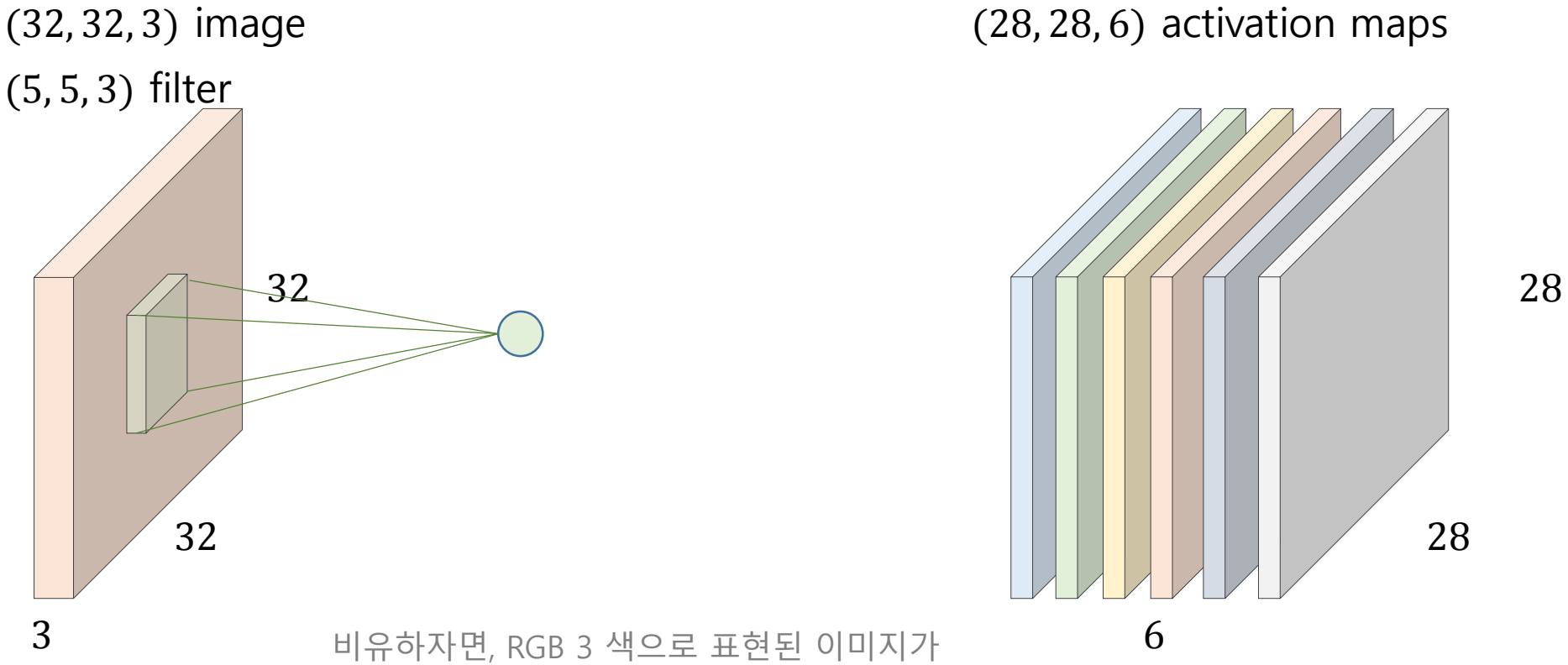


$(28, 28, 1)$ activation map



Convolutional Neural Network

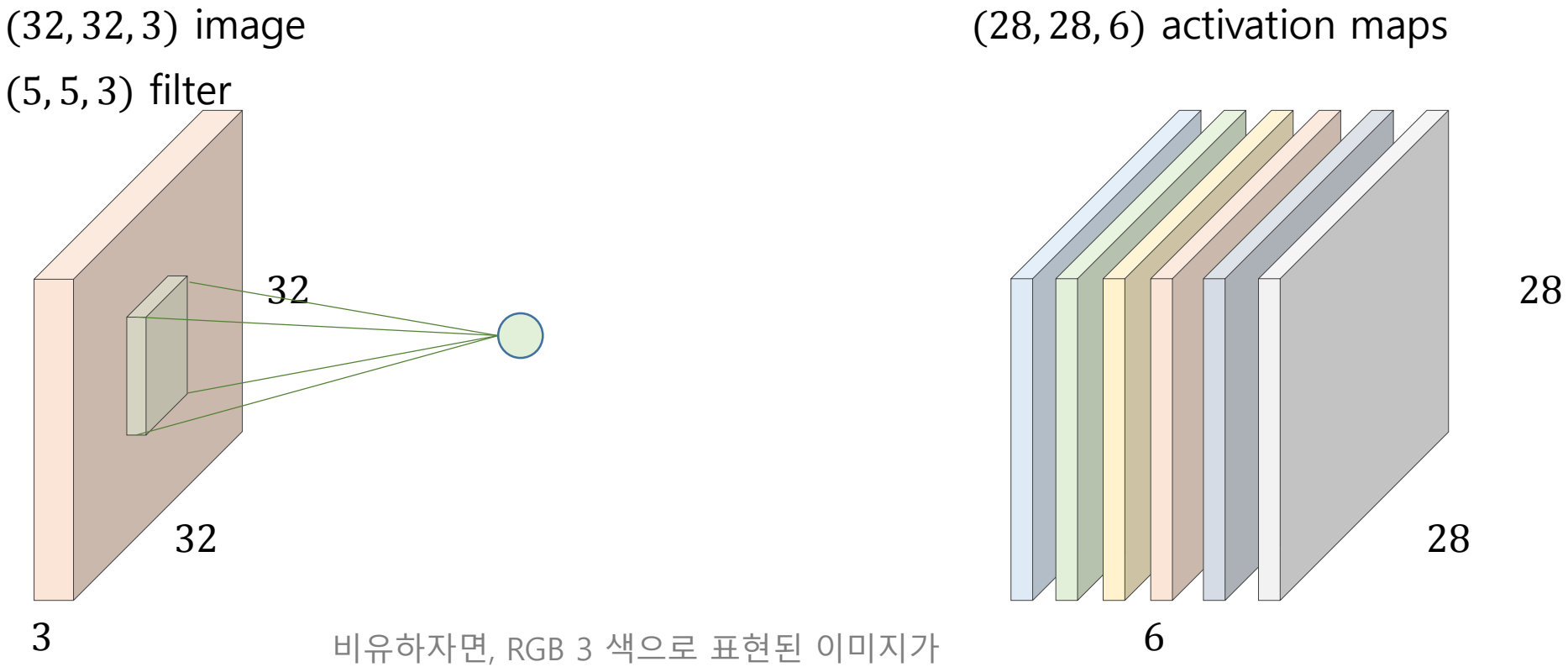
- Filter 별로 activation maps 를 쌓습니다. 각 관점별로 이미지를 해석한 결과로 이미지의 representation 을 변환합니다.



비유하자면, RGB 3 색으로 표현된 이미지가 6 색으로 변환된 것과 같습니다.

Convolutional Neural Network

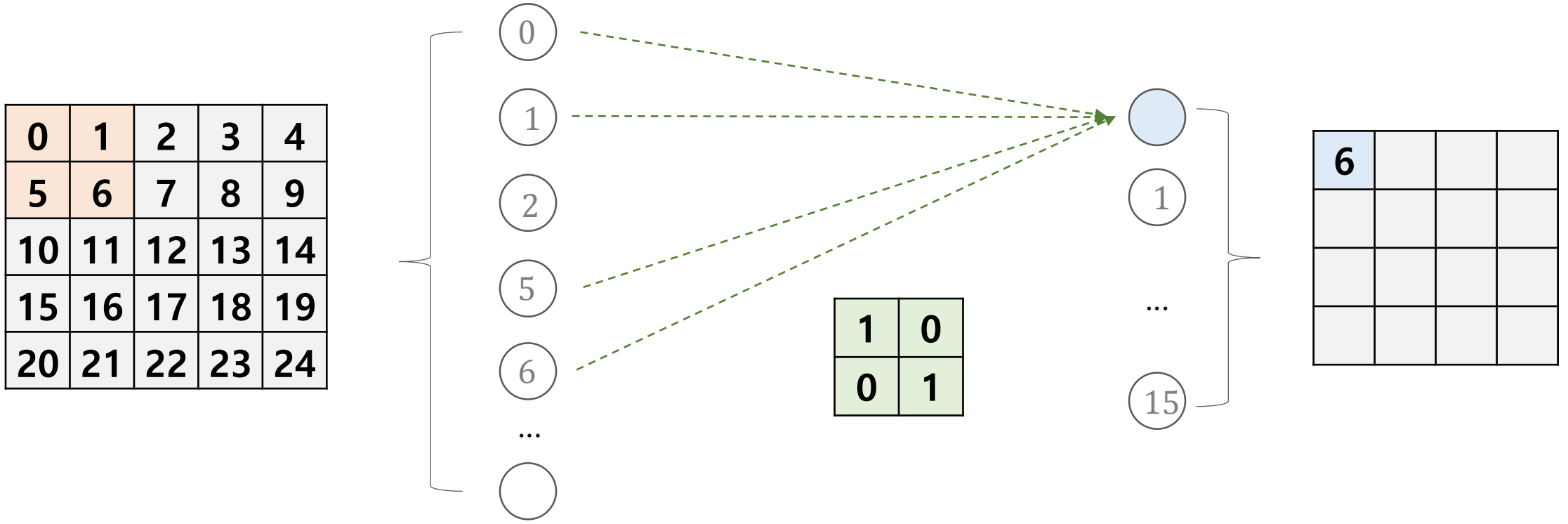
- 6 개의 filters 가 적용된 activation maps 은 다음 convolution layer 에서 6 channel 의 새로운 이미지(벡터)로 해석됩니다.



비유하자면, RGB 3 색으로 표현된 이미지가 6 색으로 변환된 것과 같습니다.

Convolutional Neural Network

- Fully connected vs. Convolutional network ?
  - Convolutional network 는 input 과 hidden 간의 일부만 연결한 구조입니다.



Convolutional Neural Network

- 이미지 데이터는 전체를 한 번에 인식하는 것보다, 각 부분에 특정한 요소들이 존재하는지가 더 중요합니다. 데이터를 local 하게 살펴보는 것이 더 좋습니다.
- CNN 은 이미지 데이터의 특징을 잘 반영할 수 있도록 구조를 설계한 neural network 입니다.
  - Locality

Convolutional Neural Network

- CNN 은 다음의 요소로 구성되어 있습니다.
  - convolution
  - subsampling
  - activation

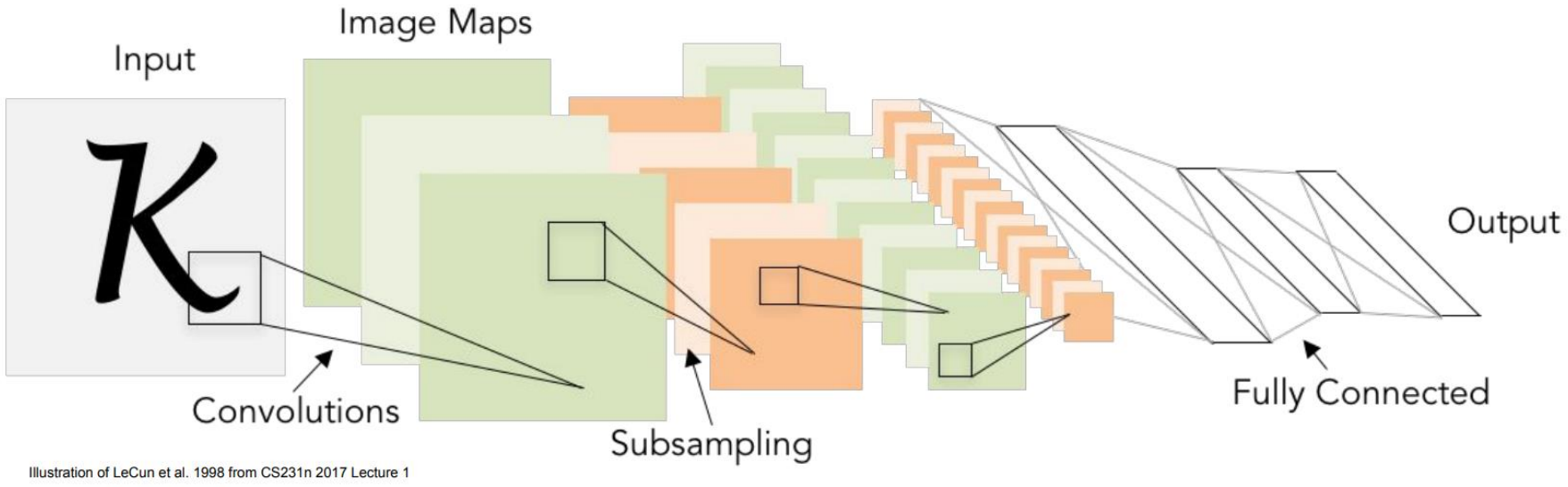


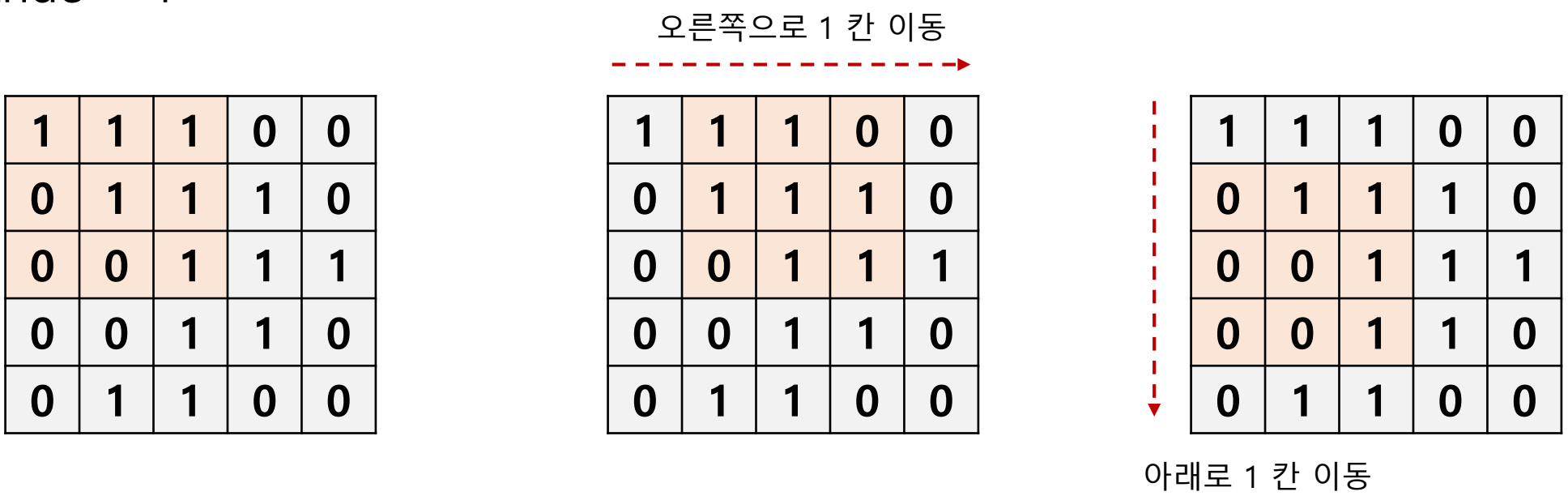
Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Convolutional Neural Network

- Convolution
  - Filters (number & size)
    - 몇 개의 필터를 이용할 것인가?
    - 각 필터의 크기는 어떻게 설정할 것인가?
  - Stride
    - 필터를 몇 칸씩 움직일 것인가?
  - Padding
    - Activation map 의 크기를 input image 와 일정하게 맞출 것인가?

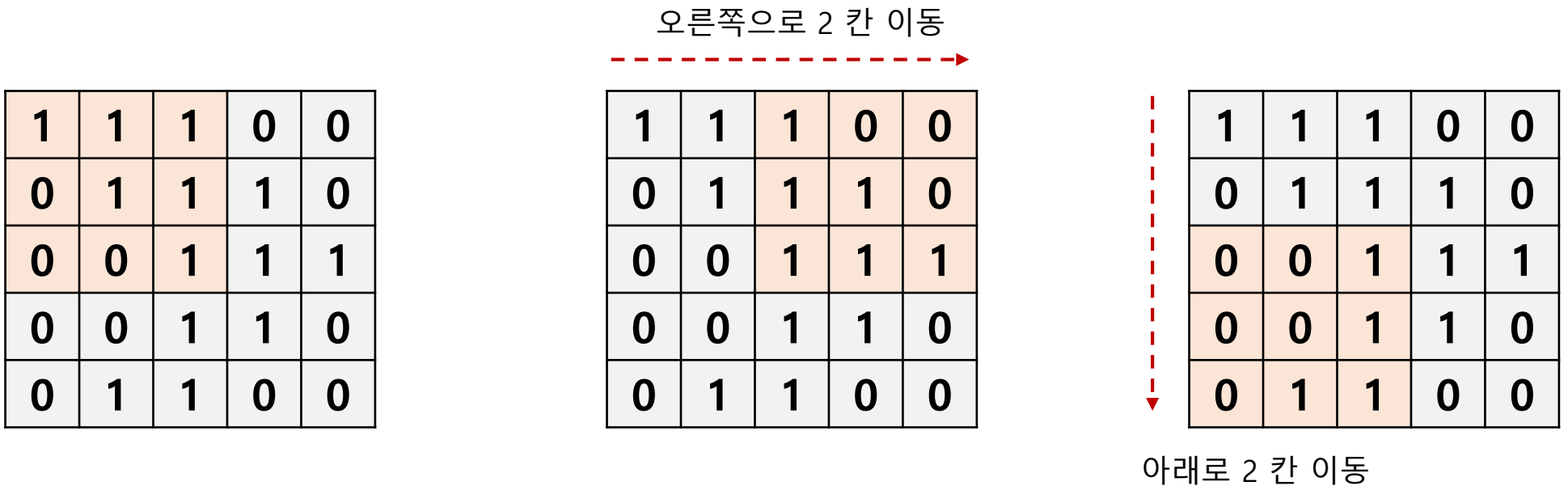
Stride

- Stride 는 filter 가 움직이는 크기입니다.
  - stride = 1



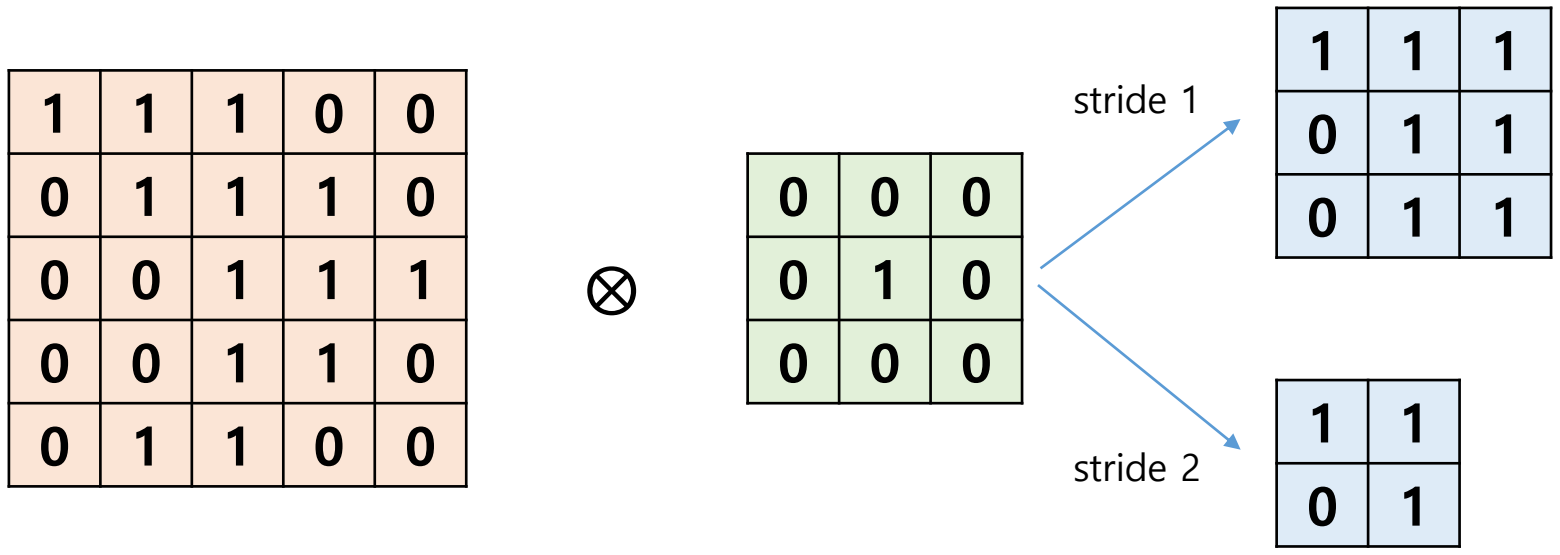
Stride

- Stride 는 filter 가 움직이는 크기입니다.
  - stride = 2



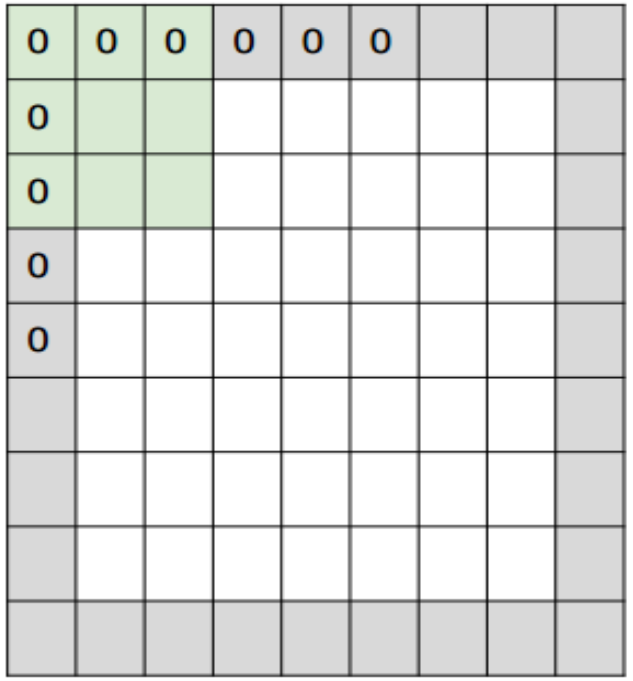
Stride

- Stride 에 따라 activation map 의 크기가 달라집니다.
  - $(5,5,1) \otimes (3,3,1)$, stride 1 → $(3,3,1)$
  - $(5,5,1) \otimes (3,3,1)$, stride 2 → $(2,2,1)$



Padding

- 필터의 크기가 (n, m, c) , $n \ or \ m > 1$ 이면
  - activation map 의 크기가 input image 보다 작아집니다.
  - 이미지의 가장자리는 필터가 적용되지 않습니다.
- (zero) Padding 은 이미지의 경계에 0 을 추가합니다.

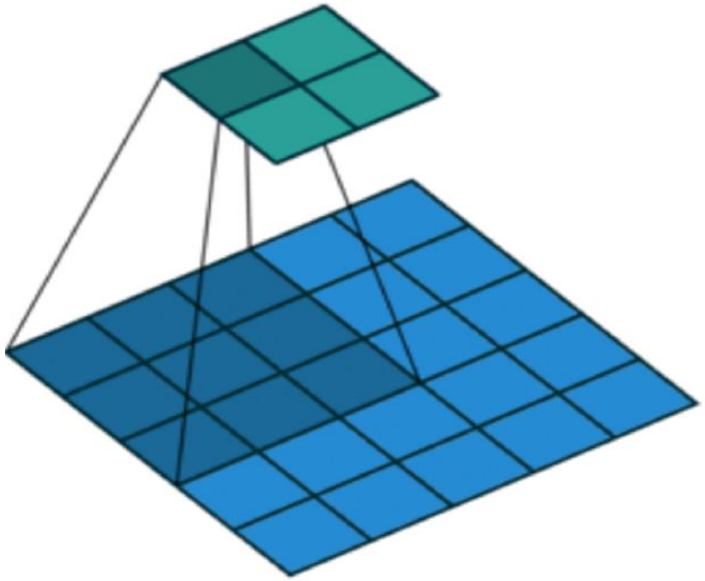


Padding

- Activation map 의 크기는 다음의 공식으로 계산할 수 있습니다.
  - $W` = \frac{W - F + 2P}{S} + 1$
    - $W`$: activation map 의 크기
    - W : input image 의 크기
    - F : filter 의 크기
    - P : padding 의 크기
    - S : stride

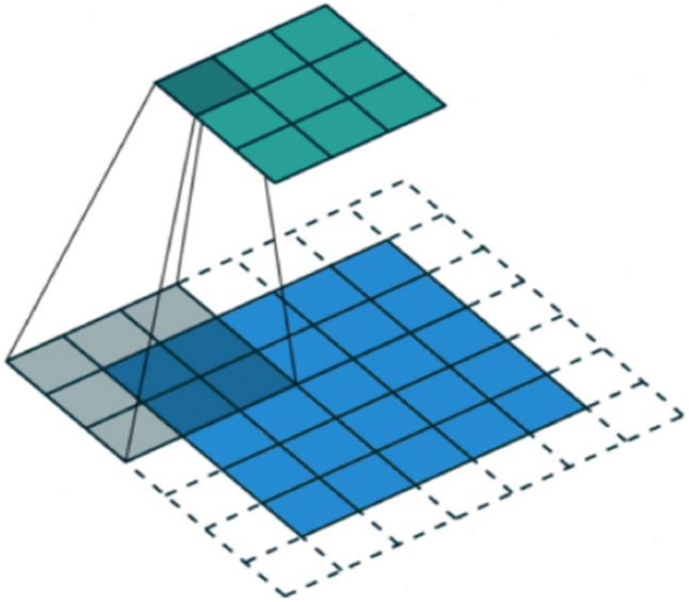
Padding

- $W` = \frac{W - F + 2P}{S} + 1$



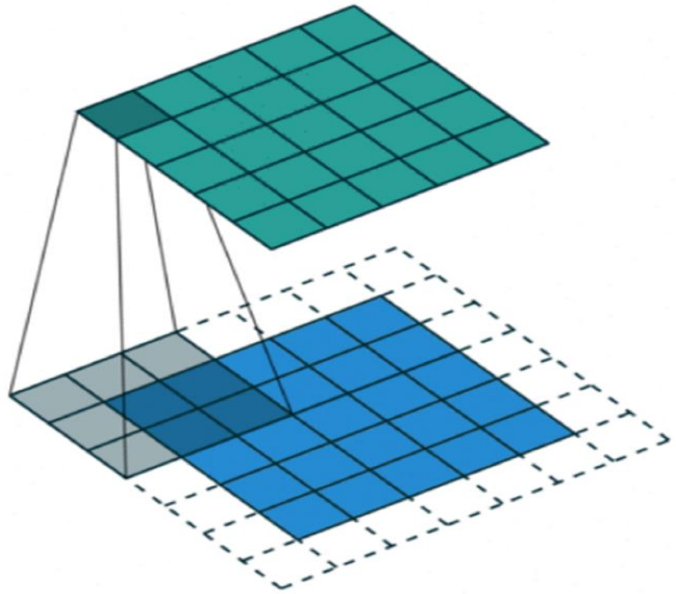
Input : (5,5), filter : (3,3)
Stride = 2
Padding = 0

$$W` = \frac{5 - 3 + 2 \times 0}{2} + 1 = 2$$



Input : (5,5), filter : (3,3)
Stride = 2
Padding = 1

$$W` = \frac{5 - 3 + 2 \times 1}{2} + 1 = 3$$

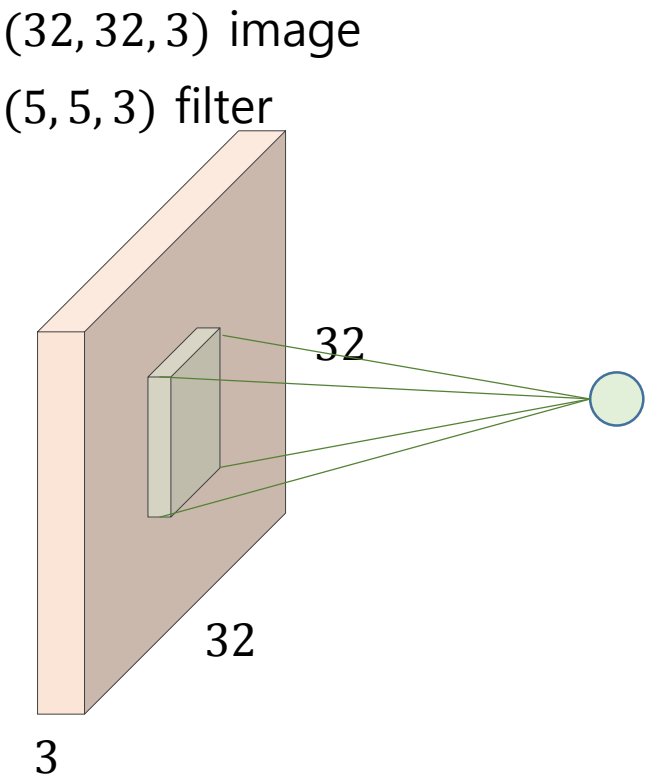


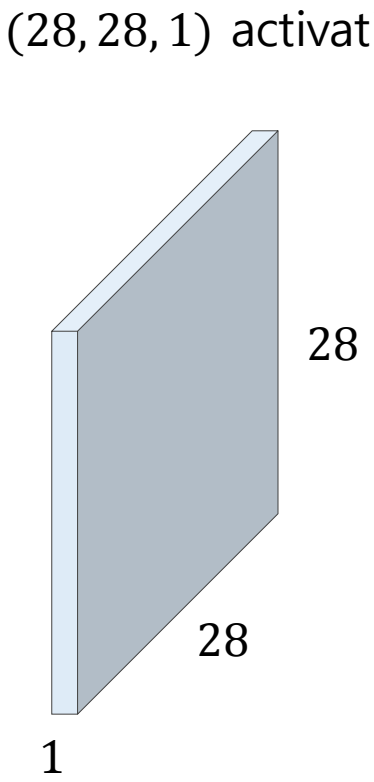
Input : (5,5), filter : (3,3)
Stride = 1
Padding = 1

$$W` = \frac{5 - 3 + 2 \times 1}{1} + 1 = 5$$

Subsampling : Pooling

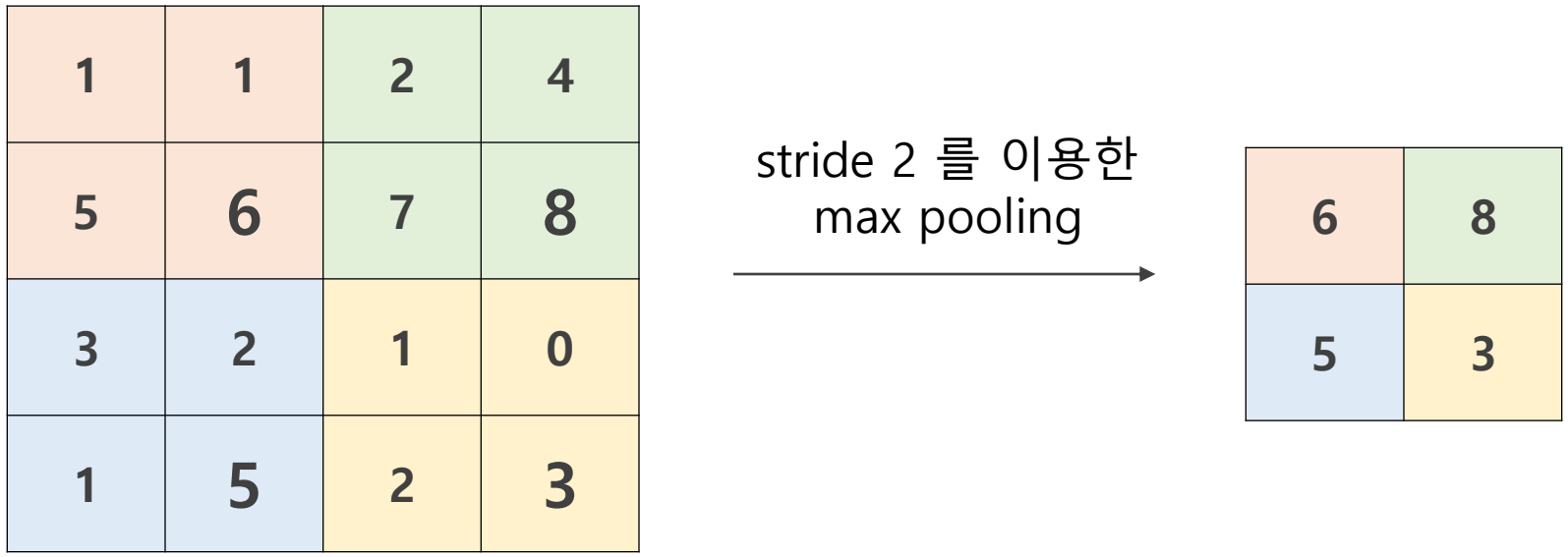
- Activation map 의 크기가 여전히 크며 $(28, 28, \#f)$, stride 를 하기 때문에 인접한 부분은 중복적인 정보를 지니기도 합니다.





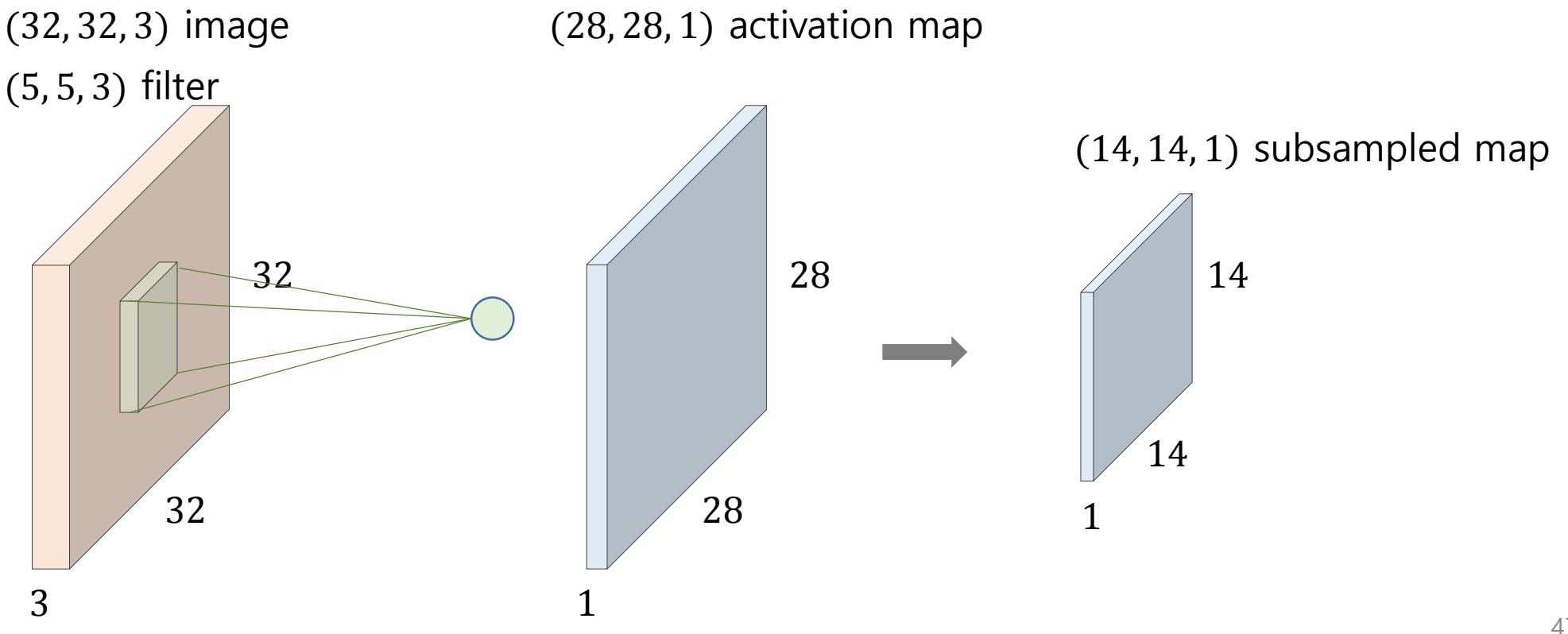
Subsampling : Pooling

- Max-pooling 은 activation map 에서 중요한 정보를 취합니다.



Subsampling : Pooling

- Pooling 은 activation map 에서 중요한 정보를 요약합니다.



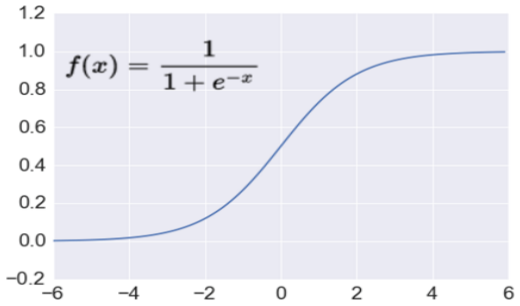
Subsampling : Pooling

- 몇 가지 자주 이용되는 pooling functions 이 있습니다.
  - Max pooling
  - Average pooling
  - k-max pooling

Activation functions

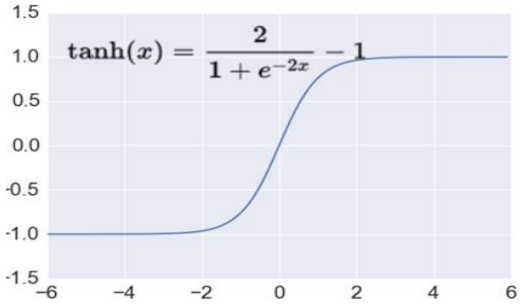
- (pooling 을 거친) activation maps 의 값에 비선형성을 더해줍니다.

sigmoid



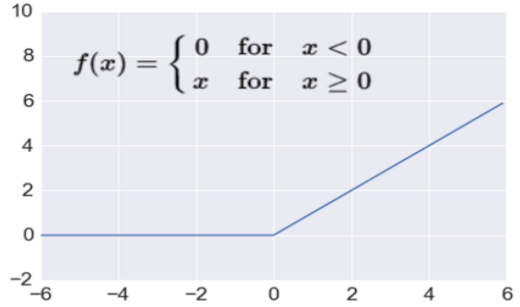
$$f(x) = \frac{1}{1+e^{-x}}$$

Hyper-tangent



$$f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

ReLU



$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

Activation functions

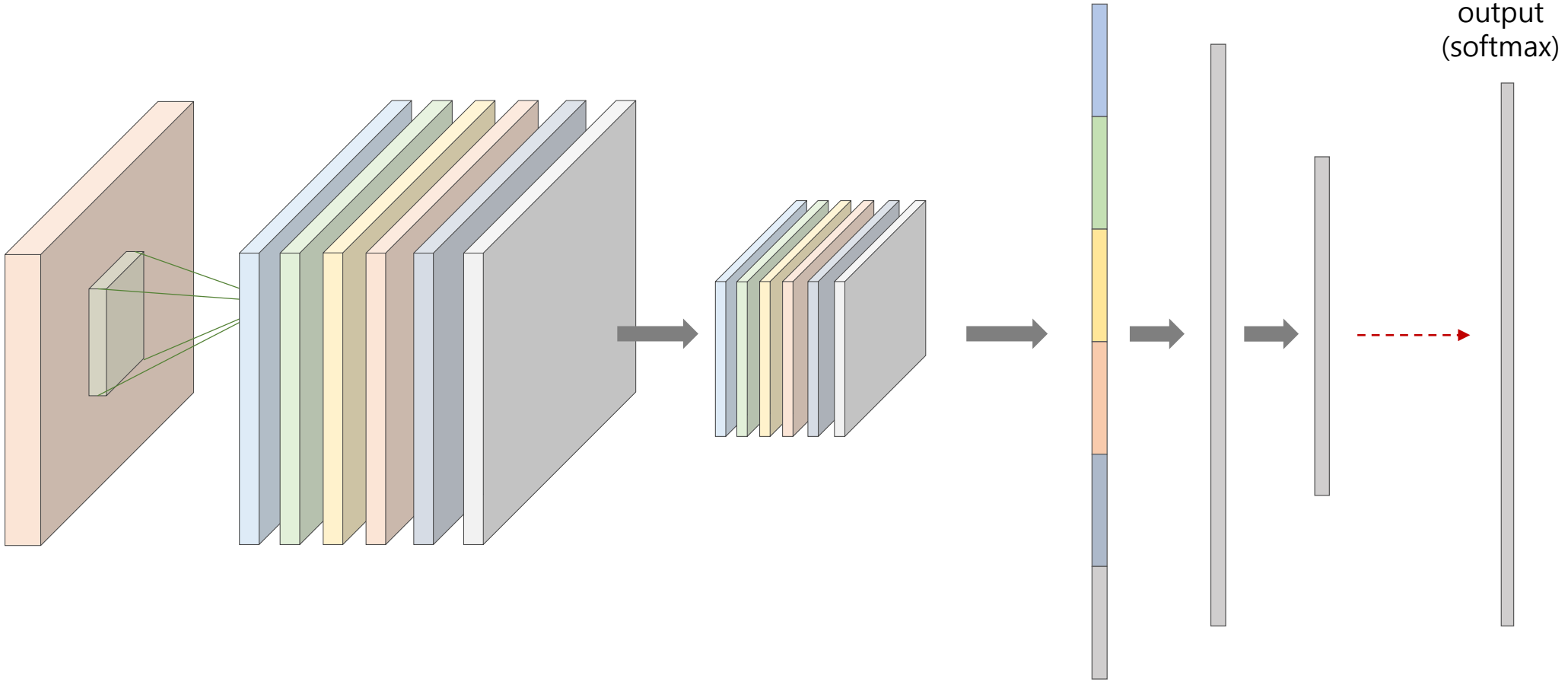
- 초기의 neural network 에서는 sigmoid 나 hyper-tangent 가 많이 이용되었지만, 최근에는 학습의 안정성과 효율성 때문에 ReLU 나 Leaky ReLU 가 많이 이용됩니다.

  - ReLU : $f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$

  - Leaky ReLU : $f(x) = \begin{cases} 0.01x & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$

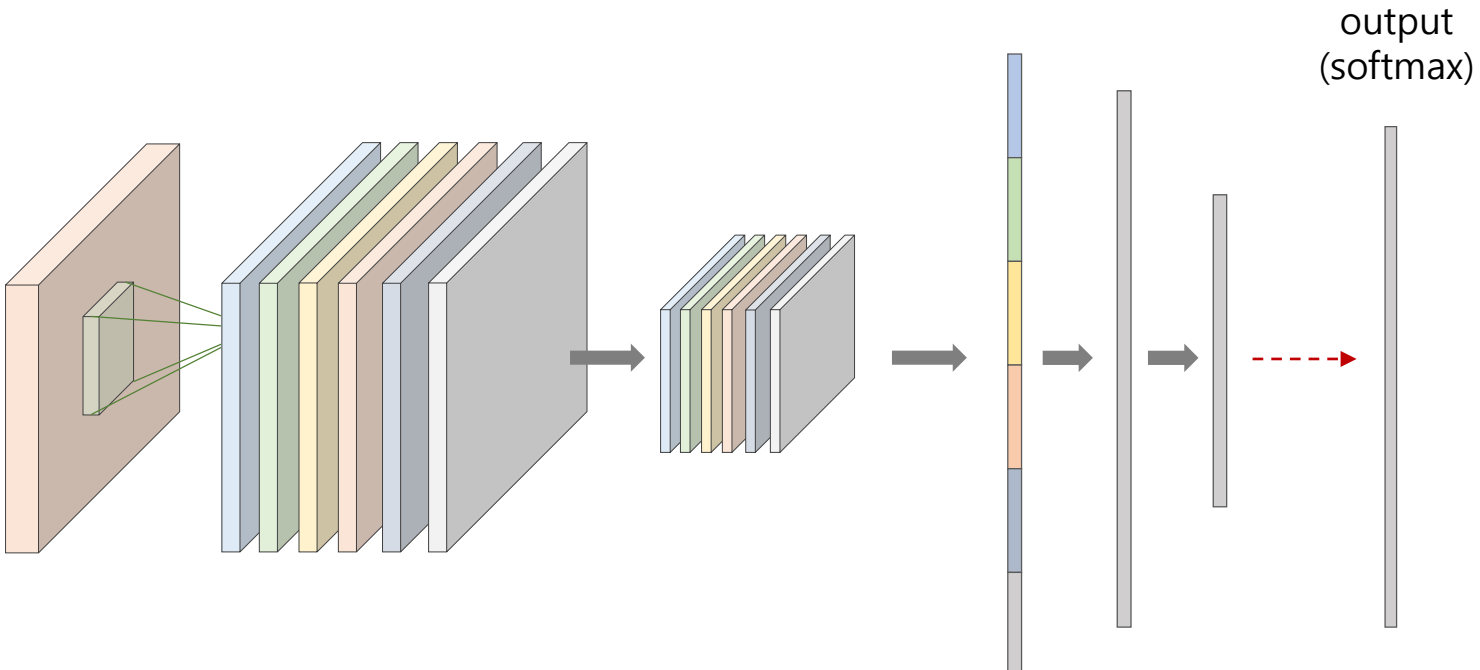
Fully connected layer

- FC 는 activation maps 를 1 개의 row 로 concatenation 합니다.



Convolutional Neural Network

- CNN 은 여러 개의 관점 (filters)으로 이미지를 해석한 뒤,
  유용한 정보만을 남겨 (pooling),
  linear separable 한 벡터로 (fully connected) 데이터의 representation 을
  변환합니다.



Revisit LeNet

- Convolution, pooling 을 거친 뒤, 2 개의 fully connected layer 를 이용하여 판별합니다.

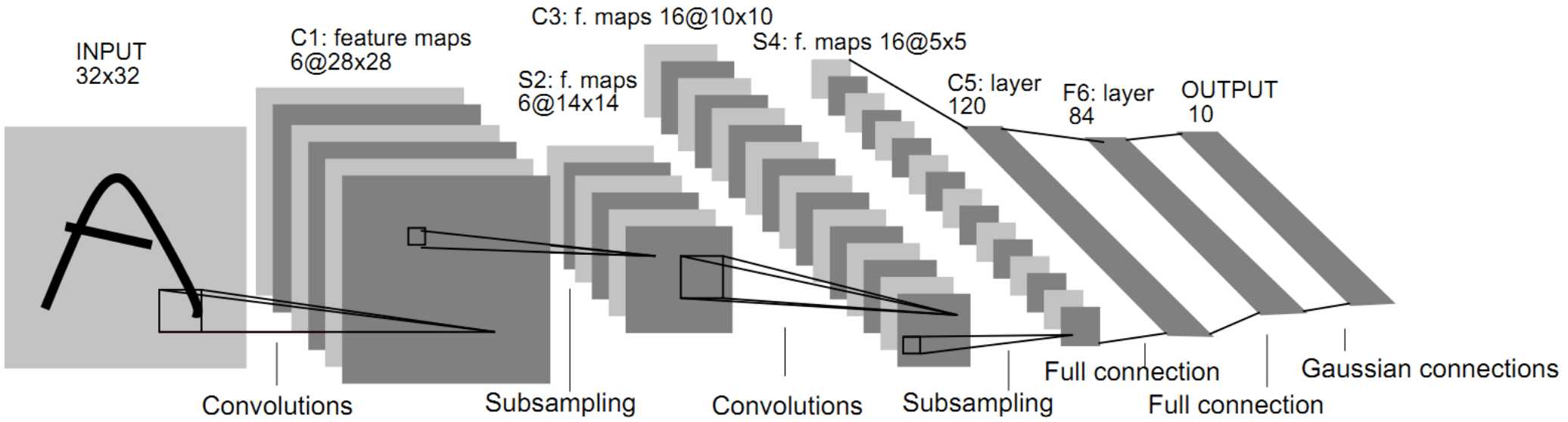
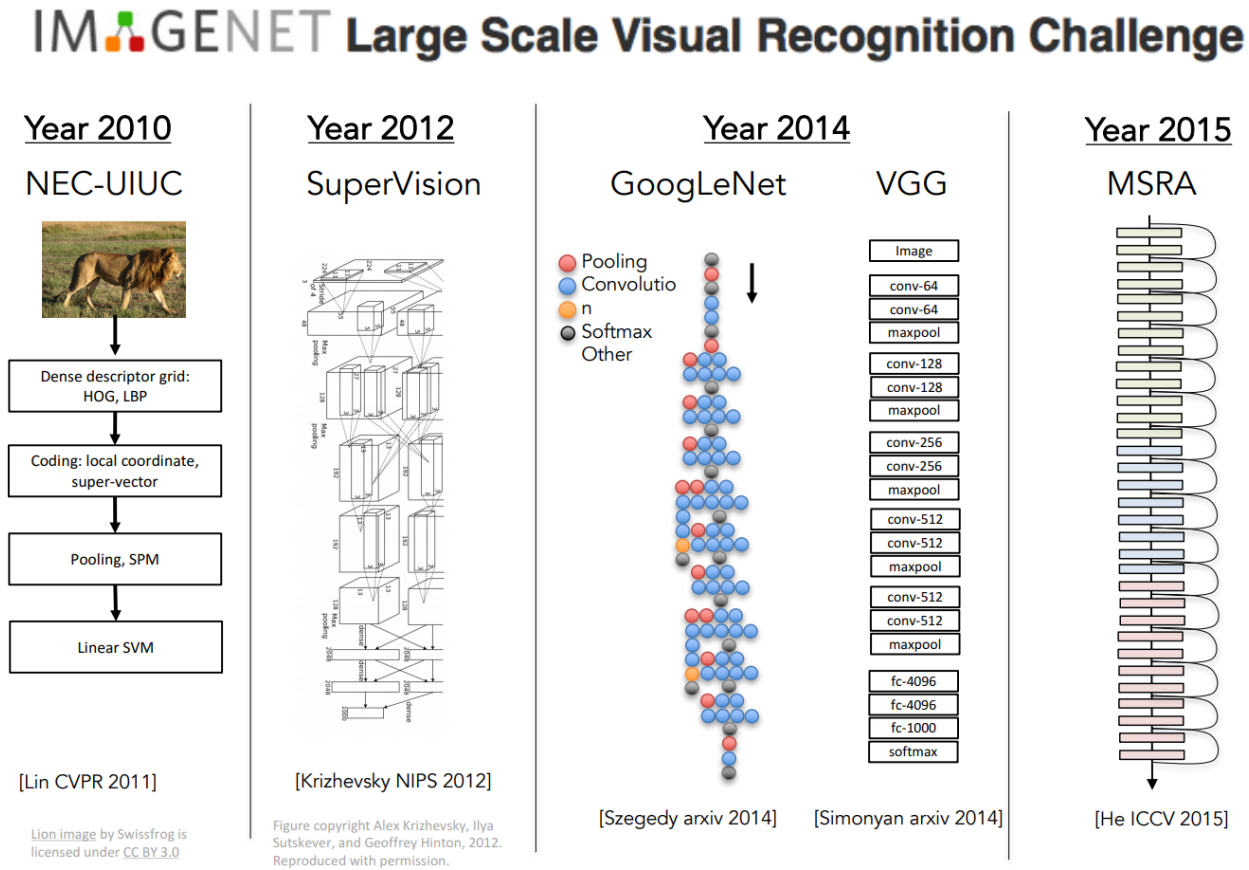


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Convolutional Neural Network

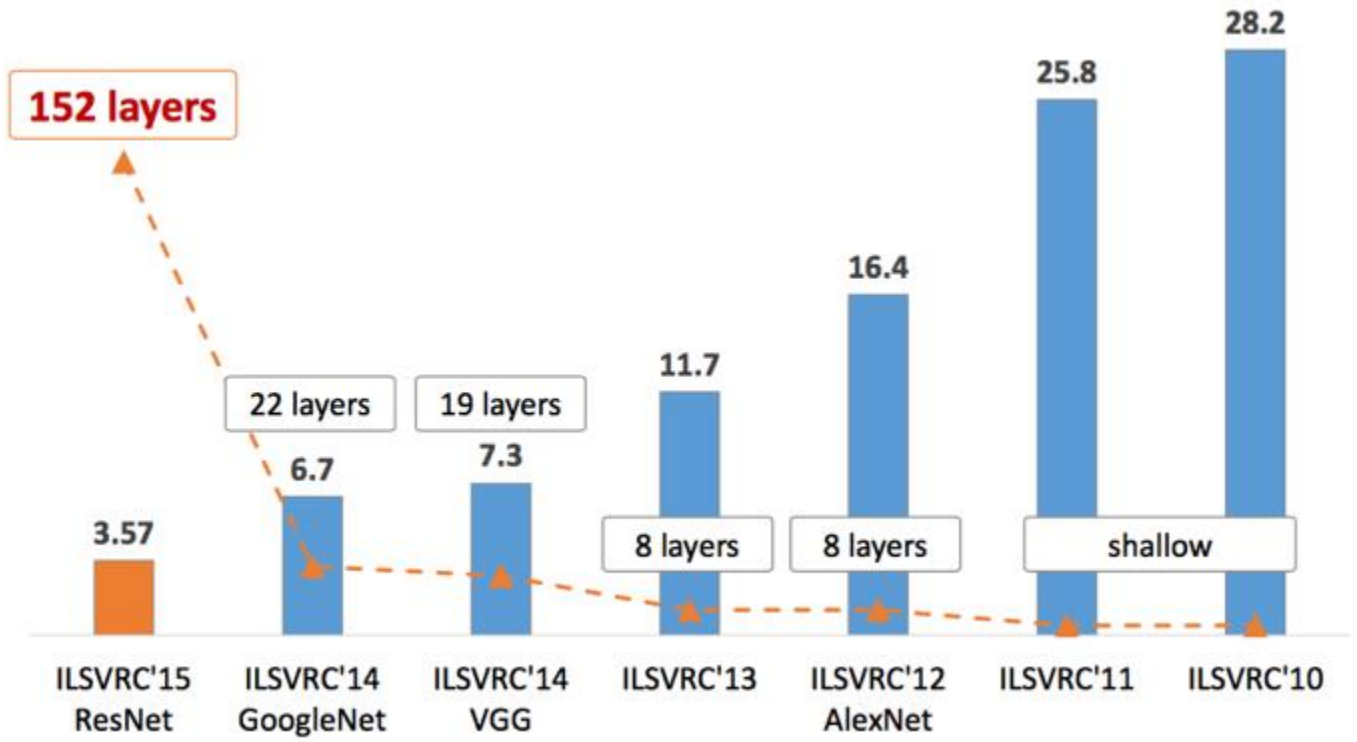
- CNN models 의 발전은 ILSVRC 과 함께 되었습니다.
  - ImageNet Large Scale Visual Recognition Challenge



* http://www.image-net.org/
* http://cs231n.stanford.edu/

Convolutional Neural Network

- CNN models 의 발전은 ILSVRC 과 함께 되었습니다.
  - Image recognition 은 사람의 인식 능력 (약 5 % 의 error) 보다도 높아졌습니다.



https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Convolutional Neural Network

- CNN 의 발전은 computer vision 에서 이뤄졌습니다.
- 이후, audio signal 이나 natural language processing 문제들에서도 CNN 이 응용되기 시작했습니다.