이화여자대학교
EWHA WOMANS UNIVERSITY

Lecture 5

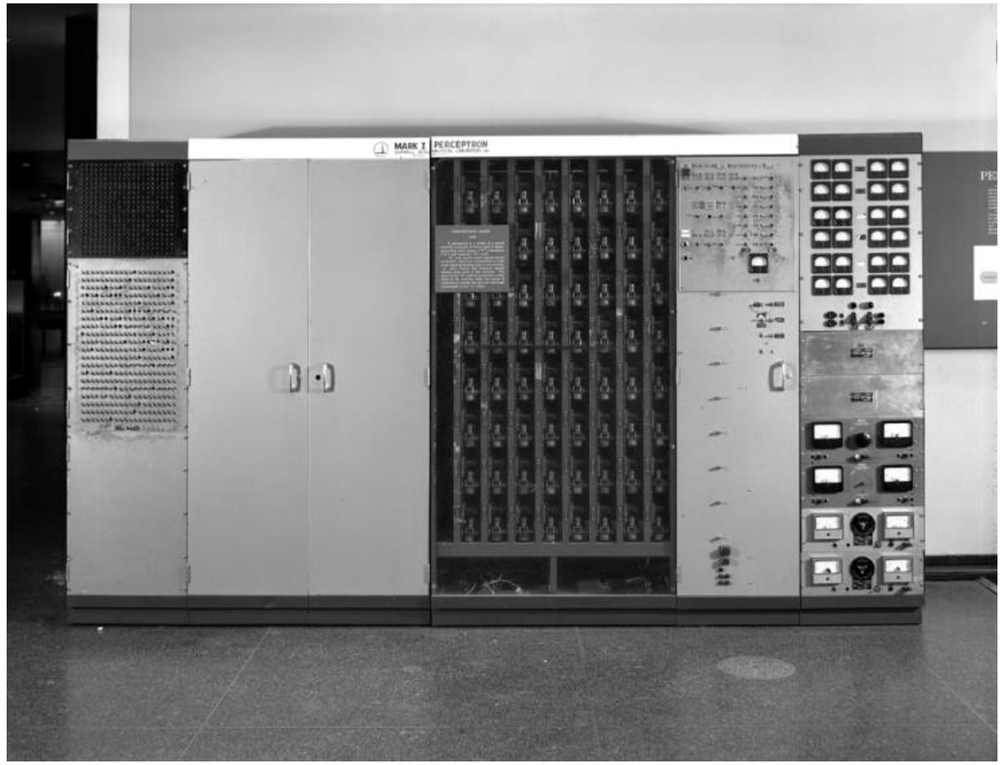
Convolutional Neural Networks

2070015 김세빈

Cotents

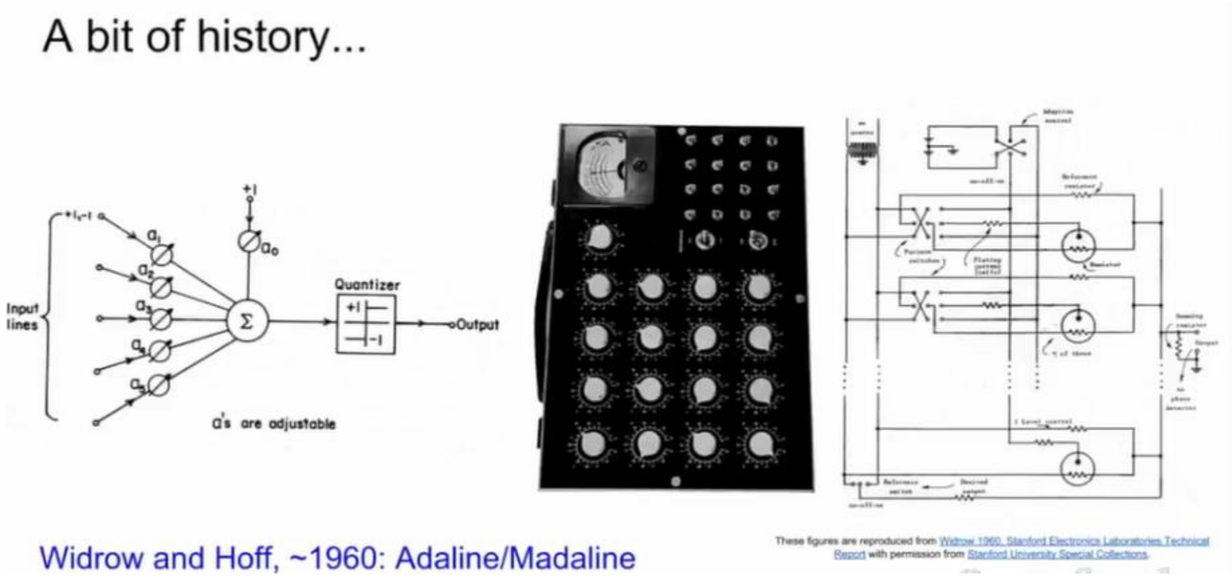
1. The History of Neural Networks

Mark I Perception machine



- 1957년, Frank Rosenblatt가 개발.
- Perception을 구현한 최초의 기계.
- Wx+b와 유사한 함수를 사용 but 출력값은 1 or 0.
- Backprop과 비슷한 Update rule.

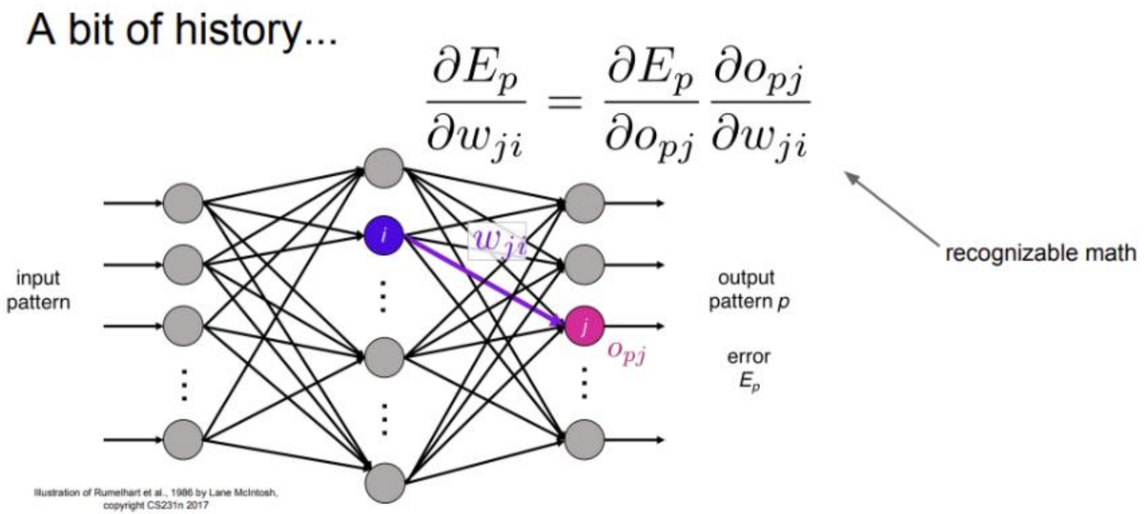
Adaline and Madaline



- 1960년, Widrow와 Hoff가 개발.

- 최초의 Multilayer Perception Network.

First Backprop



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

- 1986년, Rumelhart가 제안.

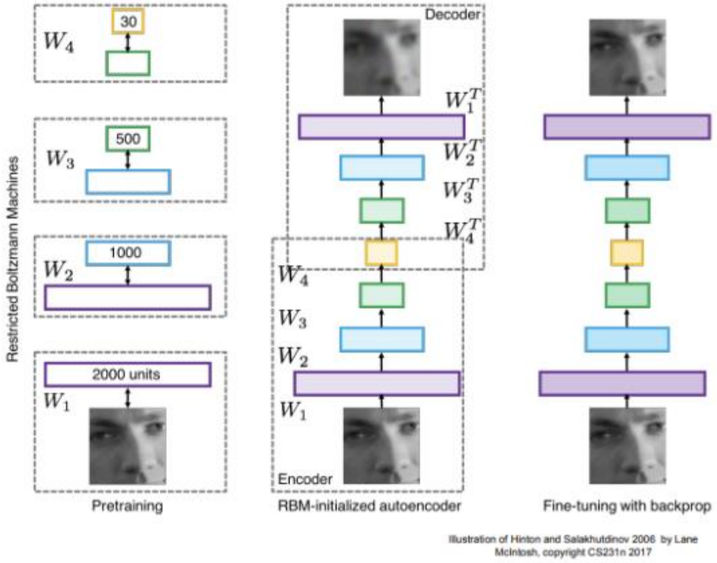
- Chain Rule과 Update Rule을 확인 가능.

DNN Deep Neural Network

A bit of history...

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning



- 2006년, Geoff Hinton과 Salakhutdinov의 논문에서 가능성 제시.

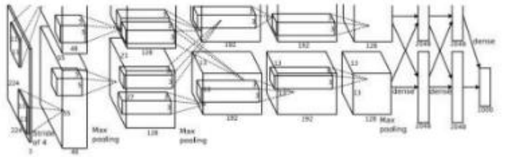
- 초기화를 위해 RBM을 활용하여 각 히든레이어의 가중치를 학습시켜야했음.

The Growth of NN Neural Network

First strong results

Acoustic Modeling using Deep Belief Networks
Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010
Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

Imagenet classification with deep convolutional neural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012



Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduc

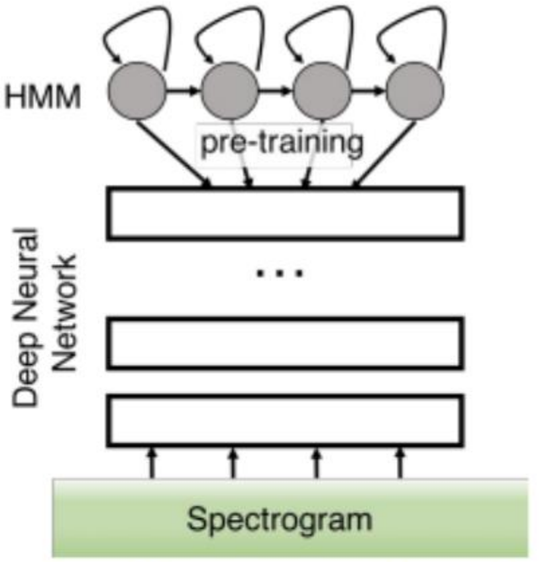
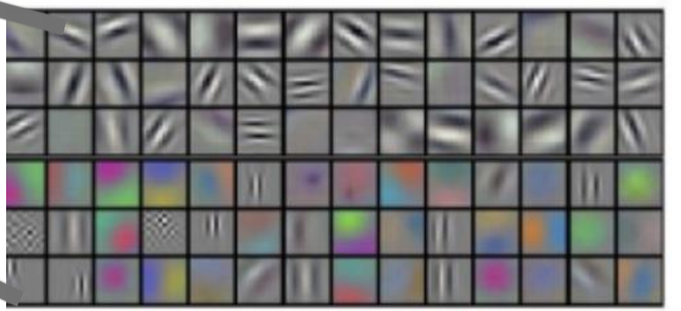


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

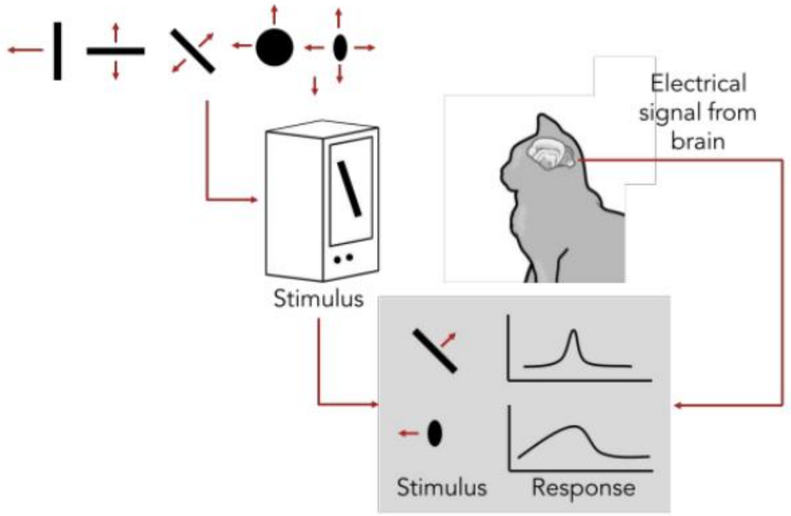


ıced with permission.

- 2012년, NN이 주목받기 시작.
- 음성인식 (Acoustic Modeling, Speech Recognition).
- ImageNet Classification에서 두각(Error 감소).

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...

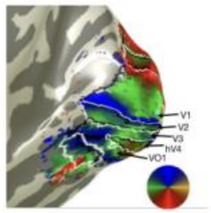


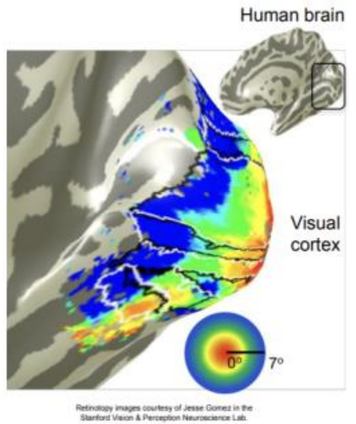
- Hubel & Wiesel이 일차시각피질의 뉴런세포를 연구.

- 뉴런들이 계층적인 구조를 지닌다는 것을 발견.

A bit of history

Topographical mapping in the cortex:
nearby cells in cortex represent
nearby regions in the visual field





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

Hierarchical organization

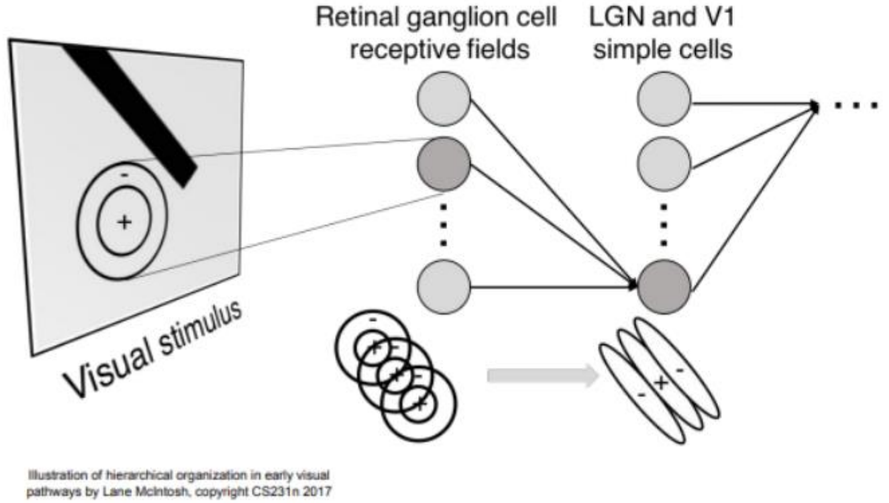
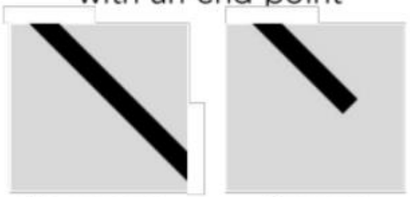


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point

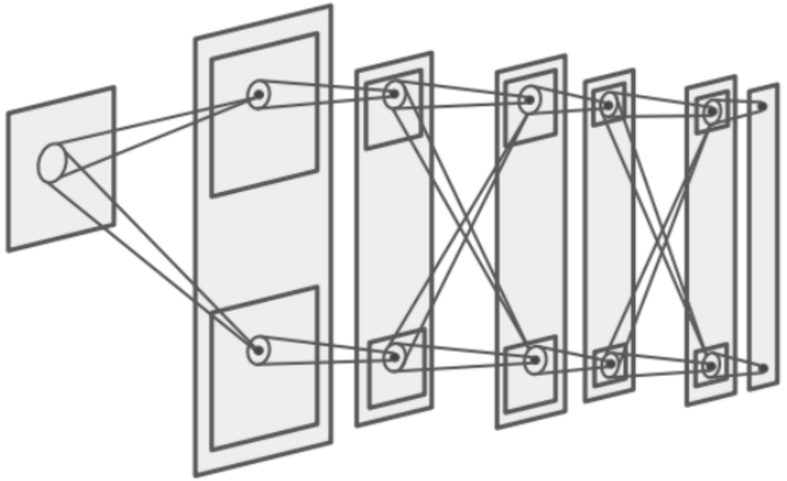


- 시각신호는 망막 신경절(Retinal ganglion)에 가장 먼저 도달.

- 이런 결과로부터 coner나 blob에 대한 아이디어를 얻음.

A bit of history:

Neocognitron
[Fukushima 1980]



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

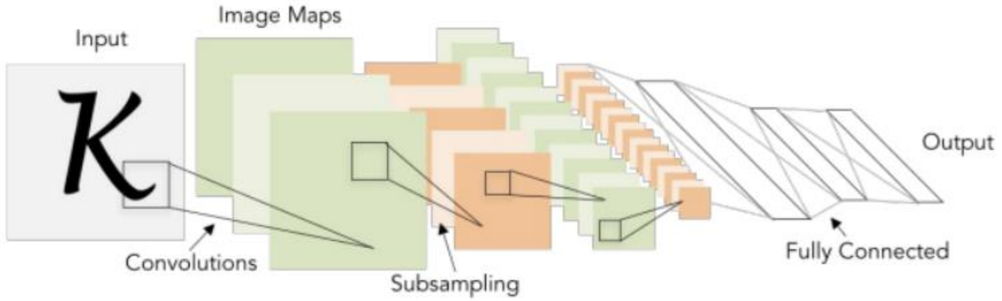
- Simple cells은 학습가능한 parameters를 가지고 있음.

- Complex cells은 pooling을 수행하여 좀 더 정밀한 반응을 보임.

A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

- Yann LeCun이 NN학습을 위해 최초로 Backprob과 gradient-based learning을 적용.

- Complex cells은 pooling을 수행하여 좀 더 정밀한 반응을 보임. (e,g, 우편번호 인식)

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

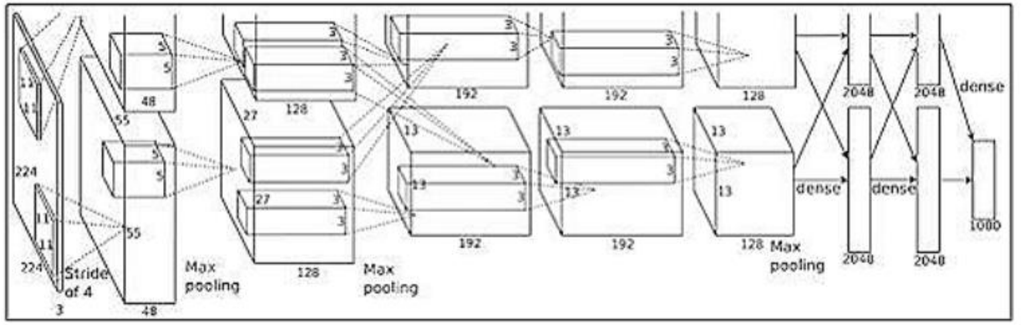


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

Fast-forward to today: ConvNets are everywhere

Classification

Retrieval

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

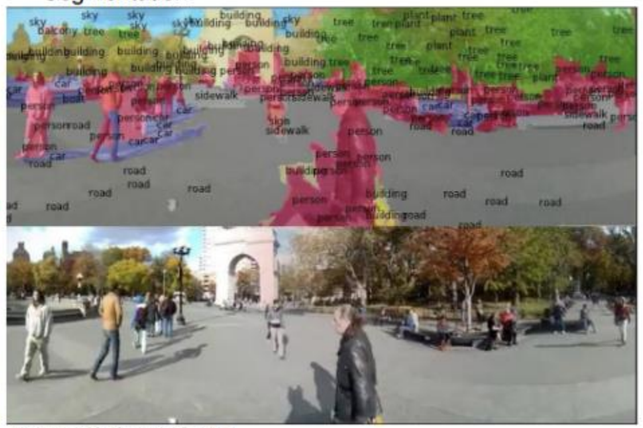
Fast-forward to today: ConvNets are everywhere

Detection

Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, 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]

Fast-forward to today: ConvNets are everywhere

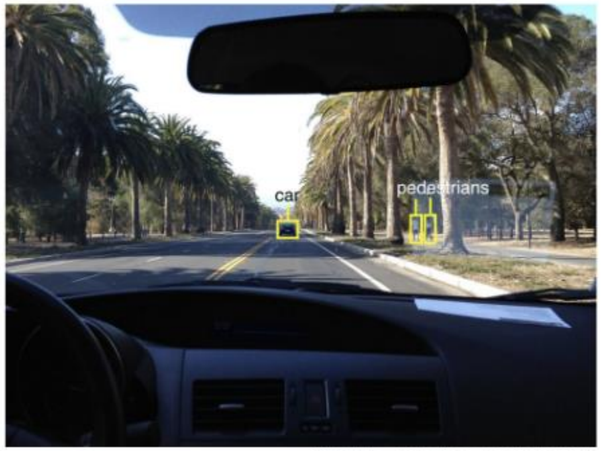


Photo by Lane McIntosh. Copyright CS231n 2017.

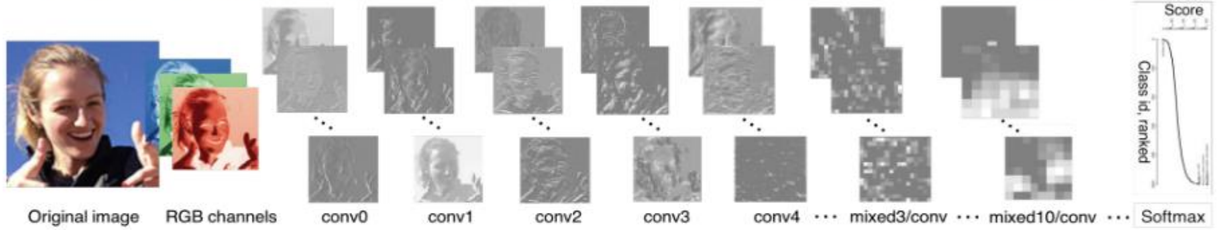
self-driving cars

This image by GBPublic_PR is licensed under CC-BY 2.0

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

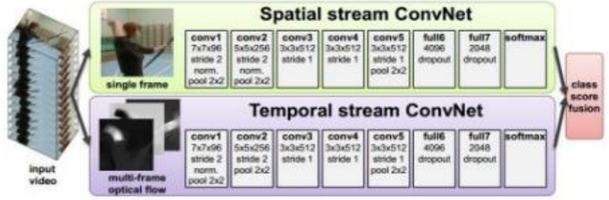
Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Fast-forward to today: ConvNets are everywhere



[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.

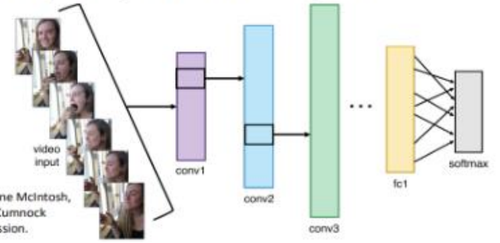
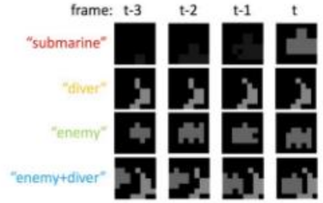


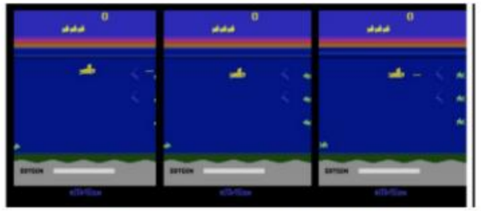
Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

Fast-forward to today: ConvNets are everywhere

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

[Toshev, Szegedy 2014]

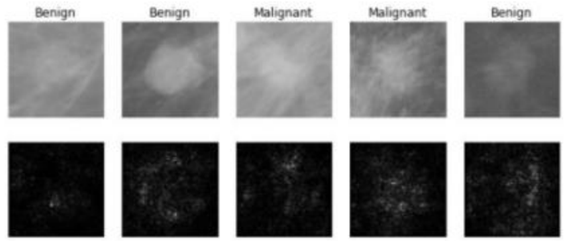




[Guo et al. 2014]

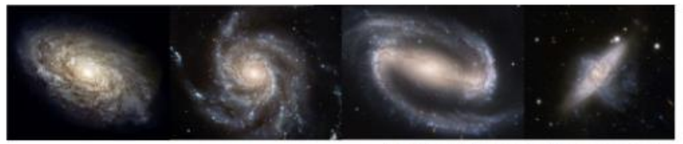
Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

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.

No errors

A white teddy bear sitting in the grass

Minor errors

A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

Captions generated by Justin Johnson using NeuralTalk2

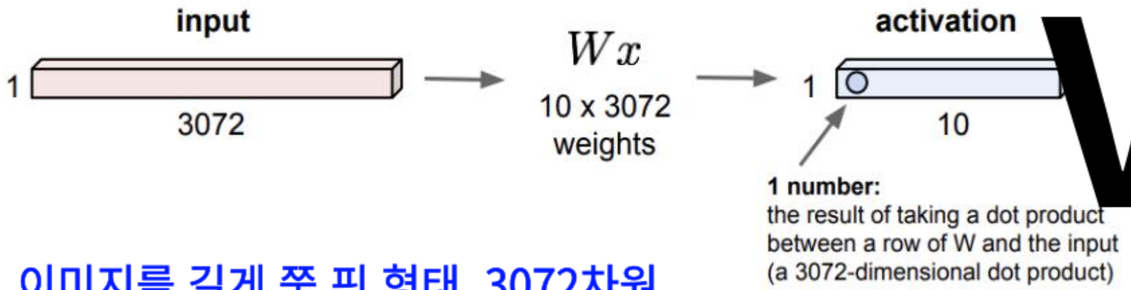
-> Image Captioning

이미지가 주어지면 그에 대한 설명을 문장으로 표현.

2. Convolutional Neural Networks (1)

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



이미지를 길게 쭉 핀 형태, 3072차원

3072차원의 입력과 10개의 행으로 구성된 Weight의 내적

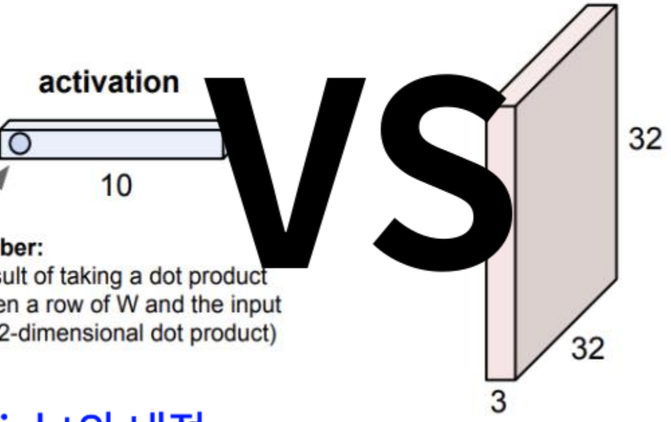
VS

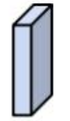
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

5x5x3 filter





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

Convolution Layer는 FC Layer와 달리 기존의 구조를 보존

작은 필터가 Weight가 되며 sliding을 통해 본래 이미지와 공간적으로 내적

Convolution Layer

Filters always extend the full depth of the input volume

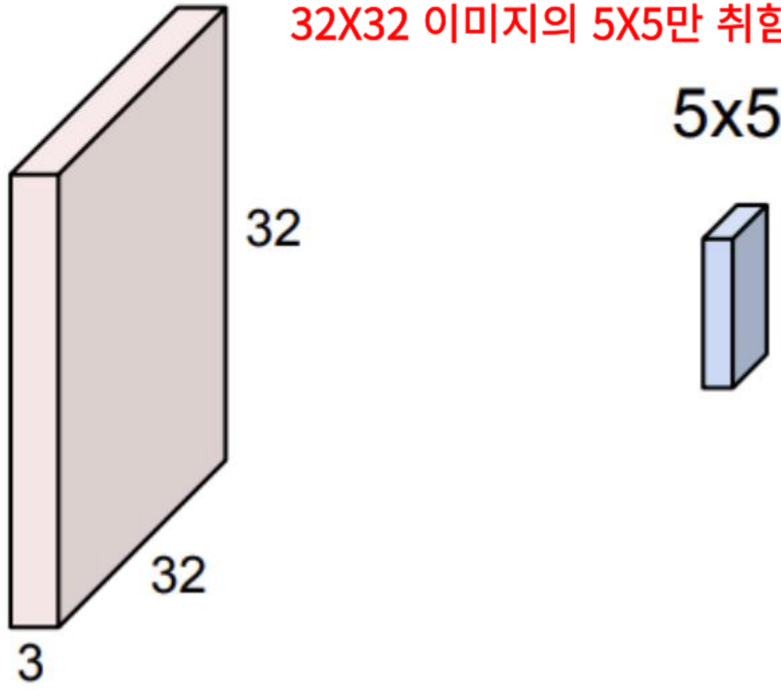
32x32x3 image

본래 이미지의 Depth만큼 확장.

32X32 이미지의 5X5만 취함.

Depth를 어디까지 늘리는 것이 효율적인지는 찾아야함.

5x5x3 filter



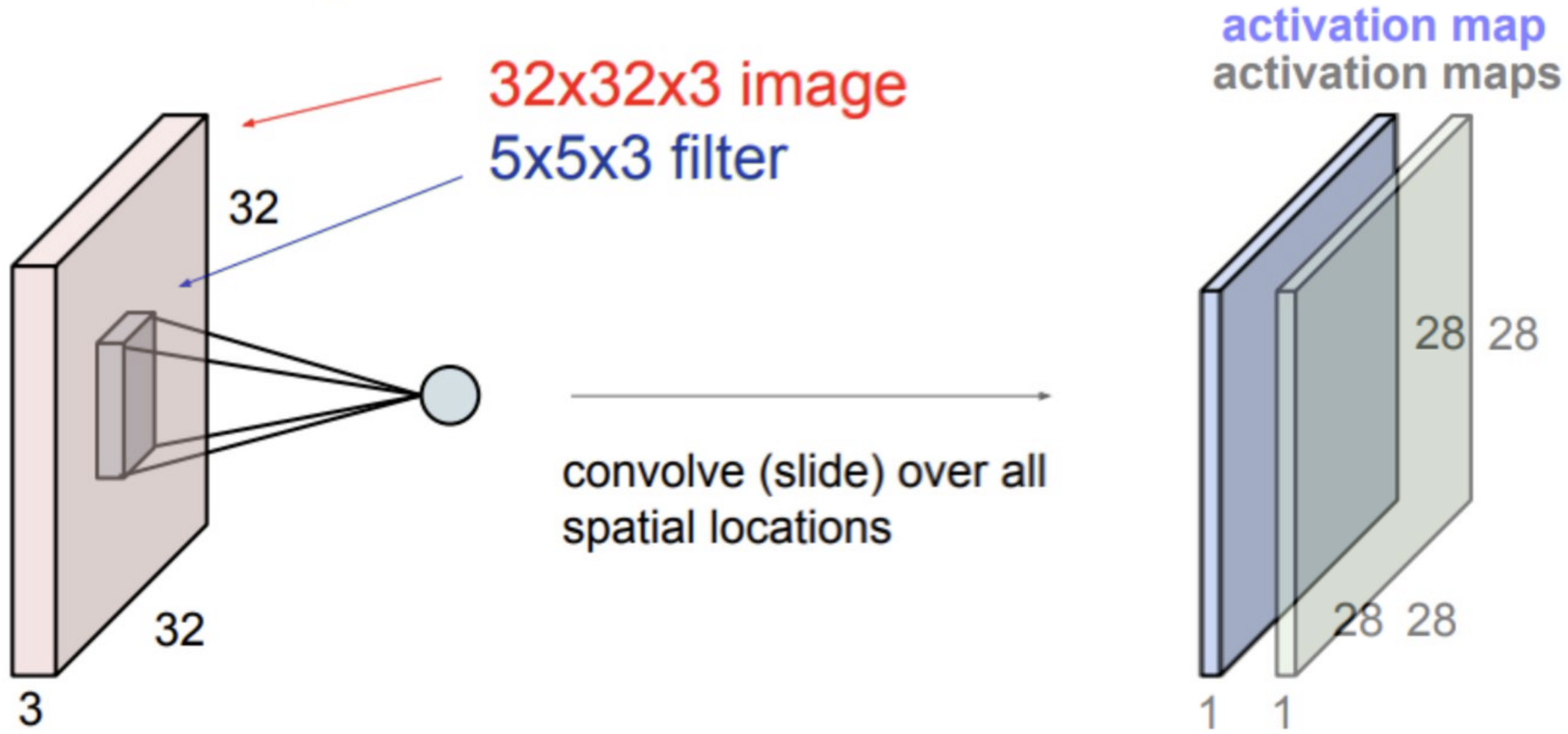
Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

Q. Convolution을 하려면 filter를 뒤집어야하지 않는가?

개념적인 정의만 사용, 엄밀한 연산과정은 아닌.

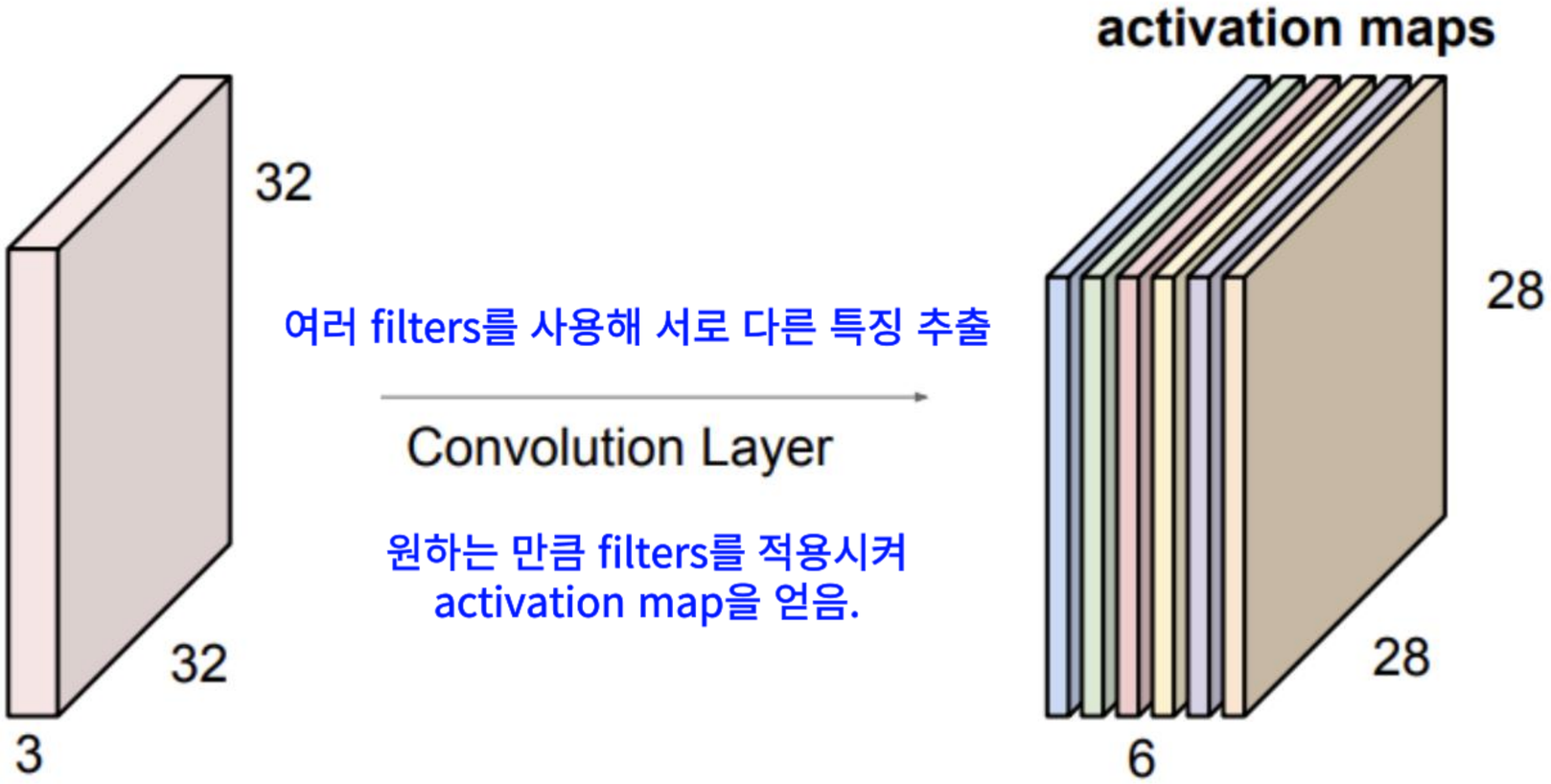
Convolution Layer

consider a second, green filter



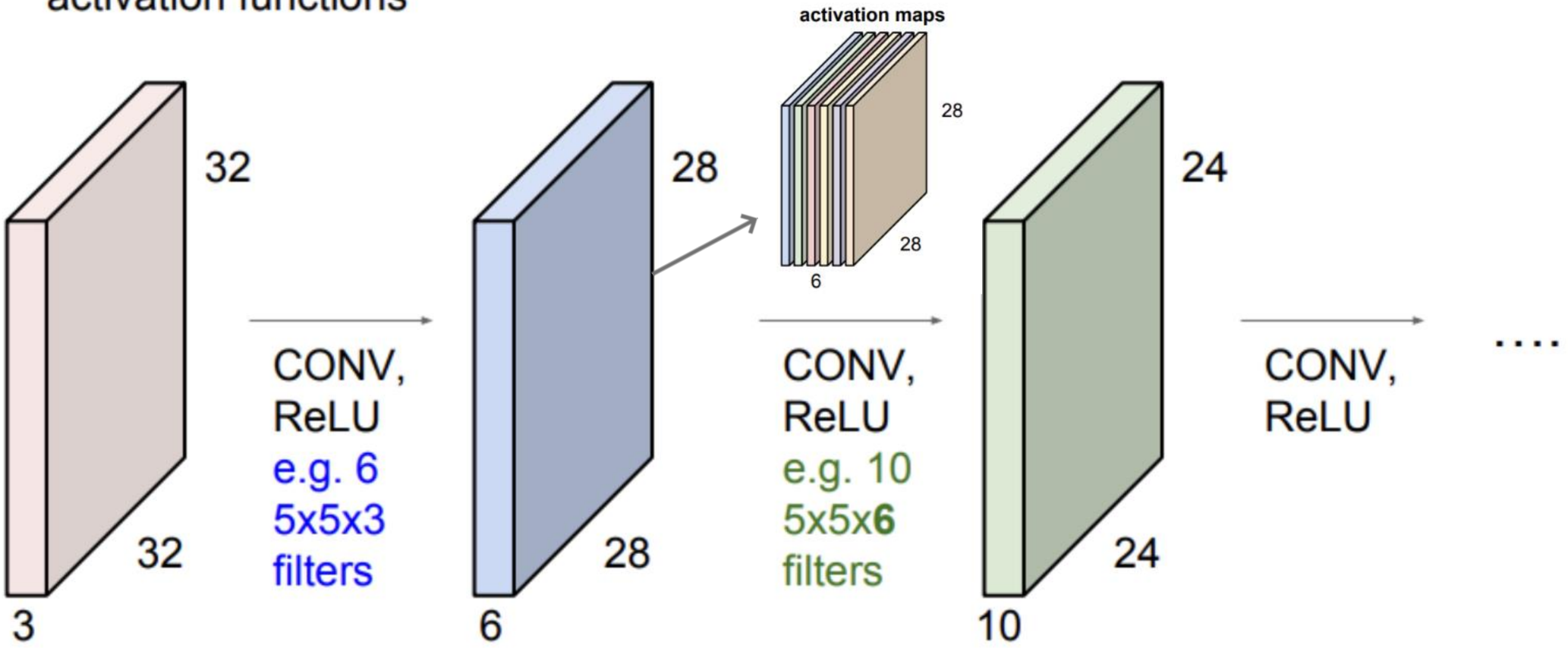
Q. 이미지의 가장자리는 filter처리가 덜 되는 것이 아닌가? **zero-padding**

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!

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



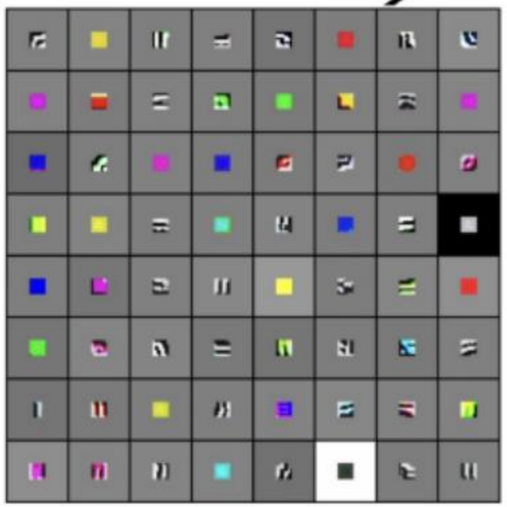
activation function을 적용, 이전의 출력이 다음의 입력이 된다.

Preview

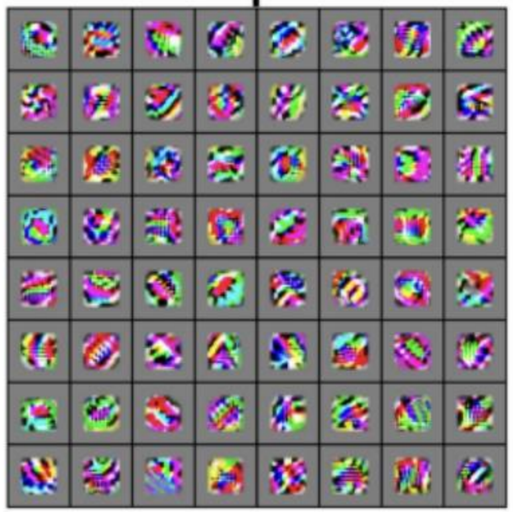
[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].

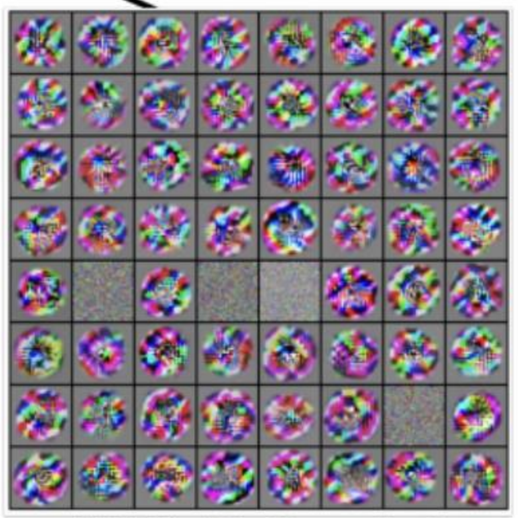
Low-level features → Mid-level features → High-level features → Linearly separable classifier



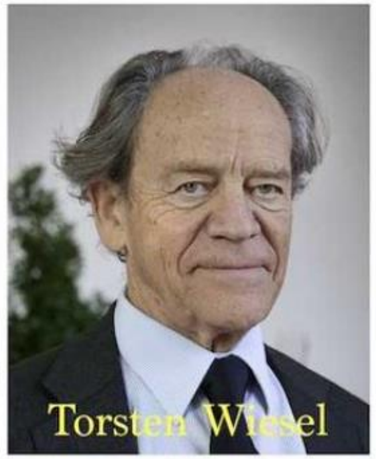
VGG-16 Conv1_1



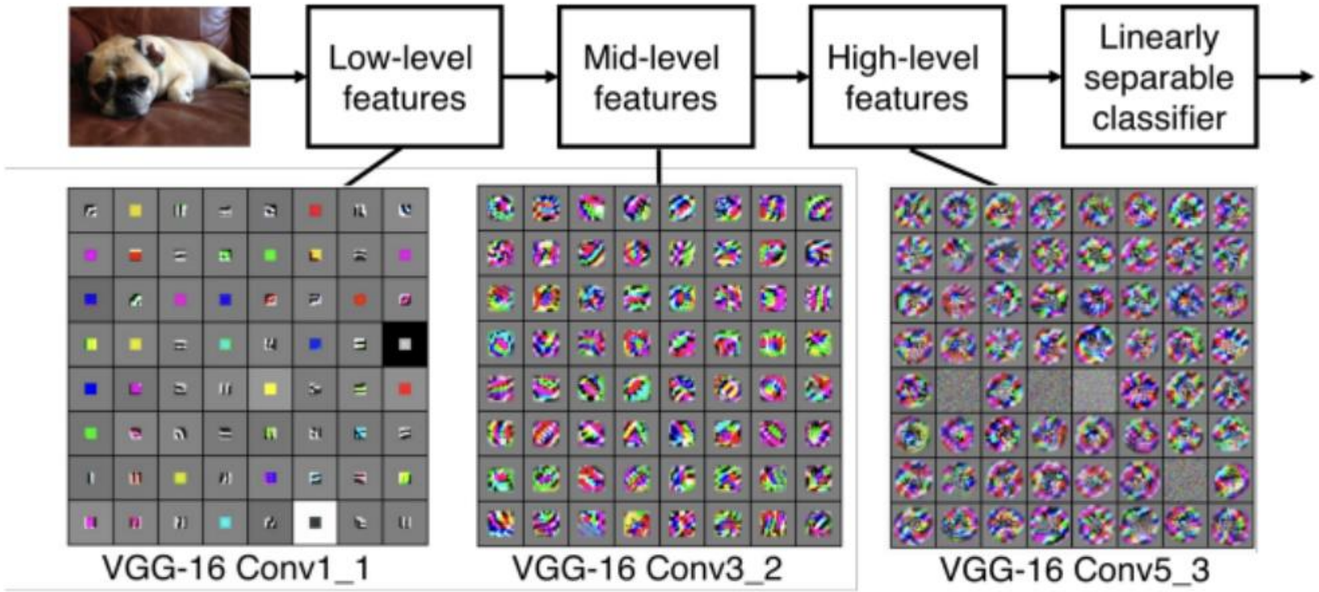
VGG-16 Conv3_2

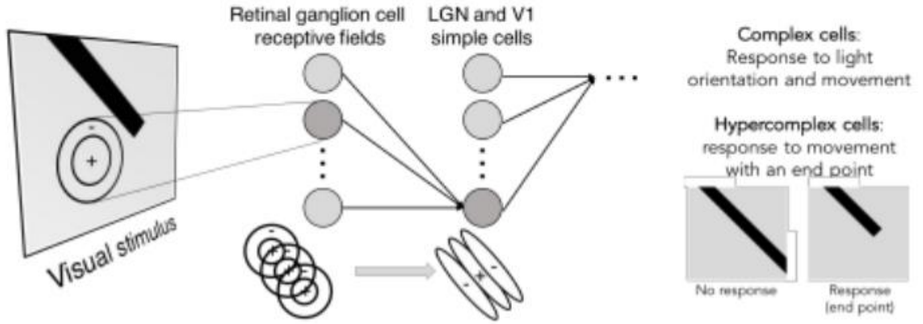


VGG-16 Conv5_3

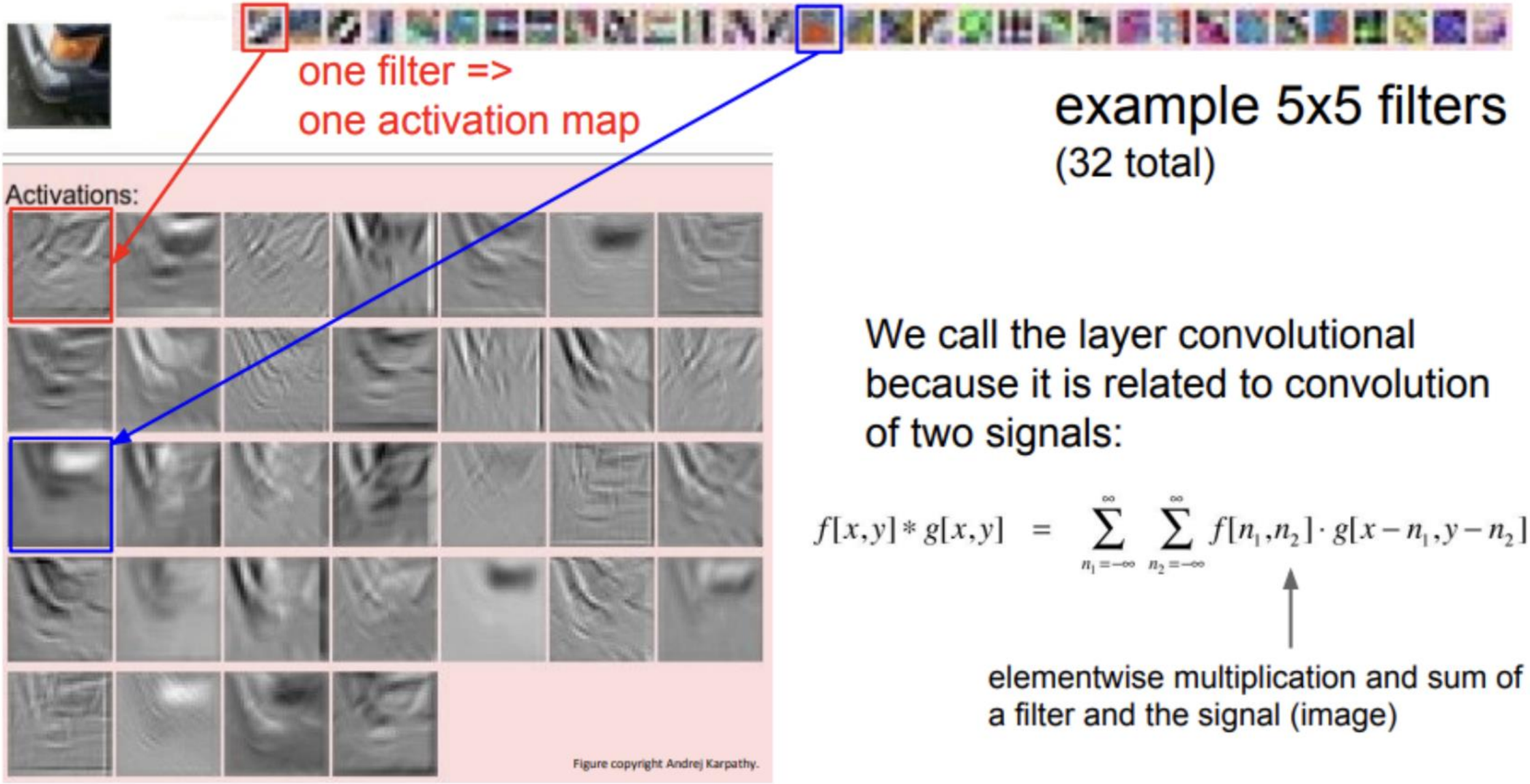


Preview





Q. 저 사진에서 시각화한 것이 무엇입니까? **뉴런의 활성을 최대화시키는 이미지 형태.**



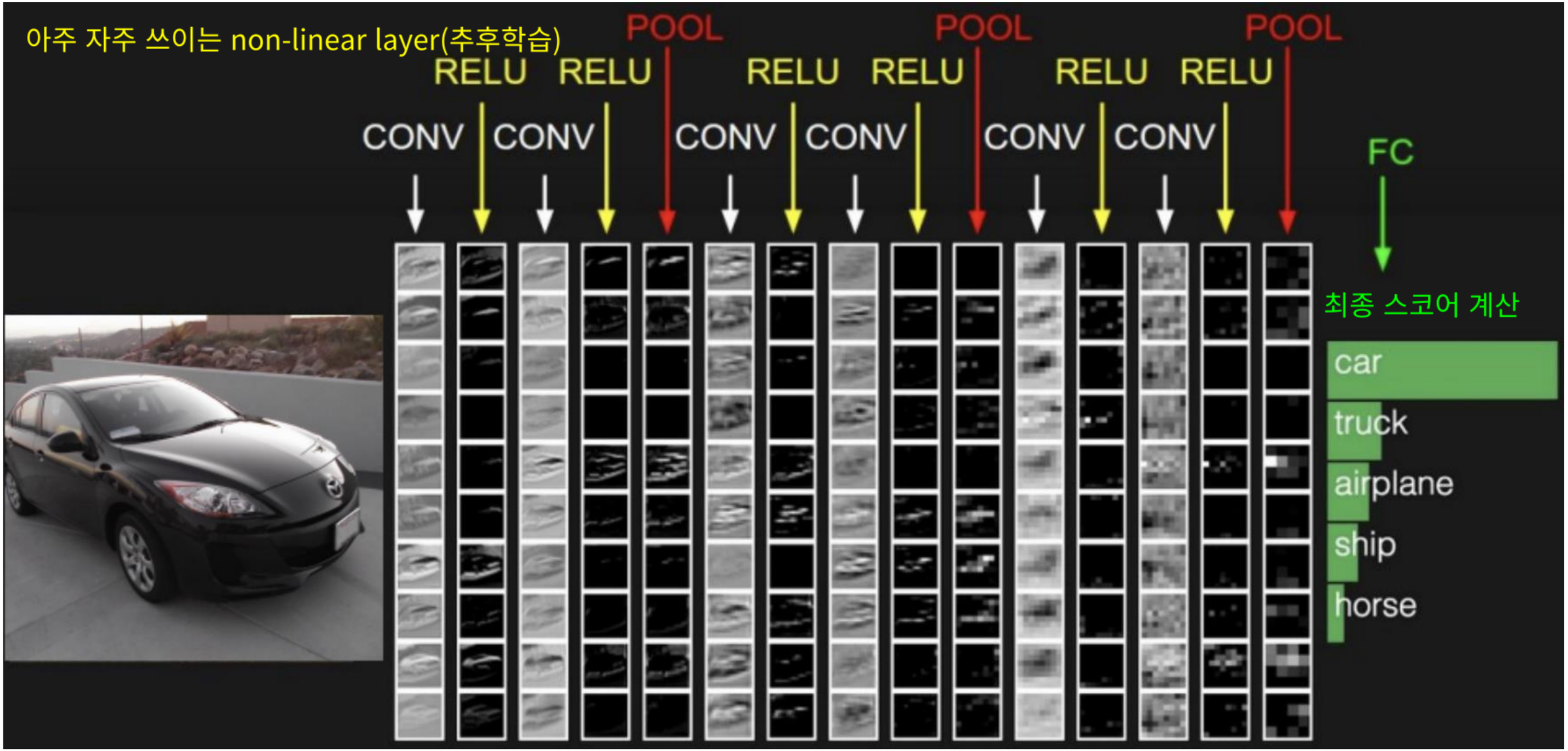
example 5x5 filters
(32 total)

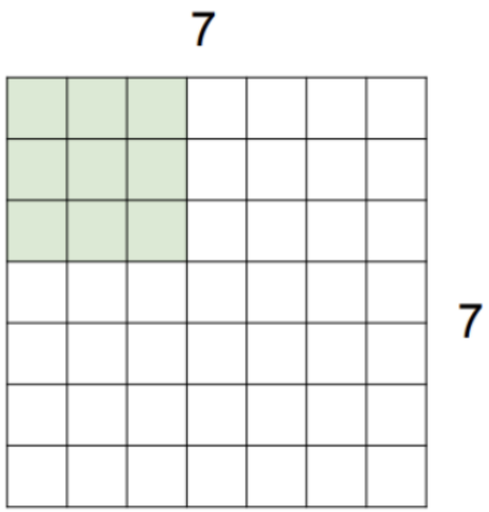
We call the layer convolutional because it is related to convolution of two signals:

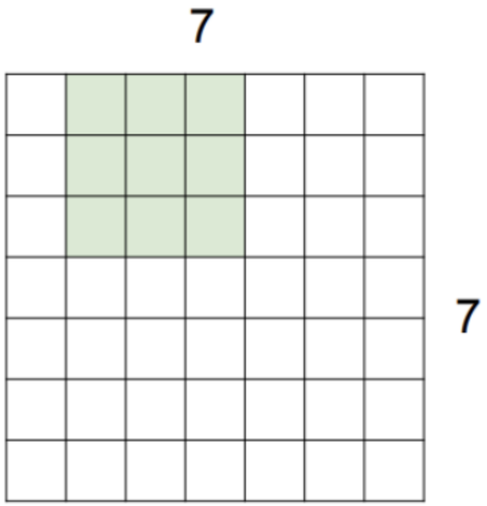
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

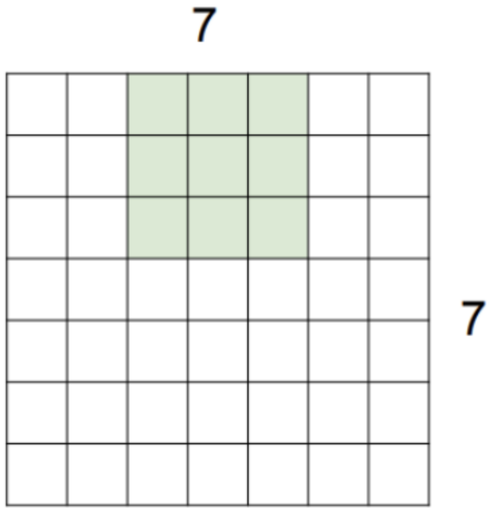
elementwise multiplication and sum of a filter and the signal (image)

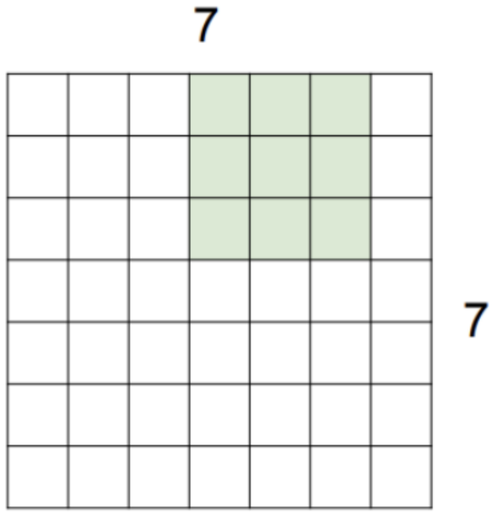
activation의 사이즈를 줄여주는 과정

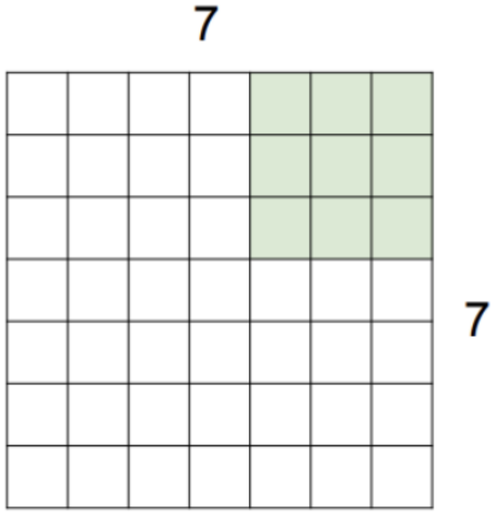



7
7

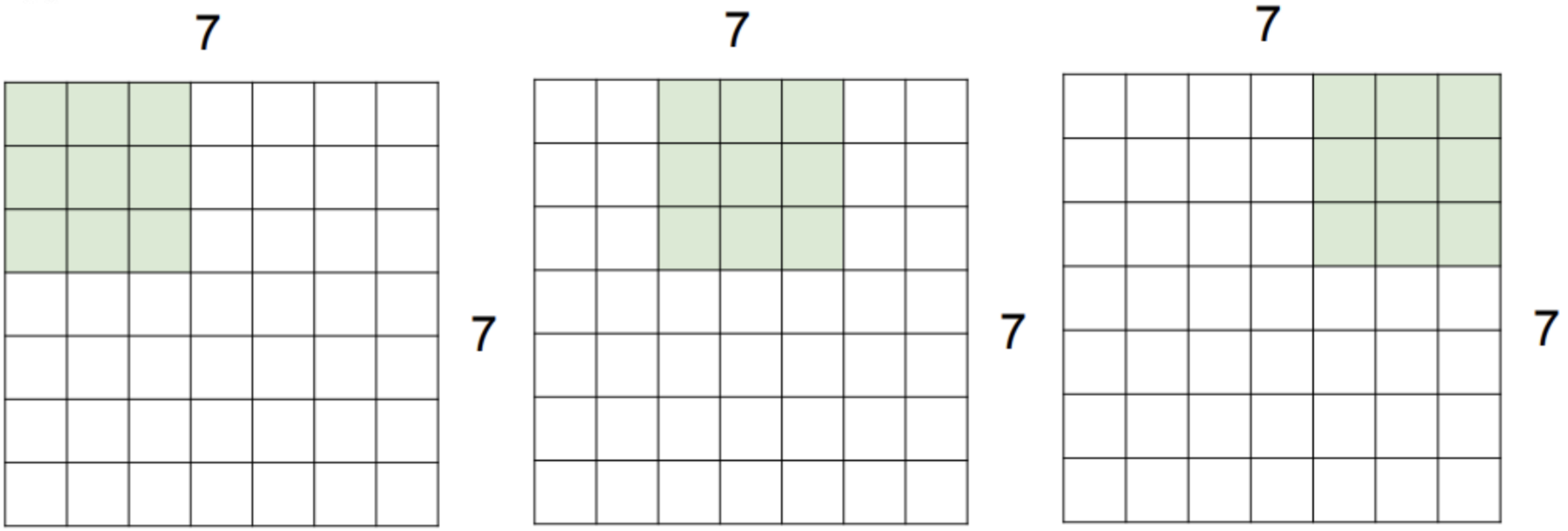

7
7


7
7

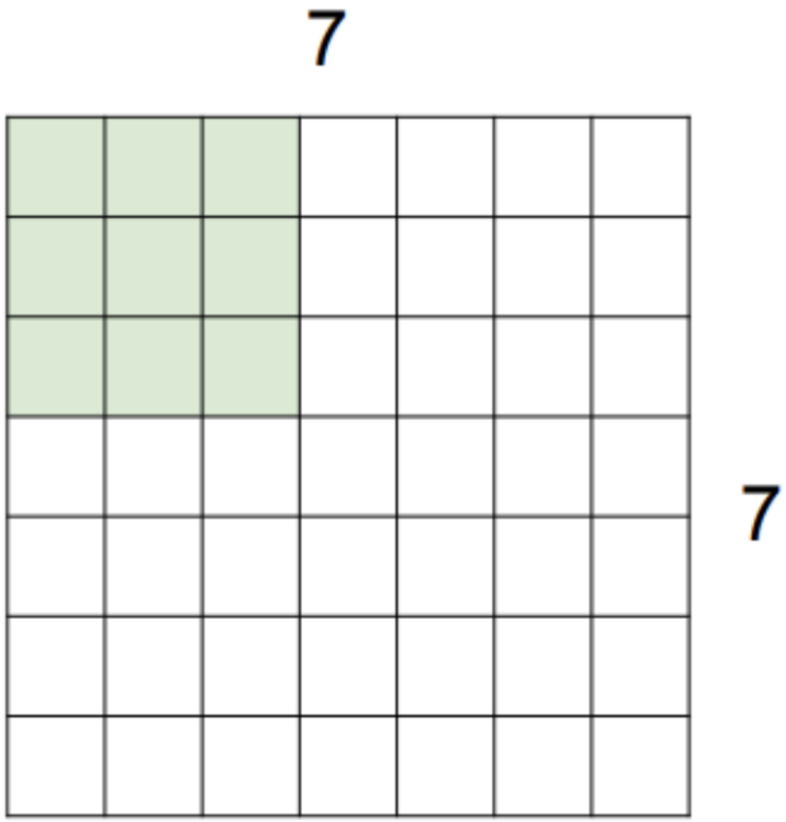

7
7

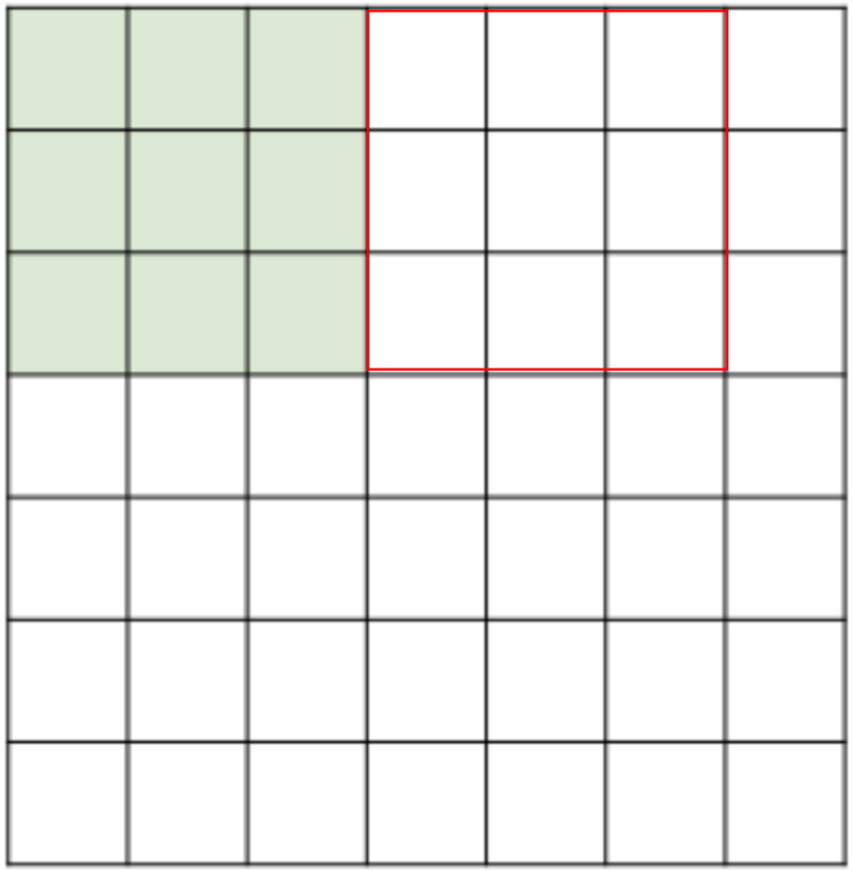

7
7

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

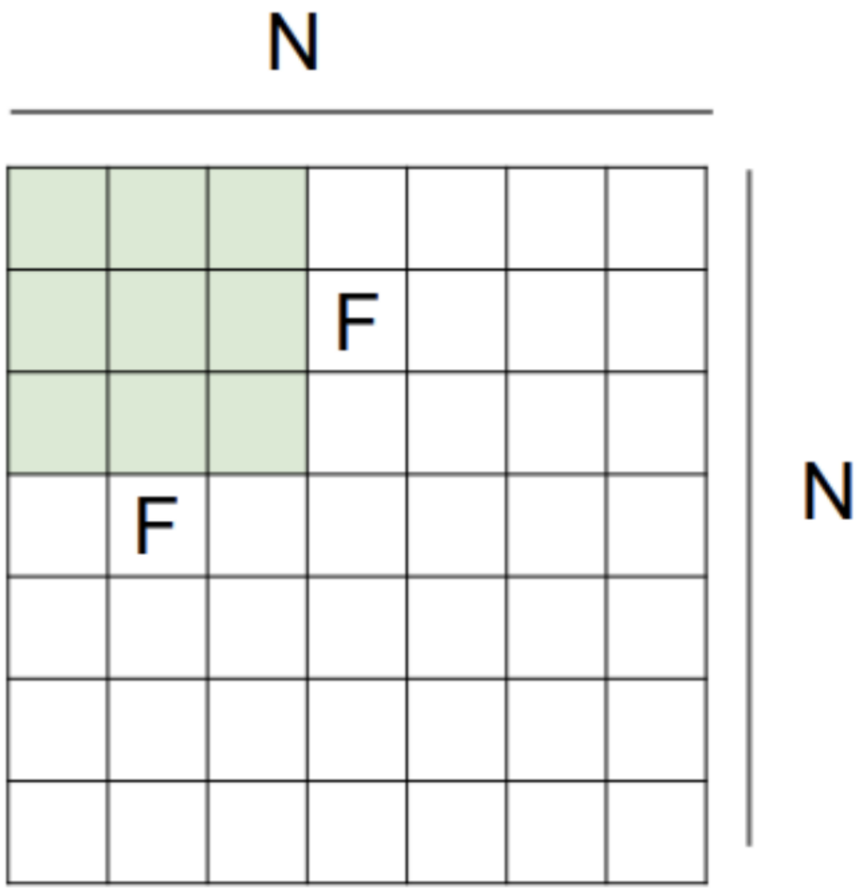


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3**





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



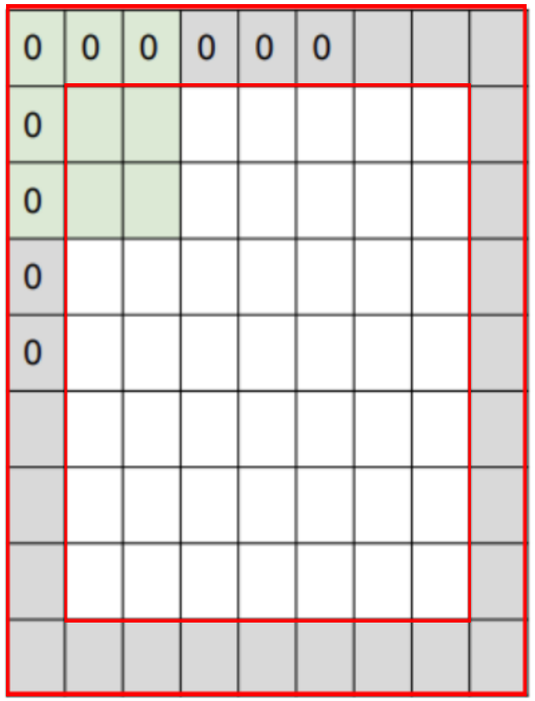
Output size:
(N - F) / stride + 1

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 :\

Zero Padding

N=9

In practice: Common to zero pad the border



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

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Q. 가장자리에 필요없는 특징이 부여되는 것은 아닌가?

모서리값을 구하는 하나의 방법, 잘 동작하는편.

Questions

Q. 정사각형이 아닌 직사각형 행렬이면 stride를 다르게 설정해야하는가?

가능. 하지만 보통 정사각형에 같은 stride를 적용. 종횡비 유지와 관련된 좋은 질문.

Q. zero-padding을 하는 이유는 무엇인가?

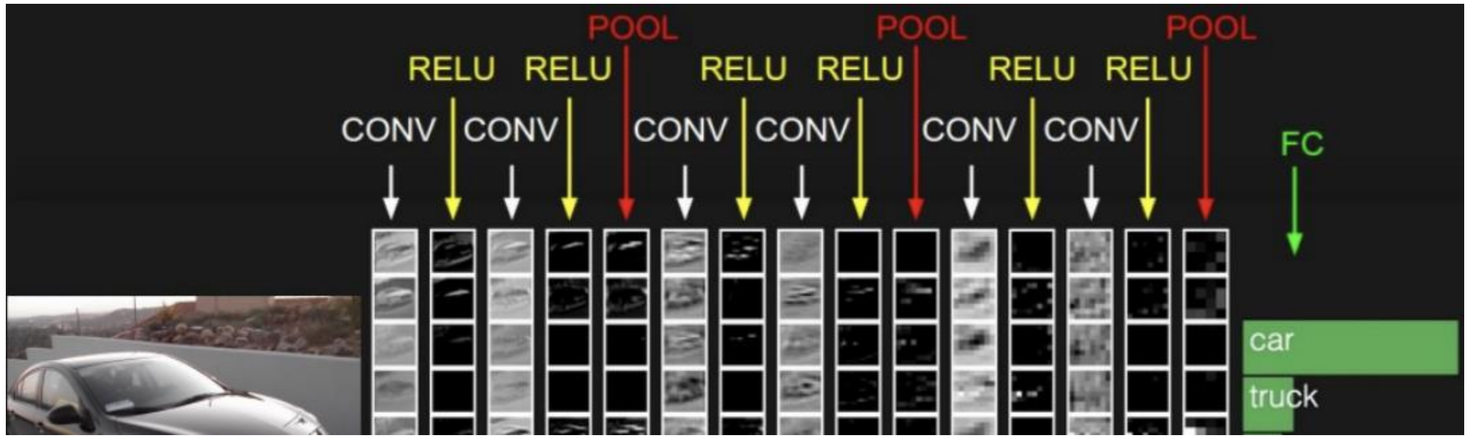
Layer를 거쳐도 입력의 size를 유지하기 위함.

Questions

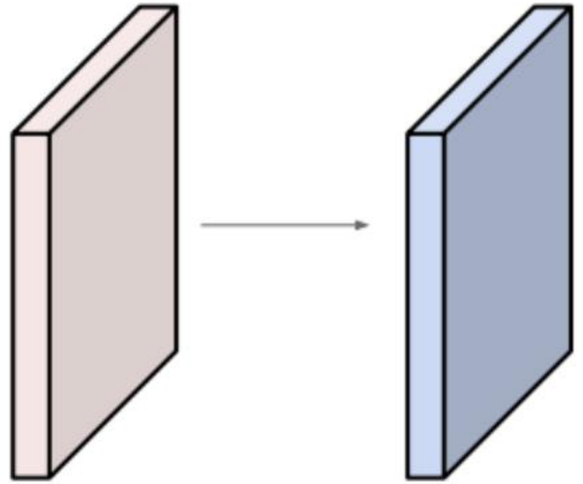
Q. stride 크기를 정하는데 필요한 직관은 무엇인가?

- stride를 크게 가져갈 수록 출력값이 점점 작아짐. (down sampling)

- activation map의 size가 model 전체의 parameter 개수에도 영향.
  FC Layer는 마지막 단계에서 ConvNet의 출력과 한 번씩 대치.



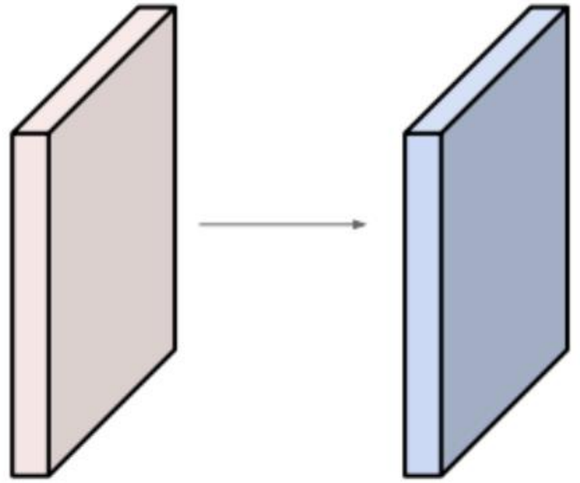
Examples time:



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

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

Examples time:



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

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

Example: CONV layer in Torch

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

SpatialConvolution

```
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])
```

Applies a 2D convolution over an input image composed of several input planes. The `input` tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane` : The number of expected input planes in the image given into `forward()` .
- `nOutputPlane` : The number of output planes the convolution layer will produce.
- `kW` : The kernel width of the convolution
- `kH` : The kernel height of the convolution
- `dW` : The step of the convolution in the width dimension. Default is `1` .
- `dH` : The step of the convolution in the height dimension. Default is `1` .
- `padW` : The additional zeros added per width to the input planes. Default is `0` , a good number is `(kW-1)/2` .
- `padH` : The additional zeros added per height to the input planes. Default is `padW` , a good number is `(kH-1)/2` .

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

If the input image is a 3D tensor `nInputPlane x height x width` , the output image size will be `nOutputPlane x oheight x owidth` where

```
owidth  = floor((width  + 2*padW - kW) / dW + 1)
oheight = floor((height + 2*padH - kH) / dH + 1)
```

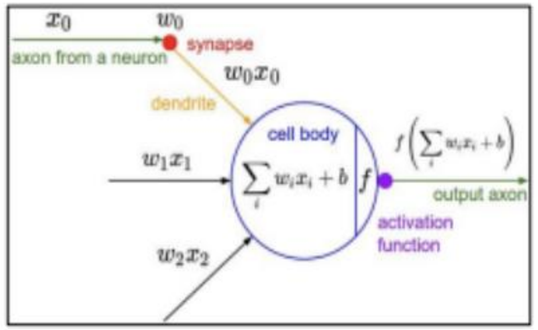

Example: CONV layer in Caffe

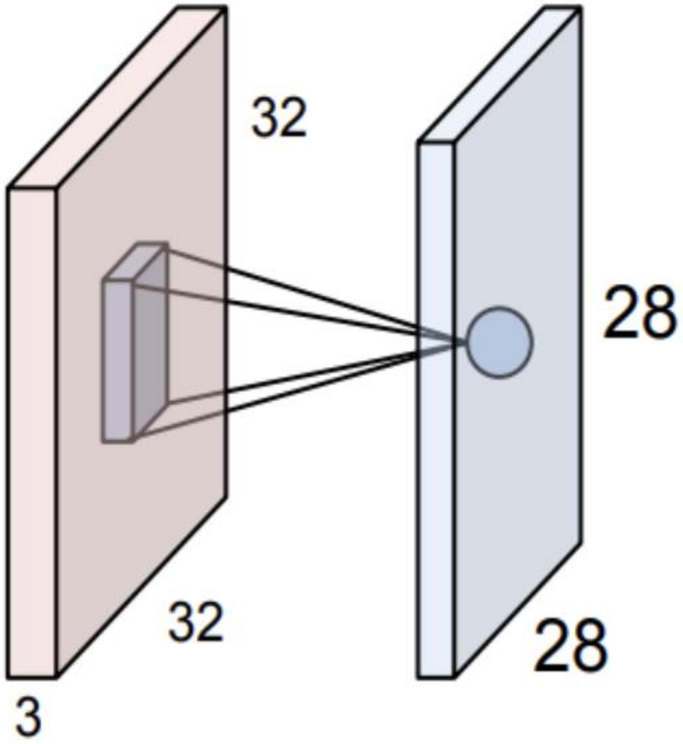
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  # learning rate and decay multipliers for the filters
  param { lr_mult: 1 decay_mult: 1 }
  # learning rate and decay multipliers for the biases
  param { lr_mult: 2 decay_mult: 0 }
  convolution_param {
    num_output: 96     # learn 96 filters
    kernel_size: 11    # each filter is 11x11
    stride: 4          # step 4 pixels between each filter application
    weight_filler {
      type: "gaussian" # initialize the filters from a Gaussian
      std: 0.01        # distribution with stdev 0.01 (default mean: 0)
    }
    bias_filler {
      type: "constant" # initialize the biases to zero (0)
      value: 0
    }
  }
}
```

The brain/neuron view of CONV Layer





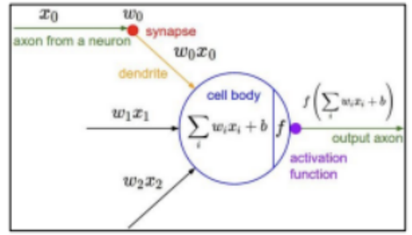
An activation map is a 28x28 sheet of neuron outputs:

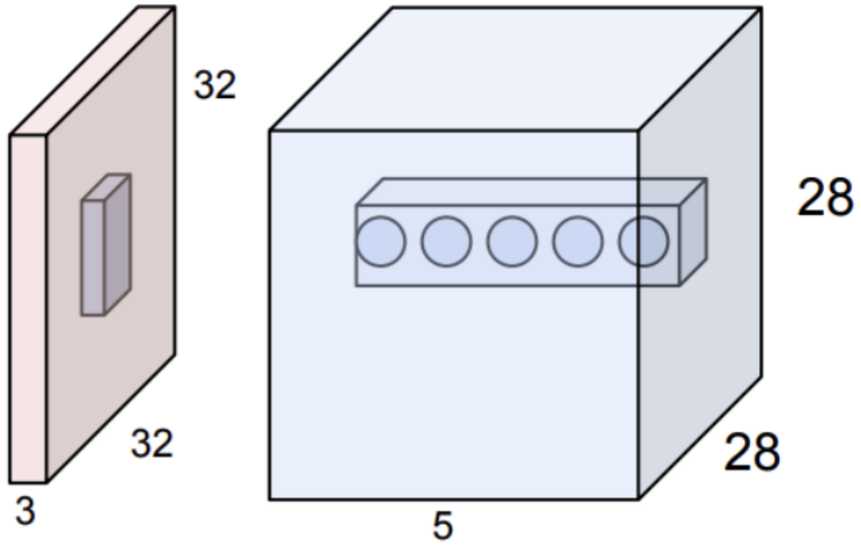
1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

한 neuron이 한 번에 수용할 수 있는 영역

The brain/neuron view of CONV Layer

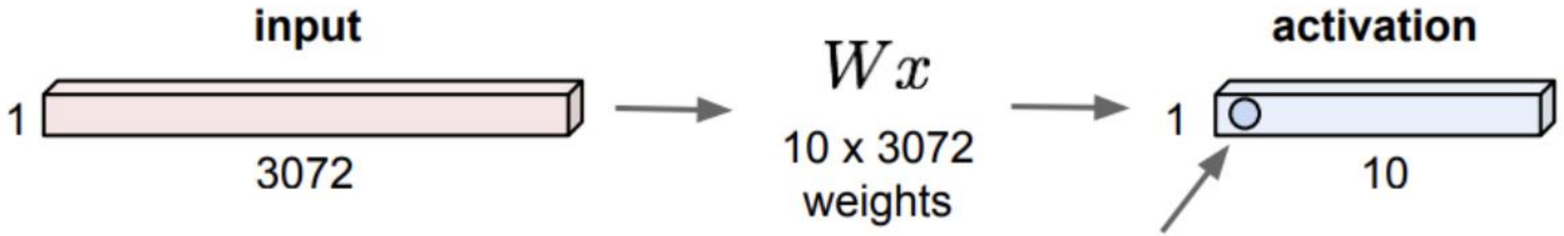




E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

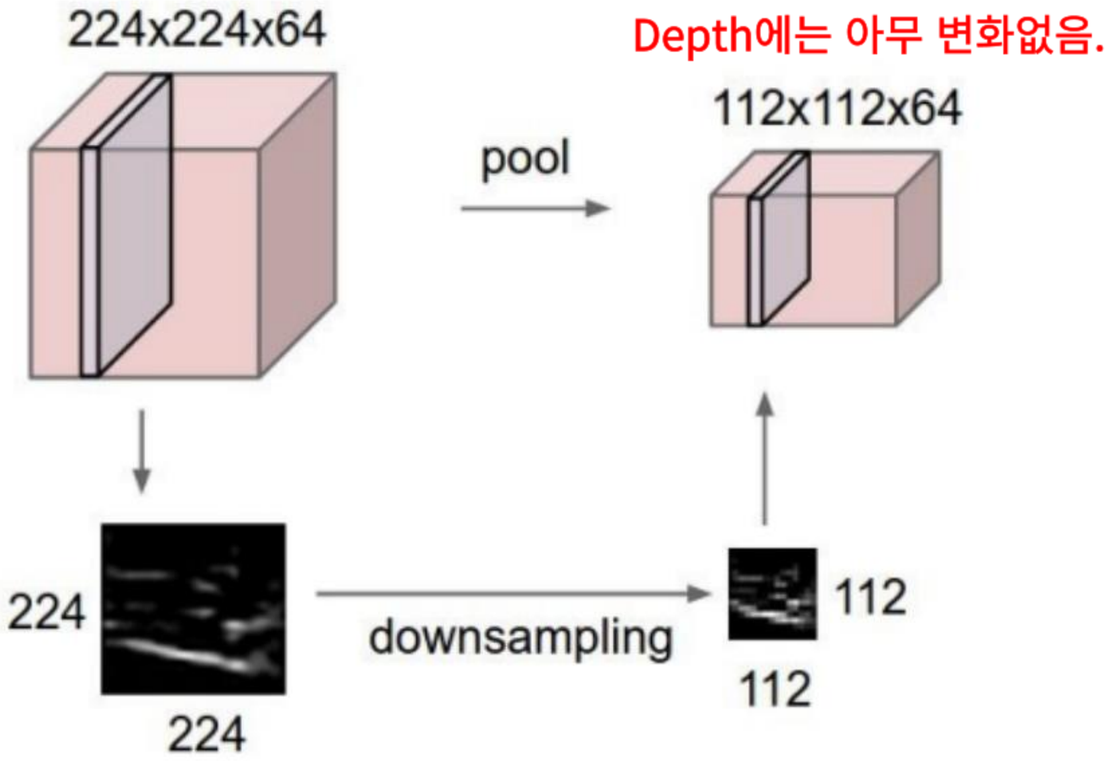
한 point에서 서로 다른 특징들을 추출.



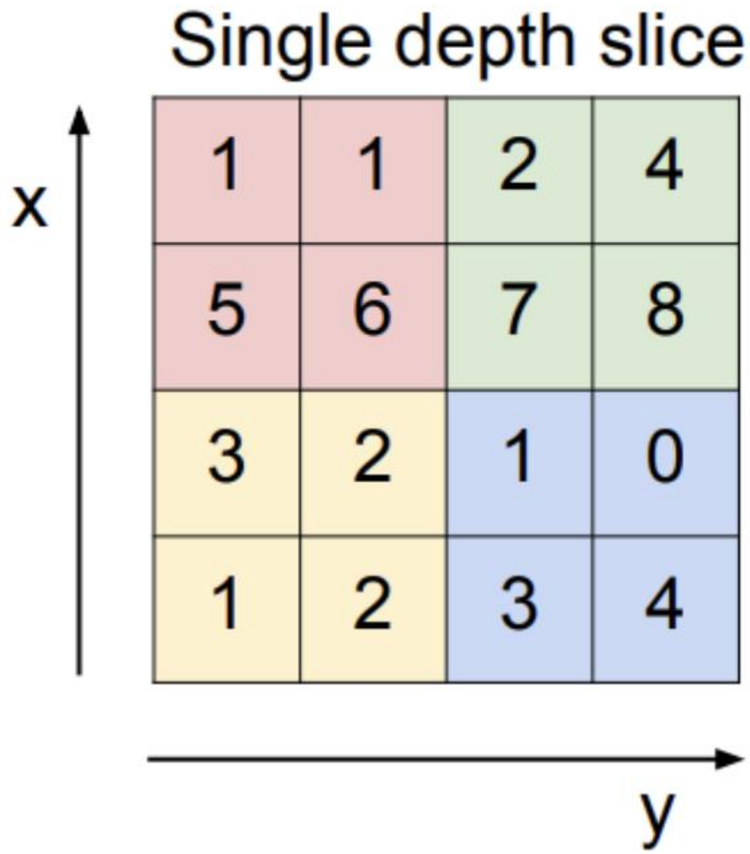
Pooling layer

Representation들을 작게 만들어 관리를 용이하게 만듬.

- makes the representations smaller and more manageable
- operates over each activation map independently:



MAX POOLING



max pool with 2x2 filters
and stride 2

6	8
3	4

Pooling은 downsampling이 목적이고 어느 영역의 수치가 크게 나오는지에 주목하므로 padding을 사용하지 않음.

THANK YOU