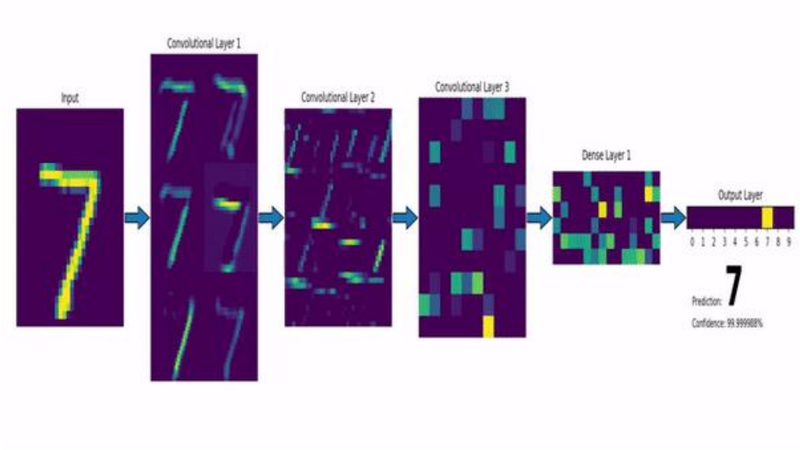
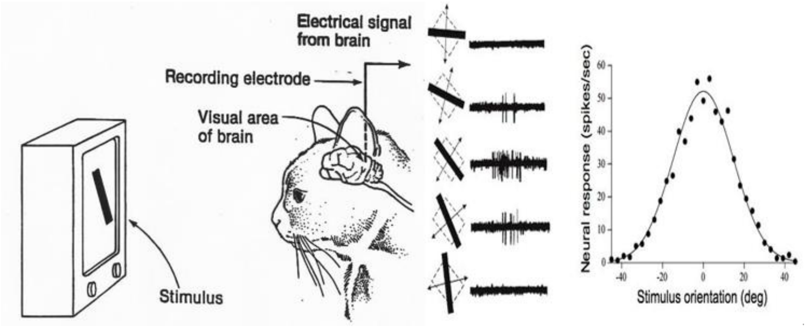
합성곱 신경망(CNN)



Convolutional Layer 1
Input
Convolutional Layer 2
Convolutional Layer 3
Dense Layer 1
Output Layer
0 1 2 3 4 5 6 7 8 9
Prediction: 7
Confidence: 99.99998%

CNN 개요

- Hubel과 Wiesel은 1958, 1959년에 고양이 시각 시스템에 이미지의 부분적 특징에 반응하는 신경세포 (Local receptive field)가 존재한다는 것을 발견하였다.
- Fukushima는1975년 동물의 시각시스템과 유사한Cognitron, 1982년 Neocognitron 네트워크를 제안하였다. Cognitron과 Neocognitron (New-cognitron)은 인간의 학습 방법과 인지 능력을 모방하여 기계를 학습시키는 방식이다. 기계 학습 실험을 통해 인간 두뇌의 기능을 더 깊게 이해할 수 있게 되었다.
- Yann LeCun은1998년에 여러개의 convolutional network를 가진 LeNet-5를 만들어 손으로 쓴 숫자들을 인식하는 시스템을 제안하였고, 이후 CNN으로 발전하게 되었다. CNN은 인간의 시각 신경세포가 시각 정보를 인식하는 방식과 유사하게 기계가 이미지를 인식할 수 있는 인공신경망이다.

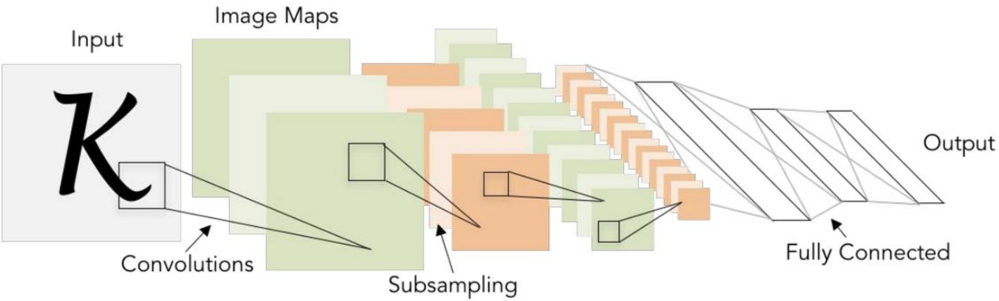


CNN (Convolutional Neural Network)

A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

http://cs231n.stanford.edu/

CNN (Convolutional Neural Network)

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

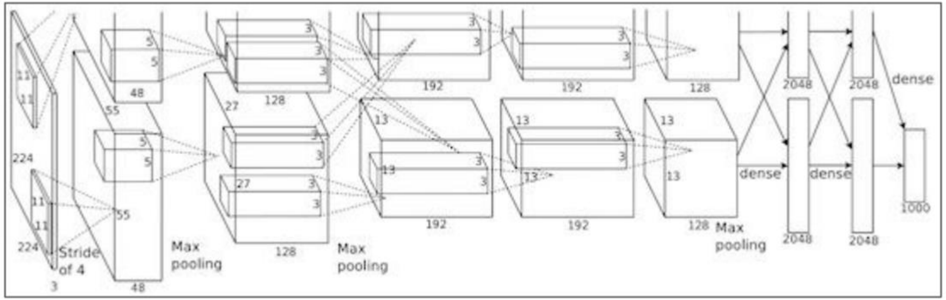


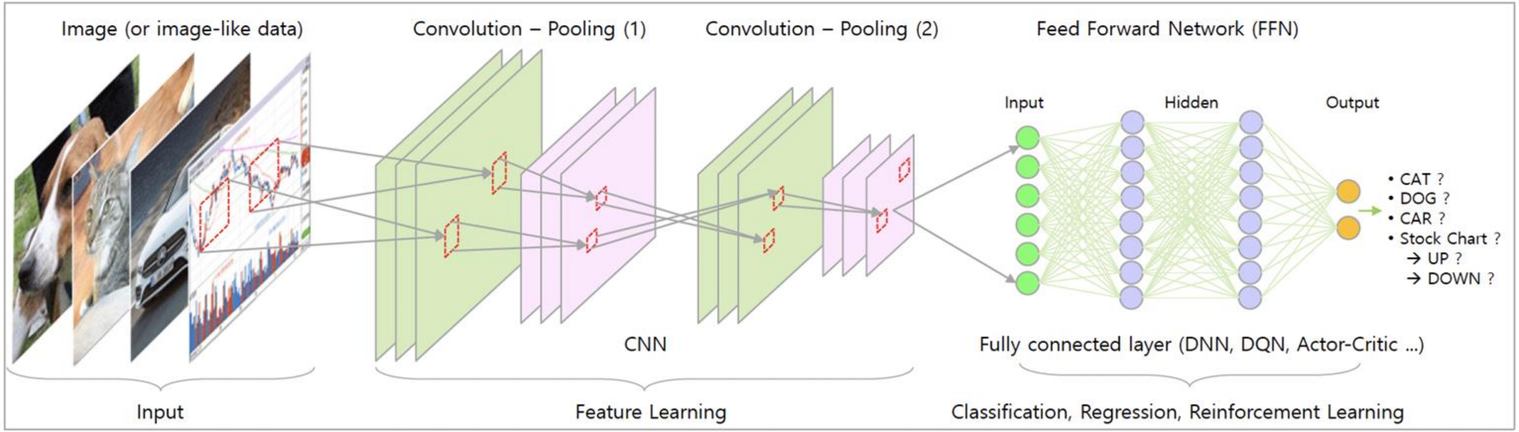
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”

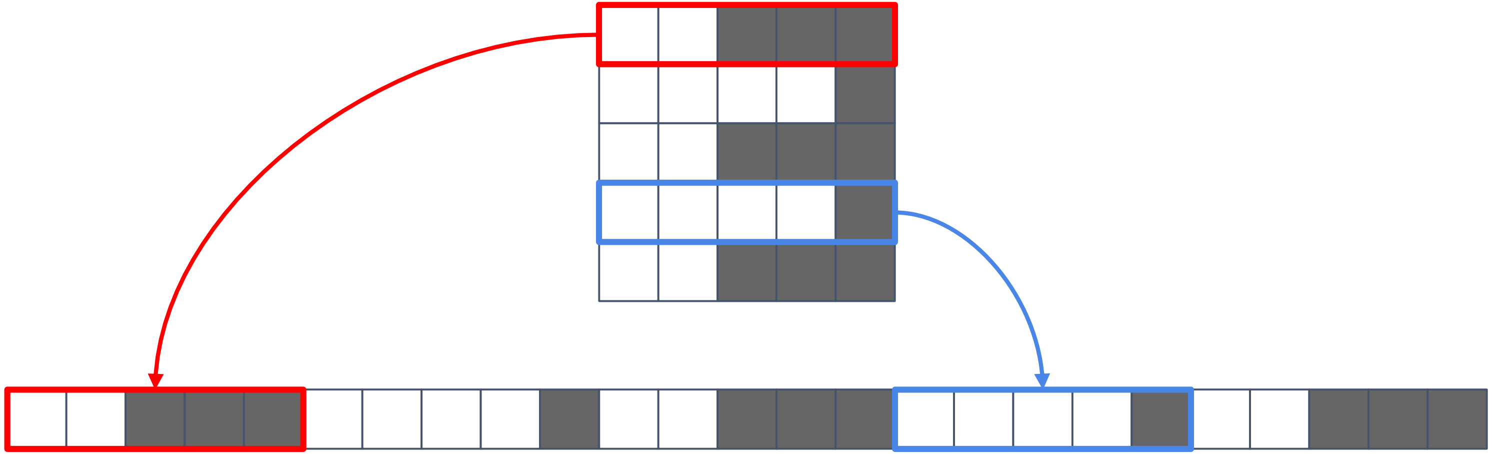
http://cs231n.stanford.edu/

CNN 개요

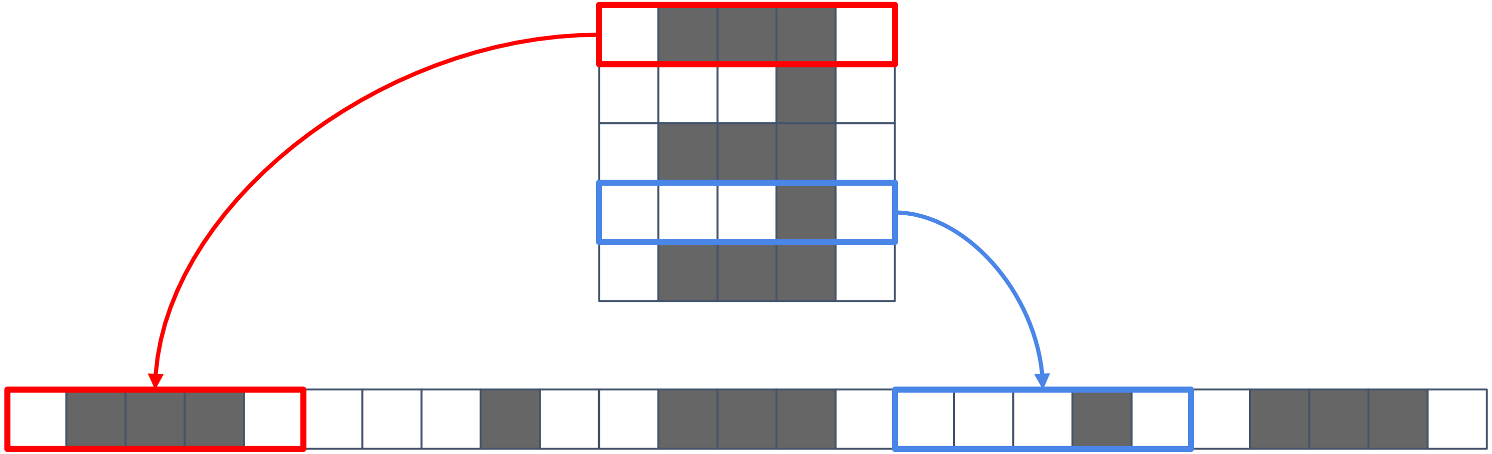
- CNN은 입력된 이미지를 여러개의 Convolution-Pooling 층을 거쳐 이미지의 부분적인 특징(feature)을 추출한다. 이 특징들은 다시 FFN로 입력되어 최종 output이 출력된다. FFN는 목적에 따라 classification 역할을 할 수도 있고, regression이나 강화학습(reinforcement learning) 네트워크 등으로 구성할 수 있다.



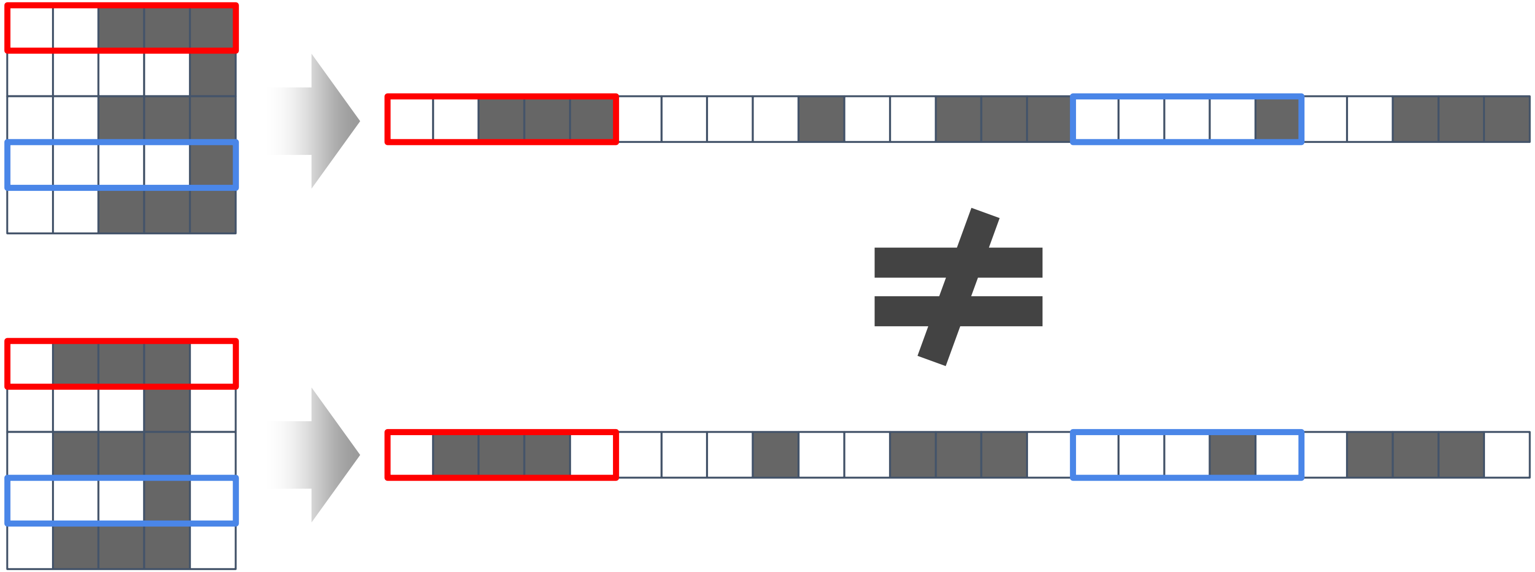
CNN (Convolutional Neural Network)



CNN (Convolutional Neural Network)

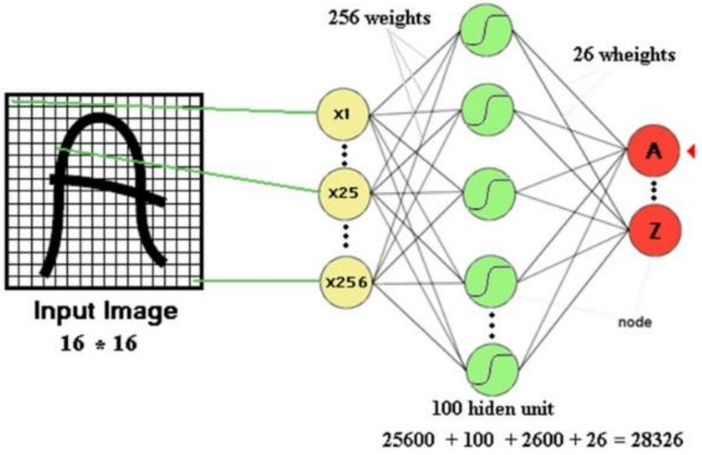


CNN (Convolutional Neural Network)



Fully connected Network의 문제점

- Image 를 fully connected Network으로 학습을 시키면 데이터의 형상이 무너진다.
- 이미지의 가까운 공간에 대한 정보를 잃게 된다.
- CNN은 이런 형상에 대한 정보를 유지시켜 준다.



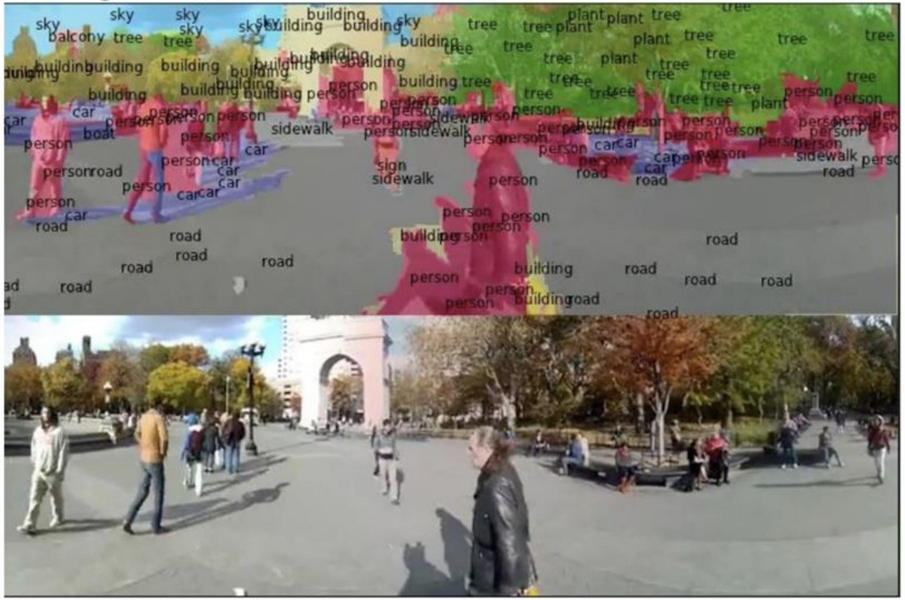
Object Detection, Segmentation

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]

http://cs231n.stanford.edu/

Self-driving car - object detection

Fast-forward to today: ConvNets are everywhere

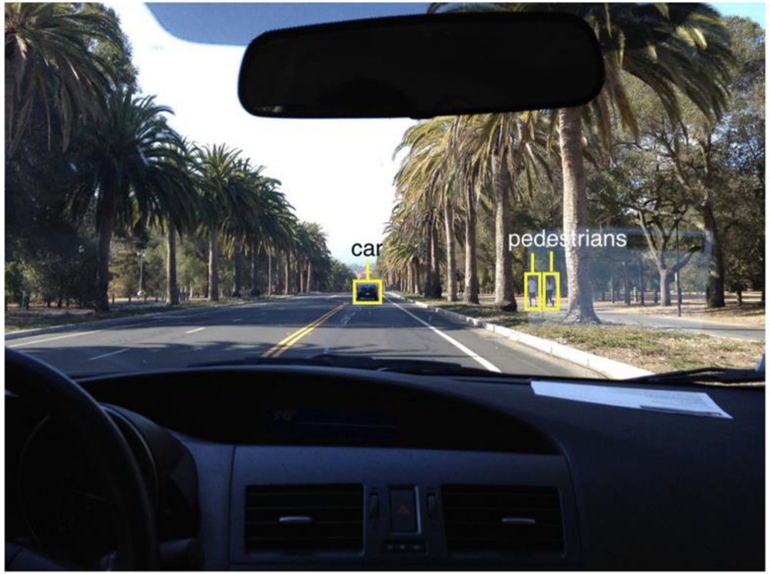


Photo by Lane McIntosh. Copyright CS231n 2017.

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.

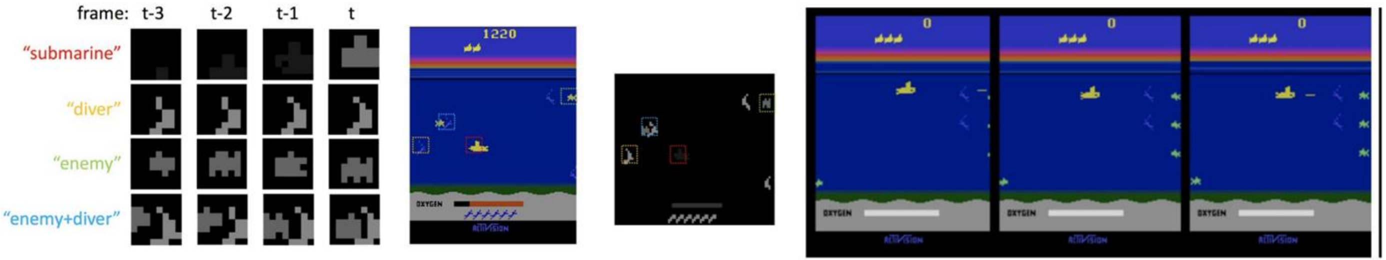
http://cs231n.stanford.edu/

Pose estimation

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.

http://cs231n.stanford.edu/

Image Captioning

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

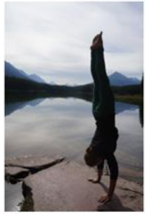
Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand



A woman standing on a beach holding a surfboard

Image Captioning

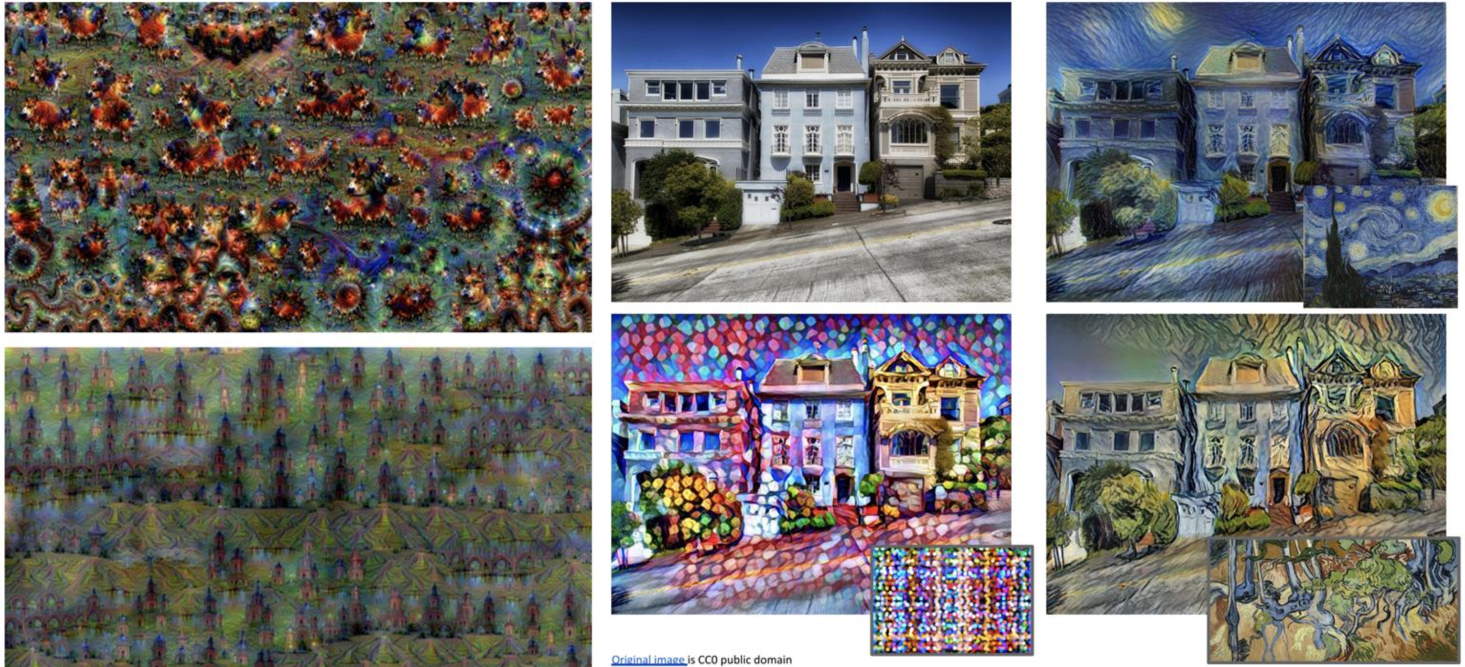
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

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

http://cs231n.stanford.edu/

Style Transfer



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

http://cs231n.stanford.edu/

Filter

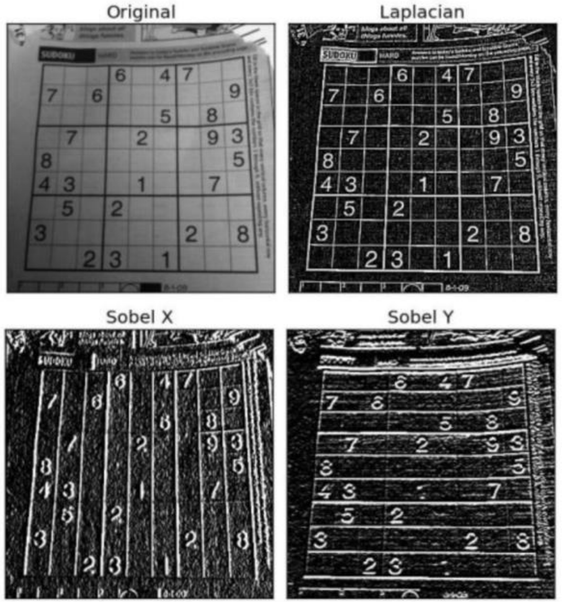
- Extract Feature

horizontal

-1	0	1
-2	0	2
-1	0	1

vertical

-1	-2	-1
0	0	0
1	2	1



https://setosa.io/ev/image-kernels/

합성곱 연산

● 필터(커널) 연산

➡ 입력 데이터에 필터를 적용해서 피쳐 맵 생성

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

입력 데이터

$\circledast$

2	0	1
0	1	2
1	0	2

필터

→

15	16
6	15

피처 맵

합성곱 연산

- 계산 과정

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix} \circledast \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & \\ & \end{bmatrix}$$

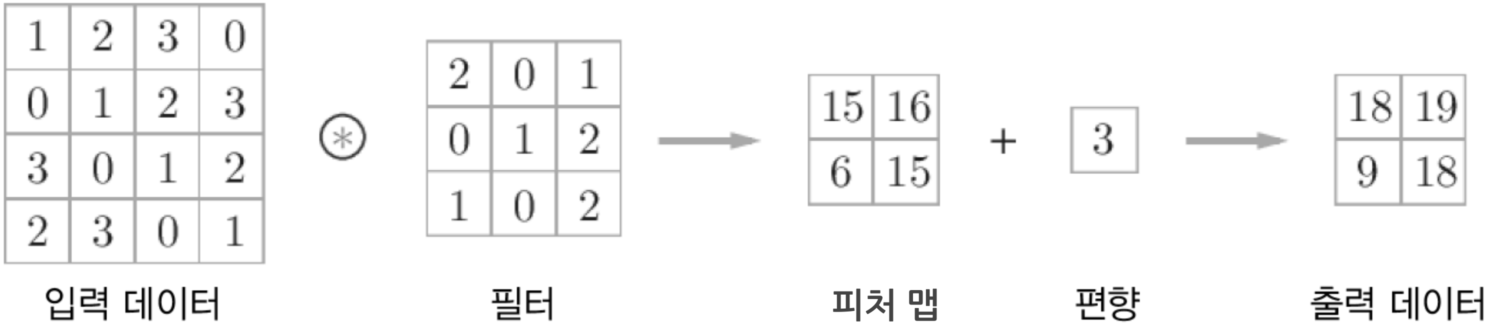
$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix} \circledast \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & 16 \\ & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix} \circledast \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & 16 \\ 6 & \end{bmatrix}$$

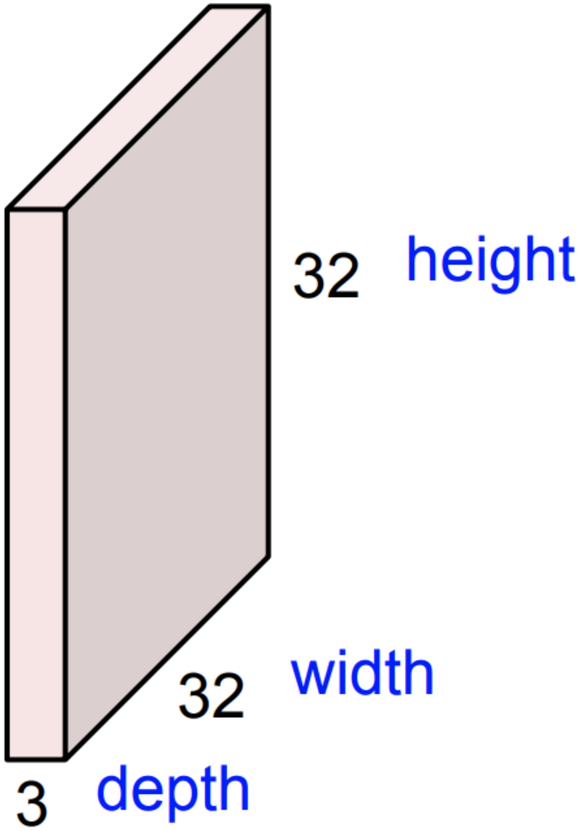
$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix} \circledast \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & 16 \\ 6 & 15 \end{bmatrix}$$

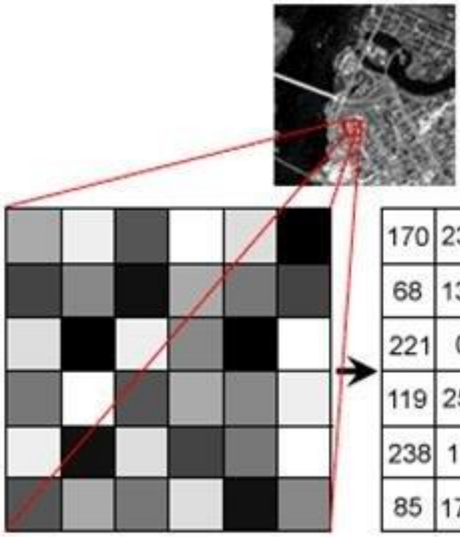
합성곱 연산

- 편향 (bias) 적용



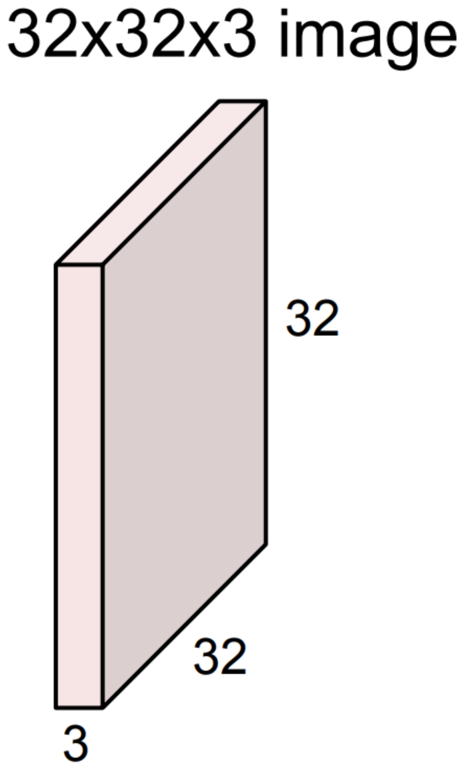
Image

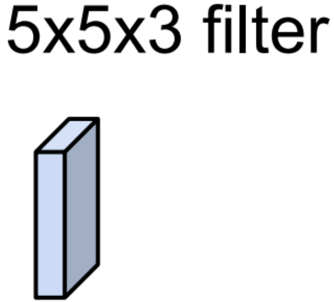




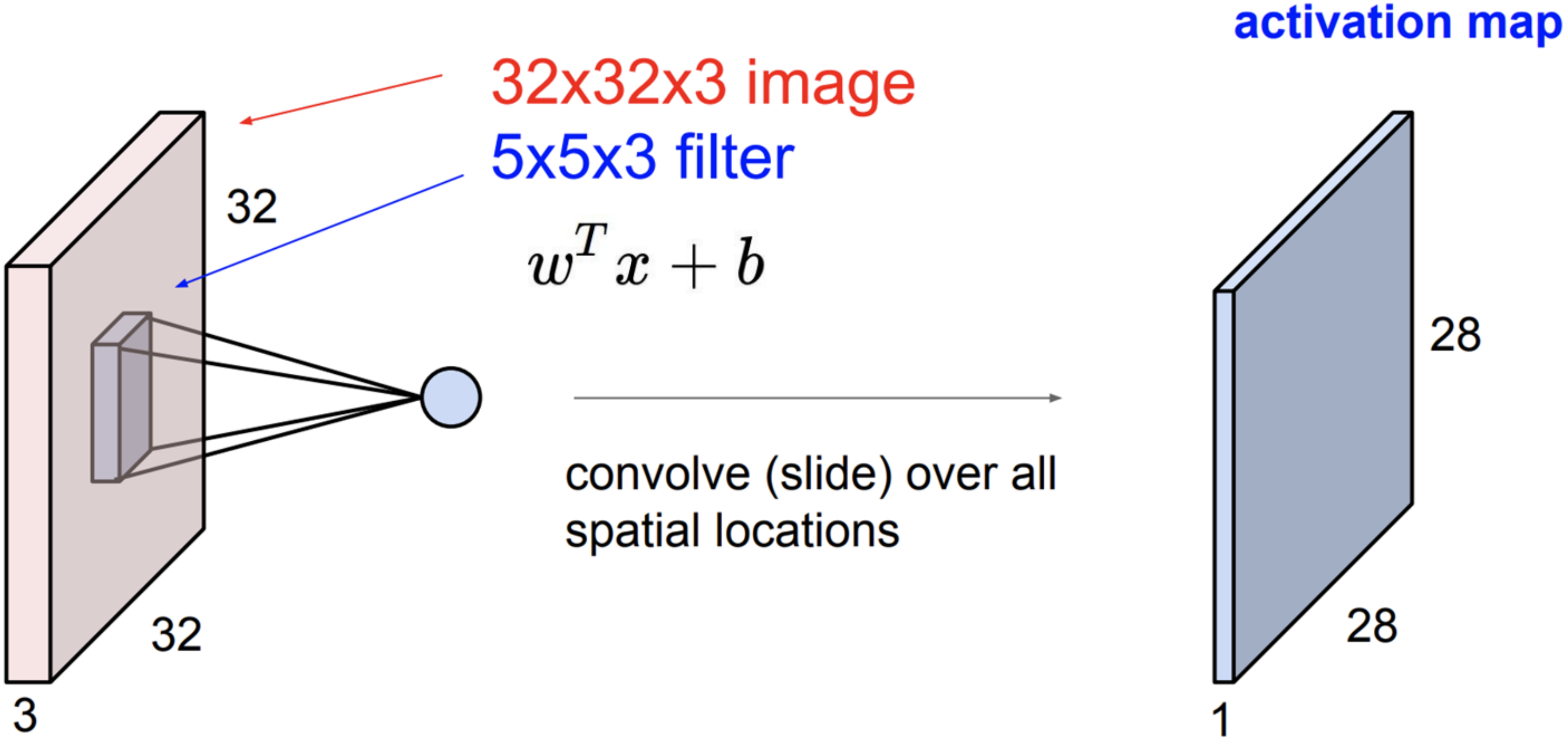
170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Filter

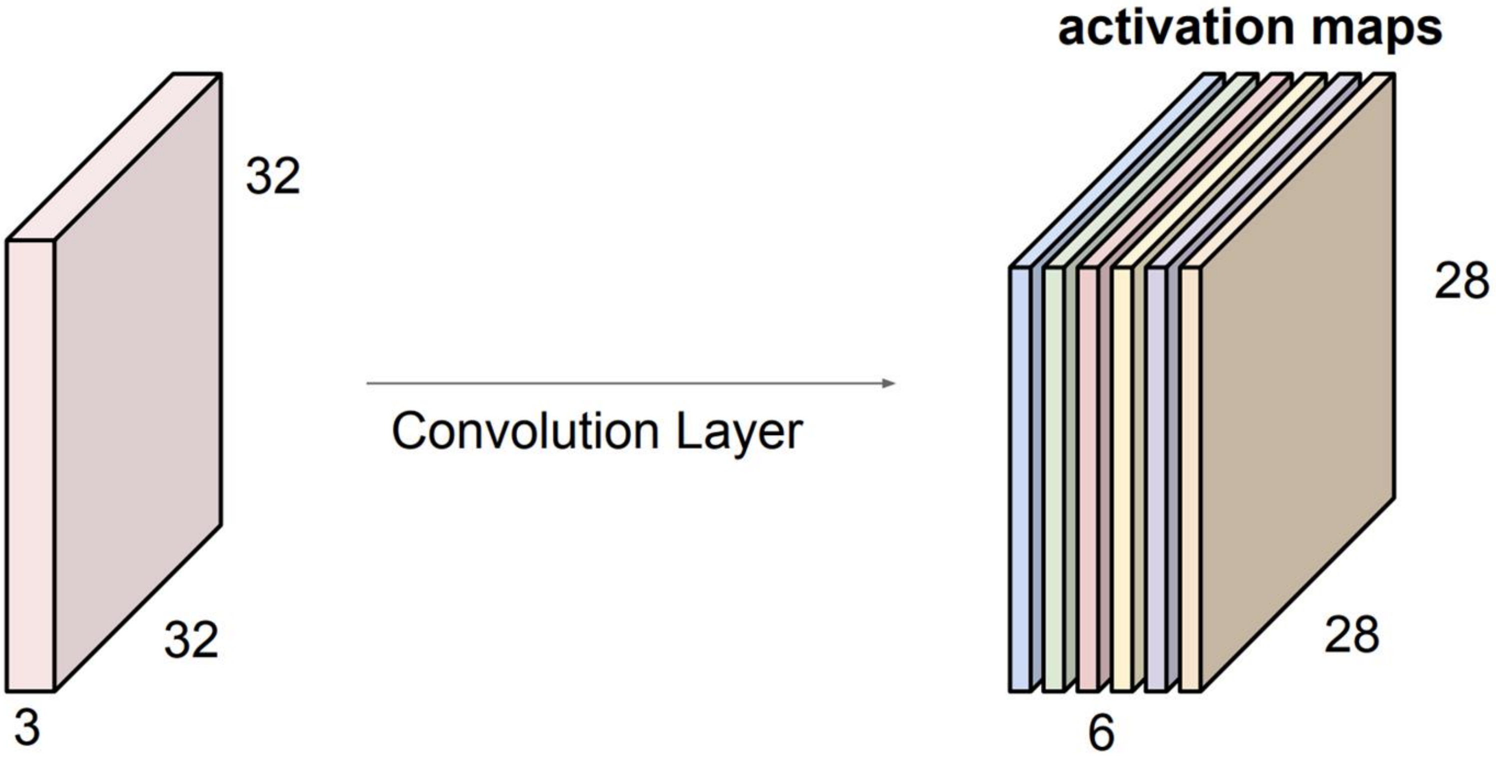




Filter


activation map
32x32x3 image
5x5x3 filter
$w^T x + b$
32
32
3
convolve (slide) over all
spatial locations
28
28
1

Filter

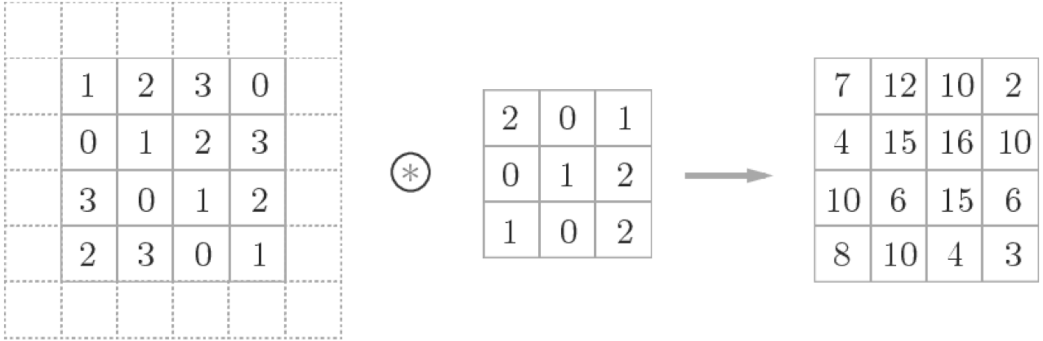


패딩(padding)

- 피처맵의 크기를 조절하기 위해서 이미지 외곽에 특정값(보통 0)으로 채우는 기법

  ➡ 필터를 계속 적용하다 보면 출력의 크기가 1일 될수 있다.

- 피처맵의 크기 조절 뿐만 아니라 입력 이미지의 외곽 데이터의 특징을 살린다



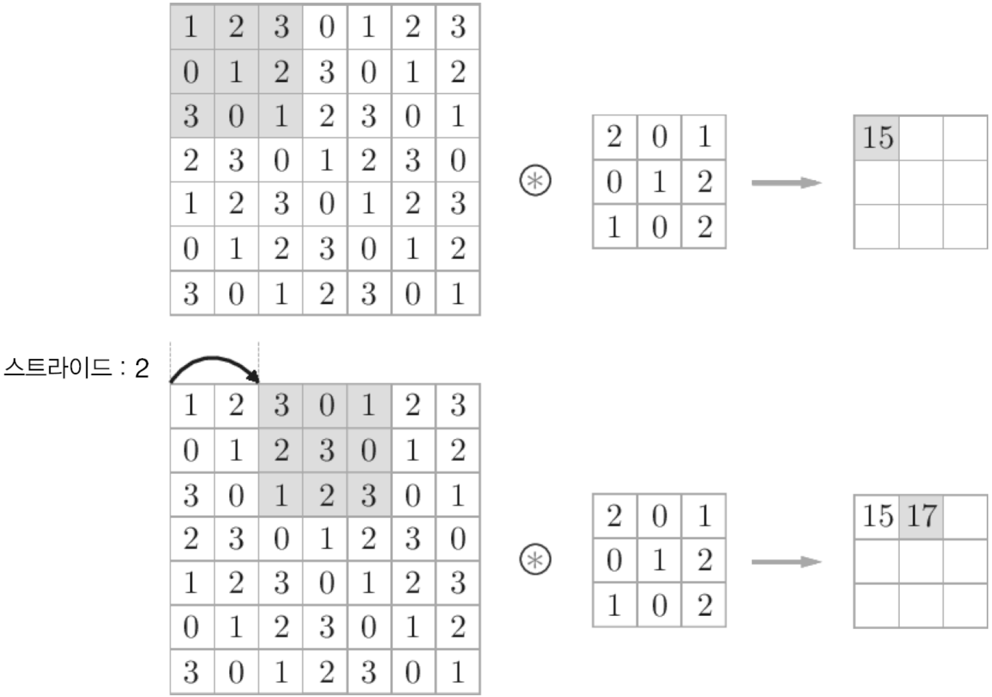
(4, 4) 입력 데이터(패딩 : 1)

(3, 3) 필터

(4, 4) 출력 데이터

스트라이드 (stride)

- 필터를 적용하는 위치의 간격



합성곱 연산

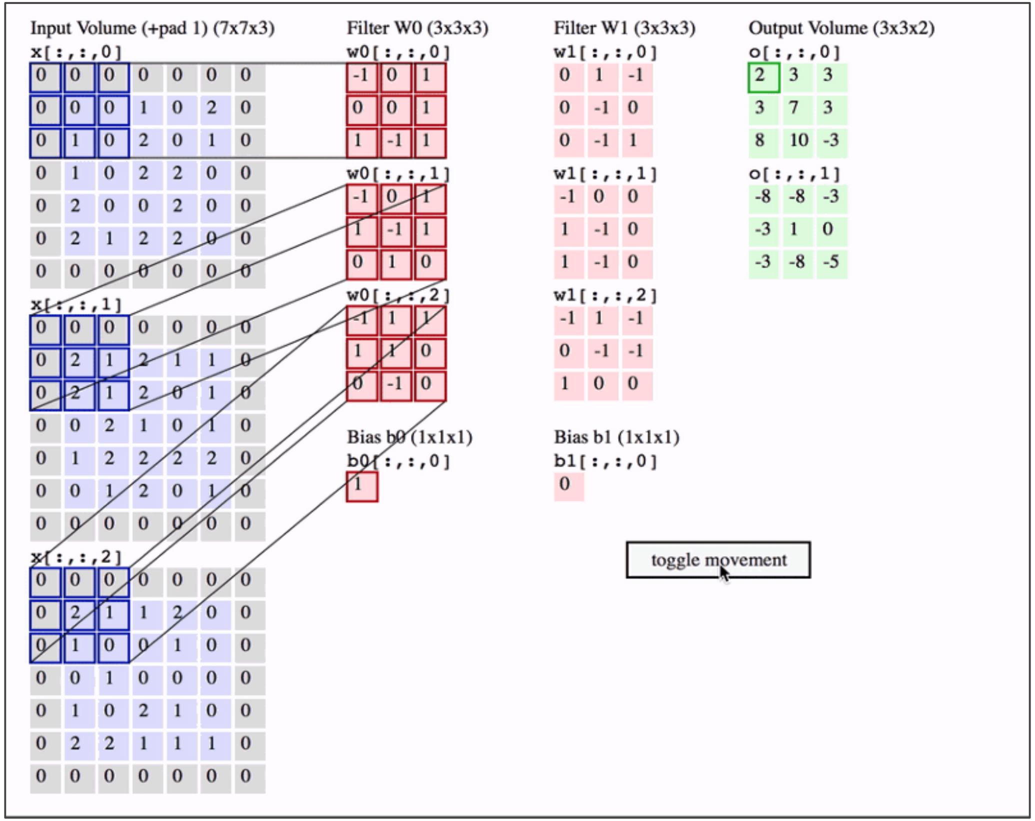
- 입력 사이즈 : H, W
- 필터 사이즈 : FH, FW
- 패딩 : P
- 스트라이드 : S
- 출력 사이즈 : OH, OW

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

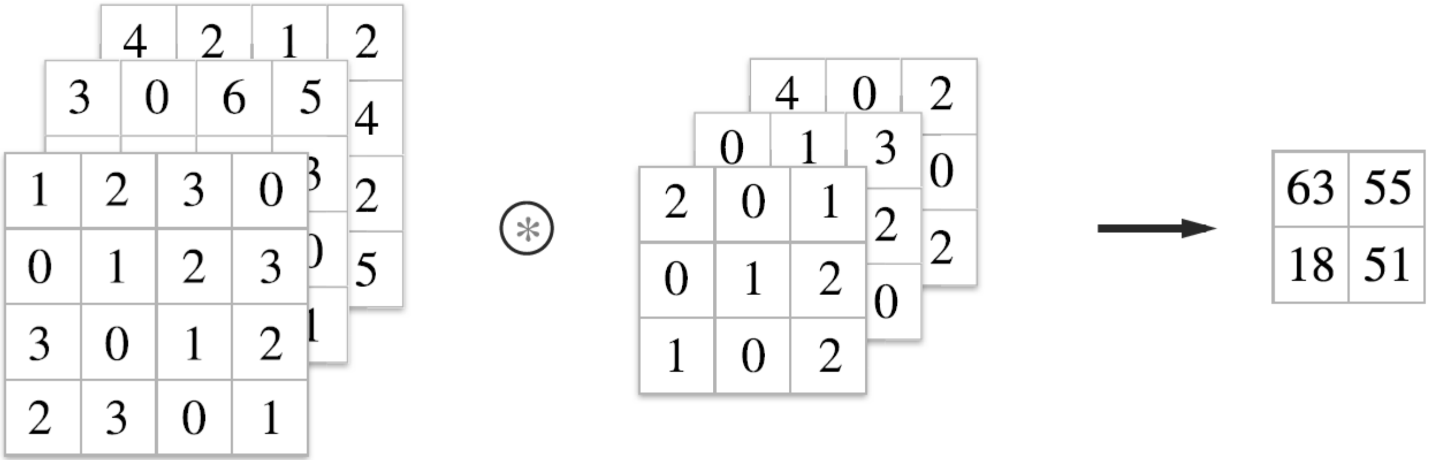
$$\underbrace{\overbrace{\begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix}}^{4}}_{\text{입력 데이터}} \circledast \underbrace{\overbrace{\begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix}}^{3}}_{\text{필터}} \longrightarrow \begin{bmatrix} 15 & 16 \\ 6 & 15 \end{bmatrix} + \underbrace{\begin{bmatrix} 3 \end{bmatrix}}_{\text{편향}} \longrightarrow \underbrace{\overbrace{\begin{bmatrix} 18 & 19 \\ 9 & 18 \end{bmatrix}}^{2}}_{\text{출력 데이터}}$$

합성곱 연산


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 1 0 2 0 1 0
0 1 0 2 2 0 0
0 2 0 0 2 0 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 2 1 1 0
0 2 1 2 0 1 0
0 0 2 1 0 1 0
0 1 2 2 2 2 0
0 0 1 2 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 1 2 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 1 0 2 1 0 0
0 2 2 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 1
0 0 1
1 -1 1
w0[:,:,1]
-1 0 1
1 -1 1
0 1 0
w0[:,:,2]
-1 1 1
1 1 0
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 1 -1
0 -1 0
0 -1 1
w1[:,:,1]
-1 0 0
1 -1 0
1 -1 0
w1[:,:,2]
-1 1 -1
0 -1 -1
1 0 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 3 3
3 7 3
8 10 -3
o[:,:,1]
-8 -8 -3
-3 1 0
-3 -8 -5
toggle movement

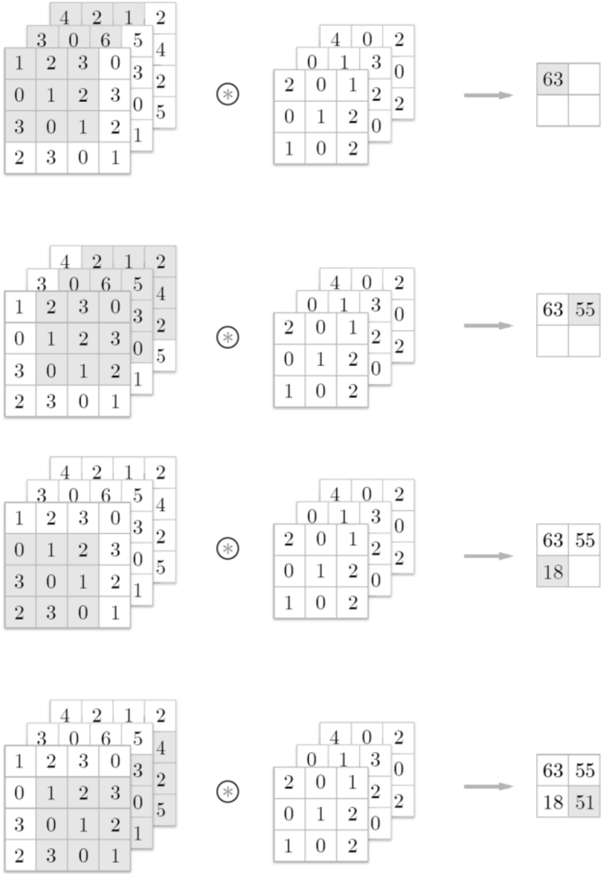
3차원 데이터

- 채널마다 수행한 뒤 그 결과를 더해 하나의 출력

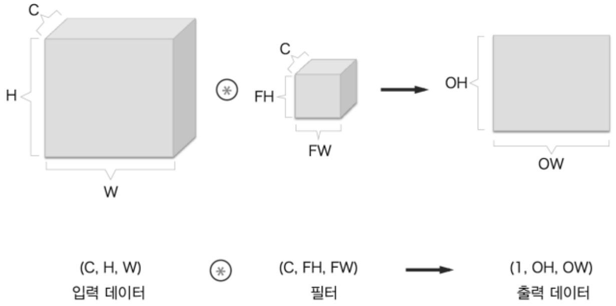


입력 데이터　　　　필터　　　　출력 데이터

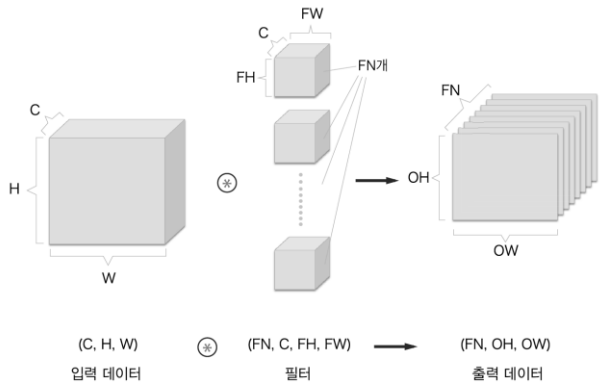
3차원 데이터 합성곱 계산 순서



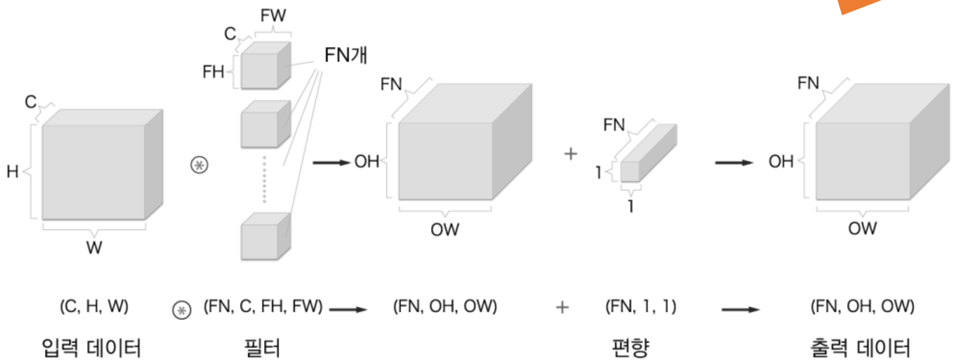
블럭으로 생각하기



필터 추가



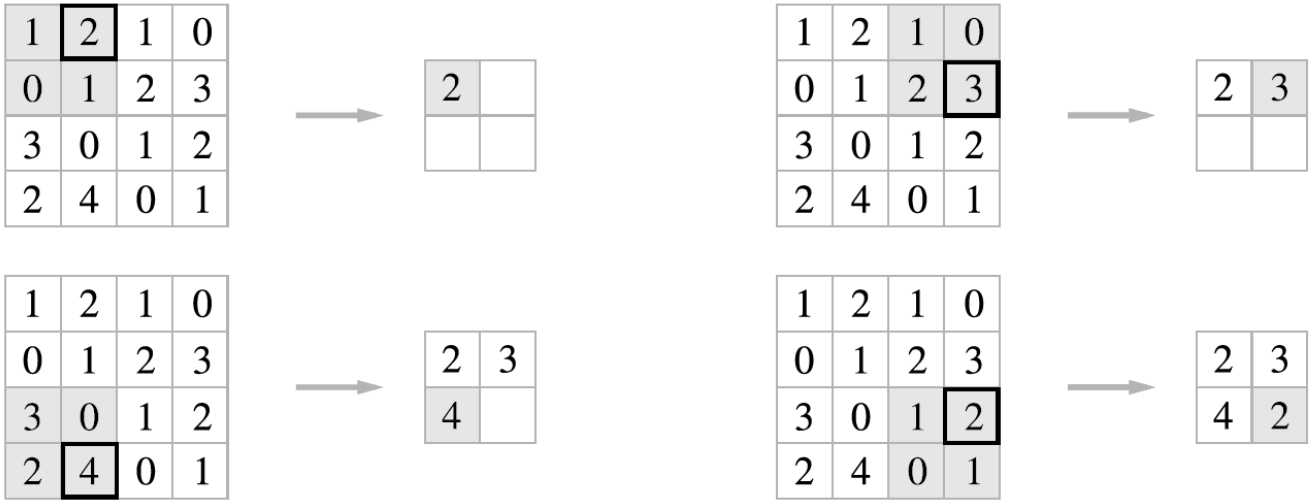
편향 추가



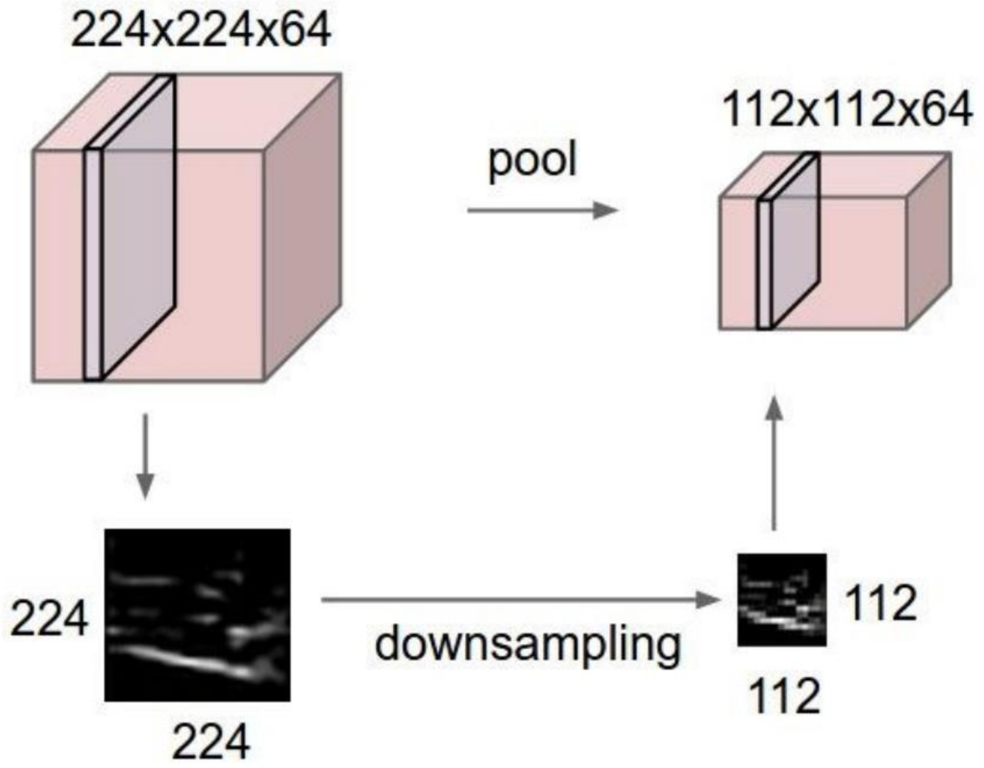
풀링 (Pooling)

- 피처맵의 크기를 줄여주는 과정 (down sampling)
- 필터와 겹치는 부분의 가장 큰 값을 선택하면 max pooling, 겹쳐지는 부분의 평균값을 취하면 average pooling.
- 가장 큰 값의 위치가 입력된 이미지와의 상관성이 가장 높기 때문에 max-pooling이 효과적
- Max-pooing은 특정 영역의 대표값을 선택하여 해당 영역을 강조하는 효과가 있다.
- Pooling을 적용하면 데이터의 상당 부분이 버려진다. 정보를 버리는 것이 비효율적일 수도 있지만(단점), 이로 인한 이득도 있다(장점).
- 예를 들어 입력 데이터의 특정 부분이 약간 틀어지거나 위치가 변해도 pooling을 적용하면 동일한 결과를 얻을 수 있다(viewpoint invariance : 시각 불변성 특성, ex: 고양이 코의 위치가 약간 달라져도 고양이 코라고 정상적으로 인식할 수 있음).
- Pooling은 convolution 처럼 weight (W)를 갖지 않는다. 즉, 학습의 대상은 아니고 down sampling을 위한 크기만 지정한다(non-trainable layer).

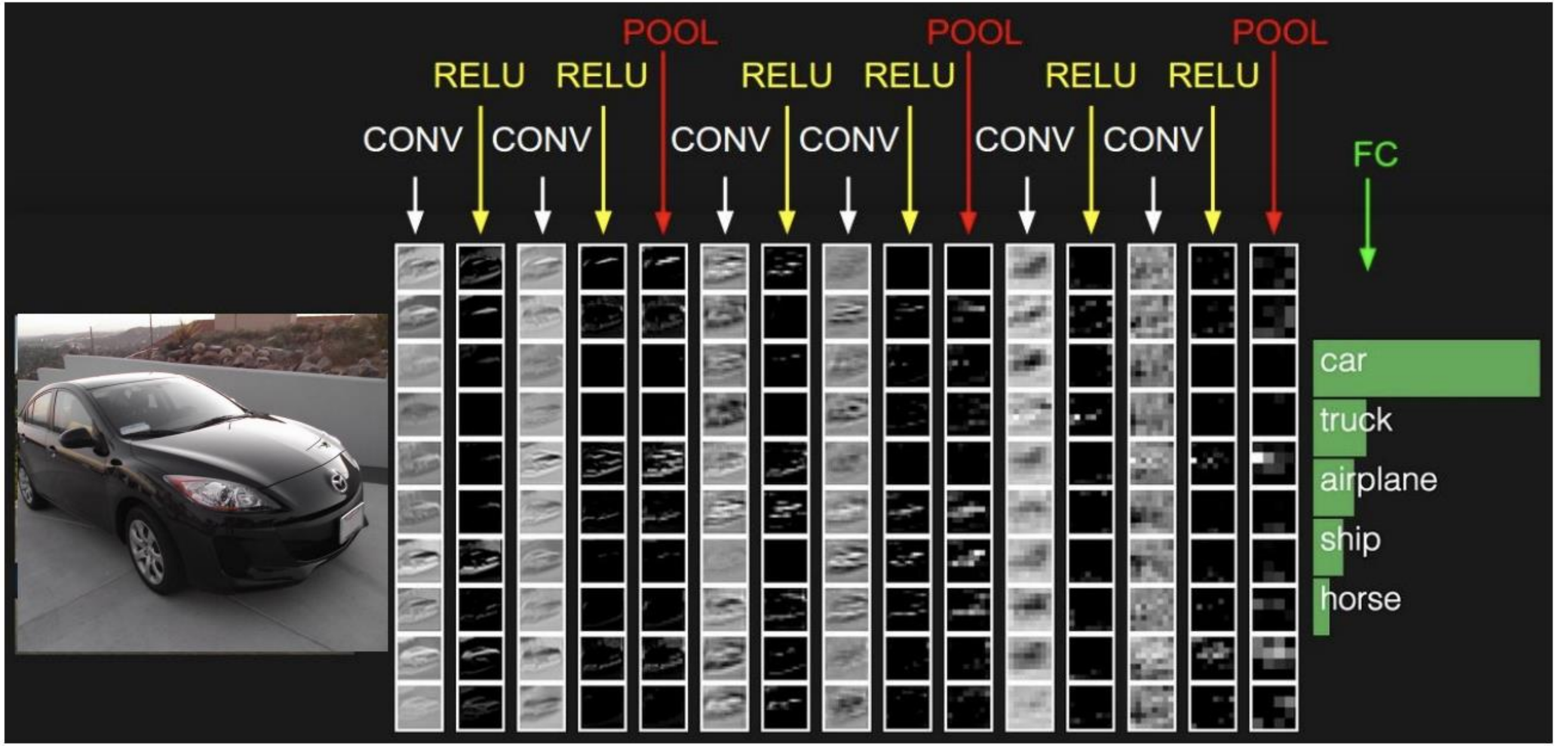
풀링의 처리 순서



Pooling



CNN (Convolutional Neural Network)

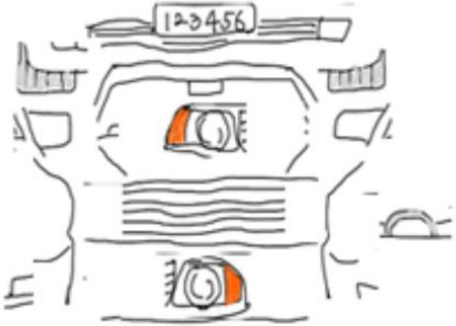


CNN

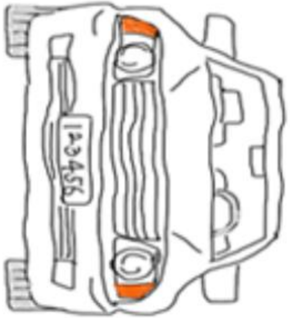
- 다량의 데이터 확보가 관건 ->**Data Augmentation** (ImageDataGenerator)

자동차

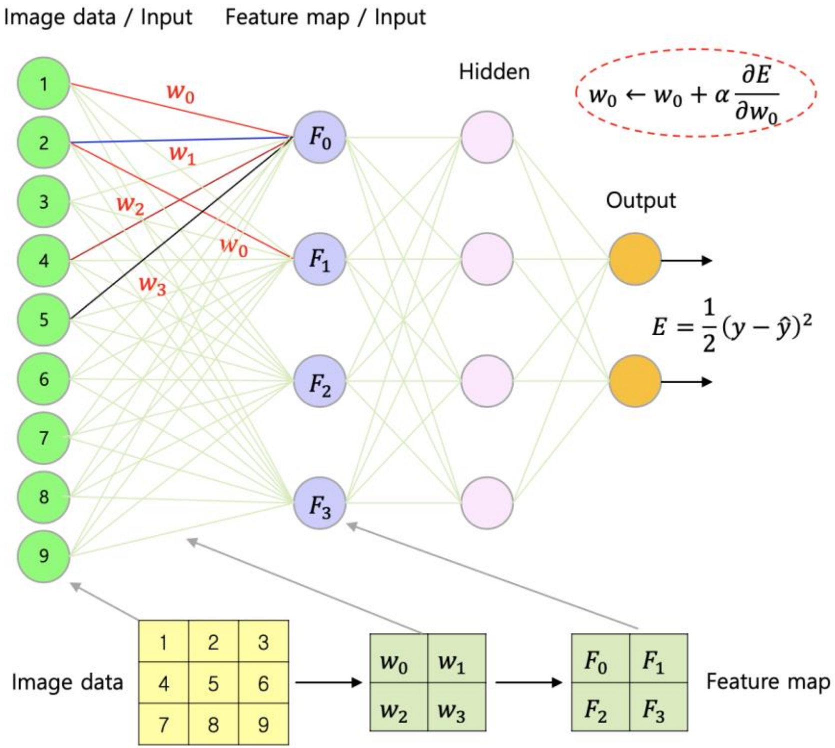
자동차 가능성 큼



??



CNN 학습

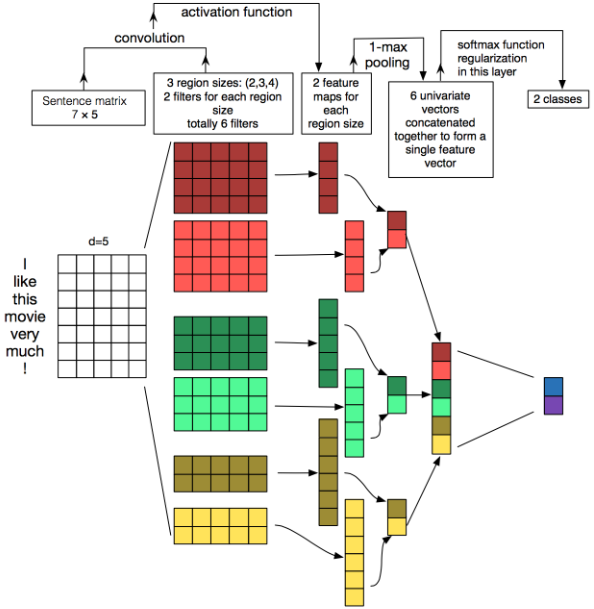


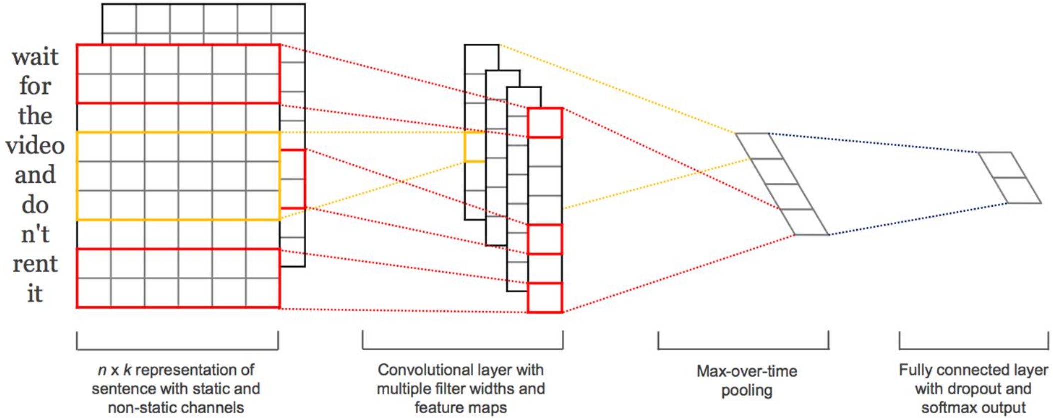
$$F_0 = 1 \times w_0 + 2 \times w_1 + 4 \times w_2 + 5 \times w_3$$

CNN for Text Classification
(Convolutional Neural Network)

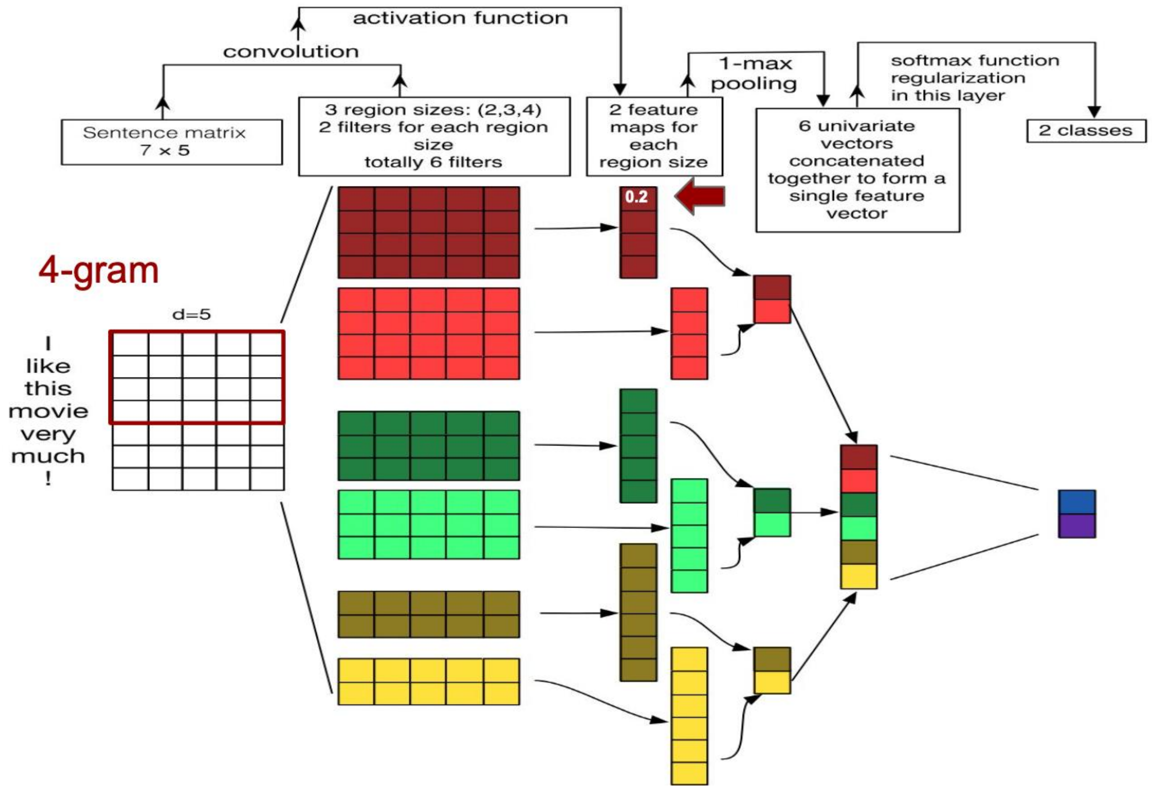
CNN

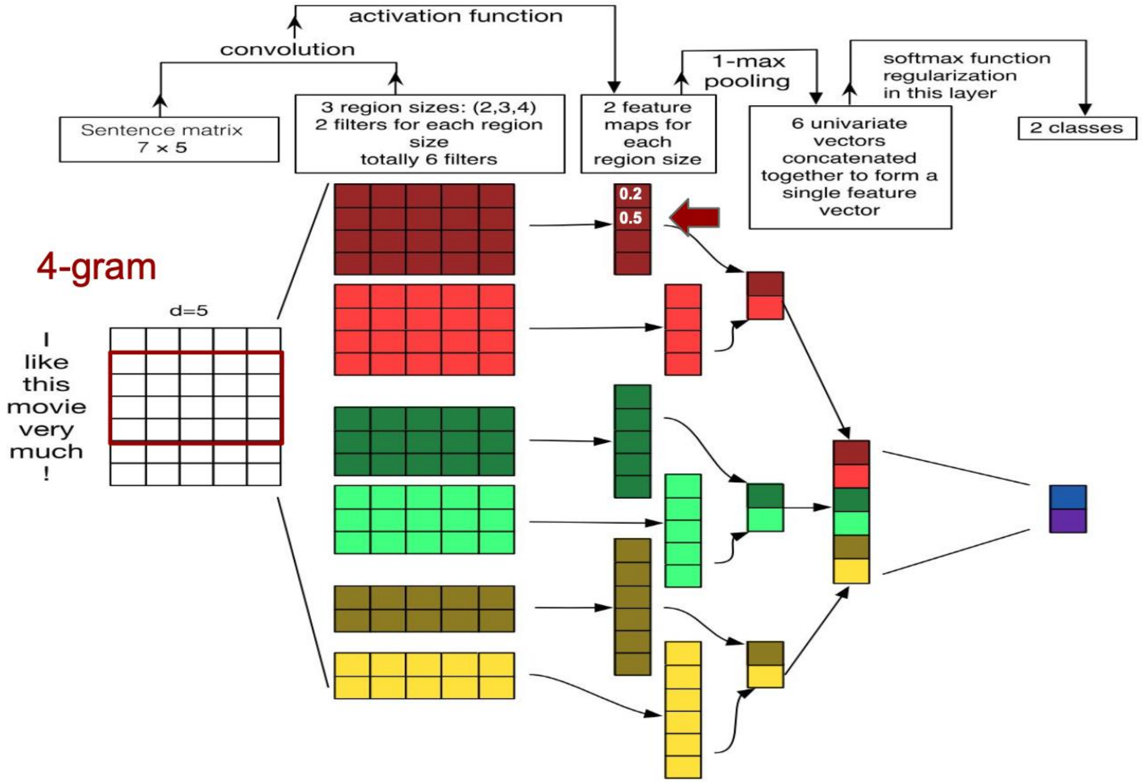
Convolutional Neural Networks for Sentence Classification



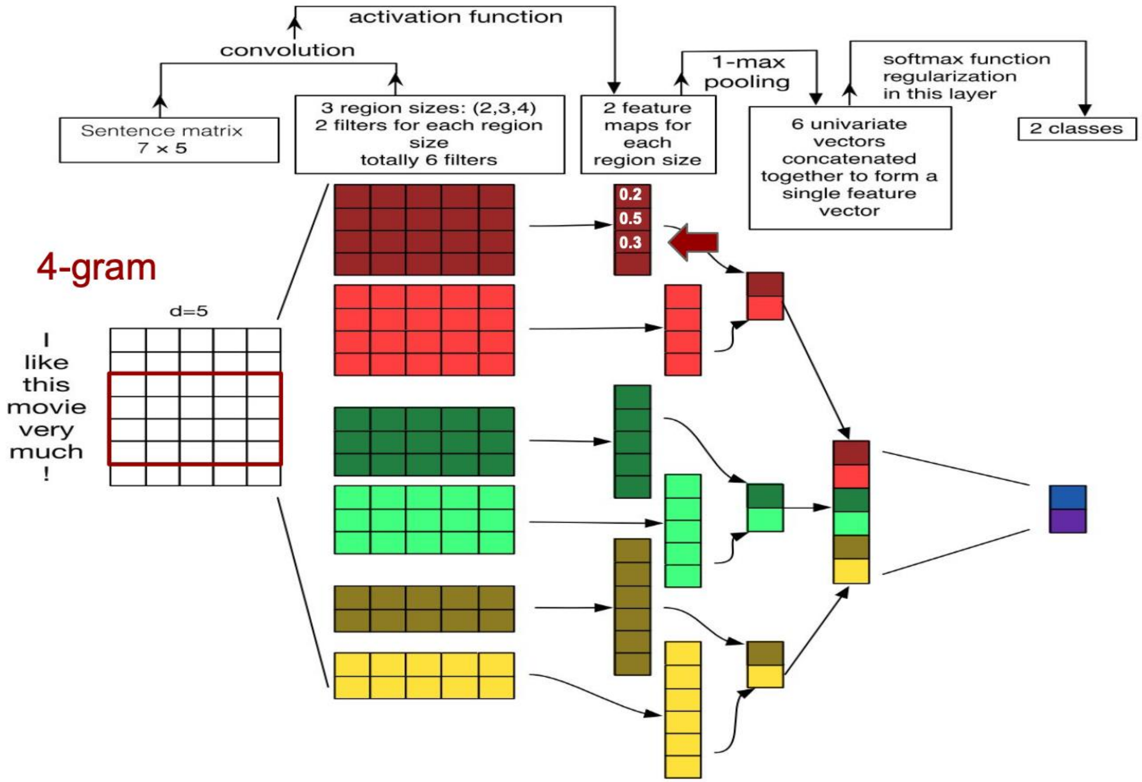


Convolutional Neural Networks for Sentence Classification

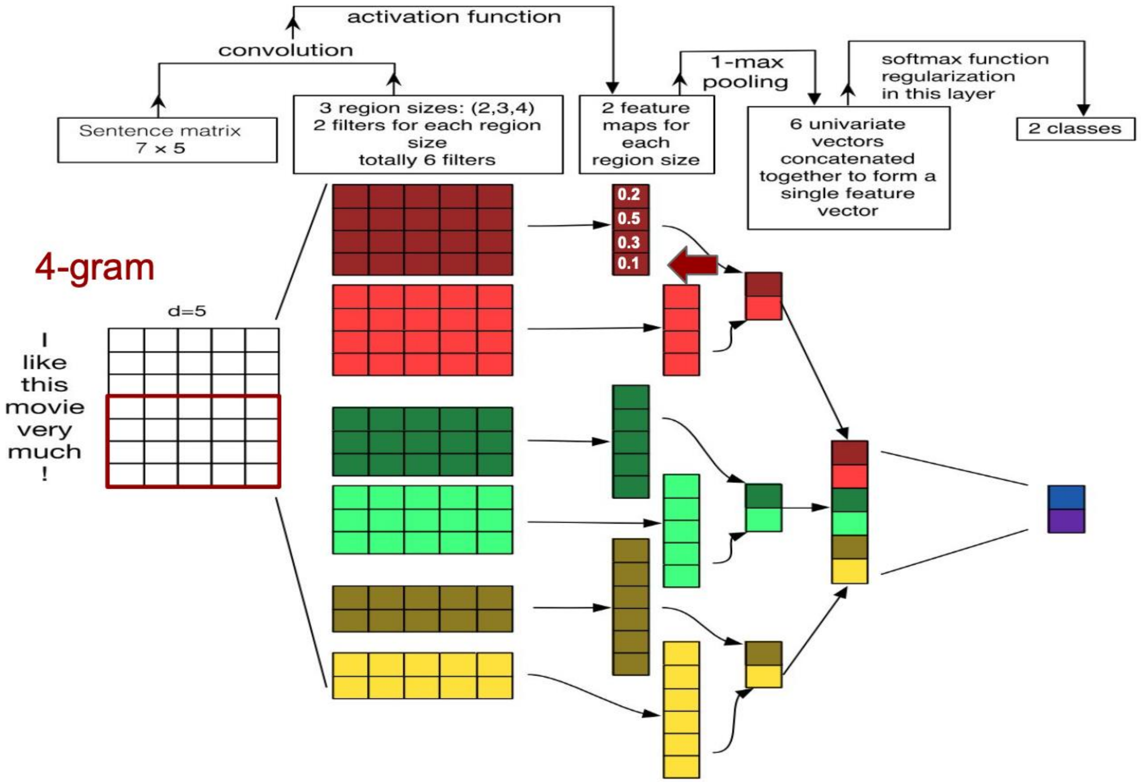




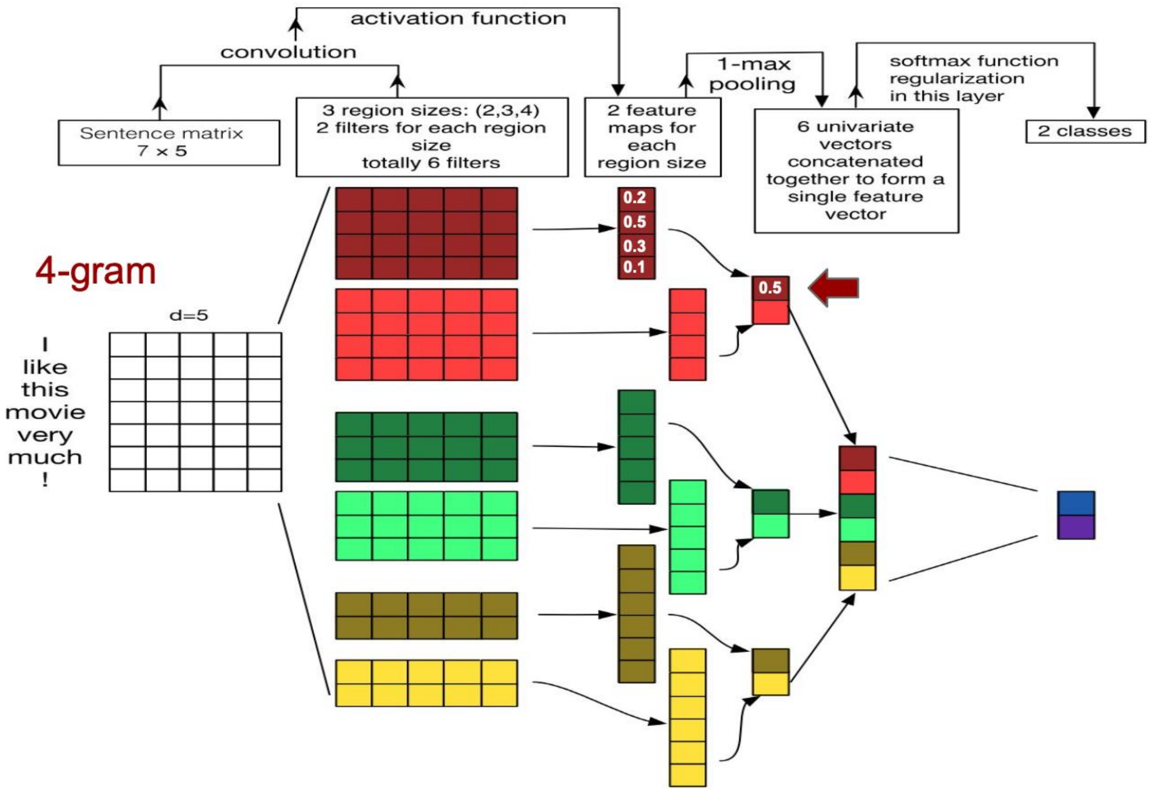
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
0.2
0.5
4-gram
d=5
I
like
this
movie
very
much
!



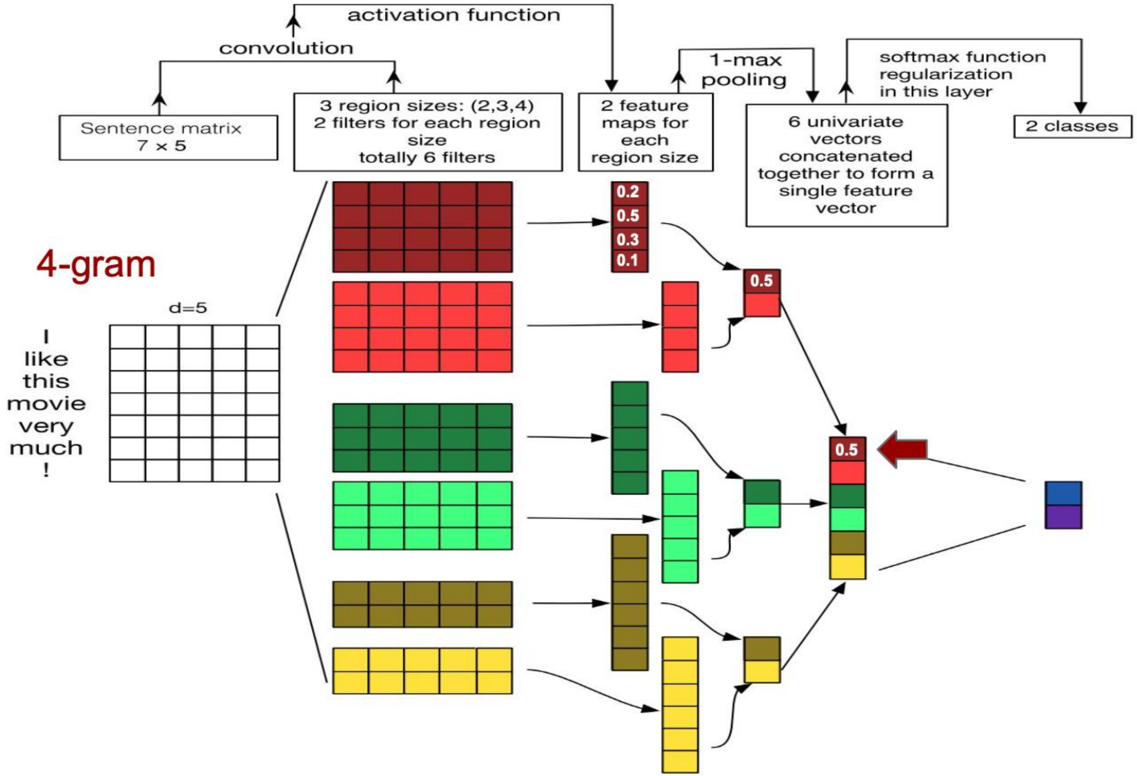
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 x 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
4-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3

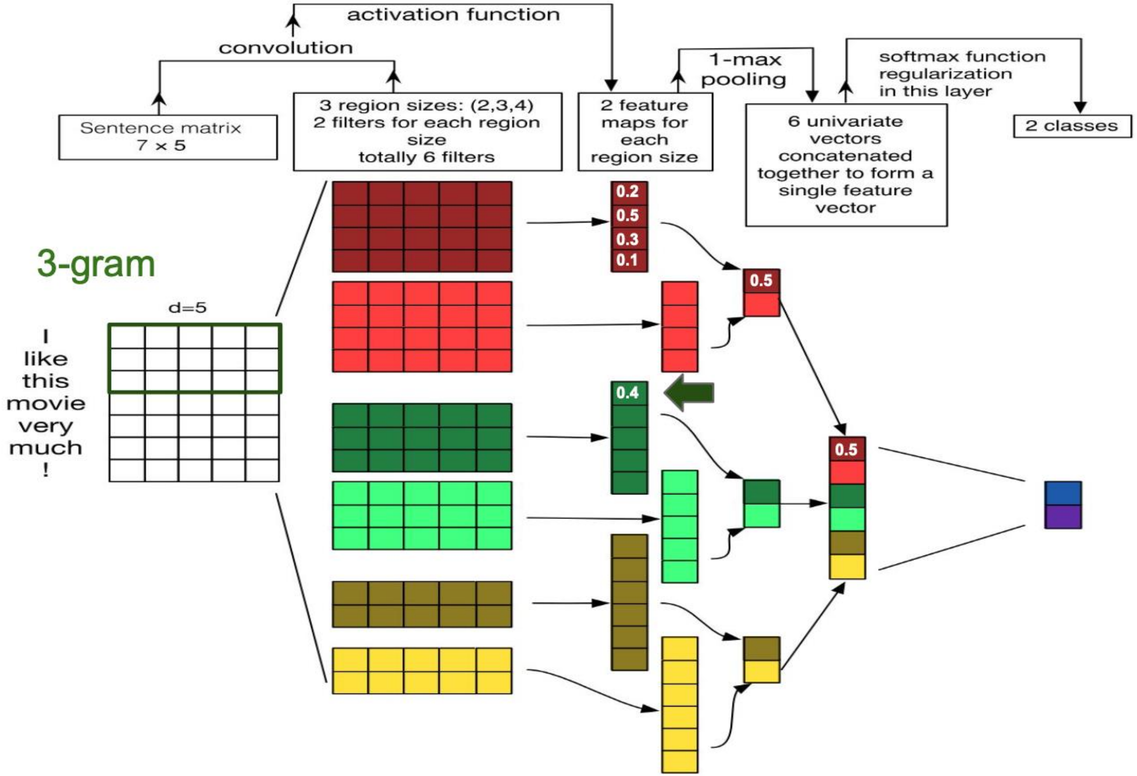


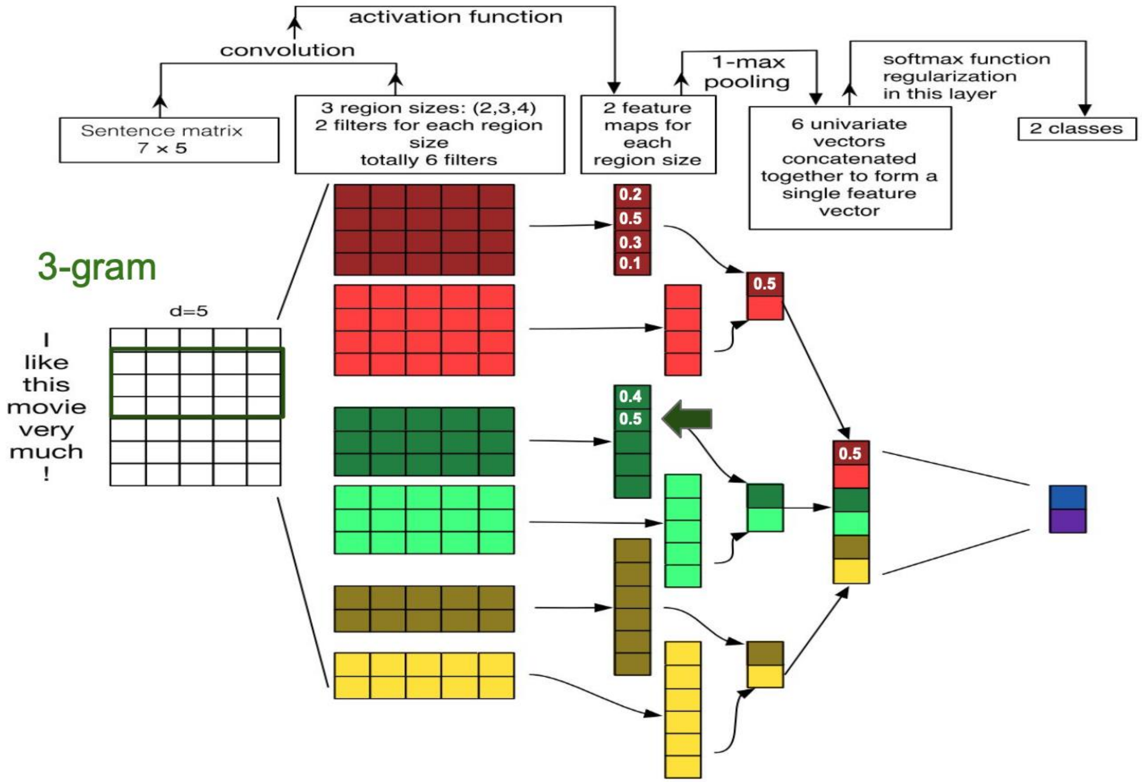
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 x 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
4-gram
d=5
I like this movie very much !
0.2
0.5
0.3
0.1

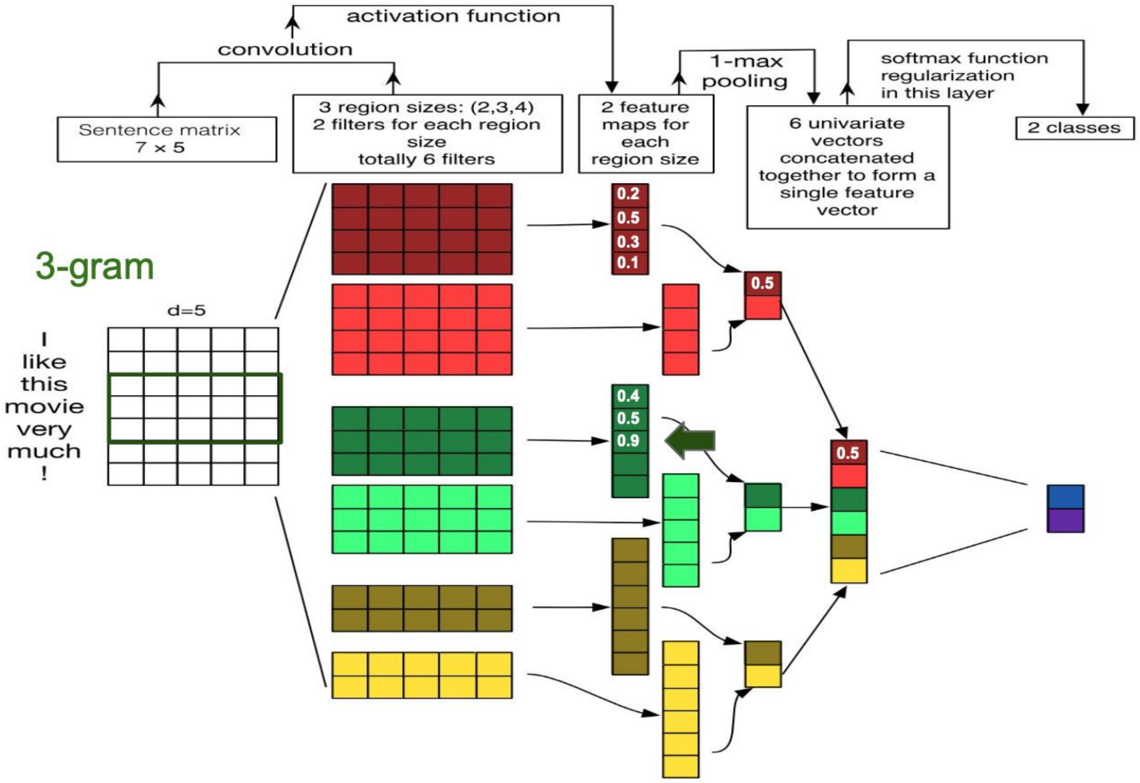

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
4-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5

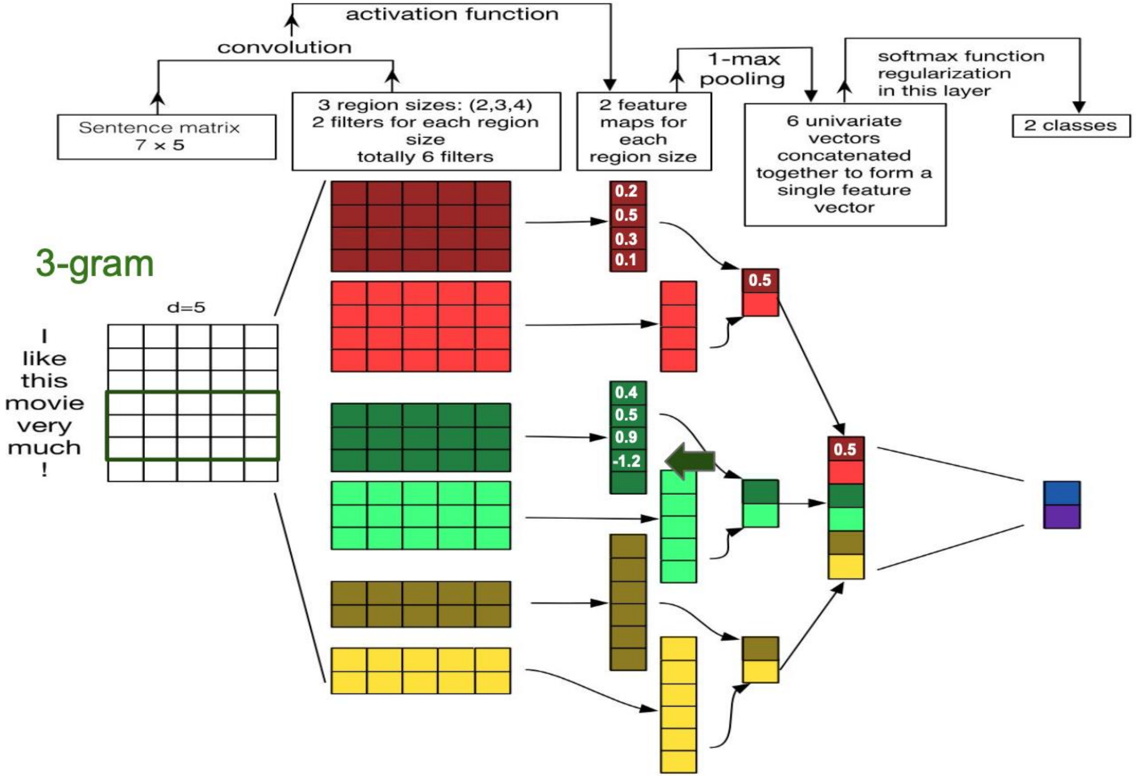
4-gram

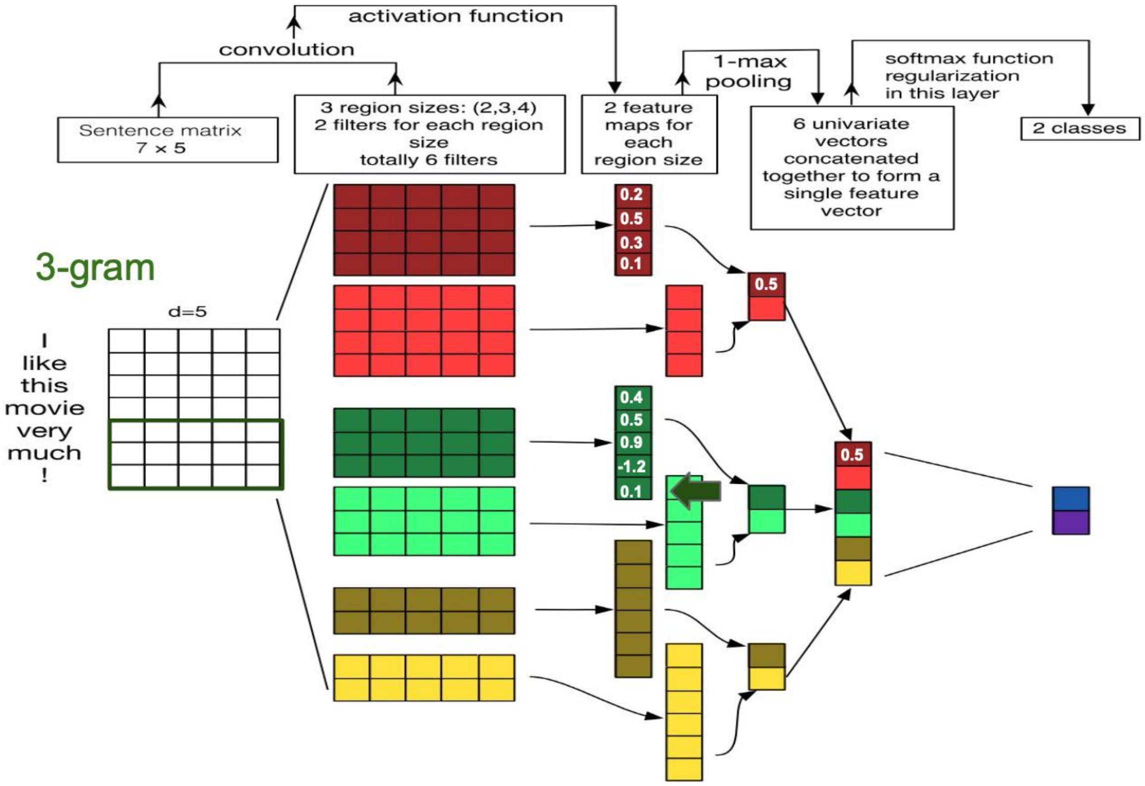

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.5


activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region size
totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5

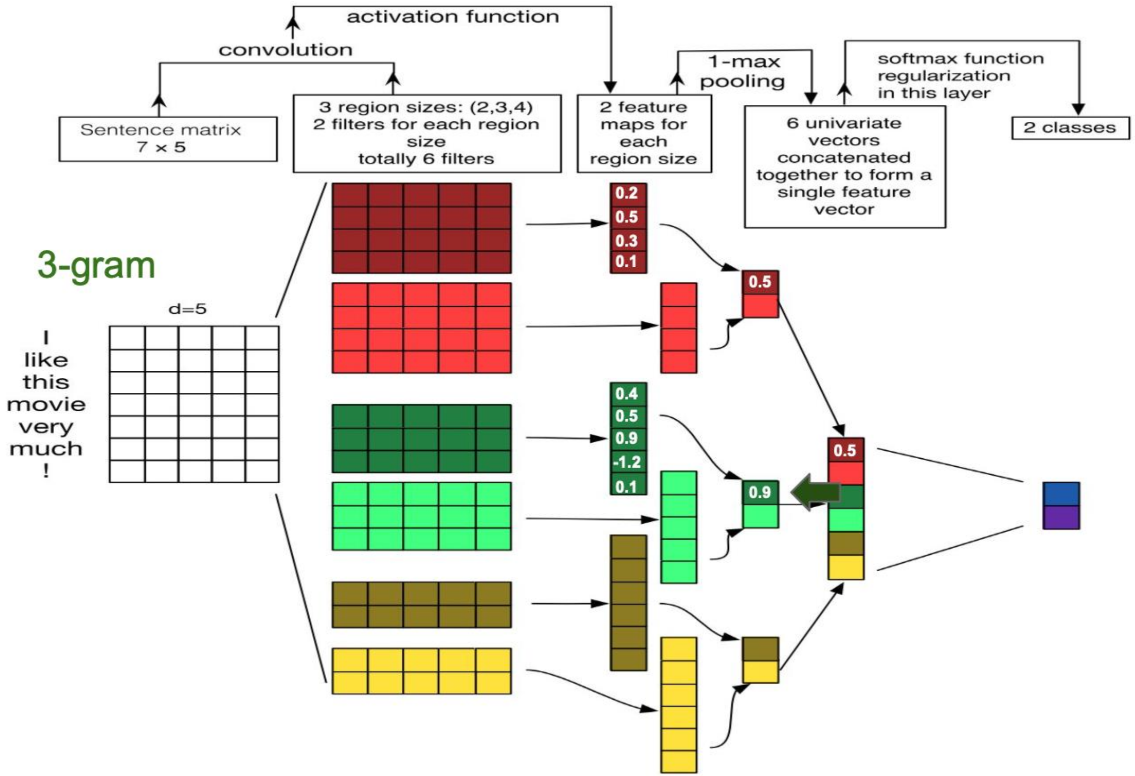

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5
0.5

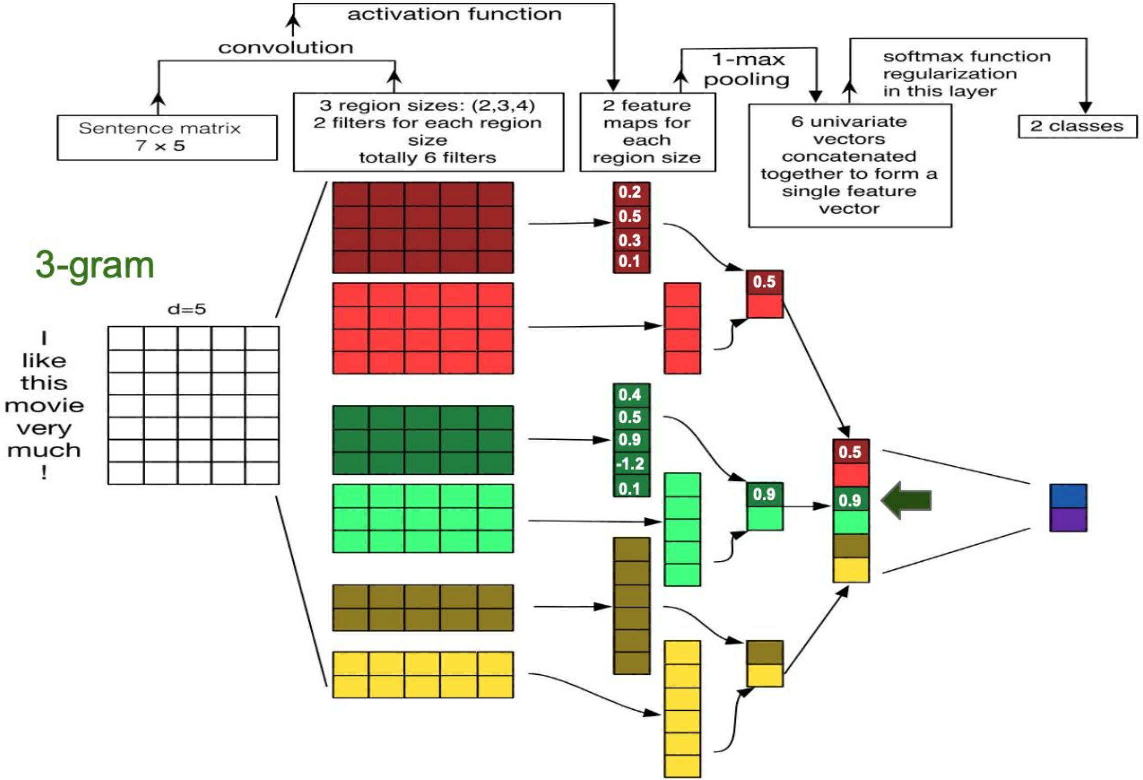

activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region
size
totally 6 filters
2 feature
maps for
each
region size
1-max
pooling
6 univariate
vectors
concatenated
together to form a
single feature
vector
softmax function
regularization
in this layer
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5
0.9
0.5


activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region
size
totally 6 filters
2 feature
maps for
each
region size
1-max
pooling
6 univariate
vectors
concatenated
together to form a
single feature
vector
softmax function
regularization
in this layer
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5
0.9
-1.2
0.5



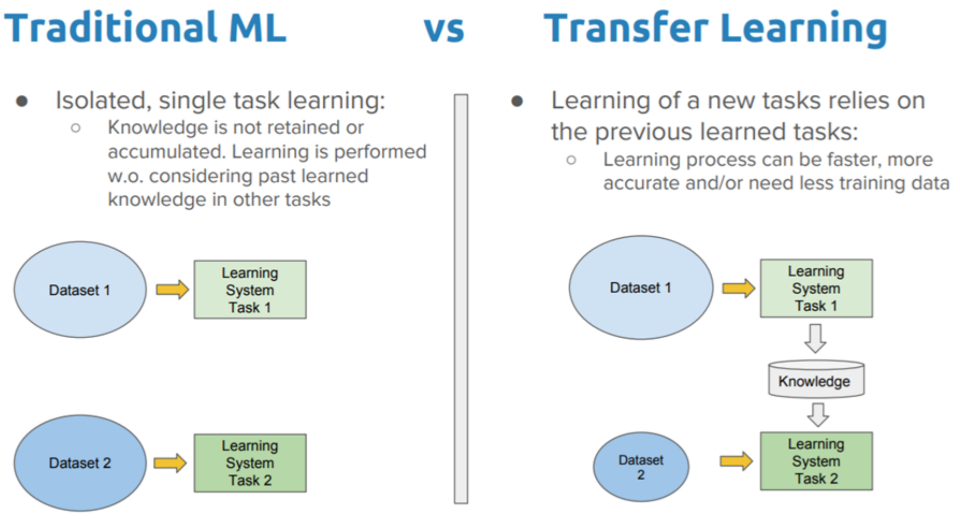
activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region
size
totally 6 filters
2 feature
maps for
each
region size
1-max
pooling
6 univariate
vectors
concatenated
together to form a
single feature
vector
softmax function
regularization
in this layer
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5
0.9
-1.2
0.1
0.5


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5
0.9
-1.2
0.1
0.9
0.5

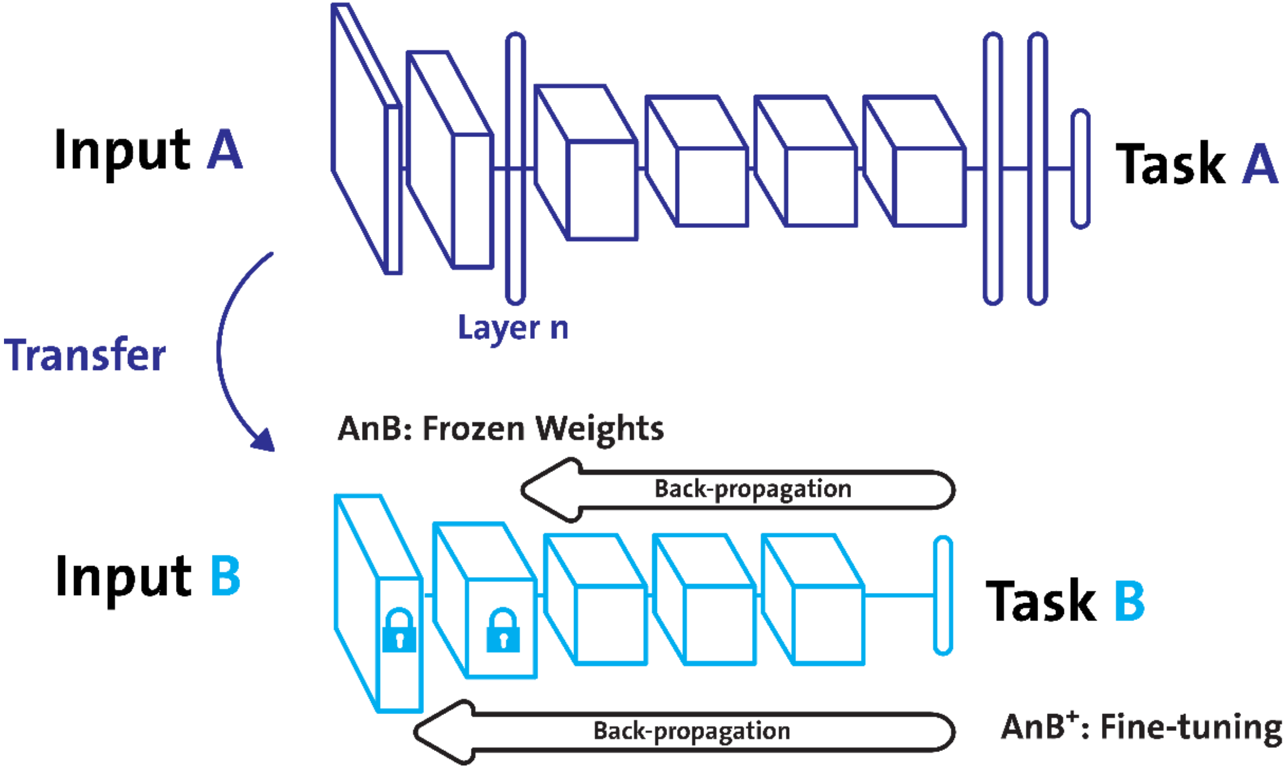

activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region size
totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
3-gram
d=5
I
like
this
movie
very
much
!
0.2
0.5
0.3
0.1
0.5
0.4
0.5
0.9
-1.2
0.1
0.9
0.5
0.9

Transfer Learning

- Transfer learning : 전이 학습은 특정 환경에서 만들어진 AI 알고리즘을 다른 비슷한 분야에 적용
- Fine-Tuning : 사전에 학습된 모델의 파라미터를 task에 맞추어 정교하게 조정하여 활용



Transfer Learning



Transfer Learning