TensorFlow (3)

고려대학교 INI Lab

Contents

01 Introduction to CNN

02 History of CNN

03 CNN structure

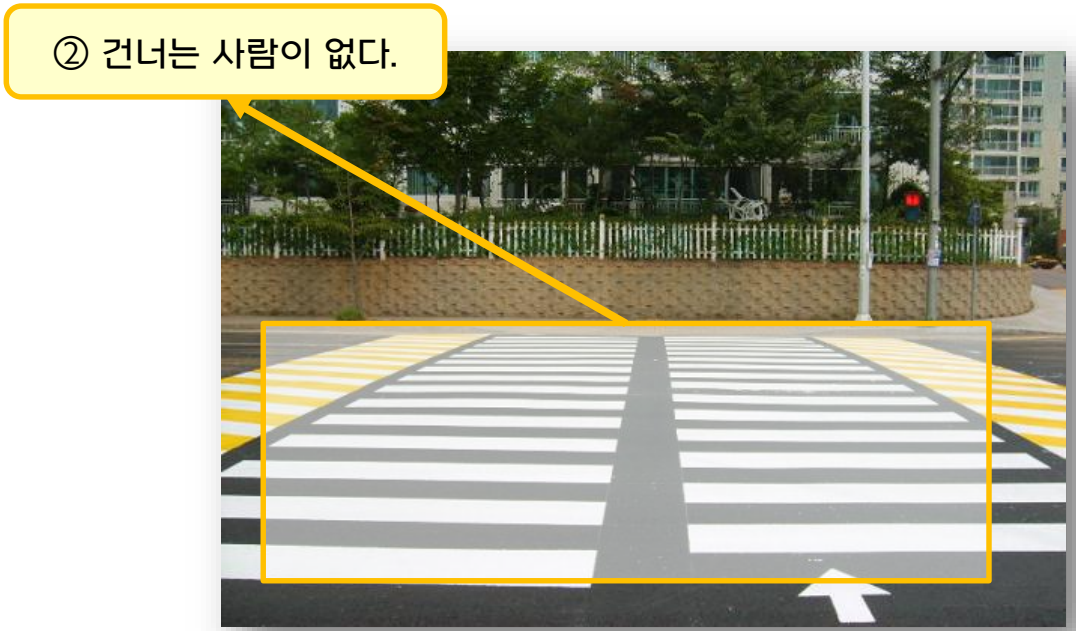
Introduction to CNN

Example of Vision Process

이런 경우에 횡단보도를 건너도 되는가?

Example of Vision Process

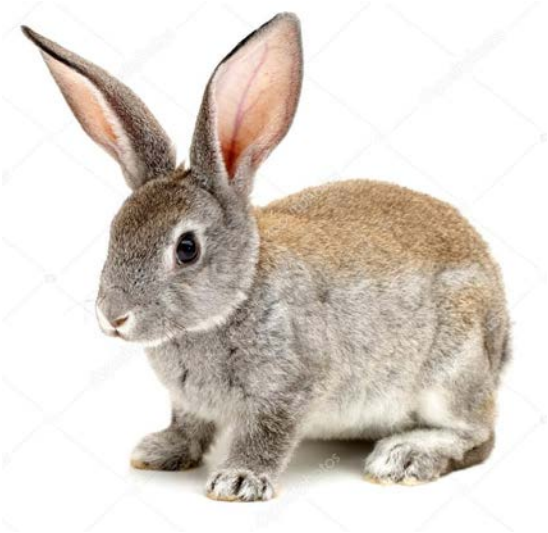
Example of Vision Process



Example of Vision Process

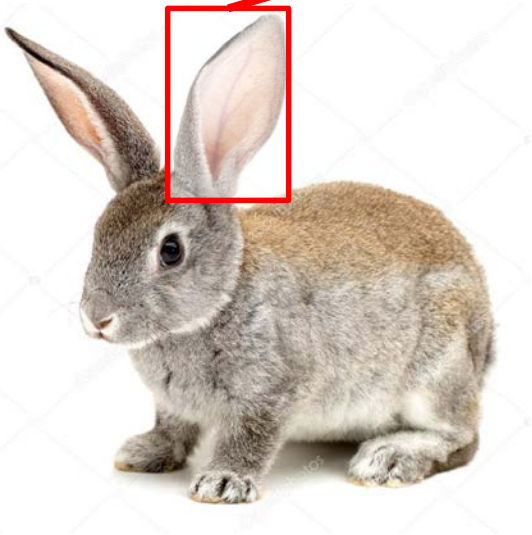
Example of Vision Process

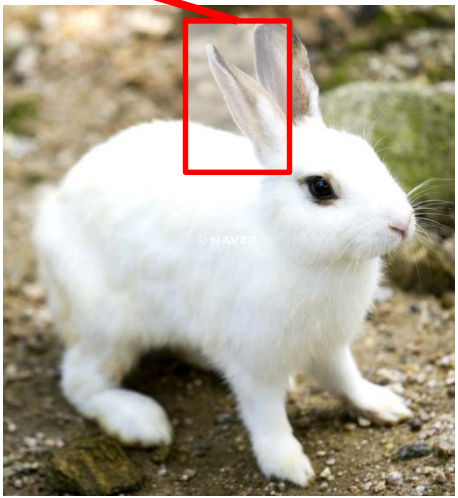
두 사진은 같은 동물의 사진인가?



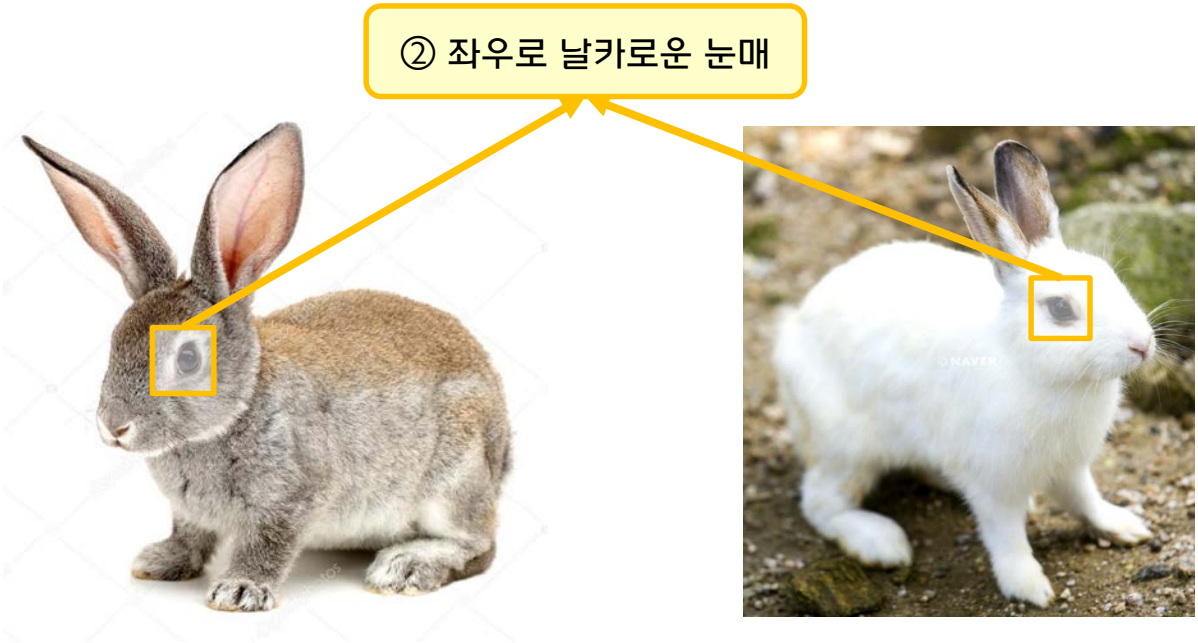
Example of Vision Process

① 뾰족한 귀 모양

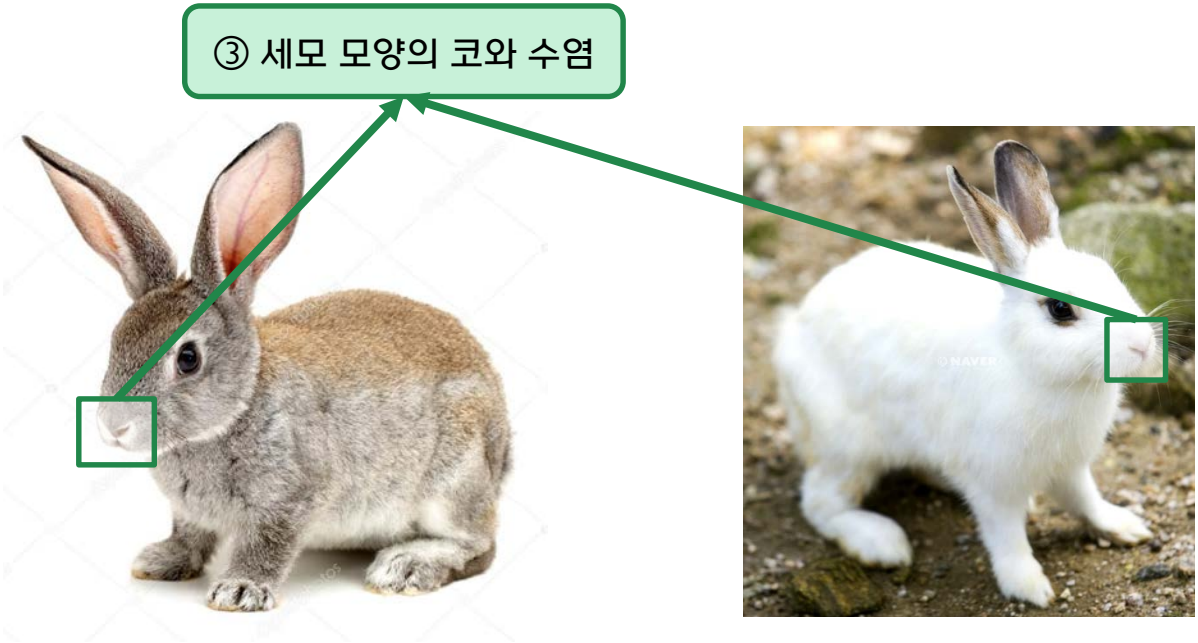




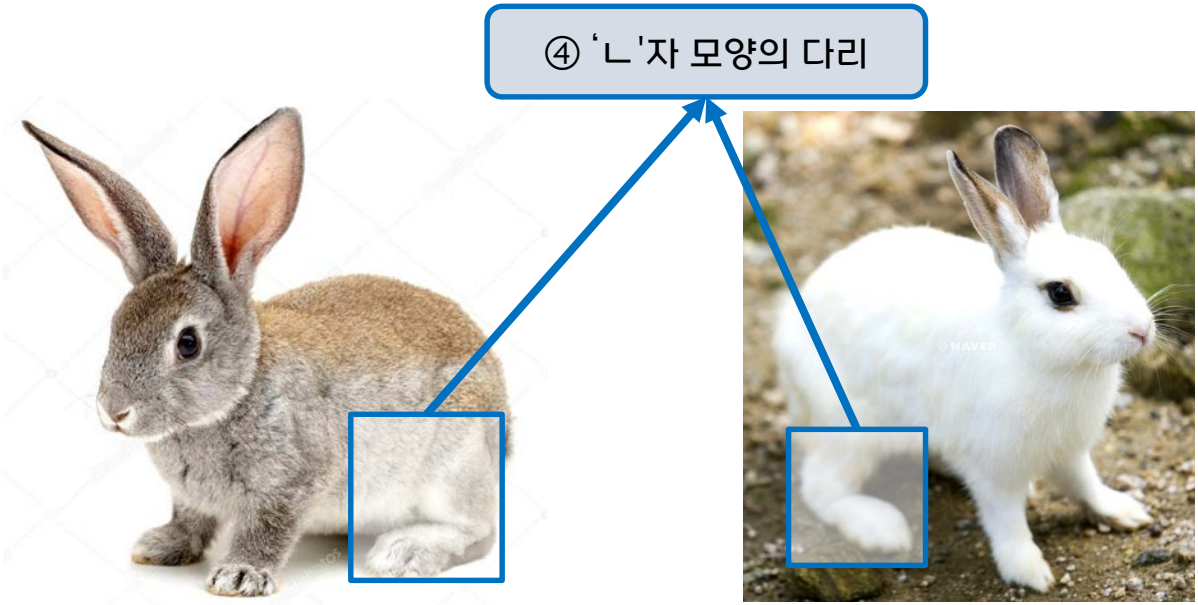
Example of Vision Process



Example of Vision Process

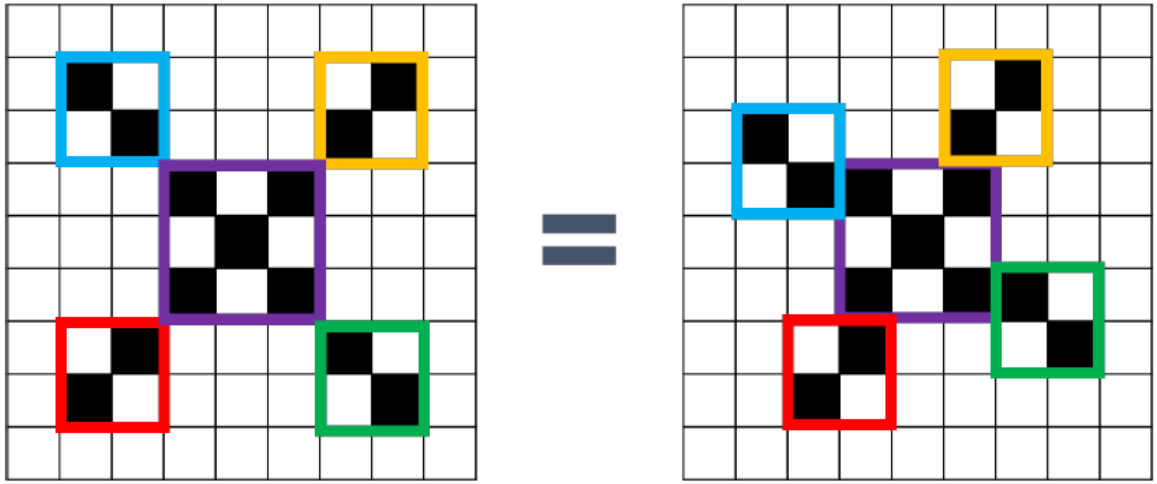


Example of Vision Process



Example of Vision Process

Feature의 특색을 학습할 수 있다면, feature의 유사성을 통해 새로운 데이터로 쉽게 학습 가능



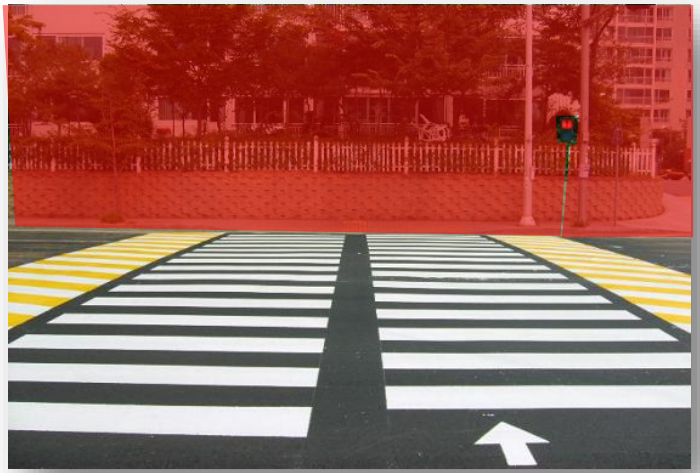
Idea

현실 세계의 실제 데이터에는 불필요한 정보가 많다

▶ 전체 데이터를 통해 판단을 내리는 것보다, 부분 데이터를 통해 판단을 내리는 것이 불필요한 정보를 제거할 수 있다.

▶ 데이터마다 불필요한 정보는 제각각 다르기 때문에, 사람이 불필요한 정보를 지정해주는 것은 비효율적

▶ 전체에서 특징을 추출하는 것이 아니라 지역적인 부분으로부터 특징을 추출하도록 모델 구축

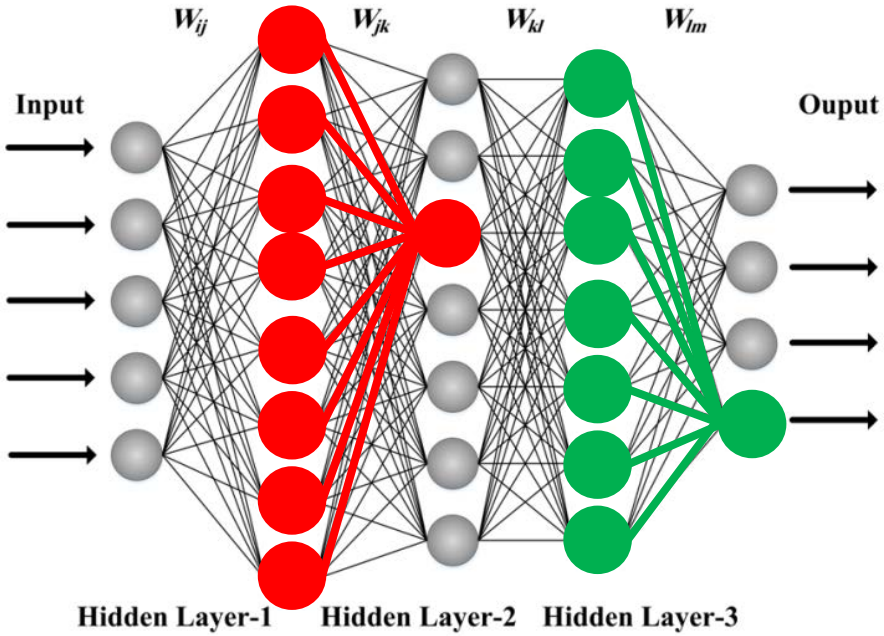


Review FFNN

이전의 모든 layer로부터 정보를 받아서 판단을 내림

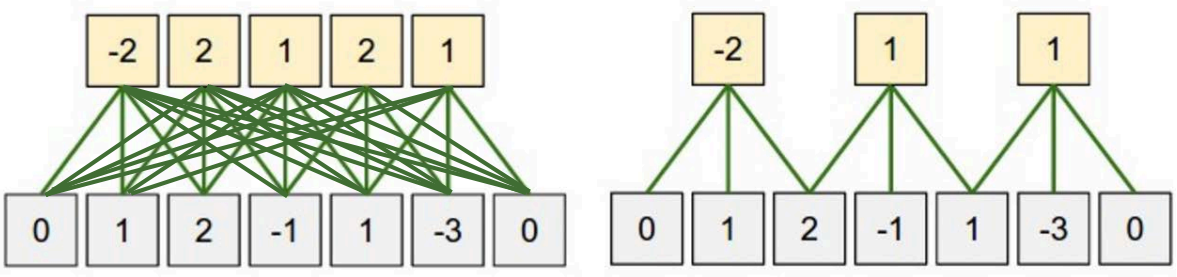
▶ 불필요한 정보까지 입력을 받기 때문에, overfit(과적합)될 가능성이 다분함

▶ 인접한 Layer끼리 모두 연결되어 있기 때문에, Fully-Connected Neural Network라고도 부름



Convolutional Neural Network

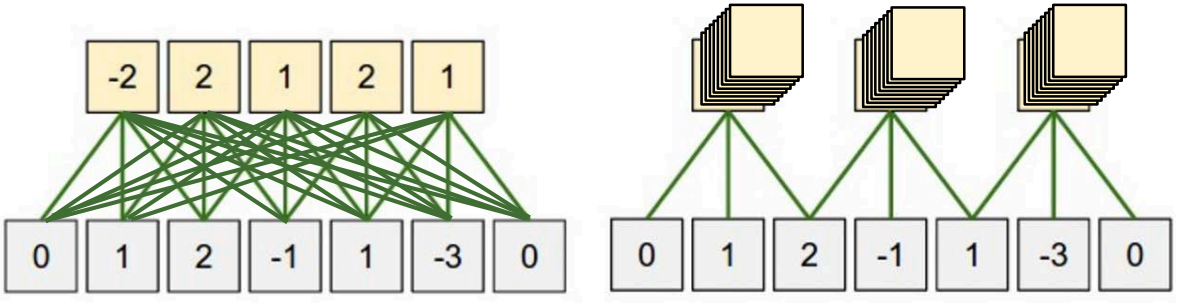
전체 데이터를 보지 않고, 일부분 데이터만을 가져와서 다음 layer에 값을 전달



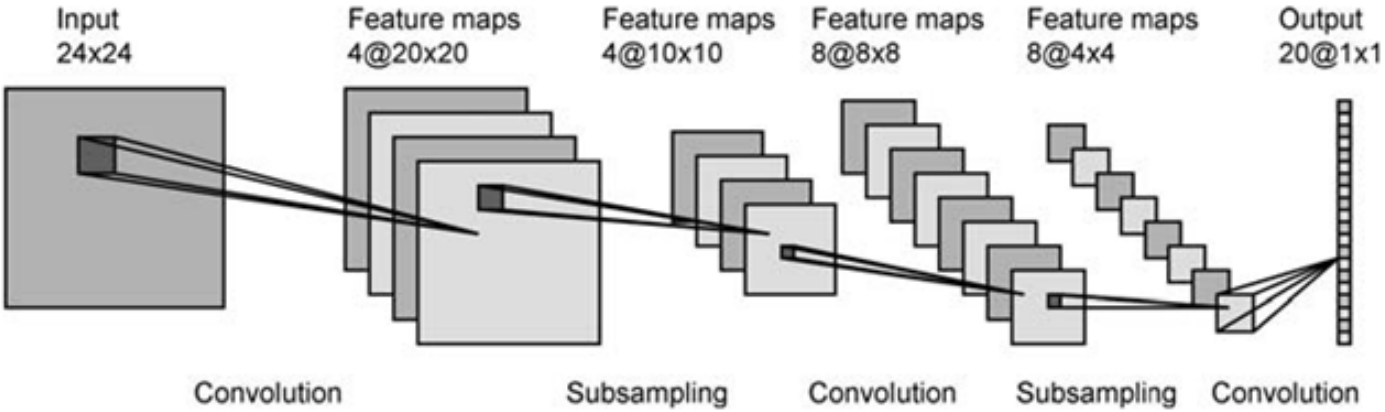
Convolutional Neural Network

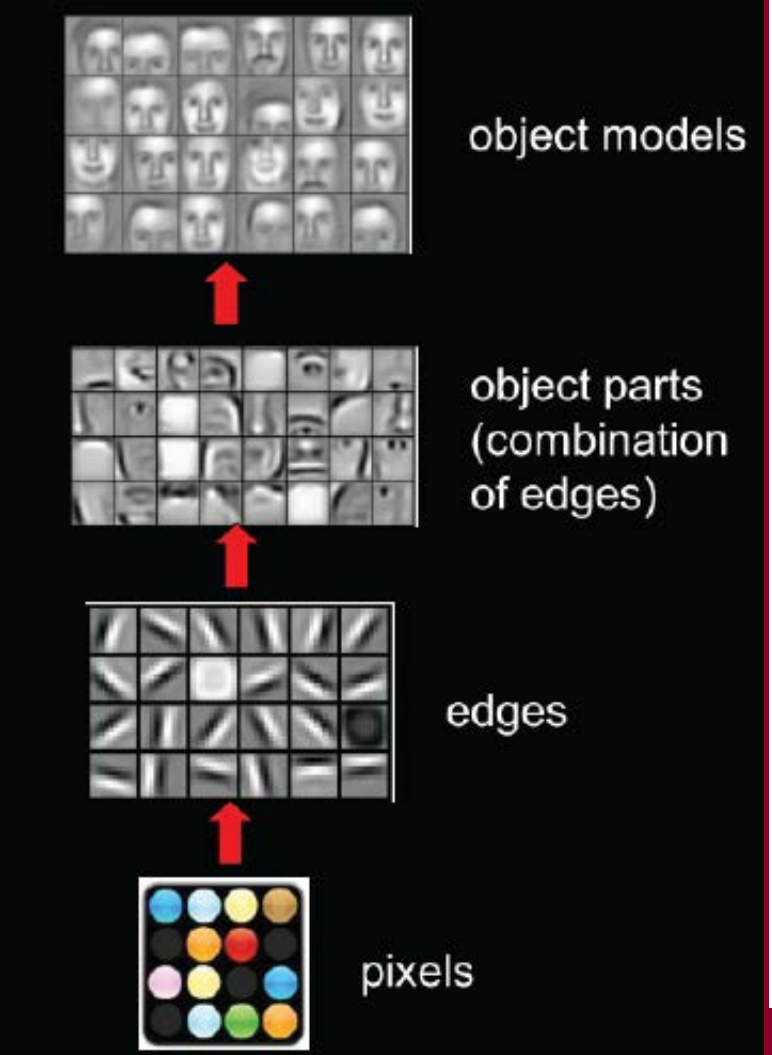
전체 데이터를 보지 않고, 일부분 데이터만을 가져와서 다음 layer에 값을 전달

▶ 일부 데이터만 보는 대신, 동일한 데이터에 대해서 다양한 정보를 학습



Convolutional Neural Network





Convolutional Neural Network

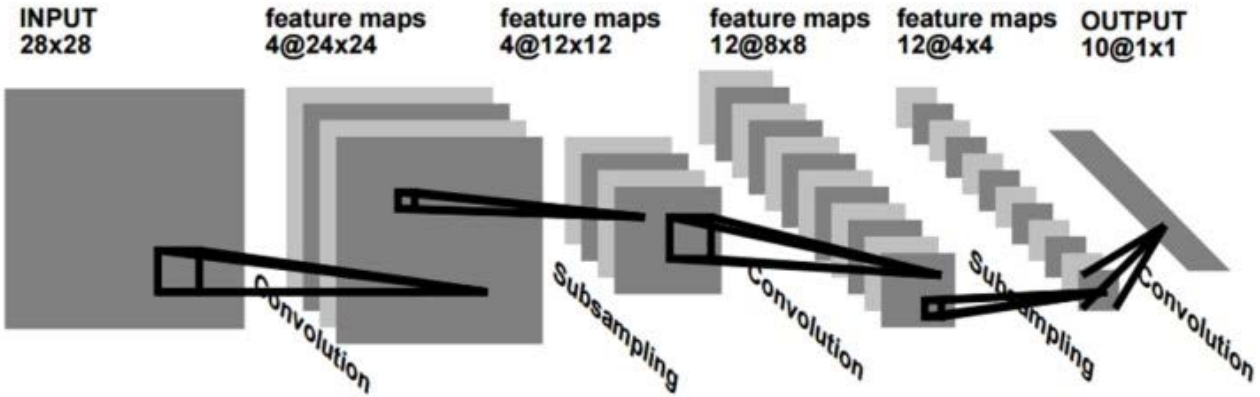
낮은 층에서는 단순한 feature 학습
단순한 feature를 조합하여 복잡한 구조 학습

History of CNN

LeNet (1990)

Yann LeCun이 제안한 최초의 CNN 모델

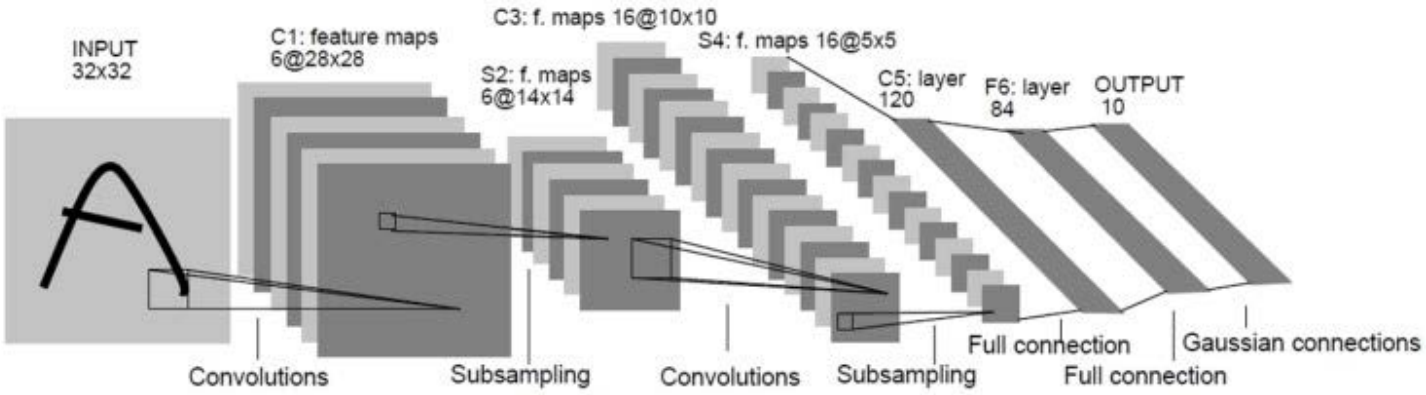
▶ feature map, shared weight, sub-sampling 등 CNN의 기본적인 구조가 이미 확립



LeNet-5 (1998)

LeNet과 구조는 동일하지만, 더 많은 feature map가 추가

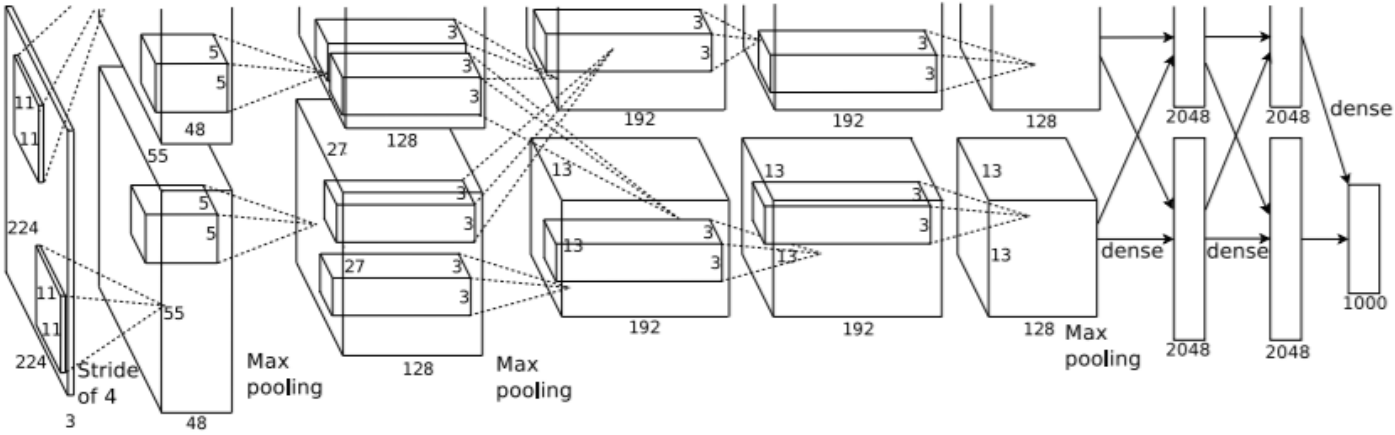
▶ Convolution 뿐만 아니라, 최종적으로 fully-connected layer를 추가



AlexNet (2012)

Alex Khrizevsky 가 구상한 모델로, ILSVRC 2012 우승

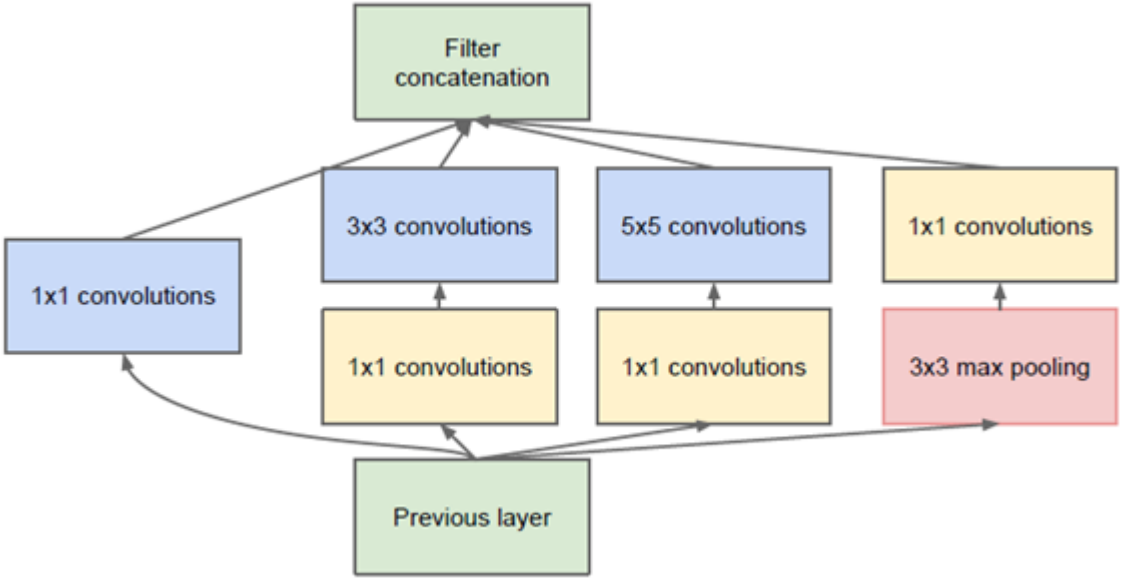
- 이 결과로 인해 CNN을 포함한 deep learning 기법이 주목받게 됨.
- AlexNet과 전반적인 구조는 유사함.
- Multi-GPU 환경에 맞는 모델 구축



GoogLeNet (2014)

구글에서 제안한 CNN 모델

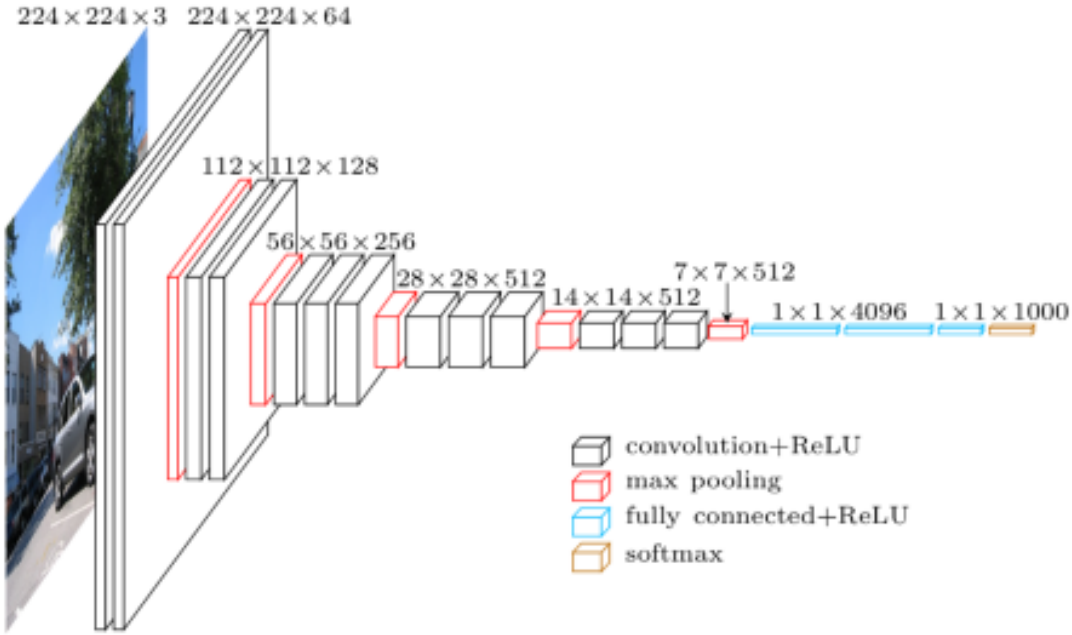
- ▶ 다양한 모양을 가지고 깊이가 각자 다른 filter를 생성하고 이를 조합하여 학습
- ▶ 기존 모델보다 한 layer에서 다양한 정보를 학습 가능



VGGNet (2014)

GoogleNet과 동시대에 발표된 모델로, 성능은 살짝 뒤떨어지나 오히려 더 많이 차용되는 모델

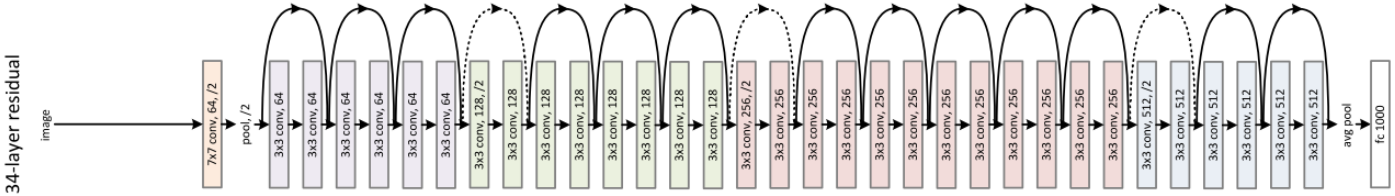
▶ AlexNet의 성능을 극복하기 위해, 모델의 구조를 바꾸는 것이 아닌 layer를 더 깊게 쌓아버림



ResNet (2015)

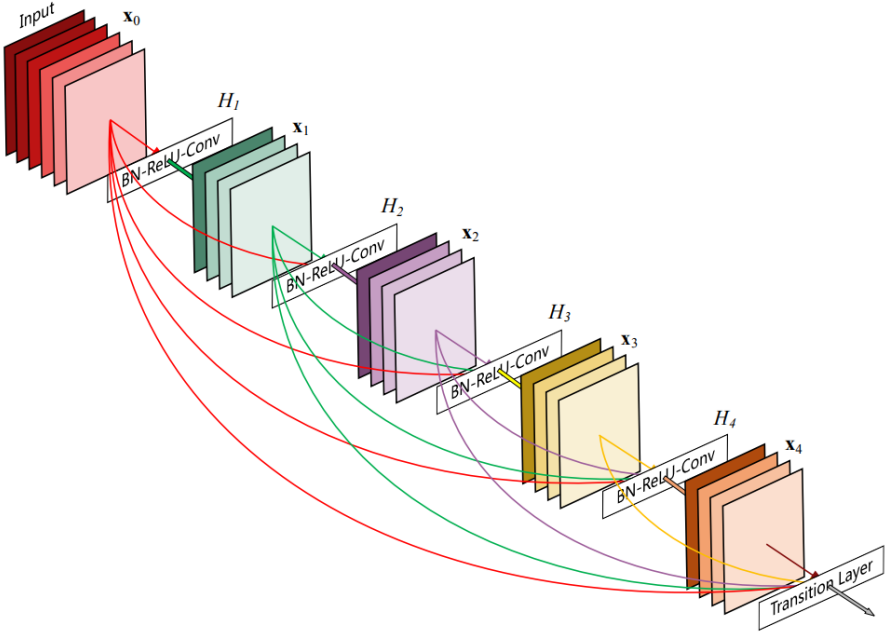
계속해서 Layer를 쌓아나가자 새로운 문제가 발생 (ResNet – 126 layer)

- ReLU를 사용했음에도 불구하고 layer가 깊어지면 gradient vanishing 문제가 발생
- 인접한 layer끼리만 데이터를 전달하는 것이 아니라, 몇 단계를 건너 뛰어서 데이터를 전달하는 구조



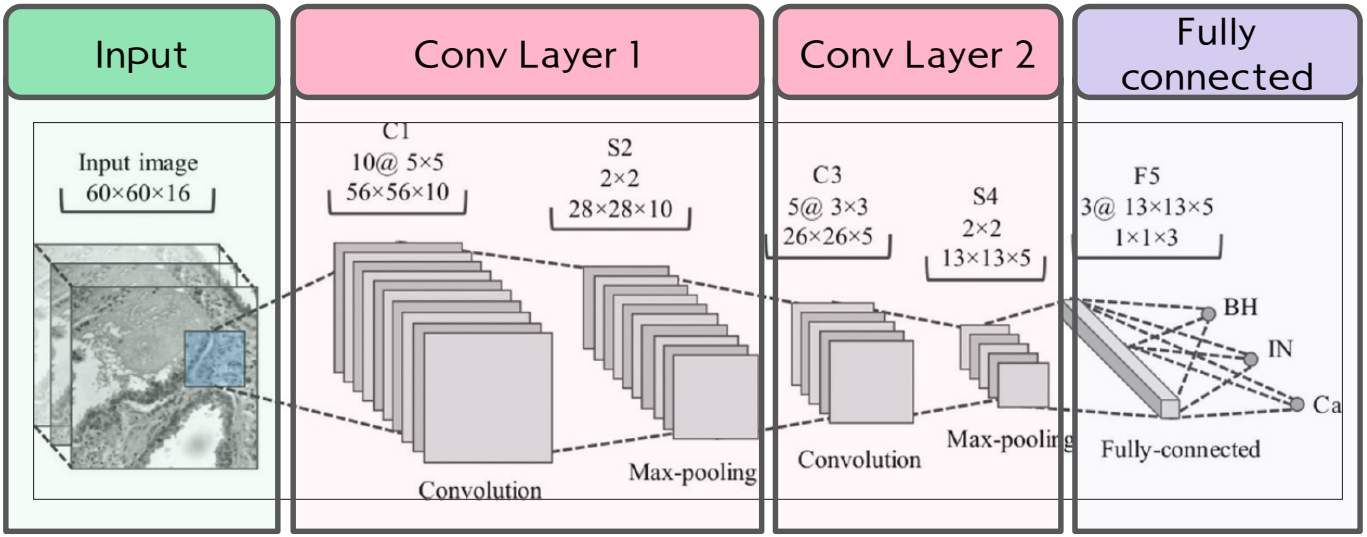
DenseNet (2016)

ResNet과 접근 방식은 유사하지만, 더 많은 통로를 만들어줌.

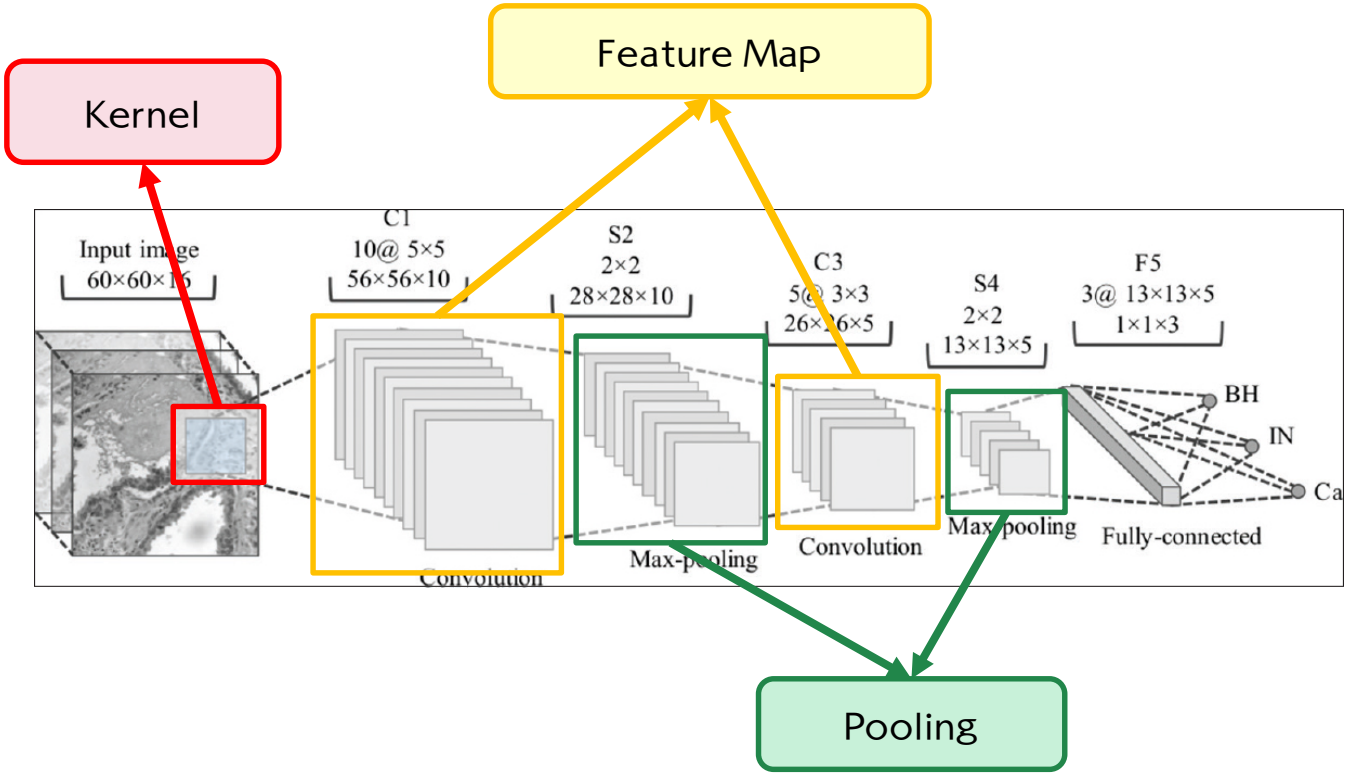


CNN Structure

CNN's Basic structure

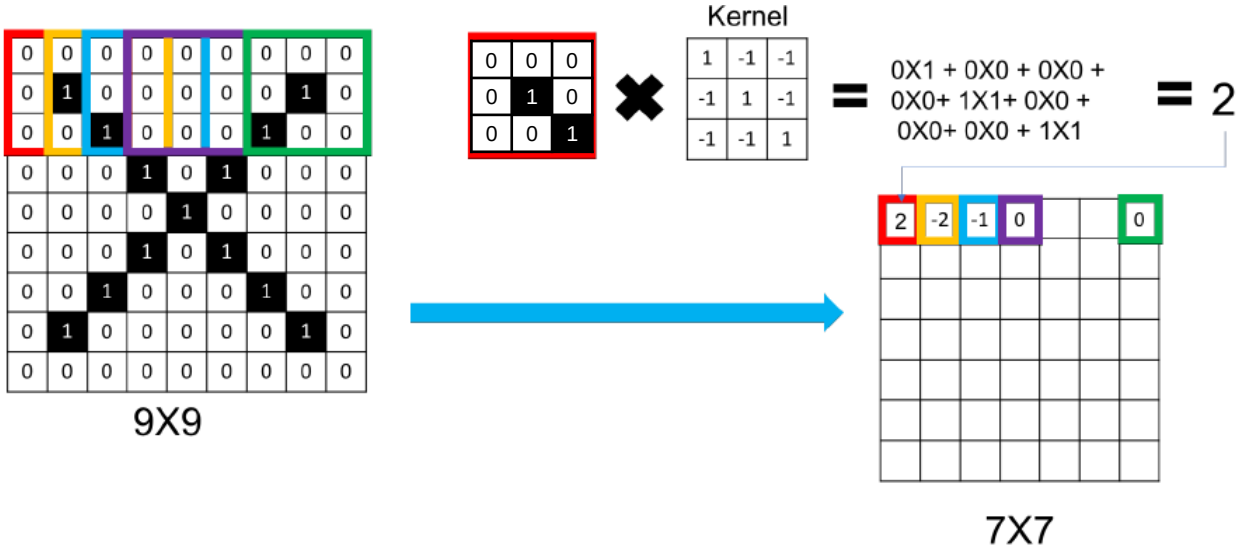


CNN's Basic structure



Convolution Calculation

Kernel : Local feature를 추출하는 weight 값

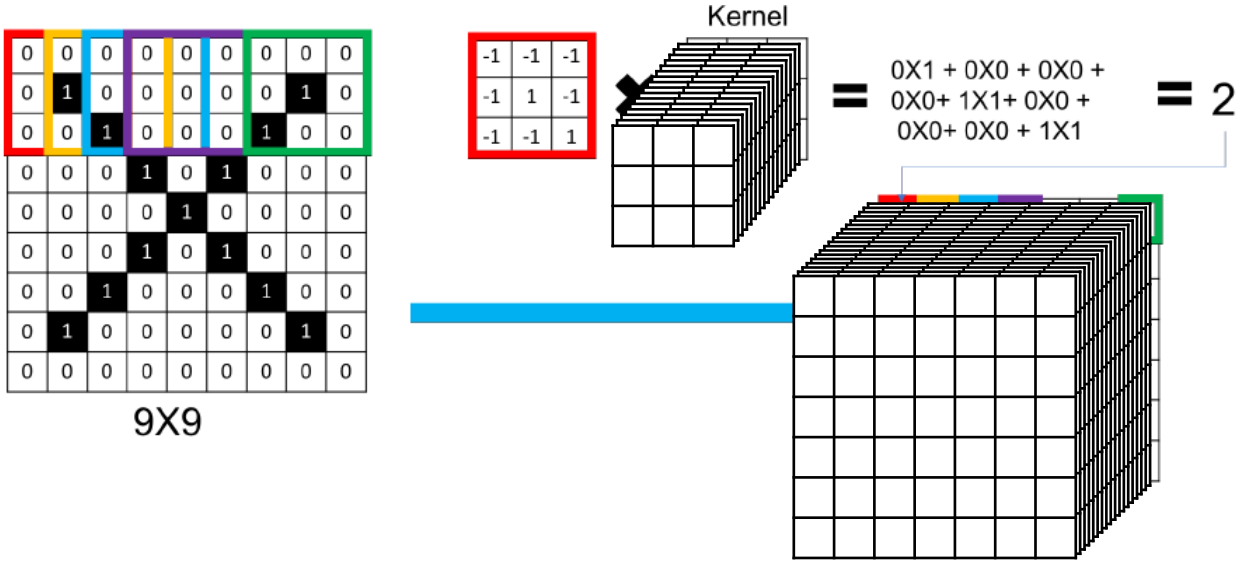


Convolution Calculation

Kernel : Local feature를 추출하는 weight 값

▶ Kernel이 많다 : 다양한 특징을 뽑아낼 수 있다

▶ 다음 번에 생성되는 feature map의 개수는 kernel의 개수와 동일

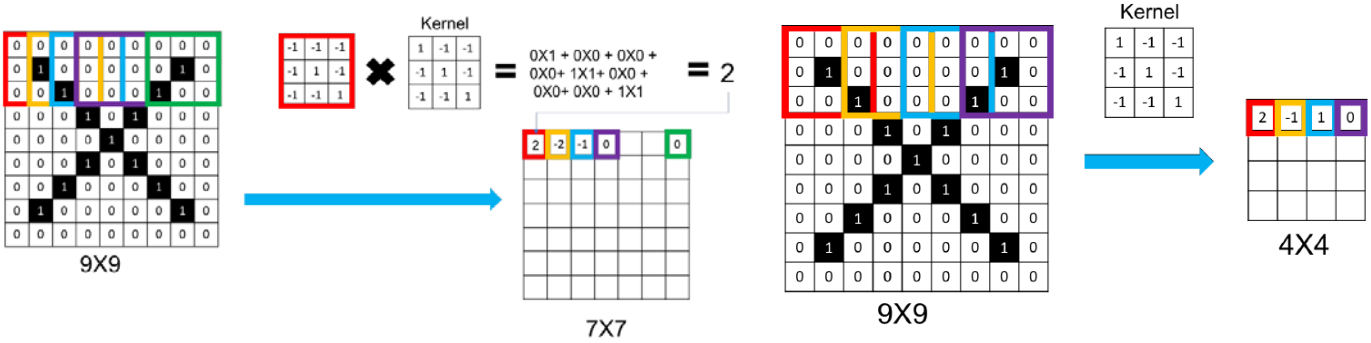


Stride

Convolution 계산 시, kernel이 움직이는 단위

▶ Stride 값이 작다 : 촘촘하게 데이터를 학습한다. Overfitting이 일어날 가능성이 높아진다.

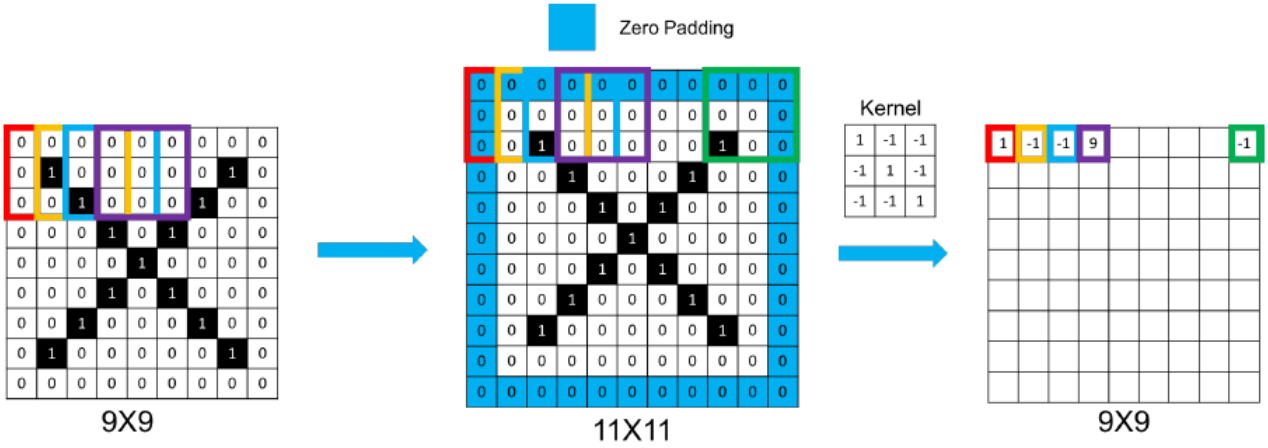
▶ Stride 값이 크다 : 느슨하게 데이터를 학습한다. 생성되는 Feature map의 크기가 작아진다. Underfitting이 일어날 가능성이 높아진다.



Padding

Convolution 연산 시 테두리에 추가적인 공간 할당

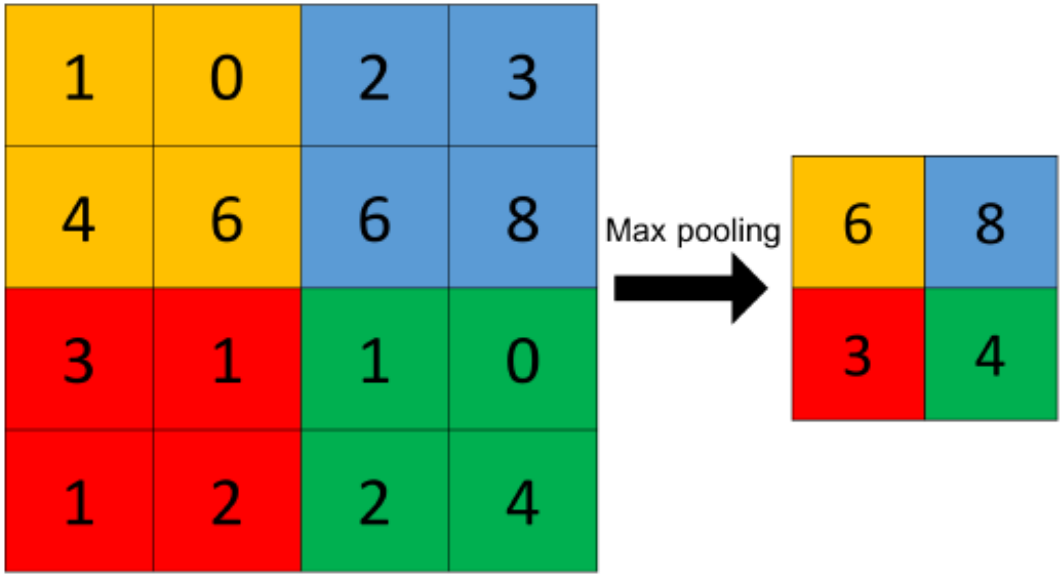
- ▶ 가장자리에 위치한 데이터도 균등하게 학습하기 위해
- ▶ 일반적으로 padding에는 잘못된 학습을 방지하기 위해 0을 넣음



Pooling Layer

파라미터를 줄여 overfitting을 방지

- ▶ Max Pooling : 가장 의미있는 데이터만을 선별해서 학습
- ▶ Average Pooling : 데이터를 평균내어서 학습



EX1. CNN with Natural Language

CNN을 언어 처리 관점에서 어떻게 바라볼 것인가?

지역적인 특징(Local Feature)을 끄집어내기 위해 사용

	A 식욕목	B 크기	C 생물	D 활동성	E 빈도
개	1	-0.8	1	0.8	0.7
곰	0.4	1	0.9	-0.5	0.1
고양이	-1	-1	1	-0.3	0.5
사자	-0.6	0.6	0.9	-0.6	0.2
호랑이	-0.7	0.8	0.8	0.3	0.1
… …					
걷다	0.1	0	-0.9	0.3	0.8
자다	0	0.1	-1	-1	0.7
달리다	-0.1	-0.1	-0.8	0.9	0.5

각 filter는 어떤 정보를 학습한 것일까?

1
-1
1
0
0

-1	0
1	0
1	-1
0	1
0	0

EX1. CNN with Natural Language

"개는 달리고 있고, 고양이는 자고 있다."

s = [개, 달리다, 고양이, 자다]

0	1	-0.1	-1	0	0
0	-0.8	-0.1	-1	0.1	0
0	1	-0.8	1	-1	0
0	0.8	0.9	-0.3	-1	0
0	0.7	0.5	0.5	0.7	0

X

f = Filter

-1	0
1	0
1	-1
0	1
0	0

=

c

2.8
-0.8
-1
-0.9
-0.2
0.9
-2.1
1
-0.9

Max Pooling

c_{pooled}

2.8
1

EX1. CNN with Natural Language

"곰은 자고 있고, 사자는 걷고 있다."

s = [곰, 자다, 사자, 걷다]

0	0.4	0	-0.6	0.1	0
0	1	0.1	0.6	0	0
0	0.9	-1	0.9	-0.9	0
0	-0.5	-1	-0.6	0.3	0
0	0.1	0.7	0.2	0.8	0

X

f = Filter

1
-1
1
0
0

-1	0
1	0
1	-1
0	1
0	0

=

c

0.3
-1.1
-0.3
-0.8

-1.4
1.5
-2.4
3.3
-1

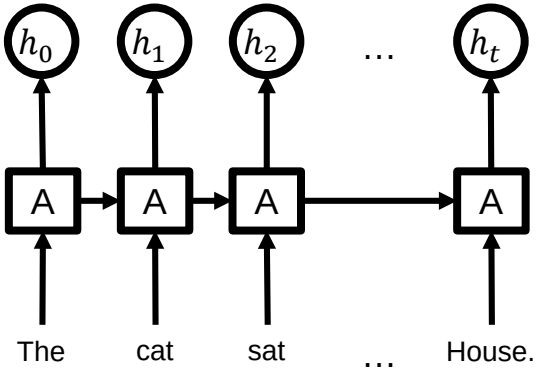
Max Pooling

c_{pooled}

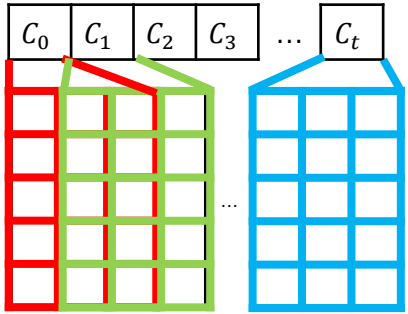
0.3
3.3

EX2. NLP with CNN vs RNN

Sequence Length



긴 sequence 정보를 이용하여
(이론상 무제한) 학습이 진행

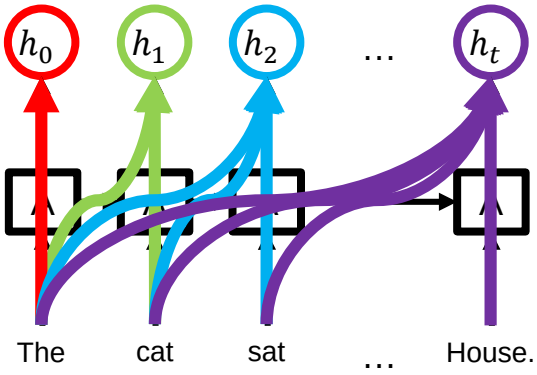


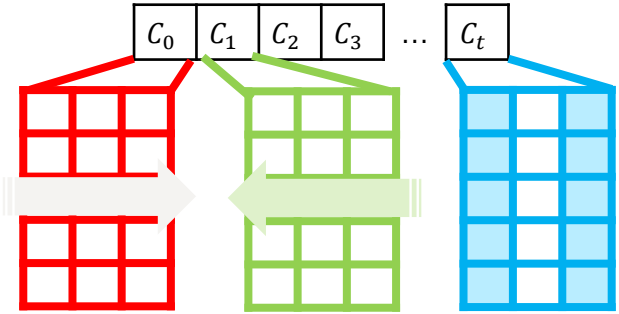
s = [the, cat, sat, on, … , the, dog, house]

정해 놓은 길이 (Filter의 크기) 내의 정보
만을 이용하여 학습이 진행

EX2. NLP with CNN vs RNN

Quality



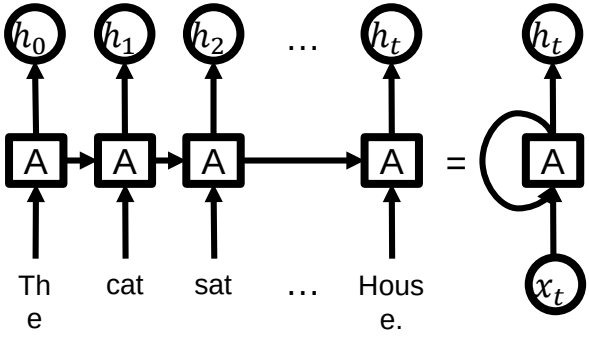


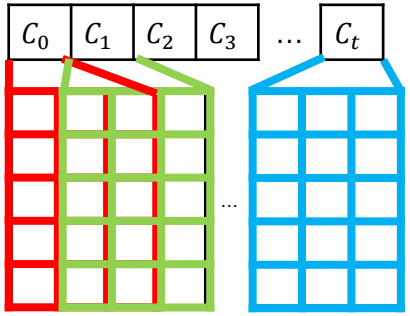
근본적으로 sequence에 따른 학습으로 부터 벗어날 수는 없음

Filter가 어떻게 학습되는지에 따라 다양한 단어의 조합으로 학습 가능

EX2. NLP with CNN vs RNN

Complexity





s = [the, cat, sat, on, … , the, dog, house]

한 sequence의 학습은 하나의 cell에서 발생하기 때문에 계산 복잡도가 낮음

여러 filter를 학습하며, 또한 각 filter의 연산 또한 RNN의 cell 연산보다 복잡

EX2. NLP with CNN vs RNN

결론

Filter의 크기가 무한하다고 가정한다면, CNN 또한 filter를 통해 sequence 정보를 학습할 수 있으며 filter에 따라서는 RNN보다 더 질 좋은 정보를 학습할 수 있음.

CNN과 RNN이 동일한 computing power로 학습한다고 가정할 경우, CNN보다 RNN이 더 효율적으로 정보를 학습할 수 있음.

short text를 사용할 경우에는, RNN 없이 CNN만을 이용해서도 합리적인 성능을 낼 수 있음.