딥러닝팀

1팀

김예찬
박시언
박윤아
정승민
김민

CONTENTS

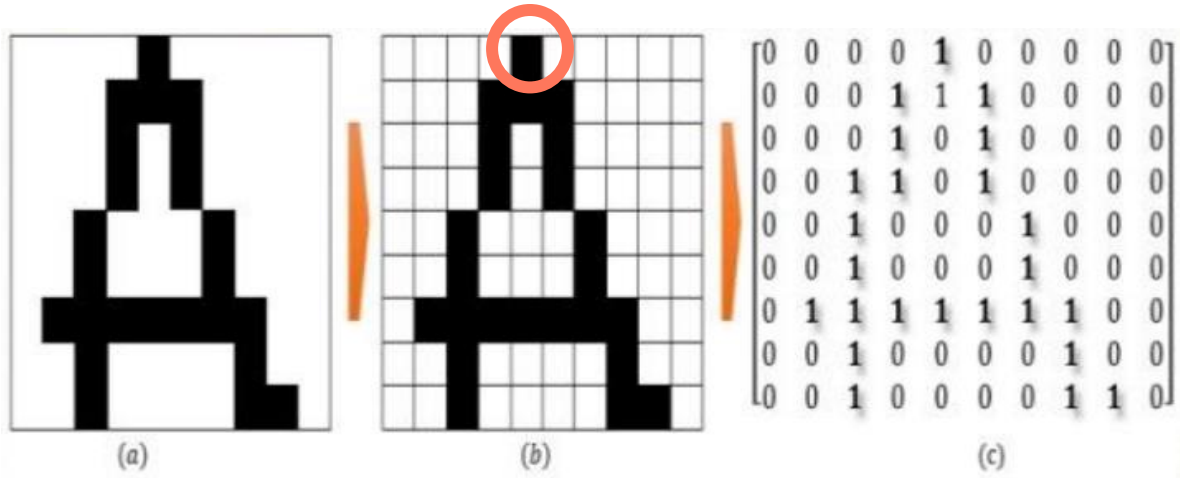
1

이미지 데이터의 특징

1 이미지 데이터의 특징

컴퓨터에서의 이미지 데이터

우리가 컴퓨터를 통해 보고 있는 모든 이미지는
픽셀(pixel)이라고 불리는 사각형의 점으로 이루어짐

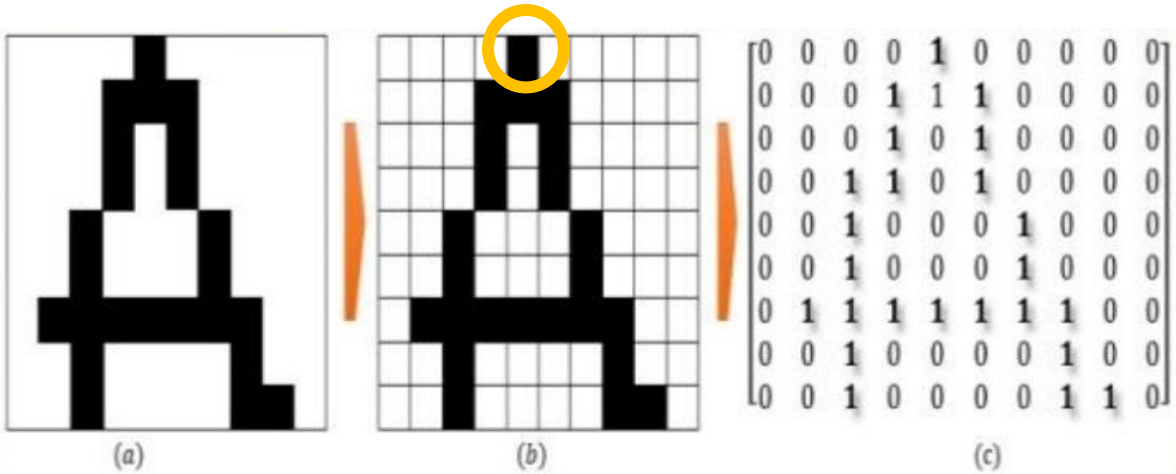


이미지 ▶ 픽셀 ▶ 행렬

1 이미지 데이터의 특징

컴퓨터에서의 이미지 데이터

만약 흑백이미지라면 픽셀을 표현할 때
우리가 컴퓨터를 통해 보고 있는 모든 이미지는
해당 위치가 검정색이면 1, 흰색이면 0으로 나타냄
픽셀(pixel)이라고 불리는 사각형의 점으로 이루어짐



이미지 ▶ 픽셀 ▶ 행렬

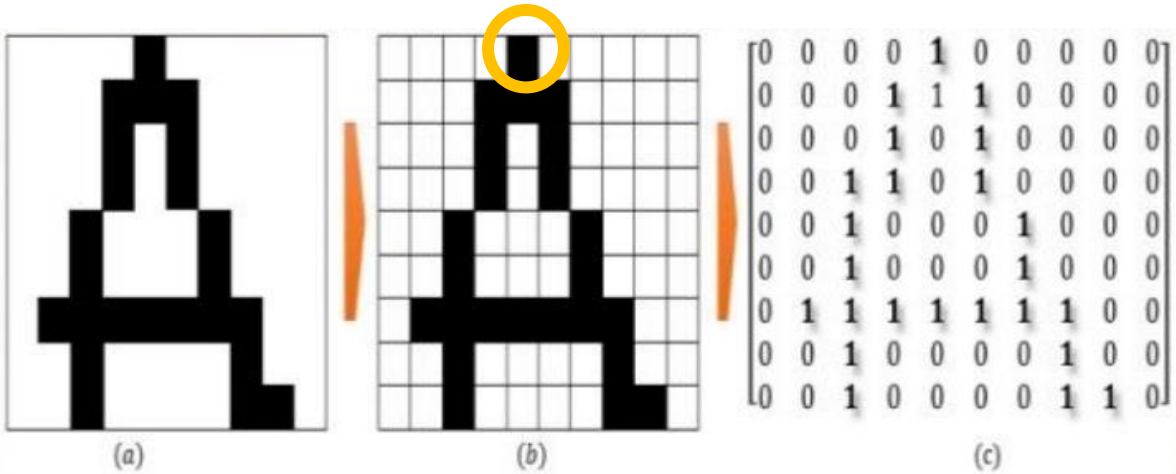
1 이미지 데이터의 특징

컴퓨터에서의 이미지 데이터

비트맵 이미지

숫자를 이용해 이미지를 행렬의 형태로 표현한 것

픽셀(pixel)이라고 불리는 사각형의 점으로 이루어짐

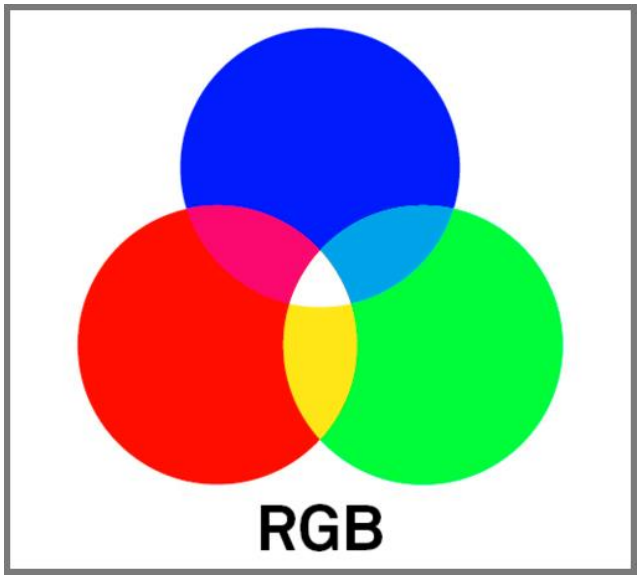


이미지 ▶ 픽셀 ▶ 행렬

1 이미지 데이터의 특징

채널

Channel



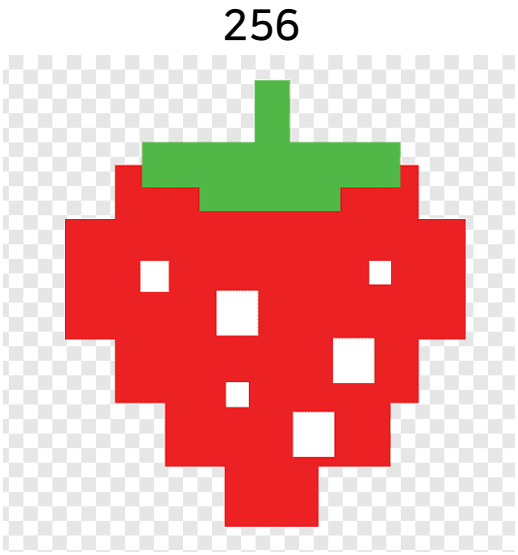
컬러 이미지는 빛의 3원색인 R, G, B 3개의 채널로 이루어짐

R, G, B 3개 채널의 범위로 색의 선명도 표현

1 이미지 데이터의 특징

채널

Channel



이미지의 가로와 세로를 구성하는
픽셀의 수가 각각 256개이고
R, G, B 세개의 색으로 이루어진 경우

> (256, 256)크기인 3개의 행렬로 표현

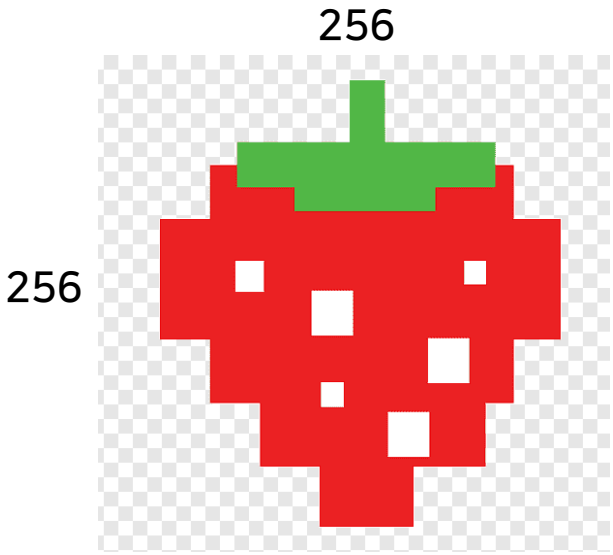
이미지 데이터 표현 ▶ (height, width, channel)

ex. *Pytorch에서는 (C, H, W) 순으로 (3, 256, 256)이라고 표현*

1 이미지 데이터의 특징

채널

Channel



이미지의 가로와 세로를 구성하는
픽셀의 수가 각각 256개이고
R, G, B 세개의 색으로 이루어진 경우

> (256, 256,3)크기인 행렬로 표현

이미지 데이터 표현 ▶ (height, width, channel)

ex. *Pytorch에서는 (C, H, W) 순으로 (3, 256, 256)이라고 표현*

2

CNN

CNN

Convolutional Neural Network

기존 신경망은 1차원 벡터만을 input으로 받아들임

2차원 이상의 데이터를 1차원 벡터 형태로 변경해야함

하지만, 이 과정에서 이미지가 가지고 있는 **공간적인 정보가 손실됨**

CNN

Convolutional Neural Network

기존 신경망은 1차원 벡터만을 input으로 받아들임

2차원 이상의 데이터를 1차원 벡터 형태로 변경해야함

하지만, 이 과정에서 이미지가 가지고 있는 **공간적인 정보가 손실됨**

CNN

Convolutional Neural Network

기존 신경망은 1차원 벡터만을 input으로 받아들임

2차원 이상의 데이터를 1차원 벡터 형태로 변경해야함

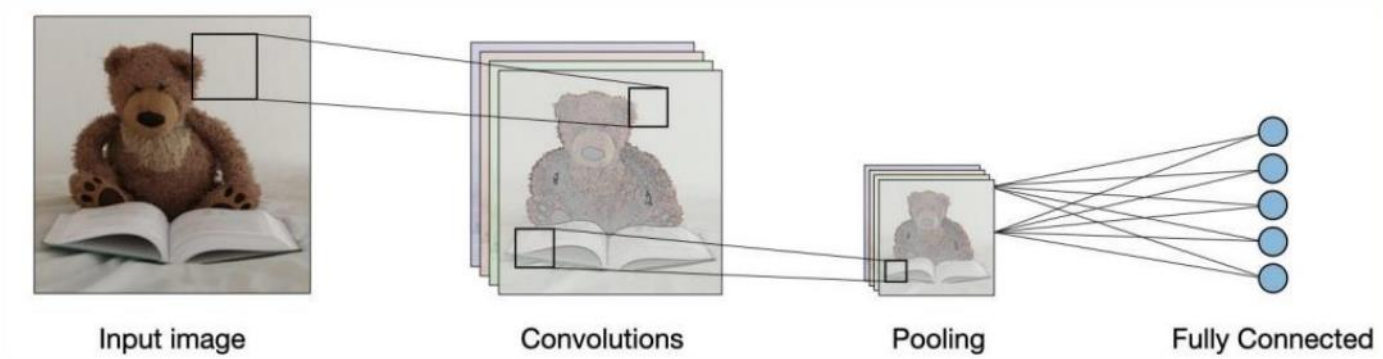
하지만, 이 과정에서 이미지가 가지고 있는 **공간적인 정보가 손실됨**

CNN

Convolutional Neural Network

강점

공간 정보의 보존



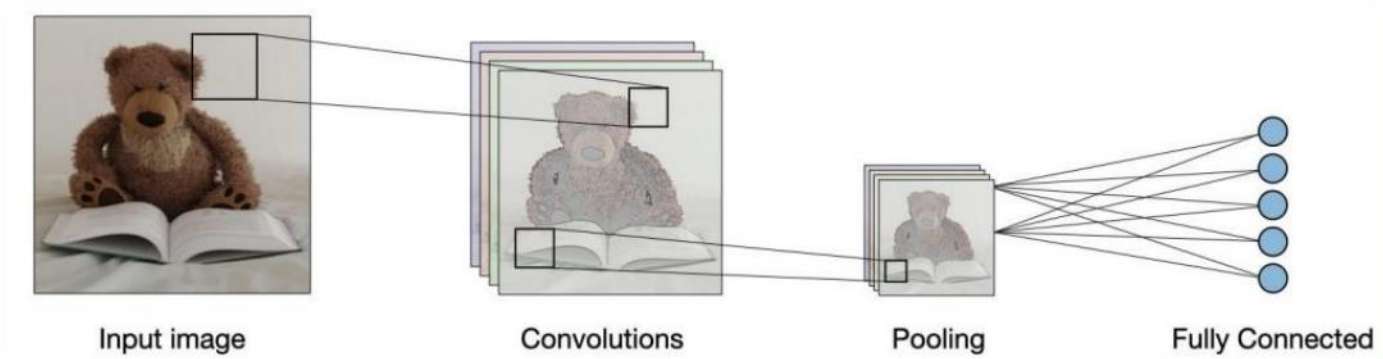
CNN

Convolutional Neural Network

강점

공간 정보의 보존

Convolutions 단계의
검은색 상자와 연결됨



검은색 상자가 표시

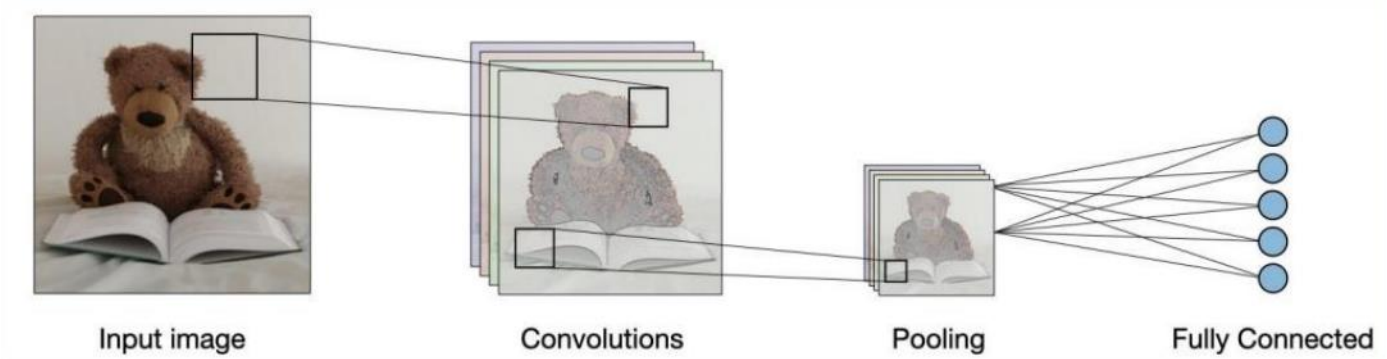
CNN

Convolutional Neural Network

강점

공간 정보의 보존

Convolutions 단계의
검은색 상자와 연결됨



검은색 상자가 표시

검은색 상자의 위치가
다음 층에서도 **그대로 유지**

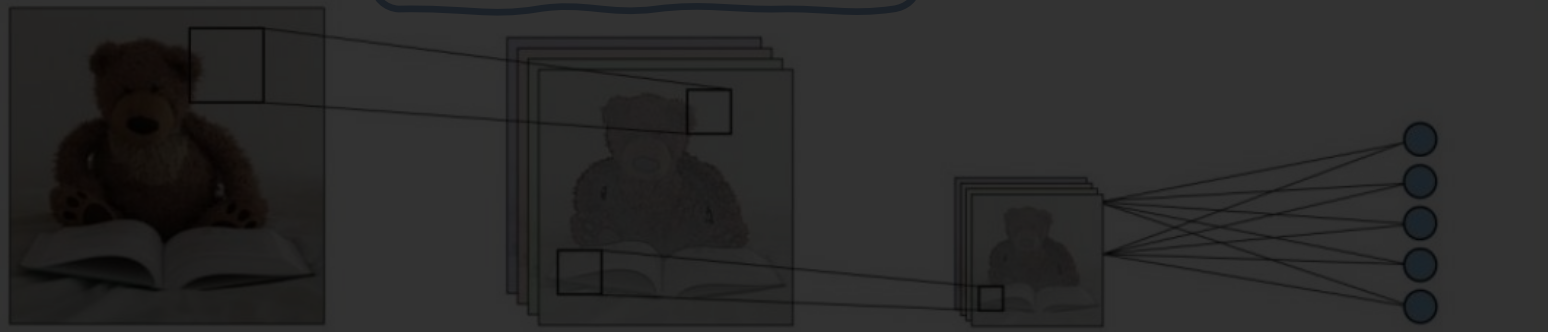
CNN (Convolutional Neural Network)

CNN의 강점

공간 정보의 보존

즉 데이터의 공간정보가 그대로 보존됨

Convolutions 단계의
검은색 상자와 연결됨



검은색 상자가 표시

검은색 상자의 위치가
다음 층에서도 그대로 유지

CNN (Convolutional Neural Network)

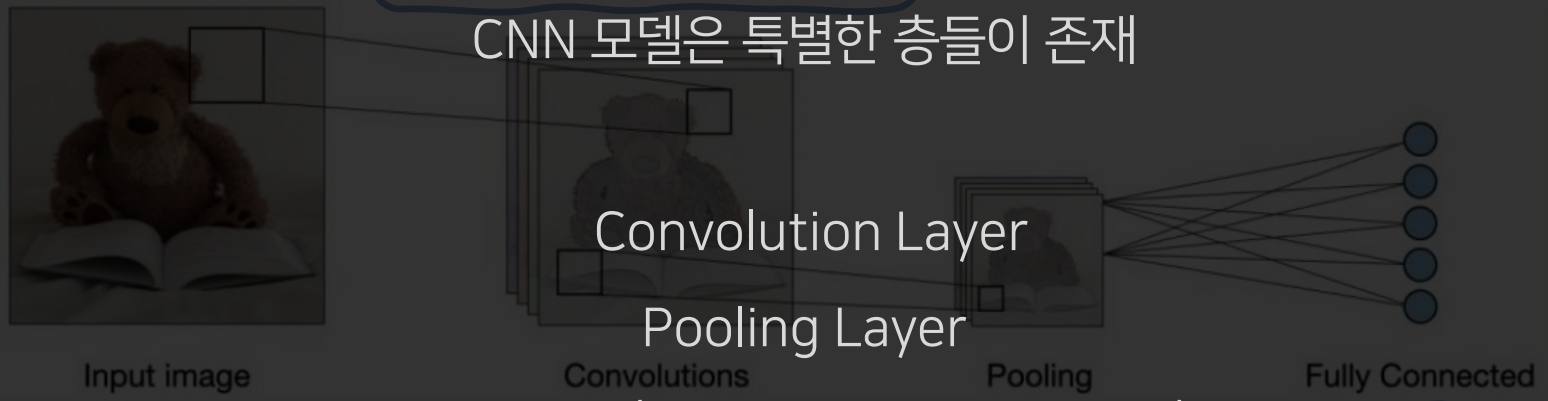
CNN의 강점

공간 정보의 보존

즉 데이터의 공간정보가 그대로 보존됨

Convolutions 픽셀의
검은색 상자와 연결됨

CNN 모델은 특별한 층들이 존재



Convolution Layer

Pooling Layer

FC Layer(Fully Connected Layer)

검은색 상자가 표시

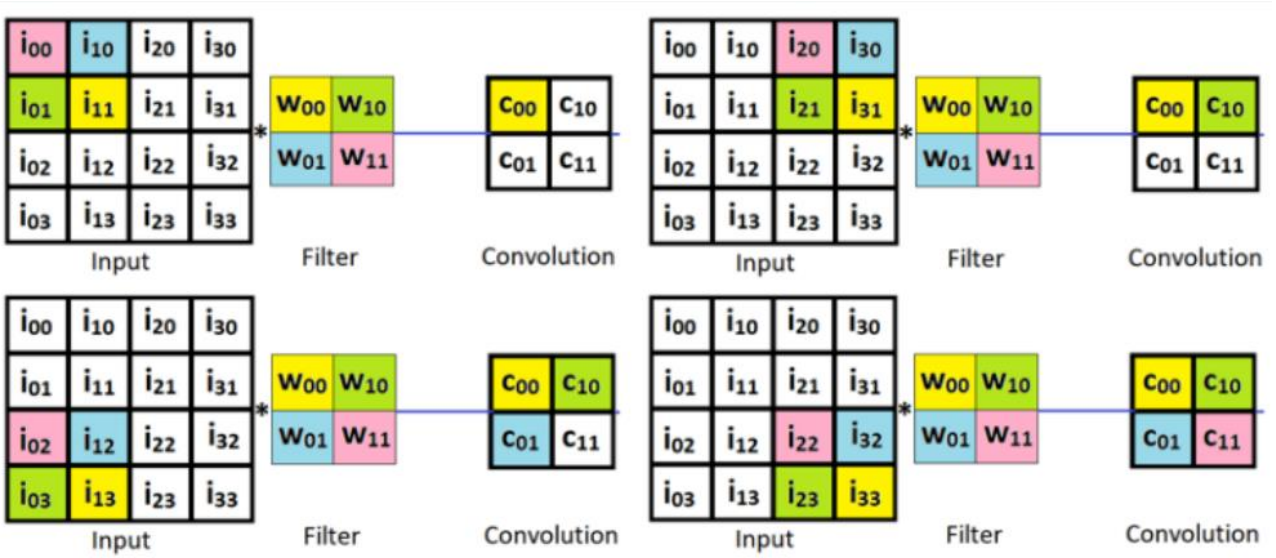
검은색 상자의 위치가
다음 층에서도 그대로 유지

Convolutional Layer

작동 원리

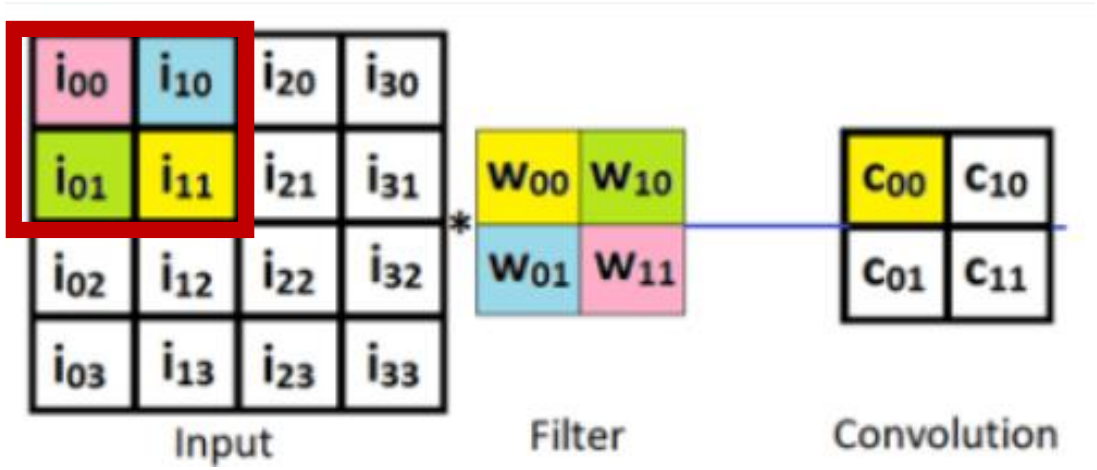
입력 데이터의 공간 정보 보존 , 입력 데이터의 특징 추출

작동 방식



Convolutional Layer

작동 원리



위 사진에서는 같은 색으로 칠해진 i와 w가 곱해진 후 더해지는 연산

필터

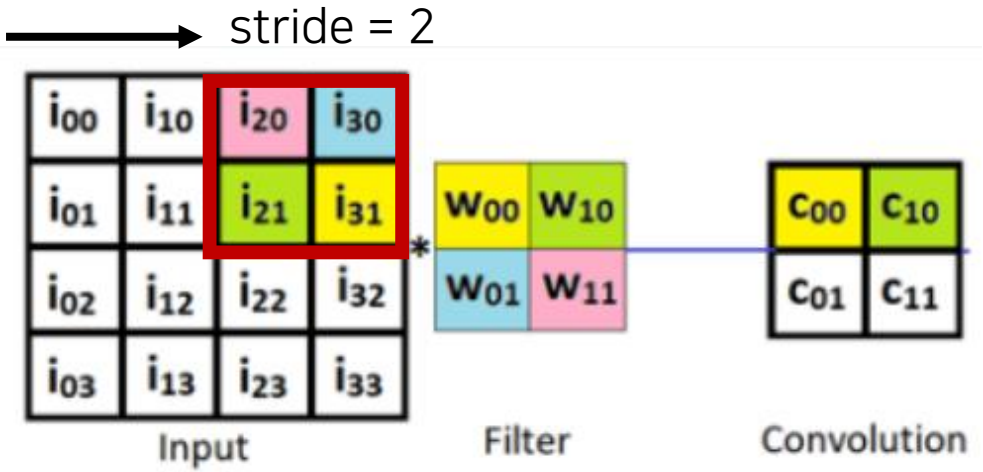
입력으로 이미지가 Convolution Layer에 들어오게 되면,

필터는 input에서 자신의 크기만큼의 부분과 가중합하는 역할

$$i_{00} \times w_{11} + i_{10} \times w_{01} + i_{01} \times w_{10} + i_{11} \times w_{00} = c_{00}$$

Convolutional Layer

작동 원리



위 예시에서는 첫번째 그림에서 두번째 그림으로 넘어갈 때 필터가 2칸을 움직임

Stride

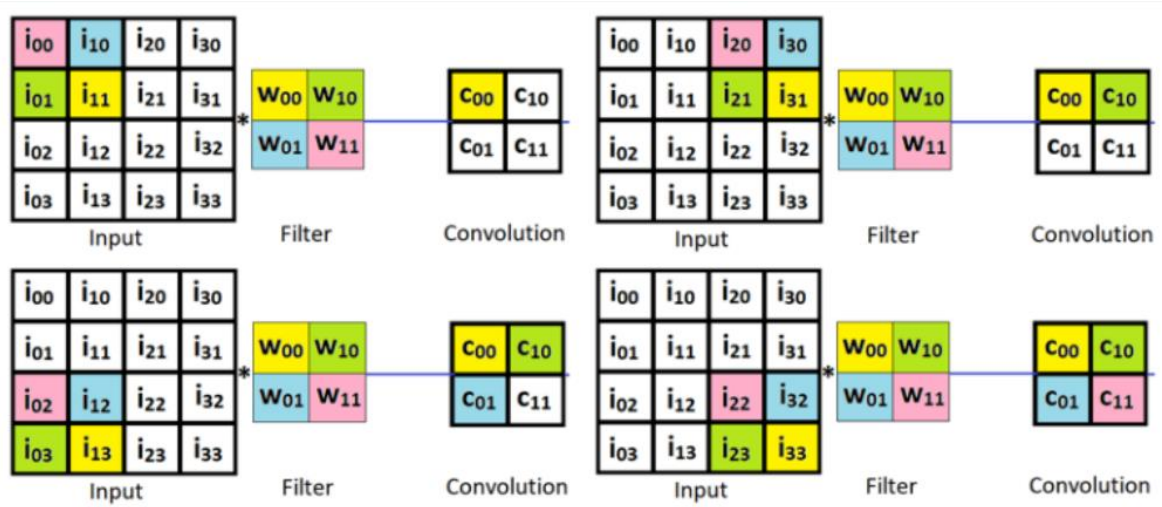
이 (2, 2)사이즈의 필터는 (4, 4)사이즈의 input을 따라

오른쪽으로 움직임

. 이것을 stride가 2라고 표현

Convolutional Layer

작동 원리



결과적으로 위 Convolution Layer예시에서는 (4, 4)의 input에 대해
(2, 2)의 필터가 stride=2 로 움직여 (2, 2)의 결과를 반환
여기서 일차적으로 반환된 결과를 Feature Map

Convolutional Layer

Feature Map

Convolution layer를 거쳐 반환된 결과

Feature map의 형태를 결정하는 하이퍼 파라미터들

하이퍼 파라미터	Pytorch 표현	Tensorflow 표현	역할
Input Channel	in_channels	미리 지정	입력 데이터의 채널 개수 지정
Output Channel	out_channels	filters	출력의 채널 개수 지정
필터 사이즈	kernel_size	kernel_size	필터의 크기 지정
Stride	stride	strides	필터의 이동 간격 지정
Padding	padding	padding	입력 데이터 주변에 붙일 padding 수 지정

Convolutional Layer

Input channel, Output channel

Input Channel

입력 이미지의 채널의 수

- 컬러 이미지 데이터: R, G, B 3개의 채널
- 흑백 이미지 데이터 : 1개의 채널

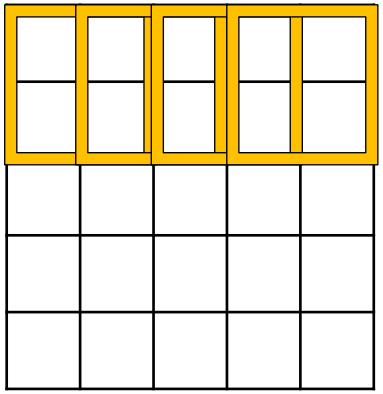
Output Channel

convolution layer가 반환하는 채널의 수

= convolution layer의 필터의 개수

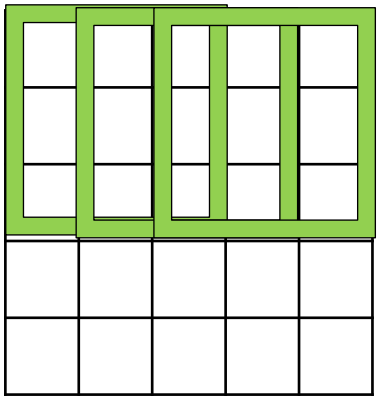
Convolutional Layer

필터 사이즈



2x2 필터

▶ 4x4 feature map



3x3 필터

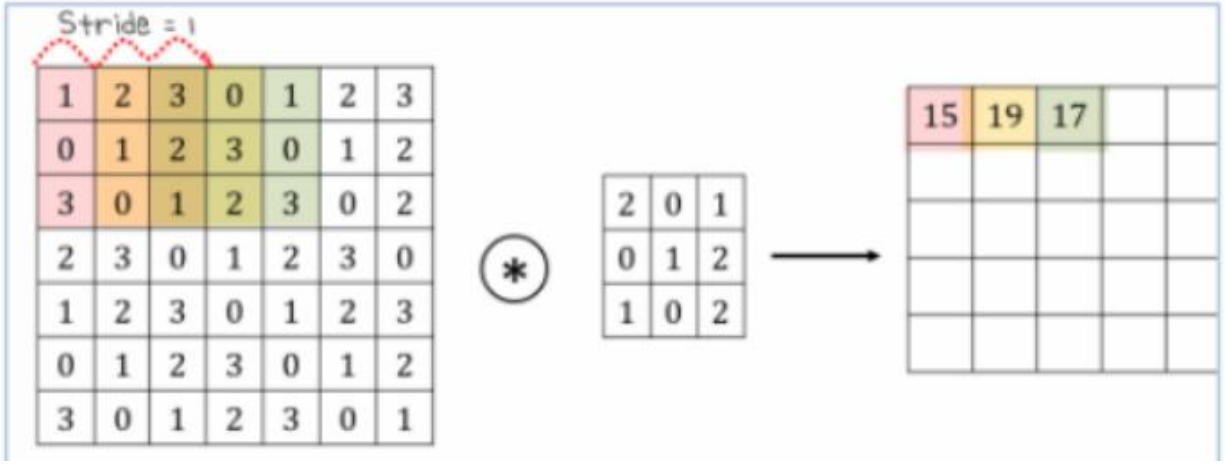
▶ 3x3 feature map

Fillter Size

- 필터의 크기가 클수록 반환되는 feature map의 크기는 작아짐
- 일반적으로 홀수로 지정

Convolutional Layer

stride



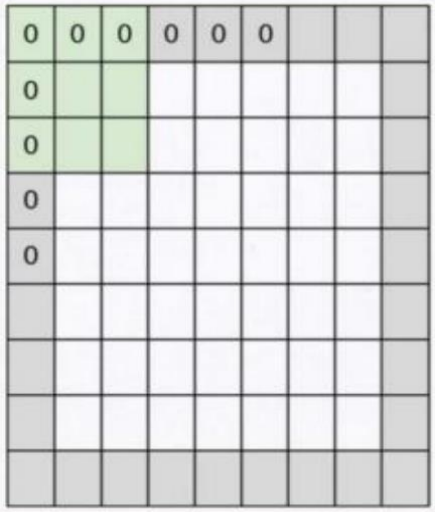
Stride

필터가 이동하는 간격

- stride가 작을수록 입력 데이터에 대해 연산이 더 많이 이뤄짐
  - Stride가 작을수록 feature map의 크기가 커짐

Convolutional Layer

padding



Zero padding

입력 데이터 중심부 근처의 값들은
필터에 여러 번 통과되지만,
가장자리의 값들은 상대적으로 적게 통과

Padding

원본 데이터의 가장자리에 새로운 값을 추가하여
가장자리의 값들도 필터를 여러 번 통과할 수 있게 조정

Convolutional Layer

Feature Map 크기 계산

$$O_n = \frac{l_n + 2P - F}{S} + 1$$

O_n = 출력의 가로 길이
l_n = 입력의 가로 길이
P = padding의 크기
F = 필터의 크기
S = stride의 크기

Q. E(32, 32, 3)의 입력 데이터, (4, 4)의 필터 10개 적용,
stride=2, padding=1 Feature map의 크기는?

Convolutional Layer

Feature Map 크기 계산

$$O_n = \frac{l_n + 2P - F}{S} + 1$$

O_n = 출력의 가로 길이
l_n = 입력의 가로 길이
P = padding의 크기
F = 필터의 크기
S = stride의 크기

Q. E(32, 32, 3)의 입력 데이터, (4, 4)의 필터 10개 적용,
stride=2, padding=1 Feature map의 크기는?

Convolutional Layer

Feature Map 크기 계산

Q. E(32, 32, 3)의 입력 데이터, (4, 4)의 필터 10개 적용,
stride=2, padding=1 Feature map의 크기는?

$$O_n = \frac{l_n + 2P - F}{S} + 1$$

일단 필터가 총 10 개이기 때문에
output channel은 10
s는 2, p는 1, i는 32, f는 4

$$\frac{32 + 2 - 4}{2} + 1 = 16$$

최종적으로 Feature Map의
사이즈는 (10, 16, 16)

Pooling Layer

pooling layer의 종류

- Max Pooling
- Average Pooling
- Min Pooling

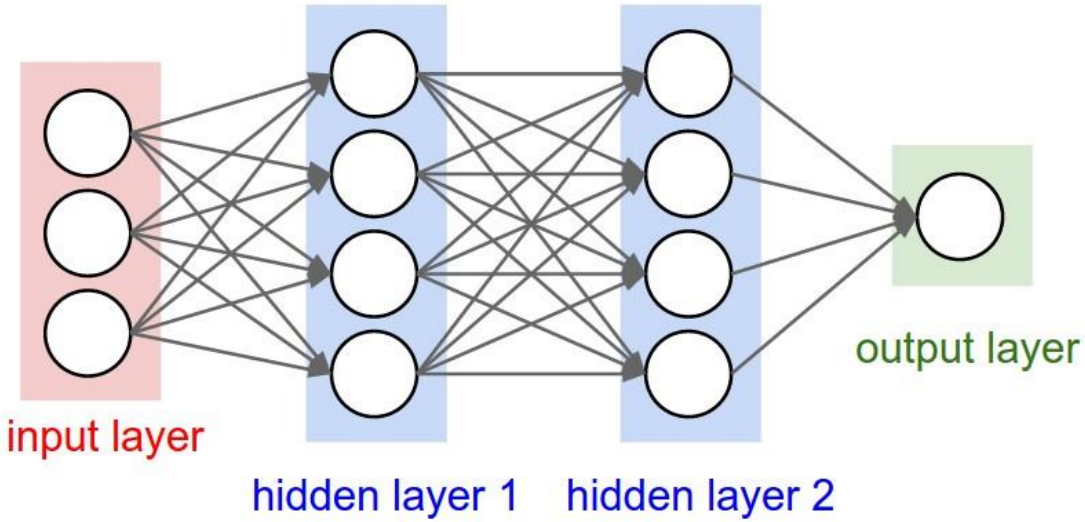
Pooling Layer

입력 데이터의 중요한 특징을 강조 & 데이터의 크기 축소

- Convolution layer와 비슷한 역할
- 가중치 없이 중요한 특징 강조

FC Layer

Fully Connected Layer



FC Layer

여러 개의 퍼셉트론이 겹쳐있는 형태

최종 단계에서 라벨 예측을 진행할 때 사용되는 층

FC layer로 2차원 이상 데이터 입력 시 벡터로의 변환 필요

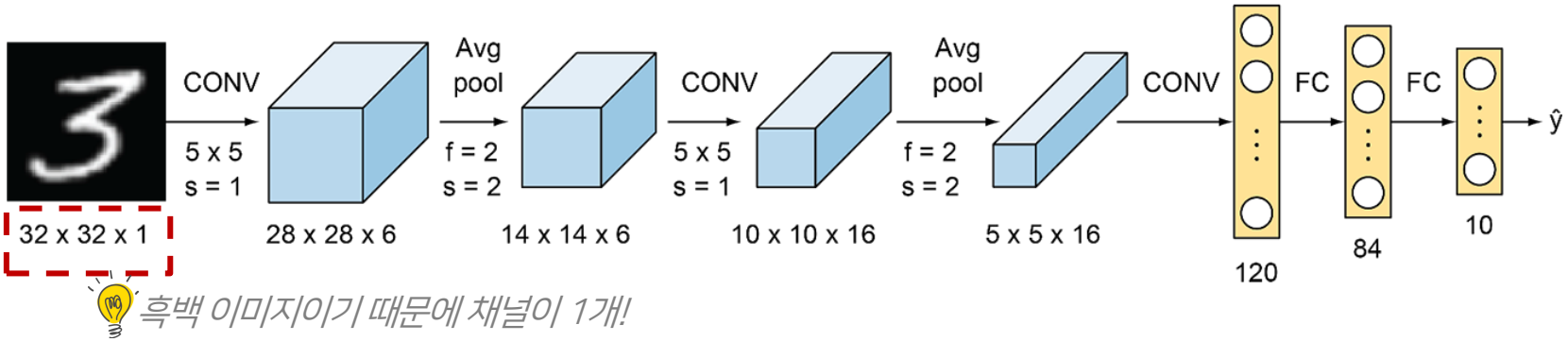
3

CNN의 발전 과정

CNN의 발전 과정

LeNet-5

소개



-CNN의 태동 같은 모델

- Convolution Layer와 Pooling Layer를 교차하여 통과시킨 후

2개의 FC Layer에 통과시켜 손글씨를 분류 하는 모델

CNN의 발전 과정

LeNet-5

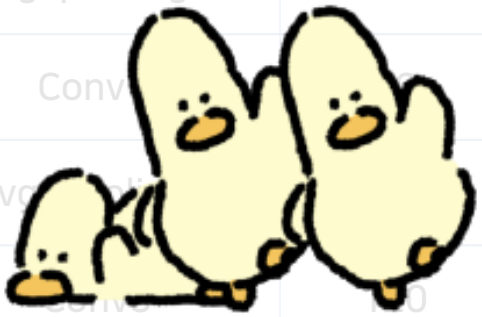
Layer	#filters/ neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully connected 2	-	-	-	10	Softmax

CNN의 발전 과정

LeNet-5

Layer	#filters/ neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv		5 * 5	1	10 X 10 X 16	tanh
Avg.		2 * 2	2	5 X 5 X 16	
Conv	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully connected 2	-	-	-	10	Softmax

활성화 함수로 tanh 사용!



CNN의 발전 과정

LeNet-5

Layer	#filters/ neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv1	6	5 * 5	1	28 X 28 X 6	Tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	Tanh
Fully connected 2	-	-	-	10	Softmax

마지막에는 Softmax로
10개로 구분!

3 CNN의 발전 과정

LeNet-5

숫자의 종류가 0~9까지

총 10개이므로

최종적으로 10으로 출력

✓ 마지막에는 Softmax로

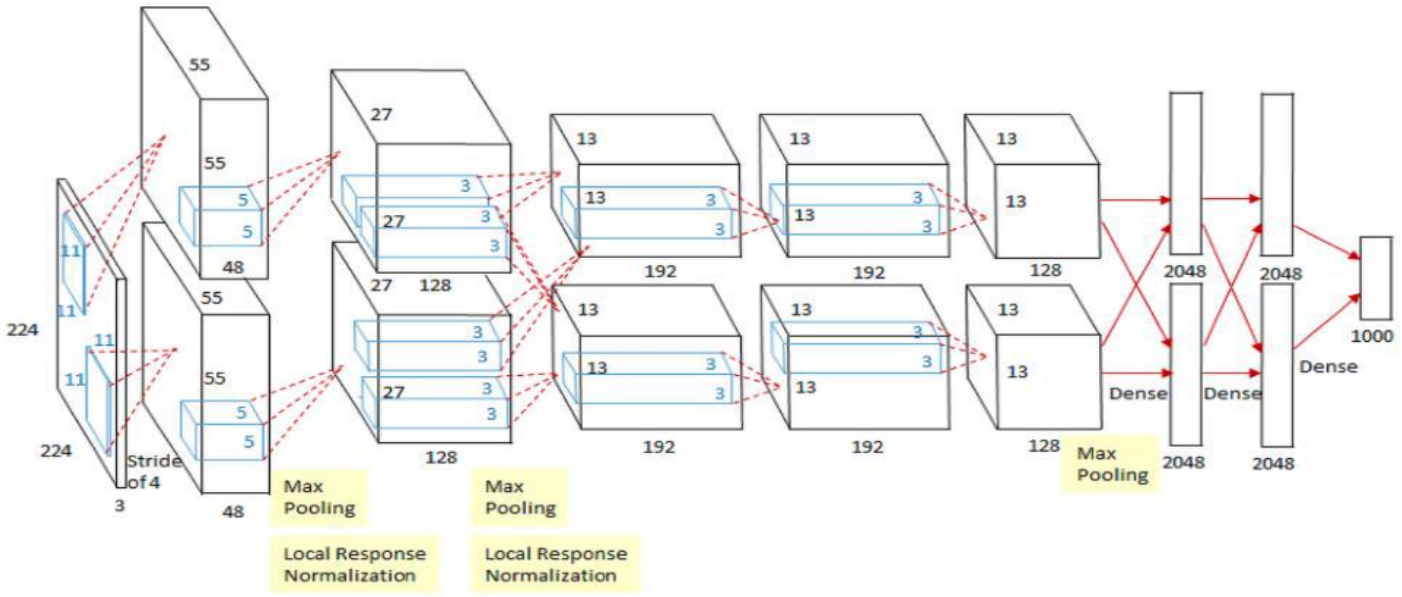
10개로 구분!

10	Softmax

CNN의 발전 과정

AlexNet

소개



~~Alex Net~~

2012년 ILSVRC에서 큰 격차로 우승

데이터의 병렬적 구조가 특징

그 시절 GPU의 한계 때문에…

CNN의 발전 과정

AlexNet

LeNet-5 VS AlexNet

	LeNet-5	AlexNet
활성화 함수	tanh	ReLU
Pooling Layer	Average Pooling	Max pooling
Drop Out	X	O
Data Agmention	X	O

CNN의 발전 과정

AlexNet

LeNet-5 VS AlexNet

	LeNet-5	AlexNet
활성화 함수	tanh	ReLU
Pooling Layer	Average Pooling	Max pooling
Drop Out	X	O
Data Augumention	X	O

ReLU를 사용함으로써
연산량은 획기적으로 줄이고
이미지의 특징을
효과적으로 뽑아냄

자세한 내용은 1주차 클린업 참고!

CNN의 발전 과정

AlexNet

LeNet-5 VS AlexNet

	LeNet-5	AlexNet
활성화 함수	tanh	ReLU
Pooling Layer	Average Pooling	Max pooling
Drop Out	X	O
Data Augumention	X	O

Max Pooling을 사용함으로써
이미지의 특징을
효과적으로 뽑아냄

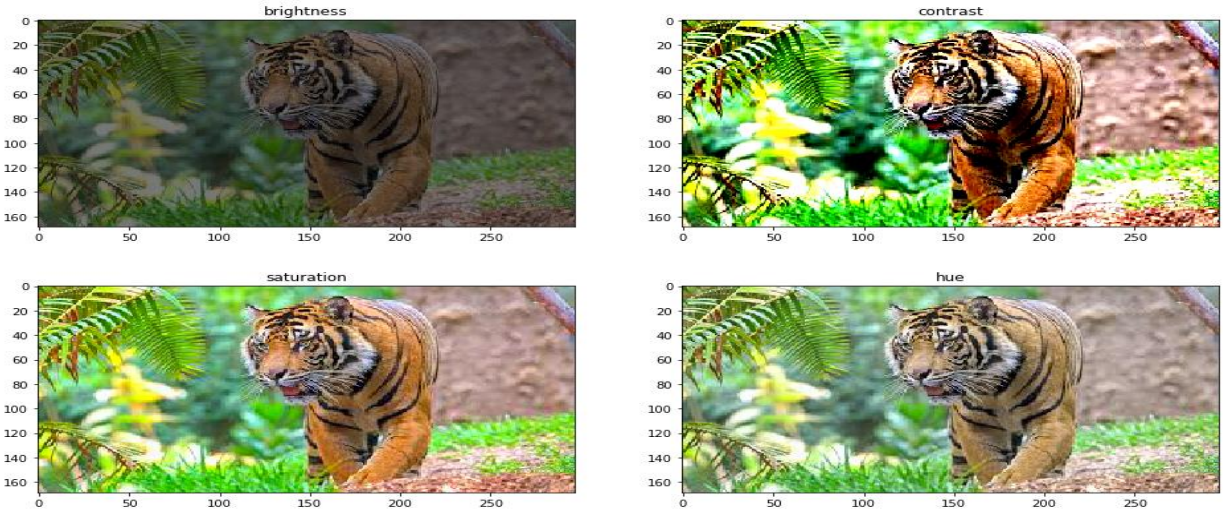
CNN의 발전 과정

AlexNet

Data Augmentation (데이터 증강 기법)

CNN모델은 사물의 위치에 강건하지 못하기 때문에

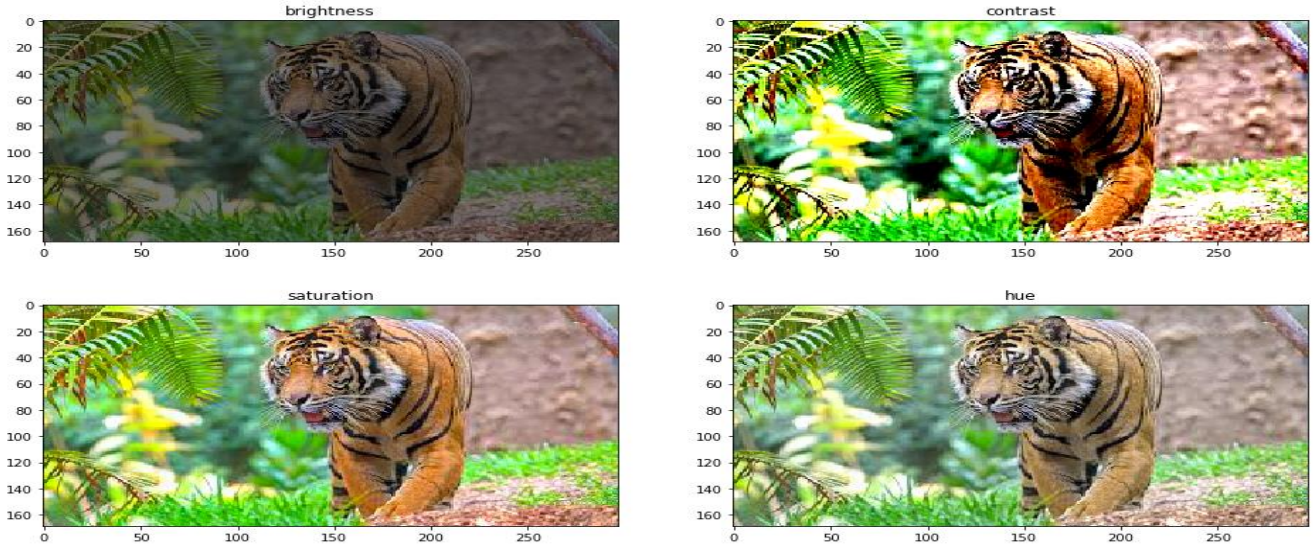
이를 해결하기 위해 Data Augmentation을 사용



CNN의 발전 과정

AlexNet

Data Augmentation (데이터 증강 기법)



데이터 증강방법으로 하나의 이미지를 사용해

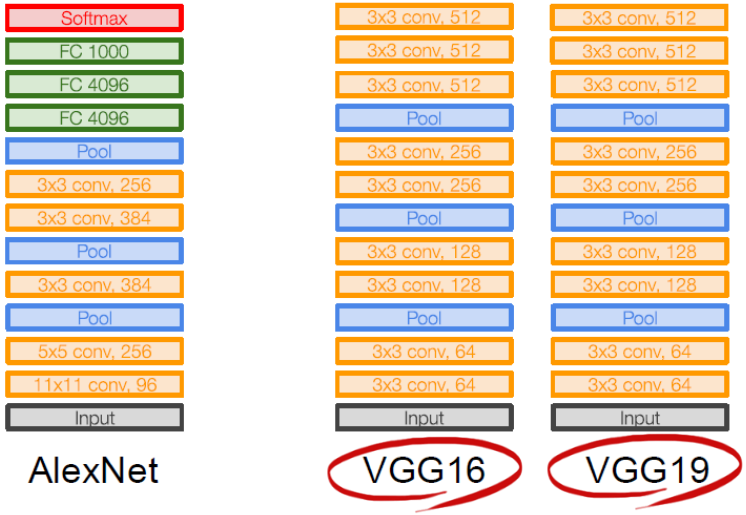
여러 비슷한 이미지를 만들어내는 것

> 실제 데이터 보다 많은 양의 데이터를 학습에 사용

CNN의 발전 과정

VGGNet

소개



VGGNet

간단한 구조임에도

2014 ILSVRC 준우승

CNN에서 Layer를

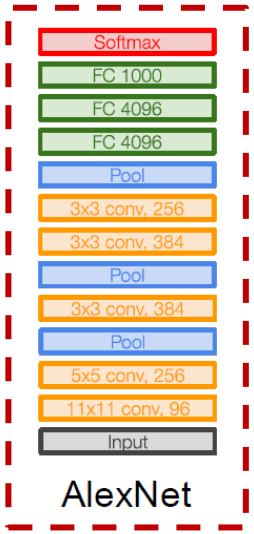
더 깊게 쌓을 수 있다면

더 높은 성능을 보일 수 있음을 증명

CNN의 발전 과정

VGGNet

소개



AlexNet의 경우

(11, 11), (5, 5)를 거쳐

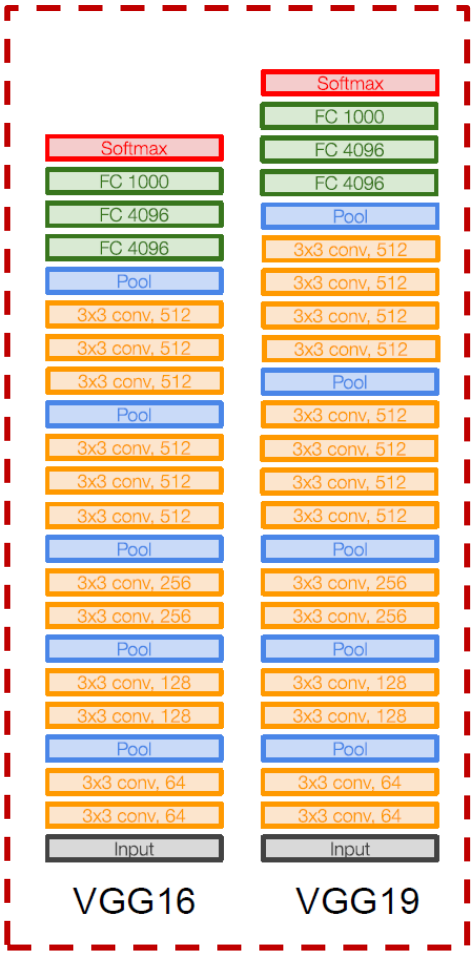
마지막에는 (3,3)까지

필터의 사이즈가 계속 줄어듦

CNN의 발전 과정

VGGNet

소개



VGGNet의 경우

stride = 1, padding =1인

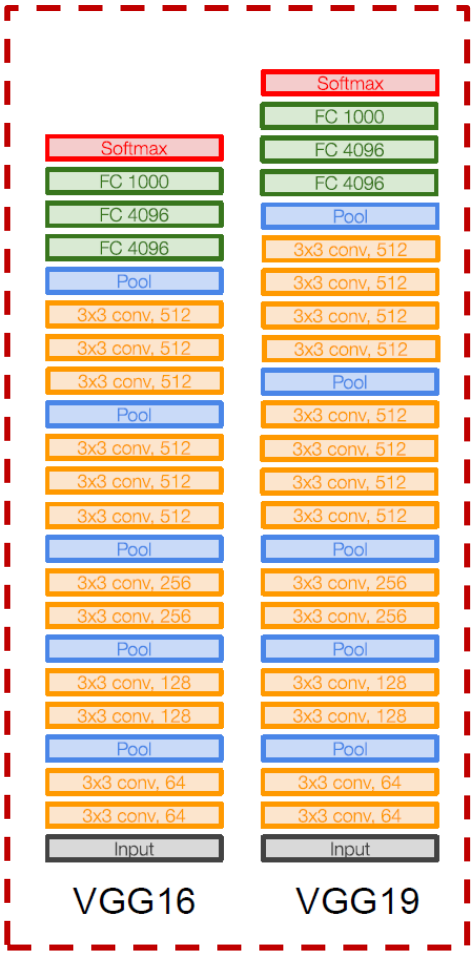
(3,3)의 필터만을

처음부터 끝까지 사용

CNN의 발전 과정

VGGNet

소개



마찬가지로

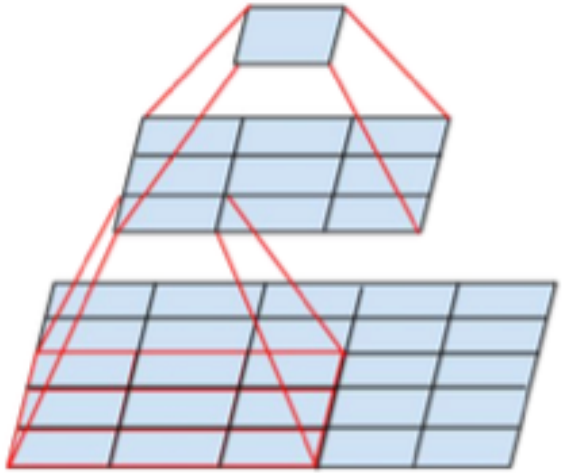
Pooling Layer 또한

Max Pooling만을 사용

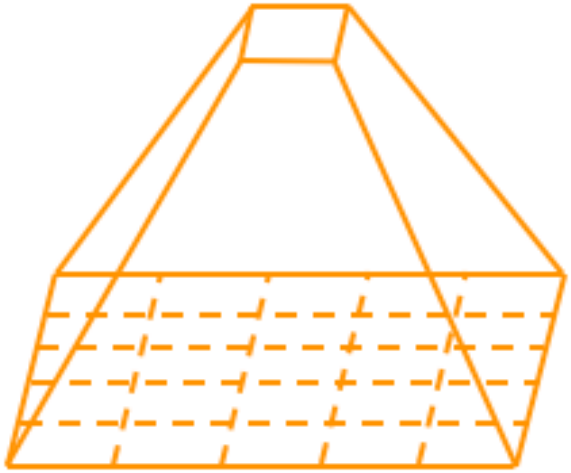
CNN의 발전 과정

VGGNet

Convolution Layer



two successive
3x3 convolutions



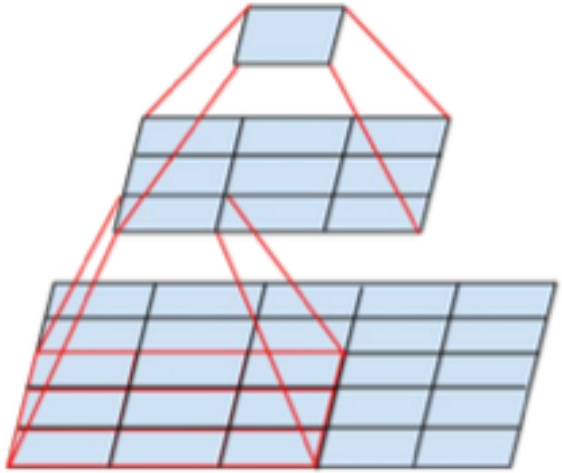
5x5 convolution

같은 (5,5) 데이터를 (1,1)로 변환

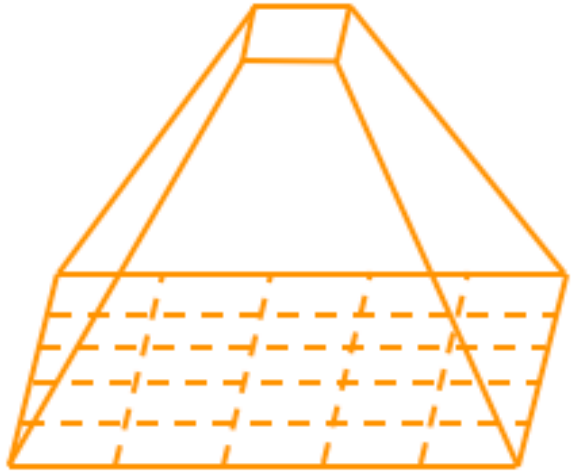
CNN의 발전 과정

VGGNet

Convolution Layer



two successive
3x3 convolutions



5x5 convolution

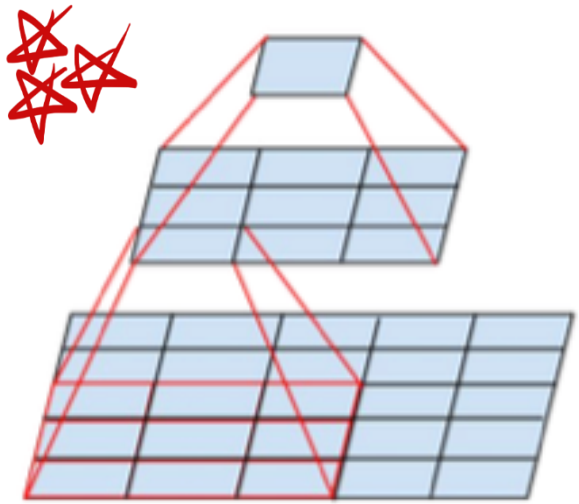
(5, 5)의 데이터에 두 번의 (3, 3) Convolution Layer를 적용한 결과

같은 데이터에 한 번의 Convolution Layer를 적용한 결과

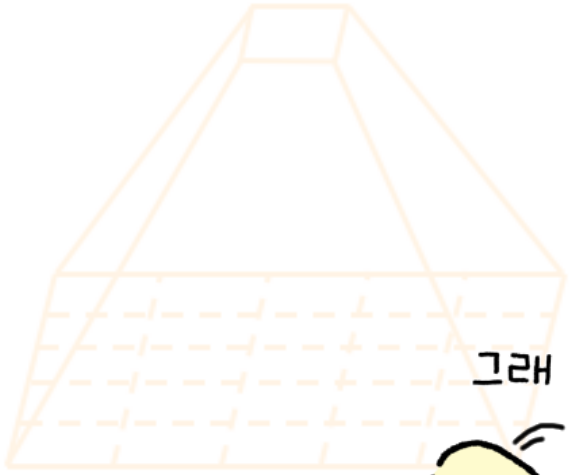
CNN의 발전 과정

VGGNet

Convolution Layer



two successive
3x3 convolutions



5x5 convolution

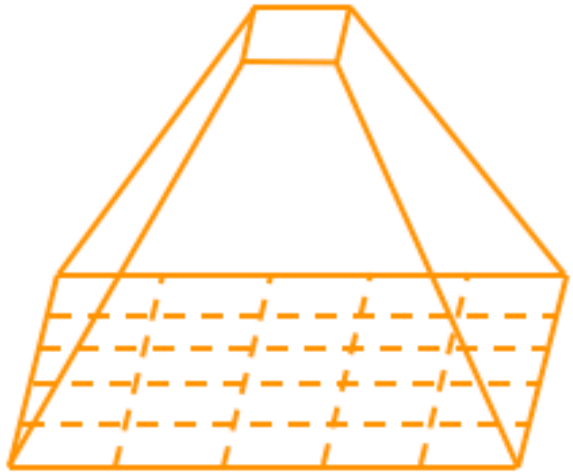
두번의 Convolution Layer 를 통과하여

비선형성을 **2번** 추가 가능!

CNN의 발전 과정

VGGNet

Convolution Layer



5x5 convolution

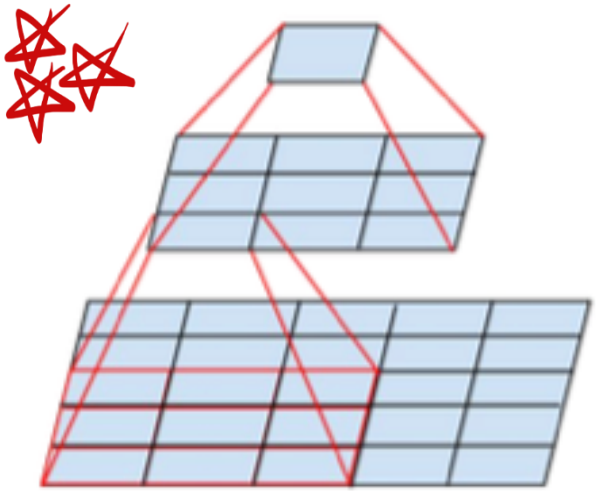
채널의 개수가 C 개일 때

파라미터 수 $5^2 \times C$

CNN의 발전 과정

VGGNet

Convolution Layer



two successive
3x3 convolutions

채널의 개수가 C 개일 때

파라미터 수 $2 \times 3^2 \times C$

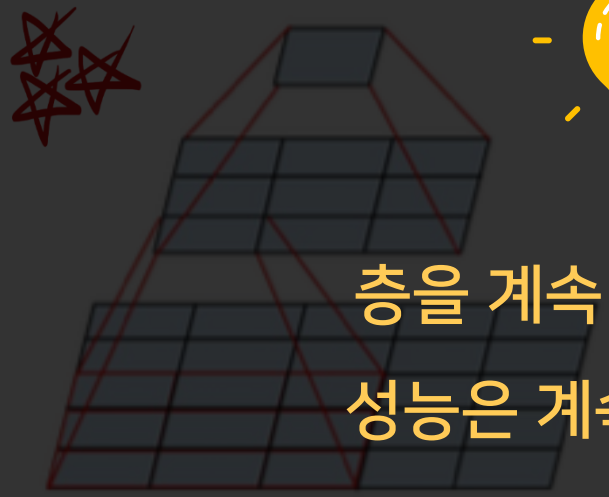
5x5 convolution

더 적은 파라미터로 좋은 성능

CNN의 발전 과정

VGGNet

Convolution Layer



two successive
3x3 convolutions

층을 계속 깊게 쌓으면
성능은 계속 좋아질까?

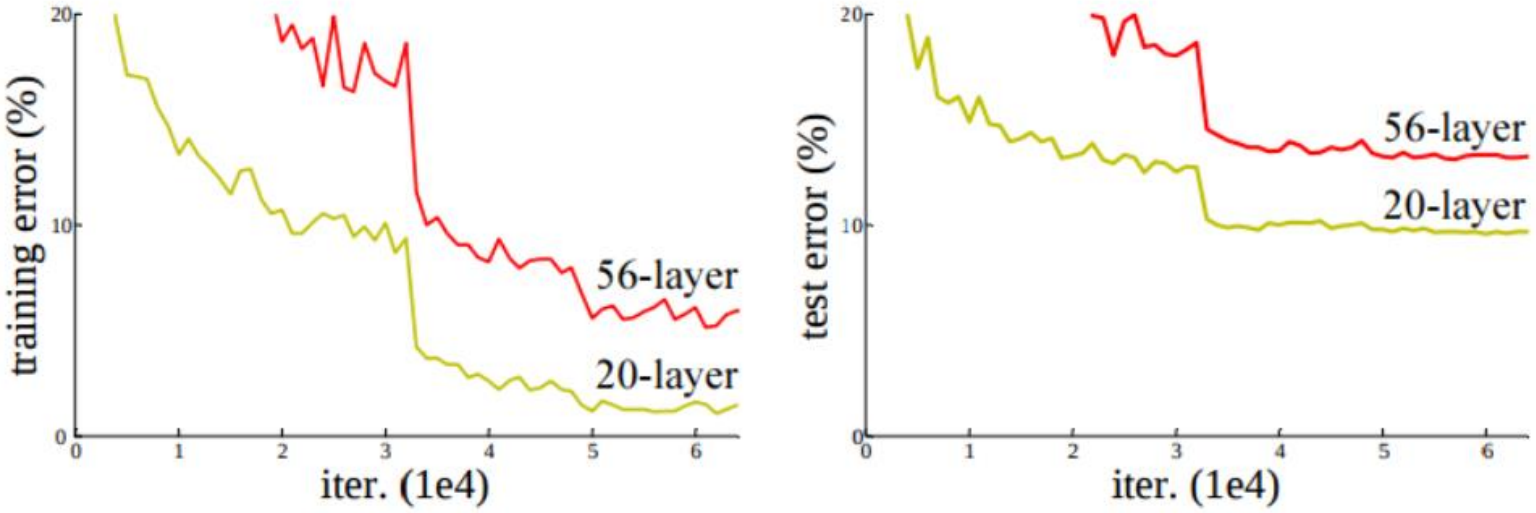
채널의 개수가 C 개일 때
파라미터 수 $2 \times 3^2 \times C$

5x5 convolution

CNN의 발전 과정

ResNet

Layer의 깊이



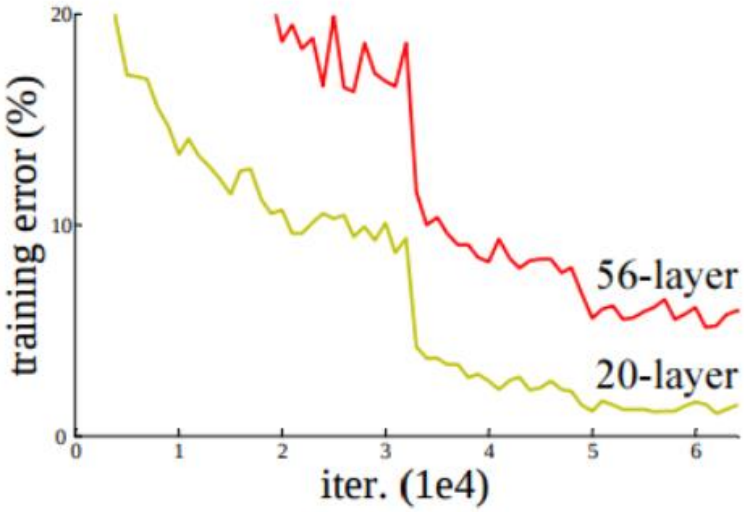
층이 너무 많아지면 오히려 성능의 감소 발생

Train, Test error 모두 순서의 역전 발생..

CNN의 발전 과정

ResNet

Layer의 깊이



Test 뿐만이 아니라

Train에서도 더 낮은 성능

단순히 과적합의 문제가

아니라, 성능이 떨어짐

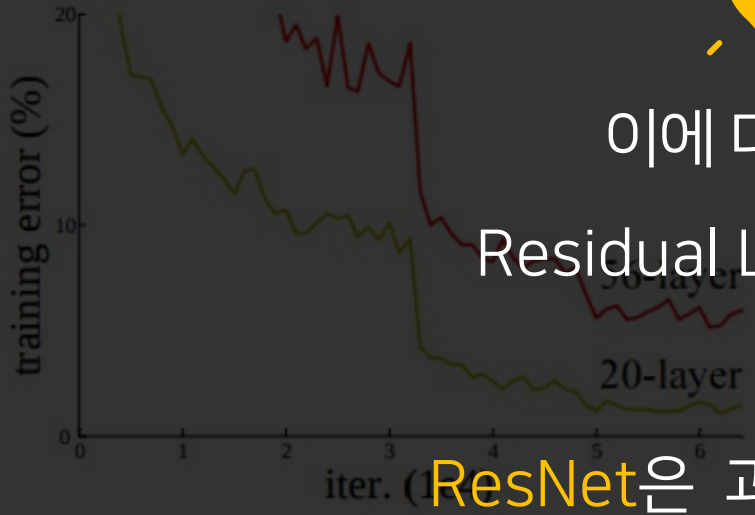
이는 최적화의 문제

층이 깊어질수록 최적화가 힘들어진다!!

CNN의 발전 과정

ResNet

Layer의 깊이



Test 뿐만이 아니라

Train에서도 더 낮은 성능

이에 대한 대책으로

Residual Learning을 사용!

과적합의 문제가

아니라, 성능이 떨어짐

ResNet은 과적합을 예방하면서

더 많은 층을 쌓을 수 있음

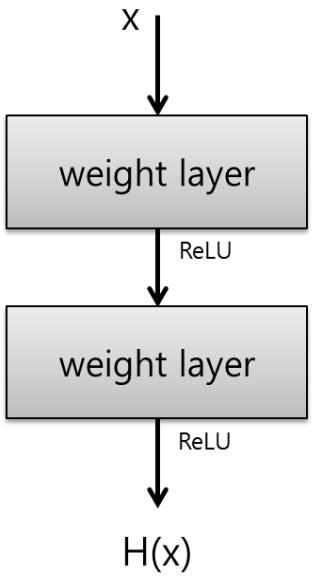
최적화의 문제

층이 깊어질수록 최적화가 힘들어진다!!

CNN의 발전 과정

ResNet

Residual Learning



기존 방식

기존의 CNN 모델은

이전의 layer 결과가 입력으로 들어가며

Convolution Layer를 통과한 결과로써

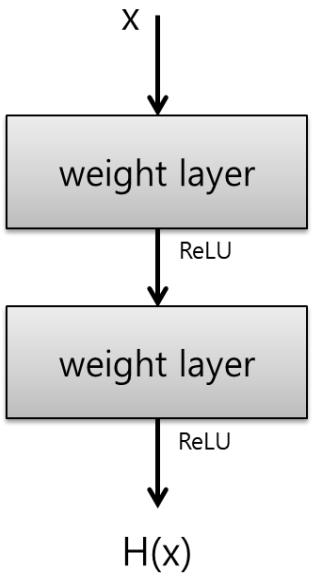
$H(x)$라는 새로운 Feature Map이 반환

Residual block

CNN의 발전 과정

ResNet

Residual Learning



기존 방식

기존의 CNN 모델은

이전의 layer 결과가 입력으로 들어가며

Convolution Layer를 통과한 결과로써

$H(x)$라는 새로운 Feature Map이 반환

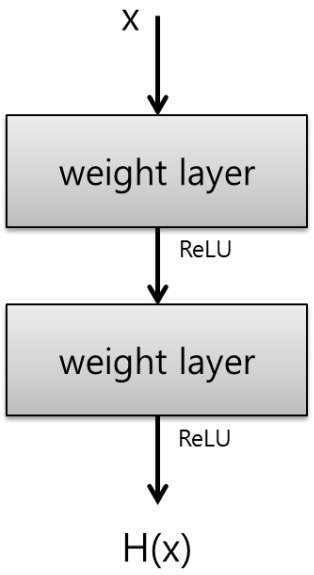
$$H(x) = f(x)$$

Residual block

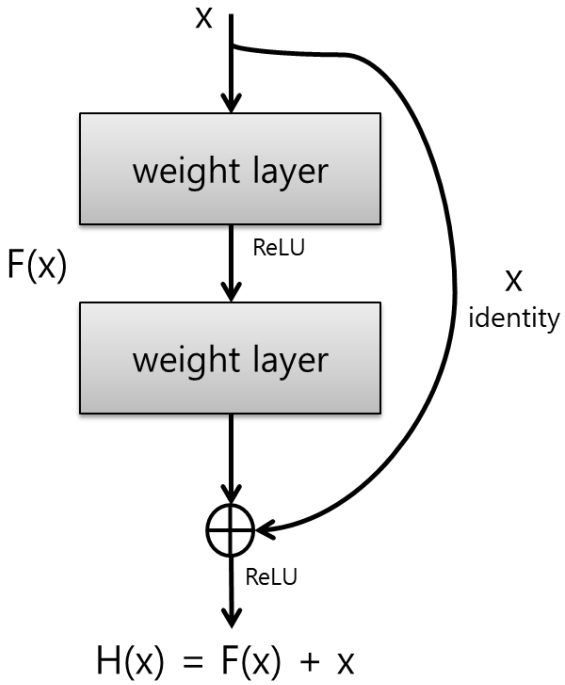
CNN의 발전 과정

ResNet

Residual Learning



기존 방식



Residual block

ResNet은 이전 layer의 결과를

입력으로 사용함과

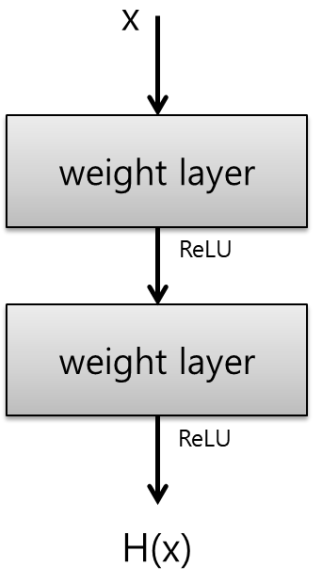
동시에 Convolution Layer

이후로 넘겨 그대로 더함

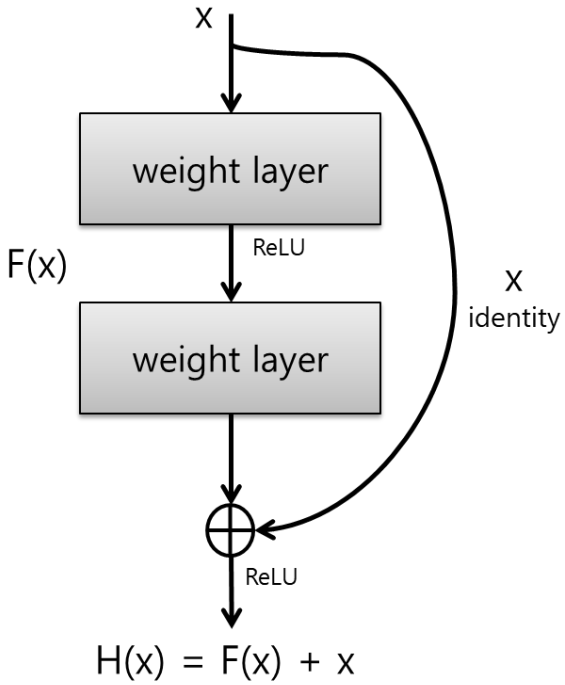
CNN의 발전 과정

ResNet

Residual Learning



기존 방식



Residual block

ResNet은 이전 layer의 결과를

입력으로 사용함과

동시에 Convolution Layer

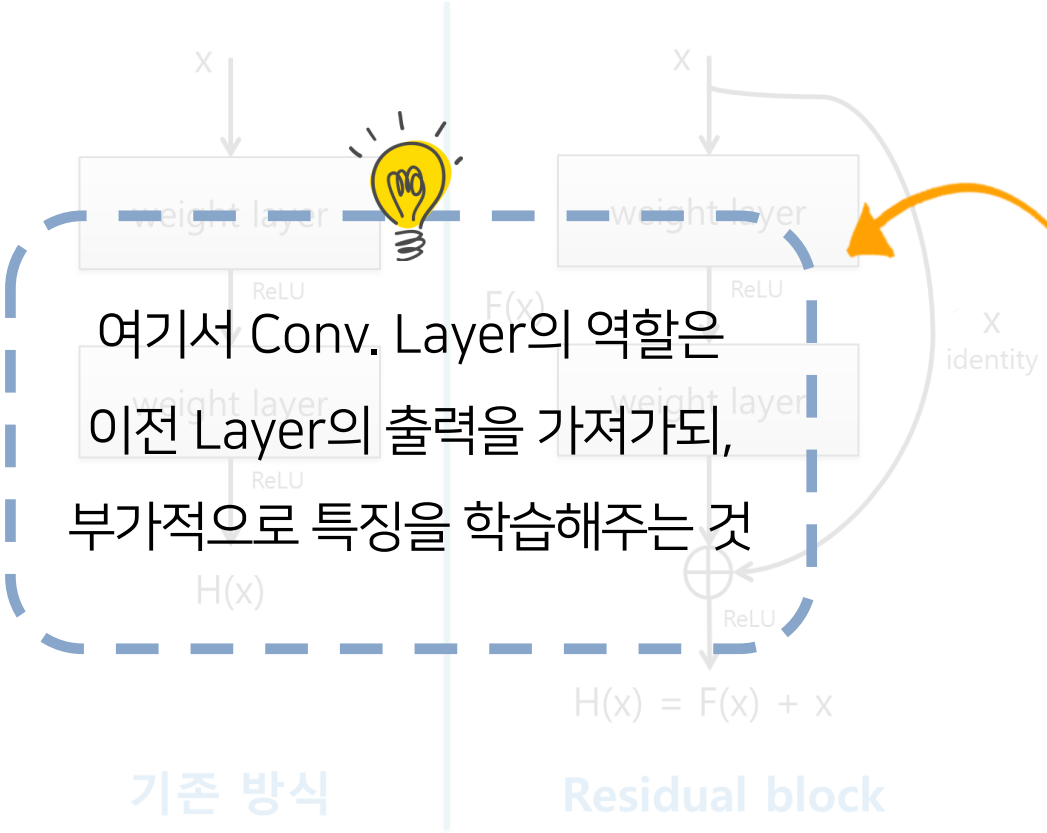
이후로 넘겨 그대로 더함

$$H(x) = f(x) + x$$

CNN의 발전 과정

ResNet

Residual Learning



여기서 Conv. Layer의 역할은
이전 Layer의 출력을 가져가되,
부가적으로 특징을 학습해주는 것

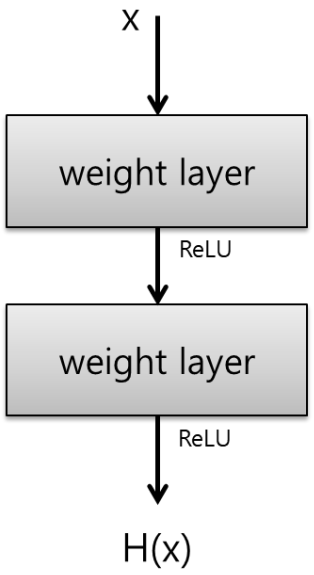
ResNet은 이전 layer의 결과를
입력으로 사용함과
동시에 Convolution Layer
이후로 넘겨 그대로 더함

$$H(x) = f(x) + x$$

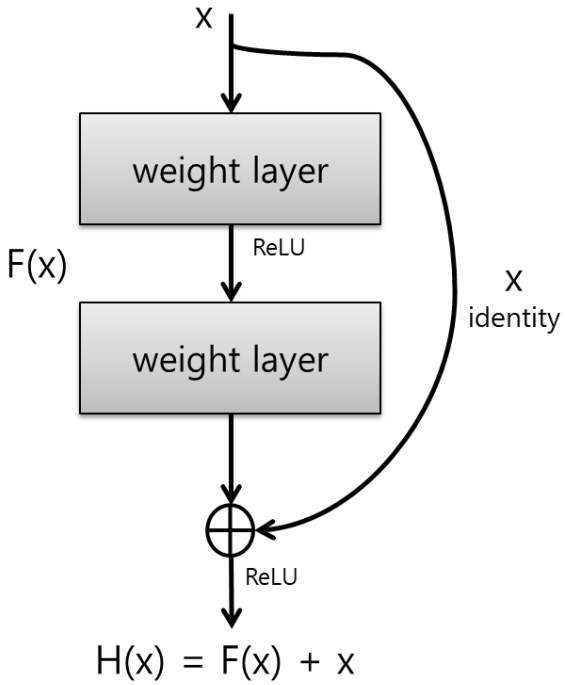
CNN의 발전 과정

ResNet

Residual Learning



기존 방식



Residual block

$$H(x) = f(x) + x$$

$$f(x) = H(x) - x$$

$$e = y - \hat{y}$$

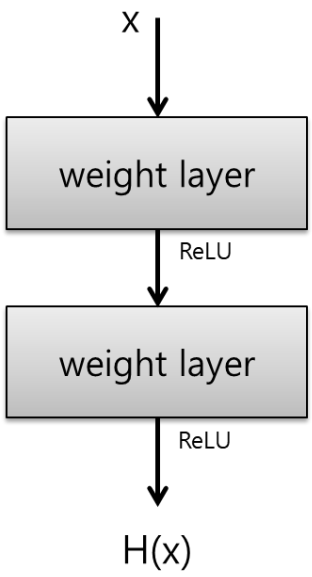
와 같은 Residual

그래서 이름이 ResNet!

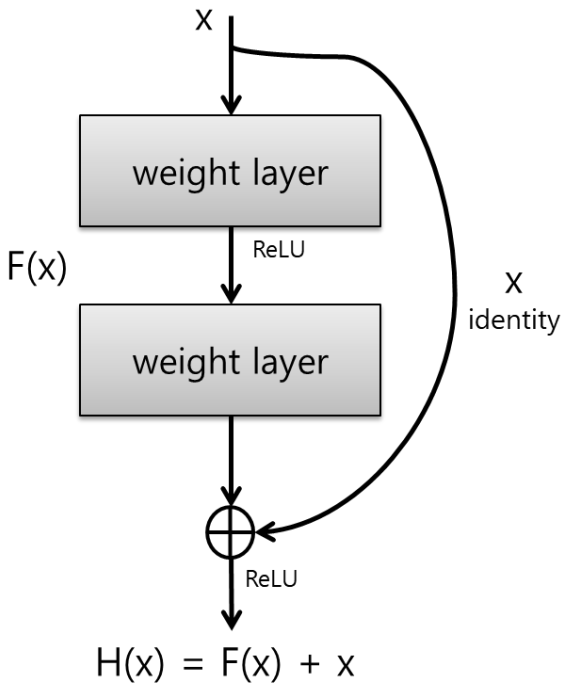
CNN의 발전 과정

ResNet

Residual Learning



기존 방식



Residual block

$$H(x) = f(x) + x$$

$$f(x) = H(x) - x$$

$$e = y - \hat{y}$$

와 같은 Residual

그래서 이름이 ResNet!

ResNet

Residual Learning

Q. 역전파시 미분값은 어떻게 나올까?

$$\frac{\partial H(x)}{\partial x} = (F(x) + x)' = F'(x) + 1$$

Residual Block 은 이전 Feature Map에서 학습되지 못한 $F(x)$ 을 최적화하는 방향으로 학습이 진행

ResNet

Residual Learning

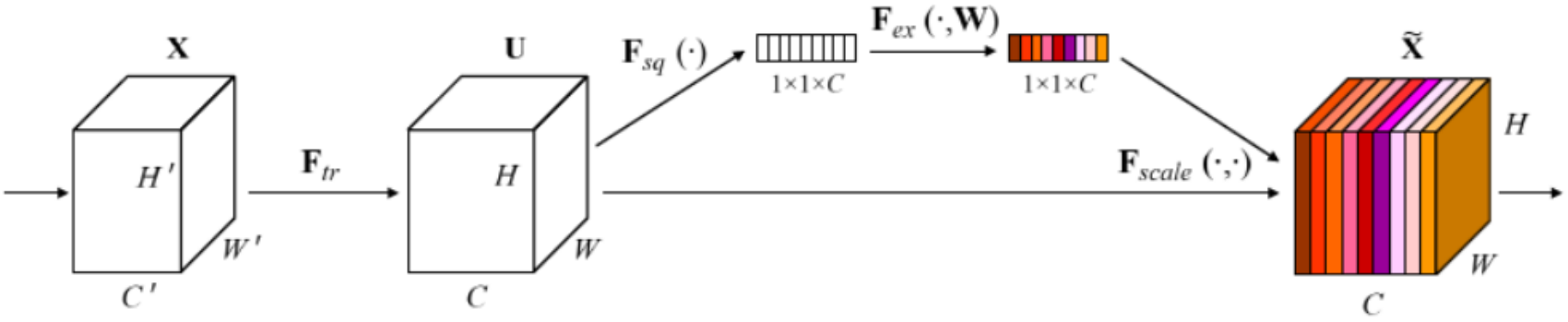
$$\frac{\partial H(x)}{\partial x} = (F(x) + x)' = F'(x) + 1$$

동시에 기존 x의 미분 값은 1이 되므로, 기울기 소실 문제 예방

⇨ 더 깊은 층을 쌓을 수 있음

CNN의 발전 과정

SENet



2017년 ILSVRC에서 1등을 차지한 모델!

SENet

SE Block을 통해 성능을 향상시킨 모델

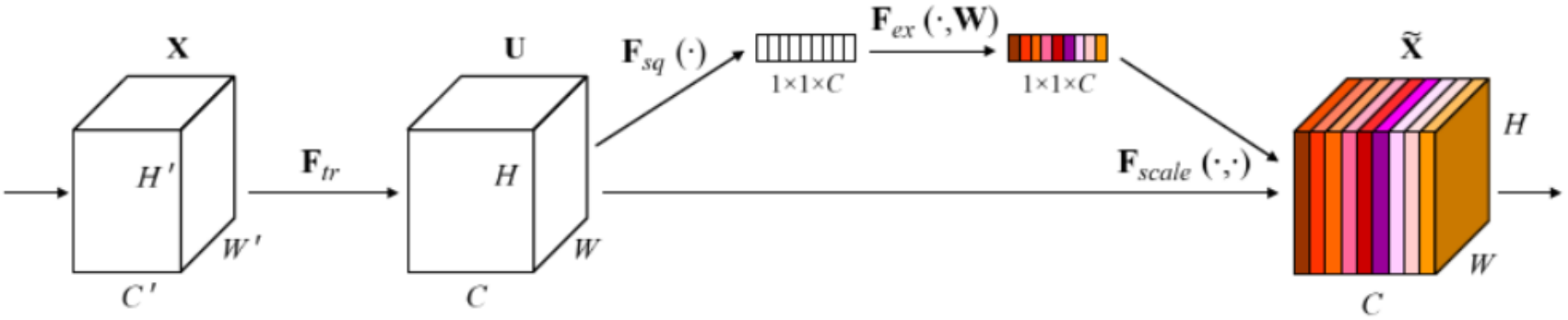
채널간의 상호작용을 이용해 성능 향상

성능은 많이 향상되는 반면, 연산량 증가는 많지 않음

CNN의 발전 과정

SENet

Se block

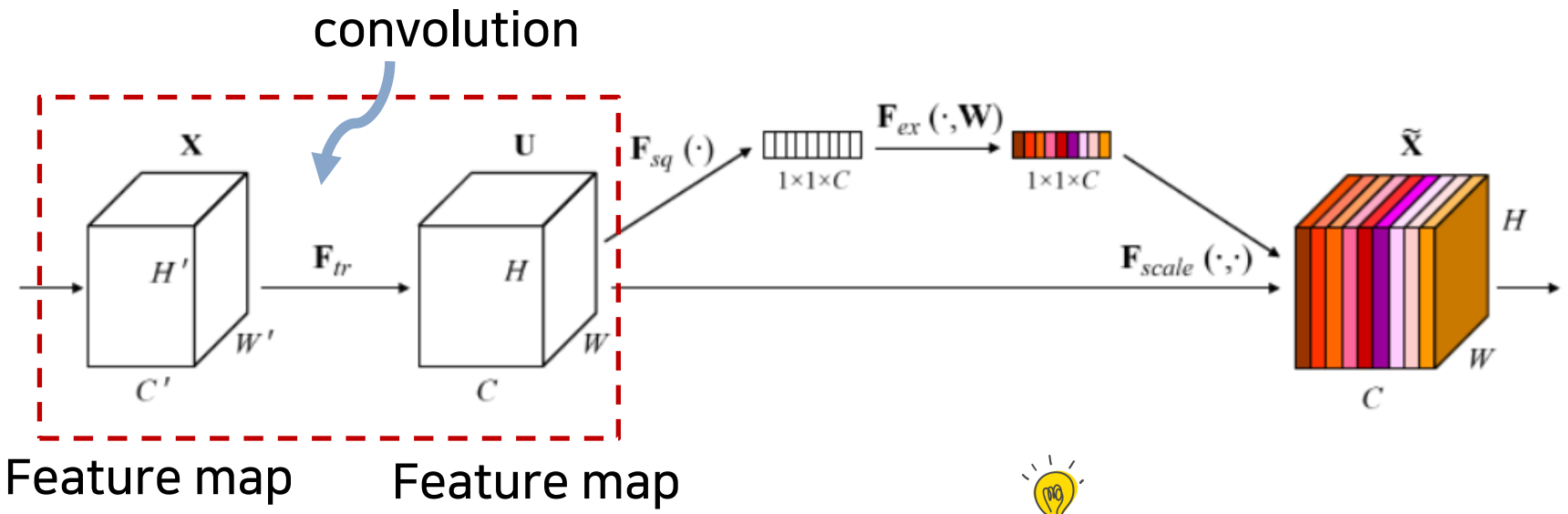


Convolution 연산을 통해
Feature map 크기가 U로 변환

CNN의 발전 과정

SENet

Se block

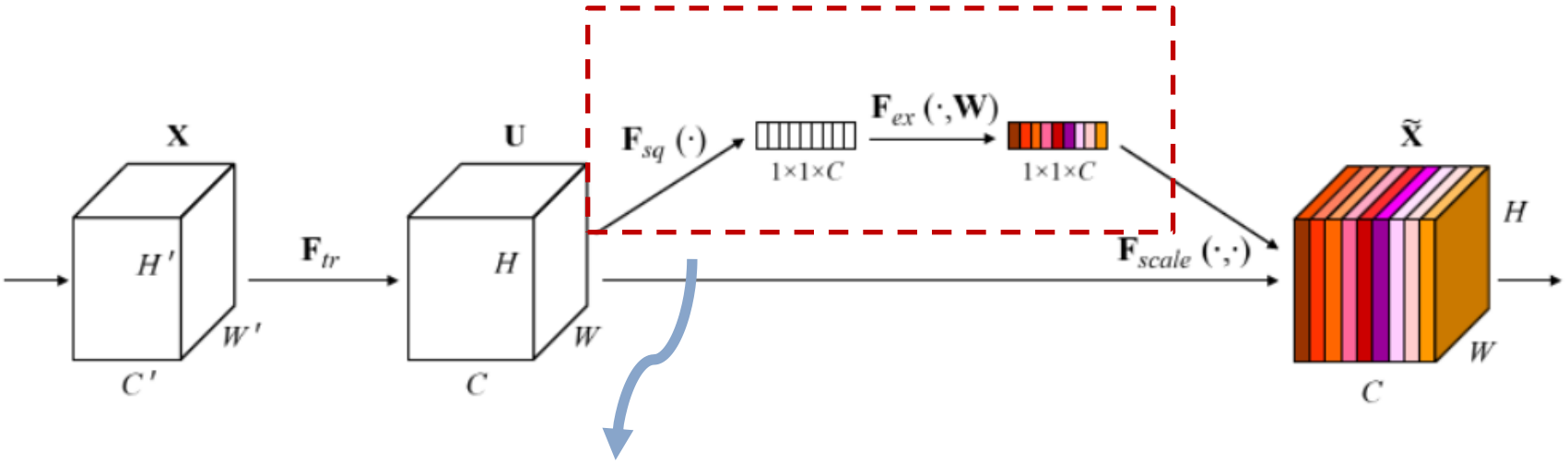


Convolution 연산을 통해

Feature map 크기가 U로 변환

CNN의 발전 과정

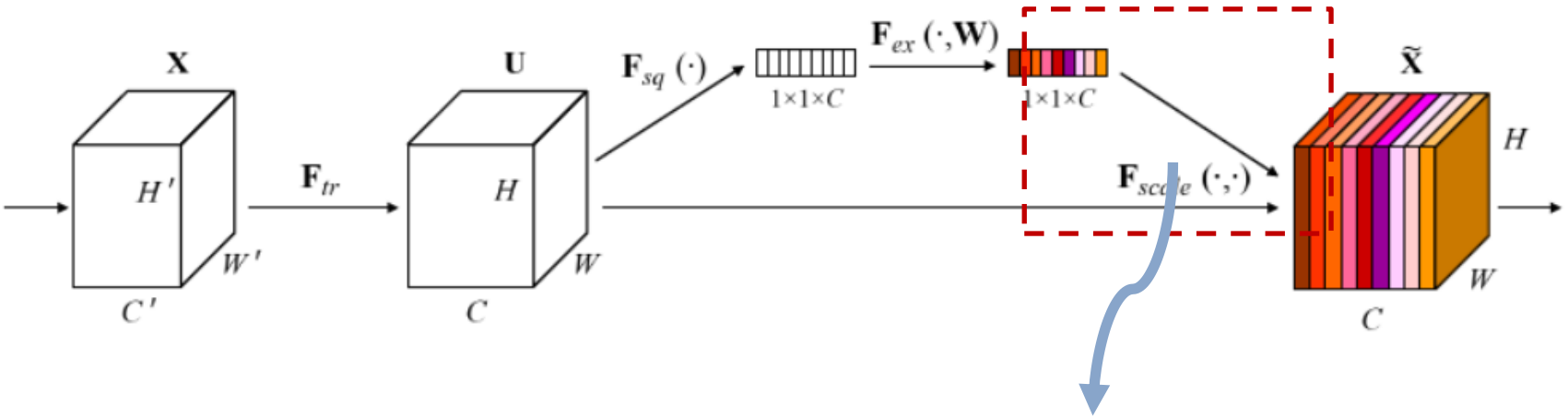
SENet



Squeeze로 1차원 변환

CNN의 발전 과정

SENet

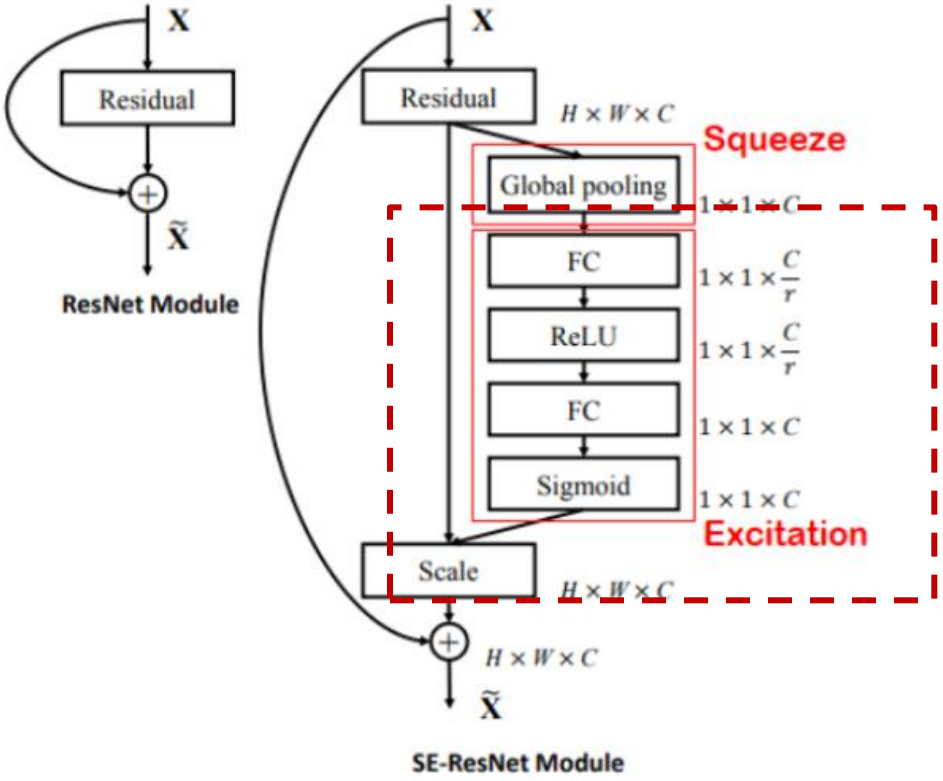


GAP(global average pooling)를 통한 값 반환

이 과정을 통해 각 채널을 하나의 숫자로 묘사 가능

CNN의 발전 과정

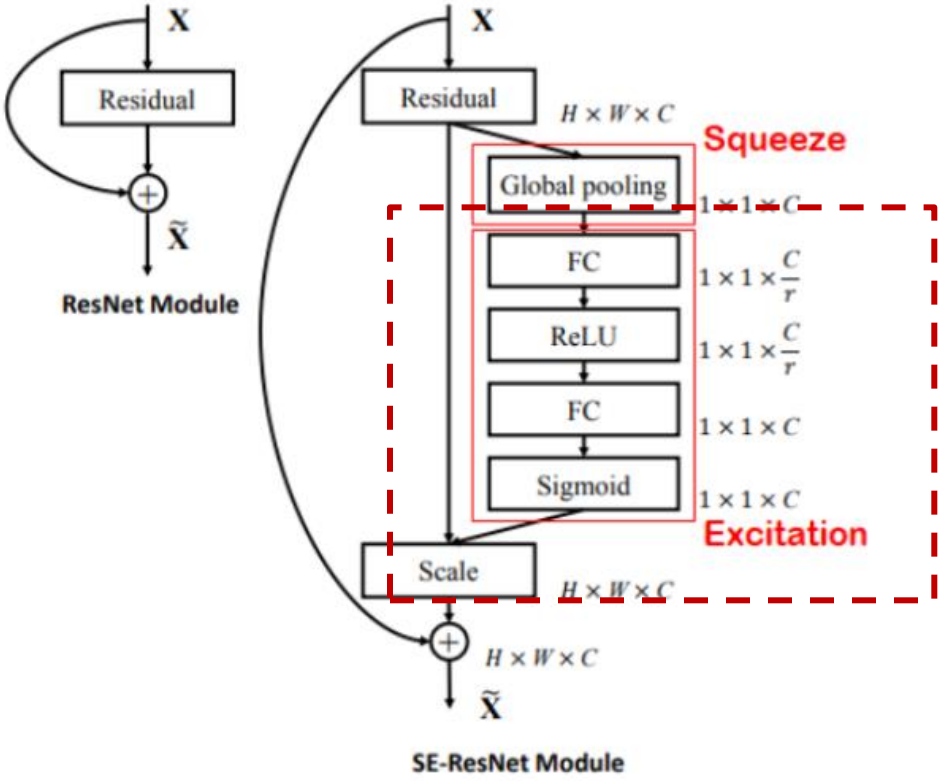
SENet



활성화 작업 돌입

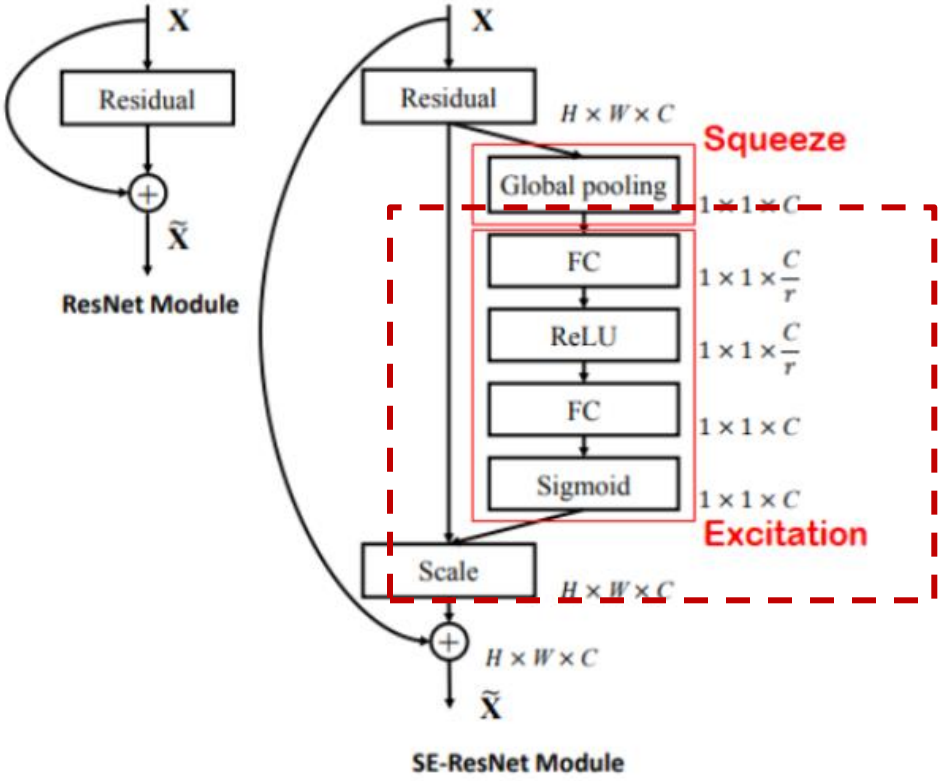
CNN의 발전 과정

SENet



① fc1에 $1 \times 1 \times C$ 벡터가 입력되어 C채널을 C/r개 채널로 축소

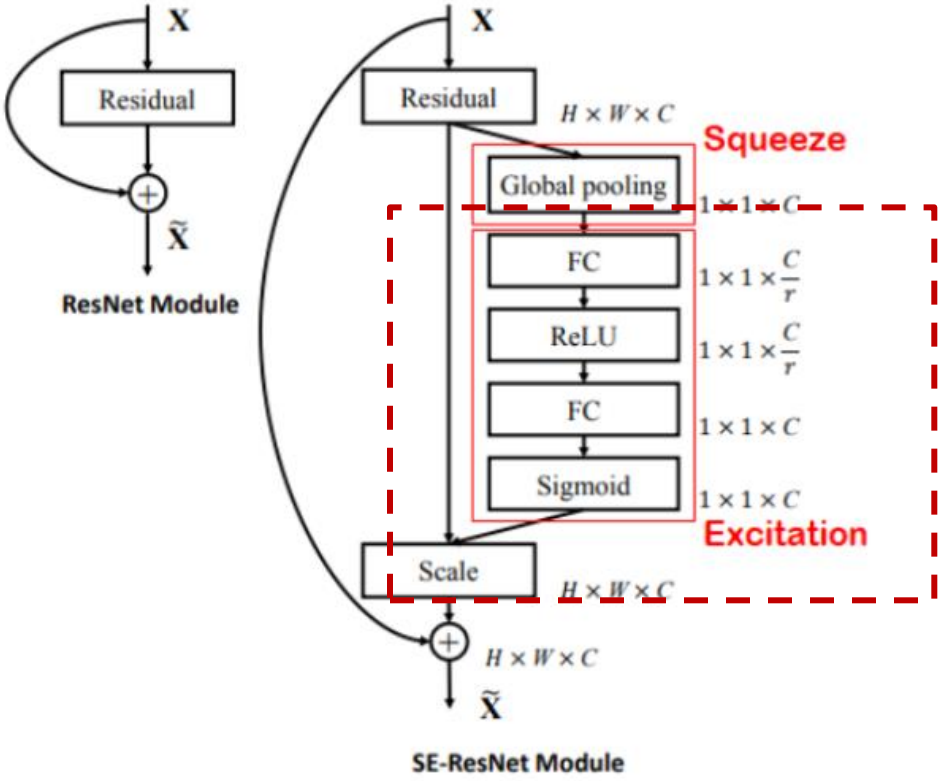
SENet



② C/r개 채널로 축소되어
$1 \times 1 \times C/r$ 가 된 벡터는
ReLU로 전달

CNN의 발전 과정

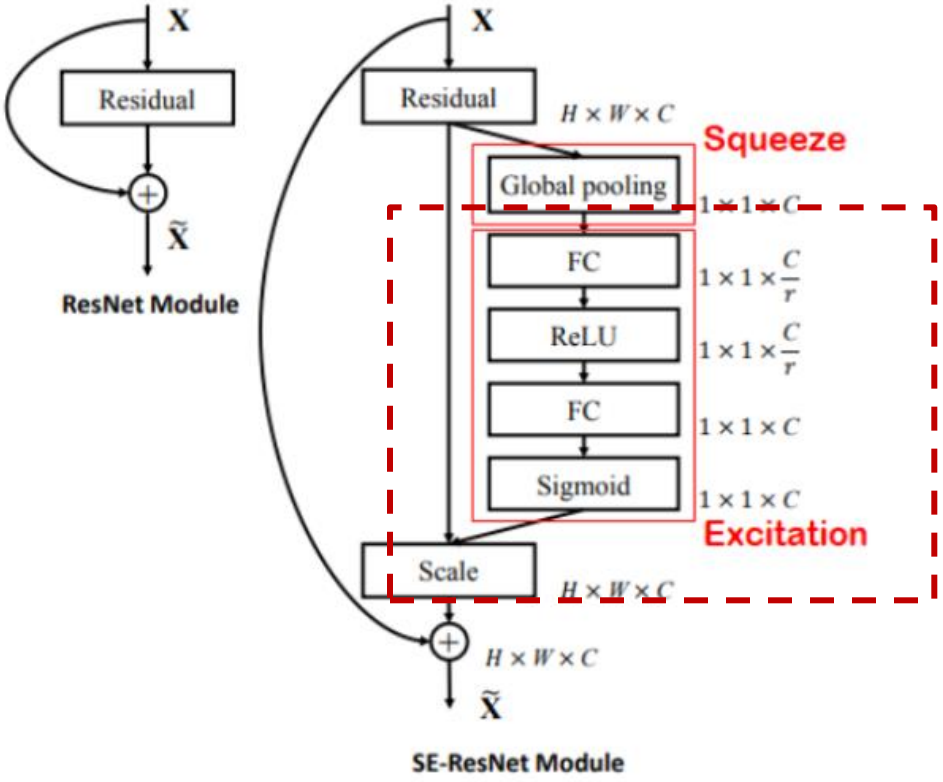
SENet



③ FC2를 통과,
FC2는 채널 수를
다시 C 로 되돌림

CNN의 발전 과정

SENet

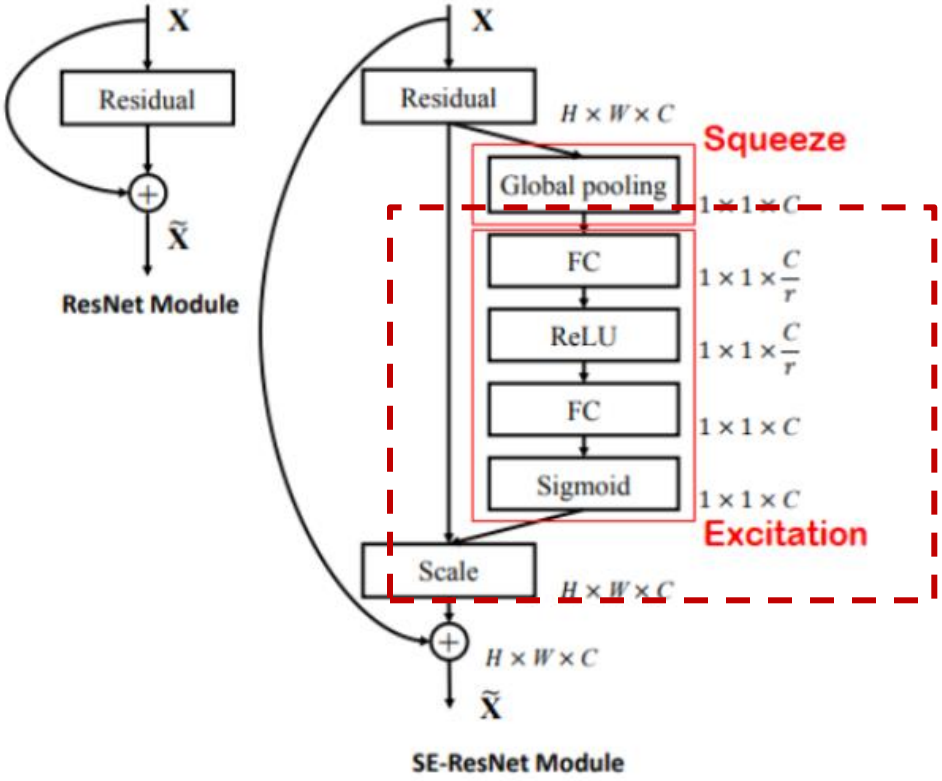


④ Sigmoid를 거쳐서
0~1 범위의 값을 지니게 됨

각 채널의 상대적 중요도를
0~1 사이 값으로 파악

CNN의 발전 과정

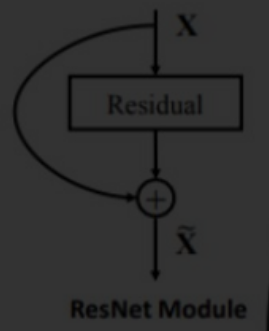
SENet

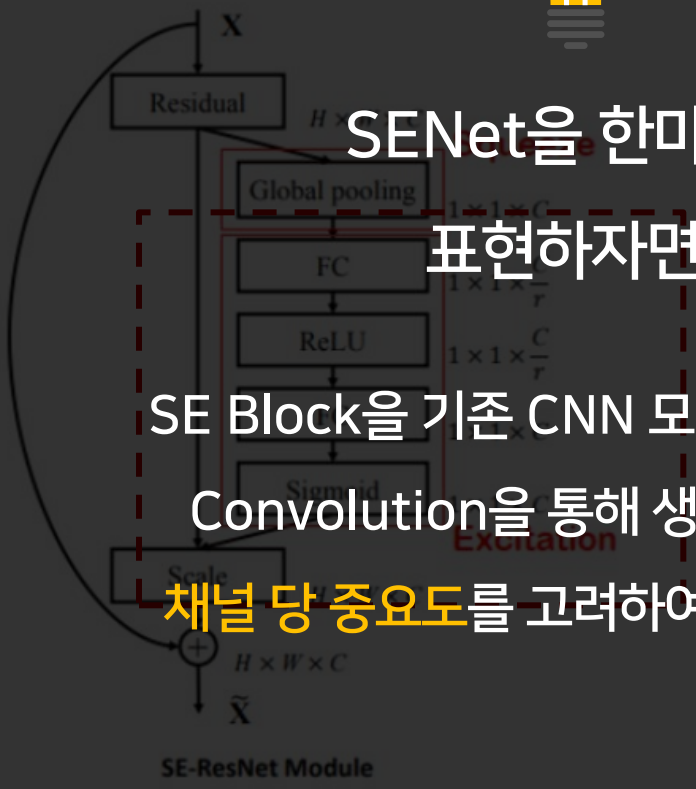


마지막으로, Feature Map과 곱해져 Feature Map의 채널에 가중치를 주게 됨

CNN의 발전 과정

SENet





SENet을 한마디로 표현하자면?

SE Block을 기존 CNN 모델에 붙이면서

Convolution을 통해 생성된 특징을

채널 당 중요도를 고려하여 재보정한 것

Sigmoid를 거쳐

0~1 사이 값으로 반환

각 채널의 상대적 중요도를

0~1 사이 값으로 파악

4

컴퓨터 비전

컴퓨터 비전

컴퓨터 비전이란

Computer Vision

컴퓨터 비전

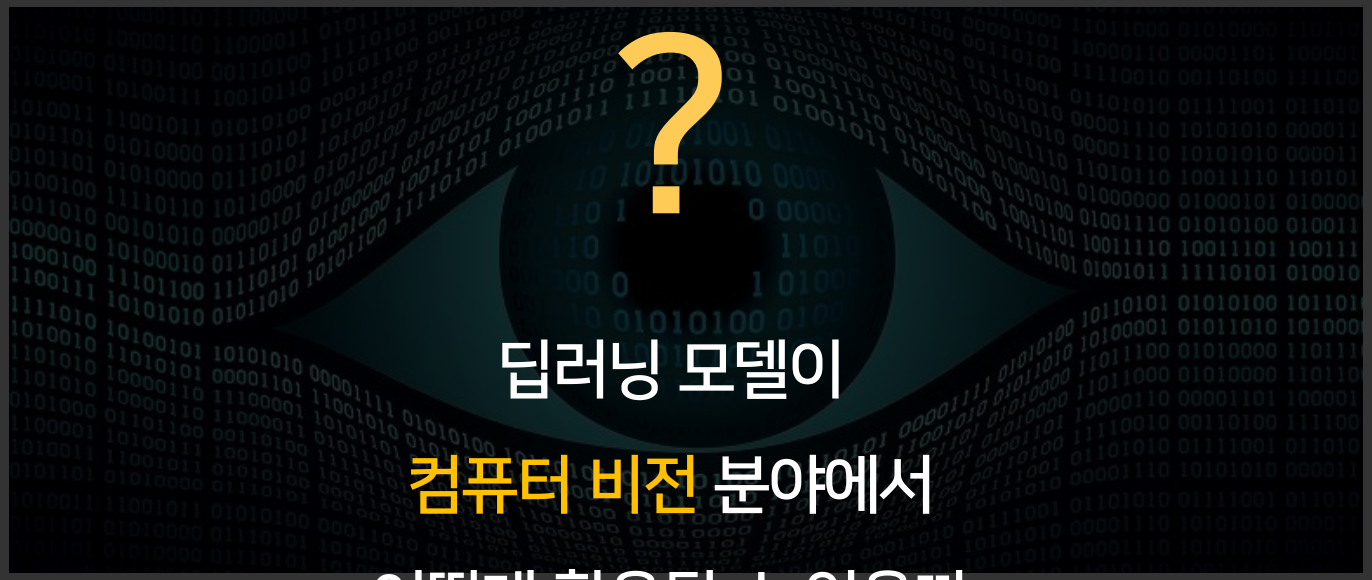
인공지능 분야 중 컴퓨터가 시각적인 체계를 이해하고

해석할 수 있도록 컴퓨터를 학습시키는 연구 분야

4 컴퓨터 비전

컴퓨터 비전이란

Computer Vision



인공지능 분야 중 컴퓨터가 시각적인 체계를 이해하고

해석할 수 있도록 컴퓨터를 학습시키는 연구 분야

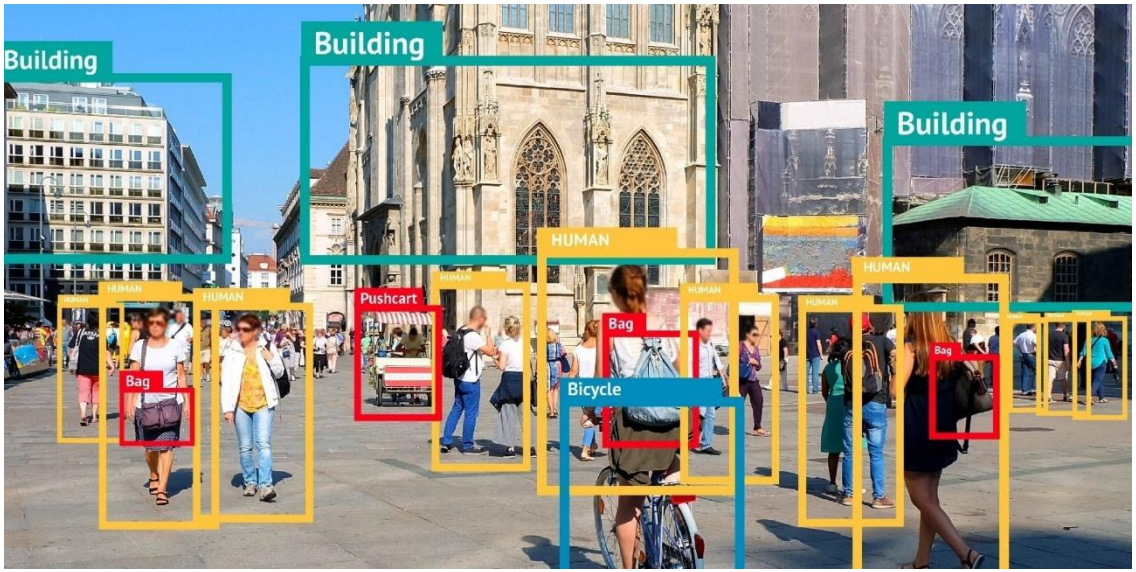
컴퓨터 비전

Objection Detection

객체 탐지

이 때 여러 객체가 존재할 때도 분류할 수 있음

이미지가 주어졌을 때, 어떤 물체가 어디에 있는지 탐지하는 과제



Objection Detection

객체 탐지

이미지가 주어졌을 때, 어떤 물체가 어디에 있는지 탐지하는 과제

Localization

어떤 범위에 객체가 존재하는지 탐지

Classification

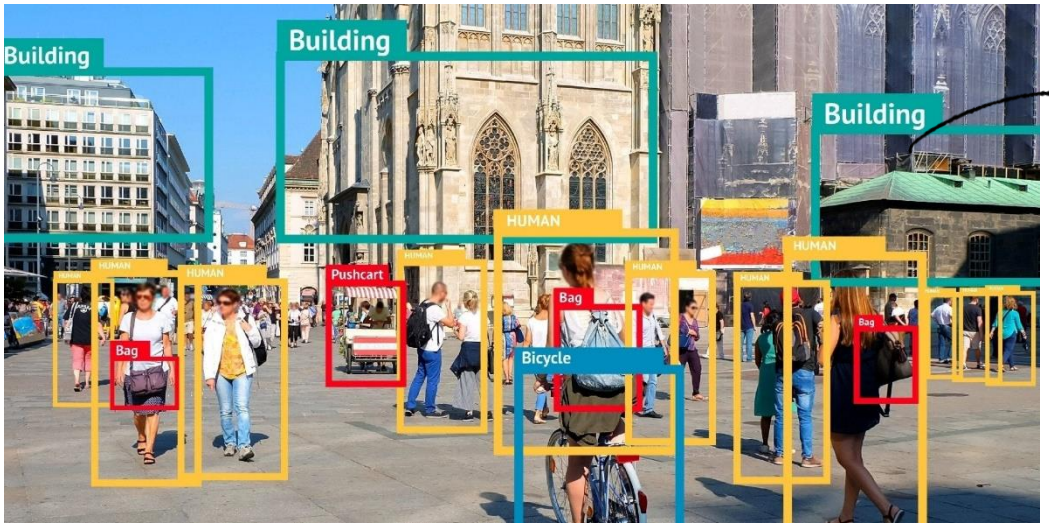
객체의 분류를 맡음

두 과정이 **모두** 일어나야 함

Objection Detection

객체 탐지

이미지가 주어졌을 때, 어떤 물체가 어디에 있는지 탐지하는 과제



Bounding Box

객체의 위치를
나타내는 경계 상자

Objection Detection

객체 탐지 종류

방법에 따라 크게 두 갈래로 나뉨

1-Stage Detector

Localization과

Classification을 동시 진행

⬇

빠른 작업 수행

상대적으로 정확도 떨어짐

VS

2-Stage Detector

Localization과

Classification을 순차 진행

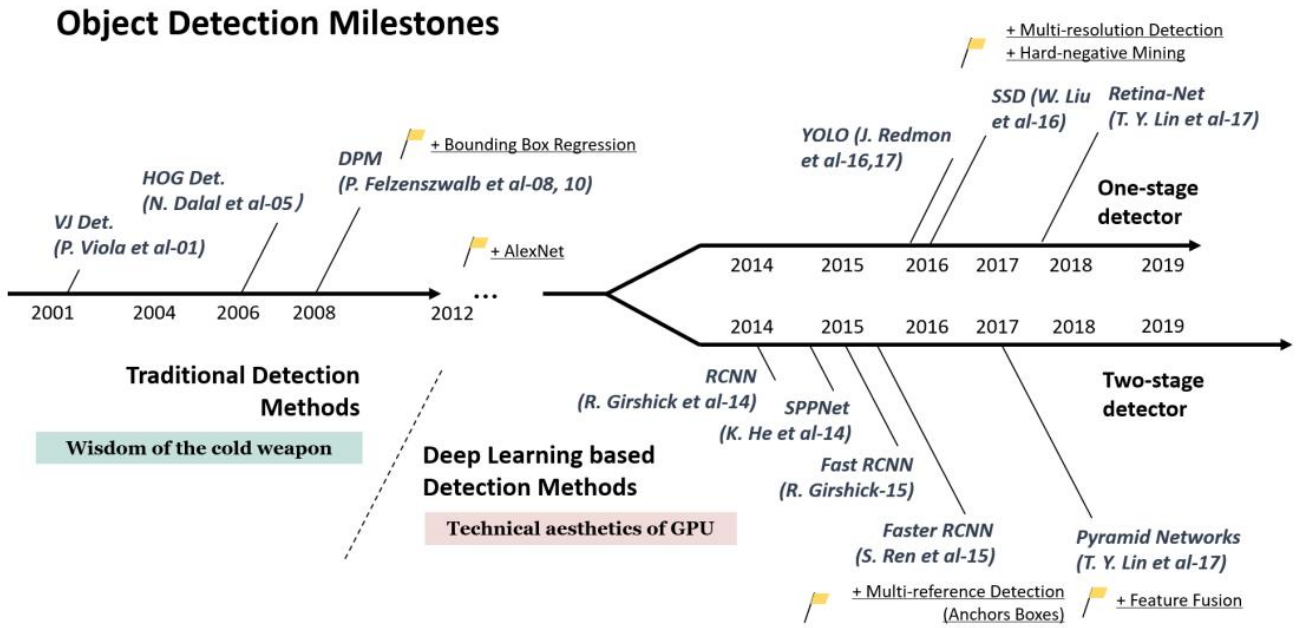
⬇

상대적으로 느림

높은 정확도를 보임

Objection Detection

객체 탐지 발전 과정

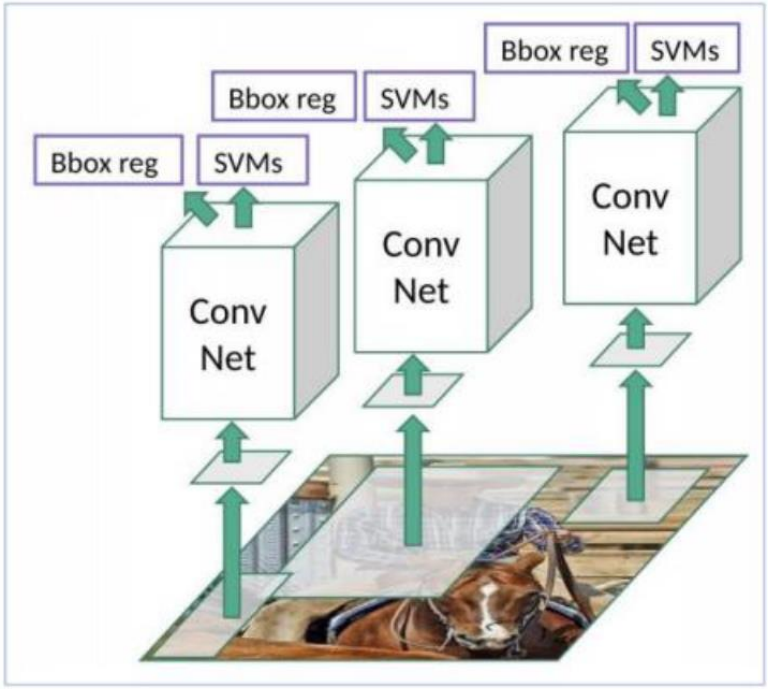


1-stage detector와 2-stage detector의 대표적인 모델들을 살펴볼 것!

RCNN

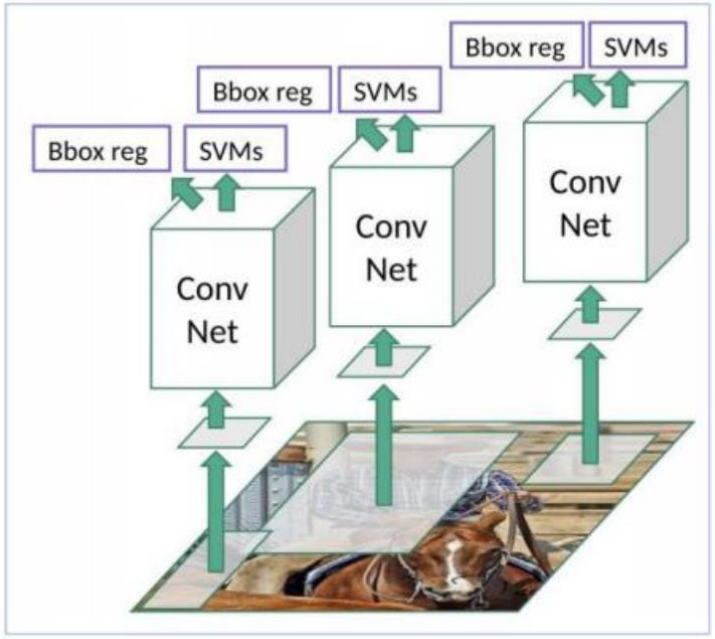
Regions with Convolutional Neuron Networks features

2-Stage Detector모델의 가장 기본적인 형태



RCNN

Regions with Convolutional Neuron Networks features



RCNN의 알고리즘 원리

이미지에서 RoI를 선별

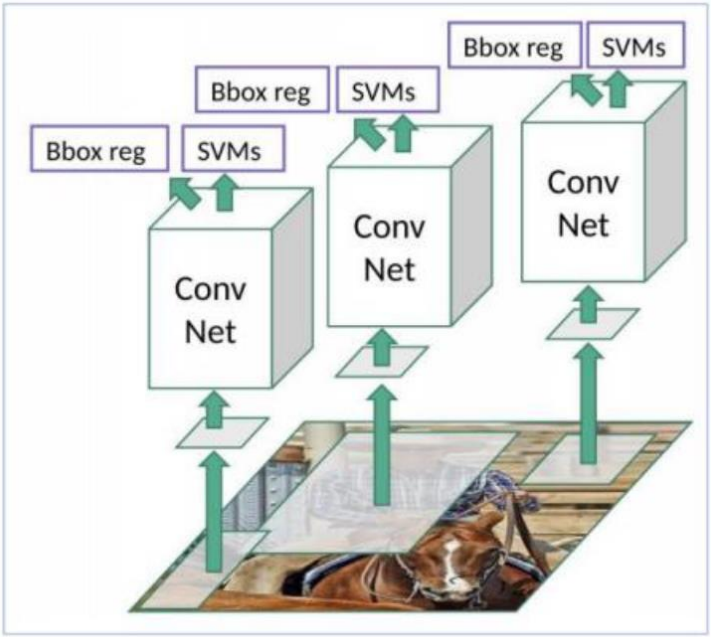
RoI: 객체가 존재할 것으로 예상되는 위치

Selective Search 알고리즘 사용

Bbox를 랜덤하고 작게 많이 생성하고,
이를 계층적 그룹핑 알고리즘을 통해
조금씩 합쳐서 RoI를 만드는 것

RCNN

Regions with Convolutional Neuron Networks features



RCNN의 알고리즘 원리

RoI를 CNN에 통과시킨 후 반환된 feature map을 바탕으로 회귀와 분류 진행

SVM을 통해 각 RoI의 라벨을 예측

localization error를 줄이기 위해 CNN feature 이용하여 Bbox regression model 수정

RCNN

Regions with Convolutional Neuron Networks features

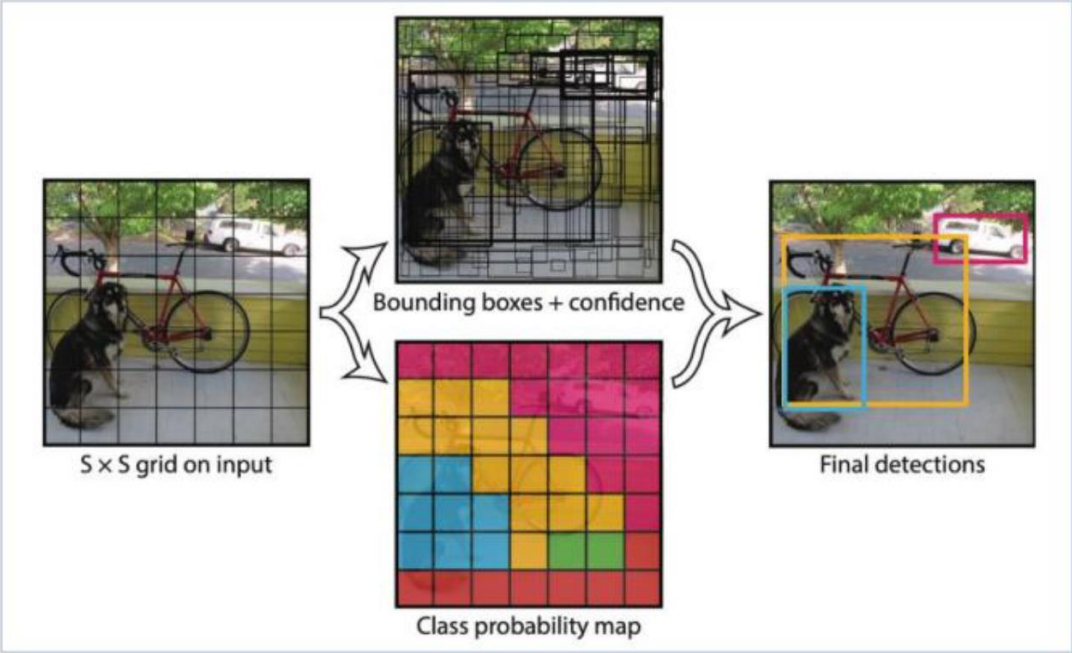
단점

- 여러 개의 CNN사용으로 시간 오래 걸림
- CNN, SVM, Bbox regression이 한번에 학습되지 않아 각 결과가 CNN을 업데이트 시키지 못함
- 회귀와 분류 모델이 모두 들어가 있어 최적화 어려움

YOLO

You Only Look Once

1-Stage Detector모델의 가장 대표적인 형태



YOLO

You Only Look Once

YOLO

이미지 전체에 하나의 모델이 한 번의 연산을 통해 bbox와 객체의 라벨 확률을 동시에 반환하는 모델

CNN

이미지를 여러 장으로 분할해 해석함

VS

YOLO

이미지 전체를 한번만 보고 객체와 객체의 위치를 예측할 수 있음

YOLO

You Only Look Once

YOLO

이미지 전체에 하나의 모델이 한 번의 연산을 통해 bbox와 객체의 라벨 확률을 동시에 반환하는 모델

CNN

이미지를 여러 장으로 분할해 해석함

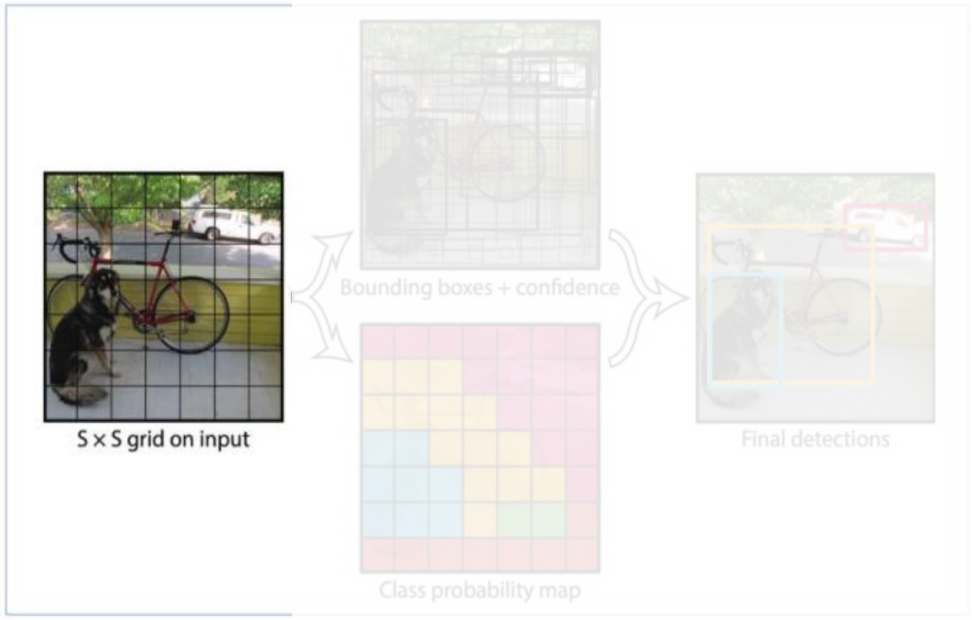
VS

YOLO

이미지 전체를 한번만 보고 객체와 객체의 위치를 예측할 수 있음

YOLO

You Only Look Once

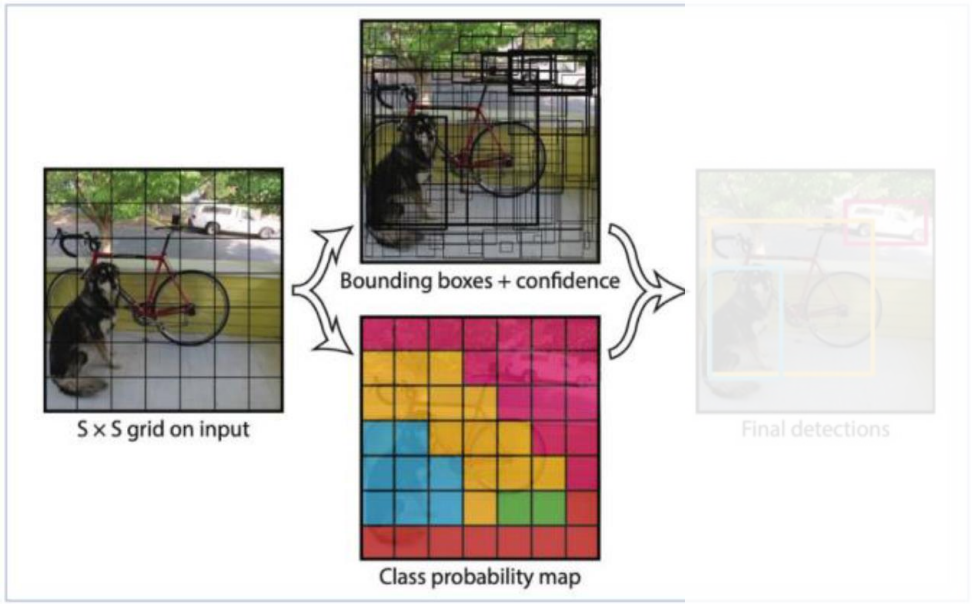


YOLO의 알고리즘 원리

①원본 이미지를 동일한 크기의 그리드(grid)로 나눔

YOLO

You Only Look Once



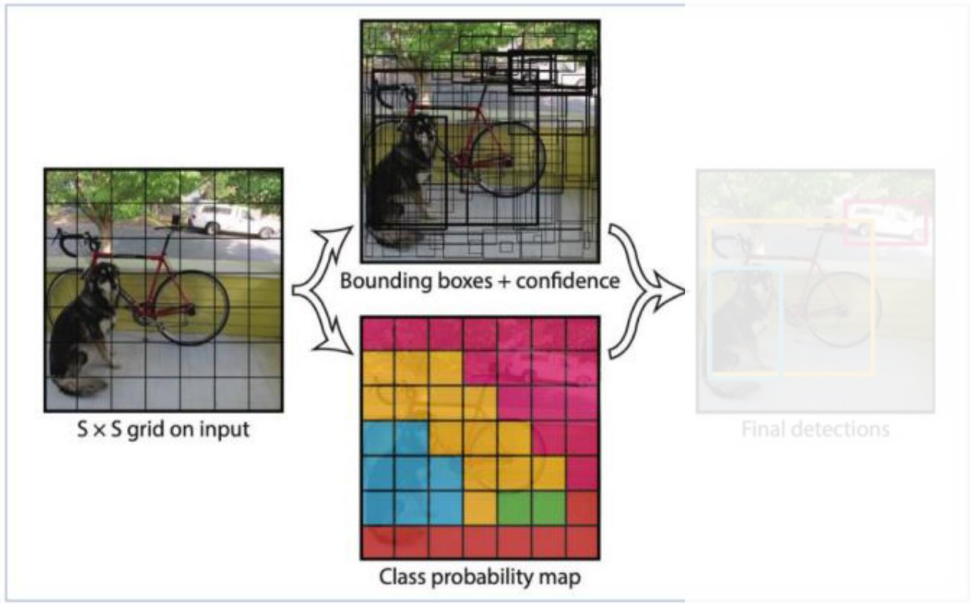
YOLO의 알고리즘 원리

②각 그리드에 대해 bbox와 confidence score을 계산

동시에 가장 높은 확률의 클래스 선별

YOLO

You Only Look Once



YOLO의 알고리즘 원리

③ 일정 이상의 확률 값을 갖는 셀들을 연결하여

최종적으로 bbox와 라벨을 반환

YOLO

You Only Look Once

장점

통합된 모델 사용으로
간단하고 빠르게
객체 검출 가능

VS

단점

2-stage detector보다
정확도가 떨어짐
특히 작은 객체 인식률이
떨어짐

Image Segmentation

이미지를 구성하고 있는 모든 픽셀을 대상으로 Class에 분류하는 작업

정확한 위치를 알아야 하기 때문에 object detection보다 복잡한 작업!

Semantic Segmentation

물체가 어디에 속하는지에
대해서만 분류를 진행

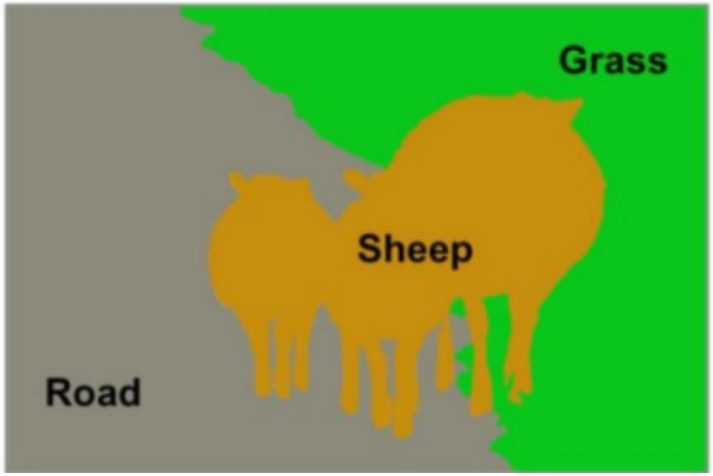
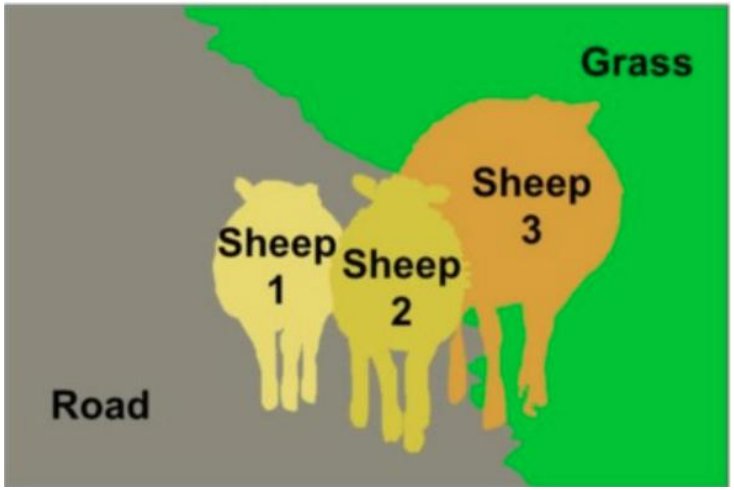


Image Segmentation

이미지를 구성하고 있는 모든 픽셀을 대상으로 Class에 분류하는 작업

정확한 위치를 알아야 하기 때문에 object detection보다 복잡한 작업!



Instance Segmentation

물체 분류 후
같은 class 내에서도
더 세부적으로 분류

Image Segmentation

출력을 "mask"라고 함

Image Segmentation

위 사진처럼 이미지를 입력 받고 각 픽셀이 어느 class에 속하는지 출력

입력 이미지 크기 = 출력 이미지 크기

Image Segmentation

일반적인 CNN같은 경우는 층을 거칠 때마다

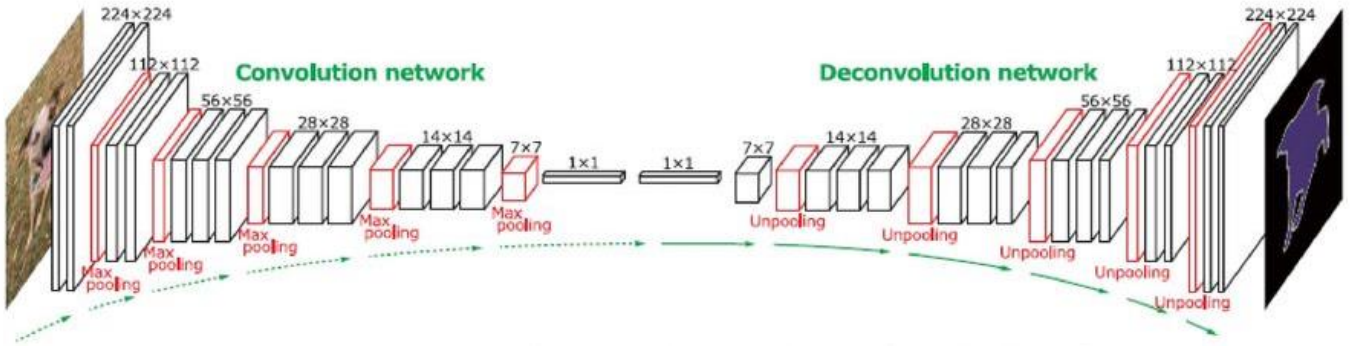
이미지의 크기가 줄어들기 때문에 그대로는 사용할 수 없음

CNN에서 이미지 크기를 줄이는 연산인 Convolution과 pooling를

역으로 연산해주어 다시 이미지를 키우는 방법 사용

Image Segmentation

일반적인 CNN같은 경우는 층을 거칠 때마다
이미지의 크기가 줄어들기 때문에 그대로는 사용할 수 없음



CNN에서 이미지 크기를 줄이는 연산인 Convolution과 pooling을
역으로 연산해주어 다시 이미지를 키우는 방법 사용

Image Generation

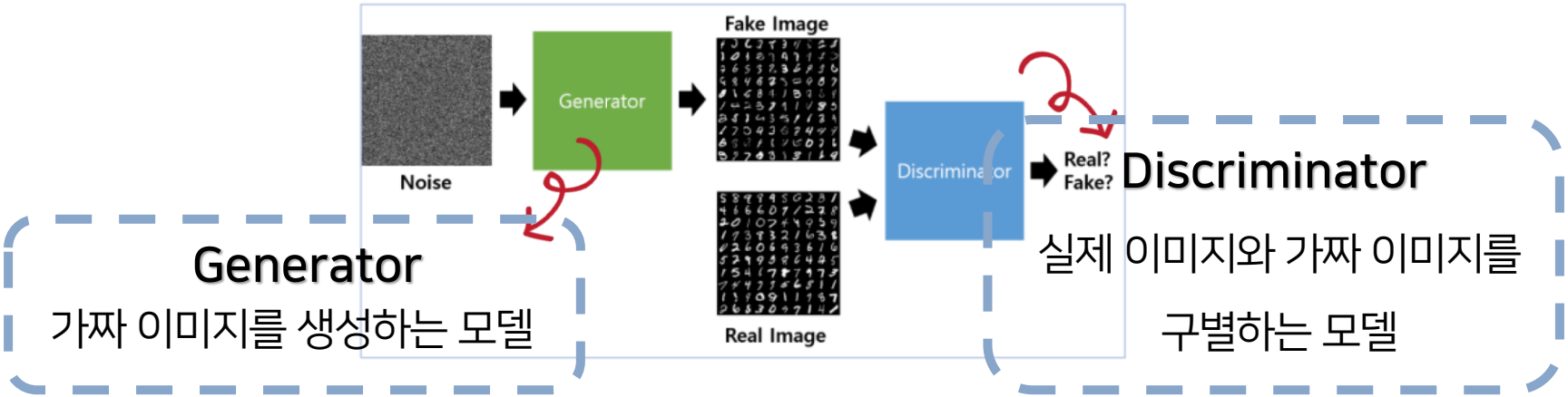
주어진 이미지를 바탕으로 새로운 이미지를 만드는 모델

비지도 학습에 속하는 딥러닝 모델들이 등장

GAN

Generative Adversarial Network

Generator와 Discriminator를 학습시켜, 진짜 같은 가짜 이미지 생성

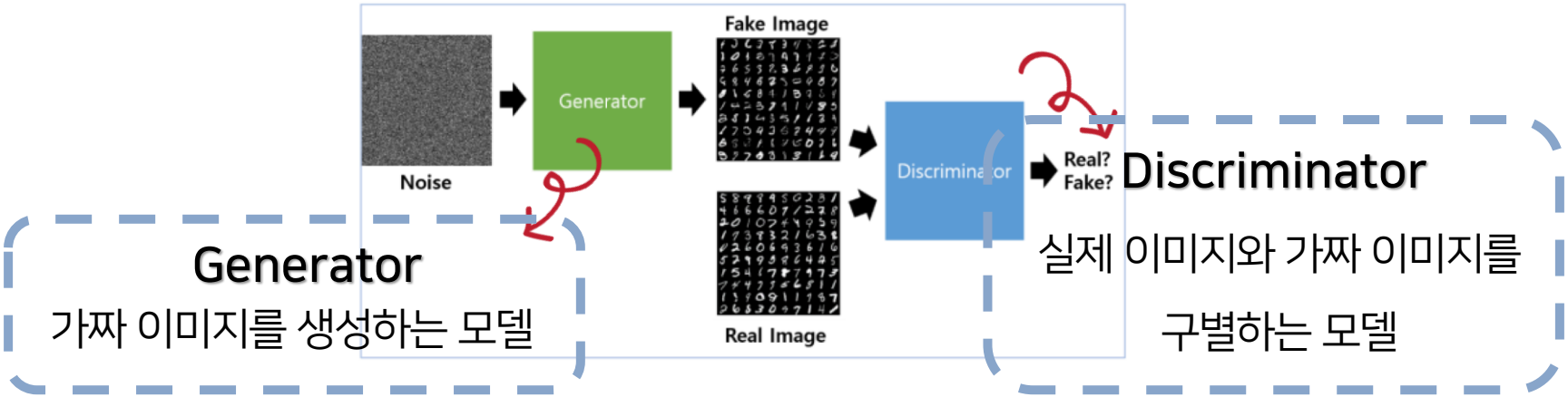


서로 적대적으로 학습하며 서로의 성능 향상시키면서
궁극적으로 진짜 같은 가짜 이미지 생성

GAN

Generative Adversarial Network

Generator와 Discriminator를 학습시켜, 진짜 같은 가짜 이미지 생성

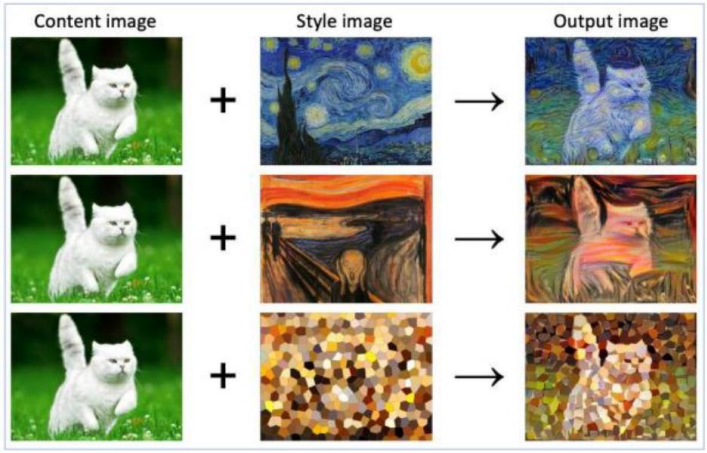


서로 적대적으로 학습하며 서로의 성능 향상시키면서
궁극적으로 진짜 같은 가짜 이미지 생성

Style Transfer

중간의 style 이미지 특징을 추출하여

입력에 해당하는 content image에 적용하는 알고리즘



Content image와 output image의 차이와

Style image와 Output의 차이를 모두 줄이는 것이 목표

5

마무리

5 마무리

다음주 예고

1. 자연어의 특징
2. 자연어의 전처리
3. RNN의 구조
4. RNN의 응용

5 마무리

“어때? 딥러닝 재미있지?”

THANK YOU