

딥러닝팀

1팀

안세현
이수정
이승우
전효림
홍지우

INDEX

1. 이미지 데이터의 특징
2. CNN의 구조
3. CNN모델의 발전
4. Deep Learning in CV

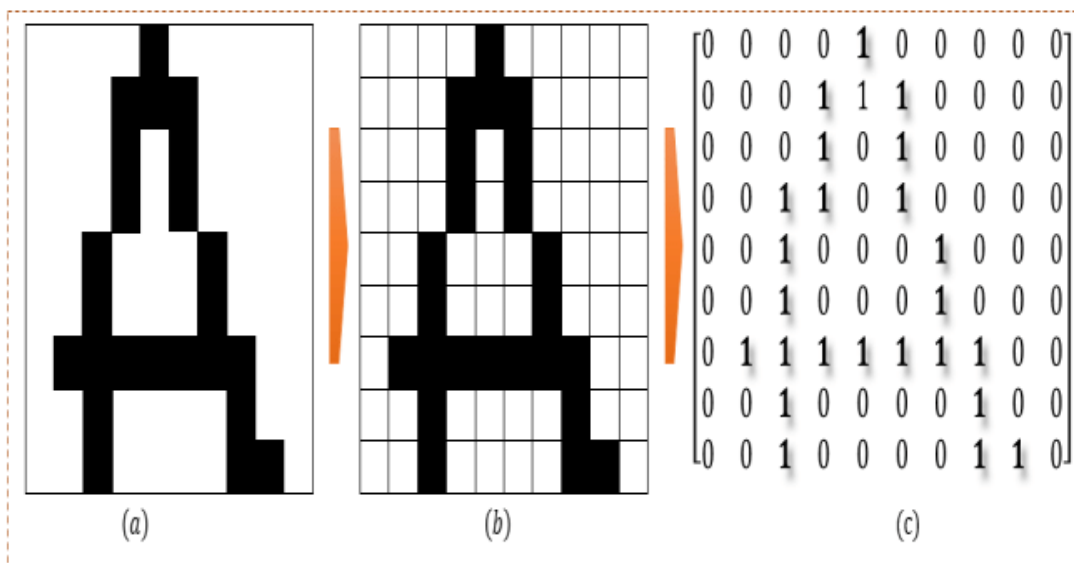
1

이미지 데이터의 특징

1 이미지 데이터의 특징

- 컴퓨터에서의 이미지 데이터

흑백 이미지



✓ 비트맵 이미지

흑백 이미지

검은색: 1 / 흰색: 0

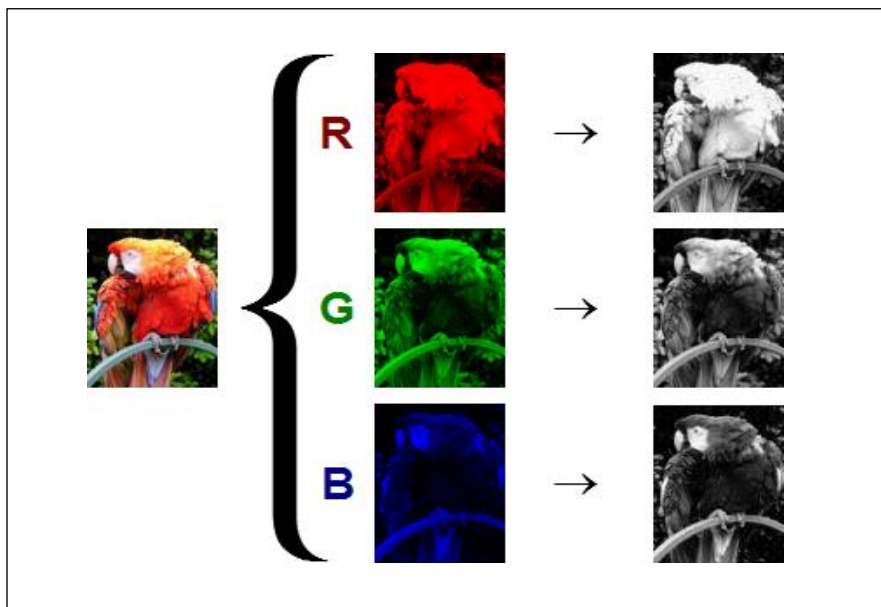
“숫자를 이용해 행렬의 형태로 표현 가능”

컴퓨터의 모든 이미지: 픽셀로 구성

1 이미지 데이터의 특징

- 컴퓨터에서의 이미지 데이터

컬러 이미지



이미지의 색을 세 색으로 표현 가능

빛의 삼원색: 빨강 / 초록 / 파랑

↓
Grayscale

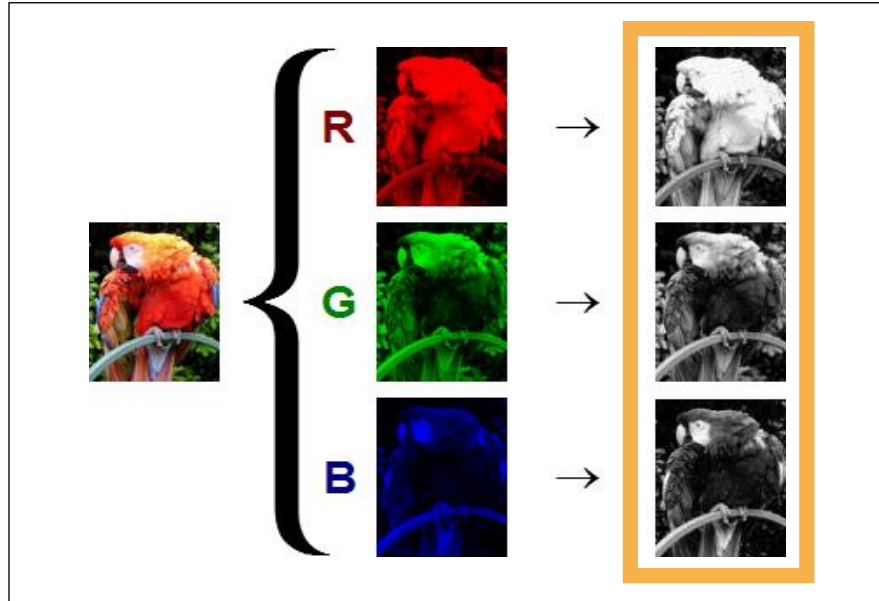
R채널, G채널, B채널

1 이미지 데이터의 특징

- 컴퓨터에서의 이미지 데이터

컬러 이미지

Grayscale로 변환한 모습



8-bit 비트맵 이미지의 경우
각 채널에서 한 픽셀은 0-255,
총 $256(2^8)$ 개 값 중 하나의 값을 가짐

1 이미지 데이터의 특징

- 컴퓨터에서의 이미지 데이터

컬러 이미지

기존 앵무새 색에 의해 가장 밝음



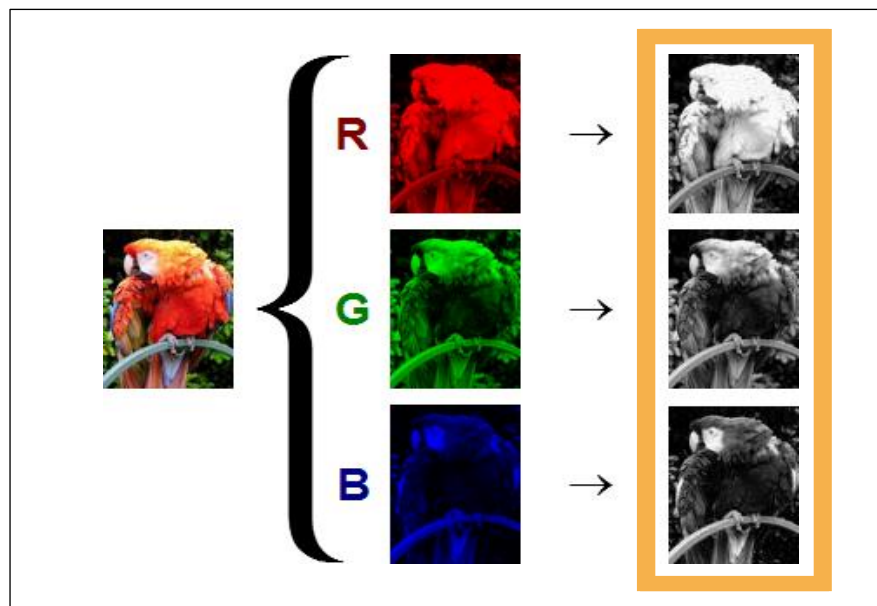
값이 클수록,
grayscale 이미지에서 하얗게 나타남

1 이미지 데이터의 특징

- 컴퓨터에서의 이미지 데이터

컬러 이미지

Grayscale로 변환한 모습



✓ 가로, 세로를 구성하는 픽셀 수가 256개이고
R, G, B 세 색으로 구성된 이미지의 경우



(Height, Width, Channel) = (256, 256, 3)
크기의 3차원 행렬로 표현 가능

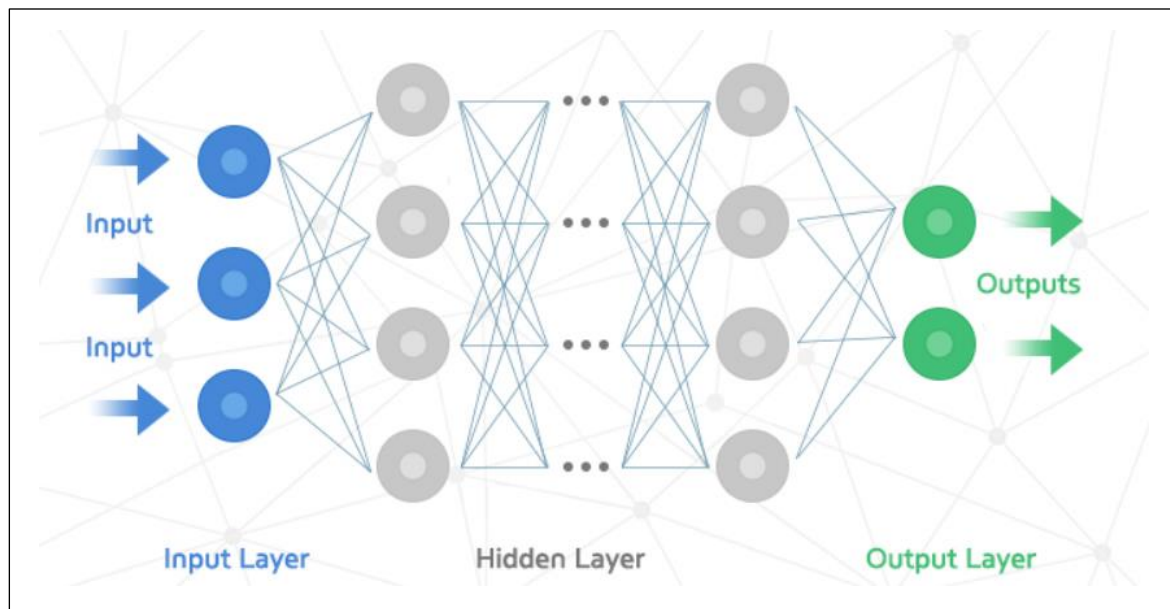
2

CNN의 구조

2 CNN의 구조

- CNN(Convolutional Neural Network) 등장 배경

기존 신경망



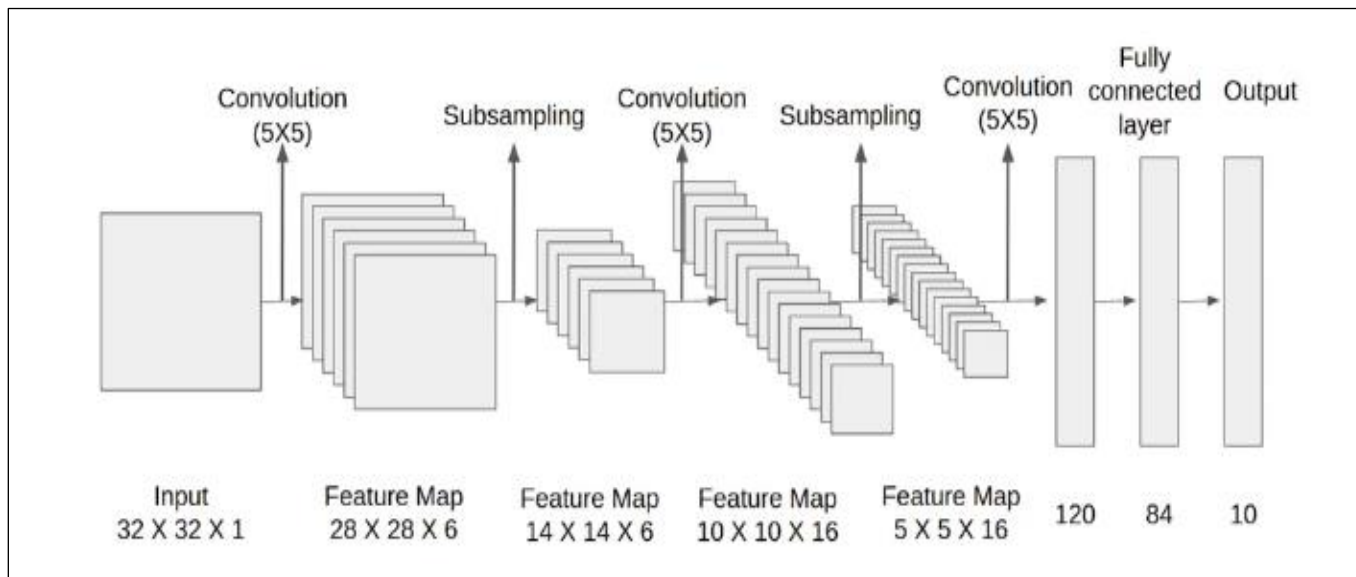
- 1차원 벡터만을 input
- 2/3차원의 경우, 1차원으로 강제 변환 필요

2 CNN의 구조

● CNN(Convolutional Neural Network) 등장 배경

LeNet

처음으로 사용된 CNN 모델



<기존 신경망>

- 1차원 벡터만을 input
- 2/3차원의 경우, 1차원으로 강제 변환 필요



이미지의 공간 정보 손실
신경망 제대로 학습 X

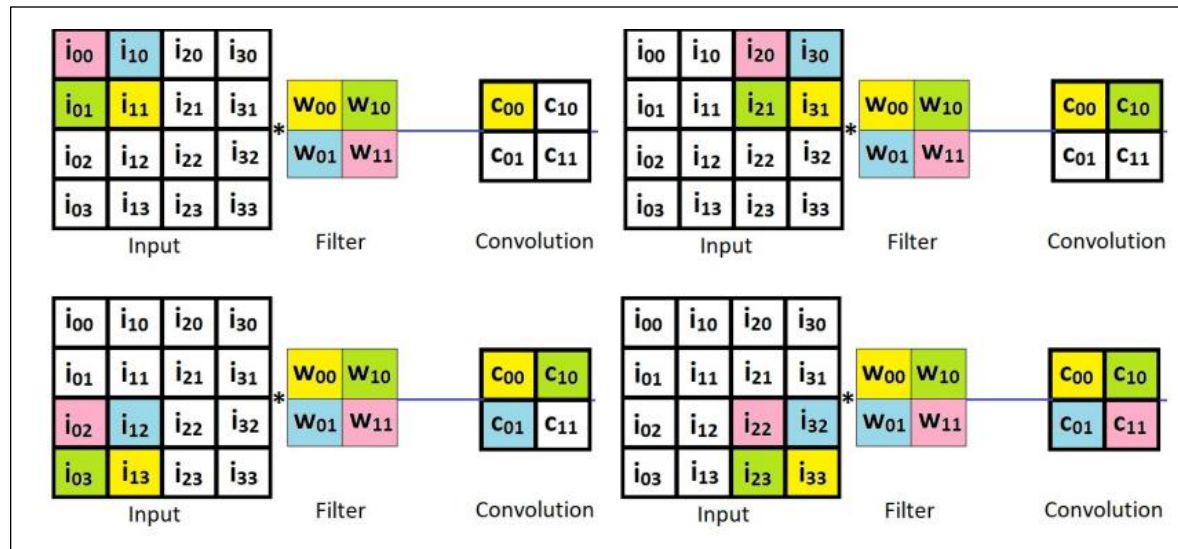


CNN 등장

2 CNN의 구조

- Convolution layer

CNN의 구조와 동작 과정



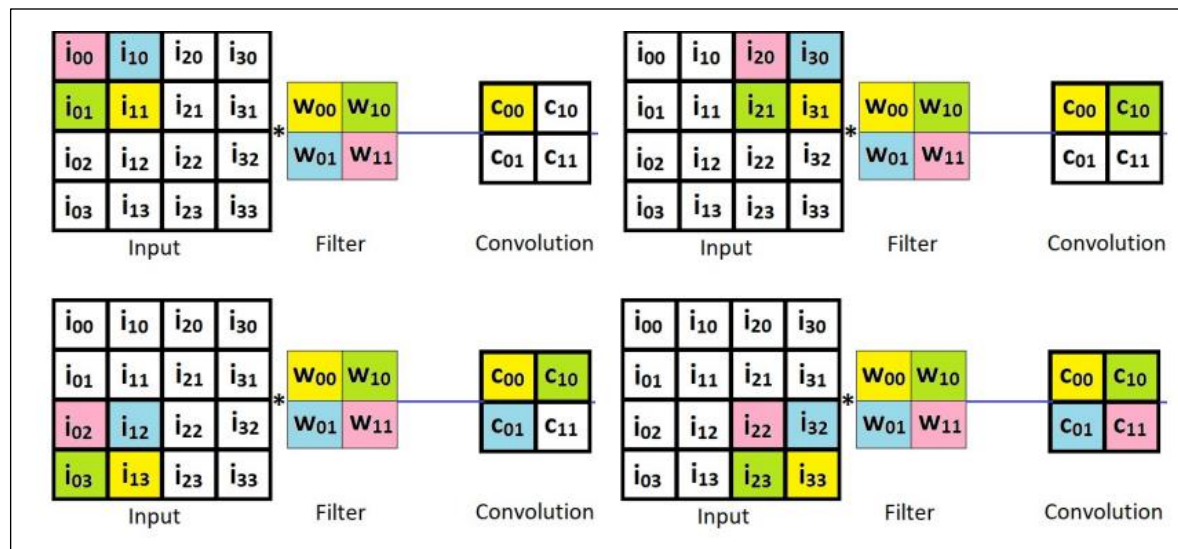
Convolution layer: CNN에서 핵심 역할을 맡는 층

필터: CNN에서 가중치 역할을 하는 파라미터
(이미지에서 어떤 특징을 뽑는 역할)

2 CNN의 구조

- Convolution layer

CNN의 구조와 동작 과정



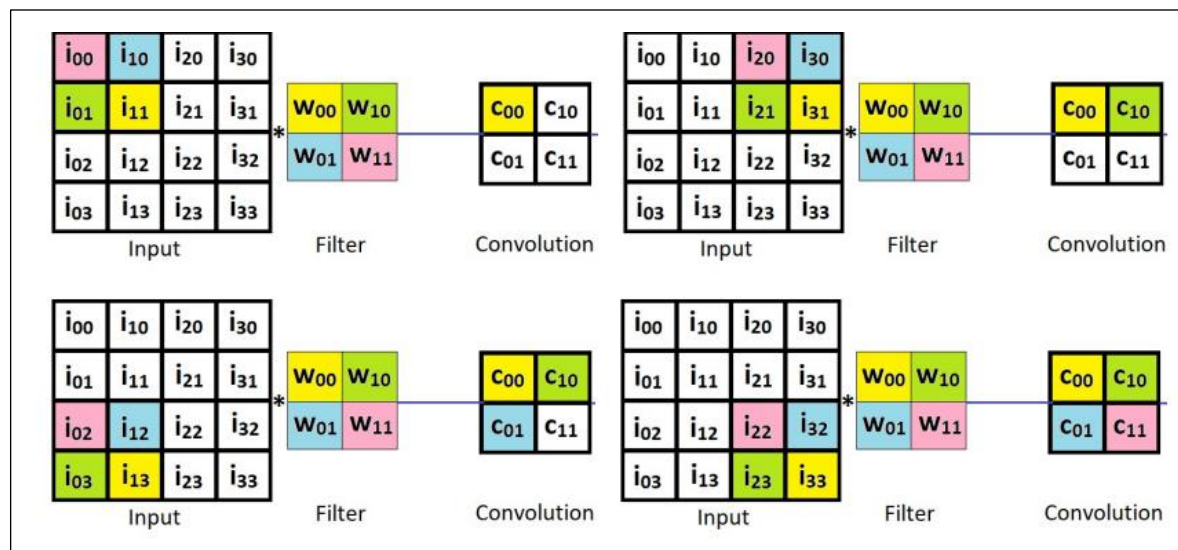
- 필터가 이미지를 순회하며 값 계산
- 합성곱 연산: 입력과 필터 사이에서 일어나는 연산

$$(c_{00} = i_{00}w_{11} + i_{10}w_{01} + i_{01}w_{10} + i_{11}w_{00})$$

2 CNN의 구조

- Convolution layer

CNN의 구조와 동작 과정



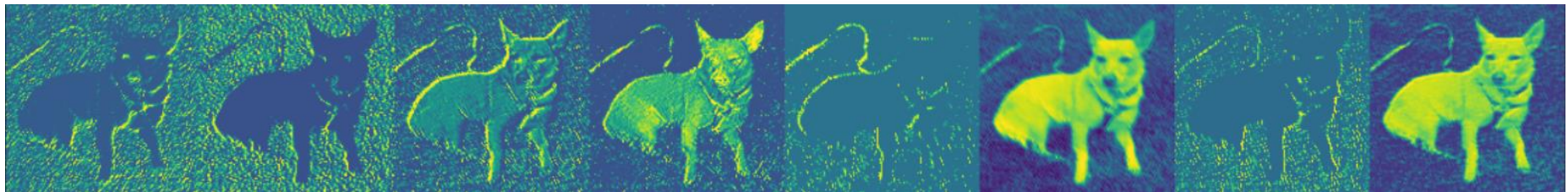
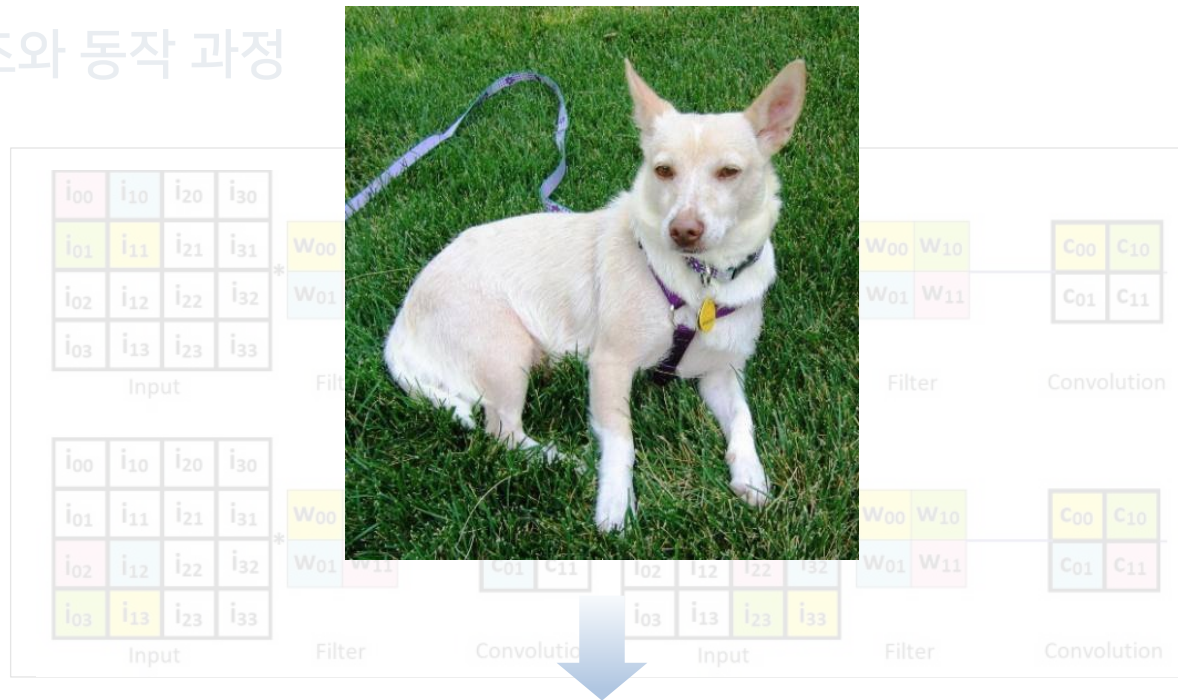
합성곱 연산을 거쳐 나온 결과 → feature map

✓ 필터를 거쳐 이미지가 갖고 있는 특징이 강조됨

2 CNN의 구조

- Convolution layer

CNN의 구조와 동작 과정



한 차례의 convolution layer를 통과한 feature map 예시

2 CNN의 구조

- Convolution layer

합성곱 연산

$$\begin{array}{|c|c|} \hline i_{00} & i_{10} \\ \hline i_{01} & i_{11} \\ \hline \end{array} * \begin{array}{|c|c|} \hline w_{00} & w_{10} \\ \hline w_{01} & w_{11} \\ \hline \end{array} = \begin{array}{|c|c|} \hline i_{00} & i_{10} \\ \hline i_{01} & i_{11} \\ \hline \end{array} \circ \begin{array}{|c|c|} \hline w_{11} & w_{01} \\ \hline w_{10} & w_{00} \\ \hline \end{array}$$

I F I 180° rotated F

필터를 180도 뒤집고

Hadamard Product(=cross-correlation)를 하는 것

두 행렬을 곱할 때

같은 위치에 있는 원소끼리 곱하는 것

2 CNN의 구조

- Convolution layer

CNN의 구조와 동작 과정

$$\begin{array}{|c|c|} \hline i_{00} & i_{10} \\ \hline i_{01} & i_{11} \\ \hline \end{array} * \begin{array}{|c|c|} \hline W_{00} & W_{10} \\ \hline W_{01} & W_{11} \\ \hline \end{array} = \begin{array}{|c|c|} \hline i_{00} & i_{10} \\ \hline i_{01} & i_{11} \\ \hline \end{array} \circ \begin{array}{|c|c|} \hline W_{11} & W_{01} \\ \hline W_{10} & W_{00} \\ \hline \end{array}$$

I F I 180° rotated F

Convolution layer



cross-correlation으로 구현



합성곱 연산의 경우,

필터를 뒤집어야 하는 추가적인 연산과 그에 따른 비용이 들기 때문

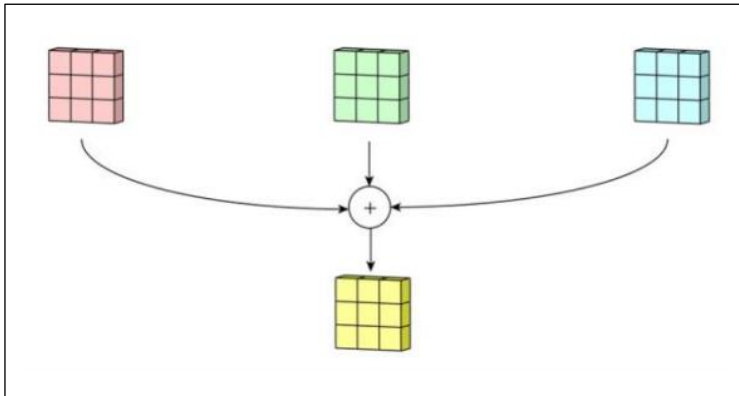
2 CNN의 구조

- Convolution layer

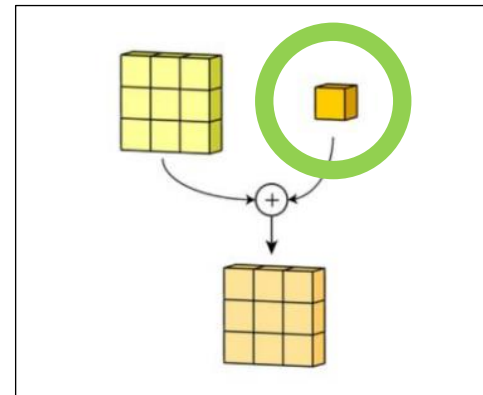
컬러 이미지가 합성곱 연산 통과하는 경우

- 각 채널마다 각각의 filter가 존재

① filter로 각 채널들의 feature map을 구함



② Bias를 더함




최종적인 feature map

2 CNN의 구조

- Convolution layer

합성곱 계층 하이퍼파라미터

CONV2D

 PyTorch에 구현

```
CLASS torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)
```

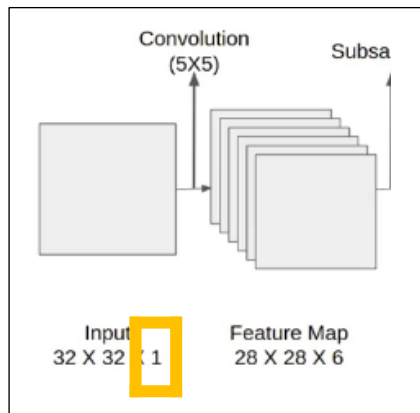
✓ 필수

in_channels	입력 이미지의 채널 개수	stride	필터가 한 번에 움직이는 정도
out_channels	출력할 채널의 수	padding	이미지의 테두리를 어떤 값으로 둘러싸는 것
kernel_size	필터의 크기		

2 CNN의 구조

- Convolution layer

합성곱 계층 하이퍼파라미터



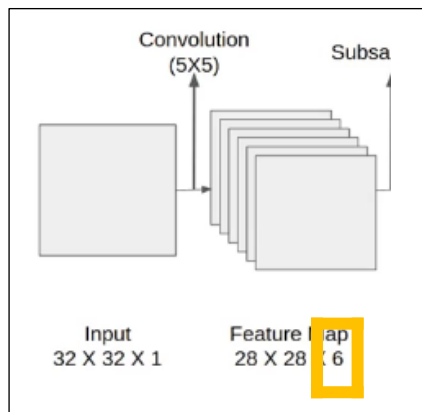
→ LeNet의 첫번째 convolution layer에
1개 채널 입력

✓ in_channels	입력 이미지의 채널 개수	stride	필터가 한 번에 움직이는 정도
out_channels	출력할 채널의 수	padding	이미지의 테두리를 어떤 값으로 둘러싸는 것
kernel_size	필터의 크기		

2 CNN의 구조

- Convolution layer

합성곱 계층 하이퍼파라미터



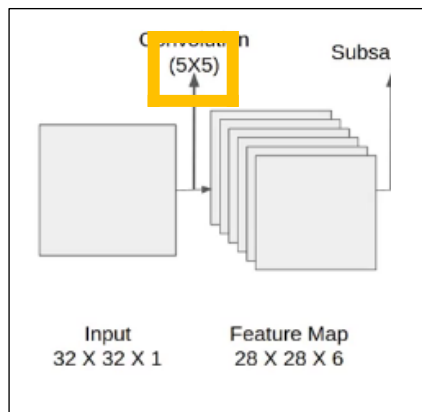
→ LeNet의 첫번째 convolution layer에서
6개 채널 출력

in_channels	입력 이미지의 채널 개수	stride	필터가 한 번에 움직이는 정도
✓ out_channels	출력할 채널의 수	padding	이미지의 테두리를 어떤 값으로 둘러싸는 것
kernel_size	필터의 크기		

2 CNN의 구조

● Convolution layer

합성곱 계층 하이퍼파라미터



- 값이 클수록 더 많은 픽셀을 대상으로 연산
 - ➔ 출력 feature map 크기 축소
 - 필터가 가진 값 = 가중치
 - ➔ 값이 클수록 가중치 많음을 의미

in_channels	입력 이미지의 채널 개수	stride	필터가 한 번에 움직이는 정도
out_channels	출력할 채널의 수	padding	이미지의 테두리를 어떤 값으로 둘러싸는 것
✓ kernel_size	필터의 크기		

일반적으로 홀수 이용

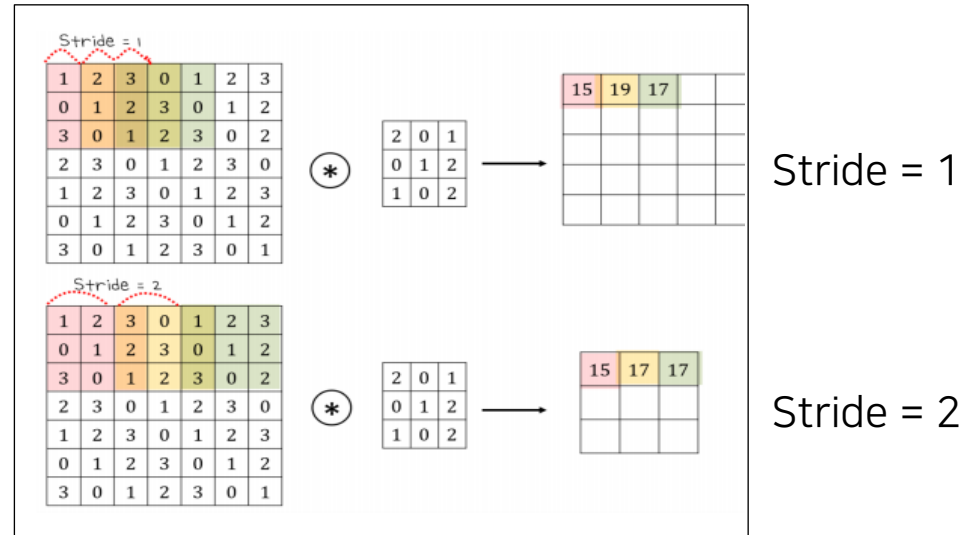
(짝수의 경우, 중심이 명확X, 이미지 왜곡 발생 가능)

2 CNN의 구조

● Convolution layer

합성곱 계층 하이퍼파라미터

값이 클수록,
feature map의 크기가 줄어들게 됨



in_channels	입력 이미지의 채널 개수	✓ stride	필터가 한 번에 움직이는 정도
out_channels	출력할 채널의 수	padding	이미지의 테두리를 어떤 값으로 둘러싸는 것
kernel_size	필터의 크기		

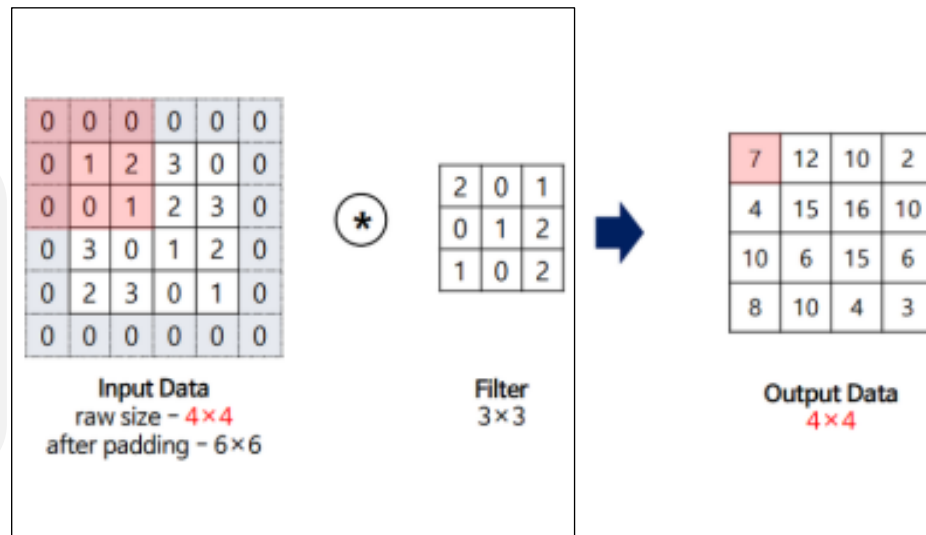
2 CNN의 구조

- Convolution layer

합성곱 계층 하이퍼파라미터

패딩 추가 시, feature map의 크기 유지 가능

입력 이미지의 모서리는 연산에 한 번 포함되고
중심부는 많이 연산 되는 문제 해결 가능



in_channels	입력 이미지의 채널 개수	stride	필터가 한 번에 움직이는 정도
out_channels	출력할 채널의 수	✓ padding	이미지의 테두리를 어떤 값으로 둘러싸는 것
kernel_size	필터의 크기		

2 CNN의 구조

- Convolution layer

합성곱 계층의 출력 크기 계산

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

합성곱 계층의 출력 feature map의 크기 계산 가능

OH(Output Height), H(Input Height), P(Padding),
FH(Filter Height), S(Stride), OW(Output Width),
W(Input Width), P(Padding), FW(Filter Width)

2 CNN의 구조

- Pooling layer

Pooling layer

Feature map의 크기를 줄이는 연산

활성화 함수를 통과한
feature map

Activation Map

12	20	30	0
8	12	2	0
34	70	37	7
112	100	22	12

Max Pooling

20	30
112	37

Average Pooling

13	8
79	18

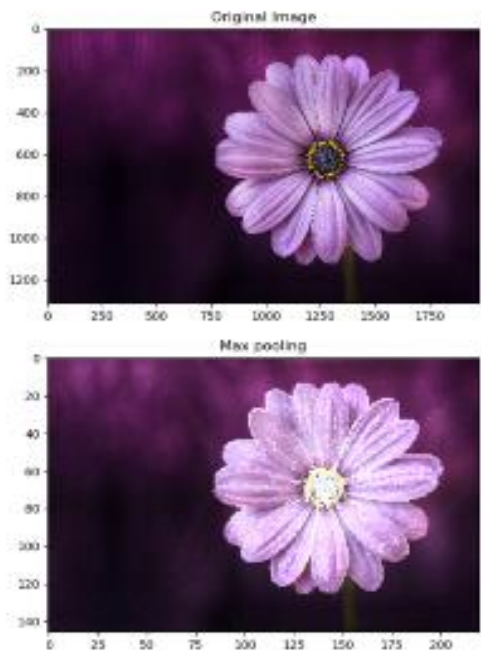
✓
필터의 크기, stride 가
2인 경우

필터(=window)의 크기, stride 값에 맞게 필터가 입력을 순회하며 값 계산

2 CNN의 구조

- Pooling layer

Pooling의 종류

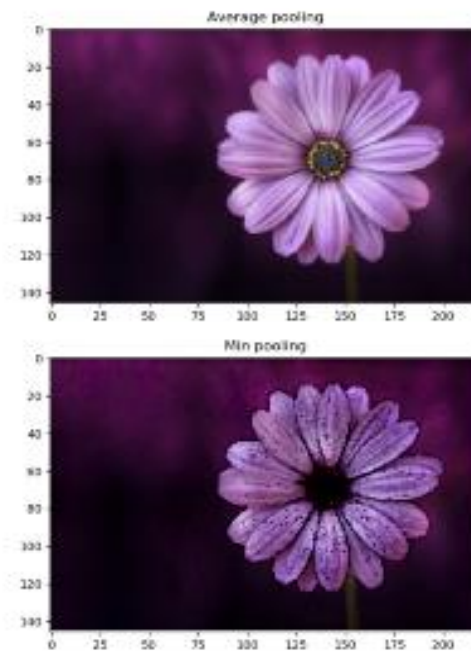


Max Pooling

필터가 덮고 있는 가장 큰 값 출력

Average Pooling

필터가 덮고 있는 값의 평균 출력



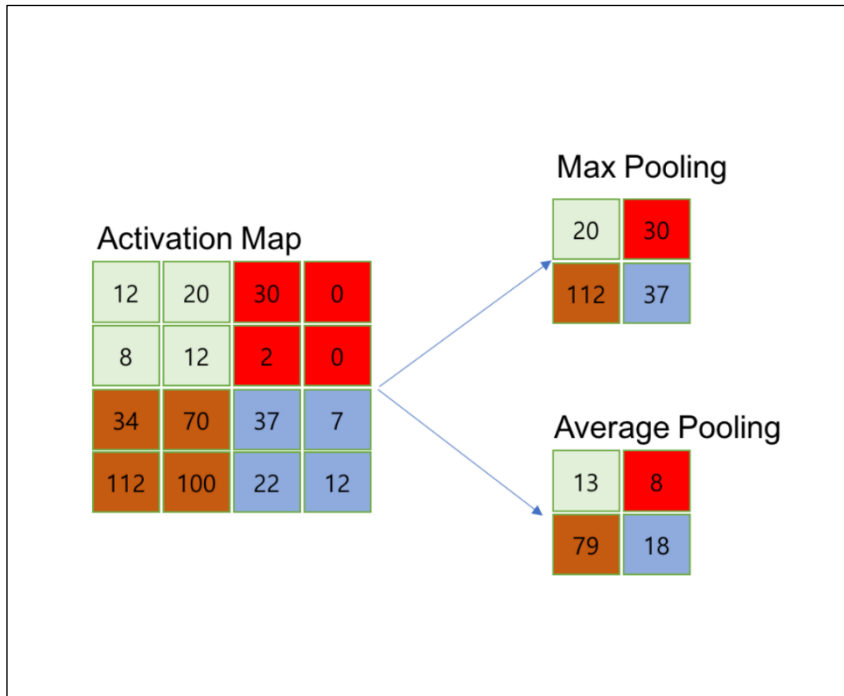
Min Pooling

필터가 덮고 있는 가장 작은 값 출력

2 CNN의 구조

- Pooling layer

Pooling layer의 장점



- Feature map에서 filter가 위치한 곳의 특징을 뽑아내 이미지 크기 축소
- 이미지에서 위치의 변화에 비교적 강건
- 추가적으로 학습해야 할 파라미터 X

3

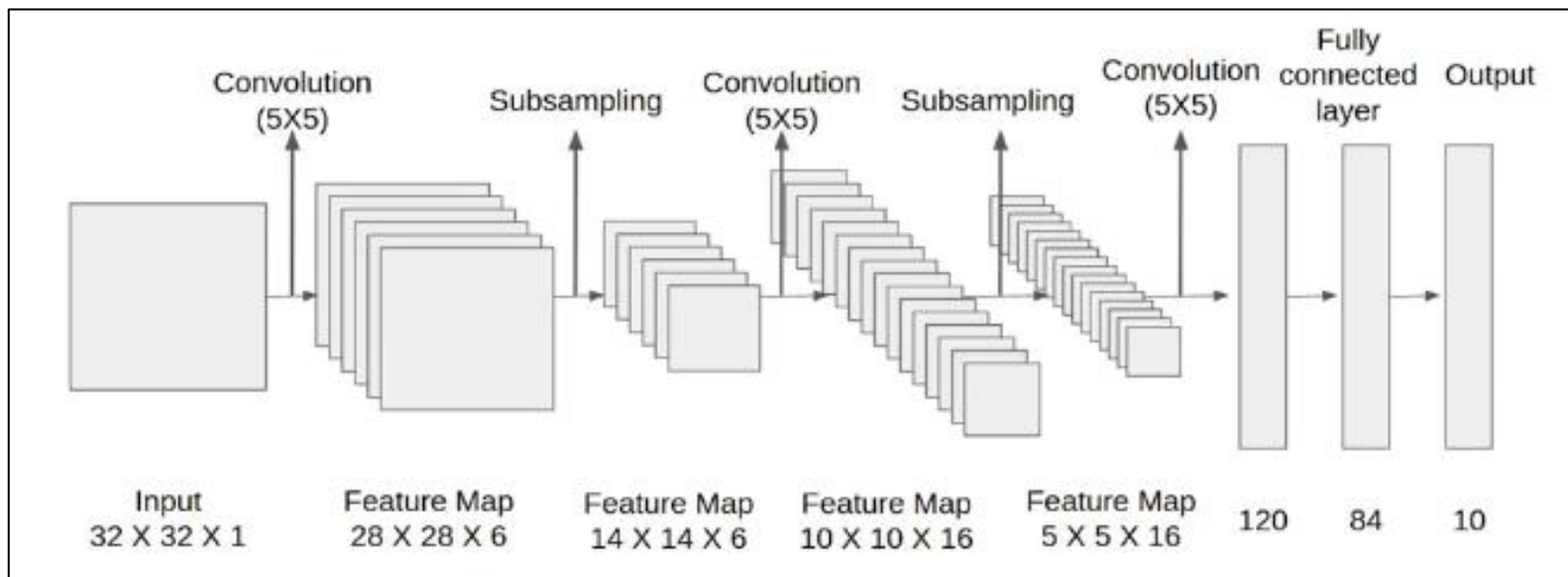
CNN 모델의 발전

3 CNN 모델의 발전

● LeNet-5

LeNet의 특징

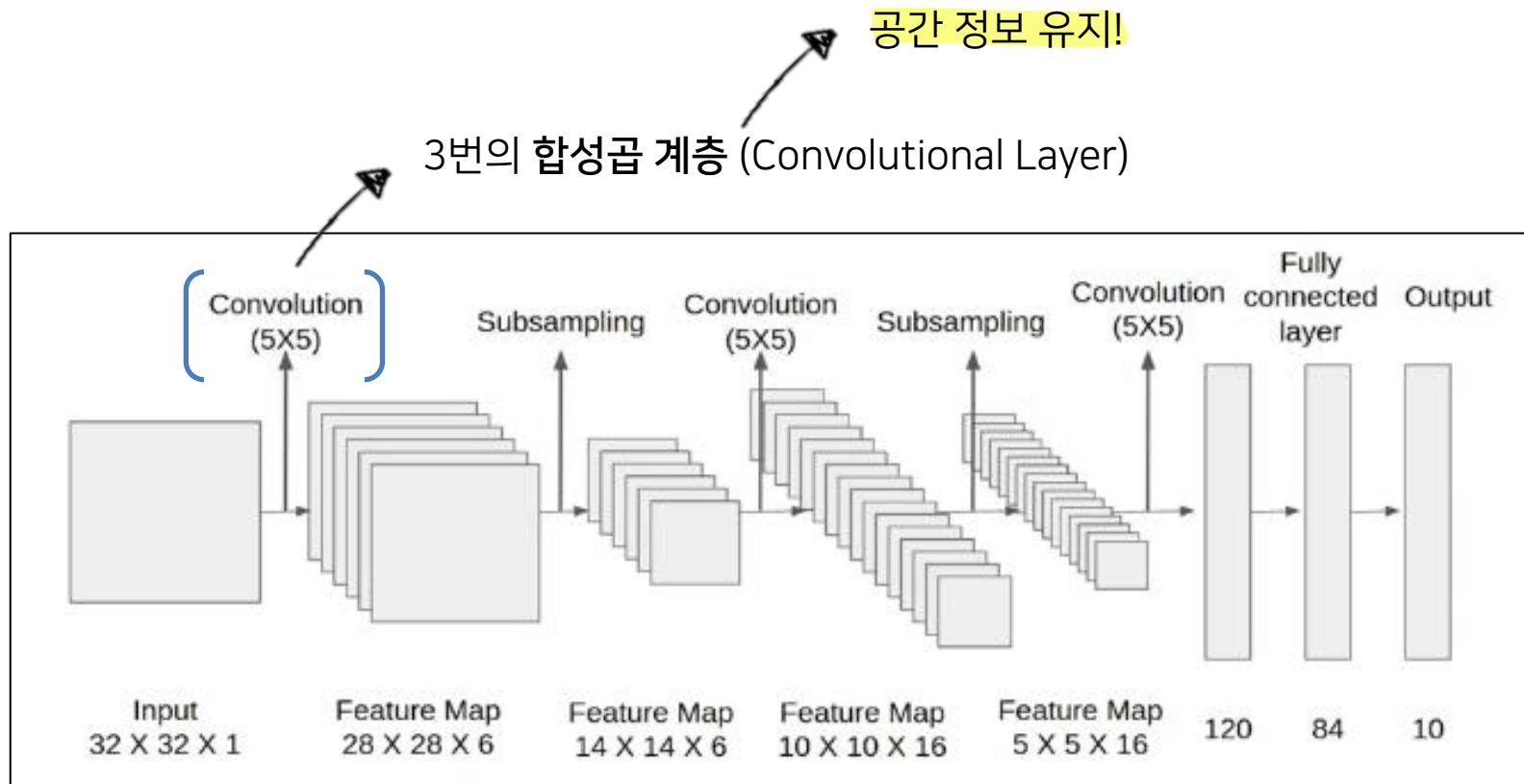
- 가장 처음 제안된 CNN 모델
- 손으로 쓴 숫자를 구분하기 위한 모델



3 CNN 모델의 발전

- LeNet-5

LeNet의 구조: Convolutional Layer

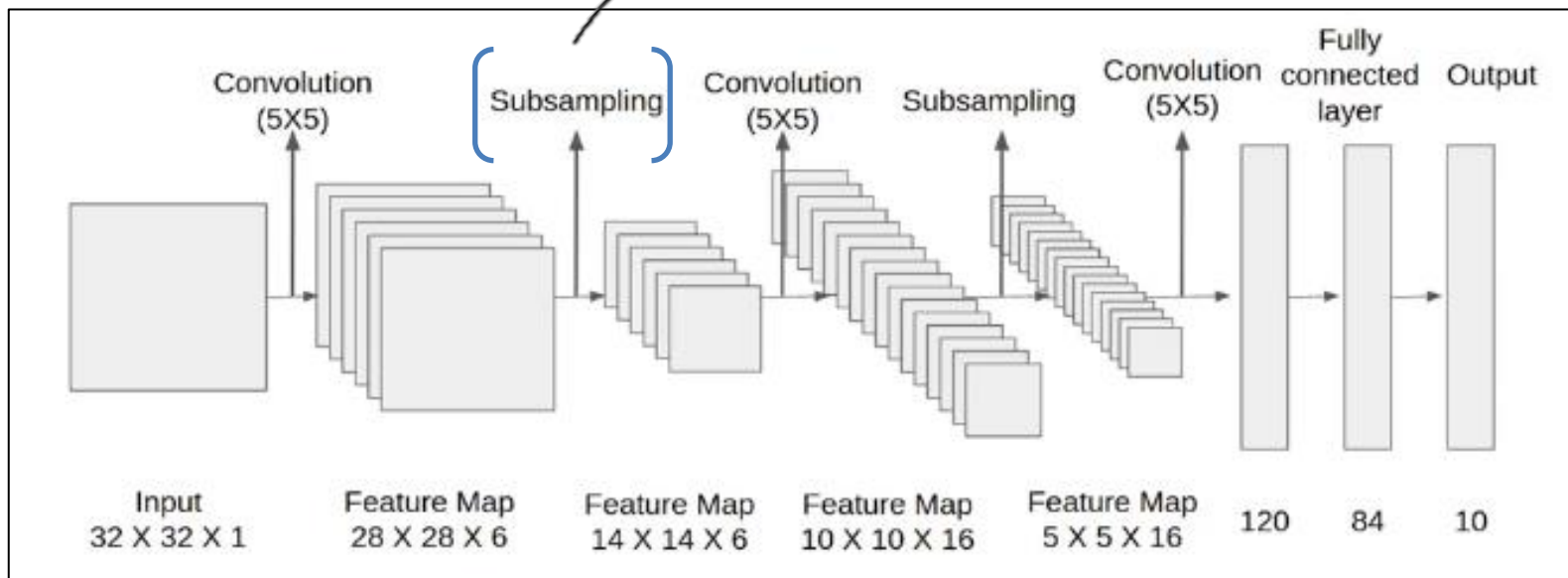


3 CNN 모델의 발전

- LeNet-5

LeNet의 구조: Average Pooling

Average Pooling 활용 (현대의 Pooling Layer)

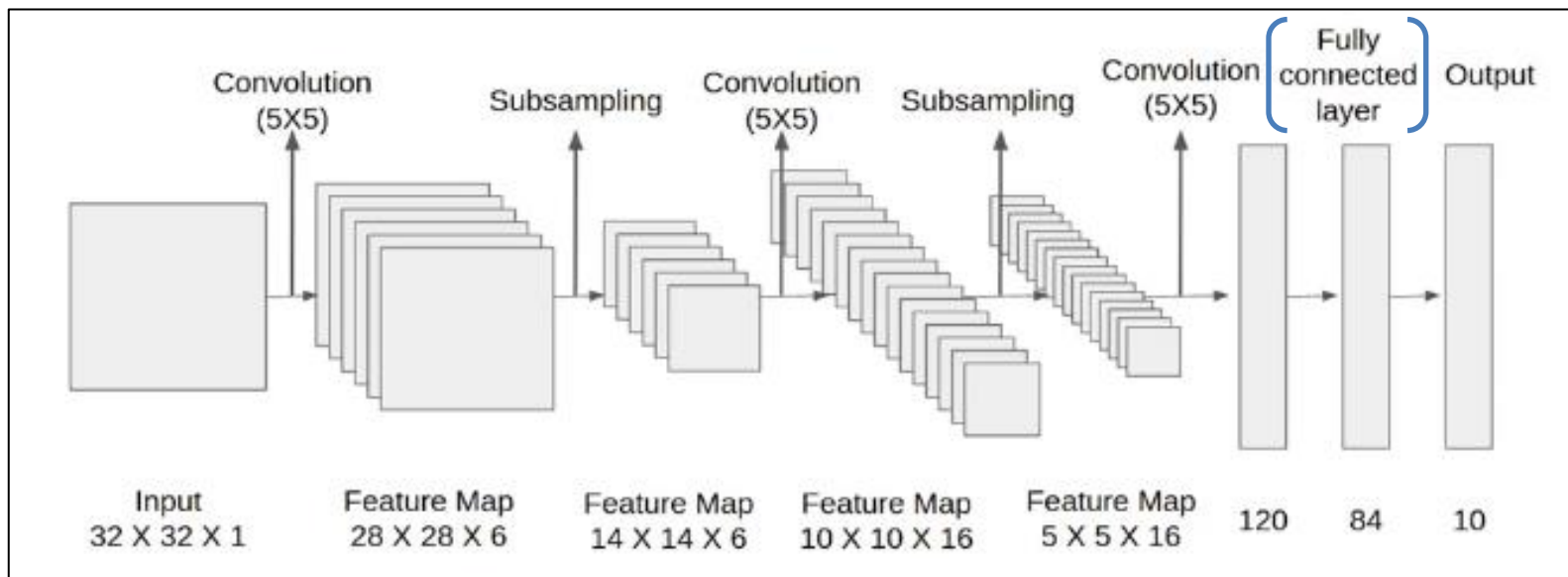


3 CNN 모델의 발전

- LeNet-5

LeNet의 구조: FC Layer

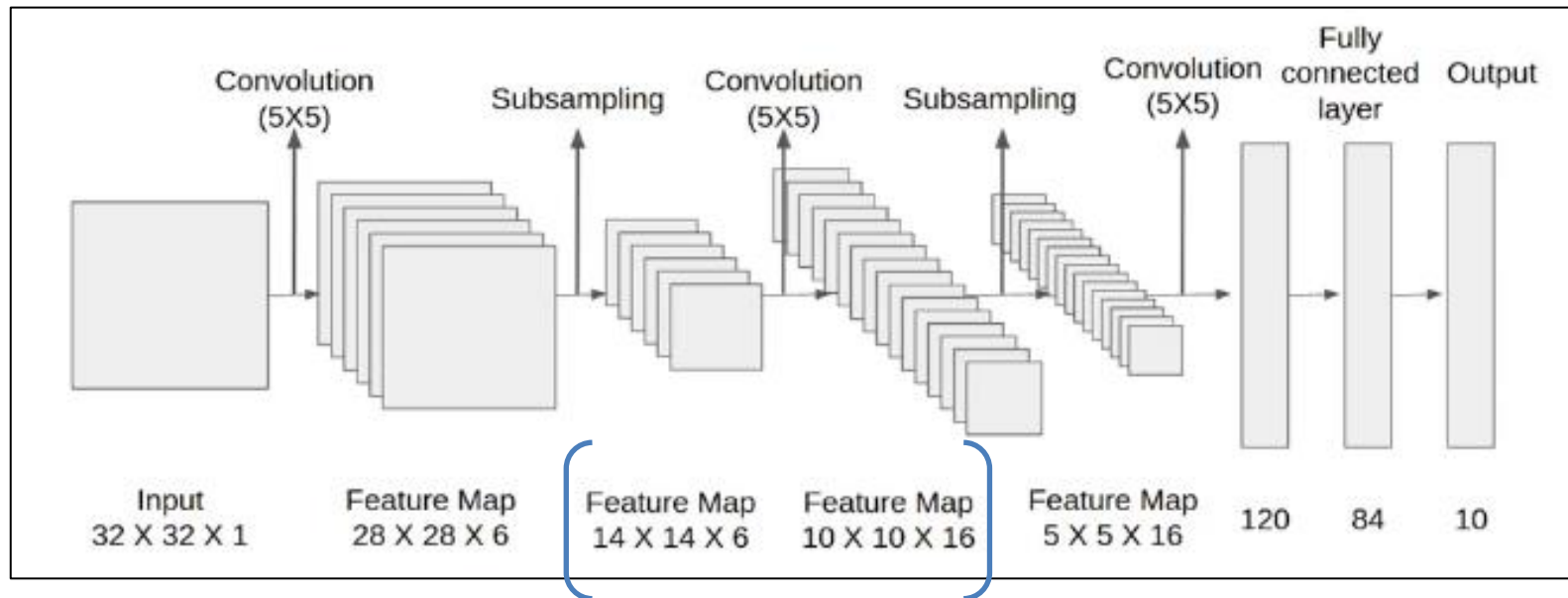
2차례의 FC Layer를 거쳐 10개의 출력값
(활성화 함수 = Softmax)



3 CNN 모델의 발전

- LeNet-5

LeNet의 구조: Feature Map



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

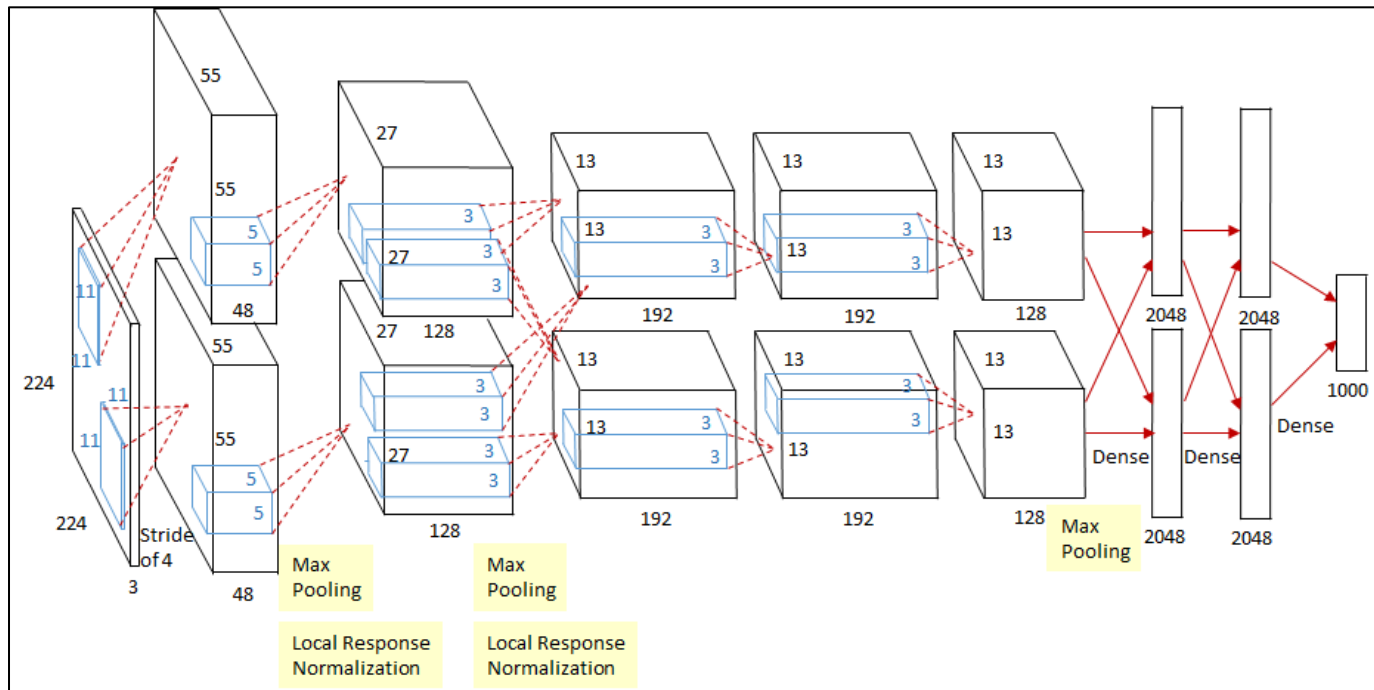
6개의 Feature Map의 **일부만을 활용**하여
16개의 새로운 Feature Map 출력

3 CNN 모델의 발전

AlexNet

AlexNet의 특징

- ILSVRC-2012에서 압도적 1등 차지
- 224 * 224의 컬러 이미지 처리를 위해 병렬 처리



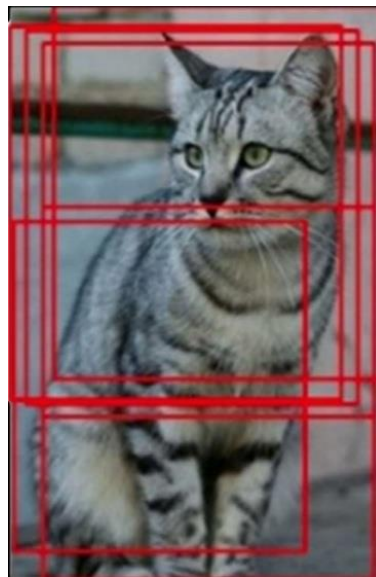
3 CNN 모델의 발전

- AlexNet

AlexNet의 구조: Data Augmentation

Size	Layer	Detail
[224x224x3]	INPUT	.
[55x55x96]	CONV1	96 11x11 filters at stride 4, pad 0
[27x27x96]	MAX POOL1	3x3 filters at stride 2
[27x27x96]	NORM1	Normalization layer
[27x27x256]	CONV2	256 5x5 filters at stride 1, pad 2
[13x13x256]	MAX POOL2	3x3 filters at stride 2
[13x13x256]	NORM2	Normalization layer
[13x13x384]	CONV3	384 3x3 filters at stride 1, pad 1
[13x13x384]	CONV4	384 3x3 filters at stride 1, pad 1
[13x13x256]	CONV5	256 3x3 filters at stride 1, pad 1
[6x6x256]	MAX POOL3	3x3 filters at stride 1, pad 1
[4096]	FC6	4096 neurons
[4096]	FC7	4096 neurons
[1000]	FC8	1000 neurons (class scores)

- Data Augmentation



256 * 256 크기의 원본 1장

Crop!

224 * 224 크기의 이미지
1024장

Flip!

224 * 224 크기의 이미지
2048장

3 CNN 모델의 발전

● AlexNet

AlexNet의 구조: Max Pooling

Size	Layer	Detail
[224x224x3]	INPUT	•
[55x55x96]	CONV1	96 11x11 filters at stride 4, pad 0
[27x27x96]	MAX POOL1	3x3 filters at stride 2
[27x27x96]	NORM1	Normalization layer
[27x27x256]	CONV2	256 5x5 filters at stride 1, pad 2
[13x13x256]	MAX POOL2	3x3 filters at stride 2
[13x13x256]	NORM2	Normalization layer
[13x13x384]	CONV3	384 3x3 filters at stride 1, pad 1
[13x13x384]	CONV4	384 3x3 filters at stride 1, pad 1
[13x13x256]	CONV5	256 3x3 filters at stride 1, pad 1
[6x6x256]	MAX POOL3	3x3 filters at stride 1, pad 1
[4096]	FC6	4096 neurons
[4096]	FC7	4096 neurons
[1000]	FC8	1000 neurons (class scores)

① ReLU 함수 통과

② Pooling Layer (Max Pooling) 통과

Window Size > Stride



Pooling Window 겹침
(약간의 성능 향상)

3 CNN 모델의 발전

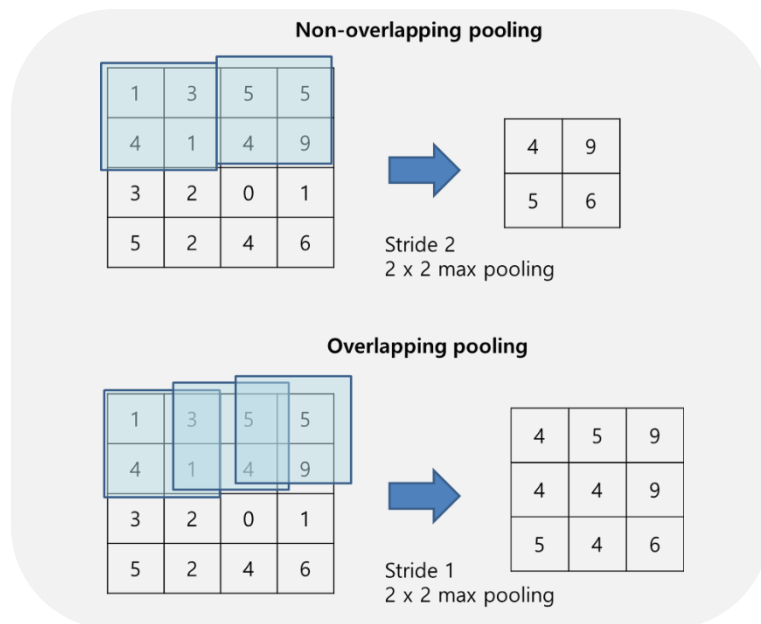
- AlexNet

AlexNet의 구조: Max Pooling

Size	Layer	Detail
[224x224x3]	INPUT	•
[55x55x96]	CONV1	96 11x11 filters at stride 4, pad 0
[27x27x96]	MAX POOL1	3x3 filters at stride 2
[27x27x96]	NORM1	Normalization layer
[27x27x256]	CONV2	256 5x5 filters at stride 1, pad 2
[13x13x256]	MAX POOL2	3x3 filters at stride 2
[13x13x256]	NORM2	Normalization layer
[13x13x384]	CONV3	384 3x3 filters at stride 1, pad 1
[13x13x384]	CONV4	384 3x3 filters at stride 1, pad 1
[13x13x256]	CONV5	256 3x3 filters at stride 1, pad 1
[6x6x256]	MAX POOL3	3x3 filters at stride 1, pad 1
[4096]	FC6	4096 neurons
[4096]	FC7	4096 neurons
[1000]	FC8	1000 neurons (class scores)

① ReLU 함수 통과

② Pooling Layer (Max Pooling) 통과



3 CNN 모델의 발전

- AlexNet

AlexNet의 구조: Normalization

Size	Layer	Detail
[224x224x3]	INPUT	•
[55x55x96]	CONV1	96 11x11 filters at stride 4, pad 0
[27x27x96]	MAX POOL1	3x3 filters at stride 2
[27x27x96]	NORM1	Normalization layer
[27x27x256]	CONV2	256 5x5 filters at stride 1, pad 2
[13x13x256]	MAX POOL2	3x3 filters at stride 2
[13x13x256]	NORM2	Normalization layer
[13x13x384]	CONV3	384 3x3 filters at stride 1, pad 1
[13x13x384]	CONV4	384 3x3 filters at stride 1, pad 1
[13x13x256]	CONV5	256 3x3 filters at stride 1, pad 1
[6x6x256]	MAX POOL3	3x3 filters at stride 1, pad 1
[4096]	FC6	4096 neurons
[4096]	FC7	4096 neurons
[1000]	FC8	1000 neurons (class scores)

① ReLU 함수 통과

② Pooling Layer (Max Pooling) 통과

③ Normalization 적용

①과 ②의 결과로 Max 값 반환
하지만, 최솟값은 0



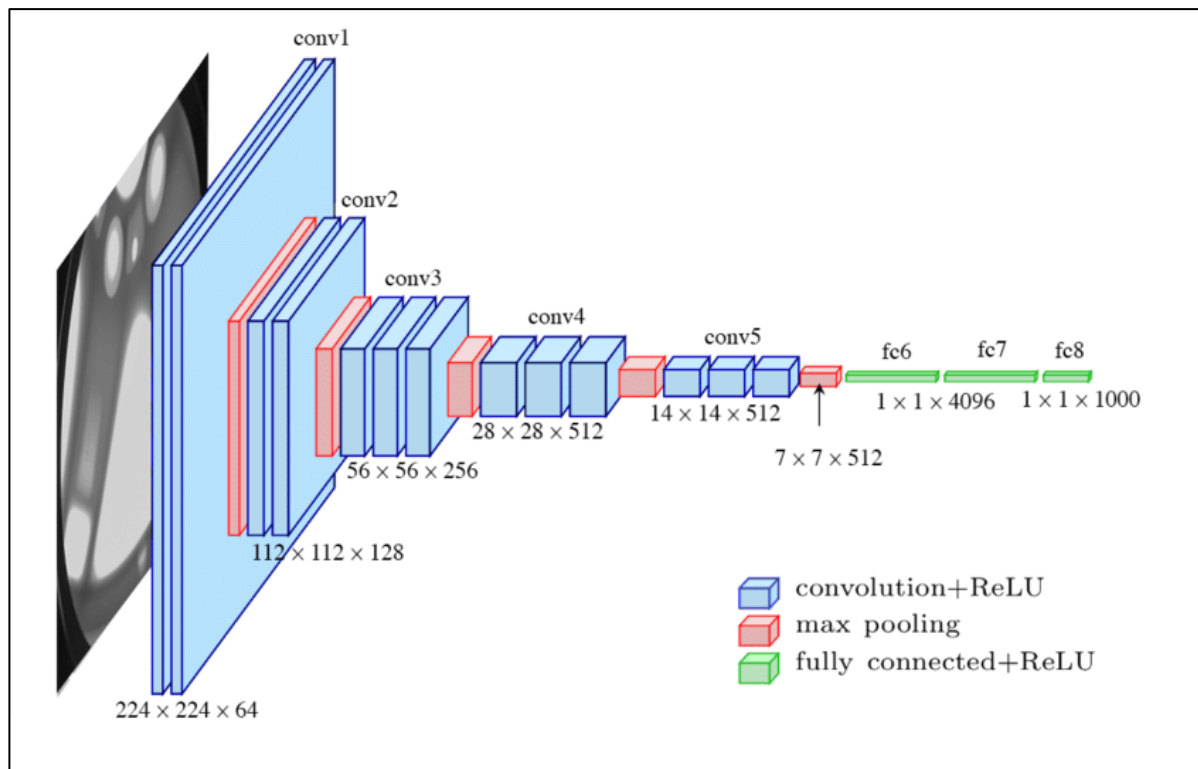
Normalization을 통해
Scale을 맞춰줌

3 CNN 모델의 발전

● VGGNet

VGGNet의 특징

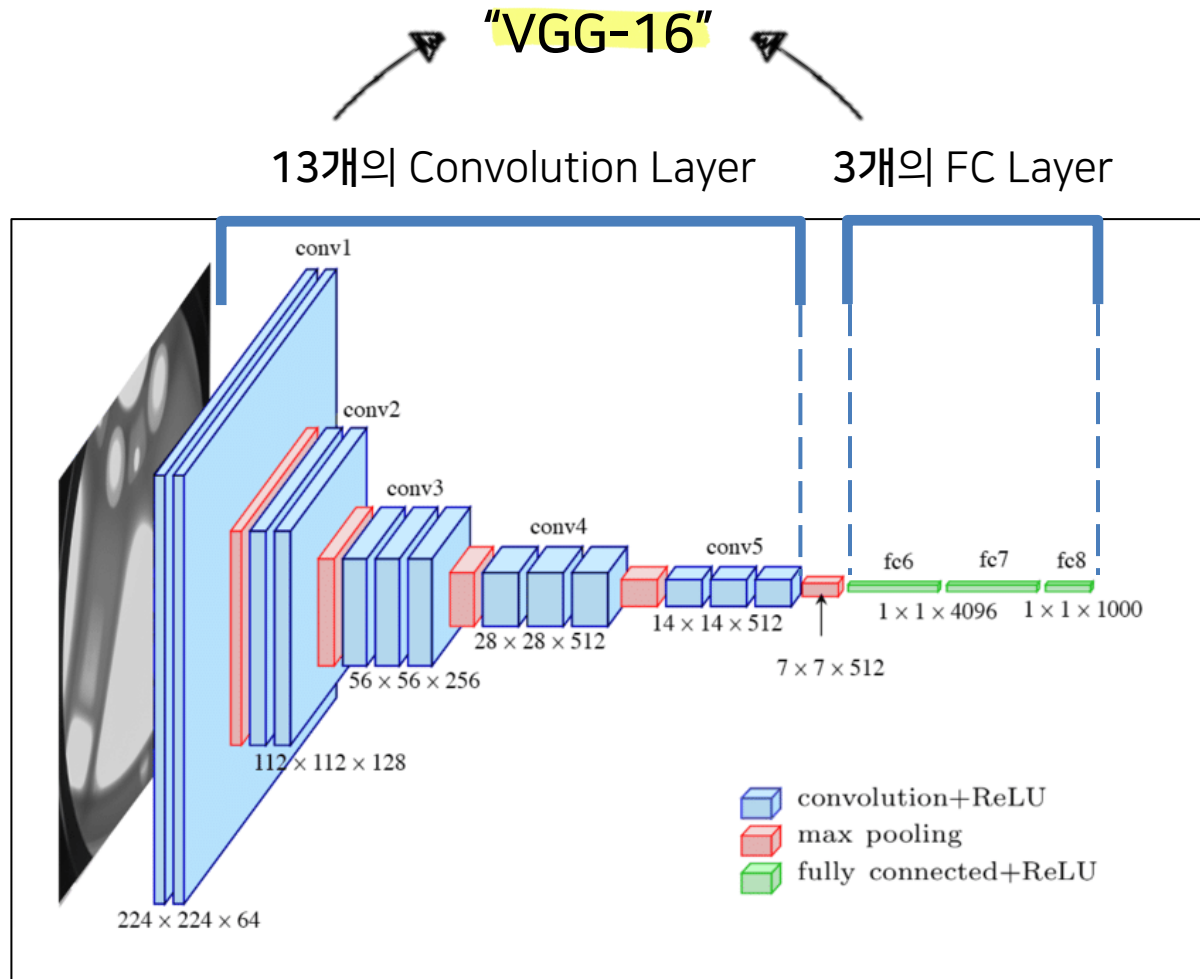
- 2014년 ILSVRC에서 1위를 한 GoogLeNet에 비해 훨씬 간단
- 층이 깊어질수록 성능이 높아짐을 확인



3 CNN 모델의 발전

- VGGNet

VGGNet의 구조: Layers



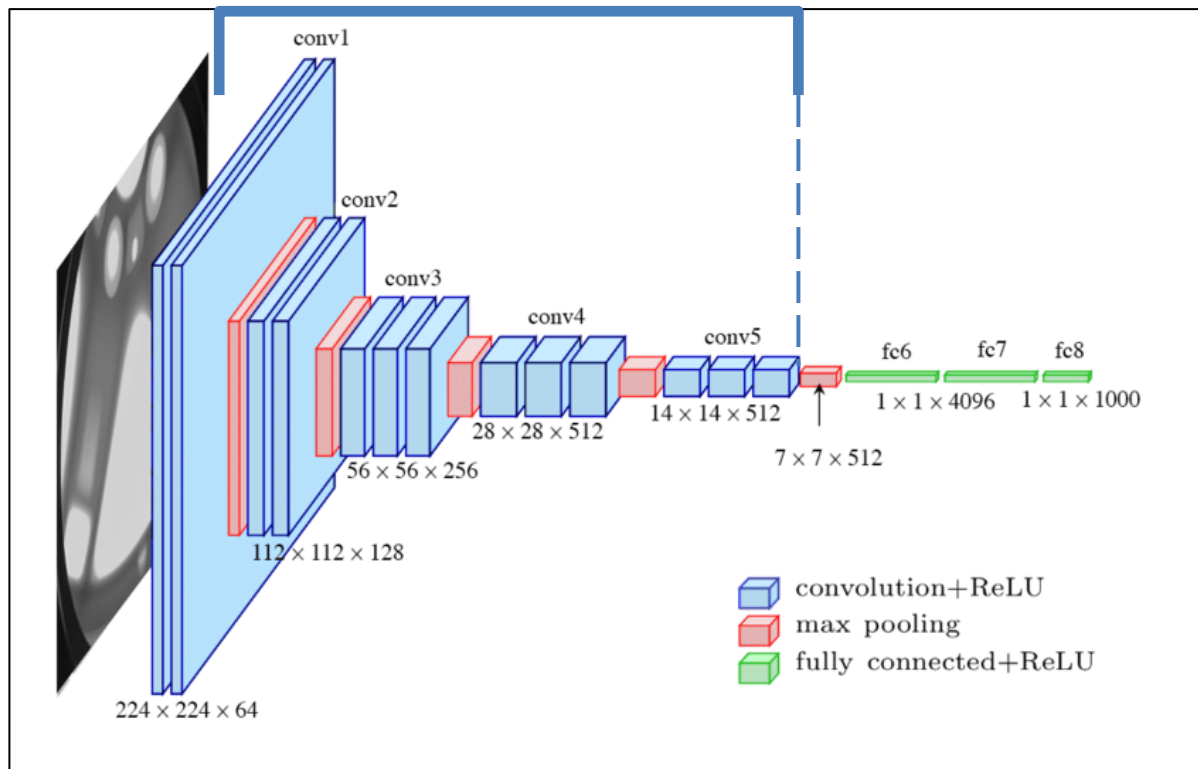
3 CNN 모델의 발전

- VGGNet

VGGNet의 구조: Layers

“VGG-16” > { LeNet: 3개의 Conv. Layer
AlexNet: 5개의 Conv. Layer

13개의 Convolution Layer



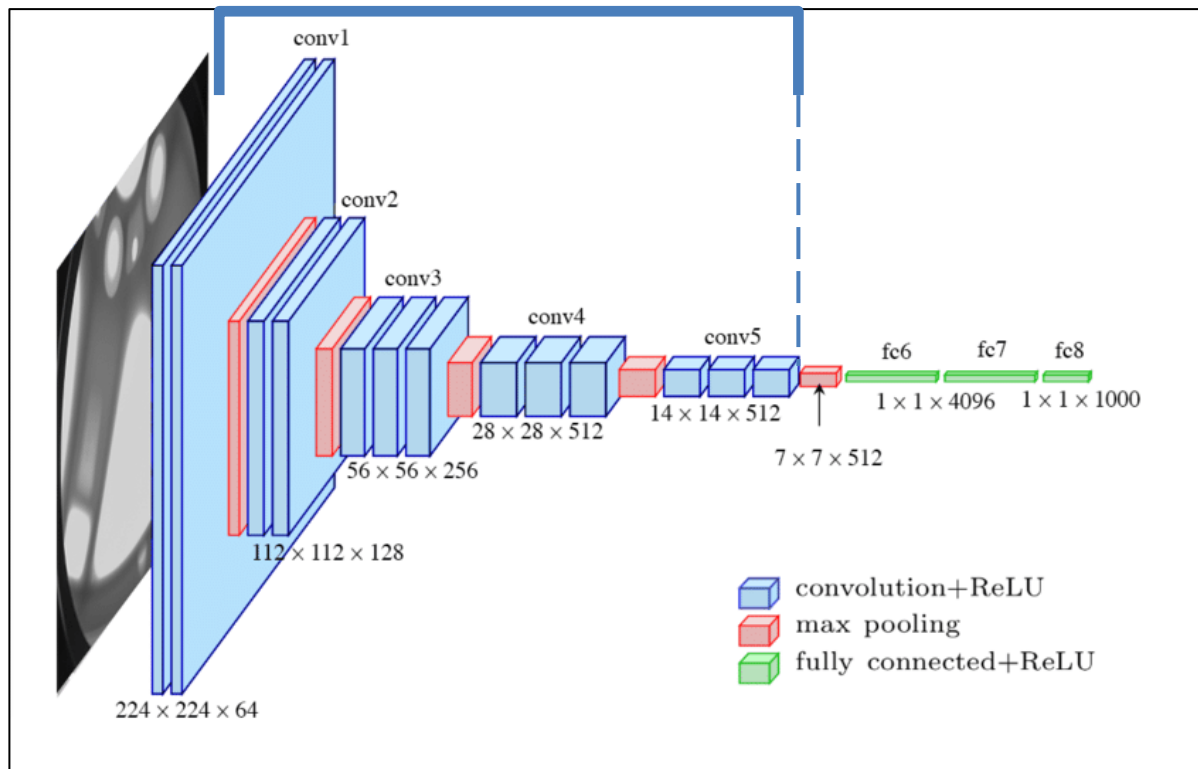
3 CNN 모델의 발전

- VGGNet

VGGNet의 구조: Layers

“VGG-16” =
13개의 Convolution Layer

층이 깊어질수록
더 좋은 성능을 보인다!

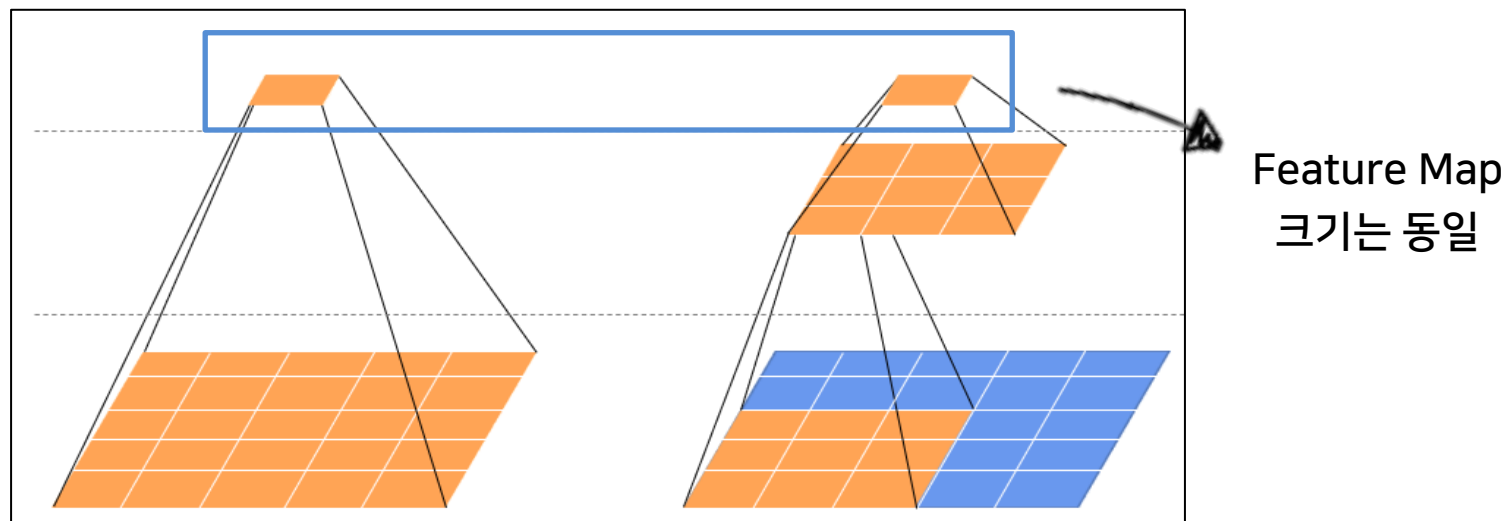


3 CNN 모델의 발전

- VGGNet

VGGNet의 구조: Filter Size

Conv. Layer의 Filter Size는 모두 (3, 3)



5 * 5 합성곱 연산 1번

3 * 3 합성곱 연산 2번

↓
파라미터: $5^2 C^2$ 개

>

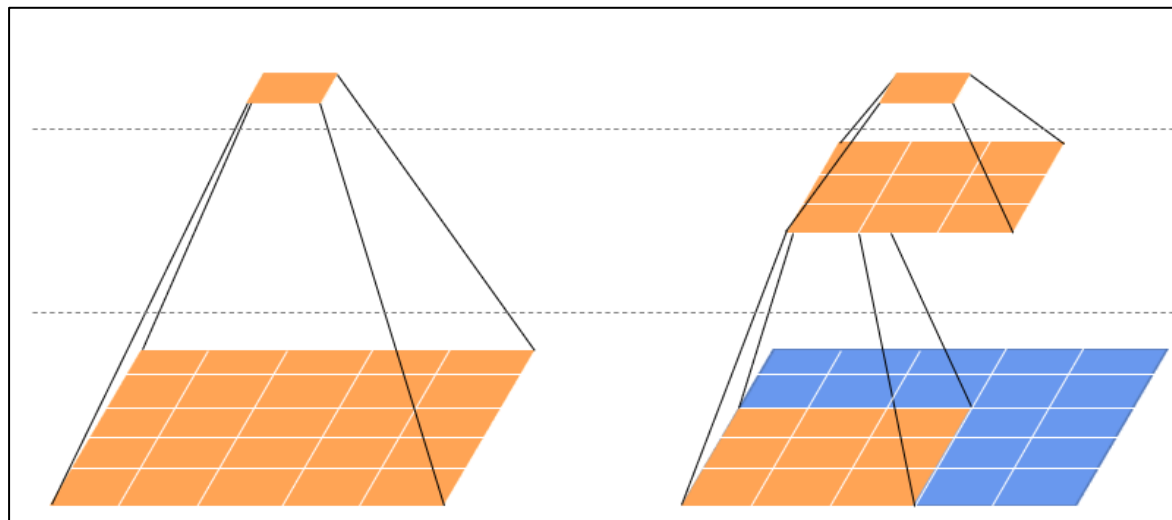
↓
파라미터: $2(3^2 C^2)$ 개

3 CNN 모델의 발전

- VGGNet

VGGNet의 구조: Filter Size

Conv. Layer의 Filter Size는 모두 (3, 3)



5 * 5 합성곱 연산 1번

3 * 3 합성곱 연산 2번



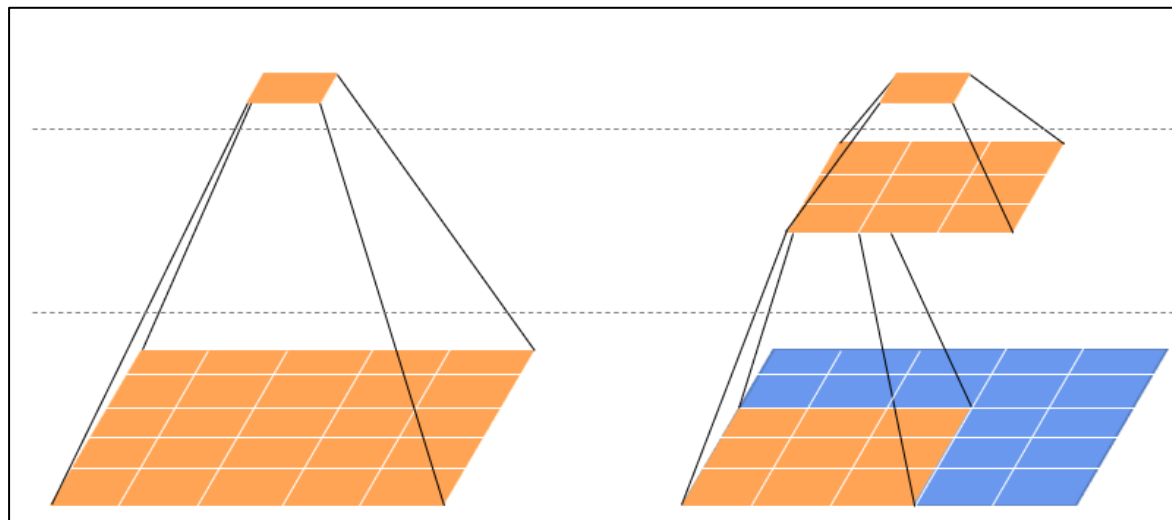
장점 : 모델에 비선형성 추가

3 CNN 모델의 발전

- VGGNet

VGGNet의 구조: Filter Size

Conv. Layer의 Filter Size는 모두 (3, 3)



5 * 5 합성곱 연산 1번

3 * 3 합성곱 연산 2번

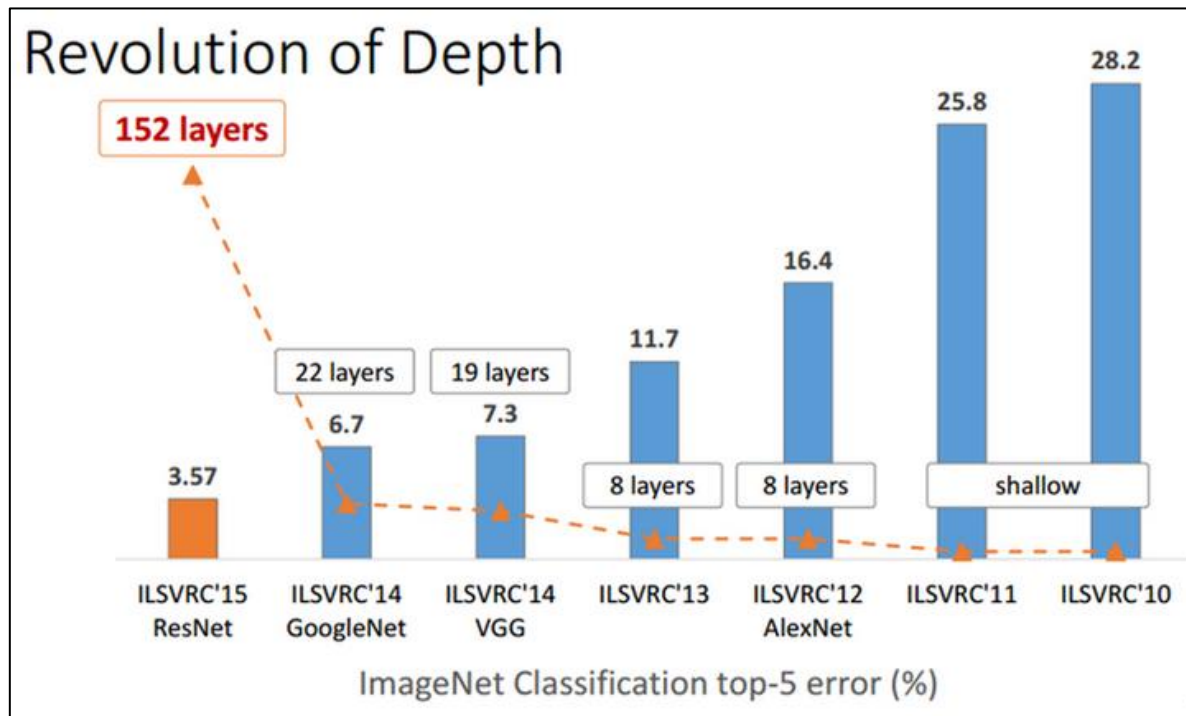
✓ 장점 : 파라미터 수를 줄여 더 가벼운 모델

3 CNN 모델의 발전

● ResNet

ResNet의 특징

- 기존 CNN과 비교해 7배가 넘는 깊이
- Residual Learning의 도입

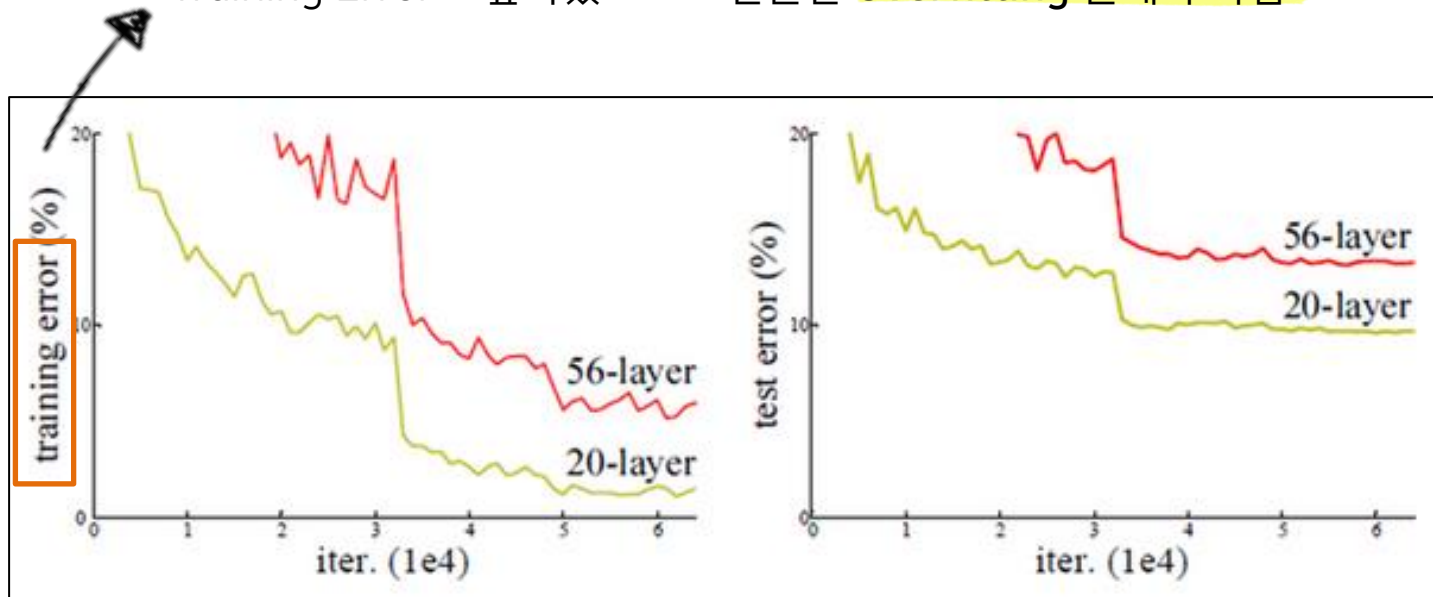


3 CNN 모델의 발전

- ResNet

ResNet의 배경

Training Error도 높아졌으므로 단순한 **Overfitting** 문제가 아님



모델의 깊이를 늘릴수록 성능이 더 좋아질 것을 기대했지만, 실제로 성능 감소

∴ Gradient Vanishing / Exploding 문제 심화

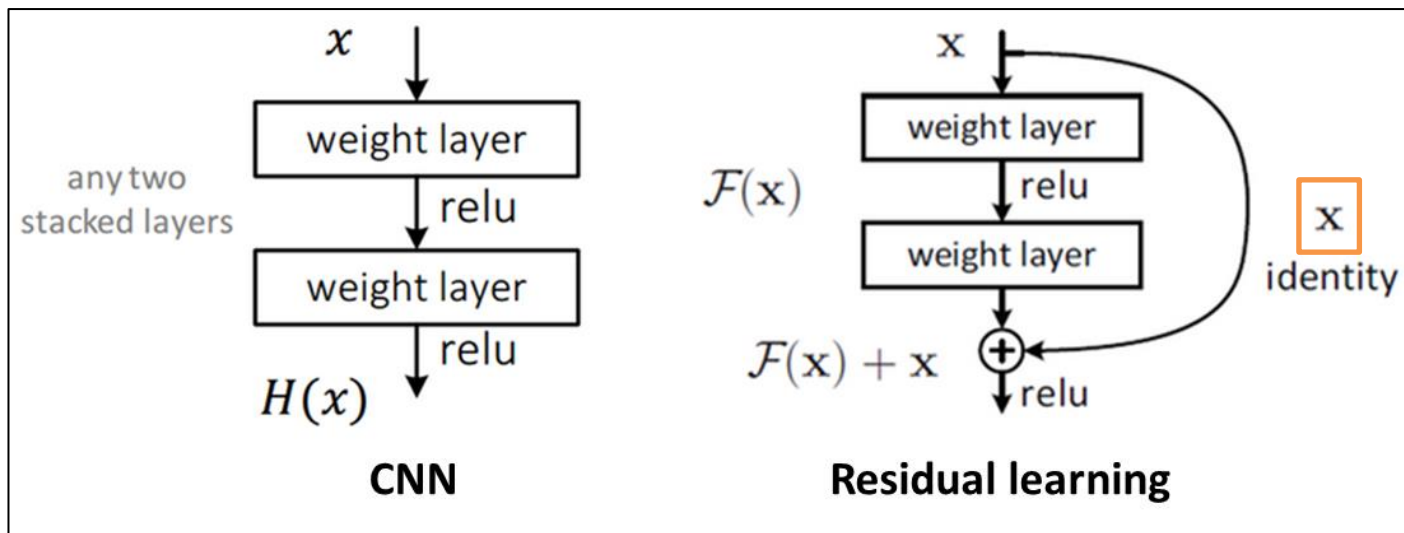


Residual Learning

3 CNN 모델의 발전

- ResNet

ResNet의 구조: Residual Learning

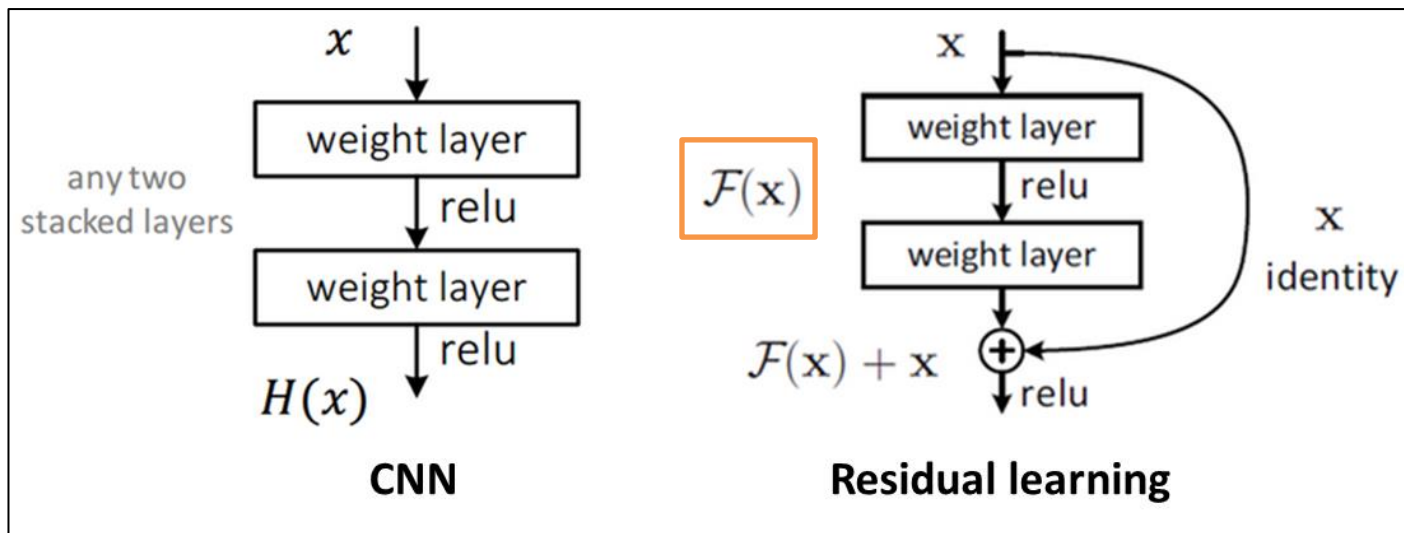


① 위 층에서 학습했던 것들(x)을 그대로 출력층으로 보낸다.

3 CNN 모델의 발전

- ResNet

ResNet의 구조: Residual Learning

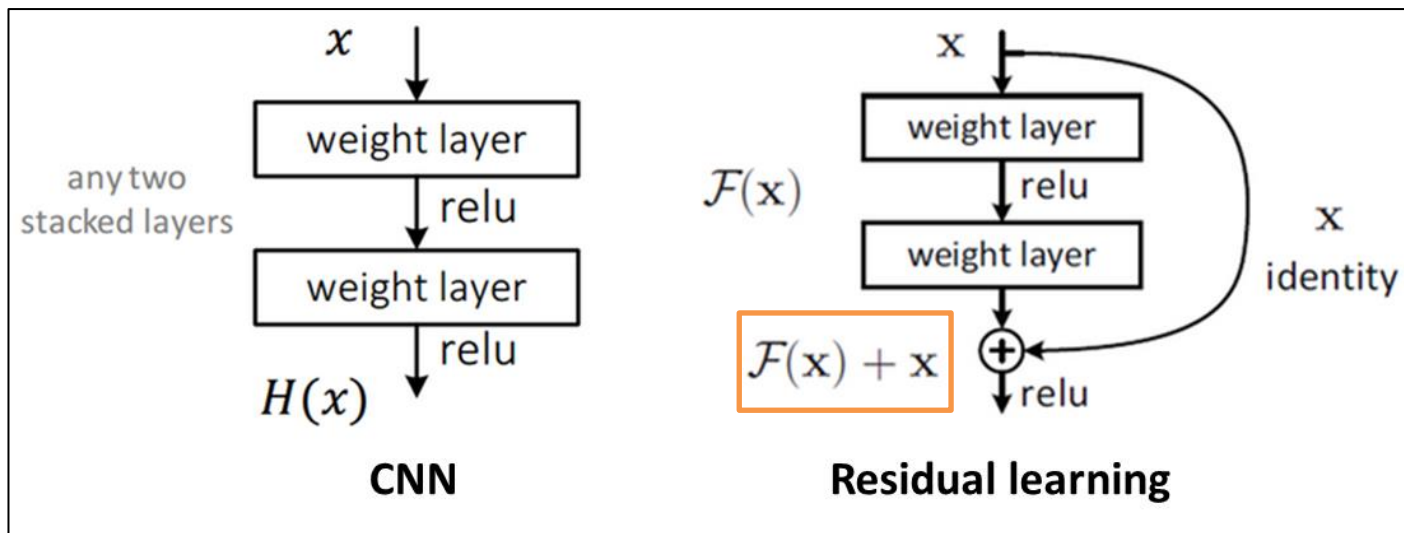


② 학습하지 못한 추가적인 정보들을 $F(x)$ 에서 학습

3 CNN 모델의 발전

- ResNet

ResNet의 구조: Residual Learning

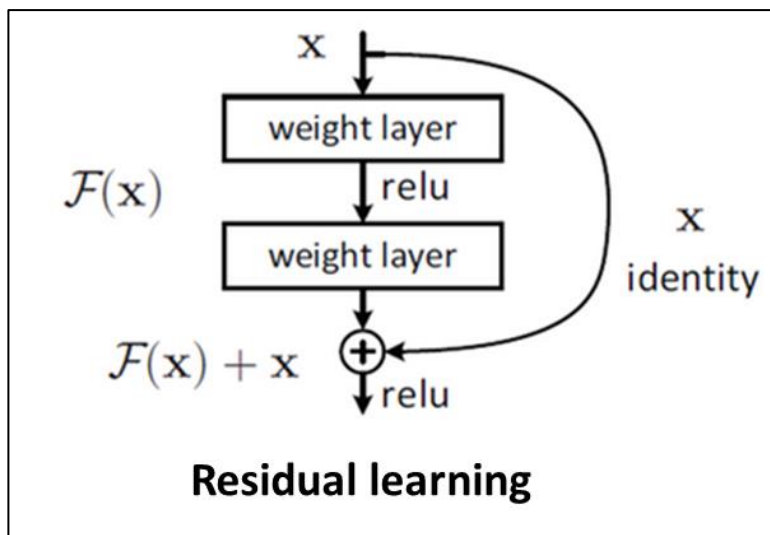


③ $F(x) + x$: 이전에 학습하지 못한 정보들을 추가적으로 학습

3 CNN 모델의 발전

- ResNet

ResNet의 구조: Residual Learning



- 역전파시 Gradient Vanishing 문제 개선

$$F(x) + x$$

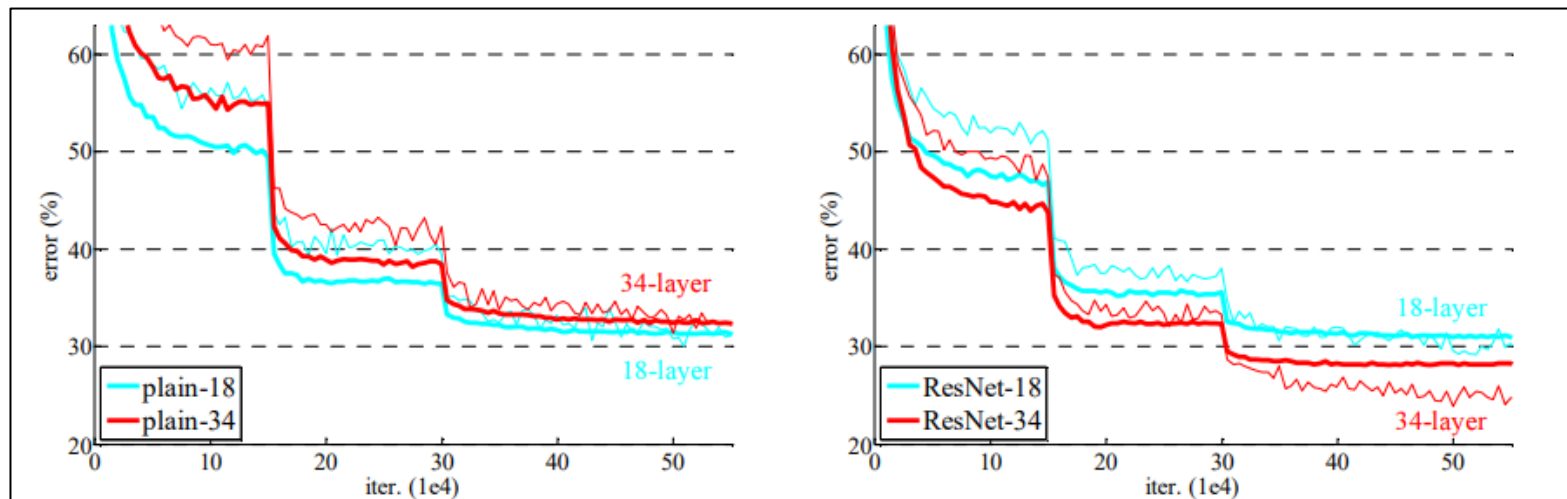
역전파

$$F'(x) + 1$$

3 CNN 모델의 발전

- ResNet

ResNet의 구조: Residual Learning



Residual Learning을 사용한 경우 층을 더 쌓았을 때 성능이 증가

3 CNN 모델의 발전

- 모델 비교

AlexNet, VGGNet, ResNet-152 비교

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B



ResNet { 성능: 매우 높음
파라미터 수: AlexNet과 유사 & VGGNet의 절반

4

Deep Learning in CV

4 Deep learning in CV

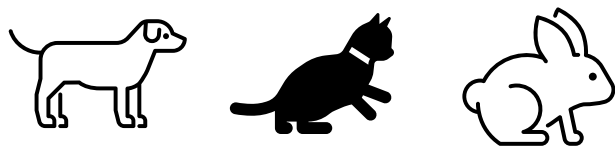
- 객체 탐지란?

Object Detection (객체 탐지)

이미지가 주어졌을 때, 어떤 물체가 어디에 있는지 찾는 작업

Classification

물체가 어떤 물체인지 분류하는 것



+

Localization

물체가 어디에 있는지 찾는 것



이 두 가지 과정을 모두 수행하는 작업

4 Deep learning in CV

- 객체 탐지란?

Object Detection (객체 탐지)

이미지가 주어졌을

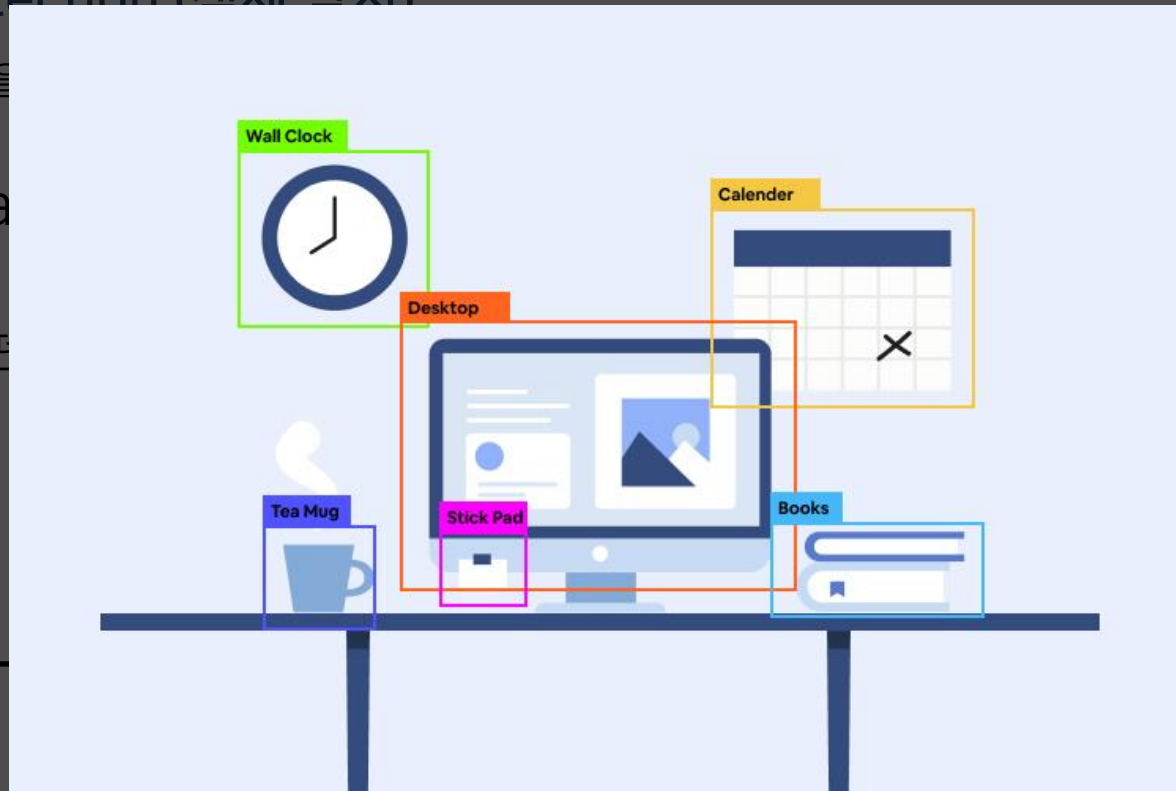
Class

물체가 어떤



ion

이 찾는 것



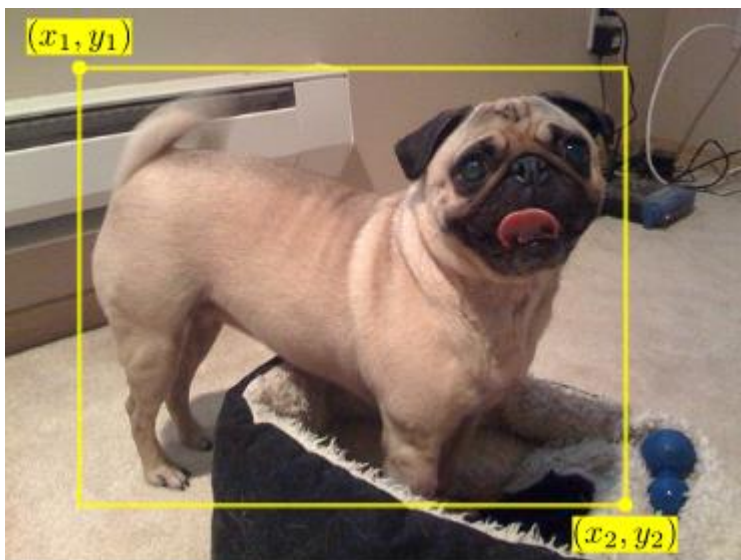
이미지에 여러 물체가 있어도 분류가 가능함

4 Deep learning in CV

- 위치 표현 방식

Bounding box (bbox)

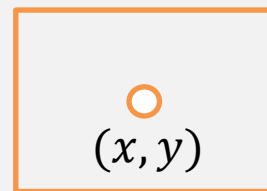
물체의 위치를 표시해주는 사각형



좌측 상단의 좌표, 우측 하단의 좌표 이용



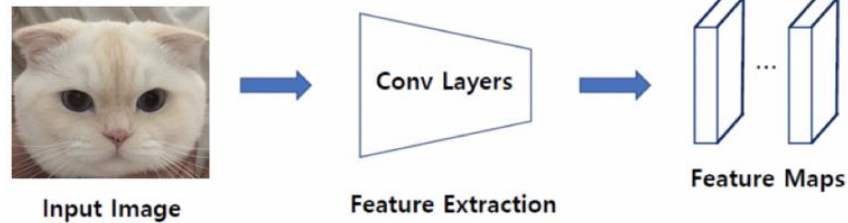
사각형 중심 또는 한 꼭짓점의 좌표와 너비, 높이 이용



4 Deep learning in CV

- 객체 탐지 모델

1-stage detector



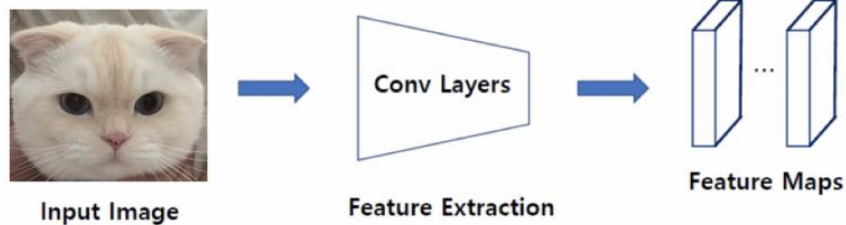
2-stage detector



4 Deep learning in CV

- 객체 탐지 모델

1-stage detector



2-stage detector

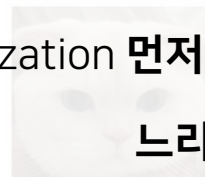


4 Deep learning in CV

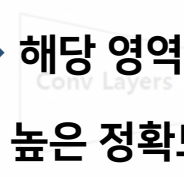
● 객체 탐지 모델

1-stage detector

Localization 먼저 고려 → 해당 영역에 대해 Classification 진행
느리지만 더 높은 정확도를 보여준다



Input Image



Feature Extraction



Feature Maps

2-stage detector



Input Image



- Selective Search
- Region Proposal Network

4 Deep learning in CV

- 객체탐지 평가지표(mAP)

PR curve

임계값 기준으로 positive/negative

		Predicted		
		Positive	Negative	
Observed	Positive	TP	FN	P
	Negative	FP	TN	N

Precision

정답이라고 예측한 것 중 실제 정답인 비율

$$\frac{TP}{TP + FP}$$

Recall

실제 정답인 것 중 정답이라고 예측한 비율

$$\frac{TP}{TP + FN}$$

4 Deep learning in CV

- 객체탐지 평가지표(mAP)

PR curve

임계값 기준으로 positive/negative

		Predicted		
		Positive	Negative	
Observed	Positive	TP	FN	P
	Negative	FP	TN	N

자세한 내용은 범주형자료분석팀 3주차 클린업에서!

Precision

정답이라고 예측한 것 중 실제 정답인 비율

$$\frac{TP}{TP + FP}$$

Recall

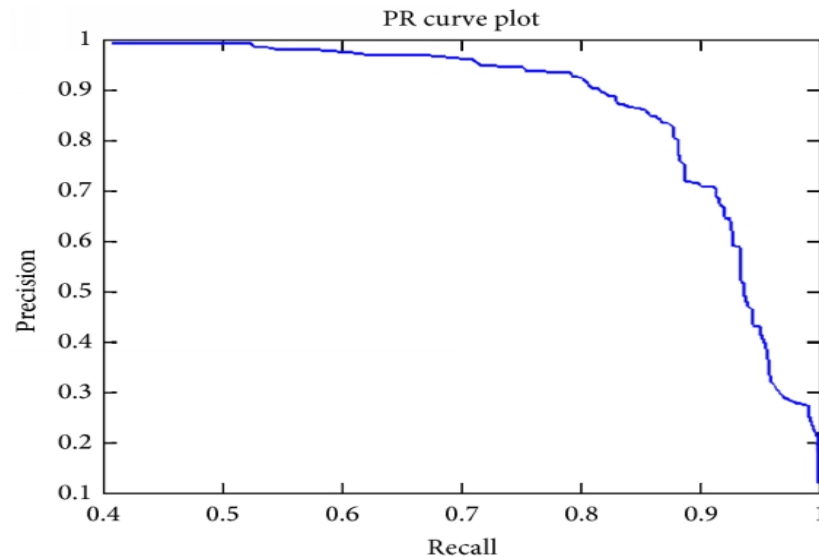
실제 정답인 것 중 정답이라고 예측한 비율

$$\frac{TP}{TP + FN}$$

4 Deep learning in CV

- 객체탐지 평가지표(mAP)

PR curve



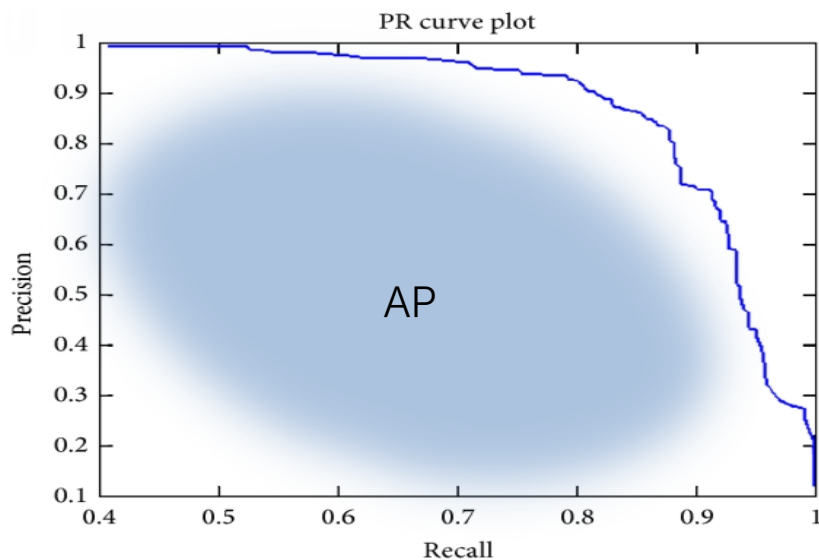
Precision과 Recall 값은 trade-off 관계

임계값을 1부터 0까지 조절해 그린 그래프 → PR Curve

4 Deep learning in CV

- 객체탐지 평가지표(mAP)

mAP (mean Average Precision)



AP : PR curve 아래의 넓이를 구한 값




mAP: 각 label마다 AP를 구한 후 평균을 낸 값

4 Deep learning in CV

- 객체탐지 평가지표(mAP)

IoU (Intersection over Union)


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

✓ IoU > threshold → TP

IoU < threshold → FP

이용해 AP와 mAP값 도출


Object detection에서 mAP를 구할 때는 IoU의 개념 적용

IoU = 실제 bbox와 예측한 bbox의 합집합을 분모에, 교집합을 분자에 둔 값

4 Deep learning in CV

- 객체탐지 평가지표(mAP)

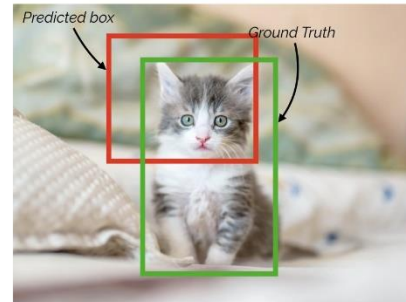
IoU (Intersection over Union)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$




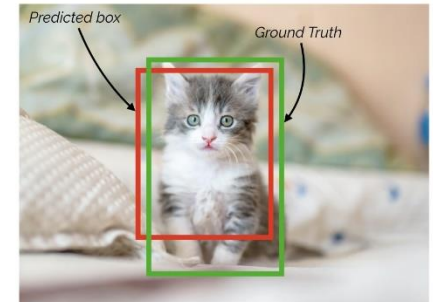
If IoU threshold = 0.5

False Positive (FP)



$\text{IoU} \sim 0.3$

True Positive (TP)

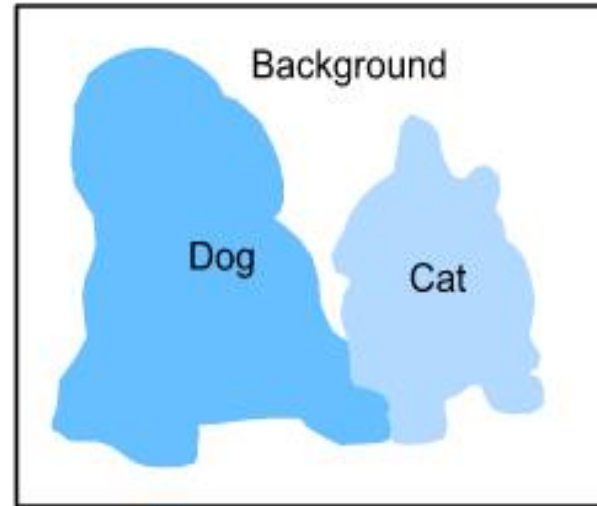


$\text{IoU} \sim 0.7$

IoU = 실제 bbox와 예측한 bbox의 합집합을 분모에, 교집합을 분자에 둔 값
0부터 1사이의 값이 나오며 클수록 정확하게 예측한 것에 해당

4 Deep learning in CV

- Image Segmentation



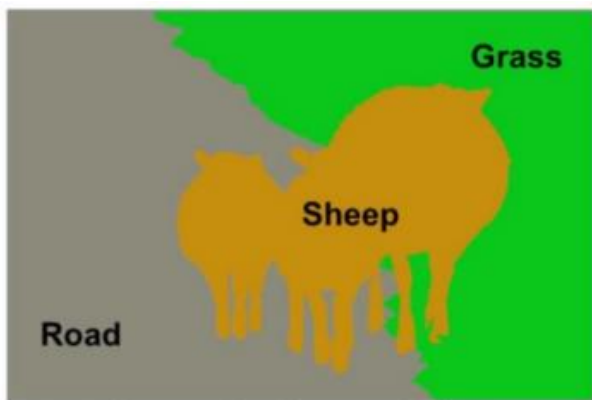
이미지를 구성하고 있는 모든 픽셀을 대상으로 class에 따라 분류하는 작업

4 Deep learning in CV

- Image Segmentation

Semantic Segmentation

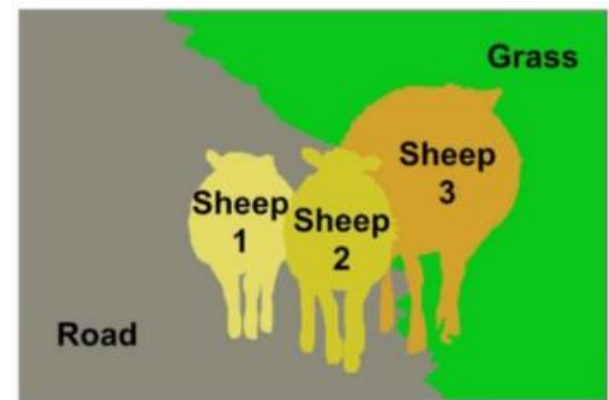
물체가 어디에 속하는지에 대해서만 분류



Semantic Segmentation

Instance Segmentation

같은 class 내에서도 세부적으로 분류



Instance Segmentation

4 Deep learning in CV

- Image Segmentation



Input



- 1: Person
- 2: Purse
- 3: Plants/Grass
- 4: Sidewalk
- 5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

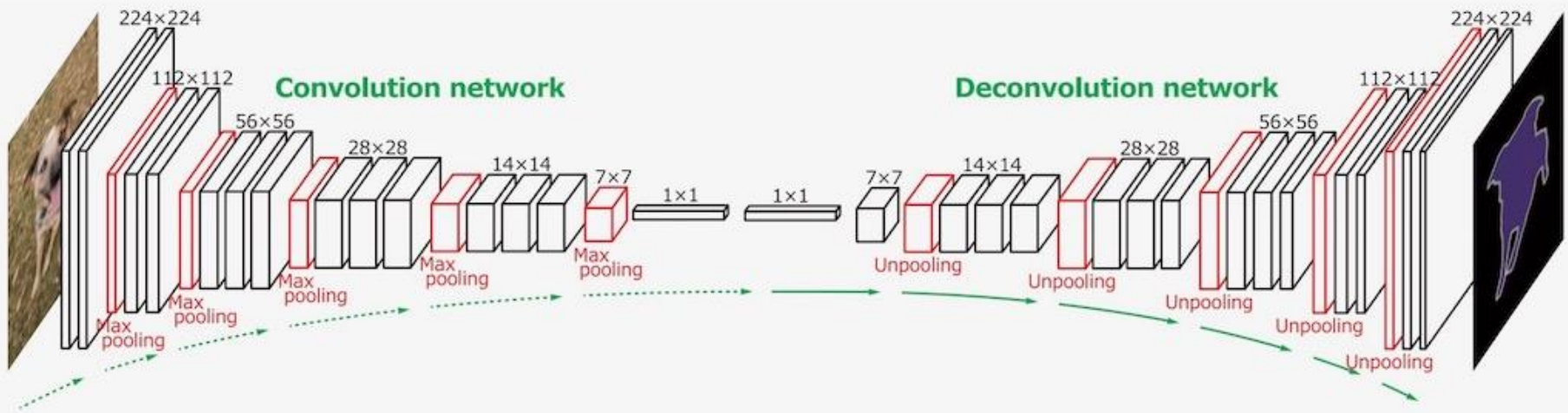
Semantic Labels

위와 같이 각 픽셀이 속하는 class 출력

입력 이미지의 크기 = 출력 이미지의 크기 (출력 = mask)

4 Deep learning in CV

- Image Segmentation



CNN의 경우 각 층을 거치며 이미지의 크기 감소

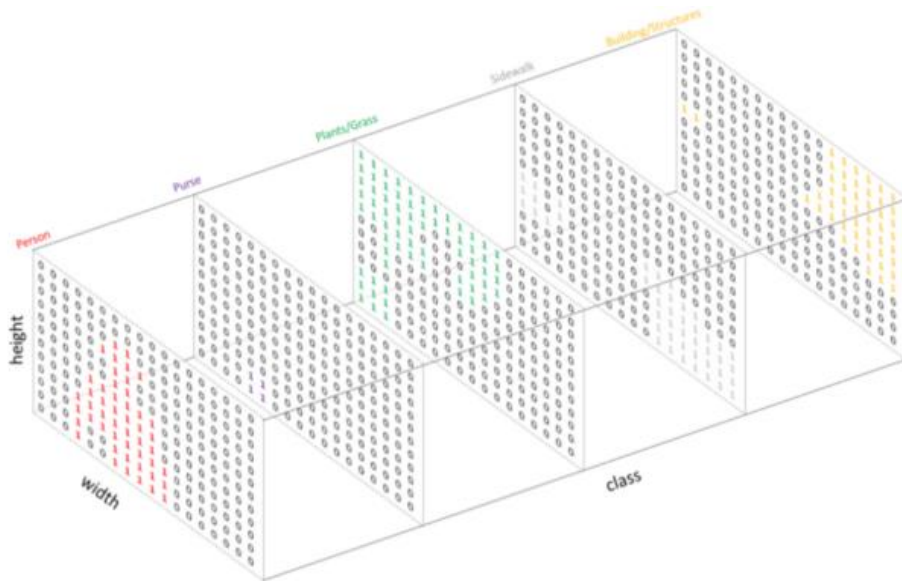


이미지의 크기를 줄이는 Convolution과 pooling을 역으로 연산

4 Deep learning in CV

- Image Segmentation

Mask(출력)를 만드는 방식



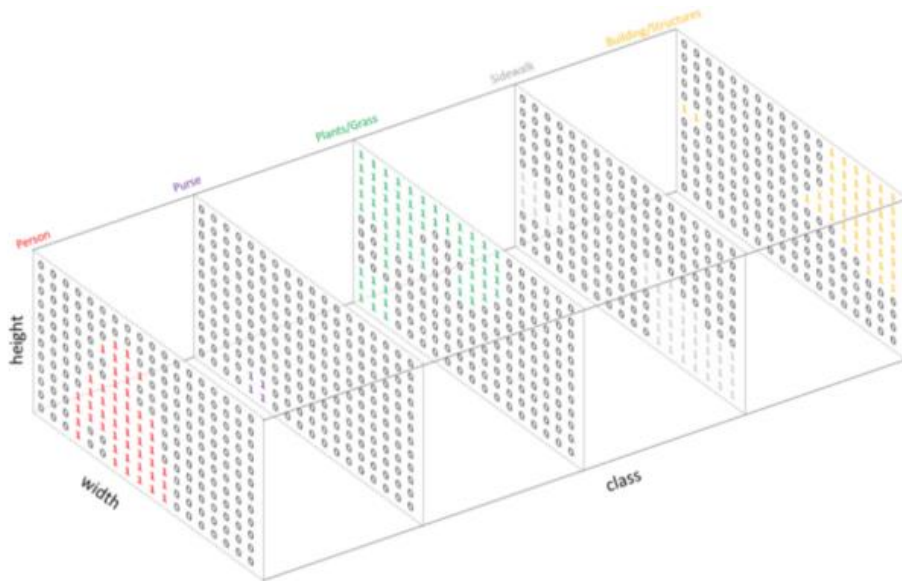
1. 출력의 채널 하나 하나는 각 class를 의미함

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

4 Deep learning in CV

- Image Segmentation

Mask(출력)를 만드는 방식

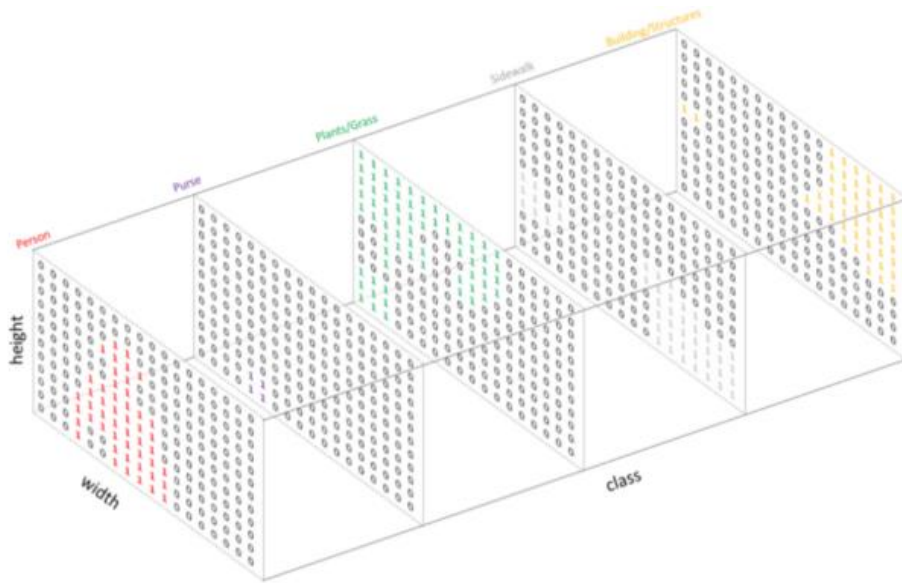


1. 출력의 채널 하나 하나는 각 class를 의미함
2. 각 채널의 픽셀 값은 활성화 함수를 거쳐 나온 값
(픽셀이 해당 class에 속할 score라고 해석가능)

4 Deep learning in CV

- Image Segmentation

Mask(출력)를 만드는 방식



1. 출력의 채널 하나 하나는 각 class를 의미함
2. 각 채널의 픽셀 값은 활성화 함수를 거쳐 나온 값 (픽셀이 해당 class에 속할 score라고 해석가능)
3. 채널들에 대해 argmax 적용해 큰 값의 채널 찾기

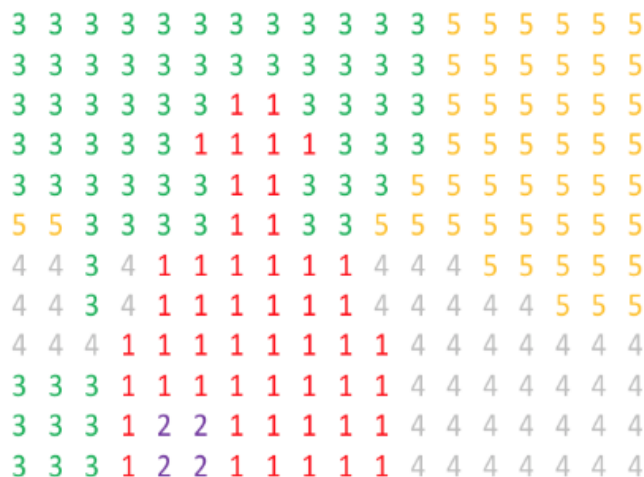
Argmax란?

입력에서 최댓값이 있는
위치(인덱스)를 알려주는 함수

4 Deep learning in CV

● Image Segmentation

Mask(출력)를 만드는 방식



Semantic Labels

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

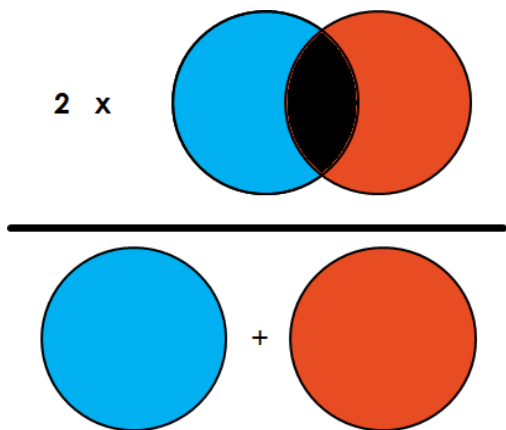
1. 출력의 채널 하나 하나는 각 class를 의미함
2. 각 채널의 픽셀 값은 활성화 함수를 거쳐 나온 값 (픽셀이 해당 class에 속할 score라고 해석가능)
3. 채널들에 대해 argmax 적용해 큰 값의 채널 찾기
4. 최종적으로 (W,H,1) 크기로 출력됨

4 Deep learning in CV

- Image Segmentation 평가지표

Dice coefficient

IoU, mPA 이외의 Image segmentation 평가지표 중 하나



$$\frac{2*|X \cap Y|}{|X| + |Y|} = \frac{2*TP}{(TP+FP) + (TP+FN)}$$



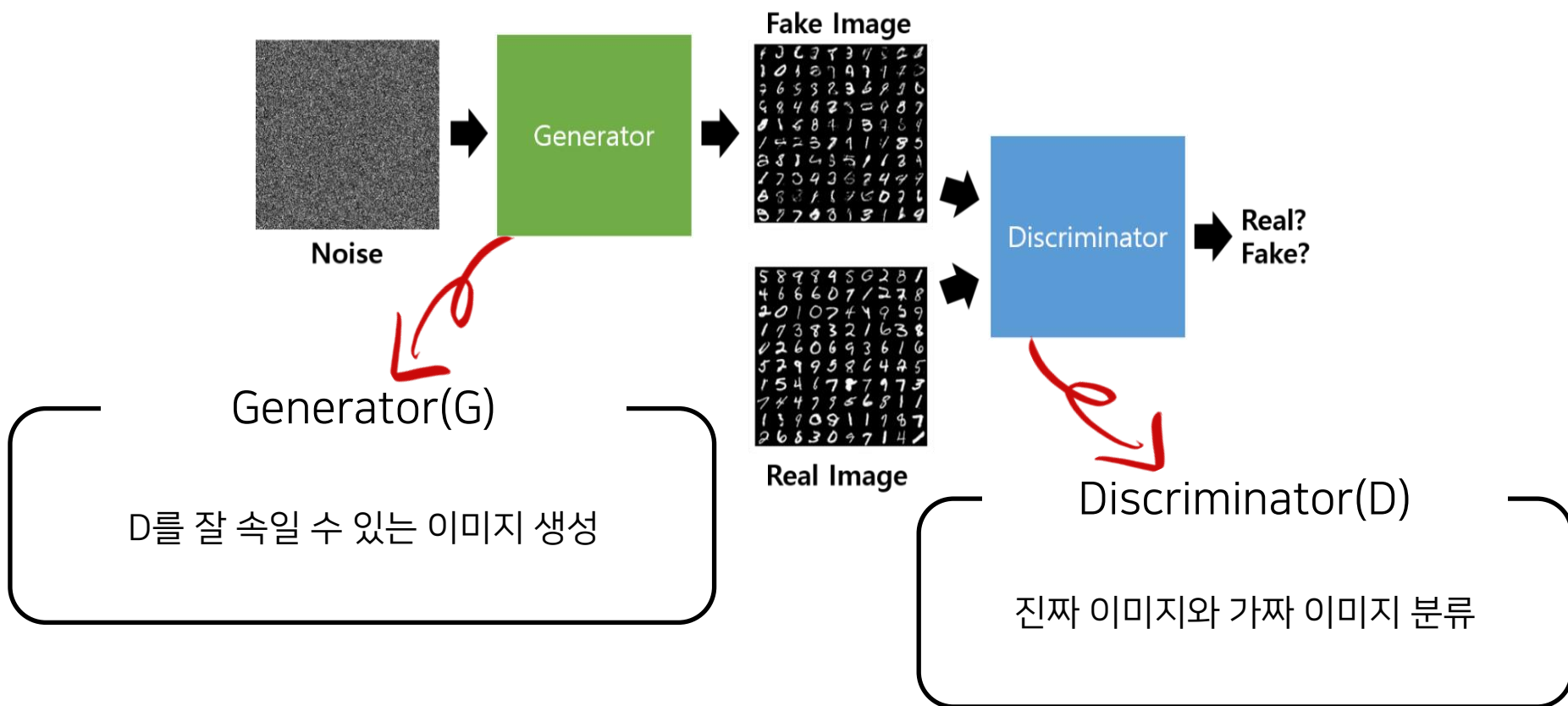
F1 score의 식과 동일해
불균형한 데이터를 다룰 때 유용함

4 Deep learning in CV

- 기타 Computer Vision

Generative Adversarial Network (GAN)

목적에 맞게 G와 D를 학습시켜 진짜 이미지와 비슷한 가짜 이미지를 생성하는 모델



4 Deep learning in CV

- 기타 Computer Vision

Generative Adversarial Network (GAN)

목적에 맞게 G와 D를 학습시켜 진짜 이미지와 비슷한 가짜 이미지를 생성하는 모델



컴퓨팅 기술의 발전에 따라 가짜 이미지의 퀄리티도 발전하는 중

4 Deep learning in CV

- 기타 Computer Vision

Style Transfer

한 이미지의 내용(content)과 다른 이미지의 스타일(style)을 합치는 알고리즘



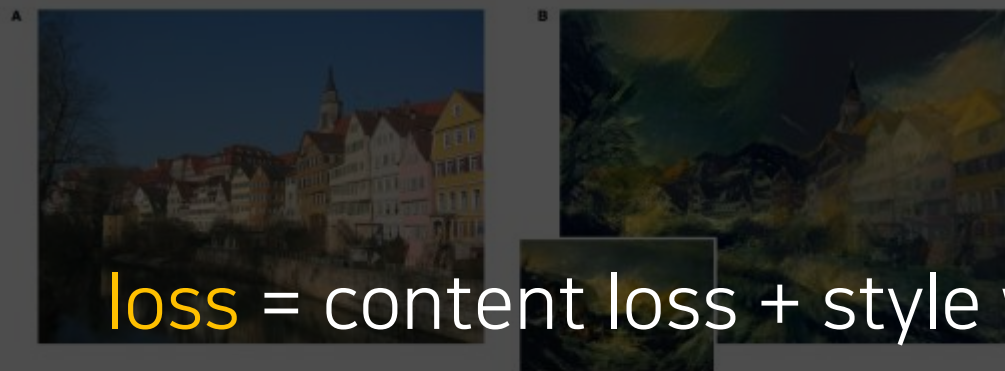
Target image와 content image의 차이
Target image와 style image의 차이를
줄이는 것이 목표

4 Deep learning in CV

- 기타 Computer Vision

Style Transfer

한 이미지의 내용(content)과 다른 이미지의 스타일(style)을 합치는 알고리즘



Target image와 content image의 차이
target image와 style image의 차이를
줄이는 것이 목표

4 Deep learning in CV

- 기타 Computer Vision

stylegan2

Gan과 style transfer의 개념이 더해진 GAN의 일종



이처럼 컴퓨터에서 이미지를 다양하게 다룰 수 있음

다음주 예고

1. 자연어 데이터의 특징
2. 자연어 임베딩
3. 여러가지 딥러닝 모델



THANK YOU

