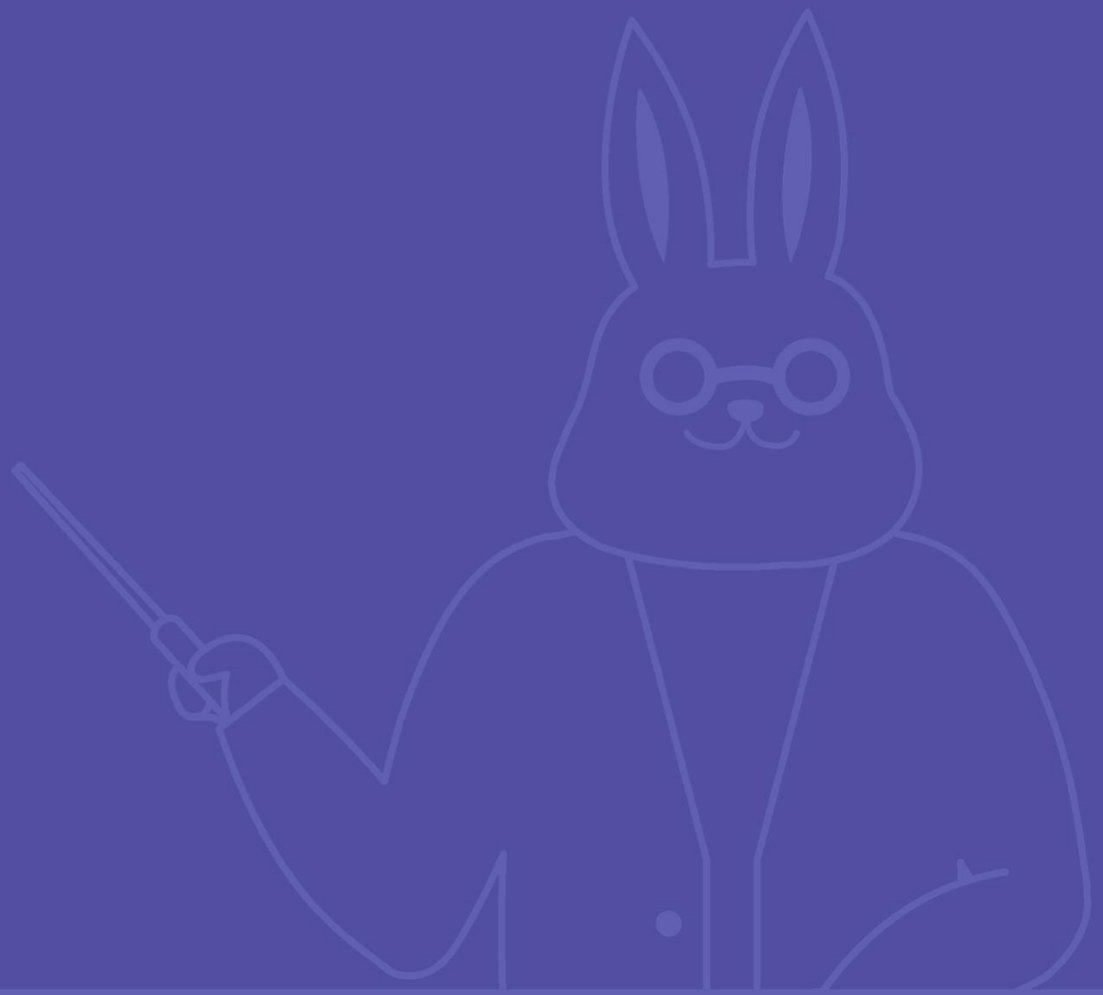
CNN/RNN

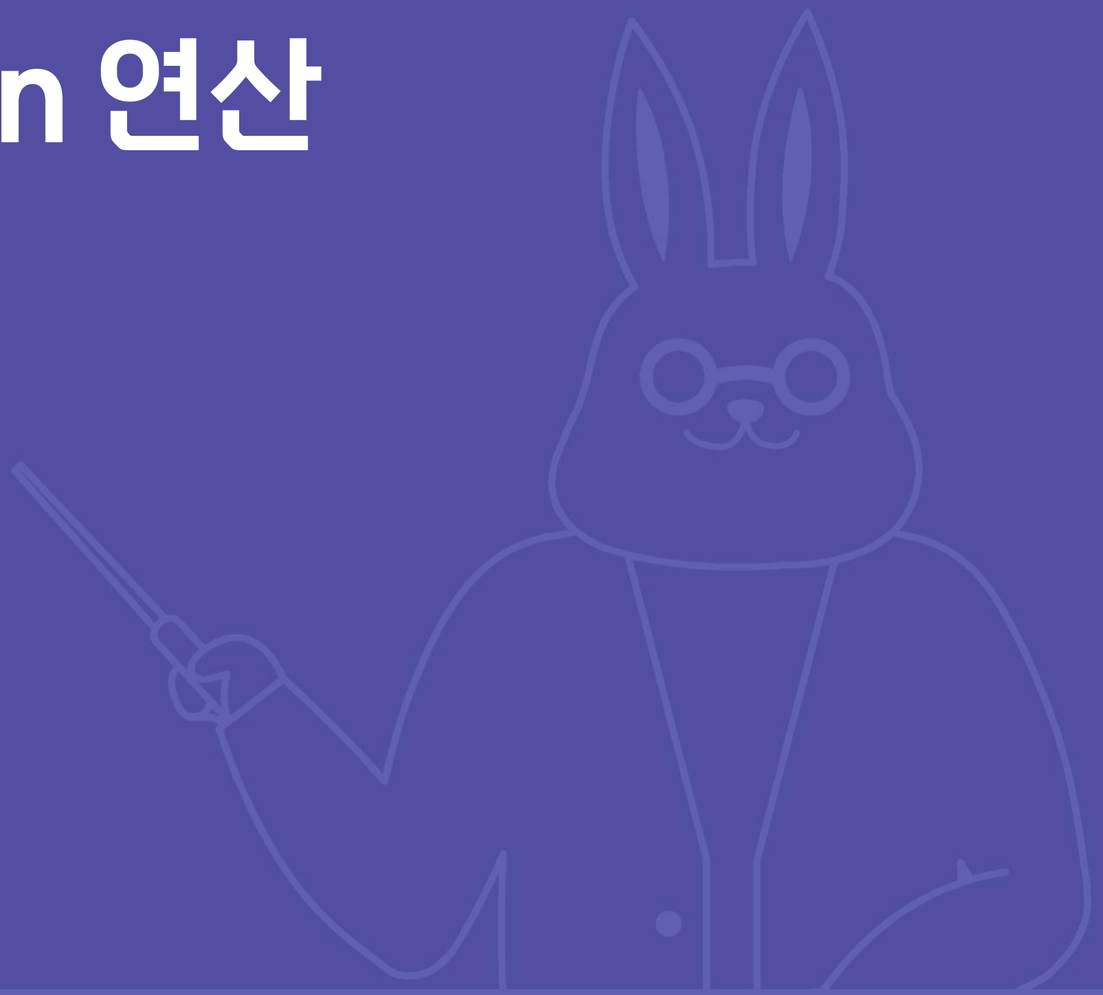
02 Convolutional Neural Network



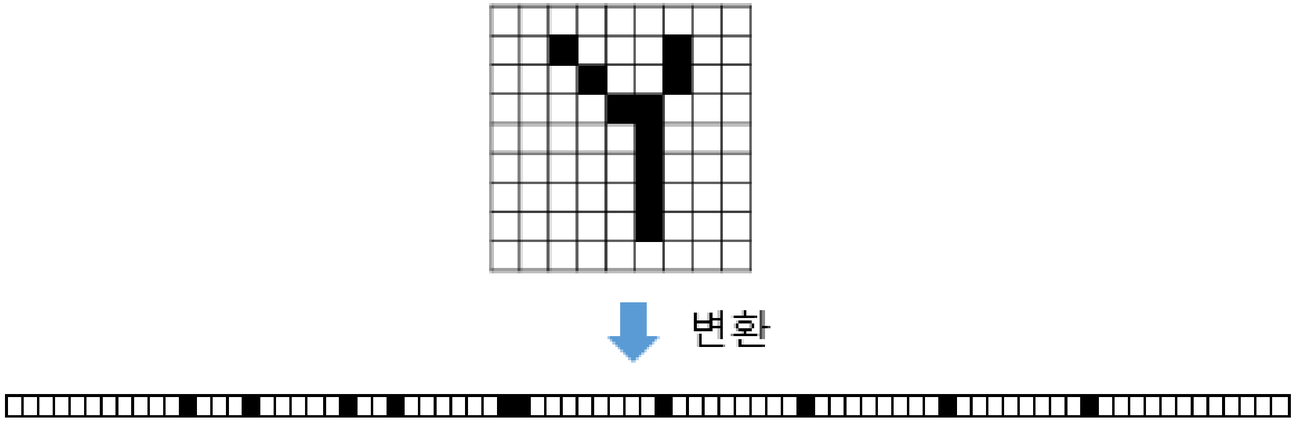
목차

01

이미지와 Convolution 연산



Fully-connected Layer와 이미지 데이터



- FC Layer는 **1차원 데이터**를 요구
- 이미지를 단순하게 1차원으로 바꾸면 **2차원 상에서 가지는 정보**를 포기해야 함
  - 이미지 내 사물 간의 거리 관계 등
  - 색의 변화 → 특히 세로로 변하는 상황
- 즉, **공간 정보(Spatial Information)**가 무너짐

Convolutional Neural Network

- 따라서 이미지 처리에 특화된 딥러닝 모델이 등장

- CNN의 대표적인 구성 요소
  - **Convolutional Layer**
  - **Pooling Layer**
  - **분류기(Classifier)**: Fully-connected layer로 구성

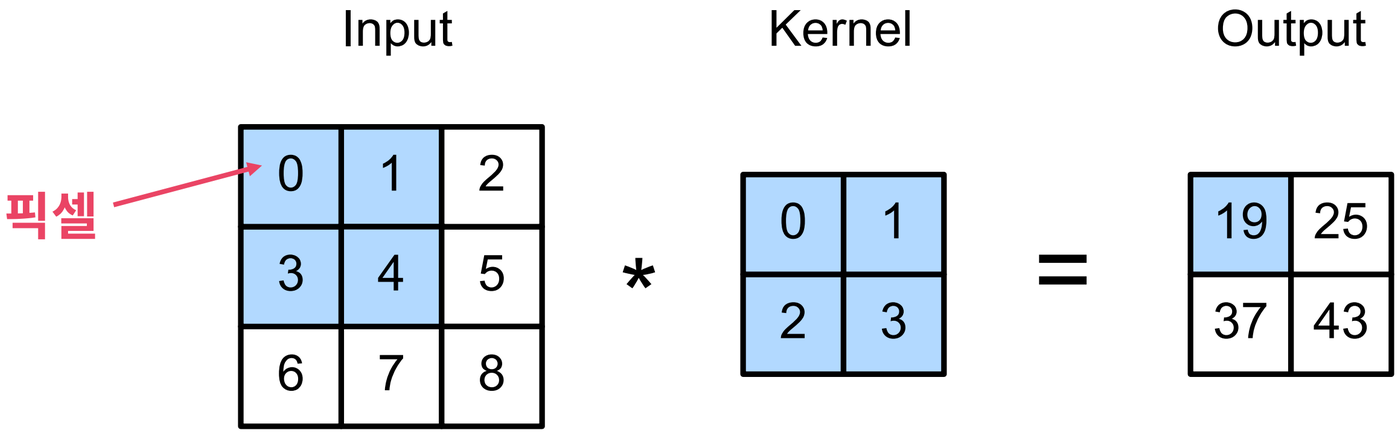
Convolution 연산

원본

커널	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
결과	 이미지 샤프닝	 경계선 감지	 블러 처리

- CNN을 구현하는 핵심 연산
- **커널**과 **Convolution 연산**
  - 전통적인 이미지 처리 분야에서 **커널(또는 필터)**이란 것이 존재
  - 이미지와 커널 간의 Convolution 연산으로 처리

Convolution 연산



- **2차원 이미지 데이터**: **행렬**로 표현 가능
  - 행렬의 각 원소는 해당 위치의 이미지 픽셀값
- **Convolution 커널**: **행렬**로 표현 가능
- Convolution 연산은 **2차원 상에서 연산이 이루어지므로** 이미지 데이터를 **변형 없이** 그대로 사용 가능

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

$*$

5	9	1
3	8	3
3	4	6

커널

$=$

181		

결과

$$7 \times 5 + 2 \times 9 + 8 \times 1 + 3 \times 3 + 6 \times 8 + 6 \times 3 + 1 \times 3 + 6 \times 4 + 3 \times 6 = 181$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

$*$

5	9	1
3	8	3
3	4	6

커널

$=$

181	204	

결과

$$2 \times 5 + 8 \times 9 + 8 \times 1 + 6 \times 3 + 6 \times 8 + 6 \times 3 + 6 \times 3 + 3 \times 4 + 0 \times 6 = 204$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

$*$

5	9	1
3	8	3
3	4	6

커널

$=$

181	204	227

결과

$$8 \times 5 + 8 \times 9 + 4 \times 1 + 6 \times 3 + 6 \times 8 + 2 \times 3 + 3 \times 3 + 0 \times 4 + 5 \times 6 = 227$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

$*$

5	9	1
3	8	3
3	4	6

커널

$=$

181	204	227
195		

결과

$$3 \times 5 + 6 \times 9 + 6 \times 1 + 1 \times 3 + 6 \times 8 + 3 \times 3 + 0 \times 3 + 3 \times 4 + 8 \times 6 = 195$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

*

5	9	1
3	8	3
3	4	6

커널

=

181	204	227
195	203	

결과

$$6 \times 5 + 6 \times 9 + 6 \times 1 + 6 \times 3 + 3 \times 8 + 0 \times 3 + 3 \times 3 + 8 \times 4 + 5 \times 6 = 203$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

$*$

5	9	1
3	8	3
3	4	6

커널

$=$

181	204	227
195	203	166

결과

$$6 \times 5 + 6 \times 9 + 2 \times 1 + 3 \times 3 + 0 \times 8 + 5 \times 3 + 8 \times 3 + 5 \times 4 + 2 \times 6 = 166$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

*

5	9	1
3	8	3
3	4	6

커널

=

181	204	227
195	203	166
179		

결과

$$1 \times 5 + 6 \times 9 + 3 \times 1 + 0 \times 3 + 3 \times 8 + 8 \times 3 + 3 \times 3 + 9 \times 4 + 4 \times 6 = 179$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

*

5	9	1
3	8	3
3	4	6

커널

=

181	204	227
195	203	166
179	206	

결과

$$6 \times 5 + 3 \times 9 + 0 \times 1 + 3 \times 3 + 8 \times 8 + 5 \times 3 + 9 \times 3 + 4 \times 4 + 3 \times 6 = 206$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 과정

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

이미지

$*$

5	9	1
3	8	3
3	4	6

커널

$=$

181	204	227
195	203	166
179	206	150

결과

$$3 \times 5 + 0 \times 9 + 5 \times 1 + 8 \times 3 + 5 \times 8 + 2 \times 3 + 4 \times 3 + 3 \times 4 + 6 \times 6 = 150$$

연산 과정: 커널이 이미지의 **노란색** 영역에 겹쳐짐

Convolution 연산 용어

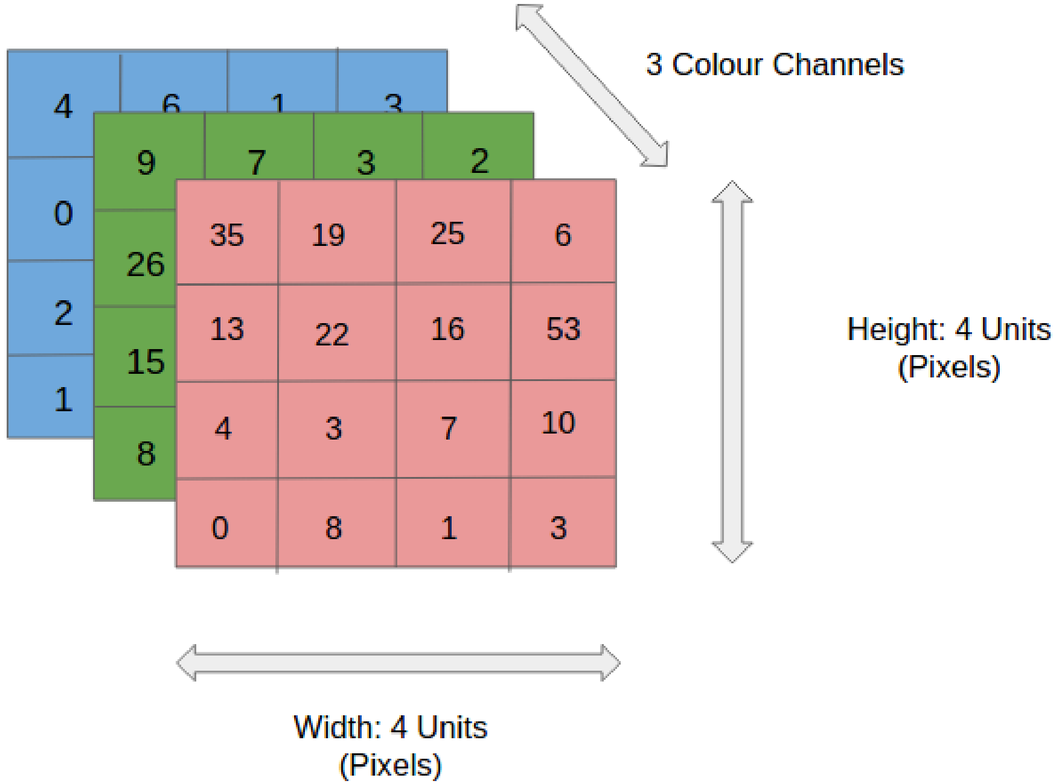
181	204	227
195	203	166
179	206	150

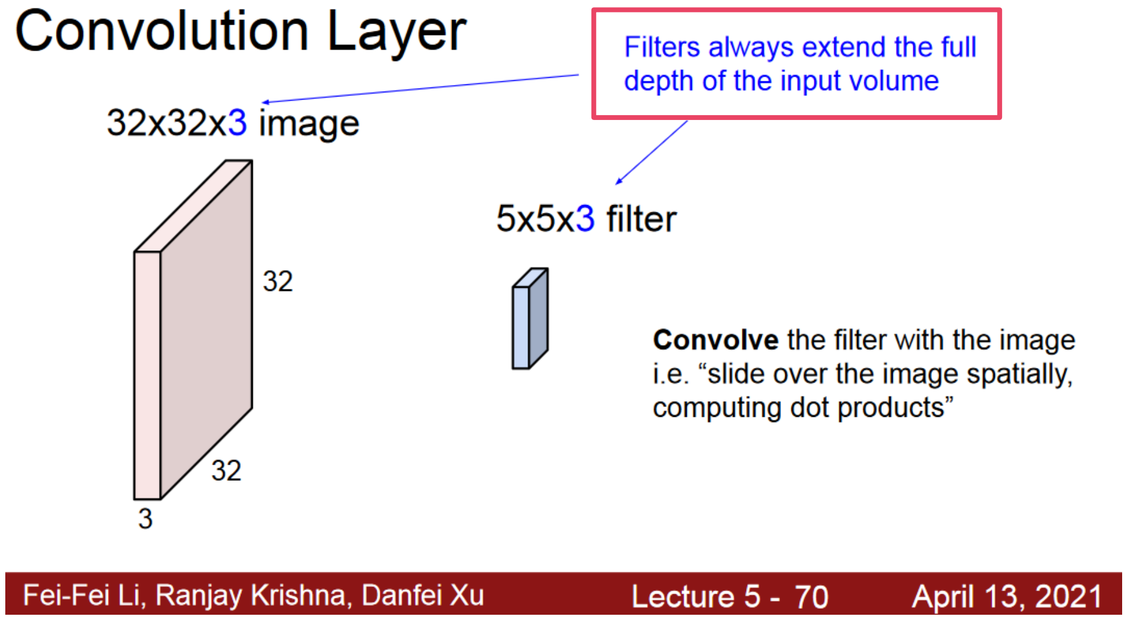
Feature Map (Activation Map)

7	2	8	8	4
3	6	6	6	2
1	6	3	0	5
0	3	8	5	2
3	9	4	3	6

- **연산 결과**: **Feature Map** 또는 **Activation Map**이라 부름
- 커널과 이미지가 **겹치는 영역: 수용 영역(Receptive Field)**

컬러 이미지의 Convolution 연산

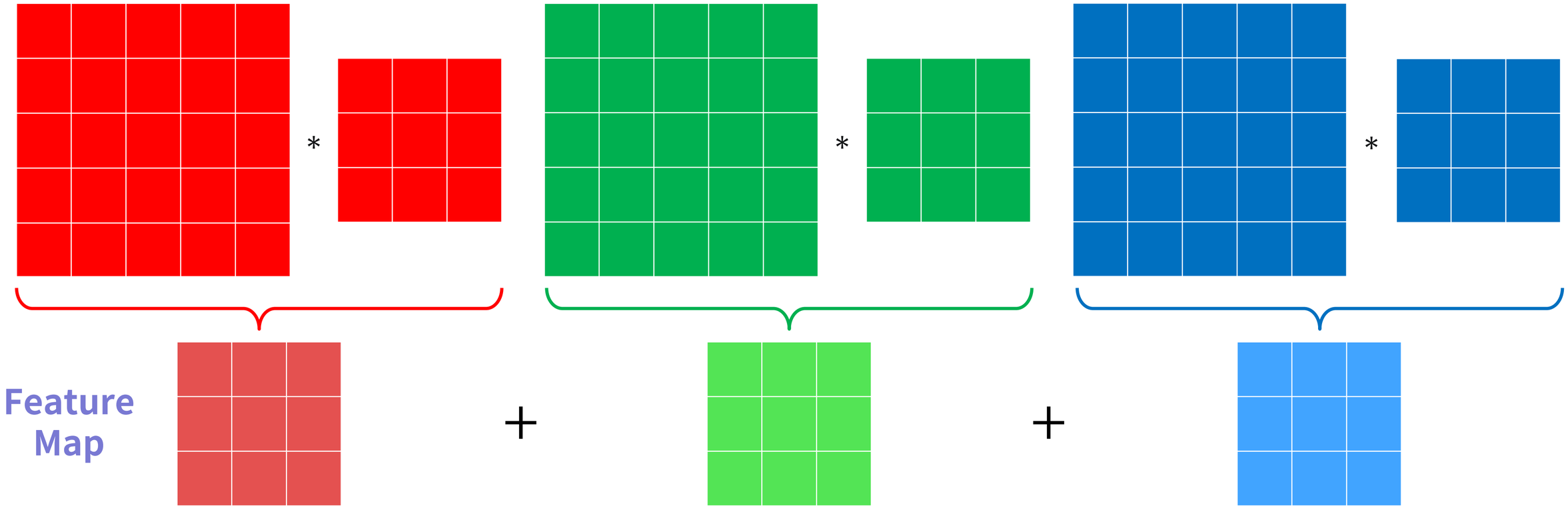




Stanford CS231n 2021 Spring

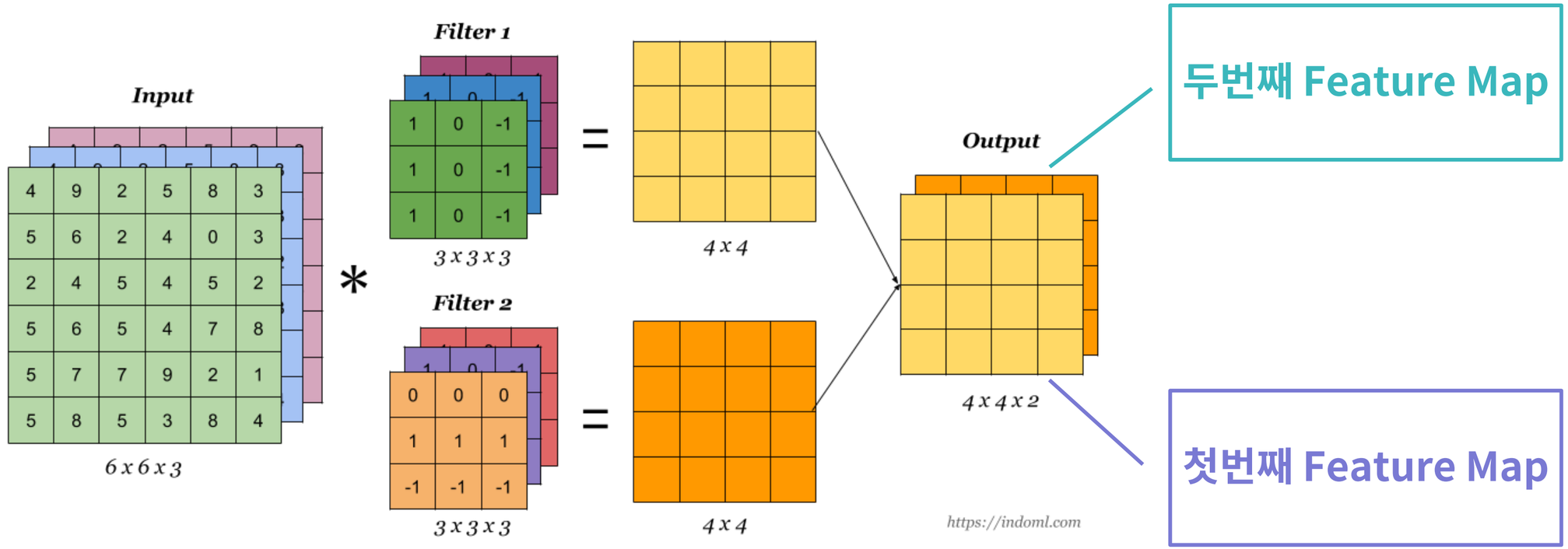
- 앞선 예시는 이미지의 채널이 1개 → 흑백 이미지
- 컬러 이미지는 채널이 3개
- 이 경우 **커널도 채널을 3개로 준비**

컬러 이미지의 Convolution 연산



각 채널 별로 Convolution 연산을 수행하고 각 결과를 더해서 **하나의 Feature Map**을 생성

Convolution 연산 확장

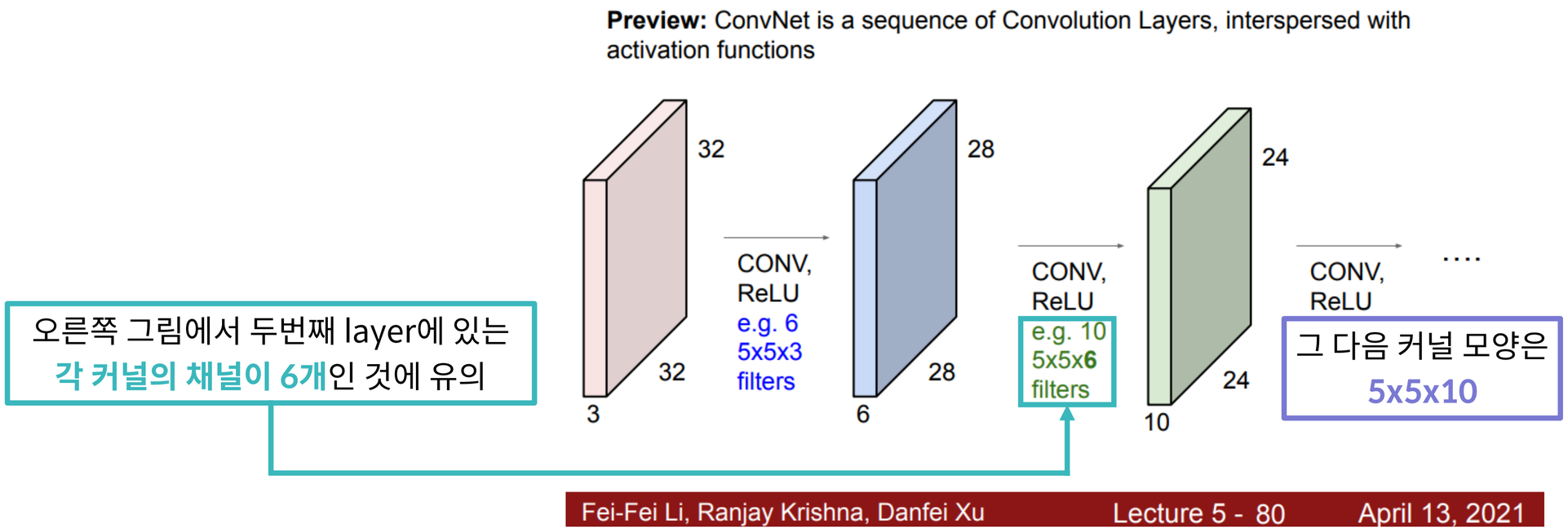


- 지금까지는 커널을 한 개만 사용 → **Feature Map도 한 개**
- **커널을 여러 개 두면 Feature Map도 여러 개 생성**

02

Convolutional Neural Network

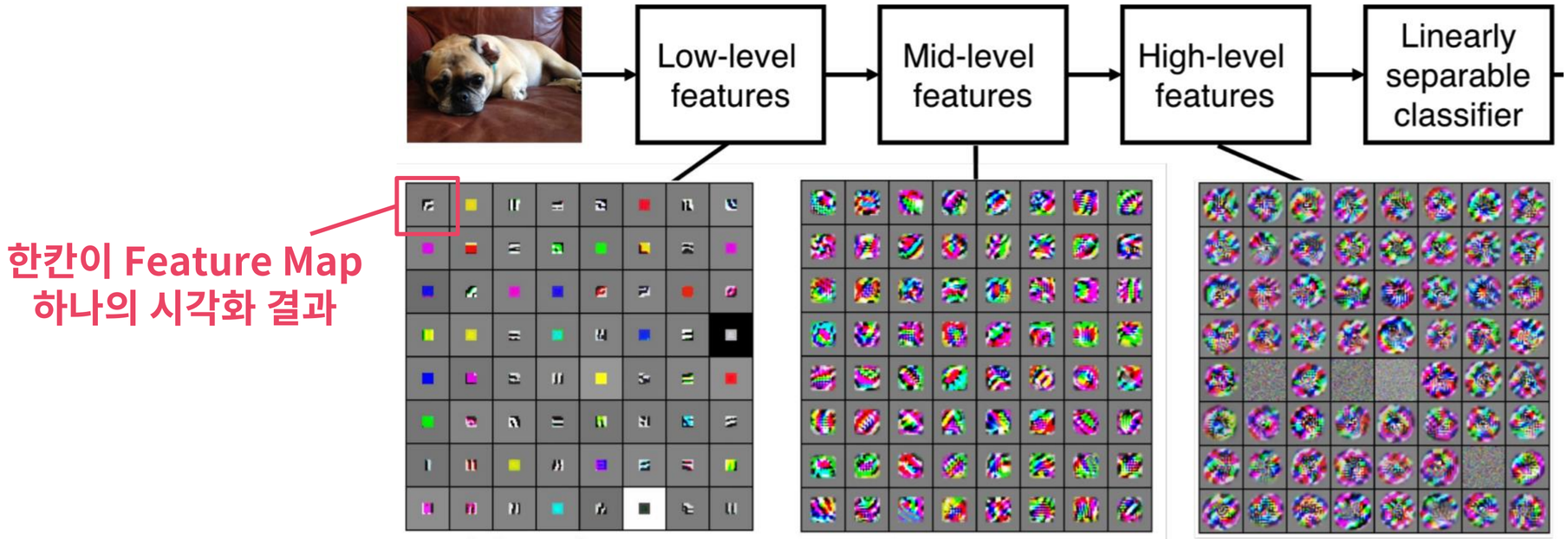
Convolutional Layer



Stanford CS231n 2021 Spring

- 지금까지 사용한 커널들은 **학습 가능한 커널**
  - 즉 커널 행렬의 각 값들이 **가중치(Weight)**
- 이러한 커널들로 이루어진 Layer를 **Convolutional Layer**라고 부름
  - 이 Layer들을 쌓아서 CNN을 구성

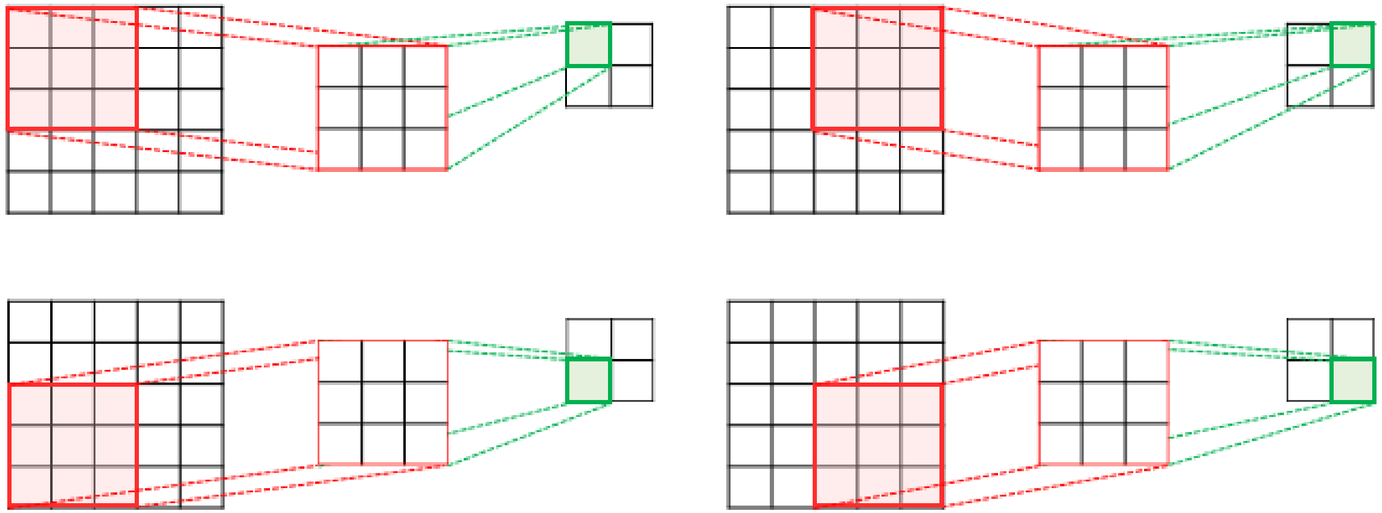
Layer 역할



Stanford CS231n 2021 Spring

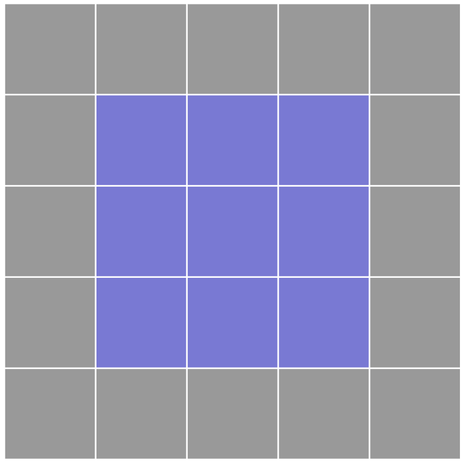
- 이미지가 가지는 **특정 Feature를 뽑아내도록 커널을 학습**
- 커널에 따라 추출하는 Feature를 다르게 학습
- 이미지 내의 **대각선, 원형, 색조** 등등이 이러한 Feature에 해당

Stride

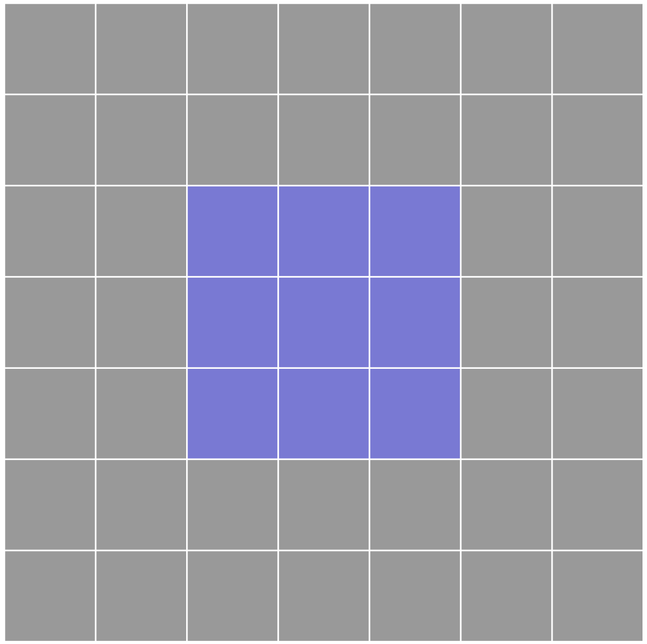


- Convolution 연산 과정을 조절하기 위한 Hyperparameter
- 커널이 **이미지 내에서 이동하는 칸수**를 조절
- 지금까지 Convolution 연산에서 보여준 예시는 모두 1칸
- 위의 그림은 **Stride가 2칸**일 경우의 예시

Padding



Padding = 1



Padding = 2

- 지금까지 예시는 Convolution 연산 결과 **Feature Map 사이즈가 계속 줄어듦**
- **Padding**을 추가하여 **Feature Map 사이즈가 줄어드는 현상 방지**
- 또한 이미지의 테두리 정보도 균일하게 활용

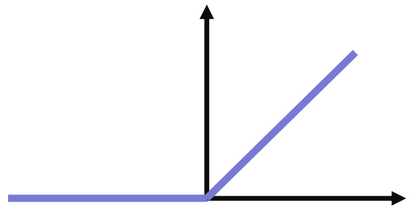
Convolutional Layer 의의

- 왜 이미지 특징을 잘 뽑아내는가?
  - Convolution 연산은 **하나의 커널이 픽셀 간의 정보**를 보게 만듦
  - 하나의 커널이 이미지 전체 영역을 보고 학습

- **Parameter Sharing**
  - 커널이 가진 **Parameter를 이미지의 모든 영역에서 공유**
  - Parameter 개수를 FC Layer에 비해 극적으로 줄임 → **과적합 방지에 유리**

Convolutional Layer 활성화 함수

2	0	-1	9	1
-2	2	2	6	-3
-2	6	4	5	9
7	5	6	-3	8
1	7	5	6	7



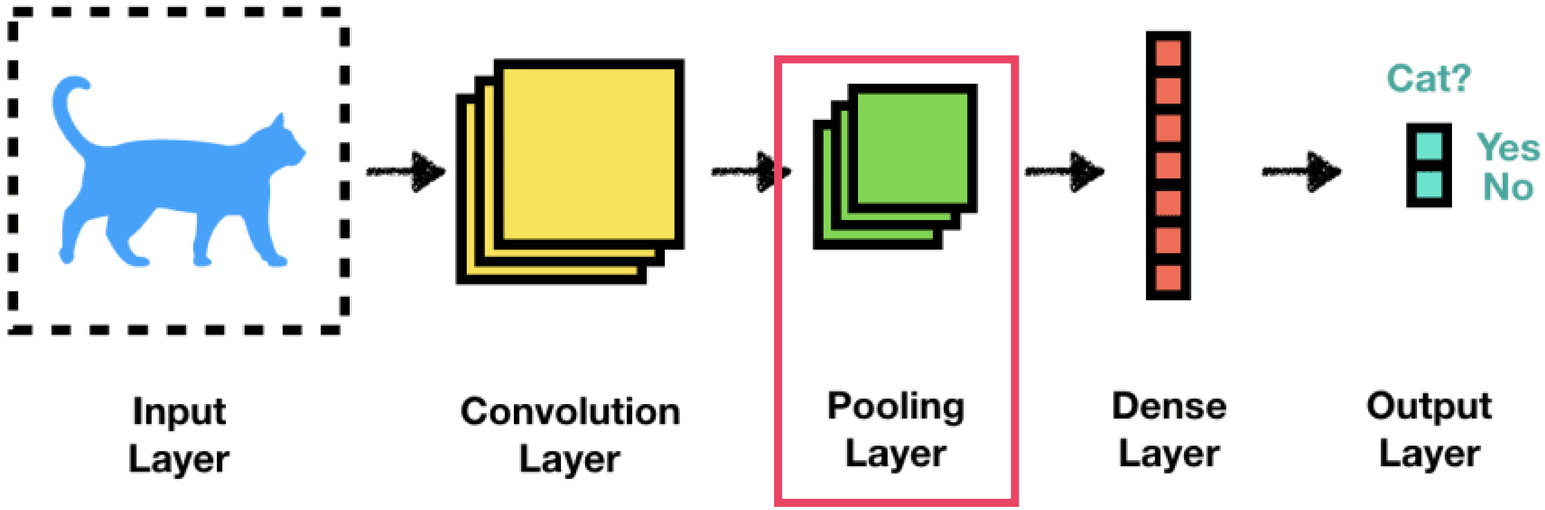
ReLU

2	0	0	9	1
0	2	2	6	0
0	6	4	5	9
7	5	6	0	8
1	7	5	6	7

- Convolution 연산 또한 **선형 연산**
  - 모두 **곱셈**과 **덧셈**으로만 이루어짐

- 따라서 FC Layer처럼 **비선형성을 추가**하기 위해 활성화 함수를 사용
  - CNN은 주로 **ReLU 함수** 사용

Pooling Layer



- CNN에서 거의 항상 같이 쓰이는 Layer
- **주 역할**: Feature Map의 사이즈를 **줄여서 Parameter 개수를 줄이는 것** → 과적합 조절

Pooling Layer – Max Pooling

12	29	2	5
21	12	17	25
18	15	2	19
17	23	16	5

Max Pooling

29	25
23	19

- 주어진 이미지나 Feature Map을 **겹치지 않는 영역**으로 분할
- 위 그림은 각 영역의 크기가 2x2가 되도록 분할
- 각 영역에서 **최대값**을 뽑아내어 새로운 Feature Map을 구성

Pooling Layer – Average Pooling

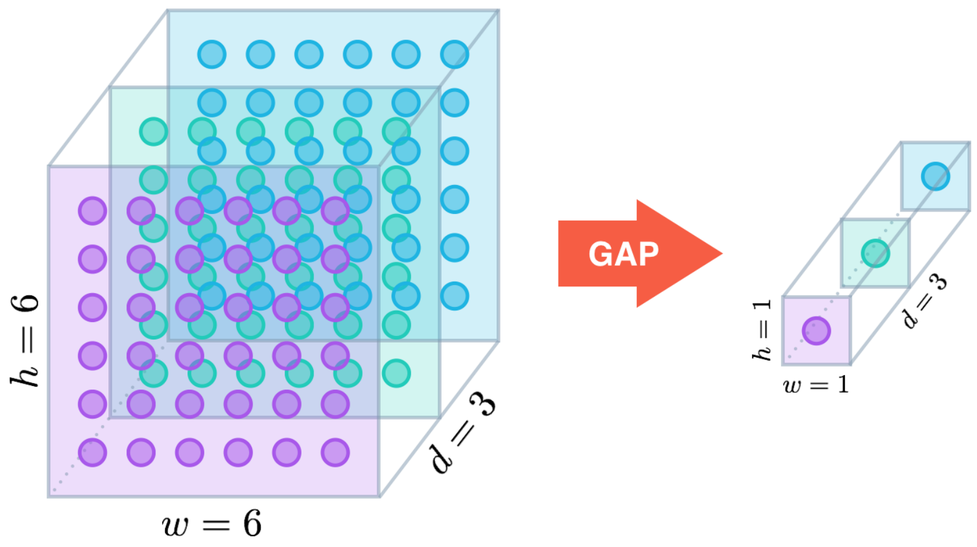
12	29	2	5
21	12	17	25
18	15	2	19
17	23	16	5

Average Pooling

18.5	12.25
18.25	10.5

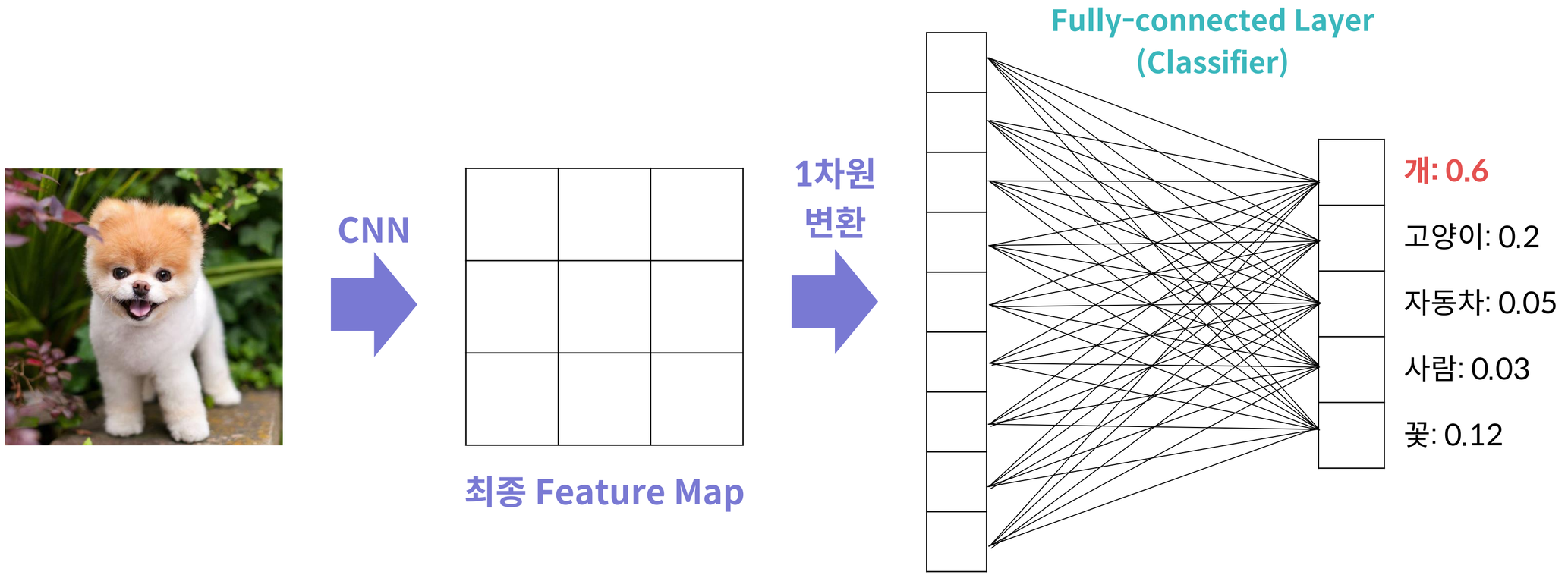
Max Pooling과 거의 동일하나, 각 영역의 **평균값**을 계산하여 새로운 Feature Map을 구성

Pooling Layer



- 일반적으로 **Max Pooling**을 많이 사용
  - Feature Map에 존재하는 Feature 중 가장 영향력이 큰 Feature만 사용
- Feature Map의 채널이 여러 개면 **각 채널별**로 Pooling 연산 수행
- **추가 Pooling Layer**
  - **Global Average Pooling**: 전체 Feature Map에서 하나의 **평균값**을 계산
  - **Global Max Pooling**: 전체 Feature Map에서 하나의 **최대값**을 계산
  - 둘 다 마찬가지로 채널 별로 연산
  - 여기선 **Global Average Pooling**을 많이 사용

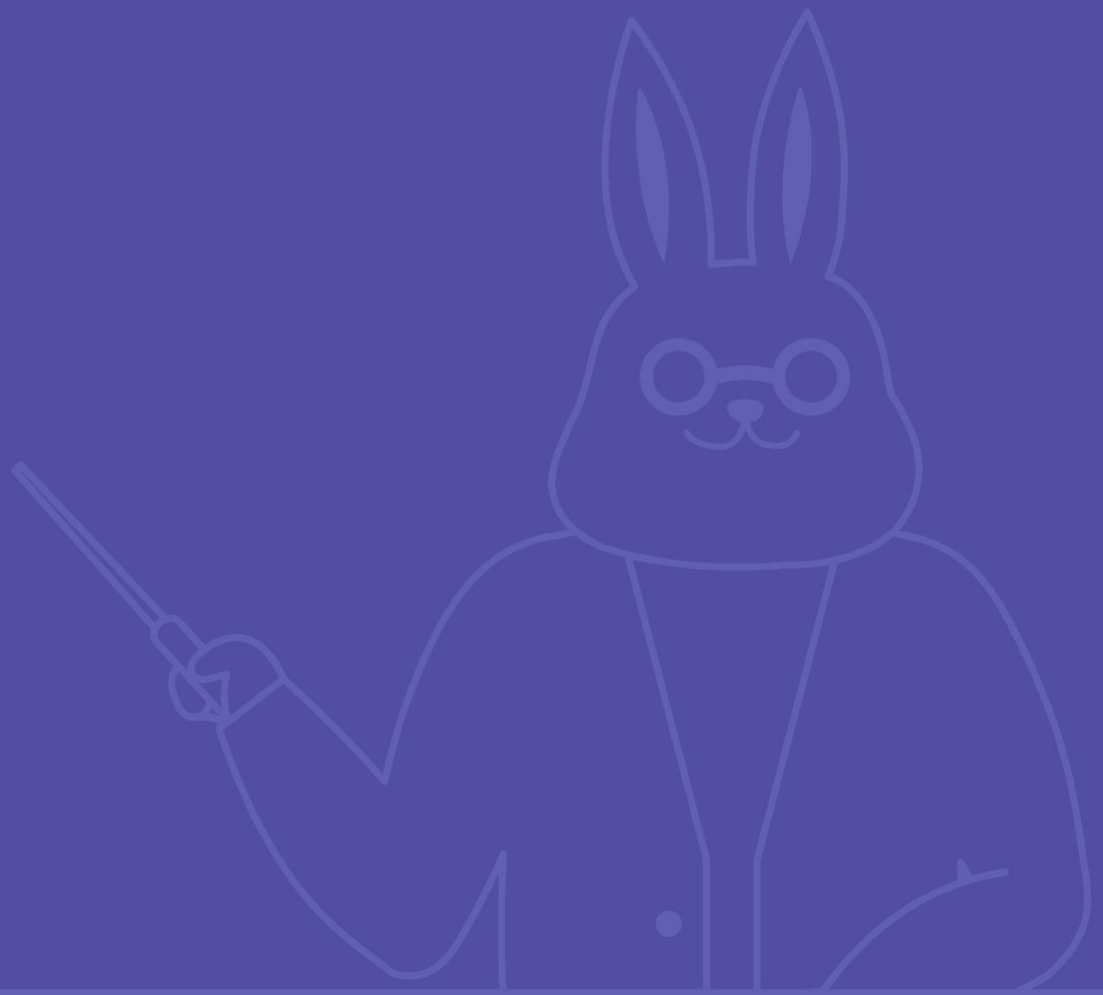
분류기 (Classifier)



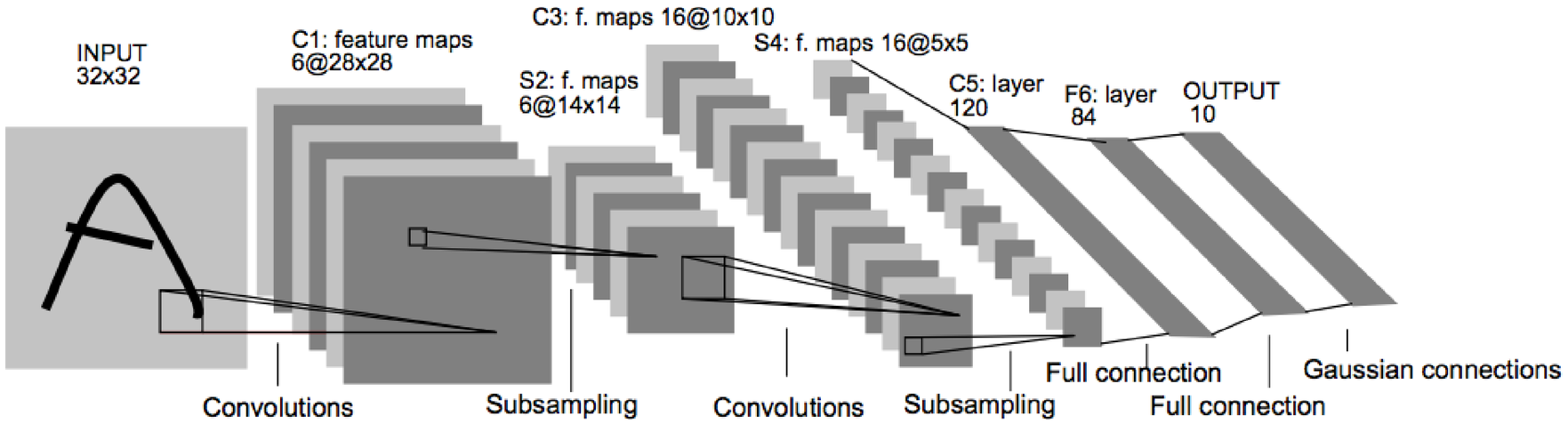
- CNN은 일반적으로 이미지 **분류** 목적으로 사용
- Feature Map을 **Fully-connected Layer에 통과시켜** 분류를 수행
- 이를 위해 Feature Map을 1차원으로 변형

03

대표적인 CNN 모델

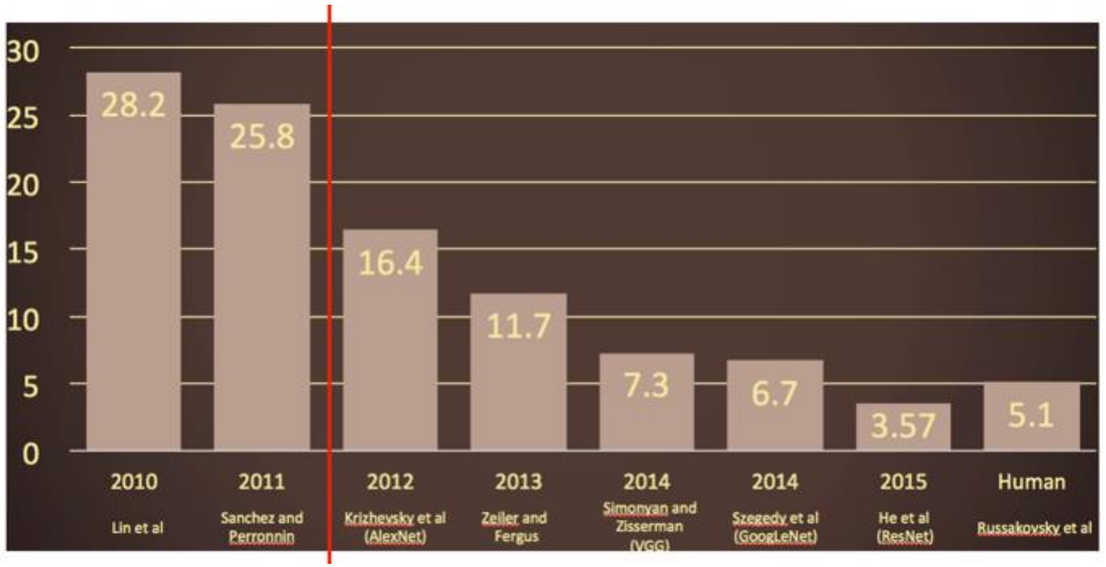


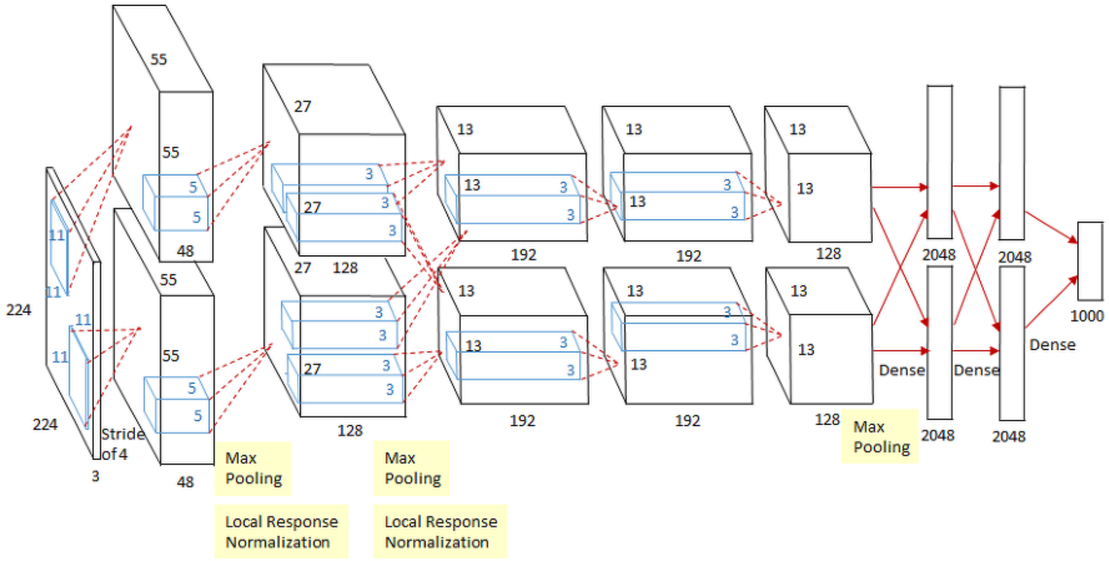
LeNet (1990)



우편번호 인식을 위한 모델

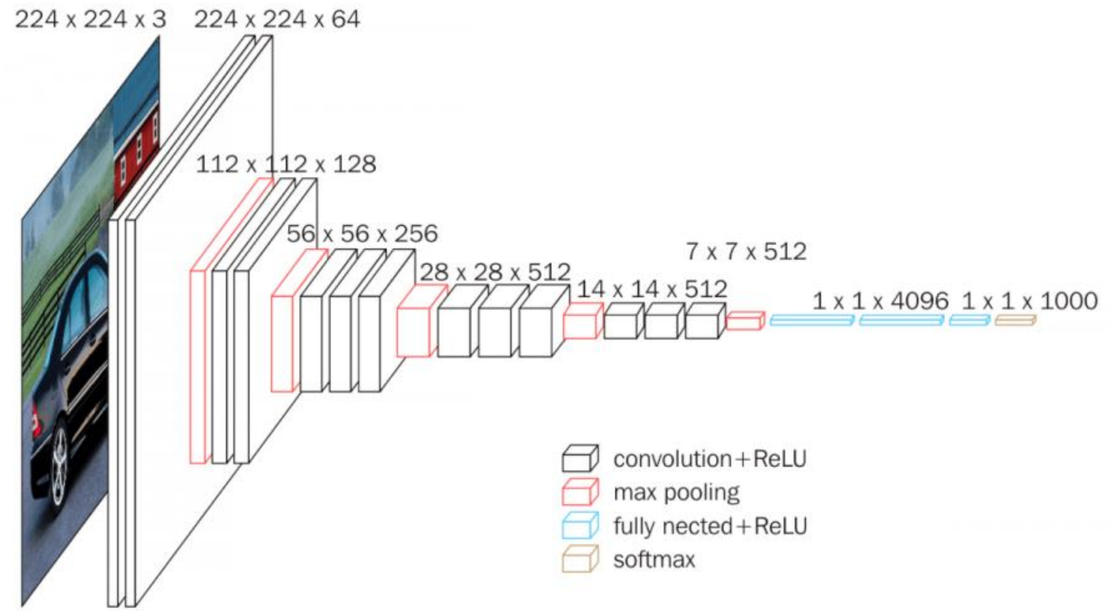
AlexNet (2012)

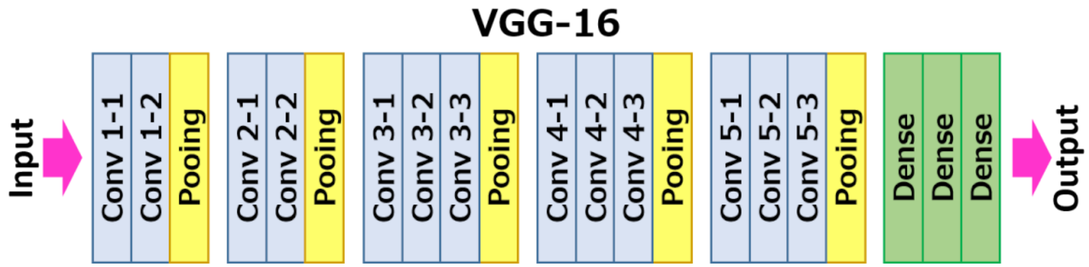




- 2012년 ImageNet Challenge 우승 → 기존 모델의 성능을 큰 폭으로 상회
- **ReLU 활성화 함수 소개**
- 딥러닝 모델 학습에 **GPU를 활용** → 이후로 사실상 모든 딥러닝 모델은 GPU로 학습

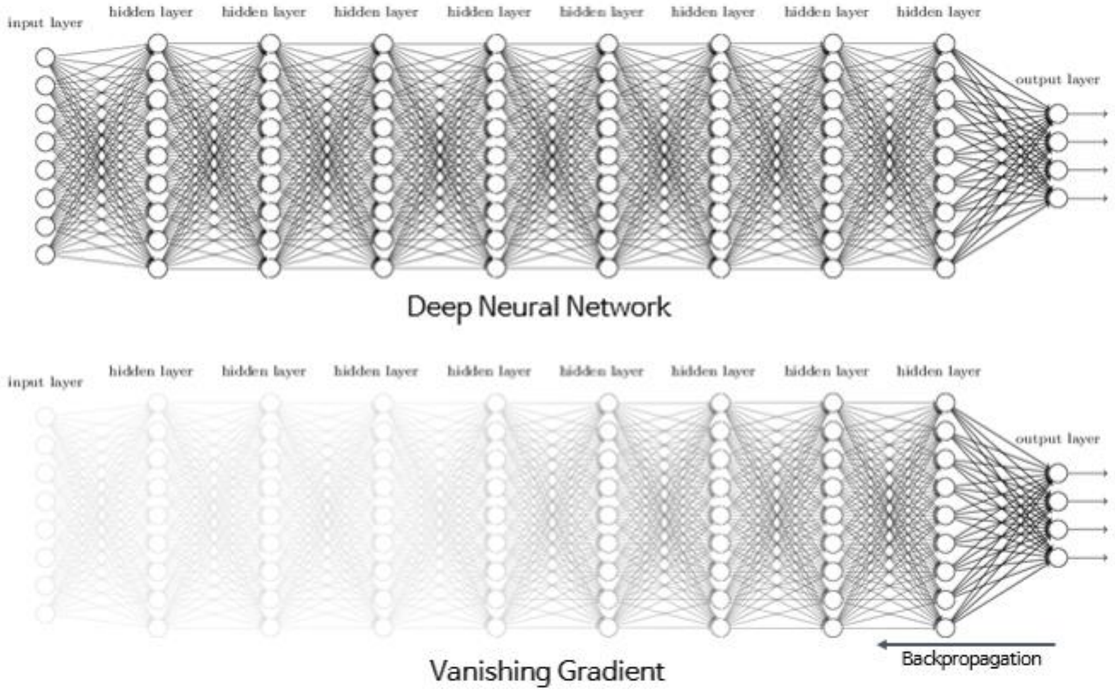
VGGNet (2014)





- 커널 사이즈를 **모두 3x3으로 통일**
- Parameter 수 증가를 억제하면서 모델 층을 더 많이 쌓을 수 있게 됨
- 층이 많을수록(즉, 모델이 깊을수록) 일반적으로 성능이 향상됨

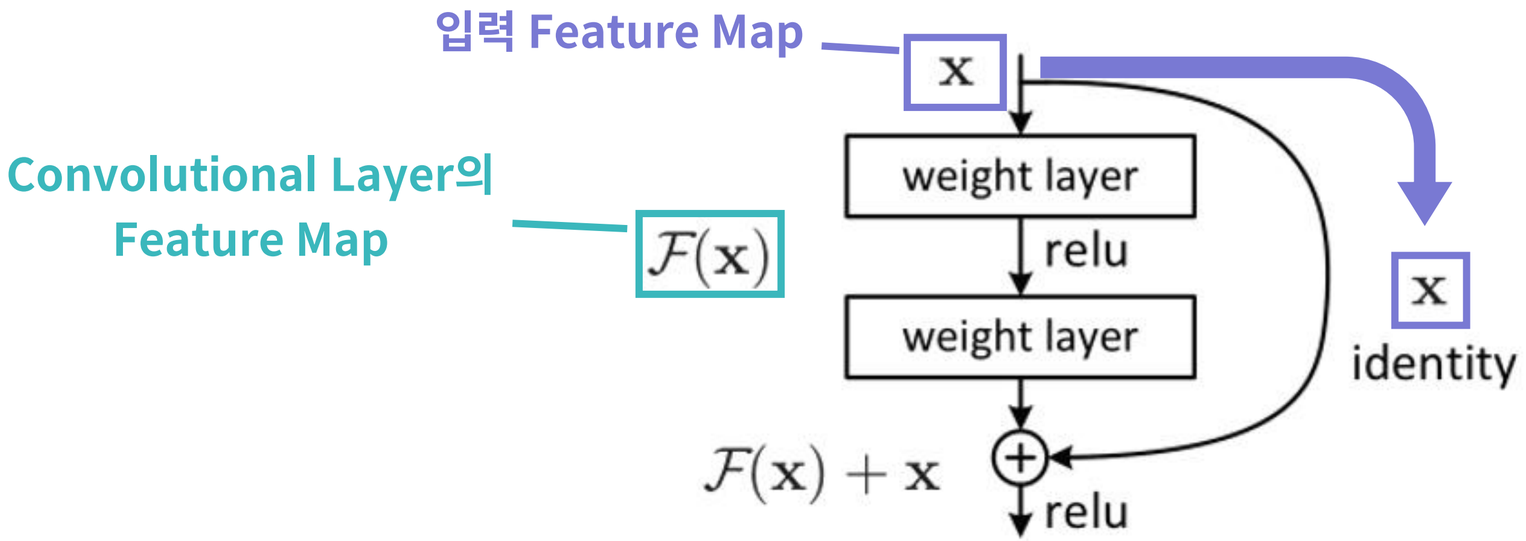
ResNet (2015)



- Layer 개수를 **최대 152개**까지 늘림
- 깊은 모델에서 필연적으로 나타나는 현상: Vanishing Gradient

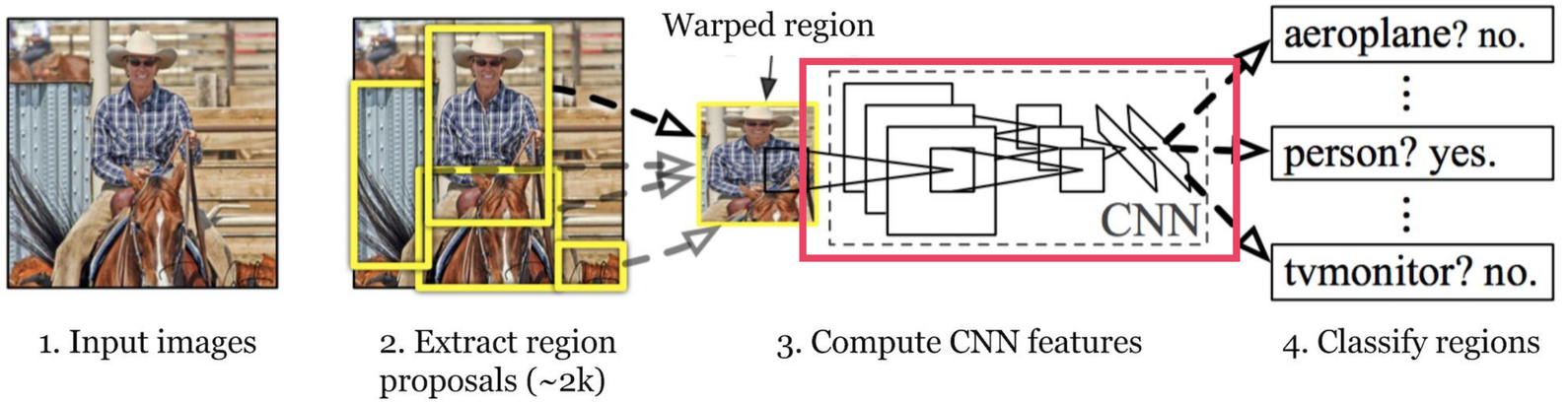
- **Vanishing Gradient (기울기 소실)**
  - 역전파 과정에서 **기울기 값이** 점점 작아지다 **0에 수렴하면서** 발생
  - 모델 학습에 **오랜 시간이 걸리거나 아예 학습이 멈추게 됨**

Residual Connection



- Vanishing Gradient 문제를 해결하기 위한 구조
- 이를 통해 Layer 개수를 극적으로 늘림
- 기존 Convolutional Layer들을 **우회**하는 연결
  - **입력 Feature Map**이 우회로를 통과하여 **Convolutional Layer의 Feature Map**과 더해짐
  - 기울기 값이 **항상 1 이상이 되어** 기울기 소실 문제를 방지

분류 작업이 아닌 경우에 사용하는 모델은?



- 지금까지 나온 모델은 모두 **분류 모델**
- 분류 작업이 **아닌** 경우에 사용하는 모델은?
  - 일반적으로 분류 모델과 유사하게 CNN을 구성
  - 모델의 **출력값, 손실 함수, 데이터셋 구성** 등이 완전히 다르게 이루어짐
  - 예) YOLO, R-CNN, U-Net 등

크레딧

/* elice */

코스 매니저
김창환

콘텐츠 제작자
김창환

강사
김창환

감수자
-

디자이너
강혜정

연락처

TEL

070-4633-2015

WEB

https://elice.io

E-MAIL

contact@elice.io