

# CNN 기초

Deep Session 4차시

# CONTENTS.

---

## 01. CNN

---

- CNN
- CV task
- MLP와 CNN
- CNN Input

## 02. CNN 구조

---

- CNN 구조
- Convolution Layer
- Pooling Layer
- Fully Connected Layer

## 03. 이미지 전처리

---

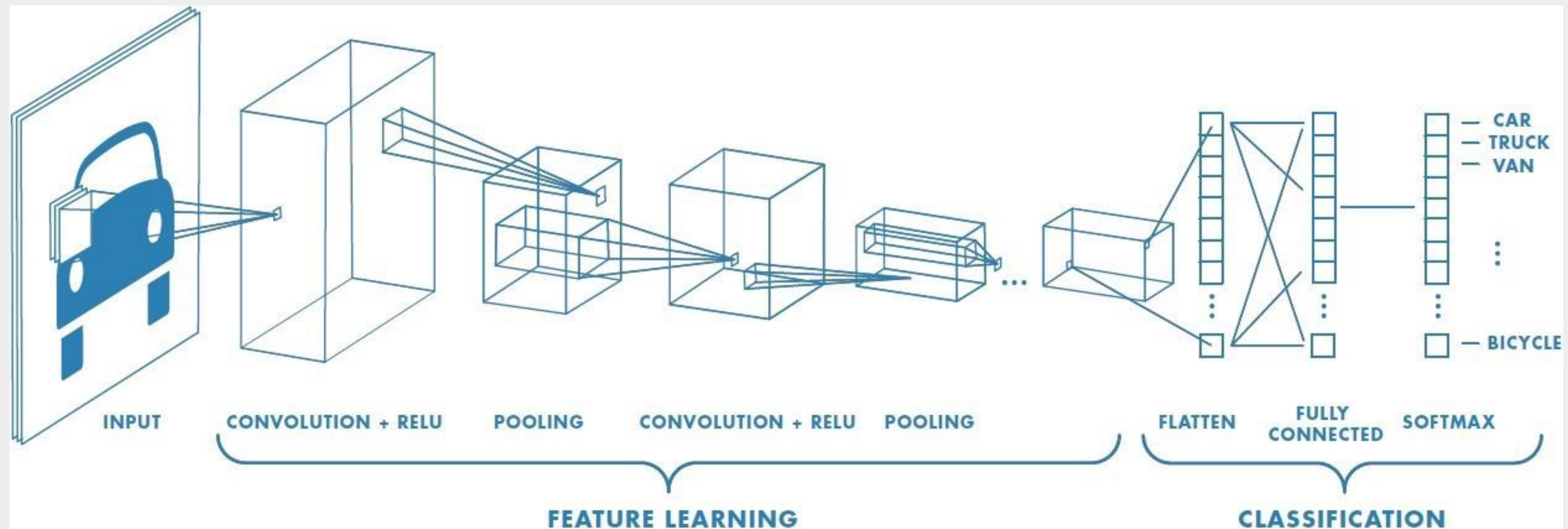
- 정규화
- Data Augmentation 기법

# 1. CNN

## CNN

### CNN (Convolutional Neural Network)

Convolution 연산을 통해 이미지의 지역 정보를 학습하는 Neural Network



# 1. CNN

## CNN

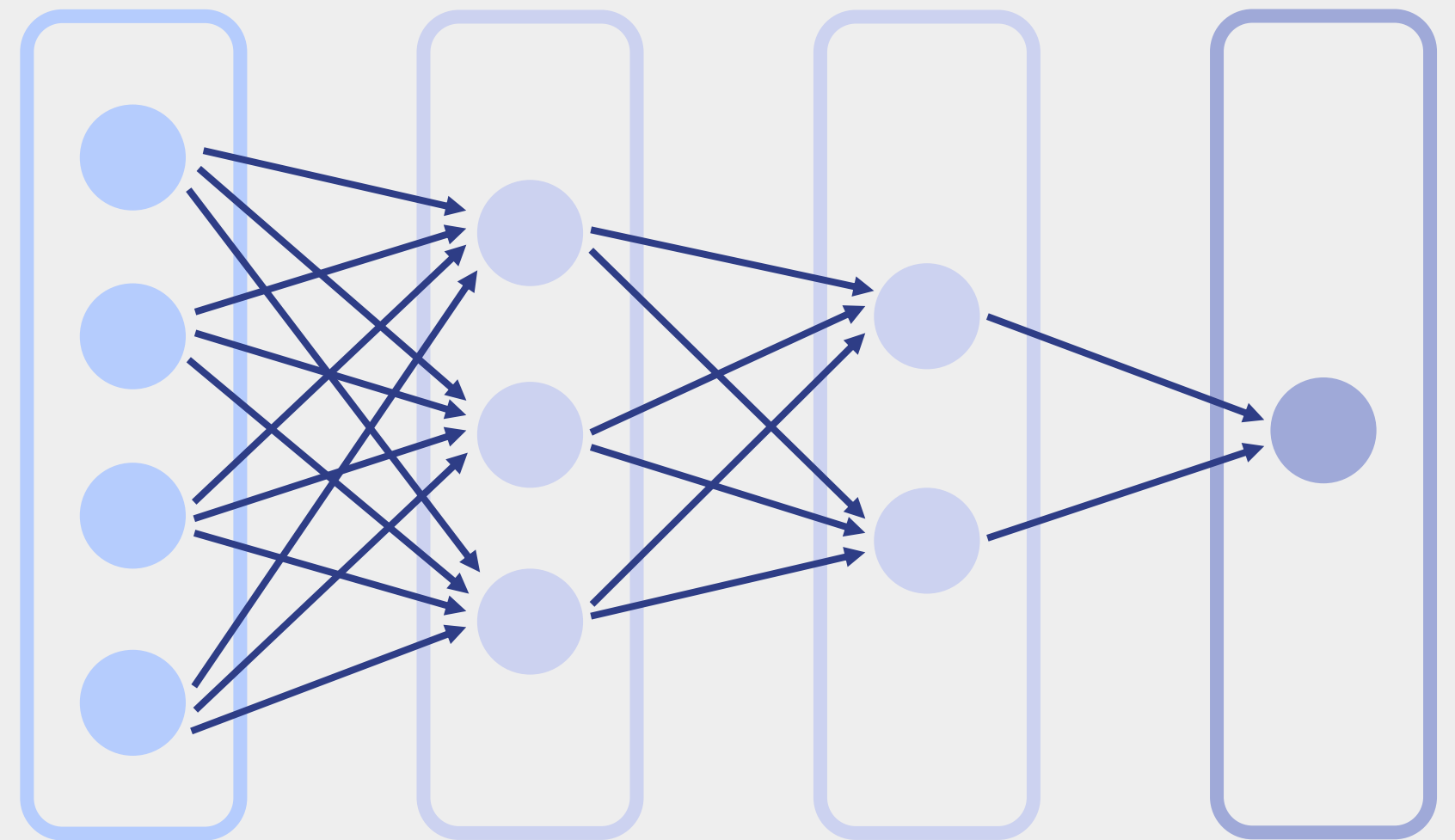
### DNN의 문제점

1. 고차원 input 데이터
2. Flexibility of Topology

: Flattening 과정에서  
이미지의 공간적/지역적 정보 손실 문제 발생

➡ 이를 해결하는 모델이 CNN

이미지의 공간 정보를  
유지한 상태로 학습이 가능한 모델



# 1. CNN

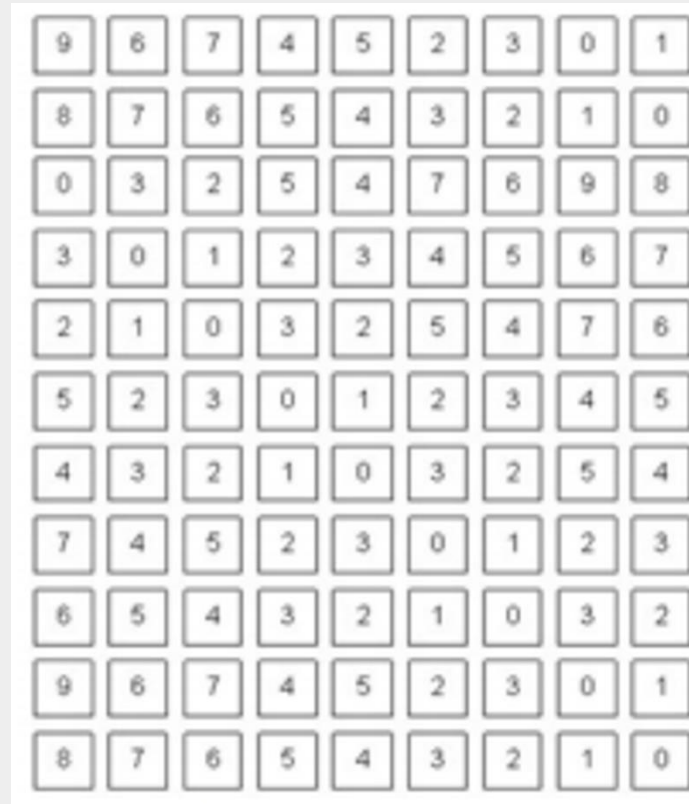
## CV task

### CV (Computer Vision)

컴퓨터가 이미지를 잘 이해할 수 있도록 하는 과제



What we see



What a computer sees

이미지가 달라지면  
(ex. 색, 위치 등)  
이미지 인식에  
어려움을 느낌





# 1. CNN

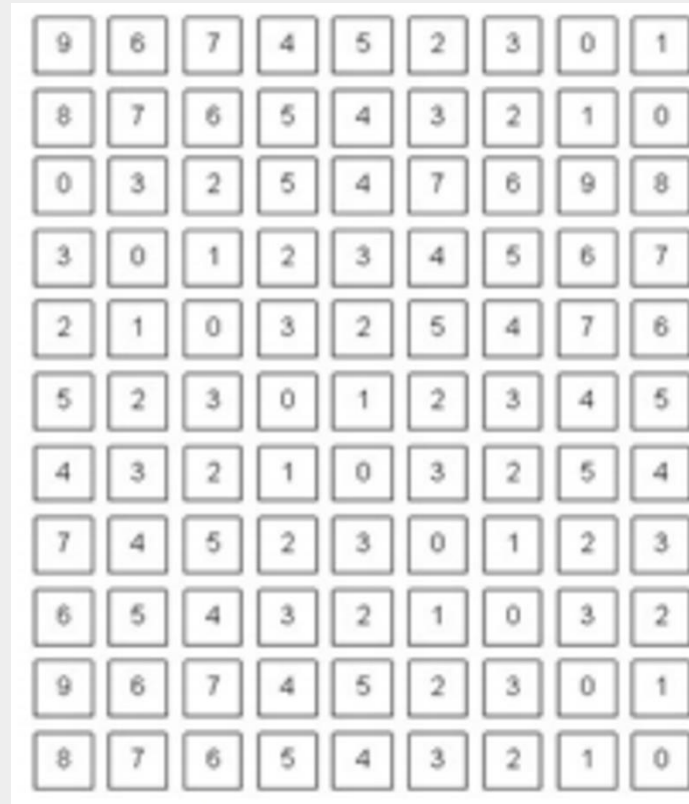
## CV task

### CV (Computer Vision)

컴퓨터가 이미지를 잘 이해할 수 있도록 하는 과제



What we see



What a computer sees

CV에서 가장 중요한 과제는  
'공간적 정보 추출'

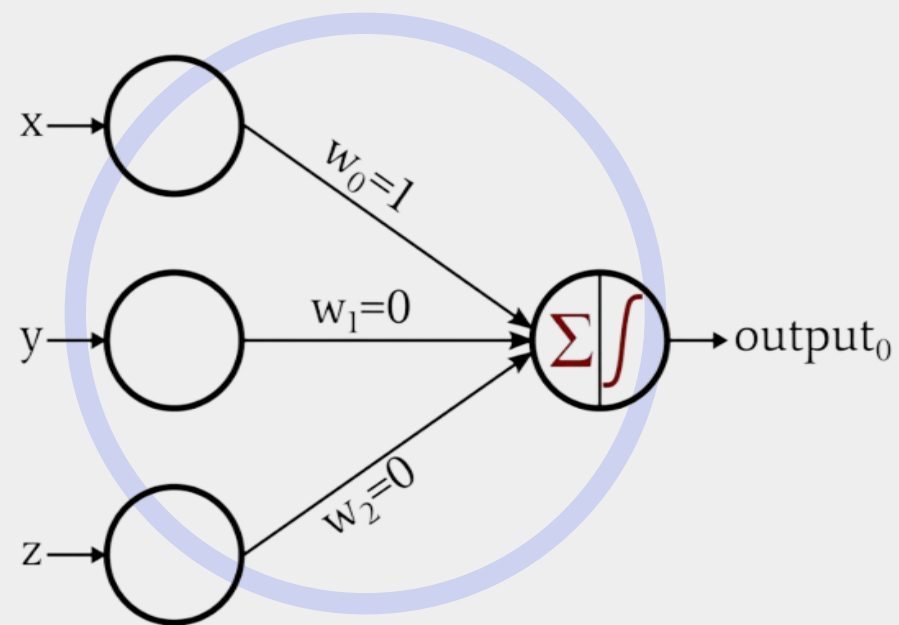


# CNN

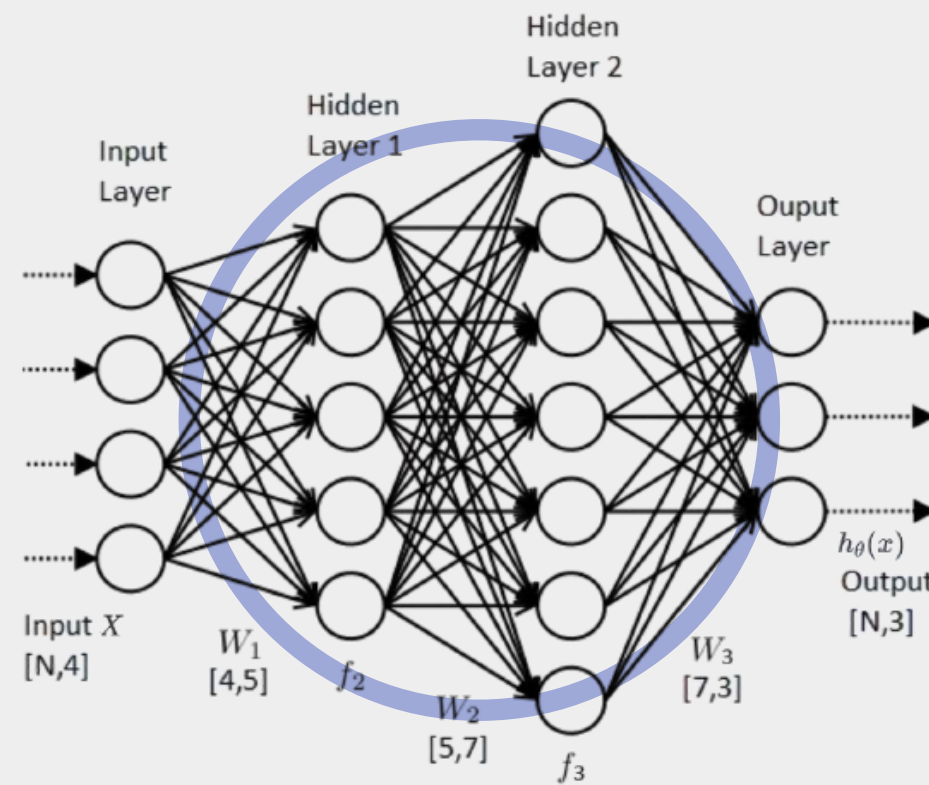


# 1. CNN

## MLP와 CNN



Perceptron



MLP

CNN

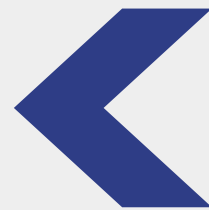
# 1. CNN

## MLP와 CNN

### MLP와 CNN

#### MLP

- input image의 3차원 형상을 고려 X
- FC Layer이 input image의 3차원 형상을 1차원으로 flatten



#### CNN

- input image의 3차원 형상을 고려 O
- 이미지의 지역 정보를 Input으로 사용
- Convolution 연산은 flatten X
- Image 안의 pixel끼리의 관계를 그대로 고려함

---

데이터의 형상 정보를 유지하는 **CNN**



# 1. CNN

## CNN input

### CNN input

입력 데이터가 이미지로 이루어졌다는 가정을 바탕으로 이루어진 모델

→ (Height, Width, Channel) 형태의 이미지

### 이미지 데이터

- 이미지가 어떤 색 공간에 저장되는지에 따라 이미지의 depth가 달라진다
- 이미지를 하나의 depth 별로 쪼개어 나타냈을 때 각 한 장의 이미지를 채널이라고 함
  - 컬러 이미지: RGB 공간에 저장, image depth = 3
  - 흑백 이미지: Gray Scale 공간에 저장, image depth = 1
- Pixel 값은 흰색의 강도를 나타낸다 -> 작을수록 검은색에 가깝고 클수록 흰색에 가깝다

## 2. CNN 구조

# CNN 구조

### CNN 구조

#### Convolution Layer

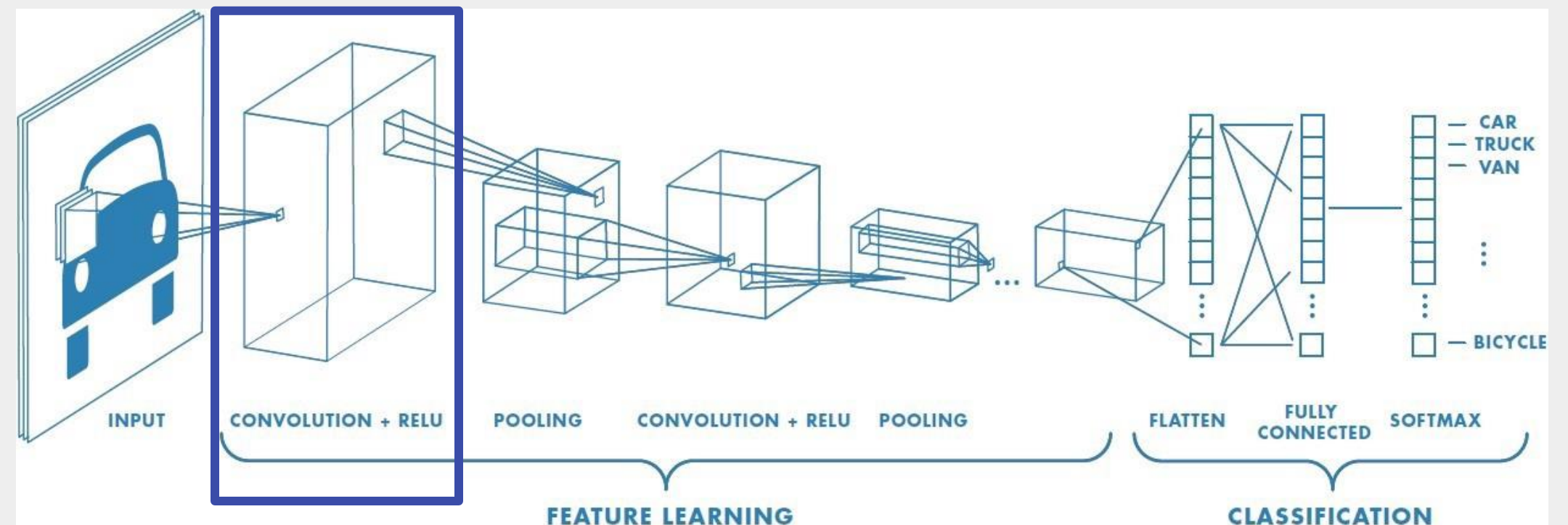
- Region Feature를 뽑아내기 위한 계층

#### Pooling Layer

- Feature Dimension을 줄이기 위한 계층

#### Fully Connected Layer

- 최종적인 분류를 위한 계층



## 2. CNN 구조

# CNN 구조

### CNN 구조

#### Convolution Layer

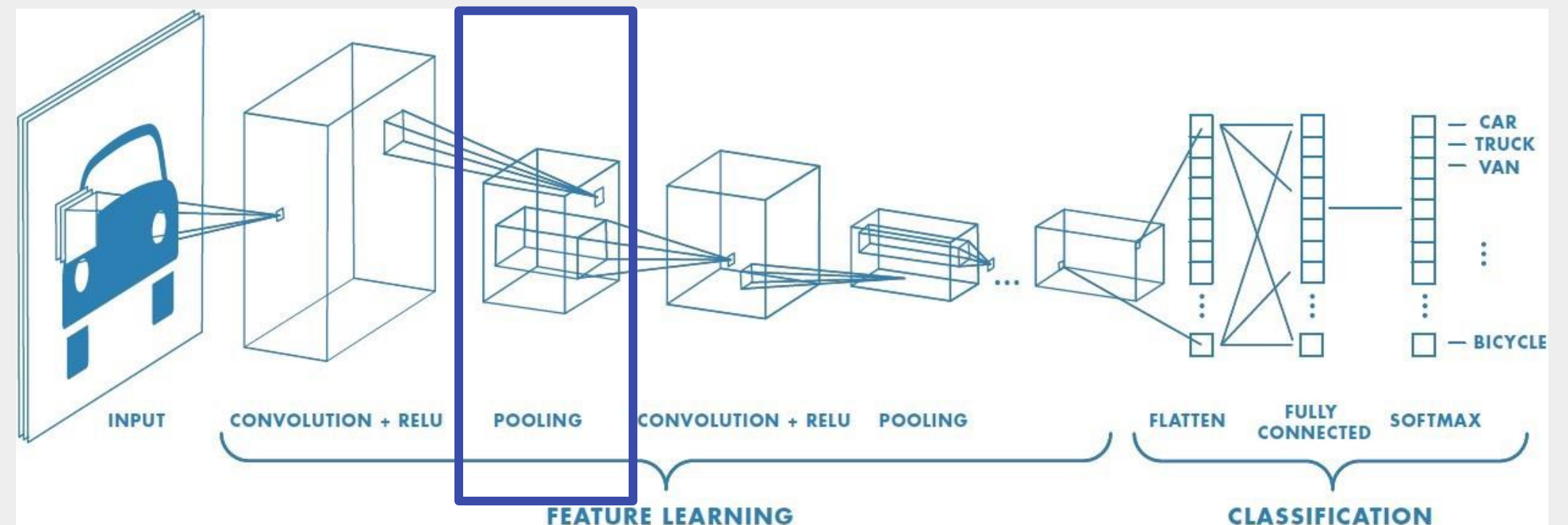
- Region Feature를 뽑아내기 위한 계층

#### Pooling Layer

- Feature Dimension을 줄이기 위한 계층

#### Fully Connected Layer

- 최종적인 분류를 위한 계층



## 2. CNN 구조

# CNN 구조

### CNN 구조

#### Convolution Layer

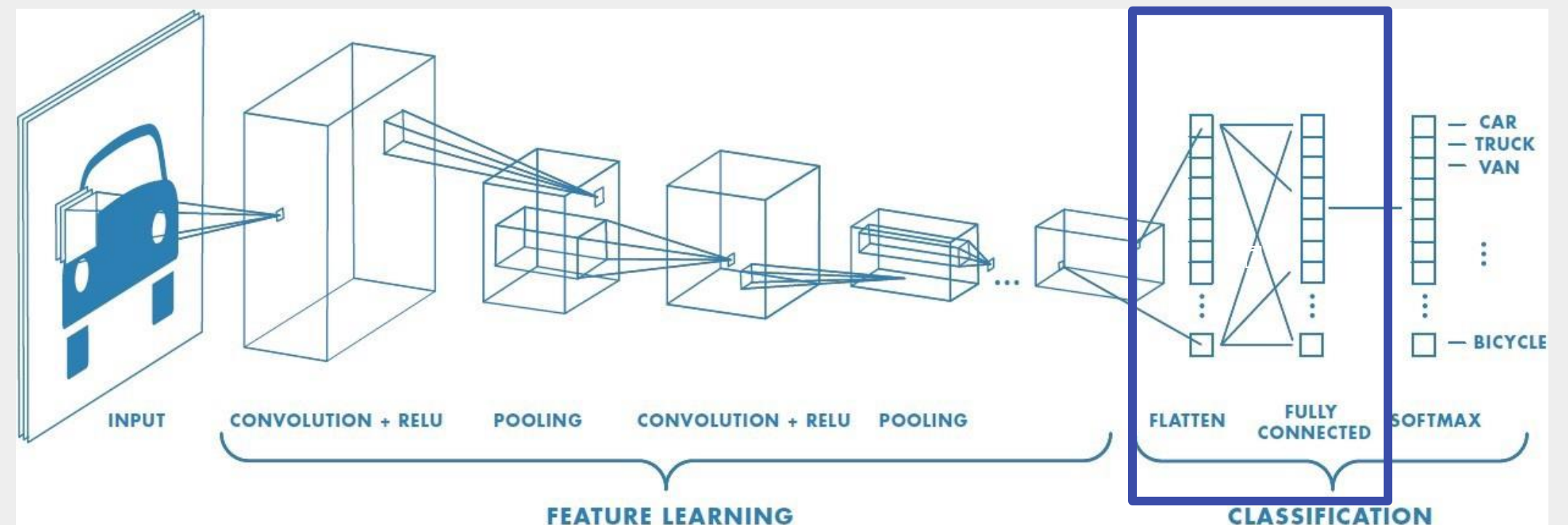
- Region Feature를 뽑아내기 위한 계층

#### Pooling Layer

- Feature Dimension을 줄이기 위한 계층

#### Fully Connected Layer

- 최종적인 분류를 위한 계층

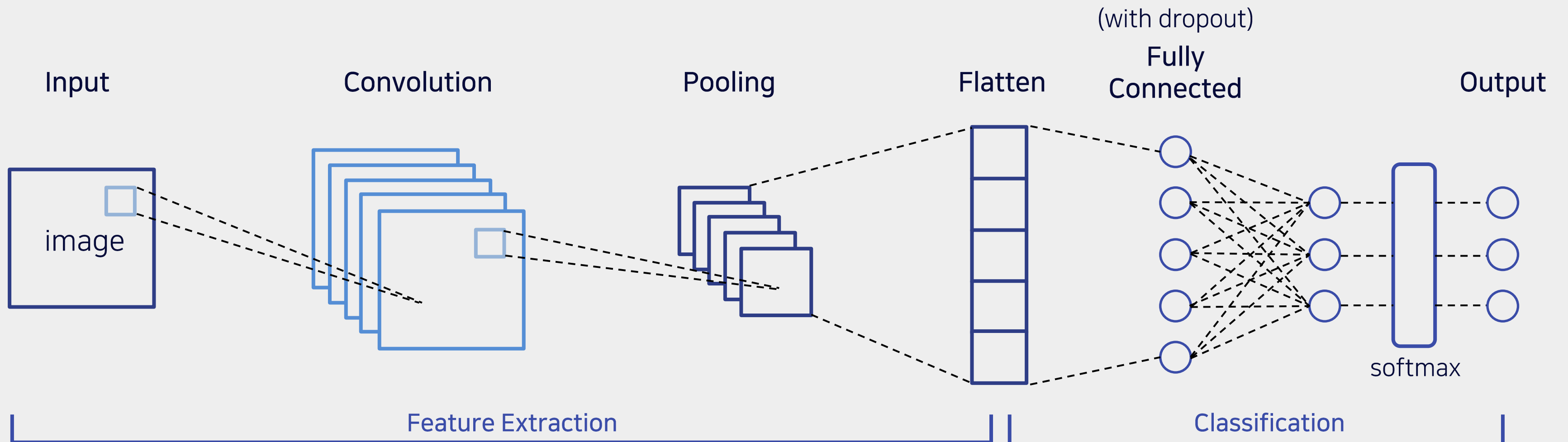


## 2. CNN 구조

# Convolution Layer

### Convolution Layer

입력 이미지와 필터 간의 convolution 연산을 통해 새로운 특성 맵(feature map)을 생성

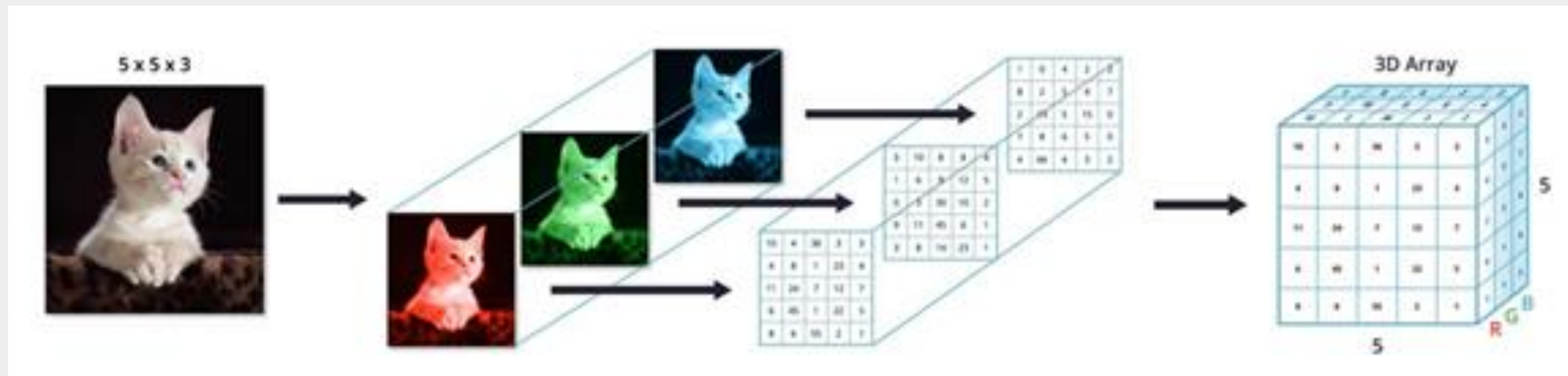


## 2. CNN 구조

# Convolution Layer

### channel

- 흑백 이미지는 2차원 형상의 데이터로, 3차원 형상으로 표현하면 1개의 채널로 구성  
ex. (5, 5, 1)
- 컬러 이미지는 각 픽셀을 RGB 3개의 실수로 표현한 3차원 형상의 데이터로, 3개의 채널로 구성  
ex. (5, 5, 3)



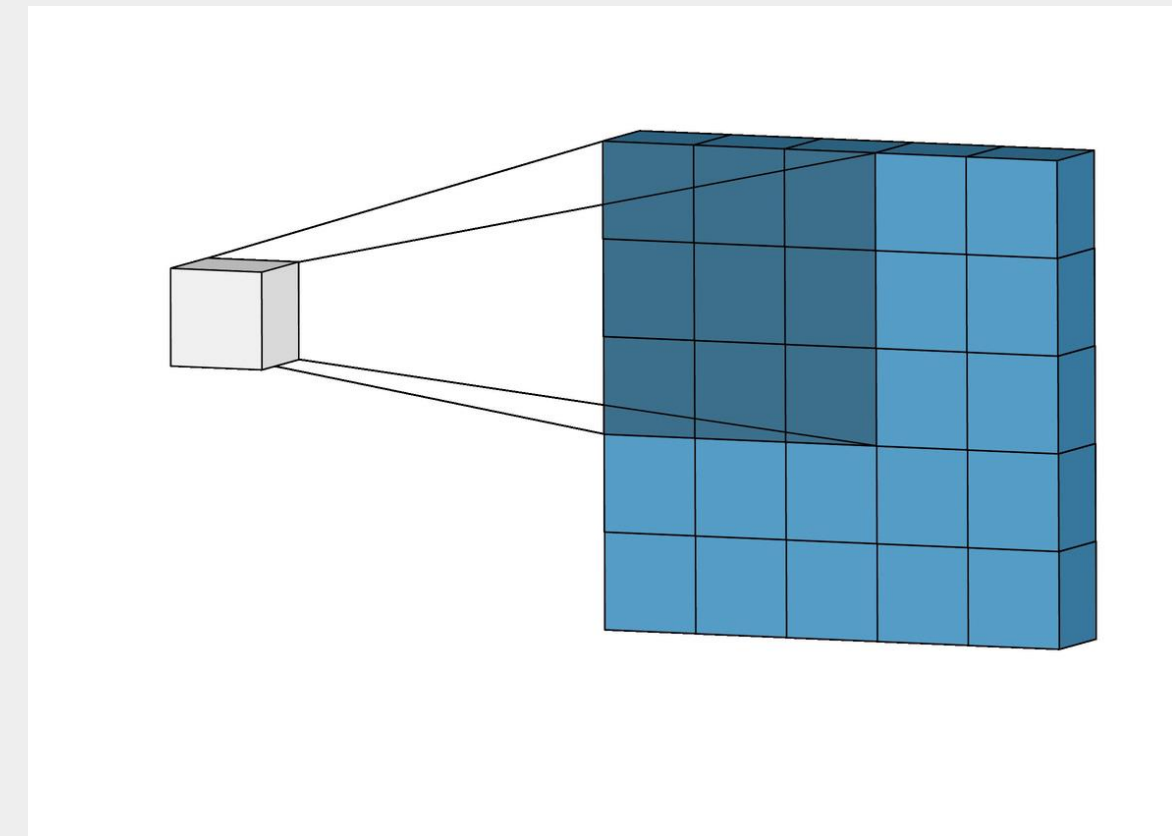


## 2. CNN 구조

# Convolution Layer

### filter(=kernel)

- 이미지의 region별 feature를 찾아내기 위한 파라미터  
➔ 커널 행렬의 원소들은 Neural Network에서의 가중치
- 필터 크기만큼 이미지를 확대해서 해당하는 부분만 확인
- 일반적으로 3x3 또는 5x5 사용

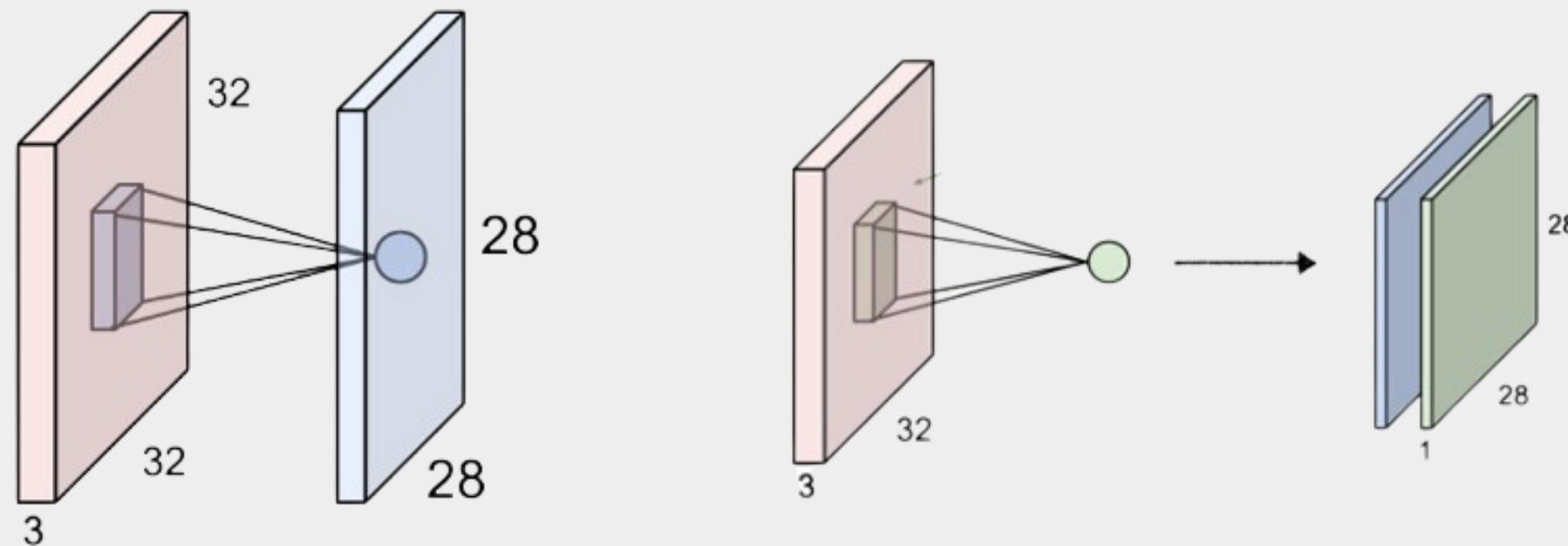


## 2. CNN 구조

# Convolution Layer

### parameter sharing

- 특정한 공간 위치( $x,y$ )에서 유용한 feature라면 다른 위치( $x_2,y_2$ )에서도 유용할 것이라는 가정에서 나온 개념
- 이미지 모든 위치에 걸쳐서 동일한 필터(가중치와 편향) 적용한다고 가정
- Specific centered structure의 경우 적합하지 않음 ex. 얼굴 사진 인식



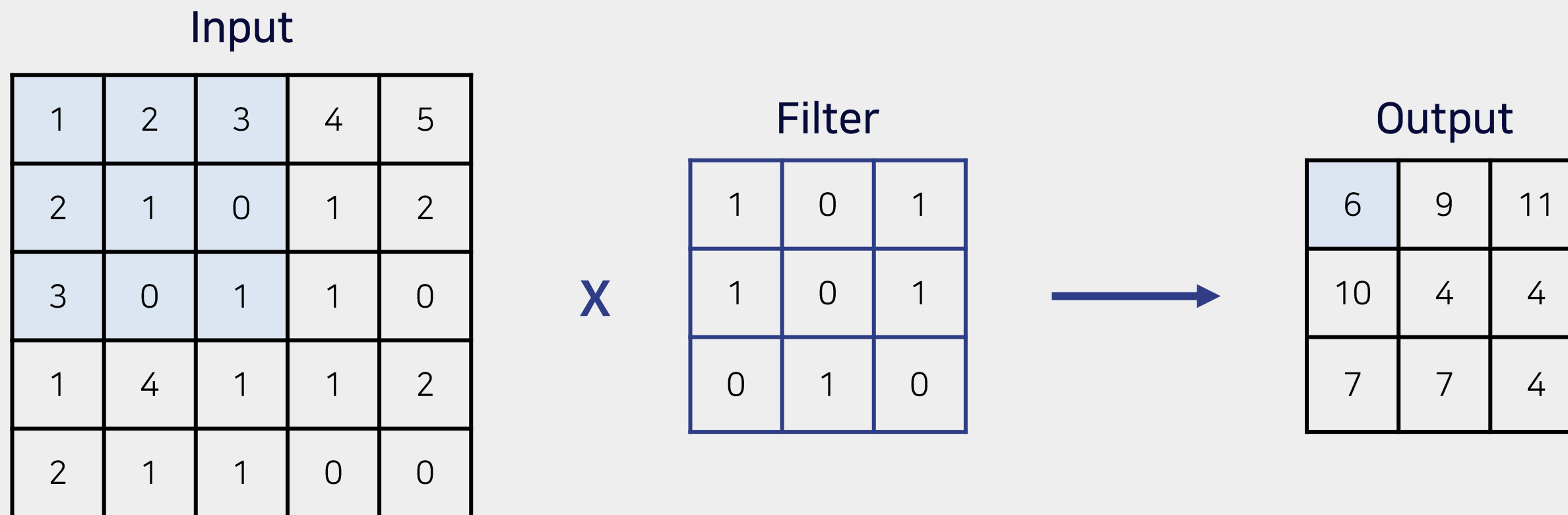
## 2. CNN 구조

# Convolution Layer

### Convolution 연산

filter가 일정 간격만큼 이동하여 Input과 대응하는 위치의 원소끼리 곱한 후 그 총합을 구하는 연산

ex. 5x5 input에 3x3 사이즈의 커널로 합성곱 연산 수행



$$(1 \times 1) + (2 \times 0) + (3 \times 1) + (2 \times 1) + (1 \times 0) + (0 \times 1) + (3 \times 0) + (0 \times 1) + (1 \times 0) = 6$$

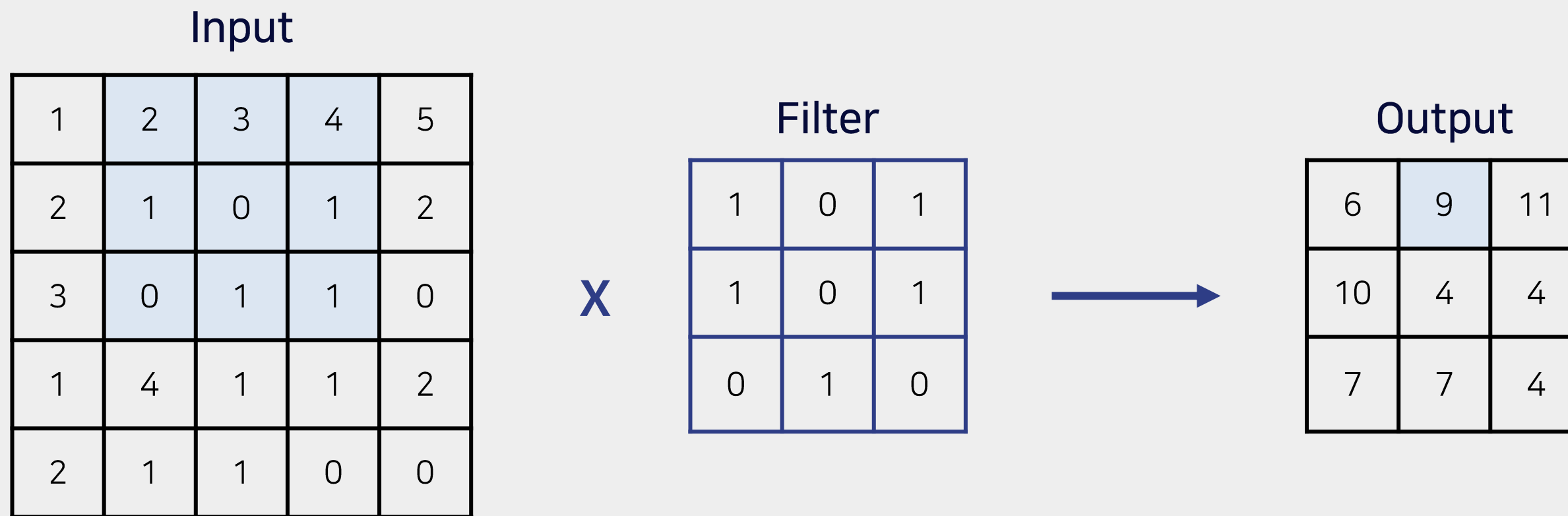
## 2. CNN 구조

# Convolution Layer

### Convolution 연산

filter가 일정 간격만큼 이동하여 Input과 대응하는 위치의 원소끼리 곱한 후 그 총합을 구하는 연산

ex. 5x5 input에 3x3 사이즈의 커널로 합성곱 연산 수행



$$(2 \times 1) + (3 \times 0) + (4 \times 1) + (1 \times 1) + (0 \times 0) + (1 \times 1) + (0 \times 0) + (1 \times 1) + (1 \times 0) = 9$$

## 2. CNN 구조

# Convolution Layer

### MLP의 가중치

Input Image

1	2	3
2	1	0
3	0	1



Input Layer

1
2
3
2
1
0
3
0
1

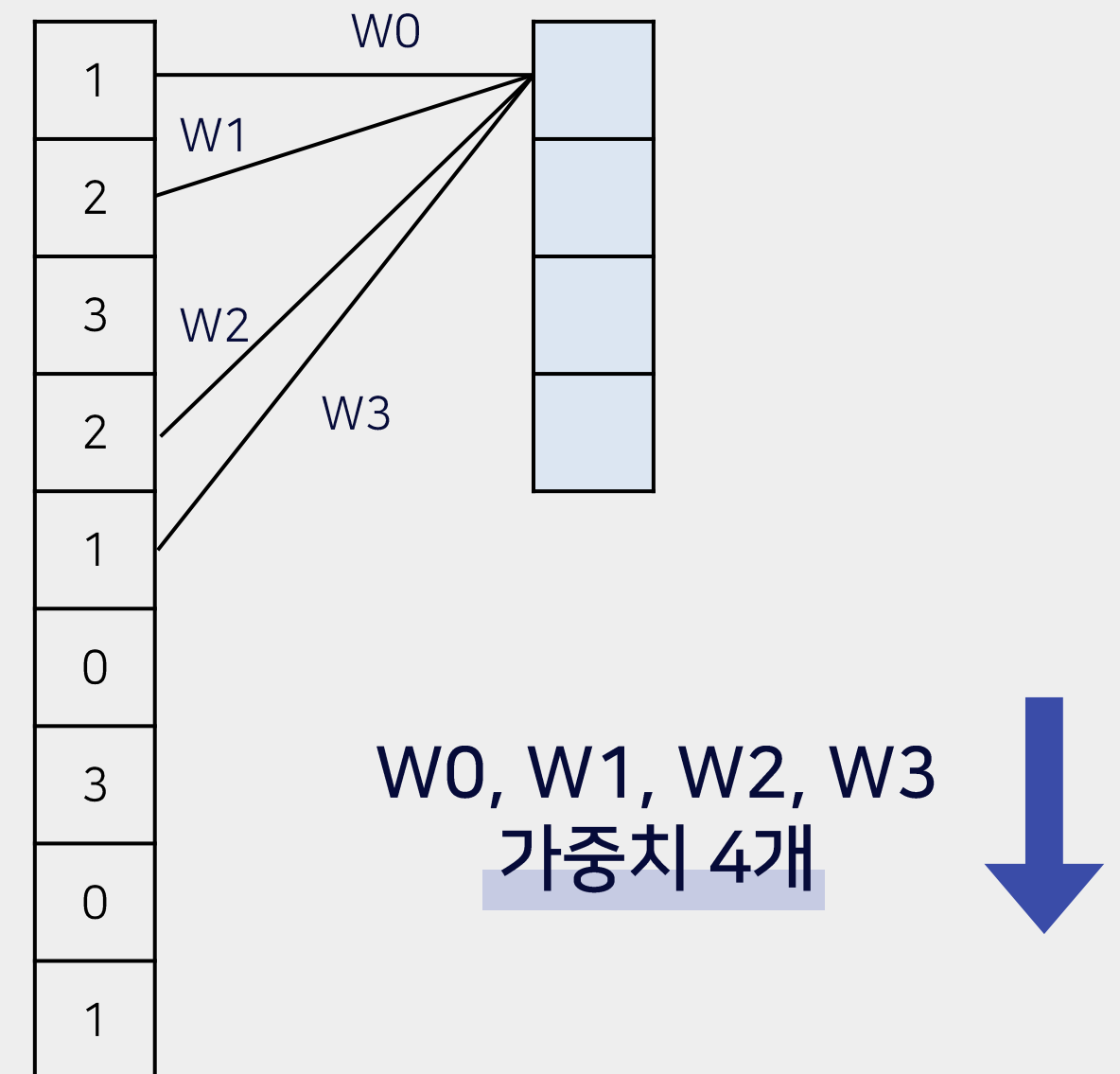
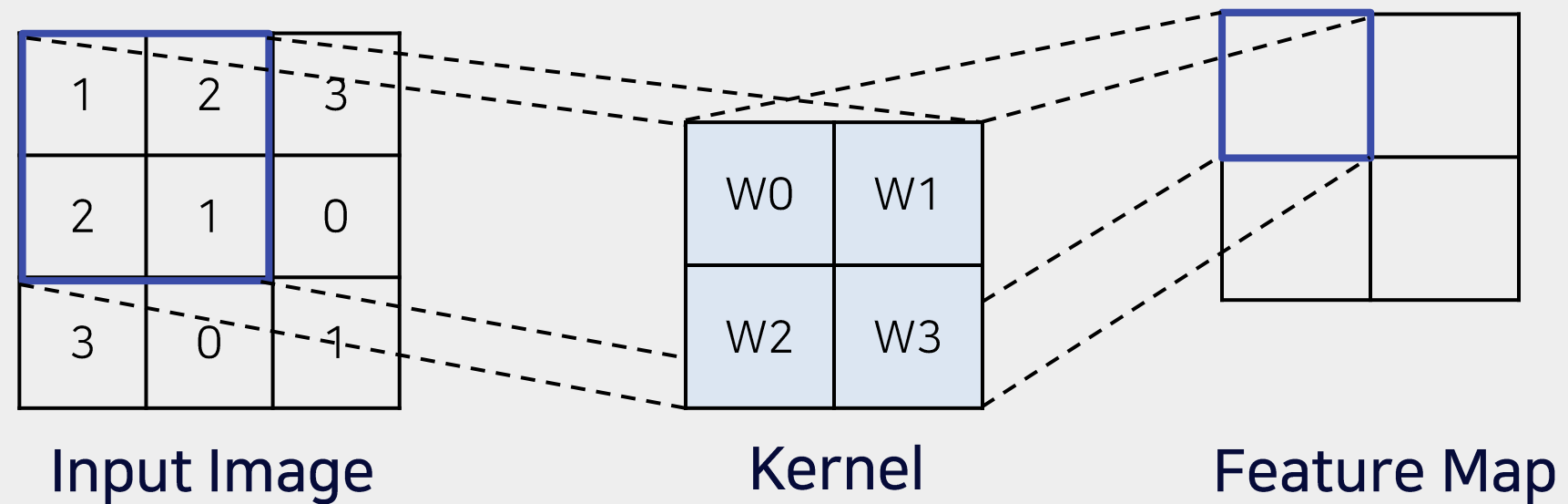
Hidden Layer


9 X 4 = 36개 의 가중치

## 2. CNN 구조

# Convolution Layer

### CNN의 가중치



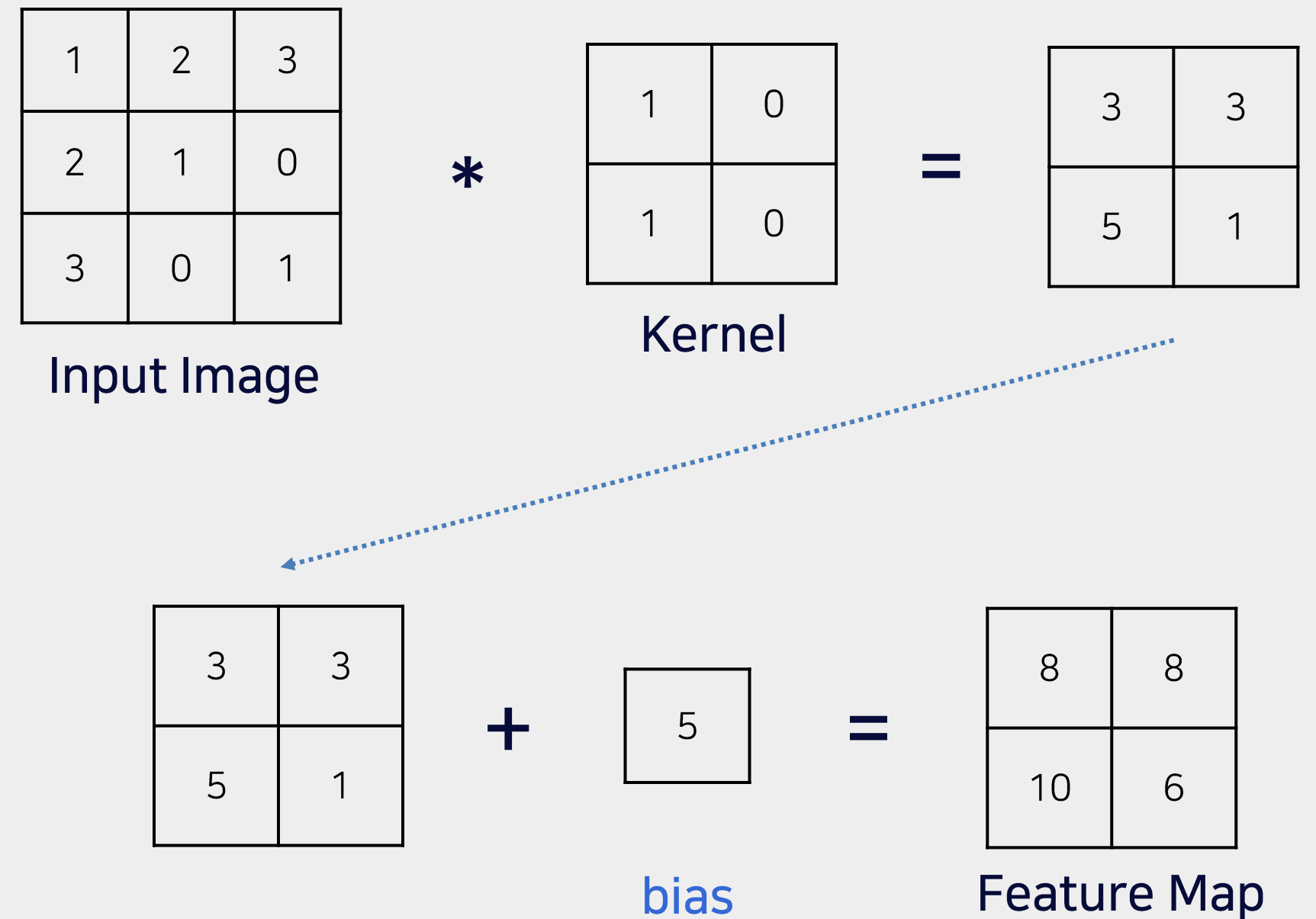


## 2. CNN 구조

# Convolution Layer

### CNN의 편향

- 커널을 적용한 뒤에 편향을 더한다.
- 편향은 하나의 값만 존재하며,  
커널이 적용된 결과의 모든 원소에 더해진다.

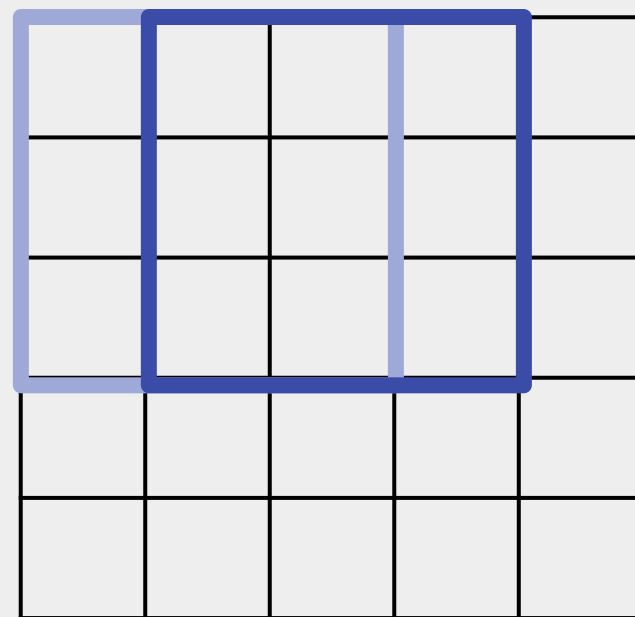


## 2. CNN 구조

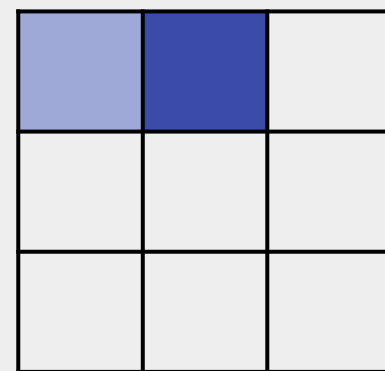
# Convolution Layer

### Hyper Parameter

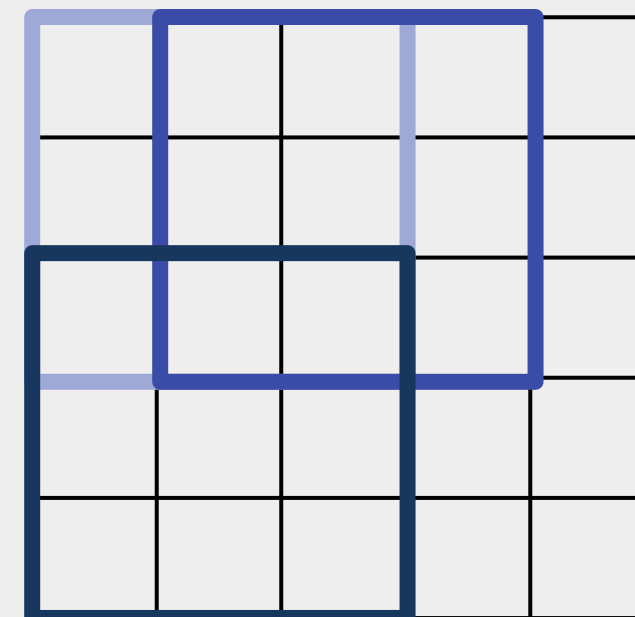
- **filter size**: filter의 크기
- **depth**: 사용하는 filter의 개수
- **stride**: filter가 입력 데이터에서 convolution 연산을 수행하는 간격



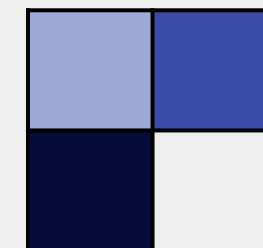
Convolution  
with Stride=1



Output



Convolution  
with Stride=2



Output

## 2. CNN 구조

# Convolution Layer

### Hyper Parameter

- padding: Convolution 연산을 수행하기 전에 데이터 주변을 특정 값 (ex.0)으로 채우는 것  
→ 출력 크기 조절을 위해 사용하는 기법

Zero Padding: Padding에 들어가는 값을 0으로 하는 것

Full Padding: 모든 요소들이 같은 비율로 연산에 참여하도록 하는 것

Same Padding: Output의 크기를 Input의 크기와 동일하도록 하는 것

Valid Padding: Padding을 안 하는 것

0	0	0	0	0
0				0
0				0
0				0
0	0	0	0	0

Padding Size = 1

## 2. CNN 구조

# Convolution Layer

### Output size

즉, feature map 크기 계산

Input에 대해 설정한 Filter(=Kernel) size, Stride, Padding에 따라 Convolution 연산을 수행

$$O_h = \text{floor}\left(\frac{I_h - K_h + 2P}{S} + 1\right)$$

$$O_w = \text{floor}\left(\frac{I_w - K_w + 2P}{S} + 1\right)$$

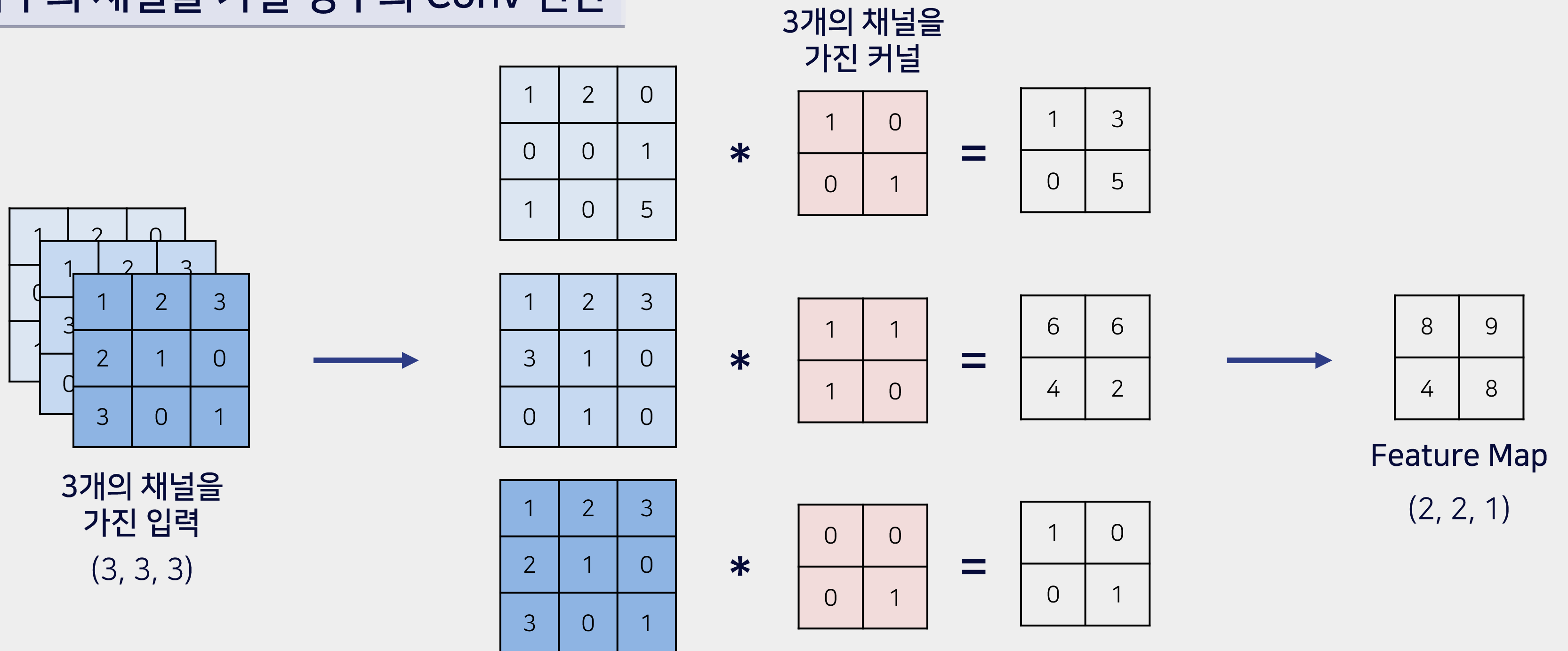
└─ 소수점 발생 시 소수점 이하를 버리는 역할

- 입력 크기 (H, W)
- 커널 크기 (K<sub>h</sub>, K<sub>w</sub>)
- 출력 크기 (O<sub>h</sub>, O<sub>w</sub>)
- padding size P
- Stride S

## 2. CNN 구조

# Convolution Layer

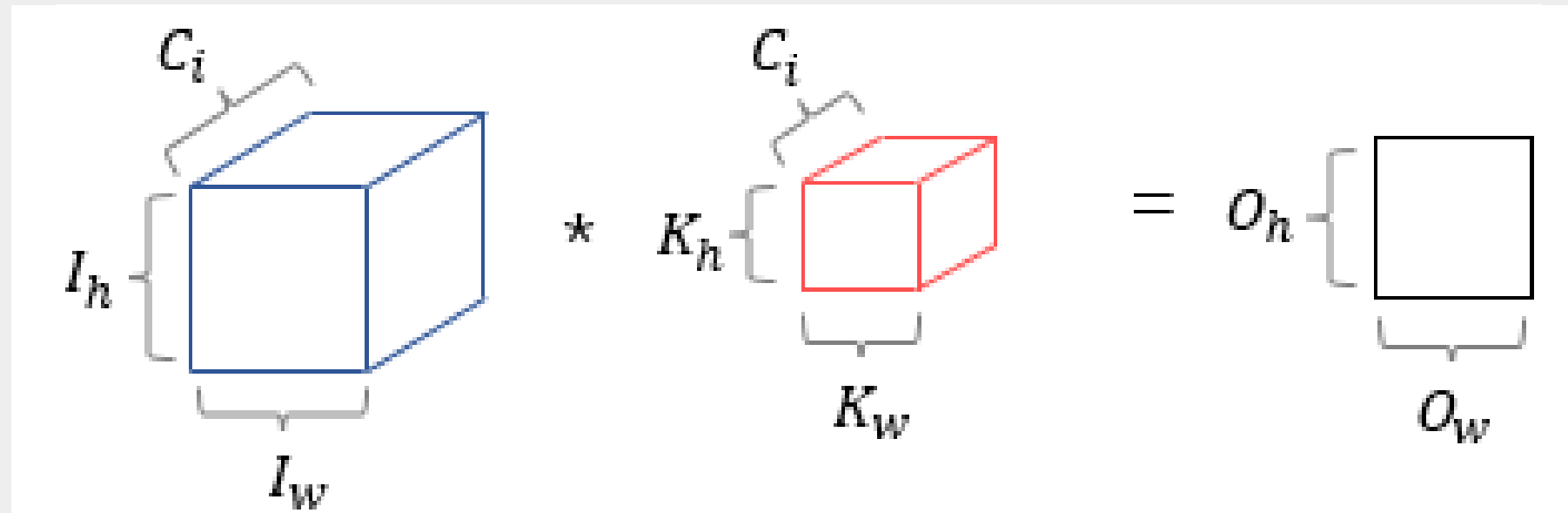
### 다수의 채널을 가질 경우의 Conv 연산



## 2. CNN 구조

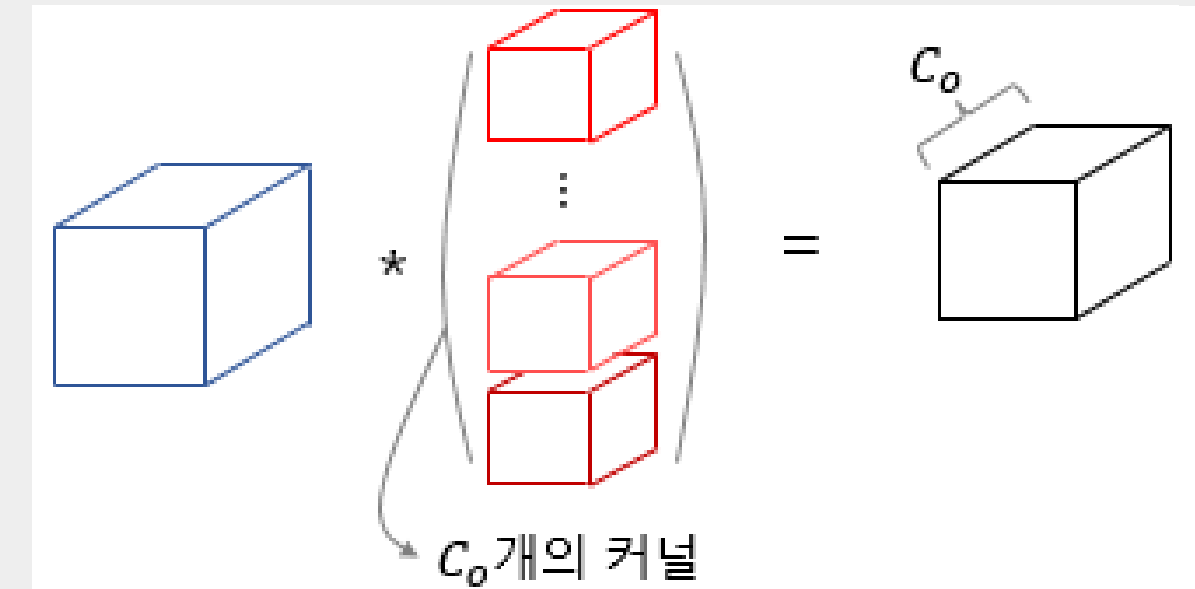
# Convolution Layer

### 다수의 채널을 가질 경우의 Conv 연산



[3차원 텐서의 Conv 연산]

입력 이미지의 채널 수 = 커널의 채널 수



[Conv 연산에서 다수의 커널을 사용한 경우]

사용한 커널 수 = feature map의 채널 수



## 2. CNN 구조

# Convolution Layer

### Activation Map

Feature Map의 각 pixel에 대해 활성화 함수를 적용한 결과  
즉, Conv Layer의 최종 출력 결과

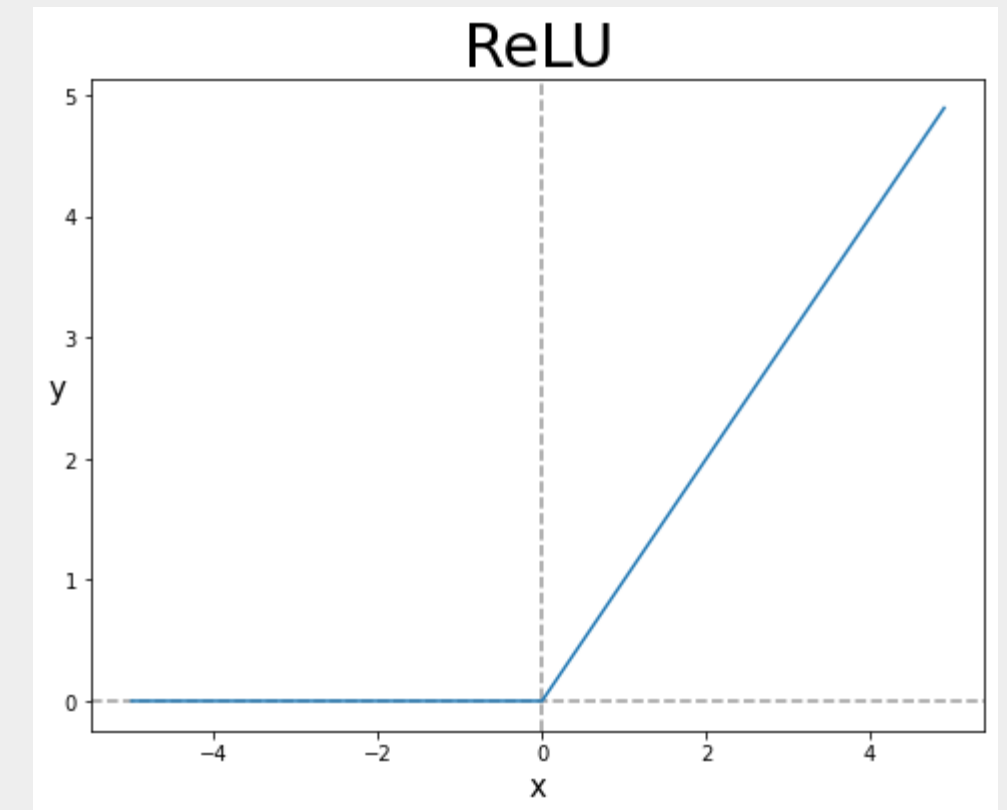
활성화 함수는 일반적으로 ReLU 함수 사용

➔ Feature map의 pixel값을 비선형 값으로 바꿔주기 위함

➔ 중요한 feature는 그 값이 유지되면서 다음 층으로 전달되고,

중요하지 않은 feature(0에 가까운 값)은 0으로 변환되어 사실상 무시됨

➔ 이미지에서 중요한 특징들만을 강조하면서도 비선형적인 문제를 효과적으로 학습함



## 2. CNN 구조

# Pooling Layer

### Pooling Layer

Convolution Layer의 출력인 Activation Map의 크기를 줄이는 연산을 수행하는 Layer

Activation Map의 인접한 부분에서 일정한 기준에 따라 픽셀 하나의 값을 뽑는다

- Max Pooling
- Average Pooling
- Min Pooling

1	2	<b>2</b>	1
4	<b>9</b>	1	0
1	5	2	<b>3</b>
4	<b>6</b>	1	2

Convolution Output



9	2
6	3

Max Pooling

## 2. CNN 구조

# Pooling Layer

### 특징

학습해야 할 parameter가 없음

- 영역에서 최댓값이나 평균을 취하는 연산만 수행하기 때문

channel 수는 변하지 않음

- 채널마다 독립적으로 계산하기 때문

feature downsampling

- 데이터의 크기를 감소시키면서도 모델의 중요한 특징을 유지

입력의 변화에 영향을 적게 받음

- 입력 데이터의 차이를 Pooling이 흡수해 사라지기 때문

최근에는 사용하지 않으려는 경향이 있음

- 많은 정보를 활용하고 학습 속도를 높일 수 있는 알고리즘이 많이 개발되었기 때문

## 2. CNN 구조

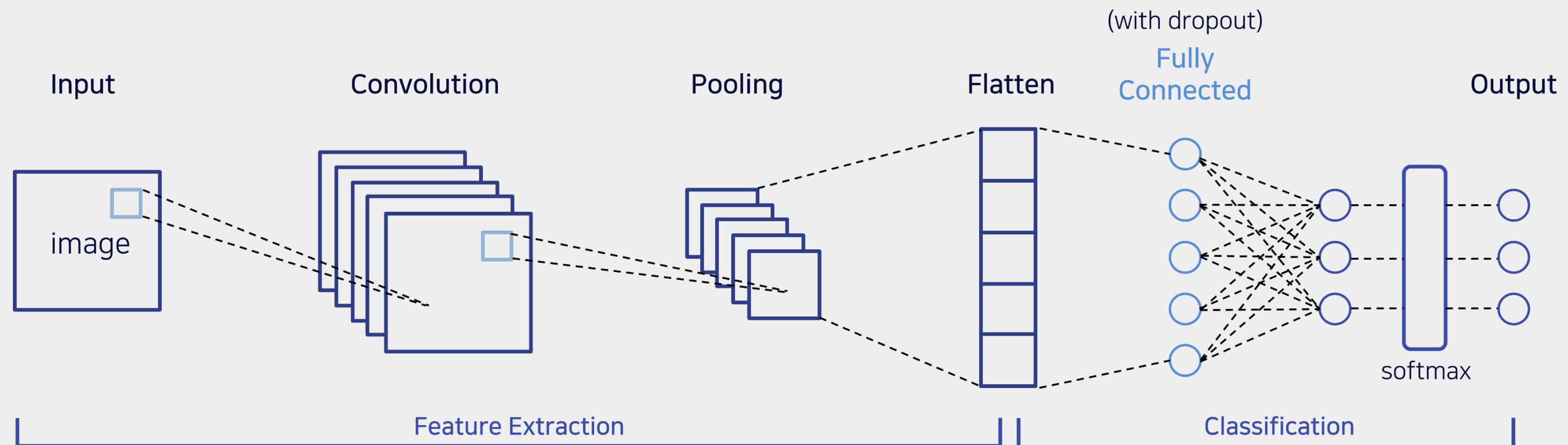
# Fully Connected Layer

### Fully Connected Layer

이미지를 정의된 라벨(클래스)로 분류하기 위한 Layer

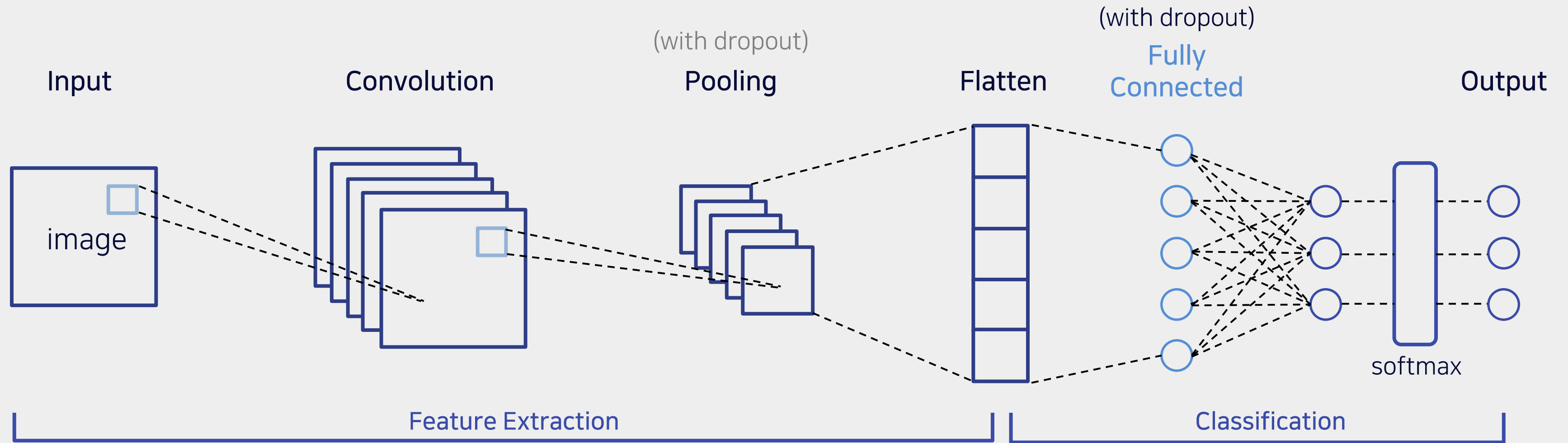
Convolution Layer, Pooling Layer를 거쳐서 나온 결과를 flattening하여 1차원 벡터로 FC Layer에 전달

➔ 여기서부터는 일반적인 MLP 구조와 동일



## 2. CNN 구조

# CNN



## 2. CNN 구조

# CNN

### CNN의 특징

- 장점

1. locally connectivity와 parameter sharing을 통해 고차원의 데이터를 오버피팅 되지 않고 다룰 수 있다.
2. 위치마다 동일한 feature를 추출하기 때문에 translation invariant 문제를 해결할 수 있다.
3. local relationship을 추출할 수 있다.

- 단점

1. 연산량이 많다.
2. 전체 데이터를 보고 특성을 추출할 수 없다.
3. 여러 Pooling층을 거치면서 작은 local 정보를 잃을 수 있다.



### 3. 이미지 전처리

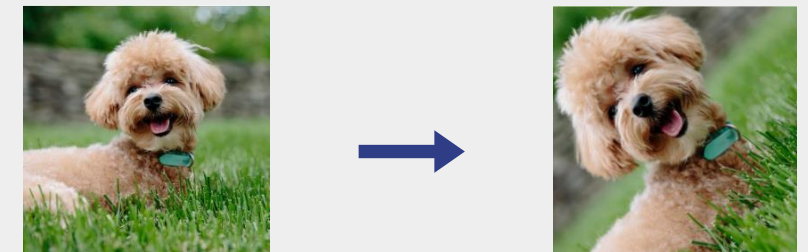
## CNN의 이미지 전처리

#### 정규화

- 이미지 픽셀 값: 일반적으로 0부터 255 사이의 값
- 이 값을 0과 1 사이로 정규화하거나, 평균을 0으로 하고 표준편차를 1로 조정

#### Data Augmentation

- 데이터를 임의로 변형해 데이터의 수를 늘려 다양한 Feature를 뽑는 방법
- 학습 시 마다 개별 원본 이미지를 변형해서 학습하는 것으로 실질적인 학습 데이터셋의 규모를 키울 수 있는 방법



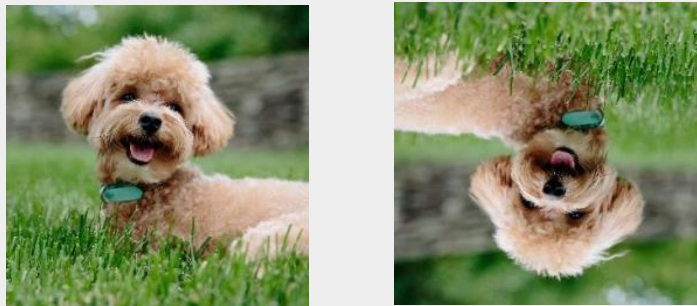
### 3. 이미지 전처리

# Data Augmentation 기법

## Data Augmentation 종류

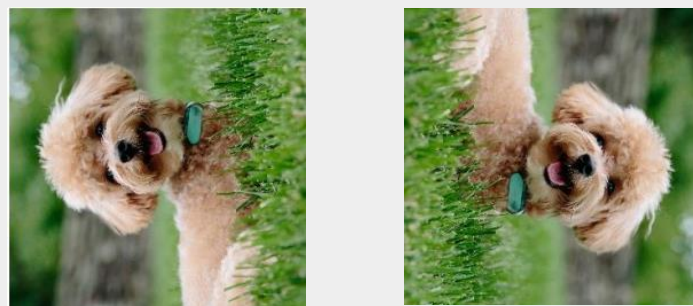
### Random Flip

- 랜덤으로 좌우/상하 반전



### Rotation

- 회전



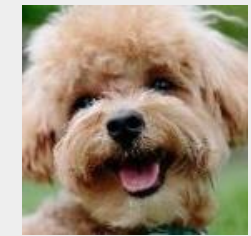
### Scaling

- 확대/축소



### Crop

- 일정 부분을 자름



### Cutout

- 이미지의 일부를 검은색 사각형 모양으로 칠하는 방법
- 0을 채워 넣는 것



강아지: 1

### Cutmix

- 여러 이미지를 일정 비율로 합치고, 이미지의 Label을 그 비율로 설정



강아지: 0.7

고양이: 0.3

### 3. 이미지 전처리

# Data Augmentation 기법

## Data Augmentation 종류

### Cutout

- PCA를 사용하여 RGB 채널의 색상강도를 조절하는 방법
- 원래의 라벨을 해치지 않으면서 색상의 변형을 일으키는 것이 가능



# 과제

1. 실습자료의 'CNN 모델 설계'에서 각 Layer를 거친 후의 shape 주석 달아보고,  
코드 속 #값\_채우기에 들어갈 값 채워보기
2. Pooling 방법 변경 및 Data Augmentation 기법 적용해보며 모델 성능 비교해보기

# REFERENCE

<https://hyen4110.tistory.com/9>

[https://velog.io/@cha-suyeon/Data-augmentation\[https://lcyking.tistory.com/entry/데이터-증강Data-Augmentation\]\(https://lcyking.tistory.com/entry/데이터-증강Data-Augmentation\)](https://velog.io/@cha-suyeon/Data-augmentation[https://lcyking.tistory.com/entry/데이터-증강Data-Augmentation](https://lcyking.tistory.com/entry/데이터-증강Data-Augmentation))

<https://hyen4110.tistory.com/9>

<https://seungbeomdo.tistory.com/40>

<https://rubber-tree.tistory.com/116>

The background is a dark blue gradient. It features several large, overlapping circles in lighter shades of blue. Two prominent white arcs, resembling a stylized rainbow or a wide smile, frame the central text. The top arc is positioned above the 'THANK YOU' text, and the bottom arc is positioned below the 'Deep Session 4차시' text.

# THANK YOU

Deep Session 4차시