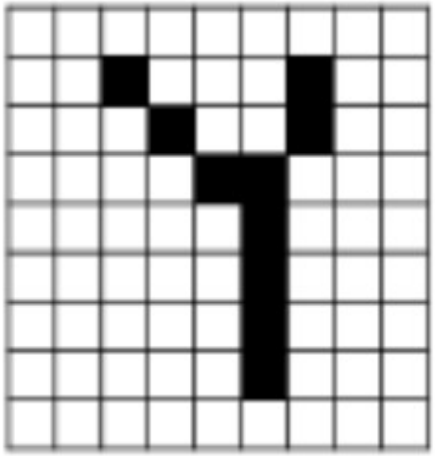
CNN
(CONVOLUTION NEURAL NETWORK)

ABOUT CNN

◆ DNN (MLP) 이미지 인식

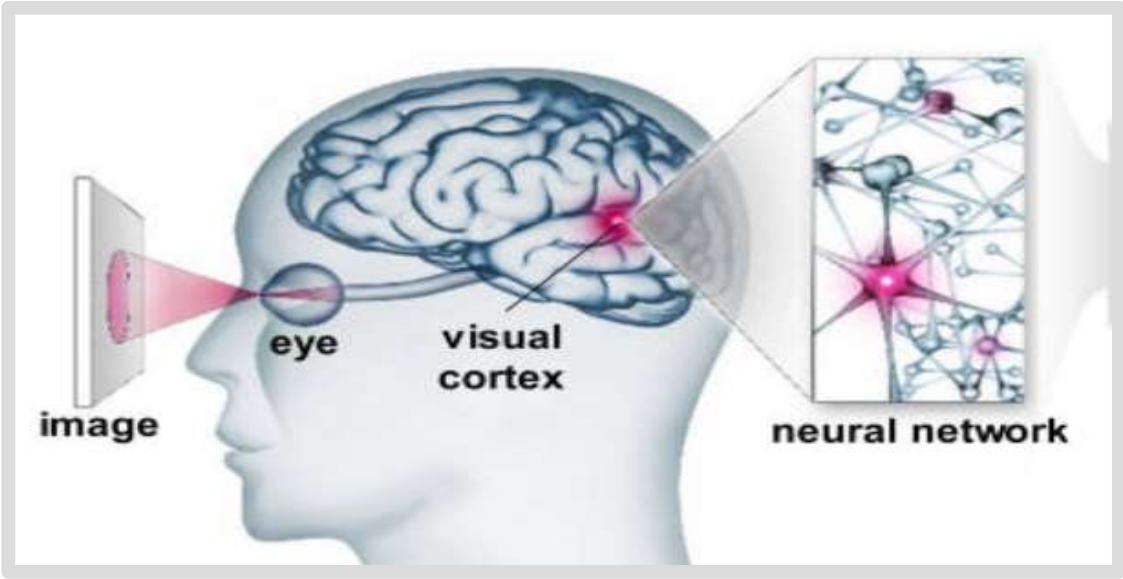


- ✓ 픽셀의 **공간정보 유실**
- ✓ 실제 이미지 데이터 보다 **배경 데이터가 더 많음**
- ✓ **픽셀 수 만큼의 가중치(W)로 연산량 증가**
- ✓ **낮은 인식률**

ABOUT CNN

◆ 사람/동물 시간 인식

- ✓ 동물의 **시각피질**(visual cortex, 視覺皮質) 구조에서 영감
- ✓ **시각 자극**이 **1차 시각피질**을 통해서 처리된 다음, **2차 시각피질** 경유하여, **3차 시각피질** 등 **여러 영역 통과**하며 모여진 정보가 계층적으로 처리되며 추상적인 **특징이 추출**되어 **시각 인식**



ABOUT CNN

◆ CNN 합성곱신경망

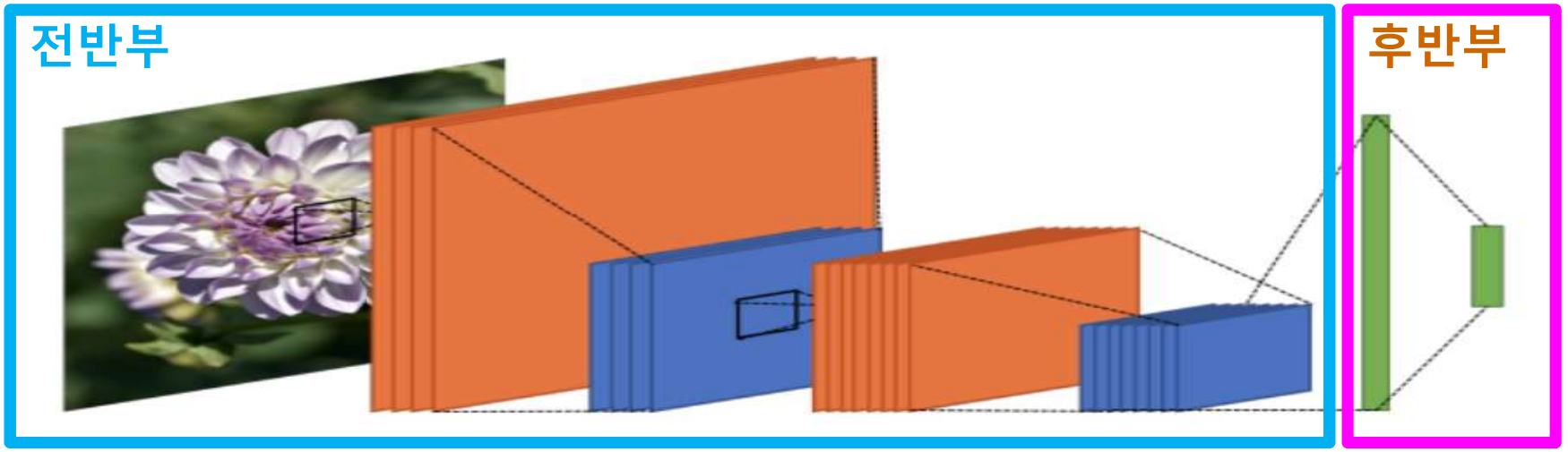
- **DNN의 한 종류로** 컴퓨터 **비전, 시각적 이미지 인식**에 주로 사용
- **텍스트 처리** 등 **여러 다른 분야에도 다양하게 활용**
- **LeNet-5은 1998년** Yann LeCun 교수가 발표한 CNN 알고리즘으로 지속적인 연구와 발전 진행, 특히 2010년 초중반에 많은 발전

LeNet-5 (1998) -> Alexnet (2012) -> GoogLeNet (2014) -> VGGNet (2014) -> Resnet (2015)

ABOUT CNN

◆ CNN 합성곱신경망

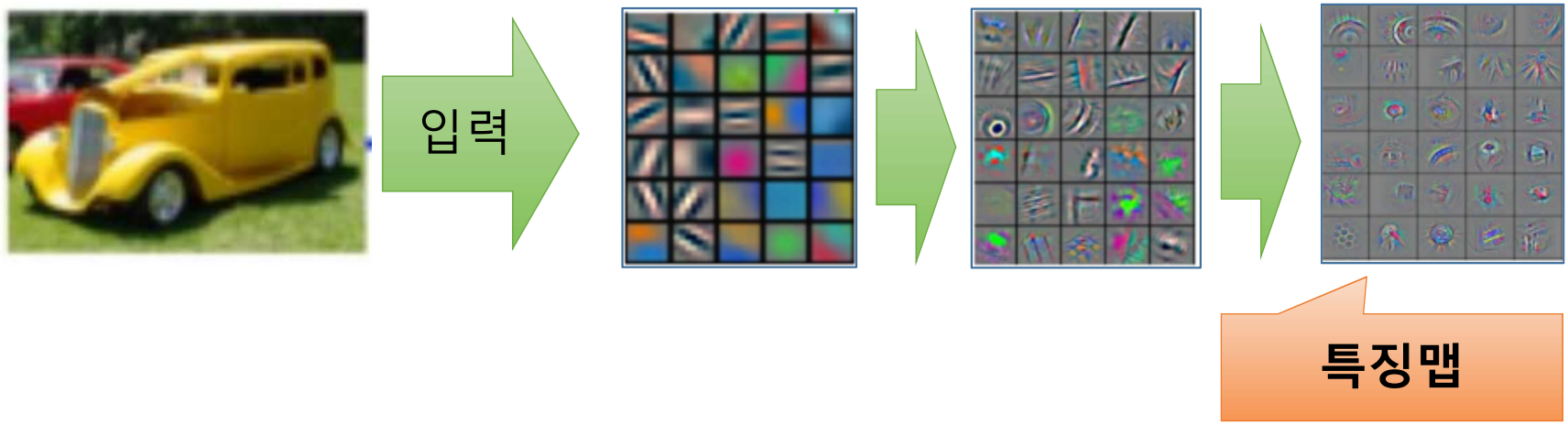
- **입력 데이터의 형상을 유지하며 특징 추출 후 분류**
  - ➢ **전반부 :** 3차원 이미지 입력 받아 특징 추출
  - ➢ **후반부 :** 특징 입력 받아 분류



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- 이미지 위를 일정 간격으로 이동하며 특징(정보)를 하나씩 추출
- 위에서 아래로 전체 이동으로 특징(정보)를 모은 특징맵을 출력하는 기능의 Layer



ABOUT CNN

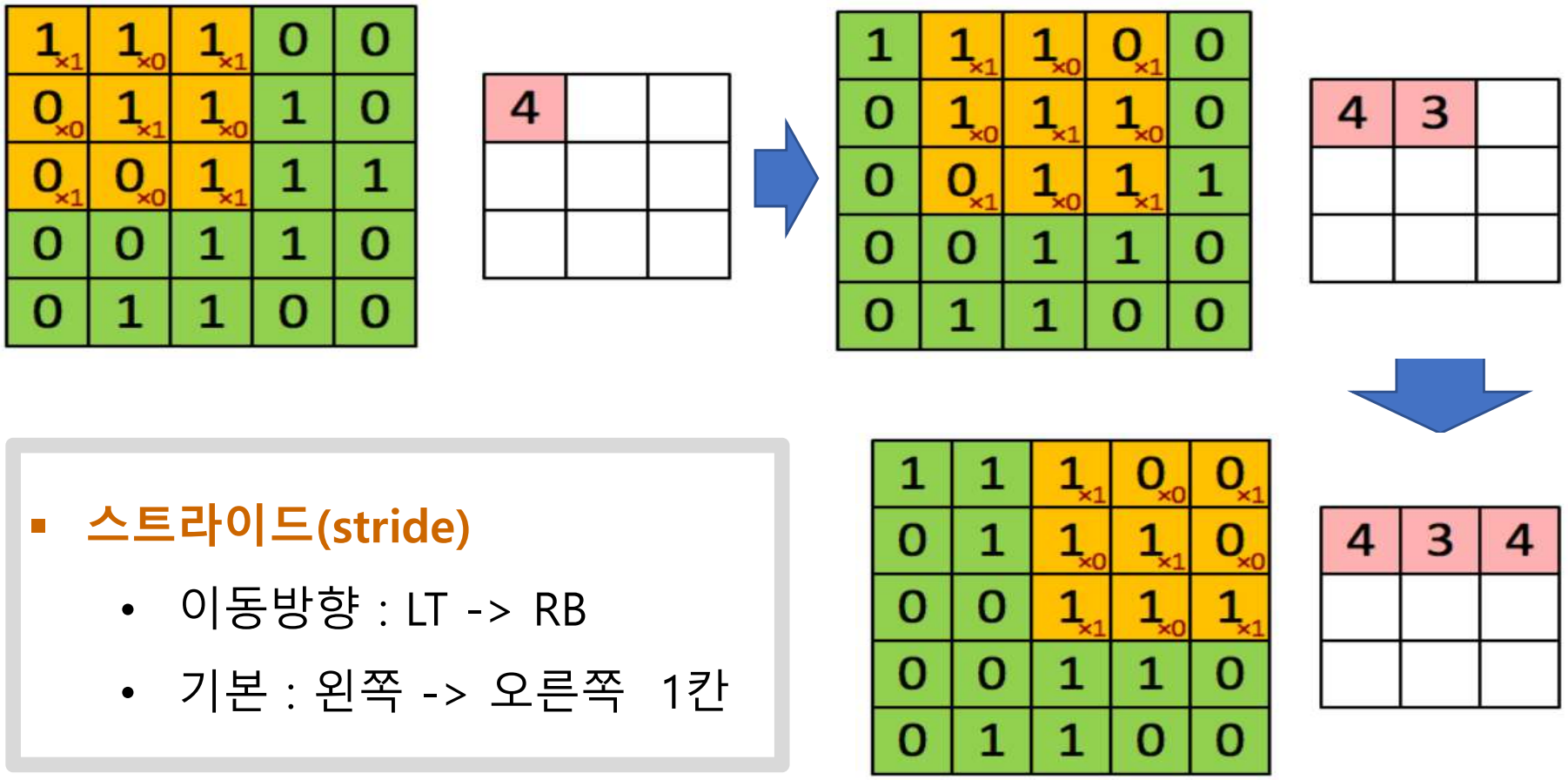
◆ 합성곱층(Conv:Convolution Layer)

- **커널/필터/마스크**
  - 가중치로 구성되며 일반적으로 3x3, 5x5 크기
  - 너무 큰 커널은 특징 추출 부족
  - 이미지 위를 일정 간격 이동

- **스트라이드(stride)**
  - 커널의 이동 방향 및 크기
  - 기본 : 왼쪽 -> 오른쪽 1칸

ABOUT CNN

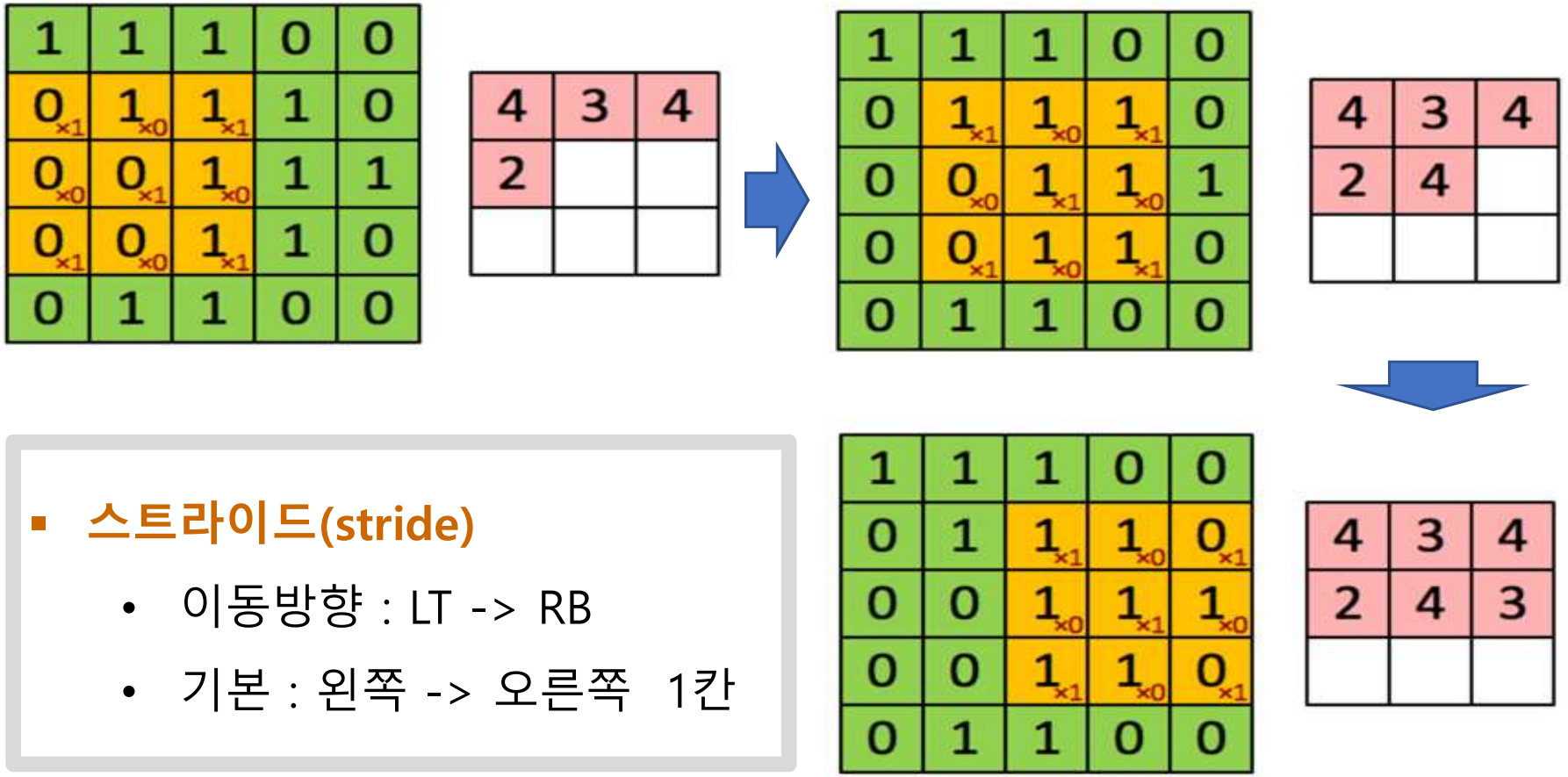
◆ 합성곱층(Conv:Convolution Layer)



- **스트라이드(stride)**
  - 이동방향 : LT -> RB
  - 기본 : 왼쪽 -> 오른쪽 1칸

ABOUT CNN

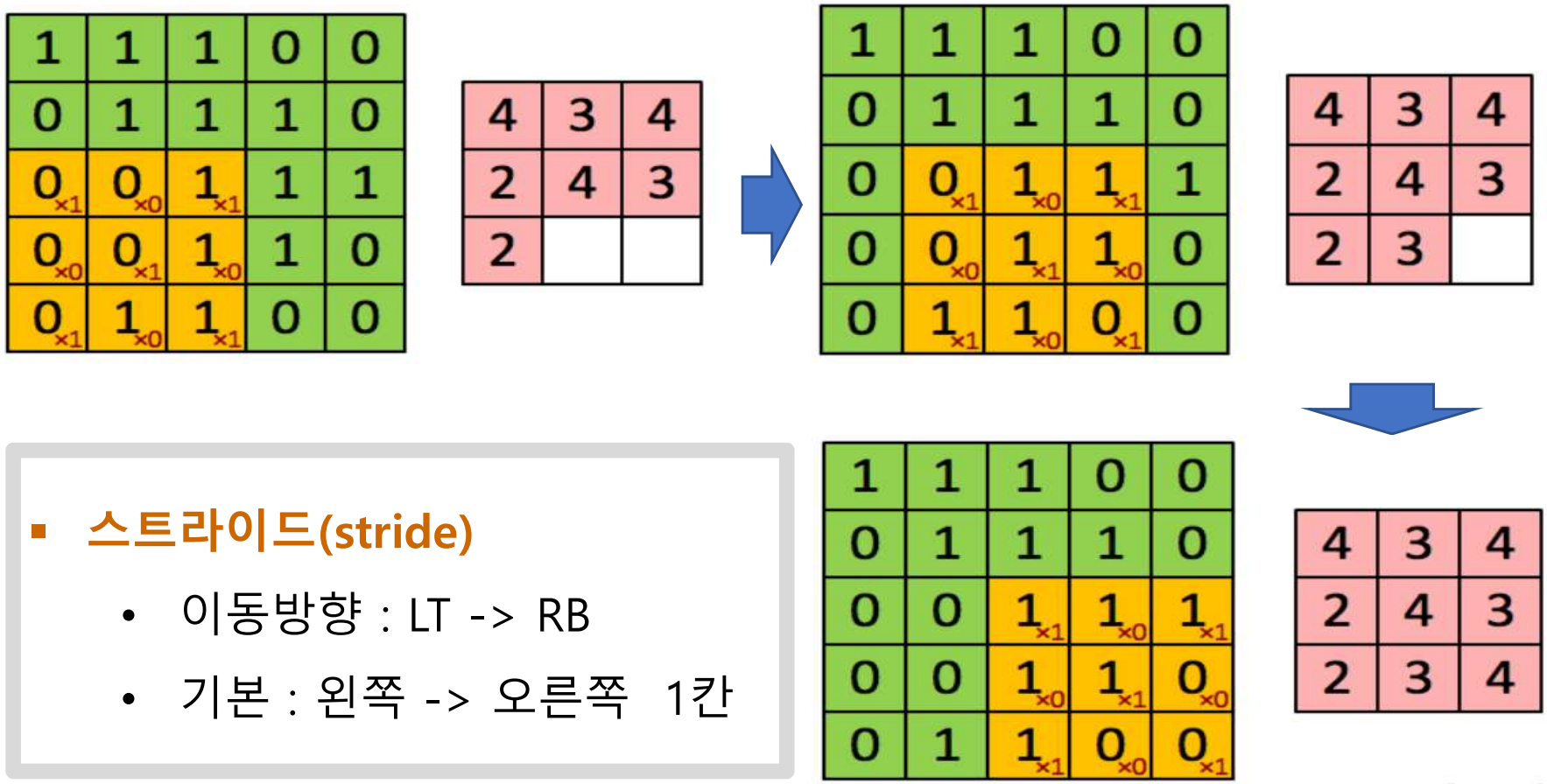
◆ 합성곱층(Conv:Convolution Layer)



- **스트라이드(stride)**
  - 이동방향 : LT -> RB
  - 기본 : 왼쪽 -> 오른쪽 1칸

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)



- **스트라이드(stride)**
  - 이동방향 : LT -> RB
  - 기본 : 왼쪽 -> 오른쪽 1칸

ABOUT CNN

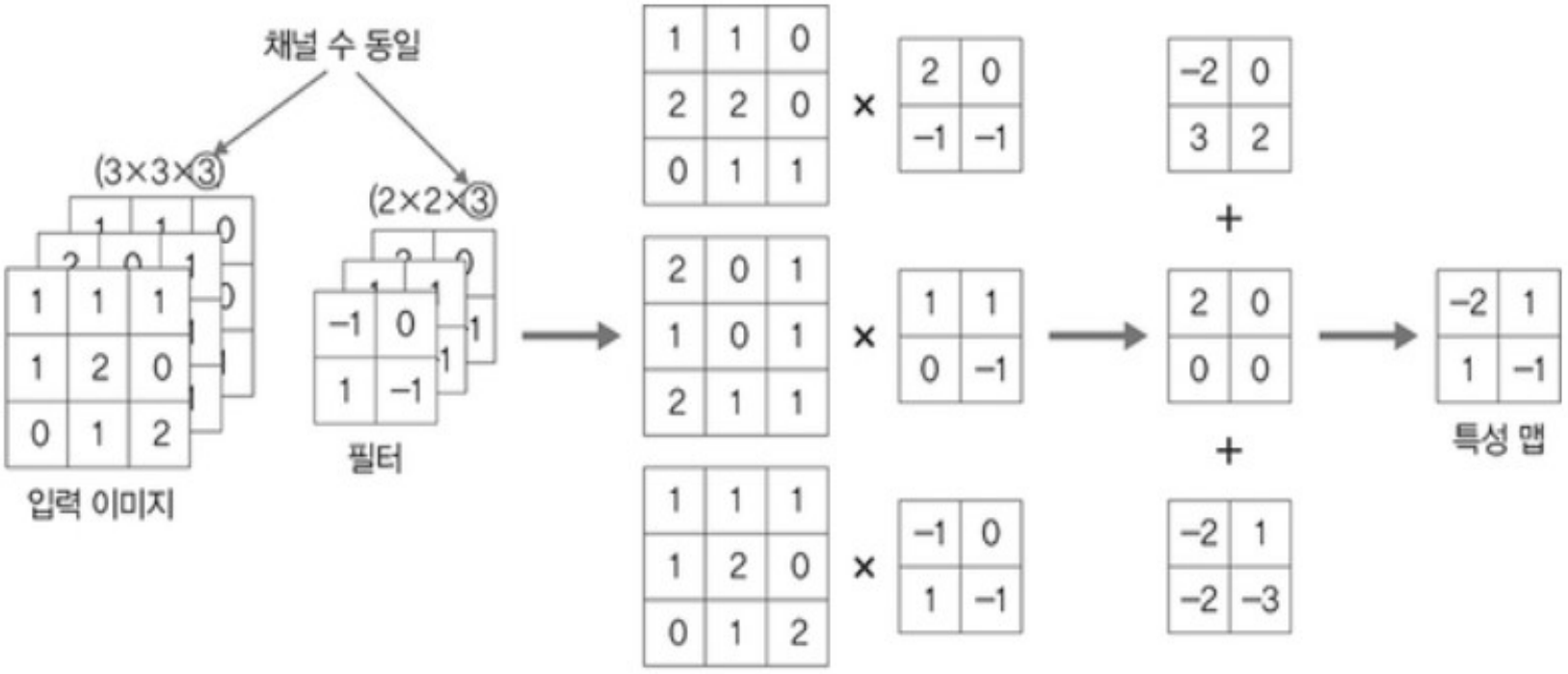
◆ 합성곱층(Conv:Convolution Layer)

- 1채널 그레이스케일 이미지

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- 3채널 컬러 이미지



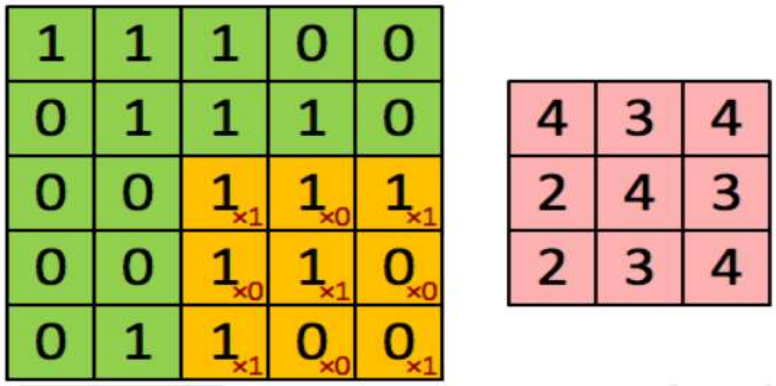
ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- **패딩(Padding)**
  - 커널(필터) 이동 시 좌우상하 모서리 부분 특징 추출 안됨
  - 입력 데이터 사면을 특정값(0)으로 채운 후 합성곱층 진행

➢ **Valid Padding**

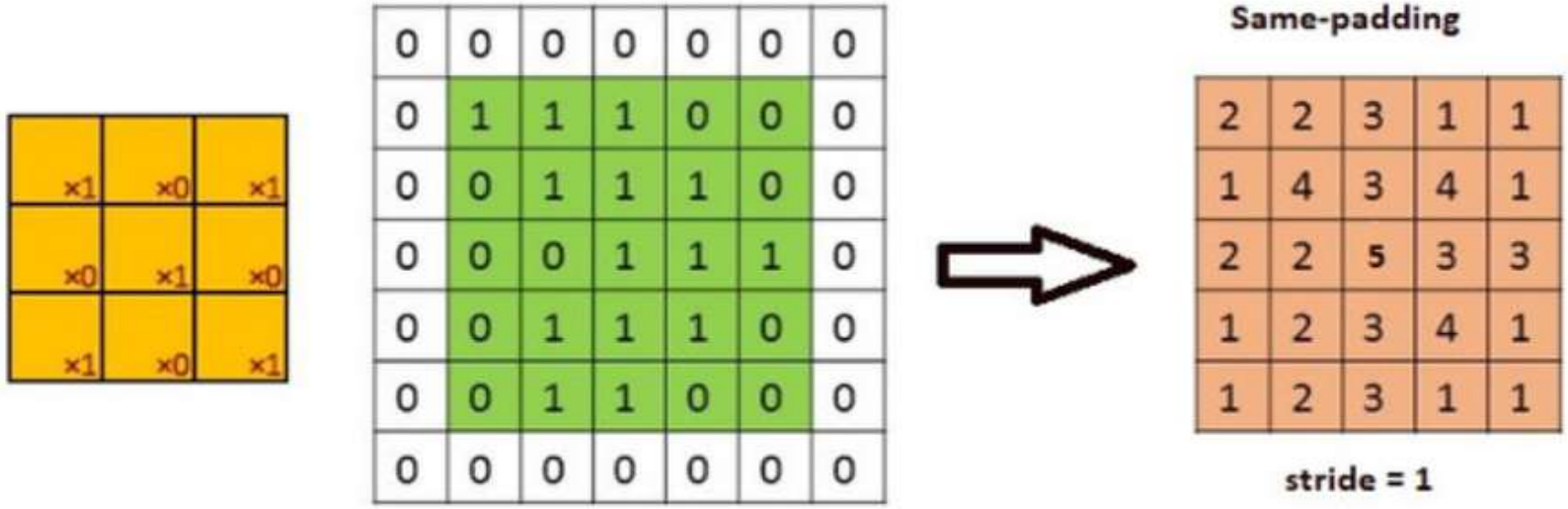
- 입력 데이터와 출력 데이터 크기 다름



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- 패딩(Padding)
  - ➢ Same Padding
    - 입력 크기와 출력 크기 동일
    - 입력 데이터 사면을 특정값(0)으로 채운 후 진행



ABOUT CNN

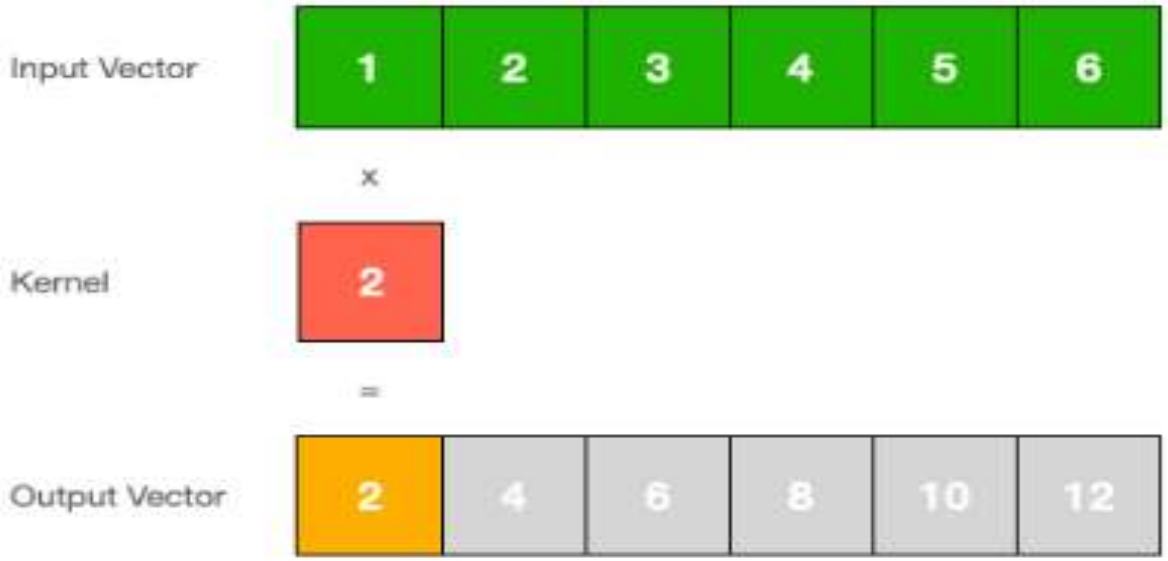
◆ 합성곱층(Conv:Convolution Layer)

- **Conv1D**
  - 커널/필터가 **시간을 축**으로 **좌우로만 이동할 수 있는 합성곱**
  - 데이터 ➔ **시계열데이터**
  - 입　력 ➔ **2D : batch, in_width, in_channels**
  - 출　력 ➔ **2D : batch, fiter_width, out_channels**
  - 사　례 ➔ **문장 분류**, 소리, 신호 평활화

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

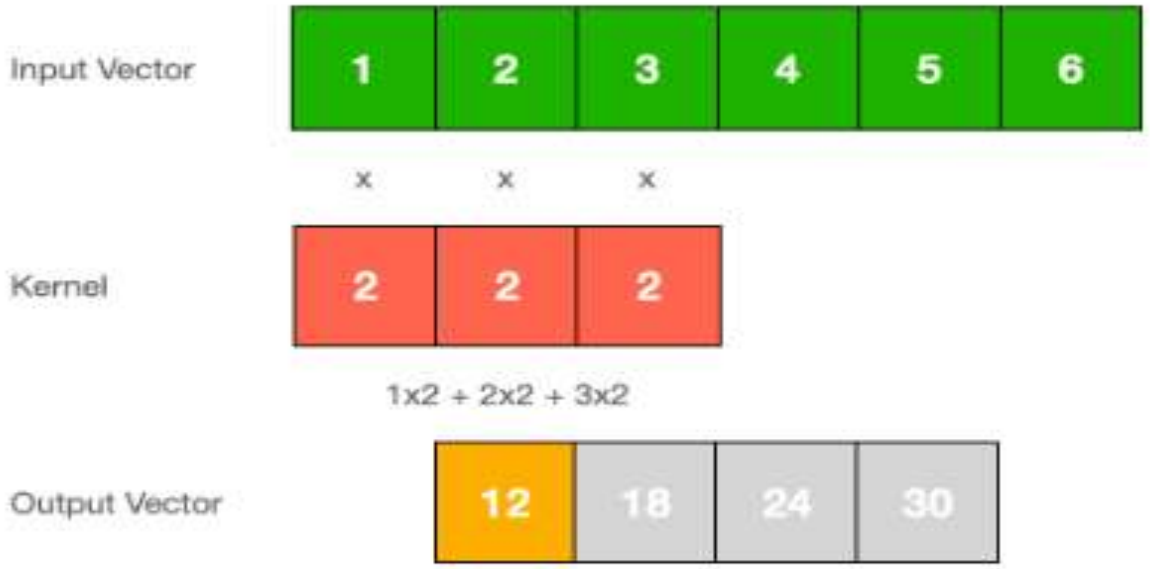
- **Conv1D**



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

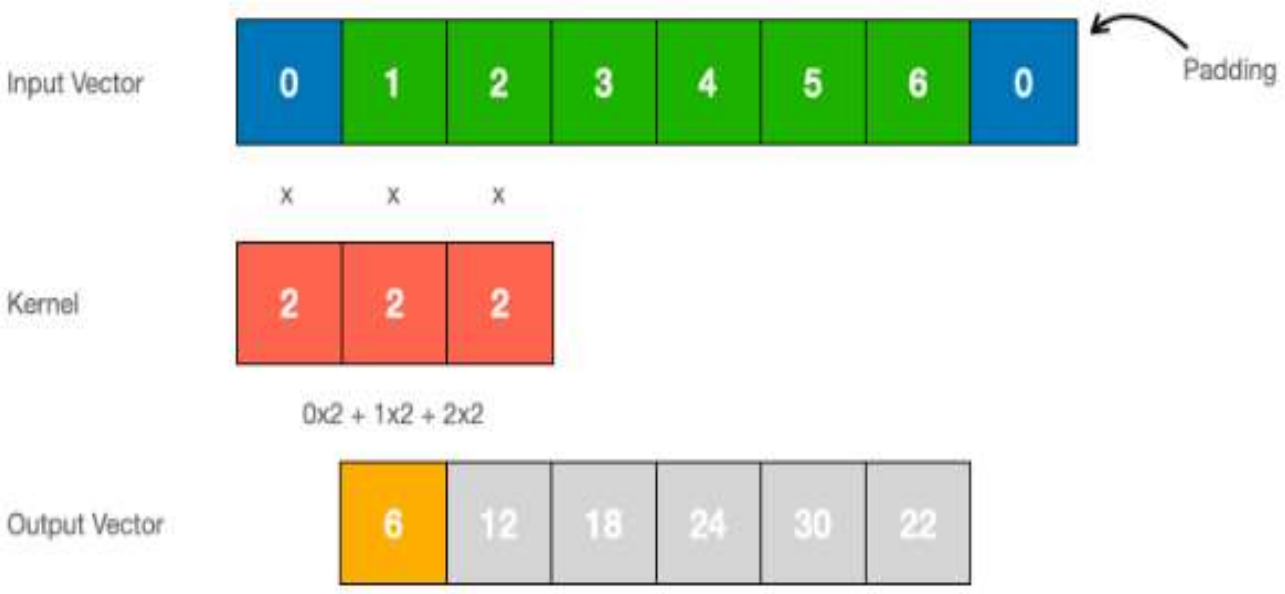
- **Conv1D**



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

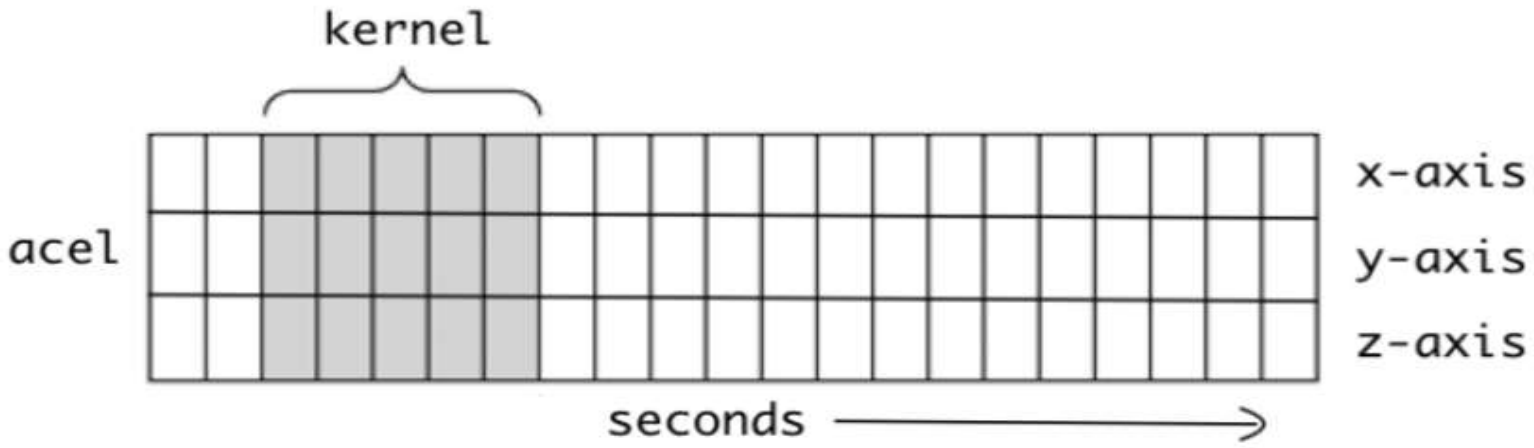
- **Conv1D**



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- **Conv1D**
  - 예) 가속도계 센싱 데이터



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- **Conv1D**

```
# 시간의 흐름에 따른 데이터 셋
X = np.array([[10,20,30],[20,30,40],[30,40,50],[40,50,60]])
y = np.array([40,50, 60, 70])

# 입력 데이터의 shape
print(f'X => {X.shape}, y =>{y.shape}')
```

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

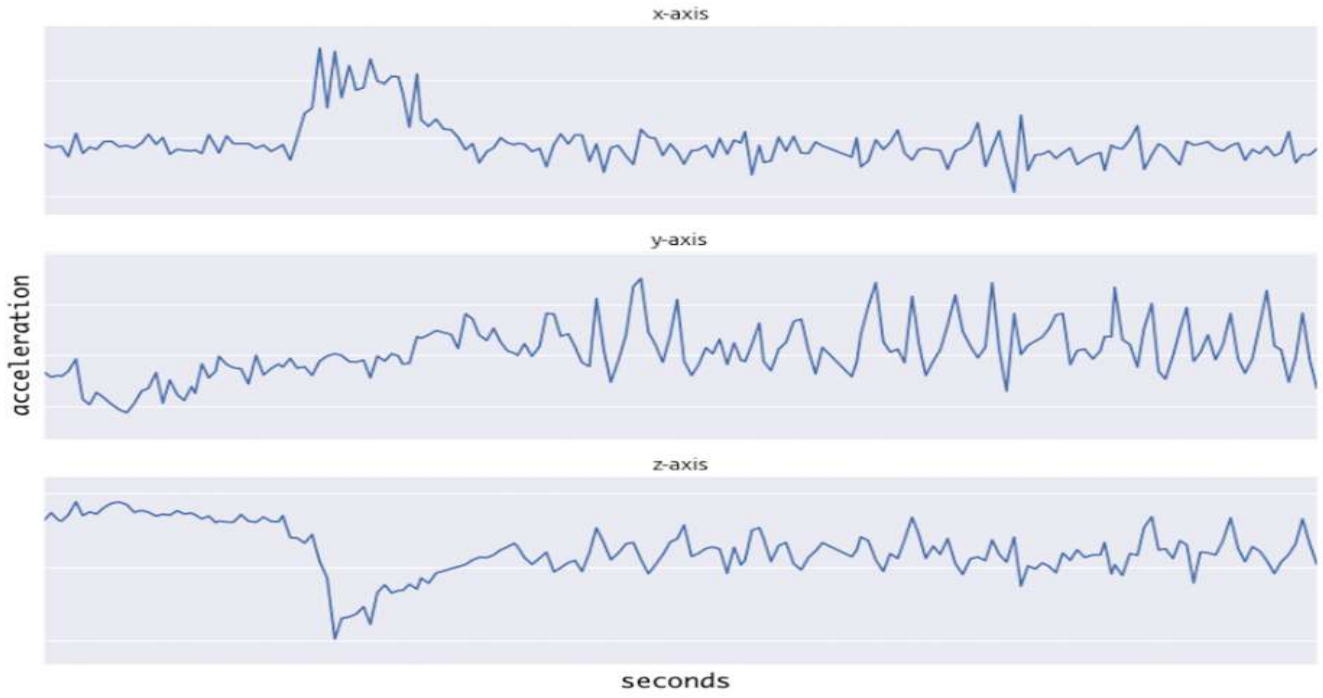
- **Conv1D**

```
model = Sequential()

# 전반부 => 특징맵 추출
model.add(Conv1D(10, kernel_size=2, input_shape = (3,1)))
model.add(MaxPooling1D())
model.add(Flatten())

# 후반부 ==> 전결합
model.add(Dense(50, activation='relu'))
model.add(Dense(1))
model.summary()
```

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

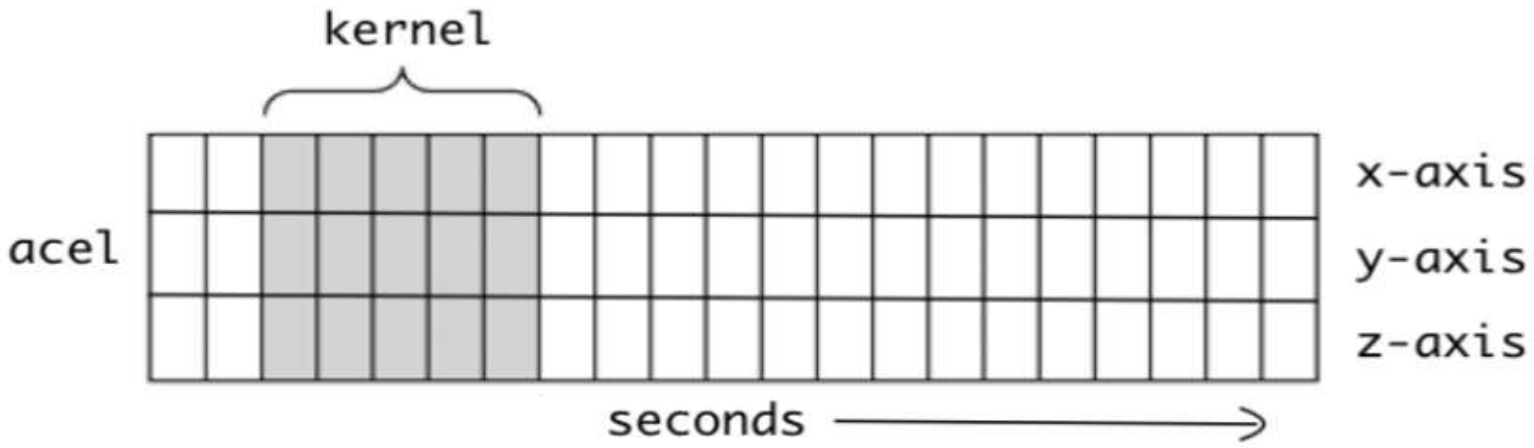
- **Conv1D**
  - 예) 가속도계 센싱 데이터



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- **Conv1D**
  - 예) 가속도계 센싱 데이터기반 행동분석



ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- **Conv1D**
  - 예) 가속도계 센싱 데이터기반 행동분석

```
model = Sequential()

# 전반부 ==> 특징맵/패턴인식
model.add(Conv1D(1, kernel_size=5, input_shape = (120, 3)))
model.add(MaxPooling1D())
model.add(Flatten())

# 후반부 ==> 전결합
model.add(Dense(50, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

model.summary()
```

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

- **Conv2D**
  - 커널/필터가 **두 개 방향으로 이동할 수 있는 합성곱**
  - 입　력 ➔ **3D : batch, in_height, in_width, in_channels**
  - 출　력 ➔ **3D : batch, filter_height, filter_width, out_channels**
  - 사　례 ➔ **이미지**

ABOUT CNN

◆ 합성곱층(Conv:Convolution Layer)

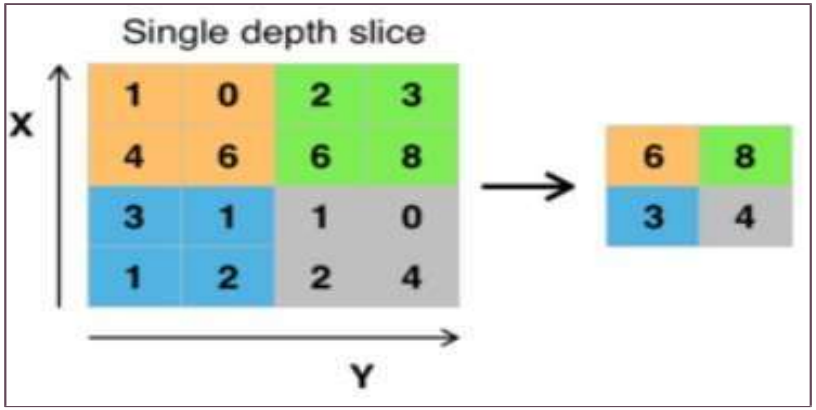
- **Conv3D**
  - 커널/필터가 **세 개 방향으로 이동할 수 있는 합성곱**
  - 입　력 ➔ **4D : batch, in_depth, in_height, in_width, in_channels**
  - 출　력 ➔ **4D : batch, filter_depth, filter_height, filter_width, out_channels**
  - 사　례 ➔ **3D 이미지 (MRI, CT 등..) 또는 비디오**

ABOUT CNN

◆ 풀링층(Polling Layer)

- **합성곱 층(합성곱 연산 + 활성화 함수) 다음**에 풀링 층 추가
- **특성 맵을 다운샘플링**하여 **특성 맵의 크기 줄**이는 **풀링 연산** 진행
- 합성곱층과 달리 **커널이 중첩되지 않음**
- 커널 크기 : 2x2
- 종류 ➔ 최대 풀링(max pooling), 평균 풀링(average pooling)

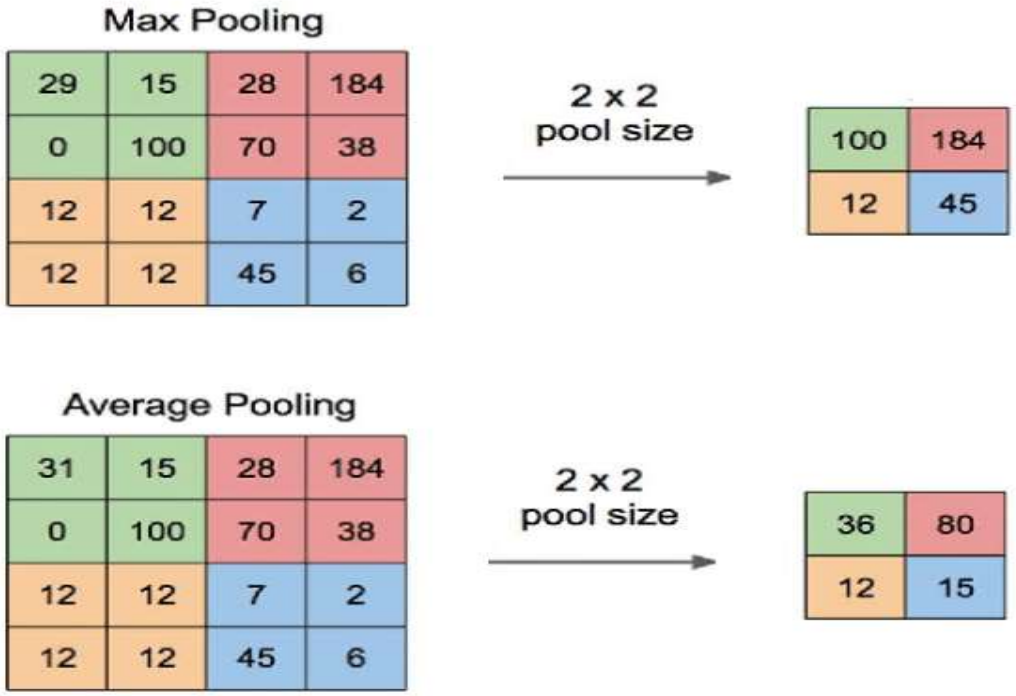
[최대 폴링 연산]



ABOUT CNN

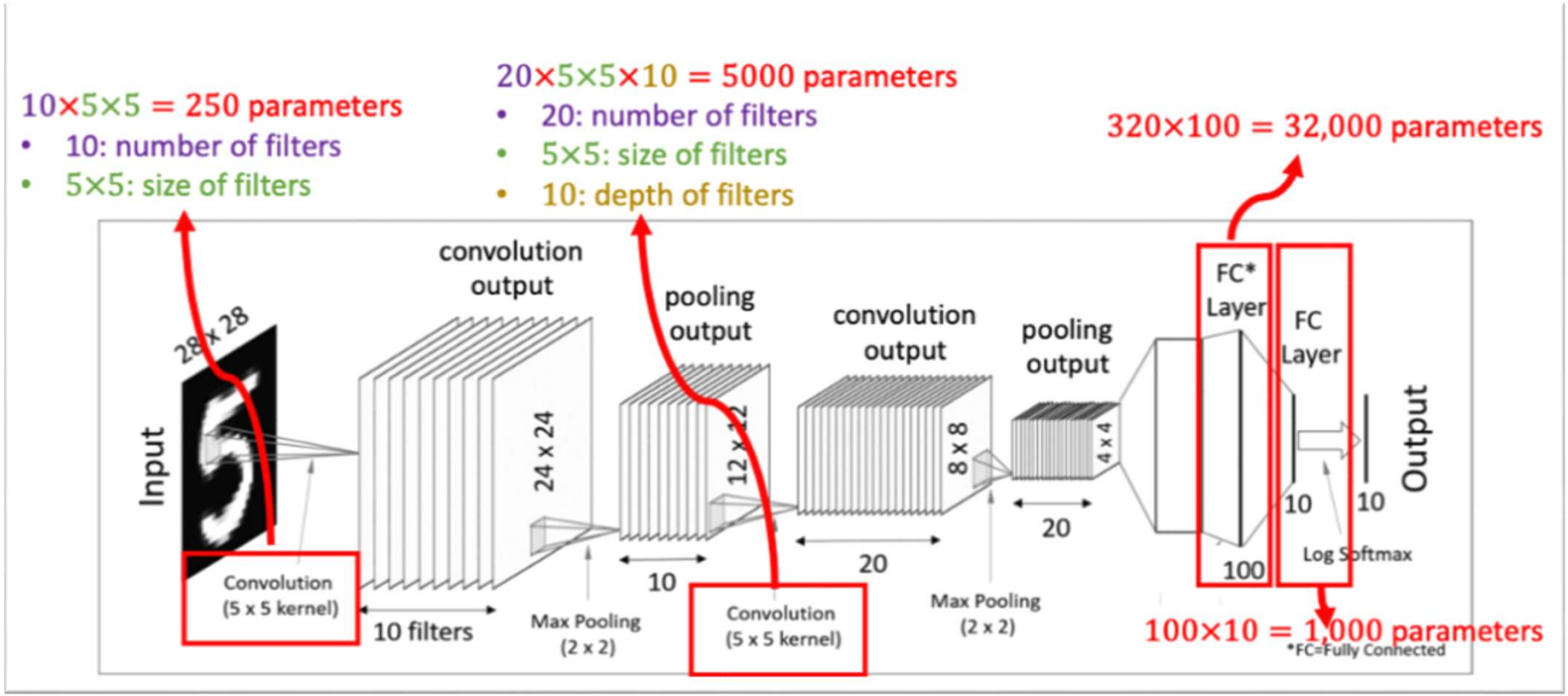
◆ 풀링층(Polling Layer)

- 통상적으로 (2,2) 크기 / 스트파이드 역시 (2,2)
- 겹치는 부분 없이 연산



ABOUT CNN

◆ CNN FLOW



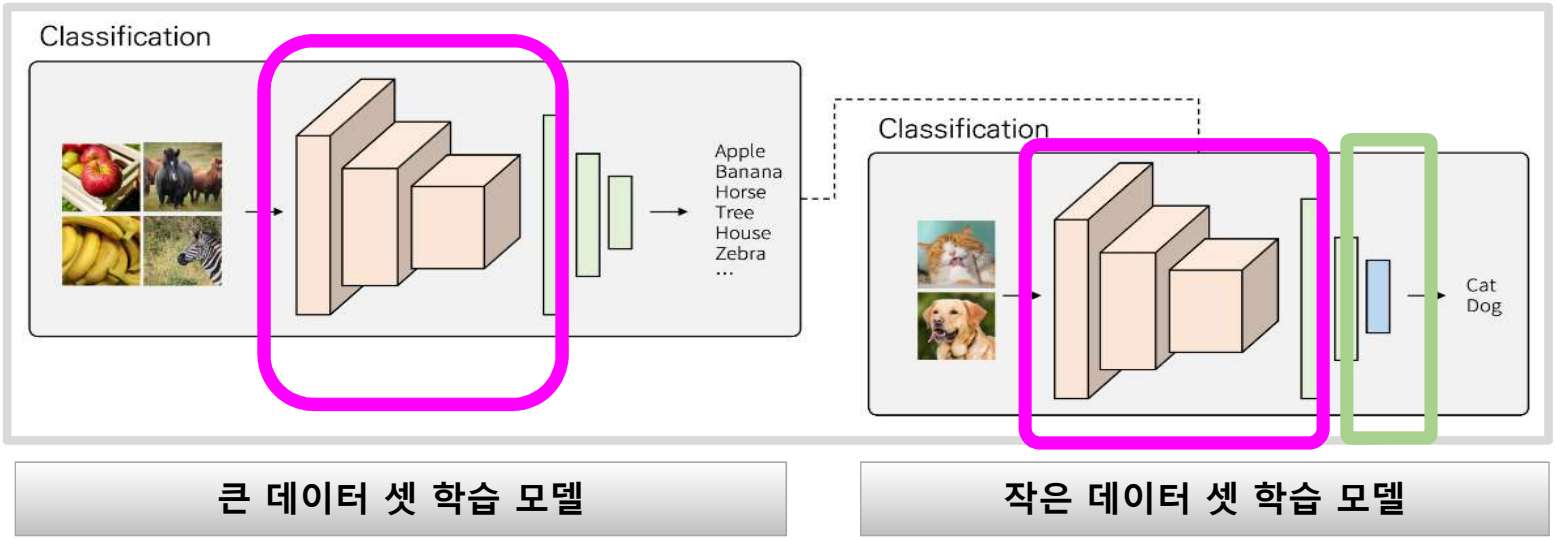
PART Ⅱ

TRANSFER LEARNING

TRANSFER LEARNING

◆ 전이학습

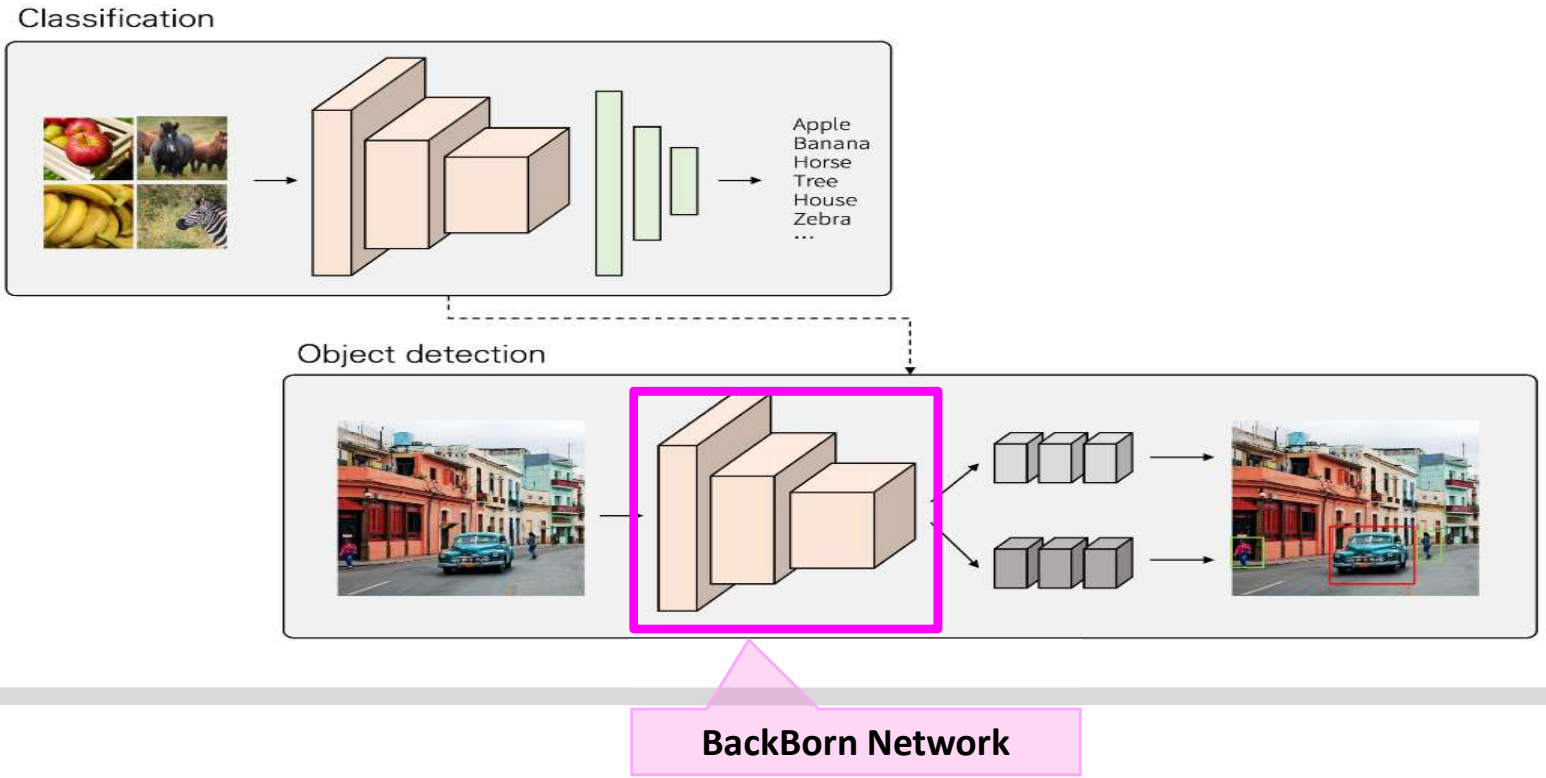
❖특정 대상을 예측하도록 **훈련된 모델을 다른 대상에 사용**하는 것



TRANSFER LEARNING

◆ 전이학습

❖훈련된 모델을 다른 작업(Task)에 적용하는 학습



TRANSFER LEARNING

◆ 전이학습

❖ Backborn Network

- 다양한 네트워크를 상호 연결하는 컴퓨터 네트워크의 일부
- 입력 이미지를 feature map으로 변형시켜주는 부분
- 대표 : VGG16, ResNet-50

❖ Nect

- Backbone과 Head를 연결
- feature map을 refinement(정제), reconfiguration(재구성)
- 대표 : FPN, PAN, BiFPN, NAS-FPN

❖ Head

- Backbone에서 추출한 feature map의 location 작업 수행
- 대표 : [1-Stage] YOLO, SSD [2-Stage] Faster R-CNN, R-FCN

TRANSFER LEARNING

◆ 전이학습

❖ **사용 이유 및 장점**

➢ **학습 빠르게 수행**

- 이미 입력되는 데이터에 대해 특징을 효율적으로 추출
- 학습할 데이터에 대해 특징 추출하기 위한 별도 학습 필요 없음

➢ **작은 데이터셋 학습 시 오버피팅 예방**

- 전이 학습을 이용해 마지막 레이어만 학습하여, 학습할 가중치 수가 줄어 과한 학습이 이루어지지 않게 할 수 있음

TRANSFER LEARNING

◆ 전이학습

❖ **사전 학습(Pre-training)**

- 학습된 모델을 만드는 과정
  - gft

❖ **미세 조정(Fine-training)**

- 사전 학습된 모델에 새로운 문제 적용하기 위해 **일부 가중치 조절하는 학습 과정**
- 전이 학습에 사용되는 기법 중 하나

TRANSFER LEARNING

◆ 전이학습

❖ 조건

➢ 사전 학습에 사용한 데이터와 새로운 데이터가 비슷한 형태

- 이미 입력되는 데이터에 대해 특징을 효율적으로 추출
- 학습할 데이터에 대해 특징 추출하기 위한 별도 학습 필요 없음

➢ 새로운 데이터보다 많은 데이터로 사전 학습 수행되었 어야 함

- 전이 학습을 이용해 마지막 레이어만 학습하여, 학습할 가중치 수가 줄어 과한 학습이 이루어지지 않게 할 수 있음

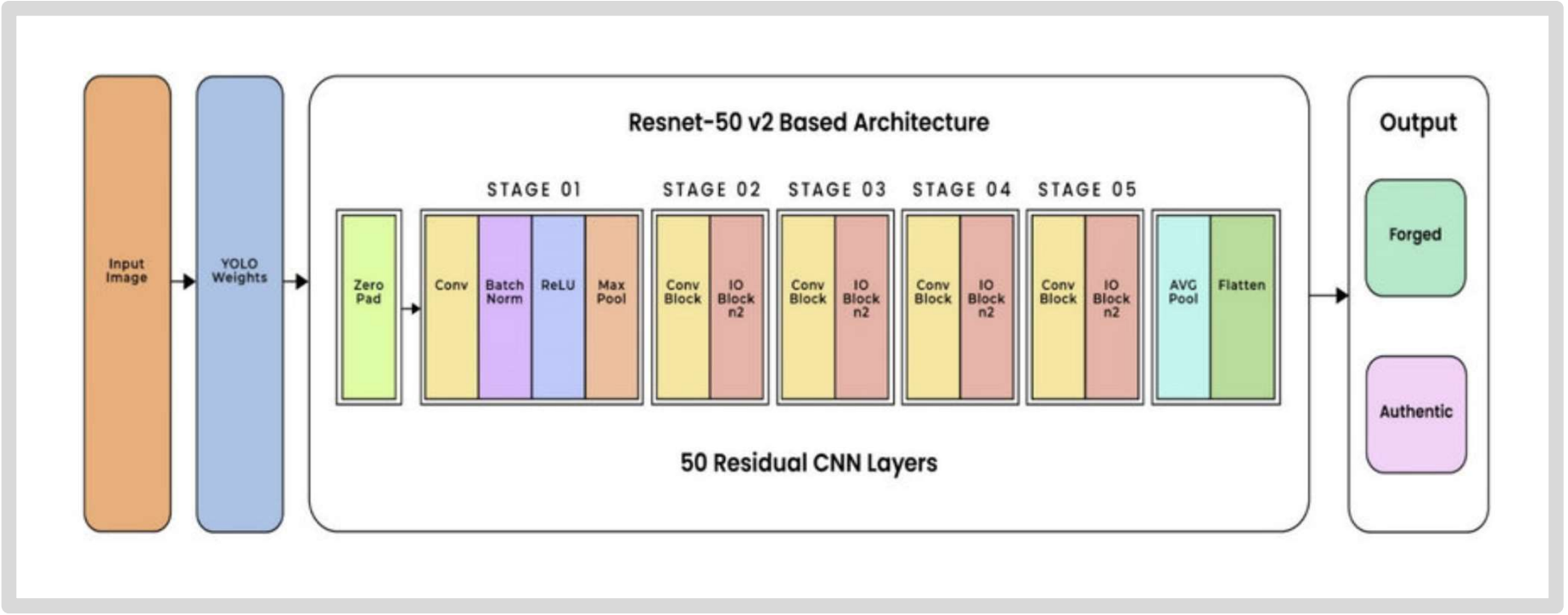
TRANSFER LEARNING

◆ Backborn Resnet-50v2

- 50개 계층으로 구성된 컨벌루션 신경망
- ResNet에서는 모델이 깊어질수록 최적화에서 멀어지는 것에 주목
- 레이어가 깊어질수록 곱해지는 미분값들이 증가함에 다라 초기값에 집중하지 못해서 CNN 자체의 성능이 떨어지게 됨 ➔ Residual Connection 기법으로 해결
- Residual Connection 란?
  - layer와 layer 사이 건너뛰는 일종의 지름길 해당하는 연결

TRANSFER LEARNING

◆ Backborn Resnet-50v2



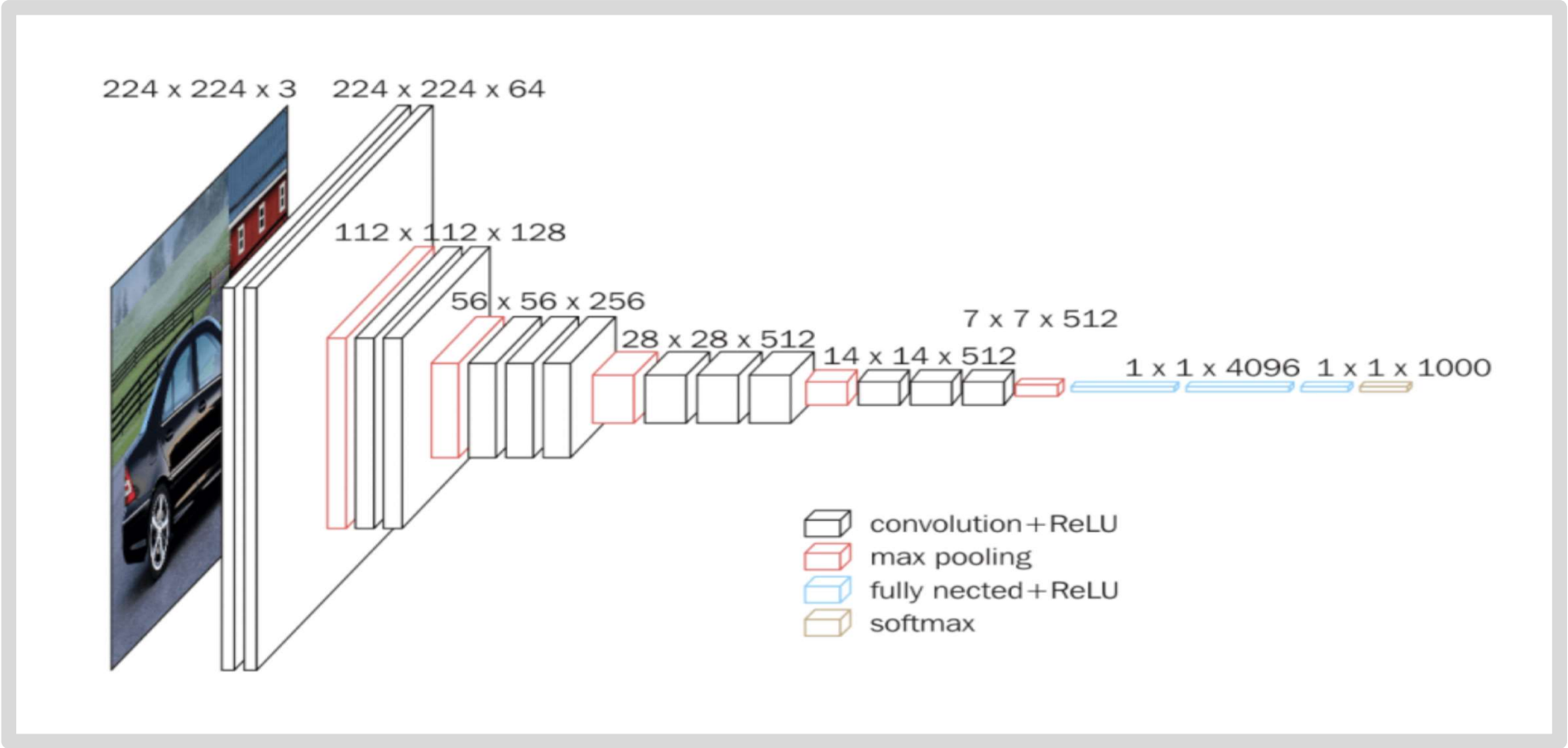
TRANSFER LEARNING

◆ Backborn VGG16

- VGGNet-N(Very Deep Convolutional network for large-scale image recognition)
- 옥스포드 대학의 연구팀 VGG에 의해 개발된 모델
- **2014년 이미지넷 이미지 인식 대회**에서 **준우승**을 한 모델
- **16개 또는 19개의 Layer**로 구성된 **모델**
- 사용하기 쉬운 구조와 좋은 성능 덕분에 우승 거둔 GoogLeNet(22Layer)보다 더 인기

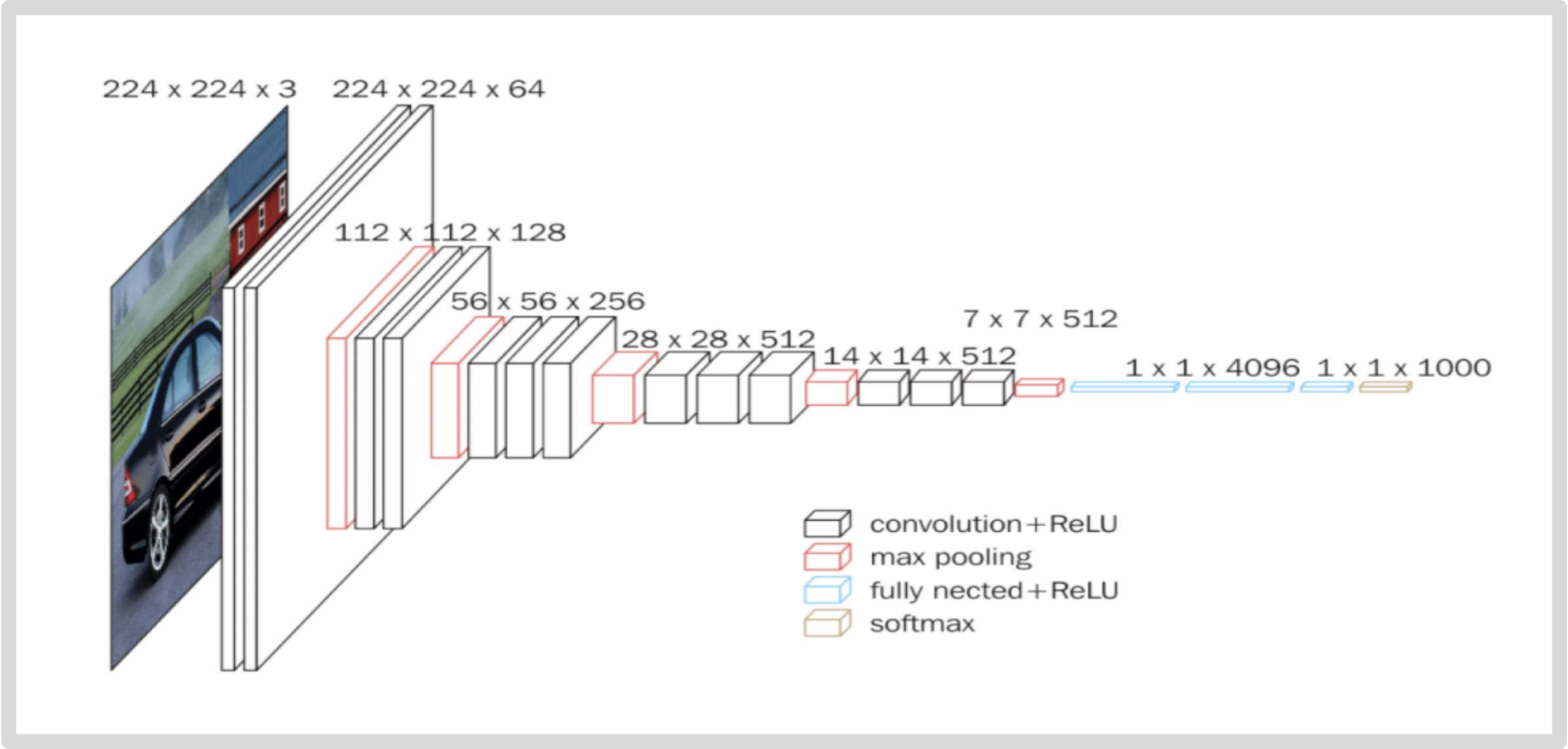
TRANSFER LEARNING

◆ Backborn VGG16



TRANSFER LEARNING

◆ Backborn VGG16



PART Ⅱ

GNN

GENERATIVE ADVERSARIAL NETWORK

ABOUT GAN

◆ GAN 생산적적대신경망

- 생성적 적대 신경망(Generative Adversarial Networks) 약자
- **2016년 구글에서 발표한 모델**
- 현재 개발되고 있는 **GAN 구조의 기초가 되는 구조**
- [초기]fully-connected 생성모델/ 분류모델

  [대체]convolution으로 대체 구성 ➔ 성능과 안전성 향상

  페이스북 AI 연구 팀이 만들어 발표한 DCGAN(Deep Convolutional GAN)

- **fractional strided convolution**
  - input 사이에 padding 더하고 convolution 시행 ➔ 크기 키움
  - 생성모델은 **고차원의 latent vector 입력받아 저차원 이미지 반환**

ABOUT GAN

◆ GAN 생산적적대신경망

- **생성자와 식별자가 서로 경쟁하며 성능 개선하여 데이터 생성**
- **비지도학습**
- 이전에는 없던 **새로운 데이터를 생성하는 모델**

- 목표
  - **판별자가 실제 데이터와 가짜 데이터를 구별 못하게 하는 것**
- 구성
  - **Generator(생성자) + Discriminator(판별자)**

ABOUT GAN

◆ GAN 생산적적대신경망

- **경찰과 위조지폐범 사이의 게임**
  - 위조지폐범은 진짜 같은 화폐를 만들어 경찰을 속이기 위해 노력하고, 경찰은 위조지폐를 잘 감별하기 위해 노력

  - **생성모델(generator)**

    ➔ 최대한 진짜 같은 데이터를 만들기 위한 학습 진행

  - **분류모델(discriminator)**

    ➔ 진짜와 가짜를 판별하기 위한 학습 진행

ABOUT GAN

◆ GAN 생산적적대신경망

❖ 학습 과정

① 분류모델/판별모델 먼저 학습

- 진짜 데이터를 진짜로 분류하도록 학습

② 생성모델 학습

- 가짜 데이터 생성하도록 학습

③ 분류모델/판별모델 가짜 학습

- 생성모델이 생성한 데이터를 가짜로 분류하도록 학습

④ 학습된 분류모델을 속이는 방향으로 생성모델을 학습

ABOUT GAN

◆ GAN 생산적적대신경망

- **생성모델(generator)**
  - 노이즈 입력 받아 다수의 층 통과하면서 **특징 맵 확장**
  - **마지막 층** 통과해서 나오는 특징 맵은 **이미지 크기와 같음**
  - 옵티마이저(optimizer) 사용하는 **최적화 과정, 컴파일 과정 없음**
  - 풀링(pooling) 과정이 없고 배운 **패딩(padding) 과정 포함**
  - 입력 크기와 출력 크기를 똑같이 맞추기 위해서
  - **배치 정규화**(Batch Normalization)라는 과정
    값을 일정하게 재배치 하여 층의 개수가 늘어나도 안정적인 학습 진행
  - 활성화 함수 ➔ ReLU() 함수, 판별자로 넘겨주기 직전에는 tanh() 함수
    출력되는 값을 -1~1 사이로 맞춤

ABOUT GAN

◆ GAN 생산적적대신경망

- **판별/분류모델(discriminator)**
  - **특징맵 크기 축소**해 나가는 구조
  - 전통적인 인공신경망의 구조
  - 가짜인지 진짜인지를 판별만 해 줄 뿐, 자기 자신 학습 안됨!
  - 판별자가 얻은 가중치

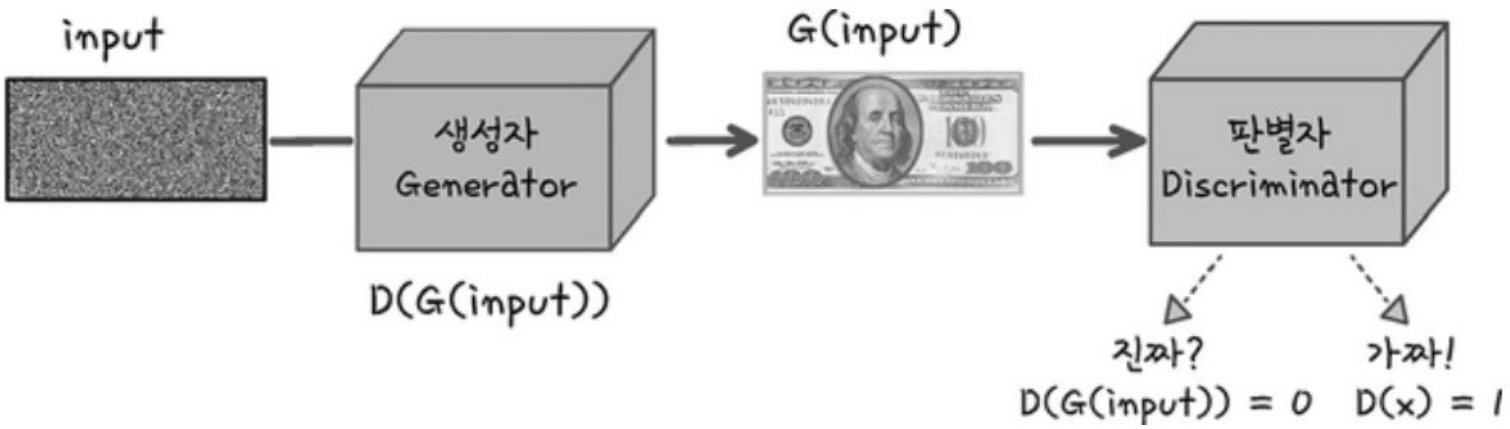
    ➔ 판별자 자신이 학습하는 데 쓰이는 것이 아니라

    생성자로 넘겨주어 생성자가 업데이트된 이미지를 만들도록 해야 함

ABOUT GAN

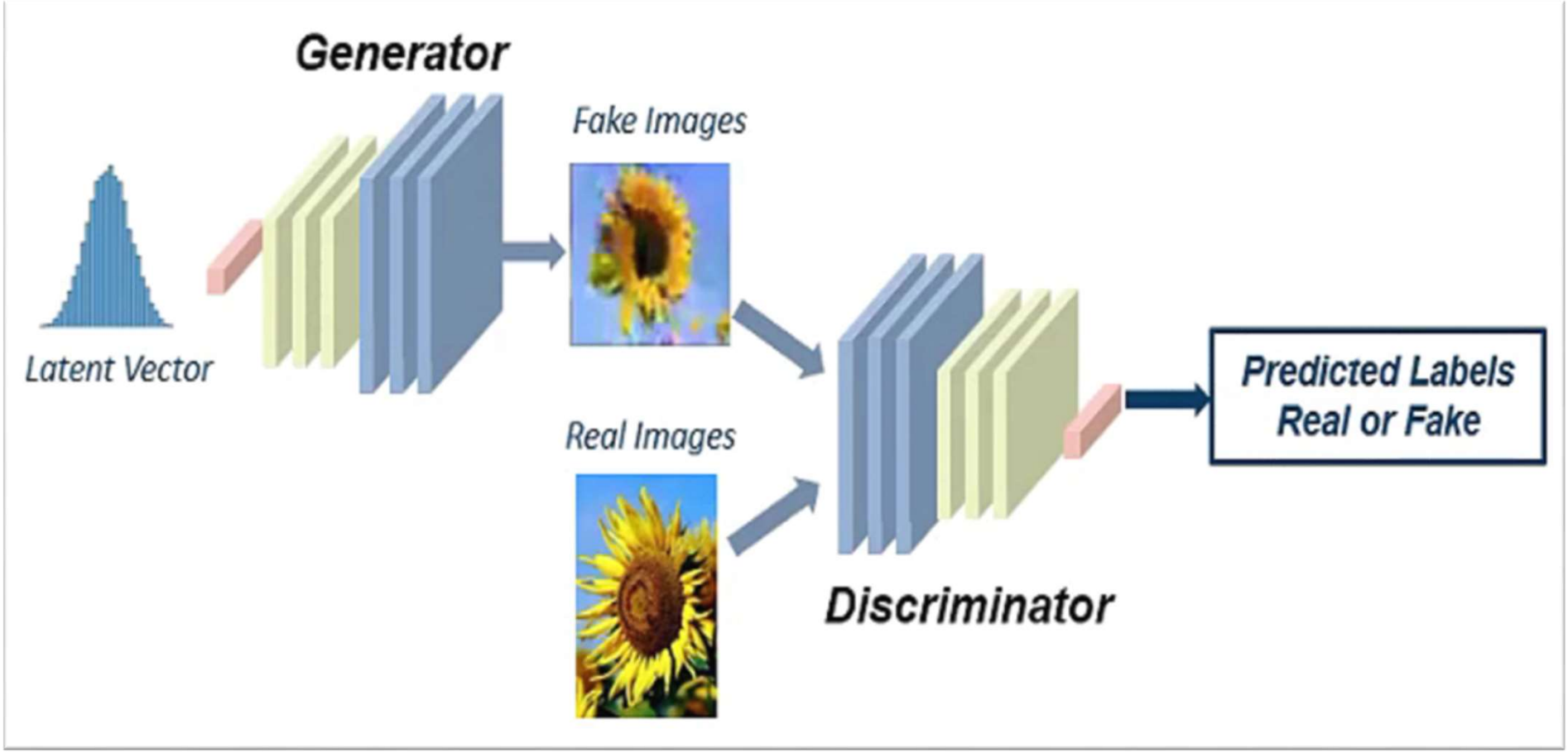
◆ GAN 생산적적대신경망

- **FLOW**



ABOUT GAN

◆ GAN 생산적적대신경망



ABOUT GAN

◆ AE(Auto-Encoder)

- 딥러닝 이용해 가상 이미지를 만드는 또 하나의 유명한 알고리즘
- **입력 데이터를 최대한 compression 후, compressed data 다시 입력 형태로 복원 시키는 신경망으로 출력 결과 알수 없음**
- GAN과 비슷한 결과를 만들지만, 다른 특징
  - ➔ GAN - 세상에 존재하지 않는 완전한 가상 이미지 생성
  - ➔ AE - **입력 데이터의 특징을 효율적으로 담아낸 이미지 생성**

- **정확한 데이터 아니어도 가능**
- **부족한 학습 데이터 수를 효과적으로 늘려 주는 효과**
- 사용 : 영상 의학 분야 등 아직 데이터 수가 충분하지 않은 분야

ABOUT GAN

◆ AE(Auto-Encoder)

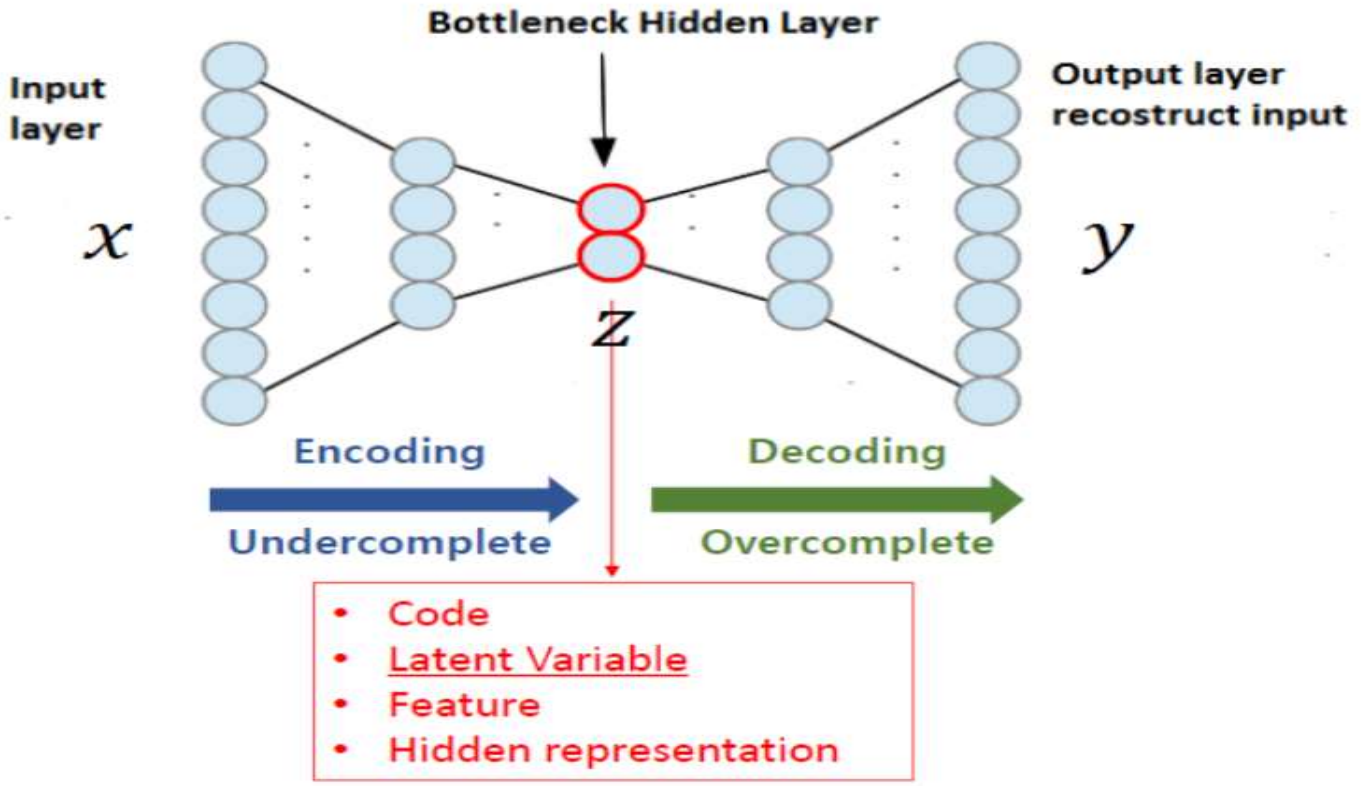
- GAN : 진짜 같아 보여도 실제로는 존재하지 않는 완전한 가상 이미지
- AE : 초점 흐릿, 윤곽 불명확, 사람의 특징 유추할 수 있는 것들 모여 이미지 생성

[GAN - AE]

ABOUT GAN

◆ AE(Auto-Encoder)

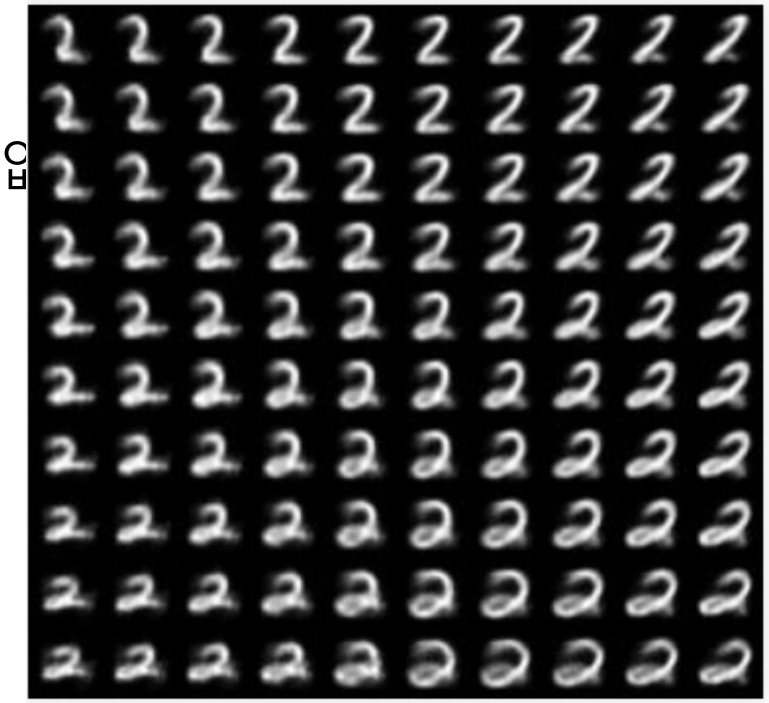
- 구성 : Encoder + Laten Code + Decder



ABOUT GAN

◆ AE(Auto-Encoder)

- 결과 시각화 이미지
  - y축 **두께** 변화
  - x축 **기울기** 변화
  - 출력 : latent vector 미리 알 수 없



ABOUT GAN

◆ AE(Auto-Encoder)

- **인코더 (Encoder) : 입력층 ➔ 은닉층**
  - 입력층보다 적은 수의 노드를 가진 은닉층 ➔ **차원 축소**
  - **소실된 데이터 복원하기 위해 학습** 시작
  - 과정 통해 **입력 데이터 특징을 효율적으로 응축한 새로운 출력이 나오는 원리**
  - 입력을 내부 표현으로 바꿈(인지 네트워크, recognition network)

ABOUT GAN

◆ AE(Auto-Encoder)

- 인코더 (Encoder) : 입력층 ➔ 은닉층

```
autoencoder.add(Conv2D(16, kernel_size=3, padding='same',
                       input_shape=(28,28,1), activation='relu'))

autoencoder.add(MaxPooling2D(pool_size=2, padding='same'))
autoencoder.add(Conv2D(8,  kernel_size=3, activation='relu',
                       padding='same'))

autoencoder.add(MaxPooling2D(pool_size=2, padding='same'))
autoencoder.add(Conv2D(8,  kernel_size=3, strides=2,
                       padding='same', activation='relu'))
```

ABOUT GAN

◆ AE(Auto-Encoder)

- **디코더 (Decoder) : 은닉층 ➔ 출력층**
  - latent vector를 잘 찾기 위한 도우미 역할
  - 내부 표현을 출력으로 바꿈(생성 네트워크, generative network)
  - 출력을 **재구성(reconstruction)**