DSAC Module4

Deep Learning (5)

2019, 2020

KPC

딥러닝 응용

사람 행동 분류 – 실습

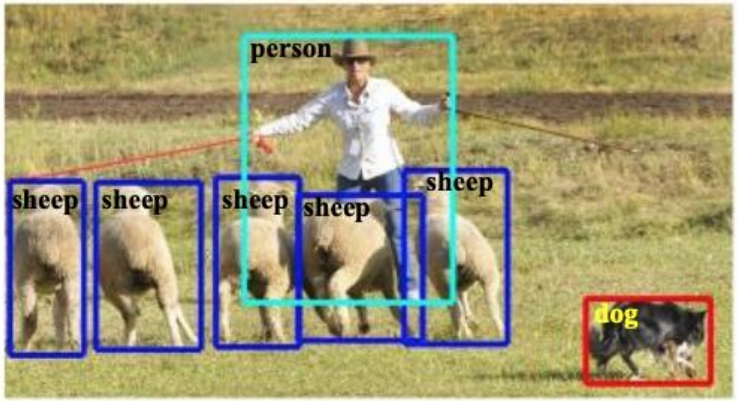
- 사람의 6가지 동작을 분류하는 모델
  - ➢ 핸드폰에 장착된 가속기 센서 데이터를 기반

객체 검출

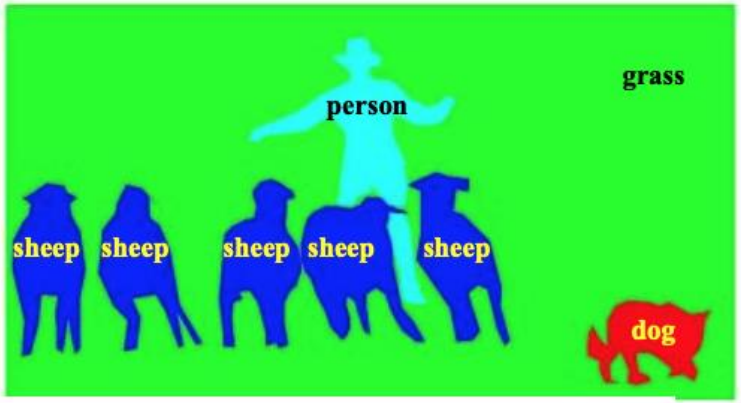
- 이미지 분류
  - 이미지에 어떤 객체들이 들어 있는지만 알아내는 것
  - 여기서는 이미지에 사람, 양, 개가 있다는 것을 찾아낸다
- 객체 검출
  - 찾아낸 객체의 위치까지 알아내는 것
  - 일반적으로는 객체가 있는 위치를 박스 형태로 찾아낸다
- 세그멘테이션
  - 객체의 위치를 박스형태가 아니라 비트 단위로 찾아낸다
  - 즉, 각 비트가 어떤 객체에 속하는지를 분류해내는 것
- 객체 인스턴스 세그멘테이션
  - 이미지를 비트 단위로 어느 객체에 속하는지를 찾아낼 뿐 아니라 각 객체를 서로 다른 객체로 구분하는 기능까지 수행

객체 검출

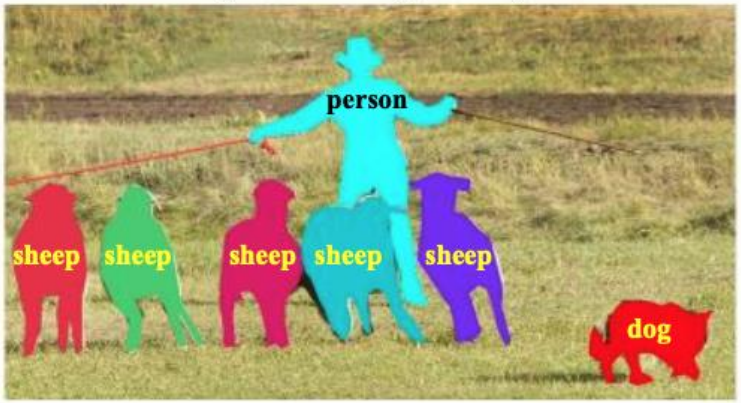
(a) Object Classification



(b) Generic Object Detection (Bounding Box)



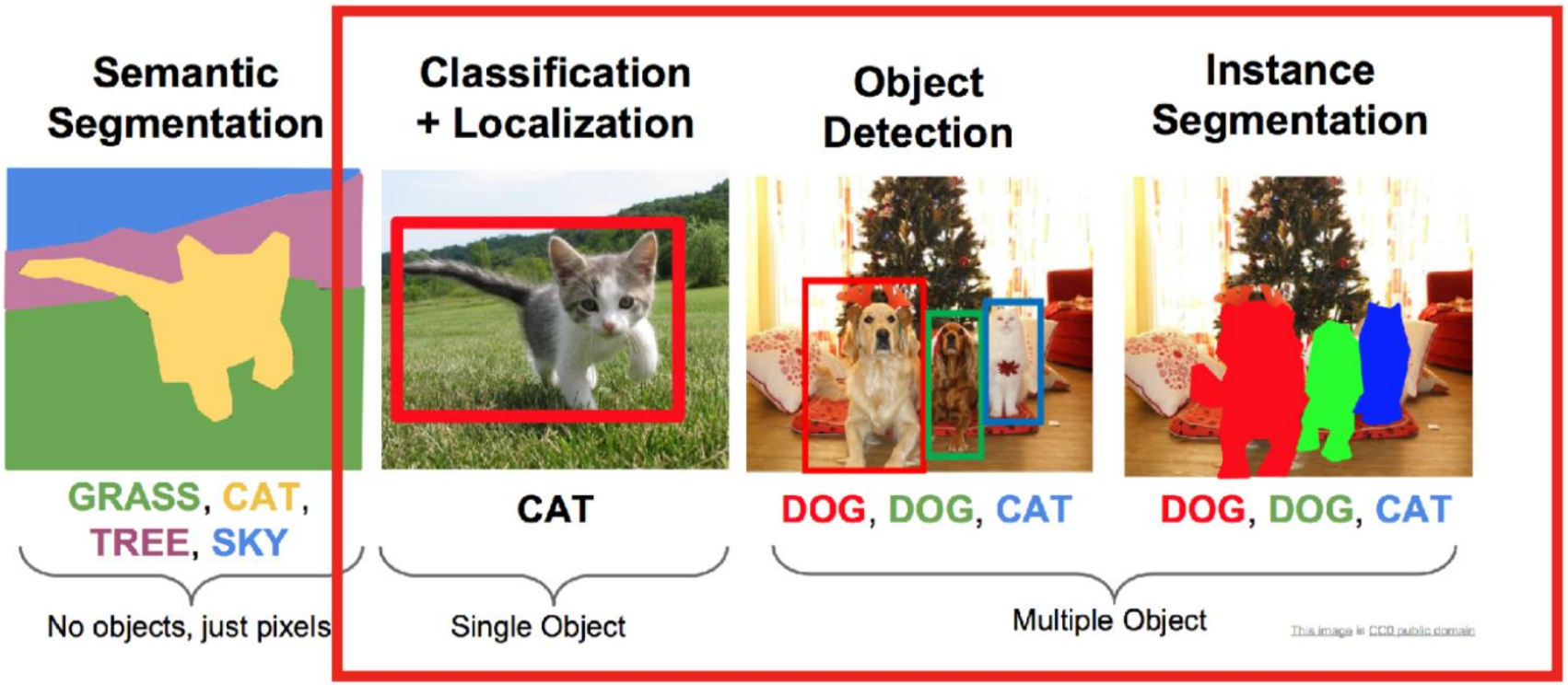
(c) Semantic Segmentation



(d) Object Instance Segmetation

이미지 객체 검출 – localization

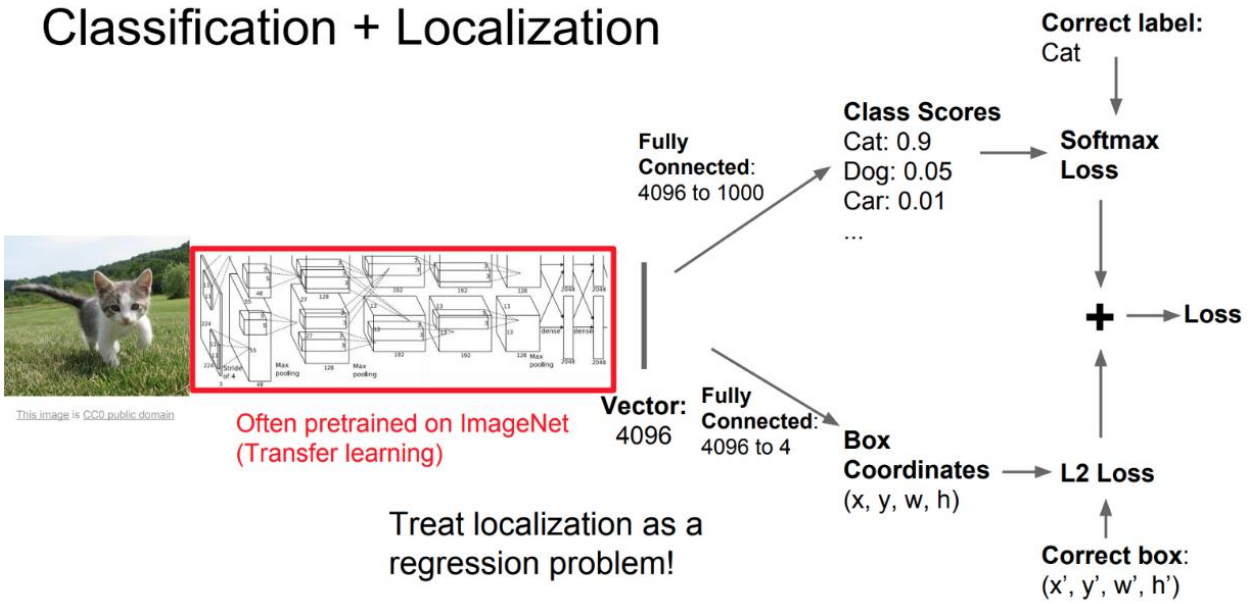
- 객체 검출, 분류에 대한 조금 다른 표현
  - ➢ 위치를 찾는 것 : localization이라는 표현을 사용



손실 함수

- 객체 검출의 2가지 작업
  - 이미지에 어떤 객체가 있는지를 분류(classification)하는 작업
  - 객체가 있는 위치(좌표)를 계산하여 예측하는 기능(localization)

- 손실 함수
  - 객체 검출의 2가지 작업을 동시에 고려하여 학습



손실 함수

- 학습을 시키려면
  - 입력 이미지에 대해서 객체의 종류와 위치를 찾아야 한다

- 레이블 – 2가지 제공해야
  - 이미지 클래스
  - 이미지가 위치한 박스의 좌표

- 출력이 두 가지 이상인 멀티 출력 모델을 만들어야 하며 손실함수는 분류에 관한 손실과 회귀에 관한 손실의 합으로 구성

$$Loss = \alpha * Softmax\_Loss + (1-\alpha) * L2\_Loss$$

- 두 가지 성분의 손실의 상대적인 비중은 α 값으로 조정

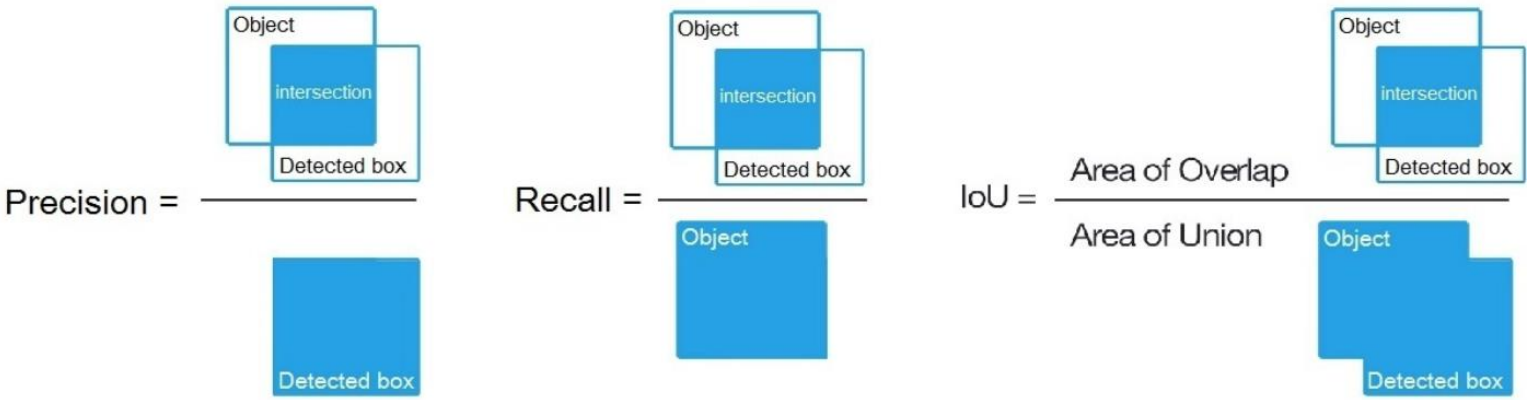
성능 평가

- 객체 검출에서는 단순히 어떤 객체가 있는지를 찾는 분류 문제가 아니므로 정확도 등 만으로 성능을 평가할 수 없고, 예측한 객체의 위치가 실제 객체의 위치와 얼마나 일치하는지를 평가해야 한다

- 성능 평가척도
  - IoU (Intersection over Union)
  - mAP (mean Average Precision)
  - AR (Average Recall)

객체검출 성능 평가 – 정밀도, 리콜, IoU

- 객체검출 성능 평가: 객체의 위치를 얼마나 실제 위치와 겹치게 잘 찾았는지 평가
  - 정밀도 (precision) : 예측한 면적분에 겹치는 면적
  - 리콜 (recall) : 실제 이미지 면적분에 겹치는 면적
  - IoU : 두 가지 면적의 전체집합 부분에 비하여 겹치는 부분의 면적의 비율



성능 평가 – IoU

- 실제 이미지 예

성능 평가 – mAP

- 객체 검출에서 검출할 객체가 여러 개가 있을 때 이들의 평균 예측 성능을 표현하는데 mAP가 많이 사용

- mAP
  - 검출작업의 평균 정확도
  - 여러 객체에 대해 각각 AP를 구하고 이의 평균치를 구한 것

- 일반적으로 mAP가 0.5 이상이면 true positive라고 판단

객체 검출 방식

- 슬라이딩 윈도우 방식 (초기)
  - 객체 검출을 위해서 전체 이미지를 작은 크기로 나누고 각 부분을 대상으로 객체를 찾는 작업을 수행
  - 이 방식은 시간이 오래 걸리고 더욱이 윈도우 크기와 객체의 크기가 다를 때 찾지 못한다
- Region Proposal (R-CNN 에 적용)
  - Sliding window 를 전체에 대해 하지 않고 물체가 있을 것 같은 영역에 대해서만 함. (스케일링 등이 필요 없어 빠름): segmentation 이용

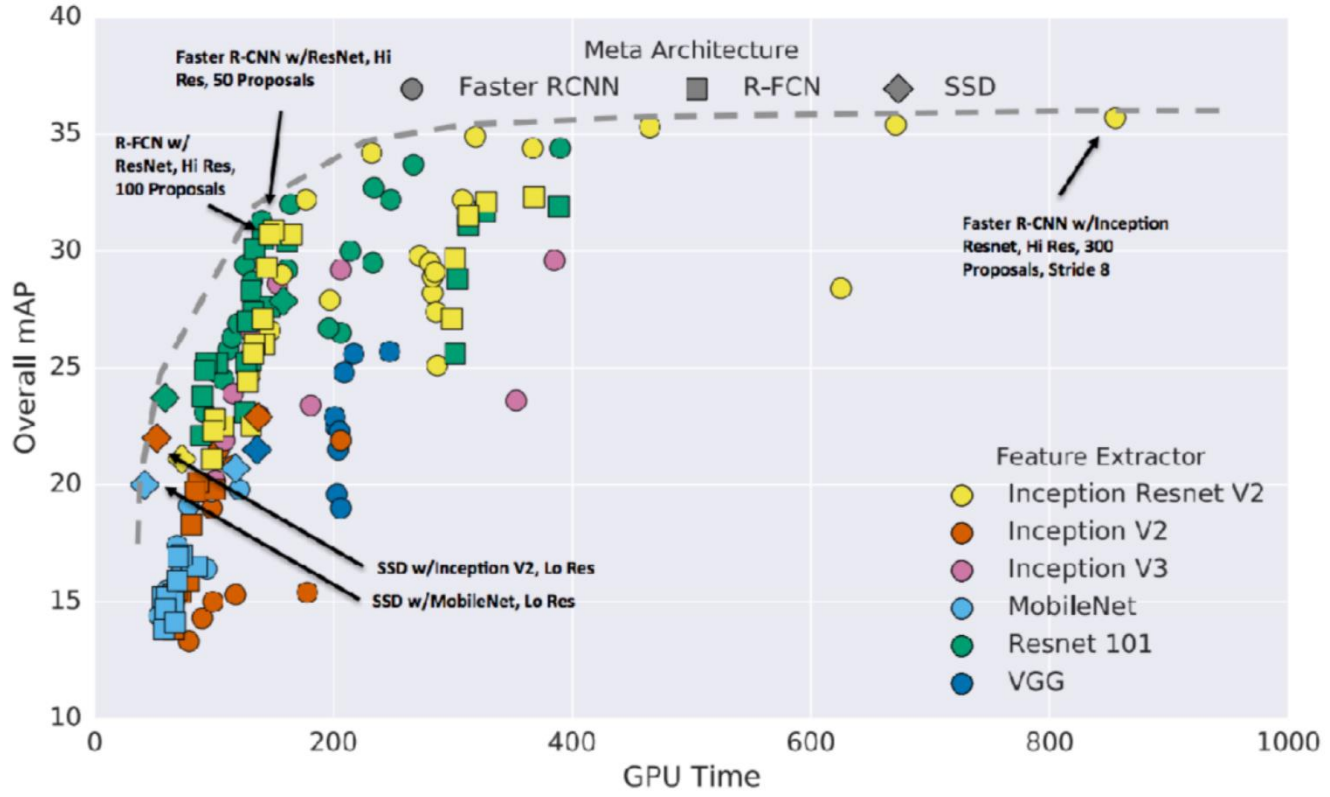
객체 검출 방식

- 한편 객체의 위치를 찾는 문제
  - 분류가 아니라 회귀문제
  - 즉, 객체의 경계 박스의 좌표 값은 회귀 문제로 예측
  - 이 방법은 사람의 자세를 예측하는 포즈 검출, 기준점 검출에서도 사용

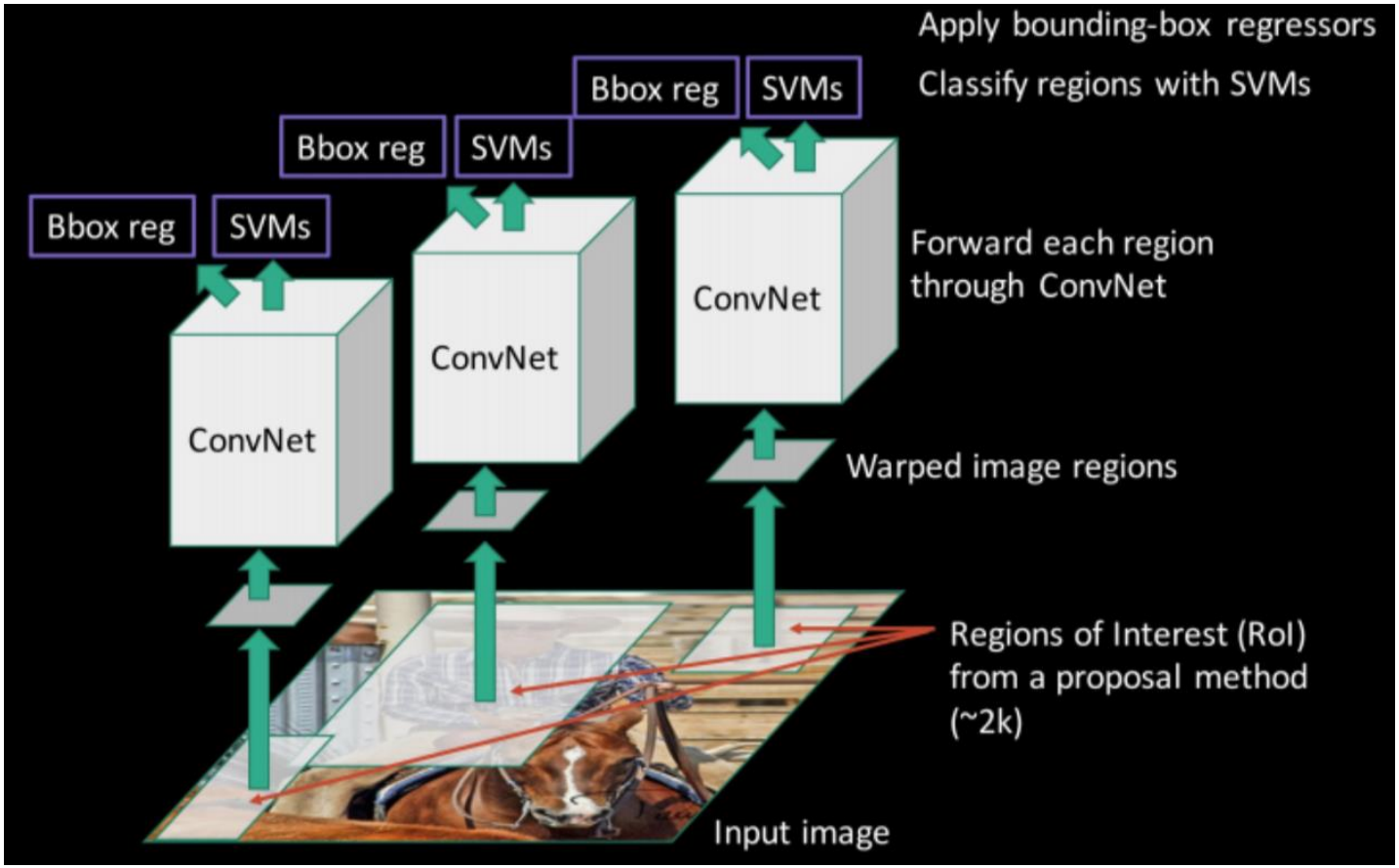
(자세 검출의 예)

객체 검출 방식 – 알고리즘 비교

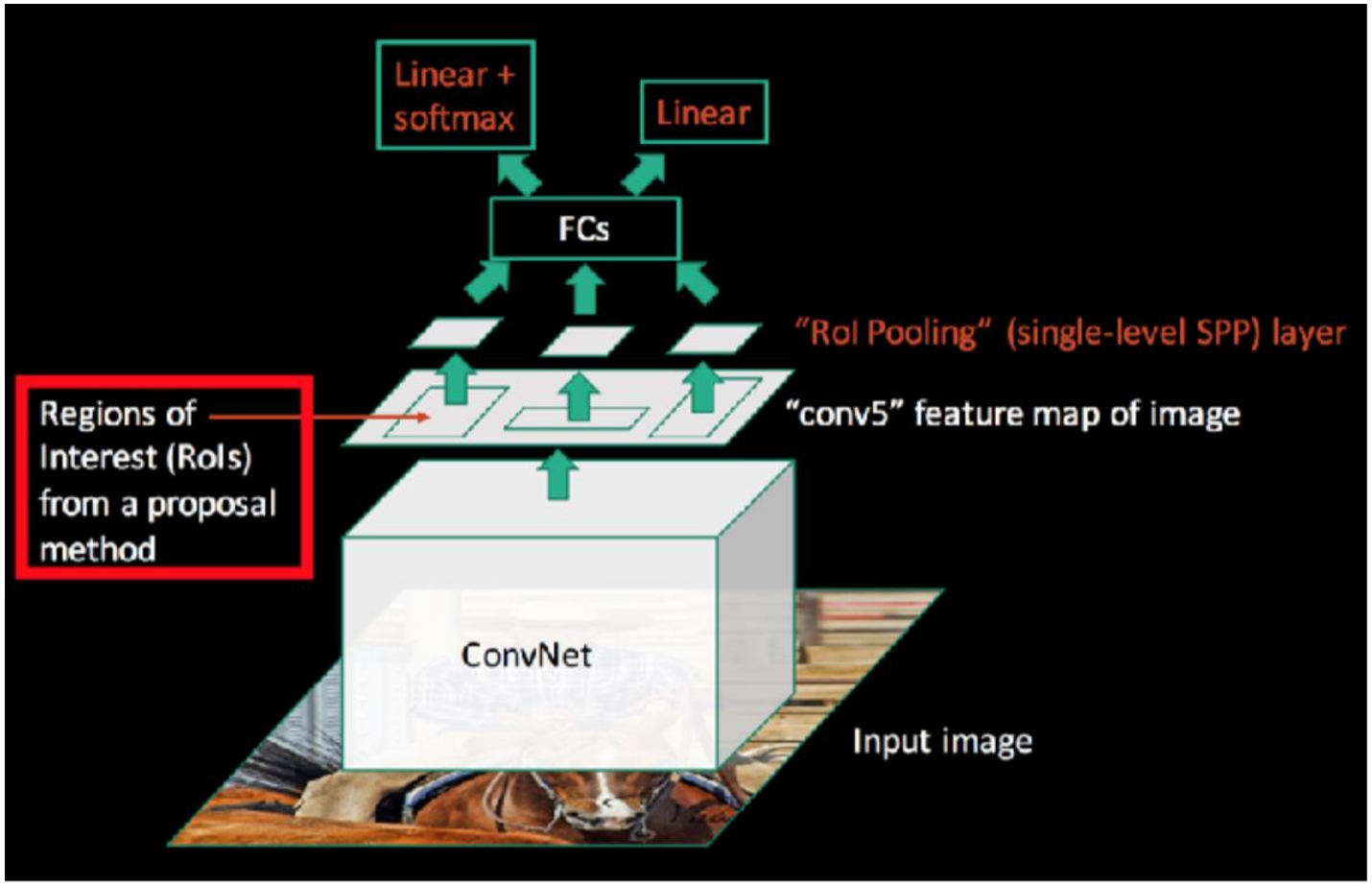
- Resnet을 사용한 Faster R-CNN등이 우수한 성능을 나타냈으나 최근 Mask-R-CNN과 Yolo등 성능이 더 개선된 방식이 소개됨
  - x 축 : 객체 검출 계산에 필요한 시간
  - y 축 : mAP 즉, 검출 성능



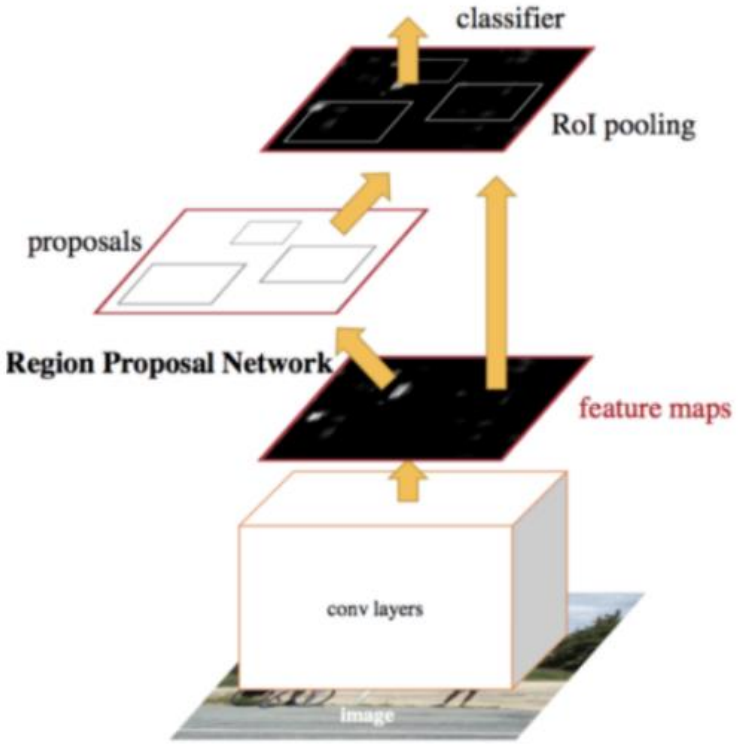
R (Regional) CNN



Fast R-CNN


Linear + softmax
Linear
FCs
"RoI Pooling" (single-level SPP) layer
"conv5" feature map of image
Regions of Interest (RoIs) from a proposal method
ConvNet
Input image

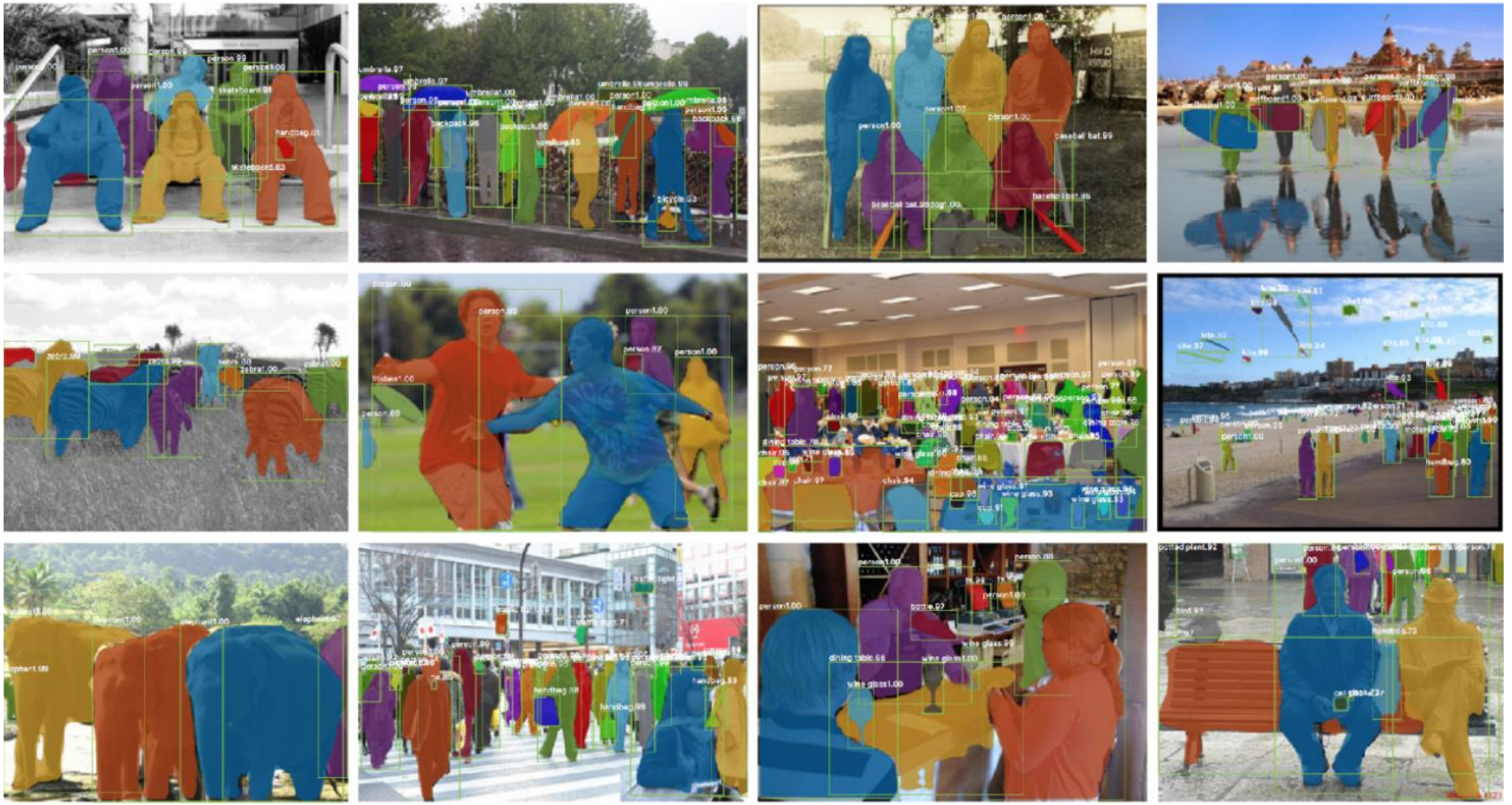
Faster R-CNN



- CNN layer 를 통해 feature map 추출
- RPN 을 통해 개체를 포함할 가능성이 높은 윈도우 생성
- 각 윈도우의 피처멥 검색하여 고정 트기로 조정한 뒤 (ROI 풀링)
- 클래스 확률과 해당 객체에 대한 정확한 경계박스 예측

Mask R-CNN

- Detectron이라는 기술로 Facebook에서 공개
- 상당히 정교하게 객체 인식과 세그멘테이션을 수행



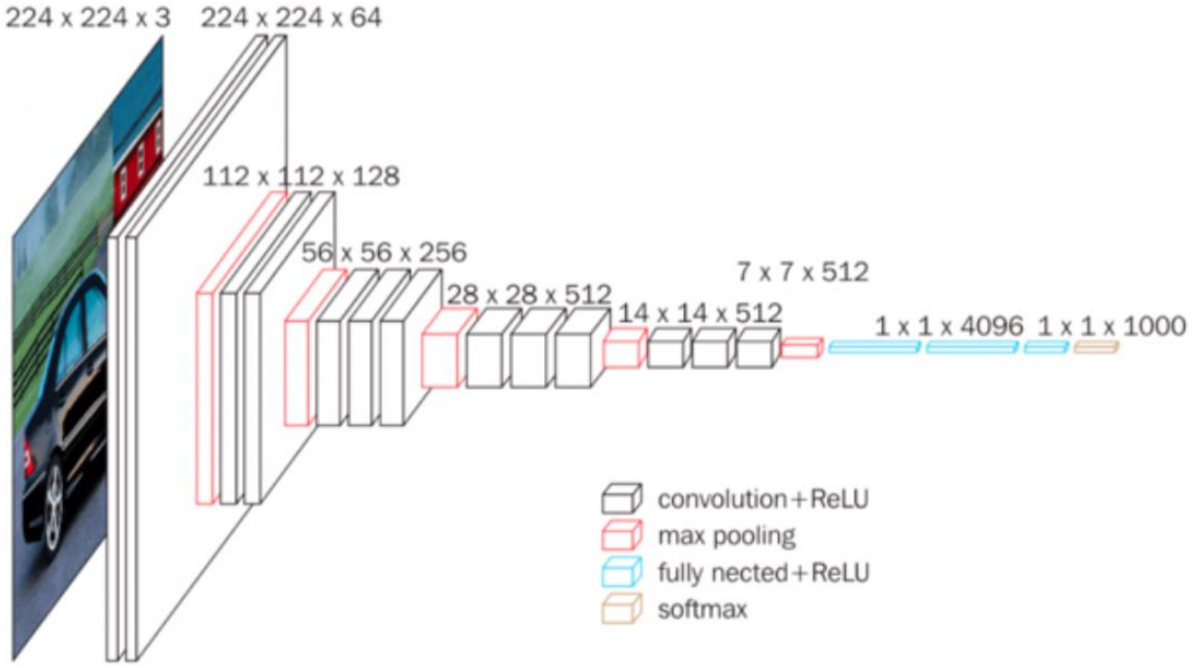
이미지 데이터 셋 (객체 검출) – ImageNet

- 분류, localization, 검출을 위한 데이터를 제공
- 일반적인 사물 1,000 종류의 1,400만 장의 사진
- 경계박스(bounding box) 좌표 값도 제공
- 각 이미지 당 평균 1.1개의 객체를 포함
- 1,000개의 객체 목록

  https://gist.github.com/aaronpolhamus/964a4411c0906315deb9f4a3723aac57

- ILSVRC(ImageNet Large Scale Visual Recognition Competition)
  - 대표적인 이미지 인식 경진대회
  - 2017년에 인간의 성능을 능가하면서 종료됨
  - 2012년 AlexNet이 큰 성과를 낸 후 딥러닝 모델이 확산
  - 2014년 우승 모델인 VGG16 모델이 전이학습에 널리 사용

VGG16 모델


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

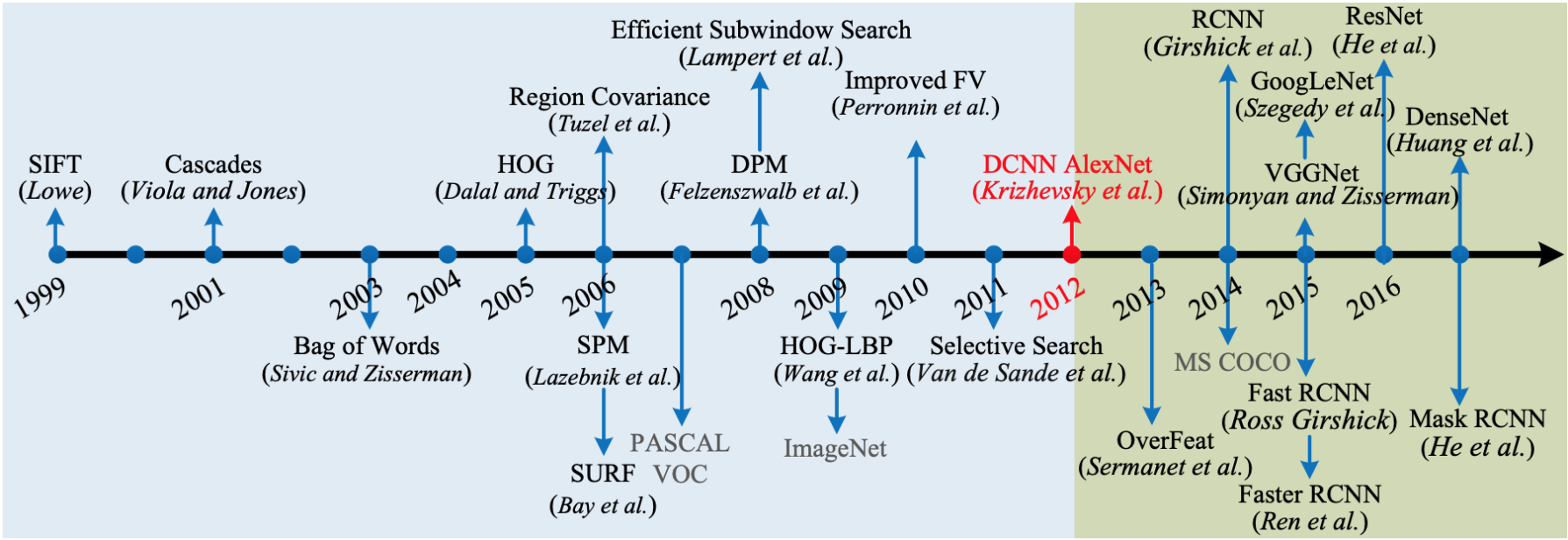

3x3 conv , 64
3x3 conv , 64
pool/2
3x3 conv , 128
3x3 conv , 128
pool/2
3x3 conv , 256
3x3 conv , 256
3x3 conv , 256
pool/2
3x3 conv , 512
3x3 conv , 512
3x3 conv , 512
pool/2
3x3 conv , 512
3x3 conv , 512
3x3 conv , 512
pool/2
fc 4096
fc 4096
fc 4096
Size:224
Size:112
Size:56
Size:28
Size:14
Size:7

이미지 데이터 셋 (객체 검출) – VOC, COCO

- VoC
  - 20개 클래스, 11,530 이미지, 27,450 annotation을 제공
  - image당 2.4개의 객체를 포함

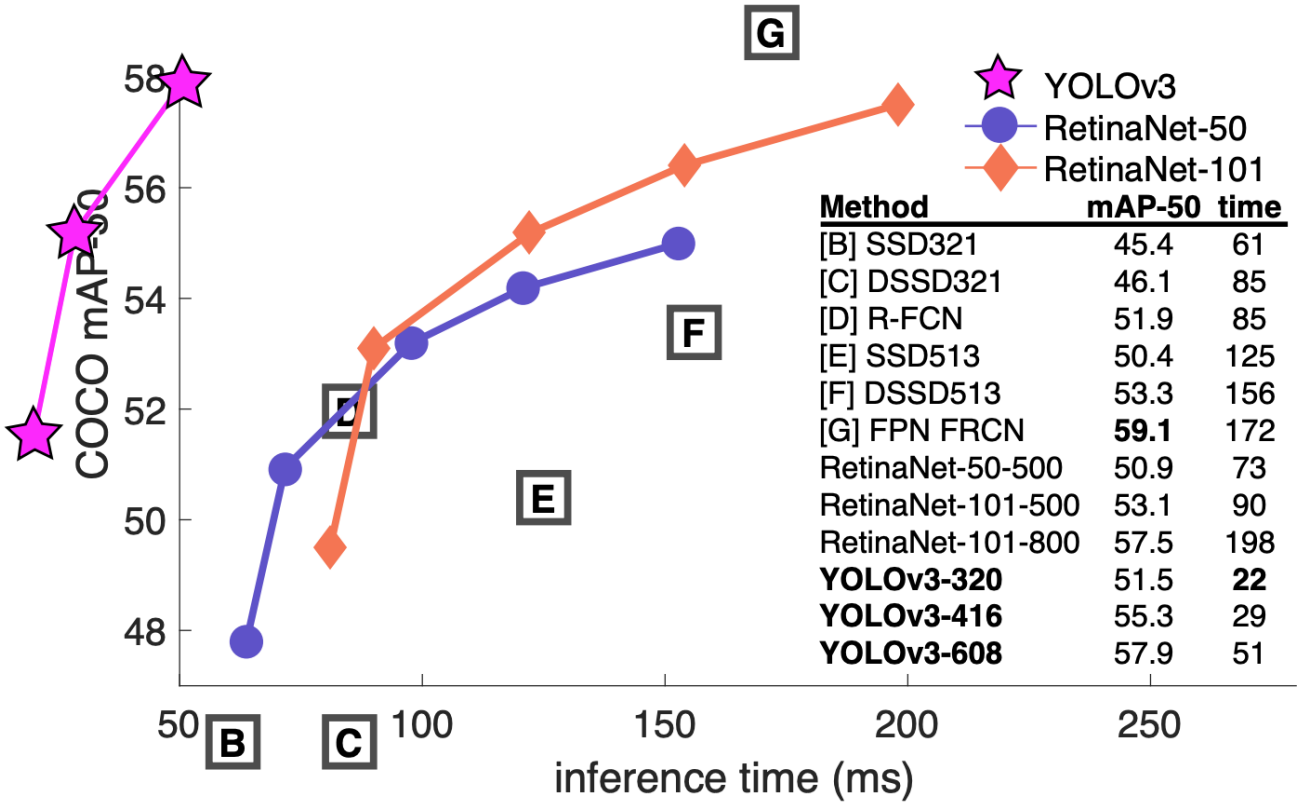
- COCO
  - MS 사가 주최하는 COCO challenge 대회가 2015년부터 운영 중이며 여기서 제공되는 데이터가 COCO(common object context)
  - 91 카테고리, 20만개의 이미지, 50만개의 annotation, 32만개 영상을 제공
  - image당 평균 7.3 개의 객체를 포함

이미지 데이터 셋 (객체 검출) – 관련 기술



YOLO (You only look once)

- 객체 검출 방식 중에 가장 속도가 빠르고 성능도 우수한 방법으로 널리 사용



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

YOLO – 동작 데모

- 객체 인식(classification)과 위치예측(localization)을 동시에 하나의 신경망으로 수행
  - ➢ 즉, 객체 예측과 경계박스를 찾는 작업을 동시에 수행하는 Single Shot Detector(SSD)을 수행
  - ➢ 기존의 다른 객체 검출 알고리즘들은 별도로 수행

- 입력 이미지 전체를 여러 지역(region)으로 나누고 경계 박스와 각 지역에 대한 객체 존재여부를 확률로 예측
  - ➢ 이 경계박스는 예측확률의 가중합으로 계산

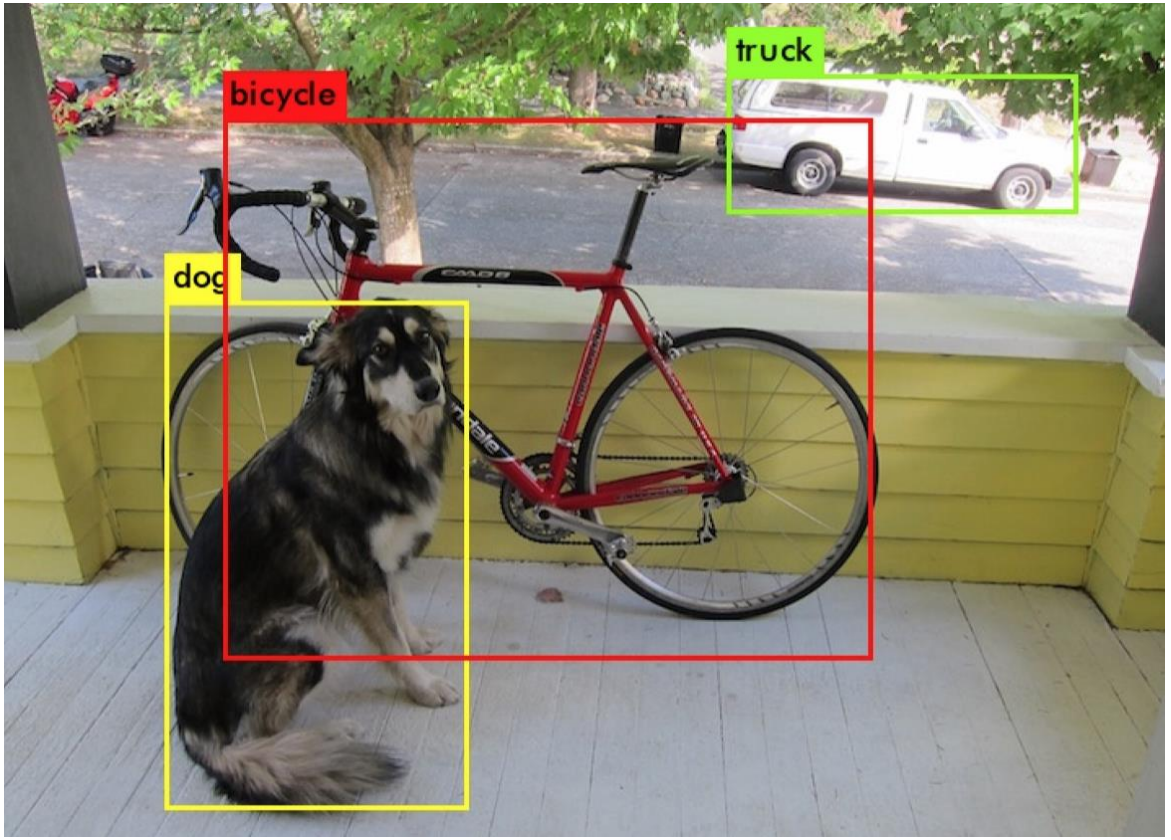
- 98개의 검출 결과를 제공 (7x7x30 = 1470 벡터 출력)

- GoogleLeNet 기반으로 동작, C로 구현된 DarkNet을 framwork로 사용

  - ❖ 참고: https://github.com/qqwweee/keras-yolo3
  - ❖ 동작 데모 : https://pjreddie.com/darknet/yolo/#demo

YOLO

- 동작 데모



YOLO – Tiny YOLO3

- 정식 YOLO보다 간단하면서 속도가 빠른 tiny YOLO 버전
- 이를 사용하려면 아래와 같이 해당 버전을 설치
  - !wget https://pjreddie.com/media/files/yolov3-tiny.weights

객체 분할 (Segmentation)

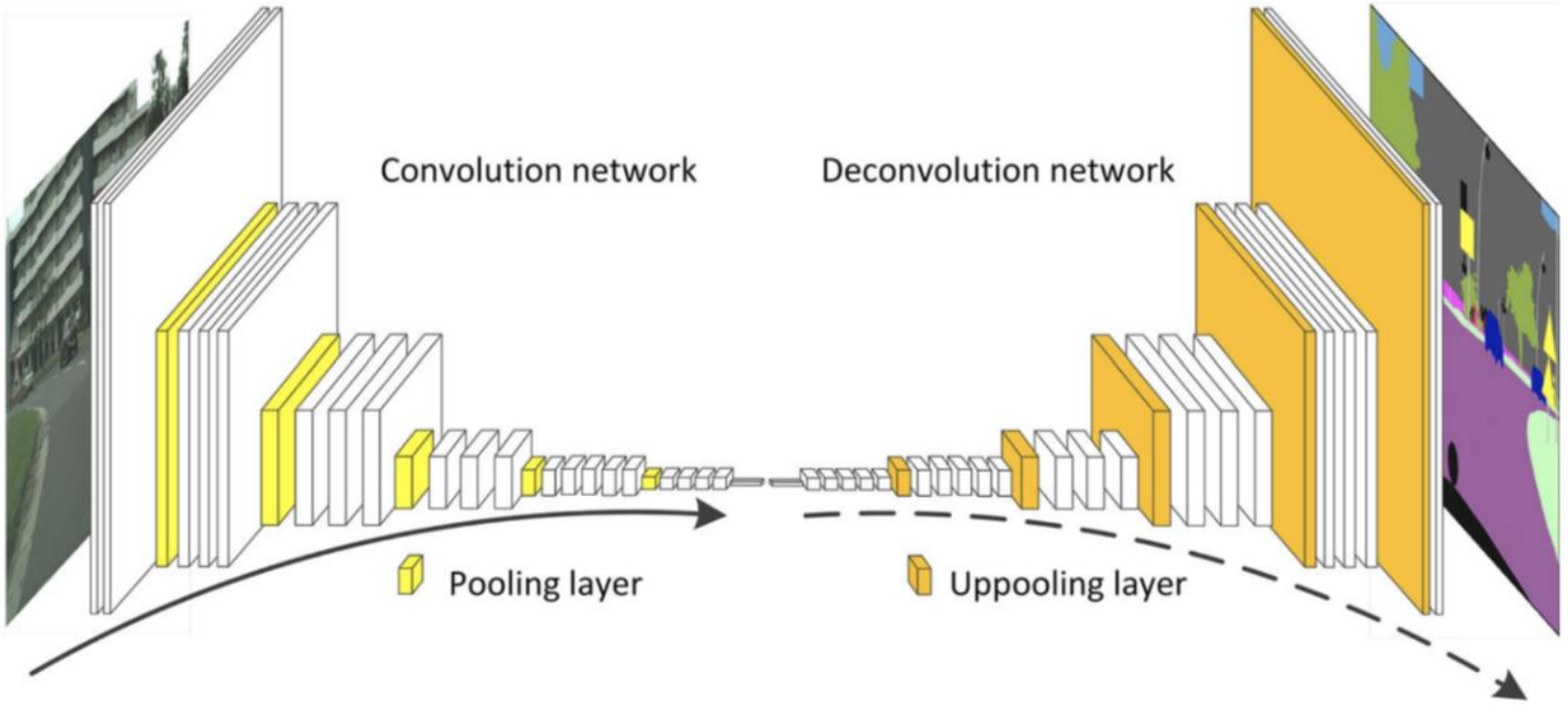
- 픽셀 단위로 객체의 클래스를 표시하는 작업

- 머신러닝 모델
  - 정보를 계속 줄여나가는 작업을 주로 수행
  - 분류와 회귀는 모두 정보를 줄인 결과 클래스 이름이나 수치를 얻는다.

- 객체 분할
  - 원래 이미지 크기의 이미지를 다시 얻어야 한다
  - 따라서 정보를 축소한 후에 다시 정보를 늘리는 작업을 수행해야
  - Deconvolution, uppooling 수행

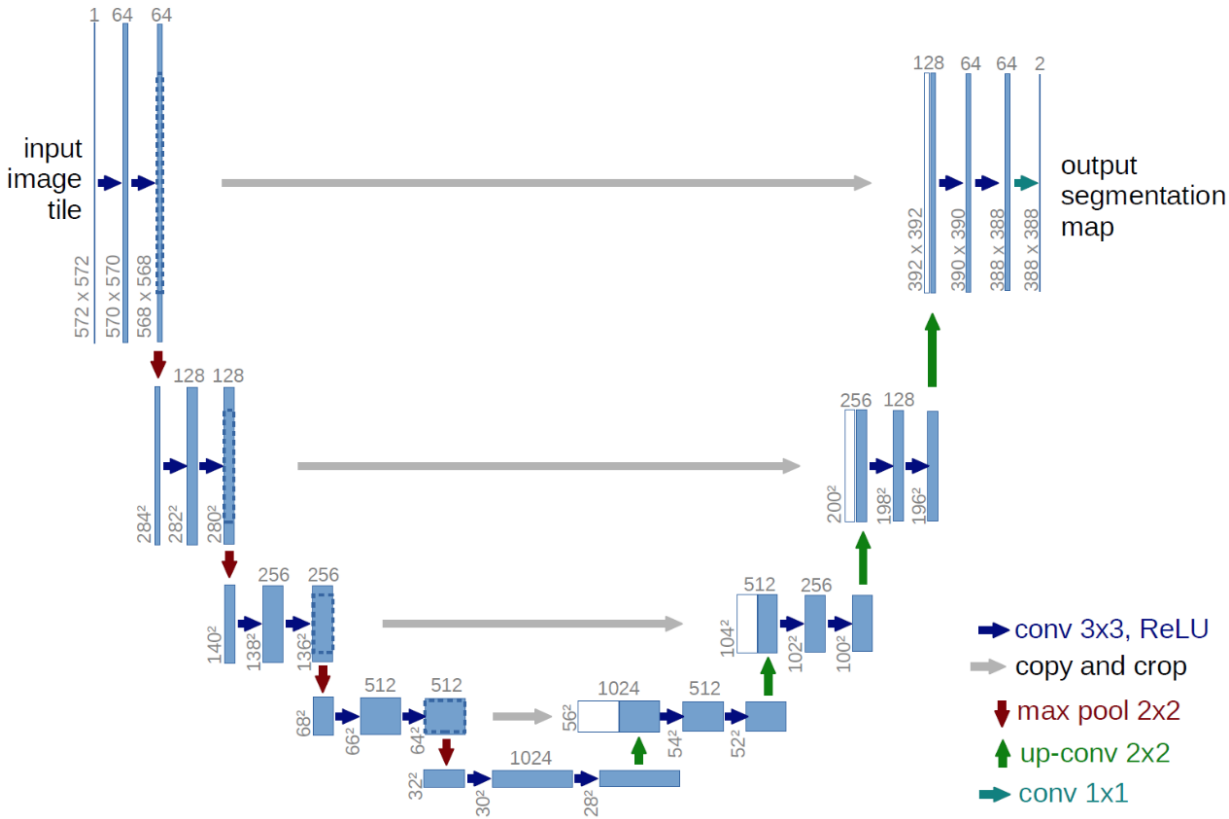
객체 분할 모델 구조

- Encoder-Decoder 구조로 구성



객체 분할 모델 구조 – U-Net

- 객체 분할을 위해서는 U자형 구조를 갖는 U-Net을 주로 사용



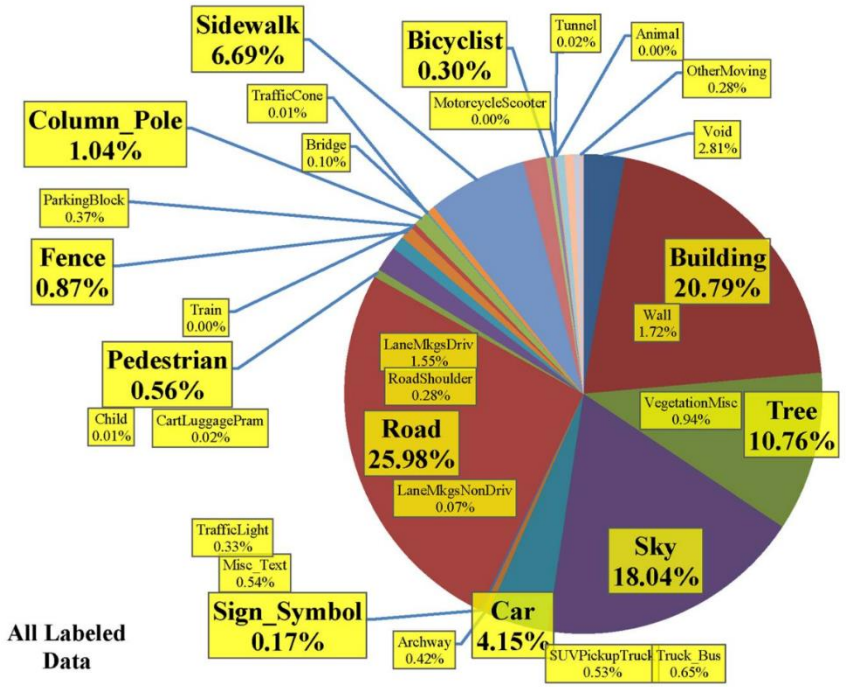
객체 분할 모델 구조 – U-Net

- 실행 결과
  - https://www.youtube.com/watch?v=ATIcEDSPWXY

- Mask R CNN과 비교 동영상 비교
  - https://www.youtube.com/watch?v=s8Ui_kV9dhw

객체 분할 – 데이터 셋

- 객체 분할을 위해서 사용되는 데이터 셋 : CAMVID, COCO 등
- CAMVID(Cambridge-driving Labeled Video Database) 데이터
  - http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/



객체 분할 – 데이터 셋

- COCO 데이터 셋
  - http://cocodataset.org/#explore

- 데이터의 특징
  - Object segmentation
  - Recognition in context
  - Superpixel stuff segmentation
  - 330K images (>200K labeled)
  - 1.5 million object instances
  - 80 object categories
  - 91 stuff categories
  - 5 captions per image
  - 250,000 people with keypoints

오토 인코더 (Autoencoder)

- 목표
  - ➢ 이미지나 시계열 데이터인 입력신호를 똑 같이 재생
  - ➢ 분류나 예측을 하는 것이 목적이 아님

- 이를 위해서 신경망은 내부적으로 가능한 적은 수의 파라미터로 입력 신호를 표현하는 능력을 학습하는 것이 목적

- 이를 통해서 차원축소, 특성 추출, 비지도 사전훈련 등을 할 수 있다

- 이러한 모델을 생성형(generative) 모델이라고 함

- 우리가 어떤 사람을 한번 보고 기억만으로 그림을 그린다면 그 사람의 특징들을 기억하면서 그리게 된다. 이미지 자체를 기억하기는 어렵다.

- 오토엔코더는 이와 같이 입력의 특성을 추출하여 이를 잘 기억하는 능력을 학습

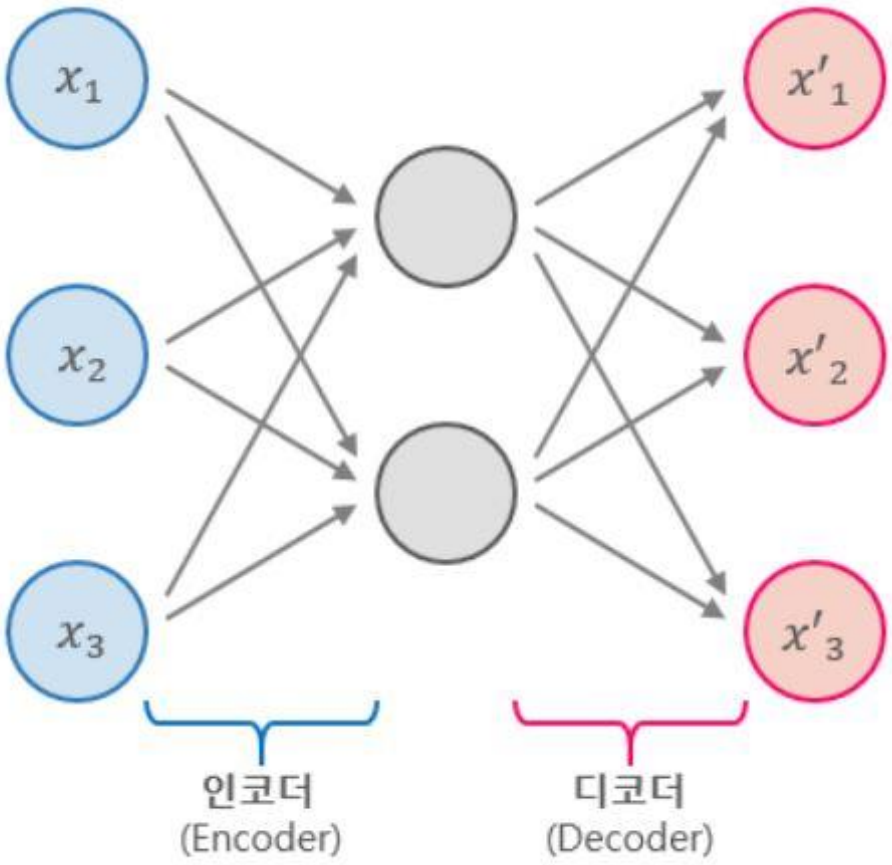
오토 인코더 (Autoencoder)

- 오토엔코더를 이용하여 학습된 모델을 이용하면 향후 이미지 인식이나 분류를 더 빨리 학습할 수 있다

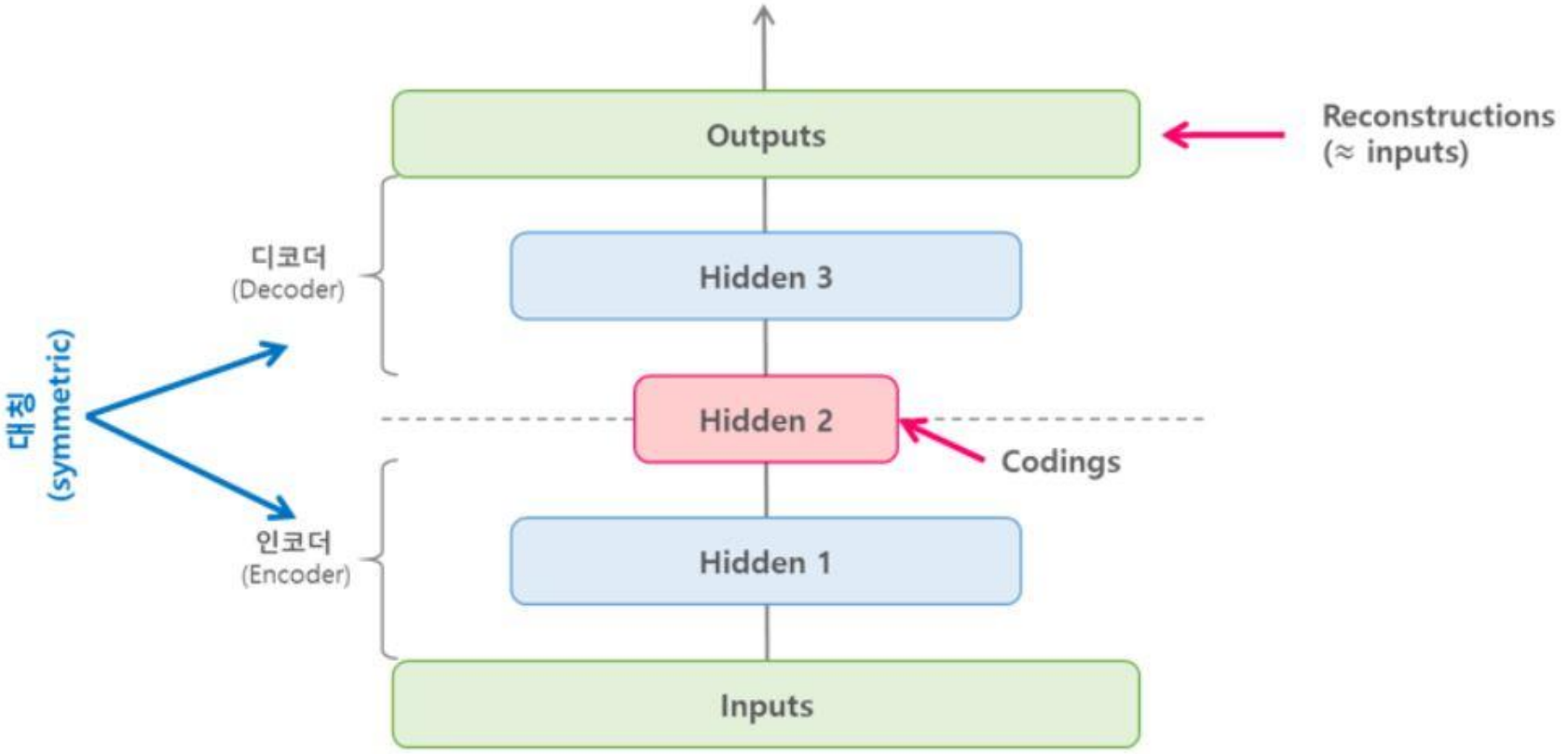
- 우리가 그림을 잘 복제하여 그리는 능력이 뛰어나면 그림에 대한 관찰과 표현력이 늘어나서 새로운 그림을 볼 때 그림의 특성을 잘 파악할 수 있는 것과 같다

- 그림 보는 능력이 생기려면 많은 그림을 그려봐야 하고 좋은 글을 쓰려면 좋은 글을 많이 읽어봐야 하는 것과 같이 오토 엔코더로 신호의 인식과 표현 능력을 학습시킬 수 있다

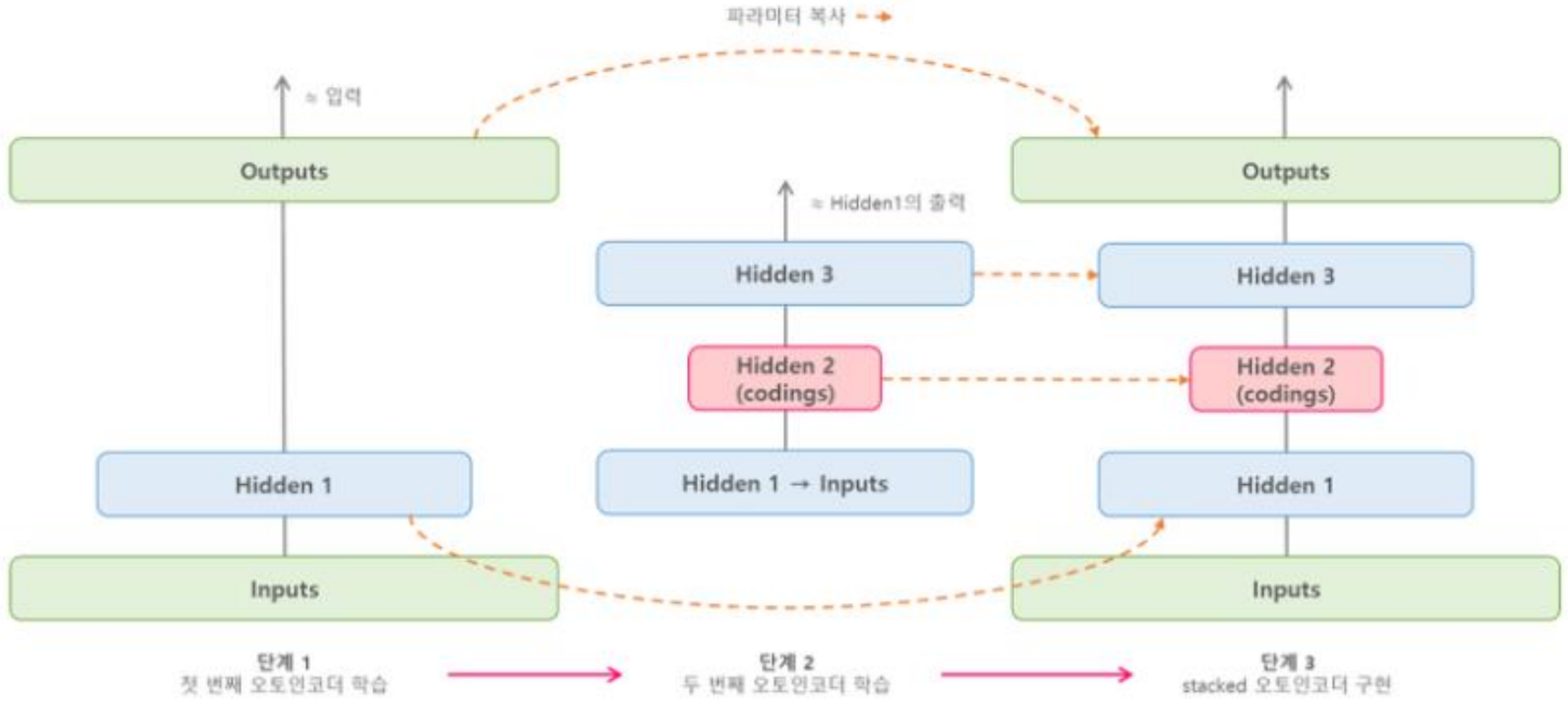
오토 인코더 (Autoencoder)



Stacked 오토 인코더

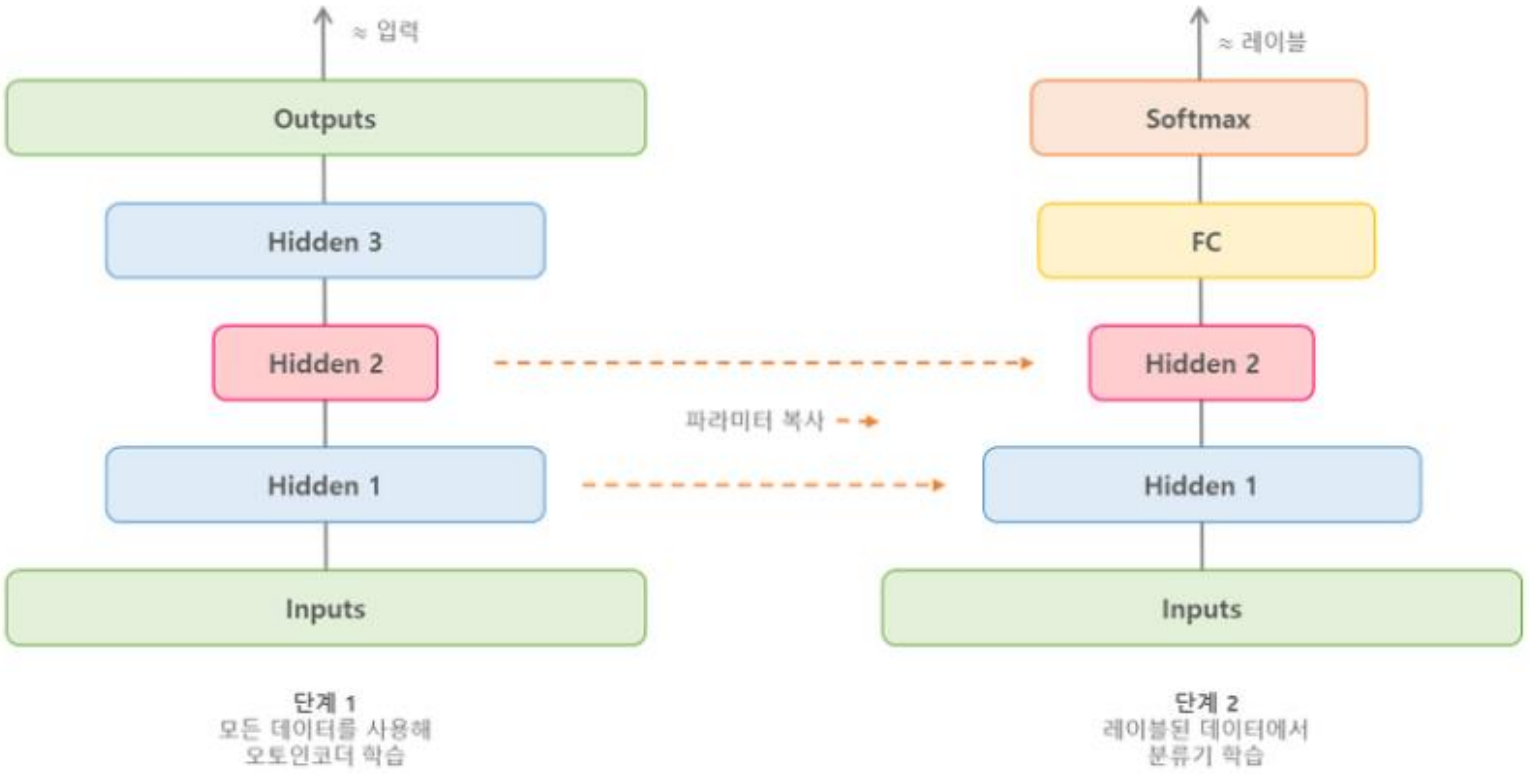


Stacked 오토 인코더 – 학습(한번에 한층씩)



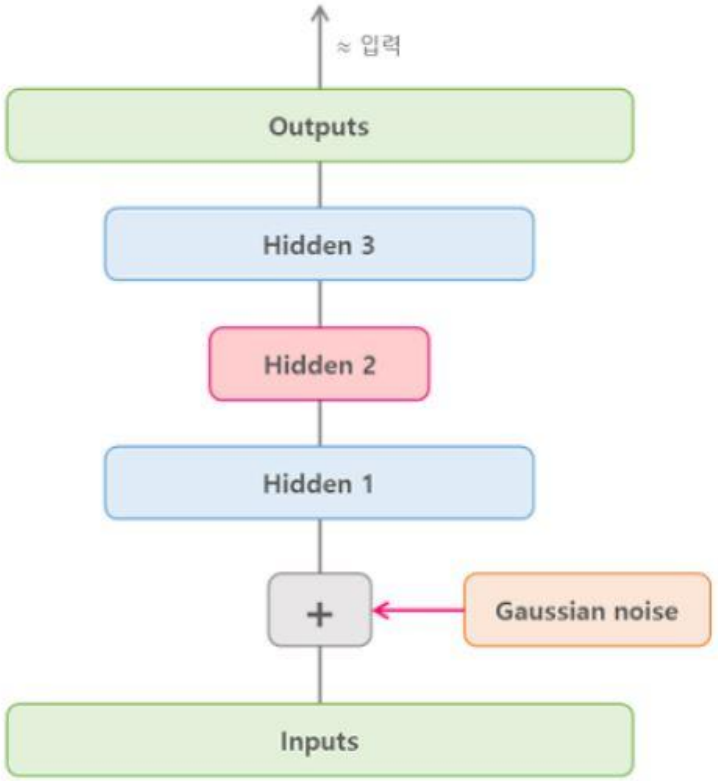
Stacked 오토 인코더 – 비지도 사전학습

- 대부분이 레이블되어 있지 않는 데이터셋이 있을 때,
  - 먼저 전체 데이터를 사용해 stacked-오토인코더를 학습
  - 다음, 오토인코더의 하위 레이어를 재사용해 분류와 같은 실제 문제를 해결하기 위한 신경망을 만들고 레이블된 데이터를 사용해 학습

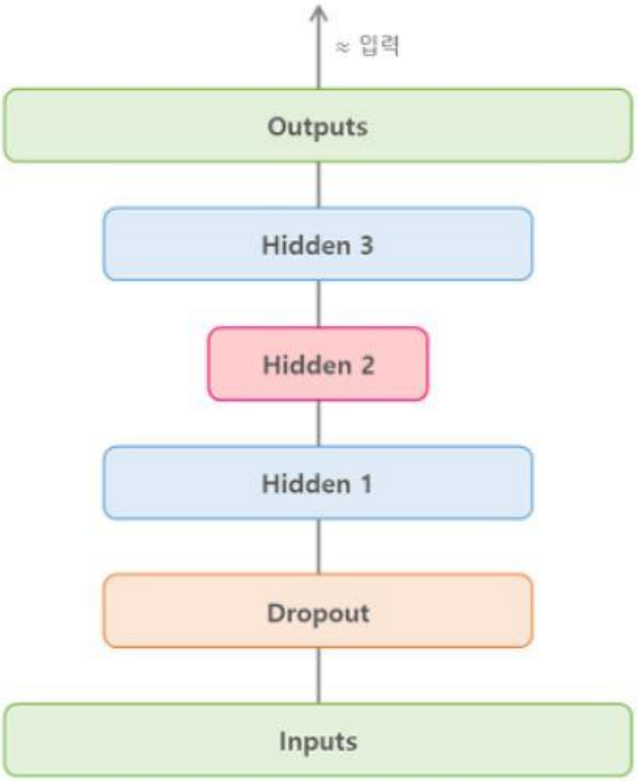


Denoising 오토 인코더

- 입력에 노이즈(noise, 잡음)를 추가
- 노이즈가 없는 원본 입력을 재구성하도록 학습



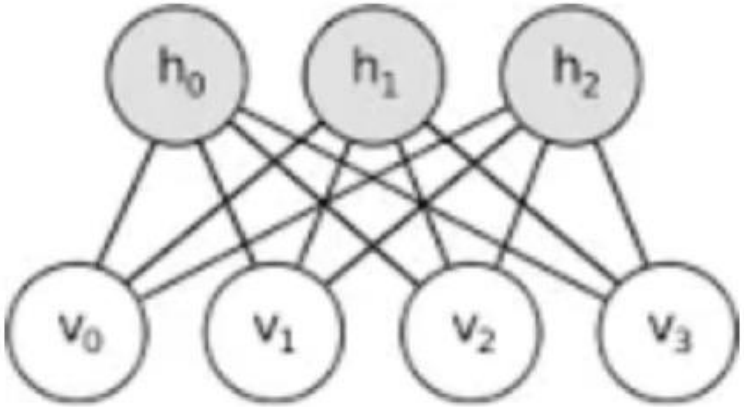
Denoising Autoencoder
With Gaussian noise



Denoising Autoencoder
With Dropout

제한 볼쯔만 머신 (RBM)

- RBM (Restricted Boltzmann machine)
- DBN(DBN, Deep Belief Network)이라고 하는 딥러닝의 일종인 심층신뢰신경망을 구성하는 기본적인 요소
- RBM을 여러겹 쌓아서 심층신뢰신경망을 만든다
- 같은 층 내부 노드간의 연결은 없고 층과 층 사이와의 연결만이 있다. 이런 점에서 제한적(Restricted)라고 불린다



제한 볼쯔만 머신 (RBM)

- RBM에서는 머신만이 이해할 수 있는 수치(가중치) 정보를 확률적으로 학습시킨다 - 노드 출력값을 확률로 결정한다
- 역전파 신경망은 목표치와의 오류를 계산하고 오류 만큼 역전파시켜 학습을 시킨다
- RBM은 단층이기에 목표치가 없고 대신 에너지라는 개념을 사용
- 에너지(E)는 마이너스(-) 부호를 가지고 가중치가 0보다 큰 수이기에 입력과 출력이 양의 상관 관계에 있을수록 에너지가 작아진다.
- 예로 입력이 -1일 때 은닉층 결과가 -1이면 에너지가 최소가 된다.

제한 볼쯔만 머신 (RBM) – 학습

- RBM의 학습은 입력에 가중치를 곱하고 더하고 확률을 구하고 확률에 근거하여 샘플링하여 은닉층의 노드값을 정한다

- 은닉층의 노드값이 결정되면 또 다시 입력층의 노드값들을 같은 방식으로 결정한다. 그러면서 에너지를 최소화하는 방향으로 가중치를 학습시킨다

비지도 딥러닝 네트워크 – 심층 신뢰망

- 심층의 신경망을 학습
  - 오차 소멸(vanishing gradient) 문제

- 즉, 심층 신경망에 대해서 오류역전파 알고리즘을 적용하면 출력층에 대해서는 가중치가 잘 교정되나 입력층 쪽으로 감에 따라서 오차값이 점차 적어져서 입력층에 대한 가중치가 교정이 잘 되지 않는 문제

- 이 문제를 해결하기 위해서
  - DBN 은 아래층(입력에 가까운 층)에서부터 위층으로 가면서 순차적으로 선훈련을 진행

- 참조:
  https://bi.snu.ac.kr/Courses/ML2016/LectureNote/LectureNote_ch5.pdf

비지도 딥러닝 네트워크 – 심층 신뢰망

- 심층 신뢰망은 크게 두 가지로 구분
  - 하나는 입력과 같은 출력을 재생성하도록 하는 오토인코더
  - 다른 하나는 분류기로 사용하는 것

- 분류 문제를 풀기 위해서는
  - 무감독학습의 마지막 은닉층을 입력으로 하고
  - 출력층을 하나 추가하여 감독학습을 시킨다

생성적 대립 네트워크 (GAN)

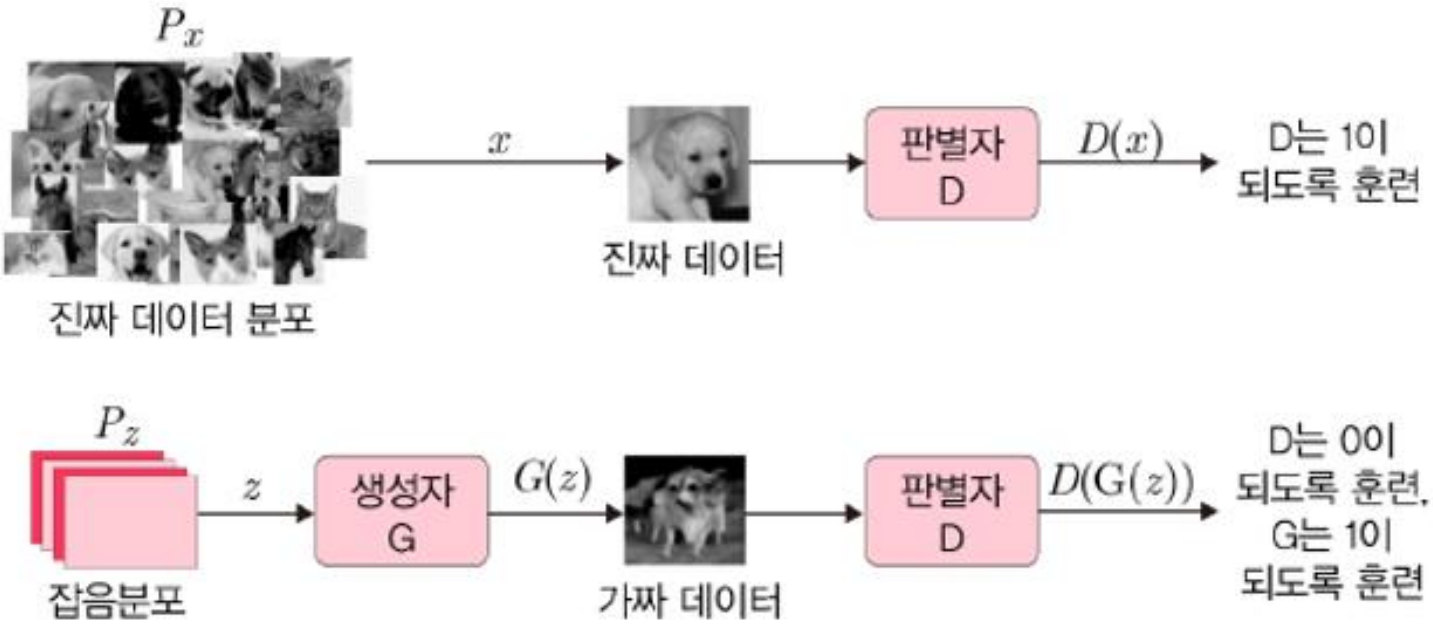
- 생성적 대립 네트워크 (Generative Adversarial Network)
  - ➢ 두 개의 신경망을 서로 경쟁시켜서 새로운 이미지를 만드는 딥러닝 모델
  - ➢ 생성(Generative) 모델과 판정(Discriminant) 모델 두 가지로 구성

- 생성 모델
  - ➢ 이미지를 새롭게 합성하여 만드는 작업
  - ➢ 판정 모델을 잘 속이면 보상을 받도록 학습

- 판정 모델
  - ➢ 합성된 이미지가 원래의 이미지인지를 판정하는 작업
  - ➢ 원본 이미지와 모방 이미지를 잘 구분하면 보상을 받도록 학습

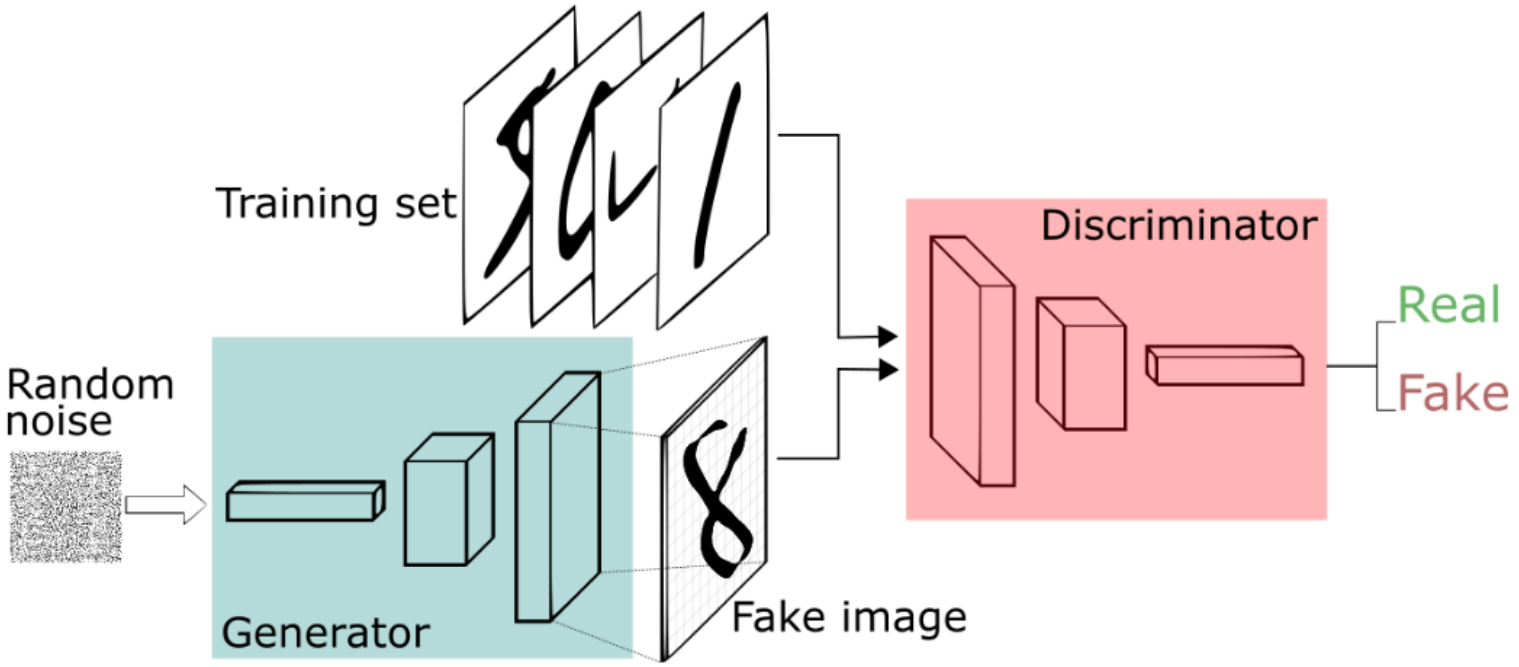
- 이 두 모델이 서로 경쟁하면서 학습하도록 하면 (창과 방패처럼) 생성 모델은 random noise로부터 원본과 비슷한 이미지를 만들어 내는 능력을 갖게 된다

생성적 대립 네트워크의 동작 원리



	생성자	판별자
입력	잡음	진짜 데이터 또는 가짜 데이터
출력	가짜 데이터	진짜 데이터일 가능성
목적	진짜 같은 가짜 데이터 생성	진짜 데이터와 가짜 데이터를 완벽하게 구별

생성적 대립 네트워크의 동작 원리



생성적 대립 네트워크의 동작 원리

- 생성자

$$\max_D E_{x \sim P_x(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$

- 판별자

$$\min_G E_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$

생성적 대립 네트워크 – 응용

- 새로운 이미지를 창작
  - ➢ 원본 이미지를 똑같이 만드는 것이 아니라, 다른 이미지의 특성을 반영하도록 하여 스타일이 바뀐 새로운 이미지를 합성할 수 있다.

- 음악 작곡
  - ➢ 주어진 음악 스타일을 반영한 새로운 곡을 작곡하도록 학습할 수 있다

- GAN의 동작 확인
  https://youtu.be/XOxxPcy5Gr4

생성적 대립 네트워크 – 동작 확인

- video

강화 학습 – 기본 원리

- 프로그램이 정답만 보고 학습하는 것이 아니라, 종합적인 외부 환경에 반응하면서 배우도록 하는 것
  - 즉, 바로 답을 가르쳐주지는 않아도 학습
  - 사람이 하는 학습과 유사

- 자동차 운전을 배울 때에도 운전하면서 나타나는 다양한 환경에 지속적으로 노출되면서 핸들과 브레이크, 액셀로 반응하면서 운전을 배운다

- 우리가 운전하는 방법을 if-then-else를 사용하는 논리적인 흐름을 따르는 프로그램으로는 구현하기 어렵지만, 우리는 편안하게 운전할 수 있다

강화 학습 – 기본 원리

- 최종 목표가 주어졌을 때 환경(주변의 여건)에 대한 나의 행동, 그리고 행동에 대한 결과를 보면서 학습하는 방법
  - ➢ 스스로 공부하는 방법을 가르치는 것

- 현재 강화학습이 적용되는 분야는 룰이 비교적 간단한 게임을 다루지만 자율주행자동차를 보면 간단한 작업이 아니다.

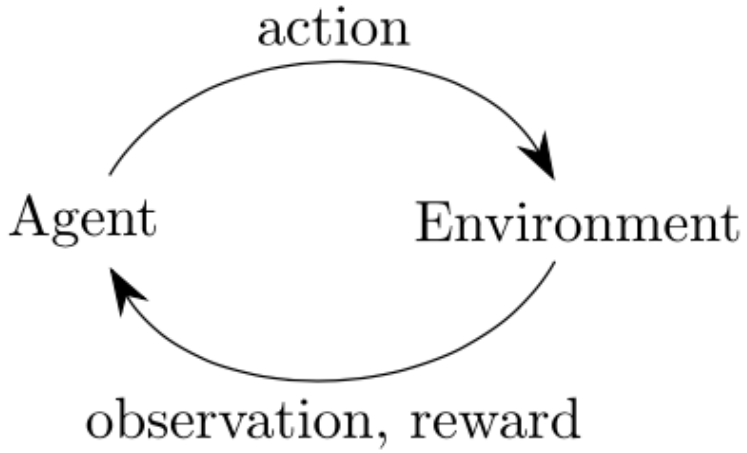
- 앞으로는 더 복잡한 일도 컴퓨터가 처리하는 시대가 올 것이며 강화학습이 점차 많은 역할을 하게 될 것

강화 학습 – 에이전트 (Agent)

- 모델을 agent라고 부름
  - ➢ 모델이 좀 더 능동적으로 학습하는 프로그램이라는 의미

- 지도학습과 달리, 매 배치 입력마다 정답을 확인하고 학습하는 방식이 아니라, 에이전트가 잘 하고 있는지 아닌지 피드백을 받으면서 학습

- agent 는 환경의 상태를 관찰하고(observe) 어떤 행동(action)을 취하면 환경은 이에 대해 보상을 하고 다시 변경된 상태를 알려주는 방식으로 동작



강화 학습 – 보상과 정책

- agent는 누적된 보상이 가장 큰 방향으로 동작
  - 긍정적인 보상(reward), 부정적인 보상(penalty)
  - 최적화 알고리즘을 정책(policy)이라고 함

- 정책
  - agent가 다음에 어떻게 행동하는 것이 바람직한지를 정하는 방식
  - 정책은 확정적인 행동으로 표현할 수도 있고, 확률적인 정책으로 표현될 수도 있다. (예: 어떤 상황에서 확률 90%로 우회전, 확률 10%로 좌회전)
  - 여기서 우회전할 확률 p가 정책 파라미터가 된다.

- 정책 탐색
  - 최적의 정책 파라미터를 찾는 것

강화 학습 – 에이전트

- agent는 초기 정보 없이 학습을 시작할 수 있다. agent 는 주어진 환경에서 처음에는 랜덤한 행동을 하고 그에 따른 보상을 보고 가능한 많은 보상을 받도록 정책을 만든다

- 강화학습은 아무런 사전 지식이 없어도 학습을 할 수 있다는 것이 특징이다. 이는 사람이 직접 체험하면서 학습하는 것과 비슷

- 지도학습에서는 각 행동(입력)마다 “정답”을 받아 학습을 하지만 강화학습에서는 이와 달리 누적된 보상을 더 중요시

- 바둑에서 한 집을 주고 두 집을 얻었다면 처음에 한 집을 주는 것을 지도 학습에서는 손실로 보지만, 강화학습에서는 두 집을 얻기 위한 바람직한 방향으로 처리해야 하며 따라서 강화학습에서는 일반 머신러닝과 다른 최적화 알고리즘이 필요

강화 학습 – 동작 원리

- 강화학습을 수행하는 기본적인 골격은 마르코프 결정 프로세스(MDP, Markov Decision Process)를 이용
  - 상태 states, S
  - 행동 actions, A
  - 보상 Reward , R
  - 정책 Policy, π
  - 가치 Value, V

- 어떤 행동들을 취하는지가 정책을 정하는 것이고, 어떤 보상들이 누적되는지를 가치를 정한다

- 반대로 정책을 따라서 행동을 취한다고도 볼 수 있다. 강화학습의 목적은 보상이 극대화 되도록 최적의 정책을 선택하는 것이다

- 이때 어떤 상태에서 어떤 행동을 하는지에 따라서 개별적인 보상이 다르므로 이들을 모두 누적한 보상을 계산

강화 학습 – DQN

- MDP를 프로그램으로 구현
  - DQN(Deep Q-Network)이 현재 널리 사용

- 어떤 상태에서 특정 행동을 했을 때의 가치를 나타내는 함수인 Q함수를 학습하는 정책기반 알고리즘으로 Q-learning 개념이 1989년에 소개되었음

- DQN(Deep Q-Network)
  - Q-learning을 구현하기 위해서 신경망 기반의 딥러닝으로 구현
  - 알파고를 만든 딥마인드에서 개발
  - 아타리 게임이나, 바둑 프로그램 등으로 성능을 입증

강화 학습 – DQN

- 강화학습에서는 Q함수의 값이 최대가 되는 행동을 취하도록 정책을 수행하는 에이전트를 만드는 것이다

- Q함수는 상태 s에서 행동 a를 지속적으로 수행한 후 얻는 최고 보상 점수이다. Q함수는 아래와 같이 표현된다

$$Q(s_t, a_t) = \max R_{t+1}$$

- 이러한 최상의 보상을 받기 위한 최적의 행동을 선택하기 위한 정책은 다음과 같이 가장 큰 Q함수를 얻는 행동들의 집합으로 정의할 수 있다

$$\pi(s) = \arg\max_a Q(s, a)$$

강화 학습 – DQN

- 즉, 정책은 Q 함수를 극대화하도록 정한다. 달리 표현하면 Q함수는 최적의 정책을 따라서 행동할 때 얻을 수 있는 최대 보상을 말하며, 현재 상태, 행동, 보상, 다음 상태가 주어지면 다음과 같이 Q함수의 현재와 미래의 관계를 정의할 수 있다.

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

- 다음 상태의 Q함수는 현재의 보상(r)과 현재 Q의 최고 값의 합이다. Q 함수는 아래와 같은 과정을 반복적으로 수행하여 도출한다.

  - 현재 상태 s를 관측
  - 현재의 정책에 따라 다음 행동 a를 선택 (e-탐색 등)
  - 행동a의 결과에 따른 보상 r과 변경된 상태 s'를 관찰
  - 현재의 보상r과 다음 단계의 가능한 최대 Q값으로부터 가치 계산
  - 상태 값 갱신

강화 학습 – DQN

- DQN은 두 개의 신경망으로 구성
  - 학습을 하면서 행동을 결정하는 기본 신경망
  - 이 행동이 적절한지 확인하는 목표 신경망
    - 목표 신경망의 기준 값은 기본 신경망으로부터 도출한 결과 값을 일정 주기마다 갱신하여 사용

Q & A