두산 Rokey Boot Camp

스터디 주간 활동 보고서

팀명	벌꿀오소리	**제출자 성명**	송주훈
참여 명단	**정찬원, 임소정, 정민섭, 송주훈**		
모임 일시	**2025년 4월 3일 21시 ~ 22시12분**		
장소	온라인	**출석 인원**	4/6
학습목표	AI 시험 대비 수업 내용 정리 후 발표		

학습내용	송주훈: AI 응용 12차시 강화학습 내용을 정리하여 발표하였다. AI 응용 12차시 강화학습 정리 by 송주훈 목차 1.강화학습이란 2.강화학습 용어 정리 3.Markov 4.벨만 방정식 5.다이나믹 프로그래밍 <1. 강화학습이란> 강화학습: 지도 학습 / 비지도 학습과 다르게 시행착오와 reward를 통해 목표를 찾아가는 기계학습. 가장 유명한 예시로는 알파고가 있다. 보상을 최대화하는 쪽으로 학습하는 방식이라고 생각하면 된다. <2. 강화학습 용어 정리> **state (s)** -현재의 상황을 나타냄 **agent** -행동을 하는 주체 혹은 알고리즘

action (a)

- 에이전트가 선택할 수 있는 의사결정의 단위

reward (r)

- 에이전트가 행동을 취한 후 환경에서 받는 피드백 (점수). 즉각적임

penalty

- 행동이 잘못되었을 때 주는 처벌 (중요한 용어는 아님)

environment

- 에이전트와 상호 작용하는 시스템. 다양한 방식으로 에이전트의 행동에 반응

observation

- 에이전트가 환경에서 가져오는 정보. state의 부분집합 혹은 같은걸로 본다고 하심.

transition probability (P)

- 특정 행동을 취했을 때 다른 상태로 전이될 확률

return(gain) (강의노트에는 G로 나와있음)

- 총 얻는 reward의 합

policy (π)

- agent가 현재 상태에 대해 선택하는 행동의 전략

deterministic policy

- 결정적으로 활동을 고르는 전략 (확률적이 아니고 특정 상태에서 특정 행동을 정함)

stochastic policy

- 확률적으로 활동을 고르는 전략

value function

- 특정 정책에 대해 해당 상태에서 시작하여 기대되는 총 보상의 기대값

state value function (v)

- 각 상태 s에서 특정 정책을 따라 행동했을 때 얻는 보상의 기대값

action value function (q)

- 상태 s에서 특정 행동 s를 취했을 때 그 이후에 정책 π에 의해 얻을 총 보상의 기대값

<3. Markov 마르코프>

마르코프 상태

- 어떤 시스템이 있을 때 현재 시스템이 어떤 상태에 있는 지 나타내는 것. 현재 상태만으로 앞으로 일어날 상황이 결정되는건 아니지만 현재 상태가 중요한 것임. 현재 상태만으로 미래의 상황을 예측.

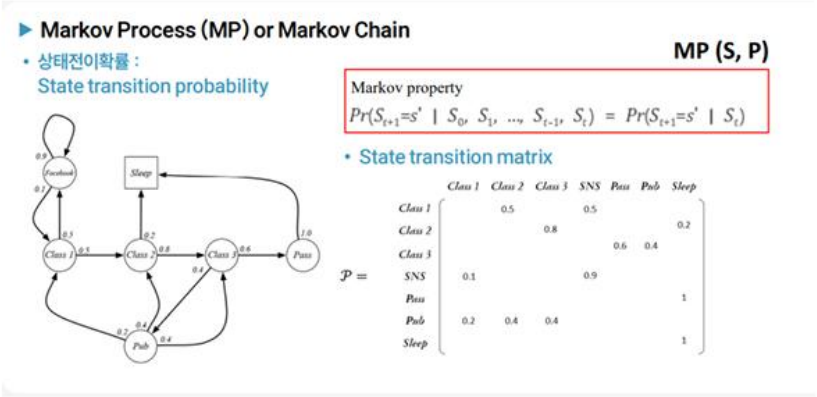
간단한 예시)

1.일반 초등학교 -> 일반 중학교 -> 일반 고등학교 -> 서울대 졸업

2.명문 초등학교 -> 영재 중학교 -> 과학 고등학교 -> 탈선으로 대학안가고 25세 양아치

누가 취업을 잘할까? 물론 과고에서 엄청난 커리어를 쌓았을 수는 있지만 마르코프 상태에선 현재 상태만 가지고 판단하므로 1번이 취업을 더 잘할 것.

마르코프 프로세스



State transition matrix를 통해 현재 상태에서 다른 상태로 전이 확률이 나타남

정민섭 : AI 개론 16차시 Model from Scratch: Resnet18에 대해 발표했다.

스터디_10차시

AI 개론 16차시 - Model from Scratch: Resnet18

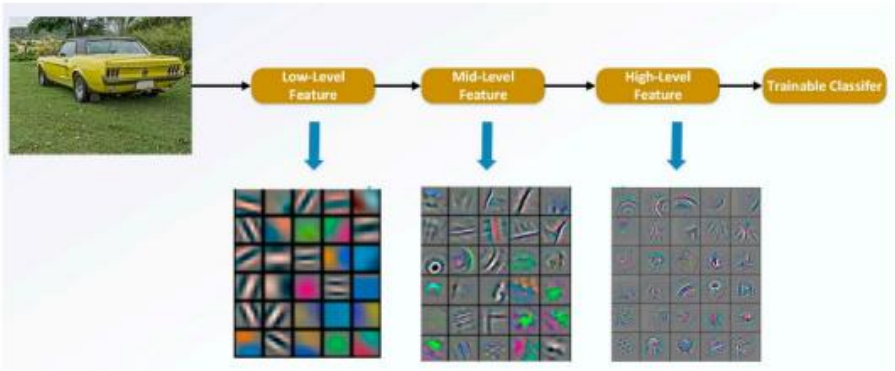
1. 모델 스크래치(Scratch) 학습이란?

- PyTorch의 사전 학습 모델을 사용하지 않고, 모델을 직접 구현하는 방식
- ResNet 아키텍처의 핵심 구성 요소인 `BasicBlock`과 `Bottleneck` 블록을 직접 구현하여 학습하는 과정을 다룬다.
- 잔차 학습(residual learning)의 구조적 의미와 구현 메커니즘을 이해하는 것이 학습 목표!!

핵심 구현 요소는 다음과 같다:

- `BasicBlock`: ResNet-18에 사용되는 기본 블록
- `Bottleneck`: ResNet-50 이상의 깊은 네트워크에 사용되는 확장형 블록
- `ResNet18` class 구현 및 인스턴스 객체 생성

2. Low-Level Feature & High-Level Feature

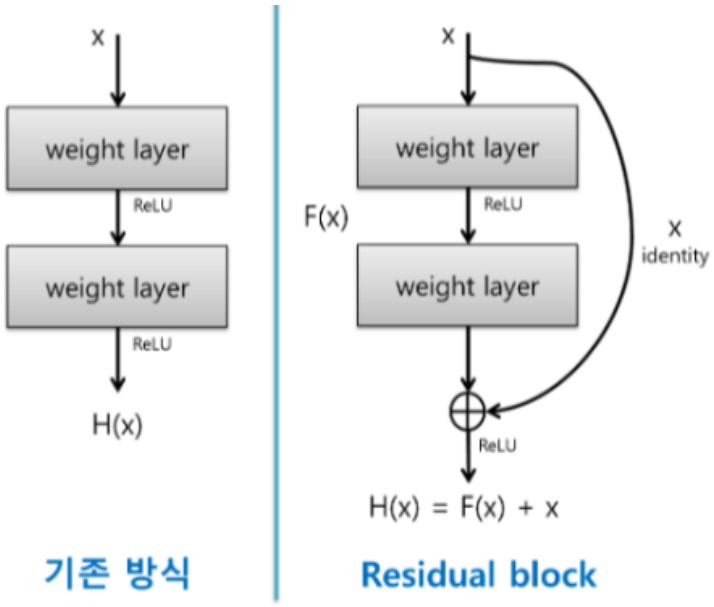


- 입력층이 가까울 수록 지역적인 low feature 추출 ex) Edge, Texture
- 출력층에 가까울 수록 전역적이고 추상적인 high feature 추출 ex) Object, Shape
- 깊은 네트워크일수록 더 복잡한 특징 학습 가능

3. 잔차 학습이란?

잔차 학습(Residual Learning)은 깊은 신경망에서의 성능 저하 문제를 해결하기 위해 제안된 방법이다. 네트워크가 깊어질수록 단순한 선형 쌓기만으로는 학습이 어려워지는 문제(degradation)가 발생하며, 이는 다음 수식으로 재정의된다

```
H(x) = F(x) + x
```



여기서 `F(x)` 는 합성곱 연산 이며, `x` 는 입력값이다. 이를 통해 신경망은 입력과 출력 사이의 변화를 학습하게 되며, 이는 다음과 같은 이점을 있다.

- 역전파 시 기울기 소실 문제 완화

- 더 깊은 네트워크 학습 가능
- 네트워크가 학습하지 않아도 최소한 `identity mapping` 을 보장

BasicBlock 클래스 (ResNet-18용)

```
from typing import Type, List, Optional

class BasicBlock(nn.Module):
  expansion = 1

  def __init__(self, inplanes:int, planes:int, stride:int=1,
               downsample:Optional[nn.Module]=None, groups:int = 1,
               dilation:int = 1, norm_layer:Optional[nn.Module]=None):
    super().__init__()

    ## Normalization layer: Skip connection
    if norm_layer is None:
      norm_layer = nn.BatchNorm2d

    ## First Convutional layer
    self.conv1 = nn.Conv2d(inplanes, planes, kernel_size=3, stride = stride,
                           padding = dilation, groups = groups, bias = False, dilation = di
lation)
    self.bn1 = norm_layer(planes)

    ## Second convolution layer
    self.conv2 = nn.Conv2d(planes, planes, kernel_size=3, stride = 1,
                           padding=dilation, groups = groups, bias = False, dilation = dil
ation)
    self.bn2 = norm_layer(planes)

    ##
    self.relu = nn.ReLU(inplace = True)
    self.downsample = downsample
```

```
    self.stride = stride

  def forward(self, x):
    identity = x

    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)

    out = self.conv2(out)
    out = self.bn2(out)

    if self.downsample is not None:
      identity = self.downsample(x)

    out += identity
    out = self.relu(out)

    return out
```

- 3×3 Conv → BatchNorm → ReLU → 3×3 Conv → BatchNorm → 잔차 연결 → ReLU
- 입력과 출력 채널이 다를 경우 `downsample` 을 통해 정렬

이 구조는 가볍고, 작은 모델(예: ResNet-18, 34)에 적합하다. 계산량이 적고, 간단한 구조로 구성되어 있어 구현 및 실험에 용이하다.

정찬원: AI 응용 3차시 객체검출, 동영상 처리에 대해 발표했다.

3장 OpenCV를 이용한 객체 검출, 동영상 처리

1. OpenCV를 이용한 컴퓨터 비전 III

- 1차 이산함수 미분
  - 전진 차분: 앞의 점과 현재 점을 이용해 미분을 계산.
    - 계산이 간단함.
    - I(x) 이후 값만 사용하므로 비대칭적인 결과가 나올 수 있음.
  - 후진 차분: 현재 점과 이전 점을 이용해 미분 계산. (전진 차분과 방향만 다름)
  - 중앙 차분: 현재 점의 앞뒤 값을 이용해 미분 계산.
    - 양쪽 방향을 고려하기에 계산량이 조금 많을지라도 대칭적이고 정확도가 높음.

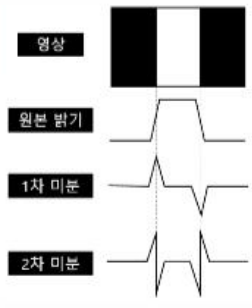
전진 차분 (Forward difference): $\frac{\partial I}{\partial x} \cong \frac{I(x+h)-I(x)}{h}$

후진 차분 (Backward difference): $\frac{\partial I}{\partial x} \cong \frac{I(x)-I(x-h)}{h}$

중앙 차분 (Centered difference): $\frac{\partial I}{\partial x} \cong \frac{I(x+h)-I(x-h)}{2h}$

- 영상과 1차 미분 + 영상 1차 미분을 이용한 엣지 검출

$$\frac{\partial I}{\partial x} \cong \frac{I(x+h)-I(x-h)}{2h}$$

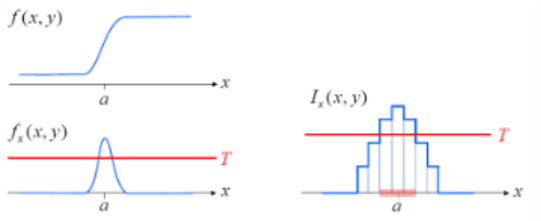


I(x): 픽셀 밝기 값. h: 한 픽셀의 거리.

원본 밝기: 흰색 (높은 밝기), 검은색(낮은 밝기)

1차 미분: 밝기 변화 급격한 지점(edge)에서 큰 값이 나타나서 엣지 검출 가능.

2차 미분: 1차 미분 변화가 극대가 되는 지점에서 0이 되며, 엣지의 정확한 위치를 찾는데 도움이 됨.

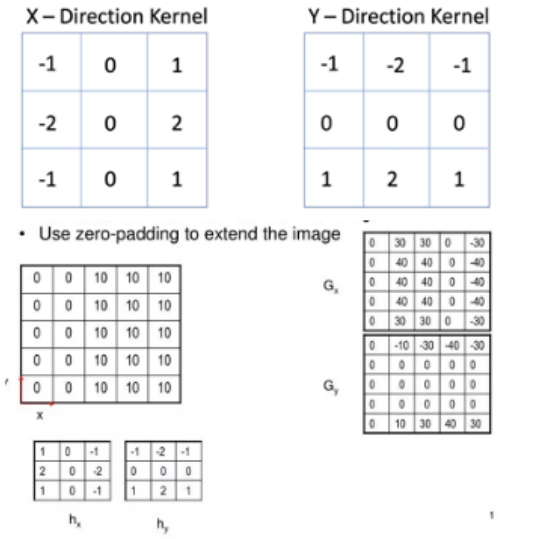


첫 번째 그래프: 영상의 밝기 분포를 나타내며 a 지점에서 급격한 밝기 변화 발생(엣지 존재)

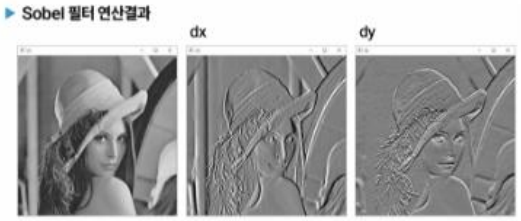
두 번째 그래프: f(x,y)를 1차 미분하여 엣지가 있는 곳에서 미분 값이 커짐. 이때 T는 임계값으로 미분 값이 T보다 크면 edge라고 판단.

세 번째 그림: 디지털 이미지의 미분: 히스토그램은 이산적인 미분 값을 의미하며 역시 T를 초과하는 부분은 edge로 판단.

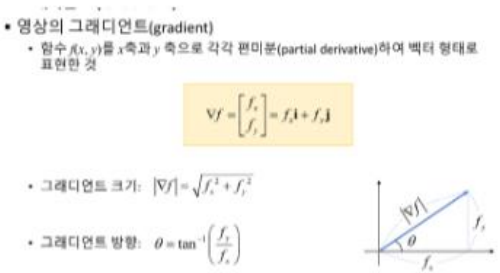
- **특징 검출: Sobel edge**
  - 중앙 차분 기반
  - 방향별 필터 적용(수평, 수직 따로)하여 각 방향의 기울기(Gx, Gy)를 구할 수 있음.



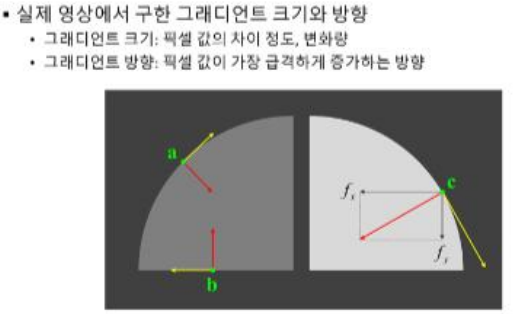
- 노이즈 억제 효과
  - 가우시안 필터와 미분을 함께 수행하여 noise에 강함.
  - 중심 계수를 2배로 두어 주변 픽셀보다 중심 픽셀 영향을 더 크게 반영.



- gradient: 각 방향의 기울기를 구한 후, 엣지 강도를 계산하여 크기가 클수록 강한 엣지를 의미함.



- 엣지의 크기와 방향을 구할 수 있으며 간단하면서 효과적인 엣지 검출이 가능함.



- sobel 필터만 사용하면 약한 엣지는 검출되지 않을 수 있고 방향을 고려하지 않으면 edge가 흐릿해질 수 있음.

- Canny edge

임소정: AI 개론 2주차에 대해 정리하여 발표했다.

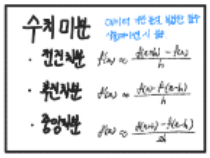
3기 컴퓨터비전 개론 2주차 요약본

1. 인공지능 학습

1.1 합성 함수

- **합성 함수란?**
  - 두 함수 f(x)와 g(x)가 있을 때, 한 함수의 출력값을 다른 함수의 입력값으로 사용하는 방식
  - 수학적 표현 및 예제

$$h(x) = f(g(x))$$



$$f(x) = 2x + 3, \quad g(x) = x^2$$

$$z(x) = f(g(x)) = 2x^2 + 3$$

  - 딥러닝에서는 여러 개의 층을 쌓아, 합성 함수를 구성하여 복잡한 문제를 해결

1.2 경사 하강법(Gradient Descent)

- **경사 하강법이란?**
  - 머신러닝 및 최적화 문제에서 널리 사용되는 알고리즘
  - 비용 함수(Loss Function)를 최소화하는 방향으로 가중치를 업데이트
  - 신경망 학습 과정에서 필수적인 최적화 기법
- **경사 하강법의 단계**
  1. **초기값 설정**: 파라미터(가중치)를 임의로 초기화
  2. **기울기 계산**: 현재 위치에서 함수의 기울기(미분값)를 계산
  3. **파라미터 업데이트**: 기울기의 반대 방향으로 이동하여 최적값을 찾음
  4. **반복**: 수렴할 때까지 위 과정을 반복

★ 학습의 power 는 숫자!

비용함수의 기울기

$$\theta_{n+1} = \theta_n - \alpha \nabla_\theta J(\theta_n)$$

θ: 파라미터 (예: 가중치).

α: 학습률 (learning rate). ← 최적화 속도 조절, 너무 크면 발산, 너무 작으면 느림, 최적값 도달 오래 걸림 계산

$\nabla_\theta J(\theta)$: 비용 함수 $J(\theta)$의 기울기.

- **경사 하강법의 원리**

  → 기울기 방향을 따라 내려가면서 손실을 줄이는 과정

- **학습률 (lr: Learning Rate)**
  - 학습률 α 값이 너무 크면 **Overshooting** (최적점을 지나침) 발생
  - 학습률 α 값이 너무 작으면 수렴 속도가 느려짐

  일반적으로 0.01 ~ 0.1

1.3 경사 하강법의 종류

Q. 경사 하강법의 변화에 해당하지 않는 것? 정답: Logistic Regression

- **배치 경사 하강법 (Batch Gradient Descent)**
  - 전체 훈련 데이터를 한 번에 사용하여 기울기를 계산
  - 장점: 전체 데이터를 사용하여 안정적인 학습 가능
  - 단점: 데이터가 많을 경우 계산 비용이 큼
- **확률적 경사 하강법 (Stochastic Gradient Descent, SGD)**
  - 훈련 데이터에서 하나의 샘플을 랜덤하게 선택하여 기울기를 계산
  - 장점: 계산 속도가 빠르고, 메모리 요구량이 적음
  - 단점: 매 스텝마다 최적화 방향이 바뀌어 학습이 불안정할 수 있음
- **미니배치 경사 하강법 (Mini-batch Gradient Descent)**
  - 훈련 데이터를 작은 배치 단위로 나누어 각 배치에서 기울기를 계산
  - 장점: 배치 경사 하강법의 안정성과 SGD의 속도를 조합
  - 단점: 배치 크기 선택에 따라 성능이 달라질 수 있음
- Momentum

1.4 경사 하강법의 장단점

- 장점

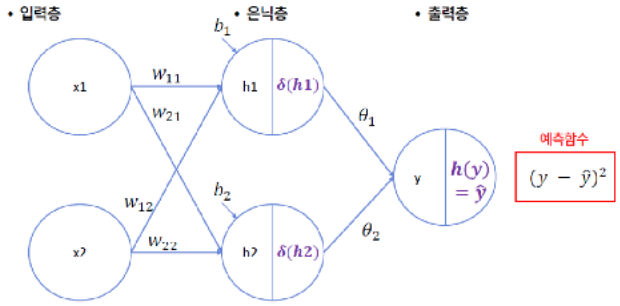
- 단순하고 직관적이며 구현이 쉬움
- 선형 회귀, 로지스틱 회귀, 신경망 등 다양한 모델에 적용 가능

- 단점
  - 지역 최솟값(Local Minimum)에 빠질 가능성
  - 학습률 설정에 따라 수렴 속도와 학습 성능이 달라짐 / 너무 크면 발산 가능성 O
  - Saddle point 정체, 에포크 부족 문제

2. 예측 함수(Object Function) 정의하기

2.1 예측 함수(Object Function)

- **예측 함수란?**
  - 입력층(Input Layer) → 은닉층(Hidden Layer) → 출력층(Output Layer) 순서로 구성
  - 입력값(x)과 가중치(Weight), 바이어스(Bias)를 조합하여 출력(y)을 계산
  - 뉴런 간의 연결을 통해 모델이 패턴을 학습하는 역할 수행



2.2 손실 함수(Loss Function)

- **손실 함수란?**
  - 개별 데이터 포인트 또는 샘플에 대한 오류를 측정하는 함수
  - 모델이 특정 데이터를 얼마나 잘 예측하는지 평가하는 역할 수행

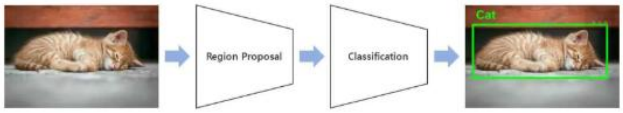
Q 6-10 클래스 불균형 문제 해결하기 가중치를 적용한 손실함수?
⇒ Focal Loss or Weighted Cross-entropy

강인우: 스터디에 참여하진 못했으나 객체 검출과 관련하여 자료를 조사하였다.

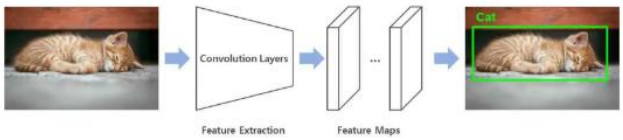
4.03 스터디

17차시. 객체검출 I (Two stage detector)

객체검출 모델 분류



(a) 2-Stage detector



(b) 1-Stage detector

Two-Stage Detector

객체 검출이 Region Proposal + Classification 2단계로 이뤄짐

- R-CNN (2014)
- Fast R-CNN
- Faster R-CNN
- Masked R-CNN (Instance segmentation)

One-Stage Detector

CNN 기반으로 객체 위치를 바로 예측

- Yolo series

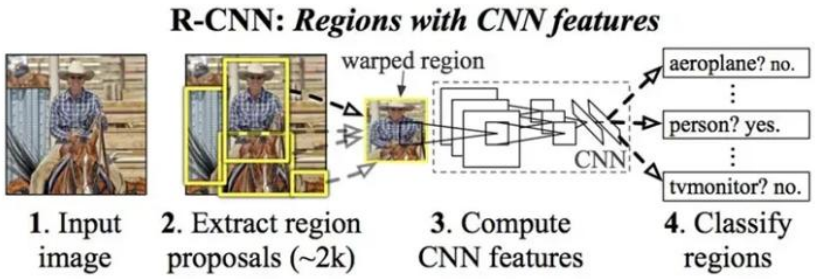
-Yolo1 ~Yolo11 (Object detection)

- SSD series

- SSD
- RetinaNet

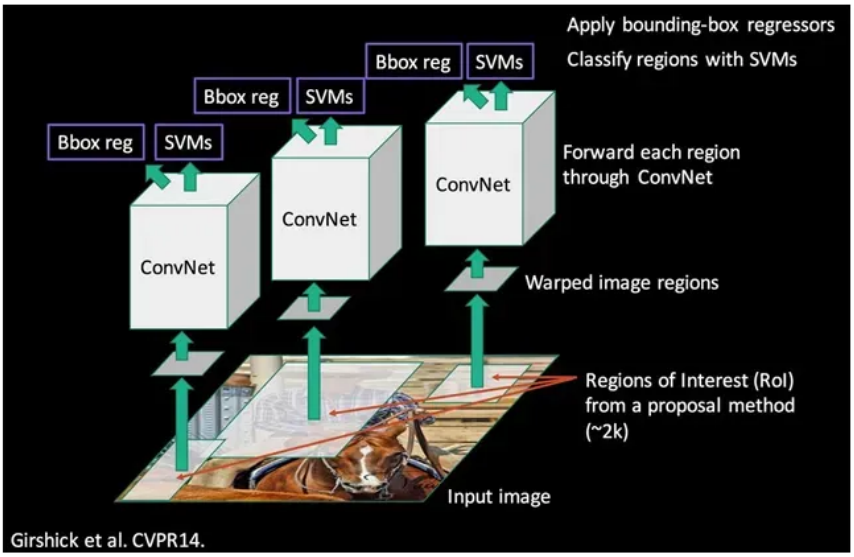
*최근에는 One-Stage Detector 모델이 정확도와 속도 모두 우수함

R-CNN



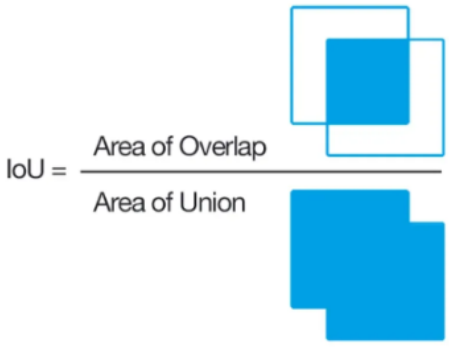
CNN을 이용한 최초의 객체 검출 모델

진행과정 요약:



1. 입력 이미지 (Input Image)
2. RoI 생성 (Regions of Interest from a proposal method ~2k)
3. RoI 크기 변환 (Warped Image Regions)
4. RoI를 ConvNet에 통과 (Forward each region through ConvNet)
5. SVM을 이용한 객체 분류 (Classify regions with SVMs)
6. Bounding Box Regressor를 적용하여 바운딩 박스 보정 (Apply bounding-box regressors)

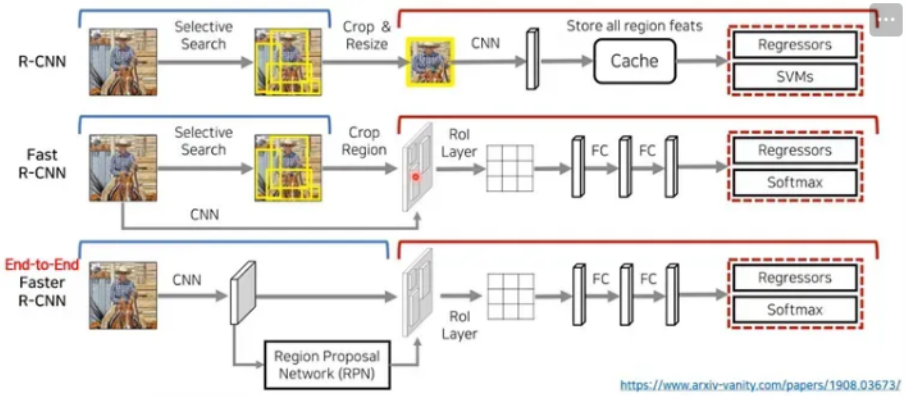
*IoU; Intersection over Union



$$\text{IoU} = \frac{\text{겹치는 영역의 넓이 (Intersection)}}{\text{전체 합집합 영역의 넓이 (Union)}}$$

- NMS 중복 제거, 객체 검출 모델에서 성능 평가를 하는 과정 등에서 사용

Fast R-CNN & Faster R-CNN



Fast R-CNN : 전체 이미지에 대해 CNN을 한 번만 수행하고, ROI Pooling을 통해 각 객체 후보 영역을 공통 feature map에서 효율적으로 추출함 (CNN 연산을 공유한다)

Faster R-CNN: Selective Search를 대신하는 RPN

ROI Pooling (Region of Interest Pooling)

ROI마다 CNN을 다시 수행하지 않고, 한 번 만든 feature map에서 Pooling으로 뽑자

⇒ 다양한 크기의 객체 후보 영역(Region Proposal)을 FC에 입력가능한 고정된 크기로 변환

RPN

feature map 상에서 anchor들을 기반으로 객체 후보 영역(Region Proposal) 을 예측하는 CNN 기반의 네트워크.

기존의 Selective Search에 비해 속도와 정확도 모두에서 크게 향상

동작과정

- **Anchor Target Generation**
  - 여러 크기의 Anchor Box를 생성하여 객체 위치 예측
- **GT(Ground Truth) 박스와 IOU(Intersection Over Union) 계산**
  - 실제 객체의 위치와 예측된 Anchor Box 간의 IOU 값을 계산하여 최적의 박스를 선택
- **Proposal 생성**
  - 최적화된 후보 영역을 선별하여 객체 검출 수행

활동평가	금일 스터디를 통해 AI 시험 대비를 할 수 있었으며 모르는 부분을 서로 채워 나갈 수 있었다.

과제	x
향후 계획	4월 2주차 스터디 휴식 4월 3주차 스터디: ROS2 패키지 탐색 후 실행해보기
첨부 자료