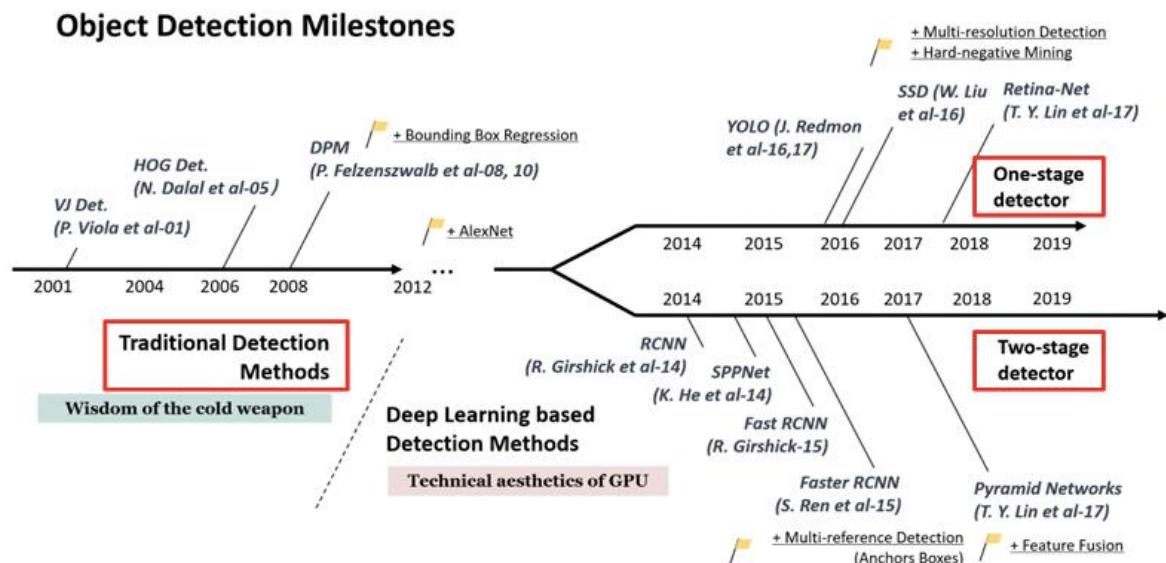


스터디 10주차 R-CNN 정리본

홍송은

Object Detection Milestones

1. 객체 검출(Object Detection)의 역사 및 흐름



Zou et al. 2019. Object Detection in 20 Years: A Survey

전통적 객체 검출 (Traditional Detection Methods)

- 2001~2012년경까지 방식:
 - Viola-Jones Detector (2001)
: Haar-like features + AdaBoost + cascade 구조
 - HOG (Histogram of Oriented Gradients) + SVM (2005)
: 물체의 (edge) 방향을 특징으로 추출
 - DPM (Deformable Part-based Model, 2008)
: 물체를 부분으로 분해하여 모델링

→ 주로 handcrafted feature 기반, 학습 없이 규칙 기반 분류가 주류를 이룸

딥러닝 기반 객체 검출 (Deep Learning-based Detection Methods)

→ CNN 기반 특징 추출기 사용, One-Stage Detector와 Two-Stage Detector 구조

2. One-Stage Detector vs Two-Stage Detector

<One-Stage Detector>

- 단일 CNN 구조에서 위치와 클래스 동시에 예측
- Region Proposal 단계 생략 \Rightarrow 바로 bounding box와 class를 예측
- 대표 모델:
 - YOLO (You Only Look Once) 시리즈: 실시간 속도에 초점
 - SSD (Single Shot MultiBox Detector)
 - RetinaNet: Focal Loss로 클래스 불균형 문제 보완

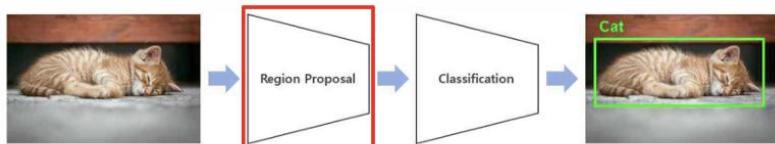
➔ 속도가 매우 빠르고 실시간에 적합
➔ 복잡한 배경/작은 객체에는 성능 저하 가능성 존재

<Two-Stage Detector>

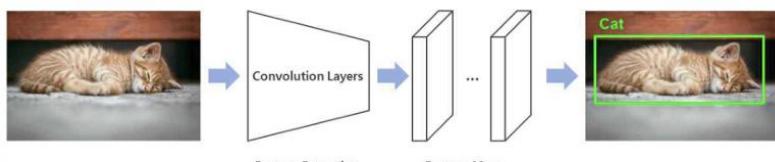
- 1단계 (Region Proposal)
: 영상 내 객체가 있을 법한 후보 영역을 제안 (e.g., RPN)
- 2단계 (Classification & Bounding Box Regression)
: 각 영역에 대해 분류 및 박스 정교화
- 대표 모델:
 - R-CNN (2014): Region Proposal + CNN + SVM
 - Fast R-CNN (2015): 하나의 CNN으로 전체 이미지를 처리, ROI Pooling 사용
 - Faster R-CNN (2015): Region Proposal을 CNN 안으로 통합 (RPN 사용)
 - Mask R-CNN (2017): Faster R-CNN에 segmentation 마스크 예측 추가

➔ 정확도는 높지만 속도가 느리고 복잡도 높음

<요약>



(a) 2-Stage detector



(b) 1-Stage detector

(a) Two-Stage Detector 구조:

1. 전체 이미지 입력
2. Region Proposal 네트워크가 후보 영역 생성
3. 각 후보 영역을 분류 및 박스 조정
4. 최종적으로 객체 분류 + 경계 상자 출력

→ 예: Faster R-CNN

(b) One-Stage Detector 구조:

1. 이미지 입력 → CNN 통과
2. CNN의 feature map에서 동시에 위치 + 클래스 예측

→ 예: YOLO, SSD

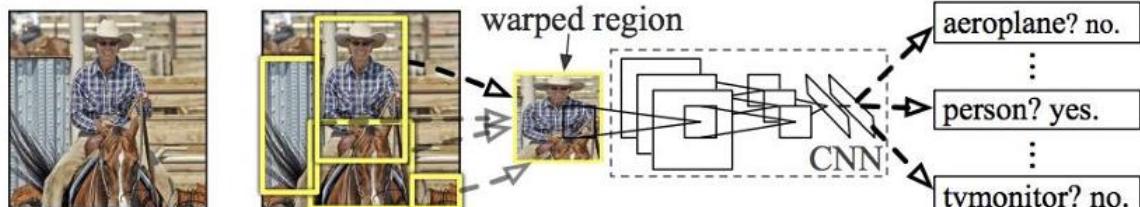
구분	대표 모델	특징
Two-Stage Detector	R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN	높은 정확도, 느린 속도, Region Proposal 단계 존재
One-Stage Detector	YOLO, SSD, RetinaNet	빠른 속도, 실시간 적용 가능, 구조 단순

R-CNN (Regions with CNN Features)

1. R-CNN이란?

1. 입력 이미지
2. Region Proposal (~2,000개) 생성
 - Selective Search를 이용해 객체가 있을 법한 후보 영역을 생성
3. CNN 특징 추출 (e.g., AlexNet)
 - 각 후보 영역을 고정 크기로 resize한 후 CNN을 통과시켜 feature vector 추출
4. SVM 분류기로 분류
 - 추출된 feature vector를 기반으로 객체 클래스를 분류
5. Bounding Box Regression으로 bounding box 닦기
 - 초기 제안된 영역을 실제 객체에 맞게 위치·크기 보정

R-CNN: Regions with CNN features



1. Input image

2. Extract region proposals (~2k)

3. Compute CNN features

4. Classify regions

<Region Proposal (Selective Search)>

- 목적: 전체 이미지에서 객체가 있을 법한 위치만 추려냄 (2,000개 내외)
- 방법:
 1. 이미지 분할 (Felzenszwalb 그래프 기반)
 2. 유사 영역끼리 병합 (Greedy 방식)
 3. 후보 영역 생성
- 장점: 비교적 빠르고 모든 객체를 잘 포괄함
- 단점: Selective Search는 학습이 불가능하고 느리다

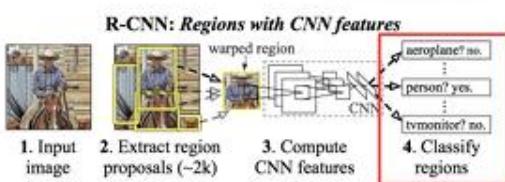
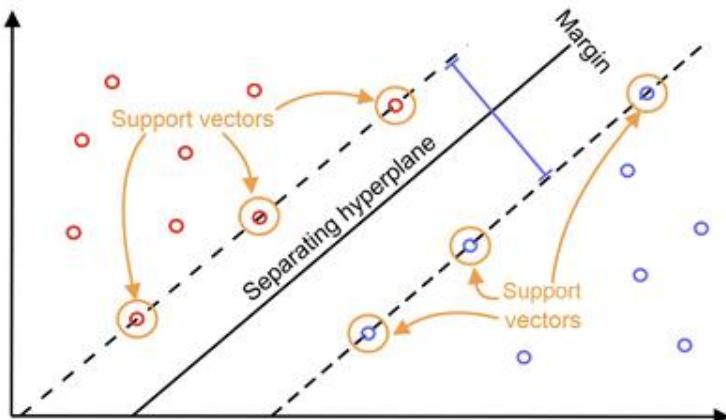
<CNN 특징 추출 (AlexNet 기반)>

- 입력된 후보 영역을 CNN에 통과시켜 고차원의 feature vector 추출
- R-CNN에서는 AlexNet(2012)을 backbone으로 사용
- 마지막 Fully Connected Layer에서 4096차원의 feature vector 생성
- 해당 벡터는 이후 분류(SVM)와 회귀(Bbox reg)에 사용됨

<SVM Classification>

- CNN으로 추출한 feature vector를 SVM 분류기에 입력하여 객체 분류 수행
- 각 클래스별로 개별 SVM이 존재
- 이진 분류를 통해 "이 영역이 특정 클래스인가?"를 판별

→ CNN은 특징을 뽑는 도구이고, SVM은 분류 도구로 활용됨 (ML 기반 분류)



6. Bounding Box Regression

- Region Proposal의 경계 상자가 정확하지 않기 때문에 보정 필요
- Bounding Box Regression은 CNN 특징 벡터를 입력받아 상자 위치를 보정
- 위치를 중심 좌표 및 너비/높이 기준으로 조정

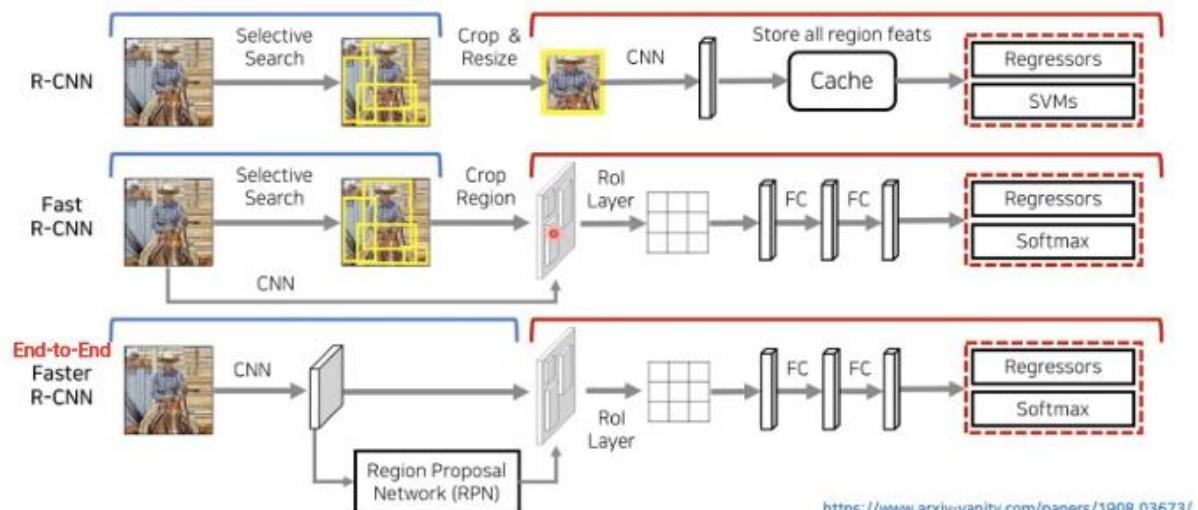
<문제점 및 한계>

항목	문제점
속도	Selective Search + CNN 개별 추출로 매우 느림 (2,000 region * CNN)
End-to-End 불가	Region Proposal과 분류기가 따로 분리되어 학습 불가능 참고) End-to-End 학습? <ul style="list-style-type: none"> - 입력부터 출력까지의 모든 과정을 하나의 네트워크로 연결하고, 전체를 한꺼번에 학습하는 방식 - 현재 CNN → SVM → BBox regressor가 모두 따로따로 학습됨 → 전체 파이프라인에서 오직 CNN만 딥러닝이고, 그 결과는 고정된 feature로 사용됨. 파라미터들이 서로 영향을 주고받으며 공동으로 학습되지 않음
공간 중복	많은 영역이 겹침에도 CNN을 계속 반복 사용

R-CNN 계열 정리: R-CNN → Fast R-CNN → Faster R-CNN

1. 전체 발전 흐름

모델	Region Proposal 방식	속도	특징 요약
R-CNN	Selective Search	매우 느림 (~50초)	각 region을 개별적으로 CNN 통과 → 비효율
Fast R-CNN	Selective Search	개선됨 (~2초)	전체 이미지를 CNN 통과 후 ROI Pooling 수행
Faster R-CNN	RPN (Region Proposal Network)	매우 빠름 (~200ms)	Region Proposal도 CNN에 통합 → End-to-End 학습 가능



2. R-CNN (2014)

- 과정
 1. Selective Search로 2,000개 region proposal 생성
 2. 각 region을 CNN (AlexNet 등)에 개별 입력해 feature 추출
 3. 추출된 feature를 SVM으로 분류
 4. Bounding Box Regression으로 위치 보정
- 한계
 - 각 proposal마다 CNN을 반복 수행 → 매우 느림
 - Multi-stage pipeline → End-to-End 학습 불가능
 - Region proposal은 고정 알고리즘 (학습 불가)

3. Fast R-CNN (2015)

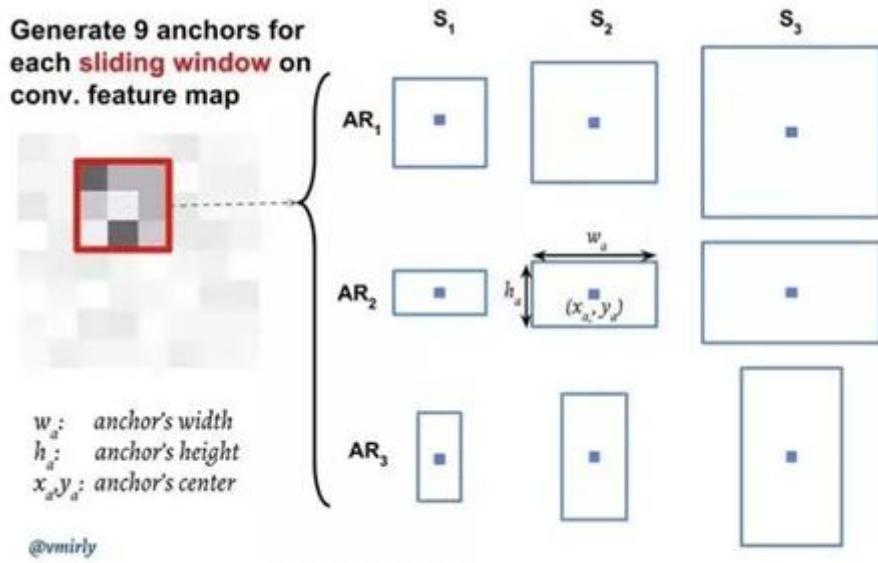
- 개선점
 - 전체 이미지를 먼저 CNN에 통과시켜 feature map 생성
 - Region proposal은 Selective Search 사용하지만,
 - 각 proposal에 대해 CNN을 다시 수행하지 않고, feature map에서 ROI만 뽑아 사용
 - ROI Pooling으로 고정 크기 feature 추출 → FC + Softmax + Regressor 연결
- 장점
 - CNN 1회만 수행 → 속도 매우 향상
 - 모든 구성 요소를 End-to-End로 학습 가능
- !!! 여전히 남은 문제
 - Region Proposal 단계(Selective Search)는 CNN 외부에 있음
→ 느리고 학습 불가능

4. Faster R-CNN (2015)

- 핵심 변화: Region Proposal마저도 CNN 내부로 통합한 구조
- 구성
 1. CNN으로 feature map 생성
 2. Feature map 위에서 RPN (Region Proposal Network) 수행
 - sliding window 방식
 - Anchor Box 개념 사용 (고정 크기 & 비율의 박스)
 - 각 위치에서 다수의 anchor를 예측 (보통 9개)
 - 각 anchor에 대해 objectness score + 박스 보정
 3. 예측된 ROI를 ROI Pooling → FC → Softmax + BBox reg
- 장점
 - 완전한 End-to-End 구조
 - Region Proposal도 학습 가능
 - 연산 효율 대폭 향상

<RPN (Region Proposal Network)>

- Anchor 기반 제안 방식
 - feature map의 각 위치마다 다양한 크기/비율의 anchor box를 설정
 - 각 anchor에 대해:
 - object인지 아닌지 (classification)
 - 위치 조정값 (regression) 예측
- Anchor 예시
 - 3가지 크기 × 3가지 비율 → 총 9개 anchor
 - 각 anchor는 ground truth와의 IOU를 계산하여 학습 대상 선정



<Non-Maximum Suppression (NMS)>

- 역할: 겹치는 박스 중 점수가 높은 것만 남기고 나머지를 제거
 - 단계
 1. IOU 계산: $\text{IOU} = \frac{A \cap B}{A \cup B}$
 2. 겹침이 큰 박스 중 하나만 선택 (보통 $\text{IOU} > 0.5$ 이면 제거)
 3. 중복 박스 제거 후 최종 객체 위치 선택
- Faster R-CNN에서는 RPN, 마지막 결과 등 여러 단계에서 적용됨