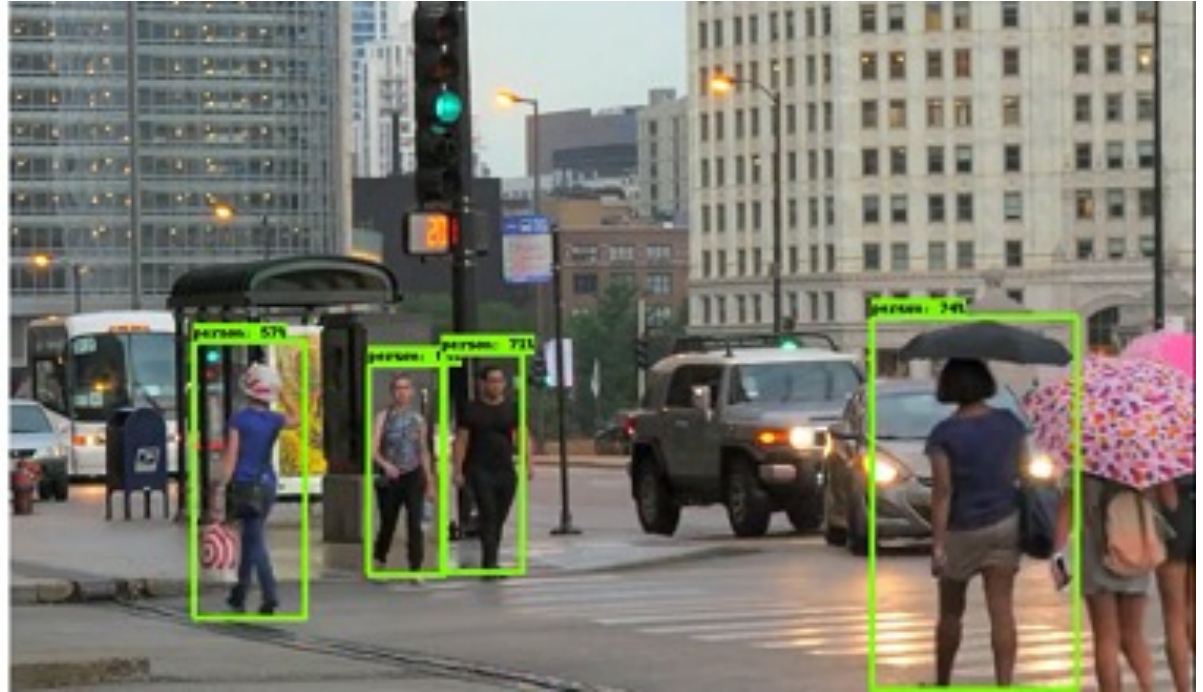


Object Detection

2024 Summer URP

이후 URP에서 경험할 task

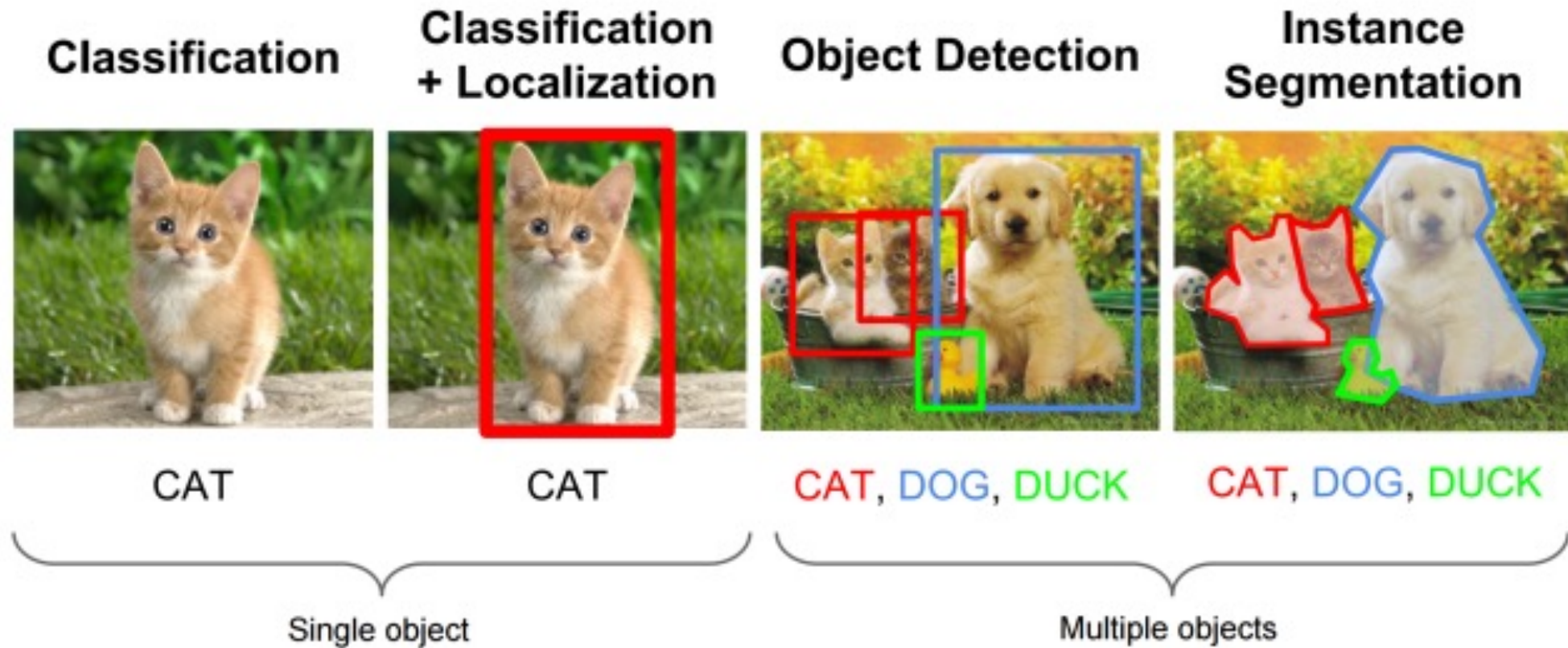
- 이후 컴퓨터 비전의 대표적인 task 중 하나인 **object detection**을 활용한 보행자 검출(**pedestrian detection**)을 진행할 예정



Object Detection

컴퓨터 비전의 다양한 task

- **Object Detection** : 이미지 내에서 객체(object)의 종류(class)와 위치(location)을 찾는 것
 - 위치(location)은 bounding box의 좌표값(x, y, w, h)을 통해 예측



Classification vs Object Detection

- 사물 탐지는 물체의 클래스 분류뿐만이 아닌 물체의 위치와 크기를 예측하기
 분류+회귀를 함께 사용하는 task

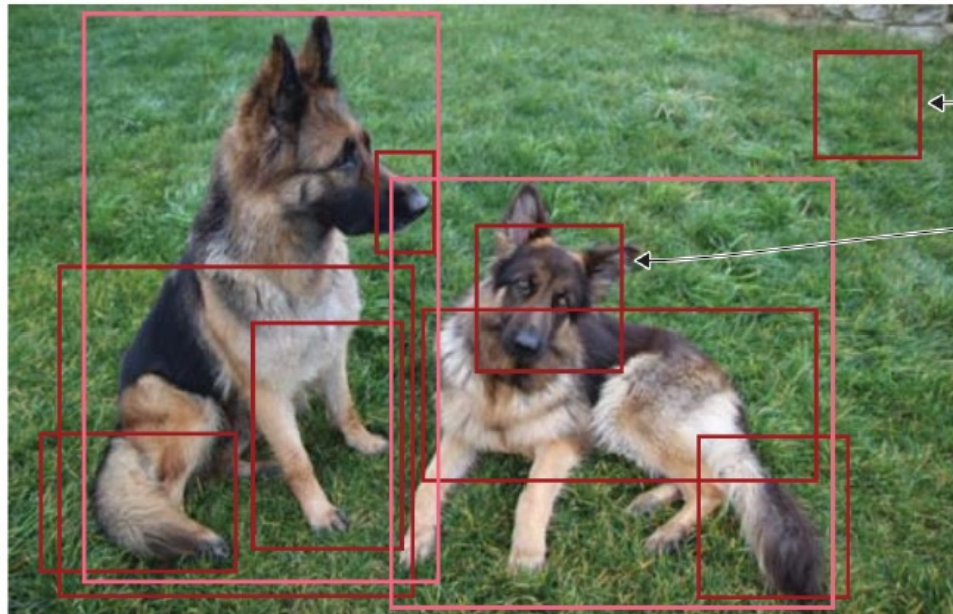
이미지 분류	사물 탐지
<p>이미지에 있는 (하나뿐인) 대상의 클래스를 예측하는 것이 목표</p> <ul style="list-style-type: none"> ▪ 입력: 하나의 대상이 있는 이미지 ▪ 출력: 클래스 레이블 (고양이, 개, 등) ▪ 출력 예: 각 클래스에 속할 확률 (고양이 84%) 	<p>이미지에 있는 (하나 이상의) 대상의 위치를 나타내는 박스와 해당 물체의 클래스를 예측하는 것이 목표</p> <ul style="list-style-type: none"> ▪ 입력: 하나 이상의 대상이 있는 이미지 ▪ 출력: 각 대상의 위치를 특정하는 경계박스(좌표)와 해당 물체의 클래스 레이블 ▪ 출력 예: <ul style="list-style-type: none"> - box1의 좌표(x,y,w,h)와 클래스에 속할 확률 - box2의 좌표와 클래스에 속할 확률 <p>좌표(x,y,w,h)에서 x와 y는 경계 박스의 중심점 좌표고, w와 h는 경계 박스의 너비와 높이이다.</p>

Object Detection Framework

- Object Detection 알고리즘의 일반적인 프레임워크
 1. **영역 제안(region proposal)** : 이미지에서 처리할 영역인 RoI를 제안하는 알고리즘
 2. **특징 추출 및 예측** : 각 box 영역의 시각적 특징 추출
 3. **비최대 억제(NMS)** : 같은 물체에 대한 중복된 박스를 처리하여 박스 하나만 남김

Object Detection Framework : region proposal

- **영역 제안(region proposal)** : 이미지에서 시스템이 처리할 영역인 RoI (Region of Interest. 관심 영역)을 제안하는 알고리즘
 - **RoI** : 이미지 내 물체가 존재할 것이라 예상되는 영역
 - 영역 제안을 통해 물체의 존재 확신도(objectness score)가 매겨진 많은 수의 box 정보 출력
 - Objectness socre가 낮은 영역은 추가 분석 없이 영역 정보 폐기



물체 존재 확신도가
낮음 (배경)

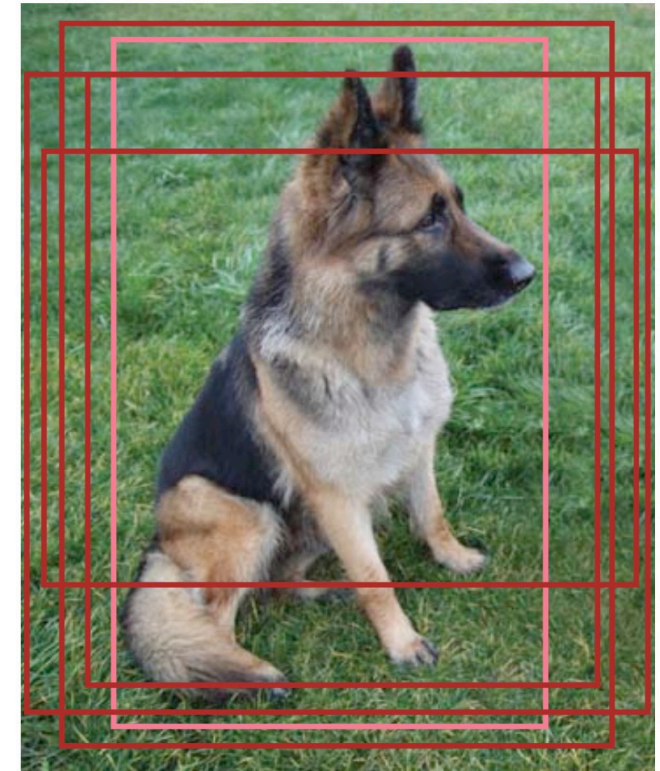
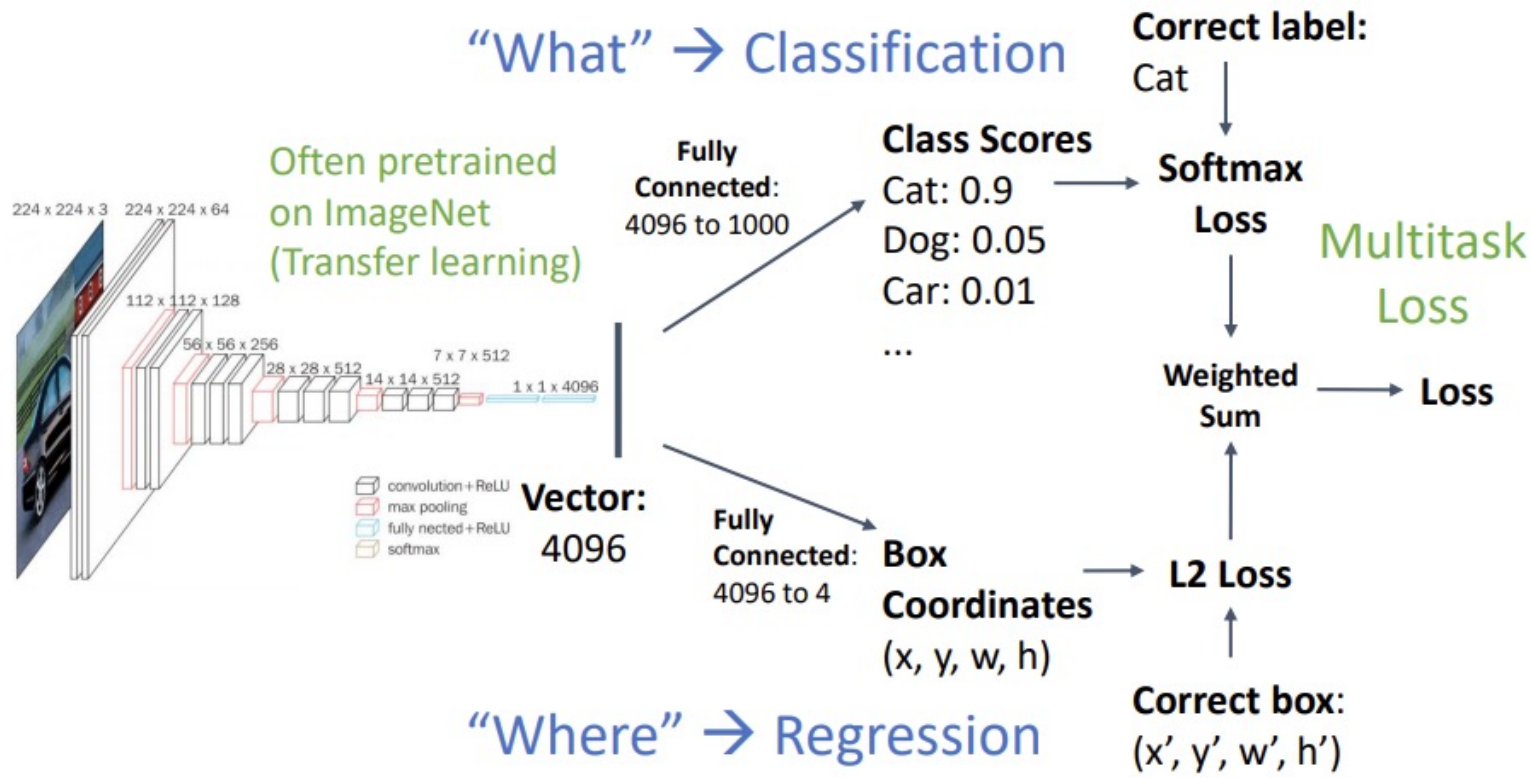
물체 존재 확신도가
높음 (전경)

Object Detection Framework : region proposal

- 딥러닝 모델은 objectness score에 따라 사전 설정된 임계값(threshold)을 초과하는 영역을 **전경(foreground)**으로 간주하고, 나머지는 **배경(background)**으로 설정
 - 임계값은 데이터셋에 따라 적절하게 조절됨
- **임계값이 너무 낮으면** -> 약간이라도 물체인 것 같은 모든 영역을 제안. 물체를 놓칠 가능성이 줄어들지만, 물체가 아닌 배경에 대해서도 오검출을 할 수 있고 계산 원을 지나치게 소모
- **임계값이 너무 높으면** -> object score가 높은 확실한 물체만 탐지하므로 물체를 놓칠 수 있음

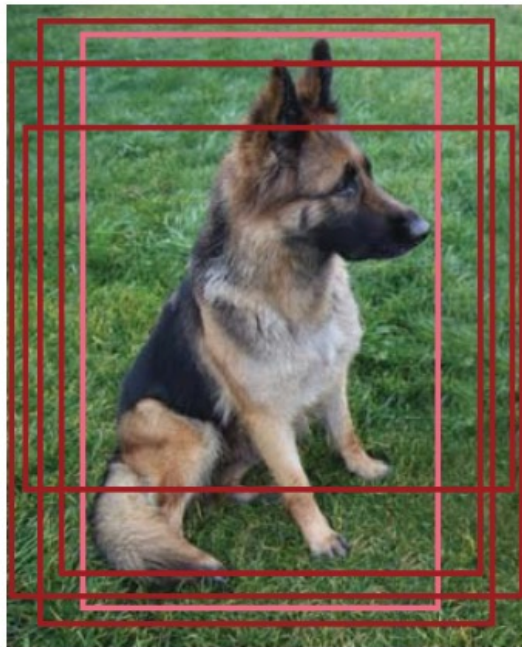
Object Detection Framework : 특징 추출 및 예측

- **특징 추출 및 예측** : 각 box 영역의 시각적 특징이 추출됨. 이러한 특징을 평가해서 물체 존재 여부와 물체의 클래스 판단



Object Detection Framework : 비최대 억제(NMS)

- 비최대 억제(Non-maximum suppression. NMS) : 사물 탐지 모델의 예측 결과, 동일한 물체에 대해 복수의 박스를 탐지했을 가능성이 높으므로 동일 물체에 대한 중복된 예측 박스를 통합하여 물체 하나 당 하나의 박스만 남도록 처리
 - 예측 확률이 가장 높은 경계 박스만 남기고 나머지는 배제



Predictions before NMS

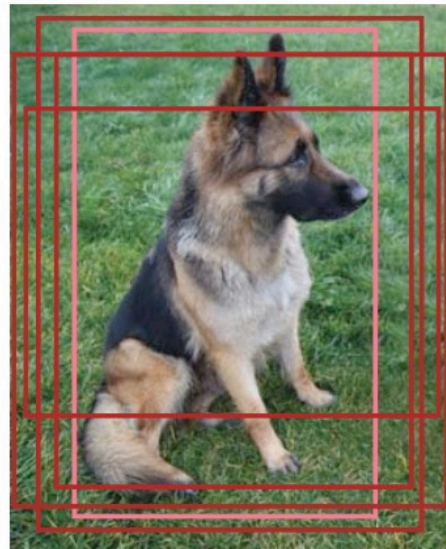


After applying non-maximum suppression

Object Detection Framework : 비최대 억제(NMS)

NMS 작동 과정

1. 예측 확률(predicted probability)이 미리 설정된 신뢰 임계값(confidence threshold)에 미치지 못하는 경계 박스 폐기
2. 남아 있는 경계 박스를 하나씩 살펴본 뒤 예측 확률이 가장 높은 것 선택
3. 선택된 박스와 예측 클래스가 같은 박스의 중복 영역 계산(중첩률. IoU)
4. 중첩률이 0인 박스는 남겨두고 중첩률이 미리 설정한 임계값(NMS임계값)보다 큰 박스는 폐기 -> 중첩률이 높은 것은 동일한 물체에 대한 예측이므로



Predictions before NMS

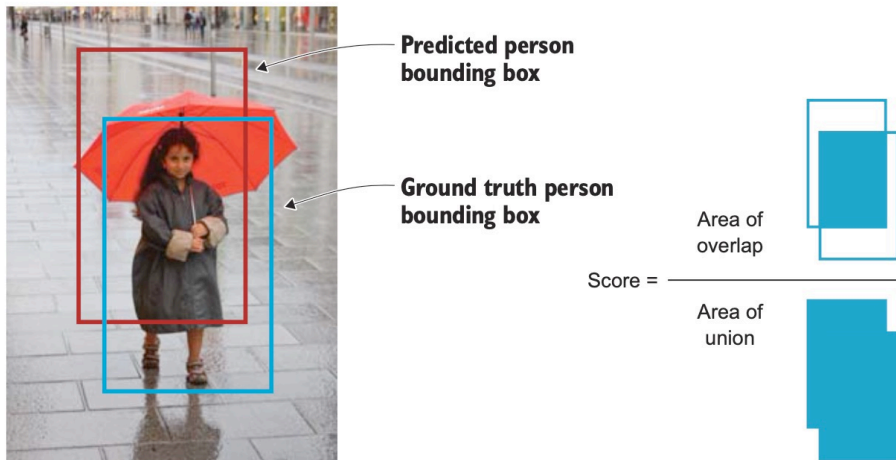


After applying non-maximum suppression

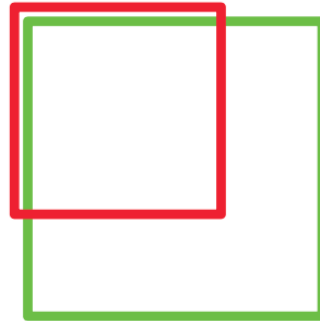
Object Detection 평가 지표

사물 탐지 성능 평가 지표

- 초당 프레임 수(frames per second. FPS) : 탐지 성능 평가 지표
- mAP(mean average precision) : 신경망 예측 정밀도 측정

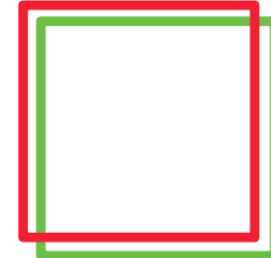


IoU: 0.4034



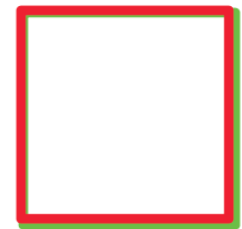
Poor

IoU: 0.7330



Good

IoU: 0.9264



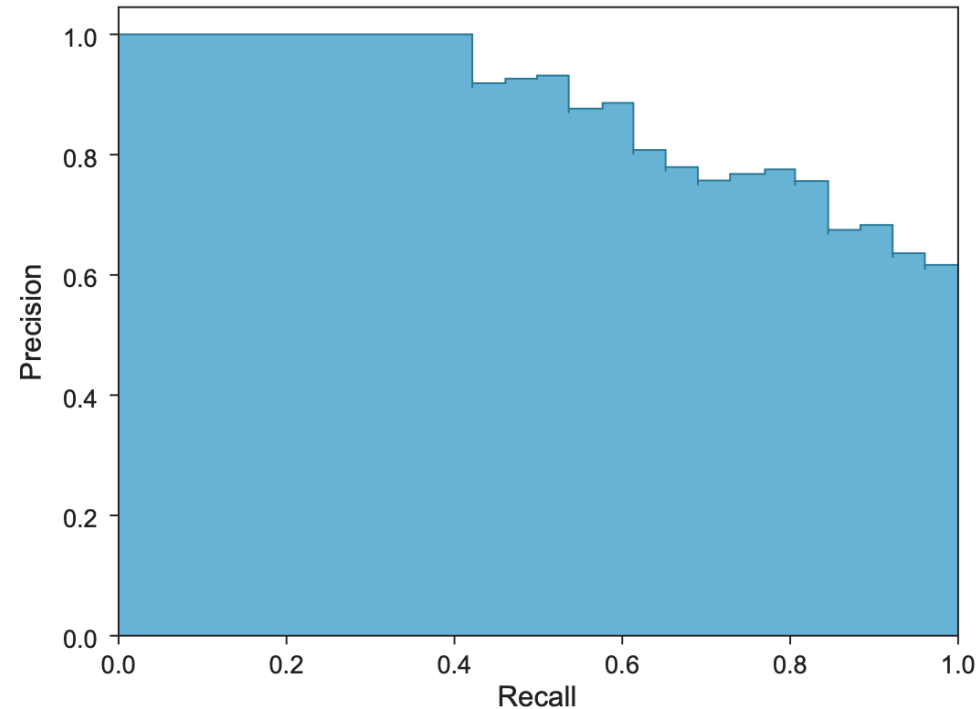
Excellent

Object Detection 평가 지표

PR(precision-recall)곡선

$$\text{재현율 (Recall)} = \frac{TP}{TP+FN}$$

$$\text{정밀도 (Precision)} = \frac{TP}{TP+FP}$$



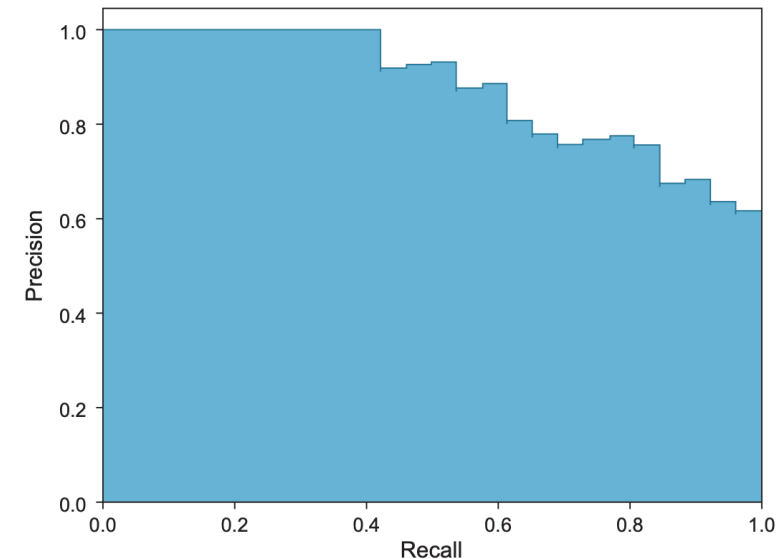
Object Detection 평가 지표

PR(precision-recall)곡선

- 각 클래스마다 PR곡선을 그려 사물 탐지 성능 평가 가능
- 물체가 아닌 것을 오검출하거나, 물체인 것을 놓치지 않아야 검출 성능이 좋다고 할 수 있음
- PR곡선 아래 면적인 **AUC(Area Under the Curve)**를 계산하여 **Average Precision(AP)**를 계산할 수 있고, 각 클래스의 AP 평균인 **mAP**를 계산하여 최종적으로 사물 탐지 모델 성능 평가

$$\text{재현율 (Recall)} = \frac{TP}{TP+FN}$$

$$\text{정밀도 (Precision)} = \frac{TP}{TP+FP}$$



Object Detection 평가 지표

mAP 정리

1. 각 bounding box의 물체 존재 확신도 계산
2. 정밀도화 재현율(precision, recall) 계산
3. 각 분류 클래스마다 확률의 임계값을 변화시키며 PR 곡선을 그림
4. PR곡선의 AUC를 계산하여 각 분류 클래스마다 평균 정밀도(AP)
5. 각 클래스의 AP 평균인 mAP(mean Average Precision) 계산

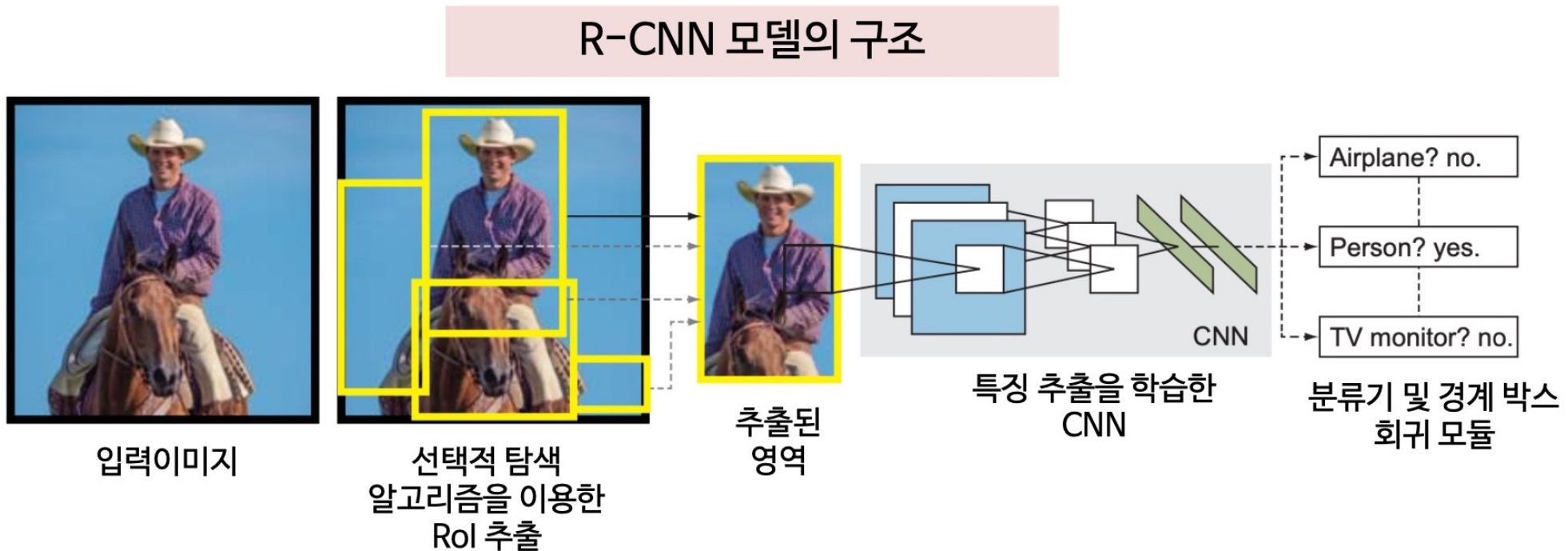
Detector 기초

Object Detector

- **Object Detector**는 크게 1. 후보 영역을 생성하는 단계와 2. 후보 영역의 클래스를 분류하는 단계로 구성
- Object Detector는 RoI 추출과 탐지를 한 번에 수행하는 1-stage detector와, 따로 수행하는 2-stage detector로 구분할 수 있음
- **2-Stage detector** : RCNN(2014), SPP-net(2015), fast RCNN(2015), Faster RCNN(2015), FPN(2017), mask RCNN(2017), G-RCNN(2021) ..
- **1-Stage detector** : Multibox(2014), AttentionNet(2015), YOLO(2016), SSD(2016), YOLOv2(2017), RetinaNet(2017), YOLOv3(2018), YOLOv4(2020), ...

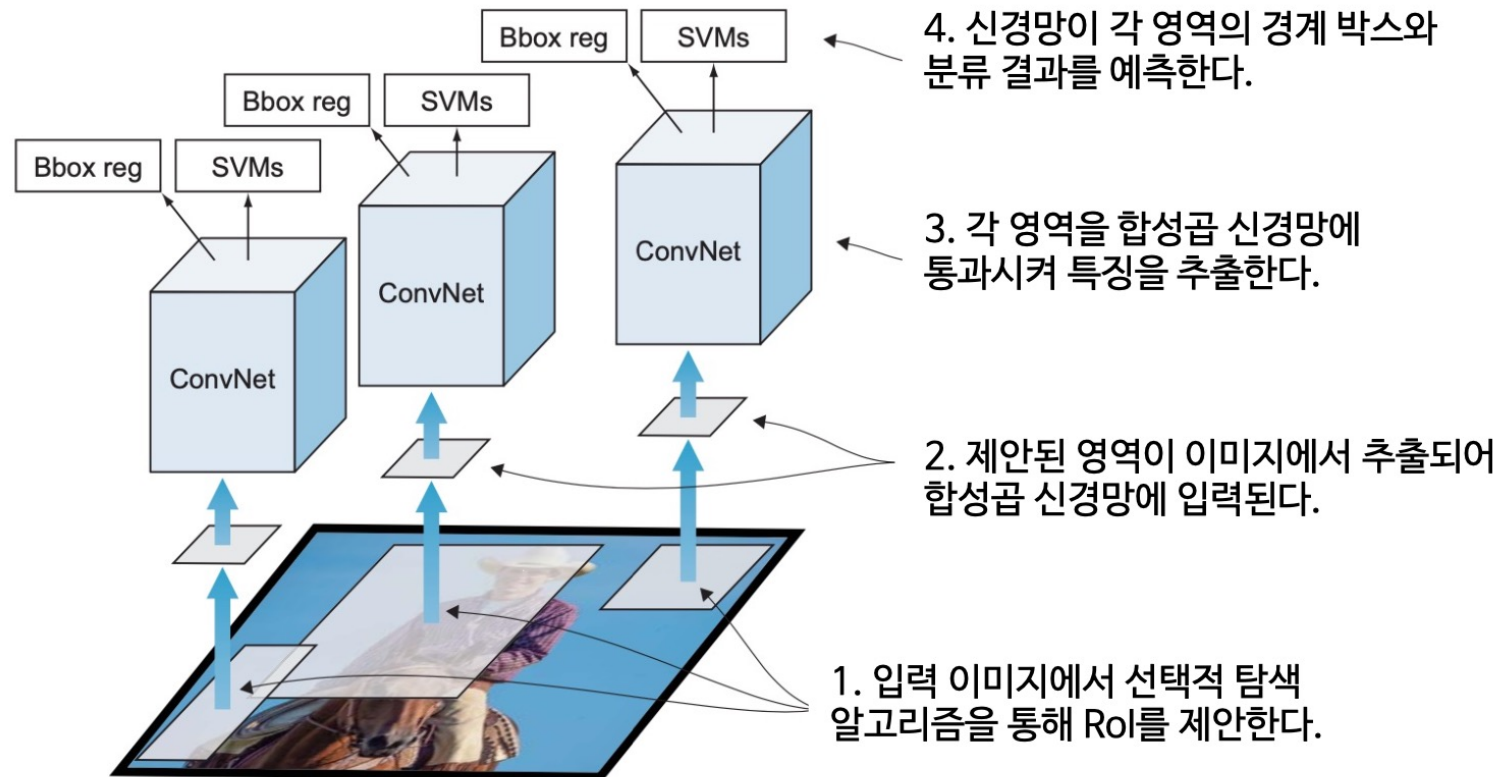
R-CNN, Fast R-CNN, Faster R-CNN

- R-CNN 계열은 CNN을 이용한 초기의 object detector로, 2-stage detector임



R-CNN, Fast R-CNN, Faster R-CNN

- R-CNN은 먼저 RoI를 제안한 다음 각 영역에서 특징을 추출하고 이 특징을 바탕으로 영역을 분류
 - RoI 추출기, 특징 추출 모듈, 분류 모듈, 경계 박스 회귀 모듈 4개의 요소로 구성

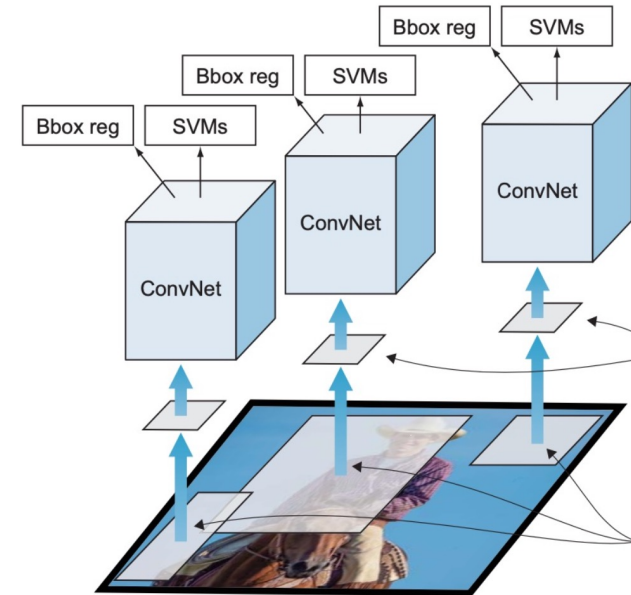
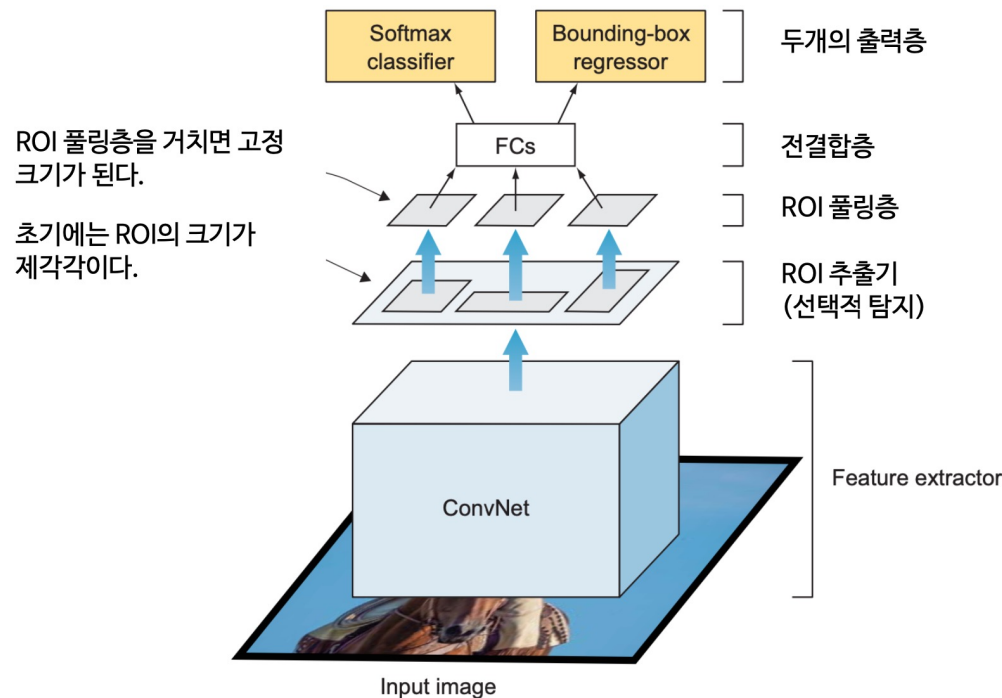


R-CNN, Fast R-CNN, Faster R-CNN

- R-CNN은 그 구조가 단순해서 이해하기 쉽고, 특징 추출에 CNN을 활용해서 제안 당시 최고 성능을 달성
- 하지만 하나의 신경망으로 구성된 시스템이 아니라 독립된 여러 알고리즘이 조합되어 동작
 - 이로 인해 사물 탐지 속도가 느리고, 학습과정이 다단계로 구성되며, 연산 복잡도가 높다는 단점이 있음

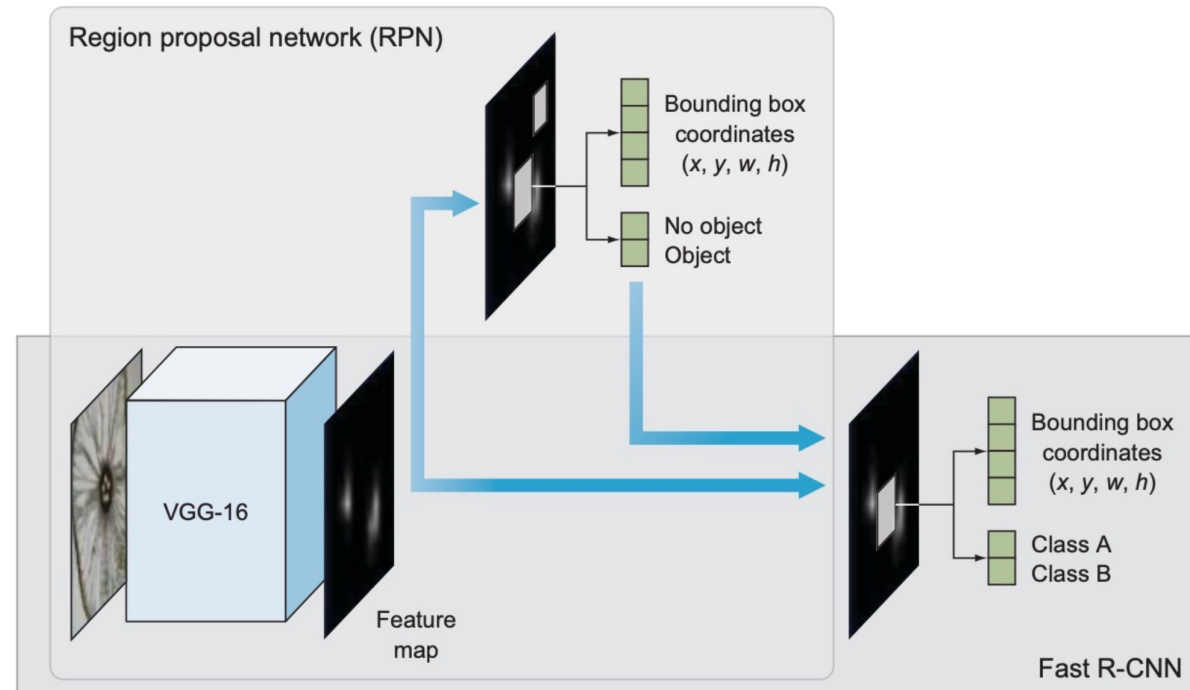
R-CNN, Fast R-CNN, Faster R-CNN

- Fast R-CNN은 R-CNN을 수정하여 개선한 모델
 - CNN feature extractor를 맨 앞에 배치해 전체 입력 이미지로부터 영역 제안
 - RCNN은 2000개 이상의 중첩된 영역을 2000개 이상의 CNN으로 처리하였지만, Fast RCNN은 전체 이미지를 하나의 CNN으로 처리할 수 있게 되었음
 - CNN에 SVM 분류기 대신 소프트맥스 계층을 추가하여 분류를 함께 수행
 - 1.RoI의 각 클래스에 속할 확률을 출력하는 softmax층, 2.최초 제안된 RoI와의 차이를 출력하는 경계 박스 회귀층 2개의 출력층을 가짐

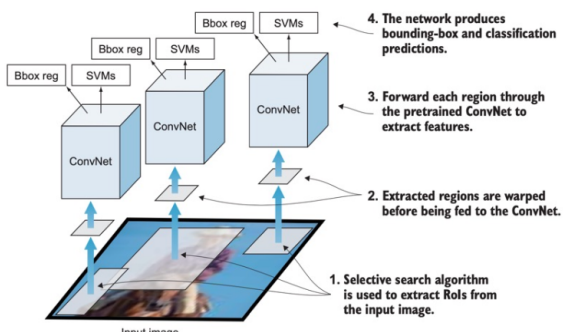
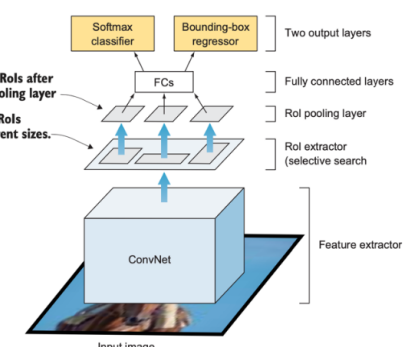
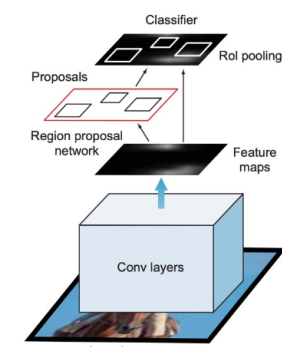


R-CNN, Fast R-CNN, Faster R-CNN

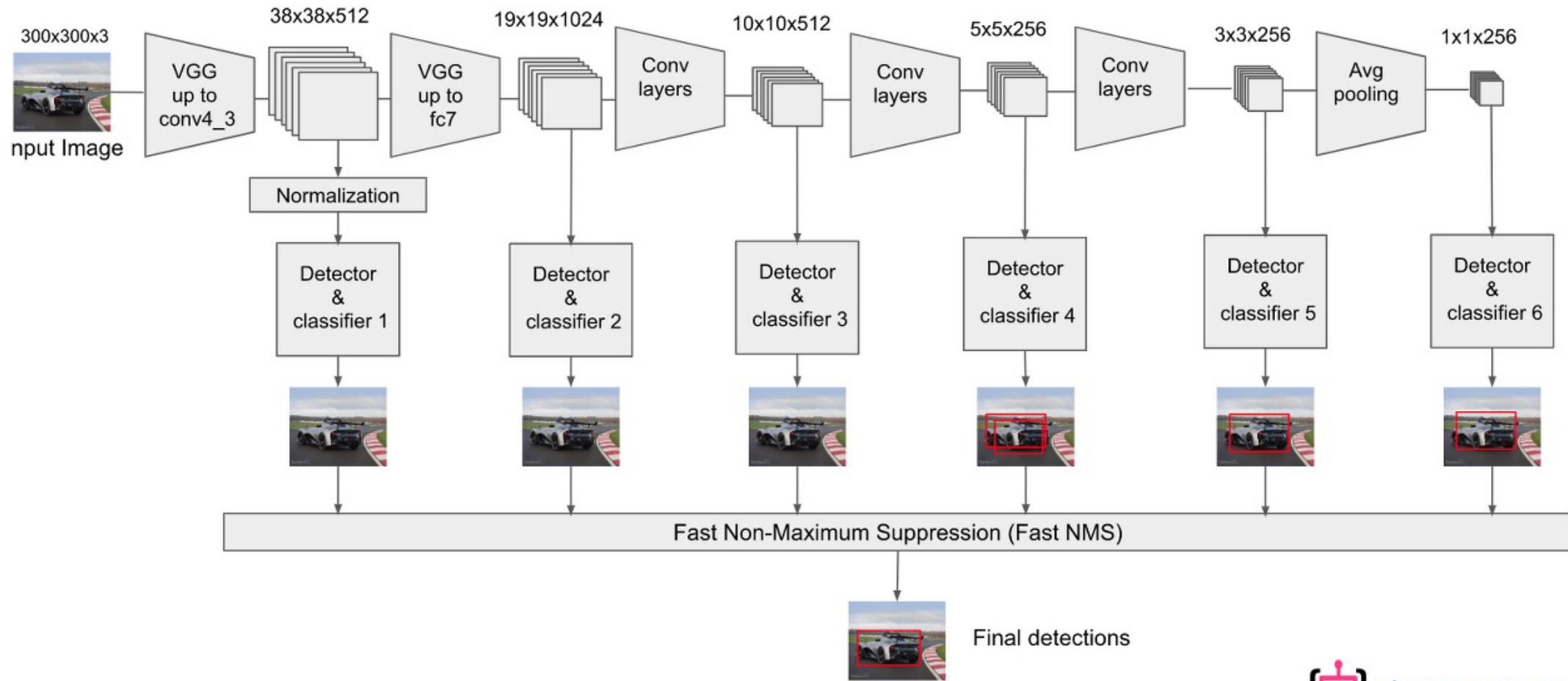
- Faster R-CNN은 영역 제안을 위해 기존의 선택적 탐색(selective search) 대신 영역 제안 신경망(region proposal network. RPN)을 사용
 - RPN은 물체 존재 확신도와 경계 박스 위치를 출력
- Faster R-CNN은 object detection pipeline 전체를 신경망에 내장시키는데 성공(end-to-end)
 - 특징 추출기, 영역 제안, RoI pooling, 분류기, 경계 박스 회귀 모듈을 파이프라인 내부로 통합



R-CNN, Fast R-CNN, Faster R-CNN 요약

	R-CNN	Fast R-CNN	Faster R-CNN
	 <p>1. Selective search algorithm is used to extract RoIs from the input image.</p> <p>2. Extracted regions are warped before being fed to the ConvNet.</p> <p>3. Forward each region through the pretrained ConvNet to extract features.</p> <p>4. The network produces bounding-box and classification predictions.</p>	 <p>Fixed-size RoIs after the RoI pooling layer</p> <p>Proposed RoIs have different sizes.</p> <p>Two output layers</p> <p>Fully connected layers</p> <p>RoI pooling layer</p> <p>RoI extractor (selective search)</p> <p>Feature extractor</p>	 <p>Classifier</p> <p>RoI pooling</p> <p>Proposals</p> <p>Region proposal network</p> <p>Feature maps</p> <p>Conv layers</p> <p>Input image</p>
정확도*	66.0%	66.9%	66.9%
특징	<ol style="list-style-type: none"> 선택적 탐색 알고리즘을 사용해서 RoI (2000개 이상)를 추출함. 별도의 합성곱 신경망을 사용해서 각 RoI에서 특징을 추출함. 경계 박스 예측과 클래스 분류를 함께 수행함 	<p>각 이미지는 CNN에 한 번만 전달되어 특징 맵이 추출된다.</p> <ol style="list-style-type: none"> CNN을 사용해서 입력 이미지에서 특징 맵을 추출함. 특징 맵에 선택적 탐색 알고리즘을 적용함. <p>이런 방식으로 2000곳 이상의 RoI를 모두 합성곱 신경망으로 처리하는 대신 전체 이미지를 한번만 처리하면 된다.</p>	<p>선택적 탐색 알고리즘을 RPN으로 대체해서 속도를 개선했다.</p> <p>전체 처리 과정을 처음부터 끝까지 딥러닝으로 구현했다.</p>
단점	RoI를 하나씩 별도로 처리하므로 처리 시간이 오래 걸린다. 또한 독립된 3개의 모델을 모두 사용해야 최종 예측을 할 수 있다.	선택적 탐색 알고리즘의 느린 속도 탓에 여전히 처리 시간이 오래 걸린다.	영역 제안에서 시간 소모가 크다. 또한 여러 시스템이 순차적으로 동작하기 때문에 전체 처리 속도가 각 시스템의 처리 속도에 영향을 많이 받는다.
이미지당 처리 시간	50초	2초	0.2초
R-CNN 대비 처리 속도	1배	25배	250배

SSD : Single Shot multibox Detector

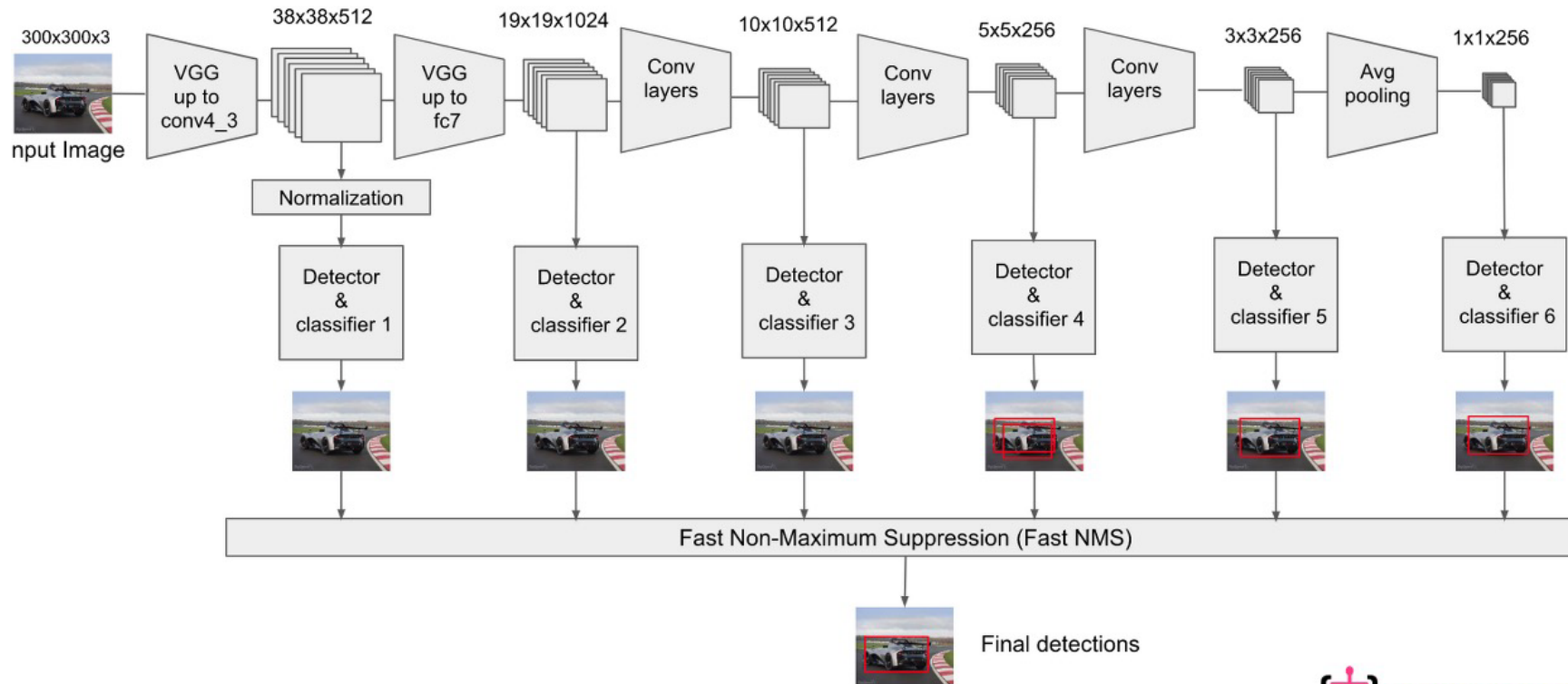


SSD : Single Shot multibox Detector

- **SSD**는 CNN 기반 object detector로, 여러 크기의 고정 크기 박스(anchor box)를 생성하고 각 box에 objectness score를 부여한 다음 NMS를 통해 최적의 탐지 결과를 제외한 나머지를 배제한다. 크게 다음 세 가지 요소로 구성된다
- **VGG backbone** : 사전 학습된 VGG backbone에서 classification layer를 제거하여 feature extractor로 사용. VGG가 아닌 ResNet 등 다른 backbone network 사용 가능
- **다중 스케일 특징층** : base convolution 뒤에 배치된 일련의 CNN filter. 필터 크기가 점진적으로 감소하며 다양한 배율로 탐지 가능
- **NMS** : 중첩되는 bounding box를 배제하고 물체별로 박스를 하나만 남김

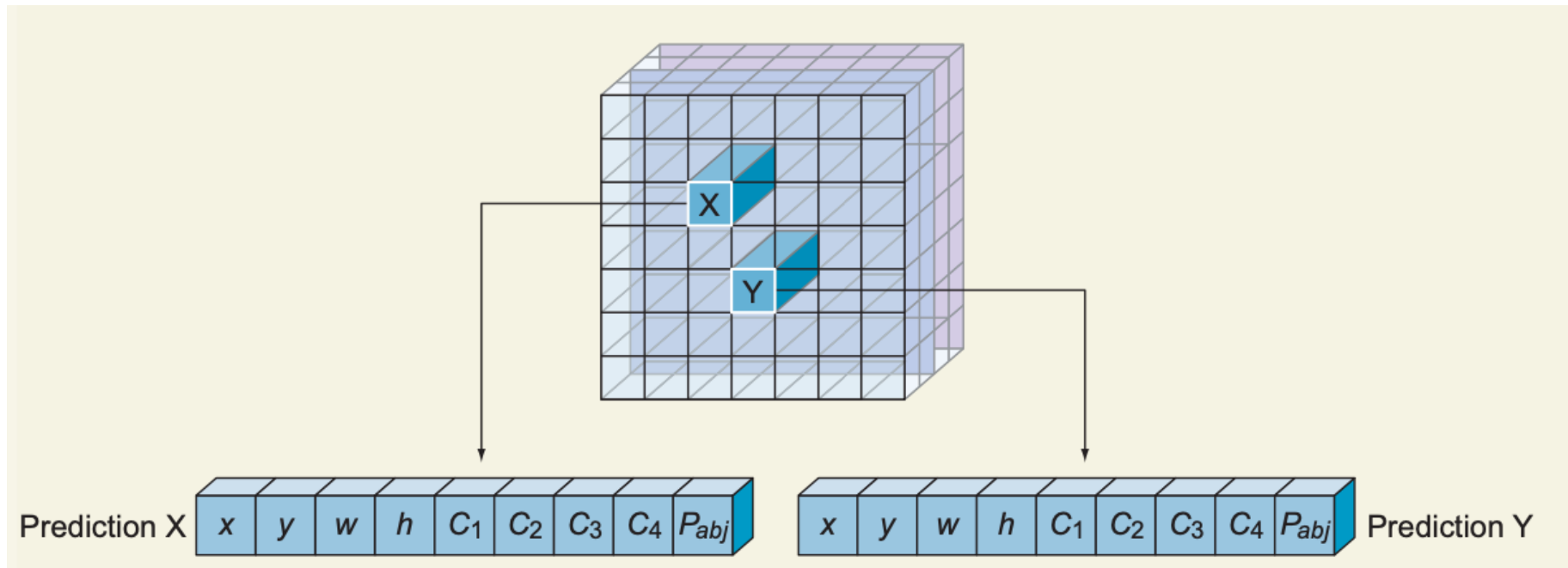
SSD : Single Shot multibox Detector

- SSD는 4_3, 7, 8_2, 9_2, 10_2, 11_2 번째 계층이 직접 예측 결과를 NMS로 전달
- 이들 계층의 필터 크기는 점진적으로 감소하는 형태를 취함
 - (38->19->10->5->3->1)



SSD : Single Shot multibox Detector

- 신경망은 각 feature에 대해 다음 값을 예측
 - Bounding Box를 경정하는 값 4개 (x, y, w, h)
 - 물체 존재 확신도 1개
 - 각 클래스의 확률을 나타내는 값 C 개
- 예측 결과는 $C+5$ 개 값으로 구성



SSD : Single Shot multibox Detector

- 논문 링크 : <https://arxiv.org/abs/1512.02325>
- 다음 README에 자세한 설명이 나와 있으므로 논문 이해 완료 후 참고자료로 사용
 - <https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection>
- 이번 Task가 필요한 이유는?
 - 논문 형식에 대한 이해
 - 논문을 읽기 위한 용어 파악
 - 논문 기반 구현 능력(원복)의 중요성
- 최대한 논문 기반으로 내용을 이해하려고 노력하면서, 논문 읽는 능력을 길러봅시다!

End
