

Object Detection

2020. 10. 28

정 준 수 Ph.D.

실습 예제

[**https://github.com/JSJeong-me/2020-11-02_Steel_AI**](https://github.com/JSJeong-me/2020-11-02_Steel_AI)

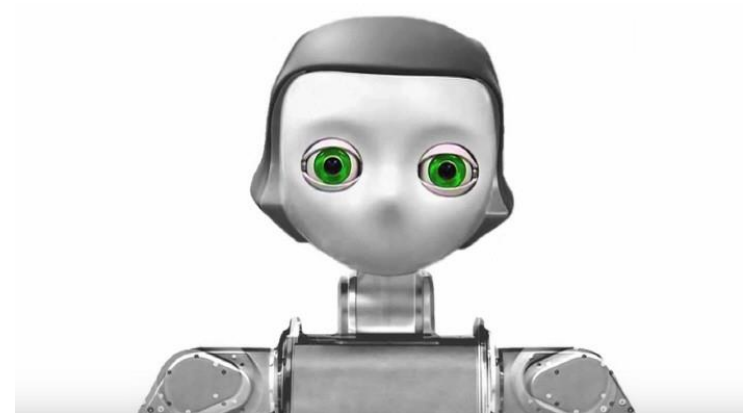
과정 목표 – 사람과 AI의 시각 정보처리 과정 비교 학습



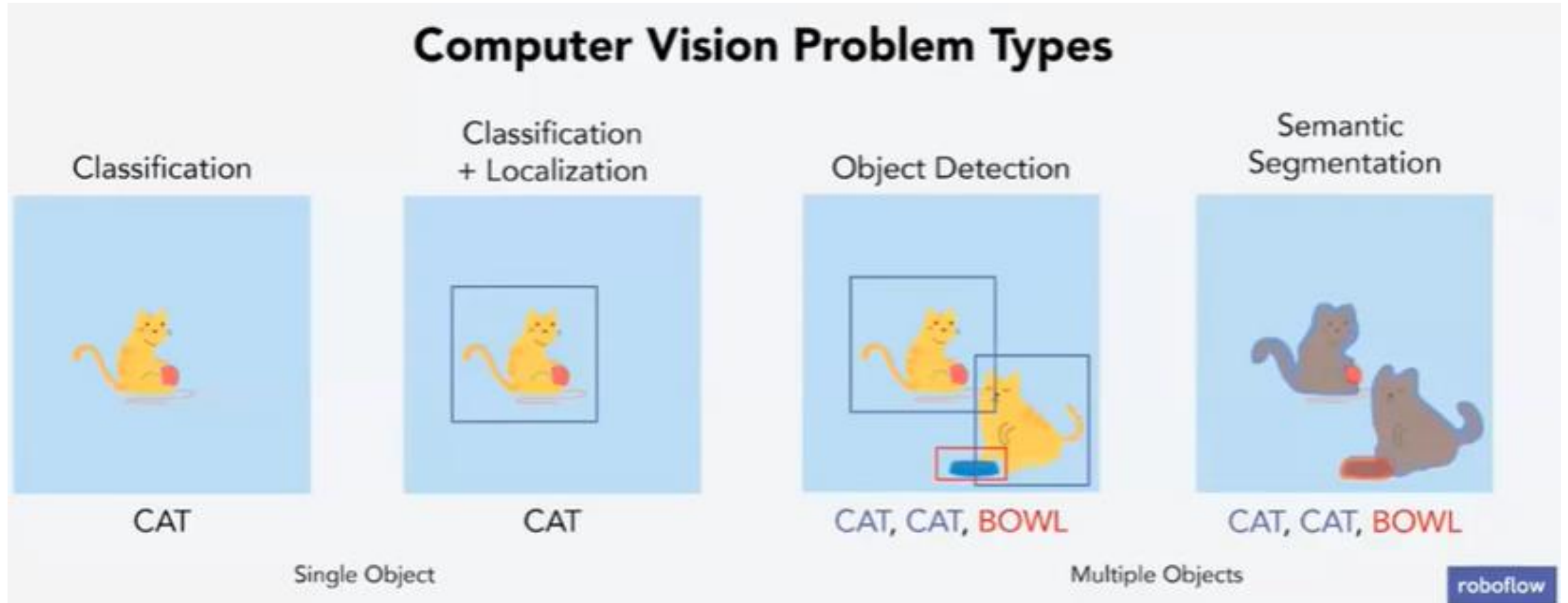
어린 고양이(Kitten)



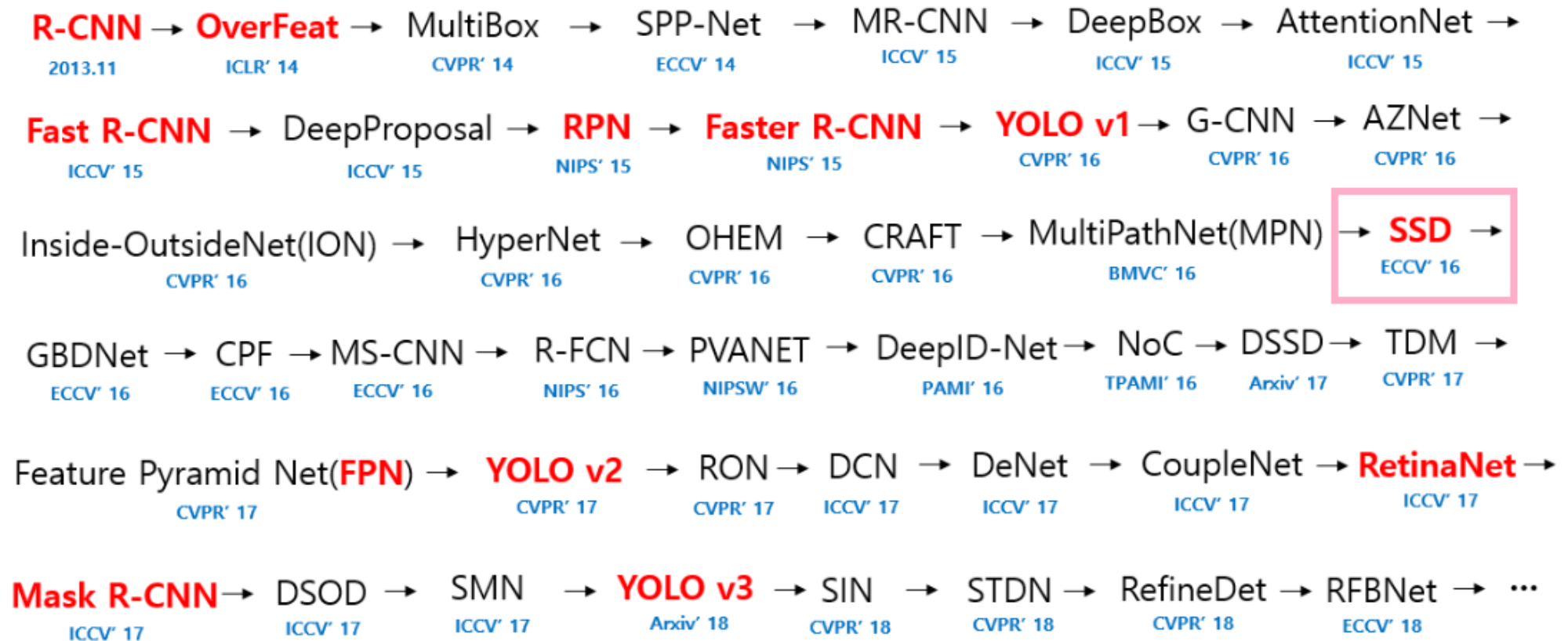
VS



Vision Tasks



Object Detection Models



YOLO - 단일 단계 방식의 객체 탐지 알고리즘

- YOLO(You Only Look Once)는 대표적인 단일 단계 방식의 객체 탐지 알고리즘입니다. YOLO 알고리즘은 원본 이미지를 동일한 크기의 그리드로 나눕니다.
- 각 그리드에 대해 그리드 중앙을 중심으로 미리 정의된 형태(predefined shape)으로 지정된 경계박스의 개수를 예측하고 이를 기반으로 신뢰도를 계산합니다. 이미지에 객체가 포함되어 있는지, 또는 배경만 단독으로 있는지에 대한 여부가 포함되겠죠. 높은 객체 신뢰도를 가진 위치를 선택해 객체 카테고리를 파악합니다.

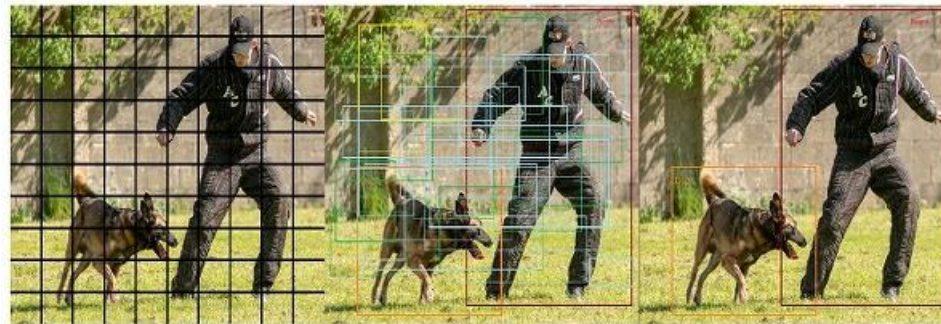


그림 0 경계박스(Bounding-box) 예측단계

YOLO

- 미리 정의된 형태를 가진 경계박스 수를 '앵커 박스(Anchor Boxes)'라고 하는데요. **앵커 박스는 K-mean 알고리즘에 의한 데이터로부터 생성되며**, 데이터 세트의 객체 크기와 형태에 대한 사전 정보를 확보합니다.
- 각각의 앵커는 각기 다른 크기와 형태의 객체를 탐지하도록 설계되어 있습니다. 그림2를 보면 한 장소에 3개의 앵커가 있는데요. 이 중 붉은색 앵커 박스는 가운데에 있는 사람을 탐지합니다.
- 이 알고리즘은 앵커 박스와 유사한 크기의 개체를 탐지한다는 뜻인데요. 최종 예측은 앵커의 위치나 크기와는 차이가 있습니다. 이미지의 피쳐(Feature) 맵에서 확보한 최적화된 오프셋이 앵커 위치나 크기에 추가됩니다.

탐지 방식

- 영역의 박스와 해당 박스의 분류를 동시에 복수개를 출력하는 네트워크
- 모두 98개($7 \times 7 \times 2$)가 제안되고, 중복되지 않고 확신도가 높은 것만 선택

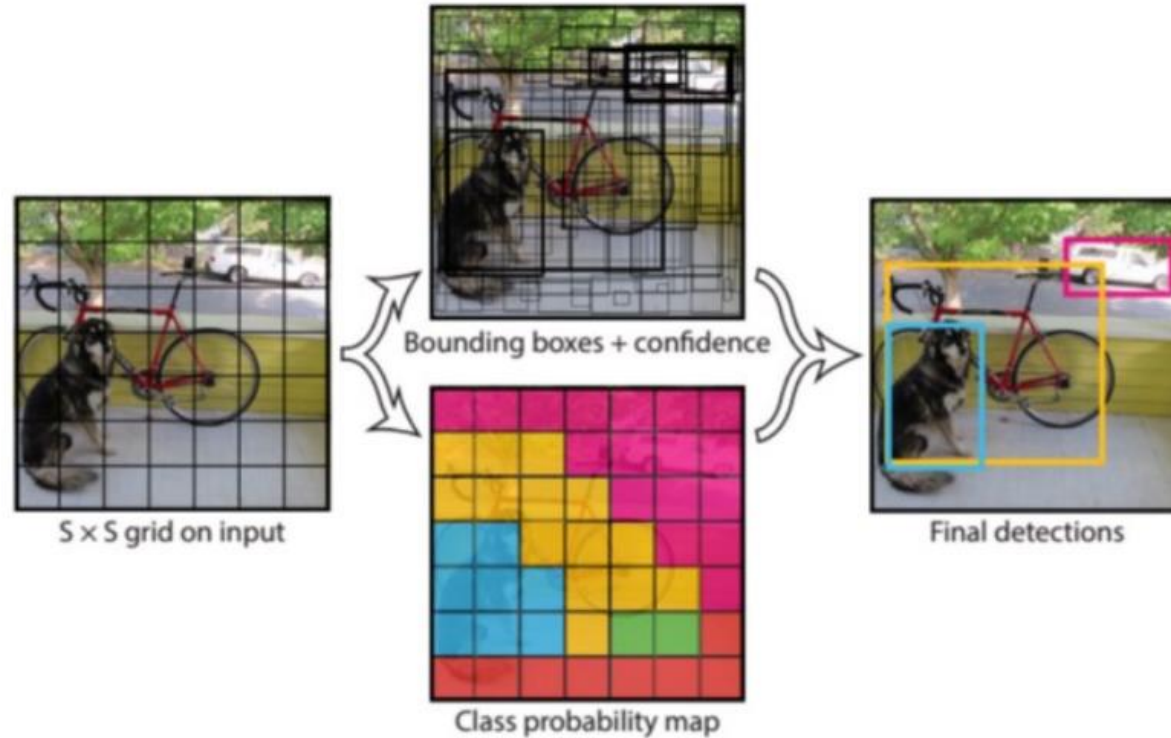


그림 1 모델 구조는 입력 이미지를 $S \times S$ 그리드로 나누고, 객체의 중심이 그리드 셀에 들어가면 해당 그리드 셀이 해당 객체를 감지함.

앵커박스과 YOLO 네트워크 아키텍처



그림 2 앵커 박스(이미지 출처: https://www.flickr.com/photos/762_photo/16751321393/)

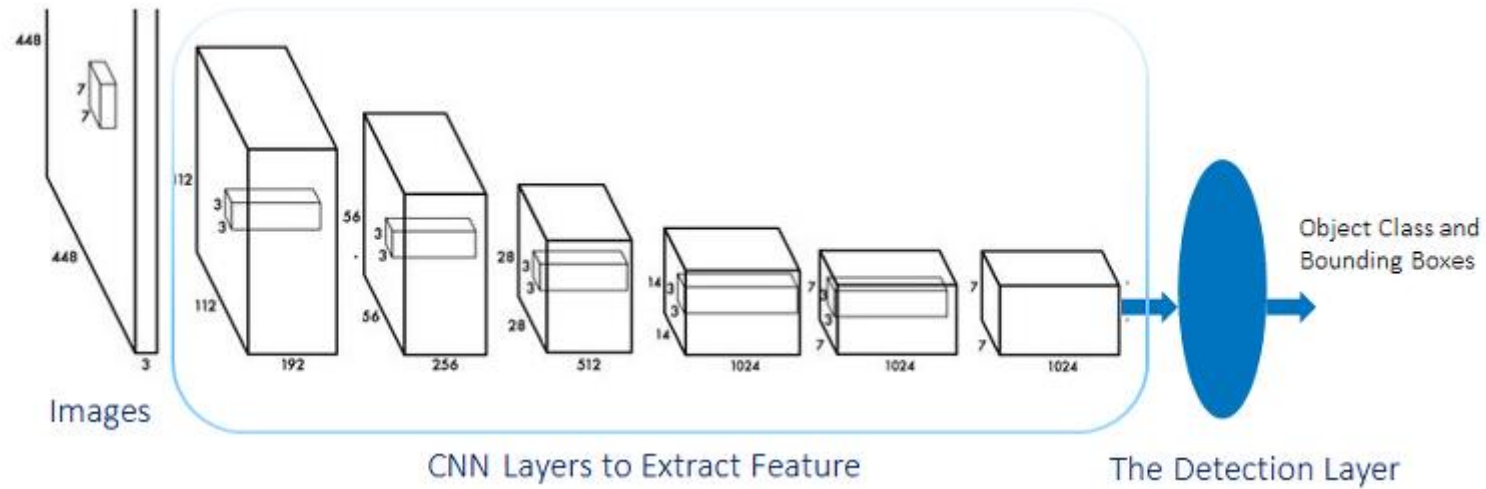
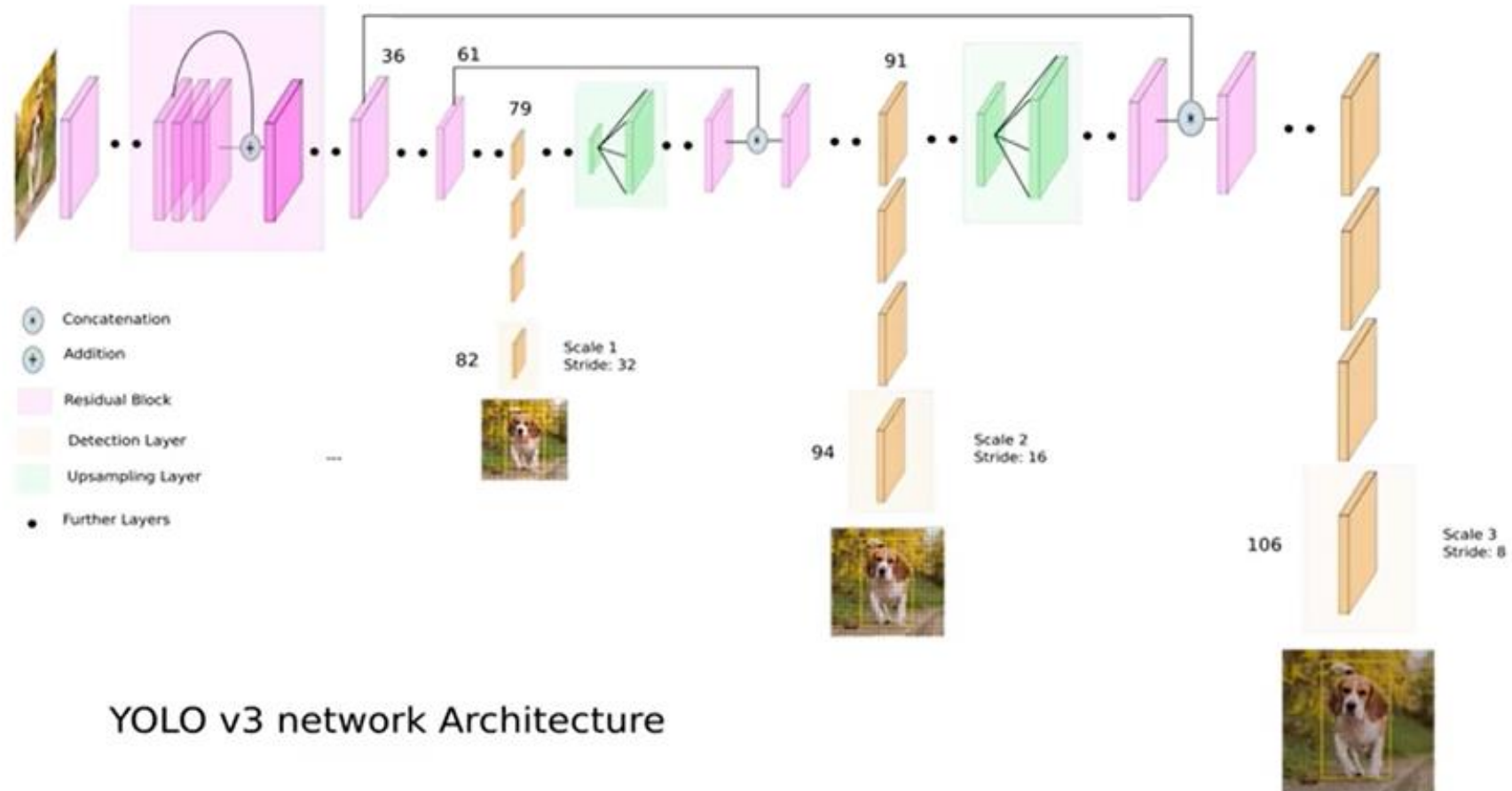


그림 3 YOLO 네트워크 아키텍처

YOLO3 Architecture

YOLO3



Faster RCNN – YOLO 이전의 객체 탐지 알고리즘

- Faster RCNN은 이단계 방식의 객체 탐지 알고리즘인데요. 그림 4에서 Faster RCNN의 단계를 살펴보겠습니다.
- 이 알고리즘 이름에 '빠른(Faster)'이라는 단어가 포함되어 있지만 단일 단계 방식보다 빠른 처리가 된다는 뜻이 아니고 이전 버전이라 할 수 있는 RCNN 알고리즘과 Fast RCNN 알고리즘 보다 빠르다는 것을 뜻하는데요. 각 관심 영역(RoI; Region of Interest)에 대한 피쳐 추출의 계산을 공유하고 딥러닝 기반의 RPN을 도입해 구현할 수 있습니다.

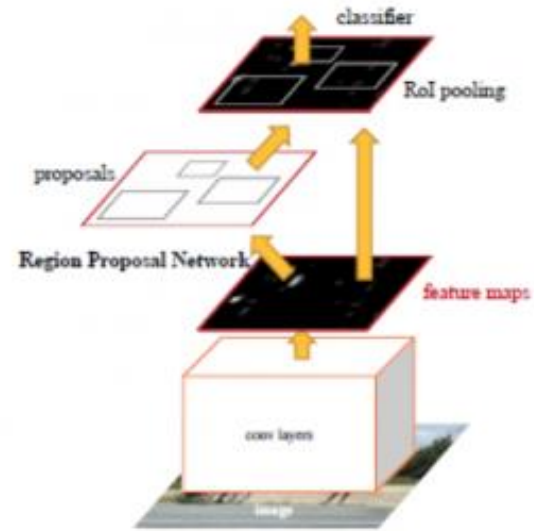
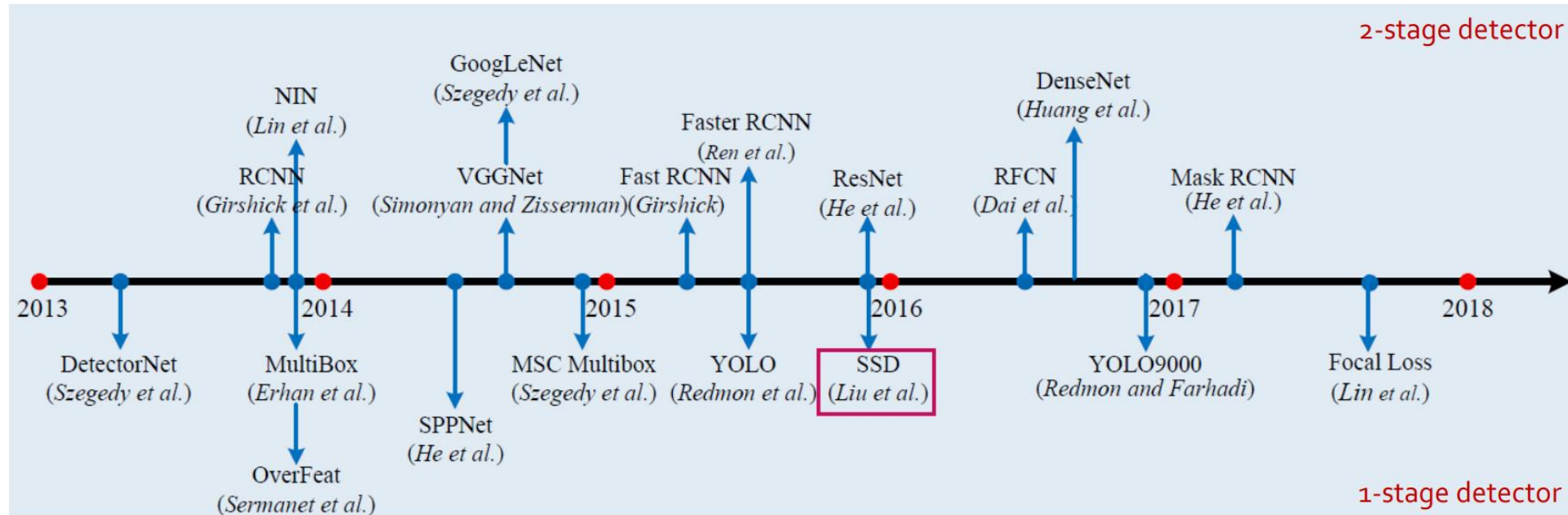


그림 4 Faster RCNN 단계

Faster RCNN

- 많은 CNN 레이어를 사용해 피쳐 맵을 추출하고 나면 RPN을 통해 개체를 포함하고 있을 가능성이 높은 윈도우가 다량으로 생성됩니다. 그런 다음 알고리즘은 각 윈도우에 있는 피쳐 맵을 검색하고, 고정 크기로 조정된 뒤(RoI 풀링) 클래스 확률과 해당 객체에 대한 더욱 정확한 경계박스를 예측합니다.
- 여기서 고려해야 할 점은 RPN이 윈도우를 생성하는 방법인데요. RPN은 YOLO와 마찬가지로 앵커 박스를 사용합니다. 하지만 YOLO 알고리즘과 다른 점은 앵커 박스가 데이터로부터 생성되는 것이 아니라 고정된 크기와 형태로 생성된다는 것인데요. 이 앵커 박스는 이미지를 보다 조밀하게 커버할 수 있습니다. RPN은 여러 객체 카테고리에 대한 분류 대신 윈도우의 객체 포함 유무에 대한 이진 분류(Binary Classification)만 수행합니다.

1-stage detection 와 2-stage detection



Milestones of object detection and recognition, including feature representations

[출처] Deep Learning for Generic Object Detection: A Survey, arxiv 2019

SSD(Single Shot MultiBox Detector)

SSD 물체 감지 모델은

- 경계 상자 가설, 각 상자의 픽셀 또는 피처 리샘플링 및 고품질 분류기를 적용

연산이 많으며, 집약적이라서 실시간 적용에는 다소 느림 감이 있음

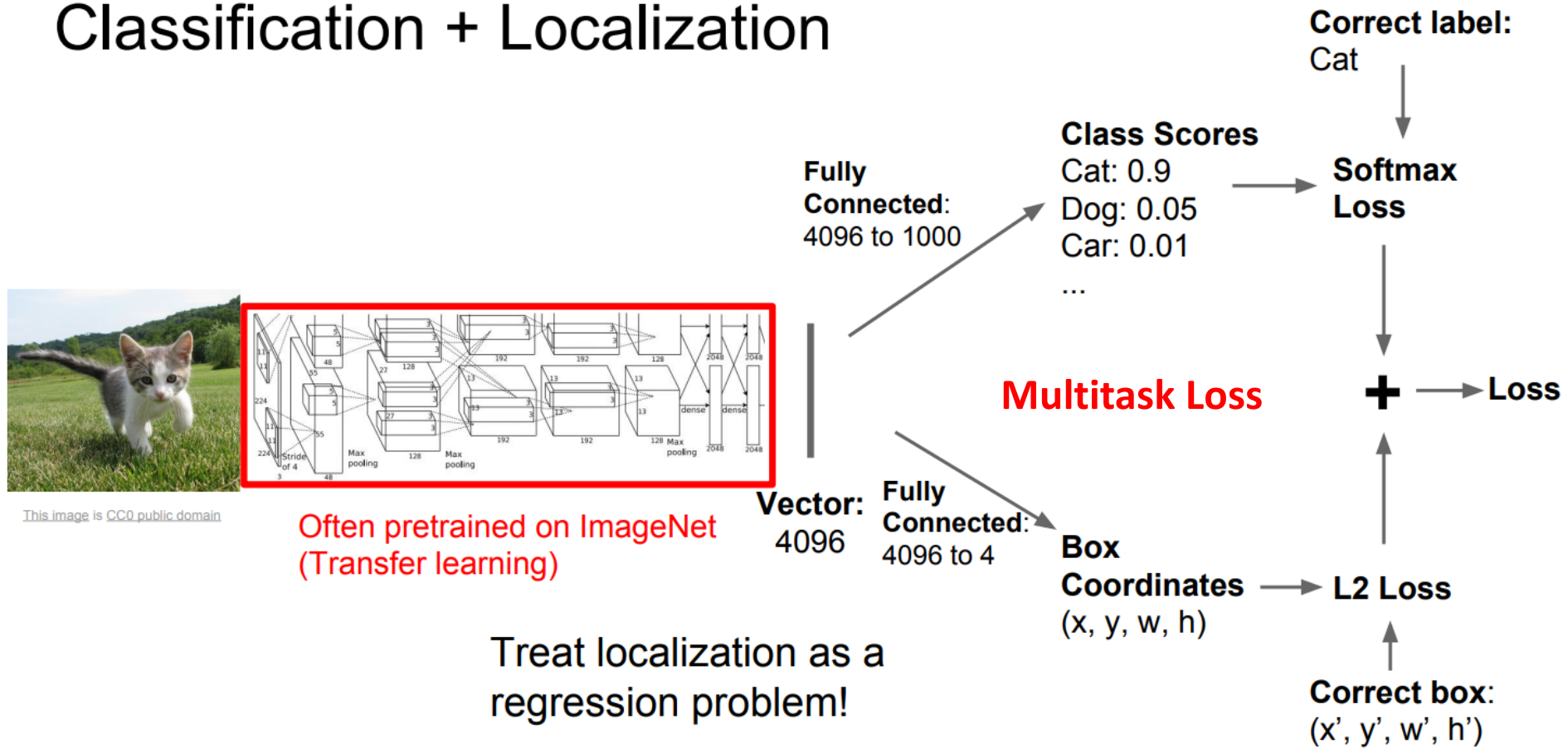
- Faster R-CNN은 mAP 73.2 %로 7FPS에서 작동합니다.

YOLO는 대폭 증가 된 속도 대비 감지 정확도는 낮음

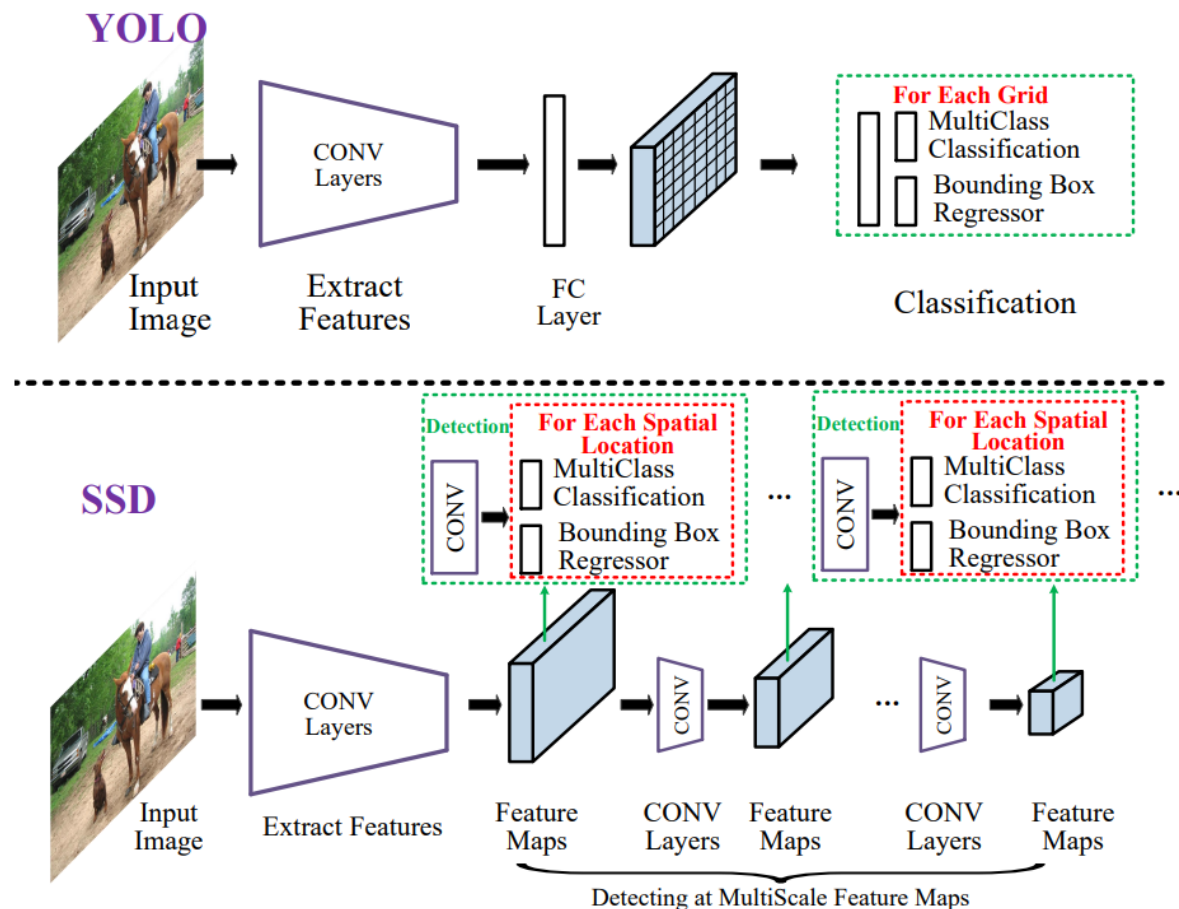
- YOLO는 mAP 63.4 %로 45FPS로 작동합니다.

Object Detection : 판별분석 + 영역구분

Classification + Localization

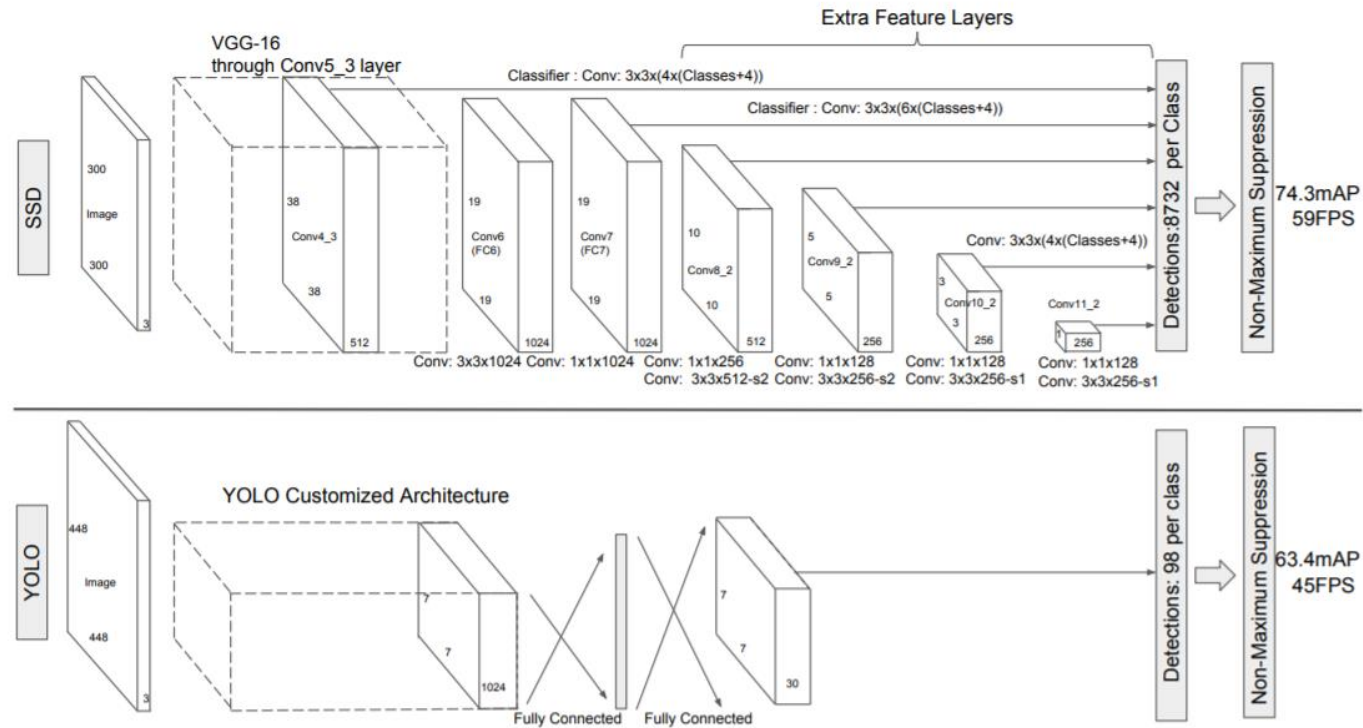


YOLO와 SSD 모델의 차이 비교



경계 상자 가설 (1 단계)에 대한 픽셀 또는 특징을 리샘플링하지 않고 접근 방식만큼 정확한 최초의 딥 네트워크 기반 객체 감지함.

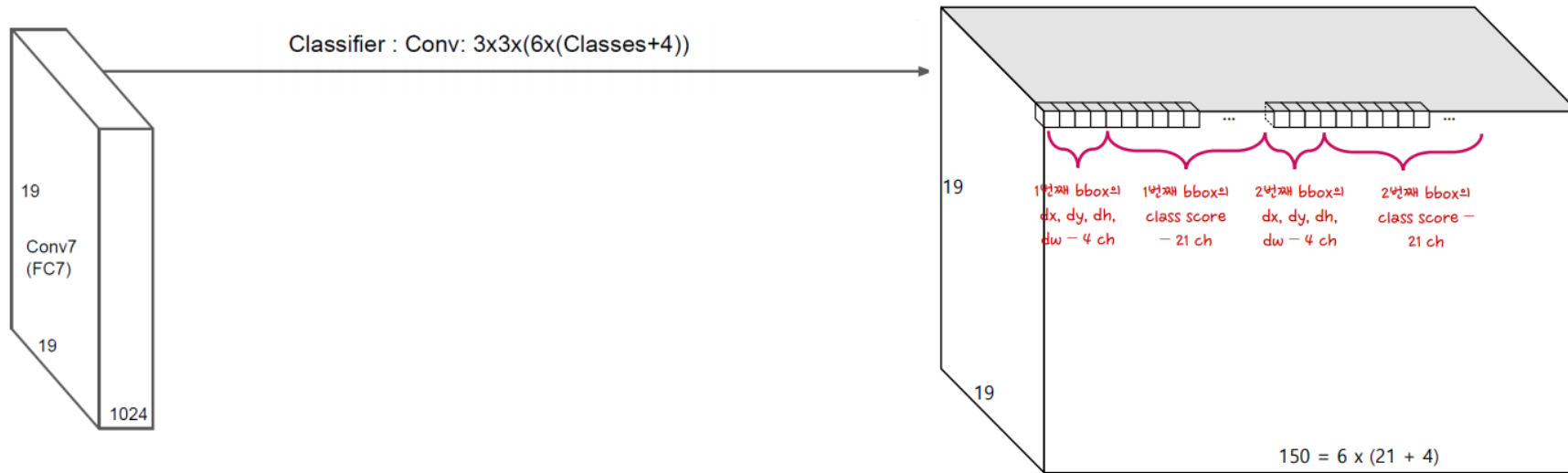
Multifl Feature Map의 활용




6개의 기본 컨볼루션 네트워크의 끝에서 피처를 가져오며, 이러한 Layer의 크기는 점진적으로 줄어들고 여러 스케일에서 탐지하여 예측가능.

Default Box 와 종횡비율

- 주어진 위치에서 k 중 각 상자에 대해 c 클래스 점수와 원래 기본 상자 모양에 상대적인 4 개의 오프셋을 계산함.
- Feature Map의 각 위치에 있는 총 $(c + 4) k$ 필터 생성.
- $M \times n$ Feature Map에 대해 $(c + 4) kmn$ 출력을 만듦.



학습방법 - 매칭 전략

$$\text{Jaccard Overlap} = \text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


The diagram illustrates the Jaccard Overlap (IoU) concept. It shows two overlapping rectangles: a light blue one and a dark blue one. The area where they overlap is shaded in a darker blue. Below the rectangles, a single large blue shape represents the union of the two rectangles. The formula above the diagram calculates IoU as the ratio of the overlapping area to the union area.

모든 Ground Truth 상자에는 최소한 1 개의 대응을 시키며, jaccard 겹침이 가장 큰 Default Box에 각 원본 영상과 일치시킴.
그런 다음 jaccard 오버랩이 임계 값 (0.5)보다 높은 모든 Ground Truth에 Default Box에 일치시킴.

.

학습 측정 방법 - Matric

- Similar to Faster R-CNN

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

set to 1 by cross validation

- N : number of default matched bboxes
- L_{conf} : classification loss → cross-entropy

$$L_{conf}(x, c) = - \sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

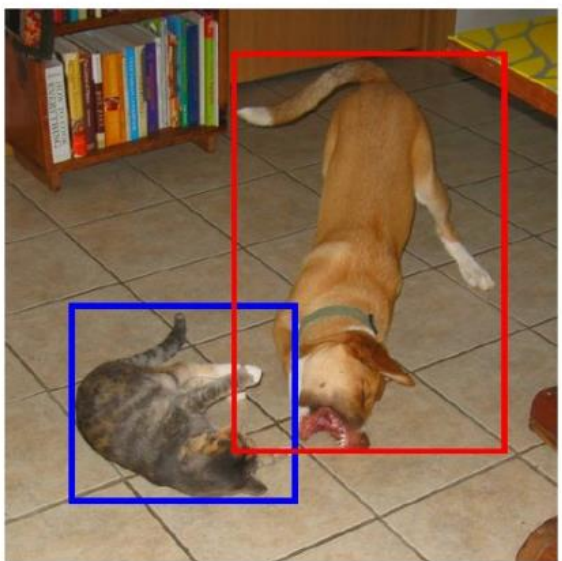
- L_{loc} : localization loss → Smooth L1 loss

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

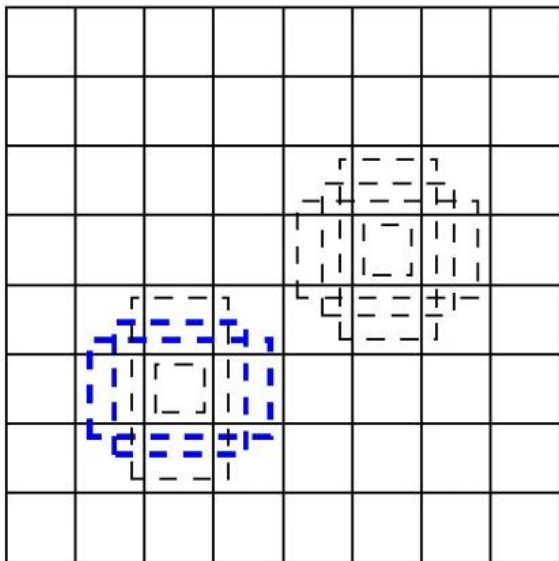
$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \quad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

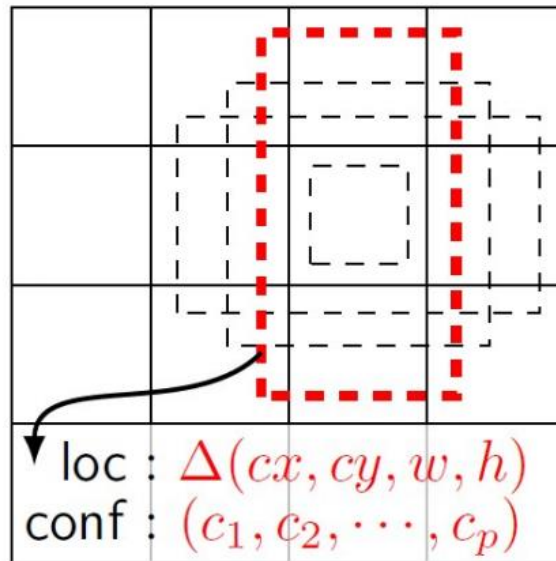
다양한 배율 Convolution Layer에서 종횡 비율에 따른 선택



(a) Image with GT boxes



(b) 8×8 feature map

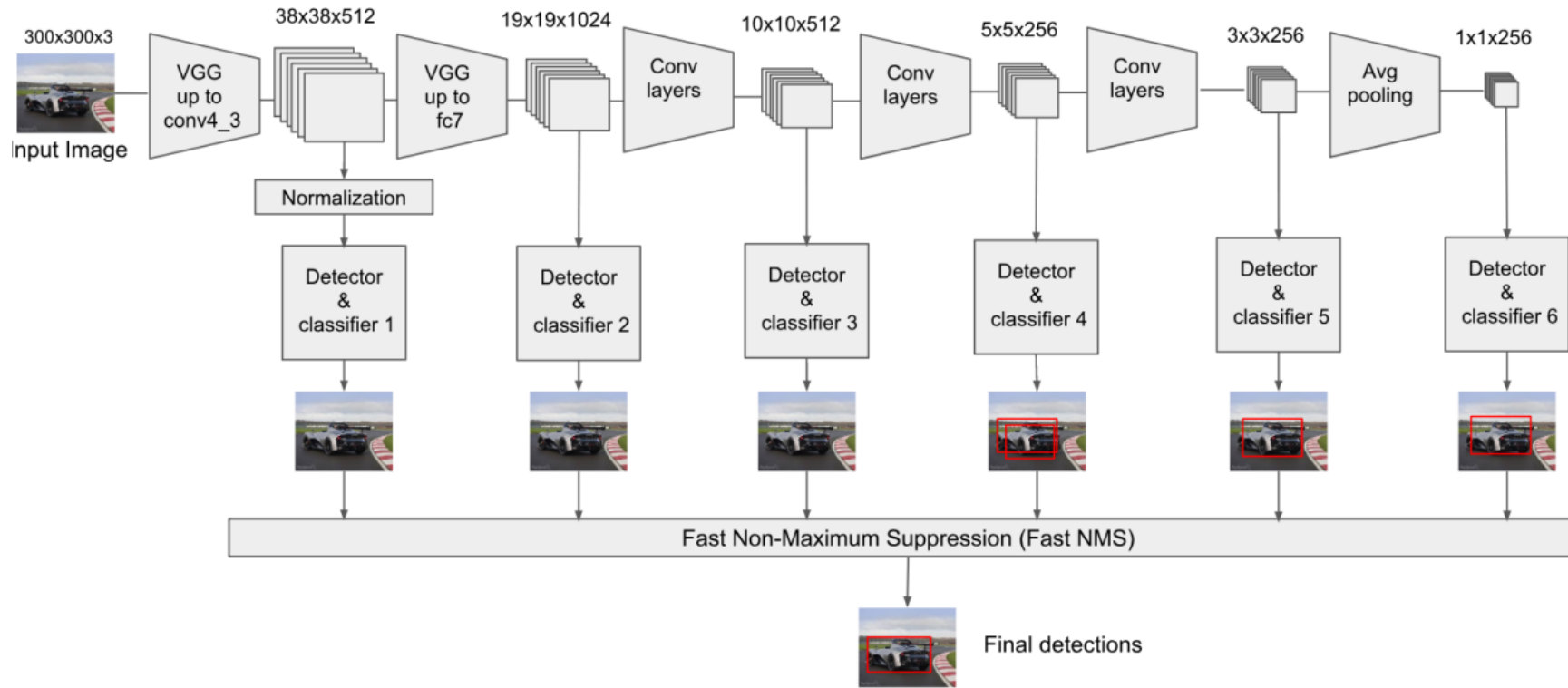


loc : $\Delta(cx, cy, w, h)$
conf : (c_1, c_2, \dots, c_p)

(c) 4×4 feature map

다른 레이어의 피쳐 맵을 사용하여 스케일 차이를 처리하며, 특정 기능 맵 위치는 이미지의 특정 영역과 개체의 특정 배율에 반응하는 법을 학습함.

기본 상자에 대한 배율 및 종횡비에 따른 선택



상대적으로 큰 물체는 종단 Layer에서 Object를 선택함.

SSD의 장점

다중 Object에 대한 SSD의 성능은,

- 다양한 스케일을 처리하기 위한 Multifl Feature Map
기능 지원
- 기본 경계 상자(Bounding Box)가 많을수록 더 나은
결과를 얻을 수 있음
- 높은 수준의 정확도와 빠른 인식 속도

실 습

강사 소개

정 준 수 / Ph.D. (heinem@naver.com)

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 전문강의, AI 교재출판 전문위원
- 現) 한국소프트웨어산업협회, AI, 머신러닝 SW전문강의
- 現) 한성대학교 교수(겸)
- 전문분야: 시각 모델링, 머신러닝(ML), RPA
- <https://github.com/jsjeong-me>

