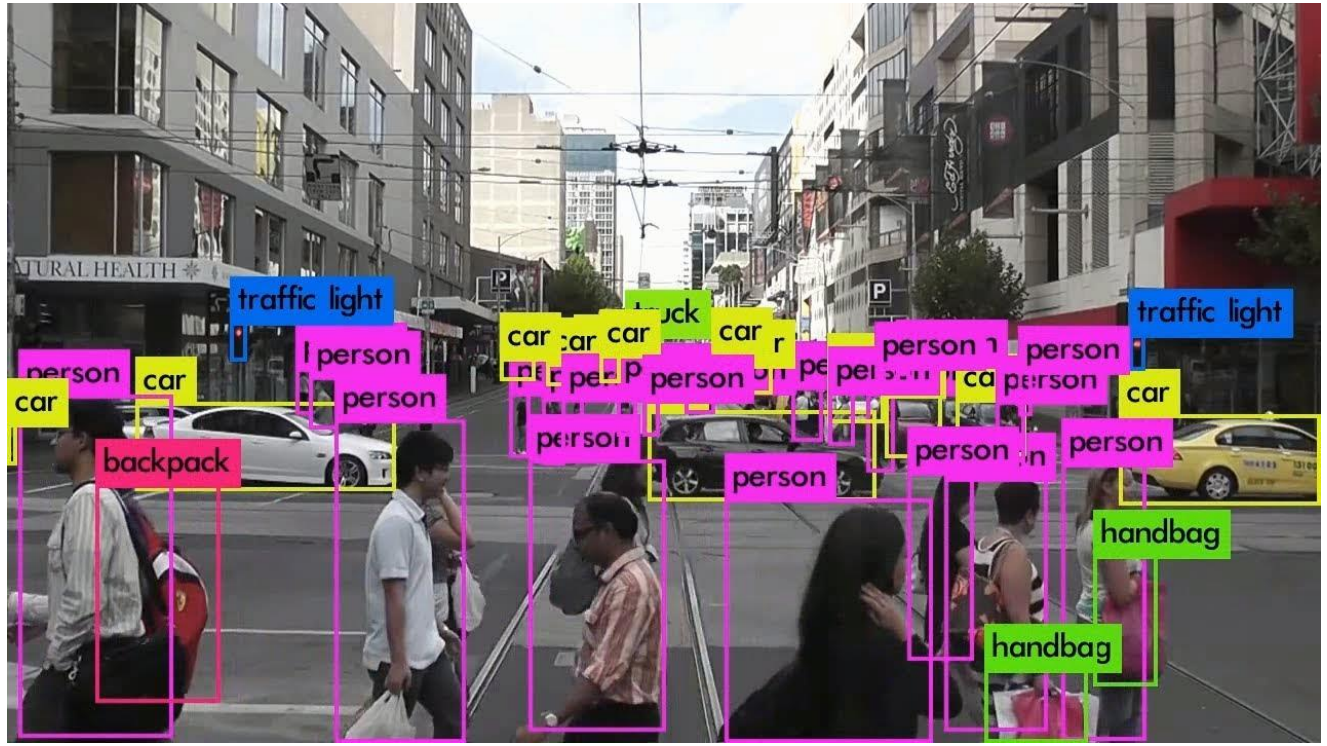
객체 탐지

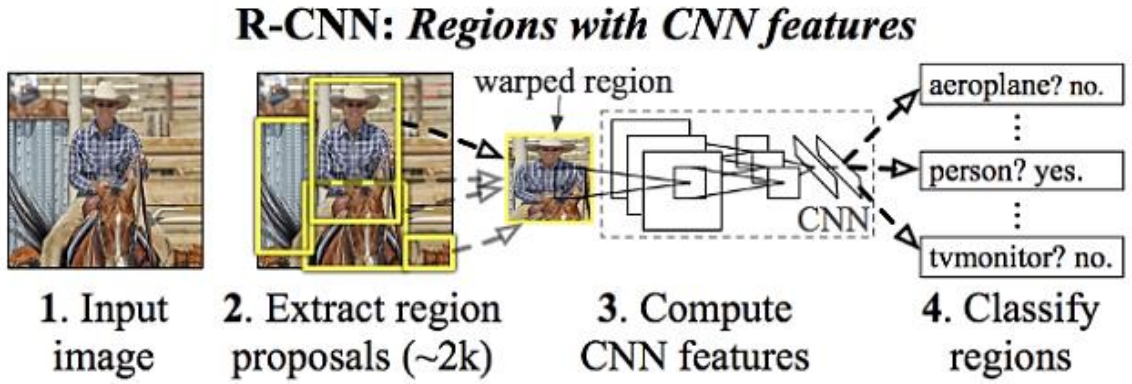
- 객체 탐지(Object Detections)
  - 객체 탐지는 이미지에서 물체를 분류(classification)한 후 정확한 위치를 경계 박스(Bounding Box 이하 BBox)로 표시(localization)하는 것
  - 영상처리 알고리즘 기반에서 딥러닝으로 변화



객체 탐지

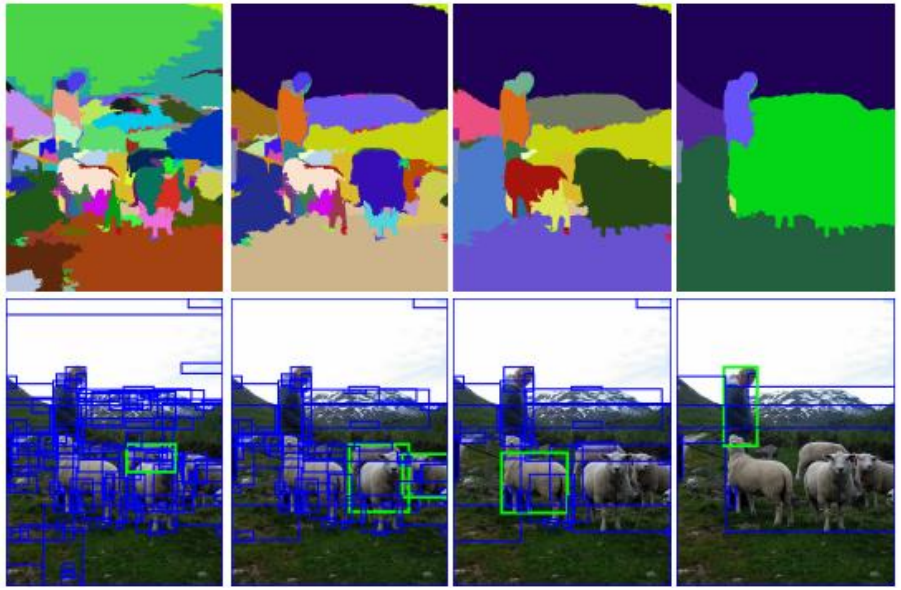
- 투샷 탐지(two-shot-detection)
  - 관심 영역의(ROI)을 슬라이딩 윈도우 이동하면서 선택적 검색(selective search) 알고리즘과 합성곱 신
    - 선택적 검색 알고리즘을 통해 지역을 제안(region proposals)하고, 이를 합성곱 신경망에 전달해 클래스 분류 및 BB
      - 이미지 내의 물체 크기는 랜덤하며 일부 객체는 다른 객체보다 경계가 덜 명확하므로 모든 객체의 크기 고려
      - 영역들을 그룹화하는 최적화된 단일 전략은 없으므로 색상, 재질(텍스쳐), 크기 등 다양한 조건을 함께 고려
    - 지역 제안과 객체 탐지 신경망의 분리로 정확도는 높지만 엄청난 연산량 요구
  - R-CNN 계열(R-CNN, Faster R-CNN, R-FCN, Mask R-CNN 등) 합성곱 신경망
    - Faster R-CNN은 지역 제안(RPN region proposals network)과 객체 탐지를 같은 합성곱 신경망에서 처리해 속도 향상

a) 크기가 다름
b) 색상이 다름
c) 질감이 다름
d) 크기와 색상이 다르지만 객체의 일부분



객체 탐지

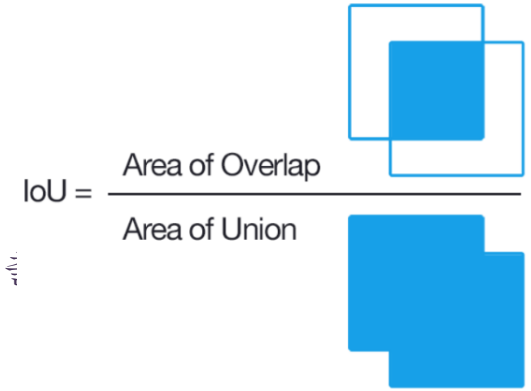
- 원샷 탐지(one-shot-detection)
  - 지역 제안 단계를 건너뛰고 최종 현지화와 객체 예측을 한 번에 산출
    - 정확도는 조금 떨어지지만 처리 속도는 매우 빠름
  - 후보 영역 추천은 세그먼테이션(segmentation)과 철저한 검색(exhaustive search) 결합
    - 세그먼테이션: 이미지 구조를 사용하여, 샘플링 프로세스 안내
    - 철저한 검색: 모든 객체의 위치를 찾음
  - SSD, RetinaNet, YOLO 합성곱 신경망 등이 있음

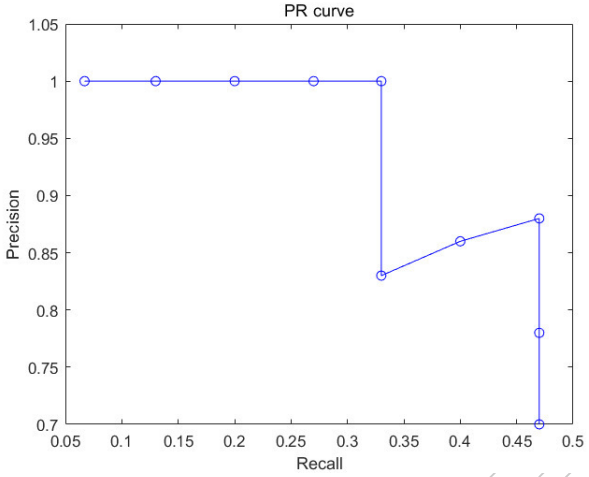


세그먼테이션

객체 탐지

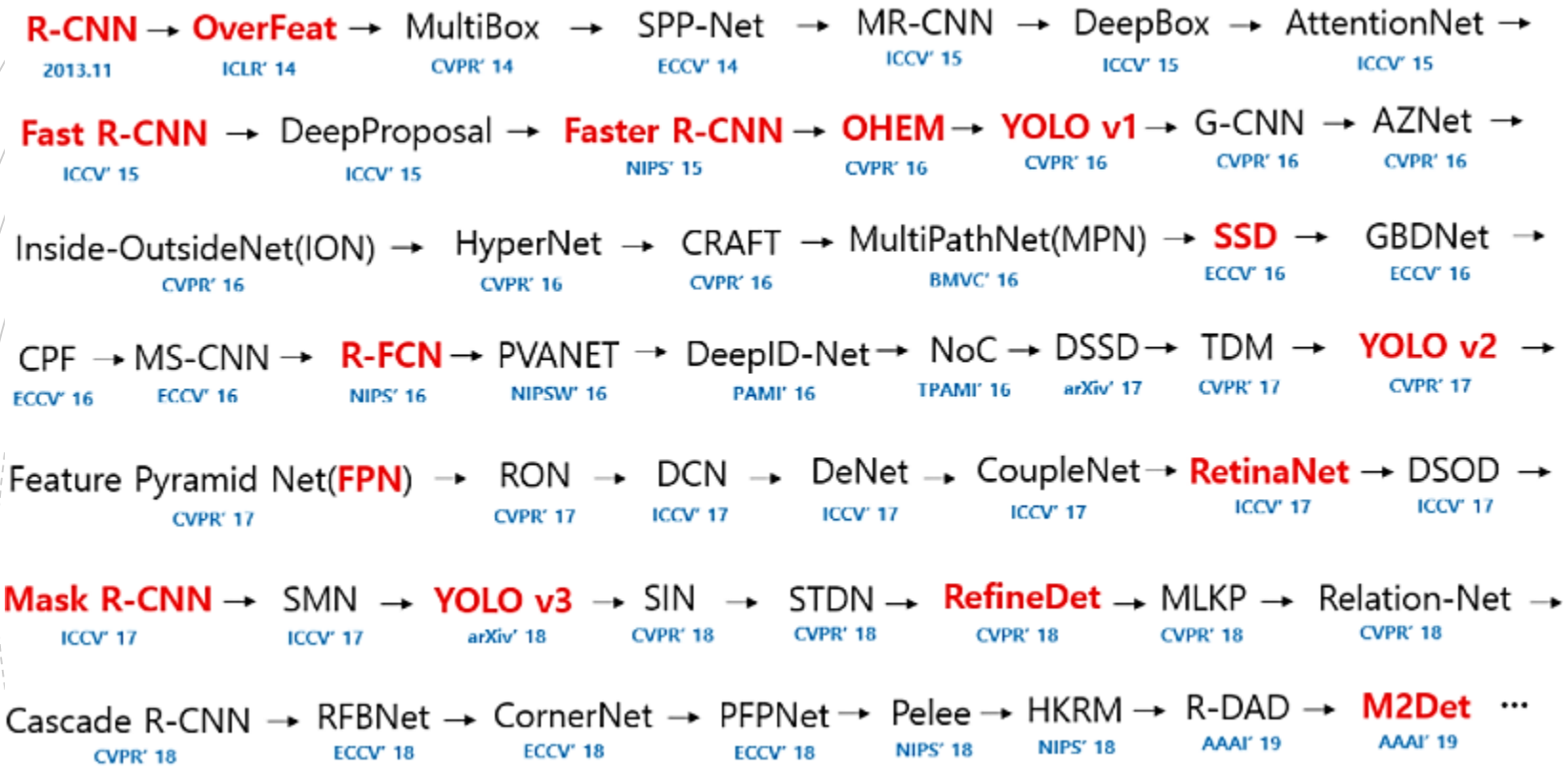
- 성능 평가 지표
  - 정밀도(Precision)는 모든 검출 결과 중 옳게 검출한 비율로 10개 중 9개를 옳게 검출했다면 0.9
    - $Precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections}$
      - TP(True Position): 검출한 결과가 옳은 것
      - FP(False Position): 검출한 결과가 틀린 것
  - 재현률(Recall)은 마땅히 검출해내야 하는 물체들 중에서 제대로 검출된 비율로 10개 중 4개만 옳게 검출
    - $Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$
      - FN(False Negative): 검출 되었어야 하는 물체인데 검출 되지 않은 것
  - IoU(Intersection of Union): 예측한 BBox와 실제 BBox의 포개진(일치) 정도
    - $IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$
  - PR(Precision-Recall) 곡선
    - 신뢰(confidence) 단계와 임계값(threshold)의 변화에 대한 정밀도와 재현률 사이 관계를 그래프로 그린 것
  - AP(Average Precision)
    - PR 곡선의 면적에 해당하는 것으로 값이 높을 수록 성능이 우수함
  - mAP(mean Average Precision)
    - 클래스가 여러 개일 때 각 클래스당 AP를 구한 후 그것을 모두 합한 다음 클래스 개수로 나눈 값





객체 탐지

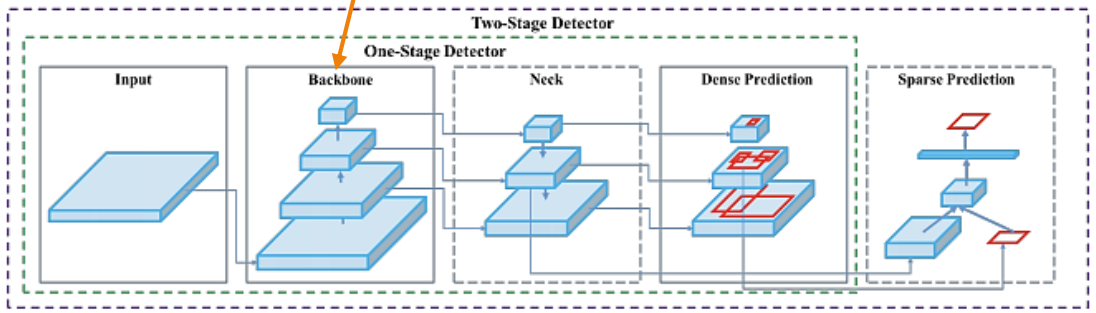
- 신경망 기반 객체 탐지 관련 논문 (전세계 유수의 비전 컴퓨팅 분야 학회)
  - 객체 탐지 기술은 의료장비를 비롯해 자율주행차, 대형마트 등 활용되는 영역이 매우 넓고, 연구도 매우 활발한 분야



ICLR: 표현 학습 국제 학회 (2013년부터 시작된 기계학습 대회)
CVPR: 컴퓨터 비전과 패턴 인식 학회(1983년부터 시작, 컴퓨터 비전 및 패턴 인식 분야 최고 권위)
ICCV: 국제 컴퓨터 비전 학회(IEEE 주최, 컴퓨터 비전 및 인공지능-패턴인식 분야에서 CVPR와 함께 최고권위 학회)
NIPS: 신경정보처리 시스템 학회(1987년부터 시작, 신경망 분야 권위 학회)
ECCV: 유럽 컴퓨터 비전 학회(ICCV와 유사)
AAAI: 전미인공지능학회(1979년부터 시작되었으며 AI 잡지 발간)

YOLO 모델

- YOLO (You Only Look Once)
  - 빠른 속도로 물체를 탐지하기 위해 만들어진 합성곱 신경망 모델로 간단한 구조와 독창적인 방법으로 높은 성능과 빠른 속도 달성
  - Darknet(https://github.com/pjreddie/darknet)은 C 및 CUDA로 작성된 오픈 소스 신경망 프레임워크 중 하나
    - 빠르고 쉽게 설치할 수 있으며 CPU 및 GPU 계산 지원
    - YOLO V1 ~ V3 저자인 Joseph Chet Redmon이 YOLO의 신경망 아키텍처 설정을 위해 개발
      - YOLO 학습을 위한 프레임워크로 사용 (Backbone)
    - YOLO V4는 Alexey Bochkovskiy

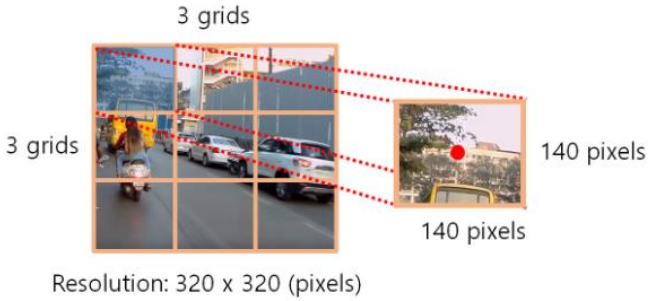


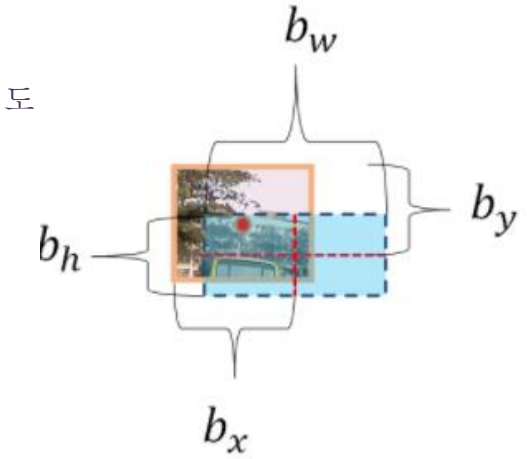
YOLO 모델

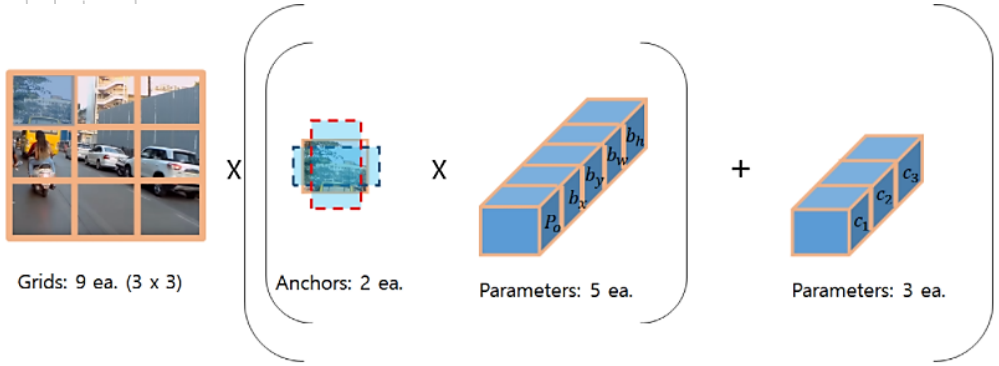
- YOLO의 역사
  - YOLO 등장 이전은 물체가 있을 만한 영역을 우선 뽑아낸 다음, 각 영역을 합성곱 신경망으로 분류와 지역화를 수행하는 투샷 탐지 방식 사용
    - 가장 빠른 모델은 Faster R-CNN으로 정확도는 73.2mAP로 나쁘지 않았지만 속도가 7fps에 그쳐 실시간으로 사용하기엔 적합하지 않음
  - YOLOv1 (2016)은 영역을 추출하는 단계 없이 바로 전체 이미지에 대한 합성곱 신경망을 통해 분류 및 지역화를 수행하는 원샷 탐지 방식 사용
    - 45fps의 속도로 Faster R-CNN보다 약 6배 향상되었으며, 경량화 모델인 Fast YOLO는 155fps의 속도로 처리 가능
    - 정확도는 63.4mAP로 Faster R-CNN보다 크게 떨어지지 않으나, 크기가 작은 물체를 잘 탐지해내지 못함
  - YOLOv2는 224x224 크기의 이미지로 200개 이상의 클래스와 서로 다른 9000개 이상의 물체 탐지
    - 배치 정규화(Batch Normalization)를 도입하고 GoogleNet 대신 DarNet-10을 백본으로 사용해 크기가 작은 물체도 잘 탐지
  - YOLOv3는 320x320 크기의 이미지를 사용해 22ms만에 51.5mAP의 정확도로 물체 탐지
  - YOLOv4는 이미지의 일부를 잘라내거나 다른 이미지의 일부를 갖다 붙이는 등의 작업을 통해 데이터셋의 과적합을 막고 더 효과적으로 데이터 학습
    - 혼동하기 쉬운 데이터를 모아 한 번 더 학습함으로써 잘못된 예측을 줄임
    - LeakyReLU, PReLU, SeLU, Mish 등의 활성화 함수를 사용해 성능 개선
    - MS COCO 데이터셋에서 43.5%의 AP와 65fps를 달성하여 거의 실시간에 가깝게 객체 탐지

YOLO 모델

- 기본 개념
  - 격자(grid) 분할
    - 여러 격자로 이미지를 S x S로 분할하면 각 셀마다 크기와 중심점을 갖게 됨
      - 클래스 확률(class probability)은 셀에 각 클래스가 포함될 확률
  - 앵커(Anchor) 박스
    - 격자는 여러 개의(일반적으로 2개) 앵커 박스를 가짐
      - BBox는 예측한 객체의 위치를 나타내고(출력), 앵커 박스는 탐지할 객체의 가정 모양(입력)으로 구분하기도
      - P_o: 신뢰도 점수 (confidence score)로 해당 셀에서 물체가 있을 확률
      - b_x, b_y: 해당 셀의 x, y 값으로 셀의 왼쪽 위 모서리 기준 거리
      - b_w, b_h : 해당 셀의 앵커 박스 폭과 높이
  - 최종 매개변수
    - 격자 수 x (앵커 수 x 앵커 매개변수) x 각 클래스 확률 매개변수

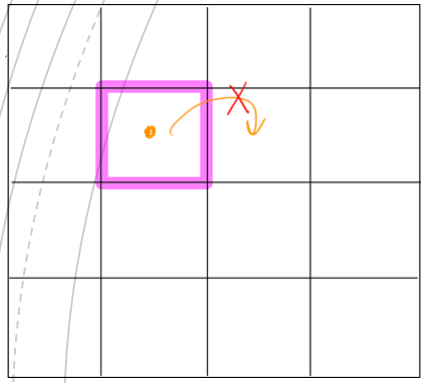


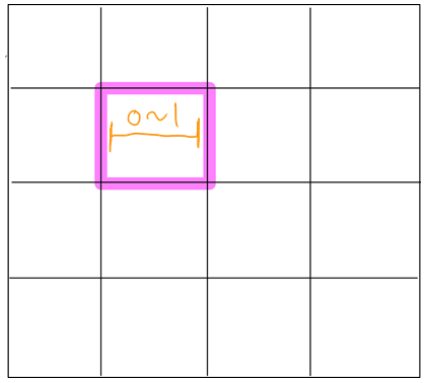


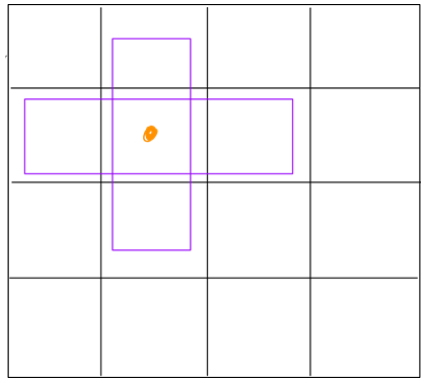


YOLO 모델

- 격자 개념은 앵커 박스 중심점 위치를 특정 셀 안으로 제한
  - 이미지를 4x4 격자로 나누고 앵커 박스의 중심점을 한 칸안에서만 움직일 수 있도록 제한하면 모든 셀에 대해 일정한 간격으로 앵커를 두는 효과를 냄
  - 모델은 셀 안에서 작은 움직임만으로 정답에 가깝게 학습할 수 있어 인식률에 대한 부담이 줄어듦
  - 가로 또는 세로 피사체 모두를 구분하기 위해 각 셀마다 가로형, 세로형 앵커 박스를 가짐

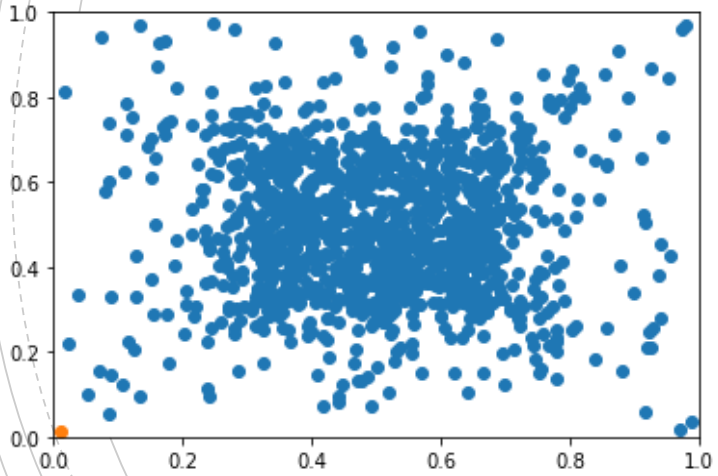


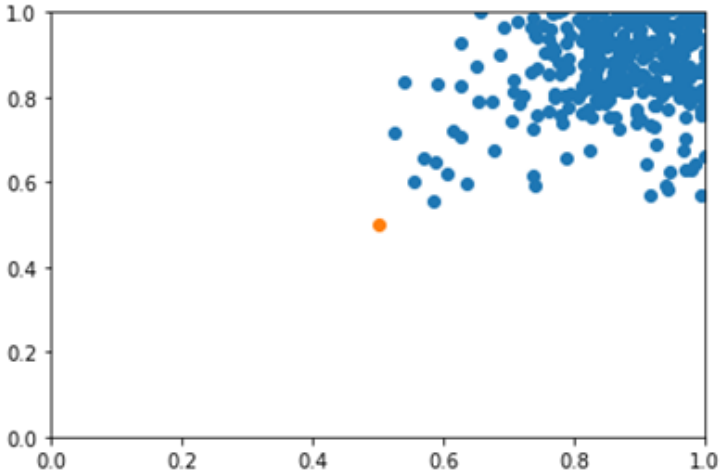




YOLO 모델

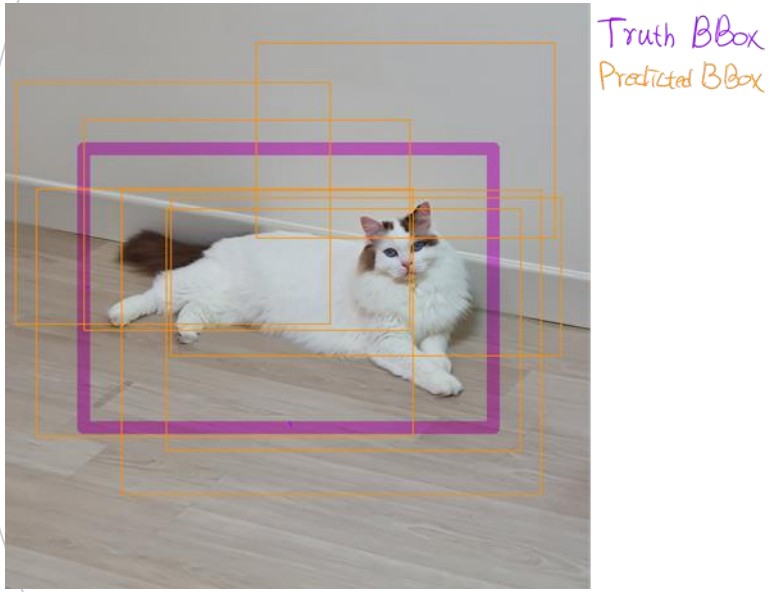
- 격자와 앵커 박스를 사용하는 이유
  - 좌표를 0 ~ 1 사잇값으로 표현할 때, 이미지 중앙에 위치한 피사체는 (0.5, 0.5)로 표현
    - 시그모이드로 결과를 출력하면 분포가 0.5 주변에 밀집할 가능성이 높음
    - 피사체가 중앙에서 멀어질수록 모델에서 학습 또는 인식할 가능성이 낮아지는 문제 발생
  - 여러 개의 앵커를 다양한 곳에 두면 분포가 한곳에 밀집하는 현상 완화
    - 앵커들의 위치를 실제 데이터 분포에 맞게 지정하면 더 좋은 결과 기대
    - 실제 데이터 위치 분포에 대해 K-평균 클러스터링(K-means Clustering) 값으로 앵커 위치 결정
    - 위치 뿐만 아니라 크기까지 적용하면 앵커 박스
  - 앵커 박스는 모델이 예측해야 할 영점 제시

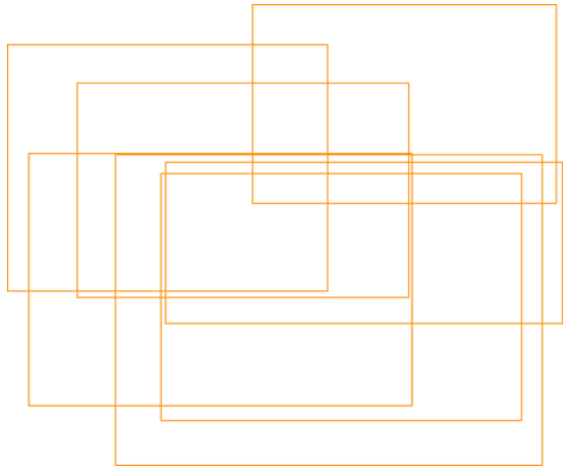




YOLO 모델

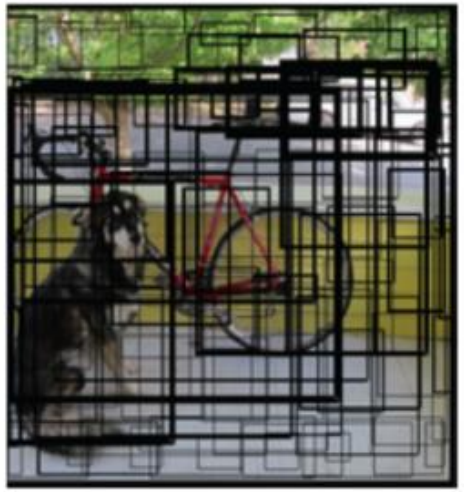
- 신뢰도
  - 객체 탐지 모델은 정답 BBox를 맞추기 위해 수 많은 BBox 생성
    - 모델 입장에서는 정답 BBox와 비슷한 예측 BBox를 최대한 많이 생성해야 높은 점수를 받을 가능성이 높아짐
  - 많은 예측 BBox가 겹쳐지면 어떤 BBox가 정답에 가까운지 파악하기 어려움
    - 모델 입장에서는 '정답' 이라는 정보가 존재 하지 않기 때문에 어떤 BBox가 정답인지 판단할 수 없음
  - 이 문제를 해결하고자 신뢰도와 비-최대 억제(NMS non-Max Suppression) 사용



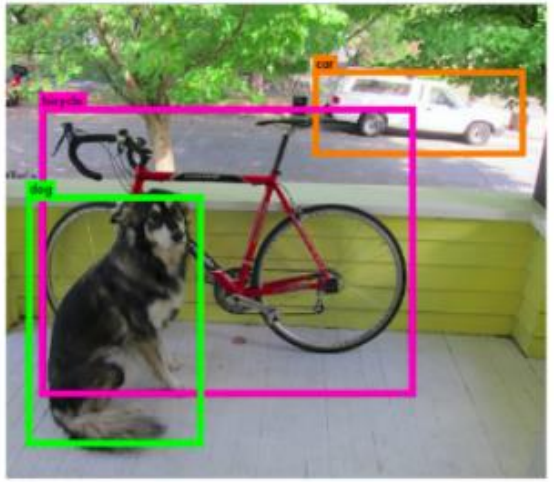


YOLO 모델

- 비-최대 억제
  - 예측한 BBox 중에서 정확한 BBox를 선택하도록 하는 기법
    - 이미지 안에서 객체는 다양한 크기와 형태로 존재하기 때문에 모델은 여러 개의 BBox를 생성하는데, 비-최대 억제는 이 중 하나의 BBox만 선택하도록 함
  - 알고리즘
    - 동일한 클래스에 대해 신뢰성 점수 순으로 정렬
    - 가장 신뢰도가 높은 BBox를 목록에서 선택
    - 선택된 BBox와 나머지 BBox의 IoU를 차례로 비교
      - 결과가 임계값보다(보통 50%) 높으면 같은 객체를 탐지한 것으로 보고 해당 BBox 제거
    - 다음 신뢰도가 높은 BBox를 목록에서 선택 후 IoU 비교 및 제거 반복



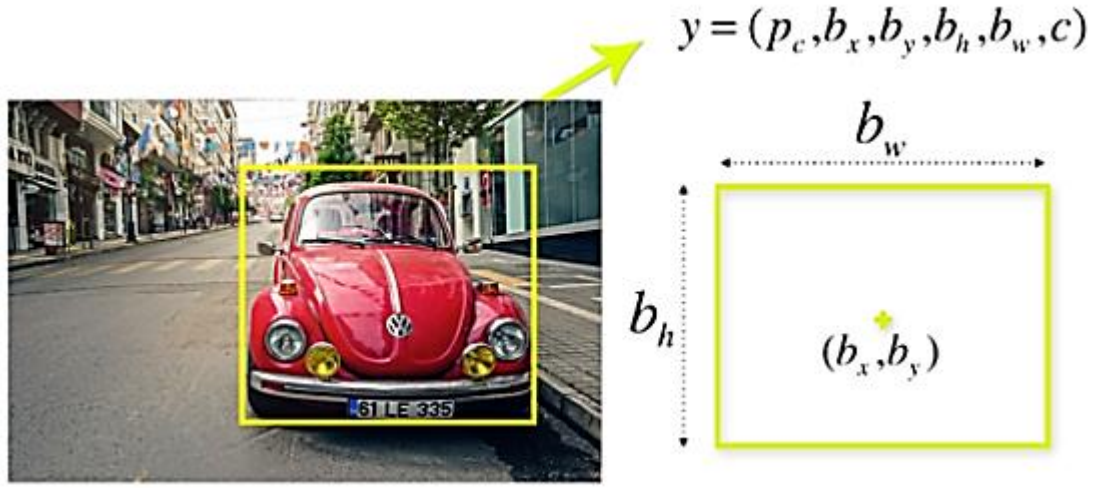
Multiple Bounding Boxes



Final Bounding Boxes

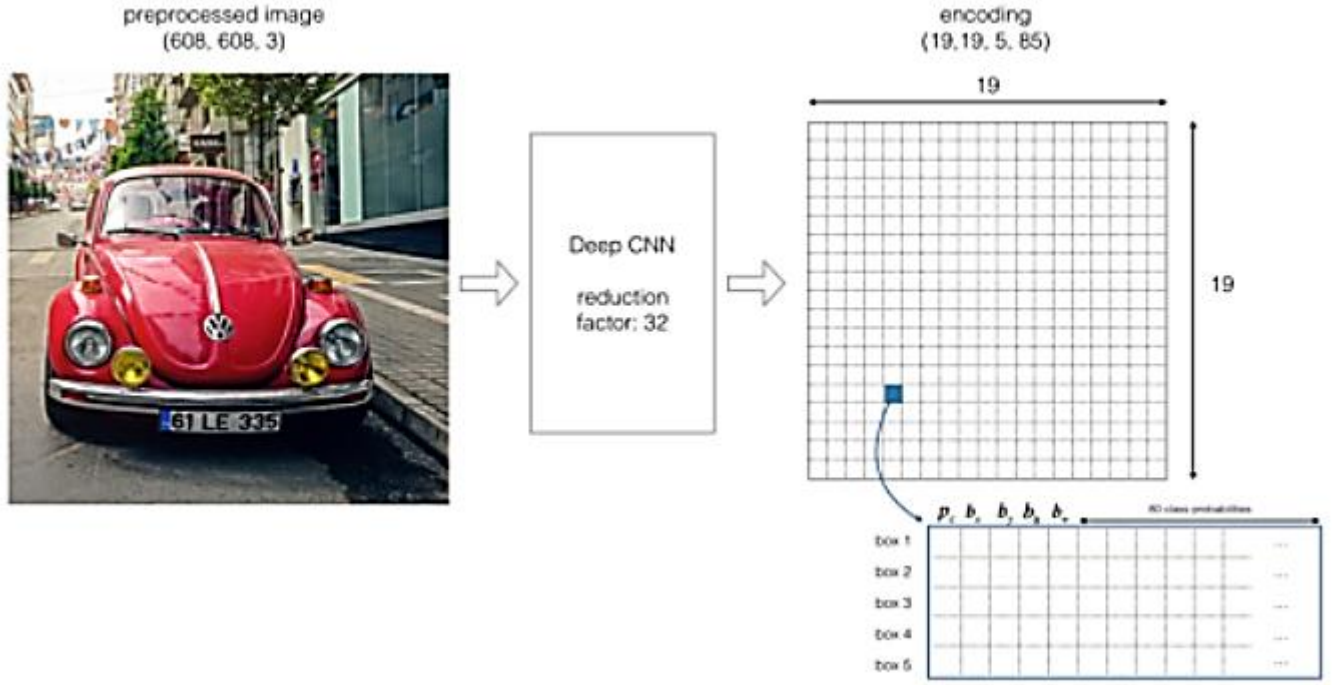
YOLO 모델

- YOLO V4 작동 방식
  - YOLO 탐색기는 객체의 클래스, BBox, BBox에 있는 객체 클래스의 확률 예측
  - 각 BBox 매개변수
    - 이미지에서 Bbox의 중심 위치 (b_x, b_y)
    - BBox 너비와 높이 (b_w, b_h)
    - 객체의 클래스(c)



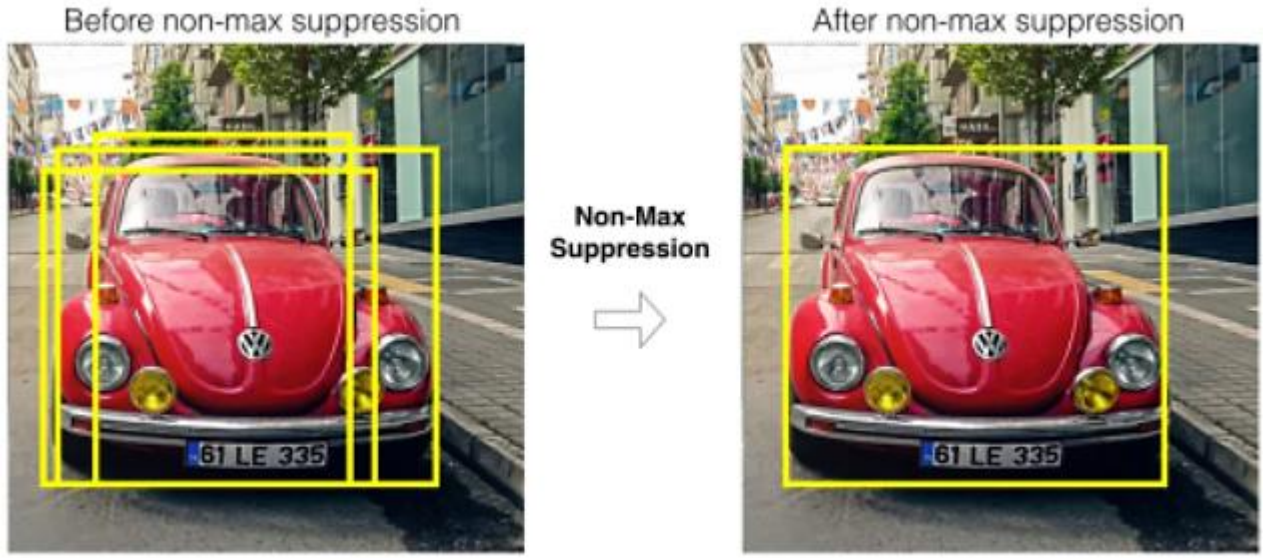
YOLO 모델

- 각 BBox는 확률 값(p_c)과 연관되며, 이는 해당 BBox에 있는 객체 클래스의 확률
  - YOLO는 흥미로운 ROI를 찾는 대신 일반적으로 19x19 격자를 사용하여 이미지를 여러 셀로 분할
  - 각 셀은 5개의 BBox를 예측하는 역할 수행
    - 셀에 하나 이상의 개체가 있을 수 있음
  - 하나의 이미지에 대해 1805개의 BBox가 생성됨



YOLO 모델

- 셀에 있는 대부분의 BBox에는 개체가 없을 수 있음
  - BBox 필터링은 개체 클래스 p_c 확률을 기반으로 수행
  - 비-최대 억제 프로세스는 원하지 않는 BBox를 제거하고 가장 높은 확률의 Bbox만 남김



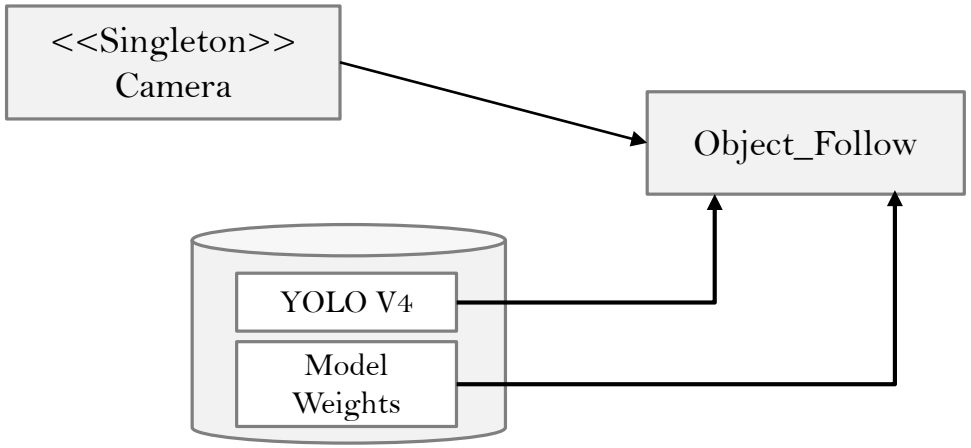
YOLO 모델

- 다크넷 프레임워크는 COCO 데이터셋 클래스 91개 중 80개 클래스 제공 (Object 2017)
  - 11개(stop sign, hat, shoe, eye glasses, plate, mirror, window, desk, door, blender, hair burush) 제외

0	person	10	fire hydrant	20	elephant	30	skis	40	wine glass	50	broccoli	60	dining table	70	toaster
1	bicycle	11	stop sign	21	bear	31	snowboard	41	cup	51	carrot	61	toilet	71	sink
2	car	12	parking meter	22	zebra	32	sports ball	42	fork	52	hot dog	62	tv monitor	72	refrigerator
3	motorbike	13	bench	23	giraffe	33	kite	43	knife	53	pizza	63	laptop	73	book
4	aero plane	14	bird	24	backpack	34	baseball bat	44	spoon	54	donut	64	mouse	74	clock
5	bus	15	cat	25	umbrella	35	baseball glove	45	bowl	55	cake	65	remote	75	vase
6	train	16	dog	26	handbag	36	skateboard	46	banana	56	chair	66	keyboard	76	scissors
7	truck	17	horse	27	tie	37	surfboard	47	apple	57	sofa	67	cell phone	77	teddy bear
8	boat	18	sheep	28	suitcase	38	tennis racket	48	sandwich	58	potted plant	68	microwave	78	hair drier
9	traffic light	19	cow	29	frisbee	39	bottle	49	orange	59	bed	69	oven	79	toothbrush

YOLO로 객체 탐지

- 실습장비는 YOLO V4 모델 기반 객체 탐지 라이브러리 지원
  - 카메라는 pop.Pilot.Camera 클래스 사용
  - pop.Pilot.Object_Follow: 객체 탐지 클래스
    - YOLO V4 기반 COCO 데이터셋로 학습한 다크넷 프레임워크 탐지 모델(매개변수) 제공



YOLO로 객체 탐지

- 객체 탐지 클래스
  - Object_Follow(camera, jupyter=True): 객체 탐지 객체 생성
    - camera: 카메라 객체
    - jupyter: 주피터 환경일 때, 노트북의 Image 위젯과 카메라 프레임 연결 유무. 기본값은 연결
  - Object_Follow.load_model(): 학습된 매개변수 로드
  - Object_Follow.detect(image=None, index=None, show=False): 객체 탐지
    - image: 탐지할 이미지, 생략하면 카메라 프레임
    - Index: 탐지할 클래스 구분. 생략하면 모든 클래스 탐지
    - show: True이면 이미지에 Bbox 및 라벨 표시
    - 반환값은 딕셔너리 타입으로 클래스에 대한 탐지 정보
      - 키 x, y의 값은 이미지에서 탐지 객체의 상대적 좌표로, 화면 정 중앙(0, 0)을 기준으로 -1 ~ 1 사잇값
      - 키 size_rate의 값은 0 ~ 1 사잇값으로 객체의 크기. 1에 가까울 수록 거리가 가깝다고 해석할 수 있음
  - Object_Follow.show(): 주피터 환경일 때 Image 위젯을 이용해 카메라 프레임과 탐지 결과 표시

YOLO로 객체 탐지

- 객체 탐지 응용
  - 카메라 객체를 인자로 객체 탐지 객체 생성
  - 학습된 모델 매개변수를 로드 한 후 detect()로 해당 클래스의 객체 탐지

```
01: from pop import Camera, Pilot
02:
03: cam = Camera(width=224, height=224)
04: of = Pilot.Object_Follow(cam)
05:
06: of.load_model()
07:
08: ret = of.detect(index='person')
09: print(ret)
10:
11: ret = of.detect()
12: print(ret)
13:
14: #of.show() #only jupyter
```

YOLO로 객체 탐지

- 실습장비가 사람을 따라 다니도록 구현
  - 이동체 움직임과 Object_Follow 객체의 person 클래스 탐지 기능 결합
  - detect()가 반환한 x, size_rate 키에 대응하는 값을 이용해 조향 및 거리 조절

```
01: from pop import Pilot, Camera
02:
03: cam = Camera(width=224, height=224)
04: bot = Pilot.SerBot()
05:
06: of = Pilot.Object_Follow(cam)
07: of.load_model()
08:
09: while True:
10:     ret = of.detect(index='person')
11:
12:     if ret is not None:
13:         steer = ret['x'] * 4
14:         bot.steering = 1 if steer > 1 else -1 if steer < -1 else steer
15:
16:         if ret['size_rate'] < 0.20:    #실제 이 값으로 대상이 멀어졌다고 판단하기 어려움
17:             bot.forward(50)
18:         else:
19:             bot.stop()
20:     else:
21:         bot.stop()
```