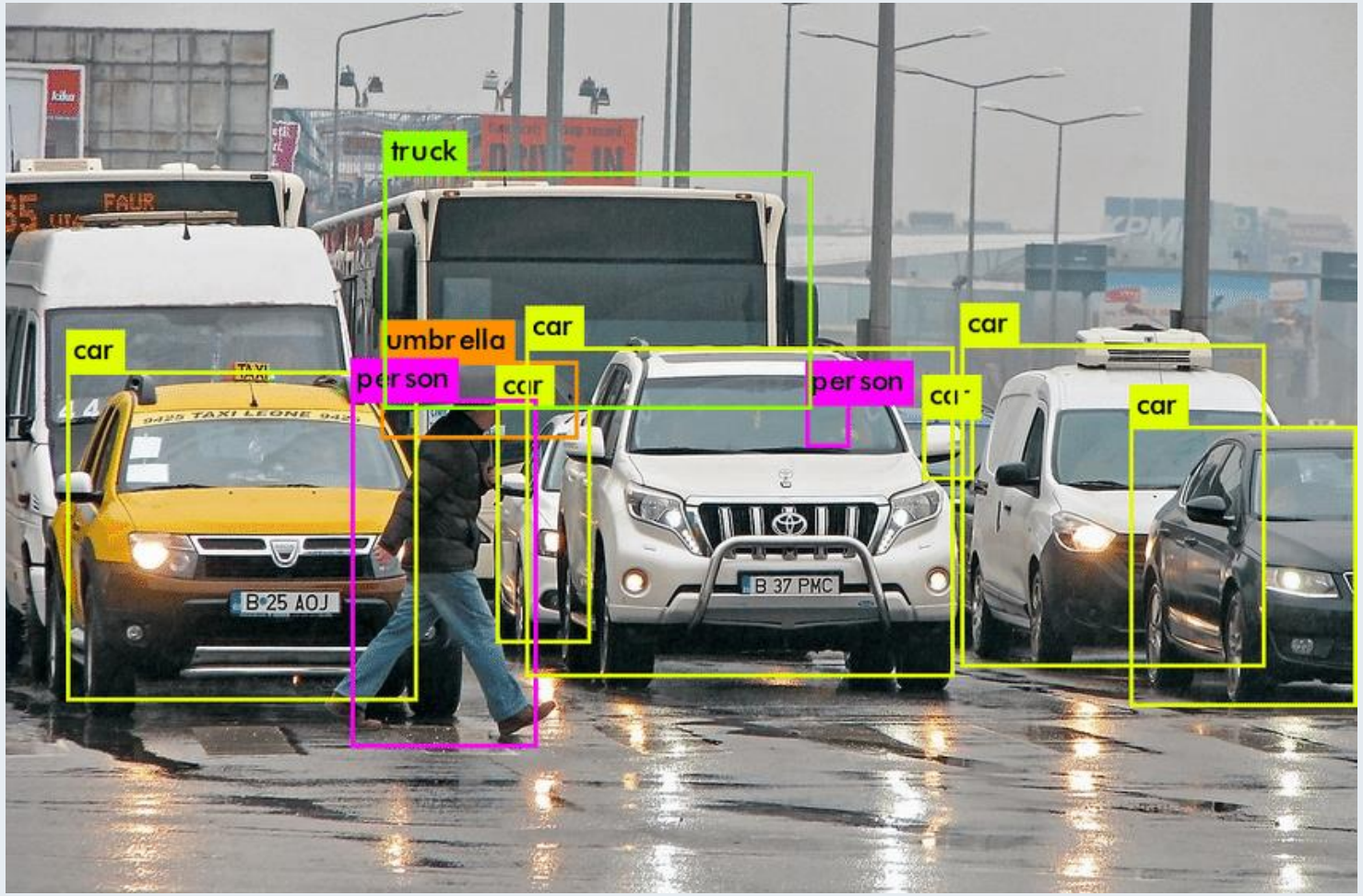
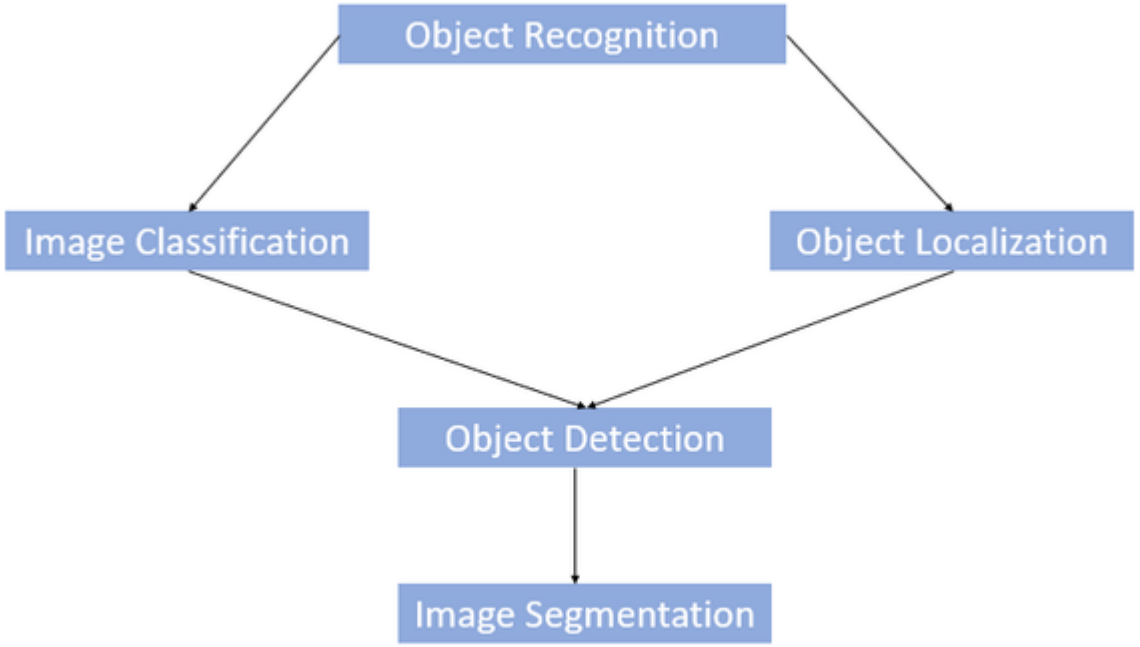
객체 탐지 (Object Detection)

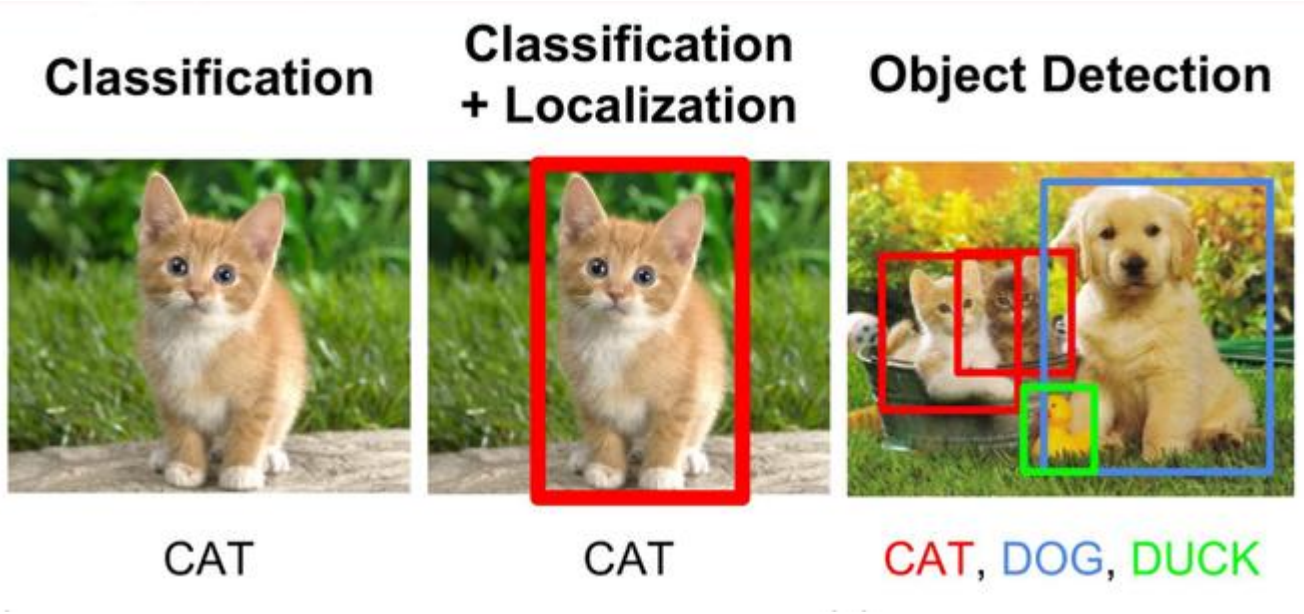


박경규

이미지 분류 vs 객체 탐지

이미지 분류는 이미지에 있는 대상이 각 클래스에 속할 확률만 출력하는 것이고,
객체 탐지는 발견한 물체의 위치를 특정하는 박스의 좌표와 각 대상의 클래스를 함께 예측합니다.





객체 탐지 프레임워크

1. 영역 제안(Region Proposal)

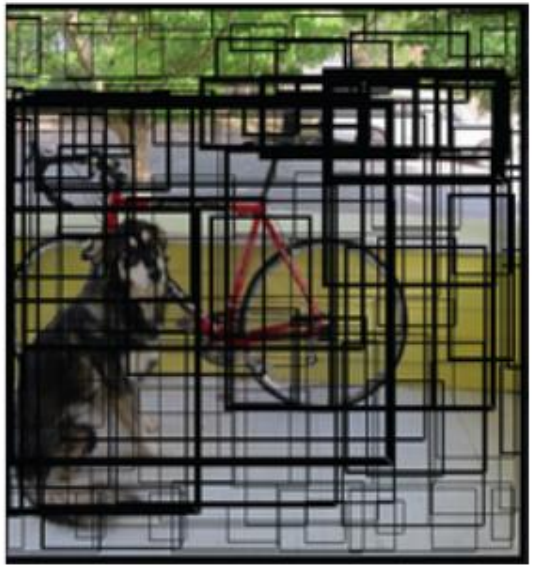
- 시스템이 이미지를 관찰하고 추가 분석이 필요한 ROI(Region Of Interest)를 제안
- ROI는 시스템이 이미지의 해당 위치에 높은 확률(Objectness Score)로 물체가 존재한다고 판단한 영역

2. 특정 추출 및 예측

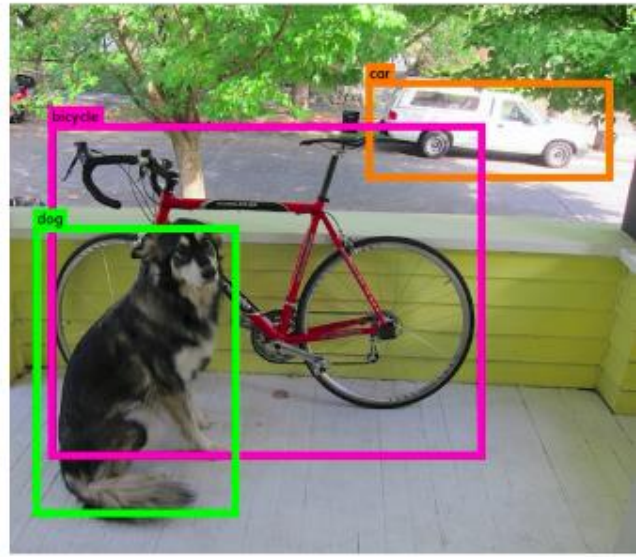
- 물체를 포함했을 가능성이 높다고 판단된 모든 영역을 신경망이 분석하여 경계박스와 클래스를 예측
- 제안되는 영역의 수가 많기 때문에 물체마다 여러 개의 경계박스가 생김

3. 비최대 억제(NMS: Non-Maximum Suppression)

- 탐지 알고리즘은 같은 물체를
  여러 번 탐지할 수 있고, 물체 하나에
  여러 개의 경계박스가 표시될 수 있음
- 비최대 억제는 예측확률이 가장 높은
  경계박스만 남기고 나머지는 배제하는 방식



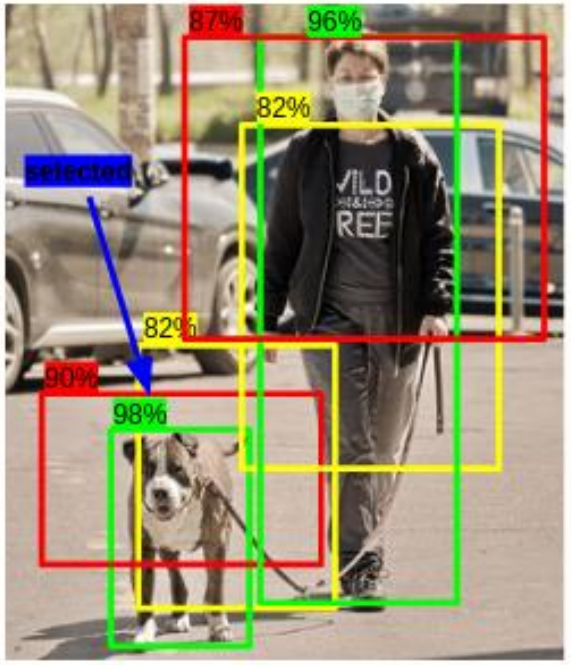
Multiple Bounding Boxes



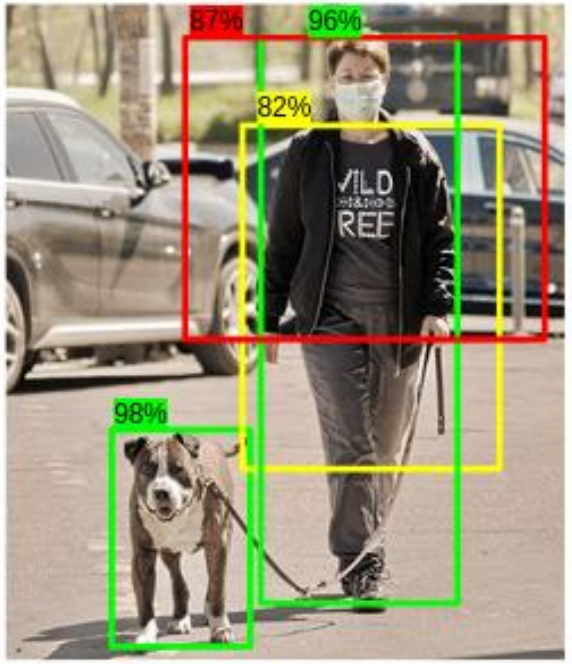
Final Bounding Boxes

비최대 억제(NMS) 알고리즘

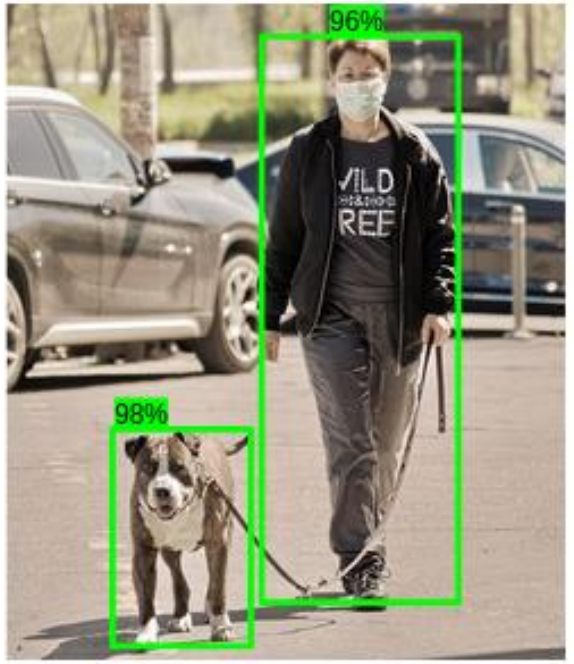
- 1단계: Objectness Score가 가장 높은 상자 선택
- 2단계: 이 상자의 중첩(합집합에 대한 교차)을 다른 상자와 비교
- 3단계: 겹침(합집합에 대한 교차) >50% 가 있는 경계 상자 제거
- 4단계: 다음으로 높은 Objectness Score 로 이동
- 5단계: 2-4단계를 반복



Step 1: Selecting Bounding box with highest score



Step 3: Delete Bounding box with high overlap



Step 5: Final Output

https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

객체 탐지 알고리즘 평가지표

■ 초당 프레임수 : FPS(Frame Per Second)

- 탐지속도를 평가하는 일반적인 지표 : R-CNN 7 FPS, SSD 59FPS

■ 평균정밀도 : mAP(mean Average Precision)

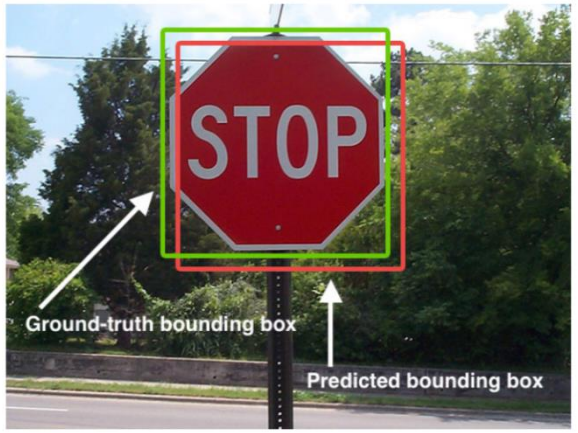
- 0~100 사이의 값으로 값이 클수록 성능이 좋음

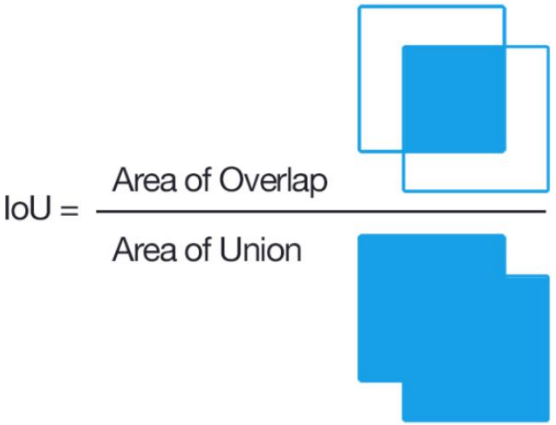
■ 중첩률 : IoU(Intersection over Union)

- 2개의 경계 박스가 중첩되는 정도를 나타내는 값

■ PR곡선 : Precision Recall curve

- confidence 레벨에 대한 threshold 값의 변화에 의한 객체 탐지기의 성능을 평가





https://bit.ly/3ofp6ws https://deep-learning-study.tistory.com/402

COCO Dataset

COCO 데이터셋은 Object Detection, Segmentation, Keypoint Detection 등을 위한 데이터셋으로,
매년 다른 데이터셋으로 전 세계의 여러 대학/기업이 참가하는 대회에 사용되고 있습니다.

```
"images": [
  ...
  {
    "license": 1,
    "file_name": "000000324158.jpg",
    "coco_url": "http://images.cocodataset.org/val2017/000000324158.jpg",
    "height": 334,
    "width": 500,
    "date_captured": "2013-11-19 23:54:06",
    "flickr_url": "http://farm1.staticflickr.com/169/417836491_5bf8762150_z.jpg",
    "id": 324158
  },
  ...
],
```

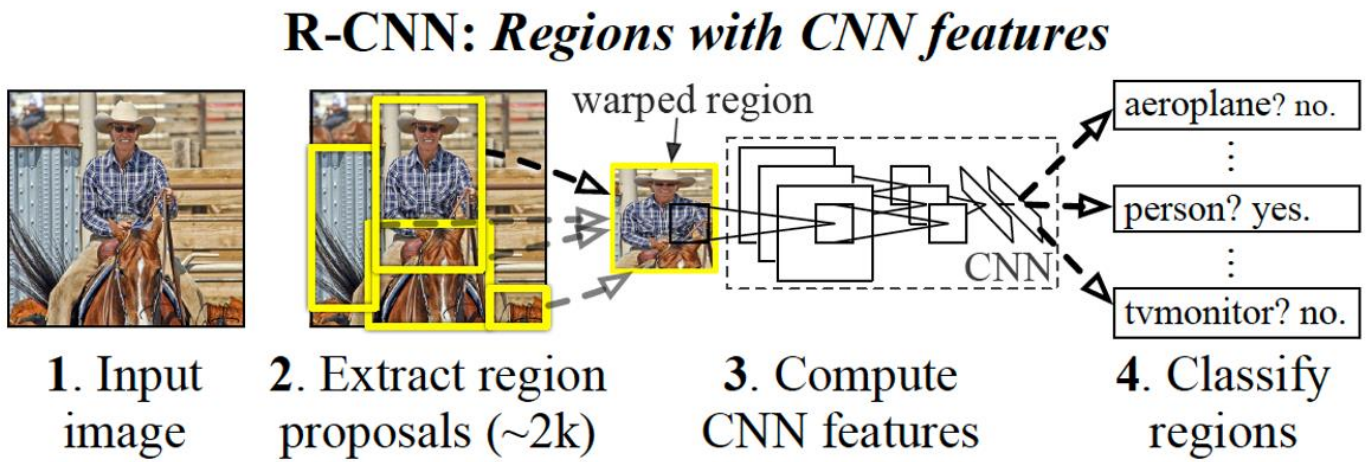
```
"annotations": [
  ...
  {
    "segmentation": [
      [
        216.7,
        211.89,
        216.16,
        217.81,
        215.89,
        220.77,

        ...
        212.16
      ]
    ],
    "area": 759.3375500000002,
    "iscrowd": 0,
    "image_id": 324158,
    "bbox": [
      196.51,
      183.36,
```

https://ukayzm.github.io/cocodataset/

R-CNN (Region-based Convolutional Neural Network)

선택적 탐색 알고리즘을 사용해서 RoI(2,000개 이상)를 추출함, 별도의 힙성곱 신경망을 사용해서 각 RoI 에서 특징을 추출하고 경계박스 예측과 클래스 분류를 수행함, 2014년에 Ross Girshick이 제안 (이미지 처리시간 50초)



■ R-CNN 구성요소

- RoI 추출기 : 선택적 탐색 알고리즘으로 물체를 포함하고 있을 가능성이 높은 이미지 영역을 고정된 크기로 추출
- 특징 추출 모듈 : 사전 학습된 합성곱 신경망에 제안된 RoI 영역을 입력해서 특징을 추출
- 분류 모델 : 서포트벡터머신 등 기존 머신러닝 알고리즘으로 분류기를 학습한 다음, 특징추출모듈에서 추출한 특징으로 해당 영역의 물체가 무엇인지 분류
- 위치 특정 모듈(경계박스 회귀 모듈. Bounding-box Regressor) : 경계 박스의 위치와 크기 예측(x, y, w, h)

https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

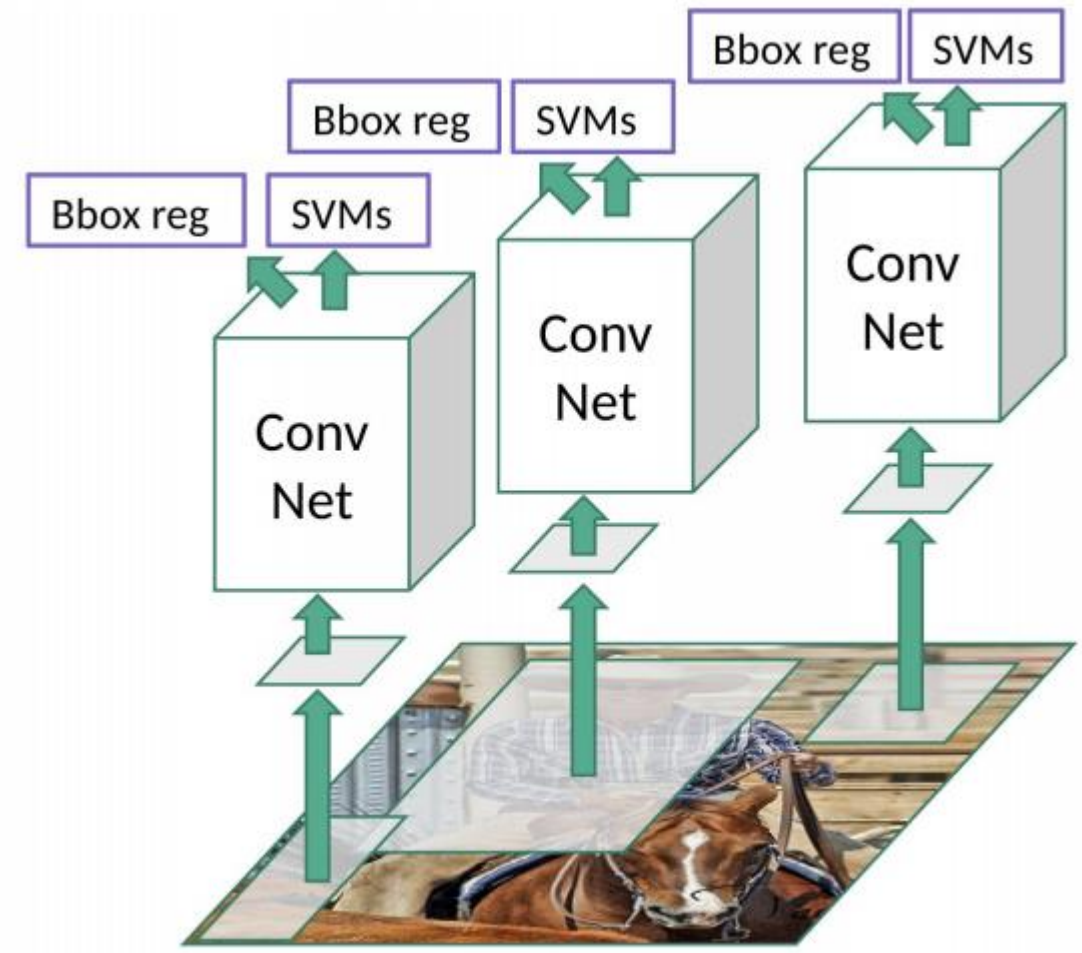
R-CNN (Region-based Convolutional Neural Network)

■ R-CNN 학습

- 특징 추출 모듈로 사용할 합성곱 신경망 학습,
  사전 학습된 신경망을 미세조정해서 사용
- SVM 분류기를 학습
- 경계 박스 회귀 모듈을 학습

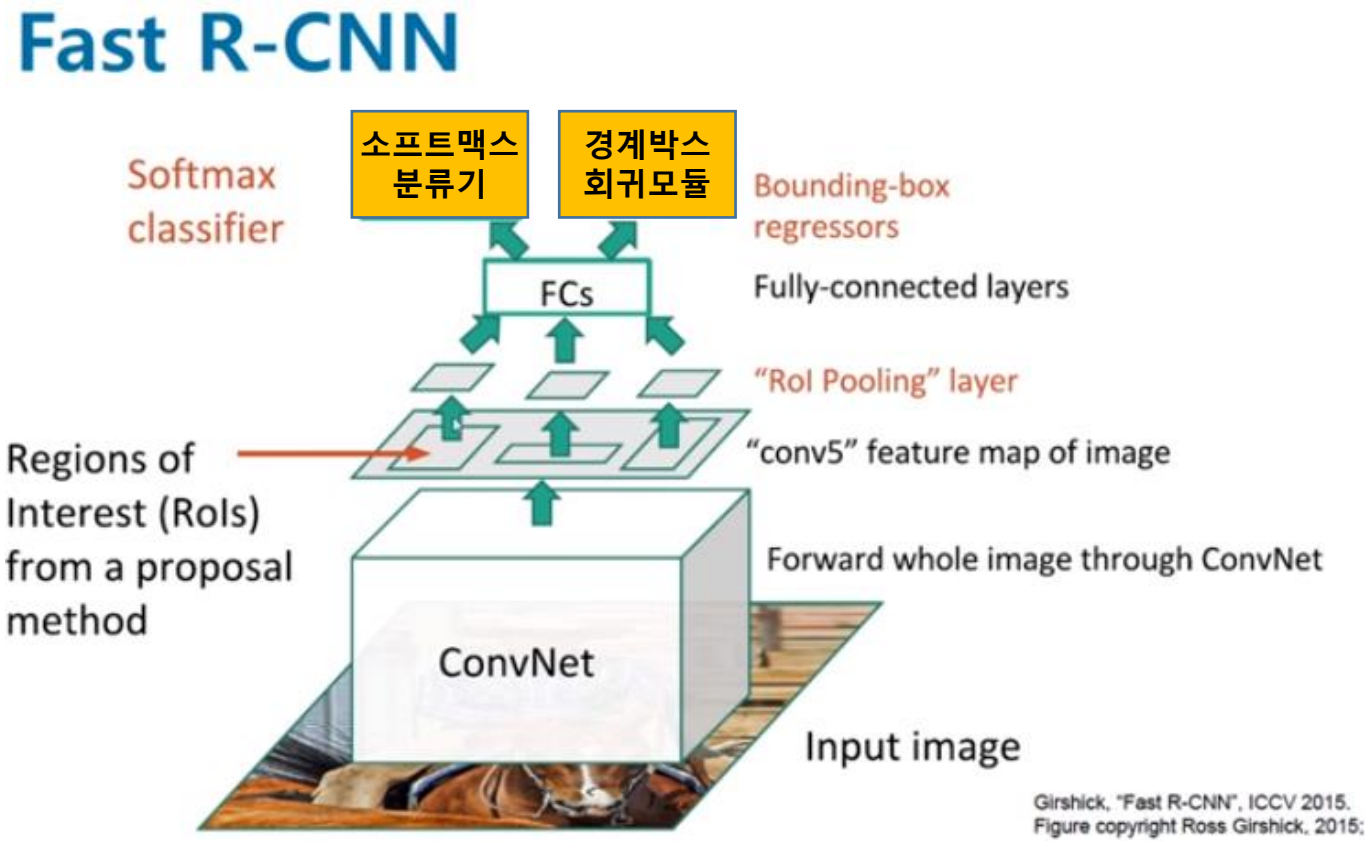
■ R-CNN 단점

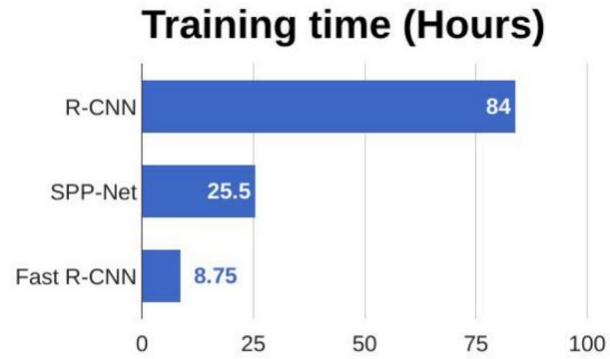
- 사물탐지 속도 느림 : 이미지 1장당 2,000 이상의 RoI가 제안됨
  RoI 하나마다 CNN 순방향 계산 필요
- 다단계 파이프라인으로 구성된 학습과정 : 학습 비용과 시간이
  많이 소요되고 이미 계산한 결과를 재사용할 수 없음
- 학습 공간 및 시간복잡도가 높음 : 수천장의 이미지를 학습하는
  시간이 수일 걸리고, 추출된 특징을 저장하기 위해
  필요한 디스크 용량이 수백GB 임

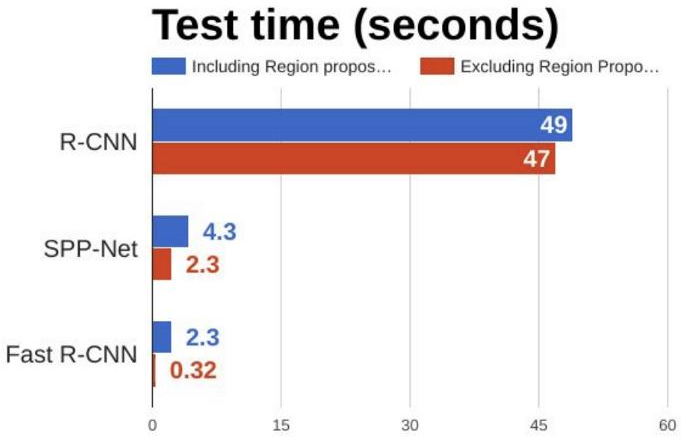


Fast R-CNN

CNN 특징 추출기를 맨 앞에 배치해서 이미지를 하나의 CNN으로 처리하고 소프트맥스층으로 분류하여
학습속도가 빠르나, 영역 제안을 생성하는 선택적 탐색 알고리즘은 여전히 속도가 느림 (이미지 처리시간 2초)

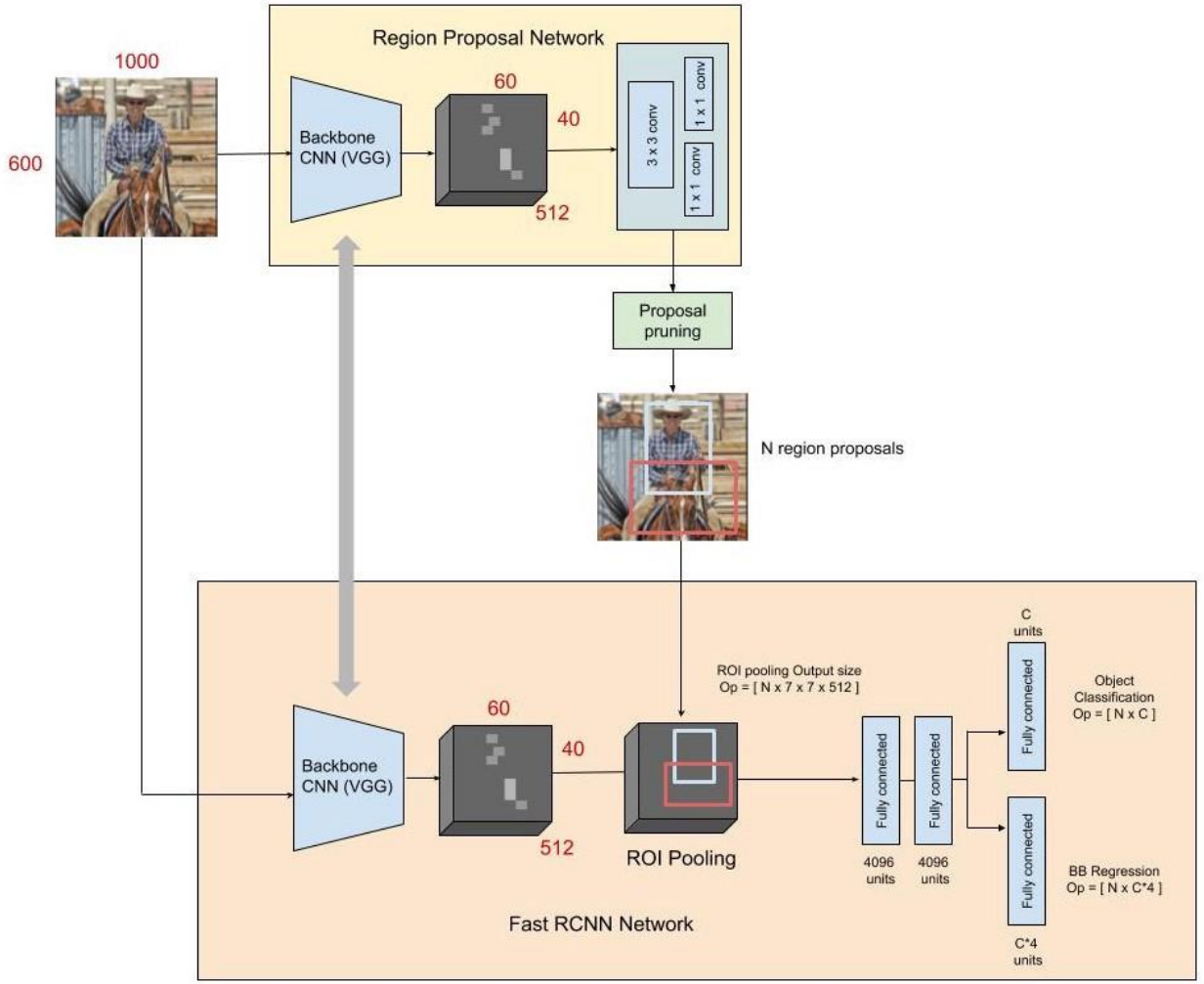






Faster R-CNN

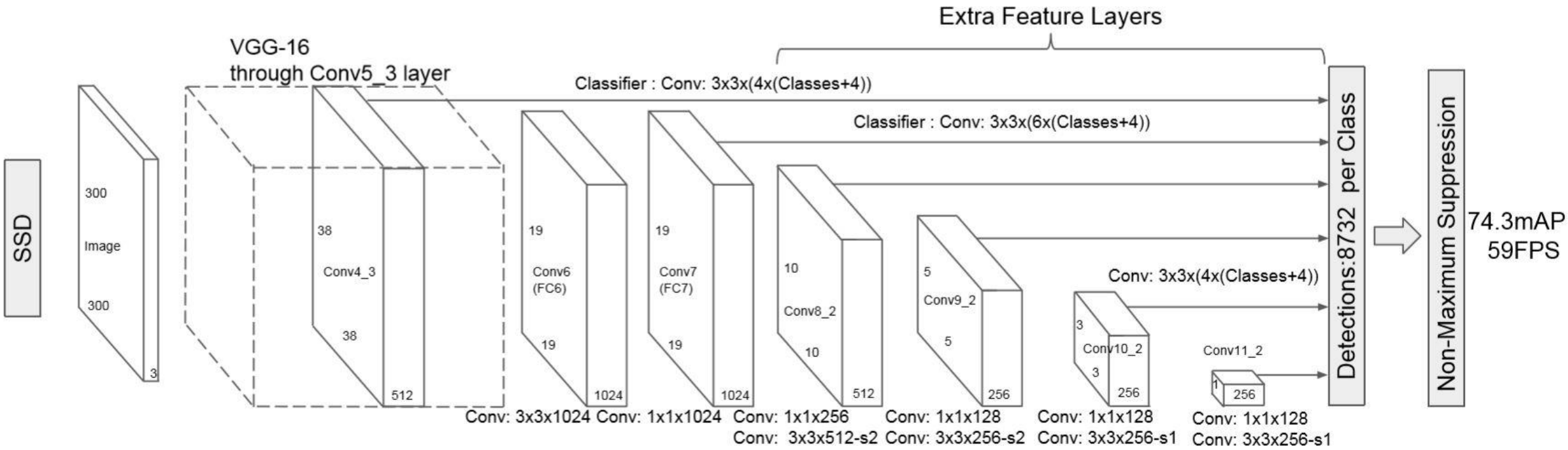
선택적 탐색 알고리즘을 영역 제안 신경망(Region Proposal Network)으로 대체해서 속도를 개선
전체 처리 과정을 처음부터 끝까지 딥러닝으로 구현, 2016년 Shaoqing Ren이 제안 (이미지 처리시간 0.2초)



https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46

SSD(Single-Shot Detector)

기본 신경망(VGG16), 물체를 탐지하는 추가 합성곱층, 최종 탐지 결과를 선정하는 비최대 억제(NMS)층으로 구성
합성곱층 7, 8, 9, 10, 11의 예측결과가 직접 NMS 층으로 전달되는 구조



PASCAL VOC와 MS COCO 데이터셋을 대상으로 59FPS의 속도와 mAP 74% 성능을 보임

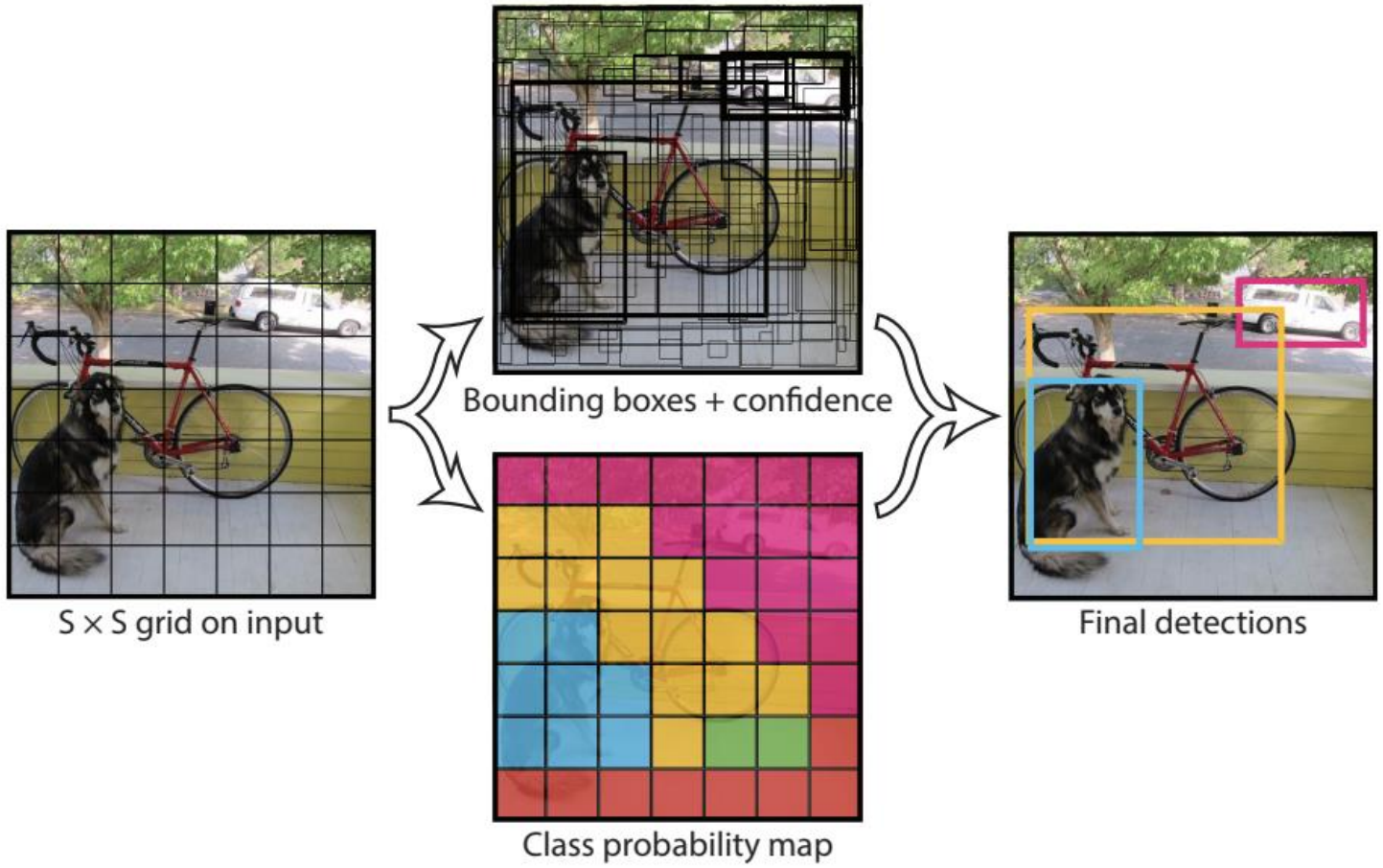
단일단계 탐지기(싱글샷) : 합성곱층에서 위치와 클래스를 한번에 예측

다단계 탐지기(R-CNN) : 경계박스 내 영역의 물체 존재 확신도를 예측한 다음 이 경계박스를 분류기로 넘겨 물체의 클래스를 예측

https://taeu.github.io/paper/deeplearning-paper-ssd/

YOLO (You Look Only Once)

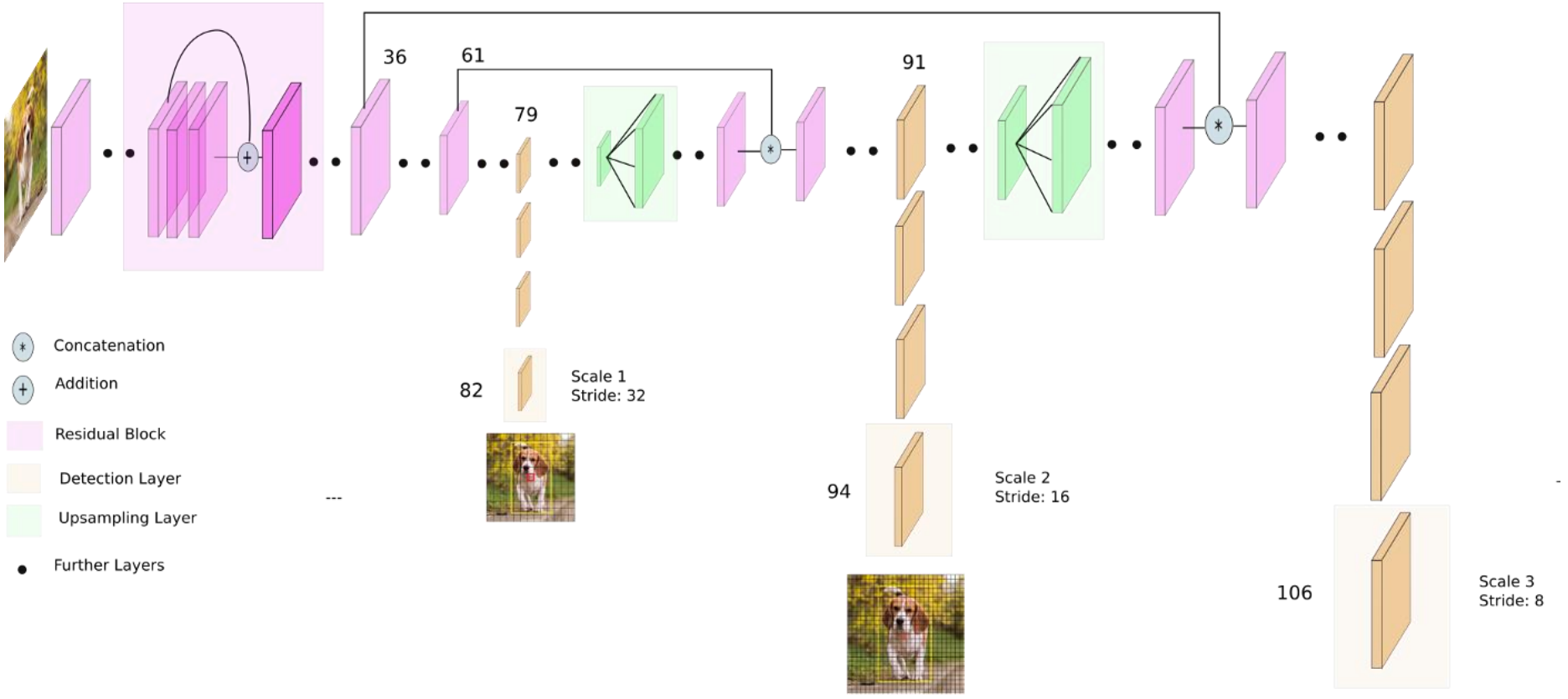
영역 제안 단계가 없으며 그 대신 입력을 격자형태로 분할하고, 분할된 영역을 대상으로 경계박스와 사물 분류 예측을 한 다음, NMS를 이용해서 최종 결과를 좁히는 방식, 2016년 Joseph Redmon이 제안



https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

YOLO (You Look Only Once)

신경망 구조는 GoogLeNet(인셉션)의 특징 추출기를 참고 하였으며, 인셉션 모듈 대신 1×1 축소층과 3×3 합성곱층 구조를 사용함(DarkNet, YOLOv3은 darknet-53 채용)



YOLO v3 network Architecture

DarkNet

YOLO https://pjreddie.com/darknet/yolo/

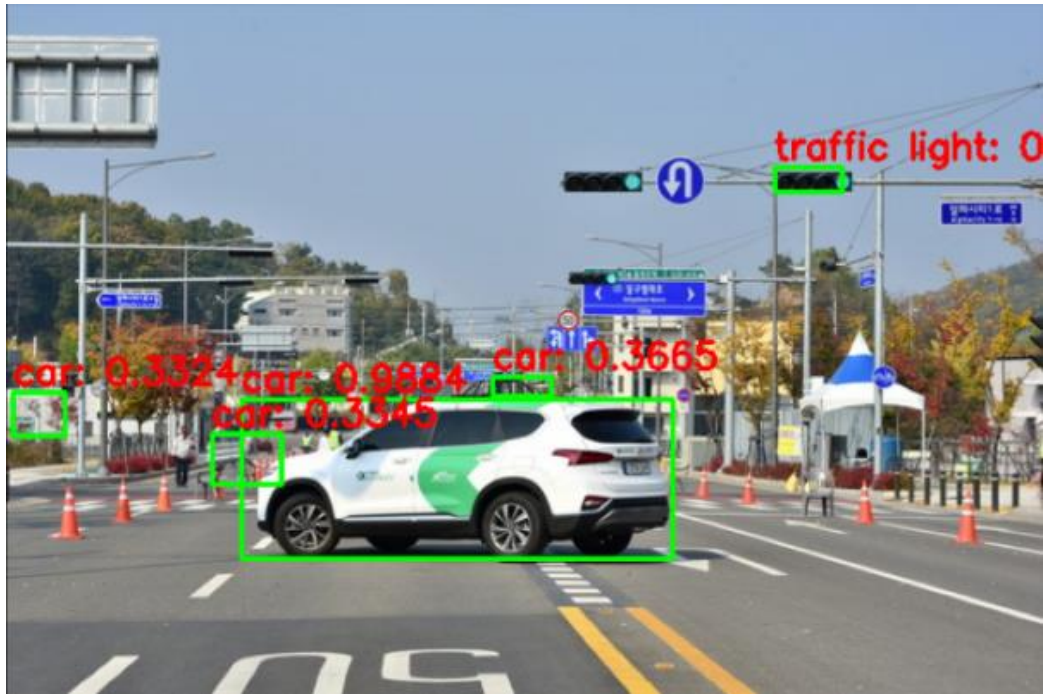
- YOLO는 빠르게 이미지에서 객체를 탐지하는 모델, Joseph Redmon이 개발
- 기존 모델들 보다 더 높은 정확도를 추구하는 것이 아닌, 근접한 정확도를 가지면서 더 많은 양의 이미지를 처리할 수 있는 실시간 객체 탐지를 하고자 등장
- OLOv1, v2, v3까지 개발하고 잠정 중단했지만(2020.02.21), 현재 다른 개발자에 의해 v4, v5까지 나온 상태

DarkNet https://pjreddie.com/darknet/

- Joseph Redmon이 독자적으로 개발한 신경망 프레임워크(Neural Network Framework)
- DNV(Deep Neural Network)들을 학습시키고 실행시킬 수 있는 프레임워크
- C, CUDA로 작성된 오픈 소스, 연산이 빠르고 설치가 쉽고 CPU 및 GPU 연산을 지원

객체탐지 실습 : SSD

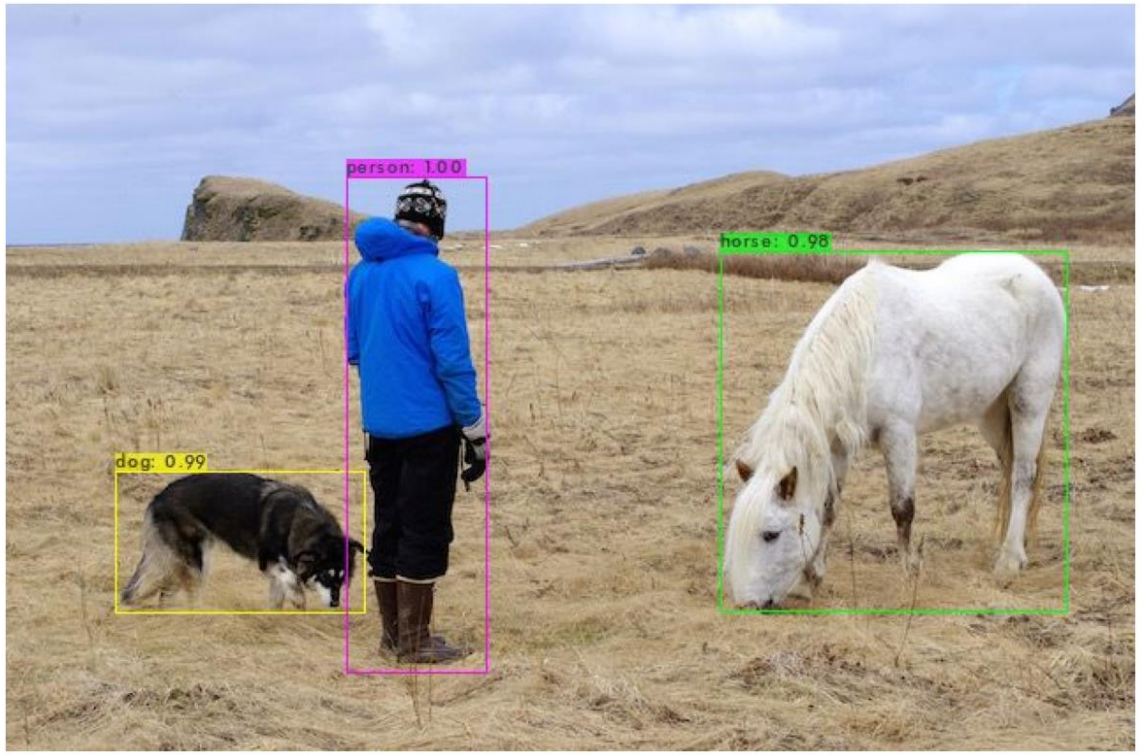
object_detection_sdd.ipynb



객체탐지 실습 : YOLO – DarkNet 설치/사용

object_detection_yolo.ipynb

```
git clone https://github.com/AlexeyAB/darknet
cd darknet/
make
wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
darknet detect cfg/yolov4.cfg yolov4.weights data/person.jpg
```



객체탐지 실습 : YOLO – 임계값(threshold) 파라미터

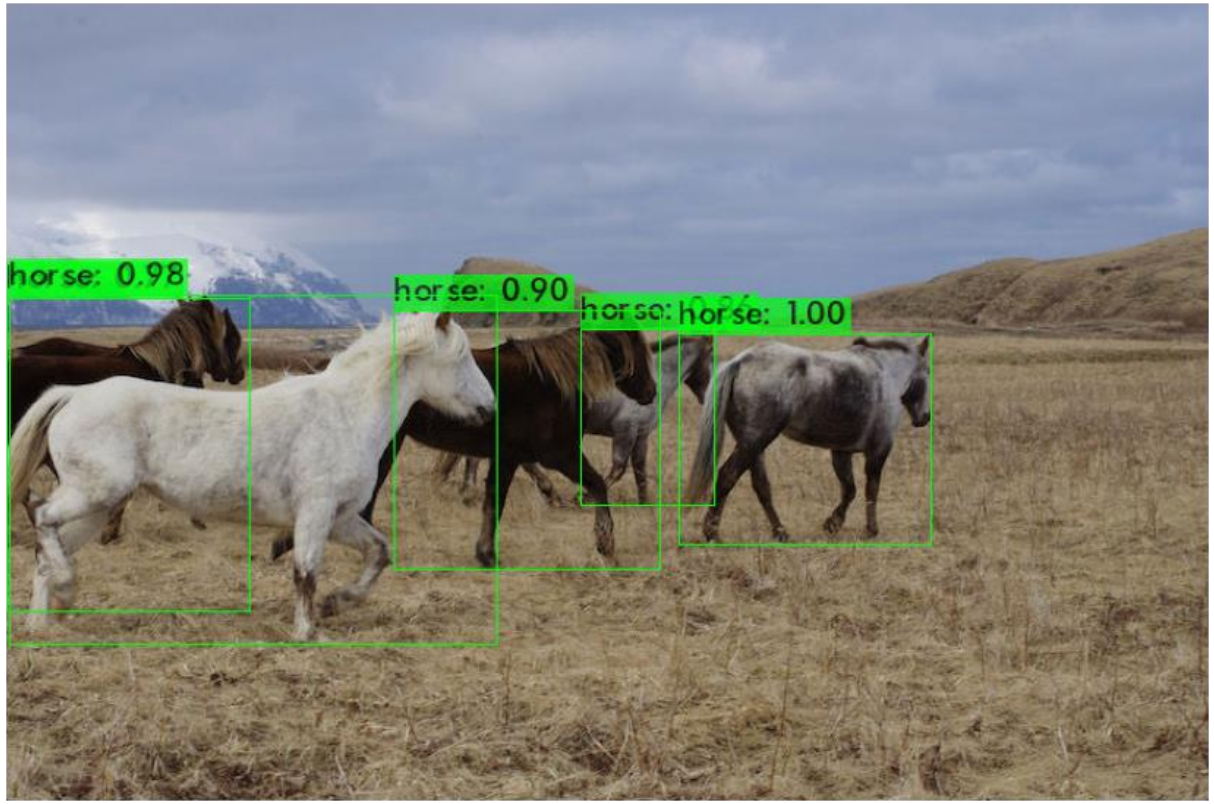
object_detection_yolo.ipynb

```
darknet detect cfg/yolov4.cfg yolov4.weights data/horses.jpg -thresh 0.9
darknet detect cfg/yolov4.cfg yolov4.weights data/horses.jpg -thresh 0.1
```

객체탐지 실습 : YOLO – ext_output 파라미터

object_detection_yolo.ipynb

```
darknet detect cfg/yolov4.cfg yolov4.weights data/horses.jpg -ext_output
```

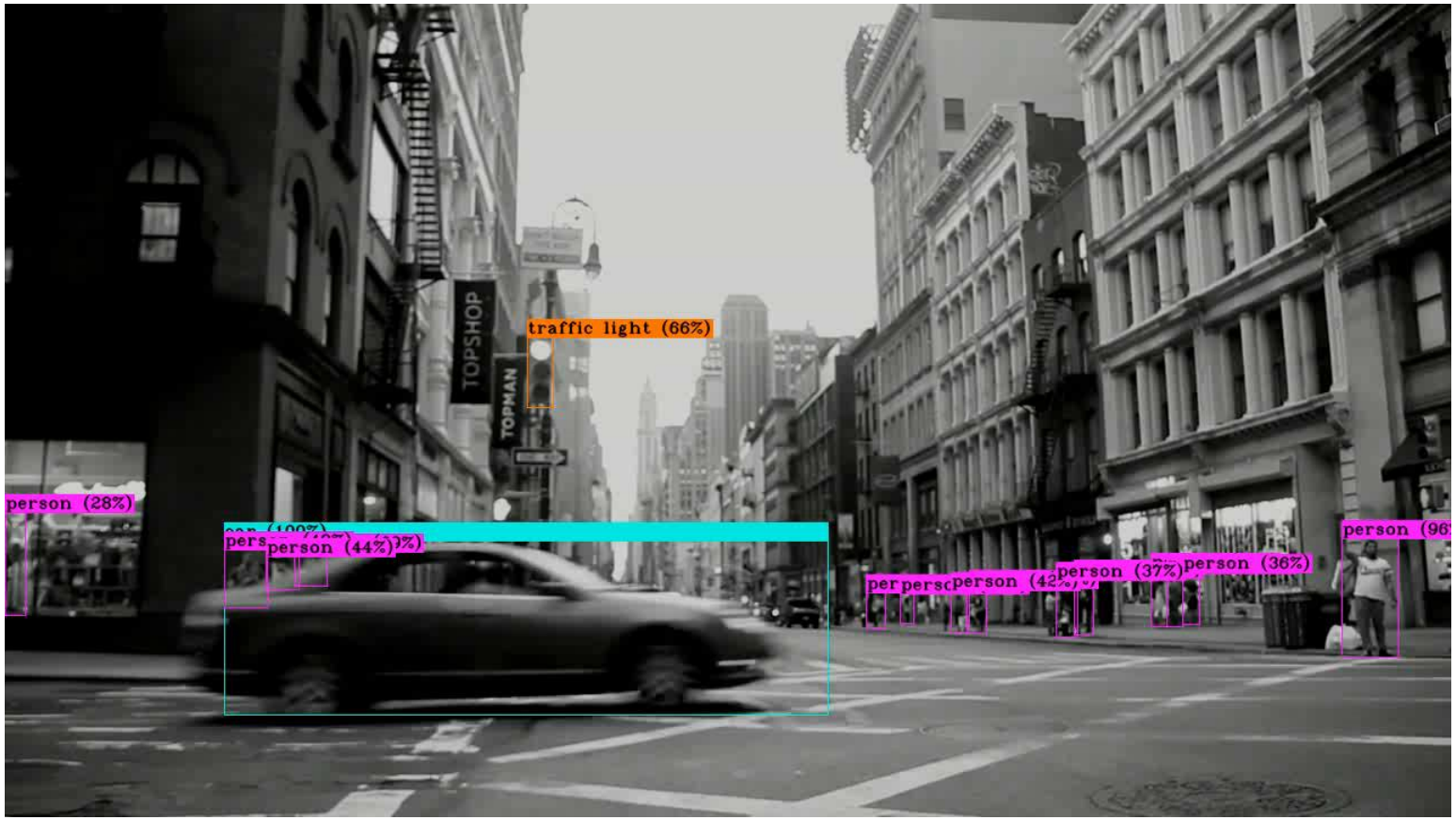



```
horse: 77% (left_x: -1 top_y: 189 width: 157 height: 202)
horse: 98% (left_x: 3 top_y: 188 width: 312 height: 224)
horse: 90% (left_x: 249 top_y: 198 width: 171 height: 166)
horse: 86% (left_x: 369 top_y: 210 width: 87 height: 113)
horse: 100% (left_x: 432 top_y: 213 width: 163 height: 135)
```

객체탐지 실습 : YOLO – 동영상에서 객체 탐지

object_detection_yolo.ipynb

```
darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights
/content/drive/MyDrive/visionai/street.mp4 -i 0 -out_filename -dont_show
/content/drive/MyDrive/visionai/street_result.mp4
```



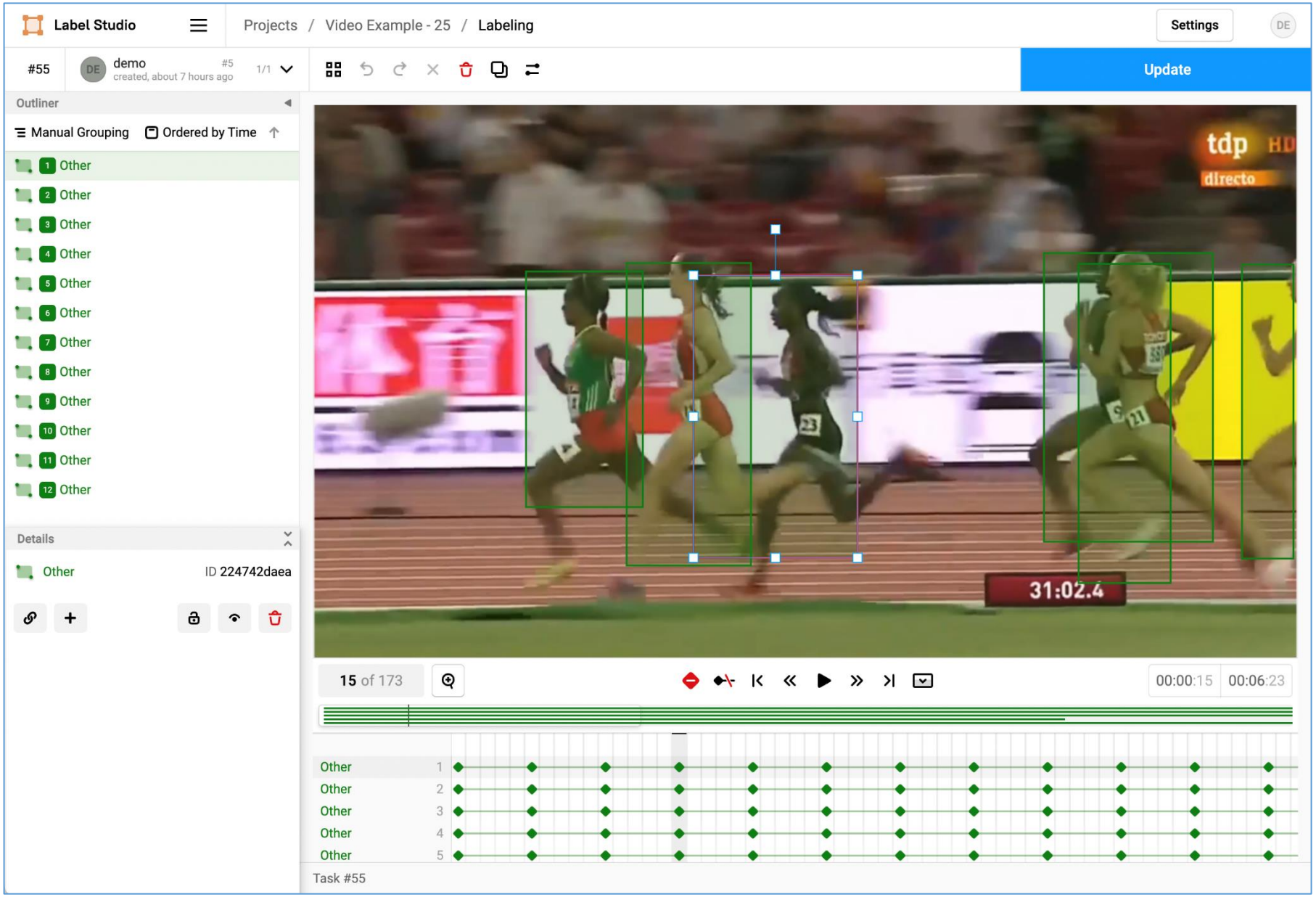
Detectron2

https://github.com/facebookresearch/detectron2

OR YOLOv5

이미지 Labeling

https://github.com/tzutalin/labelImg



THANK YOU

kgpark88@gmail.com