

object detection with yolo



해커톤 목표

1. 머신러닝을 사용할것
2. 최대한 쉽고 간편한것을 사용할것
3. 실생활에 쉽게 적용가능해야 할것
4. **TODO** 다양한 실생활에 적용해 보자

Object Detection 아는 척 하기 시작

Object Detection란?

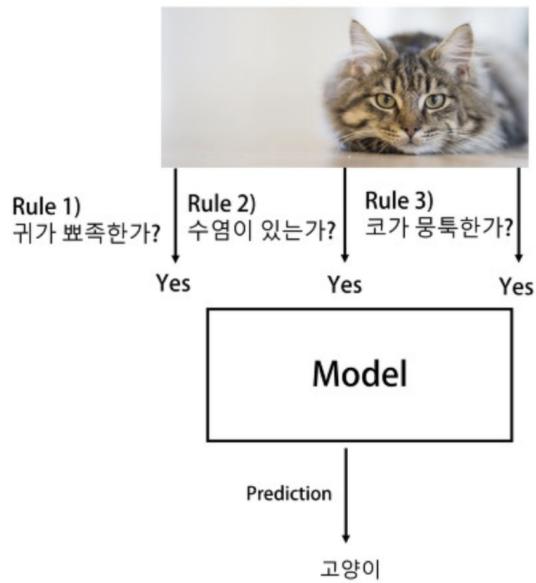
Object Detection == 물체 검출

즉, 카메라나 다른 센서를 이용하여 자동차, 사람, 동물, 물건 등을 검출.

추가로 이게 어떤 것을 검출 했는지 나타는 Classification(분류) 이란 단어를 사용

object detection Overview

Rule-based approach



Data-driven approach

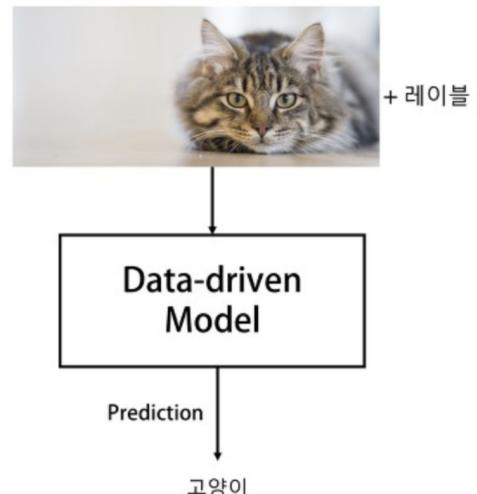
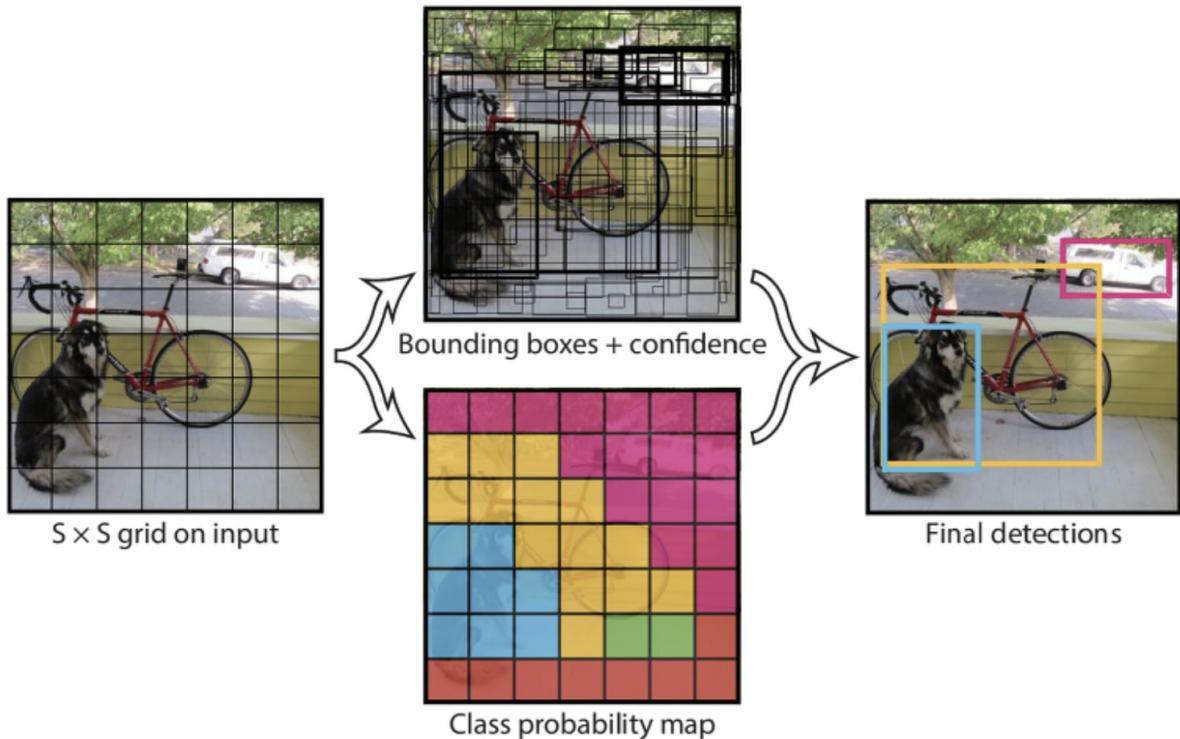


그림. Rule-based approach vs. Data-driven approach

Object Detection엔 이런 알고리즘이 쓰였어요

- Nearest neighbor (NN): 학습 데이터셋을 저장한 후에, 예측 단계에서는 투입된 이미지와 가장 가까운 데이터의 레이블을 통해 예측 하는 방법
- K-nearest neighbor (KNN) : NN은 하나의 레이어만 보고 판단한다. 많은 레이어를 보고 판단방법
- Convolutional Neural network(CNN) : 선택적알고리즘을 통해 약 2000개로 쪼개서 추출
→ r-cnn → fast r-cnn → faster r-cnn 등으로 진화
- You Only Look Once(yolo) : 이미지를 여러번 필터하지 않고 한번 보고 끝낸다!

YOLO



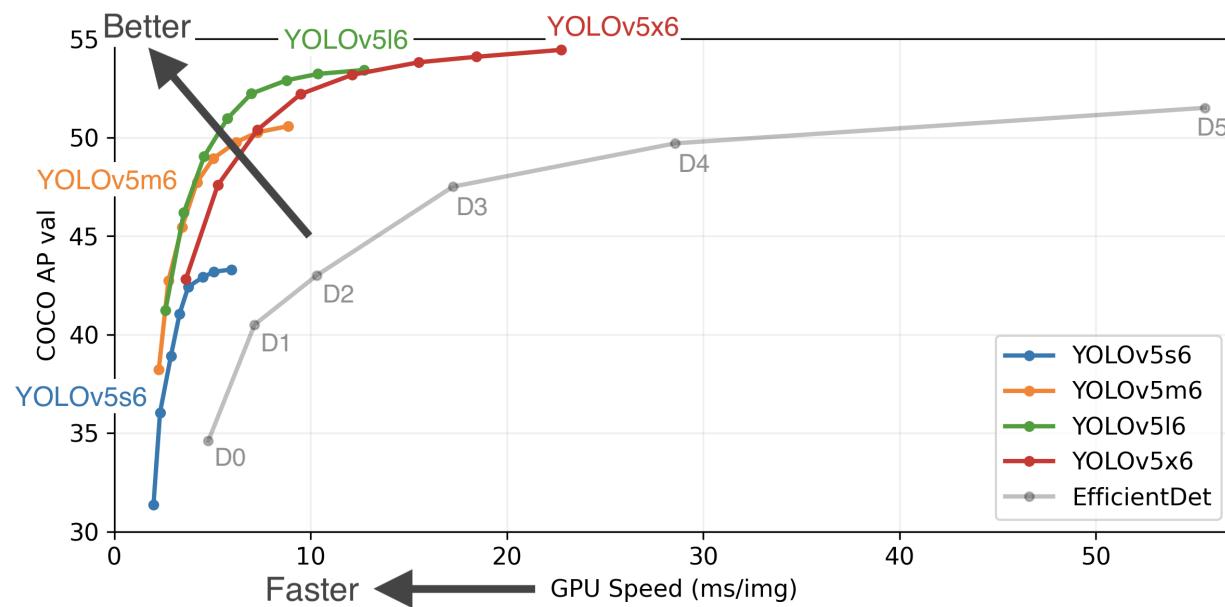
yolo 동작 방식

1. 이미지의 특정 간격으로 그리드를 나눈다
2. 나눈 그리드마다 여러 크기의 박스들을 만들어줍니다.
3. 해당 박스 안에 우리가 했던 물체가 있는지 검사합니다.
4. 나눈 그리드를 하나로 합치면서 검출된 물체도 하나로 합쳐줍니다. (NMS / ION)

하지만!!!

yolo는 빠르지만 정확도는 높지 않아요.(특히 작은 객체들이 몰려 있는 경우 검출을 잘 못해요)
즉, 속도를 취하고 정확도를 포기했어요!!

그중에서도 yolov5 빠르다!!



yolov5?? 그 전에도 있었나 보네?

YOLOv3

- 2018년 4월 출시했다. 백본 아키텍쳐 Darknet 53을 기반으로 만들어져 있으며, YOLO를 만든 Josept Redmon이 발표했다.

- 하지만 Josept Redmon은 YOLOv3을 마지막으로 더이상 개발하지 않는다고 밝혔다. (군사적으로 쓰이는것을 반대했다고 한다)

YOLOv4

- 2020년 4월 출시했다. v3에 비해 AP, FPS가 각각 10%, 12% 증가했다.
- CSPNet 기반의 backbone(CSPDarkNet53)을 설계하여 사용했다.

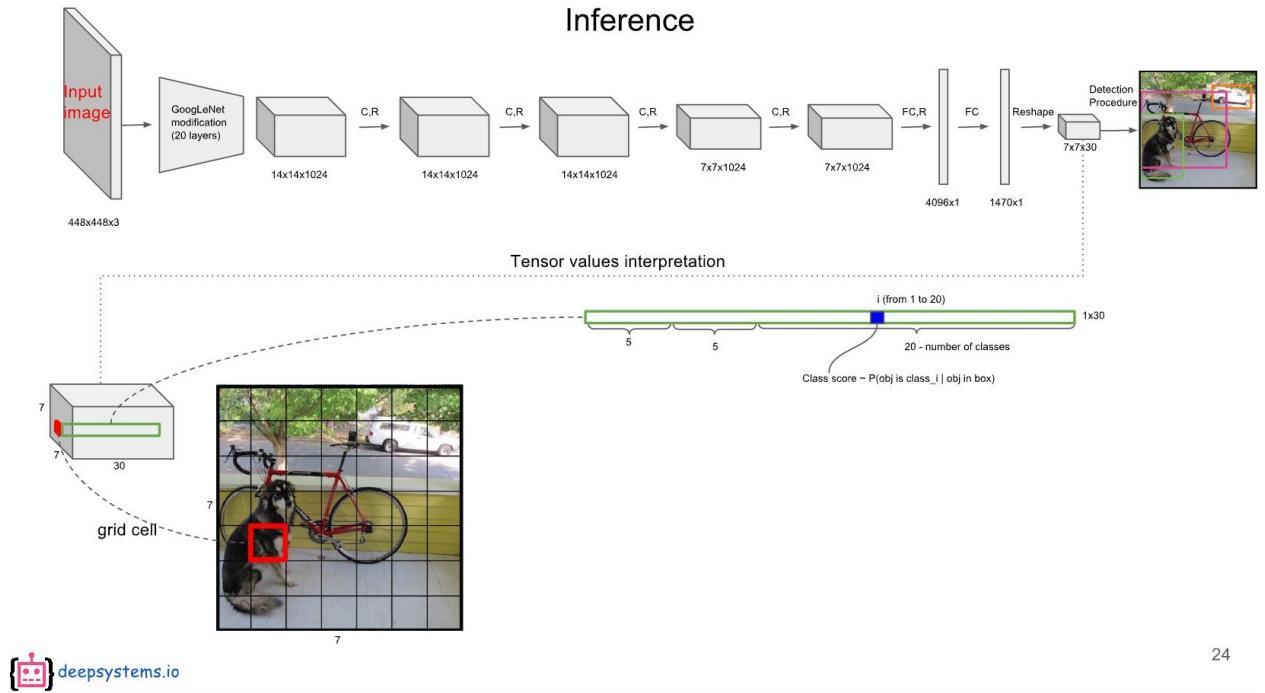
YOLOv5

- 2020년 6월 출시했다. v4에 비해 낮은 용량과 빠른 속도를 가지고 있다. (성능은 비슷하다)
- YOLOv4와 같은 CSPNet 기반의 backbone을 설계하여 사용했다.
- 처음 출시될 때 논문이 함께 출시되지 않았고, 이름을 YOLOv5로 하는것에 대한 논란이 있다.

PP-YOLO

- 가장 최근인 2020년 7월 출시됐다. YOLOv4보다 정확도가 속도가 더 높다.
- v3모델을 기반으로 하지만, Darknet3백본을 ResNet백본으로 교체했고 오픈소스 깊이 학습 플랫폼인 PaddlePaddle에 바탕을 두고 있다.

네트워크는 이렇게 구성되어 있어요(yolov1)



네... 이런 loss 함수를 쓴다고 하네요..(yolov1)

$$\begin{aligned}
& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)
\end{aligned}$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the j th bounding box predictor in cell i is “responsible” for that prediction.

위 수식은 YOLOv1의 로스 함수입니다. 😷 무시무시하게 생겼지만 사실 자세히 뜯어보면 그렇게까지 험악한 친구는 아닙니다 😊 기본적으로 좌표, 크기, 컨피던스, 클래스 확률의 평균제곱오차를 계산한 후 모두 더하는 구조로 되어있습니다.

“responsible”이라는 개념은 그리드 셀에 대해서도, 바운딩 박스에 대해서도 쓰입니다. 하지만 로스의 관점에서 “responsible”은 결국 바운딩 박스에 대한 개념으로 생각하면 됩니다. 즉 하나의 물체에 대한 예측을 책임지는 유일한 바운딩 박스를 결정하는 문제입니다. 수식을 보면 예측에 대해 “responsible”하지 않은 박스의 좌표 크기 로스 텀은 0이 되는 것을 알 수 있습니다. 즉 책임이 없는 박스들에 대해서는 좌표와 크기의 로스를 면제해주자는 아이디어입니다.

논문 발표 시간 아닙니다.

죄송합니다. 있어보이고 싶었어요.

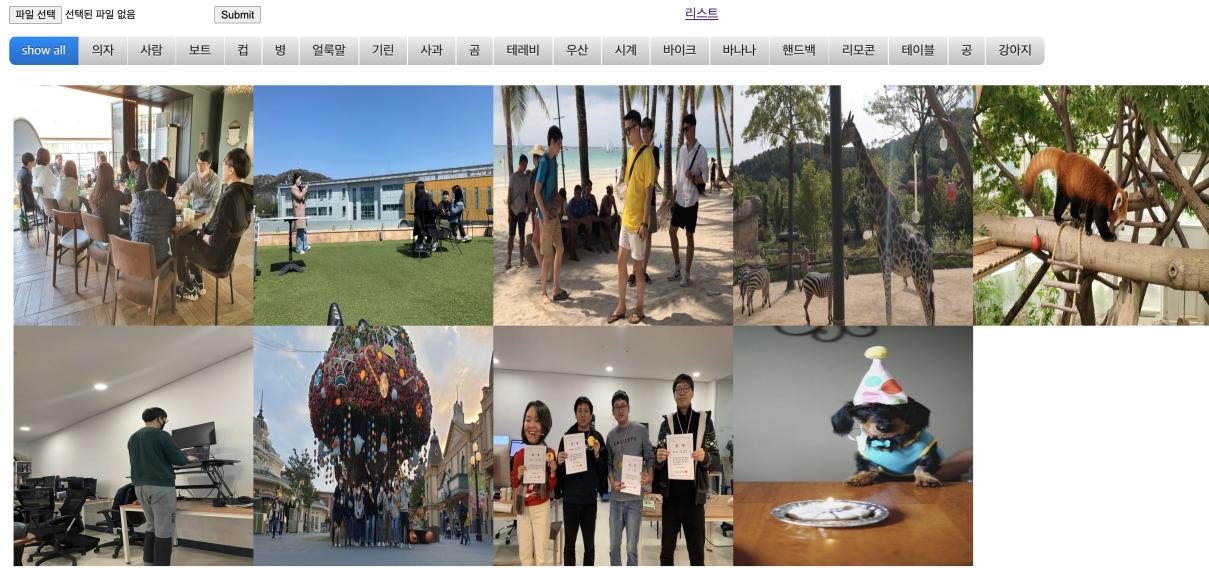
먼저 yolo로 이런걸 간단하게 만들 수 있어요

일단 한번 보시죠.

그래서 무엇을 만들었나요?

object detection로 포토 앨범 만들기

메인화면



상세 화면 (확인된 객체 boxing)

- 사람
- 시계
- 바이크
- 우산

aaad61bfb699c1b524b2900bd4626976

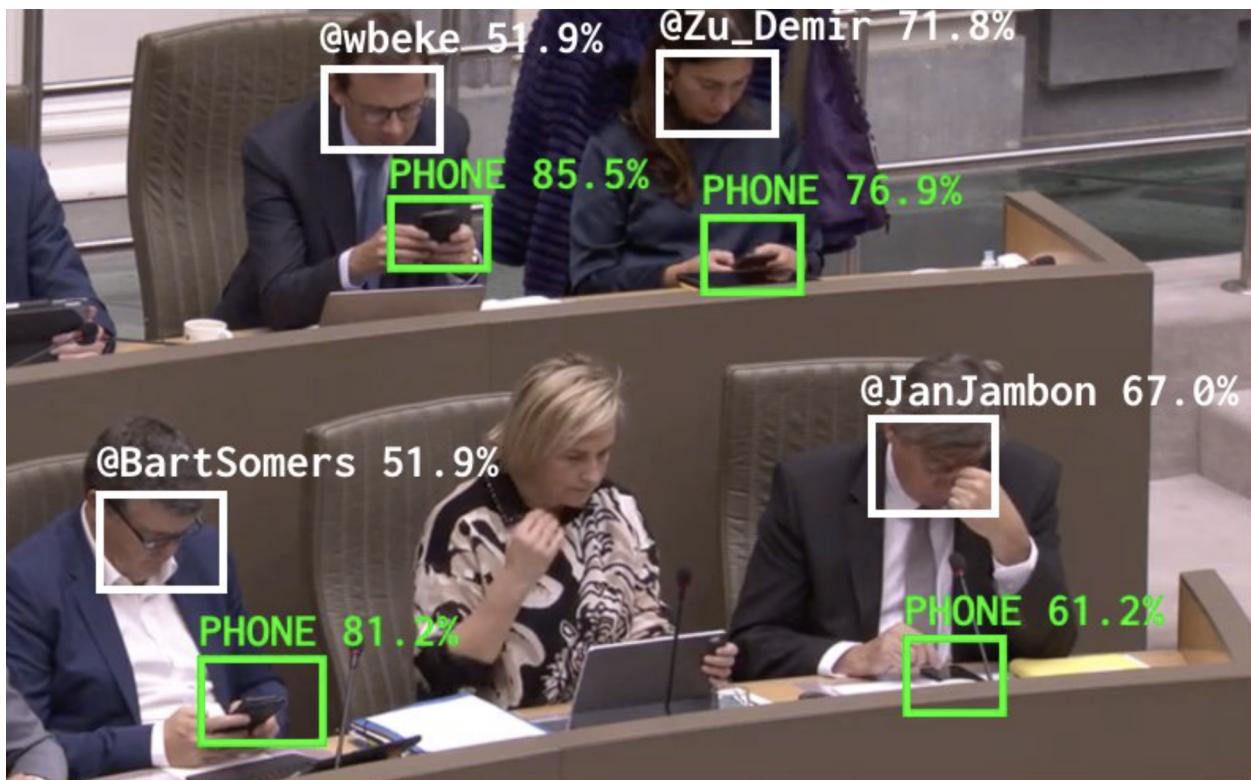


한계점

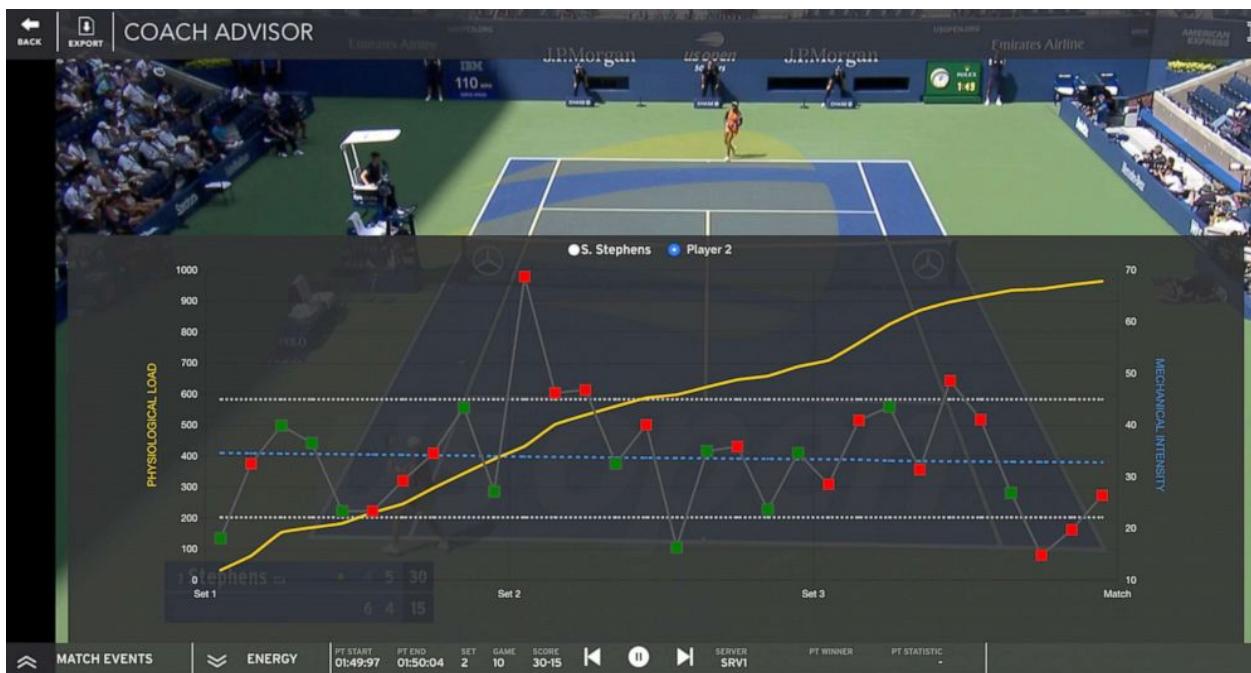
- 퍼블릭 데이터 셋은 기존 데이터셋보다 정확도가 떨어짐 (60000 : 5000)
- 학습시간 32코어 CPU로 6시간!!
- 레이블링 노가다

다음 스텝으로 이런게 가능 합니다.

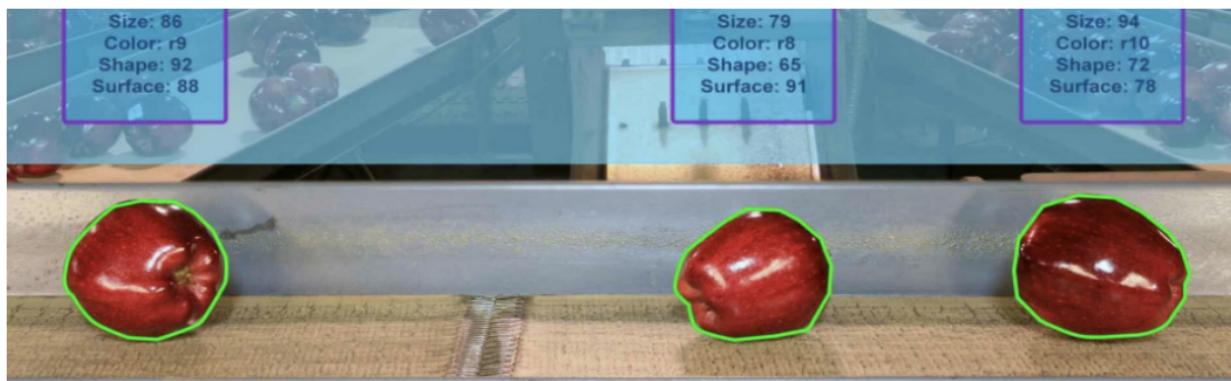
- 벨기에 의원 감시 프로젝트



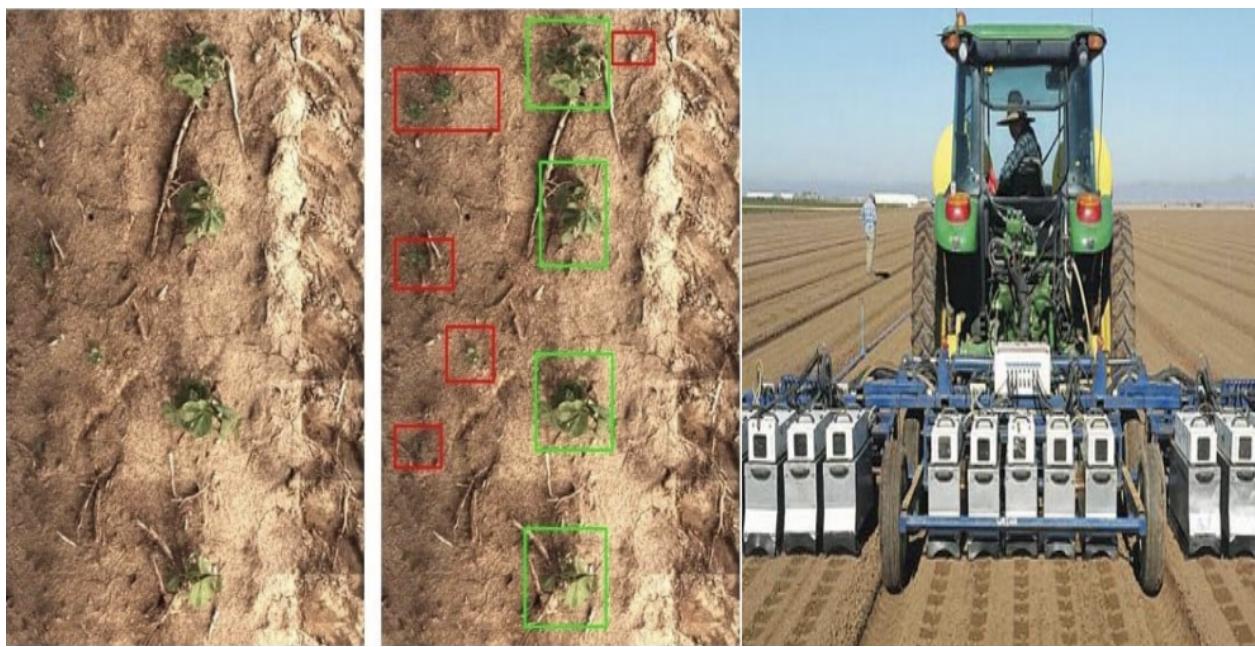
IBM + 테니스 코치 프로그램



- 과일 등급 판정



제초제 살포



질문은 아래의 참고사항을 봄주시면 되요!!

참고

<https://medium.com/curg/you-only-look-once-다-단지-한-번만-보았을-뿐이라구-bddc8e6238e2>

<https://deepflowest.tistory.com/181>

<https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/>

<https://mickael-k.tistory.com/24?category=798521>

<https://mickael-k.tistory.com/27>

<https://nuguziii.github.io/survey/S-001/>

<https://blog.naver.com/sogangori/220993971883>

https://www.youtube.com/watch?v=c18ILApJ1OU&ab_channel=SKplanetTacademy

https://www.youtube.com/watch?v=mKAEGSxwOAY&ab_channel=TheAIGuy

https://www.youtube.com/watch?v=NsxDrEJTgRw&ab_channel=VenelinValkov

<https://towardsdatascience.com/yolo-v4-or-yolo-v5-or-pp-yolo-dad8e40f7109>

<https://www.youtube.com/watch?v=NM6lrxxy0bxs>

yolov4 tensorflow

<https://keyog.tistory.com/21>

<https://keyog.tistory.com/22>

https://www.youtube.com/watch?v=hxwEqXCgQO4&ab_channel=빵형의개발도상국

모델 훈련

https://www.youtube.com/watch?v=hTCmL3S4Obw&ab_channel=JayBhatt