Google Developer Student Clubs
Hanyang

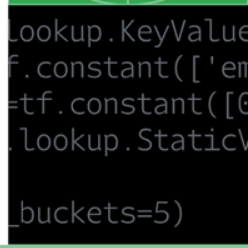
ML/DL 스터디

basic

김남호, 이지환, 김찬원
GDSC Hanyang

Google Developer Student Clubs
Hanyang University

9주차 Kaggle - Yolo를 활용한 선수 tracking



목차

YOLO 이론 설명

You Only Look Once (YOLO)

1. CNN을 활용한 객체 식별 및 분류 모델

2. 탐지 속도, 정확도 모두 높음

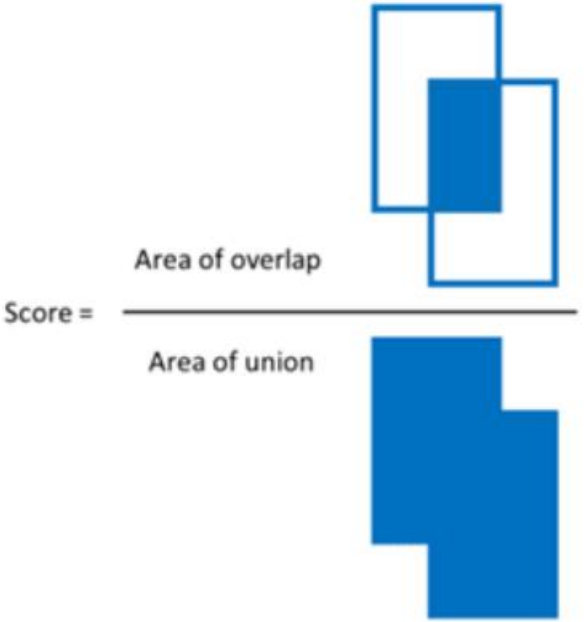
Input: 이미지
Output: 이미지내의 객체들의 bounding box, (분류)를 예측

사용 기술: -CNN(like googlenet)
-ReLU
-Batch 정규화
-Dropout =>overfitting방지
-IOU

https://www.datacamp.com/blog/yolo-object-detection-explained

IOU

Intersection over union



예측된 바운딩 박스(혹은 세그멘테이션 결과)와 실제 객체의 영역 사이의 겹치는 정도를 측정하는 척도

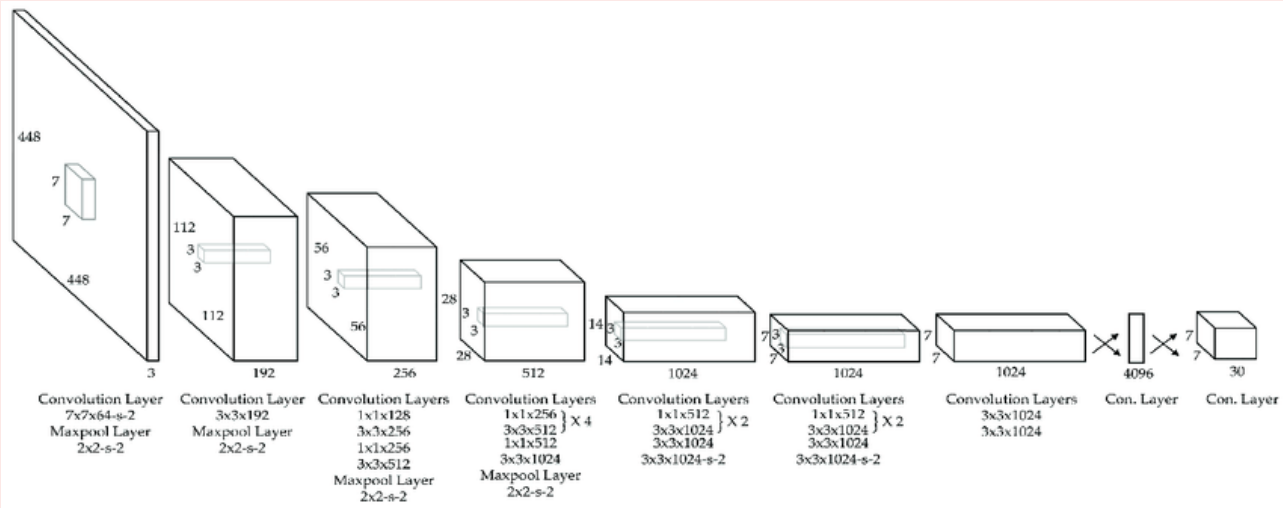
YOLO Architecture

24개의 convolution layer

4개의 max pooling layer

2개의 FC layer

Activation func: ReLU (최종layer는 다름)



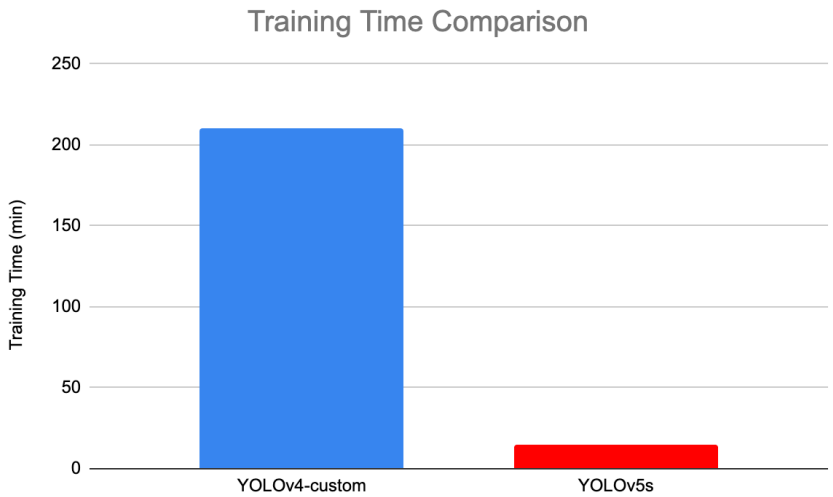
YOLO Architecture

Final loss func: Binary cross entropy(YOLOv2)

YOLOv5가 현재 사용됨(2020.06출시)

버전향상에 따른 범용성도 증가

주요 변화: 초반layer를 묶어 -> Focus layer



YOLOv5 🚀 **Focus 레이어**에 대한 많은 관심을 받았기 때문에 여기에 짧은 문서를 작성했습니다. YOLOv3 아키텍처를 YOLOv5로 발전시킬 때 Focus 레이어를 직접 만들었으며 다른 소스에서 채택하지 않았습니다. 포커스 레이어의 **주요 목적은** 레이어를 줄이고, 매개변수를 줄이고, FLOPS를 줄이고, **CUDA 메모리를 줄이고, mAP에 미치는 영향을 최소화하면서 전진 및 후진 속도를 높이는** 것입니다.

https://github.com/ultralytics/yolov5/discussions/3181m1

모델 활용 아이디어

모델 개선

개선 불가

<이유>
- YOLOv5 연구, 논문자료가 거의 없음.

- YOLO내부 구조와 그 기술(비전AI)을 모두 공부하는데 있어서 시간적 제약.

=> 베이스코드를 활용해서 의미 있는 작업을 시도하자.

원래 구상

베이스코드(YOLOv5, Bytetrack)를 활용

=> 골 넣는 선수의 자취를 표현하고자 함.

=> 골 넣는 선수의 자취(오프더볼)를 표현하고자 함.

=> 각 프레임별로 tracker_id활용해서 위치를 표현하는 것 까지 진행.

한계

구도의 움직임에 따른 점 위치 수정 불가
(단순히 이미지 위에 점을 찍으면, 구도에 따라 점을 이동시킬 수가 없음)

➔ SOL: 고정물체를 활용, 카메라 속도이용해서 점위치를 뺄셈한다.

➔ 다른 트래커를 사용한다.(Norfair)


CHE 0
03:07
0 MNC
deliveroo
PayPal

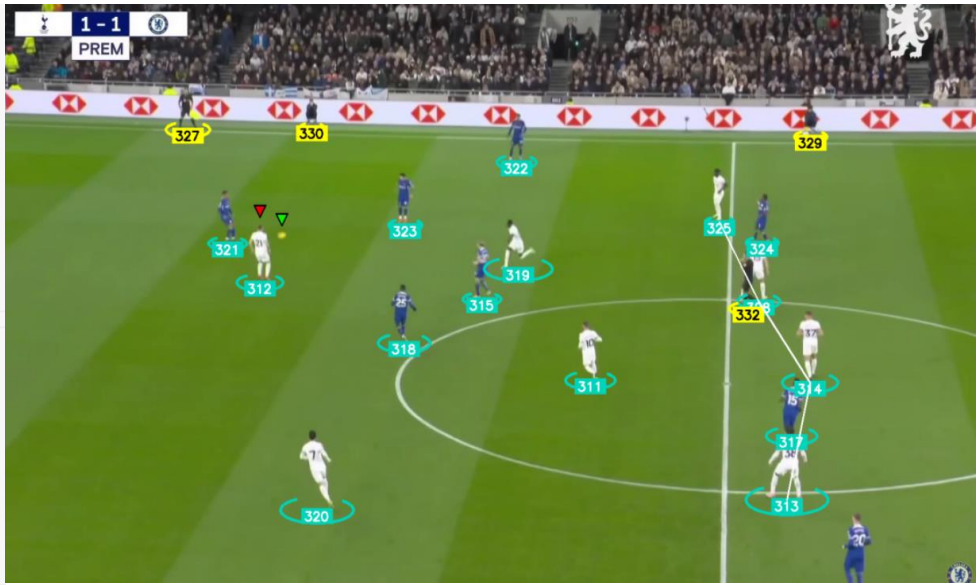
해결방안

*시간, 노력 등의 제약을 고려하여 기존 방안은 중단.

=> 수비수 4명이 수비시에 서있는 라인 상황을 선을 이용해서 표현하는 것을 활용 마무리.

베이스 코드 수정/활용

```python
import time
class LineAnnotator:
    def __init__(self, line_color: Color, line_thickness: int):
        self.line_color = line_color
        self.line_thickness = line_thickness
        self.tracker_ids = list()

    def annotate(self, image: np.ndarray, detections: List[Detection]) -> np.ndarray:
        annotated_image = image.copy()
        if len(self.tracker_ids) == 0:
            plot_image(annotated_image, 16)
            time.sleep(2)
            cnt = input("How many players do you want detection(if you want pass, enter 'pass'): ")
            if cnt == 'pass':
                return annotated_image
            for _ in range(int(cnt)):
                self.tracker_ids.append(int(input("input tracker_id: ")))

        detections_target = list()
        for id in self.tracker_ids:
            for detection in detections:
                if detection.tracker_id == id:
                    detections_target.append(detection)

        xy_list = list()
        for detection_t in detections_target:
            xy_list.append(detection_t.rect.bottom_center.int_xy_tuple)

        for i in range(len(xy_list)-1):
            cv2.line(annotated_image, xy_list[i], xy_list[i+1], color=self.line_color, thickness = self.line_thickness)

        return annotated_image
```



활용 방안

1. 첫 프레임에서 수비수 tracker_id확인
1-1. 확인불가할 경우(겹쳐서 확인불가) pass -> 다음 프레임으로 넘겨서 확인.

2. 확인된 수비수들에 대해 리스트화 -> x,y좌표 리스트 생성 및 저장

3. 각 프레임마다 해당 수비수들에 대한 좌표에 대해 Line_annotator 클래스 활용해서 annotate. (직접 생성 클래스)

4. 전체 프레임에 대해 반복(각 bounding box 변화의 예측 with 바이트트랙)

개선해야 할 점

Feedback

1. 경기장 전체에 대한 영상, YOLO에 대한 사전 지식 있었다면..

2. 사전학습된 모델을 다음번에는 Fine tuning 시도.

3. 이미지에 대해 다루는 것이 미숙함.
   - 수학적으로 카메라 이동에 따른 프레임단위 변화에 대해 기민하게 대처하지 못함.

4. CNN이 아닌 ViT도 객체 감지 및 이미지 분류에 사용해보도록 공부하자.

발표를 들어 주셔서 감사합니다.

김남호, 이지환, 김찬원
Google Developer Student Clubs