Google Developer Student Clubs
Hanyang

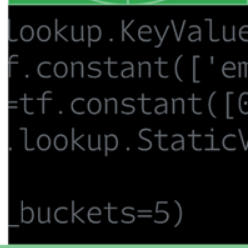
ML/DL 스터디

basic

김남호, 이지환, 김찬원
GDSC Hanyang

Google Developer Student Clubs
Hanyang University

7주차 Kaggle - Yolo를 활용한 선수 tracking



목차

베이스 코드(모델) 분석

<The whole code flow>

1. Data 영상을 프레임화
2. 기학습된 YOLOv5모델로 우리가 정한 선수와 공이라는 객체를 탐지
3. 선수, 심판, 공 등에 대한 탐지 표시를 위한 annotator 생성
4. Bytetracker를 통한 연속된 프레임에서의 객체 탐지 트래킹
5. 공 소유자의 표시를 위한 기준 및 함수 생성

최종 트래킹 반영된 영상 생성 및 출력!

<데이터 및 모델, 라이브러리 초기 설정 및 다운>

1. Training data 다운
2. YOLOv5 모델 다운
3. 커스텀 모델 실행(player 감지)
   -by roboflow

Google Developer Student Clubs

```
!kaggle competitions files -c dfl-bundesliga-data-shootout | grep clips | head -10
```

```
clips/08fd33_0.mp4      19MB  2022-07-29 14:23:09
clips/0a2d9b_2.mp4      20MB  2022-07-29 14:23:09
clips/0a2d9b_6.mp4      20MB  2022-07-29 14:23:09
clips/0a2d9b_4.mp4      18MB  2022-07-29 14:23:09
clips/08fd33_9.mp4      18MB  2022-07-29 14:23:09
clips/0a2d9b_8.mp4      19MB  2022-07-29 14:23:09
clips/0a2d9b_9.mp4      18MB  2022-07-29 14:23:09
clips/08fd33_6.mp4      19MB  2022-07-29 14:23:09
clips/08fd33_1.mp4      18MB  2022-07-29 14:23:09
clips/08fd33_8.mp4      20MB  2022-07-29 14:23:09
```

```
%cd {HOME}
!kaggle competitions files -c dfl-bundesliga-data-shootout | ₩
grep clips | head -20 | ₩
awk '{print $1}' | ₩
while read -r line; ₩
  do kaggle competitions download -c dfl-bundesliga-data-shootout -f line – pclips – –q
  done
```

Install YOLOv5

```
%cd {HOME}
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -r requirements.txt

import torch
import utils
display = utils.notebook_init()
```

<영상 프레임화 및 모델 활용>

1. 비디오의 프레임 구성
2. 모델 활용 프레임단위 예측
3. 바이트트랙 다운 및 활용
4. Annotator 설정(선수, 공 표시용)

Google Developer Student Clubs

Use custom model - single frame

```
from typing import Generator

import matplotlib.pyplot as plt
import numpy as np

import cv2

%matplotlib inline


def generate_frames(video_file: str) -> Generator[np.ndarray, None, None]:
    video = cv2.VideoCapture(video_file)

    while video.isOpened():
        success, frame = video.read()

        if not success:
            break

        yield frame

    video.release()


def plot_image(image: np.ndarray, size: int = 12) -> None:
    plt.figure(figsize=(size, size))
    plt.imshow(image[...,::-1])
    plt.show()
```

```
SOURCE_VIDEO_PATH = f"{HOME}/clips/08fd33_4.mp4"
```

<중간 결과>

선수, 공, 심판, 골키퍼를 모두 구분
(프레임단위)

각 프레임 변화에 따른 이동된
객체는 Bytetracker가 추적해줄 것.

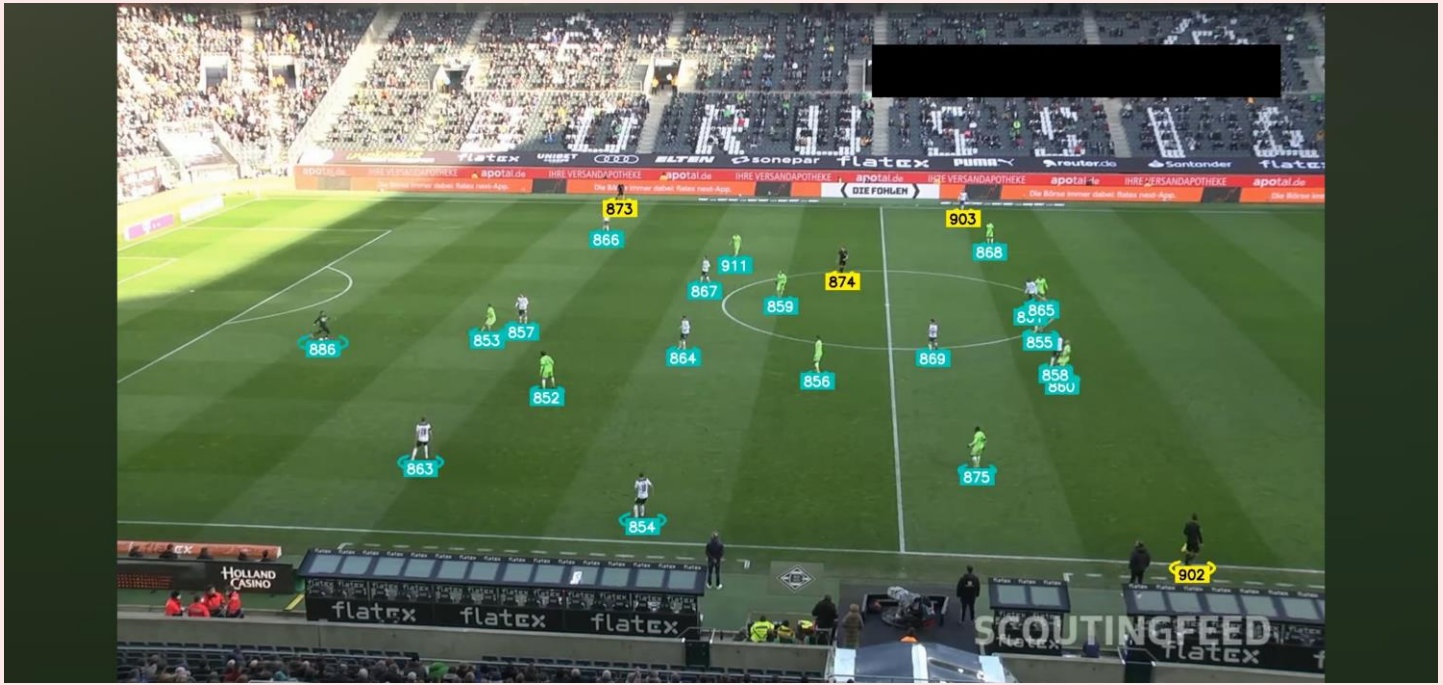
<공 소유자 표시>

1. 공 소유의 기준 반경 및 픽셀 설정 및 Marker함수 세팅
2. 단일 프레임에 대해 적용
3. FULL 비디오에 대해 모델,Annotator 등 적용 후 출력
4. FULL 비디오의 annotation에 대한 트래킹함수 설정

```python
from typing import List, Optional

#player가 공 소유하는지 판단해줌
# resolves which player is currently in ball possession based on player-ball proximity
def get_player_in_possession(
    player_detections: List[Detection],
    ball_detections: List[Detection],
    proximity: int
) -> Optional[Detection]:
    if len(ball_detections) != 1:
        return None
    ball_detection = ball_detections[0]
    for player_detection in player_detections:
        if player_detection.rect.pad(proximity).contains_point(point=ball_detection.rect.center):
            return player_detection
```

<최종 영상 모델 활용 및 트래킹>



<활용한 기술 및 라이브러리>

1. 기학습된 YOLOv5모델 활용
2. Bytetrack으로 프레임단위 트래킹에 활용
3. Cv2 로 비디오 생성 및 출력
4. Pandas... etc

모델 활용 아이디어

아이디어 선정

pre-trained model (Yolov5)이 포함된 Baseline code를 활용할 수 있는 방안에 대해 여러 아이디어를 제시

그 중 **실현 가능성이 높고 시도 가치(기대 효과)가 있는 아이디어**를 선정함

<아이디어>

득점자의 Off the ball 움직임 분석

- 실현 가능성 높음
- 해당 프로젝트를 통해 득점자들의 '**득점으로 연결 짓는 움직임**'에 대한 다양한 분석 결과를 도출할 수 있을 것이라 기대

"통계는 보통 선수들이 90분 간 평균적으로 약 3분 정도 공을 소유한다고 말한다. (중략) 그렇다면 여기서 얻어지는 가장 중요한 결론은 바로 이것이다. 공을 소유하지 않는 87분 간 무엇을 할 것인가? 바로 이것이야말로 훌륭한 선수를 결정짓는 기준이다." - 요한 크루이프

아이디어 구현 결과 예시 자료

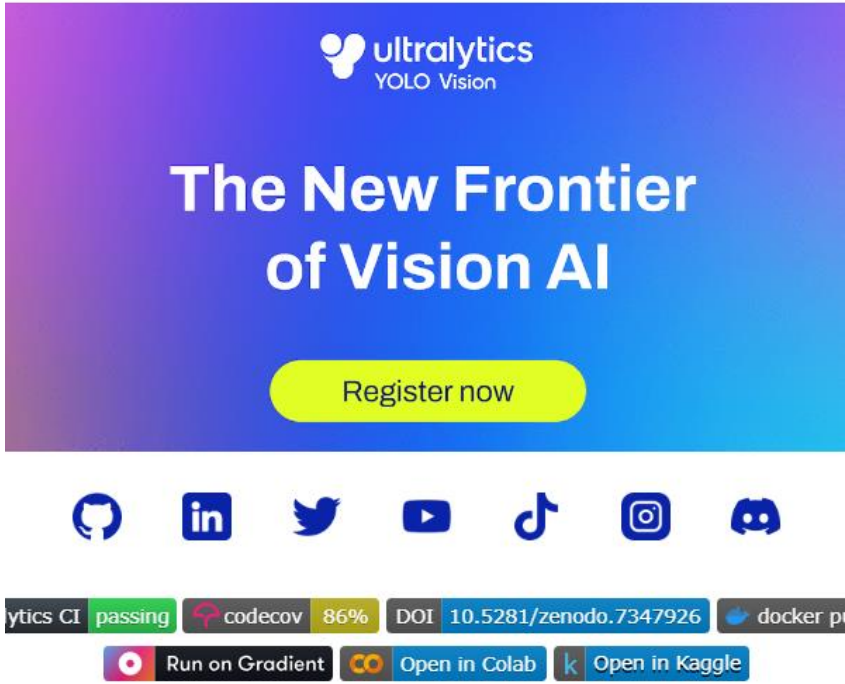
To-Do List (with 코드 리뷰)

1. 학습 시간과 파라미터를 고려, YOLOv5이 아닌 최신 버전 또는 다른 모델 사용

2. 선수 객체 인덱싱 이후 득점자 Labeling

3. 위치 정보를 통해 득점자 자취 표시
   - 공을 받기 전/받은 후로 구별

4. 선수 속력을 계산하여 표시



Issues

1. YOLO & ByteTrack Model 이해

2. YOLOv5를 다른 모델로 교체 시
수정되어야 하는 부분을 사전에 파악
   - YOLOv8 Model 우선 고려

3. 추론의 경우에도 GPU 사용이 필수적
   - Colab Pro의 컴퓨팅 단위 필요

4. 여러 사용자 정의 클래스 & 함수 코드 해석

Google Developer Student Clubs

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a

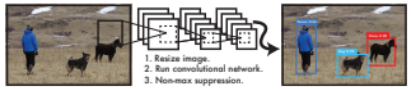


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This

자료 수집

데이터 선정 기준

선수들 간의 패스로 골을 넣는 영상 데이터를 수집

데이터 선정 기준

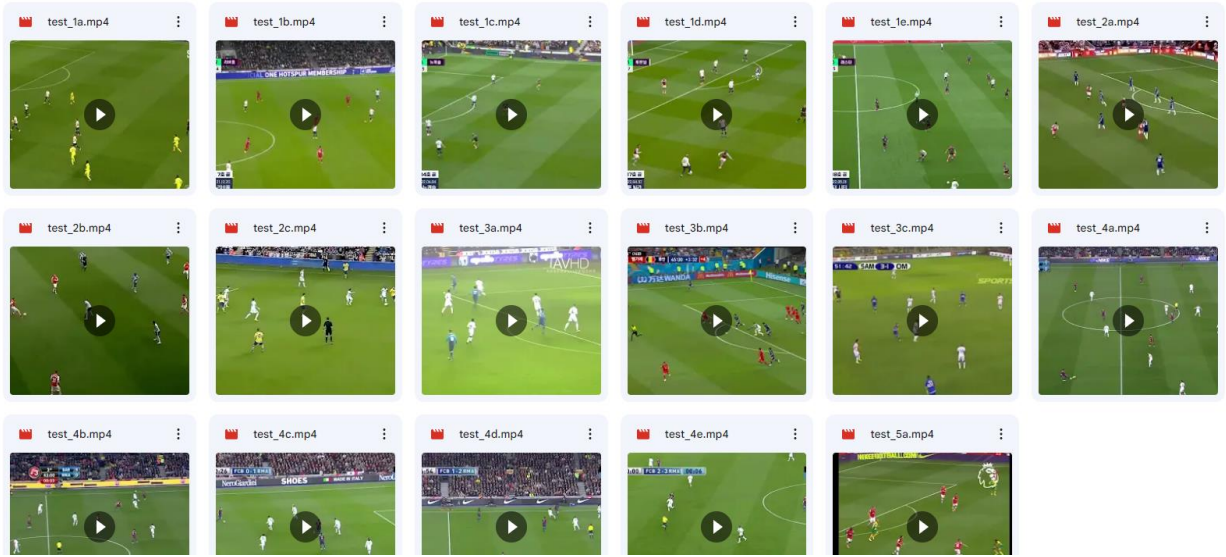
1. 전체적인 흐름에서 선수들과 공의 움직임이 보이는가
2. 갑작스러운 화면 전환이 되지 않는가
3. 공의 움직임이 명확히 보이는가

등을 고려하여 데이터를 선정

수집한 데이터

이를 통해 수집한 다음과 같이 데이터들을 통해

경기에서 골을 넣을 때의 선수들의 움직임을 추론해보려고 함



발표를 들어 주셔서 감사합니다.

김남호, 이지환, 김찬원
Google Developer Student Clubs

활용란

Google Developer Student Clubs