2019년 1학기 이재우 교수님 인공지능과 보안기술

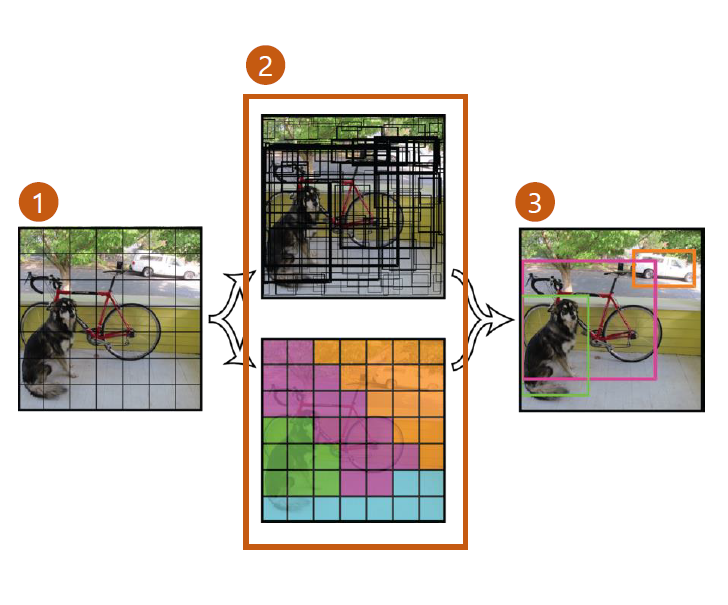
20144260 광고홍보학과 김정현

**< YOLO 프로젝트 기말 보고서 >**

1. 프로젝트 소개

**개요**

YOLO(You Only Look Once)는 CNN을 활용한 실시간 객체 검출 알고리즘이다. Fast R-CNN 등 기존의 객체 검출 알고리즘은 매우 정확한 결과값을 제공하지만 실시간 객체 검출이 필요한 상황에 사용할 만큼 빠른 속도를 제공하지 못했다. YOLO는 정확도의 수준을 낮추는 대신, 매우 빠른 속도로 결과값을 제공한다.

**작동방식**

[Figure ] Single-Stage Method

기존의 Fast R-CNN은 Two-Stage Method를 활용하였는데, Two-Stage Method란 feature map에서 오브젝트가 있을 것이라 예상되는 지점을 제안한 뒤(Region Proposal) Pooling 및 객체의 클래스 값을 계산하는 2단계 방식을 말한다. 반면에 YOLO는 Single-Stage Method를 활용하는데, Region Proposal을 수행하지 않고 한 단계에서 모든 작업을 동시에 수행한다. YOLO는 미리 프레임을 n\*n 그리드로 분할한 상태에서 CNN을 가동하는데, 각 그리드 셀을 중심으로 Bounding box를 생성해 오브젝트의 위치를 예측하고 가장 정확한 것으로 계산된 Bounding box 내 오브젝트의 class를 판단해 이를 결과값으로 제공한다.

**사용되는 알고리즘**

YOLO는 기본적으로 CNN으로 구성되는데, 오픈소스 CNN인 Dakrnet을 사용한다. 또한 이미지, 영상 및 실시간 카메라 인식을 위해 OpenCV를 사용한다. 코드 상에서는 Bounding Box가 지나치게 많이 생성되는 문제를 방지하고 검출의 정확도를 높이기 위해 NMS를 사용한다.

2. 프로젝트 코드 구성

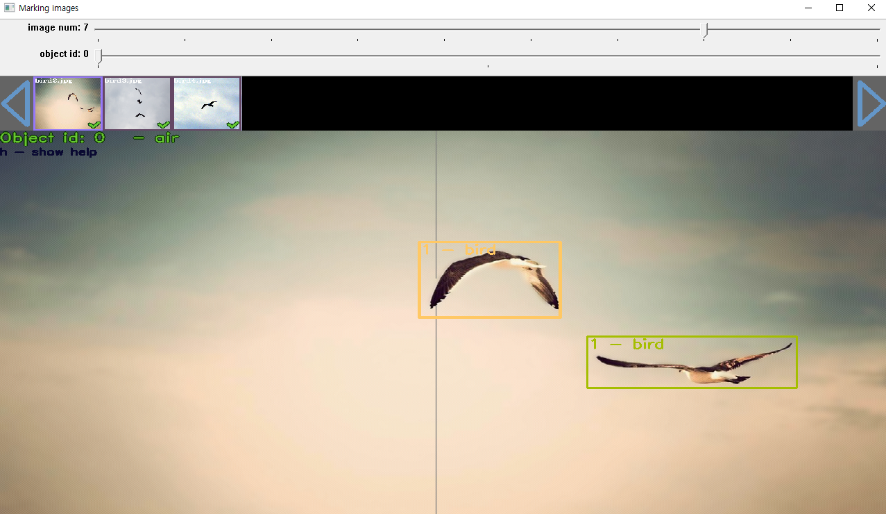
**기본 파일**

1. Configuration  
    알고리즘을 가동하는 데 있어 필요한 대부분의 정보들이 저장된 파일이다. GPU의 사양이나 학습에 소요되는 시간 및 데이터 양에 따라 batch, epoch 등의 값을 조정해야 하고, 검출하고자 하는 객체의 종류(class)에 따라 class와 filter값을 조정해 주어야 한다. YOLO에서 사용하는 filter의 값은 5 \* (class \* 5)로, class의 개수에 따라 그 값을 정확히 계산하여 configuration 파일을 수정해야 한다. (Filter 값의 수식이 위와 같은 이유는 class 당 필요한 attribute가 5개(x좌표, y좌표, 가로, 세로 길이, class값)이고, class 1개당 5개의 Bounding Box를 형성하기 때문이다. )
2. Data(Classes)  
    data 파일에는 검출하고자 하는 객체의 종류를 입력한다. class의 종류와 개수에 따라 Weights 파일과 긴밀한 연관관계를 갖는다.
3. Weights  
    Weights 파일에는 객체의 클래스를 검출하기 위해 수행되는 CNN layer의 가중치 정보가 들어 있다. 대부분의 class에 따른 가중치 정보는 웹 검색을 통해 구할 수 있으며, 학습을 통해 새로운 class에 대한 가중치 정보를 생성할 수 있다.

**코드 파일**

코드는 python과 pytorch로 작성된다. Pytorch의 torch.nn, torch.autograd 등의 기본적인 라이브러리와 opencv 라이브러리를 추가적으로 import한다.

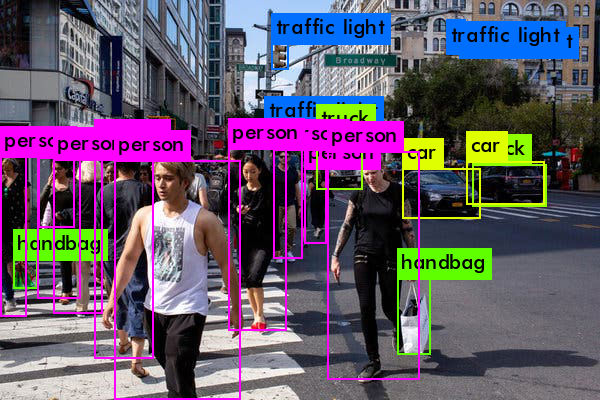
1. Darknet  
    전체 시스템의 architecture에 해당하는 코드들로 구성된다. Configuration 파일에서 필요 정보를 읽어오고, Input 이미지/동영상을 분석하는 module을 생성하고, 가중치 파일을 불러온 뒤 순전파(forward)를 진행하기 위한 전체적인 구조들이 코드화 되어 있다.
2. Util  
    Darknet을 보조하는 기능들로 이루어져 있다. 입력 이미지를 정해진 규격에 따라 분석하기 위해 해당 이미지의 transform을 resizing한 뒤 분석이 끝나면 다시 원래 이미지 사이즈로 되돌리는 함수, Bounding box를 최소화하기 위한 IoU, 그리고 결과 화면을 이해하기 쉽게 표시하는 함수 등이 포함되어 있다.
3. Detector  
    앞선 1), 2)를 기반으로 YOLO 시스템을 실행시키는 파일이다. Input 파일을 읽어오고 Darknet으로 보낸 뒤, 분석 결과에 따라 Output을 생성한다.

3. 학습 및 객체 검출 테스트

[Figure ] labeling 예시

YOLO는 사용자가 쉽게 학습을 수행할 수 있게 YOLO\_MARK 시스템을 제공한다. 시스템 상에서 제공되는 labeling 기능을 통해 dataset의 이미지에서 검출하고자 하는 클래스의 위치와 크기를 설정하여 저장할 수 있다. Labeling 데이터와 dataset 이미지, 가중치 파일과 configuration 파일을 사용해 학습을 진행하는데, batch\_size를 수정해 학습시간을 조정하고, epoch(max\_batch)를 수정해 가중치의 정확도를 높일 수 있다.

[Figure ] 테스트 결과



4. 개선점

**학습**

먼저 YOLO 알고리즘을 처음 활용하는 사람의 입장에서, 학습을 위한 데이터, 클래스의 개수, 세대 크기 그리고 학습에 소요되는 시간 등에 대한 안내가 전혀 없어 어려움을 겪을 수 있다. 대략 클래스 당 333개의 데이터가 필요하며 120세대 이후부터 객체 검출이 가능해지는데, 데이터와 세대에 따른 학습 시간 및 정확도를 예측해 안내해주면 사용이 매우 편리해질 것이다.

**코드**

YOLO는검출한 객체의 클래스에 대한 정확도(class confidence) 값을 계산할 때 sigmoid 함수를 사용하는데, CNN 알고리즘에서 높은 수행능력을 보이는 ReLU 함수를 적용하는 방향으로 코드를 개선할 수 있을 것이다.