RetinaNet을 이용한 음식 영역 추출하기

60150649 권 강 현
명지대학교
kgh9959@naver.com

Abstract— **지금까지 Object Detection 은 머신러닝에서 가장 뜨거운 주제 중 하나였다. 2013 년 R-CNN 논문발표를 시작으로 7 년 간 Fast R-CNN, Faster R-CNN, YOLO, Mask R-CNN, M2Det 등의 알고리즘이 발표되며 꾸준한 성능 향상이 이루어지고 있다. 이 프로젝트에서는 RetinaNet 의 구현체를 가지고 커스텀 데이터셋에 적용해 학습해보는 것을 목표로 한다.**
Keyword— *RetinaNet*

Ⅰ. INTRODUCTION

이 주제는 2019년 AI Starthon이라는 인공지능 경진 대회에서 가져온 것이다. 기존에 존재하는 Object Detection 알고리즘들을 통해 이미지 데이터에서 객체 탐지를 해보는 것이 목적이다.

Ⅱ. RELATED WORK

Object Detection에 대한 가장 전통적인 연구는 R-CNN(region based CNN)이다. 전통적인 R-CNN은 첫째 단계로 주어진 영상(이미지) 내에서 물체가 존재하는 위치를 Region of Interest(ROI)로 추청한 뒤, 둘째 단계로 그 ROI 각각에 존재하는 물체가 무엇인지 제각각 CNN으로 판별해내는 방식으로 작동했다.

하지만 2015년 Ross Girshick이 제안한 Fast R-CNN이라는 논문을 통해서 앞선 두 단계의 과정을 하나로 묶어내는 ROI pooling 방법을 제안하며 학습 속도의 큰 향상을 가져왔다. 그럼에도 불구하고 Fast R-CNN의 경우 GPU기반으로 작동하는 CNN 부분 연산은 빠르지만, 여전히 CPU기반으로 작동하게 되는 region proposal 추출 시간이 상대적으로 너무 오래 걸린다는 약점을 가지고 있었다.

그래서 2015년 같은 해에 그 문제를 개선한 Faster R-CNN이라는 논문이 발표되었다. 이 논문의 핵심은 region proposal 생성에 관련된 외부 알고리즘을 neural network로 만들어서 그걸 CNN 내부로 가져오는 것이었다. 이로써 Faster R-CNN은 R-CNN의 기본으로 자리잡으며 여타 object detection 관련 알고리즘에 비해 좋은 성능을 보여주게 되었다.

그리고 2017년 Kaiming He는 Mask R-CNN이라는 논문을 통해 주목할만한 R-CNN 계열의 성능 향상을 가져왔다. 이 논문의 핵심은 이미지에 존재하는 각각의 픽셀이 객체인지 아닌지를 masking하는 CNN을 추가하는 것이었다.

그리고 Object Detection에 관련하여 가장 최근 주목된 연구는 2019년 발표된 M2Det이라는 알고리즘이다. 이 알고리즘은 영상 속에서 실시간으로 크기가 변화하는 물체들을 탐지하는 작업에 Multi-level & Multi-scale 특징을 사용했다. 이 연구는 자신들의 방법이 기존의 유명한 YOLO 보다 좋은 성능을 냈다고 보고하고 있다.

Ⅲ. RetinaNet

이상적인 목표는 가장 최근에 나온 M2Det이라는 뉴럴넷에 대한 논문을 보고 내가 직접 구현하여 적용하는 것이었다. 하지만 도저히 할 수 없어서 어쩔 수 없이 누군가 구현해 놓은 구현체를 가져다 쓰기로 했다. 그래서 처음에는 깃헙에서 LeeDongYeun이라는 분이 M2Det를 Keras implementation 해 놓은 것을 가져다 쓰려 했지만, initial comit 이후로 관리되고 있지 않을 뿐더러, 현재 tensorflow2.0과의 호환성에도 문제가 있는 것 같아 다른 것을 찾게 되었다.

그러던 와중 비록 2017년에 발표된 뉴럴넷이긴하지만 Keras implamentation 되어 2020년 현재까지 관리되고 있는 레포지토리를 찾게 되어 그것을 사용하기로 했다(github.com/fizyr/keras-retinanet). RetinaNet은 2017년에 Tsung-Yi Lin, Kaiming He 외 3명의 Facebook 연구진들이 발표했다.[1]

RetinaNet을 소개하는 논문은 2017년 이전에

발표되었던 SSD나 YOLO와 같은 one-stage Network의 Dense Object Detection은 R-CNN 계열의 two-stage Network에 비해 속도는 빠르지만 성능은 낮았다고 평가하며, 그 이유를 극단적인 클래스 불균형이 있다고 파악했다. 이에 RetinaNet은 클래스 분류에 흔히 사용되는 크로스 엔트로피 함수를 조금 수정한 Focal Loss 함수를 제안한다고 논문 초록에서 밝혔다.[2]

Ⅳ. RetinaNet for Food Detection

RetinaNet을 통해 Object Detection을 실험해 볼 데이터는 AI hub에서 제공하는 '2019 AI Starthon X 네이버'에서 수집했다. 수집한 로우 데이터는 iamge가 약 4만장, 각 이미지에 대한 label(bounding box 좌표) 값을 담은 텍스트 파일이 약 4만개였다. 이 모든 것을 가지고 모델을 학습시키는 것은 주어진 환경에서는 불가능했다. 그래서 약 250여장의 image만 뽑아서 진행했다.

학습 방법은 github.com/fizyr/keras-retinanet에 릴리즈 되어 있는 pre-trained model을 가져다가 transfer learning하는 방식으로 진행했다. 코드 작성의 뼈대는 깃헙에서 클론한 keras-retinanet 레포지토리에 들어가 있는 코드(keras-retinanet/examples/ResNet50RetinaNet.py)를 바탕으로 하여 웹 자료들을 참고하며 작성했다.[3]

Ⅴ. 결과

실습코드는 이미 COCO dataset으로 pre-trained 되어 있는 RetinaNet을 가져와서 feature extractor로 쓰며 본인이 가지고 있는 custom dataset에 transfer learning 시키는 것으로 작성되었다. 돌이켜보면 너무나 간단한 코드이기 때문에 리뷰할 코드도 없는 수준이다. 컴퓨터 성능이 여유롭지 않아서 충분한 데이터를 train에 활용하지 못하고, 충분히 많은 epoch을 돌지는 못했지만, 10번의 epoch을 도는 동안 Loss가 아주 바람직한 모양으로 떨어지는 것을 확인했다.

사실 이번 프로젝트에서 목표는 proposal에서도 밝혔다시피 논문을 보고 그걸 구현해서 내가 가지고 있는 custom dataset에 적용해서 object detection을 해내는 것이었다. 하지만 앞서 밝혔다시피 아직 본인의 수준에서는 직접 구현을 할 수 없다는 것을 깨달았고, 결국 남이 만들어놓은 구현체를 그냥 clone해서 사용하게 되었다. 그래서 사실 이번 프로젝트에서는 RetinaNet 논문을 보면서 그것이 도대체 어떻게 구현되었는지 깃헙에서 clone해온 레포지토리를 뒤져보면서 구조와 워크 플로우를 파악하는데 많은 시간을 할애했다.

하지만 레포지토리 파일들을 보면 볼수록 뭔가 핵심적이지 않은 파일들이 좀 많다는 것을 느꼈다. 그래서 Mask R-CNN이나 YOLOv3 같은 다른 유명한 알고리즘들에 대해 사람들이 만들어 놓은 다른 keras implementaion 레포지토리들을 찾아보니 한결 간명한 구조로 만들어져 있는 것들도 많았다. 흐름을 빨리 잡지 못한 것이 아쉽다.

Reference

[1] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

[2]http://blog.naver.com/PostView.nhn?blogId=sogangori&logNo=221087066947

[3]https://github.com/curiousily/Deep-Learning-For-Hackers