AI 학습을 위한 탑 인식 방법에 대한 연구

강은수[O], 고병국[*], 이조순[*], 최하진[*], 김준오[*], 이병권[**]
[O]동국대학교 멀티미디어공학과,
[*]동국대학교 멀티미디어공학과,
[**]서원대학교 멀티미디어과
e-mail: sonic747@seowon.ac.kr

A Study on Tower Recognition Method for AI Learning

Eunsu Kang[O], Ko Byeongguk[*], JoSun Lee[*], Choi Hajin[*], Jun O Kim[*], Byongkwon Lee[**]
[O]Dept. of Multimedia Engineering, Dongguk University,
[*]Dept. of Multimedia Engineering, Dongguk University,
[**]Dept. of Multimedia Seowon University

● 요 약 ●

본 논문에서는 AI 학습을 위한 데이터 수집을 위해 윈도우 환경에서 YOLO 시스템을 사용한 객체 인식에 대한 방법을 제안한다. 이 방법은 아나콘다, 리눅스 등의 가상환경을 요구하지 않기 때문에 실사용 이전 사전 환경설정 작업 시간을 최소화한다. 또한 이 방법은 Visual Studio, OpenCV, CUDA 등 익숙한 플랫폼 및 라이브러리를 요구하기 때문에 다른 사람들에게 편안한 작업환경 제공한다. 또한 기존의 COCO 데이터셋을 사용한 YOLOv3가 아닌 추가 학습 방법을 제안함으로써 보다 보편적인 객체 인식이 가능하다. 따라서 빠른 시간 내에 자신이 원하는 객체를 인식할 수 있는 시스템을 구축하는 방법을 제안한다.

키워드: YOLO, 윈도우 환경, 작업시간 최소화, 편안한 작업환경

I. Introduction

최근 원본과 다른 문화재 복원 작업에 대하여 논란이 존재한다. 원본에 대한 데이터가 존재할 경우 문화재 복원 작업이 원활하게 이루어질 수 있지만, 그렇지 않은 경우 원본과 유사한 복원 작업에 어려움이 생긴다.

따라서 손실된 문화재 영상을 GAN을 통하여 원본의 영상으로 복원시키는 연구를 진행하고자 한다. 특히 문화재 중 탑에 대한 연구를 진행하려 하며, 본 연구는 사전작업으로써 GAN에 필요한 데이터셋을 추출하기 위해 딥러닝을 통한 탑 인식을 하고자 한다.

II. Preliminaries

1. Related works

1) 경계 상자(Bounding Box)

이미지에서 객체들은 다양한 영역에 분포되어 있다. 이런 이유로, 이미지에서 객체를 추출하기 전에 객체가 어느 범위 내에 있는지를 경계 상자를 먼저 확인해야 한다.

Fig. 1. Bounding Box

2) 객체 인식 기법

① CNN : CNN은 정확도 높은 객체 추출이 가능하지만, 윈도우 검출 슬라이딩에서 획득한 수많은 이미지를 CNN으로 실행하는 것은 불가능했다. 영역(region)을 고려한 R-CNN 알고리즘을 사용하여 이 문제를 해결할 수 있다. 이 방법은 객체 분류에 필요한 경계 상자의 수를 줄일 수 있다. 이 방법은 선택적 검색을 위해, 질감, 강도, 컬러, insideness 등의 로컬 정보를 사용한다. 하지만, 선택적

검색에 의해 생성된 2000개의 region은 CNN처리에 많은 시간이 소요된다[1].

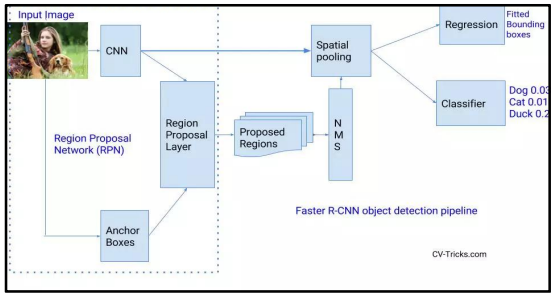

Fig. 2. R-CNN Structure

② SSD : SSD는 객체 검출 속도 및 정확도 사이의 균형이 있는 알고리즘이다. SSD는 한 번만 입력 이미지에 대한 CNN을 실행하고 형상 맵(feature map)을 계산한다. 경계 상자 및 객체 분류 확률을 예측하기 위해 이 형상 맵을 3 × 3 크기로 CNN을 수행한다. SSD는 CNN처리 후 경계 상자를 예측한다. 이 방법은 다양한 스케일의 물체를 검출 할 수 있다[2].

속도와 정확도의 균형을 추구할 수 있으나 환경설정 및 시스템 사용 자체에는 어려움이 있는 편이다.

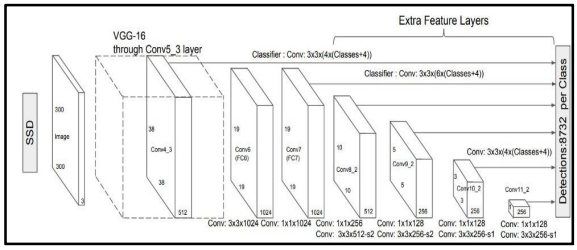

Fig. 3. SSD Structure

③ YOLO : YOLO는 각 이미지를 S x S 개의 그리드로 분할하고, 그리드의 신뢰도를 계산한다. 신뢰도는 그리드 내 객체 인식 시 정확성을 반영한다. 다음 그림과 같이 처음에는 객체 인식과는 동떨어진 경계 상자가 설정되지만, 신뢰도를 계산하여 경계 상자의 위치를 조정함으로써, 가장 높은 객체 인식 정확성을 가지는 경계 상자를 얻을 수 있다[3].

그리드에 객체 포함 여부를 계산하기 위해, 객체 클래스 점수를 계산한다. 이 결과로 총 S x S x N 객체가 예측된다. 이 그리드의 대부분은 낮은 신뢰도를 가진다. 신뢰도를 높이기 위해 주변의 그리드를 합칠 수 있다. 이후, 임계값을 설정해 불필요한 부분은 제거할 수 있다.

YOLO는 단순한 처리로 속도가 매우 빠르다. 기존 다른 real-time 비전 기술과 비교할 때, 2배 정도 높은 성능을 보인다. 이미지 전체를 한 번에 바라보는 방식으로 클래스를 분별한다[4].

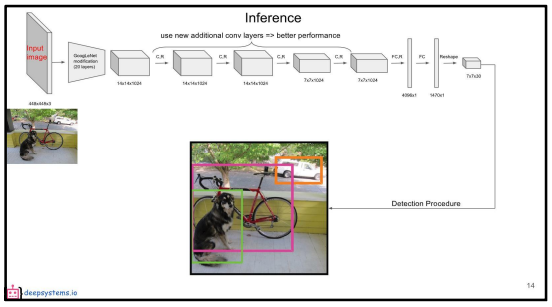

Fig. 4. YOLO Structure

④ 비전에서 객체를 검출할 때, 속도와 정확도는 다음 그림과 같이 trade-off 관계가 있다. 그러므로, 활용 목적에 따라 적절한 알고리즘을 응용하는 것이 필요하다. 본 연구에서는 빠른 환경설정과 빠른 결과 도출을 위해 YOLO를 사용하였다.

Fig. 5. Speed and Accuracy Graph

III. The Proposed Scheme

YOLO 진행 사전에 윈도우에서의 환경설정이 필요하다. 다음 이미지는 YOLO 환경설정 이전에 설치가 되어있어야 하는 프로그램들이다.

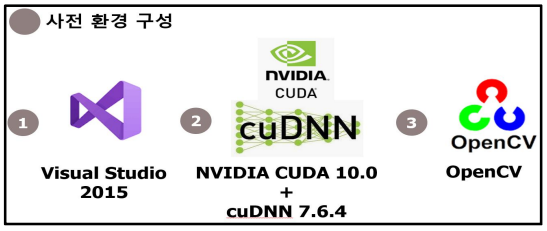

Fig. 6. Pre-prepared environment

① Visual Studio 2015 : 프로젝트 코딩 및 릴리즈
② CUDA & cuDNN : 학습에서의 GPU 사용
③ OpenCV : 이미지 처리 및 입출력

1. YOLOmark 환경설정

① 링크 : https://github.com/AlexeyAB/Yolo_mark
② 압축 해제 후 디렉토리 내에 있는 yolo_mark.sln를 실행한다.
③ 프로젝트 속성에 진입하여 구성 및 플랫폼을 Release / x64로

변경한다.

④ 구성속성=>C/C++=>일반=>추가 포함 디렉터리에 OpenCV의 include 경로를 추가한다.

⑤ 구성속성=>링커=>일반=>추가 라이브러리 디렉터리에 OpenCV의 lib 경로를 추가한다.

⑥ 솔루션 빌드하면 x64 폴더에 yolo_mark.cmd 및 yolo_mark.exe가 성공적으로 생성된다.

2. YOLO_Darknet 환경설정

① 링크 : https://github.com/AlexeyAB/darknet

② darknet-master/build/darknet 경로의 darknet. sln을 실행한다.

③ YOLOmark와 같이 프로젝트 속성에서 구성 및 플랫폼을 Release / x64로 변경한다.

④ 구성속성=>C/C++=>일반=>추가 포함 디렉터리에 OpenCV 및 CUDA의 include 경로를 추가한다.

⑤ 구성속성=>링커=>일반=>추가 라이브러리 디렉터리에 OpenCV 및 CUDA의 lib 경로를 추가한다.

⑥ 구성속성=>CUDA C/C++=>Device=>Code Generation을 수정한다. NVIDIA GPU의 compute 값을 수정하는 것인데 아래의 링크를 참조한다.

링크 : https://developer.nvidia.com/cuda-gpus

⑦ 다음 명시된 OpenCV 및 CUDA dll파일을 darknet의 x64 폴더안에 넣어준다.

경로 : C:/opencv/opencv/build/x64/vc14/bin

opencv_world3410.dll (3410은 버전명인 3.4.10)

경로 : C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.0/bin

cusolver64_100.dll

curand64_100.dll

cudart64_100.dll

cublas64_100.dll

⑧ 솔루션을 빌드하면 x64 폴더에 darknet.exe 가 성공적으로 생성된다.

3. YOLOmark를 통한 학습용 이미지 라벨링

① Yolo_mark-master/x64/Release/data 경로로 진입한다.

② obj.data 내용을 수정한다.

Classes : 학습할 객체의 수 (ex) 탑 인식 = 1)

Train : 학습 이미지 데이터 경로가 담긴 텍스트

Valid : 검증 이미지 데이터들의 경로가 담긴 텍스트

Names : names파일 경로

Backup : 학습 시 가중치 파일 저장 경로

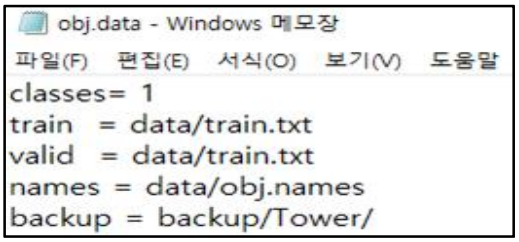

Fig. 7. obj.data contents

③ obj.names 내용을 수정한다. 이때 내용은 obj.data의 classes 값만큼 객체의 이름을 기입해주면 된다.

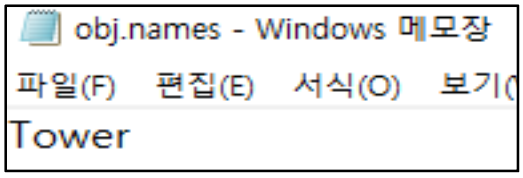

Fig. 8. obj.names contents

④ x64/Release 경로의 yolo_mark.cmd를 실행시며 마우스 드래그를 통하여 라벨링 작업을 시작한다.

Fig. 9. executed Yolo_mark.cmd

⑤ 라벨링에 성공하면 라벨링 내용이 txt파일로 만들어진다. 이때 값은 라벨 인덱스, 중심좌표(x, y), 길이(x, y) 총 5개의 값을 가진다. 또한 train.txt에 라벨링된 이미지의 경로가 저장된다.

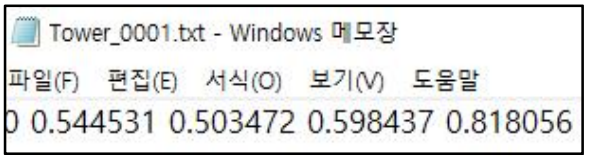

Fig. 10. Labeled txt contents

4. YOLO_Darknet을 통한 학습

① 3에서 YOLOmark의 data 경로내의 파일을 전부 Darknet-master/build/darknet/x64/data 경로로 복사한다.

② Yolo_mark-master/x64/Release 경로에 있는 yolo-obj.cfg 파일을 Darknet-master/build/darkn et/x64/cfg 경로로 복사해준다.

③ 복사된 yolo-obj.cfg 파일을 학습 형태에 맞게 수정한다.

max_batches : 학습 최대 반복수

최하단 [region]의 classes : obj.data의 classes값

[region] 바로 위의 [convolutional]의 filter 수정

- filters = (classes + 5) * 5

④ 아래 링크를 통해 기본 학습 모델 x64에 다운로드

http://pjreddie.com/media/files/darknet53.conv.74

⑤ cmd창을 열어 darknet폴더의 x64폴더로 이동

⑥ 학습 코드 실행

darknet detector train .data경로 .cfg경로 darknet 53.conv.74

정상적으로 실행되면 아래와 같이 그래프창이 나타나며, 지정된 반복회수마다 backup 디렉토리에 가중치 파일이 저장된다.

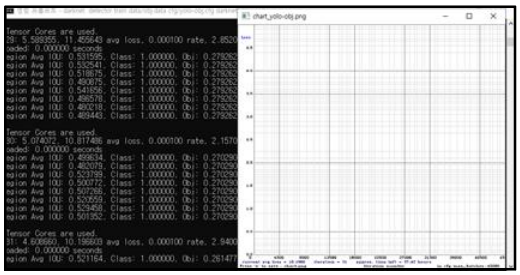

Fig. 11. Executed YOLO_Darknet

5. YOLO_Darknet을 통한 객체 인식

① 가중치 파일을 통한 객체 인식 코드 실행

darknet detector test .data경로 .cfg경로 .weights경로 test이미지 경로

아래의 이미지는 탑 데이터셋으로 학습된 가중치파일로 탑을 인식해본 결과이다.

Fig. 12. Recognized Tower with weights

IV. Conclusions

AI 학습에 있어서 객체 인식은 가장 중요한 부분 중 하나이다. 본 연구는 탑에 GAN을 적용하여 새로운 탑을 생성하기 이전에 GAN에 필요한 데이터셋을 추출하기 위한 사전연구이다. YOLO 시스템을 이용하여 탑에서 충분한 특징점을 추출하여 잘 인식할 수 있는지를 연구했다.

본 연구에서 제안하는 방법은 아나콘다, 우분투 등 특별한 가상환경을 요구하지 않으며, 환경설정 또한 간편하다. 따라서 딥러닝을 통해 객체 인식 모델을 개발 또는 테스트하고자 하는 사람들에게 도움을 줄 수 있다.

ACKNOWLEDGEMENT

이 (성과)는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2020R1A2C100766811).

REFERENCES

[1] Lee Yong-Hwan and Kim Youngseop, "Comparison of CNN and YOLO for Object Detection" Journal of the semiconductor & display technology, Vol. 19, No. 1, pp. 85-92, 2020.

[2] Liu, Wei , Anguelov, Dragomir , Erhan, Dumitru , Szegedy, Christian , Reed, Scott , Fu, Cheng-Yang , Berg, Alexander C. "SSD: Single Shot MultiBox Detector" Lecture Notes in Computer Science, Vol. 2016, No. 9905, pp. 21-37, Oct. 2016.

[3] Joseph Redmon and Ali Farhadi, ”YOLOv3: An Increm ental Improvement” Tech Report, 2018

[4] Yoon Kim, ”Convolutional Neural Networks for Sentence Classification” Empirical Methods in Natural Language Processing, pp.1746-1751, Oct. 2014.