일반논문 (Regular Paper)
방송공학회논문지 제25권 제3호, 2020년 5월 (JBE Vol. 25, No. 3, May 2020)
https://doi.org/10.5909/JBE.2020.25.3.386
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

물체인식 딥러닝 모델 구성을 위한 파이썬 기반의 Annotation 툴 개발

임 송 원[a], 박 구 만[a]‡

Development of Python-based Annotation Tool Program for Constructing Object Recognition Deep-Learning Model

Song-Won Lim[a] and Goo-man Park[a]‡

요 약

본 논문에서는 물체인식 딥러닝 모델을 구성하는데 필요한 데이터 레이블링 과정을 하나의 프로그램에서 사용할 수 있는 Annotation 툴을 개발했다. 프로그램의 인터페이스는 파이썬의 기본 GUI 라이브러리를 활용하였으며, 실시간으로 데이터 수집이 가능한 크롤러 기능을 구성하였다. 기존의 물체인식 딥러닝 모델인 Retinanet을 활용하여, 자동으로 Annotation 정보를 제공하는 기능을 구현했다. 또한, 다양한 물체인식 네트워크의 레이블링 형식에 맞추어 학습할 수 있도록 Pascal-VOC, YOLO, Retinanet 등 제각기 다른 학습 데이터 레이블링 형식을 저장하도록 했다. 제안하는 방식을 통해 국산 차량 이미지 데이터셋을 구축했으며, 기존의 물체인식 딥러닝 네트워크인 Retinanet과 YOLO 등에 학습하고, 정확도를 측정했다. 차량이 진입하는 영상에서 실시간으로 차량의 모델을 구별하는 정확성은 약 94%의 정확도를 기록했다.

Abstract

We developed an integrative annotation program that can perform data labeling process for deep learning models in object recognition. The program utilizes the basic GUI library of Python and configures crawler functions that allow data collection in real time. Retinanet was used to implement an automatic annotation function. In addition, different data labeling formats for Pascal-VOC, YOLO and Retinanet were generated. Through the experiment of the proposed method, a domestic vehicle image dataset was built, and it is applied to Retinanet and YOLO as the training and test set. The proposed system classified the vehicle model with the accuracy of about 94%.

Keyword : Annotation, GUI(Graphical User Interface), Tkinter, Crawling, Retinanet, YOLO

a) 서울과학기술대학교 일반대학원 미디어 IT 공학과(Seoul National University of Science And Technology)
‡ Corresponding Author : 박구만(Goo-man Park)
E-mail: gmpark@seoultech.ac.kr
Tel: +82-2-970-6430
ORCID: https://orcid.org/0000-0002-7055-5568
※ This study was supported by the Research Program funded by the SeoulTech(Seoul National University of Science and Technology).
· Manuscript received January 17, 2020; Revised February 28, 2020; Accepted March 6, 2020.

Ⅰ. 서 론

최근 컴퓨터 비전 인식 문제는 딥러닝을 활용한 여러 분야로 확장하면서[6,10], 다양한 신경망(Neural Network) 이 구현되고 있으며, 중요성이 커지고 있다. 이에 따라, 딥러닝 신경망 구성을 위한 학습데이터 생성 과정에서 레이블링(Labeling) 작업을 수행하기 위한 Annotation 프로그램은 필수적이다[11]. 특히, 프로그램의 기능에 따라 요구되는 노동력과 시간은 크게 달라질 수 있다. 최근에는 아마존의 AWS, IBM 의 IBM Watson Studio 와 같은 클라우드에서 이미지 데이터에 대한 머신러닝 방식에 따른 레이블링 작업을 할 수 있는 플랫폼 서비스를 제공하고 있다. 대표적으로, AWS의 'SageMaker Ground Truth' 서비스 경우, 사용자가 제공한 대량의 Law 이미지를 레이블링 데이터로 제공한다. 클라우드는 사용자가 보낸 데이터에 대해서 Annotation 작업으로 처리하는 과정을 거친다. 따라서, 학습데이터를 생성하는 과정은 시간과 비용이 뒤따른다.

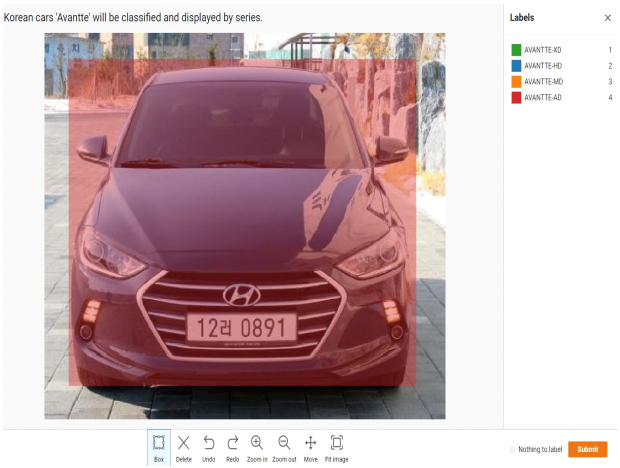

그림 1. AWS의 Sagemaker Ground Truth 레이블링 작업
Fig. 1. AWS Sagemaker Ground Truth Labeling Operation

본 논문에서는 GUI(Graphical User Interface) 기반의 사용자 편리성을 포함한 레이블링 방식을 개발하여 시간과 비용을 절약하도록 하였다. 구현한 프로그램을 통해 110종 모델의 차량에 대한 이미지 데이터를 수집하고, 차량 학습 데이터셋도 구성하였다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 GUI 프로그램에 대한 전체 구조를 설명하고, 3장에서는 각 기능에 대한 구현 방법에 대해 언급하였다. 특히, 사용자가 편리하게 Annotation 기능을 수행하도록, 기존의 물체인식 네트워크[4]와 GUI 프로그램을 연동하고, 물체인식을 통해 레이블링 데이터를 생산하는 방법을 제안하였다. 4장에서는 차량 이미지 데이터셋에 대한 생성 과정을 설명하였으며, 5장에서는 기존의 딥러닝 모델에 대한 실험 및 활용에 대한 검토를 한 후, 6장에서 본 논문에 대한 결론을 맺었다.

Ⅱ. GUI 프로그램 구조

기존의 Annotation 프로그램은 특정 딥러닝 네트워크에만 학습이 가능한 레이블링 형식만을 제공하며, 많은 양의 이미지 데이터를 사용자가 박스 형태로 직접 번거롭게 그려야 한다. 하지만 제안하는 프로그램은 사용자가 버튼 클릭만으로 Annotation 기능을 수행하고, 사용자가 원하는 이미지 데이터도 직접 수집할 수 있는 기능이 있다.

그림 2는 구현한 GUI 프로그램의 전체적인 구조도이다. 파이썬 기반의 GUI 라이브러리 Tkinter 를 활용하여, 전체적인 인터페이스를 구성했다. 우선, 프로그램에서 사용할 수 있는 각각의 기능들을 직접 구현하고 모듈화하여, Tkinter에 포함시켰다. 구현한 기능에는 사용자가 원하는 이미지 데이터를 수집할 수 있는 크롤링(Crawling) 기능과 사물을 자동으로 인식하고, 레이블링 과정을 동시에 수행할 수 있는 Annotation 기능이 있다.

우선 크롤링 기능에 있어서 Chrome, Firefox와 같은 기존의 웹브라우저를 조작할 수 있는 Selenium 라이브러리를 활용하여, 특정 웹사이트에서 원하는 이미지 데이터를 저장할 수 있는 다수의 크롤링 모듈을 구현했다. 특정 웹사이트는 사용자가 클릭으로 선택할 수 있도록 버튼 인터페이스를 마련하였다. 또한 저장한 영상 데이터에 대해서 양을 증가시켜주는 Augmentation 모듈을 추가하여, 데이터의 양을 증가시키면서 특정 폴더에 저장하도록 했다.

물체인식은 케라스(Keras) 기반으로 구현한 레티나넷(Retinanet)을 통해, COCO 데이터셋[1]으로 학습하고, 학습

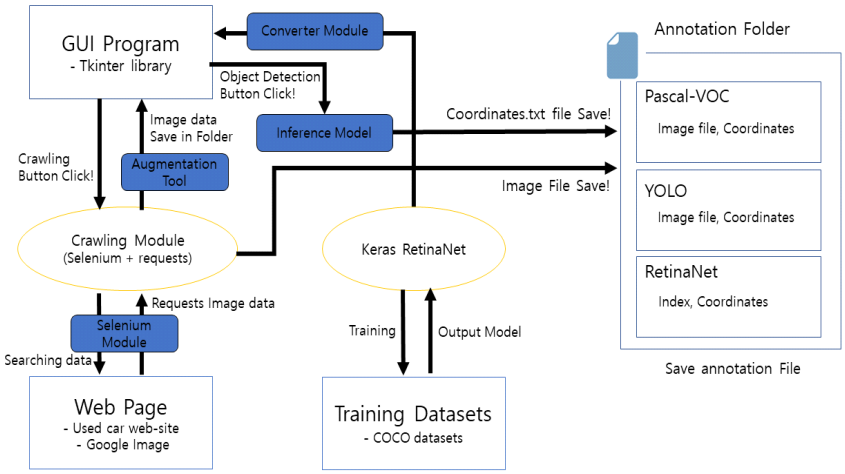

그림 2. GUI 프로그램 구조도
Fig. 2. GUI Program Construction

된 가중치 파일을 GUI 프로그램과 연동했다.

또한 Annotation 기능은 저장한 영상 데이터에서 물체인식을 통해, Pascal-VOC[2], YOLO(You Only Look Once)[3], Retinanet[4] 등 여러 가지의 형식으로 저장하도록 구현했다. 각각의 구현한 기능에 대한 자세한 방법은 다음 장에서 제시했다.

Ⅲ. 프로그램 구현 방법 및 Retinanet 학습 모델

기존의 GUI 라이브러리는 PyQT, PyGUI, WxPython, PySide 등 매우 다양하지만, 본 프로그램에서는 파이썬에서 기본으로 제공하는 라이브러리인 Tkinter를 활용했다. Tkinter는 다른 라이브러리에 비해서 많은 기능은 없지만, 사용자가 GUI 프로그램을 구성하는데 쉽게 접근할 수 있는 장점이 있다. Tkinter는 프로그램의 윈도우 구성 및 버튼 조작, 옵션 메뉴, 영상 표시, 텍스트 표시 등 인터페이스 구성에 필요한 기본적인 기능을 주요 위젯 함수로써 제공한다. 또한 GUI 프로그램에 연결하는 각각의 모듈도 위젯 기능을 통해 연동할 수 있다. Tkinter의 위젯 기능을 활용하여 구현한 기능들은 다음과 같다.

1. 보간법

그림 3은 구현한 Annotation 프로그램의 인터페이스를 나타낸다. 읽어 들인 차량의 영상을 보면 해상도가 다양해도 깨짐 없이 올바르게 나타냈다. GUI 프로그램에서 전처리 없이 영상을 불러올 경우, 영상이 잘리거나, 화소가 뭉개진 형태로 나오는 등의 문제가 발생하기 때문에, 적합한 보간법(Interpolation)을 적용해야만 한다. 본 프로그램에서는 인접한 16개 화소의 화소값과 거리에 따른 가중치의 곱을 사용하여 결정하는 Bicubic 보간법[5]을 사용하여, 해상도 변화에 상관없이 영상이 올바르게 보이도록 했다. 적용한 보간법 모듈은 Tkintert에 포함시켜 불러오는 영상이 가로, 세로의 불규칙한 비율에도 올바로 표출한다.

2. 데이터 수집 기능

기존의 프로그램은 불러온 영상에 대한 Annotation 정보만을 제공할 뿐, 영상 데이터를 수집하는 것은 사용자가 별도로 구성해야 한다. 제안하는 프로그램은 사용자가 영상 데이터도 수집할 수 있도록 별도의 크롤링 모듈을 구현하고 GUI 프로그램과 연동함으로써, 사용자가 하나의 프로그램에서 데이터 수집과 Annotation 기능을 동시에 사용할 수

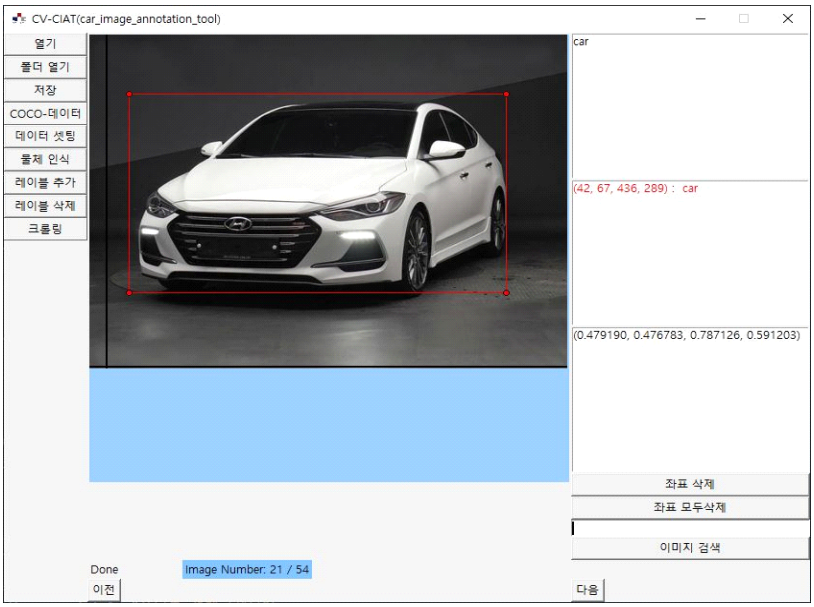

그림 3. 구현한 GUI 어노테이션 프로그램
Fig. 3. Implemented GUI Annotation Program

있도록 했다.

크롤링 모듈은 크게 두 가지의 기능으로, 첫 번째는 특정 웹사이트의 이미지를 저장할 수 있는 모듈이다. 파이썬 기반의 Selenium 라이브러리를 활용하여 특정 웹사이트에서 이미지 데이터를 수집할 수 있는 모듈을 구현했다. 모듈에는 국내 유명 중고차 사이트 6곳에서 차량의 이미지를 저장 할 수 있다. 중고차 사이트가 아닌 다른 특정 웹사이트에서도 Selenium을 통해 데이터를 수집할 수 있는 기능을 구현하고 GUI 프로그램과 연동 할 수 있다. Selenium은 Chrome, Firefox와 같은 웹브라우저(Web-Browser)를 조작할 수 있는 모듈로, 사용자가 수집하고 싶은 데이터에 접근하여 저장할 수 있도록 다양한 언어 기반의 함수로 구성되어 있다. 그림 4와 같이, 구현한 모듈에서는

그림 4. 중고차 사이트 크롤링 모듈 구현
Fig. 4. Implementing a used car site crawl module

중고차 사이트에서 제조사별, 연식별, 모델명 별로 정리되어있는 차량의 정보를 받고, 사용자는 원하는 차량의 이미지를 저장할 수 있다.

두 번째 크롤링 기능은 사용자가 찾고 싶은 사물의 키워드를 GUI 프로그램에 입력하면 Google, Bing, Baidu와 같은 웹사이트의 이미지 검색 페이지로 연결하여 일치하는 영상을 저장해준다. 그림 5는 사용자가 찾고자 하는 영상의 키워드에 대해서 검색하는 모습이다. 그림과 같이 '공유'라는 연예인의 이름을 프로그램의 인터페이스 검색 창에 기입하고, 이미지 검색 버튼을 클릭하면, 해당 이미지에 대해 저장할 수량을 묻는 명령창이 나온다. 그림 5의 우측과 같이 사용자가 원하는 수량에 따라 해당 키워드인 '공유'라는 폴더가 생성되고, 다수의 이미지를 저장한다.

3. Augmentation 기능

물체인식 딥러닝 모델의 학습 효과를 높이기 위해서는 학습데이터의 양이 많아야 좋은 학습 결과로 이어진다. 그러므로 데이터를 생성하는 과정에서 영상의 양을 증가시키는 과정은 필수적이다. 구현한 프로그램에서는 크롤링 모듈을 통해 저장한 영상에 대해, 회전이나 확대 및 자르기와 같은 효과뿐 아니라 다양한 노이즈(Noise) 및 필터(Filter) 등의 효과를 사용자의 선택에 따라 적용할 수 있다. 또한 기존의 Augmentation API(Application Programing Interface) 모듈인 'imgaug' 모듈을 프로그램에 추가하여, 기존의 이미지에서 안개 및 눈과 같은 날씨 데이터도 생성할 수 있도록 구성했다. 그림 6은 차량 영상에 대해, Augmentation 과정을 수행한 결과를 보여준다.

4. Annotation 기능

영상을 기존의 물체인식 딥러닝 모델에 학습하기 위해서는 레이블링된 데이터를 포함해야 한다. 레이블링 데이터는 객체가 어떠한 사물인지에 대한 클래스 정보와 영상 내에서 인식하고자 하는 객체의 후보 영역 좌표로 구성된다. 후보 영역 좌표의 경우 대표적으로 R-CNN(Region Convolutional Neural Network)[6] 방식을 따른다. 기존에 다양한 레이블링 프로그램은 이미지 내의 객체를 사용자가 일일이 경계 상자(Bounding Box) 형태로 그려야하는 Annotation 작업으로, 많은 시간이 소모된다.

본 연구에서 제시하는 방식은 사용자의 수작업을 없애기 위해, 저장한 이미지 데이터를 객체 인식 과정을 통해서 나오는 Annotation 정보를 저장하는 것이다. 이를 위해서 GUI 프로그램에서 물체 인식 기능을 포함하려면 기존의 딥

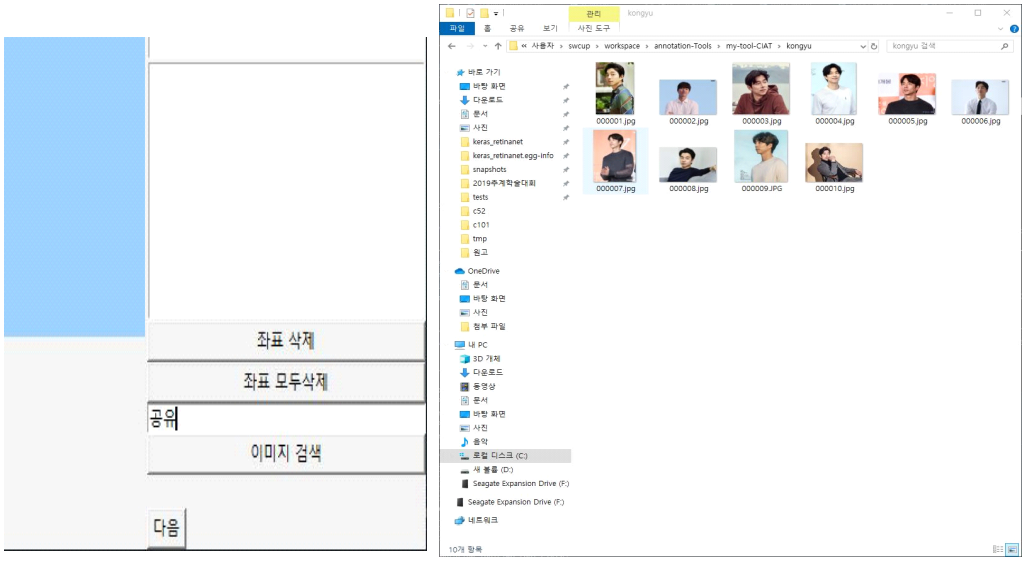

그림 5. 키워드 검색 크롤링 모듈
Fig. 5. Crawling module for keyword search

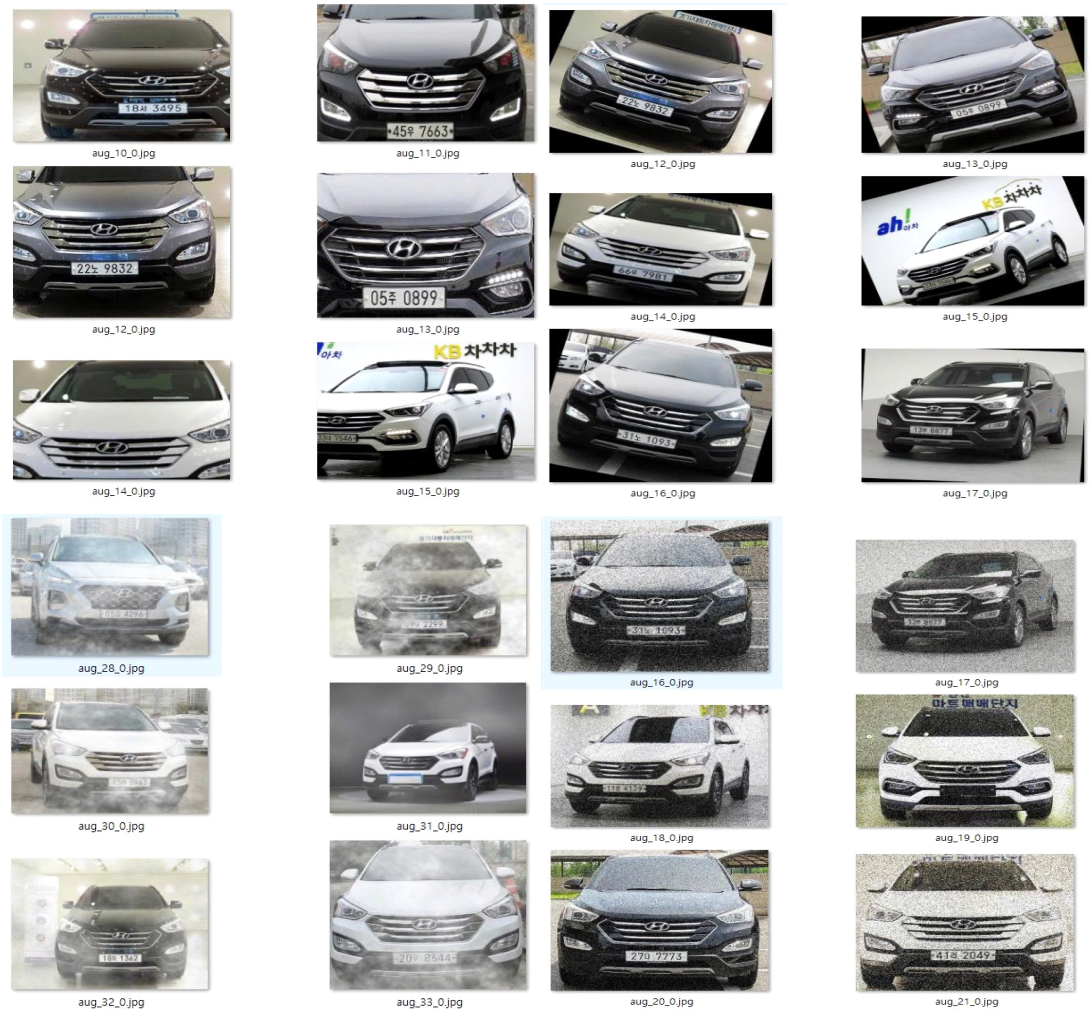

그림 6. Augmentation 이미지 데이터
Fig. 6. Augmentation Image Data

러닝 네트워크로 학습된 가중치 파일과 연동해야 한다.

우선, 케라스 기반의 Retinanet을 활용하여 대표적인 사물들의 이미지를 학습하기 위해 80개 범주의 사물 이미지를 제공하는 COCO Classification 데이터셋[1]을 활용했다. Retinanet 모델을 기반으로 COCO 데이터셋을 학습하였으며, 학습 결과 중, 가장 정확도가 높은 가중치 파일을 선별했다. 케라스 Retinanet 에 구성되어 있는 Convert Module을 통해 대용량 처리가 가능한 'hdf5' 포맷으로 가중치 파일을 변환한 후, Tkinter 상에서 Tensorflow 를 통해 변환한 가중치 파일을 불러왔다. 이로써 GUI 프로그램에서 사물을 인식할 수 있는 기능을 구성할 수 있다.

또한, 학습한 COCO 데이터셋의 80개 사물 범주를 사용자가 선택할 수 있도록 그림 7과 같이 메뉴 옵션을 추가하였다. 예를 들어, 사람을 인식하고자 하면 'person' 이라는 카테고리를 선택하여 사용자가 불러온 영상 내에서 사람을 인식할 수 있다. 그림에서와 같이 영상 데이터에 대한 레이블링 작업에서 하나의 영상에서도 많은 사물의 범주를 포함해야하는 경우가 생긴다. 구현한 프로그램에서는 한 번의 버튼 클릭만으로도 이미지 내의 모든 객체를 인식하고, 이에 대한 절대 좌표와 R-CNN 계열의 YOLO 네트워크의 상대 좌표까지 변환함으로써, 사용자가 학습 데이터를 구성하는데 필요한 시간을 줄일 수 있다. 특히, R-CNN 계열

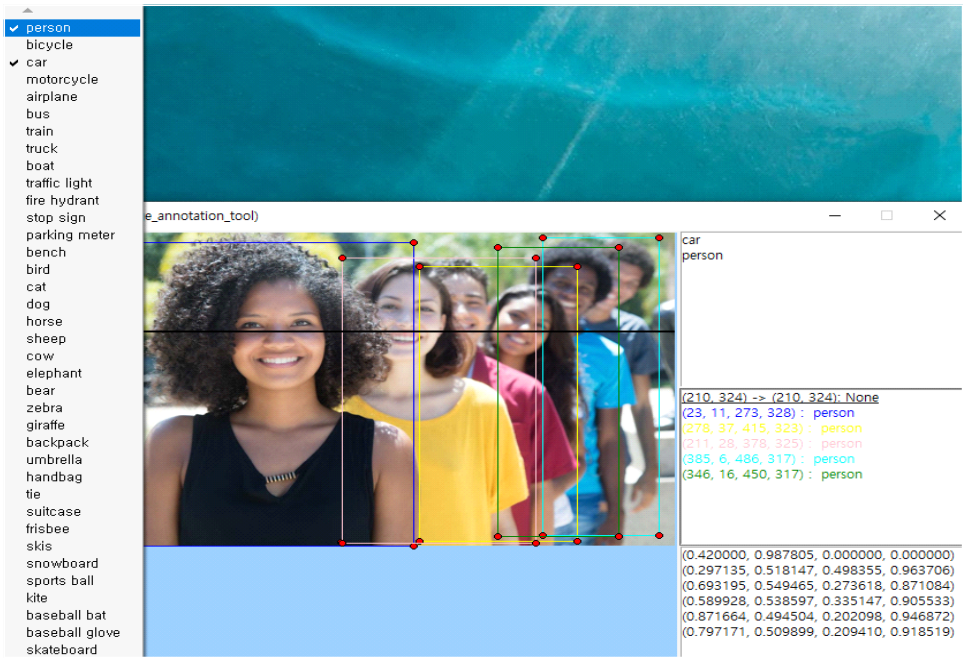

그림 7. GUI 프로그램 상에서 객체인식 모습
Fig. 7. Object recognition on the GUI program

의 YOLO 네트워크의 경우 Annotation 정보는 단순 절대 좌표가 아닌, Bounding Box Regression에 따른 4개의 좌표로 구성되도록 정의하였다[7].

x, y, w, h는 x, y 좌표는 경계 박스의 중심좌표를 의미하고, w, h는 경계 박스의 너비, 높이를 각각 의미하며, 이러한 객체의 후보 영역 박스가 실제 값인 Ground Truth 와의 차이를 줄여나가는 수식으로 정의된다[8]. GUI 프로그램에서 나타낸 객체의 절대 좌표 정보를 Bounding Box Regression 정의에 따른 상대 좌표로 변환하기 위해, Tkinter 상에서 수식 1과 같은 해당 식을 추가했다.

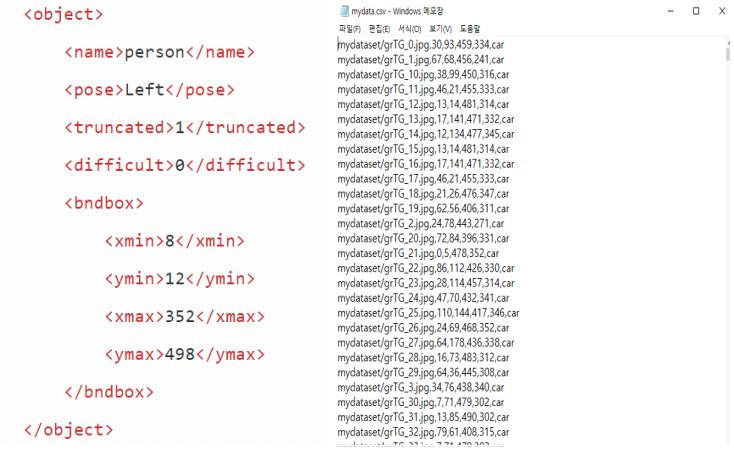

그림 8. (좌측부터) Pascal-VOC 형식과 Retinanet의 .csv 파일 형식
Fig. 8. Pascal-VOC format and .csv file format from Retinanet

$$\begin{aligned} t_x &= (x - x_a)/w_a \\ t_y &= (y - y_a)/h_a \\ t_w &= \log(w/w_a) \\ t_h &= \log(h/h_a) \end{aligned} \quad (1)$$

이밖에도 이미지 데이터셋 형식인 Pascal-VOC 형태로도 Annotation 정보를 저장하기 위해서 XML(Extensible Markup Language) 형식의 파일로 저장했다. Pascal-VOC의 레이블링 형태는 이미지 객체에 대한 클래스 정보와 인식한 영역의 좌표 정보를 XML 형식의 파일로 구성해야 한다. 특히 클래스 정보와 인식한 절대 좌표 정보를 포함하는 .csv 파일 형식으로 저장하도록 했다.

Ⅳ. 영상 데이터 셋 생성

구현한 GUI 프로그램을 통해, 많은 양의 데이터를 수집할 수 있으며, 빠르게 Annotation 기능을 수행할 수 있다. 이러한 기능을 활용하여, 대량의 영상 데이터 셋을 구성해보았다. 수집한 영상 데이터는 차량에 대한 이미지 데이터로 외국의 경우, 차량을 모델명별로 수집하고 정리한 데이터셋이 케글(Kaggle)에 공개됐지만[9], 아직 국내에서는 국산 차량에 대한 이미지를 수집한 데이터셋이 없다. 제조사별, 연식별, 모델명 별로 체계적으로 다양한 차량의 이미지를 확보하는 과정은 많은 시간이 소모되기 때문이다.

구현한 프로그램의 크롤링 모듈을 활용하면 제조사별, 연식별, 모델명 별로 저장한 차량 이미지 데이터 셋을 수집할 수 있다. 수집한 데이터셋은 KCD(Korea Cars Dataset)라는 이름으로 정의하고, 총 110종의 차량에 대한 이미지를 다양한 해상도와 각도로 구성했다. 구축한 차량 이미지 데이터는 그림 9와 같이 데이터를 수집하는 특정 중고차 사이트 별로 나누었다. 그리고 특정 사이트 폴더 내에서 차량의 연식별 모델명 별로 영상이 저장되도록 했다. 또한 영상의 크기별, 각도별로 총 3가지의 유형으로 나뉘어 저장하고, 영상의 양을 증가시키는 Augmentation 모듈을 적용한 결과의 영상들을 추가로 저장하여, 그림 10과 같이 약 9만장에 달하는 차량 이미지 데이터셋을 구축했다.

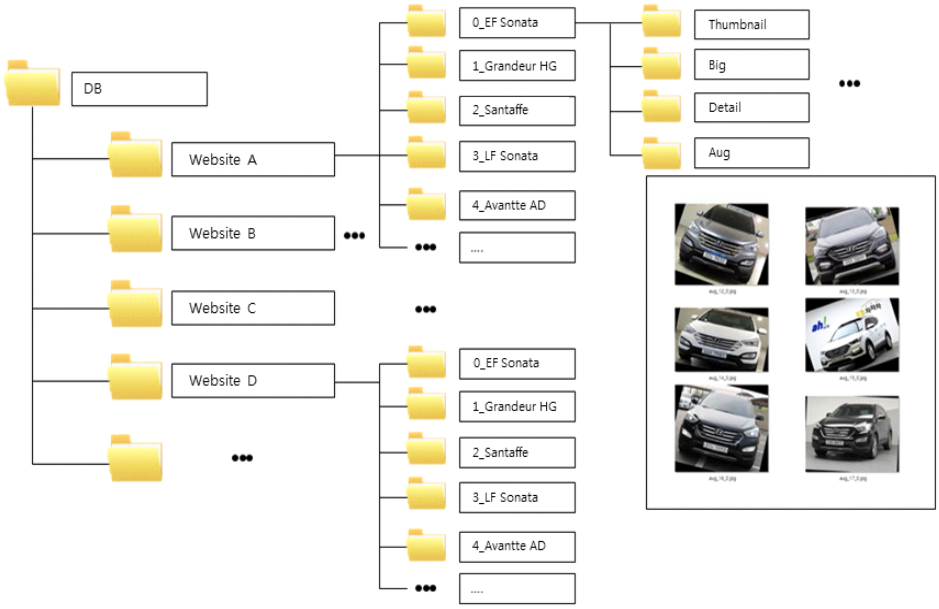

그림 9. GUI 프로그램에서 수집한 차량 이미지 저장구조
Fig. 9. Vehicle image data storage structure

그림 10. KCD(Korea Cars Dataset) 생성
Fig. 10. Create a KCD

V. 실험 및 평가

구성한 차량 데이터셋을 통해 기존의 물체인식 네트워크에서 적합한 학습 결과로 나타나는지를 평가해보았다. 무엇보다 효과적인 학습데이터 구성을 찾기 위해, 학습데이터의 구성을 표 1과 같이 4가지로 나누어 보았다. 구성된 학습데이터는 기존의 물체인식 모델인 Retinanet과 YOLO를 통해 학습하고, 정확성을 평가했다.

표 1. 학습데이터 구성
Table 1. Training data Composition

	Training Data Composition	Quantity
Data 1	Original Data	101(class) x 40
Data 2	Vehicle Left Image Data	101(class) x 55
Data 3	Original + Affine, Crop, Gray-scale	101(class) x 65
Data 4	Original + Affine, Crop + Noise + Weather	101(class) x 80

학습데이터의 구성은 Augmentation을 적용하지 않은 원본데이터인 Data 1, 진입하는 차량의 모습에 맞게 좌측 차량의 이미지만을 구성한 Data 2와 회전, 자르기, 흑백 효과를 적용한 Data 3 그리고 Data 3에서 추가로 노이즈와 날씨 데이터를 적용한 Data 4 로 구성된다. 학습데이터는 총 101종의 차량 이미지에 대해 학습을 했으며, 본 연구에 테스트 CPU의 사양은 i7-7700 3.60GHz의 GPU GeForce GTX-1080, RAM 32GB와 같다.

YOLO 모델에서 학습하는 경우에는 버전 2와 버전 3으로 나누어있기 때문에 각각의 다른 버전에 학습하고, 정확도를 비교해보았다. 비교한 정확성의 결과는 표로 작성하여 나타냈다. YOLO 버전 2에 필요한 학습 파라미터는 Subdivision을 8, 버전 3에서는 Subdivision을 32와 64 로 학습을 진행했다. 여기서 Subdivision은 Mini-batch 를 뜻하며, 한번 학습할 때 구성되는 데이터의 양을 말한다. 학습의 횟수는 데이터 구성마다 가장 높은 정확도로 평가하였다. 테스트 방법은 그림 11과 같이 차량이 진입하는 영상 2 가지를 준비하고, 그림 12와 같이 진입 영상의 차량 모델명을 적은 정답지를 작성했다. 그리고 그림 13과 같이 진입 영상에 대한 실시간으로 인식한 결과를 정답지와 비교하여 정확성을 평가했다. 야간에 진입하는 차량을 평가하기 위한 방법으로 IR(Infrared Rays) 영상의 스틸샷을 준비했다.

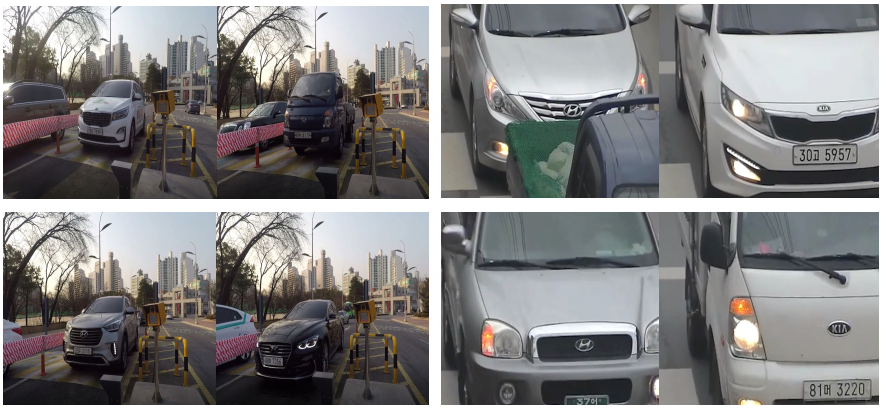

그림 11. (좌측부터) 테스트 영상1, 2
Fig. 11. Test Video 1, 2

	model_name_old	manufacturer_	model_name_new	car_type	frame_count						
1	그랜저HG	HYUNDAI	GRANDEUR HG	Mid	0	37	그랜저 IG	HYUNDAI	GRANDEUR IG	Mid	4111
2	그랜저 IG	HYUNDAI	GRANDEUR IG	Mid	102	38	K5 MX	KIA	K5 JF	Mid	4284
3	LF 쏘나타 뉴 라이즈	HYUNDAI	New Rise SONATA	Mid	216	39	스포티지 R	KIA	SPORTAGE R	Compact	4443
4	아반떼XD	HYUNDAI	AVANTTE XD	Compact	338	40	그랜저HG	HYUNDAI	GRANDEUR HG	Mid	4510
5	포터 2	HYUNDAI	POTER 2	Small	428	41	K5 MX	KIA	K5 JF	Mid	4620
6	LF 쏘나타	HYUNDAI	LF SONATA	Mid	570	42	싼타페 TM	HYUNDAI	SANTAFFE TM	Mid	4707
7	YF 쏘나타	HYUNDAI	YF SONATA	Mid	715	43	아반떼XD	HYUNDAI	AVANTTE XD	Compact	4808
8	SM3 네오	SAMSUNG	SM3 Neo	Compact	831	44	스포티지 KM	KIA	SPORTAGE KM	Compact	4908
9	쏘렌토 BL	KIA	SORENTO BL	Mid	966	45	더 suv 스포티지 R	CHEVOLET	The SUV SPORTAGE	Compact	5041
10	크루즈 J300	CHEVOLET	CRUSE J300	Compact	1091	46	올란도	CHEVOLET	ORLANDO	Compact	5143
11	K5 TF	KIA	K5 TF	Mid	1175	47	그랜저 IG	HYUNDAI	GRANDEUR IG	Mid	5323
12	봉고3 후기형	KIA	BONGO3 2	Small	1306	48	더 뉴 맥스크루즈	HYUNDAI	MAXCRUSE	Mid	5476
13	K5 MX	KIA	K5 JF	Mid	1451	49	포터 2	HYUNDAI	POTER 2	Small	5598
14	뉴 SM5 EX1	SAMSUNG	New SM5 EX1	Mid	1578	50	포터 2	HYUNDAI	POTER 2	Small	5680
15	YF 쏘나타	HYUNDAI	YF SONATA	Mid	1668	51	뉴 SM5 L43	SAMSUNG	New SM5 L43	Mid	5807
16	SM3 네오	SAMSUNG	SM3 Neo	Compact	1768	52	제네시스 G80	GENESIS	GENESIS g80	Full	5926
17	YF 쏘나타	HYUNDAI	YF SONATA	Mid	1846	53	i30 FD	HYUNDAI	i30 FD	Compact	6066
18	싼타페 TM	HYUNDAI	SANTAFFE TM	Mid	1927	54	아반떼AD	HYUNDAI	AVANTTE AD	Compact	6189
19	QM6	SAMSUNG	QM6	Mid	2001	55	LF 쏘나타	HYUNDAI	LF SONATA	Mid	6345
20	YF 쏘나타	HYUNDAI	YF SONATA	Mid	2146	56	i30 GD	HYUNDAI	i30 GD	Compact	6466
21	LF 쏘나타	HYUNDAI	LF SONATA	Mid	2270	57	뉴 모닝 SA	KIA	New MORNING	Small	6593
22	NF 쏘나타 transform	HYUNDAI	NF SONATA	Mid	2471	58	올 뉴 투싼 TL	HYUNDAI	All New TUCSON	Compact	6701
23	NF 쏘나타 transform	HYUNDAI	NF SONATA	Mid	2584	59	그랜저TG	HYUNDAI	GRANDEUR TG	Compact	6875
24	K5 SX	KIA	K5 JF	Mid	2651	60	트락스 LT	CHEVOLET	TRAX LT	Small	7016
25	올 뉴 쏘렌토UM	KIA	SORENTO UM	Mid	2761	61	K5 TF	KIA	K5 TF	Mid	7147
26	아반떼HD	HYUNDAI	AVANTTE HD	Compact	2898	62	K5 MX	KIA	K5 JF	Mid	7282
27	K5 MX	KIA	K5 TF	Mid	2967	63	더 뉴 K7 VG	KIA	The New K7	Mid	7440
28	레이 TAM	KIA	Rei	Small	3061	64	아반떼MD	HYUNDAI	AVANTTE MD	Compact	7557
29	LF 쏘나타	HYUNDAI	LF SONATA	Mid	3168	65	더 뉴 레이	KIA	The New Rei	Small	7640
30	코나 OS	HYUNDAI	KONA OS	Small	3283	66	싼타페 SM	HYUNDAI	SANTAFFE SM	Mid	7761
31	아반떼HD	HYUNDAI	AVANTTE HD	Compact	3524	67	싼타페 DM	HYUNDAI	SANTAFFE DM	Compact	7853
32	i30 GD	HYUNDAI	i30 GD	Compact	3639	68	그랜저 IG	HYUNDAI	GRANDEUR IG	Mid	7968
33	투싼 ix LM	HYUNDAI	TUCSON ix	Compact	3725	69	그랜드 스타렉스	HYUNDAI	Grand STAREX	Full	8079
34	QM3	SAMSUNG	QM3	Small	3830	70	올란도	CHEVOLET	ORLANDO	Compact	8253
35	스포티지 KM	KIA	SPORTAGE KM	Compact	4007	71	YF 쏘나타	HYUNDAI	YF SONATA	Mid	8404
						72	LF 쏘나타	HYUNDAI	LF SONATA	Mid	8523

그림 12. 테스트 영상1 정답지
Fig. 12. Test Video 1 Answer Sheet

표 2. 테스트 영상1 학습데이터 구성에 따른 정확도 평가
Table 2. Test Image 1 Accuracy evaluation according to learning data composition

Test	Training Model	Training Data Composition	Accuracy
A	YOLO v2: darknet-19	Original Data	89.90%
B	YOLO v2: darknet-19	Original + Affine, Crop, Gray-scale	94.30%
C	YOLO v3: darknet-19	Original Data	87.28%
D	YOLO v3: darknet-19	Vehicle Left-Image Data	90.48%
E	YOLO v3: darknet-19	Original + Affine, Crop, Gray-scale	93.33%
F	YOLO v3: darknet-19	Original + Noise + Weather	92.67%

표 3. 테스트 영상2 학습데이터 구성에 따른 정확도 평가
Table 3. Test Image 2 Accuracy evaluation according to learning data composition

Test	Training Model	Training Data Composition	Accuracy
A	YOLO v2: darknet-19	Original Data	88.30%
B	YOLO v2: darknet-19	Original + Affine, Crop, Gray-scale	91.20%
C	YOLO v3: darknet-19	Original Data	88.28%
D	YOLO v3: darknet-19	Vehicle Left-Image Data	83.89%
E	YOLO v3: darknet-19	Original + Affine, Crop, Gray-scale	90.33%
F	YOLO v3: darknet-19	Original + Noise + Weather	87.12%

표 4. IR 이미지 정확성 평가
Table 4. IR image accuracy evaluation

Test	Training Model	Training Data Composition	Accuracy
A	YOLO v2: darknet-19	Original Data	94.65%
B	YOLO v2: darknet-19	Original + Affine, Crop, Gray-scale	96.40%
C	YOLO v3: darknet-19	Origianl Data	90.00%
D	YOLO v3: darknet-19	Vehicle Left-Image Data	86.28%
E	YOLO v3: darknet-19	Original + Affine, Crop, Gray-scale	91.71%
F	YOLO v3: darknet-19	Original + Noise + Weather	90.22%

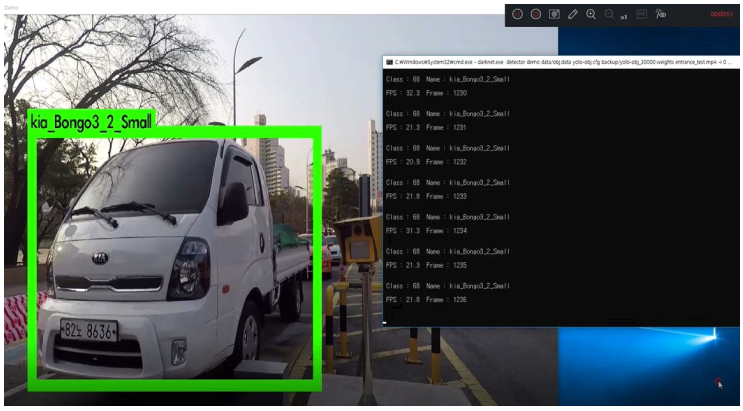

그림 13. 차량 데이터 셋으로 학습한 YOLO 인식 결과 모습
Fig. 13. YOLO model recognition result learned with vehicle dataset

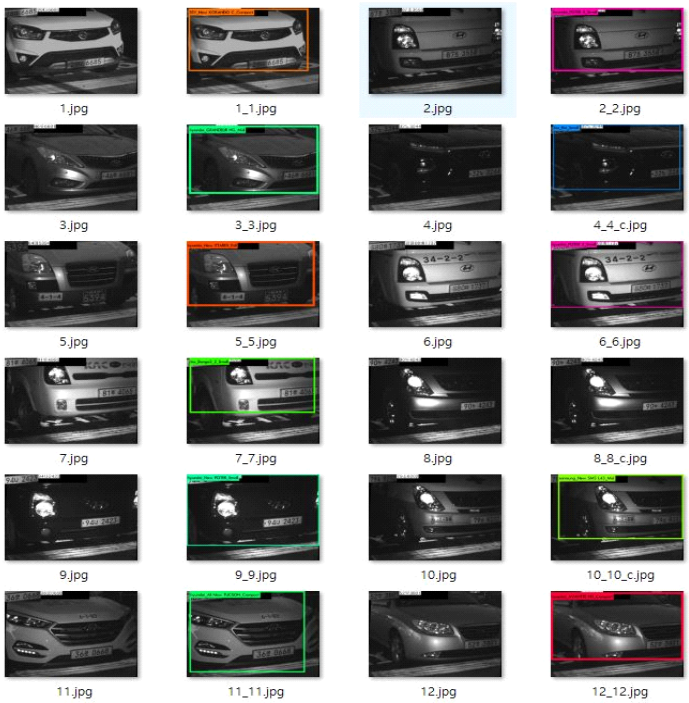

그림 14. IR 이미지 YOLO 모델 인식 결과 모습
Fig. 14. IR image recognition result

차량이 진입하는 영상에 대한 평가 결과를 표 2와 표 3에 나타냈다. 준비한 테스트 영상을 통해 공통적으로 높은 정확성을 기록한 학습데이터 구성이 있었다. 높은 정확성을 보인 경우에는 표 1의 Data 3과 같은 회전, 자르기(Crop)와 같은 기본적인 Augmentation을 적용한 것과 원본 데이터를 조합했을 때, 가장 높은 정확성을 기록했다. 날씨데이터를 조합한 경우에도, 표 2의 F와 같이, YOLO v3에서 학습한 결과의 정확성이 약 93%를 기록하였다.

YOLO 모델에서 정확성이 높은 결과로 드러난 학습데이터 Data 3으로 Retinanet에서도 학습하고 그 결과를 별도의 테스트 모듈을 구성하여, 그림 15와 같이 정확성을 나타나도록 하였다. Retinanet의 평균적인 정확도는 약 91%를 기록했다.

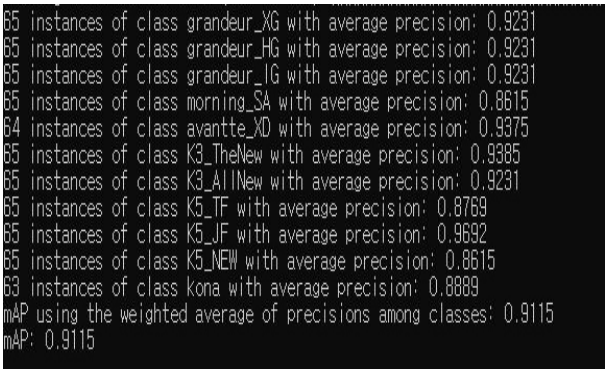

그림 15. 레티나넷 학습 결과
Fig. 15. Retinanet recognition result

Ⅵ. 결 론

본 논문에서는 영상 데이터를 딥러닝 네트워크에 학습하기 위해 필요한 데이터 구성을 사용자가 하나의 프로그램에서 수행할 수 있도록 구현했다. 사용자가 영상 데이터를 실시간으로 수집할 수 있고, 수집한 영상을 버튼 클릭만으로 객체에 대한 Annotation 정보를 획득하고, 다양한 딥러닝 네트워크에 학습할 수 있는 레이블링 파일 형식으로 저장했다. 그러므로 사용자는 학습데이터를 구성하는데 필요한 시간과 비용을 줄일 수 있다. 기존의 Annotation 프로그램이 발전해야하는 모델로서 방안을 제시했으며. 구현한 프로그램을 활용하여 110종의 차량에 대한 약 9만장의 데이터셋을 구축했다. 또한 구성한 데이터셋을 YOLO와 Retinanet에 학습하고 차량을 구별하는 정확성을 평가했다. 차량 진입 영상과 IR 영상에 대한 정확도 평가를 통해 구현한 프로그램이 대량의 정확한 데이터셋을 구성하는 것을 확인하였다.

참 고 문 헌 (References)

[1] Tsung-Yi Lin, Microsoft COCO: Common Objects in Context, Computer Vision and Pattern Recognition (cs.CV), Submitted on 1 May 2014 (v1) last revised 21 Feb 2015 (this version, v3), arXiv:1405.0312 [cs.CV]

[2] Mark Everingham, Luc Van Gool, The PASCAL Visual Object Classes (VOC) Challenge, Int J Comput Vis (2010), Received: 30 July 2008 / Accepted: 16 July 2009 / Published online: 9 September 2009.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi "You Only Look Once: Unified, Real," Computer Vision and Pattern Recognition 2016, pp. 779-788, June 2016, arXiv:1506.02640.

[4] Tsung-Yi Lin, Focal Loss for Dense Object Detection, Computer Vision and Pattern Recognition (cs.CV), Submitted on 7 Aug 2017 (v1), last revised 7 Feb.

[5] Mr. Pankaj S. Parsania, A Comparative Analysis of Image Interpolation Algorithms, International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 1, January 2016.

[6] Karen Simonyan, VERY DEEP CONVOLUTIONAL NETWORK SFORLARGE-SCALE IMAGE RECOGNITION, Published as a conference paper at ICLR 2015.

[7] Shaoqing Ren, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Computer Vision and Pattern Recognition, Submitted on 4 Jun 2015 (v1), last revised 6 Jan 2016.

[8] Joseph Redmon, YOLO9000: Better, Faster, Stronger, Computer Vision and Pattern Recognition, Submitted on 25 Dec 2016, arXiv:1612.08242.

[9] 3D Object Representations for Fine-Grained Categorization, Kaggle, Stanford Cars Dataset. Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei, 4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.

[10] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7132 - 7141, 2018.

[11] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2019.

저 자 소 개

임 송 원

- 2018년 : 청운대학교 방송영상학과 학사
- 2020년 : 서울과학기술대학교 일반대학원 미디어IT공학과 석사
- ORCID : https://orcid.org/0000-0002-0944-4527
- 주관심분야 : 컴퓨터 비전, 영상 분석

박 구 만

- 1984년 : 한국항공대학교 전자공학과 공학사
- 1986년 : 연세대학교 전자공학과 석사
- 1991년 : 연세대학교 전자공학과 박사
- 1991년 ~ 1996년 : 삼성전자 신호처리연구소 선임연구원
- 1996년 ~ 1999년 : 호남대학교 전자공학과 조교수
- 1999년 ~ 현재 : 서울과학기술대학교 전자IT미디어공학과(나노IT디자인융합대학원 정보통신미디어공학전공 파견) 교수
- 2006년 1월 ~ 2007년 8월 : Georgia Institute of Technology Dept.of Electrical and Computer Engineering, Visiting Scholar
- ORCID : https://orcid.org/0000-0002-7055-5568
- 주관심분야 : 컴퓨터 비전, 멀티미디어통신