Journal of Knowledge Information Technology and Systems(JKITS), Vol. 16, No. 1, pp. 9-19, February 2021
https://doi.org/10.34163/jkits.2021.16.1.002

Journal of Knowledge Information Technology and Systems

ISSN 1975-7700 (Print), ISSN 2734-0570 (Online)

http://www.kkits.or.kr

A Machine Learning System for Crosswalk Detection

Ju-Seok Park[1], **Gyesik Lee**[*2]

[1]*Department of Computer Science and Engineering, Hankyong National University*
[2]*School of Computer Engineering & Applied Mathematics, Hankyong National University*

ABSTRACT

There are some places where it is difficult for the blind to judge because of different types of braille blocks, mainly caused by broken braille blocks or the reinstallation of braille blocks due to rapid road revisions. Therefore braille blocks installed on crosswalks often cause accidents for the blind. To prevent this, the crosswalk has a walking signal voice guidance system and auxiliary devices. There have been various attempts to install such systems to prevent accidents. But there are many uninstalled places, or installed devices often don't work properly, making it difficult to use them in practice. We would like to present a crosswalk detection algorithm using Deep Learning. If the algorithm is mounted on a small device, it can be attached to the user's body to prevent unexpected situations, unlike systems that need to be installed on all crosswalks. It is economical and can effectively prevent accidents by alerting the user. The crosswalk detection algorithm outputs sound when it detects a crosswalk, indicating that there is a crosswalk ahead of the user. To this end, the object detection algorithm YOLOv3 and the newral network framework were used. In this project, the model we trained learned from images of crosswalks taken directly and showed an average recognition rate of 92% or higher.

KEYWORDS: Machine learning, Crosswalk, Object detection, , Darknet, OpenCV

ARTICLE INFO: Received 4 December 2020, Revised 2 January 2021, Accepted 9 February 2021.

[*]Corresponding author is with the School of Computer Engineering & Applied Mathematics, Hankyong National University, 327 Jungang-ro, Anseong-si, Gyeonggi-do, 17579, KOR EA.

E-mail address: gslee@hknu.ac.kr

1. 서 론

횡단보도는 접근하는 차량이 정지하여 있는 동안 보행자가 안전하게 도로를 횡단할 수 있도록 하기 위한 교통안전시설물이다. 그러나 횡단보도에서 차량 대 보행자 사고는 빈번히 발생하고 타 교통사고 유형보다 사망자 비율이 높다. 보행자 교통사고 유형을 2009년부터 2019년까지 살펴보면, 총 보행자 교통사고 545,094명 중 차대보행자 사고가 139,980명, 횡단 중 사고는 61,737명으로 전체 보행자 사고 중 8.8%를 차지하며, 횡단 중 사망자 수는 2,414명으로 전체 보행자 사고 사망자 수 20,155명의 8.4%를 차지하는 것으로 나타났다[1-2].

일반 보행자의 사망률이 보행자 사고율에서 높은 비율을 차지하는 현황에서, 시각장애인의 점자블록 보도 만족도는 교통약자들의 보도 만족도 조사 결과 중 가장 낮은 34점이다[3]. 교통약자의 이동편의 증진법에 따라 위험 장소 및 변화지점에 횡단보도 진입 부분에는 점형 블록, 진행 방향에는 선형 블록을 설치해야 한다. 하지만 제대로 관리되지 않은 점자블록은 안전하게 유도하지 못하고 오히려 시각장애인을 위험한 곳으로 내몰고 있다.

본 연구에서는 횡단보도에서 시각장애인의 안전을 확보하기 위한 횡단보도 탐지알고리즘을 딥러닝 알고리즘 중에서 영역기반 합성곱 신경망(Region-based convolutional neural network, R-CNN)을 기반으로 한 YOLOv3 알고리즘을 이용하여 개발하였다. 평균 정확도 90% 이상과 향후 개발될 시각장애인 보조기기 개발에 도움이 되는 핵심 기술로써 개발하는 것이 목표이다.

본 논문의 구성은 다음과 같다. 2절과 3절에서는 횡단보도 탐지알고리즘을 개발하기 위한 시스템 개발환경과 구현과정을 다룬다. 4절은 개발된 알고리즘의 성능을 검증하며, 5절 결론으로 논문을 마무리한다.

2. 시스템 개발환경

본 연구에서는 횡단보도 이미지를 인공신경망에 제공하여 학습시키고 시각장애인에게 횡단보도의 위치를 제공함으로써 점자블록의 오설치로 인한 교통사고 예방시스템 개발을 목표로 한다. 시스템 구현을 위한 시스템 환경은 다음과 같다.

표 1. 학습 시스템 환경
Table 1. Learning System Environment

구분	내용
CPU	i7-8700k
GPU	RTX 2080Super
RAM	32GB
OS	Ubuntu 18.04
프로그래밍 언어	Python 3, C
모델	YOLOv3
인공신경망	Darknet
라이브러리	OpenCV 3.4.0, cuDNN 7.5.0.56
Toolkit	CUDA 10.1.105

본 연구에서는 시스템에 YOLO 모델 중 가장 최신 모델은 아니지만 많은 검증으로 최적화가 잘 되어 있는 YOLOv3 모델을 사용하였다. 횡단보도 탐지알고리즘은 사용자가 걸어다니며 실시간으로 탐지해야 하기에 실시간에 가까운 처리 속도가 필요하다. YOLO 모델은 다양한 R-CNN 알고리즘 중에서 뛰어난 처리 속도와 성능을 보여준다[4-6].

<그림 1>은 YOLOv3가 객체를 검출하는 과정을 나타낸 것이다. YOLOv3는 입력 이미지에 대해 격자 모양의 SxS 그리드 사각형으로 영역을 구분하고 한 개의 사각형 당 총 B개의 영역을 지정한 후, 각 영역에 대하여 클래스의 확률을 예측한다[7].

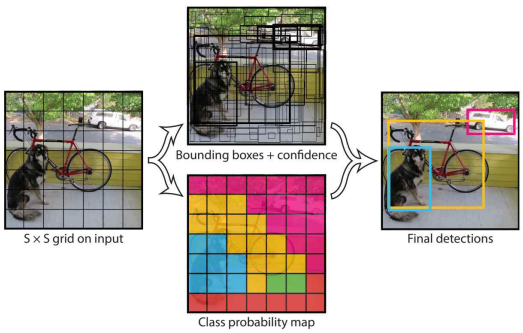

그림 1. 객체 검출 방법
Figure 1. object detection method

각 boundary box는 객체의 위치, 크기, box confidence score로 총 5개의 인자로 구성되어 있다. 여기서 box confidence score는 식 (1)을 가진다. 식 (1)은 box가 객체를 포함하고 있을 가능성과 boundary box가 얼마나 정확한지를 보여준다. 식 (2)는 탐지된 객체가 특정 클래스에 포함될 확률이다. 최종 학습된 모델은 지정한 임계치보다 높은 confidence score를 가지는 영역에 대해 조건부 확률이 제일 높은 클래스로 분류된다. YOLO 모델에 가중치(weight)가 학습되는 방식은 오차함수를 나타내는 식 (3)으로 설명할 수 있다. 식 (3)에서 (x,y)는 예측 영역의 중심 좌표를 의미하고, (w,h)는 예측 영역의 넓이와 높이를 의미한다. $l_{ij}^{obj}, l_{ij}^{noobj}, l_{i}^{obj}$에서 obj, noobj은 객체가 존재함과 존재하지 않음을 의미하고 i, j는 각각 그리드 사각형과 예측 영역을 의미한다. C_i는 confidence score를, λ_{coord}, λ_{noobj}는 loss의 균형을 맞추기 위한 balancing 파라미터로 각각 balance rate coord, balance rate nocoord를 나타낸다. $P_i(c)$는 그리드 사각형 i에 대한 클래스 c의 조건부 확률을 의미한다. 오차함수를 통해 YOLO를 검출하고자 하는 객체의 실제 영역과 예측 영역 간의 중심 좌표와 넓이, 높이의 차가 최소가 되도록 하며 객체가 존재하는 영역과 존재하지 않는 영역의 confidence score 차가 최소가 되도록 가중치를 학습한다[4, 7-9].

$$\text{Confidence Score} : P_r(object)\cdot IoU \tag{1}$$

$$\text{Confidence Class Probability} : P_r(class_i|object) \tag{2}$$

$$\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B} l_{ij}^{obj}[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2] \\
&+\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B} l_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w_i}})^2+(\sqrt{h_i}-\sqrt{\hat{h_i}})^2] \\
&+\sum_{i=0}^{s^2}\sum_{j=0}^{B} l_{ij}^{obj}(C_i-\hat{C_i})^2 \\
&+\lambda_{noobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B} l_{ij}^{noobj}(C_i-\hat{C_i})^2 \\
&+\sum_{i=0}^{s^2} l_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{P_i}(c))^2
\end{aligned} \tag{3}$$

YOLOv3에서 사용된 Network는 Darknet-53이다. Dakrnet-53은 YOLOv2에서 사용되는 Darknet-19 아키텍처에 ResNet에서 제안된 skip connection 개념을 적용하여 레이어를 훨씬 더 많이 쌓는 방식으로 개선한 Network이다. <표 2>을 보면 3x3 convolution과 1x1 convolution으로 이루어진 block을 연속해서 쌓는다. 그 후 MaxPooling 대신 convolution의 stride를 2로 취해 피쳐맵의 해상도를 줄여나간다. 또한 skip connection을 활용하여 Residual 값을 전달하고, 마지막 레이어에서 Average Pooling과 Fully Connected Layer를 통과한 뒤, Softmax를 거쳐 분류 결과를 출력한다. 아키텍처만 보면 기존의 ResNet과 큰 차이점은 없지만, <표 3>을 보면 ResNet-101과 ResNet-152를 비교했을 때 정확도에서 큰 차이는 없으나 FPS가 더 높다는 것을 강조하고 있다[10].

표 2. Darknet-53[11]
Table 2. Darknet-53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

표 3. Darknet 성능 비교[11]
Table 3. Darknet Performance Comparison

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

<표 3>의 각 네트워크는 동일한 설정으로 훈련되었으며 256x256의 단일 크롭 정확도로 TitanX 환경에서 측정하였다. Darknet-53은 ResNet-101 보다 성능이 좋고 속도가 1.5배 빠르며, ResNet-152와 성능이 비슷하고 속도가 2배 빠르다. 또한 Darknet-53은 초당 부동소수점 연산 수치가 가장 높게 측정되는데, 이는 Network 구조가 GPU를 더 잘 활용하고 속도가 더욱 빨라지는 것을 의미한다 [10].

딥러닝과 머신러닝 분야에서 빠른 학습을 위해 CPU가 아닌 GPU의 사용은 필수로 요구된다[12]. CUDA는 NVIDIA에서 개발한 GPU Develop Toolkit 이다. 기존 Single-Core로 MultiProcessing, Multi-Threading을 사용하여 Multi-core 연산이 가능했으나 딥러닝과 채굴 분야에서 연산 속도가 매우 느리다. 이에 개발자들이 주목한 것이 CUDA Core이다. 각 Core별 Clock은 CPU가 앞선다. RTX 2080Super와 동 세대로 출시된 Intel i7-10700k의 경우 1개 Core의 Max Clock은 5.10GHz에 이르며 8개의 코어를 가지고 있다. 반면, RTX 2080Super의 1개 Core의 Max Clock은 1.82GHz이며 CUDA Core 수는 3,072개이다. 직렬연산의 경우 Core당 Clock이 높은 CPU의 연산 속도가 더 빠르지만, 병렬연산의 경우는 Core의 수가 많은 CUDA Core가 압도적으로 빠르다. <그림 3>은 병렬도가 높은 데이터 요소 처리에 대한 GPU와 CPU의 처리 속도 차이를 나타낸다. 호환성 유지를 위해 아키텍처 개선이 제한된 CPU와는 다르게 GPU 아키텍처는 이러한 제한이 적기 때문에 성능이 비약적으로 발전했다 [13-14]. 따라서 많은 데이터를 이용하여 학습시키는 머신러닝 분야에서 병렬처리 성능이 높은 GPU를 사용함으로써 학습 소요시간 단축이 가능하다.

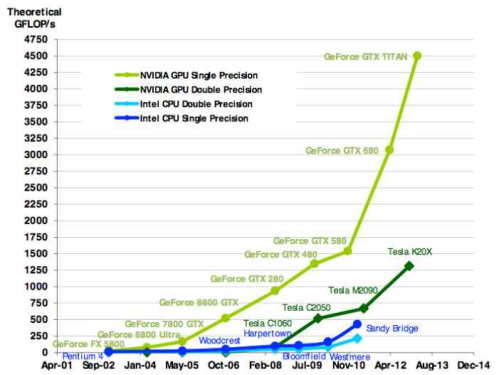

그림 2. CPU 및 GPU의 초당 부동 소수점 작업 수[15]
Figure 2. floating-point operations(ps) for CPU and GPU

cuDNN(CUDA Deep Neural Network)은 NVIDIA CUDA를 위한 GPU 가속화 라이브러리 기초 요소로, Convolution, Pooling, Nomalization, Activation과 같은 일반적인 루틴을 빠르게 이행할 수 있도록 하는 라이브러리이며 Caffe2, Keras, MATLAB, PyTorch, TensorFlow 등 널리 사용되는 딥러닝 프레임워크를 가속화하는 데에 사용된다[16]. 실제 시스템 구현 시 cuDNN을 적용하여 학습할 때 걸리는 시간과 그렇지 않은 경우는 약 10-20배의 속도 차이가 있었다. <그림 4>에서는 현재 공개되어있는 cuDNN 8.0과 cuDNN 7.6 버전에서의 속도차이를 나타낸다. 본 시스템에서는 그래픽카드 드라이버와의 호환성을 위해 이전 버전인 cuDNN 7.5.0.56 버전을 사용하였다.

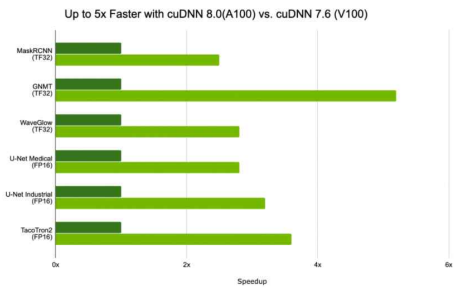

그림 3. cuDNN 두 버전의 속도차이[16]
Figure 3. speed difference between two versions of cuDNN

OpenCV(Open Source Computer Vision)는 공개 소프트웨어로 누구나 자유로운 수정, 변경, 재배포가 가능하고 위와 같은 영상처리 및 컴퓨터 비전에 사용할 수 있다. 이는 C/C++로 개발되었고, 윈도우즈(Windows), 리눅스(Linux), 맥(Mac) 등의 운영체제에서 사용이 가능하다. 응용프로그램의 작성 또한 호환성이 높아 C/C++, C#, Python, Matlab 등 타 언어와의 연동이 쉽다. OpenCV의 주요 목적 중 하나는 사용하기 쉬운 컴퓨터비전 기반구조를 제공함으로써 정교한 컴퓨터비전 응용프로그램을 쉽고 빠르게 만들 수 있도록 도와주는 것이다.

가중치를 CPU를 사용하여 학습하는 경우 일반적으로 매우 오랜 시간이 걸리므로 CUDA, cuDNN을 이용해 GPU 자원을 사용해 학습 시간을 단축시킨다.

3. 횡단보도 탐지를 위한 시스템 구현

구현된 횡단보도 탐지시스템의 작동원리는 <그림 2>와 같다. 카메라는 사람의 움직임에 따라 항시 촬영하고 있으며, 카메라에 횡단보도가 탐지되는 순간, 시스템은 비프음으로 사용자에게 경고성 알람을 울리는 구조이다.

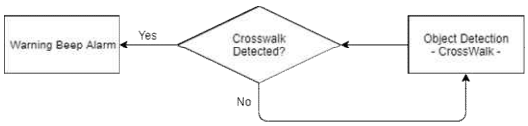

그림 4. 횡단보도 탐지 구조
Figure 4. crosswalk detection structure

다음으로 지도학습에 사용될 데이터 수집을 진행한다. 데이터는 경기도 평택시와 안성시에서 Galaxy S20+로 촬영한 횡단보도 이미지 2,500장과 촬영한 영상에서 프레임 단위로 추출한 이미지 2,500장으로부터 생성하였다. Yolo_mark에서 지도학습을 위한 라벨링을 실시하였으며, 학습 시 입력한 이미지의 크기는 608x608을 적용하였다. 시스템 구현 초기에 횡단보도를 바라보는 방향에 따라 세 개의 Class로 구분하여 학습시킬 예정이었기에 세 개의 Class로 구분하여 데이터를 수집한 후 학습을 진행하였다. 사용된 세 개의 Class는 <그림 5>와 같다.

그림 5. 횡단보도 방향에 따른 클래스 구분
Figure 5. class classification according to crosswalk direction

이후 세 개의 Class로 <표 1>의 환경에서 학습을 진행하여 최종 테스트를 하였을 때 최종 결과는 <표 4>와 같다. 약 20시간 정도를 학습시켜 검증 이미지로 평균 정확도는 80%를 나타냈다. 본 연구의 최종 목표는 평균 90% 이상의 정확도이기에 만족스럽지 못한 결과였다.

표 4. 3개의 클래스를 이용한 학습 결과
Table 4. Learning results with three classes

구분	내용
클래스	white_cross_f white_cross_n white_cross_d
이미지	5,000개
학습량	30,000회
손실률	0.2
학습 시간	24시간
평균 정확도	80%

평균정확도를 높이기 위해 학습 전 최적화를 실시하였다. 먼저 학습에 방해가 될 수 있는 요소가 포함된 이미지를 제거하여 5,000장에서 엄밀히 선정한 2,500장을 남기고 학습에서 배제하였다. 두 번째로 학습 시 입력되는 이미지의 크기를 416x416으로 변경하였다. 정확도는 조금 떨어지지만, 실시간 탐지를 기반으로 하는 횡단보도 탐지시스템에는 정확도보다 속도가 우선이라고 판단하여 입력 이미지의 크기를 감소시켰다[17]. 세 번째로 학습량을 증가시켰다. 이전 학습에서는 28,000회 이후로 손실률이 감소되지 않아 30,000회로 종료하였지만 위 두 가지를 변경한 후 학습을 진행하였을 때 약 77,000회까지 손실률이 감소되었다. 네 번째로 Class를 한가지로 통합했다. 사용자가 보는 방향에 따라 횡단보도의 모습을 세 개의 Class로 분류한 것과 분류하지 않고 한 개의 Class로 학습을 진행하였을 때 후자의 방법에서 정확도가 좀 더 높았다. 네 조건을 변경한 후의 학습 결과는 <표 5>와 같다.

표 5 최종 학습 결과
Table 5. final learning result

구분	내용
클래스	white_cross
이미지	2,500개
학습량	77,700회
손실률	0.02
학습시간	48시간
평균정확도	92%

<그림 6>에서는 제작된 횡단보도 탐지시스템의 모듈을 나타낸다. 실제 상용 기기에 적용한다고 가정하여 모듈 구성도를 작성했으며, 적용할 대표적인 두 기능은 부팅 시 자동실행 기능과 탐지알람 기능이다.

auto_start.sh은 셸 스크립트로 작성하여 기기 부팅 시 자동으로 파일을 실행하도록 되어 있으며, 횡단보도를 탐지하기 위한 촬영코드가 담긴 detect_start.py를 실행한다. detect_start.py는 Darknet과 YOLOv3를 사용하여 횡단보도를 탐지한다. 이때 인공신경망 Darknet에 cross_beep.pl 파일은 횡단보도가 탐지되는 순간 비프음을 출력하도

록 작성하였다. 개발의 편의를 위해 비프음을 사용했지만 사운드 출력장치나 진동 출력장치를 추가한 후 소스코드를 작성하여 사용자에게 다른 방법으로도 전달하는 것도 가능하다.

본 시스템의 카메라가 실시간으로 작동하며 횡단보도가 사용자의 시선에 보이는 순간만 알람을 출력한다. 그리고 최근 카메라를 이용하여 타인의 신체를 촬영하여 이를 보존, 배포하는 '몰래카메라' 또는 '디지털 성폭력' 범죄가 증가했다. 해당 범죄들은 피해자 입장에서 자신도 모르게 촬영당하며 촬영물이 인터넷상에서 확산되는 피해 또한 유발하기에 범죄 피해에 대한 두려움 수준이 높으며 일상생활에도 큰 피해를 입힌다. 이런 보안 문제에 대해 본 시스템은 촬영하는 영상을 따로 저장하지 않아 위와 같은 범죄를 사전예방이 가능하다[18].

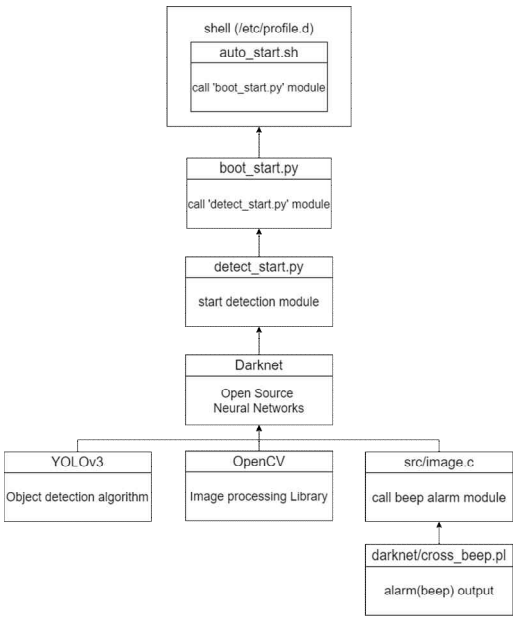

그림 6. 시스템 모듈 구성도
Figure 6. System Module Configuration Diagram

4. 검증

최종 검증을 위해 이미지, 영상, 웹캠 세 가지 방법의 탐지테스트를 진행하였다. 이미지는 초기에 설정한 사용자의 시선에 따른 횡단보도의 방향인 가로, 세로, 사선 방향의 이미지를 샘플로 테스트하였으며, 영상은 사용자가 걸어다니는 가정하에 직접 촬영한 영상과 횡단보도가 촬영된 뉴스 영상으로 테스트하였다. 웹캠 테스트는 PC 환경에서 이루어졌기 때문에 현장에서 촬영하는 것이 아닌 4K 이미지를 인쇄한 후 부착하여 촬영하여 사용하였다. 사용된 웹캠은 Logitech C920이며 FHD 화질, 평균 20-40프레임의 환경에서 촬영되었으며 사용자의 움직임을 가정하여 여러 방향에서 촬영하였다.

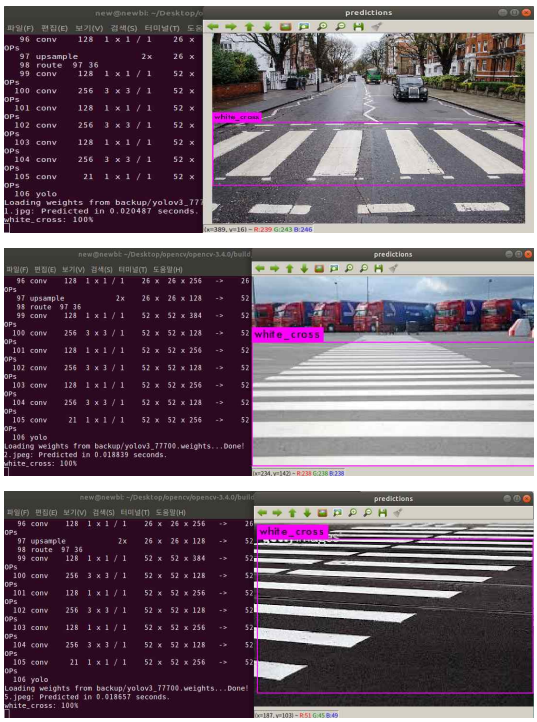

그림 7. 이미지에서의 탐지 테스트 결과
Figure 7. Detection test results from image

그림 8. 영상에서의 탐지 테스트 결과
Figure 8. Detection test results from Video

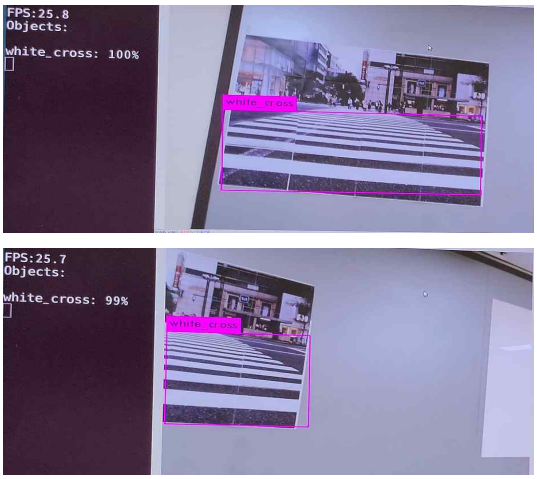

그림 9. 웹캠을 사용한 탐지 테스트 결과
Figure 9. Detection test results using webcam

결과적으로 학습을 위한 최적화 후의 정확도는 크게 증가하였다. <그림 7>을 보면 이미지에서 횡단보도 탐지는 평균 95-100%를 유지하였음을 확인할 수 있다. <그림 8>은 영상에서의 횡단보도 탐지는 평균 95%이상을 유지하였음을 보여준다. 그리고 <그림 9>에서 웹캠으로 촬영했을 때의 정확도가 평균 92% 이상을 유지함을 확인할 수 있으며, 이는 기대목표 이상의 정확도를 달성하였음을 의미한다.

검증과정에서 탐지 불가능한 현상이 몇 가지 발견되기도 하였다. 첫째, 횡단보도에 다른 개체가 있는 경우이다. <그림 8> 2번째 그림에서 택시가 지나가고 있는데, 부분적으로 겹치는 경우 탐지되는 경우가 70% 정도이며 경우에 따라 탐지되지 않을 수도 있다. 둘째, 이형 모양의 횡단보도이다. 지도학습을 위해 준비한 데이터셋의 횡단보도는 전부 흰색 바탕에 직사각형 모양의 횡단보도이다. 하지만 최근 들어 다양한 색이나 횡단보도의 시작이 직사각형이 아닌 경우가 많이 있으며, 이런 경우에는 아예 탐지되지 않거나 되더라도 정확도가 낮은 경우가 있다. YOLO는 버전에 따라 정확도의 차이가 있으므로 추후 이런 문제점을 해결하기 위해 YOLO 알고리즘의 상위 버전에 본 문제점을 해결하기 위한 변동사항이 있는 경우, 해당 버전을 사용하거나 이형 모양의 횡단보도의 클래스를 추가하여 연구가 진행되어야 한다[19-20].

5. 결 론

본 논문에서는 시각장애인의 횡단보도 보행 중 사고를 막기 위해 개발한 횡단보도 탐지시스템을 소개하였다. 시스템 개발환경에서는 객체탐지 알고리즘인 YOLOv3와 인공신경망인 Darknet을 활용하였으며, 학습모델의 빠른 훈련을 위해 NVIDIA GPU와 CUDA Core, cuDNN을 사용하였다. 시스템 테스트을 위해서 단순 이미지 테스트와 실제 사용자가 걸어다닐 때 보이는 횡단보도의 위치와 크기를 가정하여 영상 테스트와 웹캠 테스트를 실시하였다. 결과적으로 이미지에서는 대부분의 상황에서 98% 이상의 정확도를 보여주었고, 영상과 웹캠 테스트에서는 평균 30프레임 이상의 환경에서 92% 이상의 정확도를 보여줌으로써 최종 목표인 90% 이상

의 정확도를 가지는 횡단보도 탐지시스템을 구현하였다. 구현에 사용된 주요 소스코드는 깃허브에 업로드되어 있으므로, 깃허브 프로젝트 사이트에서 참고할 수 있다[21].

이미지 인식에 비해 웹캠을 이용한 인식률이 떨어지는 것으로 보아 데이터셋이 부족했다고 볼 수 있다. 또한 사용된 대부분의 데이터셋은 흰색바탕의 횡단보도이다. 최근 다양한 모양과 색깔의 횡단보도가 생겨나고 있으므로, 앞으로 원활한 탐지를 위해서는 데이터셋의 추가가 필수로 요구된다.

유사 식물 중, 생명에 위협적인 식물들의 이미지를 딥러닝으로 분류하는 연구[22] 등 머신러닝과 딥러닝 분야에서 위험을 방지하기 위한 개체 탐지에 관한 연구는 계속되고 있다. 이에 향후 개발 방향으로 횡단보도 뿐만 아니라 시각장애인의 사고를 유발할 수 있는 개체들을 탐지하여 안전을 보장하며 추후 상용기기 제작과정의 핵심기술로 사용될 수 있는 시스템을 개발하고자 한다.

References

[1] Traffic Accident Analysis system, http://taas.koroad.or.kr/, Nov. 4, 2020.

[2] J. W. Kim, M. Y. Jung, D. S. Kang, J. W. Hong, and S. B. Lee, *The setting in the range of traffic accident on the crosswalk*, The Korean Society of Safety, Vol. 26, No.4, pp. 120-126, 2011.

[3] S. M. Jo, *A study on the recognition of walking environment based on convolutional neural networks for the visually impaired*, Daegu University Graduate School of Rehabilitation Sciences, pp. 1-65, 2020.

[4] Y. H. Lee, and Y. S. Kim, *Comparison of CNN and YOLO for object detection*, Journal of the semiconductor & display technology, Vol.19, No.1, pp. 85-92, 2020.

[5] K. M. Lee, and C. H. In, *An optimal YOLOv3-tiny based object detection algorithm,* The Korea Institute of Intelligent Transport Systems, pp. 59-60, 2018.

[6] B. S. Kim, and J. Y. Kim, *A study on the YOLOv3 training data collection method for detecting of road speed sign*, Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 666-669, 2019.

[7] H. J. Lee, W. S. Lee, I. H. Choi, and C. K. Lee, *Detection model of fruit epidermal defects using YOLOv3: a case of peach*, Information systems review, Vol. 22, No. 1, pp. 113-124, 2020.

[8] D. G. Kim, *OpenCV programing*, kame publisher, 2010.

[9] J. Y. Seo, and M. B. Park, *A traffic light recognition and color detection based on YOLOv3*, Academic Conference and Exhibition of the Korean Society of Automotive Engineers, pp. 882-883, 2019.

[10] Review: ResNet - Winner of ILSVRC 2015, https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8, nov 4, 2020.

[11] J. Redmon, and A. Farhadi, *YOLOv3: An Incremental Improvement*, arXiv:1804.02767, 2018.

[12] S. M. Suh, *Effective implementation for fast deep learning algorithm*, Journal of Knowledge Information Technology and Systems(JKITS), Vol. 14, No. 5, pp. 553-561, 2019.

[13] T-H. Lee, and B-H. Hwang, J-H. Yun, and M-R. Choi, *A road region extraction using*

OpenCV CUDA to advance the processing speed, Journal of Digital Convergence, Vol. 12, No. 6, pp. 231-236, 2014.

[14] Y. M. Gwon, *High-speed processing of satellite image using GPU*, Department of Electronic Information and Communication Engineering, Chungnam National University, 2012.

[15] Suresoft, https://m.blog.naver.com/suresofttech/221151961585, Nov. 4, 2020.

[16] NVIDIA, https://developer.nvidia.com/cudnn, Nov. 4, 2020.

[17] S-B. Shim, and S-I. Choi, *Development on identification algorithm of risk situation around construction vehicle using YOLO-v3*, Journal of Korea Academia-Industrial cooperation Society, Vol. 20, No. 7, pp. 622-629, 2019.

[18] S. H. Shin, *Spatial features affecting the occurrence of hidden camera crime*, The Journal of Police Science, Vol. 19, No. 1, pp. 59-84, 2019.

[19] S. J. Lee, and G. M. Park, *Proposal for license plate recognition using synthetic data and vehicle type recognition system*, Journal of Broadcast Engineering, Vol. 25, No. 5, pp. 776-788, 2020.

[20] S-J. Bae, H-J. Choi, and G-M. Jeong, *YOLO model FPS enhancement method for determining human facial expression based on NVIDIA Jetson TX1*, Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 12, No.5, pp. 467-474, 2019.

[21] J. S. Park, detection_crosswalk, *https://github.com/NEEMYO0303/detection_crosswalk*, 2020.

[22] H. J. Shin, S. I. Lee, H. W. Jeoung, and J. W. Park, *Indoor plants image classification using deep learning and web application for providing information of plants,* Journal of Knowledge Information Technology and Systems(JKITS), Vol. 15, No. 2, pp. 167-175, 2020

횡단보도 탐지 머신러닝 시스템

박주석[1], 이계식[2]

[1] *한경대학교 컴퓨터공학과 학부생*

[2] *한경대학교 컴퓨터응용수학부 교수*

요 약

점자블록이 파손되고 급격한 도로개발로 인해 기존의 의도와 다르게 설치된 점자블록이 시각장애인의 생존을 위협하고 있다. 이를 방지하기 위해 횡단보도에 보행신호 음성안내시스템 등의 보조 기기를 갖추고 있지만, 미설치된 곳이 많고 제대로 작동하지 않아 실제적인 활용에 어려움이 발생하고 있다. 이에 잘못 설치된 점자블록으로 인해 발생하는 시각장애인의 교통사고를 방지하기 위해 머신러닝을 활용한 횡단보도 탐지시스템을 개발하여 사용자에게 위험상태를 알려주는 시스템을 제안한다. 소개하는 시스템은 YOLOv3, Darknet 등 심층신경망과 직접 촬영한 횡단보도 사진을 이용하여 학습되었으며, 92% 이상의 정확도 성능을 보인다.

감사의 글

본 논문은 2017학년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2017R1D1A1B05031658).

Ju-Seok Park is a student of the Department of Computer Science and Engineering at Hankyong National University. His interests include machine learning and artificial intelligence.

E-mail address: hears0303@gmail.com

Gyesik Lee received his bachelor's degree in the Department of Mathematics from Seoul National University. He received the M.S. degree and the Ph.D. degree in the Department of Mathematics and Computer Science from University of Münster in Germany. He had research positions at INRIA, AIST, and Seoul National University. He is a professor in the School of Computer Engineering & Applied Mathematics at Hankyong National University. His current research interests include machine learning, data science, and logic in computer science. He is a member of the KKITS.

E-mail address: gslee@hknu.ac.kr