KICSP 한국 융합신호처리학회

융합신호처리학회논문지 Vol. 23, No. 3 : 150~159 September. 2022
DOI : 10.23087/jkicsp.2022.23.3.006

헬멧 착용 여부 및 쓰러짐 사고 감지를 위한 AI 영상처리와 알람 시스템의 구현

조용화[1], 이혁재[1*]
[1]경남대학교 정보통신AI공학과

Implementation of an alarm system with AI image processing to detect whether a helmet is worn or not and a fall accident

Yong-Hwa Jo[1], Hyuek-Jae Lee[1*]
[1]Department of Information & Communication AI Engineering, Kyungnam University

요 약 본 논문은 실시간 영상 분석을 통해서 산업현장에서 활동하는 여러 근로자의 영상 객체를 추출해 내고, 추출된 이미지로 부터 개별 영상 분석을 통해 헬멧의 착용 여부와 낙상 사고 여부를 확인하는 방법을 구현한다. 근로자의 영상 객체를 탐지하기 위해서 딥러닝 기반 컴퓨터 비전 모델인 YOLO를 사용하였으며, 추출된 이미지를 이용하여 헬멧의 착용여부를 판단하기 위해 따로 5,000장의 다양한 헬멧 학습 데이터 이미지를 만들어서 사용하였다. 또한, 낙상사고 여부를 판단하기 위해서 Mediapipe의 Pose 실시간 신체추적 알고리즘을 사용하여 머리의 위치를 확인하고 움직이는 속도를 계산하여 쓰러짐 여부를 판단하였다. 결과에 신뢰성을 주기위한 방법으로 YOLO의 바운딩 박스의 크기를 구하여 객체의 자세를 유추하는 방법을 추가하고 구현하였다. 최종적으로 관리자에게 알림 서비스를 위하여 텔레그램 API Bot과 Firebase DB 서버를 구현하였다.

• 주제어 : 합성곱 신경망, YOLO, 객체 검출, 딥러닝, 산업현장

Abstract This paper presents an implementation of detecting whether a helmet is worn and there is a fall accident through individual image analysis in real-time from extracting the image objects of several workers active in the industrial field. In order to detect image objects of workers, YOLO, a deep learning-based computer vision model, was used, and for whether a helmet is worn or not, the extracted images with 5,000 different helmet learning data images were applied. For whether a fall accident occurred, the position of the head was checked using the Pose real-time body tracking algorithm of Mediapipe, and the movement speed was calculated to determine whether the person fell. In addition, to give reliability to the result of a falling accident, a method to infer the posture of an object by obtaining the size of YOLO's bounding box was proposed and implemented. Finally, Telegram API Bot and Firebase DB server were implemented for notification service to administrators.

• Key Words : Convolutional Neural Network (CNN), YOLO, Object detection, Deep learning, Industrial site

Received 29 August 2022, Revised 22 September 2022, Accepted 23 September 2022

* **Corresponding Author** Hyuek-Jae Lee, Department of Information & Communication AI Engineering, Kyungnam University, 7 Kyungnamdaehak-ro, Masanhappo-gu, Changwon-si, Gyeongsangnam-do 51767, Korea.
E-mail: hyuek@kyungnam.ac.kr

Ⅰ. 서론

산업현장은 외부환경과 위험 요소에 쉽게 노출되어 작업을 진행한다. 이러한 위험성 때문에 다양한 신체를 보호하는 개인 보호구를 의무적으로 착용하도록 규정하고 있다. 개인 보호구로는 헬멧, 안전화, 안전 장갑 등 여러 종류가 있다. 특히 건설 현장에서는 두부 보호를 위한 헬멧 착용 관리가 필수적으로 요구되며, 이에 대해 현장 안전 관리자는 안전교육과 함께 현장 순회 점검과 헬멧 착용 여부를 확인하고, 미 착용자에게 헬멧 착용을 지시하고 있다. 하지만 2020년 산업재해 현황 분석에 따르면 상해 부위 중 두부 손상으로 인한 경우가 전체의 42%를 차지한다[1]. 두부 손상의 대부분은 떨어지는 물건에 의한 사고 발생이 아닌 낙상에 의한 손상이 대부분이다. 이러한 통계의 이유로는 실제 현장에서 답답함, 번거로움 등의 이유로 헬멧을 미착용하거나 제대로 착용하지 않는 경우가 많고, 안전관리자의 교육과 점검을 통해 작업자에게 안전 장비의 중요성을 인식시키지만 현장에 있는 모든 근로자에 대한 실시간 관리는 실질적으로 어려운 실정이다. 이에 따라 헬멧 등 개인 보호구의 착용 상태를 보다 효율적으로 관리하는 등 개인 보호구의 착용 문화를 정착하기 위한 시스템적인 개선이 필요하다[2].

본 논문에서는 산업현장 근로자의 헬멧 착용 상태를 안전 관리자가 실시간 육안으로 확인하는 대신 카메라 등 영상 촬영 장치를 이용하여 실시간 알람으로 확인할 수 있는 딥러닝 기반의 컴퓨터 비전 모델의 응용 가능성을 알아보고자 한다.

Ⅱ. 전체 시스템 개요와 구조

산업현장에서 사용되는 헬멧은 흰색을 대부분 사용하지만, 경우에 따라서는 파랑 등 다른 색을 사용하는 경우가 있다. 일반적으로 형태와 내부는 동일하게 구성되어 있는 경우가 많다. 이를 토대로 학습 데이터는 3개의 헬멧을 사용하는데, 기본적인 흰색과 파란색, 디지털 문양의 헬멧을 사용하였다. 산업현장에서의 낙상사고는 떨어짐, 넘어짐 등으로 나뉜다. 본 논문에서는 낙상 사고의 예측을 목적으로 하는 것이 아니라 사고 사후를 빨리 확인해서 긴급 알람을 발생시키는 시스템을 구현하고자 한다. 낙상 사고에 대한 사고 유무를 확인하기 위해서는 먼저 객체(사람)를 탐지해야 하는데 이는 헬멧을 확인하기 위해 사용되는 YOLO를 이용하고자 한다. 추출된 이미지는 Mediapipe의 Pose 실시간 신체 추적 알고리즘을 통해서 자세를 확인할 수 있다. 전체적인 자세 구조를 신체 포인트별로 알려주는데, 머리에 대한 포인터(머리 부분은 11개의 포인트로 구성)의 평균을 구하고, 프레임별로 슬라이딩윈도우 평균을 지속적으로 측정하게 되면 머리의 움직임 속도를 구할 수 있게 된다. 그렇게 구해진 머리의 속도가 특정 낙하 속도에 도달하게 된다면, 낙상사고로 판단한다. 최종적으로는 YOLO 바운딩 박스 크기의 가로, 세로 비를 통해 객체가 쓰러졌는지의 여부를 검증하는 방법을 제안한다. 얻어진 결과는 스마트 폰을 통해 안전 관리자에게 실시간 보고 되도록 SNS 알람 시스템을 구현 다.

2.1 산업현장에서의 헬멧 착용 탐지

딥러닝 기반 컴퓨터 비전은 인공지능의 한 분야로서 카메라와 같은 영상 장치로부터 얻은 이미지 데이터를 이용하여 특정 객체를 자동으로 찾거나 추적하는 기술이 대표적으로 활용된다. 그중에서 실시간으로 객체를 탐지하는 분야에 있어서 뛰어난 모델인 YOLO를 사용하여 본 논문에서 제안하는 시스템을 구현하였다. 객체 탐지 분야에서 CNN(Convolutional Neural Network)을 기반으로 하는 모델 중 R-CNN (Regions with CNN features)이 주목을 받고 있는데, R-CNN, Fast R-CNN, Faster R-CNN [3] 순으로 대략 발전하였다. 처음 R-CNN은 객체 탐지 속도가 매우 느렸으며, Fast R-CNN도 ~2 FPS(Frame Per Second)의 성능을 보였는데, 실시간 구동을 위해서 필요한 >30 FPS 성능에는 많이 못 미쳤다. Faster R-CNN가 최대 ~7 FPS를 보여주는 시점에서 YOLO가 정확도는 약간 떨어지지만 45 FPS를 보여주면서 단번에 실시간 객체 탐지 분야에서 높은 순위를 차지했다[4]. 이후에 SSD(Single Shot MultiBox Detector)가 제안되었고, 최근에는 YOLOv4, v5가 잇달아 발표되면서 객체 탐지 분야에 절대 강자로 자리 잡고 있다.

YOLOv4 [5]는 2020년 4월, YOLOv5 [6]는 2020년 6월에 발표되었는데, 개발자는 서로 다르고 성능은 비슷하다. 본 논문에서는 v5 버전을 사용하였다[6,7]. v5

는 v4에 비해 낮은 용량과 Darknet이 아닌 PyTorch로 구현되어 있기 때문에 보다 쉽게 환경 구성하고 구현할 수 있다는 장점을 갖는다.

2.2 산업현장에서의 쓰러짐 탐지

Mediapipe는 구글에서 제공하는 AI 프레임워크로서, 비디오 형식 데이터를 이용한 다양한 비전 AI 기능을 파이프라인 형태로 손쉽게 사용할 수 있도록 제공된다[8]. AI 모델개발 및 수많은 데이터셋을 이용한 학습도 마친 상태로 제공되므로 라이브러리를 호출하여 사용하기만 하면 된다. 기본적인 얼굴인식 이외에도 Pose, Hands, HairSegmentation 등 다양한 컴퓨터 비전 AI 기능들을 제공하며, 다양한 개발환경 및 언어를 지원하고 있다(Android, iOS, C++, Python, JS, Coral). 또한 오픈소스 프로젝트로서 소스가 공개되어 있기 때문에 원하는 부분을 수정하여 추가로 개발할 수도 있으며, 최근 요가 자세 교정에 많이 응용되고 있다[9]. 솔루션별로 상세한 기술자료 및 예제 등이 풍부하게 제공되고 있으며, 학습모델을 범위나 용도별에 따라 구분하여 사용할 수 있도록 Lite, Full, Heavy 버전 등으로 구분하여 제공되기 때문에 각자 환경이나 목적에 따라 적정한 모델을 골라 사용하기만 하면 된다. 본 논문에서는 그림 1과 같이 처음에 YOLOv5를 가지고 사람 객체를 추출한 다음에 Mediapipe의 Pose를 이용하여 실시간 신체 골격 33개의 키 포인트를 추출하여 쓰러짐의 여부를 판단하려고 한다. 판단 여부는 33개의 키 포인트 중 얼굴 부위 11개 포인트의 평균 움직임 속도와 YOLOv5의 바운딩 박스의 가로와 세로 넓이 비율로 판단한다.

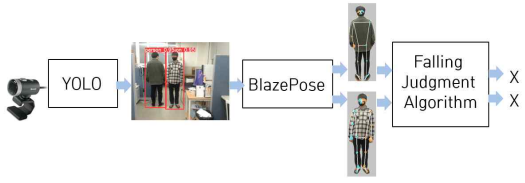


Fig. 1. Structure of a fall accident detector using YOLO object detection and Mediapipe Pose

2.3 실시간 알람과 DB 서버 구현

산업현장에서의 사고는 보통 안일함에서 일어난다. 헬멧은 근로자에게는 불편함을 느끼게 하고, 잠깐이라도 벗어두려고 한다. 그러한 안일함으로 인한 인명사고는 높은 비중을 차지한다. 그러한 문제를 없애기 위해 관리자는 직접 근로자를 확인하거나 안전교육을 따로 실시하기도 한다. 하지만 직접적으로 관리자가 확인하며 다니기에는 실질적으로 힘들며, 교육은 일시적인 효과만 보이기 일쑤인데, 그러한 문제를 본 논문에서 해결하고자 한다.

텔레그램 API Bot을 통해 실시간 SNS 전송을 구현하고[10], 현장 상황 데이터를 DB 서버에 저장하여 추후에 어디서든 열람할 수 있는 시스템을 구현한다. 동작 절차는 다음과 같다. 카메라 또는 영상 매체를 통해서 헬멧을 쓰지 않은 작업자 또는 쓰러져 있는 작업자를 발견하게 되면, 현장 시스템의 FTP 서버에 이미지를 올리고, TCP 소켓을 통해서 상황 정보를 관리자에 전송한다. 이미지와 상황 정보는 텔레그램을 통해서 알림 메시지로 전송과 동시에 파이어베이스 DB[11] 서버에 저장된다. DB에 저장된 데이터는 안드로이드 스마트 폰 앱을 통해 검색할 수 있는 App 서비스를 구현한다. 그림 2에 실시간 알람과 DB 서버에 대한 데이터 전달 흐름도를 보였다.

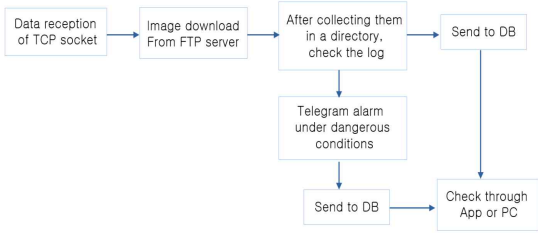


Fig. 2. Flow chart of real-time alarm and data delivery to DB server

III. 제안 시스템의 구현

3.1 헬멧 착용 유무의 판단

3.1.1 헬멧 데이터 수집

특정 객체 검출을 진행하기 위해서는 학습시키고자 하는 이미지 데이터셋을 만들어서 학습을 진행해야 한다. COCO, Mars, Market1501 등과 같은 데이터셋은 구글링으로 쉽게 찾아볼 수 있지만, 헬멧의 착용 여부를 확인해야 하는 시스템이므로 직접 이미지 데이터셋을 만들어야 한다. 이미지 데이터셋을 만드는 방법은 학습시키고자 하는 객체가 포함되어 있는 이미지를 모으

고, 해당 객체를 찾을 수 있도록 라벨링을 진행하면 된다. 라벨링 파일은 객체의 위치와 객체의 클래스 번호가 입력되어 있는 txt 파일을 뜻한다. 데이터를 수집하기 위해서 영상 촬영을 진행하였고, 영상은 총 3가지 종류를 준비하였다. 영상은 한사람이 3가지의 헬멧을 착용하는 모습을 각각 촬영하였다. 해당 영상을 프레임별로 나눠서, 총 5,000개, 즉 학습용 3,500장, 검증용 1,500장의 학습 데이터를 만들었다. 라벨링은 YoloLabel이라는 소프트웨어를 이용하여 라벨링을 진행하였다[12]. 각 라벨에 해당하는 부분을 체크하고 넘어가면 자동으로 좌표 내용을 txt 파일로 저장시켜준다. 그림 3과 같이 Helmet(연두색)은 헬멧을 착용, NoHelmet(초록색)은 헬멧을 착용하지 않은 사람으로 라벨링 하여 이미지 데이터셋을 만들었다.

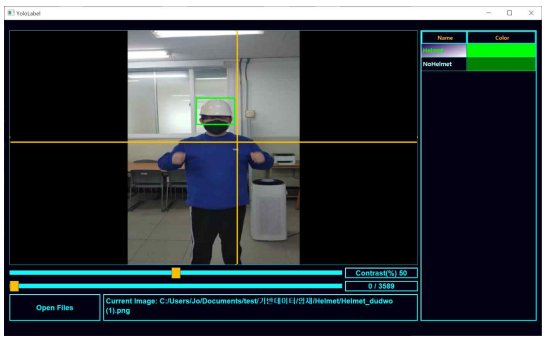


Fig. 3. Labeling of a training image dataset

라벨링은 다음과 같은 기준으로 수행하였다. 첫째, 얼굴 전체가 아닌 귀를 경계 부분으로 한정, 둘째, 중복된 데이터는 되도록 삭제, 셋째, 흐릿한 이미지의 경우 삭제, 마지막으로 사람 머리에 착용하고 있는 헬멧에 대해서만 라벨링을 수행하였다.

3.1.2 YOLO 학습 데이터 평가

학습용 이미지 데이터셋을 7:3으로 나눠서 학습용과 검증용으로 사용한다. 이렇게 나누는 이유는 YOLO에서 학습을 진행하면서 스스로 검증을 진행하기 때문이다. YOLO는 손실을 계산하기 위해서 예측값과 실제값 사이의 sum-squared error를 사용하며 이는 box_loss, obj_loss, cls_loss(classification loss)의 세 가지 요소로 구성되어 있다. box_loss는 최종 예측에 포함된 경계박스를 찾아내어 x, y 좌표 w, h 값에 대한 예측값과 실제값의 차를 구해 sum-squared error를 계산하여 구한다. obj_loss는 객체를 찾아야 했는데 못 찾은 인덱스에 대해 신뢰도 값의 차를 구해 손실 값에 더해서 구한다. cls_loss는 모든 객체가 있다고 판단된 인덱스들에 대해 모든 클래스의 예측값과 실제값의 차이를 구한 뒤 손실에 더해서 구한다.

Precision과 Recall의 정의는 그림 4와 같다. 합성곱 신경망을 사용한 객체 검출 모델에서의 정확도는 mAP(mean Average Precision)로 측정된다. mAP는 경계 상자가 레이블에 얼마나 잘 맞는지와 경계 상자에 대한 예측된 클래스가 얼마나 정확한지를 나타내는 지표이다.

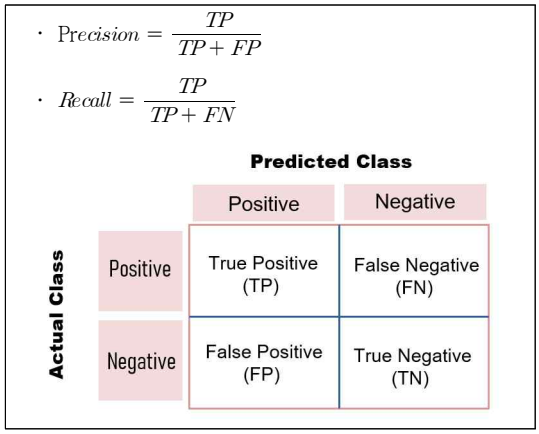


Fig. 4. Definition of Precision and Recall

3.1.3 학습 진행

전체 시스템 구현은 intel Core i9 10920X@ 3.50GHz, 64GB RAM, Nvidia GeForce RTX 3090 컴퓨터 환경에서 수행되었다. 학습의 전반적인 진행은 YOLO에서 제공하는 tarin.py를 통해서 진행된다. 학습에 필요한 학습 이미지 데이터가 있는 경로를 yaml 파일에 기록하여 저장해야 하며, 표 1은 학습에 사용한 yaml 파일 내용이다.

Table 1. Content of a yaml file for training

```
# Classes
nc: 2  # number of classes
names: ['Helmet', 'NoHelmet']  # class names

train: yolo_custom_train₩Train
val: yolo_custom_train₩Val
```

클래스의 수와 클래스 이름을 지정할 수 있으며, train 경로와 val 경로를 입력해주면 yaml 수정은 끝이다. 그 후 사전 학습 데이터 모델 중 하나를 골라야 하는데, 전이 학습을 통해서 평균적인 mAP를 높이기

위해서다. 모델은 YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x 순으로 크기가 커지며, 크기가 크면 클수록 복잡해지고 정확성이 높아지는 대신, 시간이 오래 걸리고 GPU의 메모리를 많이 차지한다. 학습이 종료되면 그림 5와 같은 학습 결과가 출력된다. 각 데이터의 의미는 그림 6의 박스에 정리해 두었다. 학습 데이터 및 검증 데이터에 대한 학습이 잘 이루어졌음을 그림 5 결과를 통해서 알 수 있다.

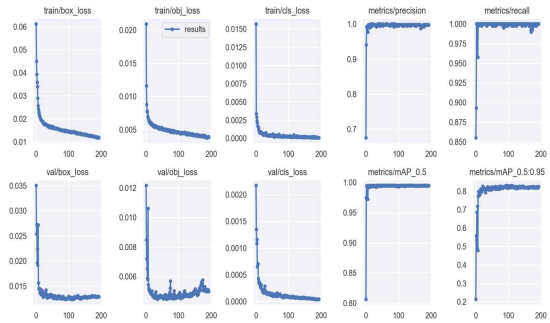


Fig. 5. Training results of the YOLO model on the helmet dataset

- 검증 정확도는 mAP로 나타내고, 훈련 및 검증 손실은 "train" 및 "val" 탭 아래에 있음. mAP는 검증 데이터 세트에 대해서만 표시됨
- train/box_loss : 학습데이터에 대한 상자손실
- train/obj_loss : 학습데이터에 대한 객체손실
- train/cls_loss : 학습데이터에 대한 클래스손실
- val/box_loss : 검증데이터에 대한 상자손실
- val/obj_loss : 검증데이터에 대한 객체손실
- val/cls_loss : 검증데이터에 대한 클래스손실
- metrics/precision
- metrics/recall
- metrics/mAP_0.5
- metrics/mAP_0.5:0.95 : IoU(Intersection over Union, 교집합 넓이/합집합 넓이 → 정답 바운딩 박스와 예측 바운딩 박스 사이에 계산)를 0.5, 0.55, 0.60 등 0.95까지 변화하면서 얻은 mAP 평균값

Fig. 6. Meaning of each data in Fig. 5

3.1.4 학습 결과 테스트

YOLO에서는 모델에 대한 mAP 계산을 도와주는 툴이 존재한다. 해당 test 툴을 이용하여 계산된 학습 모델의 결과물을 표 2에 정리 하였으며, 검증용 1,500장 중 580장의 이미지가 사용되었다.

Table 2. Performance outputs of YOLO training model

Class	Images	Labels	P	R	mAP @.5	mAP @.5:.95
all	580	527	0.895	0.749	0.821	0.557
Helmet	580	456	0.975	0.991	0.993	0.82
NoHelmet	580	71	0.814	0.507	0.65	0.293

이미지 580개(검증 데이터)를 이용하여 테스트를 진행하였고, 각 class별로 전체 이미지 수에 대한 Labels와 Precision(P, 정밀도)와 Recall(R, 재현율)이 나온다. Helmet의 mAP@.5는 0.993, NoHelmet의 mAP@.5는 0.65로 많은 차이를 보여준다. 실제 성능에서 NoHelmet 부분이 미흡한 것을 확인했다. 배치 사이즈는 16, YOLOv5x를 사용했고, Epoch는 299까지 학습시켰다. 학습데이터에 대한 인식 테스트 결과를 그림 7에 나타냈고, 사진에서 알 수 있듯이 헬멧을 쓴 경우와 안 쓴 경우를 확실히 구분한다. 간혹 헬멧 착용 미착용 머리 자체를 인식하지 못하는 경우는 있지만, 오인식 되는 경우는 한 번도 없는 것을 확인했다. 인식 threshold를 0.8로 지정한 경우인데, 0.6 정도로 낮추면 조금씩 오인식 확률이 올라간다.

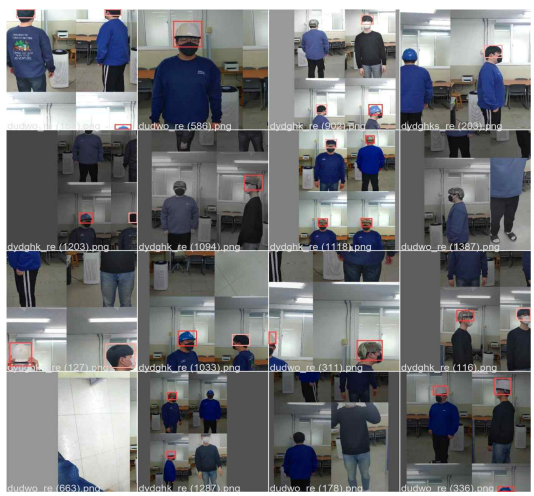


Fig. 7. Test results of the trained YOLO model

Fig. 8. Example of confusion when two labels are detected at the same time in the case of a woman with hair covering her ears

헬멧을 착용한 경우에 대해서는 인식률이 매우 높아 오인식되는 경우가 없었다. 그러나, 오인식되는 경우를 분석해 보았는데, 그림 8과 같이 여성의 경우, 귀 부분을 머리카락이 가리고 있는 경우 두 개의 라벨이 동시에 뜨며, 혼동을 유발한다. 원인은 라벨링 기준을 귀를 기준으로 잡았기에 일어난 문제이다. 두 명의 인물을 동시에 잡았을 때 인식 정확도(%)가 한 사람일 때 보다 약간 떨어지는 경우가 생기고, 여성 인물이 있을 경우에 정확도가 확연하게 내려가는 것을 확인했다. 문제 해결을 위해 학습 데이터를 목 부분까지 늘려서 추가로 진행하였고, 2~3명의 사람이 들어가 있는 사진을 1,000장 추가하였다. Background Image는 앞에서 언급한 False Positive(FP)를 줄여주는 효과를 준다. 새로운 학습 데이터를 이용하여 다시금 학습을 진행하였다. 학습에 대한 결과는 표 3과 같으며, mAP@.5에서 헬멧을 쓴 경우, 안 쓴 경우 각각 0.995와 0.977 높은 정확도를 보여주는 것을 알 수 있다.

Table 5. Performance outputs of the modified training image dataset

Class	Images	Labels	P	R	mAP @.5	mAP @.5:.95
all	580	527	0.992	0.978	0.986	0.757
Helmet	580	456	0.998	0.998	0.995	0.825
NoHelmet	580	71	0.986	0.958	0.977	0.689

해당 학습 모델을 토대로 전혀 학습에 참여하지 않은 사람에 대해서도 그림 9와 같이 잘 구분하는 것을 확인할 수 있었다.

Fig. 9. Test results of a person not included in the training dataset

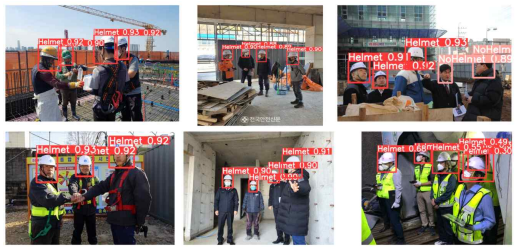
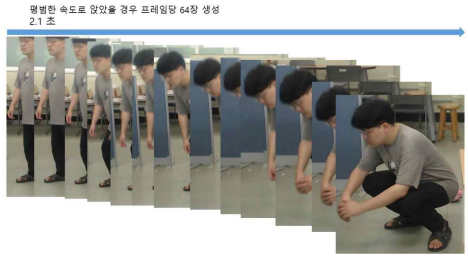


Fig. 10. Test results of photos in industrial sites

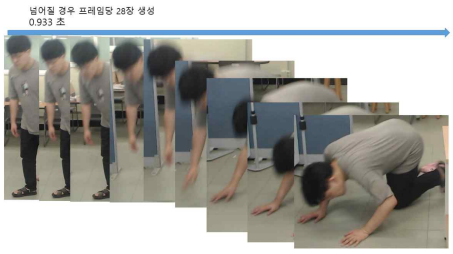
일반적인 산업현장에서의 상황을 테스트하기 위해 그림 10과 같은 사진을 대상으로 수행해 본 결과, 매우 잘 동작함을 알 수 있다.

3.2 쓰러짐 유무의 판단

Mediapipe Pose[8]를 이용하여 실시간 신체 골격 33개의 키 포인트 중 0~10번 머리 부분의 포인트들을 이용하여 쓰러짐 여부를 판단한다. 사람의 머리 부분 키 포인트 11개를 평균하여 중심값을 찾고, 그 중심값에 대해 10프레임의 슬라이딩 윈도우를 만들어 시간 평균값을 매번 산출하도록 하였다. 슬라이딩 윈도우의 시간 평균 중심값을 지속적으로 기록하면서 일정 구간에서 y 방향으로 위치의 급격한 변화가 생기는 것을 관찰하는 것으로 낙상 발생 유무를 확인하고자 한다.



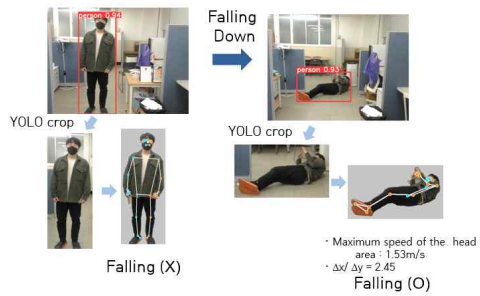
(a)



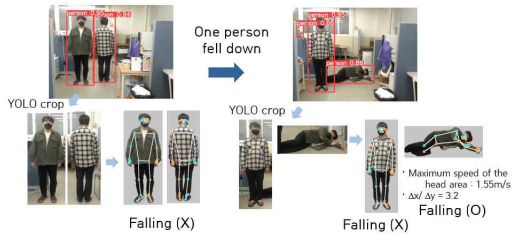
(b)

Fig. 11. Frame rate measurement of (a) normally sitting (b) falling

그림 11(a)와 같이 평범하게 앉았을 경우 프레임당 64장 생성이 되어 전체 프레임 30이니 2.1초 정도 걸린다고 볼 수 있다. 반면 빠르게 넘어질 경우 그림 11(b)와 같이 프레임당 28장을 생성하여 0.933 초 걸린다. 이처럼 중심값을 기준으로 빠르게 y축의 값이 변화하는 속도를 구하는 것으로 낙상 발생의 유무를 일차 확인할 수 있다. 예를 들어, y 방향으로 키 180cm를 가진 사람이 0.933초 만에 땅에 쓰러졌다고 가정했을 때 1.8/0.933 = 1.93m/s 속도가 나온다. 문제는 2차원 카메라에 잡히는 사람의 키는 원근에 따라 많이 달라짐으로 키에 대한 기준값 설정이 필요한데, YOLO의 바운딩 박스 y 높이를 기준으로 계산했다. 사람마다 약간 다를 수 있으나, 200cm (YOLO의 바운딩 박스는 사람 키를 더 크게 잡기 때문임)라고 잡고, 머리 중심점으로부터는 180cm라고 계산하였다. 10프레임의 슬라이딩윈도우 형태로 평균 속도를 계산하여 1.2m/s 이상의 속도가 검출되면, 다음 단계로 YOLO의 바운딩 박스 Δx와 Δy의 비를 체크하도록 하였다. 즉 y축 속도만으로 낙상을 판단하는 것은 오류가 발생하기 쉽기 때문이다. 이를 추가적으로 판별할 수 있는 방법은 YOLO에서 검출된 바운딩 박스의 Δx와 Δy의 비를 가지고 최종 판단을 내리며, $\Delta x/\Delta y > 1.0$ 조건을 만족하면 최종적으로 쓰러졌다고 판단을 한다.



(a)



(b)

Fig. 12. Testing Results of (a) when one person falls and (b) when one of the two standing people falls

그림 12(a)와 (b)에 한사람이 쓰러지는 경우, 두 사람이 서 있다가 한사람이 쓰러지는 경우를 테스트한 결과를 보였다. 카메라가 비치는 정면으로 쓰러지는 경우는 오동작이 가끔 발생하는데, 카메라 각도에 영향을 받는다. 본 논문에서 제안한 쓰러짐 검출 구현은 기본적인 방법을 구현한 것이고, 좀 더 정밀하게 검출하기 위해서는 움직임 패턴을 나타내는 시계열 데이터를 반영해야 할 것으로 생각된다. 향후 참고문헌 [13]와 유사한 LSTM 혹은 GRU 딥러닝 모델을 이용한 방법을 더 진행될 예정이다.

3.3 텔레그램 SNS 알림 전송 및 DB 서버 구현

스마트폰 SNS 서비스는 텔레그램을 이용하고, API 봇을 이용하여 알림 전송을 구현한다. 전송된 데이터의 저장은 구글의 Firebase DB 서버를 사용하며, 일괄적으로 아래와 같이 동작하도록 구축한다.

STEP ① 카메라 시스템이 헬멧을 쓰지 않은 작업자 또는 쓰러져 있는 작업자 발견

STEP ② 현장시스템의 로컬 FTP 서버에 이미지를 올리고, TCP 소켓을 통해 알림 메세지 전송, TCP 서버는 받은 메시지를 분석하여 긴급하게 관리자에 알람을 보내려면 텔레그램 SNS 송신 프로그램에게 전달

STEP ③ 텔레그램 SNS 송신 프로그램은 FTP 이미지와 전송 메시지를 관리자 스마트폰 텔레그램에 전송하고, 동시에 파이어베이스 DB에 저장

STEP ④ 사후 결과 분석을 위해 DB에 저장된 데이터는 스마트폰 앱을 통해서 검색할 수 있도록 구현

3.3.1 현장 상황을 보고 받기 위한 FTP 및 TCP 서버

헬멧 착용 유무와 낙상 사고 발생을 알려주기 위해서 TCP 소켓 서버와 FTP 서버를 이용한다. 전술한 헬멧 미착용 혹은 낙상사고가 발생하면, 알람 신호를 TCP 소켓에 담아 TCP 서버에 전송함과 동신에 FTP 서버에 이미지를 전달하며, FTP 서버는 pyftpdlib 라이브러리를 사용했다. 그림 13은 FTP를 통해서 받은 이미지와 TCP 소켓 통신에서 받은 메시지를 로그 파일 형태로 생성하여 저장한 txt 파일을 보여준다.

Fig. 13. Images received through FTP and a log file received from TCP socket communication

3.3.2 관리자 텔레그램 SNS 알람 구현

텔레그램에서는 API Bot을 생성하여 특정 입력을 주면 출력 값을 채팅으로 보내는 역할을 수행하는 유저를 만들 수 있으며 이를 채팅 봇이라 한다. 채팅 봇을 만들기 위해서는 텔레그램 메신저에 접속을 한 후 대화상대에서 BotFather을 검색하여 추가한 다음 /start라는 메시지를 보낸 다음 채팅 봇을 만들기 위한 /newbot 메시지를 보낸다. 마지막으로 채팅 봇 이름을 지정해주면, access token과 chat id를 확인할 수 있게 된다. access token과 chat id를 확인하여 API를 사용할 수 있으며, 그림 14는 텔레그램으로 전송한 내용과 소스의 일부이다.

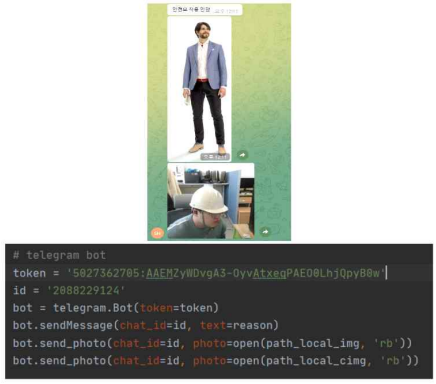


Fig. 14. Images sent through telegram and its source code

3.3.3 현장에서의 상황 저장을 위한 DB 서버 구현

Firebase realtime database를 이용한 데이터베이스를 사용하기 위한 pyrebase, firebase_admin 라이브러리와 데이터를 업로드하기 위해서 firebase console에서 프로젝트를 생성해야 한다. 그 후 프로젝트 설정에서 서비스 계정 항목에서 Firebase admin SDK에서 새 비공개 키를 생성하고 나서 키를 다운 받는다. 받은 파일은 .json 형식으로 이루어져 있으며, 데이터베이스 규칙도 설정해야 한다. Realtime Database 구조는 json 형식을 따르며, 각 항목에 따라서 key(태그)와 value(값)를 가진다. 설정의 Firebase SDK snippet에서 CDN을 체크 한 후 firebaseConfig를 저장한다. 이미지를 Firebase Storage에 저장한 후 데이터베이스에는 url 형식으로 저장해야 하므로 storage를 사용하기 위한 config가 필요하다.

Fig 15. Stored image and alarm message in the Firebase DB

구글 storage에 업로드할 경로를 설정한 뒤 로컬 파일이 있는 위치를 가져와서 사용자가 지정한 storage 경로에 파일을 업로드한다. Storage 경로에 있는 이미지 파일을 url로 변경해서 이미지를 저장한다. 그다음 TCP 통신에서 받은 데이터와 FTP 서버에서 다운받은 이미지를 텔레그램으로 알람을 보내고, 그 후 데이터베이스에 알람을 보냈던 내용과 기록을 그림 15와 같이 저장하였다.

3.3.4 Firebase DB 저장 데이터를 읽기 위한 App 구현

데이터베이스에 저장된 데이터를 관리자가 보기 쉽게 앱을 이용해서 확인 가능하도록 만들었다. 처음 화면을 로딩하는 것을 기다리기 위해 스플래쉬 액티비티를 사용하였으며 그 후 라디오 버튼을 통해 검색 조건을 설정할 수 있도록 만들었다. 검색조건은 사유(알람 발생 이유)와 시간(발생 시간)이며 이 두 버튼은 라디오 그룹으로 묶여 있기 때문에, 하나의 라디오 버튼을 선택 바로 데이터를 확인할 수 있는 리사이클러뷰로

넘어가게 된다. 리사이클러뷰에 Firebase Realtime Database에 있는 데이터를 연결하기 위해서 우선 안드로이드 스튜디오의 패키지에 Firebase를 연결해 줘야 한다. Firebase와 안드로이드를 연결하기 위해 필요한 순서는 첫 번째 Firebase 프로젝트 생성, 두 번째 google-service.json 파일을 다운 후 안드로이드 스튜디오 프로젝트 안에 삽입, 그리고 Firebase SDK 추가, 마지막으로 안드로이드와 Firebase 연결을 확인하면 된다.

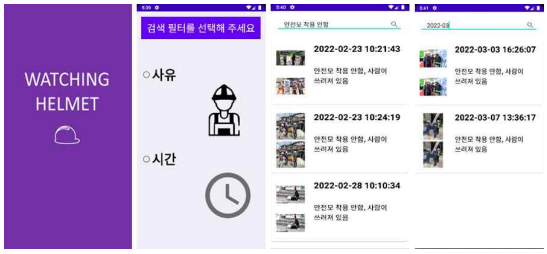


Fig 16. Implementation of Android App that can search contents of Firebase DB

프로젝트 생성을 완료하게 되면 안드로이드와 연결하는 버튼을 통해 앱 등록을 해준다. 패키지 이름과 앱 닉네임 그리고 디버그 서명 인증서를 입력한 뒤 구성 파일 다운로드에서 google-service.json 파일을 다운로드 받을 수 있다. 이 파일은 Firebase에서 발급하는 증명서인데, 이 안에는 Firebase에 필요한 여러 API가 담겨있다. 안드로이드 스튜디오 내부의 biled.gradle에 동기화하면 Firebase를 선택한 뒤 Realtime Database를 클릭하면 1번과 2번 항목이 있는데 이 두 곳을 전부 connected 시켜주면 안드로이드 스튜디오와 Firebase 연동이 끝나게 된다. 그림 16은 Firebase DB의 내용을 열람할 수 있는 안드로이드 App 구현에 대한 동작 사진이다.

Ⅳ. 결론

본 논문에서는 딥러닝 알고리즘을 이용한 컴퓨터 비전 기반 헬멧 착용 여부 검출 및 낙상 유무를 실시간으로 판단하고 관리자에게 알람의 형태로 보내는 시스템을 구현하였다. 사람 객체를 찾기 위해 YOLOv5 기본 모델을 사용하였고, 헬멧 검출은 데이터 셋을 따로 만들어 학습시킨 모델을 사용하였다. 최종 학습된 모델은 검출 정확도를 나타내는 지표 mAP가 평균 0.98 보였고, 건설 현장의 여러 복잡한 상황에서도 헬멧 착용 여부를 대부분 정확하게 검출할 수 있음을 알 수 있었다. Mediapipe Pose를 이용하여 사람 머리 부분의 추락 속도를 구했고, 그로부터 낙상 유무를 판단하는 데 사용하였다. 최종적인 쓰러짐은 YOLO의 바운딩 박스 가로, 세로비와 추락 속도를 연계하여 판단하였다. 카메라 각도와 연관되어 있어 어떤 특정한 각도에서는 오류가 발생하는데, 향후 연구가 더 필요하다. 검출된 헬멧 미착용과 쓰러짐은 텔레그램을 사용하여 관리자에게 알람과 함께 사진 형태로 전송되고 동시에 DB에 저장하며, App 어플을 통해서 간단하게 데이터에 접근할 수 있도록 구현했다.

헬멧 미착용 검출에 있어 일부 특정한 상황에서는 미검출이나 오검출이 발생하였는데, 이러한 경우는 학습되지 않은 형태인 경우, 헬멧과 배경 사이의 명암이나 색상 차이가 작은 경우, 대상이 너무 먼 거리에서 촬영된 이미지에서 주로 나타났고, 형태가 변형된 경우에도 나타났다. 이러한 문제들은 학습 이미지 데이터의 다양성 부족과 잘못된 라벨링에서 일어난 문제이다. 따라서, 이러한 문제점을 보완하기 위해서는 산업 현장을 배경으로 촬영된 이미지, 거리, 크기, 형태 등 다양한 조건에 대한 보다 세분화된 이미지 데이터를 추가로 확보하여 학습시키는 것이 필요하다.

REFERENCES

[1] Analysis of Industrial Accidents in 2020, Available: https://url.kr/7xfh6u

[2] Jeung, Sueng Hyo, Lee, Yong-Soo, Kim, ChangEun, "Improved System for Establishing a Culture to Wear Personal Protective Gear," Journal of the Korea Institute of Construction Safety, 2(1), pp. 16-20, 2019.

[3] S. Ren, K.He, R. Girshick, and J. Sun (2016). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" https://arxiv.org/pdf/1506.01497.pdf (Faster-CNN)

[4] J. Redmon, S. Divvala, R. Girshick, and Ali Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection" https://arxiv.org/pdf/1506.02640.pdf (YOLO)

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection" https://arxiv.org/pdf/2004.10934.pdf (YOLOv4)

[6] https://zenodo.org/record/6222936 (YOLOv5)

[7] YOLOv5 in PyTorch > ONNX > CoreML > TFLite, Available: https://url.kr/gpsmfj

[8] Mediapipe - Google, Available: https://url.kr/hay576

[9] S.Garg, A.Saxena, and R. Gupta, "Yoga pose classification: a CNN and MediaPipe inspired deep learning approach for real-world application," Journal of Ambient Intelligence and Humanized Computing, 03 June 2022.

[10] Receiving push notifications with Telegram Bot API, Available: https://shared.co.kr/196

[11] FireBase Realtime Database, Available: https://firebase.google.com/docs/database?hl=ko

[12] YOLO_Label github, Available: https://url.kr/ulnvh4

[13] Yoon-Kyu Kang, Hee-Yong Kang, and Dalsoo Weon, "Human Skeleton Keypoints based Fall Detection using GRU," Journal of the Korea Academia-Industrial cooperation Society Vol. 22, No. 2 pp. 127-133, 2021.

저자소개

조 용 화 (Yong-Hwa Jo)

2020년 2월 : 경남대학교 정보통신공학과(공학사)
2020년 3월~현재 : 경남대학교 정보통신공학과 석사과정
관심분야 : 딥러닝 및 인공지능

이 혁 재 (Hyuek-Jae Lee)

1994년 2월 : KAIST 전기및전자공학과(공학박사)
1994년 3월~1995년 7월 : LG전자기술원 선임연구원
1995년 7월~2000년 7월 : ETRI 선임연구원
2000년 8월~2001년 11월 UC Davis: post-doc & asst. researcher
2001년 12월~2002년 12월 ROSWIN-USA, Inc CEO
2003년 9월~현재 : 경남대학교 정보통신AI공학과 교수
관심분야 : 광통신/광스위칭, 초고속통신망, 딥러닝응용