각종 재난 상황에서의 경보 알림 시스템
(Alert Notification System for Various Disastrous Situations)

양 희 범, 김 현 모, 구 희 원, 박 상 홍

(Yang Hee beom, Kim Hyeonmo, Koo Hee won, Park Sang Hong)

부경대학교 전자공학과

Ⅰ. 서론

2022년 한 해 동안 각종 재난 사건들로 인해 수많은 인명 피해가 발생하였다. 여름에는 태풍 힌남노의 영향으로 포항 아파트 지하 주차장에서 미처 대피하지 못한 7명의 사상자가 발생하였고, 그 해 10월에는 이태원에서 할로윈 축제를 즐기려던 수많은 인파들이 고지대 골목에서 뒤엉켜 압사 사고가 발생하였다. 이와 같이 재난은 예상치 못한 시간 및 장소에서 발생함에도 불구하고 이에 대처할 시설물이나 경보 시스템이 부재한 실정이다.

이러한 이유로 본 논문에서는 효과적으로 재난상황을 인식하여 사용자들에게 이에 대한 정보를 제공할 수 있는 시스템을 제안한다. 제안된 시스템은 라즈베리 파이 기반의 모니터링 시스템과 각종 센서로 구성되어 있으며, 인구 밀집 구역 및 지하 주차장과 같이 사고 발생에 취약한 공간에 설치되어 위험이 감지될 경우 어플을 통해 사용자에게 경보를 보낸다. 또한 사용자들은 어플을 통해 재난에 대한 수치값을 실시간으로 확인할 수 있다.

Ⅱ. 본론

1. 화재 객체 인식

본 연구에서 객체 감지를 위해 사용한 CNN 훈련 모델은 Effiecientet(2019)이다. EfficientDet은 ImageNet에서 기존 ConvNet보다 8.4배 작으면서 6.1배 빠르고 더 높은 정확도를 갖는 모델로, 경량화된 모델이 요구되는 라즈베리 파이 내에서 객체 감지를 수행하기에 적합하다.

	Input Size Rinput	Backbone Network	BiFpN		Box/class #layers Dclass
			#channels Wbifpn	#layers Dbifpn	
D0(φ = 0)	512	B0	64	3	3
D1(φ = 1)	640	B1	88	4	3
D2(φ = 2)	768	B2	112	5	3
D3(φ = 3)	896	B3	160	6	4
D4(φ = 4)	1024	B4	224	7	4

[표 1] Scaling configs for EfficientDet D0-D4

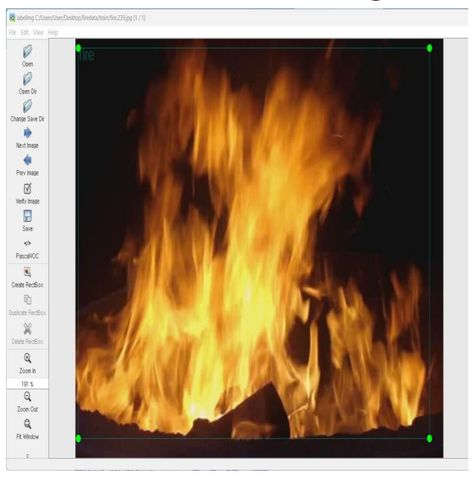

그림 1. 객체 감지를 위한 이미지 라벨링

그림 2. 데이터가 저장된 모습

이미지 라벨링 툴(LabelImg)을 사용하여 그림 1과 같이 훈련을 위한 이미지를 각각 라벨링 하면 그림 2와 같이 데이터셋이 생성된다. 본 연구에서는 EfficientDet-D0 모델을 사용하여 훈련을 진행하였다. 모델 훈련에 사용할 총 단계 수(num_steps)는 40000, 훈련 단계당 사용할 이미지 수(batch_size)는 4로 설정하였다. 학습된 모델은 TFLite 파일로 변환하고 파이 카메라가 실시간으로 캡처한 프레임에서 해당 모델을 통한 객체 감지 추론을 수행한다.

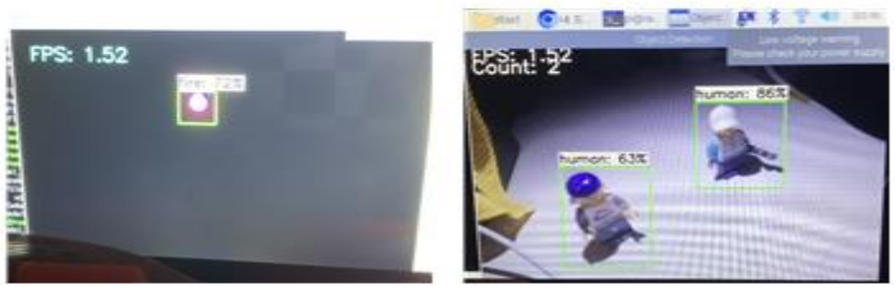

그림 3. 객체 감지 및 실시간 카운팅

본 연구에서는 라즈베리파이 3B+ 모델과 브레드보드를 연결하여 카메라를 통해 객체가 감지되면 LED가 켜지도록 설계하였다. LED가 꺼졌을 때는 0, 켜졌을 때는 1의 값이 서버로 즉시 전송되고 이와 센서값을 대조하여 화재 발생 여부를 판단한다. 동시에, 실시간으로 변화하는 사람의 수를 측정하여 사용자가 해당 공간에 대한 인구 밀집을 확인할 수 있다(그림 3참조).

2. Flask 서버

사용자에서 알림을 보낼 app에 필요한 데이터를 저장하고 수집된 정확한 데이터를 보여주기 위해 flask 서버를 이용하였으며 log열람을 위한 페이지를 분리하여 최근에 수집된 데이터들을 사용자가 직접 확인할 수 있게 하였다(그림 4)(그림 5).

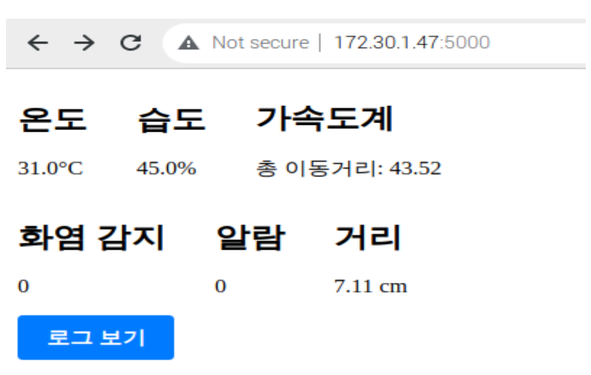

그림 4. Flask 서버 메인 화면

그림 5. 센서로부터 수집된 데이터 기록

4. App Inventor

어플리케이션을 구현하기 위해 사용한 프로그램은 App Inventor로, Flask 서버와 어플을 연동하기 위해서 Flask 서버의 외부접근이 가능해야한다. 이를 위한 방식으로는 ngrok 방식을 택하였다. Flask 서버의 데이터들을 어플로 불러오기 위해 GET 방식으로 데이터를 요청하면 데이터 값을 json형식으로 return한다. json형식의 데이터를 전역변수에 저장하여 list형식으로 나열한 후 split하여 각 변수에 저장한다. 이 저장된 변수들을 label에 저장한다.

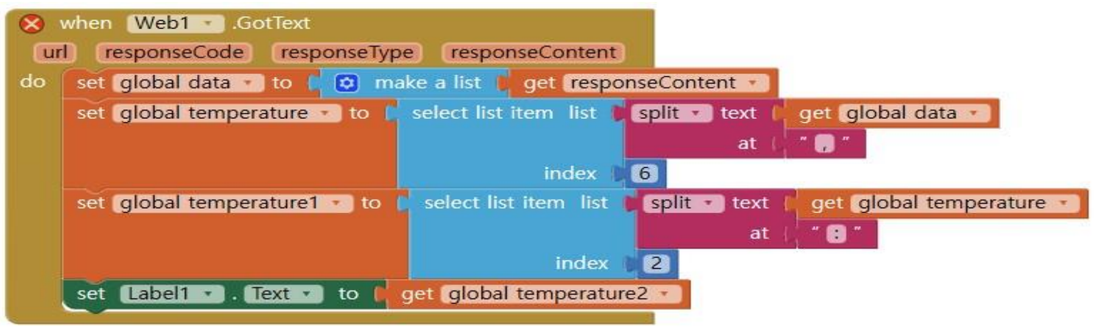

그림 6. App inventor 블록 구성도

그림 7. 앱 인벤터 화면 구성

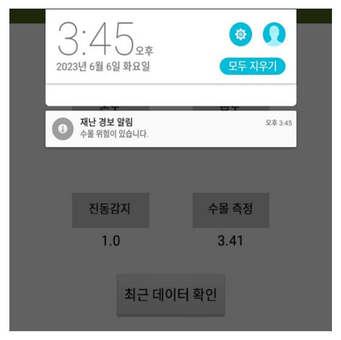

그림 8. 재난 경보알림

그림 7는 온습도 센서, 가속도 센서, 초음파 센서에 관한 실시간 데이터 값을 label에 표기한 것이고, 각 센서의 로그 데이터를 최신순으로 표기한 것이다.

객체 탐지 결과와 불꽃 감지 센서의 데이터를 비교하여 두 데이터 모두 참의 값이 들어왔을 경우 화재재난 알림을 보낸다(그림 8). 기울기 센서 데이터와 초음파 센서 데이터를 통해서 임계값 이상일 경우 지진 재난 알림과 수몰 위험 알림을 보낸다.

Ⅲ. 결론

본 논문에서는 객체 탐지를 위해 CNN의 EfficientNet 모델을 이용하여 데이터셋을 학습시켜 인식 모델을 생성하였으며, 객체탐지 대상의 정확도가 낮은 경우에도 비교적 정확하게 식별하기 위해 센서 데이터와 대조하여 최종적으로 판단하는 알고리즘을 적용하였다.

다만, 하드웨어 사양의 한계로 카메라 프레임이 낮게 측정되는 단점이 있고, 이로 인해서 빠르게 이동하는 객체는 영상에서 객체 대상을 탐지하기 위한 적절한 프레임을 추출하기 어렵다는 문제가 존재한다. 이를 보완하기 위해 본 논문에서는 카메라와 감지할 객체를 특정한 위치에 고정하고 장애물이 없는 깔끔한 배경에서 촬영을 진행하였으며 보다 정확한 결과를 얻기 위해서는 더 가벼운 인식기법의 적용이 필요할 것이다.

REFERENCES

[1] Mingxing Tan, Quoc V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2020/09/11
[2] https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images
[3] https://www.kaggle.com/datasets/ihelon/lego-minifigures-classification
[4] https://github.com/freedomwebtech/tensorflow-lite-custom-object
[5] https://github.com/freedomwebtech/tensorflow-lite-bullseye
[6] https://colab.research.google.com/github/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Train_TFLite2_Object_Detction_Model.ipynb
[7] http://puravidaapps.com/notification.php
[8] https://egeasy.tistory.com/entry/%EC%95%B1%EC%9D%B8%EB%B2%A4%ED%84%B0-%EA%B0%95%EC%A2%8C-5-API-JSON-%EC%82%AC%EC%9A%A9%EB%B0%A9%EB%B2%95-%EB%B0%8F-Dictionary-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8-%ED%99%9C%EC%9A%A9%ED%95%98%EA%B8%B0