2024 남서울대학교 지능정보통신공학과 졸업논문집 Vol.25

YOLO v5를 이용한 금연 구역 흡연 방지 시스템

Smoking Detection System in No-Smoking Areas Using YOLO v5

강희원*, 김웅빈**, 한규범***
Hee-Won Kang*, Woong-Bin Kim**, Kyu-Beom Han***

* 남서울대학교 지능정보통신공학과 4학년 hhw9790@naver.com
** 남서울대학교 지능정보통신공학과 4학년 kimob1210@naver.com
*** 남서울대학교 지능정보통신공학과 4학년 gksrbqja1234@naver.com

요 약

본 논문에서는 YOLO v5를 이용하여 금연 구역에서의 흡연을 감지하는 시스템을 구현하였다. 이 시스템은 라즈베리파이, 카메라, 블루투스 스피커, 부저를 사용하여 구성하였으며, YOLO v5와 학습시킨 모델을 통해 카메라가 담배를 인식하면 즉각적으로 부저와 블루투스 스피커를 통해 경고를 전달하도록 구현되었다. 또한, 해당 금연 구역을 관할하는 관리자가 실시간으로 해당 구역을 감시할 수 있도록 관리자 웹 페이지를 구축하였다. 즉각적인 경고가 무시되는 경우를 대비하여, 담배가 인식된 순간의 모습을 캡처하여 시간 정보를 포함한 사진 파일로 저장함으로써 법적 처벌을 위한 증거 자료로 활용될 수 있다.

키워드 : YOLO v5, , 흡연 감지, 라즈베리파이, 블루투스 스피커, 실시간 경고 시스템

I. 서 론

21세기는 인공지능과 사물인터넷(IoT) 기술이 급격하게 발전하고 있는 시대이다. 이러한 기술들은 공공 안전과 보건 분야에서도 중요한 역할을 하고 있으며, 금연 구역에서의 흡연 문제 해결에 대한 수요도 증가하고 있다. 간접흡연으로 인한 비흡연자의 건강 피해와 환경 오염 문제는 사회적 문제로 대두되고 있으며, 금연 구역에서의 흡연을 효과적으로 감지하고 방지할 수 있는 스마트 솔루션의 필요성이 더욱 절실해지고 있다.

따라서 본 논문에서는 금연 구역에서의 흡연을 감지하고 즉각적인 경고를 제공하는 시스템을 제안한다. 이 시스템은 라즈베리파이, 카메라, 블루투스 스피커, 부저를 이용하여 구현되었으며, YOLO v5 모델을 통해 담배를 실시간으로 인식한다. 또

한, 관리자가 금연 구역을 실시간으로 감시할 수 있는 웹페이지를 구축하며, 후속 조치에 대비하여 흡연 시, 그 장면을 캡처하여 법적 처벌을 위한 증거 자료로 활용할 수 있도록 설계되었다. 이를 통해 공공장소의 쾌적함을 유지하고 비흡연자의 건강을 보호하는 데 기여하고자 하였다.

II. 하드웨어 구성

본 논문에서 구현한 시스템의 하드웨어 구성은 그림 1과 같다. 이 시스템은 라즈베리파이, 카메라, 블루투스 스피커, 부저로 구성되어 있다. 카메라는 실시간으로 금연 구역을 감시하며, 현장의 영상을 라즈베리파이에 전송한다. 이 카메라는 높은 해상도와 빠른 프레임 속도로 실시간 감시가 가능하도록 선택되었다. 라즈베리파이는 카메라로부터 전송된 실시간 영상 데이터를 처리한다. 이 과정에서 라즈베리파이는 YOLO v5 모델을 사용하여 담배를 인식하며, 담배가 인식되면 즉각적으로 조치를 취한다. 부저는 담배가 인식되었을 때 라즈베리파이의 신호에 따라 5초간 경고음을 발생시킨다. 이는 금연 구역에서의 즉각적인 주의 환기를 위해 설계된 것이다. 부저 경고 후, 라즈베리파이는 블루투스 스피커를 통해 경고 문구를 송출한다. 이 추가적인 경고는 흡연자의 주의를 명확하게 환기시키기 위함이다.

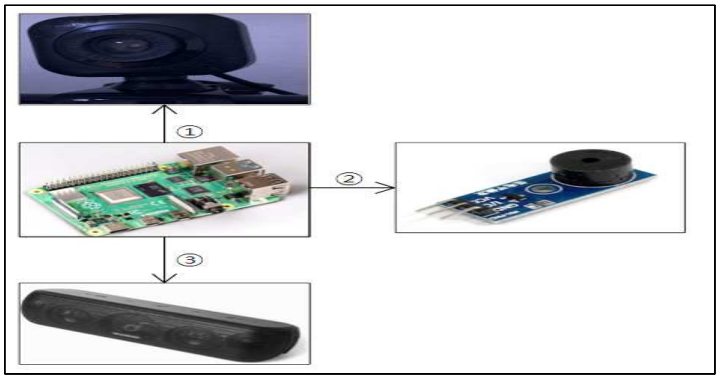

그림 1. 하드웨어 구성도

라즈베리파이와 각 구성 요소 간의 연결 및 데이터 흐름은 다음과 같이 이루어진다.

카메라 → 라즈베리파이 (①): 실시간 현장 영상 데이터 전송.

라즈베리파이 → 부저 (②): 담배 인식 시 즉각적인 경고음을 발생.

라즈베리파이 → 블루투스 스피커 (③): 부저 경고 후 경고 문구 송출.

이와 같은 하드웨어 구조는 금연 구역에서의 흡연 행위를 신속하게 감지하고 경고를 전달하는 데 최적화되어 있다. 실시간 데이터 처리와 경고 시스템의 조화를 통해 효과적인 금연 구역 관리를 목표로 한다.

III. 소프트웨어 구성

본 논문에서 사용된 주요 도구는 다음과 같다.

- Roboflow – 담배 인식 모델을 만들기 위해 필요한 데이터 셋 수집, 라벨링, 및 이미지 증강 작업을 수행하기 위해 사용하였다. 다양한 이미지 전처리 및 증강 기법을 제공하여 데이터 셋의 품질을 향상시키는 데 도움을 주었다(그림 2).
- Google Colab – Roboflow에서 편집하고 완성한 데이터 셋을 활용하여 모델을 학습시키기 위한 환경으로 사용하였다. Google Colab은 무료로 GPU를 제공하며, Python 환경에서 딥러닝 모델을 효율적으로 학습시킬 수 있는 플랫폼이다(그림 3).
- YOLO V5 – 본 연구의 중심이 되는 객체 검출 모델로, YOLO V5를 사용하여 담배 인식 모델을 구축하였다. YOLO V5는 높은 정확도와 실시간 처리 능력을 갖춘 객체 검출 모델로, 본 연구의 목적에 부합하는 최적의 선택이었다(그림 4).

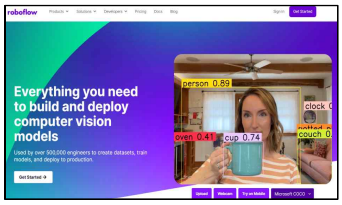

그림 2. roboflow

그림 3. Google colab

그림 4. YOLO v5

1. 모델 학습 과정

본 논문에서는 담배 이미지를 roboflow 사이트에 업로드하여 라벨링 작업을 수행하고, 이미지 전처리 및 증강 기법을 활용하여 약 25,470장의 담배 이미지 데이터를 확보하였다. 이 데이터 셋은 담배 인식 모델을 만들기 위한 주요 자원으로 사용되었다. 다음 그림 5는 데이터 셋 구축 과정을 시각적으로 보여준다.

그림 5. 데이터 셋 구축 과정

이후, 구축된 데이터 셋을 기반으로 Google Colab에서 YOLO v5모델을 학습시켰다. YOLO v5 파일 안에 포함된 train.py를 사용하여 담배 인식 모델인 cigaratte.pt를 추출하였다.

2. 모델 학습 명령어

Google Colab에서 yolov5 모델 학습을 위한 주요 명령어는 다음과 같다.

① !pwd

: 현재 작업 디렉토리를 출력한다.

② !curl -L "데이터 셋 저장된 주소의 암호화 키값" > roboflow.zip

: 주어진 URL에서 roboflow.zip파일을 다운로드한다.

③ !unzip roboflow.zip

: 다운로드한 ZIP 파일을 압축 해제한다.

④ !rm roboflow.zip

: 압축 해제 후 ZIP 파일을 삭제한다.

⑤ !git clone https://github.com/ultralytics/yolov5.git

: YOLOv5 저장소를 GitHub에서 로컬 디렉토리로 클론한다.

⑥ %cd /content/yolov5/

: 현재 작업 디렉토리를 /content/yolov5/로 변경한다.

⑦ !pip install -r requirements.txt

: requirements.txt파일에 명시된 모든 파이썬 패키지들을 설치한다.

⑧ %cat /content/data.yaml

: /content/data.yaml파일의 내용을 출력한다.

⑨ !python train.py --img 416 --batch 16 --epochs 50 --data /content/data.yaml --cfg ./models/yolov5s.yaml --weights yolov5s.pt —name cigarette_yolov5s_results

: YOLOv5 모델을 사용하여 담배 인식 모델을 훈련한다. 이 명령어는 이미지 크기를 416x416으로 설정하고, 배치 크기를 16으로, 에포크 수를 50으로 설정하여 모델을 학습시킨다. 최종 모델과 결과는 cigarette_yolov5s_results디렉토리에 저장된다.

추가 설명으로 --img 416: 입력 이미지의 크기를 416x416으로 설정,

--batch 16: 배치 크기를 16으로 설정, 이는 훈련 시 한 번에 처리할 이미지 수를 의미(배치 크기가 클수록 메모리 사용량이 증가), --epochs 50: 전체 훈련 데이터셋을 50번 반복하여 훈련한다 라는 의미를 가진다.

3. epoch별 YOLO v5 모델의 손실 값 및 성능 지표 변화 분석

그림 6은 YOLO v5 모델의 학습 과정 중에서 중요한 에포크별 성능 지표와 손실 값을 나타낸 것이다.

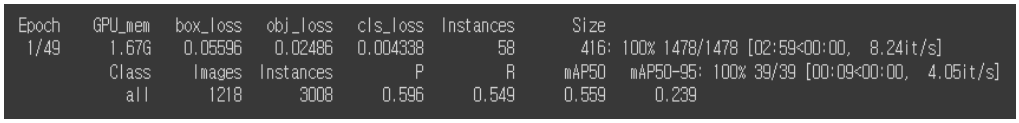

그림 6. 모델 학습을 두 번째 에포크한 성능 결과

- Epoch 및 GPU 메모리 사용량: 학습은 총 50번의 에포크 중 두 번째 에포크를 나타내며, GPU 메모리 사용량은 1.67GB로 기록되었다. 손실 함수 (Loss): 경계 상자(box)의 위치와 크기를 예측하는 데 사용되는 손실 값(box_loss)은 0.05595로 나타났다. 이는 경계 상자의 예측이 비교적 정확함을 의미한다. 객체가 있는지 없는지를 판단하는 객체 손실 값(obj_loss)은 0.02486으로, 객체 존재 유무를 잘 예측하고 있음을 시사한다. 객체의 클래스를 예측하는 클래스 손실 값(cls_loss)은 0.004393으로, 클래스 예측의 정확도가 높음을 나타낸다. 인스턴스 수: 현재 에포크 동안 학습에 사용된 이미지 수는 58장이다. 이미지 크기 및 처리 속도: 학습에 사용된 이미지의 크기는 416 픽셀이며, 100번의 배치(batch)당 1478개의 이미지를 처리했다. 총 학습 시간은 2시간 59분이고, 처리 속도는 초당 8.24 이미지였다.

- 클래스 및 전반적인 성능 지표: 평가된 클래스에 대해 학습에 사용된 이미지 수는 1218장이고, 객체 인스턴스 수는 3008개이다. 모델의 정밀도(Precision, P)는 0.596으로 예측한 객체 중 실제로 맞은 객체의 비율이 59.6%임을 나타낸다.

재현율(Recall, R)은 0.549로 실제 객체 중 모델이 맞게 예측한 객체의 비율이 54.9%임을 보여준다. Intersection over Union (IoU)이 0.5일 때의 평균 정밀도(mAP@0.5)는 0.559로, 55.9%의 값을 나타낸다. IoU 임계값이 0.5에서 0.95까지 변화할 때의 평균 정밀도(mAP@0.5:0.95)는 0.239로, 23.9%의 값을 나타낸다.

평가를 하면 첫 번째 에포크의 결과는 모델이 학습 초기 단계임에도 불구하고 양호한 성능을 보이고 있다. 학습의 마지막 에포크(Epoch)인 50번째 에포크의 결과는 그림 7과 같다.

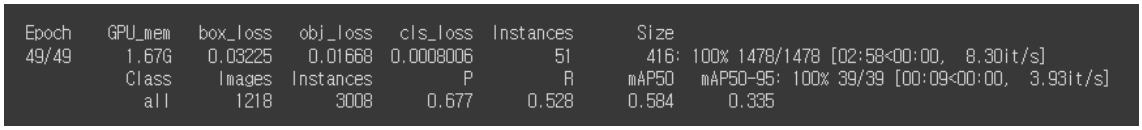

그림 7. 모델 학습을 50번째 에포크한 성능 결과

- Epoch 및 GPU 메모리 사용량 : 학습은 총 49번의 에포크 중 마지막 에포크인 50번째 에포크를 나타내며, GPU 메모리 사용량은 1.67GB로 기록되었다. 손실 함수 (Loss): 경계 상자(box)의 위치와 크기를 예측하는 데 사용되는 손실 값(box_loss)은 0.03255로, 초기 에포크에 비해 감소하였다. 이는 경계 상자의 예측이 더욱 정확해졌음을 의미한다. 객체가 있는지 없는지를 판단하는 객체 손실 값(obj_loss)은 0.01688로 감소하여 객체 존재 유무를 더욱 잘 예측하고 있음을 시사한다. 객체의 클래스를 예측하는 클래스 손실 값(cls_loss)은 0.000805로 매우 낮아져, 클래스 예측의 정확도가 크게 향상되었음을 나타낸다. 인스턴스 수: 현재 에포크 동안 학습에 사용된 이미지 수는 51장이다. 이미지 크기 및 처리 속도: 학습에 사용된 이미지의 크기는 416픽셀이며, 100번의 배치(batch)당 1478개의 이미지를 처리했다. 총 학습 시간은 2시간 58분이고, 처리 속도는 초당 8.30 이미지였다.

- 클래스 및 전반적인 성능 지표 : 평가된 클래스에 대해 학습에 사용된 이미지 수는 1218장이고, 객체 인스턴스 수는 3008개이다. 모델의 정밀도(Precision, P)

는 0.677로, 예측한 객체 중 실제로 맞은 객체의 비율이 67.7%임을 나타낸다. 재현율(Recall, R)은 0.528로, 실제 객체 중 모델이 맞게 예측한 객체의 비율이 52.8%임을 보여준다. Intersection over Union (IoU)이 0.5일 때의 평균 정밀도(mAP@0.5)는 0.584로, 58.4%의 값을 나타낸다. IoU 임계값이 0.5에서 0.95까지 변화할 때의 평균 정밀도(mAP@0.5:0.95)는 0.335로, 33.5%의 값을 나타낸다.

최종 평가를 하면 마지막 에포크의 결과는 모델이 학습을 통해 성능이 개선되었음을 보여준다. 특히, 손실 값(box_loss, obj_loss, cls_loss)이 초기 에포크에 비해 상당히 감소하였고, 정밀도(P)와 재현율(R)이 각각 67.7%와 52.8%로 향상되었다. 또한, mAP 값이 IoU 임계값 0.5에서 58.4%로 나타났으며, 더 높은 IoU 임계값 범위(0.5에서 0.95)에서도 33.5%로 증가하였다. 이러한 결과는 모델이 다양한 임계값에서도 객체를 정확히 검출할 수 있음을 시사한다.

4. 학습한 모델 그래프화 성능 분석

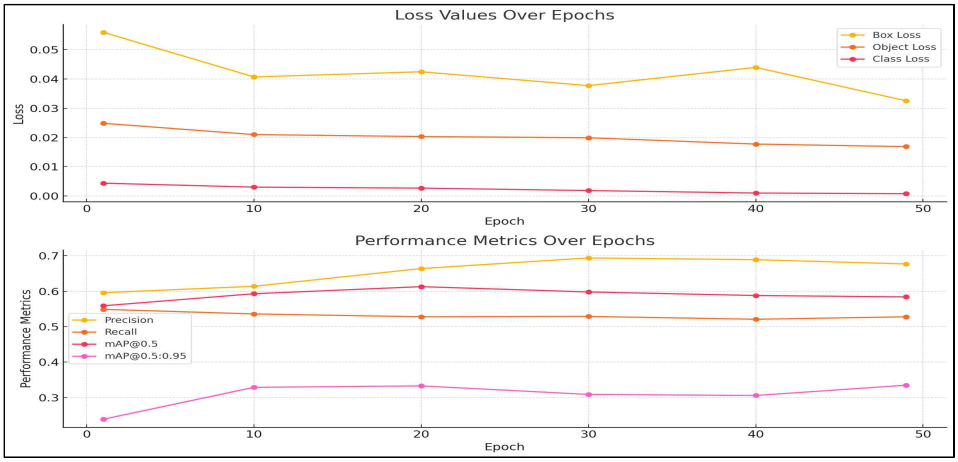

그림 8. 학습 시킨 모델의 손실 값과 성능 지표 그래프화

YOLO V5 모델의 학습 과정을 분석하기 위해 첫 번째 에포크부터 마지막 에포크까지의 손실 값과 성능 지표를 그림8과 같이 그래프로 나타냈다. 그래프는 손실 값

의 변화를 보여주는 상단 그래프와 성능 지표의 변화를 보여주는 하단 그래프로 구성된다.

상단 그래프는 손실 값 변화 (Loss Values Over Epochs) Box Loss, Object Loss, Class Loss의 변화를 보여준다.

- Box Loss: 첫 번째 에포크에서 0.05595였으나, 마지막 에포크에서는 0.03255로 감소하였다. 이는 경계 상자의 예측이 점차 정확해졌음을 나타낸다.
- Object Loss: 첫 번째 에포크에서 0.02486이었으나, 마지막 에포크에서는 0.01688로 감소하였다. 이는 모델이 객체의 존재 유무를 판단하는 데 있어 더 정확해졌음을 시사한다.
- Class Loss: 첫 번째 에포크에서 0.004393이었으나, 마지막 에포크에서는 0.000805로 크게 감소하였다. 이는 모델이 객체의 클래스를 예측하는 데 있어 매우 높은 정확도를 달성했음을 의미한다.

하단 그래프는 성능 지표 변화 (Performance Metrics Over Epochs) Precision, Recall, mAP@0.5, mAP@0.5:0.95의 변화를 보여준다.

- Precision: 첫 번째 에포크에서 0.596이었으나, 마지막 에포크에서는 0.677로 증가하였다. 이는 예측한 객체 중 실제로 맞은 객체의 비율이 높아졌음을 의미한다.
- Recall: 첫 번째 에포크에서 0.549이었으나, 마지막 에포크에서는 0.528로 약간 감소하였다. 이는 실제 객체 중 모델이 맞게 예측한 객체의 비율이 약간 줄어들었음을 나타낸다.
- mAP@0.5: 첫 번째 에포크에서 0.559이었으나, 마지막 에포크에서는 0.584로 증가하였다. 이는 IoU 임계값 0.5에서의 평균 정밀도가 높아졌음을 의미한다.
- mAP@0.5:0.95: 첫 번째 에포크에서 0.239이었으나, 마지막 에포크에서는 0.335로 크게 증가하였다. 이는 다양한 IoU 임계 값에서의 평균 정밀도가 크게 향상되었음을 의미한다.

종합 분석하면, 첫 번째 에포크와 마지막 에포크의 비교 결과, 모델의 성능이 전반

적으로 향상되었음을 확인할 수 있다. 특히, 손실 값(box_loss, obj_loss, cls_loss)이 크게 감소하였으며, 이는 모델의 예측 정확도가 향상되었음을 나타낸다. Precision과 mAP 값이 증가하였고, 특히 mAP@0.5:0.95가 큰 폭으로 증가하여 모델이 다양한 임계 값에서도 정확하게 객체를 검출할 수 있음을 시사한다. 이러한 결과는 모델이 학습을 통해 더 정확하고 신뢰성 있는 객체 검출을 수행할 수 있게 되었음을 보여준다.

5. 시스템 구현 주요 코드 설명

완성된 모델인 cigaratte.pt와 YOLO v5 안에 있는 detect.py를 사용하여, 담배 인식 시 부저와 블루투스 스피커를 통해 경고 문구를 알리는 기능을 구현하였다. 또한, 카메라에서 담배가 인식된 순간의 모습을 캡처하여 시간 정보를 포함한 사진 파일로 저장하는 기능도 함께 구현되었다. 각 기능별 구현 프로그램 코드는 부록 A와 같다. 그림 9는 담배 인식시, 이미지를 저장하는 것을 구현한 화면이고 그림 10은 관리자가 금연 구역을 실시간 감시하고 있는 화면의 모습을 웹을 통해서 볼 수 있게 하기 위해 구현한 웹 페이지 화면이며, 그림 11은 관리자가 로그인을 통해 인증 성공시, 실시간 화면의 모습을 보여주고 있는 모습을 구현한 화면이다.

그림 9. ③을 구현한 화면

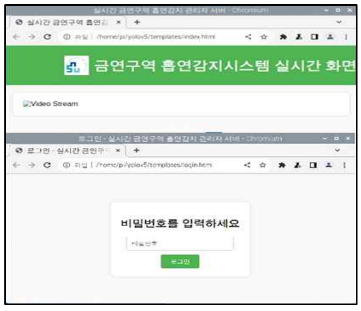

그림 10. ④,⑥,⑦을 구현한 화면

그림 11. ⑤을 구현한 화면

6. 시스템 소프트웨어 구성도

기능 구현을 완료한 후, 그림 12는 시스템의 소프트웨어 구성도를 보여준다. 이

구성도는 시스템의 데이터 흐름, 주요 구성 요소 간의 관계, 그리고 최종 모델이 어떻게 동작하는지를 명확히 설명한다.

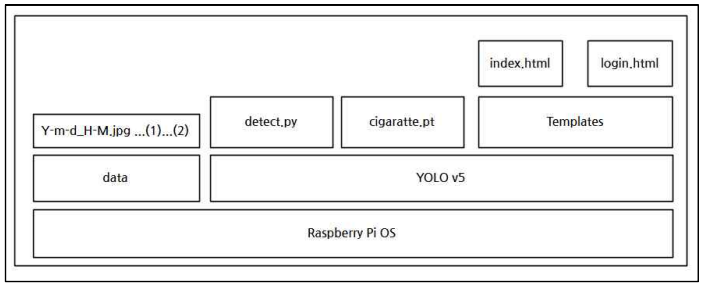

그림 12. 시스템 소프트웨어 구성도

- Raspberry Pi OS : 시스템의 운영체제 역할을 하며, 모든 소프트웨어와 하드웨어가 이 기반 위에서 작동한다.
- YOLO v5 : 담배 인식 모델을 실행하는 핵심 소프트웨어로, 카메라로부터 실시간으로 제공되는 영상을 분석하여 담배를 인식한다.
- data: 담배가 인식된 순간을 캡처한 이미지 파일을 모아두는 저장소이다..
- detect.py : 담배를 인식하고 경고 시스템을 작동시키는 스크립트 파일이다.
- cigaratte.pt : 학습된 YOLO v5 모델 파일로, 담배를 인식하는 역할을 한다.
- Templates : 관리자 웹페이지에 사용되는 HTML 템플릿 파일들 (index.html, login.html).
- Y-m-d_H-M.jpg : 담배가 인식된 순간을 캡처한 이미지 파일로, 시간 정보를 포함하여 저장된다.

7. 작품의 최종 모델 디자인

다음 그림 13은 작품의 최종 모델 디자인이며, 시스템 외관의 눈 부분에는 카메라가 설치되어 실시간으로 영상을 제공하며, 내부에는 부저와 라즈베리파이, 블루투스 스피커 등의 경고 시스템이 포함되어 있다. 담배가 인식될 경우, 부저가 울리고 블루투스 스피커를 통해 경고 메시지가 송출된다.

그림 13. 금연 구역 흡연 방지 시스템 모델 디자인한 외형

Ⅳ. 결 론

본 연구에서는 금연 구역에서의 흡연을 실시간으로 감지하고 경고를 제공하는 시스템을 개발하였다. YOLO v5 모델을 활용하여 담배를 인식하고, 라즈베리파이와 카메라를 이용하여 실시간 데이터를 처리함으로써 금연 구역에서의 흡연 행위를 신속하게 감지할 수 있었다. 최종 모델의 성능 평가 결과, Precision이 0.677, mAP@0.5가 0.584로 나타나, 모델이 효과적으로 담배를 인식할 수 있음을 확인하였다. 그러나 본 연구는 특정 데이터 셋에 기반하여 모델을 학습시켰으며, 다양한 환경에서의 성능 검증이 부족한 한계가 있다. 향후 연구에서는 더욱 다양한 환경에서의 데이터 셋을 추가로 수집하고, 모델의 성능을 개선하는 것이 필요하다. 또한, 이 시스템은 공공장소의 쾌적함을 유지하고 비흡연자의 건강을 보호하는 데 중요한 역할을 할 것으로 기대된다.

참고문헌

[1] 조앤 유 (2021). YOLO 객체 탐지: 실습과 이론. 서울: 한빛미디어.

[2] 매튜 리처드슨, 숀 월리스 (2016). 라즈베리파이로 배우는 간단한 IoT 프로젝트. 한빛미디어.

부록 A. 기능별 구현 프로그램 코드

① 부저 울리기 기능

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.OUT)
def activate_buzzer():
    GPIO.output(23, GPIO.HIGH)
    time.sleep(5)  # 5초 동안 부저 울리기
    GPIO.output(23, GPIO.LOW)
```

② 개체별 인식 시 경고 문구 알림

```
alert_messages ={
            "cigarette": "여기는 금연구역! 담배 대신 미소를 피워 보는 건 어떨까요?",
            "person": "좋은 하루 입니다"
}
```

```
        def play_alert_message(label):
            message =get_time_based_message()
            if message is None:
                message =alert_messages.get(label)
            if message:
                tts =gTTS(text=message, lang='ko')
                tts.save("/tmp/message.mp3")
                  threading.Thread(target=subprocess.run,  args=(["mpg123",
"/tmp/message.mp3"],)).start()
```

③ 담배 인식 시 이미지 저장

```
def save_detected_image(img, label, save_dir):
    if label =="cigarette":
        now =datetime.now()
        filename =now.strftime("%Y-%m-%d_%H-%M") +".jpg"
        folder_path ="/home/pi/datafile"
        os.makedirs(folder_path, exist_ok=True)
        path =os.path.join(folder_path, filename)
        cv2.imwrite(path, img)
        print(f"Saved detected image to {path}")
```

추가로, 금연 구역을 실시간으로 감시할 수 있도록, 관리자가 접근할 수 있는 관리자 웹페이지를 구축하였다.

④ Flask를 사용한 라우팅

```
@app.route("/")
def login():
    return render_template('login.html')
@app.route('/index')
```

```
def index():
    return render_template("index.html")
@app.route('/authenticate', methods=['POST'])
def authenticate():
    password =request.form.get('password')
    if password ==PASSWORD:
        return redirect(url_for('index'))
    else:
        return "비밀번호가 잘못되었습니다.", 401
```

⑤ 비디오 피드 생성 및 전송

```
@app.route("/video_feed")
def video_feed():
    return Response(generate(), mimetype="multipart/x-mixed-replace; boundary=frame")
def generate():
    global outputFrame, lock
    while True:
        with lock:
            if outputFrame is None:
                continue
            flag, encodedImage =cv2.imencode(".jpg", outputFrame)
            if not flag:
                continue
                                    yield(b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n'+bytearray(encodedImage) +b'\r\n')
```

⑥ HTML 템플릿 - index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>실시간 금연구역 흡연감지 관리자 서버</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f3f3f3;
        }
        header {
            background-color: #4CAF50;
            color: white;
            padding: 10px 0;
            display: flex;
            align-items: center;
            justify-content: center;
         }
        header img {
            height: 50px; /* 로고의 크기를 조정합니다 */
            margin-right: 10px; /* 제목과 로고 사이의 간격을 조정합니다 */
        }
        main {
            width: 90%;
            margin: 20px auto;
            background-color: white;
            padding: 20px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
```

```
        }
        img {
            display: block;
            max-width: 100%;
            height: auto;
            margin: 0 auto;
        }
    </style>
</head>
<body>
    <header>
            <h1>금연구역 흡연감지시스템 실시간 화면</h1>
    </header>
    <main>
        <img src="{{ url_for('video_feed') }}" alt="Video Stream">
    </main>
</body>
</html>
```

⑦ HTML 템플릿 – login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>로그인 – 실시간 금연구역 흡연감지 관리자 서버</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
```

```
    padding: 0;
    background-color: #f3f3f3;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
 #login-form {
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    text-align: center;
}
input[type="password"] {
    width: 200px;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
 input[type="submit"] {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
```

```
    </style>
  </head>
  <body>
    <div id="login-form">
      <h2>비밀번호를 입력하세요</h2>
      <form action="/authenticate" method="post">
        <input type="password" name="password" placeholder="비밀번호
">
        <br>
        <input type="submit" value="로그인">
      </form>
    </div>
  </body>
  </html>
```