

이미지 분석 프로젝트 보고서

(객체 인식을 통한 차량 방치사고 대비)

목 차

제 1 장 - 배경

제 2 장 - 주제

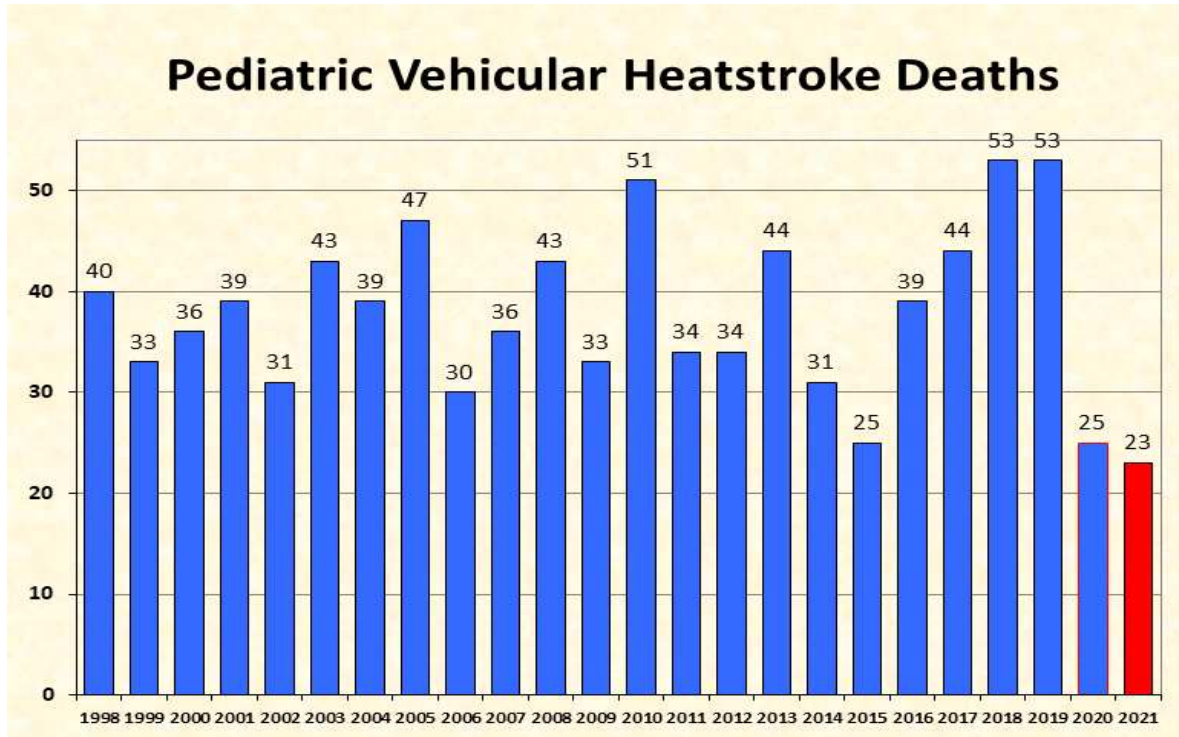
제 3 장 - 진행 과정
파이썬 코드

제 4 장 - 활용 방안

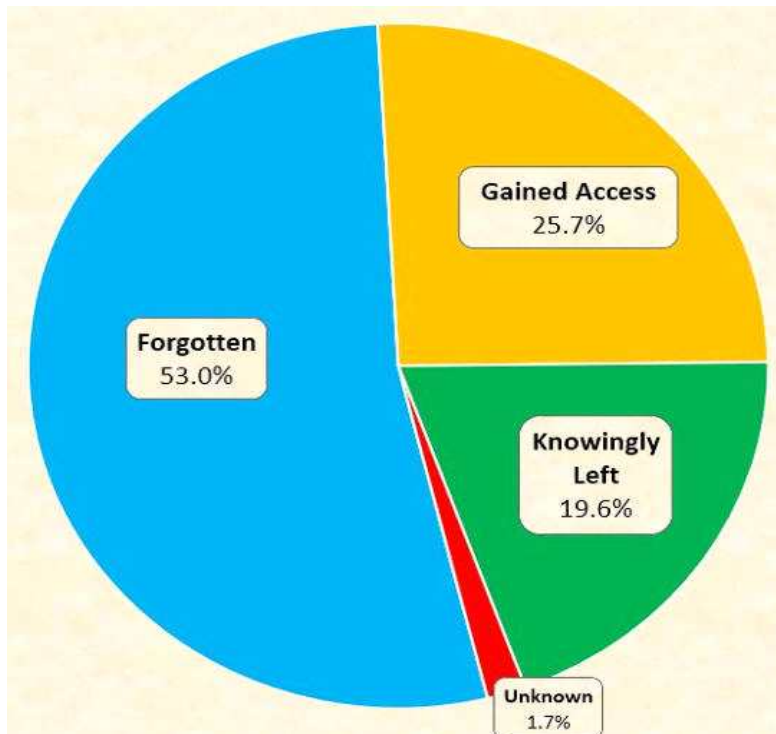
제 5 장 - 관련 자료

선정 배경

과거 뉴스를 보면서 영유아의 차량 방치로 인한 인명 사고에 관한 내용을 보았던 기억이 있어 관련 자료를 찾아보면서 알아본 결과 우리나라 외에도 많은 국가에서 이러한 사고로 인해 많은 인명 피해가 있음을 알 수 있었다.



해당 자료는 미국에서 매년 차량 방치로 인해 사망한 사례에 대한 자료로 이처럼 기술은 발달했지만 줄지 않고 꾸준히 발생하고 있음을 알 수가 있다.



또한 50퍼센트 이상이 차량에 타고 있음을 잊어버려서 발생한 사고임을 알 수 있다.

우리나라의 사고 사례들을 살펴보면 주로 어린이집 통학 차량에서 발생하는 경우가 많다. 통학 차량은 주로 버스나 승합차를 이용하기에 아이가 만약 차량에서 자고 있거나 장난을 친다고 숨어있는 다면 직접 가서 보지 않는다면 알아채지 못할 수 있다.


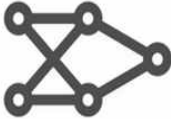


이를 해결해보고자 해당 내용을 결정하게 되었다.

주제

운전자나 인솔자의 방심, 실수로 인하여 발생하는 영유아 차량 방치사고를 방지하고자 차량에 설치되어 있는 차량용 블랙박스와 차량용 cctv 영상을 이용하여 차량 운행이 종료되고 나서 일정 시간이 지나고 난 뒤 즉 시동이 꺼지고 난 뒤 해당 영상을 이용하여 YOLOv5의 코코데이셋을 이용하여 사람(영유아)가 탐지된다면 운전자나 운행 책임자에게 휴대폰이나 메일을 통하여 경고 알람을 보내서 차량 내부를 확인하도록 하여 차량에 남아있는 사람을 차에서 내려 사고를 사전에 방지하도록 한다.

진행 과정

먼저 해당 주제는 사람을 탐지하는 것이 주된 목적이기에 해당 모델을 훈련시켜 만들려고 했지만 이미 cocodata를 이용해 사람을 탐지하는 모델이 이미 존재했기에 해당 모델을 사용하여 진행하기로 하였다. 해당 모델은 yolov5를 사용하기에 정확도와 속도에 따른 네 가지 모델이 존재한다.

			
Small YOLOv5s	Medium YOLOv5m	Large YOLOv5l	XLarge YOLOv5x
14 MB _{FP16} 2.2 ms _{V100} 36.8 mAP _{COCO}	41 MB _{FP16} 2.9 ms _{V100} 44.5 mAP _{COCO}	90 MB _{FP16} 3.8 ms _{V100} 48.1 mAP _{COCO}	168 MB _{FP16} 6.0 ms _{V100} 50.1 mAP _{COCO}

다음과 같이 네 가지가 존재하는데 small은 정확도는 가장 낮지만, 탐지 속도는 가장 빠르다. x-large 모델은 정확도는 가장 높고, 속도는 가장 느리다는 특징을 가지고 있다.

이에 어느 모델이 해당 주제를 진행하는 데 있어 가장 적합한지 알아보기 위해 주피터 노트북(jupyter notebook) 환경에서 약 13초가량의 영상을 기준으로 네 가지 모델을 탐지시켜본 결과 YOLOv5s 모델의 소요 시간은 3분 32초가량 걸렸고 총 18명의 탑승 인원 중 최대 9명을 탐지하였고, YOLOv5m 모델은 4분 15초의 시간이 걸렸고 최대 탐지 인원은 13명이었다. YOLOv5l 모델은 7분 47초의 시간이 걸렸고 최대 탐지 인원은 14명이었고, YOLOv5x 모델은 21분 23초의 소요 시간이 걸렸으며 15명의 사람이 탐지되었다.

해당 실험 결과로 동영상으로 탐지하여 사람을 찾는 것은 부적합하다고 판단하여, 동영상인 아닌 캡처를 하여 해당 이미지를 탐색하여 사람을 탐지하는 방식이 더 빠르고 적합하다고 판단하여 이미지를 탐지해 보았다.

이에 해당 이미지를 0.4의 정확도를 가지고 모델별로 탐지를 시켜본 결과 YOLOv5s 모델의 경우 6명을 탐지하였고, YOLOv5m 모델은 9명, YOLOv5l은 10명, YOLOv5x 모델은 10명의 사람을 탐지하였다.



※실험 이미지



※YOLOv5s (conf=0.4)



※YOLOv5m (conf=0.4)



※YOLOv5l (conf=0.4)



※YOLOv5x (conf=0.4)

모델별로 같은 이미지를 가지고 실험해본 결과 large와 x-large의 탐지하는 인원의 수가 동일하여 더 빠르게 결과를 보여줄 수 있는 large 모델이 해당 프로젝트로 진행하는데 가장 적합하다고 판단하였다.

이후 조금 더 많은 인원을 찾아내기 위해 동일한 이미지로 정확도를 조절해 가며 실험해 본 결과 정확도가 낮을수록 많은 인원을 탐지하지만, 사람이 아닌 것을 탐지하거나 겹쳐서 검출하는 경우가 있었고 반대로 정확도를 높게 지정하여 실험해 본 결과 사람을 완벽하게 탐지하지만 높은 정확도로 인해 뒤에 있거나 정면을 바라보지 않고 신체 일부만 나왔을 때 탐지를 못하는 모습을 보였다. 그중 가장 적절한 정확도는 0.3이라고 판단하였다. 이를 기준으로 YOLOv5l 와 정확도 0.3을 이용하여 다른 이미지에도 실험해본 결과

no	confidence	small	medium	large	X-large
1	0.3	8/16(0.5)	12/16(0.75)	11/16(0.69)	13/16(0.81)
2	0.3	7/18(0.39)	8/18(0.45)	6/18(0.3)	11/18(0.61)
3	0.3	12/13(0.92)	11/13(0.85)	11/13(0.85)	14/13(1.07)

4	0.3	7/9(0.78)	4/9(0.45)	6/9(0.66)	5/9(0.56)
5	0.3	10/21(0.48)	13/21(0.62)	12/21(0.57)	13/21(0.62)

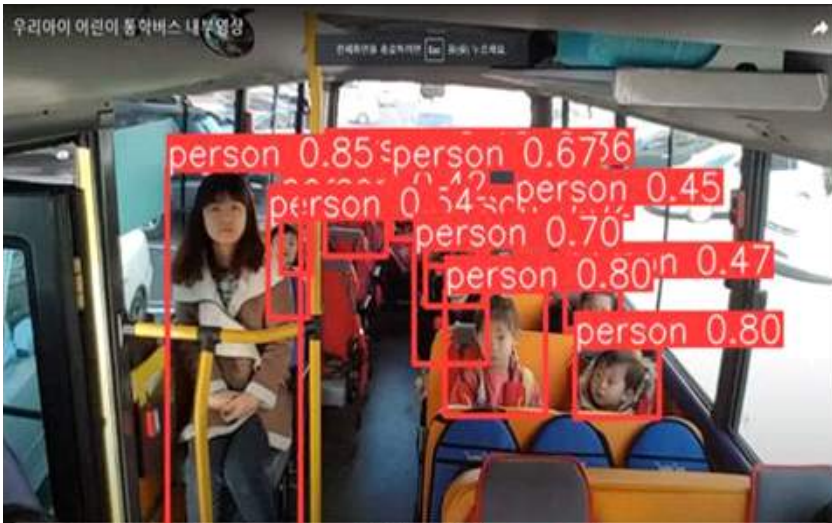
다음과 같은 결과값을 얻게 되었다. x-large 모델이 large 모델보다 검출률이 더 높게 나오는 것을 알 수 있었다.

따라서 large 모델이 아닌 x-large 모델을 사용하기로 하였으며 모델 크기에 따른 소요 시간은 동영상인 아닌 이미지를 탐지하는 것이기에 체감할 정도로 크게 차이가 나지 않기 때문에 x-large 모델을 사용하여 진행하는 것이 해당 프로젝트를 진행하는데 가장 적합하다는 결과를 도출하였다.

이후 다양한 캡처 이미지를 이용하여 x-large 모델로 실험해 본 결과 가장 정확하고 많은 인원을 탐지하여 최선의 모델임을 알 수 있게 되었다.

해당 이미지들은 YOLOv5x 모델로 실행시켜 나온 결과들이다.





파이썬 코드

colab을 통해 사용하는 경우

YOLOv5 model download

!git clone <https://github.com/ultralytics/yolov5.git>

yolov5 폴더로 이동

%cd yolov5

필수 패키지 설치

%pip install -qr requirements.txt

환경 확인 // gpu

from yolov5 import utils

display = utils.notebook_init()

import torch

model = torch.hub.load('ultralytics/yolov5', 'yolov5l')

주피터 노트북을 사용하는 경우

1) 이미지 탐지

YOLOv5 model download

#C:\Users\kimhy\.cache\torch\hub\ultralytics_yolov5_master /// yolov5 저장 위치

!git clone <https://github.com/ultralytics/yolov5.git>

yolov5 폴더로 이동

%cd yolov5

필수 패키지 설치

#pip install -qr requirements.txt

Packages

import cv2

import torch

Model download & load - pretrained model(COCO dataset)

model = torch.hub.load('ultralytics/yolov5', 'yolov5l')

Model settings

```

model.conf = 0.2
model.iou = 0.45
model.classes = [0]
model.multi_label = False
model.max_det = 1000

### Source
img = "C:\\Users\\kimhy\\Desktop\\image_analysis\\image5-3\\buskid.jpg "

### Image detection
results = model(img)

### 결과 저장
results.save("YOLOv5_out")

### 인식 결과
results.pandas().xyxy[0]

### 결과 영상 출력
results.display(render=True)
result_img = results.imgs[0]
show_img = cv2.resize(result_img,dsize=(0,0),fx=1,fy=1,
                      interpolation=cv2.INTER_LINEAR)
cv2.imshow('',show_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

2) 동영상 탐지

```

### YOLOv5 model download
#C:\Users\kimhy\.cache\torch\hub\ultralytics_yolov5_master /// yolov5 저장 위치
!git clone https://github.com/ultralytics/yolov5.git

### yolov5 폴더로 이동
%cd yolov5

### 필수 패키지 설치
#pip install -qr requirements.txt

### Packages
import cv2

```

```

import torch

### Model download & load - pretrained model(COCO dataset)
model = torch.hub.load('ultralytics/yolov5', 'yolov5l')

### Model settings
model.conf = 0.2
model.iou = 0.45
model.classes = [0]
model.multi_label = False
model.max_det = 1000

import warnings
warnings.filterwarnings('ignore')

#원본영상
file = "C:\\Users\\kimhy\\Desktop\\image_analysis\\image5-3\\file\\bus_video.mp4"
cap=cv2.VideoCapture(file)

while cv2.waitKey(20)<0:
    ### 반복 재생 설정
    if True & (cap.get(cv2.CAP_PROP_POS_FRAMES) ==
cap.get(cv2.CAP_PROP_FRAME_COUNT)):
        cap.set(cv2.CAP_PROP_POS_FRAMES,0)

    ### Capture frame-by-frame
    ret,frame=cap.read()
    if ret:
        ### Frame 출력
        cv2.imshow("VideFrame",frame)
    else:
        print("Can't recieve frame.Exiting...")
        break

### when everything done, release the capture
cap.release()
cv2.destroyAllWindows()

### 영상 연결
img = "C:\\Users\\kimhy\\Desktop\\image_analysis\\image5-3\\file\\bus_video.mp4"

```

```

cap = cv2.VideoCapture(img)

### 코덱
fourcc = cv2.VideoWriter_fourcc(*'FMP4')

### frame width, height, fps(초당 frame 수)
width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
fps = cap.get(cv2.CAP_PROP_FPS)

### 영상 저장 설정
out = cv2.VideoWriter('G19_19s_out.mp4',fourcc,fps,(int(width),int(height)))

### 영상 출력
while cv2.waitKey(1) < 0:
    ### Capture frame-by-frame
    ret, frame = cap.read()

    ### 영상을 읽지 못하면 종료
    if not ret:
        print("Can't recieve frame. Exiting..")
        break

    ### 객체 인식
    results =model(frame)
    results.display(render = True)
    frame = results.imgs[0]

    ### Frame 출력
    cv2.imshow("Video Frame",frame)

### When everythig done, release the capture
cap.release()
out.release()
cv2.destroyAllWindows()

warnings.filterwarnings(action='default')

### 결과 영상 출력

```

```
results.display(render=True)
result_img = results.imgs[0]
show_img = cv2.resize(result_img,dsize=(0,0),fx=1,fy=1,
                      interpolation=cv2.INTER_LINEAR)
cv2.imshow('',show_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


활용방안

먼저 앞선 주제에서 알림을 통해 운행 책임자에게 알리는 것을 어플등과 연계를 통해 조금 더 쉽게 알림을 줄 수 있도록 하고 확인했다는 것을 사진을 찍어 차량 내부 사진을 업로드하여 완료하는 방식으로 해당 사진에서 사람이 있는지 다시 한번 탐지하여 이중으로 확인을 하도록 한다.

코로나로 인원 제한이 있는 지금과 같은 시기에 매장이나 특정 장소에 몇 명이 있는지 확인하는 데에도 활용할 수 있을 것이다.

관련 참고 자료

<https://github.com/ultralytics/yolov5>

<https://ultralytics.com/>

<https://bigdata-analyst.tistory.com/195?category=883085>

<https://www.youtube.com/watch?v=T0DO1C8uYP8>

<https://public.roboflow.com/>

<https://cocodataset.org/#home>

<https://www.noheatstroke.org/index.htm>

http://news.heraldcorp.com/culture/view.php?ud=201807180929174580929_1&md=_BL

[https://www.chosun.com/international/international_general/2021/09/11/AJJU4H3XFGXZENJUDECA
G4RBE/](https://www.chosun.com/international/international_general/2021/09/11/AJJU4H3XFGXZENJUDECA
G4RBE/)