임베디드응용 및 실습

최종보고서

이름	김한별
학번	2022180034
학과	정보통신공학전공
제출일	2024.12.11

1. 서론

이번 프로젝트의 목표는 자율주행 트랙을 한바퀴 돌면서 지정된 물체가 감지되면, 자동차가 긴급 정지되고, 지정된 물체가 사라지면 다시 자율 주행하는 것이다. 이 프로젝트를 수행하기 위해 다음과 같은 과정을 거쳤다.

1) 자율주행 트랙 데이터셋 수집
2) 수집한 데이터셋으로 자율주행 학습
3) 물체 감지 학습
4) 긴급 제동할 물체 지정
5) 물체가 감지되면 긴급제동
6) 긴급 제동시 LED와 부저 기능 추가

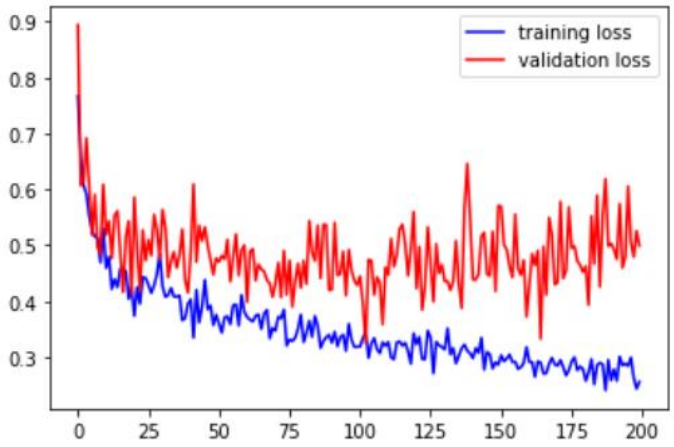
2. 자율주행 구현

1) 자율주행 트랙 데이터셋 수집

총 **1,234장의 데이터셋**을 수집하였다. 직진, 좌회전, 우회전의 비율은 대략 **3:1:1**이다. 키보드의 방향키를 눌러서 데이터셋을 모으도록 하였는데, 위쪽 방향키는 0(직진), 왼쪽 방향키는 1(좌회전), 오른쪽 방향키는 2(우회전)이다. 키보드를 누를 때마다 사진이 찍히고 파일명을 0, 1, 2로 끝마치며 사진이 저장되게 된다. 자동차가 한바퀴를 도는 동안 대략 0.5초 간격으로

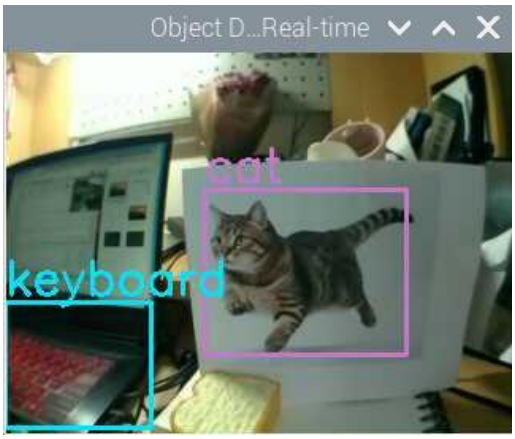
키보드를 눌러 사진을 찍도록 하였다. 코너의 경우에는 회전 1번, 직진 1번을 0.3초 간격으로 눌러 데이터셋을 모았다. 그 결과로, 직진 데이터셋을 회전보다 더 많이 모으게 되었다.

2) 수집한 데이터셋으로 자율주행 학습



처음 주어진 학습 코드에선, 배치사이즈가 각각 train_bSize = 20, valid_bSize = 100 이었다. 하지만 메모리 용량이 부족해 커널이 자꾸 죽는 문제가 발생하여, 배치 사이즈를 줄이도록 하였다. 최종적으로 학습한 배치 사이즈는 train_bSize = 20에 valid_bSize = 10이다. 더불어 배치 사이즈를 줄이면, 학습이 제대로 되지 않을 것 같아 Epochs의 학습 수를 100에서 300으로 늘렸다. 결과적으로 **손실률이 0.3** 정도 도출되었다. 손실률을 더 낮추기 위해 학습수를 600번으로도 늘려보았는데, 과적합 문제가 발생해 오히려 자율주행이 전보다 되지 않았다. 여러 번 학습을 돌리고, 직접 테스트 해본 결과 **valid_bSize가 10이며 Epochs를 300번** 학습 시켰을 때가 더 정확해 이 모델로 자율주행을 구현하였다.

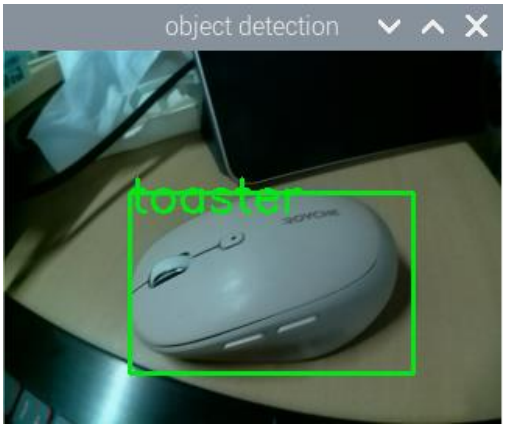
3) 물체 감지 학습



먼저, 주어진 얼굴검출 코드를 수정해 실시간으로 객체를 탐지할 수 있도록 하였다. 물체 감지는 사람이 제일 잘 탐지 되었고, 그 다음으로는 labtop, chair 등의 큰 물체의 경우가 제일 탐지가 빨랐다.

4) 긴급 제동할 물체 지정

처음 긴급제동을 위해서 물체를 지정할 때 물병이나 마우스 등으로 시도해보았다. 하지만 물체 크기가 너무 작아 실시간으로 감지가 너무 느렸고 마우스의 경우 'mouse'로 잘 인식하다가, 'toaster'로 인식이 되는 오류가 빈번히 발생하였다.

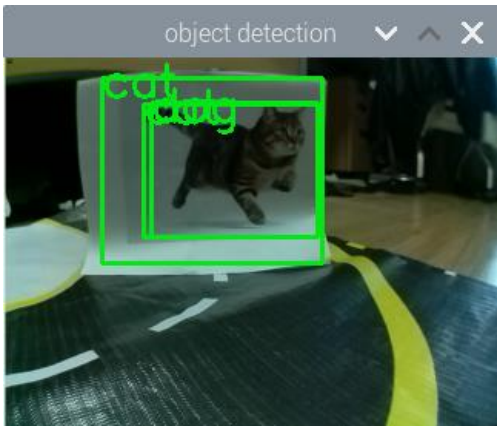


따라서 인식이 그나마 빠르고 정확한 동물로 선택하였고, 그 중, **고양이**로 선택하게 되었다.

5) 물체 감지 후 긴급제동

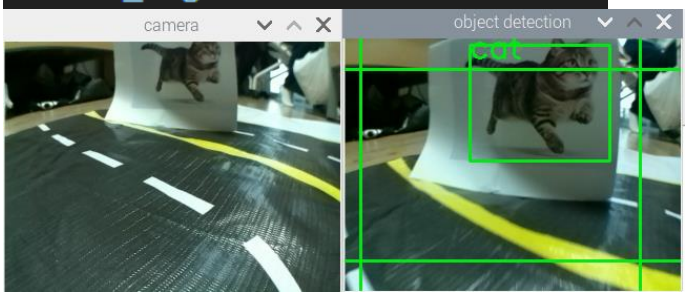
```
if class_name == 'cat' or class_name == 'dog':
    print("stop fpr detected")
    car.motor_stop()
    emergency_stop()
```

주어진 코드 내에서 if문을 수정해 **'cat'과 'dog'**를 탐지하면 모터가 정지할 수 있도록 코드를 작성하였다. 처음엔 'cat'만 감지하도록 하였지만, 모델이 고양이를 'cat'와 'dog' 둘 다로 인식하여, 'dog'을 감지하여도 멈출 수 있도록 수정하였다.



또한 객체 탐지 카메라의 중앙에서 인식할 수 있도록 고양이 사진을 프린트해 받침대 앞에 붙여서 고양이 사진을 세울 수 있도록 하였다. 결과적으로, 멀리서도 'cat' detection이 잘 이루어졌다.

```
cnt = 0
num_skip = 10 #지연시간 조절
mul_f = 1
size_img = 200
```



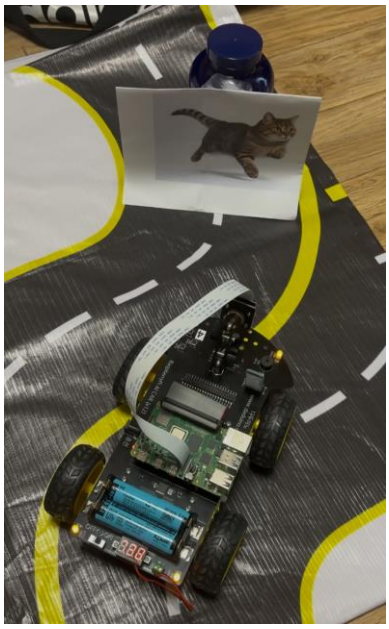
긴급제동 부분을 구현하는데 있어 object detection 카메라의 지연시간이 너무 길어 정작 detection을 했지만 제동이 되지 않거나, detection이 멈추었는데도 다시 출발하지 않는 오류가 발생하였다. 모든 문제는 자율주행의 기본 camera와 object detection의 프레임 딜레이가 차이가 커 발생한 것으로 판단해 최대한 object detection 카메라의 딜레이를 낮추도록 시도하였다. 하여 **size_img를 기존 300에서 200**으로 낮추었고, **num_skip을 10**으로 늘려 객체 감지 수행 주기가 느려지게 만들어 최대한 딜레이 격차를 줄이고자 하였다. 그 결과로 대략 2초에 한번 프레임이 움직여 detection을 수행하고 'cat'이나 'dog' 탐지가 발생하면 모터 주행을 멈출수 있도록 하였다.

6) 긴급 제동시 LED와 부저 기능 추가

```python
# GPIO 핀 설정
LED_PIN1 = 20  # LED가 연결된 핀 번호
LED_PIN2 = 21
LED_PIN3 = 16
LED_PIN4 = 26
BUZZER = 12

# GPIO 초기화
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(LED_PIN1, GPIO.OUT)
GPIO.setup(LED_PIN2, GPIO.OUT)
GPIO.setup(LED_PIN3, GPIO.OUT)
GPIO.setup(LED_PIN4, GPIO.OUT)
GPIO.setup(BUZZER, GPIO.OUT)
p = GPIO.PWM(BUZZER, 261)

def emergency_stop():
    for _ in range(5):  # 5번 점멸
            GPIO.output(LED_PIN1, GPIO.HIGH)
            GPIO.output(LED_PIN2, GPIO.HIGH)
            GPIO.output(LED_PIN3, GPIO.HIGH)
            GPIO.output(LED_PIN4, GPIO.HIGH)
            p.start(50)
            p.ChangeFrequency(261)
            time.sleep(0.5)  # 0.5초 켜기
            GPIO.output(LED_PIN1, GPIO.LOW)
            GPIO.output(LED_PIN2, GPIO.LOW)
            GPIO.output(LED_PIN3, GPIO.LOW)
            GPIO.output(LED_PIN4, GPIO.LOW)
            p.stop()
            time.sleep(0.5)  # 0.5초 끄기
            print("Emergency alert: Activating LED")
```



물체가 감지되면 4개의 LED와 부저가 0.5초 간격으로 점멸하고, 경고음이 송출될 수 있도록 코드를 작성하였다.

```python
if class_name == 'cat' or class_name == 'dog':
    print("stop fpr detected")
    car.motor_stop()
    emergency_stop()
```

긴급제동 LED/부저 기능을 함수로 작성해 자율주행 함수 내의 긴급제동 부분에서 함수 호출 방식으로 기능하도록 작성하였다.

7) 전체 코드

```
import cv2 as cv
import numpy as np
import threading, time
import SDcar
import sys
import time
from tensorflow.keras.models import load_model
import RPi.GPIO as GPIO

classNames = {
    0: 'background', 1: 'person', 2: 'bicycle', 3: 'car', 4: 'motorbike', 5:
'aeroplane',
    6: 'bus', 7: 'train', 8: 'truck', 9: 'boat', 10: 'traffic light', 11:
'fire hydrant',
    12: 'N/A', 13: 'stop sign', 14: 'parking meter', 15: 'bench', 16: 'bird',
17: 'cat',
    18: 'dog', 19: 'horse', 20: 'sheep', 21: 'cow', 22: 'elephant', 23:
'bear',
    24: 'zebra', 25: 'giraffe', 26: 'N/A', 27: 'backpack', 28: 'umbrella', 29:
'N/A',
    30: 'handbag', 31: 'tie', 32: 'suitcase', 33: 'frisbee', 34: 'skis', 35:
'snowboard',
    36: 'sports ball', 37: 'kite', 38: 'baseball bat', 39: 'baseball glove',
40: 'skateboard', 41: 'surfboard',
    42: 'tennis racket', 43: 'bottle', 44: 'wine glass', 45: 'cup', 46:
'fork', 47: 'knife',
    48: 'spoon', 49: 'bowl', 50: 'banana', 51: 'apple', 52: 'sandwich', 53:
'orange',
    54: 'broccoli', 55: 'carrot', 56: 'hot dog', 57: 'pizza', 58: 'donut', 59:
'cake',
    60: 'chair', 61: 'couch', 62: 'potted plant', 63: 'bed', 64: 'dining
table', 65: 'toilet',
    66: 'tv', 67: 'laptop', 68: 'mouse', 69: 'remote', 70: 'keyboard', 71:
'cell phone',
    72: 'microwave', 73: 'oven', 74: 'toaster', 75: 'sink', 76:
'refrigerator', 77: 'book',
    78: 'clock', 79: 'vase', 80: 'scissors', 81: 'teddy bear', 82: 'hair
drier', 83: 'toothbrush'
}



def id_class_name(class_id, classes) :
    for key, value in classes.items():
        if class_id == key:
            return  value
```

```
speed = 80
epsilon = 0.0001

def object_detection_thread():
    global frame, object_detection_enabled, class_name

    model = cv.dnn.readNet('/home/pi/hello-git/log-
git/week13/frozen_inference_graph.pb',
                                    '/home/pi/hello-git/log-
git/week13/ssd_mobilenet_v2_coco_2018_03_29.pbtxt')
    cnt = 0
    num_skip = 10 #지연시간 조절
    mul_f = 1
    size_img = 200
    # COLORS 배열 정의



    while True:
        cnt += 1
        if object_detection_enabled is True and cnt % num_skip == 0:

            print('cnt', cnt)

            lock.acquire()
            imagednn = frame.copy()
            #lock.release()
            print('frame id', id(frame))
            #print('image dnn id', id(imagednn))
            lock.release()

            image_height, image_width, _ = imagednn.shape

            #print('size img : ',size_img)

            starttime = time.time()
            model.setInput(cv.dnn.blobFromImage(imagednn, size=(size_img,
size_img), swapRB=True))
            output = model.forward()

            class_name = -1
            for detection in output[0, 0, :, :] :
                confidence = detection[2]
                if confidence > .3:
                    class_id = detection[1]
                    class_name = id_class_name(class_id, classNames)
```

```
                    print(str(str(class_id) + " " + str(detection[2]) + " " +
class_name))

                    box_x = detection[3] * image_width
                    box_y = detection[4] * image_height
                    box_width = detection[5] * image_width
                    box_height = detection[6] * image_height


                    cv.rectangle(imagednn, (int(box_x), int(box_y)),
(int(box_width), int(box_height)), (23, 230, 5), thickness=2)
                    cv.putText(imagednn, class_name, (int(box_x),
int(box_y+.05*image_height)), cv.FONT_HERSHEY_SIMPLEX, 1, (23, 230,5 ), 2)

            elapsed = time.time() - starttime

            cv.imshow('object detection', imagednn)

        if is_running is False:
            break
        if cnt >= 1000000:
            cnt = 0


def key_cmd(which_key):
    #print('which_key', which_key)
    is_exit = False
    global object_detection_enabled
    global enable_AIdrive
    if which_key & 0xFF == 82:
        print('up')
        car.motor_go(speed)
    elif which_key & 0xFF == 84:
        print('down')
        car.motor_back(speed)
    elif which_key & 0xFF == 81:
        print('left')
        car.motor_left(30)
    elif which_key & 0xFF == 83:
        print('right')
        car.motor_right(30)
    elif which_key & 0xFF == 32:
        car.motor_stop()
        enable_AIdrive = False
        print('stop')
    elif which_key & 0xFF == ord('q'):
        car.motor_stop()
        print('exit')
```

```
        enable_AIdrive = False
        is_exit = True
        print('enable_AIdrive: ', enable_AIdrive)
    elif which_key & 0xFF == ord('e'):
        enable_AIdrive = True
        print('enable_AIdrive: ', enable_AIdrive)
    elif which_key & 0xFF == ord('w'):
        enable_AIdrive = False
        car.motor_stop()
        print('enable_AIdrive 2: ', enable_AIdrive)
    elif which_key & 0xFF == ord('t'):  # 't' 키 입력 시 객체 감지 활성화
        object_detection_enabled = True
        print("Object detection enabled")
    elif which_key & 0xFF == ord('r'):  # 'r' 키 입력 시 객체 감지 비활성화
        object_detection_enabled = False
        print("Object detection disabled")
    return is_exit


# GPIO 핀 설정
LED_PIN1 = 20  # LED가 연결된 핀 번호
LED_PIN2 = 21
LED_PIN3 = 16
LED_PIN4 = 26
BUZZER = 12

# GPIO 초기화
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(LED_PIN1, GPIO.OUT)
GPIO.setup(LED_PIN2, GPIO.OUT)
GPIO.setup(LED_PIN3, GPIO.OUT)
GPIO.setup(LED_PIN4, GPIO.OUT)
GPIO.setup(BUZZER, GPIO.OUT)
p = GPIO.PWM(BUZZER, 261)

def emergency_stop():
    for _ in range(5):  # 5번 점멸
            GPIO.output(LED_PIN1, GPIO.HIGH)
            GPIO.output(LED_PIN2, GPIO.HIGH)
            GPIO.output(LED_PIN3, GPIO.HIGH)
            GPIO.output(LED_PIN4, GPIO.HIGH)
            p.start(50)
            p.ChangeFrequency(261)
            time.sleep(0.5)  # 0.5초 켜기
            GPIO.output(LED_PIN1, GPIO.LOW)
            GPIO.output(LED_PIN2, GPIO.LOW)
            GPIO.output(LED_PIN3, GPIO.LOW)
```

```
            GPIO.output(LED_PIN4, GPIO.LOW)
            p.stop()
            time.sleep(0.5)  # 0.5초 끄기
            print("Emergency alert: Activating LED")


def drive_AI(img):
    #print('id', id(model))
    global class_name
    img = np.expand_dims(img, 0)
    res = model.predict(img)[0]
    #print('res', res)
    steering_angle = np.argmax(np.array(res))
    #print('steering_angle', steering_angle)

    print('class name in drive AI', class_name)

    if class_name == 'cat' or class_name == 'dog':
        print("stop fpr detected")
        car.motor_stop()
        emergency_stop()

    elif steering_angle == 0:
        print("go")
        speedSet = 40
        car.motor_go(speedSet)
    elif steering_angle == 1:
        print("left")
        speedSet = 20
        car.motor_left(speedSet)
    elif steering_angle == 2:
        print("right")
        speedSet = 20
        car.motor_right(speedSet)
    else:
        print("This cannot be entered")


def main():
    global frame
    try:
        while( camera.isOpened() ):
            starttime = time.time()
            lock.acquire()
            ret, frame = camera.read()
            frame = cv.flip(frame,-1)
            lock.release()
```

```
            cv.imshow('camera',frame)
            # image processing start here
            crop_img = frame[int(v_y/2):,:]
            crop_img = cv.resize(crop_img, (200, 66))
            #show_grid(crop_img)
            #cv.imshow('crop_img ', cv.resize(crop_img, dsize=(0,0), fx=2,
fy=2))

            if enable_AIdrive == True:
                starttime = time.time()
                drive_AI(crop_img)
                elapsed = time.time() - starttime

            # image processing end here
            is_exit = False
            which_key = cv.waitKey(20)
            if which_key > 0:
                is_exit = key_cmd(which_key)
            if is_exit is True:
                cv.destroyAllWindows()
                break

    except Exception as e:
        exception_type, exception_object, exception_traceback = sys.exc_info()
        filename = exception_traceback.tb_frame.f_code.co_filename
        line_number = exception_traceback.tb_lineno

        print("Exception type: ", exception_type)
        print("File name: ", filename)
        print("Line number: ", line_number)
        global is_running
        is_running = False

if __name__ == '__main__':

    model_path = '/home/pi/hello-git/log-
git/week13/lane_navigation_20241207_0900.h5'
    model = load_model(model_path)
    '''print('id', id(model))
    print(model.summary())'''

    model_obj = cv.dnn.readNet('/home/pi/hello-git/log-
git/week13/frozen_inference_graph.pb',
                               '/home/pi/hello-git/log-
git/week13/ssd_mobilenet_v2_coco_2018_03_29.pbtxt')

    #test_fun(model)
    W = 320
```

```python
H = 240 # 바꿔도됨
camera = cv.VideoCapture(0)
camera.set(cv.CAP_PROP_FRAME_WIDTH,W)
camera.set(cv.CAP_PROP_FRAME_HEIGHT,H)
camera.set(cv.CAP_PROP_FPS, 30)

v_x = W
v_y = H
v_x_grid = [int(v_x*i/10) for i in range(1, 10)]
moment = np.array([0, 0, 0])

_, frame = camera.read()

lock = threading.Lock()

t_task1 = threading.Thread(target = object_detection_thread)
t_task1.start()

car = SDcar.Drive()

is_running = True
enable_AIdrive = False
object_detection_enabled = False
class_name = -1
try :
    main()
finally:

    is_running = False
    t_task1.join()

    camera.release()
    cv.destroyAllWindows()

    car.clean_GPIO()
    print('end vis')
```

8) 구현 시스템 개요

- **하드웨어 구성**

  - 라즈베리파이
  - 카메라(전면 환경 영상 획득)
  - 모터 (전/후진 및 좌/우회전 구동)
  - LED 및 부저(비상 상황 경고)

- **소프트웨어 구성**

  - Python 기반 OpenCV 라이브러리(영상 처리)
  - TensorFlow Keras 모델 로딩 (사전 학습된 차선 인식 모델)
  - OpenCV DNN 모듈을 통한 객체 탐지(SSD MobileNet V2 COCO 모델 활용)
  - RPi.GPIO 라이브러리를 통한 GPIO 제어 (모터, LED, 부저)

- 알고리즘 개요

  - id_class_name : 객체 인식 모델에서 얻은 class_id를 통해 classNames에서 실제 클래스 이름 문자열 반환
  - object_detection_thread : 별도의 스레드에서 동작하며, frame에서 객체 인식 수행. 신뢰도가 0.3 이상이면 해당 객체를 class_name 변수로 업데이트하고 영상에 박스를 그림.
  - Key_cmd : 사용자 키보드 입력에 따라 주행 방향 수동 제어/ai 주행 모드/객체 인식 모드
    - 주행방향 수동 제어 : up/down/left/right/space 키
    - AI 주행 모드 : e 활성화 / w 비활성화
    - 객체 인식 모드 : t 활성화 / r 비활성화
    - q : 프로그램 종료
  - emergency_stop : 5회 반복으로 LED 점멸, 부저 송출

- drive_AI : AI 모드 활성시 함수 호출, 차선 인식 모델의 결과를 바탕으로 차량 주행 방향 제어, 객체 인식 결과에 따라 특정 물체가 있을 경우 긴급 정지 수행
    - 출력값이 0일 경우 직진
    - 출력값이 1일 경우 좌회전
    - 출력값이 2일 경우 우회전
- Main
    - 차선 인식 모델(Keras) 로드
    - 객체 인식 모델(OpenCV DNN) 로드
    - 카메라 초기화 (해상도, FPS 설정)
    - 객체 인식 스레드 시작
    - SDcar.Drive() 객체를 통해 모터 제어 장치 초기화
    - 전역변수 초기화 후 main() 호출
    - main() 루프 종료 후 카메라, 윈도우, GPIO 정리 및 프로그램 종료

3. 결론

이번 프로젝트를 통해 라즈베리파이를 활용한 자율주행 및 객체 감지 시스템 구현을 목표로 하여 다양한 과정을 거쳐 성공적으로 자율주행을 수행할 수 있었다.

먼저, 자율주행 트랙 데이터셋을 수집하는 데 있어 사진을 찍는 환경이 중요하단 것을 깨달았다. 기숙사에서도 사진을 찍어보고, 테스트를 할 강의실에서도 사진을 찍어보았는데 확실히 실제 테스트 환경에서 데이터셋을 모아 학습한 것이 더 큰 정확도를 보였다.

또한 이를 학습해 모델을 구축하는 과정에서 데이터 균형과 학습 파라미터 최적화의 중요성을 인식하였다. 결국엔 0.3 정도의 손실률을 달성할 수 있었지만, 메모리가 큰 CPU 환경에서 커널이 죽지 않는다는 보장을 가지고 대량의 데이터셋으로, 1000번의 횟수 이상으로 학습을 돌려 정확도를 100퍼센트까지 끌어보고 싶다는 생각을 하였다.

긴급제동 부분에 있어서, 카메라 딜레이가 커 바로바로 객체를 탐지하지 못하는게 아쉬움이 컸다. 특히 작고 복잡한 형태의 물체 인식에서 발생한 오류를 완전히 해결하지 못한 점은 개선이 필요하다. 학습 모델의 용량이 크고, 카메라 해상도가 높아도 바로바로 탐지를 할 수 있는 환경에서 이러한 한계를 극복해보고 싶다.

마지막으로 프로젝트를 실행하면서 여러 코드를 올바르게 통합하는 과정이 어려움을 느꼈다. 특히, 코드에 오류가 발생했을 때 수정 과정에서 코드의 길이가 길다 보니 윗줄에서 수정한 부분을 밑줄에서는 반영하지 못하거나, 기존에 정상적으로 작동하던 코드가 반복적인 수정 과정에서 오히려 오류를 유발하는 등의 문제가 자주 발생했다. 이러한 경험을 통해 코드를 바로 수정하기보다는 수정 전 백업을 반드시 해두는 것의 중요성을 깨달았다. 또한, 코드가 길고 복잡할수록 처음부터 끝까지 구조를 차근차근 살펴보며 전체적인 흐름을 파악하고 코드 내에서 작은 오타나 불일치를 잡는 것이 필요하다는 점을 배웠다.