분류번호 : 2023-2-4106-07

졸업작품 최종보고서

딥러닝을 활용한 자동 재활용 쓰레기 분류기

지도교수 : 도 현 락 교수님

제 출 일 : 2023년 11월 10일

제 작 자 : 17101501 이 윤 혁

17101529 정 지 혜

18101552 정 민 규

서울과학기술대학교 전자IT미디어공학과

목 차

요 약

제 목 : 딥러닝을 활용한 자동 재활용 쓰레기 분류기

이 보고서는 프로젝트의 설계 동기부터 시작해 최종적인 목표를 이루기 위한 실시간 객체인식과 음성 데이터 분석을 진행한 과정과 동작부를 어떻게 구현했는지에 대해 다룬다.

우리나라뿐만 아니라 세계적으로 쓰레기 문제가 심각하다. 현재 우리나라는 쓰레기 수거 후 재활용 가능 쓰레기를 분류하고 분류되지 않은 나머지는 매립하거나 소각하고 있다. 다른 나라들에 비해 분리수거 체계가 잘 되어있다고 말하곤 하지만 분리수거 된 쓰레기 중에서도 재활용되는 것들은 절반도 채 되지 않는 실정이다. 이러한 쓰레기 문제에 조금이라도 도움이 되고자 이 프로젝트를 설계하게 되었다.

이 프로젝트는 기본적으로 실시간 객체인식과 음성 데이터 분석을 이용한다. 아두이노와 USB 카메라, USB 마이크 그리고 컴퓨터를 연결해 사용한다. 또한 쓰레기를 플라스틱, 캔, 유리 세 재질로 나누고 이 때문에 2개의 서보모터도 함께 사용했다.

실시간 객체인식을 위해 각종 쓰레기 사진을 수집했다. 하지만 실제로 찍은 쓰레기 사진은 데이터 셋을 구축하기에 충분하지 않아 인터넷의 여러 쓰레기 사진을 다운 받고 roboflow 사이트에 사진을 업로드하여 학습하고 데이터 셋을 구축하였다. 이후 YOLOv5를 이용해 딥러닝을 진행하였다.
음성 데이터 분석을 위해 쓰레기를 떨어뜨릴 때 나는 소리를 수집했다. 음성 데이터는 어느 판에 떨어뜨리느냐에 따라 소리가 확연히 달라지기 때문에 정해둔 판에 쓰레기를 다각도로 떨어뜨리면서 소리를 녹음하고 수집하였다. 이후 MFCC를 추출하여 학습하여 개발한 CNN모델을 이용해 음성 데이터를 분석하였다.

플라스틱, 캔, 유리 세 재질로 구분되기 때문에 동작부에는 두 개의 서보모터와 회전판이 존재한다. 제일 위, 적외선 센서에 쓰레기를 인식시키는 것이 시작이다. 첫 번째 회전판은 재질을 구분하기 위한 분류판으로, 모든 판단이 진행되는 곳이다. 먼저 그 위에 놓인 쓰레기를 USB 카메라(웹캠)를 통해 실시간 객체인식으로 재질을 분류하고 신뢰도를 추출한다. 그 신뢰도가 0.7 이상인 경우 객체 인식만으로 재질을 판단하여 분류하고 신뢰도가 0.7 미만인 경우에 음성 데이터 분석을 이용해 재질을 판단하여 분류한다. 이는 직접 떨어뜨리는 과정에서 난 소리를 USB 마이크를 통해 학습된 데이터를 거쳐 재질을 판단한다.

실시간 객체 인식과 음성 데이터 분석, 두 가지를 이용하기 때문에 정확도가 높을 것으로 예상되며 이 분류기가 사람들에게 편리함을 주고 쓰레기 문제를 해결하는 데 도움이 될 수 있기를 바란다.

표 목 차

그 림 목 차

사 진 목 차

Ⅰ. 개 요

1-1. 작품의 목표

사용자가 따로 분리수거할 필요 없이, 학습된 데이터를 바탕으로 실시간 객체 인식과 음성 데이터 분석을 활용해 쓰레기를 자동으로 분류한다.
분리수거가 필요한 쓰레기를 분류판 위에 떨어뜨린 후, 객체 인식과 음성 데이터 분석을 통해 자동으로 물체의 재질을 판단하고 플라스틱과 캔, 유리로 분류할 수 있도록 한다.
이를 공공장소, 아파트나 공동주택의 분리수거장, 쓰레기 처리장 등 다양한 곳에서 사용할 수 있도록 한다.

1-2. 제작 배경

2019년 말부터 시작되어 현재까지 이어지고 있는 COVID-19. 이로 인해 배달 음식 주문량이 증가했고 그에 따라 플라스틱 사용량이 늘어나 '코로나 트래시(Trash)' 라는 신조어가 생기기도 했다. 또한 약 5년 전에 비해 플라스틱류 쓰레기의 발생량은 2배 이상 증가했다.
현재 재활용이 가능한 쓰레기도 일반 쓰레기와 함께 혼합 배출되는 경우가 허다하고, 재활용으로 분류되어 들어온 쓰레기도 30~40%만 재활용되고 있는 상황이다. 면적이 작은 우리나라는 더 이상 쓰레기를 매립할 공간도 부족하다.
이러한 시점에서 자동으로 분리수거를 해주는 분류기가 있다면 조금이라도 쓰레기 문제를 해결할 수 있다고 생각되어 제작하게 되었다. 실시간 객체 인식과 음성 데이터의 MFCC를 추출하여 학습하여 개발한 CNN모델을 이용해 물체의 재질을 판단하고 분류할 수 있다는 점은 활용 가치가 매우 높을 것으로 예상된다.

1-3. 설계 진행일정

작품의 설계는 크게 1학기와 2학기로 나누어 진행하였다. 1학기에는 아이디어 회의 및 프로젝트 주제를 결정하고 관련 지식을 습득한 뒤 자료를 수집하여 각종 데이터 학습을 목표로, 2학기에는 더 많은 데이터 학습과 동작부 구현을 목표로 진행하기로 하였다.
실시간 객체인식에 대한 데이터는 많은 편이었지만 음성 데이터는 비교적 적은 편이라 하나씩 녹음해야 했기 때문에 예상보다 시간이 오래 소요되었다.

	연구개발내용	추진일정								
		1학기				여름방학		2학기		
		3월	4월	5월	6월	7월	8월	9월	10월	11월
1	아이디어 회의 및 프로젝트 주제 결정									
2	프로젝트 설계 관련 지식 학습									
3	관련 자료 수집 및 기본 코드 학습									
4	객체인식 및 음성 데이터 학습									
5	동작부 구현									
6	문제점 보완 및 수정									
7	최종 보고서 작성									

Table 1. 설계 진행일정

Ⅱ. 시스템의 설계[분석, 합성]

2-1. 시스템 설계

2-1-1. 설계목표

YOLOv5를 이용한 실시간 객체 인식과 음성 데이터의 MFCC를 추출하여 학습시킨 CNN모델을 개발하여 물체의 재질을 구분하도록 한다.

2-1-2. 기능분석

객체인식을 수행하여 YOLOv5모델을 이용해 물체의 재질을 판별하고 그와 동시에 물체가 떨어지는 음성의 데이터를 전처리하여 CNN 모델을 이용해 독립적으로 한 번 더 재질을 판별하도록 한다. 이는 두 종류의 기계학습을 실행하여 부족한 신뢰도를 보완하기 위함이다. 각각의 판별 결과를 출력하고 이 결과를 시리얼 통신으로 아두이노에 송신한다. 객체 인식의 결과를 출력하고 신뢰도가 0.7 이상이면 이 결과를 최종 결과로 판단한다. 만약 객체 인식을 통한 판별 결과의 신뢰도가 0.7 미만인 경우, 음성 데이터를 이용한 판별 결과를 최종 결과로 판단하게 된다. 최종 판별 결과가 정해지면 이 결과에 따라 아두이노에서는 2개의 서보모터를 순차적으로 제어하여 물체를 서로 다른 곳으로 분리한다.

2-1-3. 동작부

2-1-3-1. 아두이노 우노 (Arduino uno)

아두이노는 오픈소스(Open Source)를 기반으로 한 단일 보드인 마이크로 컨트롤러(Microcontroller)로 완성된 보드와 관련 개발 도구 및 환경이다. 그중 아두이노 우노는 가장 많이 사용되는 보드로써 총 44개의 핀과 단자들로 구성되어 있어 여러 센서들의 제어에 이용될 수 있으며 다양한 응용이 가능하다.

Fig 2-1. 아두이노 우노 보드

2-1-3-2. 서보모터 (Servo Motor)

서보모터는 DC 모터에 귀환회로를 추가하여 정확한 위치제어가 가능하게 구성된 모터이다. 본 프로젝트에서는 MG995를 사용하였으며 이 서보모터는 기어가 메탈로 되어 있으며, 토크가 타 서보모터보다 높아 더 강한 힘을 낼 수 있다.

Fig 2-2. MG995

2-1-3-3. 프레임

2개의 서보모터의 회전 각도에 따라 물체를 분류할 수 있도록 2단으로 설계하였으며, 총 3개의 재질로 분류되도록 구상하였다. 서보모터에 의해 회전하는 분류판은 떨어질 때 음성 데이터가 명확하게 추출될 수 있도록 하기 위해 나무판으로 제작했으며, 전체 프레임 또한 작품의 안전성을 위해 목재로 재질을 정하였다.

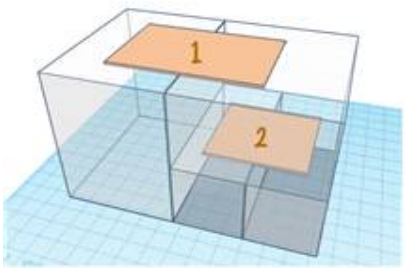
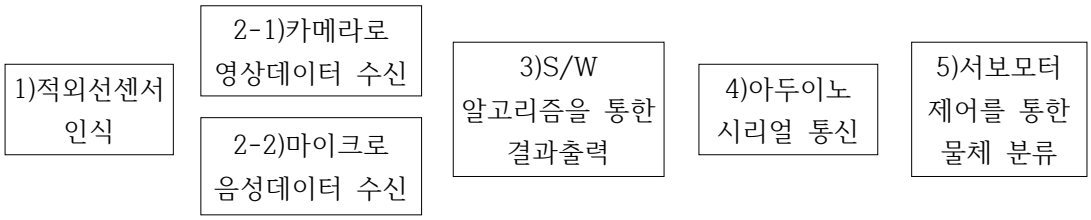


Fig 2-3. 작품 구상도

2-2. 시스템 구성 요약



2-3. S/W algorithm

2-3-1. YOLOv5

YOLO 모델은 객체인식을 수행하기 위한 심층 신경망으로, bounding box coordinate와 classification을 동일 신경망 구조를 통해 실행하는 one stage detector이다.
오픈소스인 YOLOv5중 s 모델을 선택하여 기존의 R-CNN 모델보다 빠르게 객체인식을 수행하도록 설계하였다.

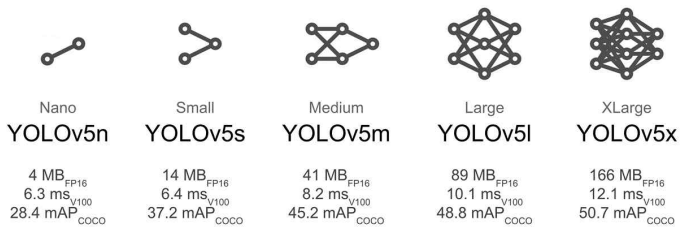


Fig 2-4. YOLOv5모델의 종류

2-3-2. MFCC (Mel-Frequency Cepstral Coefficient)

MFCC는 '음성데이터'를 '특징 벡터'(Feature)화 해주는 알고리즘이다. 입력된 소리 전체를 대상으로 하는 것이 아니라, 일정 구간(Short time)씩 나누어, 이 구간에 대한 스펙트럼을 분석하여 특징을 추출하는 기법이다. 다음 그림은 MFCC의 Mel-Cepstral을 추출하는 과정을 나타낸 것이다.

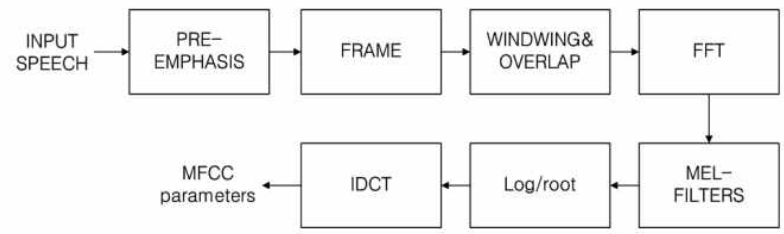


Fig 2-5. Mel_Cepstral 추출 과정

① Pre-emphasis (고대역 강조 필터) : 음성신호는 고대역일수록 에너지값이 작아지는 현상이 있다. 이를 위해 음성의 특징을 추출할 때 고대역을 증가시키기 위해 고대역 강조 필터 식을 통과시킨다.

② Frame Blocking : 음성신호는 단구간,즉 프레임별로 처리하는데 프레임 간의 음성 정보의 손실을 고려하여 임의의 N개의 샘플로 블로킹(Blocking)하여 하나의 프레임으로 사용하고 다시 임의의 N개의 샘플만큼 이동하여 중첩시킨 후 다시 N개의 샘플로 블로킹하여 다음 프레임으로 사용한다.

③ Windowing : 음성신호는 주기적인 신호와 비주기적인 신호의 두 가지 성분으로 이루어진 준주기적 신호이다. 그렇기 때문에 주기적인 윈도우를 취해 음성신호를 좀 더 주기적인 신호로 만들어야 한다.

④ FFT (Fast Fourier Transform) : FFT는 DFT와 IDFT를 계산하는데 효과적인 알고리즘이다. FFT는 큰 정수의 빠른 계산에 의한 디지털 신호처리와 미분의 방정식들의 부분적인 해결을 빠르게 계산이 가능하기 때문에 다양한 종류에 적용되기 때문에 매우 중요하다.

다음은 MFCC의 전체적인 수식이다.

$$MFCC(d) = \sum_{k=1}^{K} X_k \cos\left[d(k-0.5)\frac{\pi}{K}\right], \quad d = 1,\ldots,D$$

여기서 K는 필터의 개수이고 D는 캡스트럴 분석 차수이다.

2-3-3. CNN (Convolution Neural Network)

CNN은 특징추출(Feature Extraction)과 이미지 분류(Classification)로 구성되어 있고, 이미지 분류에 최적화되어있다. 특징추출 단계에서 Filter로 이미지의 주요 특징을 추출하는 Convolution layer와 특징을 강화하고 크기를 줄이는 Pooling layer를 반복 실행하고, 이미지 분류 단계에서 3차원 컬러 이미지나, 여러 장의 4차원 이미지를 인공신경망의 입력 데이터 형태인 1차원 배열로 평면화시켜 Feature Map을 얻고, Fully-connected 신경망에 보내서 분류한다.

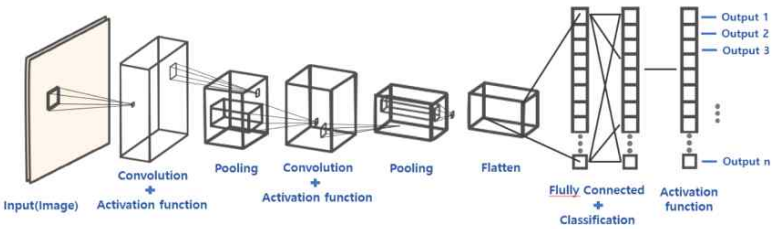


Fig 2-6. CNN 네트워크의 구조

2-3-4. 전체 알고리즘

1. 적외선 센서가 물체 탐지 후 아두이노로 신호 송신

2. 물체 탐지 신호를 수신하면 음성 녹음 시작/ 객체 인식 실행

3. 녹음된 음성파일 전처리 및 MFCC 추출 /웹캠 비디오 스트림 캡쳐

4. YOLOv5로 재질 판별 후 신뢰도 출력 & CNN모델로 재질 판별 및 판별결과 출력 / 두 결과를 아두이노로 송신

5-1. 객체인식의 판별결과가 신뢰도 0.7 이상
-> 객체인식의 판별결과를 최종판별결과로 지정

5-2. 객체인식의 판별결과가 신뢰도 0.7 미만
-> MFCC를 이용한 판별결과를 최종판별결과로 지정

6. 최종 판별결과에 따라 두 개의 서보모터 각도 제어

Ⅲ. 시스템의 제작[합성, 제작]

3-1. 개발환경 구축

기본적으로 파이썬을 사용하여 개발을 진행했기 때문에 Python IDE인 Pycharm을 베이스로 가상환경을 만들어 사용하였고, 학습모델의 경우에는 gpu를 무료로 사용할 수 있는 환경인 Google Colab을 사용했다.

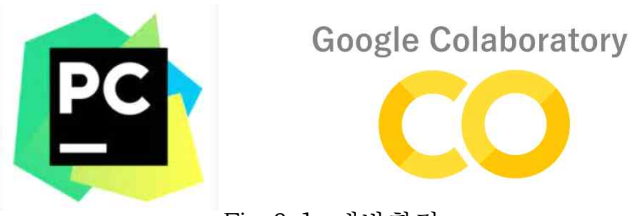


Fig 3-1. 개발환경

3-2. 모델구축

3-2-1. YOLOv5 model

```
from google.colab import drive
import os

drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).
```

```
[ ]  # 작업 경로를 MyDrive로 변경하여 구글 드라이브에 접속 후 바로 작업 디렉토리를 확인 가능

print('현재 작업 경로 :', os.getcwd())
os.chdir('/content/drive/MyDrive')
print('변경된 작업 경로 :', os.getcwd())
```

```
현재 작업 경로 : /content/drive/My Drive/yolov5
변경된 작업 경로 : /content/drive/MyDrive
```

```
[ ]  # 처음에 한번만 실행 !
!git clone https://github.com/ultralytics/yolov5  # YOLOv5 레퍼지토리 clone
```

```
fatal: destination path 'yolov5' already exists and is not an empty directory.
```

```
[ ]  !pip install -r requirements.txt
```

```
Collecting gitpython>=3.1.30 (from -r requirements.txt (line 5))
  Downloading GitPython-3.1.34-py3-none-any.whl (188 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.6/188.6 kB 3.1 MB/s eta 0:00:00
Requirement already satisfied: matplotlib>=3.3 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 6)) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 7)) (1.23.5)
Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 8)) (4.8.0.76)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 9)) (9.4.0)
```

```
[ ]  #@title
     # 필요한 패키지 다운로드 및 임포트
     %cd yolov5
     %pip install -qr requirements.txt # install dependencies
     %pip install -q roboflow

     import torch
     import yaml
     from IPython.display import Image, clear_output  # to display images

     print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

```
/content/drive/MyDrive/yolov5
Setup complete. Using torch 2.1.0+cu118 (Tesla T4)
```

Fig 3-2. YOLOv5 model-1

```
[ ]  !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="a1Drrbu0nA0IkZb4tdcj")
     project = rf.workspace("schoolcapstoneproject").project("capstonetrash")
     dataset = project.version(5).download("yolov5")
```

```
     Requirement already satisfied: roboflow in /usr/local/lib/python3.10/dist-packages (1.1.9)
     Requirement already satisfied: certifi==2023.7.22 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2023.7.22)
     Requirement already satisfied: chardet==4.0.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.0.0)
     Requirement already satisfied: cycler==0.10.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.10.0)
     Requirement already satisfied: idna==2.10 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.10)
     Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.5)
     Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7.1)
     Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.23.5)
     Requirement already satisfied: opencv-python-headless==4.8.0.74 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.8.0.74)
     Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (9.4.0)
     Requirement already satisfied: pyparsing==2.4.7 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.4.7)
```

```
[ ]  # 사용할 데이터셋 경로 및 데이터셋의 yaml 파일 경로 지정
     data_dir = '/content/drive/MyDrive/yolov5/CapstoneTrash-5'
     data_yaml = '/content/drive/MyDrive/yolov5/CapstoneTrash-5/data.yaml'
```

```
[ ]  # 데이터셋 yaml 파일 확인
     with open(data_yaml) as f:
         film = yaml.load(f, Loader=yaml.FullLoader)
         display(film)
```

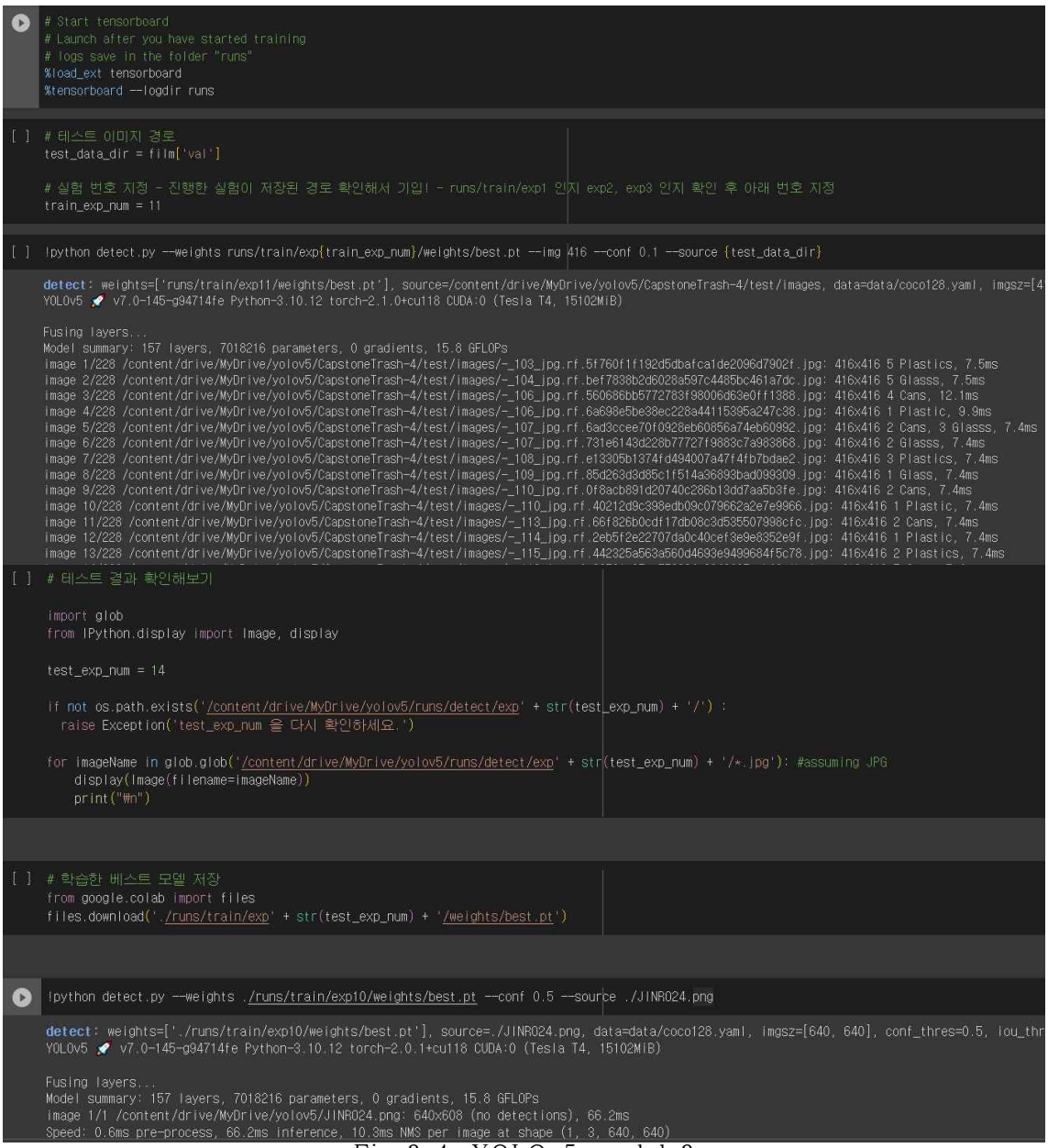
```
     {'names': ['Can', 'Glass', 'Plastic'],
      'nc': 3,
      'roboflow': {'license': 'CC BY 4.0',
       'project': 'capstonetrash',
       'url': 'https://universe.roboflow.com/schoolcapstoneproject/capstonetrash/dataset/5',
       'version': 5,
       'workspace': 'schoolcapstoneproject'},
      'test': '../test/images',
      'train': 'CapstoneTrash-5/train/images',
      'val': 'CapstoneTrash-5/valid/images'}
```

```
[ ]  # yaml 파일의 train, val 데이터가 있는 경로 수정 (기존 경로 -> 구글 드라이브에 저장된 경로로)
     film['train'] = '/content/drive/MyDrive/yolov5/CapstoneTrash-5/train/images'
     film['val'] = '/content/drive/MyDrive/yolov5/CapstoneTrash-5/test/images'

     with open(data_yaml, 'w') as f:
         yaml.dump(film, f)

     print('변경된 yaml 파일 :')
     with open(data_yaml) as f:
         film = yaml.load(f, Loader=yaml.FullLoader)
         display(film)
```

```
[ ]  !python train.py --img 416 --batch 16 --epochs 220 --data {data_yaml} --weights yolov5s.pt --cache
```

```
     2023-11-05 18:33:11.315896: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to reg
     2023-11-05 18:33:11.315951: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to regi
     2023-11-05 18:33:11.315994: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to r
     train: weights=yolov5s.pt, cfg=, data=/content/drive/MyDrive/yolov5/CapstoneTrash-5/data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=220,
     github: ⚠ YOLOv5 is out of date by 93 commits. Use 'git pull' or 'git clone https://github.com/ultralytics/yolov5' to update.
     YOLOv5 🚀 v7.0-145-g94714fe Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)

     hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05
     ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5 🚀 in ClearML
     Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
     TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
     Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
     100% 755k/755k [00:00<00:00, 15.4MB/s]
     Overriding model.yaml nc=80 with nc=3

                      from  n    params  module                                  arguments
       0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
       1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
       2                -1  1     18816  models.common.C3                        [64, 64, 1]
       3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
       4                -1  2    115712  models.common.C3                        [128, 128, 2]
       5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
       6                -1  3    625152  models.common.C3                        [256, 256, 3]
       7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
       8                -1  1   1182720  models.common.C3                        [512, 512, 1]
       9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
      10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
      11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
      12          [-1, 6]  1         0  models.common.Concat                    [1]
      13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
      14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
      15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
```

Fig 3-3. YOLOv5 model-2



```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

```
# 테스트 이미지 경로
test_data_dir = film['val']

# 실험 번호 지정 - 진행한 실험이 저장된 경로 확인해서 기입! - runs/train/exp1 인지 exp2, exp3 인지 확인 후 아래 번호 지정
train_exp_num = 11
```

```
!python detect.py --weights runs/train/exp{train_exp_num}/weights/best.pt --img 416 --conf 0.1 --source {test_data_dir}
```

```
detect: weights=['runs/train/exp11/weights/best.pt'], source=/content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images, data=data/coco128.yaml, imgsz=[4
YOLOv5 🚀 v7.0-145-g94714fe Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
image 1/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_103_jpg.rf.5f760f1f192d5dbafca1de2096d7902f.jpg: 416x416 5 Plastics, 7.5ms
image 2/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_104_jpg.rf.bef7838b2d6028a597c4485bc461a7dc.jpg: 416x416 5 Glasss, 7.5ms
image 3/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_106_jpg.rf.560686bb5772783f98006d63e0ff1388.jpg: 416x416 4 Cans, 12.1ms
image 4/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_106_jpg.rf.6a698e5be38ec228a44115395a247c38.jpg: 416x416 1 Plastic, 9.9ms
image 5/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_107_jpg.rf.6ad3ccee70f0928eb60856a74eb60992.jpg: 416x416 2 Cans, 3 Glasss, 7.4ms
image 6/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_107_jpg.rf.731e6143d228b77727f9883c7a983868.jpg: 416x416 2 Glasss, 7.4ms
image 7/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_108_jpg.rf.e13305b1374fd494007a47f4fb7bdae2.jpg: 416x416 3 Plastics, 7.4ms
image 8/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_109_jpg.rf.85d263d3d85c1f514a36893bad099309.jpg: 416x416 1 Glass, 7.4ms
image 9/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_110_jpg.rf.0f8acb891d20740c286b13dd7aa5b3fe.jpg: 416x416 2 Cans, 7.4ms
image 10/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_110_jpg.rf.40212d9c398edb09c079662a2e7e9966.jpg: 416x416 1 Plastic, 7.4ms
image 11/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_113_jpg.rf.66f826b0cdf17db08c3d535507998cfc.jpg: 416x416 2 Cans, 7.4ms
image 12/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_114_jpg.rf.2eb5f2e22707da0c40cef3e9e8352e9f.jpg: 416x416 1 Plastic, 7.4ms
image 13/228 /content/drive/MyDrive/yolov5/CapstoneTrash-4/test/images/-_115_jpg.rf.442325a563a560d4693e9499684f5c78.jpg: 416x416 2 Plastics, 7.4ms
```

```
# 테스트 결과 확인해보기

import glob
from IPython.display import Image, display

test_exp_num = 14

if not os.path.exists('/content/drive/MyDrive/yolov5/runs/detect/exp' + str(test_exp_num) + '/') :
  raise Exception('test_exp_num 을 다시 확인하세요.')

for imageName in glob.glob('/content/drive/MyDrive/yolov5/runs/detect/exp' + str(test_exp_num) + '/*.jpg'): #assuming JPG
    display(Image(filename=imageName))
    print("₩n")
```

```
# 학습한 베스트 모델 저장
from google.colab import files
files.download('./runs/train/exp' + str(test_exp_num) + '/weights/best.pt')
```

```
!python detect.py --weights ./runs/train/exp10/weights/best.pt --conf 0.5 --source ./JINR024.png
```

```
detect: weights=['./runs/train/exp10/weights/best.pt'], source=./JINR024.png, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.5, iou_thr
YOLOv5 🚀 v7.0-145-g94714fe Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
image 1/1 /content/drive/MyDrive/yolov5/JINR024.png: 640x608 (no detections), 66.2ms
Speed: 0.6ms pre-process, 66.2ms inference, 10.3ms NMS per image at shape (1, 3, 640, 640)
```

Fig 3-4. YOLOv5 model-3

3-2-2. CNN model(MFCC)

```
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import itertools
import librosa
import IPython.display as ipd
import matplotlib.pyplot as plt


midi_file = "/content/drive/MyDrive/capstonedata/capstone_3mat.wav"

#파일에는 3개의 재질의 소리가 100개씩 3초 간격으로 존재

materials = [0, 1, 2]
num_materials = 100
sec = 3

audio = []
inst = []
for inst_idx, num_mat in itertools.product(range(len(materials)), range(num_materials)):
  material = materials[inst_idx]
  offset = (material*num_materials*sec) + (num_mat*sec)
  print('material: {}, num_mat: {}, offset: {}'.format(material, num_mat, offset))
  y, sr = librosa.load(midi_file, sr=None, offset=offset, duration=3.0)
  audio.append(y)
  inst.append(inst_idx)
```

```
Mounted at /content/drive
material: 0, num_mat: 0, offset: 0
material: 0, num_mat: 1, offset: 3
material: 0, num_mat: 2, offset: 6
material: 0, num_mat: 3, offset: 9
```

```
[ ]  #-------------<MFCC를 이용한 머신러닝 오디오 분류>-----------------------
     audio_mfcc = []

     for y in audio:
         ret = librosa.feature.mfcc(y=y,sr=sr, hop_length=1024)
         audio_mfcc.append(ret)

     mfcc_np = np.array(audio_mfcc, np.float32)
     inst_np = np.array(inst, np.int16)

     print(mfcc_np.shape, inst_np.shape)
```

```
(300, 20, 141) (300,)
```

```
[ ]  mfcc_np = mfcc_np.reshape((300, 20 * 141))
```

```
[ ]  from sklearn.preprocessing import MinMaxScaler

     scaler = MinMaxScaler()
     scaler.fit(mfcc_np)
```

```
▾ MinMaxScaler
MinMaxScaler()
```

Fig 3-5. CNN model-1

```
[ ]  #--------------------<CNN모델 구성>--------------------

     from keras.utils import to_categorical
     from sklearn.model_selection import train_test_split

     mfcc_np = np.array(audio_mfcc, np.float32)
     mfcc_array = np.expand_dims(mfcc_np, -1)
     inst_cat = to_categorical(inst_np)

     train_x, test_x, train_y, test_y = train_test_split(mfcc_array, inst_cat, test_size=0.2)

     print(train_x.shape)
     print(test_x.shape)
     print(train_y.shape)
     print(test_y.shape)
```

```
(240, 20, 141, 1)
(60, 20, 141, 1)
(240, 3)
(60, 3)
```

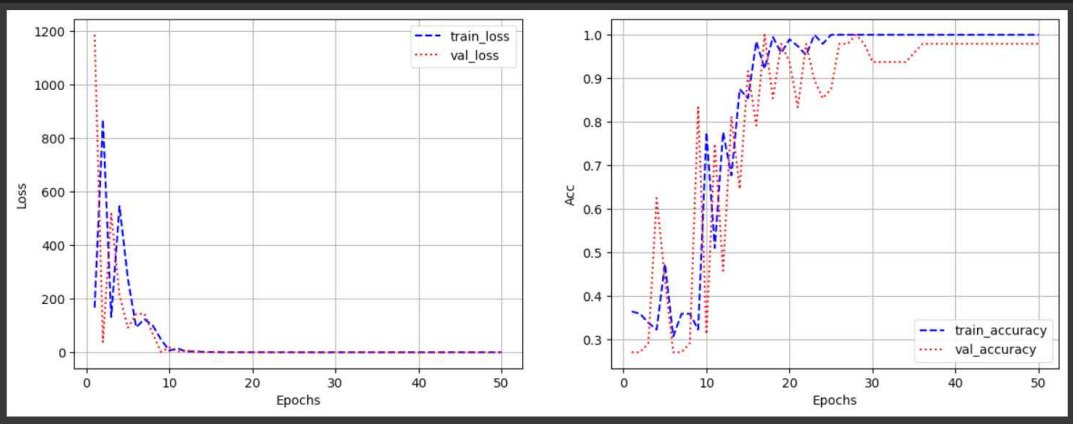
```
[ ]  from keras.layers import Conv2D, MaxPool2D, Flatten, Input, Dense
     from keras.models import Sequential, Model

     def model_build():
       model = Sequential()

       input = Input(shape=(20, 141, 1))

       output = Conv2D(128, 3, strides=1, padding='same', activation='relu')(input)
       output = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(output)

       output = Conv2D(256, 3, strides=1, padding='same', activation='relu')(input)
       output = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(output)

       output = Conv2D(512, 3, strides=1, padding='same', activation='relu')(input)
       output = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(output)

       output = Flatten()(output)
       output = Dense(512, activation='relu')(output)
       output = Dense(256, activation='relu')(output)
       output = Dense(128, activation='relu')(output)

       output = Dense(3, activation='softmax')(output)

       model = Model(inputs=[input], outputs=output)

       model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])

       return model
```

```
[ ]  model = model_build()
     model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 20, 141, 1)]      0

 conv2d_2 (Conv2D)           (None, 20, 141, 512)      5120

 max_pooling2d_2 (MaxPoolin  (None, 10, 71, 512)       0
 g2D)

 flatten (Flatten)           (None, 363520)            0

 dense (Dense)               (None, 512)               186122752

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 3)                 387

=================================================================
Total params: 186292483 (710.65 MB)
Trainable params: 186292483 (710.65 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig 3-6. CNN model-2

```
[ ]  history = model.fit(train_x, train_y, epochs=50, batch_size=128, validation_split=0.2)

Epoch 1/50
2/2 [==============================] - 13s 608ms/step - loss: 166.2719 - acc: 0.3646 - val_loss: 1185.1622 - val_acc: 0.2708
Epoch 2/50
2/2 [==============================] - 0s 112ms/step - loss: 868.9875 - acc: 0.3594 - val_loss: 38.4162 - val_acc: 0.2708
Epoch 3/50
2/2 [==============================] - 0s 114ms/step - loss: 131.5576 - acc: 0.3385 - val_loss: 517.1033 - val_acc: 0.2917
Epoch 4/50
2/2 [==============================] - 0s 111ms/step - loss: 545.7604 - acc: 0.3229 - val_loss: 214.1429 - val_acc: 0.6250
Epoch 5/50
2/2 [==============================] - 0s 123ms/step - loss: 275.9817 - acc: 0.4740 - val_loss: 91.6172 - val_acc: 0.4375
Epoch 6/50
2/2 [==============================] - 0s 112ms/step - loss: 93.5375 - acc: 0.3073 - val_loss: 142.1851 - val_acc: 0.2708
Epoch 7/50
2/2 [==============================] - 0s 115ms/step - loss: 123.3433 - acc: 0.3594 - val_loss: 144.2932 - val_acc: 0.2708
Epoch 8/50
2/2 [==============================] - 0s 121ms/step - loss: 99.9679 - acc: 0.3594 - val_loss: 68.8218 - val_acc: 0.2917
Epoch 9/50
2/2 [==============================] - 0s 115ms/step - loss: 47.9611 - acc: 0.3229 - val_loss: 1.6337 - val_acc: 0.8333
Epoch 10/50
2/2 [==============================] - 0s 114ms/step - loss: 6.3301 - acc: 0.7760 - val_loss: 17.7383 - val_acc: 0.3125
Epoch 11/50
2/2 [==============================] - 0s 114ms/step - loss: 15.2166 - acc: 0.5104 - val_loss: 2.5233 - val_acc: 0.7500
Epoch 12/50
2/2 [==============================] - 0s 117ms/step - loss: 2.1310 - acc: 0.7760 - val_loss: 7.5385 - val_acc: 0.4583
Epoch 13/50
2/2 [==============================] - 0s 111ms/step - loss: 3.3742 - acc: 0.6771 - val_loss: 1.6137 - val_acc: 0.8125
Epoch 14/50
2/2 [==============================] - 0s 113ms/step - loss: 1.2139 - acc: 0.8750 - val_loss: 2.4798 - val_acc: 0.6458
Epoch 15/50
2/2 [==============================] - 0s 114ms/step - loss: 0.9486 - acc: 0.8542 - val_loss: 0.3637 - val_acc: 0.9167
Epoch 16/50
2/2 [==============================] - 0s 114ms/step - loss: 0.1192 - acc: 0.9844 - val_loss: 2.2079 - val_acc: 0.7917
Epoch 17/50

[ ]  plot_history(history.history)
```




```
[ ]  model.evaluate(test_x, test_y)

2/2 [==============================] - 0s 108ms/step - loss: 0.1620 - acc: 0.9667
[0.1619589775800705, 0.9666666388511658]
```

Fig 3-7. CNN model-3

3-3. 모델을 이용한 판별 알고리즘

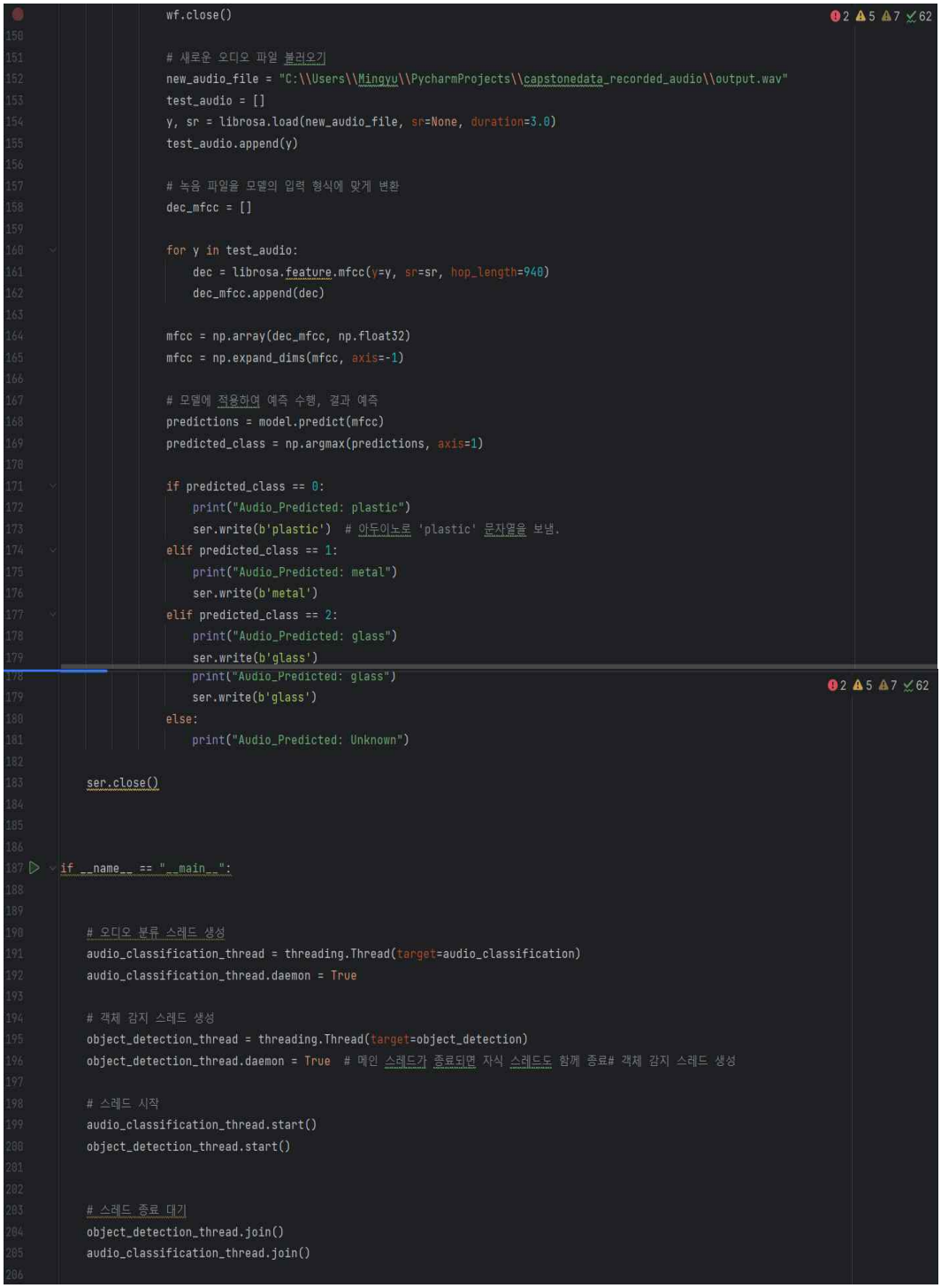
각 판별이 독립적으로 실행되도록 하기 위해 멀티 스레딩을 사용하였다.

```python
import cv2
import torch
from pathlib import Path
import random
import serial
from tensorflow.keras.models import load_model
import pyaudio
import numpy as np
import wave
import librosa
import threading
import time

# 아두이노와 연결된 시리얼 포트 설정 (적절한 포트 이름으로 변경하세요)
arduino_port = 'COM5'

# 시리얼 통신을 열고 아두이노와 연결합니다.
ser = serial.Serial(arduino_port, baudrate: 9600)

# 모델을 로드할 때 이미 로드 되어 있는지 확인 하고, 없는 경우에만 로드 합니다.
if 'model' not in locals():
    model = load_model('C:/Users/Mingyu/PycharmProjects/capstonedata_cnn_model/mfcc_model')

CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100

p = pyaudio.PyAudio()

# 모델 파일의 경로 설정 (적절한 경로로 변경하세요)
# 모델 파일의 경로 설정 (적절한 경로로 변경하세요)
model_path = 'C:/Users/Mingyu/PycharmProjects/capstonedata_cnn_model/exp10/weights/best.pt'

# YOLOv5 모델을 초기화합니다.
yolo_model = torch.hub.load( repo_or_dir: 'ultralytics/yolov5', model: 'custom', path=model_path)

# 클래스별 랜덤 색상을 생성합니다.
num_classes = len(yolo_model.names)
class_colors = [(random.randint( a: 0, b: 255), random.randint( a: 0, b: 255), random.randint( a: 0, b: 255)) for _ in range(num_classes)]

# 웹캠 비디오 스트림을 캡처합니다.
cap = cv2.VideoCapture(1)

# 객체 감지를 수행하는 함수


1 usage
def object_detection():
    last_send_time = 0
    object_detected_time = 0
    detection_triggered = False  # 객체 감지 트리거 변수

    while True:
        ret, frame = cap.read()

        if not ret:
            break

        # 프레임을 YOLOv5 모델에 전달하여 객체 감지를 수행합니다.
        results = yolo_model(frame)
```

Fig 3-8. 판별 알고리즘-1

```
        # 현재 시간을 가져옵니다.
        current_time = cv2.getTickCount() / cv2.getTickFrequency()

        if ser.inWaiting() > 0:
            arduino_data = ser.readline().decode('utf-8').strip()
            if arduino_data == "Object Detected" and not detection_triggered:
                object_detected_time = time.time() + 2.2  # 적외선 센서가 물체를 감지한 시간 기록 + 2.2초
                print("물체 감지됨, 2초 후에 객체인식 시작")
                detection_triggered = True  # 객체 감지 트리거 활성화

        if detection_triggered and time.time() >= object_detected_time:
            for detection in results.pred[0]:
                class_id = int(detection[-1].item())  # 클래스 ID를 추출합니다.
                class_name = yolo_model.names[class_id]  # 클래스명을 가져옵니다.
                confidence = detection[4].item()  # 신뢰도(확률)를 추출합니다.

                # 객체 감지 박스의 좌표를 추출합니다.
                bbox = detection[:4].tolist()
                x1, y1, x2, y2 = map(int, bbox)

                # 클래스별 랜덤 색상을 사용하여 객체 감지 박스를 그립니다.
                color = class_colors[class_id]
                cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)

                # 클래스명과 신뢰도를 화면에 출력합니다.
                label = f"{class_name}: {confidence:.2f}"
                cv2.putText(frame, label, org: (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, color, thickness: 2)

                # 클래스명과 신뢰도를 노트북 화면에 출력하거나 다른 작업을 수행합니다.
                print(f"Class: {class_name}, Confidence: {confidence}")
                print(f"Class: {class_name}, Confidence: {confidence}")

                # 클래스명과 신뢰도를 아두이노로 전송합니다.
                data = f"Class: {class_name}, Confidence: {confidence}\n"
                ser.write(data.encode())  # 데이터를 아두이노로 전송합니다.

                # 결과를 화면에 출력하거나 다른 후속 작업을 수행합니다.
                detection_triggered = False  # 객체 감지가 수행된 후 트리거 비활성화

            # 결과를 화면에 출력하거나 다른 후속 작업을 수행합니다.
        cv2.imshow(winname: "YOLOv5 Object Detection", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()




# 오디오 분류를 수행하는 함수
1 usage
def audio_classification():
    while True:
        if ser.inWaiting() > 0:
            arduino_data = ser.readline().decode('utf-8').strip()
            if arduino_data == "Object Detected":
            if arduino_data == "Object Detected":
                print("Object detected, recording started")

                stream = p.open(format=FORMAT,
                                channels=CHANNELS,
                                rate=RATE,
                                input=True,
                                frames_per_buffer=CHUNK,
                                input_device_index=2)

                print('start recording')

                frames = []
                seconds = 3
                for i in range(0, int(RATE / CHUNK * seconds)):
                    data = stream.read(CHUNK)
                    frames.append(data)

                print('record stopped')

                stream.stop_stream()
                stream.close()

                recorded_file_path = r'C:\Users\Mingyu\PycharmProjects\capstonedata_recorded_audio\output.wav'

                wf = wave.open(recorded_file_path, mode: 'wb')
                wf.setnchannels(CHANNELS)
                wf.setsampwidth(p.get_sample_size(FORMAT))
                wf.setframerate(RATE)
                wf.writeframes(b''.join(frames))
```

Fig 3-9. 판별 알고리즘-2



```
            wf.close()

            # 새로운 오디오 파일 불러오기
            new_audio_file = "C:\\Users\\Mingyu\\PycharmProjects\\capstonedata_recorded_audio\\output.wav"
            test_audio = []
            y, sr = librosa.load(new_audio_file, sr=None, duration=3.0)
            test_audio.append(y)

            # 녹음 파일을 모델의 입력 형식에 맞게 변환
            dec_mfcc = []

            for y in test_audio:
                dec = librosa.feature.mfcc(y=y, sr=sr, hop_length=940)
                dec_mfcc.append(dec)

            mfcc = np.array(dec_mfcc, np.float32)
            mfcc = np.expand_dims(mfcc, axis=-1)

            # 모델에 적용하여 예측 수행, 결과 예측
            predictions = model.predict(mfcc)
            predicted_class = np.argmax(predictions, axis=1)

            if predicted_class == 0:
                print("Audio_Predicted: plastic")
                ser.write(b'plastic')  # 아두이노로 'plastic' 문자열을 보냄.
            elif predicted_class == 1:
                print("Audio_Predicted: metal")
                ser.write(b'metal')
            elif predicted_class == 2:
                print("Audio_Predicted: glass")
                ser.write(b'glass')
                print("Audio_Predicted: glass")
                ser.write(b'glass')
            else:
                print("Audio_Predicted: Unknown")

    ser.close()



if __name__ == "__main__":


    # 오디오 분류 스레드 생성
    audio_classification_thread = threading.Thread(target=audio_classification)
    audio_classification_thread.daemon = True

    # 객체 감지 스레드 생성
    object_detection_thread = threading.Thread(target=object_detection)
    object_detection_thread.daemon = True  # 메인 스레드가 종료되면 자식 스레드도 함께 종료# 객체 감지 스레드 생성

    # 스레드 시작
    audio_classification_thread.start()
    object_detection_thread.start()


    # 스레드 종료 대기
    object_detection_thread.join()
    audio_classification_thread.join()
```

Fig 3-10. 판별 알고리즘-3

3-4. 아두이노 제어코드

```
#include <Servo.h>
#include <SoftwareSerial.h>

Servo servo1; // 첫 번째 서보 모터
Servo servo2; // 두 번째 서보 모터
int servoPin1 = 9; // 첫 번째 서보 모터 핀
int servoPin2 = 11; // 두 번째 서보 모터 핀
int irSensorPin = A0;  // IR sensor analog pin


void setup() {
  servo1.attach(servoPin1); // 첫 번째 서보 모터 핀 설정
  servo2.attach(servoPin2); // 두 번째 서보 모터 핀 설정
  Serial.begin(9600); // 시리얼 통신 시작 (아두이노와 연결된 시리얼 통신)
  servo1.write(78);
  servo2.write(85);
  delay(500);
}

void loop() {

  int irValue = analogRead(irSensorPin);
  if (irValue < 500) {  // Adjust this threshold according to your IR sensor
    Serial.println("Object Detected");  // Print to the serial monitor
  } else {
    Serial.println("No Object");  // Print to the serial monitor
  }
  delay(250);  // Delay for stability

  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n'); // 블루투스 모듈을 통해 데이터 수신
    // 예상 데이터 형식: "Class: Plastic, Confidence: 0.85"

    // 데이터에서 클래스와 신뢰도 추출
    String objectClass = "";
    float confidence = 0.0;

    if (parseData(data, objectClass, confidence)) {
      Serial.println("Received data: " + data);
      Serial.println("Object Class: " + objectClass);
      Serial.println("Confidence: " + String(confidence, 2)); // 소수점 둘째 자리까지 표시

      if (confidence >= 0.7) {
        if (objectClass == "Glass") {
          // Plastic 클래스일 때 서보 모터 각도 변경
          servo1.write(130); // 130도(왼쪽)로 변경
          delay(3000); // 각도 변경 후 일정 시간 유지 (1초)
          servo1.write(78); // 초기 위치로 복귀
        } else if (objectClass == "Can" || objectClass == "Plastic") {
          // Can 클래스 또는 Glass 클래스일 때 서보 모터 각도 변경
          servo1.write(20); // 20도(오른쪽)로 변경
          delay(3000); // 각도 변경 후 일정 시간 유지 (1초)
          servo1.write(78); // 초기 위치로 복귀
        } else {
          // 다른 클래스일 때 서보 모터는 아무 동작도 하지 않음
          servo1.write(78); // 초기 위치로 복귀
        }

        delay(1000);

        if (objectClass == "Can" && confidence >= 0.7) {
        // Plastic 클래스이고, Confidence가 0.8 이상일 때 서보 모터 각도 변경
        servo2.write(130); // 130도(왼쪽)로 변경
```

Fig 3-11. 아두이노 제어코드-1

```
            if (objectClass == "Can" && confidence >= 0.7) {
            // Plastic 클래스이고, Confidence가 0.8 이상일 때 서보 모터 각도 변경
            servo2.write(130); // 130도(왼쪽)로 변경
            delay(3000); // 각도 변경 후 일정 시간 유지 (1초)
            servo2.write(85); // 초기 위치로 복귀
          } else if (objectClass == "Plastic" && confidence >= 0.7) {
            // Can 클래스 또는 Glass 클래스이고, Confidence가 0.8 이상일 때 서보 모터 각도 변경
            servo2.write(20); // 20도(오른쪽)로 변경
            delay(3000); // 각도 변경 후 일정 시간 유지 (1초)
            servo2.write(85); // 초기 위치로 복귀
          } else {
            // 위 조건을 만족하지 않을 때 서보 모터는 아무 동작도 하지 않음
            servo2.write(85); // 초기 위치로 복귀
          }

          } else {
            // confidence가 0.7 미만인 경우, 두 번째 코드 실행
            String inputString = Serial.readString();
            inputString.trim(); // 앞뒤 공백 제거

            if (inputString.equals("glass")) {

              servo1.write(130);
              delay(2500); // 각도 변경 후 일정 시간 유지
              servo1.write(78); // 초기 위치로 복귀

             } else if (inputString.equals("metal")) {

            servo1.write(20);
            delay(3000);
            servo1.write(78);
            delay(1000);
            delay(1000);

            servo2.write(130);
            delay(3000);
            servo2.write(85);
            } else if (inputString.equals("plastic")) {

            servo1.write(20);
            delay(3000);
            servo1.write(78);
            delay(1000);

            servo2.write(20);
            delay(3000);
            servo2.write(85);
            }
          }
        }
      }
    }

    bool parseData(String data, String &objectClass, float &confidence) {
      int classIndex = data.indexOf("Class: ");
      int confidenceIndex = data.indexOf("Confidence: ");

      if (classIndex != -1 && confidenceIndex != -1) {
        // 데이터에서 클래스와 신뢰도 추출
        objectClass = data.substring(classIndex + 7, confidenceIndex - 2);
        confidence = data.substring(confidenceIndex + 12).toFloat();
        return true;
      }
      return false;
    }
```

Fig 3-12. 아두이노 제어코드-2

3-5. 동작부 구현

아래 그림과 같이 총 3가지의 재질을 분류하도록 하였으며, 2단으로 구성되었다. 첫 번째 서보모터의 작동 후 약 1초의 딜레이가 있으며 이후에 두 번째 서보모터가 작동하여 분류를 원활하게 하도록 하였다.
분류판은 물체가 떨어지는 음성이 명확하게 들리는 나무판으로 제작했으며, 첫 번째 분류판 위에는 적절한 각도로 객체 인식에 이용할 USB 카메라(웹캠)를 설치했다. 또한 분류판 옆에 USB 마이크를 설치하여 음성 데이터를 잘 받아들일 수 있도록 구현하였다.

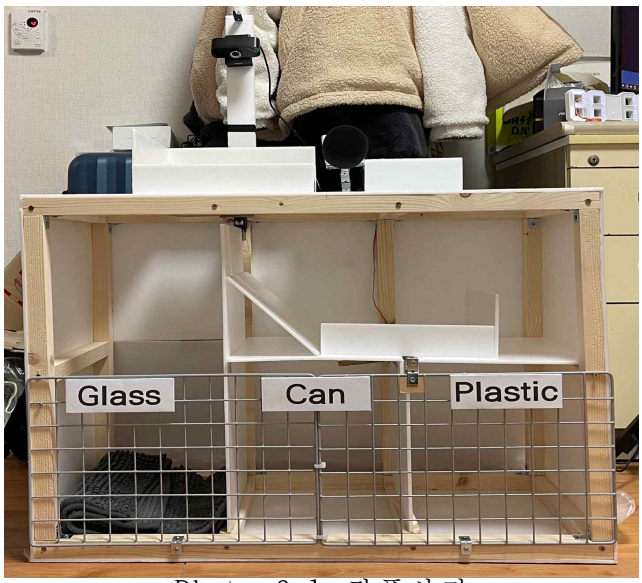


Photo. 3-1. 작품사진

Ⅳ. 시스템의 동작 검증[시험, 성능평가]

4-1. 적외선 센서의 동작 여부

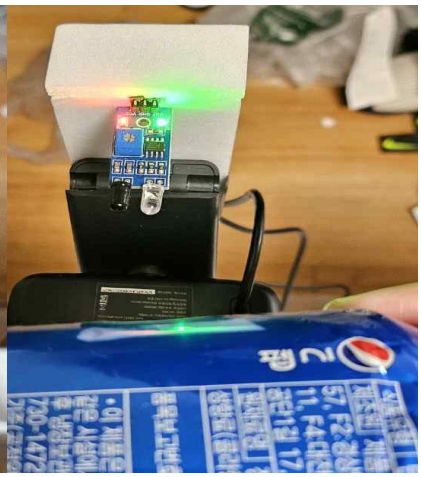


Photo 4-1. 아무것도 감지되지 않았을 때의 적외선 센서
Photo 4-2. 투입하려는 쓰레기가 감지되었을 때의 적외선 센서

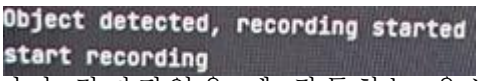


Fig 4-1. 쓰레기가 감지되었을 때 작동하는 음성분석 시스템

가변저항을 이용해 적외선 센서의 범위를 조절하여 같은 공간의 다른 물체와의 간섭을 없앴다. 투입하려고 하는 쓰레기가 약 5~10cm 이내의 공간으로 들어가야 적외선 센서가 물체를 인식한다. 물체가 이렇게 인식이 되면 적외선 센서가 아두이노로 정보를 보낸 후, 아두이노에 입력된 정보를 파악한 파이썬이 음성분석 시스템을 작동시키는 원리이다.

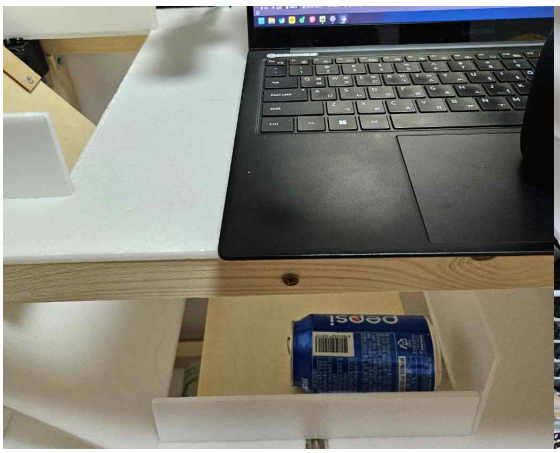
4-2. 실시간 객체 인식 정보를 통해 동작하는 경우

Photo 4-3. 쓰레기 투입구에 캔 재질의 쓰레기를 투입하려는 모습

웹캠이 설치된 쓰레기 투입구에 캔 쓰레기를 넣어보았다.

Fig 4-2. Yolov5가 판별한 쓰레기 종류와 신뢰도

실시간 객체 인식이 쓰레기를 캔(Can)으로 판단하였고, 신뢰도는 0.84 이었다. 이 신뢰도가 기존 설정한 임계값인 0.7 이상이므로 음성 데이터 분석 없이 바로 분류를 시작하였다.



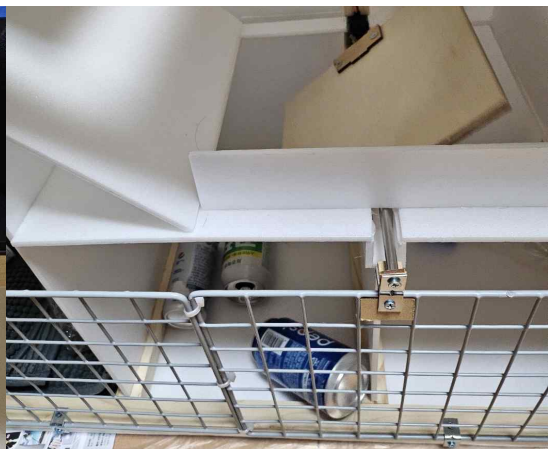


Photo 4-4. 1차 분류판이 오른쪽으로 기울어져 2차 분류판으로 들어간 모습
Photo 4-5. 2차 분류판이 왼쪽으로 기울어져 최종 분류칸(Can)으로 투입된 모습

캔으로 분류될 시, 1차 분류판이 첫 번째 서보모터에 의해 오른쪽으로 기울어지고, 2차 분류판이 두 번째 서보모터에 의해 왼쪽으로 기울어져 최종적으로 가운데 분류 칸인 캔(Can) 구역으로 쓰레기가 투입된다.

4-3. 음성 인식 딥러닝 모델의 판단을 이용하여 동작하는 경우

Photo 4-6. 1차 분류판이 왼쪽으로 기울여져 유리(Glass)로 투입된 쓰레기

실시간 객체인식은 유리(Glass), 신뢰도 0.55로 판단하여 보류된 상태이다. 따라서 음성 데이터 분석을 이용해 판별된 재질로 쓰레기가 이동하도록 아두이노를 프로그래밍하였다. 그 결과, 1차 분류판이 왼쪽으로 기울어져 유리(Glass) 구역으로 쓰레기가 투입된 것을 확인할 수 있다.

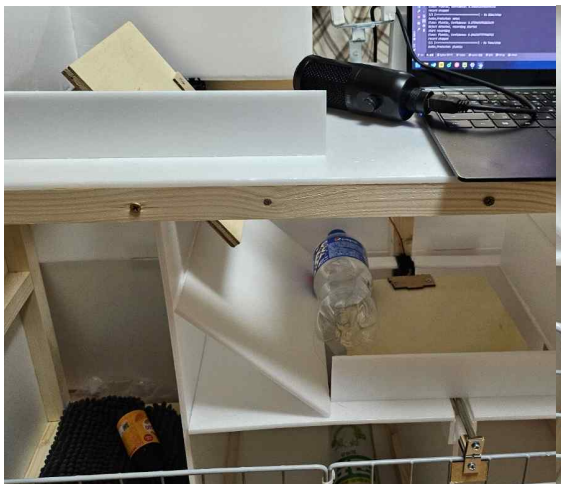



Photo 4-7. 1차 분류판이 오른쪽으로 기울어져 2차 분류판으로 들어간 상황
Photo 4-8. 2차 분류판이 오른쪽으로 기울어져 최종 분류칸(Plastic)으로 투입된 모습

실시간 객체 인식으로는 유리 즉, Glass라 판별 되었다. 하지만 신뢰도가 0.34인 관계로 보류된 상태이다. 때문에 음성 데이터 분석을 이용해 판별한 재질로 쓰레기가 이동하도록 아두이노를 프로그래밍하였다. 그 결과, 1차 분류판이 오른쪽으로 기울어지고 2차 분류판도 오른쪽으로 기울어져 가장 오른쪽의 플라스틱(Plastic) 구역으로 쓰레기가 투입된 것을 볼 수 있다.

Ⅴ. 결론 [결과도출]

5-1. 목표로 설정했던 기능

USB 카메라(웹캠)을 이용한 실시간 객체 인식 시스템과 USB 마이크를 이용한 실시간 음성 분석 시스템을 구축하려 하였다. 이 두 시스템을 동시에 작동시킨 후 더 높은 신뢰도를 가진 시스템으로 물체의 재질을 구분하는 것이 목표였다.
처음 아이디어 회의부터 2학기 초반까지 동작부는 컨베이어 벨트와 로봇팔을 결합하여 구현하려 했다. 컨베이어 벨트 위로 지나가는 쓰레기를 아두이노와 로봇팔을 활용하여 할당된 공간으로 투입하려 했으나 비용적 문제와 실제 구현이 불가능하다 판단되어 폐기하였다.

5-2. 개발결과 시스템의 기능

USB 카메라(웹캠)을 이용한 실시간 객체 인식 시스템과 USB 마이크를 이용한 실시간 음성 분석 시스템의 구축에 성공하였다. 실시간 객체인식은 YOLOv5 모델을 사용했으며 실시간 음성분석은 MFCC 모델을 사용하였다.
실시간 객체인식 시스템의 판별 결과를 우선시하되, 보조적으로 음성분석 시스템을 사용한다. 객체인식의 신뢰도가 설정한 값인 0.7 미만일 경우, 음성 데이터 분석 결과를 이용하여 쓰레기의 재질을 플라스틱, 캔, 유리로 분류한다.
동작부는 아두이노와 2개의 서보모터를 사용하여 구현했다. 불필요한 리소스 사용 방지와 분류 시스템의 정확도 향상을 위해 적외선 센서를 이용했다. 적외선 센서에 물체가 인식되면 시스템이 작동되도록 설계하였다. 이후 객체인식과 음성분석을 통해 플라스틱, 캔, 유리의 세 재질로 분류가 가능하다.

5-3. 목표설정 시스템과 개발결과 시스템 완성도

완성도 100%. 원래 생각했던 쓰레기 재질 구분 알고리즘은 살짝 변경되었지만 가장 큰 목표였던 객체인식과 음성 데이터 분석 시스템을 이용하여 실시간으로 재질을 구분하는 알고리즘 구축엔 성공하였다.
동작부 또한, 본래 결정했던 컨베이어 벨트와 로봇팔이 아닌 2개의 서보모터를 이용하여 분류판을 조작하고 회전하는 방식으로 바뀌었지만 결과적으로 기능에는 큰 차이가 없다.

5-4. 주요 활용 분야

공원이나 길거리 등 사람들이 많이 왕래하는 공공장소에 큰 크기의 자동 재활용 쓰레기 분류기를 설치하면 분리수거율이 상승될 것이라고 생각된다. 또한 아파트, 빌라 등 사람들의 집단 거주 구역에 설치하면 길가의 쓰레기 문제도 해결될 수 있을 것으로 보인다.
뿐만 아니라 학교나 놀이터 등 학생들이 자주 다니는 곳에 설치하면 딥러닝과 코딩 등 공학 설계에 관한 학생들의 흥미를 유발할 수 있을 것으로 예상된다.

Ⅵ. 참 고 문 헌

[1] 환경부, 「전국폐기물발생및처리현황」, 2019, 2023.03.11., 폐기물 발생현황_생활폐기물

[2] 코로나시대 폐기물 통계 : 동향과 쟁점, 김고운・이혜진, 서울연구원 연구보고서, 2022

[3] 권선미・윤우성・정유민, “[쓰레기대란]① “쓰레기 버릴 곳 없어”...10년내 쓰레기 대란 온다” *연합뉴스*, 2021년 7월 23일, https://www.yna.co.kr/view/AKR20210719145400501

[4] 김형환, “[현장에서] “쓰레기 분리배출 제멋대로... 재활용 30-40% 그쳐” “ UPI뉴스, 2020년 2월 18일, https://www.upinews.kr/newsView/upi202002170085

[5] 박진원・김영진, “YOLOv5 기반의 딥러닝 성능 개선 연구“, 한국통신학회 학술대회논문집 2022.6, 2022, 1592-1593

[6] 김현규. “화자확인을 위한 MFCC 성능 향상에 관한 연구.“ 국내석사학위논문 광운대학교, 2007. 서울

[7] 윤원정. “MFCC를 활용한 딥러닝 기반 음성 인식 모델에 관한 연구.“ 국내석사학위논문 한세대학교 대학원, 2022. 경기도

Ⅶ. 부 록

부품리스트

품 명	규 격	제작사 및 일련번호	단가(원)	구입처 (상호명 또는 인터넷사이트)
아두이노 우노 Uno R3 SMD 호환보드	아두이노 우노 호환보드	OEM	4,400	에듀이노
아두이노 모터 드라이버 쉴드	L298P 쉴드	OEM	17,000	에듀이노
아두이노 적외선(IR) 송수신 센서 모듈	적외선(IR) 송수신 센서 모듈	OEM	2,200	에듀이노
메탈기어 서보모터 M995	360도 회전 서보모터	OEM	3,580	알파마이크로
점퍼선 케이블	쉴드 및 모터 연결	CODBOT	2,600	송파메이커스페이스
USB 웹캠	앱코 APC850 FHD	앱코	24,300	인터파크
USB 마이크	브리츠 BE-STM300	브리츠	42,140	쿠팡

1) 품명에는 H/W소자 및 S/W package 등도 포함됨.
2) 규격에는 전기적 특성과 용도등의 규격을 말함.

Ⅷ. 감사의 글

본 작품의 연구는 서울과학기술대학교 전자IT미디어공학과에서 이루어졌으며 설계부터 완성에 이르기까지 많은 분들의 도움을 받았습니다. 먼저 주제 선정부터 이론의 정립 등 작품의 완성이 이루어지기까지 학문적 지도는 물론 모든 일을 깊은 관심으로 지도해 주신 도현락 교수님께 감사의 말씀을 드립니다.

딥러닝을 활용한 자동 재활용 쓰레기 분류기라는 주제를 선정하고 만들기까지 쉽지만은 않은 여정이었습니다. 계획했던 주제를 폐기하고 생각했던 구현을 아예 뒤엎기도 하면서 프로젝트를 진행해왔습니다.

이번 졸업 작품 프로젝트를 위해 지난 수업에서 배운 지식을 활용하기도 했고, 처음 접하는 부분도 많아 새로운 내용을 공부해야 할 때도 있었습니다. 이러한 과정 속에서 공학 설계 능력을 키우고 또 다른 지식과 경험을 얻었습니다. 이렇게 얻은 지식과 경험이 졸업 후 사회에 나가서도 유용하게 쓰일 수 있기를 바랍니다.

매주 작품의 진행 상황을 파악하며 피드백 주신 도현락 교수님 그리고 매주 서로의 발표를 경청하며 함께 노력해온 같은 반분들과 저희 작품에 관심을 가지고 지켜봐 주신 분들에게 감사의 인사를 드립니다.

마지막으로 어려운 부분이 있을 때 주저 않고 도움을 주신 모든 분들과 본 프로젝트의 완성을 위해 때때로 밤을 새우며 1년간 함께 노력해 온 팀원들 덕분에 무사히 작품 제작을 마쳤습니다. 졸업 후에도 더욱 발전하는 공학도가 될 수 있도록 노력하겠습니다. 감사합니다.

2023년 11월 10일

이 윤 혁

정 지 혜

정 민 규