딥러닝과 귓바퀴의 구조를 이용한 소리의 방향 분석

조용우 김가온 조현태
서울과학고등학교

Sound localization using Deep learning and Ear flaps

Yongwoo Cho, Gaon Kim, Hyuntae Cho
Seoul Science High School

본문 초록

이 연구의 목적은, 다양한 귓바퀴 모양과 인공지능 알고리즘을 이용하여, 보다 효과적인 방향 탐지 장치를 만드는 것이다. 수많은 인간과 동물이 하듯, 기계 또한 귓바퀴 모양을 사용한다면 훨씬 더 적은 수의 마이크로 효과적인 방향 구분이 가능할 것이다. 두 개의 귓바퀴가 각각 접합된 마이크를 통해서 모든 구 좌표계의 각 단위 방향에서 녹음된 박수 소리를 딥러닝 데이터 수집기를 통해서 수집한 이후, CNN을 활용한 인공지능 알고리즘으로 소리 방향 분석의 정확도를 회귀 문제로 접근하여 구하는 연구이다.

Abstract

The purpose of this study is to create a more effective directional detection device using various earlobe shapes and artificial intelligence algorithms. Like so many humans and animals do, machines will be able to distinguish direction effectively with much fewer microphones if they use the shape of the ear. After collecting clapping sounds recorded in each unit direction of all spherical coordinate system through a microphone bonded to each, the accuracy of sound direction analysis is obtained by approaching a regression problem with an artificial intelligence algorithm using CNN.

I. 서론(Introduction)

1. 필요성과 목적

다양한 로봇과 전자제품들에서, 소리의 방향을 구면 좌표계 상에서 간단한 모듈로 쉽게 추정할 수 있다면, 인간을 위한 서비스의 질을 한 단계 더 끌어올릴 수 있을 것이라고 믿는다. 인명 구조용 로봇과 같은 경우 어떠한 소리에 대해서 방향을 구분해내는 능력이 당연히 필요하고, 그 정확도가 높을수록 더 많은 사람을 구해낼 수 있을 것이다. 또한, 소리를 이용해서 더 적은 수의 카메라를 이용한 탐지 또한 가능하며, 모듈을 간소화 하여 더 다양한 제품들을 저렴하게 만들 수 있을 것이다.

그러므로 우리 연구의 목적은, 인체의 귓바퀴 모양을 이용하여, 마이크 두 개를 이용하여 소리의 방향을 구면좌표계 상에서 찾을 수 있는 인공지능 모델과, 귓바퀴 형태를 탐구하는 것이다. 굳이 마이크 두 개로 한 이유는, 마이크의 수가 늘어날수록 비용과 휴대성이 감소하고, 인간이 두 개로 하는데 인공지능이 못할 리가 없다고 생각했기 때문이다.

2. 선행연구 및 관련 이론

음원탐지 및 위치 추정 알고리즘을 이용한 방재용 IoT 디바이스 시스템 설계 - 길민식1, 곽동걸2

위 연구에서는 4개의 마이크로폰 어레이를 이용해 소리 방향 구분 모듈을 만들었으며, 기본적인 TDOA(Time difference of arrival)을 이용해서 모듈을 제작하였다. 이로써, 기본적인, 귓바퀴가 적용되지 않은 소리 방향 구분 모듈은 어떤 식으로 구현되는지 확인할 수 있었다. [7]

위키백과에 따르면, 딥러닝은 다음과 같이 정의된다. “심층 학습(深層學習) 또는 딥 러닝(영어: deep structured learning, deep learning 또는 hierarchical learning)은 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화(abstractions, 다량의 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 작업)를 시도하는 기계학습 알고리즘의 집합으로 정의되며, 큰 틀에서 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한 분야라고 이야기할 수 있다.” [2]

인공신경망 훈련은 다음과 같이 이루어진다. 임의의 인공신경망을 준비한다. 이러한 인공신경망에, 예시 입력을 넣어준다. 그 후, 예시 입력에 대한 인공신경망의 출력을 구한다. 그 후 예시 입력에 대한 정답(예상 출력)이랑 인공신경망의 출력을 대조해서, 오차만큼 인공신경망의 가중치를 수정해준다. 이렇게, 수많은 개수의 예시 입력과 출력으로 인공신경망을 교육해주면, 우리가 원하는 인공신경망을 얻을 수 있는 것이다. [1]

딥러닝의 알고리즘 중, 음성 인식을 위해 쓰이는 알고리즘으로는, CNN (Convolution Neural Network), RNN (Recurrent Neural Network), LSTM (Long Short Term Memory)같은 것들이 있다.

CNN은 주로 이미지를 처리하는데 쓰이는 알고리즘으로, 필터를 반복적으로 적용시켜 이미지를 필요한 만큼 추상화시켜 필요한 것만 남기는 것이다. 그런 후, 일반적인 인공신경망을 통해서 주로 어떤 이미지인지 인식하고 구분한다. 이를 음성인식 알고리즘에 적용시키는 방법은, 음성 파일을 이미지로 변환하는 함수를 하나 추가로 적용시키는 것이다. 말만 들으면 비효율적으로 들릴수도 있겠지만, 짧은 길이의 소리 분석에는 높은 효율과 속도로 인해 많이 쓰인다. [3]

RNN은 시간상 순서를 가지고 있는, 순차적인 자료형을 구분하기 위한 알고리즘으로, 마찬가지로 짧은 길이의 소리를 분석하기 위해 쓰여진다. 하지만 오차가 완전히 되먹임 되지 않고, 속도가 무척이나 느려 긴 순열을 분석하는데 쓰이지는 않는다. [4]

LSTM은 RNN의 개선된 형태로, 이전 노드들의 기억력을 임의로 조정해 연산속도와 오차 분석을 더 체계적으로 바꾸었다. LSTM은 보다 더 긴 순열도 분석 가능하다. [5]

추가로, LSTM과 유사한 GRU라는 구조도 있다고 한다. 본 연구에서는 1-2초 내외의 짧은 길이의 소리 신호임과 더불어, STFT (short term fourier transform) method를 통한 음향 분석을 더 쉽고 간편하게 활용 가능한 CNN 알고리즘이 가장 적절하다 판단되어 CNN으로 실험하였다. [1]

II. 연구방법 및 이론(Method & Theory)

1. 도구 및 재료

연구의 단계는, 크게 두 단계로 이루어졌다. 소리의 방향을, 구 좌표계에서 5도 단위로 나누게 되면, 소리 샘플 하나당 2592(360/5 * 180/5)개의 소리를 녹음해야 된다. 다양한 귓바퀴에 적용 가능한 범용적인 TSM (train set maker)를 제작해야 했기에, 직접 손수 녹음을 하기보단 효율적으로 소리를 녹음해줄 장치를 만드는 과정이 필수이다. 그래서, 우선적으로 TSM을 제작한 후, TSM으로 소리 녹음을 마쳐 귓바퀴에 대한 소리를 수집한 이후, 딥러닝 알고리즘을 통한 회귀적 성능 분석이 필요했다.

다음은 TSM을 제작하기 위한 도구와 재료이다.

TSM-1 (Trainset Maker)를 만들기 위해서, 3D 프린터를 통한 모델링으로 설계를 하고, 스텝모터와 마이크가 달린 TSM-1을 제작하였다. 마이크 두 개, 스텝 모터 두 개, 8-15mm 베어링 4개와 라즈베리파이, 모터드라이버 등으로 설계하였다. 또한 스피커 부속은 Bose Companion II 로 사용하였다.

이 때 스텝 모터는 정해진 각도만큼을 정밀하게 돌아갈 수 있는 모터로, 특정한 각도만큼 돌아간 후 소리를 분석해야 하는 본 연구에서 사용해야 할 모터이다.

TSM-1에서는 Polar angle 회전을 맡는 스텝모터에 연결된 3D 프린터로 출력된 기어가, 스텝모터의 발열로 인해 녹는 문제가 발생하였다. 스텝모터의 발열은 과도한 토크가 걸릴시에 일어나기에, 이를 해결하기 위해 TSM-2는 회전부의 경량화와 스텝모터를 SG-90 서보모터로 대체하여 해결하였다. 또한 마이크를 일반적인 다이나믹 마이크에서 가벼운 핀마이크로 대체하여서 과도한 토크 문제를 해결하였다.

딥러닝 모듈은 Google Colab Pro를 통해 실행시켰으며, 귓바퀴 또한 3D 프린터로 출력하였다.

2. 실험 과정

귓바퀴 형태 A 하나를 제작한 이후, CNN을 이용한 알고리즘으로 실험을 진행할 것이다. 귓바퀴 A에 대한 음향 파일을, 여러 개의 박수 소리를 특정 각 단위로 녹음하여서 수천개의 파일을 각각 “소리파일명_방위_L/R”)의 파일명으로 저장할 것이다.

이렇게 저장된 소리 파일은 딥러닝 알고리즘을 통해 처리되어, 일부는 train_data, 나머지는 validation_data 로 나뉘어 설계된 CNN 인공지능 모델을 훈련시키는데 활용된다. 이때, Validation data의 Accuracy 척도로 훈련의 정확성을 비교할 것이다. CNN 인공지능 모델의 구조와 원리에 대해서는 결과 부분에서 자세히 다루도록 하겠다.

이때 박수 소리로 입력된 좌우 파형 두 가지는 소리의 음색, 소리의 시간 지연, 귓바퀴로 인해 회절된 소리의 음압 차이 등을 모두 고려하기 위해서 딥러닝 알고리즘을 선택했다. 다른 선행 연구에서는 딥러닝 알고리즘을 사용하지 않아 각 변수를 1개, 2개 정도만 고려해 알고리즘을 만들었기 때문에 제한된 범위에서만 데이터를 얻을 수 있었고, 정확한 sound localization을 구현할 수 없는 것에 반해 본 연구에서 사용한 딥러닝 알고리즘은 단순하지만 변수의 손실이 없는 방법으로 소리의 파형 전체를 분석하기 때문에 모든 변인을 고려할 수 있고, 따라서 보다 개선된 정확한 알고리즘 생성이 가능해진다.

III. 결과(Data & Result)

우리의 연구는, 크게 두 부분으로 나누어진다.

1. TSM의 설계와 제작, 시행착오
2. 데이터 분석과 인공지능

초기에, 인공지능에 대한 정보가 거의 없었기 때문에, 인공지능에 대한 공부를 시작하고, LTE ANN 강의와 <딥러닝 첫걸음> 과 같은 책을 읽어서 스스로 인공지능의 전반적인 개념과, 과제연구에 대한 큰 그림을 그려 나갔다. 이때, 위의 선행연구 및 관련 이론에서 서술한, 인공지능의 기본 원리와 여러 알고리즘, 오차 분석 함수, 활성함수에 대해서 이해를 하게 되었다.

딥러닝 공부를 시작함과 동시에, TSM을 설계하게 되었다. TSM은 인공지능 알고리즘을 돌리기 위한 소리 데이터 셋 수집기로, 여러 방향에서 들리게 되는 소리를 정확한 방위에서 사람 대신 녹음 해주는 역할을 하는 장치이다. 이 장치를 설계하고 구동하는데 시행착오가 많아 제작 기간이 많이 연장되었다. 이러한 시행착오에 대해 하나 둘 씩 서술을 해보려고 한다.

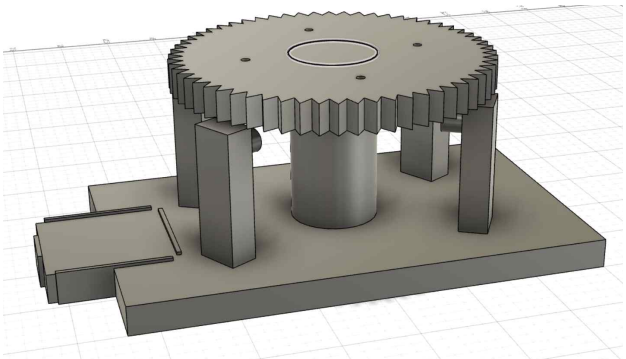
마지막으로, TSM을 이용해서 데이터 수집을 하고, 인공지능 모델을 설계했고, 그에 맞추어 데이터를 전처리한 후 넣어서 방향을 구분하는 모델을 러닝시켜 그의 성능을 확인해 보았다.

3 – 1 : TSM 제작 과정

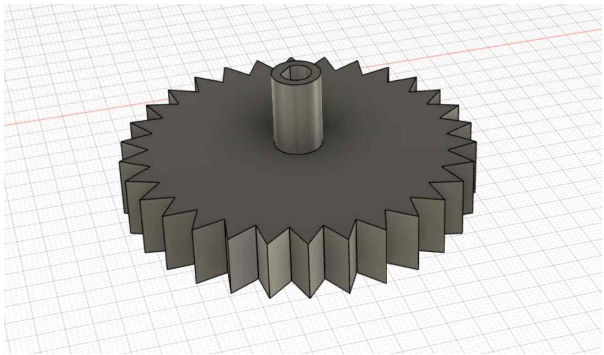
TSM – 1 의 설계이다.

초기에, TSM의 설계를 하였을 때는, 스피커가 긴 축을 통해서 회전하고 마이크가 달린 헤드가 한 축을 기준으로 회전하면서 소리를 녹음하는 구조를 생각했다. 하지만, 그러려다 보니 과도하게 부피가 크고, 사용이 어려우면서 많은 재료가 소비 될 것 같아, 구조를 단순화 하여 다시 설계를 해보았다.

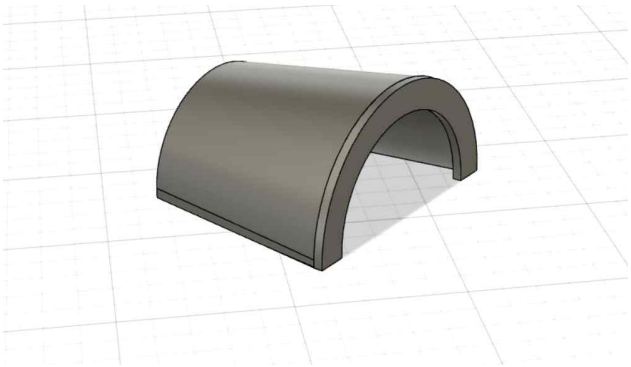
두 번째 설계는, 귓바퀴 구조물과 마이크가 달려있는 헤드가 두 축을 기준으로 3차원상 모든 방위를 가리킬 수 있도록 회전하고, 스피커는 고정되어있는 설계를 하였다. 실제로 완성 또한 이 방식으로 하였으며, 아래는 다음의 설계를 통해서 뽑아낸 부품들이다.



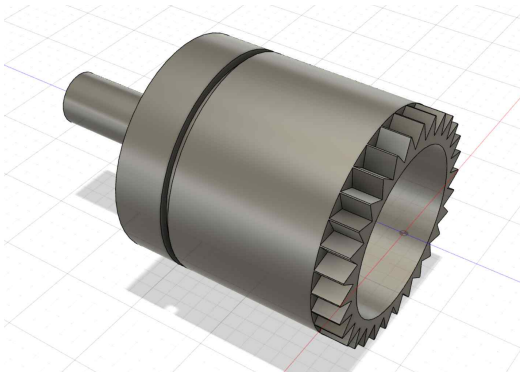
(A)



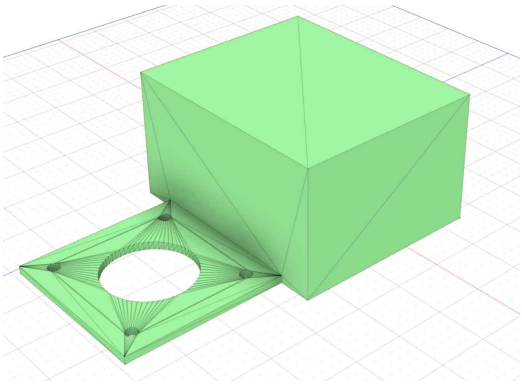
(B)



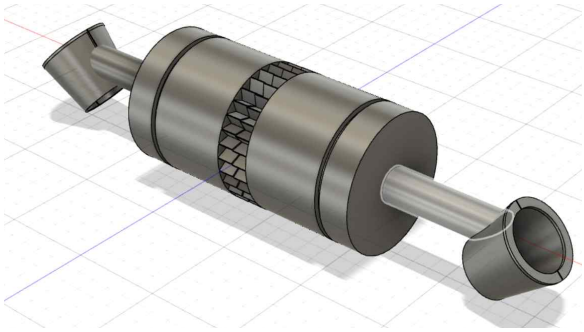
(C)



(D)

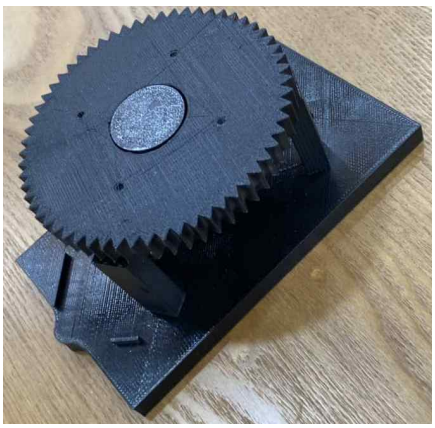


(E)



(F)

Fig 1. (A) : TSM의 밑판과, Azimuth angle 회전을 담당하는 기어이다. (B) : TSM의 Polar angle 회전을 담당하는 기어 (C): 마이크 고정대 부분 (D): 마이크 회전 고정대 (E): 스텝모터 마운트 (F) : 마이크 회전자의 전체 모습

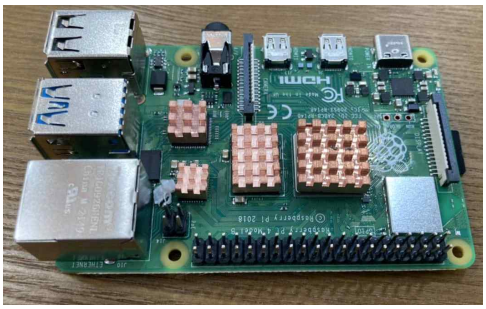


(A)

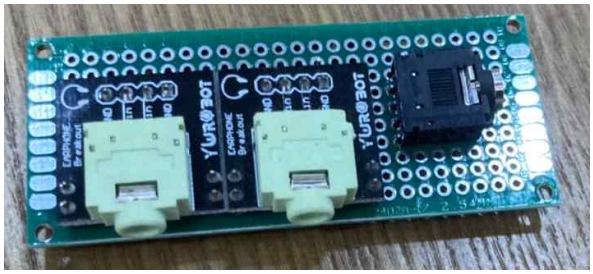
(B)

(C)

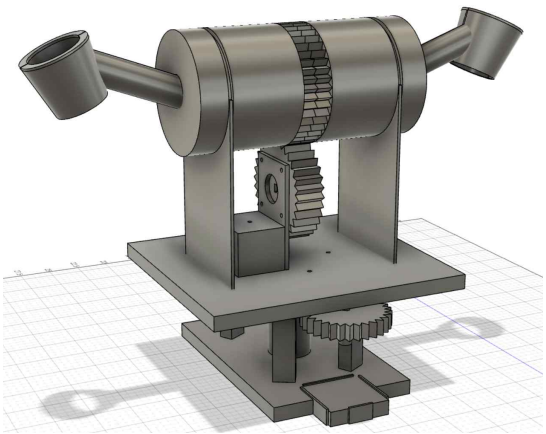
(D)



(E)



(F)



(G)

```python
import RPi.GPIO as GPIO
import time

#프로그램을 위한 변수와 상수들
low_stepper_pins = (13, 11, 15, 12)
high_stepper_pins = (16, 18, 19, 21)

singlestep = 1.8

ls_ratio = 60/30 * (1/ (singlestep/2))
hs_ratio = 1

time_ = 0.01

class Stepper :
    def __init__(self, a, ratio, pins):
        self.angle = a #현재 물체의 각도 (스텝모터 각도 아님)
        self.ratio = ratio #기어비
        self.pin = pins
        self.tense = 0
        self.i = 0
        self.positive = 0
        self.negative = 0
        self.v = 0
```

(H)

Fig 2. (A) : <하단부, 기어 결착모습> (B) : <하단부에 볼베어링을 장착한 모습> (C): <마이크 회전자의 모습> (D) : <스텝모터마운트와 스텝모터 결착모습> (E) :<라즈베리 파이의 모습> (F) : <3.5mm모노단자2개-3.5mm스테레오 단자 변환부의 모습> <G>:전체적인 수음기의 형태> <H> : <스테퍼 모터를 동작시키는 코드>

위의 부품들을 이용해서, 실제 완성된 TSM을 만들게 되었다.

처음 전체적인 형태를 만든 후 마이크 회전자의 모습을 처음 모델링 해보았다. 그러나 중간부분과 마이크 부분의 결속이 약해 무거운 무게의 마이크를 들어올리던 중 결속이 부러지는 문제가 생겼다. 또한, 마이크 회전자의 톱니를 너무 작게 만들어서 실제 회전이 불가능한 점도 문제가 되어 톱니 한 피스의 크기를 키우고 전체적인 형태를 보강한 마이크 회전자를 완성했다.

톱니-1은 하단부 스텝모터에 달려 상단부를 회전시키는 역할을 하는 톱니이다. 처음엔 상단부가 무거우므로 기어비를 높여 해결하기 위해 아래 톱니를 총 4피스로 만들었었다. 설계까지는 좋았으나 피스 사이의 각도가 너무 커져 하중이 걸릴 시 한 피스당 두 피스가 돌아가는 등 문제들이 발생하였다. 해결방법으로 보다 센 토크를 가지는 모터를 구하고 기어비를 줄여 회전이 안정적인 형태로 바꾸게 되어 현재의 톱니-1이 되었다. 또한, 마이크의 무게중심이 회전축에 더 가까이 붙을 수 있도록 설계를 변경하여서, 회전 관성을 최대한 줄여 정지와 구동 시 안정성을 늘리고자 하였다.

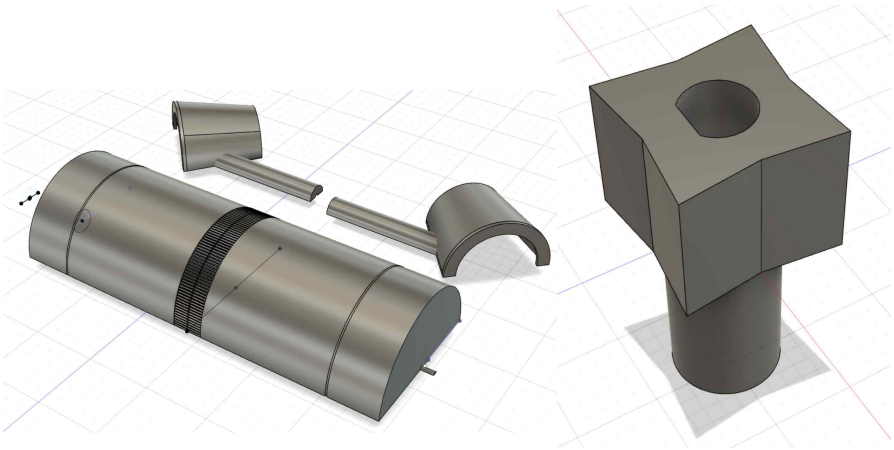


Fig 3. (A) : 마이크 회전자의 초기 형태, (B): 톱니-1의 초기 형태

아래 그림은 TSM-1의 구동 모습이다.

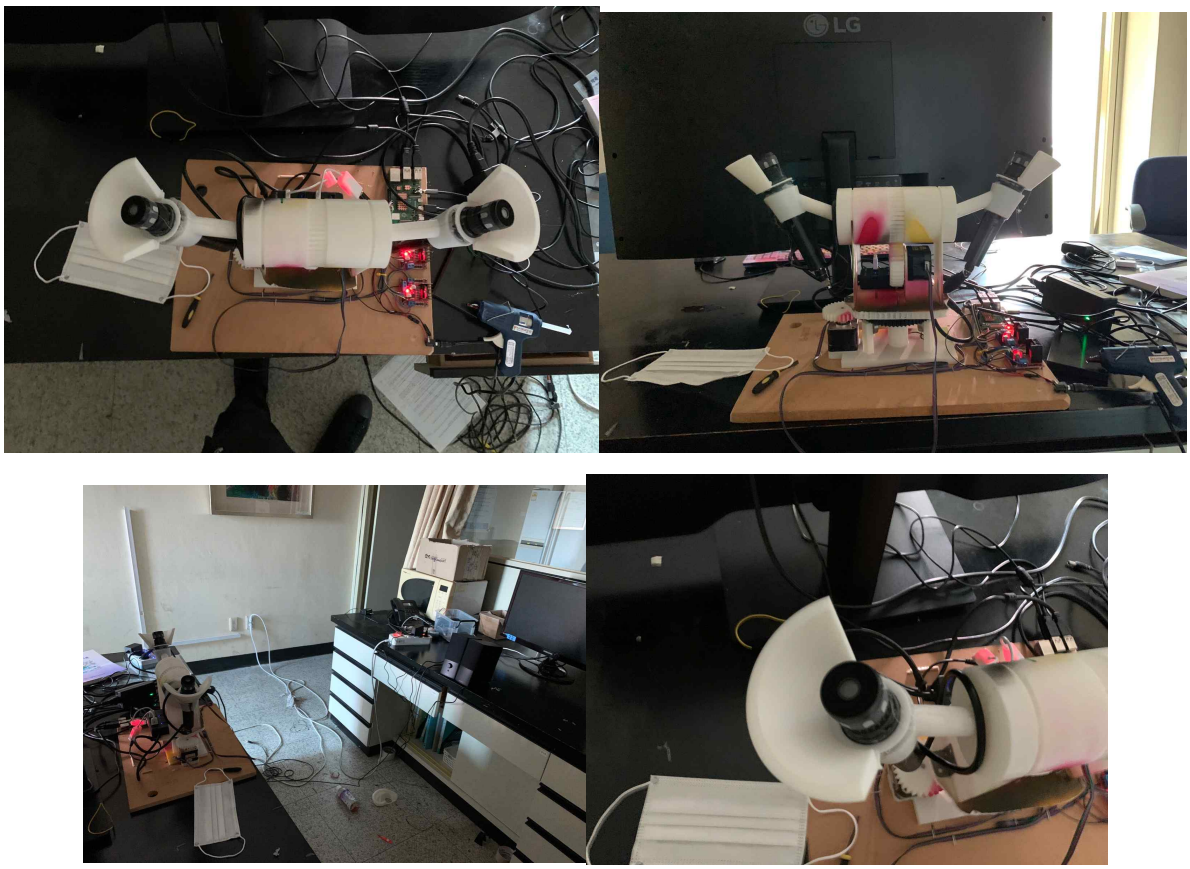


Fig 4. (A) 위에서 바라본 1차 TSM의 구동 모습. 위에 마이크와 귓바퀴 구조물이 있고, 우측에 라즈베리 파이와 L298N 두 개를 확인할 수 있다. (B) 전면에서 바라본 모습, 스텝모터 두 개를 확인 가능하다. (C) 스피커의 위치와 TSM의 위치를 확인 가능하다. (D) 귓바퀴 구조물 1의 모습과, 부착된 형태이다.

그러나, 단순히 마이크의 무게 중심 위치를 바꾸는 것 만으로는 과도한 토크 문제가 해결 되지 않았다. TSM-1의 모든 문제를 해결했다고 생각하고 시험삼아 데이터를 수집하던 도중, Polar angle 회전을 담당하던 스텝 모터가 계속된 발열로 열에 약한 3D 프린트 된 기어 부속을 녹여 더 이상 돌아가지 않았다. 이를 해결하기 위해, TSM-2는

아예 마이크를 바꾸고, 발열이 없는 가벼운 서보모터로 Polar angle 회전을 도맡아 하기로 하였다.

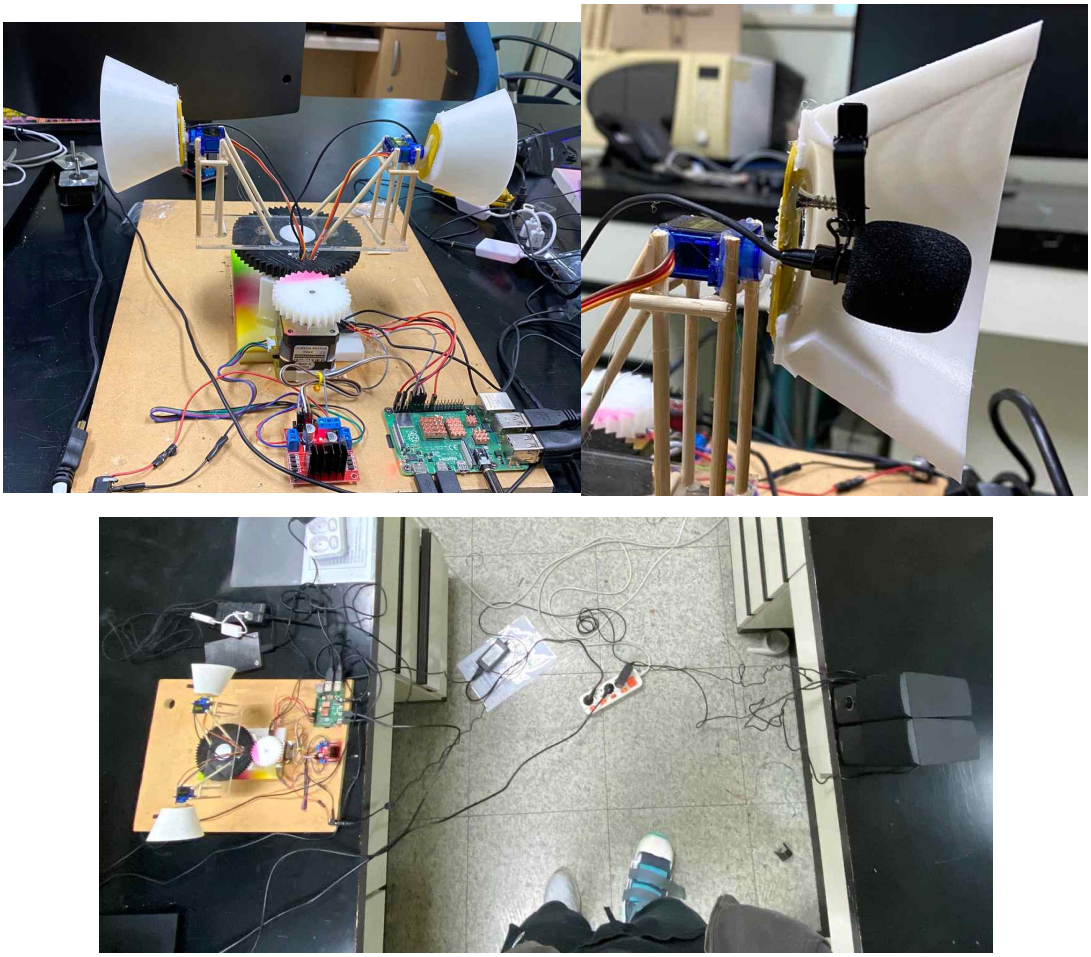


Fig 5. (A) 앞에서 바라본 TSM-2의 모습. 회전자가 경량화되었다는 것을 한눈에 볼 수 있다. (B) TSM-2의 마이크는 핀 마이크로 바뀌었다. (C) 스피커의 위치와 TSM의 위치를 확인 가능하다.

그러나, TSM-2에서는 서보모터의 알 수 없는 문제로 인해 자꾸 동작 도중 틱틱거리는 소리와 함께 움직였다. 그대로의 각도를 유지하면서 틱틱거리는 것이 때에 따라 나아졌다가도, 전력 공급이 원인인지, 아니면 라즈베리파이를 통한 제어에 하드웨어적인 문제가 있는 것인지 알 수 없었다.

TSM을 만들면서, 라즈베리 파이에 대한 코드 또한 설계하고 탐구 하였다. 라즈베리 파이는 초소형 컴퓨터/프로세서로, 우리 프로젝트에서는 모터와 마이크의 효율적이고 값 싼 제어를 위해서 필요하다.

라즈베리 파이는 세 가지를 정확하게 조절해주어야 한다.

1. 스텝 모터 (stepper motor)
2. 소리 재생
3. 소리 녹음

이다. 각각에 대해서 처음 써보는 기계 장치와 모듈인지라, 그 과정의 원리와 시행착오에 대해서 설명하고자 한다.

스테퍼 모터는, 특정 각도만큼만을 무한히 돌아갈 수 있는, 회전 각을 정밀하게 조절할 수 있는 모터이다. 우리 연구에서는 이 모터를 구면 좌표계 상에서 모든 방향에 있는 소리를 구현하기 위해서 사용한다. 각각의 스테퍼 모

터가, 구면좌표계 azimuth angle와, polar angle를 각각 맡는다. 스텝 모터는 17hs3430을 이용하였다. 모터의 특성상 큰 전류를 사용하기 때문에, 외부전원이 따로 필요하였다. 그래서, 12V 아답터를 이용해 전류를 공급해 주었다. 또한, L298N 모터 컨트롤러를 이용하여 스텝 모터의 각을 제어하였다. 스텝 모터를 움직이기 위해서는 스텝 모터의 전자석 여러쌍 (종류에 따라 다르다)을 각기 다르게 제어해주어야 한다. 스텝 모터 제어 코드는 아래의 보충 자료에 첨부하였다. 그 코드는 [8]의 내용을 많이 참고하여서 만들었다.

두 번째로 제어해주어야 할 것은, 소리 재생이다. 소리 재생은 pygame의 audio method를 이용하여서, 전체적인 프로그램의 구성과 소리 재생을 구성하였다. subprocess의 mplayer를 쓸수도 있었지만, pygame을 이용해 UI를 구성할거라 그냥 익숙한 것으로 짰다. 이 코드를 짜는 도중에, 파일을 열 수 있게 해주는 filename = tkinter.filedialog.askdirectory() 코드를 이용하여서 파일 여는 것을 구현하였다.

라즈베리 파이의 기본 파이썬 IDLE에, pygame이 설치가 똑바로 되어 있지 않아 이를 해결하기 위해서 많은 시간을 들였다. 결과적으로 라즈베리 파이 OS를 통째로 다시 설치하고 해결되었다.

세 번째로 라즈베리 파이로 다루어야 할 것은, 마이크 입력을 통한 소리 녹음이다. 우리는 마이크 두 개에 대한 입력을, 한쪽은 Left, 한쪽은 Right 전용 mono channel 로 사용할 것이였기 때문에, 소리를 분배해주는 장치를 만들어 주었다. 위에 있는 (F)가 소리 분배 장치이다.

그러나, 실제 완성된 코드를 가지고 테스트를 해보았을 때, 소리는 좌우가 동일하게 들렸다. USB 사운드카드를 분명 stereo로 샀지만, 출력만 stereo로 되고, 녹음하는 것은 mono로 녹음 되는 것이었다. 하는 수 없이 USB 사운드 카드를 하나 더 주문해서, 각각 따로 마이크에 연결을 해서 녹음을 받게 되었다.

소리 입력을 파이썬으로 통제하는 것 또한 많은 조사가 필요했다. 소리 입력은 우선 terminal에서 arecord 명령어를 이용해서 제어 할 수 있는데, 이를 사용하는 명령어는 arecord –f cd –D plughw:1, 0 —duration = 10 audio.wav 식으로 하면 됐다. -f cd 는 format(음질)을 CD로 받겠다는 뜻이고, -D plughw:는 1번 카드의 0번 디바이스로 녹음을 받겠다, -duration은 시간, 마지막에 파일 경로와 이름을 넣어주면 녹음이 됐다. 그러나, 문제는 파이썬을 이용해서 terminal input을 제어하는 것이었다.

이 terminal input을 파이썬 코드로 제어하는 것은, subprocess.Popen으로 해결했다. subprocess는 파이썬 코드에서 다른 코드나 terminal 명령어를 실행할 수 있게 해주는 모듈이다. shlex라는 라이브러리 함수로 명령어를 적당히 처리해서 넣어주면, 원하는대로 녹음이 되었다. 다음은 코드의 일부이다.

```
L_rec = 'arecord -D plughw:1,0 -t wav --duration=' + str(audio.get_length()) + ' -f cd ' +file_dir+ file_name+'L' + '.wav'
R_rec = 'arecord -D plughw:4,0 -t wav --duration=' + str(audio.get_length()) + ' -f cd ' + file_dir+file_name +'R'+ '.wav'

audio.play()
subprocess.popen(shlex.split(L_rec))
subprocess.popen(shlex.split(R_rec))
time.sleep(audio.get_length())
```

또한 핀 마이크의 AUX 4핀 단자와 3핀 단자의 호환 문제 또한 있었으나, 이는 변환젠더로 바로 해결 하였다.

다음은 TSM의 실행 화면 중 일부이다.

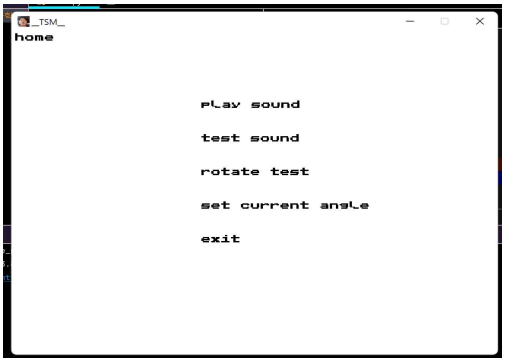


Fig 6. TSM의 실행 MAIN 화면 모습.

TSM은 다음과 같이 버튼을 마우스로 클릭하는 방식으로 구동 된다.

Play Sound : File Directory 함수를 통해서, 파일 탐색기를 통해 찾은 파일 경로 내부에 있는 모든 소리 파일을 모든 방위에 대해서 녹음하고 저장해주는 역할을 한다.

Test Sound : 소리가 잘 나오는지 테스트 하는 버튼이다.

Rotate Test : TSM의 회전자 (Azimuth, Polar) 회전을 테스트 하는 버튼이다.

Set Current Angle : 현재의 각도를 (0, 0) 원점으로 잡는다.

Exit : 종료

3-2 인공지능 모델

CNN을 이용한 인공지능 모델을 설명해보겠다.

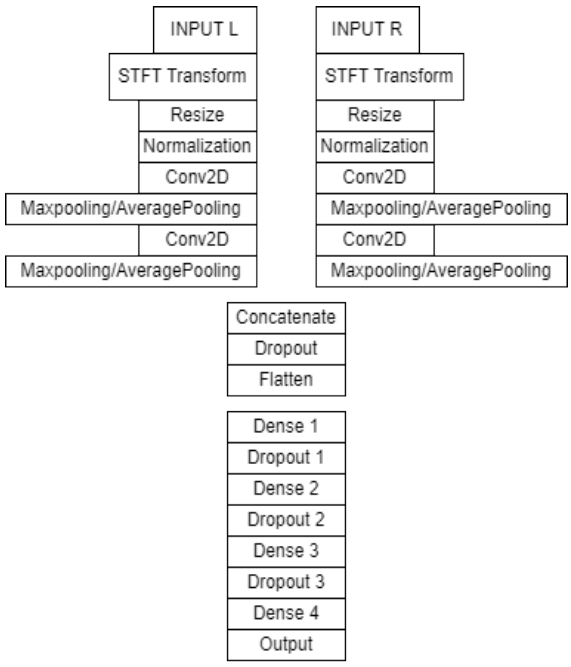


Fig 7. Model의 구성도이다

CNN-Model은 좌우 마이크의 입력을 따로 받아서 다음과 같이 병렬적으로 입력 신호를 처리한다. Input-STFT Transform-Resize-Normazlize and Convolution 순서대로 이루어져 있다. 그 이후, 좌우 신호를 마치 대뇌에서 종합하여 처리하듯, Concatenate Layer에서 이들 노드를 병합하여 Dense Layer를 처리하는 방식으로 모델을 구성하였다. 모델의 구조는 Google의 Wide & Deep [9]모델에서부터 영감을 얻었으며, OREILLY Hands-On 2nd Edition 310p의 코드[10], Tensorflow의 Simple_audio.ipynb[11], TensorflowIO의 포럼 [12]를 참고하여서 구성하였다. 각각

의 Layer에 대한 Activation Function은 Relu Function으로 통일하였다.

CNN은 Convolution 이라는 뜻으로, 특정 크기의 행렬을 계속해서 주어진 2차원(혹은 3차원)배열에 곱해주어, 그 크기를 줄여나가면서 중요한 정보만 남기는 것이다. Pooling은, Convolution의 데이터 크기를 감소시키며 가장 중요한 데이터만 남기는 과정으로, 이 설계에서는 Max-Pooling과 Average-Pooling으로 나누어 실험을 할 것이다. Max-Pooling은 주어진 격자 내에서 최댓값, Average-Pooling은 평균만 남겨서 압축을 하는 형태이다. CNN은 주로 이미지 처리를 하는 딥러닝에서 주로 사용되는데, 이를 짧은 소리 처리에도 활용할 수 있다.

1단계는, 소리를 Spectrogram, 즉 작은 이미지로 바꾸는 과정이다. 우리가 처리 할 소리는 1초 미만의 매우 짧은 소리들로 구성되어 있다. 이 소리를 Spectrogram으로 바꾸는 방식은 여러 가지가 있는데, 연구에서 사용할 방식은 STFT이다. 첫 번째 STFT는 Short Term Fourier Transform의 약자이다. Fourier Transform 이라는 것은 푸리에 변환으로, 여러 소리가 섞였을 때 그 소리의 음을 분석할 때 쓴다. 그러나 이 소리는 시간에 따라서 그 음이 빠르게 변하니까, 짧은 단위 시간 동안의 푸리에 결과를 Spectrogram으로 바꾸어주는 것이다. 이렇게 되면, 소리의 시간에 대한 주파수 정보, 주파수별 세기의 정보를 모두 얻을 수 있어 ITD (Interaural Time Difference), ILD (Interaural Level Difference), MSC (Monaural Spectral Cues)모두 고려한 방향 구분이 가능하다는 것이다.

2단계는 Convolution Layer로, 인공신경망에 적용하여 훈련하기에는 필요없는 정보가 많아 부적합한 데이터를 필요한 것만 추려서 압축을 하고, 추상화하는 Layer로, 위에서 설명한 Pooling기법으로 점차 크기를 줄여나간다.

3단계는 Dropout으로, 과적합을 방지하기 위해서 넣어주는 Layer이다. 과적합이란 학습하는 데이터에 과도하게 모델이 학습되어서, 되려 새로운 데이터에 대한 정답률이 낮아지는 것을 뜻한다.

4단계는 Dense Layer로, CNN의 출력을 점차 줄여나가면서, 우리가 필요한 데이터로 변환을 해가는 것이다. 우리가 최종적으로 필요한 데이터는 azimuth angle과 polar angle 두 가지이다. 둘 다 변역이 제한되어 있는 답안이라, 마지막 Layer는 Tanh으로 구성하였다. (Tanh의 출력값은 –1~1이다). 오차 분석 함수는 MSE를 사용하였고, learning rate = 0.00001인 SGD를 적용하여 훈련시켰다.

데이터 전처리 -

오디오 데이터의 전처리를 하는 과정이 까다로웠는데, Stereo로 녹음된 44100 Samplerate 1초 길이의 소리를 np.array로 변환 시킨 후에, 좌측 소리를 x0, 우측 소리를 x1으로 나누어 재배치하였다. 또한, y 값은 azimuth하고 polar angle로, 각각 360과 180으로 나누어서 정규화 시켜 주었다.

또한, 15000개 상당의 녹음된 파일들은 Google Drive를 통해 불러오고 unzip 시켜주었다.

IV. 고찰(Discussion)

TSM으로 더 많은 데이터 수집을 해서, 귓바퀴가 있을 때 없을 때, 그 각각의 형태에 대한 정확도 분석을 하려고 했으나, 시간 상 너무 부족하여서 다양한 데이터를 수집할 시간이 부족하였다. 초기 계획에 따르면 다양한 인공지능 알고리즘 (RNN, LSTM)을 구현해보고자 하였지만, 마찬가지로 시간이 부족하여서 간소화한 방식으로 구현을 마치고, 이러한 구현의 가능성을 확인하고 정확도를 알고리즘 내부 변수의 차이에 대해 비교하는 방식으로 연구를 변형하였다. 또한, 소리는 녹음되는 방향 뿐만이 아닌 소리가 발생한 공간의 입체적인 구조 또한 소리의 음향에 꽤나 큰 영향을 준다고 하는데, 이러한 영향은 무시된 연구인 것 같아, 다음에 후속 연구를 한다면 다양한 알고리즘을 구현해, 더 많은 데이터셋을 수집한 채로 공간의 입체적인 구조까지 분석 가능하도록 만들고 싶다.

딥러닝의 특성상, 훈련된 모델의 성능은 검증 가능하지만, 해당 모델이 어떠한 방식으로 답을 얻어내는지, 그 내부 알고리즘은 알기 어렵다. 현재의 모델 역시 마찬가지로, CNN 내부의 Pooling Layer에서, Mean pooling이 Max pooling 보다 월등한 성능을 가지는 것을 설명하기가 어렵다. 또한, 제어 가능한 수많은 변수에 대해 가장 적절한 형태가 무엇이고 그 이유는 무엇인지에 대한 고찰이 부족한 분야인 것 같다. 이러한 인공지능에 대한 원론적인 연구 또한 진행해보고 싶다.

V. 결론(Conclusion)

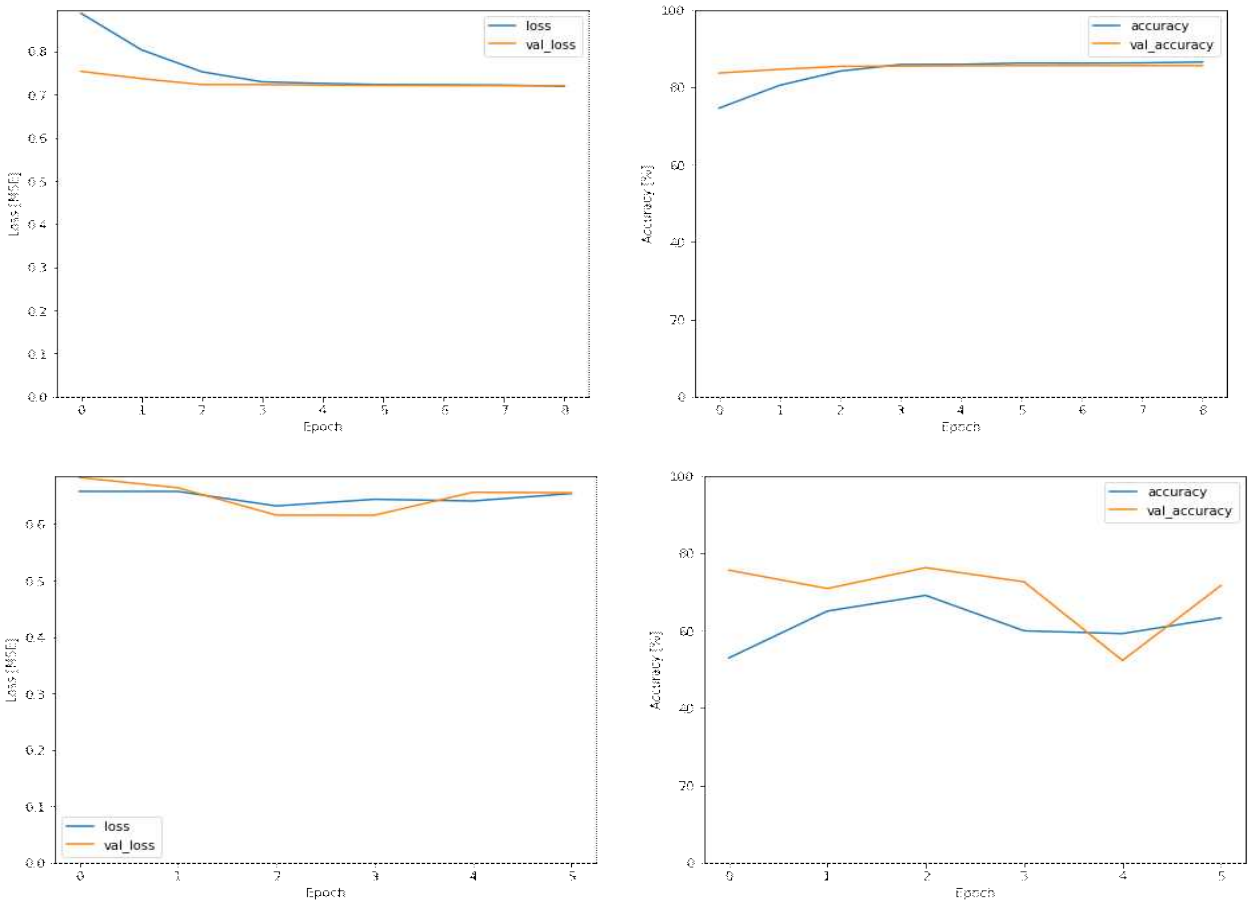


Fig 1. (a) Model (Mean Pooling)의 훈련 결과 : loss: 0.7193 - accuracy: 0.8646 - val_loss: 0.7207 - val_accuracy: 0.8554 (b) Model (Max Pooling)의 훈련 결과 : loss: 0.6540 - accuracy: 0.6325 - val_loss: 0.6553 - val_accuracy: 0.7159

그림은 Model을 Max Pooling과 Mean Pooling 각각으로 나누어 훈련한 결과이다. 임의의 자료에 대한 정확도를 나타내는 데이터인 Val_accuracy가 각각 0.8554와 0.7159인 것으로 보아, Mean Pooling이 Max Pooling보다 더 정확한 분석을 보여준다는 것을 알 수 있었다. 또한 Validation accuracy의 감소 양상을 볼 때, Mean Pooling이 훨씬 안정적으로 감소하므로 의도한대로 훈련이 되고 있다는 것을 알 수 있다.

비록 CNN의 변환과정에서, Mean Pooling과 Max Pooling이 어떠한 요소를 부각시키고 소거시키는지, 정확하고 명료하게 알 수 있는 방법이 없음에도 불구하고, 명확한 훈련 정확도 차이를 지니는 것으로 보아 ILD, ITD, MSC등의 요소를 Mean Pooling 이 훨씬 더 잘 잡아낸다는 것을 추론할 수 있었다. 귓바퀴의 구조와 딥러닝을 활용해서 소리 방향 구분에서 약 85.5%의 정확도를 가지는 모델을 설계하고 구성할 수 있었다.

VI. 참고 문헌(Reference)

[1] 서울과학고 동아리 LTE ANN 1, 2, 3, 4강

[2] 위키피디아, 딥러닝, https://ko.wikipedia.org/wiki/%EB%94%A5_%EB%9F%AC%EB%8B%9D

[3] 위키피디아, RNN, https://en.wikipedia.org/wiki/Recurrent_neural_network

[4] 위키피디아, CNN, https://en.wikipedia.org/wiki/Recurrent_neural_network

[5] 위키피디아, LSTM, https://en.wikipedia.org/wiki/Long_short-term_memory

[6] 김성필, 딥러닝 첫걸음, 한빛미디어, 2016, 전체

[7] 길민식1, 곽동걸2, 음원탐지 및 위치 추정 알고리즘을 이용한 방재용 IoT 디바이스 시스템 설계91.

[8] electronicshub.org, https://www.electronicshub.org/raspberry-pi-stepper-motor-control/

[9] Wide & Deep Model https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html

[10] OREILLY Hands-on Machine Learning, https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

[11] Tensorflow simple_audio.ipynb https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/audio/simple_audio.ipynb#scrollTo=zRxauKMdhofU

[12] Tensorflow forum , https://www.tensorflow.org/