과제번호	4.20494

2020학년도 학부생 연구프로그램(UGRP)

최종보고서

과 제 명	나만을 위해 연주하는 머신러닝 밴드 세션 생성 프로그램				
지원분야	■ Track 1: 그룹형 프로젝트				
연구기간	2020. 4. 20. ~ 2021. 1. 12.				
연 구 비	371,210				
지도교수	성명	유환조		학과	컴퓨터공학과
참여자 인적사항	구분	학과	성명	학번	연락처
	연구 책임자	컴퓨터공 학과	정준혁		
	공동 연구원	컴퓨터공 학과	손주은		
	공동 연구원	전자전기 공학과	윤선경		
	공동 연구원	전자전기 공학과	주효진		

2020학년도 학부생연구프로그램 최종보고서를 아래와 같이 제출합니다.

2021 년 1 월 13 일

연구 책임자 : 정 준 혁 (인)

연구지도교수 : 유 환 조 (인)

포항공과대학교 교육혁신센터장 귀하

과제번호	4.20494

2020학년도 학부생 연구프로그램(UGRP)

연구비 집행 결산보고

연구책임자(학생)	정준혁	소속/학번	컴퓨터공학과
과제명	나만을 위해 연주하는 머신러닝 밴드 세션 생성 프로그램		

■ 연구비 집행표

세부비목	당초 예산(원)	집행 예산(원)
연구장비 및 시설비	754,000	0
시약·재료비 및 전산 처리	100,00	371,210
시작품 제작비	475,700	0
수용비 및 수수료	0	0
기술정보활동비	0	0
국내여비	428,800	0
합계	1,758,500	371,210

☞ 연구비 예산에는 당초 연구비로 배정받은 금액을, 집행한 연구비는 중 최종 집행한 연구비를 기입함.

과제번호	4.20494

2020학년도 학부생 연구프로그램(UGRP)

최종보고서

연구책임자(학생)	정준혁	소속/학번	컴퓨터공학과
과제명	나만을 위해 연주하는 머신러닝 밴드 세션 생성 프로그램		

○ 연구목적

이 과제는 임의의 보컬 멜로디를 입력받아, 기타와 베이스, 키보드, 드럼 등의 악기 반주를 생성하여, All-Recorded(보컬+반주)와 Music-Recorded(반주) 파일을 출력하는 기기를 구현하는 데에 목적이 있다. 그리고 우리 팀의 기기를 사용하는 사람들은 두 가지 경험을 기대할 수 있다.

첫째로, 자신만의 멜로디로 제작된 음악을 즐길 수 있을 것이다. AR(보컬+반주)을 감상하거나, MR 반주를 노래방처럼 재생하여 노래 부를 수 있다.

둘째로, 일반인(비 음악가)이 작곡에 느끼는 진입장벽이 낮아질 것이다. 일반인이 작곡에 진입장벽을 느끼는 것은, 노래를 흥얼거리기는 쉽지만, 그 노래에 맞는 악기 반주를 만들기가 어렵기 때문일 것이다. 우리는 기존의 수많은 음악가들이 음악을 창작한 패턴을 활용함으로써, 사용자의 노래에 적절한 반주를 제공한다. 이로부터 사용자는 자신의 노래에 악기 반주를 덧붙이는 것에 편안하게 다가갈 수 있게 될 것이다.

○ 연구내용 및 진행

1. 프로젝트 구조

Creative Karaoke는 그림1과 같이 크게 Client side와 Server side로 이루어져있다.

Client side는 Raspberry Pi 와 모니터, 스피커, LED의 출력 장치와 마이크, 스위치의 입력 장치로 구성하였다. 사용자가 모니터로 출력되는 GUI를 확인하며 스위치로 원하는 장르와 모드, 템포 설정을 진행한다. 마이크로 사용자가 부른 노래를 녹음하고, 이 때 Fixed 모드 선택 시 bpm을 LED로 출력한다. 이후 Server side에서 완성된 음악을 raspberry Pi에서 받아 스피커로 출력한다.

Server side는 Node js와 Express 기반의 Web server로 구현하였다. Web server에서 실행하는 Creative Karaoke 프로그램은 Google의 오픈소스 라이브러리 Tensorflow를 기반으로 한다. Client side에서 사용자의 노래와 함께 요청을 받으면 Creative Karaoke 프로그램을 실행하여 생성된 음악을 보낸다.

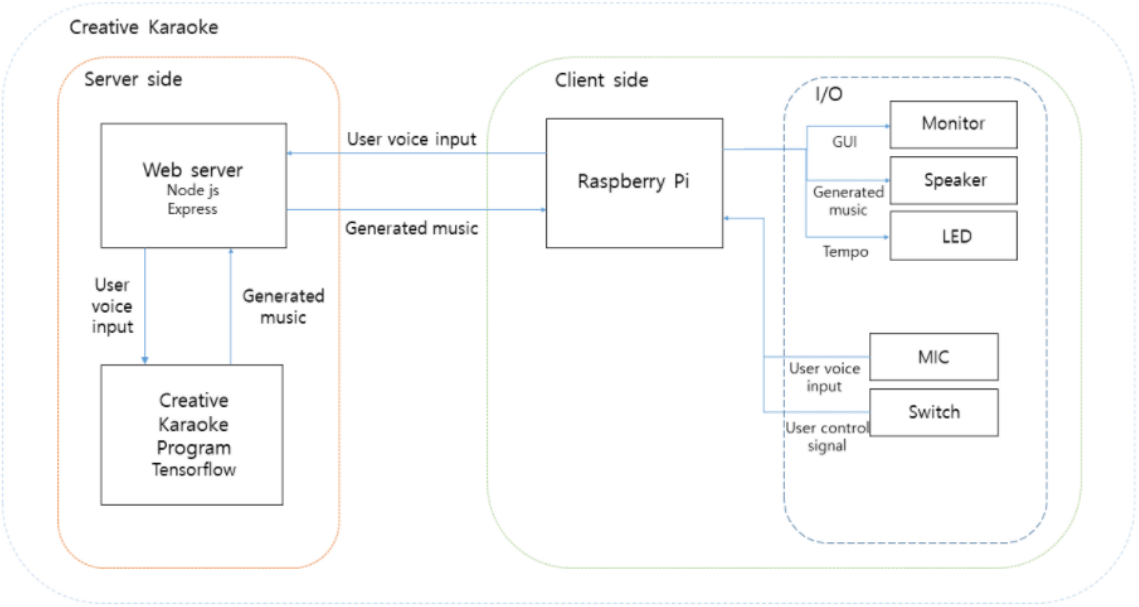

그림 1. Creative Karaoke의 구조

2. 기능 별 연구 내용 및 구현 방법

2.1. Client side

2.1.1 User flow와 UX 디자인

Client side의 제 1목적은 사용자의 편의에 적절히 맞아 떨어져야 한다는 것이다. 사용자 편의를 고려하기 위해 User flow를 먼저 확인했으며, 아래가 그 내용이다.

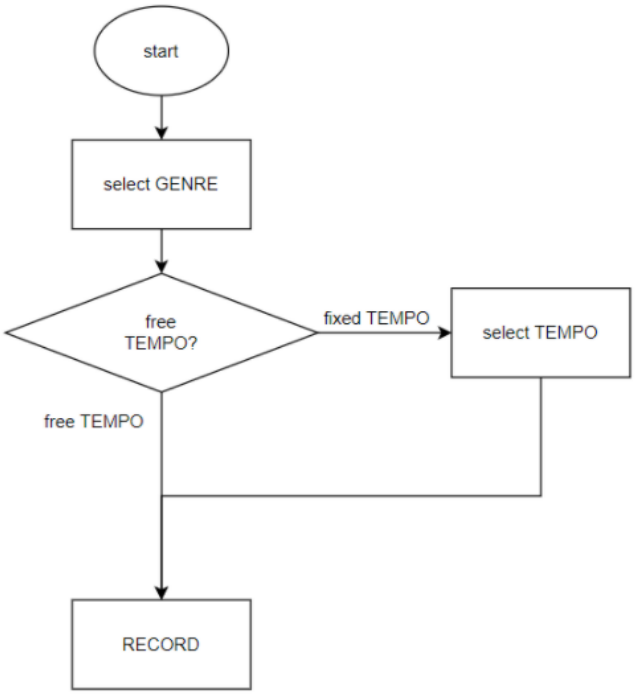

그림 2. User flow

악기 반주 (밴드 세션) 생성 프로그램에 필요한 input은 사용자의 목소리 외에도 장르와 템포가 있다. 사용자는 장치를 사용하면서 세 동작 (장르 선택, 템포 선택과 녹음)을 하게 되는데, 세 동작의 적절한 순서를 생각해보았다. 사용자는 부를 노래의 스타일을 결정한 후에 노래를 부를 것이므로, 장르 선택을 1번 동작으로 결정하였다. 또, 일정한 템포에서 노래를 부른다면 템포가 결정되어야 그에 맞추어 녹음할 수 있으므로, 템포 선택을 2번, 녹음을 3번 동작으로 결정하였다.

그리고 아래는 장르선택, 템포선택, 녹음의 순서의 순으로 디자인한 UX이다.

<장르 선택>

장르 선택 단계에서 사용자는 ROCK, JAZZ, CLASSICAL, DISCO 중 하나를 선택해야 한다. 장르를 선택할 때, 모니터에서는 아래의 화면을 띄울 것이다. 직관적으로 알 수 있듯이, 보라색 사각형이 현재 사용자가 선택한 장르이며, 사용자는 버튼 세 개를 이용할 수 있다. 왼쪽 버튼을 누르면 사각형이 왼쪽으로 이동하고, 오른쪽 버튼을 누르면 사각형이 오른쪽으로 이동한다. 그리고 가운데 버튼을 누르면 사각형이 위치한 장르로 선택되는 방식이다.

그림 3. 장르 선택 화면

<템포 선택>

위에서 가운데 버튼을 누르면 모니터는 아래의 화면으로 전환된다. 사용자는 fixed tempo와 free tempo 중에 하나를 고르면 되는데, fixed tempo는 기기가 제공하는 템포에 맞추어 노래하는 방식이며, free tempo는 말 그대로 자유롭게 노래하는 방식이다. 이 때 사용자는 버튼 두 개를 사용할 수 있으며, 1번을 누르면 fixed, 2번을 누르면 free tempo가 선택된다.

그림 4. 템포 선택 화면

fixed tempo를 선택할 경우, 초기 설정 값은 120 bpm이며, <장르 선택>에서와 마찬가지로 버튼 세 개를 이용할 수 있다. 왼쪽 버튼을 누르면 bpm 값을 낮출 수 있고, 오른쪽 버튼을 누르면 bpm 값을 높일 수 있고, 가운데 버튼을 누르면 bpm 값을 확정(즉, tempo를 선택)할 수 있다. 이 때, 사용자는 4개의 LED에 불이 들어오는 것을 보고 박자를 확인할 수 있다. 4개의 LED는 아래 그림처럼 일렬로 배치되어 있으며, O/X/X/X, O/O/X/X, O/O/O/X, O/O/O/O 의 네 상태를 번갈아가며 출력한다.

free tempo를 선택할 경우, 사용자는 별도의 조작을 하지 않아도 된다.

<녹음>

Fixed tempo에서 가운데 버튼을 누르거나, free tempo를 선택한 후에는 아래의 화면으로 전환된다. 이 때 사용자는 두 버튼을 사용할 수 있는데, 왼쪽 버튼을 누르면 녹음이 시작되고, 오른쪽 버튼을 누르면 녹음이 종료된다. 한편, fixed tempo를 선택한 경우에는 녹음이 종료될 때까지 4개의 LED에서 박자를 확인할 수 있다. 즉, LED의 박자에 맞추어 노래를 부르면 된다.

그림 5. 녹음 화면

<녹음 이후>

녹음을 마치면 더 이상 사용자가 할 일은 없다. 인공지능의 곡 처리 속도에 따라 30초에서 1분 이내로 기다리면 생성된 MR 파일을 들어볼 수 있다. 기다리는 동안에 모니터에는 아래의 왼쪽 화면이 띄워진다. 그리고 생성된 MR 파일이 재생되면서는 오른쪽 화면이 띄워진다.

프로그램이 종료된 후에는 제일 처음 장르를 고를 수 있는 화면으로 되돌아가게 되고, 다시 처음부터 과정을 반복할 수 있다.

그림 6. 녹음 이후의 화면

2.1.2 GUI 구현 방식

GUI는 Python Tkinter 라이브러리로 구현하였다. 화면의 경우 gif 이미지를 이용하여 보여주었으며, tempo와 같이 이미지로 보여주지 못하는 부분은 글자를 이용하여 보여주었다.

Tkinter 라이브러리에서 메인 루프를 실행하게 될 경우에 키보드나 마우스 이외의 Input을 받지 못하므로 우리가 구현한 버튼의 handler를 다른 thread로 열어서 사용자가 입력한 버튼 값을 받게 하였고, 사용자가 한 과정을 모두 마칠 경우에는 다른 thread를 또 열어서 다음 과정을 수행하도록 하였다. 각 과정은 함수로 만들어 thread로 실행할 수 있게 만들어두었다.

2.1.3 Buttons/LEDs

사용자의 입력을 받는 Button의 경우 복잡한 키보드를 두지 않고 사용자에게 필요한 버튼만을 이용하여 사용자의 User Experience를 높이고 키보드나 마우스가 없는 환경에서도 편리하게 이용할 수 있도록 하였다.

Raspberry Pi 4의 GPIO 핀들을 활용했다. 아래에 사용자가 button을 누르는 것을 인식하는 코드의 일부를 적고 설명했다. 언어는 python이다.

```
--------------------
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
        if GPIO.input(17) == False:
```

```
            print("Button is pressed")
----------------
```

GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)의 세 번째 항은 라즈베리파이 내부에 이미 만들어진 풀업 저항을 SW로 활성화하는 코드이다. 풀업 저항을 선택할 경우 button이 눌렸을 때 false, 풀다운 저항을 선택할 경우 button이 눌렸을 때 true임을 주의하며 코드를 작성하였다. 또한, button은 용도에 따라 두 개 혹은 세 개가 동시에 사용되므로, 각 button을 별개의 GND에 연결하여, 풀다운 저항으로 연결된 여러 개의 button 인식에 혼선이 생기지 않도록 하였다.

LED의 깜빡임을 제어하는 코드는 위와 크게 다르지 않다. GPIO.setup에서 GPIO.IN 대신 GPIO.OUT을 적으면 되고, LED를 켜고 싶을 때 GPIO.output(핀 번호, True)를, 끄고 싶을 때 GPIO.output(핀 번호, False)를 적으면 된다.

2.2. Server side

2.2.1 Web server

초기 계획에서는 Raspberry pi에서 직접 Creative karaoke 프로그램을 실행하여 결과물을 출력하고자 하였으나, Raspberry Pi 4의 OS 환경과 프로그램의 요구사항이 호환되지 않아 별도의 서버를 구매하여 Node.JS Express기반 웹 서버를 구축하였다. 서버의 경우 머신러닝을 돌릴 수 있는 환경을 Google Cloud Platform에서 구축하였다.

Raspberry Pi에서 사용자가 GUI를 통해 입력한 녹음 데이터, 녹음 모드, 템포, 장르 정보를 웹 서버로 보내면 웹 서버는 정보를 Creative Karaoke 프로그램의 인자로 사용하여 실행한다. 프로그램이 성공적으로 수행되면 4개의 음악 세션 파일을 생성하게 되고, 성공했다는 신호를 client에 전송하게 된다. 이를 확인한 client는 주소 '/0, /1, /2, /3'에서 생성된 각 세션의 데이터를 가져온다.

2.2.2 Creative Karaoke program

Github Repository: https://github.com/UGRP2020/Creative_Karaoke

2.2.2.1 Data Flow Diagram

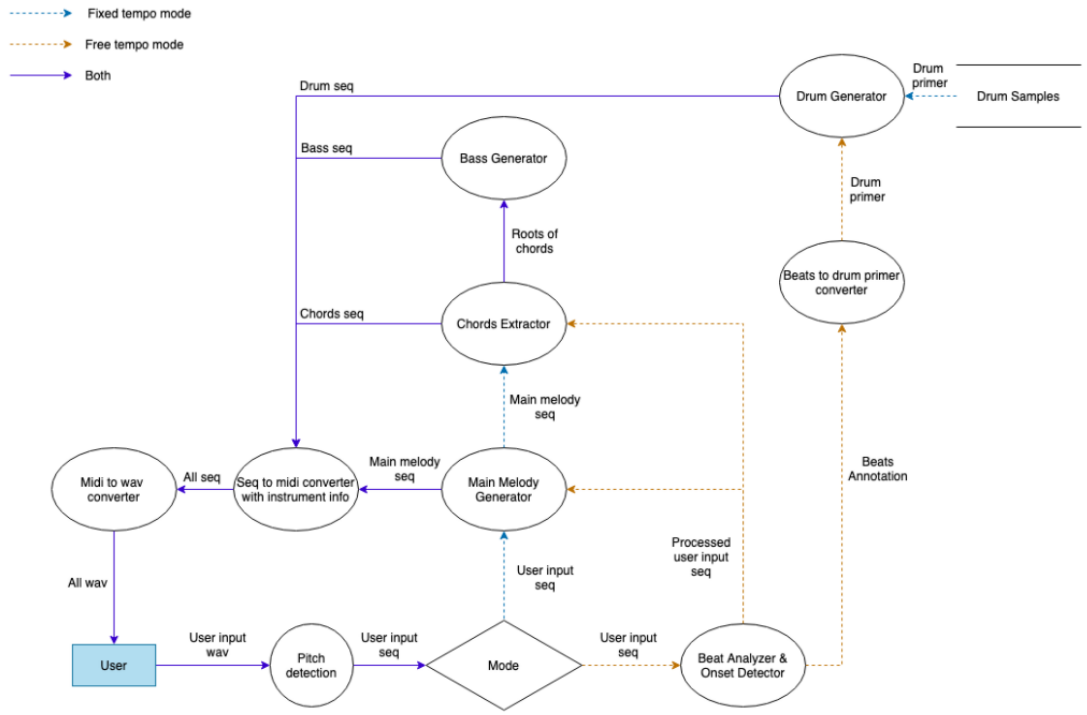

그림 7. Data flow

2.2.2.2 프로세스 별 세부 기능 및 구현 방법

(i) Pitch detection 및 correction

Pitch detection 은 사용자의 노래를 RNN 모델의 인풋으로 사용될 수 있는 형태로 양자화하는 단계이다. 두 가지 방법으로 pitch detection 을 진행했다. 첫 번째 방법은 멜로디아 논문 기반의 atmm 라이브러리를 활용하는 방법 [1]이다. 인풋으로 사용자 노래를 wav 파일 형식으로 입력하면, tempo, smoothness 등의 인자를 입력받아 미디파일과 같은 양자화 된 형태로 바꾸어주는 라이브러리이다. 사용자 노래에는 불안정한 음정이 섞여있을 수밖에 없는데, 이를 다듬어주는 pitch correction 작업을 smoothness 인자를 통해 조절할 수 있다. 여러 input 에 대해 실험해본 결과, 다른 pitch 를 가진 음들이 연속적으로 나오는 것은 부른 이의 의도대로 잘 변환해주지만 같은 pitch 의 음들을 연속적으로 부를 때는 하나의 긴 음으로 인식한다는 단점이 있었다. 또한 tempo 도 직접 입력해야하기 때문에 tempo 를 염두에 두지 않고 부르는 경우에는 사용하기 곤란하다는 단점이 있다.

atmm 라이브러리의 이러한 단점들 때문에 추가적인 pitch correction 알고리즘을 개발했다. 먼저 crepe 라이브러리를 통해 사용자의 음원에서 각 time stamp 에서의 frequency 및 음이 있을 확률 (&confidence&) 정보를 얻을 수 있었다. 이를 바탕으로 confidence 가 상대적으로 낮아지는 지점을 음과 음 사이의 쉼으로 인식하게 했다. 또한 불안정한 음정의 경우에는 frequency를 양자화했을 때 두 음 사이를 짧은 시간 안에 왔다 갔다하는 양상을 확인할 수 있다. 이러한 &trill (트릴)& 들의 산술 평균을 구해 더 우세한 하나의 음으로 합쳐주는 기본적 trimming 작업을 진행했다.

(ii) Beat Analyzer & Onset Detector

각 파트별 음원을 생성할 때 공통된 마디구조가 없다면 같이 들었을 때 싱크가 맞지 않아 듣기에 상당히 낯선 음악이 될 것이다. 때문에 각 파트 생성 전에 하나의 마디 구조를 생성했다. 이를 위해서

Fixed Tempo 및 Free Tempo 의 두가지 모드를 제공하고 있다. 먼저 Fixed Tempo 모드에서는 사용자로부터 템포를 입력받고 녹음 시작 버튼을 누르는 순간을 곡의 시작으로 인식하기 때문에 마디 구조가 자동으로 결정이 된다.

반면에 free tempo 모드에서는 사용자가 부른 곡의 템포 혹은 마디 구조를 직접 파악해야한다. 이를 위해 beat detection 을 해주는 AUBIO 라이브러리를 이용해 곡에서 강박으로 인식되는 시간을 추출해 rnn 모델들이 인식할 수 있는 annotation 형식으로 변환해주었다. 이렇게 뽑아낸 beat 정보를 바탕으로 beat 와 beat 사이를 한 덩어리로 처리해 이후의 chord extraction 및 melody generation 을 진행했다.

(ⅲ) Main melody generator

사용자의 노래를 기반으로 32마디 길이의 Main 멜로디를 생성하는 프로세스이다. 사용자가 어떠한 길이의 노래를 입력하더라도 제약없이 사용할 수 있도록 하기 위해 사용자의 입력이 지정된 길이보다 짧은 경우, 사용자의 멜로디를 Primer melody로 사용하여 어울리는 새로운 멜로디를 길이에 맞게 작곡한다.

멜로디의 작곡을 위해 Tensorflow 기반 라이브러리 Magenta의 Melody RNN을 활용하였다. 학습 데이터의 경우 Song galaxy midi site [2]에서 대중가요를 python crawling을 통해 수집하여 미디 데이터셋(9,569개의 미디 파일)을 형성하였다. Note sequence 타입으로 변환 후, attention RNN을 배치 사이즈 64, RNN layer 사이즈 [64, 64]로 설정하고 9000 training step 만큼 학습시켰다. Note sequence는 Magenta에서 음악 데이터를 다루는 데이터 타입으로, Pitch, start time, end time, velocity, qpm 정보를 가지는 Notes의 sequence로 이루어져 있다.

학습이 완료되었을 때 모델의 checkpoint, metadata 등의 정보를 담고 있는 Bundle 파일을 생성하고, 이를 Melody Generator에서 활용하며 멜로디를 작곡한다. 기존 RNN이 입력 시퀀스의 길이가 길어졌을 때 출력 시퀀스의 정확도가 떨어지는 문제점을 보완하는 Attention RNN의 특성 [3]에 따라 입력한 사용자의 노래와 높은 연관성을 가지는 멜로디가 출력된다. 이는 멜로디에 적당한 반복과 일관성을 형성하여 사용자가 &좋은 음악'으로 인식할 확률을 높여준다.

Fixed tempo mode에서는 User input sequence와 지정된 템포를 인자로 받아 멜로디를 생성하고, Free tempo mode에서는 Beat analysis 와 Onset detection을 거쳐 beat annotation 정보를 가지고 있는 processed user input sequence를 통해 멜로디를 생성한다.

(ⅳ) Chords Extractor

Main 멜로디에서 코드 진행을 추출하는 프로세스이다. 추출한 코드를 음악 세션으로 사용하고, 코드에서 근음을 추출하여 Bass generator에 넘겨준다. 코드 추출은 Magenta에서 제공하는 Chords inference 함수를 활용해 진행하였다.

Fixed tempo mode에서는 Main melody를 받아 마디 별로 코드를 추출한다. Free tempo mode에서는 Beat annotation 정보를 가지는 user input sequence를 입력받아 beat 마다 코드를 추출한다.

(ⅴ) Bass Generator

음악 세션 중 베이스를 생성하는 프로세스이다. 베이스는 곡을 구성하는 코드의 근음을 기반으로 하여 리듬에 맞게 연주해야 한다.

Fixed tempo mode에서는 Chords extractor에서 받은 근음 sequence의 모든 note에 대해 Melody

RNN에 primer melody로 입력하고, 한 마디 길이의 멜로디를 생성해 이어 붙이는 방식으로 베이스를 생성하였다. 멜로디의 랜덤한 정도를 나타내는 temperature를 Main 멜로디보다 낮게 설정하여 안정감 있는 베이스 멜로디를 생성할 수 있도록 하였다.

Free tempo mode에서는 Chords extractor에서 받은 근음 sequence가 beat annotation 정보를 가지고 있으므로, 이를 melody RNN에 primer로 입력하여 곡의 전반에서 베이스가 user input sequence에서 추출한 beat를 따르도록 하였다.

(vi) Drum Generator

음악 세션 중 드럼을 생성하는 프로세스이다. 드럼은 곡의 전반적인 리듬을 결정하므로, 미디 사이트 [4] 각 장르에 맞는 드럼 샘플을 구해 Primer 드럼으로 사용하였다.

드럼 라인은 Magenta에서 제공하는 Drum RNN을 활용해 생성하였다. 9 Piece drum kit(bass drum, snare drum, closed hi-hat, open hi-hat, three toms, crash cymbal, ride cymbal)을 모두 활용하는 drum kit configure를 설정하였다. 장르 선택이 Classical인 경우에는 드럼 라인을 생성하지 않는다.

Fixed tempo mode에서는 Jazz, Rock, Disco에 맞는 드럼 샘플에서 사용자가 선택한 장르에 따라 Primer를 입력해 드럼 라인을 생성하였다. Free tempo mode에서는 드럼 샘플을 사용하지 않고, Beat analyzer에서 추출한 beat annotation 정보를 바탕으로 bass, snare, hi-hat을 적절히 조합해 직접 drum primer를 생성해 드럼 라인을 만든다.

(vii) Seq to Midi converter with Instrument info

Note sequence 타입으로 생성된 각 음악 세션을 장르에 따라 악기를 설정하고 미디 데이터로 출력하는 프로세스이다. 악기 설정은 General MIDI sound set [5]을 따르도록 하였다. 장르 별 악기 설정은 표와 같다.

	Jazz	Disco	Rock	Classical
Main Melody	Alto Sax(66)	Synth Voice(55)	Overdriven Guitar(30)	Violin(41)
Chords	Grand Piano(1)	Voice Oohs(54)	Electric Piano 2(6)	String Ensemble 1(49)
Bass	Slap Bass 1(37)	Slap Bass 1(37)	Electric Bass(pick)(35)	Contrabass(44)
Drum	Jazz Drum	Jazz Drum	Standard Drum	-

표 1. 장르별 악기 설정

○ 연구결과 및 활용방안

1. 연구결과

본 연구는 사용자가 입력한 멜로디에 대해서 이와 어울리는 3가지 악기 세션을 자동으로 생성하는 장치를 제작하는 것을 목표로 진행되었다. Client side와 server side로 나누어 장치를 제작하였다.

Client side는 Raspberry pi와 세 출력장치 (모니터, 스피커, LED), 두 입력 장치(마이크, button)으로 구성하였다. GUI 구현과 button, led 제어하는 방법을 배우고, user flow를 고려해 client side를 완성하였다.

Server side는 Node js와 Express 기반의 Web server에서 Tensorflow 기반의 Creative Karaoke 프로그램을 실행하도록 구현하였다. 사용자의 노래에 어울리는 악기 세션을 생성하기 위해서는 음악의 구성에 대한 이해가 필요했다. 곡 전체가 같은 리듬을 가지도록 하기 위해 사전에 지정된 tempo를 따라 녹음하도록 하는 Fixed tempo mode와 사용자의 노래에서 리듬을 추출하는 Free tempo mode를 구현하였다. 다른 방식으로 지정된 리듬에 따라 각 세션을 생성하는 프로세스를 구현하였는데, Main melody generator 및 Bass generator에서 Attention RNN을 직접 학습시켜 새로운 멜로디를 생성하였다. Chords extractor는 생성된 멜로디에서 코드 진행을 예측하여 메인 멜로디와 같은 길이의 세션으로 출력하도록 하였다. Drum generator는 장르에 따라 미리 저장된 드럼 샘플 또는 사용자의 노래에서 추출한 리듬을 기반으로 직접 생성한 드럼 라인을 Primer로 사용하여 Drum RNN을 통해 생성하였다. 각각의 프로세스에서 생성한 세션에 장르별로 다른 악기 소리를 씌워 최종적으로 출력하였다.

2. 활용방안

사용자는 장치를 사용한다는 것 자체로 두 경험을 기대할 수 있는데, 본 보고서의 연구목적에서도 기술했듯이, 장치의 사용자는 자신만의 노래를 감상하고 가창하며 즐길 수 있고, 작곡에 느끼는 장벽을 완화할 수 있다. 한편, 본 연구의 결과물 장치는 단순한 장치 사용 외에도 다양한 활용방안이 있을 것이다.

먼저, 향후 노래방 산업과 협력해 나만을 위한 MR에 맞춰 노래하는 것이 자연스러운 일이 될 수 있다. 예를 들어, 사용자가 불렀던 멜로디와 결과물(AR과 MR)을 사용자과 mapping하여 저장해두었다가, 사용자가 노래방에서 본인의 id에 접속하면 결과물에 접근할 수 있게 하는 방식이 될 것이다.

또, 혼자 집에서 노래를 부를 때, 본인의 노래에 맞춰 화음을 쌓아주는 기계로도 활용될 수 있다. 특히 코로나 바이러스로 인해 노래방에 가지 못하고 집에서 혼자 노래를 부르는 사람이 많다는 점에서 활용성이 커질 것으로 보인다.

○ 참고문헌

[1] J. Salamon and E. Gómez, "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics", IEEE Transactions on Audio, Speech and Language Processing, 20(6):1759-1770, Aug. 2012.

[2] Song Galaxy, <https://songgalaxy.com/multi.php?cat=Pop> (2020.07.30.).

[3] Chaudhari, S., Polatkan, G., Ramanath, R., & Mithal, V. An attentive survey of attention models. arXiv 2019. arXiv preprint arXiv:1904.02874.

[4] Drum beats, <http://midi.teragonaudio.com/files/beats.htm> (2020.12.29)

[5] "GM 1 Sound set", MIDI Association,
<https://www.midi.org/specifications-old/item/gm-level-1-sound-set> (2021.01.10)