**컴퓨터공학 종합설계 6조 S/W 최종 보고서**

**우상욱, 고미진, 황혜영**

**- 목차 -**

1. **어플리케이션 개요**

**1.1 기능**

**1.2 개발 동기**

**1.3 기존 개발 사례**

1. **시스템 구성**

**2.1 사용 라이브러리**

**2.2 시스템 / 앱 구조도**

**2.3 데이터 플로우 시나리오**

1. **구현 방법**

**3.1 채점 기능**

**3.2 보컬 분리 기능**

**3.3 UI**

1. **성능 평가 및 보완점**

**4.1 채점 부분**

**4.2 그 외**

1. **참고 자료**
2. **발표 링크**

**1. 음악 파일 채점 어플리케이션 “아무곡나” 개요**

**1.1 기능**

|  |  |
| --- | --- |
| **기 능** | **기능의 간략한 설명** |
| 주파수 분석  (실시간, 단번에) | 사용자가 실시간으로 노래를 부르는 것을 스마트폰 마이크로 입력받아 실시간으로 주파수 분석을 하고 음정 체크(실시간) 분리된 보컬에 대해 단번에 주파수 분석을 하고 음정체크(단번에) |
| 보컬 분리 | 음원에서 보컬 파트를 추출해 음정 답안지를 만드는 데 사용함 |
| 파일 시스템 | 사용자와 서버의 사이에 있는 파일 공유 storage |
| 서버 사용자 간 통신 | 사용자와 파일공유 storage, 파일 공유 storage와 서버 간의 통신 |
| 점수 생성 | 정답음정과 사용자 음정의 유사도를 구해 점수 도출 |
| pitch shifting | 사용자의 키 조절 정도를 받아 음악의 키 조절 |

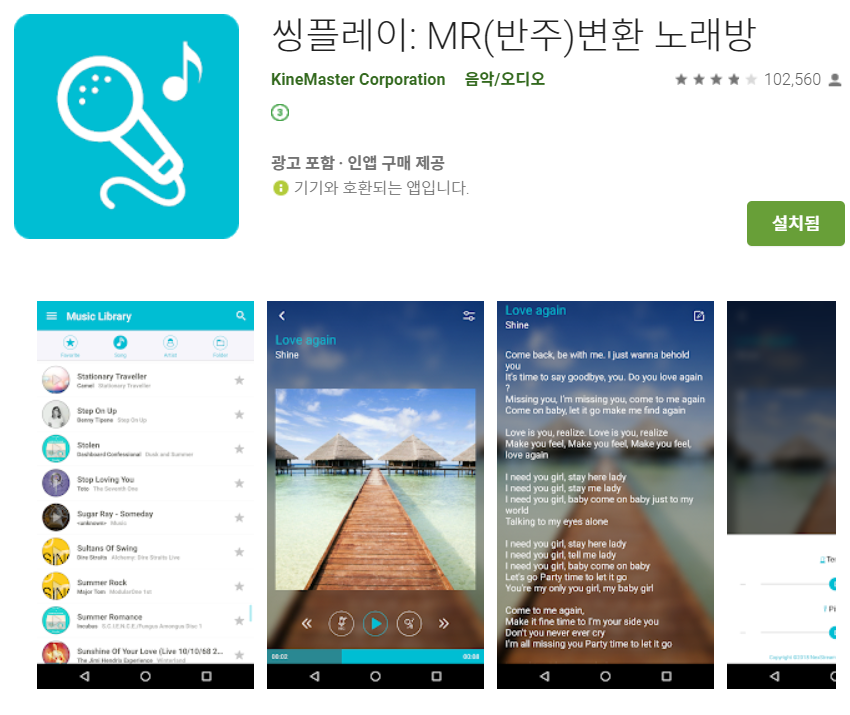
**1.2 개발의 필요성**

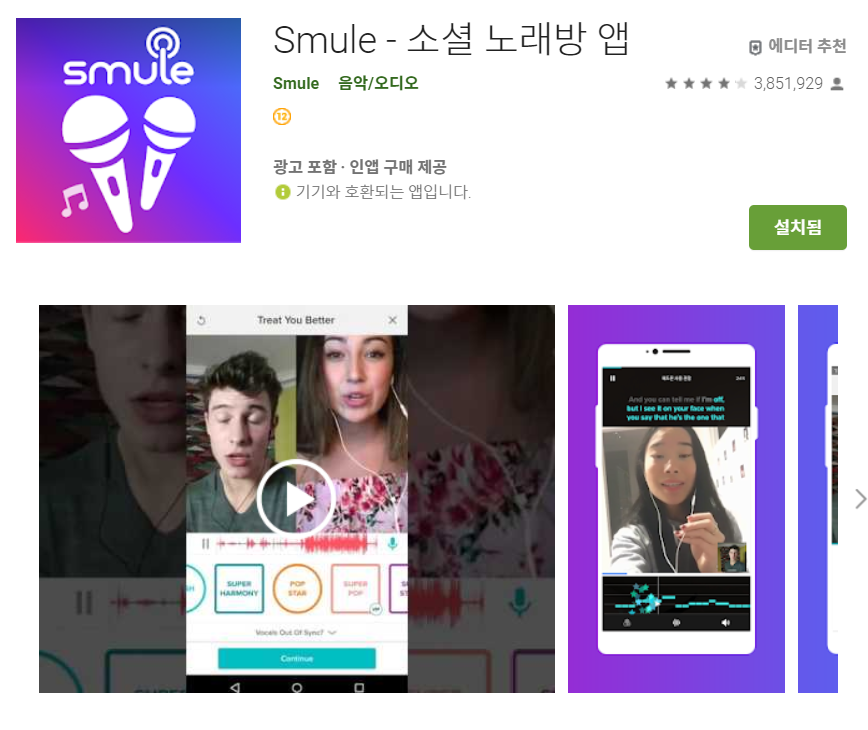
노래를 부를 때 음정을 맞추기 어려운 사용자를 위해 모바일에서의 음정 채점 기술을 도입해 자유롭게 어디서나 음정을 정확하게 맞출 수 있도록 트레이닝 할 수 있게 한다.

노래방 기기에서 지원하지 않는 다양한 노래와 기존의 노래라 할 지라도 퍼펙트 싱어와 같이 음정 채점을 즐기고 싶어하는 소비자들의 니즈를 충족시킬 수 있다. 또한 기존의 유사 서비스들은 정답 음정의 레이블링이 필요하다는 점에서 음정 채점을 제공하는 곡의 수가 한정적이다. 이 앱은 보컬의 정답음정을 자동으로 생성해 주어 음정 레이블링의 수고를 덜고, 사용자가 원하는 아무 음악에 대해서 곧바로 음정채점을 진행할 수 있도록 하는 데 의의가 있다.

**1.3 기존 개발 사례**

팀에서 개발한 “아무곡나”와 기존에 나와있는 비슷한 사례들을 선정하여 비교해보았다. 사례 선정 기준은 기존에 사용되고 있는 프로그램 중 구글 플레이스토어에 출시된 경우 1000만 이상의 다운로드 수를 가지고 있거나, Tj 노래방 퍼펙트 스코어 같은 경우 이미 전국의 노래방에 탑재되어 많은 사람들이 이용하고 있고, 저희가 개발한 기능과 겹치는 기능이 존재하며, 서로 다른 자신만의 기능이 있는 것들을 기준으로 선정하게 되었다.

  
**[[1]](#footnote-1)** 씽플레이 : 가지고 있는 파일을 MR로 변환해주는 기능의 노래방 어플



**[[2]](#footnote-2)** Smule : SNS와 같이 자신의 노래를 공유 및 연결해주는 기능의 노래방 어플



[[3]](#footnote-3) Tj 노래방 퍼펙트 스코어 : Tj media의 노래방 기계에 탑재된 기능으로 음정과 박자, 노래의 바이브레이션 등의 부수적인 기술까지 채점해주는 프로그램

아래 표는 기존 유사 개발 사례들과 본 프로젝트와의 차이점을 간략하게 정리한 내용이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 씽플레이 | Smule | Tj 노래방 퍼펙트 스코어 | 아무곡나 |
| 음정 채점 기능 탑재 | X | 탑재되었지만 부정확함 | ○ | ○ |
| 가사 지원 | ○ | ○ | ○ | X |
| MR 생성 | 깔끔하지 않게 생성 | 기존에 등록된 MR만 생성 | 노래방에 등록된 MR만 생성 | X |
| 모든 음악 지원 | ○ | X | X | ○ |
| 정답 음정 UI 지원 | X | ○ | ○ | ○ |
| 채점 결과 표시 | X | X | ○ | ○ |

위 표의 내용과 같이 본 프로젝트는 음정 채점을 지원하면서 기존 노래방에 없거나, 자작곡 같이 음원 사이트에 등록되지 않은 곡이든지 보컬 추출과 피치 디텍팅을 통한 자동 음정 레이블링 과정을 통해 음정 채점이 가능하도록 차별점을 두었다.

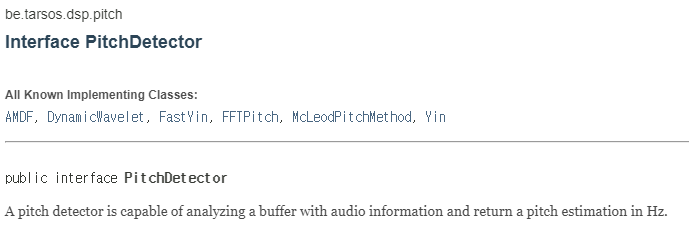
**1.4 시스템 제약사항**

|  |  |
| --- | --- |
| **기 능** | **관련된 제약사항** |
| 주파수 분석 | (실시간) 사용자의 목소리와 섞이지 않도록 이어폰으로 원래 음악을 듣는 것 권장 (단번에) 안드로이드 환경으로 전달된 분리된 보컬에 대해 주파수 분석을한다. |
| 보컬 분리 | 8기가바이트 이상의 내장메모리를 가진 그래픽 카드가 필요하다. 그렇지 않고 cpu로 돌리게 되면 4분 길이의 음원 기준 보컬 추출 시간 소요가 2~2분 정도 관측된다. |
| 파일 시스템 | 안드로이드 자바에서, 서버 측 파이썬에서 파이어베이스 storage에 접근 가능 해야한다.  storage의 업데이트에 서버가 빠른 속도로 반응할 수 있어야 한다. |
| 점수 생성 | 정답 음정과 사용자 음정이 반음 정도 차이 나는 선에선 정답으로 인정, 유사도의 퍼센티지 도출 |
| pitch shifting | 사용자가 불러야할 정답 보컬과 듣고 따라부르는 원본 음악 둘 다에 적용되어야한다. |

**2. 시스템 구성**

**2.1 사용 라이브러리**

**2.1.1 TarsosDSP**

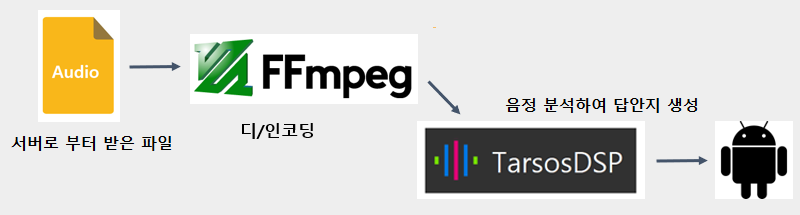
자바 기반의 실시간 오디오 처리 라이브러리이다. 실시간 녹음에서의 주파수 측정과 분리된 보컬 파일의 음정을 분석하는데 사용된다. 자바 기반으로 안드로이드에서도 일부 기능을 지원함으로 사용하기 더 용이하다.  


위는 음정 분석에 사용한 인터페이스 클래스이다. 음악 파일에 대해서, 음정을 분석한 후에 해당 음에 대한 주파수와 시간 대를 EventHandler로 함수로 얻어 낼 수 있다.  
 음정 분석에 사용한 라이브러리의 알고리즘은 FFT\_YIN 알고리즘으로, 일반 YIN 알고리즘에 좀 더 빠른 속도를 위해 푸리에 변환을 사용한 것이다. 사람 목소리가 아닌 다른 소리에 대한 인식률이 낮기 때문에 목소리의 주파수를 분석하는데 있어서 더 효율적이다.

음정 분석에서의 한계점이 존재하는데 음악에서

**2.1.2 FFmpeg**

TarsosDSP에서 사용 가능한 형태로 음악 파일들을 디코딩, 인코딩 하는데 사용되는 라이브러리이다.

 개발 초기에 FFmpeg 자체를 안드로이드 NDK를 이용하여, 직접 컴파일하여 앱에 구현하려 하였지만, 안드로이드 최신 버전에서 안드로이드 OS 내부의 커널을 이용하여, FFmpeg 명령어를 직접 실행하는 것을 자체를 보안 문제로 인해 더 이상 지원하지 않았다. 대신에 안드로이드 자바 환경에서 코드로 FFmpeg 명령어를 사용할 수 있게 해주는 mobile-FFmpeg 라이브러리로 대체하였다.

**2.1.3 Demucs**

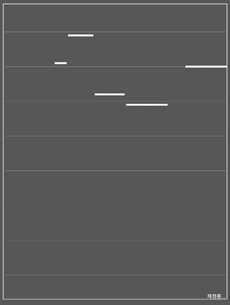
python기반의 음악 소스 별 분리 딥러닝 모델로 보컬을 뽑아내기 위해 사용하였다.

**2.1.4 Librosa**

python 기반의 음악 분석 라이브러리로 pitch shifting을 위해 사용하였다.

**2.1.5 MPAndroidChart**

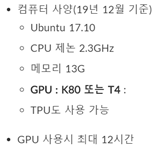
MPAndroidChart : Philipp Jahoda가 개발한 안드로이드를 위한 차트 라이브러리. 안드로이드 상에서 여러 종류의 차트를 그리고 조작할 수 있도록 해준다. 본 프로젝트는 Line Chart를 사용하여 데이터를 표시하고, 화면 뷰포트를 움직여 시간이 흐를수록 정답 음정 차트가 움직일 수 있도록 구현하였다.



**2.2 사용 프레임워크**

**2.2.1. colab**

구글의 코랩은 무료로 다음 사양의 서버 컴퓨터를 ipynb의 백엔드로서 제공한다.

****

또한 다음과 같은 추가적인 특징을 갖는다.

- 구글 드라이브를 파일시스템 마운팅 하여 읽고 쓸 수 있다.

- cuda, cudnn, 기본적인 python 라이브러리가 미리 설치되어있다.

- linux 커맨드를 ipynb에서 사용할 수 있으며 시스템 파일의 조작이 제한적으로 가능하다.(추가적 라이브러리를 설치할 수 있다.)

- firewalled 되어있어 패키지를 다운받는 것 이외의 통신은 제한된다.

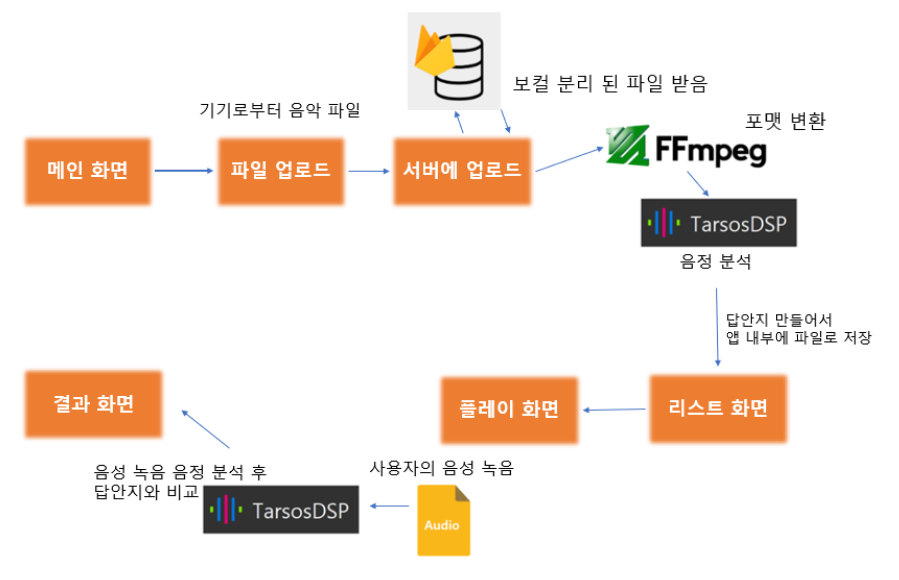
**2.2.2 firebase storage**

bucket에서 발급받은 인증 키 json파일을 통해 인증 된 사용자에 한해서 파일 업로드 다운로드가 가능하다.

android java와 서버 python에서 둘 다 접근 가능하다.

bucket에 있는 파일들의 목록을 불러오는 것 가능하다.

**2.2 앱 구조도 (사용자 관점)**

**[[4]](#footnote-4)**화면 순서도

**[[5]](#footnote-5)**화면 별 기능

**메인 화면**

**업로드**

**녹음**

**옵션**

**파일 업로드**

**내부 저장소에서 업로드**

**외부 저장소에서 업로드**

**파일 변환**

**대기 화면**

**진행도**

**곡 장르 선택**

**키 조절 선택**

**음악 리스트**

**음악**

**시작**

**미리 듣기**

**결과 화면**

**점수**

**녹음**

**실시간 음정 비교 아이콘**

**정답 음정 차트**

**옵션**

**더 쉽게**

**옵션**

**더 세세하게**

**옵션**

**2.3 데이터 플로우 시나리오**

외부와의 통신이 제한된 colab 컴퓨터를 서버로서 사용하기 때문에, 파일을 업로드 다운로드 할 수 있는 클라우드 데이터베이스인 firebase storage를 파일의 경유지로 사용하여 음정 분리 및 키 변환 요청을 보낸다. colab 컴퓨터는 storage 버킷을 확인하며 새로운 파일들을 fetch하는 방법을 사용했다.

1) 사용자가 새로운 음악에 대해서 음정 채점을 하고자 할 때



위 그림에서 화살표는 해당 파일의 이동경로이며 화살표 위 숫자는 진행 순서이다.

colab 컴퓨터에서 새로운 음악파일들을 다운로드 하여 보컬 분리 및 pitch shifting 하여 처리 결과물을 다시 스토리지에 올린다.

이후 이 파일을 올렸던 안드로이드 핸드폰이 처리 결과물을 다운 받아 주파수 분석을 통해 음정 답안지를 생성, 로컬 스토리지에 저장한다.

2) 사용자가 한 번 음정 채점을 받았던 음악에 대해 다시 음정 채점을 하고자 할 때



사용자의 핸드폰에 음정 답안지가 남아 있을 것이므로 보컬분리, 피치 변경, 주파수 분석을 하지 않고 음정 답안지를 로드 해 사용한다.

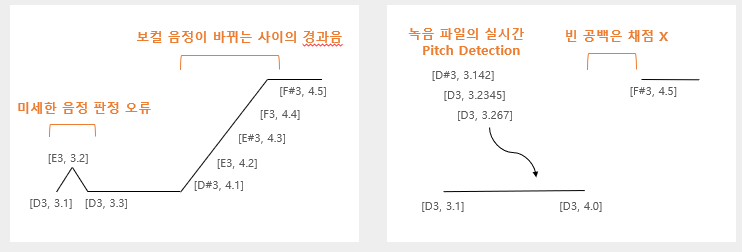
**3. 구현**

**3.1 채점 기능**

**[ [ ‘음계’, ‘타임 스탬프’ ], [ ‘음계’, 타임 스탬프’ ], … ] 형태로 저장**

**하나의 음 당 2개의 리스트(시작 시간 - 끝나는 시간)로 판단**

* **기본 아이디어(답안지)**

****

음정 분석의 간격 자체는 위 그림과 다르게(위는 예시) 약 0.05초 간격으로 이루어지게 된다. 분리된 결과에서 근소하게 흔들리는 음은 채점에서 도움이 되지 않기 때문에 제거 하고, 음에서 음으로 옮겨가는 과정에서의 경과음도 그 간격에 있어서 너무 좁은 간격에서 변화할 경우, 채점에 사용되지 않도록 구현하였다.

구현 시에는 3개의 함수로 나뉘며 각각 인자로 다음과 같이 가진다

|  |  |  |
| --- | --- | --- |
| **함수명** | **인자** | **기능** |
| TrimAnswerSheet | 다른음으로 인정되는 같은 음의 시간 간격 | 음이 연속적으로 나타나지 않고 바뀔 경우 null로 대체, 같은 음이더라도 간격이 길경우 삭제 |
| removeNull\_sNotes | 채점에 사용될 수 있는 음 하나의 최소길이 | null값 아이템 삭제 + 음이 너무 짧을 경우 삭제 |
| extendTimeStamp | 삭제되어 빈 공간을 채울 음의 길이 | 삭제되어 빈 공간이 된 음을 확장시켜서 시간을 늘림 |

3가지 함수를 인자 값들을 조정하며, 호출을 반복하여 답안지를 만들게 된다. 장르 별로 각 인자값을 조절하여 다른 방법으로 답안지를 만들 수 있다.

* **진행 과정(답안지)**

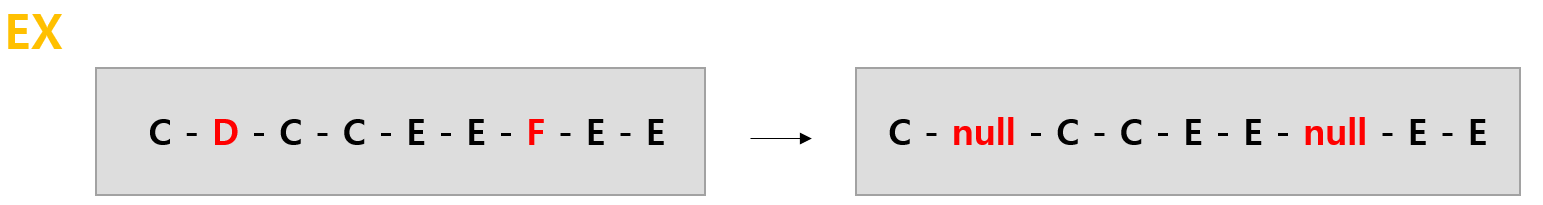
음정 분석 결과를 리스트에 저장 <음정, 시간> 쌍

실제 구현에서는 아래 과정이 순서대로 처리되지 않지만, 구현 설명을 위해 따로따로 설명하였다.

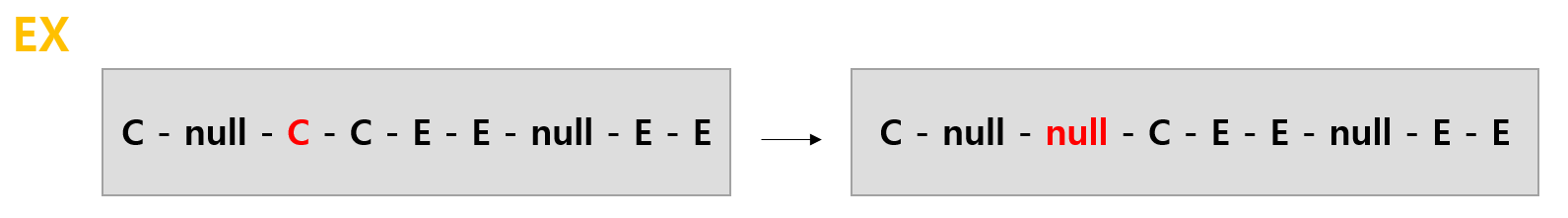
**\*위 과정을 설명함에 있어서 \*표시가 되어있는 특정값들은 모두 함수 내에서 답안지를 만드는 과정에 있어서 조정이 가능한 값이다.**

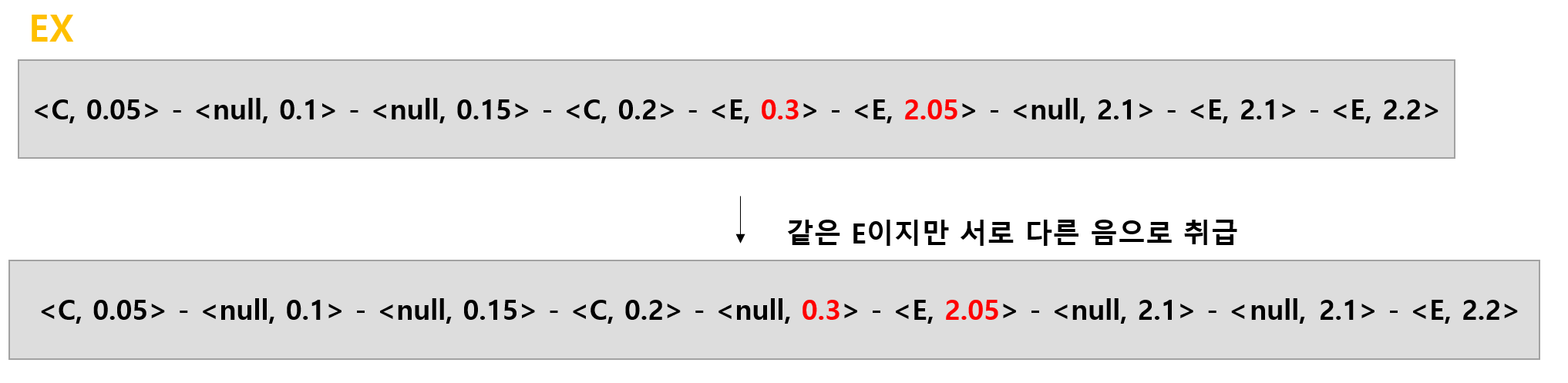
**실제 구현에서는 곡의 장르에 따라 인자를 따로 두었으며, 해당값들은 모두 경험적인 실험에 의해 얻어진 값이다.**

**1 TrimAnswerSheet(param1) 함수**

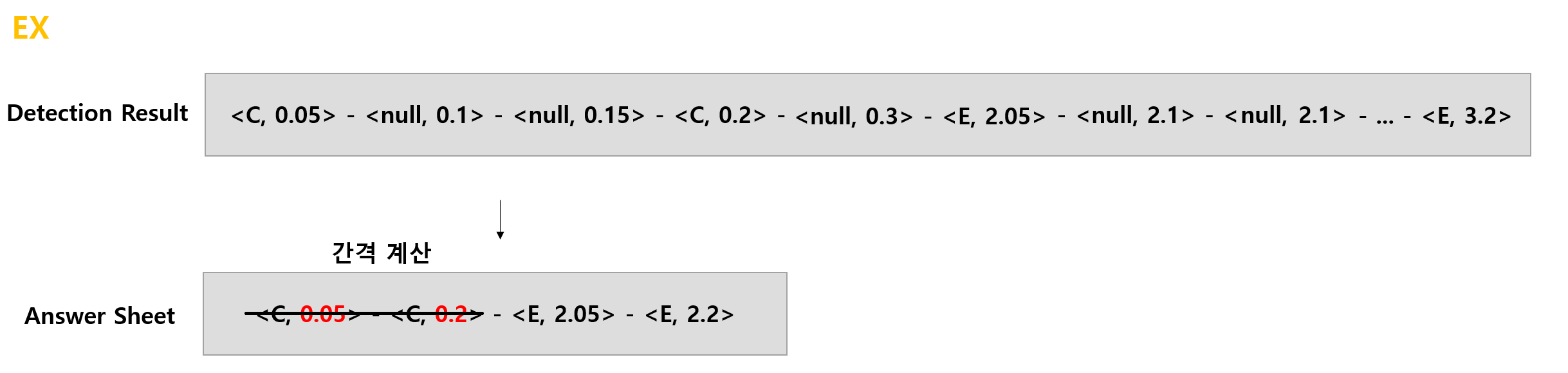
1-1 리스트의 모든 음정에 대해 연속적이지 않은(같은 음과 쌍을 이루지 않는) 음을 Null로 대체 한다. (리스트원소를 도중에 지울 경우, 총 길이의 변화로 반복문에서 index 참조가 구현 상에서 어려워지므로 null 값을 사용) 음정에 있어서 더 많이 반복되는 음을 구하고, 노이즈가 발생한 부분을 배제하기 위함이다.  


이 때 음정의 분석 단위 시간이 약 0.05초 이기 때문에 서로 다른 음만 계속 반복되어 모든 음이 null로 대체 되는 경우는 드물다.

1-2 이 과정에 있어서 한 음에 대해 한쌍의 원소만 리스트에 남기기 위해 같은 음 사이에 중복되는 음 또한 null 값으로 대체한다.  


1-3 위 과정에 있어서 서로 같은 음이더라도 **\*특정 시간(param1)**을 넘어가는 간격을 이룰 경우 서로 다른 음으로 취급하여, null로 대체 될 수 있다.

**2 removeNull\_sNotes(param1) 함수**

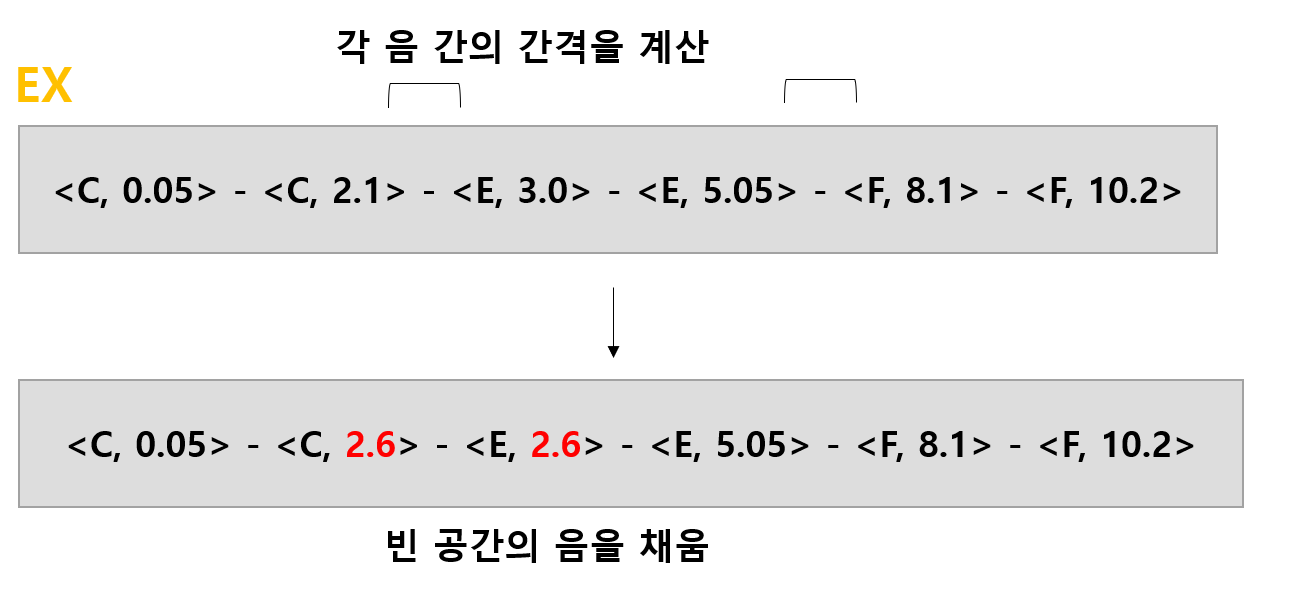
2-1 실제 답안지 값이 들어갈 답안지 리스트를 만들어서 null 값을 제외한 나머지 원소들을 추가한다. 이 때, **\*특정 길이(param1)**보다 짧은 음의 경우, 추가하지 않는다.

**3 extendTimeStamp(param1, param2) 함수**

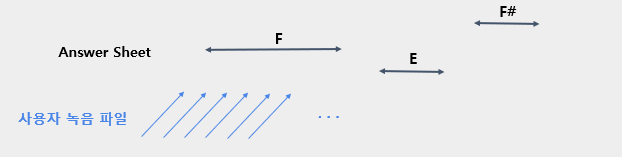
3-1 첫번째와 두번째 단계에서 음정 측정에 대해서 자주 나타나지 않거나, 노이즈 간섭 또는 흔들리는 음, 부정확한 음을 지우는 데에 중점을 두었다.

하지만 이에 대한 결과로 채점을 할 음이 얼마 남지 않게 되게 될 경우 채점의 정확도에도 영향을 주게 된다.(기본적으로 채점은 답안지의 음의 간격에 따라 답안에 필요한 입력 수를 계산하기 때문)

이를 방지하기 위하여, **\*특정 길이(param1)**의 가까이 있는 음에 대해서 서로의 음에 대하여, 답안시간대를 확장하는 과정을 거치게 한다. 이때 늘어나는 \***특정 최대값(param2)**을 지정할 수 있다.

이는 특정음 간격이 가까울 경우에, 보통은 해당 음 사이에 다른 음이 존재했을 가능성이 높고, 

* **기본 아이디어 및 과정(채점)**



각 음의 길이를 **\*특정 길이 단위**로 나누어 결과 수를 정하여, 필요한 입력 수를 <정답으로 인정되는 입력수, 시간> 채점 리스트로 만든다. 사용자 녹음 파일도 보컬 분리 파일과 같이 음정 분석을 하여 <음정, 시간>의 입력 리스트로 만들어 분석을 하게 되고, 이를 답안지와 비교하여 이를 채점리스트에 입력 수를 저장한다.

정답에 필요한 총 입력수와 들어온 입력 수를 퍼센트로 점수로 사용하였다.

\*채점에서 영향을 주는 인자로는 정답으로 인정하는 음정차이, 음정 길이를 나누는 단위의 길이가 있다. 이 역시 장르와 난이도에 따라 조정하였다.

**3.2 보컬 분리 기능**

보컬 분리를 위하여 facebookresearch팀에서 개발한 demucs라는 오픈소스 딥러닝 모델을 사용하였다. demucs의 특징은 다음과 같다.

- U-Net convolutional architecture

- 인코더와 디코더 지원(다양한 포맷의 파일에 대해 사용 가능)

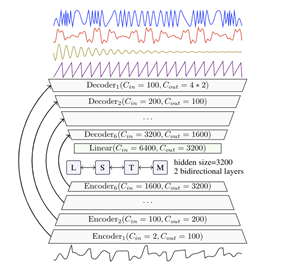
- multiple audio channel 지원

- 기존에 개발돼 있던 모델인 [Wave-U-Net](https://github.com/f90/Wave-U-Net) 과 [SING](https://github.com/facebookresearch/SING)을 기반으로 함

- MusDB dataset(84 개의 노래)+150개의 추가 곡으로 트레인

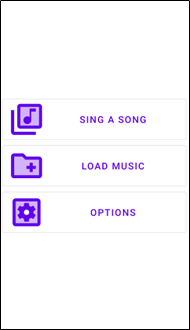
- 타 모델에 비해 더 좋은 성능 자랑

(4.2 demucs 성능분석 표에서 이 프로젝트에 사용된 모델의 성능은 빨간 박스를 친 부분이다.)  
- MIT 라이선스(상업적 이용, 수정 가능)



위 사진은demucs의 구조이다. mixture wave 형식으로 인풋을 받았을 때 4개의 악기(보컬, 드럼, 베이스, 나머지)로의 분리를 아웃풋으로 생성한다.

**3.3 UI**

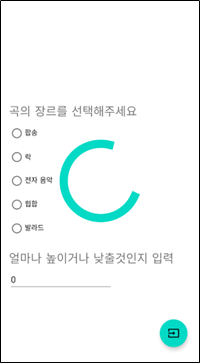
**3.3.1 메인 화면** 

어플리케이션을 실행했을 때, 가장 먼저 보이는 메인 화면이다. 녹음 버튼, 음악 업로드 버튼, 그리고 옵션 버튼으로 이루어져 있다.

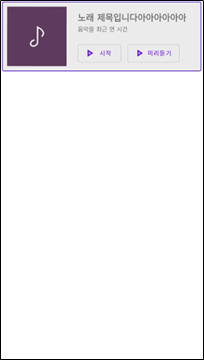
**3.3.2 업로드 화면**



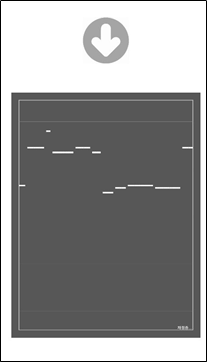
업로드 버튼을 눌렀을 때 기기 내, 또는 외부에서 음악 파일을 업로드할 때 보이는 화면이다. 좌측 사진은 휴대폰 내 다운로드 폴더에서 음악을 선택 중인 화면이다. 디자인은 안드로이드 기본 디자인을 사용한다.

**3.3.3 대기 화면**

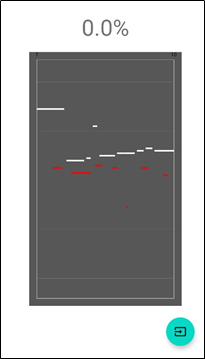
음악 파일의 업로드와 변환이 진행되면서 사용자가 기다리는 동안 보여주는 화면이다. 가운데에 Progress Bar가 있어 파일의 변환이 얼마나 이루어졌는지 확인할 수 있고, 곡의 장르와 키(key)를 얼마나 조절할 지 선택할 수 있는 옵션이 존재한다. 여기서 선택한 옵션에 따라 곡의 키를 바꾸거나, 채점 방법을 바꾸게 된다.

**3.3.4 음악 리스트 화면**

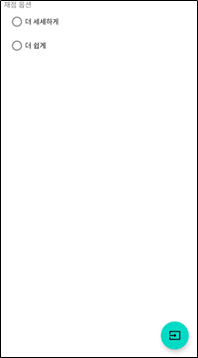
음악 파일 업로드가 완료되면, 좌측과 같이 음악 리스트에서 확인할 수 있다. 각 객체에는 음악 정보와, 미리듣기 버튼, 녹음을 시작할 수 있는 시작 버튼이 포함되어 있다.

**3.3.5 플레이 화면**

음악 녹음 시 보여지는 화면이다. 화면 상단의 아이콘은 실시간 사용자의 녹음된 음정과 정답 음정을 비교해서 일치한다면 좌측 화면처럼 체크 표시를, 사용자가 더 높은 음을 부르고 있다면 위쪽 화살표를, 낮다면 아래쪽 화살표를 표시한다. 아이콘 밑에는 원곡의 정답 음정을 차트로 표시해 시간에 따라 움직일 수 있도록 구현하였다.

**3.3.6 최종 점수 확인 화면**

녹음이 끝나고 사용자가 녹음한 노래와 정답 음정을 비교해서 최종적인 점수를 화면에 표시한다. 점수는 100점 만점이며, 점수 하단에는 플레이 화면과 같이 음정 차트를 보여준다. 플레이 화면의 차트와 다른 점은, 정답 음정만이 아니라 사용자가 녹음한 음정을 차트로 그려 전반적으로 어떻게 노래를 불렀고, 어떤 부분에서 틀렸는지 표시하였다. 정답 음정은 흰색 차트로, 사용자 녹음 음정은 붉은색 차트로 표현하였다. 사용자 음정과 정답 음정이 일치할 경우 사용자 음정은 표시되지 않기 때문에 일치 하지 않는 부분만 확인 가능하다.

**3.3.7 옵션 화면**

채점 옵션을 설정할 수 있는 옵션 화면이다. 엄격한 채점 기준을 원한다면 “더 세세하게” 옵션을, 그렇지 않다면 “더 쉽게” 옵션을 선택하여 채점 알고리즘의 인자를 변경할 수 있다.

**3.4 pitch shifting**

librosa의 pitch\_shifting 함수를 사용하였다. 이 함수의 내부적 작동 방식은 음악파일의 속도를 변경하여 음의 높낮이를 조절한 다음 원래 음악의 길이에 맞추어 리샘플링 하는 것이다.

이 때, 다양한 형식의 음악파일을 librosa에 로드하기 위해서 ffmpeg 코덱을 python module로 서버에 설치하여 사용하였다.

**3.5 colab 서버 관리**

다음은 이 프로젝트에서 코랩 서버를 사용하는 방법이다.

1) 구글 드라이브에 서버 측 소스코드 업로드

2) colab과 연결된 ipynb에서 구글 드라이브 마운팅(파일시스템 마운팅) 및 linux 커맨드를 이용한 탐색과 실행

3) firebase storage bucket에 업로드 할 권한을 주기 위해 해당 bucket에서 발급받은 인증 키 json파일을 업로드 한 후 환경변수에 이 키 파일의 경로를 지정한다.

4) 모델로드를 미리 한 후 서버처리 요청(보컬분리+pitch shifting)이 들어올 때까지 대기한다.

**3.6 안드로이드 내에 여러 세부 구현 사항**

* **답안지 제작 채점에 대한 난이도, 장르에 따른 Option Flag**안드로이드 SharedPreference 클래스를 사용하여, 구현하였다.

기본적으로 안드로이드 앱 내부의 캐시저장소에 간단한 키값을 저장하는데 사용하는 클래스로 이번 프로젝트 내에서 Static 클래스로 구현하여, 앱 내부의 어떤 코드에서도 쉽게 값을 변경, 불러오기가 가능하여, 채점, 답안지 만들때나, 사용자의 입력으로 쉽게 값을 바꿀 수 있도록 하였다.

* **만들어진 답안지의 로컬 저장 방법**

만들어진 답안지는 파일 오브젝트를 직렬화하여, 각각 파일로 저장하였다.

리스트 화면에서는 기본적으로 안드로이드 앱 내부에서만 접근 가능한 고유 파일 시스템의 디렉토리를 불러와 화면에 표시하는 식으로 구현하였다.

하지만 이런 식으로 구현 하였을 경우, 저장된 답안지 파일을 제외하고도, 앱 내부에서 사용하는 파일들 모두가 불러와지기 때문에 만들어진 답안지 파일들의 이름을 관리하는 리스트 파일을 따로 구현하여, 이를 통해 리스트 화면에서 답안지들의 동기화를 완성하였다.

**4. 성능 평가 및 한계**

**4.1 채점 성능**

* **성능 평가 기준 : 답안지를 만든 보컬 파일을 입력으로 사용**

직접 불러서 나온 점수로 하기엔 객관적이지 않고, 여러 노래에 대해서 시험을 하는데 있어서 부르는 사람의 가창력이 따라주지 않는 문제점도 존재하였다.

그 외에 노래에 대해서 악보를 보고 음을 추출하여, 파일로 만들어서 입력시키기에는 여러 노래에 대해서 악보를 구하거나 채보하는데 있어서 비용의 효율이 따라주지 않았다.

이에 따라 답안지를 만드는데 사용한 보컬 파일(서버를 통해 원곡에서 분리된)을 그대로 입력으로 사용하는 방법으로 결정하였다. 그대로 입력으로 음정을 분석 만들어진 답안지와 비교 채점하는 과정을 거쳤다.

이를 채점 점수를 지표로 사용하여 평가와 답안지 제작 및 채점 과정에서 조절가능한 여러 인자에 대해서 수정을 하였다.

* **여러 곡에 대한 평가 및 조정**

기본적으로 5가지의 장르(발라드, 팝, 힙합, 전자 음악, 락) 곡을 분류하여, 선별하였다.

하지만 현대음악에 있어서 여러 장르가 합쳐진 곡도 존재하기에 이런 곡들은 따로 부분적으로 분석을 하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| **장르** | **노래 이름**  **/ 아티스트** | **정답 음정수 / 총 정답 답안 음정수 (점수)** | **비고** |
| 발라드 | Someone Like you / Adele | 2536/3164 (80%) |  |
| 야생화  / 박효신 | 3104/3510 (88%) |  |
| 락 | Sk8er Boi /  Avril Lavigne | 1325/2002 (66%) |  |
| 어쩌다 마주친 그대 / 송골매 | 1220/1784 (68%) |  |
| 비와 당신/  노브레인 | 924/2204 (40%) | 분리 안된 기타소리가 음정 분석에 잡힘 |
| 팝 | Y / 프리스타일  feat. 케이 | 2082/2672 (77%) | 2키 낮춘 곡  랩과 여성 보컬 섞여있는 곡 |
| Signal / 트와이스 | 886/2306 (37%) | 잦은 전자음 미분리와 멤버별 합창시 음정 분석 성능 하락 |
| DNA / 방탄소년단 | 1178/2713 (43%) |
| 힙합 | 친구여 / 조PD feat.인순이 | 453/2128 (21%) | 랩 자체의 분석 성능 하락/피처링 파트에서 코러스 부분의 경우 정확도 급락 |
| Not Afraid / Eminem | 1063/2585 (41%) | 랩 자체에 음정이 없음  주변의 비트 음악의 영향에 영향을 받음 |
| 전자 음악 (EDM) | Clarity / Zedd feat.Foxes | 2241/3201 (70%) |  |
| Titanium / David Guetta ft.Sia | 2057/3165 (65%) |  |

* **발라드**

기본적으로 피아노를 베이스로 하는 경우가 많고, 크게 보컬을 덮는 튀는 음이 없어서 성능에 있어서 문제가 적다.

* **락**

기본적으로 드럼소리가 보컬에 섞여 나오는 경우 일부, 답안지에 영향을 받았고, 비와 당신의 경우, 곡 자체의 음질 자체도 문제가 있었으나 특정 파트의 분리되지 않은 기타 솔로가 그대로 답안에 반영되는 바람에 정확도가 해당 부분에 있어서 급락하게 되었다.

이에 대해선 아래와 같은 파트의 주파수 대역이 가장 확률이 높았다.  
****

답안지를 만들 때, 락 장르의 경우 음정 분석의 함수 내에서 반환하는 확률값이 높을때, 즉, 사람의 목소리일 확률이 더 높을 때만 음정을 분석하여 결과값으로 사용, 답안지를 만드는식으로 보완하였다.

* **팝**

팝송의 경우, 워낙 종류가 다양하여, 정의하기 힘들지만, 분리가 힘든 다양한 악기를 사용하는 경우가 아닐 때는 높은 수준의 정확도를 보였다.

하지만 현대 음악으로 넘어와서, 다양한 음향이 사용되는 경우 분리 성능이 하락하였으며, 보컬과 겹치게 될 경우, 음정 분석이 튀는 경향이 두드러졌다.

이를 통해, 노이즈가 섞이는걸 막기 위해, 답안지를 만드는 과정에서 짧은 음을 지우는 인자를 늘리고, 지워진부분을 채우는 범위를 확장하는 식으로 인자들을 조정하였다.

트와이스와 방탄소년단과 같은 그룹의 노래에서 여러 멤버가 노래를 부르는 경우

답안지가 만들어지는 경우는 드물었으며, 두 명중 한명의 음으로 번갈아 분석되거나, 간섭을 일으켜, 아예 상관없는 음이 되는 경우도 있었다.

이는 프로젝트의 제한 사항으로 보인다.

* **힙합**

랩에 대해서 음정 변화를 세세하게 인식하기에 있어서 정확도가 떨어졌다. 발라드에서 바이브레이션과 같은 미세한 음정 변화도 잡아내는 것에는 문제가 없었으나 랩과 같이 주파수의 변화가 크지 않지만, 그 주파수에 있어서 음정이 정의되지 않은 음(사람이 말하는 것과 같음)의 경우, 경계가 애매하게 표시되어 이 경우가 답안지에 반영되지 않는 경우가 있었다.

이는 채점에 있어서 허용되는 음정 범위를 늘림으로써 일부 해결할 수 있었지만, 근본적인 해결책은 아니다.

그와 별개로 친구여와 같은 노래는 다른가수가 랩중에 피처링을 하여, 두명의 음이 겹치는 경우가 많았고, 이는 정확도 문제를 해결할 수 없는 정확도 하락으로 이어졌다.

* **전자음악**

최초 가장 성능에 있어서 문제가 있을 것으로 예상되었지만, 에초에 전자음과 보컬의 음향 크기 차이가 있기 때문에 보컬 분리와 성능에 영향이 적었다.

하지만 그 외에 기본적으로 사람의 목소리에 이펙트가 들어가는 경우가 많았기 때문에 이를 인식하는데 있어서 문제가 생길 것으로 고려해 음정 인식 함수에서 사용하는 확률(사람 목소리일 확률)을 낮추어 채점에 사용하여, 이펙터가 사용되어도 채점의 문제를 줄였다.

* **한계점**

**1 보컬 분리 성능 자체에서 곡에 따라 영향이 큼**

일단, 보컬 분리 자체가 잘 되지 않으면, 답안지가 제대로 만들어지지 않게 된다.

특히 가장 영향을 받게 되는 부분은 사람의 주파수와 비슷한 기타 대역은 잡힐 높은 편이며, 가장 분리가 잘 되지 않는 것은 악기가 아닌 인공적으로 만들어진 여러 전자음들의 분리 성능이 떨어지는 편이였다. 이와 같은 음들이 보컬과 겹치게 될 경우, 문제가 더 심각해졌다.

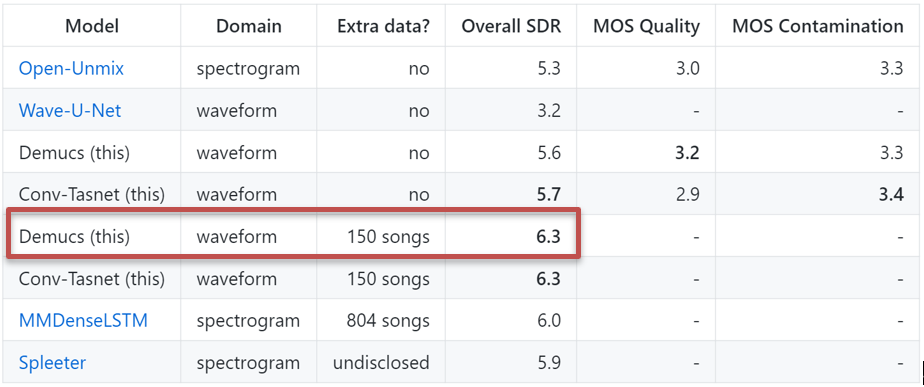
그 외에 음질이 좋은 곡들도 분리 후에 노이즈가 생겨, 이를 음정 분석하는데 있어서, 애로사항이 있다.

**2 음정 분석하는 과정에 있어서, 합창(여러 음성이 겹치는)의 경우 여러 음이 중구난방 잡힘**

음정을 분석하는 알고리즘의 한계로, 여러가지음을 한번에 잡아낼 수 없기 때문에 이는 채점에서 정확도에 큰 영향을 주게 된다.

**4.2 그 외**

**A. demucs 보컬분리 성능**

****

Overall SDR : 4개 소스에 대한 평균적 정확도

MOS Quality : 사람이 들었을 때 1에서 5까지의 정확도 점수

MOS Contamination : 다른 소스의 소리가 포함되지 않았는지 대한 1에서 5까지의 점수

최근에 개발된 딥러닝 모델들 중에서 최고 성능을 보인다.

**B. 처리 시간**

- firebase storage 업로드 시간 2초/1 곡

- 보컬분리 시간 3초(GPU : T4) 혹은12초 (GPU : K80)

(코랩에서 할당해주는 GPU를 선택할 수 없기 때문에 랜덤성 있음)

- pitch shifting 시간 10초/1곡

(한곡은 4분짜리라고 가정)

- - pitch detection : 전체에 대해 1초 미만, 마이크 인풋에 대해 실시간으로 가능

**5. 참고자료**

[**https://play.google.com/store/apps/details?id=com.smule.singandroid (Smule)**](https://play.google.com/store/apps/details?id=com.smule.singandroid%20(Smule))

[**https://play.google.com/store/apps/details?id=com.nexstreaming.app.singplay**](https://play.google.com/store/apps/details?id=com.nexstreaming.app.singplay) **(씽플레이)**

[**http://www.tjmedia.com/**](http://www.tjmedia.com/) **(TJ 미디어)**

[**https://github.com/JorenSix/TarsosDSP**](https://github.com/JorenSix/TarsosDSP)**(tarsosDSP)**

[**https://www.ffmpeg.org/**](https://www.ffmpeg.org/)**(FFmpeg)**

[**https://github.com/tanersener/mobile-ffmpeg**](https://github.com/tanersener/mobile-ffmpeg)**(mobile-FFmpeg)**

[**http://mroy.chez-alice.fr/yin/index.html**](http://mroy.chez-alice.fr/yin/index.html)**(YIN 알고리즘 관련)**

[**https://github.com/facebookresearch/demucs**](https://github.com/facebookresearch/demucs)**(Demucs)**

[**https://librosa.org/librosa/\_modules/librosa/effects.html#pitch\_shift**](https://librosa.org/librosa/_modules/librosa/effects.html#pitch_shift)**(librosa)**

[**https://github.com/PhilJay/MPAndroidChart**](https://github.com/PhilJay/MPAndroidChart)**(MPAndroidChart)**

**6. 영상**

**발표 링크 :** [**https://youtu.be/3LCKPKOcUVg**](https://youtu.be/3LCKPKOcUVg)

**시연 영상:** [**https://drive.google.com/file/d/15cnRqq0WtWIpFnsBaFH5VqfkWbRW7kg2/view**](https://drive.google.com/file/d/15cnRqq0WtWIpFnsBaFH5VqfkWbRW7kg2/view)

**시연 영상(풀버전) :** [**https://youtu.be/b67ZqnuGpdo**](https://youtu.be/b67ZqnuGpdo)

1. 사진 출처: 구글 플레이스토어 – 씽플레이 어플리케이션 [↑](#footnote-ref-1)
2. 사진 출처: 구글 플레이스토어 – Smule 어플리케이션 [↑](#footnote-ref-2)
3. 사진 출처: Tj Media [↑](#footnote-ref-3)
4. 화면 순서도 [↑](#footnote-ref-4)
5. 화면 별 기능 [↑](#footnote-ref-5)