

종합설계 프로젝트

수행 보고서

프로젝트 명	
영상 처리와 딥러닝을 이용한 악보 코드 변환 프로그램 Music Score Chord Conversion Program using OpenCV, Deep Learning	
팀 번호	
S2-7	
제출일	
2020.06.27	
팀원	팀장 임영규
	팀원 김민지
	팀원 문지수
지도 교수	공기석 교수님 (인)
	박정민 교수님 (인)

- 문서 수정 내역 -

작성일	대표 작성자	버전(Revision)	수정 내용	
2020.01.04	문지수	1.0	수행 계획서	최초 작성, 양식 설정
2020.01.07	공동 작성	1.1	수행 계획서	서론 내용 최초 작성
2020.01.09	공동 작성	1.2	수행 계획서	본론(1-3) 내용 최초 작성
2020.01.13	문지수	1.3	수행 계획서	작성 내용 양식 수정
2020.01.16	김민지	1.4	참고문헌	참고문헌 추가, 양식 수정
2020.01.18	공동 작성	1.5	개발 일정	개발 일정 추가
2020.01.20	문지수	1.6	개발 내용	개발 내용 추가
2020.01.20	임영규	1.7	수행 계획서	번호 서식 수정
2020.01.21	공동 작성	1.8	수행 계획서	전체 양식 수정
2020.01.22	공동 작성	1.9	목차, 참고문헌	페이지 추가, 양식 수정
2020.01.23	공동 작성	2.0	시험 시나리오	모듈 별 플로우 차트 추가
2020.02.25	공동 작성	2.1	상세 설계	상세 설계 추가
2020.02.26	문지수	2.2	상세 설계	상세 설계 표 수정
2020.02.29	문지수	2.3	수행 계획서	계획서 발표 후 개발 수정 사항 추가
2020.03.01	문지수	2.4	수행 계획서	계획서 발표 후 개발 수정 사항 추가 페이지 수정
2020.03.05	김민지	2.5	수행 계획서	개발 환경 수정
2020.04.17	문지수	3.0	프로토타입	목차 설정
2020.04.23	문지수	3.1	프로토타입	프로토타입 내용 추가
2020.04.24	공동 작성	3.2	프로토타입	프로토타입 내용 수정 이전 변경사항 수정
2020.04.28	공동 작성	3.3	프로토타입	프로토타입 내용 수정 이전 변경사항 수정
2020.04.30	공동 작성	3.4	프로토타입	2.1 개발 내용 수정 2.4 상세 설계 내용 추가 2.5.1 프로토타입 구현 현황 2.5.2 프로토타입 개발 2.5.3 프로토타입 테스트 및 분석
2020.05.01	문지수	3.5	프로토타입	목차 내용 추가 및 수정 1.6 개발 환경 변경 내용 수정 2.4 상세 설계 변경 내용 수정 2.5 프로토타입 변경 내용 수정
2020.06.20	문지수	4.0	최종 데모	최종 데모 목차 설정
2020.06.21	김민지	4.1	최종 데모	1.5 개발 일정 변경 내용 수정 1.6 개발 환경 변경 내용 수정 2.2 문제 및 해결 방안 변경 내용 수정

				2.3 시스템 및 시험 시나리오 변경 내용 수정 2.6.3 서버와 앱 연동 테스트 추가 2.6.4 서버와 데이터베이스 연동 테스트 내용 추가 2.7.1 Coding Rule(서버-앱) 내용 추가
2020.06.22	문지수	4.2	최종 데모	최종 데모 수정 내용 통합 2.5.2.2 서버 연동 구현 현황 내용 추가 2.5.3 프로토타입 개발 변경 내용 수정, 앱 화면 구성 변경 수정 2.7.4 Application 내용 추가
2020.06.23	임영규	4.3	최종 데모	2.1.1 악보 인식 모듈 내용 추가 2.4 상세 설계 내용 추가
2020.06.24	김민지	4.4	최종 데모	2.7.3 Server & Database 내용 추가
2020.06.25	문지수	4.5	최종 데모	최종 데모 수정 내용 통합 2.7.5 조옮김 설명 내용 추가 2.7.6 Demo 앱 실행 화면 추가
2020.06.26	문지수	4.6	최종 데모	2 장 목차 추가 및 설정 2.6 프로토타입 이 후 전체 구현 현황 표 추가 2.6.3 서버와 앱 연동 구현 현황 표 추가 2.6.4 서버와 데이터 베이스 연동 구현 현황 표 추가
2020.06.26	공동 작성	4.7	최종 데모	2.6.1 음표 인식 정확도 표 추가 2.6.2 음표 박자 인식 표와 그래프 분석 추가 2.7 OurChord 전체 소프트웨어 구성도 추가 2.7.1 Algorithm 순서도 추가 2.7.2 Demo 시나리오 추가

- 문서 구성 -

진행 단계	프로젝트 계획서 발표	중간 발표1 (2월)	중간 발표2 (4월) <small>(코로나19로 발표 연기)</small>	학기말 발표 (6월) <small>(코로나19로 발표 연기)</small>	최종 발표 (10월)
기본 양식	계획서 양식				
포함되는 내용	I.1 서론(1~6) I.2 본론(1~3) 참고자료	I.2 본론(1~4) 참고자료	I.2 본론(1~5) 참고자료	I.2 본론(1~7) 참고자료	X

이 문서는 한국산업기술대학교 컴퓨터공학부의
 “종합설계” 교과목에서 프로젝트
 “영상 처리와 딥러닝을 이용한 악보 코드 변환 프로그램
 (Music Score Chord Conversion Program using OpenCV, DeepLearning)” 을
 수행하는 (S2-7 : 임영규, 김민지, 문지수) 가 작성한 것으로
 사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

1. 서론

1.1	작품 선정 배경 및 필요성	5
1.2	기존 연구/기술 동향 분석	7
1.3	개발 목표	7
1.4	팀 역할 분담	8
1.5	개발 일정	8
1.6	개발 환경	9

2. 본론

2.1	개발 내용	10
2.2	문제 및 해결 방안	13
2.3	시스템 및 시험 시나리오	15
2.4	상세 설계	16
2.5	Prototype 구현	22
2.6	시험/테스트 결과	38
2.7	Coding & Demo	42

3. 결론

3.1	연구 결과
3.2	작품 제작 소요 재료 목록

참고자료	49
------------	----

1. 서론(Introduction)

1.1 작품 선정 배경 및 필요성

음악 전공자가 아닌 사람들은 조표(샹, 플랫)가 많은 악보를 연주함에 어려움을 느끼는 경우가 많다. 세션 연주자들은 각자의 악기에 맞추어 하나의 코드(chord)로 맞추어 연주를 해야 하는 경우가 많다. 또한 대부분의 악보는 저작권 문제로 MIDI 파일로 제공되지 않고, PDF 파일로 제공되는 경우가 많아 음표 수정 및 삭제가 불가능한 경우가 많다.

조옮김을 통해 음악을 시작하려는 사람들에게 연주하기 쉬운 악보 제공을 통해 악기 연주에 자신감 부여할 수 있다. 세션 연주자들에게는 서로의 악기 코드(chord)를 빠르고 쉽게 맞출 수 있다. 또한 수정이 용이한 MIDI 파일 제공을 통해 악보 수정이 용이하고, 다른 코드의 악보를 별도로 구매해야 하는 불편함을 없애고자 한다.

1.2 기존 연구/기술 동향 분석

1.2.1 기존 연구 분석

기존 악보 코드 변환 프로그램은 Musecore가 있다. Musecore는 GNU(General Public License) 일반 공중 사용 허가서를 따르는 자유 및 오픈 소스 프로그램이다. 온라인 악보 공유 플랫폼을 제공한다. 악보 PDF 파일을 MIDI 파일로 변환해 주는 기능이 포함되어 있지만 Musecore 프로그램 내에서 MIDI 파일 변환이 불가능하여, Musecore 웹 페이지에 접속하여 MIDI 파일로 변환해주어야 한다. 변환이 된 MIDI 파일도 조표가 있는 경우 제대로 음표가 인식되지 않은 경우가 매우 많았다.

forScore는 기존 악보 위에 레이어를 추가하

여 추가한 레이어 위에 표기 기호 스탬프를 통해 악보 수정이 가능하다. 하지만 주요 표기 기호는 별도의 추가 유료 구매가 필요하다. 위의 악보 수정과 마찬가지로 빈 악보 레이어 위에 기호 스탬프를 추가하여 작곡이 가능하였다. 추가로 MIDI 악기 연결 시 악기 연주의 녹음이 가능하고, 메트로놈, 피치, 조율 기능을 제공한다.

Piano Companion는 선택한 코드(chord)에 대한 안내를 제공하고, 다양한 화음의 진행을 확인 가능하였고, 오선지에 코드 표시가 가능하였다. 또한 피아노 왼손 연습곡 하농(Hanon) 아래 그림 1 과 같은 스케일 악보를 제공하였다. 하지만 사용자가 이해하기 어려운 UI(User Interface)를 가지고 있다.



그림 1. 스케일 악보 예시

Fig 1. Scale score example

악보 바다는 피아노 악보, 기타, 베이스, 드럼, 건반 악보, 타브(TAB) 악보, 멜로디, 코드 악보, MR 및 동영상 악보를 다양한 코드로 유료 제공하는 웹사이트이다. 유료로 조옮김 신청이 가능하지만 조옮김 신청 시 조옮김 악보를 제공받는데 시간이 오래 걸린다는 불편함이 있다.

위 기존 연구의 낮은 음표 인식률을 개선하여 높은 음표 인식률을 통해 더 정확한 음표 더 정확한 조옮김 악보를 제공하고, MIDI 파일을 제공하여 악보 수정을 손쉽게 할 수 있다. 또한 편리한 UI를 통해 모든 사용자가 쉽게 사용할 수 있도록 한다.

1.2.2 기술 동향 분석

악보 코드 변환 프로그램(이하 “OurChord”라 한다.) 개발을 위해 먼저 악보 파일 이미지에 대한 영상 처리가 필요하다. 영상 처리는 입출력이 영상인 모든 형태의 정보 처리를 가리키며, 사진이나 동영상을 처리하는 것이 대표적인 예이다. 악보 영상 처리를 위해 OpenCV 라이브러리를 사용한다.

OpenCV는 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리로 실시간 이미지 프로세싱(Processing)에 중점을 둔 라이브러리이다. 47만 명의 OpenCV 라이브러리 사용자 커뮤니티가 존재하고, 1800만 건 이상의 다운로드 수, 구글, 야후, 마이크로 소프트, 인텔과 같은 회사들도 OpenCV 라이브러리를 사용한다.

OpenCV는 C++, Python, Java 인터페이스와 Windows, Linux, Android 및 Mac OS를 지원한다. 머신 러닝(Machine Learning)에서 사용되는 OpenCV는 Python 라이브러리 Top 10에 포함되어 있다. OpenCV 라이브러리에서 물체 추적 분야에서 자주 사용되는 패턴 매칭(Pattern Matching) 함수를 제공한다. TensorFlow, Torch/PyTorch 및 Caffe의 딥러닝 프레임워크를 지원한다. 병렬 프로그래밍, CUDA(Computer Unified Device Architecture), OpenCL, 그래픽 성능 처리 장치와 GPU(Graphics Processing Unit)를 통해 영상 처리 속도를 향상시킨다.

악보 영상 처리 후 음표 구분을 위해 기계 학습(Machine Learning)이 필요하다. 기계 학습은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있는 알고리즘과 기술을 개발하는 것이다. 전 세계적의 머신 러닝 회사에서도 텐서플로우(TensorFlow)를 사용한다.

텐서플로우는 기계 학습을 위한 엔드 투 엔드 오픈소스 플랫폼으로 데이터 흐름 프로그래밍을 위한 오픈소스 소프트웨어 라이브러리이다. 텐서플로우는 깃허브(GitHub)에서 가장 인기있는 프로젝트 중 하나이다. 아래의 그림 2를 통해 상호 연결된 텐서플로우 커뮤니티를 확인할 수 있다.



그림 2. 상호 연결된 텐서플로우 커뮤니티

Fig 2. Interconnected TensorFlow Community

텐서플로우는 아래의 그림 3에서 확인할 수 있듯이 2019년 깃허브 작업 프로젝트 기여자 수 5위를 기록하였다.

* 오픈 소스 프로젝트 기여자 수

01	microsoft/vscode	19.1k
02	MicrosoftDocs/azure-docs	14k
03	flutter/flutter	13k
04	firstcontributions/first-contributions	11.6k
05	tensorflow/tensorflow	9.9k
06	facebook/react-native	9.1k
07	kubernetes/kubernetes	6.9k
08	DefinitelyTyped/DefinitelyTyped	6.9k
09	ansible/ansible	6.8k
10	home-assistant/home-assistant	6.3k

그림 3. 2019년 깃허브 오픈 소스 프로젝트 기여자 수

Fig 3. GitHub Open Source Project Contributors in 2019

텐서플로우는 케라스(Keras) 같은 직관적인 API(Application Programming Interface)를 통해

즉각적인 모델 반복 및 손쉬운 디버깅이 가능하다.

케라스(Keras)는 TensorFlow 딥러닝 모델 설계와 훈련을 위한 고수준 API로 사용자 친화적, 모듈화 및 구성 가능성, 쉬운 확장성을 가진다.

국제 머신 러닝 학회(ICML: International Conference on Machine Learning)의 논문 제출 수가 매년 증가하는 것을 나타낸 그림 4를 통해 기계 학습에 대한 관심이 해를 거듭할수록 늘어남을 알 수 있다.

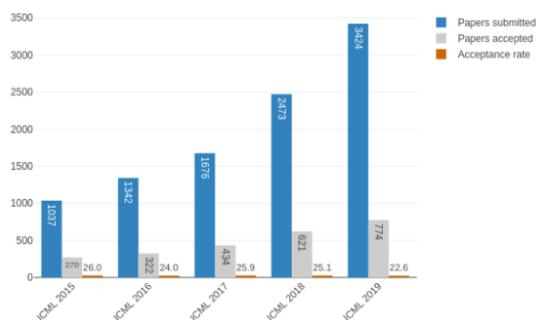


그림 4. 2015 ~ 2019 년도 국제 머신 러닝 학회의 논문 제출 수

Fig 4. Number of papers submitted by the ICML from 2015 to 2019

1.3 개발 목표

OurChord 개발을 위해 첫번째, 악보 PDF 파일을 PNG 파일로 변환한다.

두번째, PNG 파일로 변환한 악보를 영상처리한다. 먼저 히스토그램(Histogram)을 사용하여 악보의 오선 좌표를 추출한다. 이 후 템플릿 매칭(Template Matching)을 사용하여 음표의 좌표를 추출하여 앞서 추출한 오선의 좌표의 정보를 통해 음표의 음계를 확인한다.

세번째, 템플릿 매칭을 통해 얻은 음표 이미지를 학습시킨 CNN(Convolutional Neural Network) 모델을 사용하여 음표의 박자를 확인한다. CNN 모델 생성 위해 음표 데이터를 각각 훈련용 240개, 검증용 120개씩 총 1,440개 직접 수집하였다. 음표의 박자 인식을 90% 달성할 것이다.

네번째, 인식한 음표데이터를 MIDI 파일로 변환하여 사용자가 쉽게 수정할 수 있도록 한다. 또한 MIDI 파일로 변환된 데이터를 통해 사용자가 원하는 코드(chord)로 조옮김 변환율 100%를 달성할 것이다.

1.4 팀 역할 분담

OurChord 개발을 위한 팀 역할 분담은 아래의 표 1 과 같다.

표 1. 팀 역할 분담

Table 1. Team Role Sharing

	임영규	김민지	문지수
자료 수집	영상 처리 알고리즘 조사 OpenCV 영상 처리 조사 CNN 딥러닝에 대한 조사 OCR, OMR에 대한 조사 MIDI 파일 구성 요소에 대한 조사		
설계	Application 환경 설계	Server 환경 설계	Database 환경 설계
구현	사용자 인터페이스 구현, 악보 파일 입출력	Server 환경 구현, Application, Server 연동	음표 데이터 DB 구현, 사용자 정보 DB 구현
공동 설계	악보 음표 추출 알고리즘 설계 코드 변환 알고리즘 설계 MIDI 파일 변환 알고리즘 설계		
공동 구현	악보 영상 처리 통한 음표 추출 추출한 음표 통한 MIDI 파일 생성 음표 추출 통한 조옮김 구현		
테스트	통합 테스트 및 유지 보수		

1.5 개발 일정

OurChord 개발 진행 상황은 아래의 표 2 와 같다.

표 2. 개발 진행 상황.

Table 2. Development Progress

항목	세부사항	진행률
자료 수집	OpenCV 영상처리 조사	100%
	R-CNN Deep Learning 조사	100%
	OCR, OMR 조사	100%
	MIDI 파일 구성 요소 조사	100%
	악보, 음표 DB 수집 및 Labeling	100%
요구사항 정의 및 분석	요구사항 분석	100%
	분석된 자료 바탕으로 요구 사항 정의	100%
	조옮김 기능	100%
	악보 오선 인식	100%
	악보 음표 인식	100%
	MIDI 파일 변환	100%
시스템 설계	악보 오선 인식	100%
	악보 음표 인식	100%
	MIDI 파일 변환	100%
구현	악보 오선 인식	100%
	악보 음표 인식	100%
	MIDI 파일 변환	100%
통합 및 테스트	시스템 모듈 통합	100%
	앱 구성도 제작	100%
	앱 구현	90%
	서버, DB 연결	100%
	앱, 서버 연결	100%
유지보수	유지보수	50%
최종 검토 및 발표	최종 확인	2020년 7월 예정

악보 오선 인식, 악보 음계, 박자 인식, MIDI 파일 변환 각각의 시스템 모듈 구현을 완료하였다. AWS(Amazon Web Server) 서버 환경 구축 및 앱과 서버의 연동을 완료하여 앱 화면 별 서버의 기능 구분을 완료하였다.

1.6 개발 환경

1.6.1 개발 언어

OurChord 개발을 위해 영상처리와 딥러닝, MIDI 파일 변환 개발을 위해 Python 을 사용하고, 사용자 인터페이스 구현을 위해 Java 를 사용 한다. 서버 구현을 위해 AWS EC2(t2.micro) 와 Python 을 사용하고, 데이터 베이스 구축을 위해 MySQL 과 AWS 의 RDS 를 사용한다.

1.6.2 사용 프레임 워크

Visual Studio Code 1.31 에서 Python 을 통해 악보 영상 처리, 딥러닝, MIDI 파일 변환을 개발하고, Android Studio 3.5/Android 9.0 (API 28), NDK 19 에서 Java 를 통해 사용자 인터페이스를 구현한다.

MySQL Workbench 8.0 을 통해 OurChord 의 데이터베이스를 구축한다. 데이터베이스를 구축 한 후 AWS(Amazon Web Server)에 템플릿 매칭을 위한 음표 데이터와 박자를 구분하기 위한 H5 모델을 업로드하였다. 구축한 데이터 베이스를 AWS 의 RDS 를 같이 사용한다.

1.6.3 주요 라이브러리

악보 영상 처리를 위해 OpenCV 라이브러리를 사용한다. OpenCV 라이브러리는 이진화(Binarization), 노이즈 제거(Noise Reduction), 기계 학습(Machine Learning) 등의 알고리즘이 존재한다. 악보 영상 처리를 위한 알고리즘을 이용하여 음표 인식을 위한 악보 이미지를 전 처리한다.

음표 인식을 위한 딥러닝 분산 처리를 위해 텐서플로우(TensorFlow) 라이브러리를 사용한

다. 텐서플로우는 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 오픈 소스 소프트웨어 라이브러리로 심볼릭 수학(Symbolic Mathematics) 라이브러리이자 뉴럴 네트워크(Neural Network)와 같은 기계 학습 응용 프로그램에 사용된다. 또한 텐서 보드(TensorBoard)를 통한 가시화를 통해 음표 학습 시 손실 함수의 경과와 중간층의 모습, 추출한 특징의 가시화를 통해 디버깅과 구축한 모델 이해를 용이하게 한다.

음표 인식을 위한 딥러닝 모듈 구현 시 케라스(Keras)를 텐서플로우와 함께 사용한다. 케라스는 수치 계산을 위한 라이브러리로 자동 미분 기능 등 딥러닝(Deep Learning)에 편리한 기술이 구현되어 있다. 또한 빠른 학습과 평가가 가능하도록 설계되어 최소한의 모듈 방식의 확장 가능성에 초점을 두고 있는 라이브러리이다. Python의 기계 학습 라이브러리인 사이킷런(scikit-learn)과 비슷한 인터페이스를 통해 학습과 평가를 진행할 수 있어 코드의 가독성이 매우 높아질 수 있다. 케라스의 자주 사용하는 것을 적절히 모듈화 한 구성을 통해 효율적인 딥러닝 모듈 코드 구현이 가능할 것으로 보인다.

추가로 Python 코드 작성 시 위의 악보 영상 처리와 딥러닝을 위해 행렬, 다차원 배열 처리를 지원하는 Numpy, 파일과 폴더를 생성 및 복사하는 Os, 매트랩(MATLAB)과 유사한 그래프 표시를 가능하게 하는 Matplotlib, MIDI 파일 변환을 위한 Pyknon 라이브러리를 사용한다.

2. 본론(Point)

2.1 개발 내용

OurChord 개발을 위한 시스템 구조는 아래의 그림 5 와 같다.

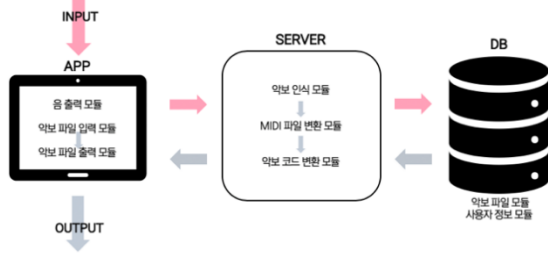


그림 5. 시스템 구성도

Fig 5. System diagram

2.1.1 악보 인식 모듈

악보를 인식하기 위해 악보의 음표 정보를 추출한다. OurChord는 사용자가 PDF 악보 파일을 업로드 하면 그 파일을 OpenCV 라이브러리를 활용하기 위하여 PNG 파일로 변환 시킨다. 이 후 음표의 음계와 박자 데이터들을 PNG 파일로 변환한 이미지 파일에서 얻는다.

악보의 음표를 추출하기 위해 먼저 오선을 추출한다. 악보 이미지에서 오선을 추출하기 위해 cv2.COLOR_BGR2GRAY 를 사용하여 흑백 이미지로 만든다. 흑백 이미지를 만든 후에 cv2.threshold를 사용하여 설정한 임계 값에 따라 높으면 255(흰색) 낮으면 0(검은색) 으로 바뀐 배열 리스트를 얻는다. 그 후 이미지의 높이 만큼의 값 0 을 가진 리스트를 만들고 이제 배열 리스트들을 확인하여 값이 0(검은색) 인 값이면 해당 y좌표를 + 1한다. 모두 확인을 하였으면 +1한 y좌표 값이 너비 값의 80% 이상 이면 오선으로 간주한다. 하여 오선의 좌표를 뽑고 그 오선 좌표 중에 중복 오선 좌표가 뺄 수 있어 1~2픽셀 정도만 차이 나는 좌표

는 제거 한다.

오선의 좌표를 뽑으면 해당 오선들의 gap을 구하여 평균 gap을 구한다. 그리고 OurChord 에서 템플릿 매칭하는 이미지의 오선 gap(define 설정 값)과 비교를 하여 얼마나 차이가 나는지 비율을 알아내고 해당 이미지를 템플릿 매칭에 알맞게 크기를 조절한다.

Resize 된 이미지에서 오선 좌표를 추출하여 이를 활용하여 오선별로 이미지를 추출한다. 템플릿 매칭과 박자 인식을 위한 악보 이미지는 오선별로 추출한 악보 이미지를 활용하여 정확도와 악보 인식 속도를 높인다.

악보의 음계를 추출하기 위해 템플릿 매칭을 사용한다. 템플릿 매칭을 통한 음계 인식 모듈 순서도는 아래 그림 6 과 같다.

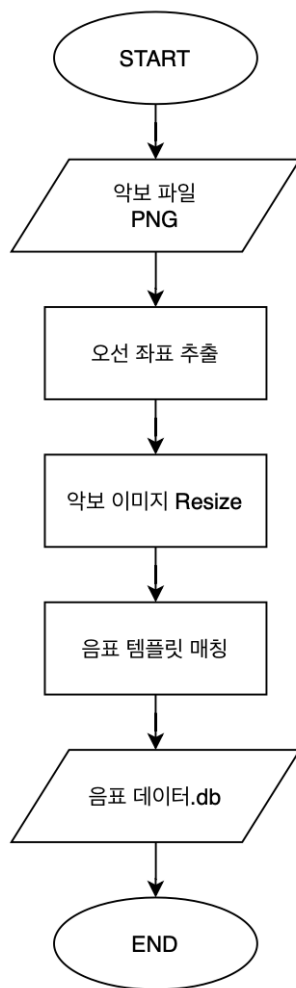


그림 6. 템플릿 매칭 통한 음계 인식 모듈
Fig 6. Melody Recognition module by Template Matching

위의 그림 6과 같이 템플릿 매칭의 정확도를 올리기 위하여 템플릿 매칭할 음표 이미지와 악보 음표의 이미지의 크기가 비슷하거나 같아야 한다. 따라서 먼저 PNG로 변환 시킨 악보 이미지에서 오선을 찾아 오선의 간격을 비교하여 악보 이미지를 확대 및 축소 시켜 비율을 맞춘다. 그 후 템플릿 매칭을 통해 음표 좌표 데이터를 얻고, 이를 활용하여 음계 데이터를 얻는다. 음계 데이터 중 아래 그림 19, 20 과 같은 음계 데이터는 따로 구하여 변경해야한다.

음계를 인식한 후 CNN 을 통해 음계에 대한 박자를 인식한다. CNN 을 통한 음계 인식 모듈 순서도는 아래 그림 7 과 같다.

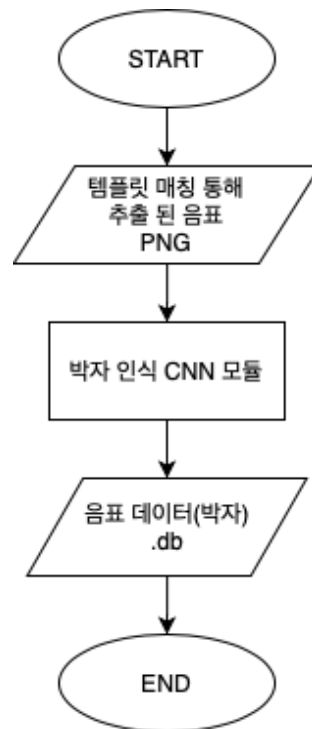


그림 7. CNN 통한 박자 인식 모듈
Fig 7. Tempo Recognition module by CNN

위의 그림 7 와 같이 CNN 을 통해 음표 박자 인식한다. 앞서 설명한 템플릿 매칭을 통해 사각형 모양의 음표 데이터를 list 에 순서대로 저장된 음표 PNG 를 CNN 모델에서 반복 훈련 시킨다. Conv2D, MaxPooling, Dense, Flatten 레이어 층을 추가하여 음표의 박자 인식하는데 최적화된 CNN 모델을 구현할 것이다. 구현한 CNN 모델을 통해 음표 박자 데이터를 학습시킬 것이다. 학습 시킨 모델을 통해 음표의 박자를 분류하여 데이터베이스에 저장한다.

악보 음표의 박자를 인식하기 위해 구현한 CNN(Convolutional Neural Network) 모델의 구조를 아래 그림 8, 그림 9 와 같이 시각화 하였다.

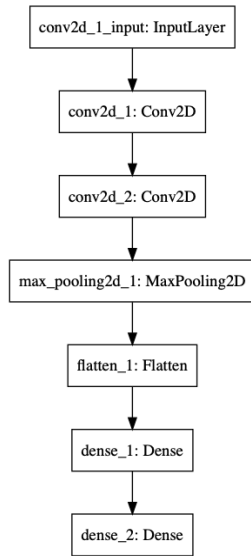


그림 8. 음표 박자 인식 모델 시각화

Fig 8. Model Visualization of Note Tempo Recognition

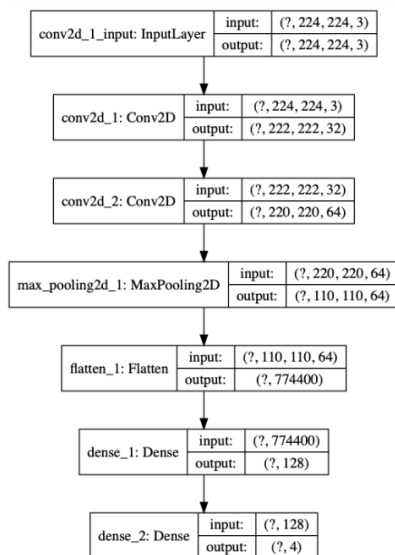


그림 9. 음표 박자 인식 모델 입/출력 시각화

Fig 9. Model Visualization of Note Tempo Recognition I/O

위의 그림 8 을 통해 박자 인식을 위한 CNN 모델이 Input 레이어를 거친 후 2 개의 Conv2D, MaxPooling2D, Flatten 레이어, 마지막 2 개의 Dense 레이어를 거친다.

그림 9 에서 '?'는 CNN 모델에 박자를 분류 할 음표 이미지 파일이다.

그림 9 를 통해 Input 레이어의 입력 값은 224 X 224 픽셀, 입력 이미지 처리 필터 채널

이 3 개이고, 출력 값은 입력과 마찬가지로 224 X 224 픽셀, 입력 이미지 채널 3개이다. 첫 번째 Conv2D 레이어의 입력 값은 필터 크기 224 X 224, 필터의 갯수 3개, 출력 값 필터 크기 222 X 222, 32 개의 필터이다. 두 번째 Conv2D 레이어의 입력 값은 필터 크기가 222 X 222 이고, 32 개의 필터를 갖는다. 출력 값의 필터 크기는 220 X 220 이고, 64 개의 필터를 갖는다. MaxPooling2D 레이어의 입력 값은 필터 크기 즉 풀의 크기가 220 X 220, 64 개의 풀을 갖고, 출력 값으로 풀의 크기 110 X 110, 64 개의 풀을 갖는다. Flatten 레이어의 입력 값은 필터 크기가 110 X 110, 64 개의 필터를 갖으며, 출력 값으로 1차원 필터 크기 774,400 를 갖는다. 첫 번째 Dense 레이어의 입력 값은 입력 뉴런의 수가 774400 개임을 뜻하며, 출력 뉴런의 수가 128 개 임을 뜻한다. 두 번째 Dense 레이어의 입력 값으로 입력 뉴런의 수가 128 개이며, 출력 값으로 출력 뉴런 수가 4 개이다.

2.1.2 악보 코드 변환 모듈

OurChord 악보 코드 변환 모듈의 순서도는 아래 그림 10 과 같다.

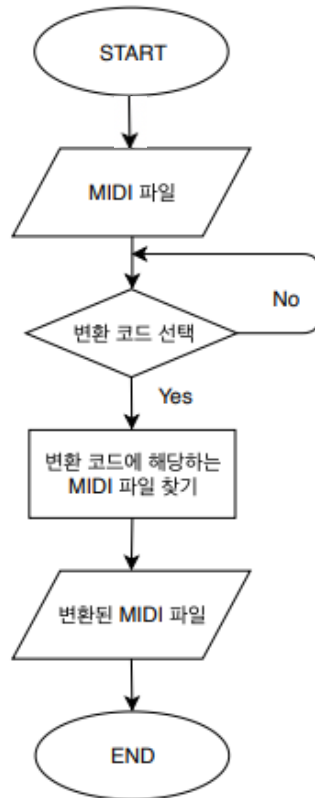


그림 10. 악보 코드 변환 모듈
Fig 10. Sheet Music chord change module

사용자가 원하는 조(chord)와 기존 악보의 조를 입력한다. 위의 MIDI 파일 변환 모듈에서 사용하는 Python 의 pyknon 라이브러리를 사용하여 사용자가 원하는 조의 새로운 MIDI 파일을 제공한다.

2.1.3 MIDI 파일 변환 모듈

OurChord MIDI 파일 변환 모듈의 순서도는 아래 그림 11 과 같다.

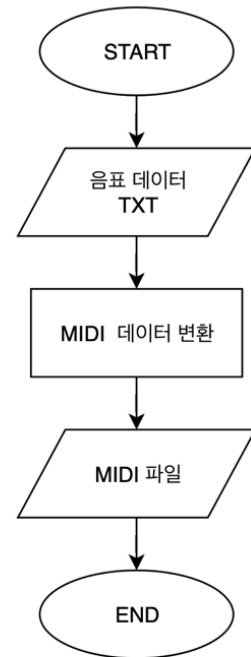


그림 11. MIDI 파일 변환 모듈 순서도
Fig 11. MIDI File Conversion Flow Chart

Python 의 pyknon 라이브러리를 사용하여 MIDI 파일을 생성할 것이다. 추출한 음표 데이터를 lists로 입력 받은 후 입력 받은 음표 데이터를 NoteSeq 함수를 사용하여 MIDI 데이터로 변환한다.

2.2 문제 및 해결 방안

OurChord 개발 시 해결해야하는 문제의 첫 번째는 PDF 파일을 PNG 파일로 변환해야 하는 것이다. 두 번째, 음과 정확한 음표 추출을 위해 오선 좌표를 알 수 있어야 한다. 세 번째 문제는 음표의 좌표를 정확하게 인식할 수 있도록 음표의 좌표를 알 수 있어야 한다.

위의 첫번째 PDF 파일을 PNG 파일로 변환해야하는 문제를 해결하기 위해 Python 의 fitz 라이브러리를 사용하여 PDF 파일을 PNG 파일로 변환하는 클래스를 생성하여 변환한다. PNG 파일로 변환 후 파일 크기가 달라 졌는지 악보 이미지가 변한 부분이 있는지 확인 (세부 크기) 한다.

두번째, 악보 오선 추출 시 cv2.HoughLinesP로 오선의 직선이 안 뽑히는 경우가 발견되었다. 모든 오선을 추출하기 위해 방법을 바꾸어 이미지를 흑백 이미지로 변환 시켜 각 픽셀의 흑백 임계 값을 확인하여 악보 오선을 추출한다.

세번째, 음표 추출을 위해 CNN과 템플릿 매칭(Template Matching)을 이용해 정확한 음표 추출한다. CNN 사용 시 각 음표 데이터를 훈련용 음표 데이터를 음표당 240개, 테스트용 음표 데이터를 음표 당 120개, 템플릿 매칭을 위한 음표 데이터를 음표 당 30개씩 아래 그림 12, 그림 13 과 같은 방법으로 라벨링 한다.

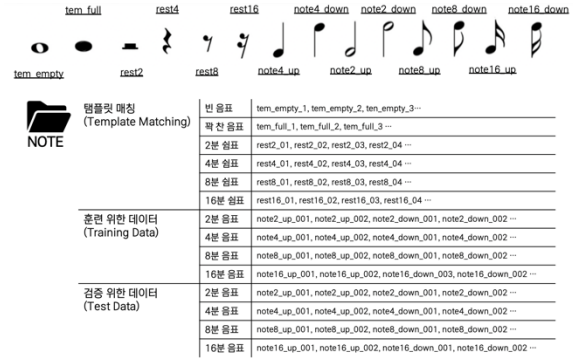


그림 12 음표 데이터 라벨링 예시

Fig 12. Note Data Labeling example

- 템플릿 매칭 위한 음표 데이터



- 훈련 위한 데이터 / 검증 위한 데이터



그림 13. 음표 데이터 라벨링 상세 예시

Fig 13. Note Data Labeling detail example

라벨링 한 음표 데이터로 음표 데이터베이스를 구축한 뒤 아래 그림 14 과 같은 음표 데이터베이스를 이용하여 CNN(Convolution Neural Network) 모델에 학습시킨다. 반복 된 학습을 통해 음표 인식률을 더 높인다. 학습 시킨 데이터를 바탕으로 음표 인식 후 음표의 좌표를 추출하여 정확한 음표 박자 데이터를 얻는다.

TemplateNo	tem_full	tem_empty	rest2	rest4	rest8	rest16
tem1	tem_full_1	tem_empty_1	rest2_1	rest4_1	rest8_1	rest16_1
tem2	tem_full_2	tem_empty_2	rest2_2	rest4_2	rest8_2	rest16_2
...

TrainNo	note2_up	note2_down	note4_up	note4_down	note8_up	note8_down	note16_up	note16_down
train1	note2_up_1	note2_down_1	note4_up_1	note4_down_1	note8_up_1	note8_down_1	note16_up_1	note16_down_1
train2	note2_up_2	note2_down_2	note4_up_2	note4_down_2	note8_up_2	note8_down_2	note16_up_2	note16_down_2
...

TestNo	note2_up	note2_down	note4_up	note4_down	note8_up	note8_down	note16_up	note16_down
test1	note2_up_1	note2_down_1	note4_up_1	note4_down_1	note8_up_1	note8_down_1	note16_up_1	note16_down_1
test2	note2_up_2	note2_down_2	note4_up_2	note4_down_2	note8_up_2	note8_down_2	note16_up_2	note16_down_2
...

그림 14. 음표 데이터베이스 예시

Fig 14. Score Database example

템플릿 매칭을 사용하여 음표 추출 확률을

높이기 위해 많은 샘플 이미지를 음표 데이터 베이스에 가지고 있을 것이다. 많은 음표 이미지를 통해 음표 추출 시 정확도를 더 올린다.

2.3 시스템 및 시험 시나리오

2.3.1 시스템 시나리오

OurChord 의 전체 시나리오는 아래 그림 15 와 같다.



그림 15. 전체 시스템 수행 시나리오

Fig 15. Full System Execution Scenario

위의 그림 15 에서 확인 할 수 있듯이 첫 번째 과정은 악보 PDF 파일을 OurChord 프로그램에 입력하는 것이다. 두 번째로 OurChord 프로그램에서 PDF 악보 파일을 전송한다. 세 번째, 서버에서 변환 된 PDF 악보를 PNG로 변환하여 악보의 음표를 인식한 후 MIDI 파일로 변환한다. 변환 된 MIDI 파일을 OurChord 프로그램에 전송한다. 네 번째 OurChord 프로그램은 웹 서버에서 받은 MIDI 파일을 출력하는 위의 과정이 전체 시스템 수행 시나리오이다.

사용자가 악보 파일 입력 시 시나리오는 아래 그림 16 과 같다.

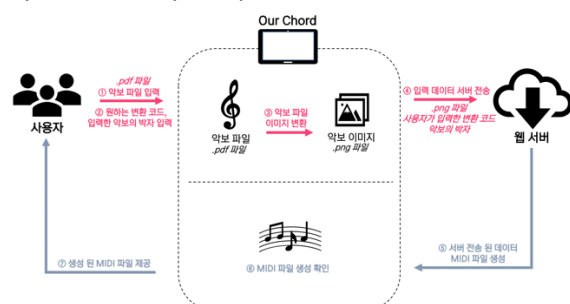


그림 16. 사용자 악보 파일 입력 시나리오

Fig 16. User Score File Input Scenario

위의 그림 16 에서 확인할 수 있듯이 첫 번째

째 사용자가 OurChord 프로그램에 악보 PDF 파일을 입력한 후, 두 번째로 사용자가 기존 악보의 코드, 변환 코드(chord)와 악보의 박자를 입력한다. 세 번째로 OurChord 프로그램에서 사용자가 선택한 PDF 파일과 두 번째 단계에서 사용자가 입력한 정보를 서버로 전송한다. 네번째, OurChord 프로그램에서 서버로 보낸 PDF 파일을 서버에서 PNG 파일로 변환한다. 다섯 번째, 서버에서 OurChord 프로그램에서 전송된 데이터를 통해 MIDI 파일을 생성하여 OurChord 프로그램에 전송한다. 여섯 번째로 OurChord 프로그램에서 MIDI 파일이 생성되었는지 확인한다. 마지막으로 생성된 MIDI 파일을 OurChord 프로그램이 사용자에게 제공한다.

악보 입력 후 조옮김 수행 시나리오는 아래 그림 17 과 같다.



그림 17. 악보 입력에 따른 조옮김 변환 시나리오
Fig 17. Chord Conversion Scenario Based on Score Input

위의 그림 17 에서 확인 할 수 있듯이 악보 입력 후 서버에서 조옮김 시 첫 번째로 서버에 악보 PNG 파일을 입력한다. 두 번째 서버에서는 악보 PNG 이미지 파일에서 음표 데이터를 추출한다. 추출한 데이터를 배열로 저장한다. 세 번째 추출한 음표 배열을 Python 의 Pyknon 라이브러리의 Noteseq 함수를 사용하여 MIDI 데이터로 변환한다. 네 번째 변환된 MIDI 데이터를 통해 사용자가 원하는 조(chord)로 MIDI 데이터를 변환한다. 다섯 번째로 변환된 MIDI 파일을 확인하고, 변환된 MIDI 파일을 OurChord 프로그램을 통해 사용자에게 출력한다.

2.3.2 시험 시나리오

악보 PDF파일 입력 후 PNG 파일로 변환된 악보가 크기 변환 없이 잘 변환되었는지 속성 자세히 보기에서 크기와 함께 확인한다.

오선 좌표 출력 시 오선의 y좌표를 print로 출력하여 개수를 확인한다. cv2.line 함수를 사용하여 오선의 위치가 맞는지 확인 한다. 다른 악보 PNG로 반복하여 테스트한다.

음표 좌표 출력 시 구한 좌표에 대하여 cv2.Rectangle 함수를 사용하여 맞게 구하였는지 실제 출력 된 결과를 확인하고 개수 확인을 한다. 만약 추출되지 않은 음표가 있다면 매칭(Matching) 확률을 조정하여 추출되지 않은 음표 사진을 라벨링 하여 lists에 추가하여 출력 한다.

MIDI 파일 변환 시 추출한 음, 박자를 MIDI 파일로 생성 후 MIDI 파일 실행을 할 수 있는 GarageBand, KMIDIPlayer 또는 vanBasco's Karaoke player 등을 사용하여 맞게 저장되었는지 비교 확인한다.

조옮김 변환 시 입력 받은 원래 조(chord)와 바꾸고 싶은 조의 차이가 정확하게 나오는지 테스트한다.

서버 확인 시 서버 연동 후 서버에서 음표 추출을 실행하여 결과 값이 잘 나오는지 확인하고 어플리케이션(Application)과 연동 되어 정확하게 입력 받고 출력 되는지 확인한다.

통합 확인 시 먼저 알고리즘 및 기능들을 전체적으로 합쳐서 테스트한다. 서버에서는 모듈 별 실행 과정 시 각 수행 모듈에 해당하는 출력을 통해 확인하고, 어플리케이션에서는 Log.w 를 통해 입력과 출력이 잘 되는지 확인

한다. 그 후 서버에 구성된 알고리즘을 입력하여 어플리케이션에서 입력하면 원하는 출력값이 나오는지 확인한다.

2.4 상세 설계

2.4.1 PDF to PNG 모듈

입력받은 PDF 파일을 PNG 파일로 형식을 변환한다. PDF 형식의 악보파일을 PNG 형식으로 변환하는 함수는 아래 표 3 과 같다.

표 3. 사용자가 입력한 PDF 파일 PNG 변환 함수

Table 3. PDF file PNG Conversion Function Entered by User

PDF2PNG()	
형식	PDF2PNG(scoreID)
리턴 값	PDF를 PNG로 변경한 악보 파일
설명	변환 된 PNG 파일을 서버에 저장한다.
예시	Image(filename = 'test.pdf')

표 3 에서 확인할 수 있듯이 사용자가 입력한 PDF 파일을 PNG 파일로 변환하기 위해 입력된 악보의 ID 를 받아온다. 이후, PDF 형식을 save() 함수를 통해 PNG 형식으로 변환하여 저장한다.

2.4.2 음 출력 모듈(튜닝 위한 음 출력)

사용자로부터 입력 받은 '조'에 해당하는 튠링 음 파일의 경로에 접근하여 해당하는 MP3파일을 출력한다. 사용자가 선택한 튠링 음을 저장하는 함수는 아래 표 4 와 같다.

표 4. 사용자가 선택한 듣고 싶은 '조' 저장 함수

Table 4. User Selected 'Chord' Save Function

getSound()	
형식	getSound()
리턴 값	사용자가 선택한 조(chord)
설명	사용자가 듣고 싶은 튠링음의 '조'값 가져온다.
예시	Sound = getSound(sound)

표 4 에서 확인할 수 있듯이 사용자가 선택한 튠링 음의 '조' 값(A, B, C 등)을 getSound() 함수를 통해 받아와 Sound변수에 저장하여

playSound() 함수로 전달한다. 출력할 음의 경로를 불러오는 함수는 아래 표 5 와 같다.

표 5. 선택한 '조'에 해당하는 소리 출력 함수

Table 5. Sound Output Function Corresponding to Selected 'Chord'

playSound()	
형식	playSound(sound)
리턴 값	해당 조의 음
설명	해당 '조'에 맞는 소리를 출력한다.
예시	Sound = playSound(sound)

위의 표 5 에서 확인할 수 있듯이 sound 변수를 통해 입력 받은 '조'를 확인하고 해당 '조'에 해당하는 MP3 파일 경로로 접근하여 playSound()함수를 통해 출력한다.

2.4.3 오선지 좌표 추출 모듈

오선지에 대한 좌표 추출을 위해 첫번째로 악보 이미지를 읽어온다. 두번째로는 악보 이미지를 이미지를 각 픽셀씩 확인하여 임계값(설정 값)과 비교 하여 검은색에 가까우면 1, 아니면 0으로 데이터를 수집한다. 후 y좌표로 데이터를 정렬하여 악보 이미지 너비의 80% 넘게 검은 색일 경우의 좌표를 추출한다. 세번째로 추출한 좌표들의 중복 좌표를 제거한다.

첫번째, 이미지를 읽어오는 함수는 아래 표 6 과 같다.

표 6. 이미지를 읽어 오는 함수

Table 6. Function of Read ScorePNG

readPNG()	
형식	readPNG()
리턴 값	해당 PNG 이미지
설명	원하는 악보 PNG 이미지를 읽어 온다.
예시	Img = readPNG(pngID)

위의 표 6 에서 확인할 수 있듯이 사용자가 입력한 PDF 악보를 PNG 로 변환하여 데이터

베이스에 저장 된 pngID 를 readPNG() 함수를 통해 원하는 악보 PNG 를 읽어 온다.

표 7. 오선지 직선 검출 함수

Table 7. Function of Detect Straight Line

detectStraightLine()	
형식	List detectStraightLine(Image)
리턴 값	검출 된 직선의 x, y 좌표
설명	이미지에서 직선 검출하여 x, y 좌표 리턴
예시	xylist[] = detectStraightLine(Img)

위의 표 7 에서 확인할 수 있듯이 전처리 된 PNG 파일에서 detectStraightLine() 함수를 통해 xylist[] 에 검출한 직선의 y좌표를 반환한다.

2.4.4 템플릿 매칭 통한 음계 분류 모듈

템플릿 매칭을 통한 음계 분류는 먼저, 악보 PNG 파일을 읽어온 후 전처리 과정을 거친다. 위의 전처리 과정은 악보의 크기를 변경을 뜻한다. 이 후 크기를 변경한 악보 파일 이미지와 매칭 이미지를 불러온다. 설정 값에 따라 템플릿 매칭을 하여 좌표를 추출한다. 좌표에 대해 중복 좌표를 제거하여 매칭된 좌표만 얻어 음계를 분류 할 것이다. 이미지를 읽어오는 함수는 앞서 오선지 좌표 추출 에서 설명한 표 6 과 같고, 악보 이미지의 크기를 변환하는 함수는 아래 표 8 과 같다.

표 8. 악보 크기 변환 함수

Table 8. Function of Staff Resize

resizeImg(stafflist)	
형식	Image resizeImg(stafflist)
리턴 값	크기를 변환한 악보 이미지
설명	템플릿 매칭 이미지와 크기를 맞추기 위한 크기 변환
예시	Img = resizeImg (stafflist)

위의 표 8 에서 확인할 수 있듯이 resizeImg() 함수를 통해 악보 이미지 음표 크

기를 템플릿 매칭할 음표 이미지 크기에 맞추어 변환하여 저장한다.

템플릿 매칭 함수는 아래 표 9 과 같다.

표 9. 템플릿 매칭 함수

Table 9. Function of Template Matching

templateMatching()	
형식	List templateMatching(Image, List)
리턴 값	매칭 된 이미지의 x, y 좌표
설명	설정 값과 매칭할 이미지를 통해 템플릿 매칭 시 추출한 x, y 좌표 리턴
예시	xynoteList[] = templateMatching(lmg, List)

위의 표 9 에서 확인할 수 있듯이 templateMatching() 함수를 통해 매칭 된 이미지의 x, y 좌표를 xynoteList[] 에 저장한다. 템플릿 매칭 후 중복 된 x, y 좌표 필터링 함수는 아래 표 10 과 같다.

표 10. 중복 매칭 필터링 함수

Table 10. Function of Duplicate Matching Filtering

filteringTem()	
형식	List filteringTem(List)
리턴 값	필터링 된 x, y 좌표 리스트
설명	매칭 된 x, y 중복 좌표 리스트들을 필터링하여 음표 좌표 리턴
예시	noteList[] = filteringTem(xynoteList)

위의 표 10 에서 확인할 수 있듯이 filteringTem() 함수를 통해 앞서 templateMatching() 함수에서 추출한 x, y 좌표 중 중복 된 좌표들을 검출한 음표의 x, y 좌표를 반환한다.

음계 분류 위한 함수는 아래 표 11 과 같다.

표 11. 음계 분류 함수

Table 11. Function of Classification Melody

noteScale(notelist, stafflist)

형식	List noteScale(notelist, stafflist)
리턴 값	저장 할 이미지 파일
설명	음표 음계를 구분하여 활용 하기 위해 List 저장
예시	scaleList = noteScale(notelist, stafflist)

위의 표 11 에서 확인할 수 있듯이 noteScale() 함수를 통해 noteList에 들어 있는 음표들의 음계를 추출한다.

음표 박자를 확인하기 위한 음표 이미지 추출하여 저장하는 함수는 아래 표 12 와 같다.

표 12. 음표 이미지 저장 함수

Table 12. Function of Save Note

noteBeat(notelist, stafflist)	
형식	Image noteScale(notelist, stafflist)
리턴 값	저장 할 이미지 파일
설명	음표 박자를 구분 하기위해 음표 이미지 저장
예시	Imagelist= noteScale(notelist, stafflist)

음표 박자 분류를 위해 악보의 음표들을 하나씩 이미지로 저장하여야 한다. 이를 위 음표 좌표 추출 함수를 통해 추출한 음표 좌표를 활용하여 이미지를 자른다. 입력 값으로는 음표 좌표 리스트, 오선 좌표 리스트가 있다. 오선 좌표 중 가운데 오선 좌표를 기준으로 음표 y 좌표가 기준보다 위에 있으면 아래로 아래에 있으면 위로 음표 이미지를 자른다. 오선 좌표 기준으로 자른 음표 이미지를 데이터베이스에 저장한다.

2.4.5 음표 박자 분류 위한 모델 생성 모듈

음표 박자 분류를 위한 모델 생성 시 훈련 데이터와 테스트할 데이터의 경로를 지정한 후 음표를 학습 시킬 모델을 생성한다. 이 후 학습 시킬 모델을 반복하여 훈련 시켜 음표 분류의 정확도를 높인다.

훈련 데이터의 경로를 불러오는 함수는 아래 표 13 와 같다.

표 13. 훈련 데이터 경로 지정 함수

Table 13. Function of Training Data Path

newTrainPNG()	
형식	Image newTrainPNG(Image)
리턴 값	훈련할 이미지 파일
설명	CNN 위해 훈련할 이미지 경로
예시	Train = newTrainPNG(Image)

위의 표 13 의 newTrainPNG() 함수에 딥러닝 시 훈련할 음표 데이터의 경로를 지정한다.

표 14. 검증 데이터 경로 지정 함수

Table 14. Function of Test Data Path

newTestPNG()	
형식	Image newTestPNG(Image)
리턴 값	테스트할 이미지 파일
설명	CNN 위해 테스트할 이미지 경로
예시	Train = newTestPNG(Image)

위의 표 14 에서 확인할 수 있듯이 newTestPNG() 함수에 딥러닝 시 검증할 음표 데이터의 경로를 지정한다.

딥러닝 시 음표 데이터를 훈련하고 검증할 모델을 구성하는 함수는 아래 표 15 와 같다.

표 15. 모델 생성 함수

Table 15. Function of Model Create

addModel()	
형식	Model = addModel(model)
리턴 값	훈련하기 위해 새로 구성 된 모델
설명	CNN 위해 훈련할 모델 구성
예시	addModel(Conv2D) addModel(MaxPooling2D) addModel(Dense) addModel(Flatten)

위의 표 15 에서 확인할 수 있듯이 addModel() 함수에 딥러닝 위한 모델을 레이어를 추가하여 생성한다.

영상 처리를 위해 Conv2D 레이어를 추가하여 음표 데이터의 경계 처리와 활성화 함수를 지정한다. Max Pooling 레이어를 추가하여 Conv2D 레이어에서 주요 값만 추출하여 작은 변화가 특징을 추출할 때 큰 영향을 미치지 않도록 한다. Flatten 레이어를 추가하여 Conv2D 레이어와 Max Pooling 레이어를 거치며 추출 된 특징을 1차원 자료로 바꾸어 준다. Dense 레이어를 추가하여 입력 뉴런 수와 출력 뉴런 수, 활성화 함수를 지정하여 음표를 정확하게 분류하기 위한 모델을 생성한다. 생성한 모델을 사용하여 음표 데이터를 훈련 시키기 위한 함수는 아래 표 16 와 같다.

표 16. 모델 훈련 함수

Table 16. Function of Model Train

trianModel()	
형식	Model = trainModel(model)
리턴 값	훈련 된 모델
설명	CNN 위해 훈련할 모델
예시	load.model(trainModel.h5)

앞서 설명한 addModel 함수를 통해 생성 된 모델을 통해 음표 데이터들을 훈련한다. 이 후 위의 표 16 에서 확인할 수 있듯이 trainModel() 함수를 통해 훈련 시 훈련 된 정보가 기록 된 H5 파일을 추출하고, 다음 훈련 시 H5 파일을 load 하여 더 정확도를 높인다.

2.4.6 박자 인식 모델 통한 음표 박자 분류

2.4.5 에서 설명한 박자 인식 위한 딥러닝 모델이 생성되면 템플릿 매칭을 통해 얻은 음표 이미지를 가지고 각 음표의 박자를 구분한다.

템플릿 매칭을 통해 얻은 음표 이미지를 가

저오는 함수는 위의 표 6 의 readPNG() 함수를 사용한다.

박자 인식 모델에 불러온 이미지를 넣어 2, 4, 8, 16분음표인지 구분한다. 박자 인식 모델에서 박자 별 예측 정확도가 0.5 이상일 경우 각 2, 4, 8, 16분음표인 것으로 인식한다. 왜냐하면 박자 인식 모델 생성 시 아래 그림 18 의 왼쪽 2분음표 모양과 오른쪽 2분음표 모양처럼 2, 4, 8, 16분음표를 구분하여 훈련하였기 때문이다. 박자 인식 모델에 8가지 분류 기준을 둔 것인데, 8가지 분류 기준의 가중치를 모두 더하면 1이다. 모델 생성 시 각 음표의 특징을 추출하도록 했기 때문에 2분음표의 특징을 모델이 인식했다면 예측 정확도 0.5를 충분히 넘길 것이기 때문에 융통성을 가지는 딥러닝 모델의 정확도를 높이기 위해 예측 정확도를 0.5 로 정했다.



그림 18. 2분음표 꼬리 위치 별 모양 예시
Fig 18. Example of Half Note tail up, down

박자 인식 모델을 통한 음표 박자 분류 함수는 아래 표 16 과 같다.

표 16. 박자 분류 함수

Table 16. Function of Classification each Note Tempo

predictTempo()	
형식	readPNG()
리턴 값	해당 PNG 의 박자
설명	예측 정확도 0.5 이상일 경우 박자를 출력한다.
예시	lists = predictTempo('2', '4', '8', '16')

2.4.7 조 변환 모듈

조 변환은 기존의 악보를 사용자가 원하는 조로 바꾸는 것이다. 먼저 기존 악보에 대한 조와 변하고 싶은 조에 대한 데이터를 얻는다. 그 후 위의 과정들을 통해 얻은 악보 음에 대한 데이터를 읽어와 기존의 음을 변하고 싶은 조의 음으로 변환한다.

입력한 악보의 코드(조)를 가져오는 함수는 아래 표 17 과 같다.

표 17. 악보의 코드를 가져오는 함수

Table 17. Function of Base Score Chord

getBaseChord()	
형식	getBaseChord()
리턴 값	사용자가 입력한 기존 악보의 조(chord)
설명	조 변환을 위해 사용자가 입력한 기존 악보의 조(chord)를 가져온다.
예시	baseChord = getBaseChord('C')

위의 표 17 에서 확인할 수 있듯이 사용자가 입력한 코드를 getBaseChord() 함수를 통해 가져온다. 사용자가 변하고 싶은 코드(조)를 가져오는 함수는 아래 표 18 과 같다.

표 18. 조옮김 악보 MIDI 파일 저장 함수

Table 18. Function of Change Score Chord

getChangeChord()	
형식	getChangeChord()
리턴 값	사용자가 입력한 변환하려는 조(chord)
설명	조 변환을 위해 사용자가 입력한 변환하고자 하는 코드를 가져온다,
예시	changeChord = getChangeChord('D')

위의 표 18 에서 확인할 수 있듯이 기존 악보의 조(chord)와 변하고 싶은 조(chord)를 읽어 오고 앞서 추출한 음표 데이터를 불러온다. 그 후 불러온 데이터들을 가지고 조 변환 함수를 통해 조 변환 한다.

조 변환 함수는 아래 표 19 과 같다.

표 19. 조옮김 악보 MIDI 파일 저장 함수

Table 19. Function of Change Note

changeNote()	
형식	List changeNote()
리턴 값	조 옮김 된 데이터
설명	조 변환 MIDI 생성 위한 데이터
예시	changeNote('C', 'D', '4')

위의 표 19 에서 확인할 수 있듯이 조 변환을 위해 사용자가 입력한 기존 악보의 음과 변환하고자 하는 조(chord)의 음, 해당 음에 대한 박자를 MIDI 데이터 변환 위해 changeNote() 함수에서 저장한다.

2.4.8 MIDI 데이터 변환 모듈

기존 악보 파일에 대한 MIDI 데이터 변환 함수는 아래 표 20 과 같다.

표 20. 기존 악보 MIDI 파일 저장 함수

Table 20. Function of Origin Socre MIDI save

saveBaseMIDI()	
형식	MIDI baseMidiList(List)
리턴 값	기존 악보 MIDI 파일
설명	기존 악보의 MIDI 데이터 저장
예시	baseMidi = saveBaseMIDI()

위의 표 20 에서 확인할 수 있듯이 saveBaseMIDI() 함수는 는 기존 악보 파일에 대한 MIDI 데이터를 저장하여 MIDI 파일을 생성해주는 함수이다. baseMidiList 에 위의 PDF to PNG, 오선지 좌표 추출, 템플릿 매칭, 딥러닝을 통해 음표 데이터를 추출해낸다. 추출한 데이터를 List 형식으로 저장한다. 추출한 음표 데이터들을 saveBaseMIDI 함수에 입력하여 MIDI 데이터로 변환한다.

조 옮김 된 MIDI 데이터 변환 함수는 아래 표 21 과 같다.

표 21. 조옮김 악보 MIDI 파일 저장 함수

Table 21. Function of Chord Conversion MIDI save

saveChangeMIDI()	
형식	Midi saveChangeMIDI(List)
리턴 값	변환 된 악보 MIDI 파일
설명	조옮김 된 MIDI 데이터 저장
예시	changeMidi = saveChangeMIDI()

위의 표 21 에서 확인할 수 있듯이 saveChangeMIDI() 함수는 조옮김 된 음표 데이터들을 통해 기존 악보에서 사용자가 변환하고자 하는 조(Chord)로 변환 된 MIDI 데이터를 생성해준다. changeMidi 에 조 변환 된 음표 데이터를 List 형식으로 저장한다. List 형식으로 조 변환 된 음표 데이터를 saveChangeMIDI 함수에 입력하여 MIDI 데이터로 변환한다.

다음 함수들은 프로토타입 개발 이 후 정확도를 더 높이고 음표 인식률을 더 높이기 위해 추가로 구현한 함수이다.

기존에는 위의 오선지 직선 검출 함수인 detectStraightLine 를 사용하여 악보 전체를 오선으로 분류하고 이미지를 추출했다. 하지만 아래 표 22 의 함수를 사용하여 악보의 오선 별 이미지를 추출하여 활용하면 예전의 전체 악보 이미지에서 템플릿 매칭한 것에 비해 속도는 빨라지고, 정확도는 올랐다.

악보를 오선 별 분류하여 이미지를 자르는 함수는 아래 표 22 와 같다.

표 22. 조옮김 악보 MIDI 파일 저장 함수

Table 22. Function of Divide Score

divideScore()	
형식	List divideScore(Image, List)

리턴 값	오선으로 구분된 이미지 리스트
설명	악보에서 오선으로 이미지를 자름
예시	divideImgList= divideScore (img, detectList)

위의 표에서 확인할 수 있듯이 divideScore() 함수는 악보 이미지를 오선의 y 좌표 리스트를 활용하여 이미지를 오선으로 자른 것이다. 위의 이미지를 활용하기 위해 악보 이미지 전체를 사용하던 부분은 모두 오선으로 구분된 이미지 활용으로 바꾸었다.

악보의 음표 중 그림 19 (이하 "잇단 음표" 라 한다.) 와 같은 음표 중 아래 그림 과 같은 음표들이 존재한다.



그림 19. 꼬리가 이어진 음표 예시

Fig 19. Example of a note with tail connected

템플릿 매칭과 딥러닝 통해 박자 구분 시 위 그림 과 같은 음표는 구분하지 못하여 추가적으로 구현하였다, 자른 오선 이미지에서 침식과 팽창을 이용하여 오선을 지운다. 이 후 픽셀의 색이 검은색인지 확인하고, 검은색이 맞다면 다음 (x+1, y) 좌표의 픽셀을 확인하며 검은색 부분의 가로 직선을 찾고 필터링하여 8분 음표 부분의 x 좌표 범위와 16분 음표 부분의 x 좌표 범위를 찾는다.

악보의 8분 음표, 16분 음표의 x 좌표 범위를 찾는 함수는 아래 표 23 과 같다.

표 23. 8분 음표, 16분 음표 x좌표 범위 찾는 함수

Table 23. Function of checkChangeListener

checkChangeListener()	
형식	List divideScore(lmag)
리턴 값	이어진 8분음표, 16분 음표 x좌표 범위

설명	8분 음표, 16음표에 해당 되는 x좌표 범위를 리턴한다.
예시	checkList= checkChangeListener (img)

위의 표 23 에서 확인할 수 있듯이 checkChangeListener() 함수는 8분 음표, 16분 음표 범위를 알 수 있다. 이 범위를 구해 잘못된 값이 입력 될 수 있는 박자들을 알맞게 8분 음표와 16분 음표로 바꾼다.



그림 20. 화음 음표 예시

Fig 20. Example of a chord note

악보의 음표 중 그림 20 (이하 "화음 음표" 라 한다.) 과 같은 화음 음표가 있다. MIDI 파일을 생성하기 위해 화음 음의 위치를 알아야 화음으로 MIDI 를 구성할 수 있다. 따라서 템플릿 매칭을 통해 음표들의 x, y 좌표를 구하여 필터링을 통해 화음 부분을 구분할 수 있도록 리스트를 추가하였다.

화음 음표를 구분할 수 있게 만드는 함수는 아래 표 24 와 같다.

표 24. 화음 음표 구분 함수

Table 22. Function of Harmony Search

harmonySearch()	
형식	List harmonySearch (List)
리턴 값	화음을 구분 할 수 있는 음표 리스트
설명	음표의 좌표를 통해 화음 위치 확인
예시	harmonyList = harmonySearch(noteList)

위의 표 24 에서 확인할 수 있듯이 harmonySearch() 함수는 음표의 좌표를 통해

화음 위치를 알 수 있다. 이 함수를 통해 얻은 정보로 MIDI 생성 시 화음으로 MIDI 를 구성한다.

2.5 Prototype 구현

2.5.1 요구 분석

‘OurChord’ 는 첫번째, 음악을 시작하려는 사람들이 연주하기에 어려움을 느끼는 기존의 악보보다 좀 더 단순한 악보로 변환하여 도전하는 것에 대한 장벽을 낮추고자 하는 요구를 최우선으로 한다. 두번째, 조옮김을 통해 음악을 시작하려는 사람들 외에도 세션 연주자들을 위해 하나의 코드로 각자의 악기를 튜닝 시 조옮김 MIDI 파일을 제공하려 한다. 세번째로 손쉽게 각자의 악기를 튜닝할 수 있도록 하는 요구에서 개발이 시작되었다.

2.5.2 프로토타입 구현 현황

OurChord 프로토타입 구현 시 공식적인 악보 파일 10 개로 각 모듈을 테스트한 정확도(%)를 아래 표 25 로 나타내었다.

표 25. 프로토타입 모듈별 구현 현황

Table 25. Prototype Each Module Process Status

	구현 현황(%)
PDF to PNG	100%
오선 좌표 추출	100%
음표 좌표 추출	95%
음표 좌표 정렬	100%
음표 사진 추출	85%
음계 추출	95%
딥러닝 모델 학습 모듈	100%
박자 인식 정확도	84%
MIDI 데이터 변환	100%
튜닝 음 출력	100%
조 변환 모듈	100%
96%	

OurChord 프로토타입 구현 시 모듈 외 서버 연동, 데이터베이스, 앱 구현 현황은 아래 표 26 으로 나타내었다.

표 26. 프로토타입 모듈 외 구현 현황

Table 26. Prototype exception Module Process Status

	구현 현황(%)
앱과 서버 통신 모듈	100%
서버 가상환경 구축	100%
앱 화면 구성	90%
서버와 DB 연동	100%
DB 구성	80%
94%	

따라서 프로토타입 전체 구현 현황은 96% 이다.

2.5.2.1 모듈별 구현 현황

프로토타입 구현을 위한 모듈에는 PDF 파일을 PNG 파일로 바꾸는 모듈, 오선 좌표 추출 모듈, 음표 좌표 추출 모듈, 음표 좌표 순서대로 정렬하는 모듈, 음표 사진을 잘라 저장하는 모듈, 음표의 게이름을 추출하는 모듈, 음표의 박자를 인식하는 모듈, 조 옮김 모듈, MIDI 변환 모듈, 튜닝 음 출력 모듈이 있다.

첫번째 PDF 파일을 PNG 파일로 변환하는 모듈은 기존 개발 환경에서 import image 를 통해 PDF 파일이 있는 경로에 접근하여 확장자를 PNG 형식으로 변경하여 저장한다. '1.pdf'의 경우 위의 모듈을 통해 '1-0.png', '1-1.png'과 같이 이미지 이름이 순차적으로 저장되는 것을 확인하였다.

두번째 오선 좌표 추출은 악보 이미지를 흑백 이미지로 바꾸어 각 픽셀을 확인하여 설정 값(임계 값)에 따라 높으면 1 낮으면 0으로 구분하여 y축 히스토그램을 만든다. 현재 이 방법으로 악보 사이즈가 다른 악보들도 정확하게 오선을 뽑는다. 15개를 테스트하여 모두 정확하게 오선을 뽑아 내었다.

음표 좌표 추출은 템플릿 매칭을 통해 구현한다. 템플릿 매칭을 하여 설정한 값에 따라

템플릿 매칭이 성공하면 좌표를 추출한다. 현재 음표 좌표 추출에서 잇단 음표와 화음 음표는 가끔 음표가 매칭이 되지 않는다. 동요 5개 가요 5개를 테스트 한 결과는 아래 그림 21과 표 27 를 통해 확인하였다. 이 후 템플릿 매칭을 오선 좌표를 이용하여 오선 부분만 템플릿 매칭을 하는 방법과 매칭할 이미지들을 추가를 하여 정확도를 올린다.

표 27. 음표 매칭 정확도

Table 27. Note Recognition Accuracy

	총 음표 개수	틀리거나 못 찾은 것	정확도
동요1	75	0	100%
동요2	66	0	100%
동요3	103	0	100%
동요4	159	0	100%
동요5	149	0	100%
가요1	350	9	97%
가요2	205	0	100%
가요3	225	11	95%
가요4	145	15	90%
가요5	310	0	100%
평균	178.7	3.5	98.2%

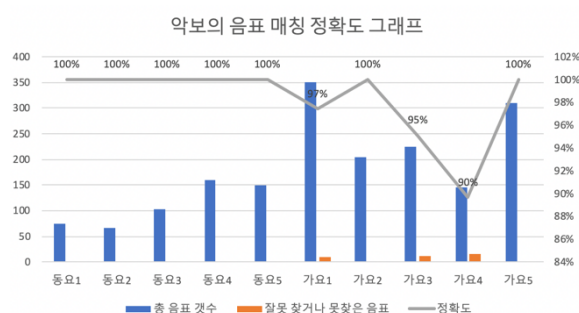


그림 21. 악보의 음표 매칭 정확도 그래프

Fig 21. Sheet Note Matching Accuracy Graph

음표 리스트들은 y 좌표로 정렬되어 있다. 이를 추출한 오선 좌표를 이용하여 오선의 차례대로 구분을 한 후에 x 좌표로 정렬한다.

음표 좌표를 이용하여 음표의 이미지를 잘라내어 박자 구분할 때 사용하려 한다. 현재 음표 사진 자르기는 오선 가운데 좌표 기준으로

위와 아래로 자르게 되어 있다. 하지만 오선을 넘여가는 것과 이어져 있는 위의 그림 29, 30 과같은 음표의 좌표들은 해당 사항에 맞지 않아 보충 작업이 필요하다.

음표의 음계 추출은 오선을 기준으로 음표 y 좌표를 활용하여 구분 하였다. 오선과 오선 사이의 갭을 활용 하여 그 사이에 음표 좌표가 있으면 해당 음을 출력하도록 구현 하였다. 현재 4옥타브의 도부터 5옥타브 미까지 구분하여 추출 가능하다. 후 3옥타브 도부터 6옥타브 도까지 추가 구분할 것이다.

음표 박자 인식은 박자 인식을 위한 딥러닝 모델을 구현하여 구현한 모델을 통해 2, 4, 8, 16분 음표를 인식한다. 현재 박자 인식 모델의 정확도가 92.7% 이기 때문에 계속해서 훈련하여 박자 인식 시 정확도를 더 높일 것이다.

음표의 박자를 인식하기 위해 딥러닝 모델을 구현하였다. 모델을 구현하기 위해 훈련용 음표와 검증용 음표 데이터를 따로 수집하였다. 훈련용 음표 데이터 2, 4, 8, 16분 음표를 각각 240개, 검증용 음표 데이터 2, 4, 8, 16분 음표를 각각 120개씩 수집하였다.

수집한 음표 데이터를 통해 박자 인식 모델을 구현하였다. 첫번째 훈련 시 박자 인식 모델의 정확도는 66.7% 정도로 낮았지만 반복된 훈련을 통해 92.7%로 정확도를 높였다. 악보 10 개에 대해 박자 인식한 정확도를 아래 표 28 과 그래프 그림 22 로 나타내었다.

표 28. 박자 인식 정확도

Table 28. Tempo Recognition Accuracy

	총 음표 개수	틀린 것	정확도
1	48	5	90%
2	35	3	91%
3	30	4	87%
4	51	11	78%
5	52	10	81%

6	55	13	76%
7	29	7	76%
8	30	3	90%
9	49	7	86%
10	38	5	87%
평균	41.7	6.8	84.2%

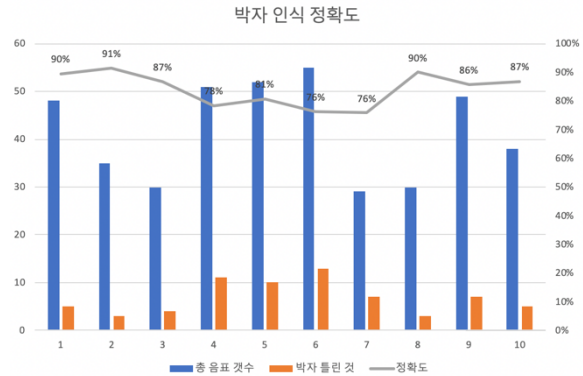


그림 22. 박자 인식 정확도 그래프

Fig 22. Tempo Recognition Accuracy Graph

초반 구현 시 음계와 박자를 모두 딥러닝 (Deep Learning)을 통해 구분하려 했다. 하지만 딥러닝을 위해 C3(3옥타브 도)부터 C6(6옥타브 도)까지 음계 별 박자 별 음표 훈련 데이터 ({(C3 부터 C6 까지음계의 수 X 훈련 데이터 수) X 2, 4, 8, 16분 음표 박자 수} = {(22 X 240) X 4} = 21,120) 검증 데이터({(22 X 120) X 4 = 10,560) 총 31,680개의 음표 데이터를 수집해야 했다.

음표의 머리 부분은 아래 그림 23 과 같이 정형화 되어 있어 템플릿 매칭을 통해 음표의 음계를 인식하는 것이 더 정확 하였다. 이 후 딥러닝을 통해 음표의 꼬리 부분을 통해 박자를 구분하는 것이 더 효율적이고 정확 하였다. 따라서 음표의 음계와 박자 인식을 위해 템플릿 매칭과 딥러닝을 모두 사용하기로 하였다.



그림 23. 음표의 머리 부분 예시

Fig 23. Example of Note Head

3옥타브 도부터 6옥타브 도까지 음에 해당하는 음표들에 대해서 조 옮김을 하는 것을 전제로 진행하고 있기 때문에, 옥타브 별 동일한 음표 즉 다른 옥타브의 같은 음의 구분을 위해 해당 음들의 위치를 숫자로 표현하였고, 미리 배열에 저장된 'C~B'의 조를 통해 사용자가 입력한 '기존 악보의 조'와 '변경할 조'의 갭 차이를 악보에 있는 모든 음표들에 동일한 갭 차이를 적용하여, 조 변환된 음표가 출력된다.

MIDI 변환을 위해 Python의 pyknon 라이브러리를 사용한다. 템플릿 매칭과 딥러닝을 통해 얻은 음계와 음표 데이터를 MIDI 데이터로 변환한다.

스피커에 출력할 'C~B#'조에 해당하는 사용자가 선택한 튜닝음을 해당 '조' MP3파일이 저장된 경로를 통해 접근하여 출력하는 방식으로 테스트하였으며, 해당 방식을 통해 원하는 튜닝 음 파일이 잘 출력되는 것을 확인하였고, 해당 모듈은 어플리케이션 자체에서 수행된다.

2.5.2.2 서버 연동 구현 현황

프로토타입 당시 서버 언어로 사용하고 있는 PHP 에서 Python 코드를 불러오는 과정에서 그 결과물이 웹에 표시되지 않아 DB 에 저장되지 않는 문제, 서버에 이미지와 Python 코드를 업로드 하는 방법, 안드로이드와의 연동 부분이 아직 미흡하였다.

프로토타입 이 후 OurChord 프로그램을 개발하며 서버 환경 구성을 완료하였고, 앱과 DB 연동 시 PHP 가 아닌 Python 을 통해 구현하여 서버와 앱, 서버와 DB 연동을 완료하였다.

2.5.2.3 문제점 및 해결 방안

서버와 데이터베이스 연동 모듈에서 서버 언어로 선택한 PHP 에서 모든 주요 알고리즘이 개발된 언어인 Python 값을 받아오는 것이 vscode 환경에서는 결과물이 잘 출력되는 반면에 웹에서는 출력되지 않아 데이터베이스에 저장이 되지 않는 문제점이 있었다. 이러한 문제점의 원인으로 예상되는 것은 PHP 코드에서 받을 수 있는 파일 형식의 차이로 발생할 수 있는 문제, Python 에서 보낸 값을 PHP 에서 받지 않는 문제등의 여러가지 문제 중 PHP 의 버전 변경, PHP 에서 Python 실행을 위한 설정 변경 등의 원인이 있었다.

위의 문제가 계속 해결되지 않을 시에는 서버 언어 또한 Python으로 변경하여 Django라는 Python 웹 프레임워크를 사용하여 Python 코드 실행 문제를 해결할 예정이었다. 하지만 APMSETUP 은 PHP5버전까지만 지원 하기때문에 Python코드가 실행되지 않는 다는 것을 확인하여 PHP7버전까지 지원하는 XAMPP로 서버 구현을 진행하여 Python 코드가 실행되지 않는 문제와 데이터베이스에 출력 값이 저장되지 않았던 문제를 해결하였다.

데이터베이스 구성 시 템플릿 매칭을 통해 저장된 음표 이미지 파일이 박자 인식을 위한 딥러닝을 위해 일시적으로 저장 되었다가 딥러닝을 통해 박자가 분류 된 후 데이터베이스에서 삭제되어야 했는데, 임시 저장을 하는 부분에서 문제가 있었다. 이 문제는 데이터베이스에 이미지 파일을 임시로 저장하기 위해 전역 임시 테이블을 구성하여 이미지 파일을 임시 저장하는 것으로 해결하였다. 또한 데이터베이스에 이미지 파일 자체를 올리는 것이 아닌 경로를 올리기 위해 먼저 서버에 음표 이미지 파일을 업로드해야 하는데 서버 구축이 완료되는 데로 이미지 파일을 업로드 할

수 있도록 업로드 파일 경로를 저장하는 테이블을 데이터베이스에 추가로 구성하여 서버에서 데이터베이스 확인 시 여러 업로드 파일들을 효율적으로 관리할 수 있도록 구현하였다.

앱 화면 구성 시 요구사항에 따른 가장 편리한 앱 화면을 구성해야했기 때문에 초기 앱 화면 디자인에 많은 수정을 거쳤다. 다양한 악보 PDF 파일과 MIDI 파일을 관리해야 하기 때문에 파일을 한 눈에 확인할 수 있는 UI(User Interface)로 디자인해야 했다. 따라서 PDF 파일과 MIDI 파일을 탭으로 구분하여 앱 화면을 구성하였다. 또한 조옮김 MIDI 파일을 바로 확인할 수 있도록 하기 위해 하단에 주요 기능 탭을 구성하였다. 앱 화면을 최대한 단순하게 하여 따로 앱 사용법을 제공하지 않아도 OurChord 앱의 모든 기능들을 손쉽게 사용 가능하도록 구성하였다.

코드 개발 환경을 기존의 방법인 PyCharm에서 Python 코드 실행을 vscode에서 실행으로 변경한 후, PDF to PNG 모듈 실행을 위한 pip 명령어를 통한 wand 설치와 ImageMagick 라이브러리를 설치 하고, 코드 수행에서 위의 라이브러리가 설치 된 가상환경에 접근할 때, ImageMagick 라이브러리가 설치되어 있지 않다는 에러와 함께 수행되지 않는 문제점이 발생하였다. ImageMagick 라이브러리 설치와 환경변수 설정까지 완료하였음에도 이러한 에러가 왜 발생하는지에 대한 원인을 아직 찾지 못했다.

개발 초반에는 오선의 좌표를 추출하기 위해 cv2.houghlinesp를 사용하였다. 이 함수는 직선을 추출 하는 함수 인데 이 때 두께와 길이 등으로 추출이 가능하였다. 하지만 이 함수로 구현을 하며 테스트 해본 결과 오선이 아닌 부분이나 오선을 추출 하지 못할 때가 많았다. 따라서 오선을 추출하는 방법 자체를 변경하

였다. 악보에서 하나의 픽셀 씩 모두 확인 하는 방법을 사용하여 검은 색 부분이 악보 이미지의 너비의 80% 넘는 y좌표를 추출하는 방법으로 오선을 추출하였다. 테스트 해본 결과 아래 그림 24 와 같은 악보 외에는 정확하게 추출 되었다. 확인해 보니 예외 사진에서는 검은색 부분이 악보 너비의 80%가 넘는 곳이 오선 말고 다른 곳도 존재 하여 생겼다. 위의 문제는 악보 너비의 퍼센트를 더욱 올려 해결할 것이다.



그림 24. 오선 추출 예외 악보 예시

Fig 24. Example Detect Staff Exception Case

템플릿 매칭으로 음표 좌표를 추출 하면서 매칭이 안되는 경우가 있었다. 원인을 분석 해 보니 음표 이미지 크기가 달라서 매칭이 안되는 것 이였다. 하여 이 문제는 매칭할 이미지들의 크기와 악보의 음표 크기를 같게 크기를 조절하여 템플릿 매칭 시 정확도를 올렸다. 그리고 매칭할 음표 이미지의 개수를 늘려 정확도를 더 높일 것이다.

음표 좌표 정렬을 구현할 때 입력 값으로는 y좌표로 정렬된 음표 리스트를 받는다 이를 오선 별로 음표를 나누고 나눈 음표들을 x좌표 순서대로 정렬해야 한다. 이 때 3차원 리스트가 구성이 되어 3차원 리스트를 정렬 및 구성하는데 어려움이 있었다. 이를 3차원 리스트를 만들기 전에 정렬하고 3차원 리스트를 넣어 해결 하였다. 박자 인식을 위해 음표의 사진을 추출할 때 오선과 오선 사이의 중간 값을 기준으로 몇 번째 오선에 있는 음표인지를 구분을 하고 구분 한 음표 리스트의 좌표들을 x좌표로 정렬 하였다.

박자 인식을 위한 정확도 높은 딥러닝 모델을 구현하기 위해 많은 어려움이 있었다. 초반 딥러닝 모델을 구현하기 위해 오선이 없는 음표 이미지를 훈련용, 검증용으로 나누어 수집하였다. 하지만 PDF to PNG 모듈 구현 시 악보를 투명 이미지로 변환하는 것으로 수정되어 오선이 없는 투명 음표 이미지를 새로 수집하여 모델을 구현하였다. 그 이후 악보를 투명 이미지로 변환하지 않고 PNG 형식으로 바꾸기로 하였다. 수집한 데이터가 인식하려는 데이터와 달라 잘못 된 모델 학습을 반복하였다.

최종적으로 박자 인식 딥러닝 모델 구현 시 인식해야 하는 데이터를 오선이 있는 음표 데이터로 정확히 하여 오선이 존재하는 2, 4, 8, 16분 음표를 수집하여 반복 된 훈련으로 정확도 높은 음표 박자 인식 모델을 구현하였고, 정확도는 훈련을 통해 더 높일 예정이다. 현재 박자 인식 딥러닝 모델은 아래 그림 25 와 같은 2, 4, 8, 16 분 음표만 인식 가능하다. 하지만 추가로 잇단 음표와 화음 음표와 같은 음표도 인식 가능하도록 구현할 예정이다.



그림 25. 2, 4, 8, 16 분 음표

Fig 25. half, quarter, eighth, sixteenth Note

MIDI 데이터 변환 시 Python으로 MIDI 데이터를 생성하는 것에 어려움이 있었다. MIDI 데이터를 다루는 것 자체가 처음이었기 때문이다. Python 코드를 통해 MIDI 데이터로 변환하기 위해 Python을 공부하던 중 pyknon 라이브러리가 악보의 요소들 즉 MIDI 데이터를 쉽게 생성하고 수정할 수 있게 해준다는 것을 알게 되었다. 이 후 pyknon 라이브러리에 대해 공부하여 MIDI 데이터 변환 시 pyknon 라이브러리를 사용하여 템플릿 매칭과 딥러닝

을 통해 얻은 음계와 박자 데이터를 가지고 MIDI 파일을 생성하였다.

마지막, 튜닝 음 출력 모듈은 사용자가 선택한 '조'에 해당하는 .mp3 파일이 존재하는 파일의 경로에 접근하여 'A~G#' 중, 해당하는 튜닝 음 파일을 제공하는 과정에서 먼저, 입력된 '조'에 해당되는 파일이 존재하는지 확인하고, 모듈 테스트 과정에서는 '소문자로 입력된 조', '입력되지 않은 경우'등을 고려하는 예외처리를 추가하여 검증하는 과정을 거쳤다. 또한, python의 경우에는 경로 작성 시, 'w'로 인해 발생한 경로 접근 불가 문제도 '/'로 수정하여 해결하였다.

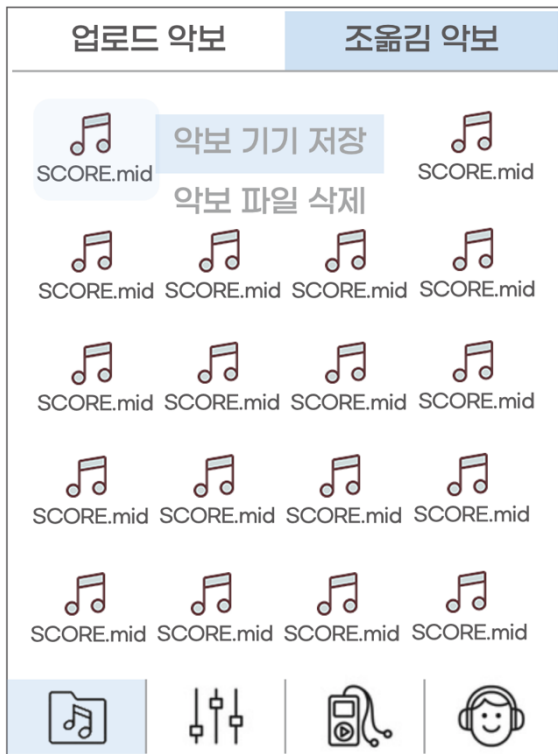


그림 28. OurChord 앱의 악보 폴더 화면
Fig 28. OurChord Application Music Sheet Screen

조옮김 설정 메뉴는 아래 그림 29 과 같이 구성되었다. 조옮김 할 PDF 악보를 선택한 후 선택한 악보의 기존 조(chord) 와 바꾸려는 조를 사용자가 선택한 뒤 'CHORD CONVERSION' 버튼을 클릭하면, 사용자가 변환하고자 하는 PDF 악보 파일을 서버에 전송하여 PNG 형식으로 변환하여 저장하고, 그 PNG 파일을 활용하여 앞서 설명한 모듈을 실행하여 음계, 박자 데이터 추출한다. 추출한 데이터들을 불러와 조 변환 및 MIDI 로 변환 후 그림 28 과 같이 표시 후 확인한다.

위의 과정을 거쳐 조옮김이 완료되어 앞서 설명한 악보 폴더 메뉴의 조옮김 악보 폴더에 기존 악보의 MIDI 파일과 조옮김 된 MIDI 파일이 함께 저장된다.

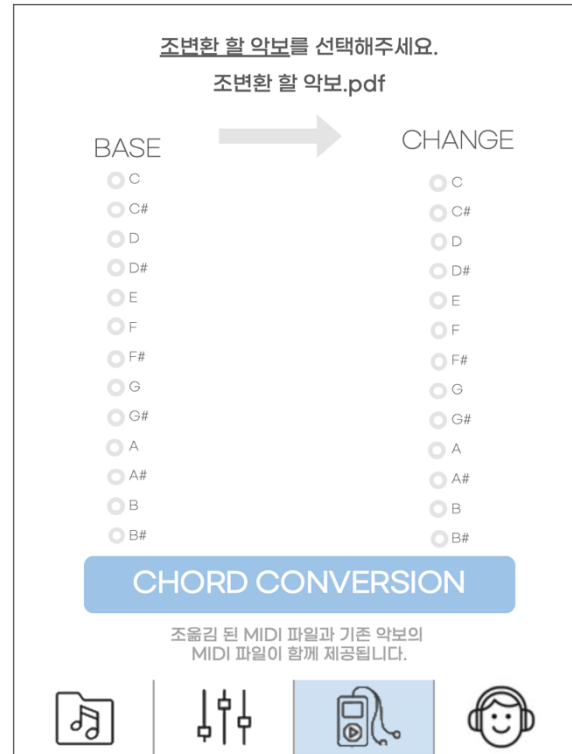
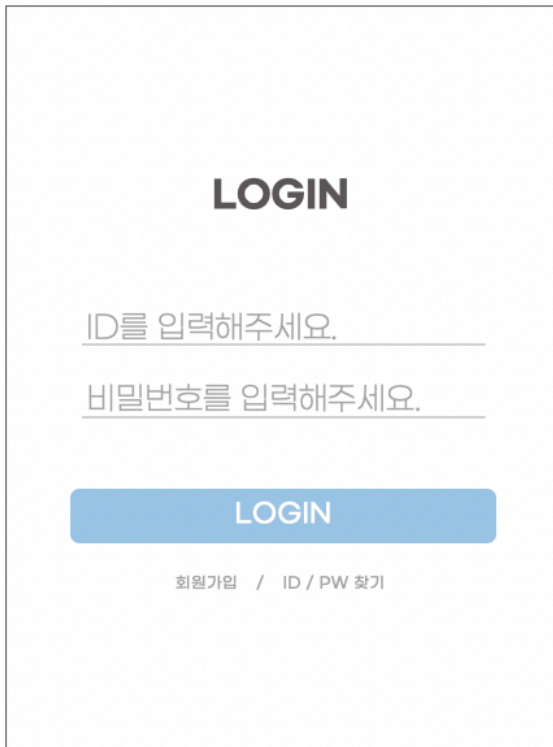


그림 29. OurChord 조옮김 설정 화면
Fig 29. OurChord Application Chord Conversion Set Screen

OurChord 로그인 화면은 아래 그림 30 과 같다. 회원 가입이 되지 않은 경우 회원 가입 글씨를 클릭하여 회원가입 화면으로 이동 할 수 있으며, ID 와 비밀번호를 잊어버렸을 경우 각 글씨를 클릭하면 ID 와 비밀번호를 찾는 화면으로 넘어가게 된다.



The image shows a login screen with a light gray background. At the top, the word "LOGIN" is centered in bold black text. Below it, there are two input fields: the first is labeled "ID를 입력해주세요." and the second is labeled "비밀번호를 입력해주세요.". Both labels are in a light gray font. Below the input fields is a blue button with the text "LOGIN" in white. At the bottom, there is a link that says "회원가입 / ID / PW 찾기" in a small gray font.

그림 30. OurChord 로그인 화면
Fig 30. OurChord Application Login Screen

OurChord 의 ID 와 비밀번호 찾기 화면은 아래 그림 31 과 같다. ID 를 잊었을 경우 이름 과 회원 가입 시 입력한 E-mail 을 입력하면 ID 를 앱 화면에서 알림을 통해 (ex) *zjisu***) 와 같이 확인할 수 있다. 비밀번호를 잊었을 경우 ID 와 E-mail 을 입력하면 ID 를 찾을 때 와 마찬가지로 (ex) *zjisu******) 로 확인할 수 있다.



The image shows a screen for finding ID and password. At the top, the text "ID(PW) 찾기" is centered in bold black text. Below it, there are two input fields: the first is labeled "이름(ID)을 입력해주세요." and the second is labeled "E-MAIL을 입력해주세요.". Both labels are in a light gray font. Below the input fields is a blue button with the text "ID(PW) 찾기" in white. At the bottom, there is a note in a small gray font that says "회원가입 시 입력한 이름과 이메일을 입력해주세요. 입력하신 이메일로 아이디가 전송됩니다."

그림 31. OurChord ID, 비밀번호 찾기 화면
Fig 31. OurChord Application Find Id, Password Screen

OurChord 의 회원 가입 화면은 아래 그림 32 와 같다. 이름과 ID, 비밀 번호, 비밀 번호 재확인, E-mail, E-mail 로 전송 된 인증코드를 모두 입력해야 한다.

ID 는 중복 확인을 거쳐야 하며, 비밀 번호는 8 자 이상 16 자 이하로 영문, 숫자, 특수 문자를 모두 포함하여야 한다. E-mail 입력 후 인증 코드 버튼을 클릭하면 입력한 E-mail 로 인증 코드가 전송되고, 전송 된 인증코드를 올바르게 입력해야 회원가입이 완료된다.



The screen is titled "회원 가입" (Member Registration) at the top center. It features a back arrow on the top left and a home icon on the top right. The registration form includes the following fields and buttons:

- 이름** (Name): A text input field.
- ID**: A text input field with a red "중복확인" (Check for duplicates) button to its right.
- PW** (Password): A text input field with a note below it: "8자 이상 16자 이하의 영문, 숫자, 특수문자 조합" (Combination of English letters, numbers, and special characters, 8 to 16 characters).
- PW 확인** (Confirm Password): A text input field with a note below it: "일치 / 불일치" (Match / Mismatch).
- E-MAIL**: A text input field with a red "인증코드" (Verification code) button to its right.
- 인증코드** (Verification code): A text input field with a red "확인" (Confirm) button to its right.
- 회원 가입** (Member Registration): A large blue button at the bottom.

그림 32. OurChord 회원 가입 화면
Fig 32. OurChord Application Member Register Screen

OurChord 의 튜닝 음 화면은 아래 그림 33과 같다. 사용자가 원하는 튜닝 음을 선택하여 클릭하면 해당 튜닝 음이 스피커에서 출력된다.



The screen displays a list of tuning options. Each option consists of a horizontal line followed by a black rectangular box. At the bottom, there is a navigation bar with four icons: a music note, a tuning fork, a speaker, and a person wearing headphones. The tuning fork icon is highlighted with a pink background.

그림 33. OurChord 튜닝 음 화면
Fig 33. OurChord Application Tuning Set Screen

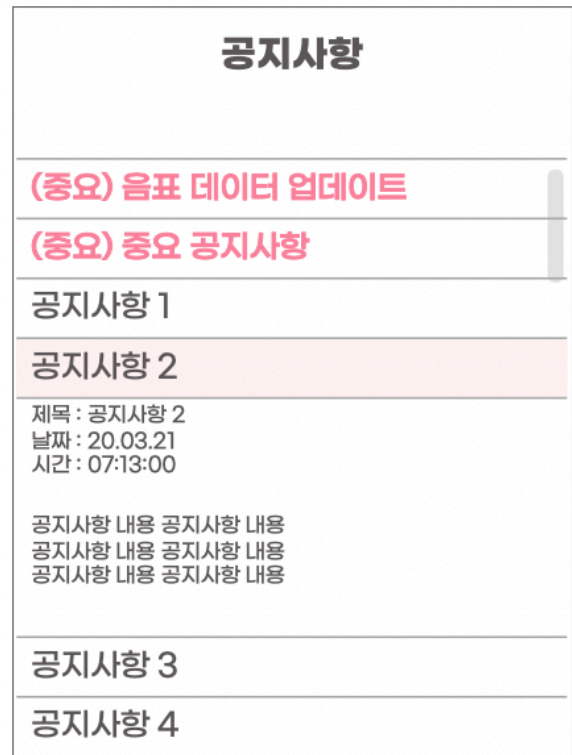
OurChord 의 개인 정보 수정 화면은 아래 그림 34와 같다. 이름, ID, E-MAIL 개인 정보를 수정한 뒤 수정 버튼을 클릭하면 개인 정보 수정이 완료된다.



The screen is titled '개인 정보 수정' (Personal Information Modification). It contains three input fields: '이름' (Name), 'ID', and 'E-MAIL'. Below these fields is a blue button labeled '수정' (Modify).

그림 34. OurChord 개인 정보 수정 화면
Fig 34. OurChord Application Individual Info Modify Screen

OurChord 의 공지사항 화면은 아래 그림 35 와 같다. 중요 공지사항은 앞에 '(중요)' 가 붙 으며, #F9688C 색으로 표시된다. 확인하고자 하는 공지사항 제목을 클릭하면 아코디언 메뉴 형식으로 공지사항의 상세 내용을 확인할 수 있다.



The screen is titled '공지사항' (Notice). It features a list of notices. The first two items are highlighted in pink and labeled '(중요) 음표 데이터 업데이트' and '(중요) 중요 공지사항'. Below these are four items labeled '공지사항 1', '공지사항 2', '공지사항 3', and '공지사항 4'. Item 2 is expanded, showing details: '제목: 공지사항 2', '날짜: 20.03.21', '시간: 07:13:00', and a list of '공지사항 내용' (Notice Content).

그림 35. OurChord 공지사항 화면
Fig 35. OurChord Application Notice Screen

OurChord 의 FAQ 화면은 아래 그림 36 과 같다. 공지사항과 마찬가지로 확인하려는 FAQ 를 클릭하면 아코디언 메뉴 형식으로 FAQ 의 답을 확인할 수 있다.



그림 36. OurChord FAQ 화면

Fig 36. OurChord Application FAQ Screen

2.5.4 프로토타입 테스트 및 분석

OurChord 프로토타입 테스트는 공식적인 악보 파일 10 개로 악보의 오선 추출, 음표의 좌표 추출 정확도, 음표 사진 추출 정확도, 음표 박자 인식 정확도를 확인하였다.

첫번째 악보 오선 추출 시 정확도는 아래 표 29 와 같다.

표 29. 오선 추출 정확도 테스트 결과

Table 29. Detect Staff Accuracy Test Result

	오선 수	정확히 뽑힌 수	정확도
악보1	50	50	100%
악보2	50	50	100%
악보3	30	30	100%
악보4	35	35	100%
악보5	25	25	100%
악보6	35	35	100%
악보7	35	35	100%
악보8	25	25	100%
악보9	60	60	100%

악보10	35	35	100%
평균 정확도			100%

두번째, 음표 좌표 추출 정확도는 아래 표 30 과 같다.

표 30. 음표 좌표 추출 정확도 테스트 결과

Table 30. Detect Note Coordinate Accuracy Test Result

	음표 수	정확히 뽑힌 수	정확도
악보1	89	89	100%
악보2	79	79	100%
악보3	344	344	100%
악보4	402	402	100%
악보5	104	104	100%
악보6	227	227	100%
악보7	123	123	100%
악보8	74	74	100%
악보9	60	60	100%
악보10	110	35	100%
평균 정확도			100%

세번째, 음표 사진 추출 정확도는 아래 표 31 과 같다.

표 31. 음표 사진 추출 정확도 테스트 결과

Table 31. Detect Note Photo Accuracy Test Result

	음표 수	정확히 뽑힌 수	정확도
악보1	89	88	98%
악보2	79	78	98%
악보3	344	322	93%
악보4	402	360	89%
악보5	104	96	92%
악보6	227	219	96%
악보7	123	115	93%
악보8	74	73	98%
악보9	60	60	100%
악보10	110	105	95%
평균 정확도			95%

마지막, 박자 인식 정확도는 아래 표 32 와 같다. 박자 인식 정확도 그래프 그림 37 과 같다.

표 32. 박자 인식 정확도 테스트 결과

Table 32. Recognition Tempo Accuracy Test Result

	음표 수	정확히 뽑힌 수	정확도
악보1	89	70	79%
악보2	79	70	89%
악보3	344	302	88%
악보4	402	321	80%
악보5	104	89	86%
악보6	227	201	89%
악보7	123	98	80%
악보8	74	65	88%
악보9	60	60	100%
악보10	110	100	91%
평균 정확도			85%

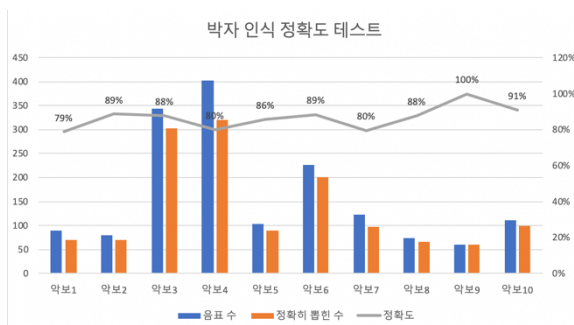


그림 37. 박자 인식 정확도 테스트 결과 그래프
Fig 37. Recognition Tempo Accuracy Test Graph

음표의 좌표는 추출 시 100% 정확도를 갖는다. 하지만 음표를 위의 2.4 상세설계 그림 2 분음표 예시에 맞추어 구분하여 음표 사진을 추출 시 평균 정확도가 95%로 떨어졌다. 추출된 음표 사진으로 박자를 인식하기 때문에 박자 인식 정확도가 85%로 더 낮아진 것을 확인하였다.

따라서 정확한 조옮김을 위해 음표 사진 추출 정확도를 더 높여야 한다. 그리고 현재 박자 인식 모델 자체의 정확도는 92.7% 이다. 연구 개발 시 음표 인식 정확도 목표는 90% 이상 이다. 프로토타입 테스트 결과 연구 개발 목표인 90% 이상을 달성하지 못했다. 90% 이상 달성을 위해 음표 사진 추출 시 정확도와 박자 인식 모델 정확도를 더 높일 것이다.

음표 사진 추출 정확도를 높이기 위해 정확히 뽑히지 않는 그림 38, 39 와 같은 음표들도

정확히 뽑힐 수 있도록 해결책을 찾을 것이다.



그림 38. 예외적인 음표 예시 1
Fig 38. Exception Note Example 1



그림 39. 예외적인 음표 예시 2
Fig 39. Exception Note Example 2

딥러닝 모델의 정확도를 높이기 위해 음표 데이터를 더 수집하고, 모델의 레이어를 더 추가하여 박자 별 음표의 특징을 더 잘 잡아내는 모델을 생성할 것이다.

2.6 시험/테스트 결과

프로토타입 이 후 계속해서 개발을 진행한 결과 현재 구현 현황은 아래 표 33 과 같다. 음계 추출 시 기존에는 잇단 음표와 화음 음표는 인식 불가능하였다. 하지만 추가로 잇단 음표와 화음 음표도 구분할 수 있도록 구현하여 음표 좌표 추출, 음계 추출을 완벽히 완성하였다. 따라서 모듈 별 98 % 구현 완료하였다.

표 33. 프로토타입 이후 전체 구현 현황

Table 33. After Prototype OurChord Process Status

	구현 현황(%)
PDF to PNG	100%
오선 좌표 추출	100%
음표 좌표 추출	100%
음표 좌표 정렬	100%
음표 사진 추출	100%
음계 추출	100%
딥러닝 모델 학습 모듈	100%
박자 인식 정확도	89%
MIDI 데이터 변환	100%
튜닝 음 출력	100%
조 변환 모듈	100%
앱과 서버 통신 모듈	100%
서버 가상환경 구축	100%
앱 화면 구현	90%
서버와 DB 연동	100%
DB 구성	100%
98%	

2.6.1 음표 인식

음표 인식 테스트를 위해 다음 그림 40, 41 과 같이 확인 하였다. 인식한 음표에 사각형을 표시하여 음표가 정확히 추출되는지 확인하였다.

프로토타입 이전에는 잇단 음표는 인식하지 못하였지만, 프로토타입 이후 잇단음표와 화음 음표도 인식가능함을 확인할 수 있다. 프로토타입 이 후 음표 인식 정확도는 아래 표 34 로 정리하였다.



그림 40. 음표 인식 테스트 예시

Fig 40. Example of Score Recognition test



그림 41. 프로토타입 이후 추가 음표 인식 확인

Fig 41. After Prototype Extra Score Recognition Check

표 34. 프로토타입 이후 음표 인식 정확도

Table 34. After Prototype Detect Note Coordinate Accuracy

	음표 수	정확히 뽑힌 수	정확도
악보1	88	88	100%
악보2	125	123	98%
악보3	79	79	100%
악보4	123	123	100%
악보5	148	148	100%
악보6	84	84	100%
악보7	103	103	100%
악보8	80	80	100%
악보9	69	69	100%
악보10	117	117	100%
평균 정확도			99.8%

2.6.2 박자 인식 테스트

표 35. 프로토타입 이후 박자 인식 정확도

Table 35. After Prototype Tempo Coordinate Accuracy

	음표 수	정확히 뽑힌 수	정확도
악보1	123	100	81.3%
악보2	148	130	87.8%
악보3	88	83	94.3%
악보4	84	79	94%
악보5	103	98	95.1%
악보6	69	55	79.7%
악보7	56	51	91%
악보8	117	112	95.7%
악보9	59	49	83%
악보10	86	76	88.3%
평균 정확도			89%

템플릿 매칭을 통해 잇단 음표와 화음 음표

의 구분이 가능해져 잇단 음표의 8, 16분 음표의 박자 구분을 추가하여 박자 인식을 위한 딥러닝 모델을 새롭게 구현하였다. 이를 통해 박자 인식 모델 정확도를 더 높였다.

프로토타입 이전과 이후 박자, 음표 인식 정확도를 아래 그림 42 로 비교하였다. 프로토타입 이전 박자 정확도는 85 %,이 후 89 %로 증가하였고, 음표 인식 정확도는 95 % 에서 이후, 99.8 % 로 증가함을 확인할 수 있다.

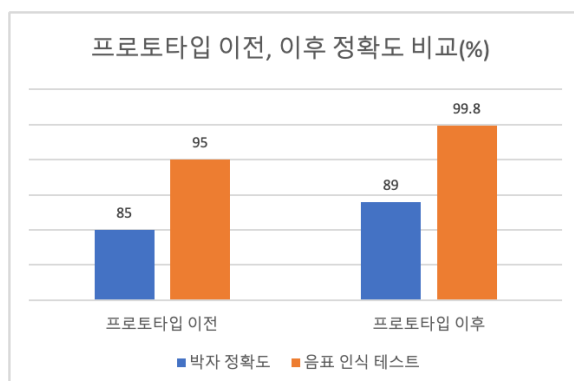


그림 42. 프로토타입 이후, 이전 정확도 비교
Fig 42. Compare Prototype Before After Accuracy

2.6.3 서버와 앱 연동 테스트

서버와 앱과의 연동 구현 현황을 아래 표 36 로 정리하였다. 서버와 앱 연동 테스트는 서버에서 앱으로부터 받은 문자열을 통해 앱의 화면을 구분하여 화면 별 모듈에 맞추어 수행하는지 확인하였다. 화면 구분을 위한 문자열은 앱의 레이아웃명의 문자열을 전송하였다. 예를 들어, 로그인 화면의 문자열은 'login - id - pwd' 와 같다.

서버에서 앱의 화면 별 기능 수행을 위해, 앱의 화면 별 모듈 이름을 딴 문자열을 통해 구분하였다. 예를 들어, 로그인 화면 부분의 문자열은 'LOGIN-ID-PWD' 'login-flottante-flot234' 과 같다. 서버에서는 앱으로부터 받은 정보를 '-' 로 구분하여 배열에 저장하고, 저장된 데이

터를 통해 기능을 수행하도록 앱으로부터 받은 정보를 서버에서 'alldis'라는 변수에 저장된 문자열에 "-"로 구분하여 이를 순차적으로 'tdata' 배열에 저장하여, 저장된 데이터를 통해 해당 기능을 수행하였다. 서버와 앱 간의 연동이 문자열을 통해 앱에서 서버로, 서버에서 앱으로 보내는 정보를 확인할 수 있도록 구현을 완료하였다.

표 36. 서버와 앱 연동 구현 현황

Table 36. Link Server and application Process Status

	구현 현황(%)
앱에서 문자열 전송	100%
서버에서 문자열 확인	100%
로그인 정보	100%
회원가입 중복 확인 정보	100%
회원가입 인증 번호 전송	100%
회원가입 인증 확인 전송	100%
PDF 파일 전송	100%
조 변환 정보 전송	100%
MIDI 파일 전송	100%
ID 찾기 정보 전송	100%
비밀번호 찾기 정보 전송	100%
개인정보 수정 정보 전송	100%
100%	

첫번째, 앱에서 로그인 화면을 실행할 경우 사용자가 'LOGIN' 버튼을 클릭할 경우 앱은 서버로 'login - 사용자가 입력한 아이디 - 사용자가 입력한 비밀번호'를 전송하고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 데이터를 통해 SQL 문 실행 후, 기존에 등록된 회원으로 확인된 경우 문자 's' 를 앱으로 return 한다.

두번째, 앱에서 회원가입 화면 중, 사용자가 중복 확인(ID) 버튼을 클릭할 경우 앱은 서버로 'id_check - 사용자가 입력한 아이디'를 전송하고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 데이터를 통해 SQL 문 실행 후, 기존에 없는 ID 값으로 확인된 경우 문자 's' 를 앱으로 return 한다.

세번째, 앱에서 회원가입 화면 중, 인증코드 버튼 클릭할 경우 앱은 서버로 'Auth - 회원 가입하려는 사용자 이름 - 회원 가입하려는 사용자 아이디 - 회원 가입하려는 사용자 비밀번호 - 회원 가입하려는 사용자 이메일'을 전송하고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 회원가입 정보가 성공적으로 등록되면 문자 's' 를 앱으로 return 한다.

네번째, 앱에서 인증코드 확인 버튼을 클릭할 경우 앱은 서버로 'checkAuth - 실제 이메일로 보내진 인증코드 - 사용자가 입력한 인증코드'를 전송하고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 데이터와 현재 사용중인 사용자의 ID를 통해 SQL 문 실행 후, 두 인증코드가 일치하면 문자 's' 를 앱으로 return 한다.

다섯째, 앱에서 PDF 파일을 서버로 전송할 경우 앱은 서버로 'upload_folder - PDF 이름 - PDF 데이터'를 255 Bytes 단위로 끊어서 전송하고 서버에서는 해당 PDF 데이터를 사용자 ID 에 맞게 생성된 디렉토리 내에 PDF 파일을 생성 및 저장한다.

여섯째, 앱에서 악보 조 변환 화면을 실행할 경우 앱은 서버로 'conversion'을 전송하고 서버에서는 현재 사용중인 사용자의 ID와 함께 해당 데이터를 배열로 저장한다.

일곱째, 앱에서 MIDI파일 확인 화면을 실행할 경우 앱은 서버로 'midiupload_folder'를 전송하고 서버에서는 현재 사용중인 사용자의 id 에 해당하는 폴더 아래에 있는 모든 MIDI파일을 파일 길이가 끝날 때까지 전송한다. MIDI파일은 사용자가 해당 화면에 접속할 때마다 처음부터 모든 파일을 다시 전송한다.

여덟째, 앱에서 아이디 찾기 화면을 실행할 경우 앱은 서버로 'find_id - 사용자가 입력한 이름 - 사용자가 입력한 이메일'을 보내주고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 데이터를 통해 SQL 문 실행 후, 일치하면 ID 찾기가 성공했다는 문자 's' 와 찾은 ID 값을 앱으로 return 한다.

아홉째, 앱에서 비밀번호 찾기 화면을 실행할 경우 앱은 서버로 'find_pwd - 사용자가 입력한 ID - 사용자가 입력한 이메일'을 보내주고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 데이터를 통해 SQL 문 실행 후, 일치하면 PWD 찾기가 성공했다는 문자 's' 와 찾은 PWD 값을 앱으로 return 한다.

마지막, 앱에서 개인정보 수정 화면을 실행할 경우 앱은 서버로 'my - 사용자가 새로 수정하고자 하는 이름 - 사용자가 새로 수정하고자 하는 아이디 - 사용자가 새로 수정하고자 하는 이메일'을 보내주고 서버에서는 해당 데이터를 배열로 저장한다. 배열에 저장된 데이터와 현재 사용중인 사용자의 ID 를 통해 SQL 문 실행 후, 일치하면 개인정보 수정이 성공했다는 문자 's' 를 앱으로 return 한다.

2.6.4 서버와 데이터베이스 연동 테스트

2.6.3 과 마찬가지로 서버와 데이터베이스 연동 테스트를 진행하였다.

서버와 데이터베이스의 연동 구현 현황을 아래 표 37 로 정리하였다. 앱의 화면에 따라 서버는 필요한 데이터베이스에 접근 시에 서버에서 해당하는 SQL 문 실행 가능 여부를 확인 및 저장하고 확인하도록 구현을 완료했다.

표 37. 서버와 데이터베이스 연동 구현 현황

Table 37. Link Server and database Process Status

	구현 현황(%)
--	----------

SQL 문 실행 확인	100%
로그인 정보 DB 저장	100%
ID 중복 DB 확인	100%
인증코드 DB 저장	100%
인증코드 DB 확인	100%
PDF DB 경로 확인	100%
조변환 정보 DB 확인	100%
ID 찾기 DB 확인	100%
비밀번호 찾기 DB 확인	100%
정보 수정 DB 업데이트	100%
100%	

첫번째, 서버에 배열로 저장된 로그인 정보가 USER 테이블에 저장된 사용자 정보와 동일한 정보가 존재한다면, 서버에서는 '정보 있음'이라는 출력과 함께 's' 라는 문자를 return 하게 되고, 정보가 존재하지 않을 경우, '정보 없음'이라는 출력과 함께 'f' 라는 문자를 return 하게 된다.

두번째, 앱에서 회원가입 화면 중, ID 중복 확인 버튼을 클릭할 경우 USER 테이블에 저장된 사용자 정보와 동일한 ID값이 존재한다면, 서버에서는 'ID 중복o'이라는 출력과 함께 'f' 라는 문자를 return 하게 되고, 정보가 존재하지 않을 경우, 'ID 중복x'이라는 출력과 함께 's' 라는 문자를 return 하게 된다.

세번째, 서버에 배열로 저장된 회원가입 중인 사용자의 정보를 USER 테이블에서 인증코드는 빈칸으로 두고, 나머지 정보를 INSERT 성공하면 서버에서는 '정보 USER 테이블에 삽입 성공'이라는 출력과 함께 's' 라는 문자를 return 하게 되고, 실패할 경우에는 '정보 USER 테이블에 삽입 실패'라는 출력과 함께 'f' 라는 문자를 return 하게 된다.

네번째, 앱에서 인증코드 확인 버튼을 클릭할 경우 먼저, USER 테이블에서 사용자가 실제로 이메일로 발급받은 인증코드를 UPDATE 한다. 이후에 현재 사용자의 실제 발급받은 인증코

드와 사용자가 입력한 인증코드가 일치하는 데이터가 EXISTS 하면 서버에서는 '인증 성공'이라는 출력과 함께 's' 라는 문자를 return 하게 되고, 실패할 경우에는 '인증 실패'라는 출력과 함께 'f'라는 문자를 return 하게 된다.

다섯째, 앱에서 악보 조 변환 화면을 실행할 경우 USER_SCORE_PDF 테이블에서 현재 사용 중인 사용자의 ID로 변환할 PDF파일의 경로를 SELECT한 결과값이 존재할 경우 'PDF 경로 찾기 성공'이라는 출력과 함께 's' 라는 문자를 return하게 되고, 실패할 경우에는 'PDF 경로 찾기 실패'라는 출력과 함께 'f' 라는 문자를 return하게 된다.

여섯째, 앱에서 아이디 찾기 화면을 실행할 경우 USER 테이블에서 사용자에게 입력 받은 사용자 이름과 이메일을 통해 ID를 SELECT한 결과값이 존재할 경우 'ID 찾기 성공'이라는 출력과 함께 's' 라는 문자를 return 하게 되고, 실패할 경우에는 'ID 찾기 실패'라는 출력과 함께 'f' 라는 문자를 return 하게 된다.

일곱째, 앱에서 비밀번호 찾기 화면을 실행할 경우 USER 테이블에서 사용자에게 입력 받은 사용자 아이디와 이메일을 통해 PWD를 SELECT한 결과값이 존재할 경우 'PWD 찾기 성공'이라는 출력과 함께 's' 라는 문자를 return 하게 되고, 실패할 경우에는 'PWD 찾기 실패'라는 출력과 함께 'f' 라는 문자를 return 하게 된다.

마지막, 앱에서 개인정보 수정 화면을 실행할 경우 USER 테이블에서 수정된 사용자 이름, ID, EMAIL을 UPDATE 성공하면 서버에서는 'USER 정보 수정 성공'이라는 출력과 함께 's' 라는 문자를 return 하게 되고, 실패할 경우에는 'USER 정보 수정 실패'이라는 출력과 함께 'f'라는 문자를 return 하게 된다.

2.7 Coding & Demo

프로토타입 이후 최종 데모 OurChord 프로그램의 소프트웨어 구성도는 아래 그림 43 과 같다.

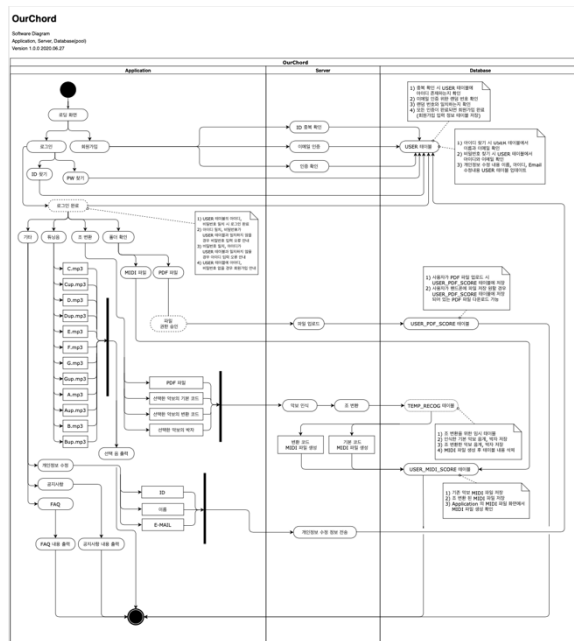


그림 43. OurChord 전체 소프트웨어 구성도

Fig 43. OurChord Software Diagram

전체 소프트웨어 구성도는 앱, 서버, 데이터베이스 관점에서 전체 구성을 확인할 수 있도록 작성하였다.

2.7.1 Algorithm

OurChord 프로그램의 알고리즘 순서도는 아래 그림 44 와 같다.

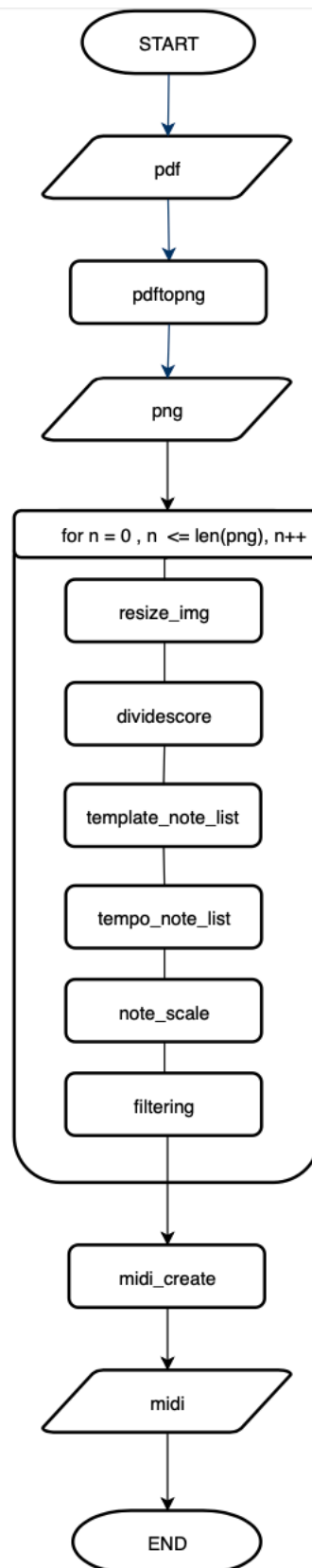


그림 44. OurChord 전체 알고리즘 구성도

Fig 44. OurChord Algorithm Diagram

악보 PDF 파일을 입력하면 악보 인식을 위해

pdftopng 함수를 통해 PDF 파일을 PNG 파일로 변경한다. 이 후 생성된 악보 PNG 이미지의 수에 따라 템플릿 매칭 정확도를 높이기 위해 resize_img 함수를 통해 리사이즈한다. dividescore 함수를 통해 악보 전체에서 하나의 오선 별로 이미지를 구분한다.

악보 전체에서 하나의 오선으로 구분한 오선 이미지에서 template_note_list 함수를 통해 음계 인식을 위한 음표의 좌표를 추출한다. tempo_note_list 함수를 통해 음표의 박자를 인식한다. 이 후 filtering 함수를 통해 중복 좌표와 잇단 음표, 화음 음표를 추출하여 음계와 박자 추출 정확도를 높인다.

filtering 함수를 통해 얻은 정확한 음표의 음계와 박자를 사용하여 create_midi 함수를 통해 MIDI 파일을 생성한다.

2.7.2 Server & Database

OurChord 프로그램에서 서버는 pymysql 라이브러리의 pymysql.connect() 를 통해 RDS에 존재하는 "OURCHORD" 데이터베이스에 접근한다. Cursor 변수를 선언하여 쿼리문에 의해 반환되는 결과를 저장하는 메모리 공간을 할당하였다.

로그인 화면 실행 시, 'SELECT EXISTS(SELECT *FROM USER WHERE ID=%s AND PWD=%s)' 쿼리문을 실행하여 기존에 가입한 사용자 정보를 확인한다. 기존 테이블 구성인 아래 그림 45 과 같다.

USERNAME	ID	PWD	EMAIL	AUTH
user1			email	
*	111	0530	who@gmail.com	NULL
김민지	flottante	1234	kkky5833@naver.com	NULL
user2	flottante2	1234	email	
user2	flottante3	1234	email	
한글	HANGUL	1234	hangul@naver.com	NULL
user0622	id0622	1234	today0622@today.com	
user2	id1	1234	email	
user2	id11	1234	email	
user1	id2	1234	email	
조원희	MINJIZANG	0406	chowond@daum.net	
오늘	today	5678	today@naver.com	NULL
user7	uu1	622	u@kpu.ac.kr	
user6	uuu	u0622	u@kpu.ac.kr	
임영규	yklm1	1234	yklm1@naver.com	NULL
문지수	zjisuo0	1234	moonjs96@naver.com	NULL

그림 45. USER 테이블 구성

Fig 45. USER table configuration

회원가입 화면에서 아이디 중복 체크 실행 시, 'SELECT EXISTS(SELECT ID FROM USER WHERE ID=%s)' 쿼리문을 실행하여 기존에 가입한 사용자 ID와의 중복을 확인한다.

회원가입 화면에서 사용자 정보 추가 시, 'INSERT INTO USER(USERNAME, ID, PWD, EMAIL, AUTH) values(%s, %s, %s, %s, %s)' 쿼리문을 실행하여 새로 회원 가입한 사용자의 정보를 추가한다.

기존 테이블 구성인 위의 그림 45 와 같다. 새로운 사용자 정보 삽입이 성공할 경우 업데이트 된 USER 테이블 정보는 아래 그림 46 과 같다.

USERNAME	ID	PWD	EMAIL	AUTH
user1			email	
*	111	0530	who@gmail.com	NULL
newnew	flot	1234	new@new.com	7894
김민지	flottante	1234	kkky5833@naver.com	NULL
user2	flottante2	1234	email	
user2	flottante3	1234	email	
한글	HANGUL	1234	hangul@naver.com	NULL
user0622	id0622	1234	today0622@today.com	
user2	id1	1234	email	
user2	id11	1234	email	
user1	id2	1234	email	
조원희	MINJIZANG	0406	chowond@daum.net	
오늘	today	5678	today@naver.com	NULL
user7	uu1	622	u@kpu.ac.kr	
user6	uuu	u0622	u@kpu.ac.kr	
임영규	yklm1	1234	yklm1@naver.com	NULL
문지수	zjisuo0	1234	moonjs96@naver.com	NULL

17 rows in set (0.00 sec)

그림 46. 업데이트 된 USER 테이블

Fig 46. Updated USER Table

PDF업로드 화면에서 사용자가 저장한 PDF 정보 추가 시, 'SELECT EXISTS(SELECT *FROM USER_SCORE_PDF WHERE PDF_ID=%s AND PDF_NAME=%s AND PDF_PATH=%s)' 쿼리문을 실행하여 기존에 저장된 PDF 정보인지 확인 후, 기존에 저장된 PDF 정보가 아니라면, 'INSERT INTO USER_SCORE_PDF(PDF_ID, PDF_NAME, PDF_PATH) values(%s, %s, %s)' 쿼리문을 실행하여 새로 저장한 사용자의 PDF 정보를 추가한다.

기존 테이블 구성인 아래 그림 47 과 같다. 새로운 PDF 정보 삽입이 성공할 경우 업데이트 된 USER_SCORE_PDF 테이블 정보는 아래 그림 48 와 같다.

```
mysql> select * from USER_SCORE_PDF;
```

PDF_ID	PDF_NAME	PDF_PATH
zjisuoo	test.pdf	./home

1 row in set (0.01 sec)

그림 47. USER_SCORE_PDF 테이블 구성
Fig 47. USER_SCORE_PDF table configuration

```
mysql> select * from USER_SCORE_PDF;
```

PDF_ID	PDF_NAME	PDF_PATH
zjisuoo	test.pdf	./home
flottante	4.pdf	/home/ec2-user/Ourchord/USER/flottante/

2 rows in set (0.00 sec)

그림48. 업데이트 된 USER_SCORE_PDF 테이블
Fig 48. Updated USER_SCORE_PDF Table

MIDI 파일 확인 화면에서 사용자가 저장한 MIDI 정보 추가 시, 'SELECT EXISTS(SELECT *FROM USER_SCORE_MIDI(MIDI_ID, BASEMIDI_NAME, BASEMIDI_PATH, CONVERSIONMIDI_NAME, CONVERSIONMIDI_PATH) values(%s, %s, %s, %s, %s)' 쿼리문을 실행하여 기존에 저장된 MIDI 정보인지 확인 후, 기존에 저장된 MIDI 정보가 아니라면, 'INSERT INTO

USER_SCORE_MIDI(MIDI_ID, BASEMIDI_NAME, BASEMIDI_PATH, CONVERSIONMIDI_NAME, CONVERSIONMIDI_PATH)

values(%s, %s, %s, %s, %s)' 쿼리문을 실행하여 새로 저장한 사용자의 MIDI 정보를 추가한다.

기존 테이블 구성인 아래 그림 49 와 같다. 새로운 MIDI 정보 삽입이 성공할 경우 업데이트 된 USER_SCORE_MIDI 테이블 정보는 아래 그림 50 와 같다

```
mysql> select * from USER_SCORE_MIDI;
```

MIDI_ID	BASEMIDI_NAME	BASEMIDI_PATH	CONVERSIONMIDI_NAME	CONVERSIONMIDI_PATH
zjisuoo	B_12.mid	./home	C_12.mid	./home

1 row in set (0.00 sec)

그림 49. USER_SCORE_MIDI 테이블 구성
Fig 49. USER_SCORE_MIDI table configuration

```
mysql> mysql> select * from USER_SCORE_MIDI;
```

MIDI_ID	BASEMIDI_NAME	BASEMIDI_PATH	CONVERSIONMIDI_NAME	CONVERSIONMIDI_PATH
zjisuoo	B_12.mid	./home	C_12.mid	./home
flottante	BA_1.pdf	/home/ec2-user/Ourchord/USER/flottante/	CC_1.pdf	/home/ec2-user/Ourchord/USER/flottante/

2 rows in set (0.00 sec)

그림50. 업데이트 된 USER_SCORE_MIDI 테이블
Fig 50. Updated USER_SCORE_MIDI Table

아이디 찾기 화면에서, 'SELECT EXISTS(SELECT ID FROM USER WHERE USERNAME=%s AND EMAIL=%s)' 쿼리문을 실행하여 찾고자 하는 사용자 정보가 있는지 확인한다. 해당 정보를 만족하는 경우, 'SELECT ID FROM USER WHERE USERNAME=%s AND EMAIL=%s' 쿼리문을 실행한다.

비밀번호 찾기 화면에서, 'SELECT EXISTS(SELECT PWD FROM USER WHERE ID=%s AND EMAIL=%s)' 쿼리문을 실행하여 찾고자 하는 사용자 정보가 있는지 확인한다. 해당 정보를 만족하는 경우, 'SELECT PWD FROM USER WHERE ID=%s AND EMAIL=%s' 쿼리문을 실행한다.

사용자 정보 수정 화면에서, 'SELECT EXISTS(SELECT *FROM USER WHERE

`ID=%s)` 쿼리문을 실행하여 수정하고자 하는 사용자 ID와 현재 사용중인 사용자 ID가 일치하면, `'UPDATE USER SET USERNAME=%s, ID=%s, EMAIL=%s WHERE ID=%s'` 쿼리문으로 사용자가 입력한 업데이트 할 정보로 수정한다. 업데이트 완료 시, 업데이트 된 사용자 정보는 아래 그림 51 과 같다.

USERNAME	ID	PWD	EMAIL	AUTH
#	111	0530	who@gmail.com	NULL
newnew	flot	1234	new@new.com	7894
김민지	flottante	1234	kkky5833@naver.com	NULL
user2	flottante2	1234	email	
user2	flottante3	1234	email	
한글	HANGUL	1234	hangul@naver.com	NULL
user0622	id0622	1234	today062@today.com	
user2	idl	1234	email	
user2	idl1	1234	email	
user1	id2	1234	email	
조원희	MINJIZANG	0406	chowond@daum.net	
오늘	today	5678	today@naver.com	NULL
happy	user1		happy@hap.com	
user7	uu1	622	u@kpu.ac.kr	
user6	uuu	u0622	u@kpu.ac.kr	
임영규	yklml	1234	yklml@naver.com	NULL
문지수	zjisuo	1234	moonjs96@naver.com	NULL

그림 51. 업데이트 된 USER 테이블

Fig 51. Updated USER Table

2.7.3 Application

OurChord 프로그램 앱 파일 구성은 아래 표 38 과 같다.

표 38. OurChord 앱 파일 구성

Table 38. OurChord Application File Construct

Java	
login.java	
register.java	
register_complete.java	
loading.java	
find_id.java	
find_pw.java	
MainActivity.java	
folder.java	
upload_folder.java	
midupload_folder.java	
tune.java	
Conversion.java	
Setting.java	
Notice.java	
Faq.java	
My.java	
res	
layout	add.png
	arrow.png
	blackpad.png
	whitepad.png
	folder.png
	tune.png
	conversion.png
	setting.png
	ourchord_logo.png
	pre.png
	renew.png
	nlue_btn.xml
	pink_btn.xml
Font	gmarketbold.ttf
	Gmarketlight.ttf
	Gmarketmedium.ttf
layout	activity_login.xml
	activity_register.xml
	activity_register_complete.xml
	activity_loading.xml
	activity_find_id.xml

	activity_find_pw.xml
	activity_main.xml
	fragment_folder.xml
	fragment_upload_folder.xml
	fragment_midiupload_folder.xml
	fragment_tune.xml
	fragment_conversion.xml
	fragment_setting.xml
	activity_notice.xml
	activity_faq.xml
	activity_my.xml
menu	bottom_menu.xml
raw	c.mp3
	cup.mp3
	d.mp3
	dup.mp3
	e.mp3
	f.mp3
	fup.mp3
	g.mp3
	gup.mp3
	a.mp3
	aup.mp3
	b.mp3
	bup.mp3
values	basechord.xml
	changechord.xml
	colors.xml
	strings.xml
	styles.xml

OurChord 앱은 로딩 화면 (loading.java) 을 3초간 보여준 후 로그인 화면 (login.java) 으로 전환된다. 로그인 화면에서 회원가입, 아이디, 비밀번호 찾기 (register.java, find_id.java, find_pw.java) 화면으로 이동할 수 있다. 로그인 화면에서 사용자가 아이디와 비밀번호를 입력하면 서버에 connect_login 함수를 통해 'login - zjisuooid - zjisuoopassword' 를 전송한다.

회원가입 화면 (register.java) 에서 중복 확인 버튼(ID)을 클릭하면 connect_id_check 함수를 통해 'id_check - zjisuooid' 를 전송하여, 중복 확인 후 인증코드 버튼을 클릭하면

sendAuth_Email 함수를 통해 사용자가 입력한 이메일로 랜덤한 4자리 숫자를 'sendAuth_Email - ****' 와 같이 전송한다. 이후 사용자가 인증 코드를 입력 한 뒤 확인 버튼을 클릭하면 checkAuth 함수를 통해 'checkAuth - ****' 를 서버에 전송하여 사용자 인증을 완료한다. 모든 입력을 완료한 뒤 회원가입 버튼을 클릭하면 void_regis_confirm 함수를 통해 'register - name - id - pwd - email' 을 서버에 전송한다.

회원가입이 완료 되면 회원가입이 완료되었다는 화면 (register_complete.java) 으로 전환되고, 로그인 버튼을 클릭하면 로그인 화면으로 이동한다.

아이디 찾기 화면 (find_id.java) 에서 아이디 찾기 버튼을 클릭하면 void_find_id 함수를 통해 'find_id - name - email' 을 서버에 전송한다. 비밀번호 찾기 화면 (find_pw.java) 에서 비밀번호 찾기 버튼을 클릭하면 void_find_pw 함수를 통해 'find_pw - id - email' 을 서버에 전송한다.

로그인 화면에서 정확한 아이디와 비밀번호를 입력하면 메인 화면 (MainActivity.java) 로 이동한다. 메인 화면은 프래그먼트로 구성하였다. bottom_menu.xml 에 폴더, 튜닝, 코드 변환, 환경 설정 메뉴를 설정하였다.

폴더 화면 (folder.java) 은 PDF 파일 화면 (upload_folder.java) 과 MIDI 파일 화면 (midiupload_folder.java) 을 프래그먼트 내 프래그먼트로 나누어 구성하였다.

PDF 파일 화면을 처음 이동하였을 경우 파일 업로드를 위해 파일 접근 권한을 승인하도록 안내 메시지를 보인다. 업로드 버튼을 클릭하면 모바일 기기 내의 파일에 접근하여 PDF 파

일을 앱에 업로드 하고, 서버에 void_pdf 함수를 통해 'upload_foler - filename' 을 전송한다.

MIDI 파일 화면은 서버에서 코드 변환하여 MIDI 파일을 앱으로 전송하면 MIDI 파일이 보여지도록 설정하였다.

튜닝 화면 (tune.java) 은 drawble 폴더의 whitepad.png, blackpad.png 를 통해 피아노 건반처럼 화면을 구성하였다. raw 폴더 내 각 코드(chord) 별 MP3 파일을 저장하여 피아노 건반의 코드에 맞는 MP3 파일을 soudpool 을 통해 MP3 가 출력된다.

코드 변환 화면 (conversion.java) 는 코드 변환하고자 하는 악보를 스피너를 통해 선택하고, 기존 코드와 변환하고자 하는 코드를 라디오 버튼을 통해 선택하도록 구성하였다. conversion 함수를 통해 'conversion - basechord - changechord - pdffilename' 을 서버에 전송한다.

설정 화면 (setting.java) 은 공지사항, FAQ, 개인 정보 수정 (notice.java, faq.java, my.java) 으로 구성하였다.

공지사항과 FAQ 는 아코디언 메뉴로 구성되어 제목을 클릭하면 제목에 해당하는 내용이 보여지도록 하였다. String 에 공지사항과 FAQ 의 제목, 내용을 입력하여 String ID 값에 따라 내용과 제목이 변경되도록 하였다. FAQ 는 변경되거나 업데이트 될 경우가 거의 없기 때문에 String 에 저장하는 것이 효율적이지만 공지사항은 주기적 업데이트가 될 수 있기 때문에 웹 뷰를 제작하여 서버의 DB 에 업로드 할 것이다.

개인 정보 수정은 사용자가 수정하고자 하는 정보를 수정한 뒤 수정 버튼을 클릭하면

connect_my 함수를 통해 'my - name - id - email' 을 서버에 전송한다.

2.7.5 조 변환 정의

조 변환을 위한 규칙은 아래 그림 52 와 같다. 아래 그림의 대문자는 조표에 따른 조 (key) 를 뜻한다. 음악에서 C, D, E, F, G, A, B 는 2 가지 의미를 갖는다.

첫번째, 조(chord) 를 뜻한다. 조는 음을 정리하고 질서 있게 하는 근본이 되는 조직, 구성이 구체적, 실제적으로 나타나는 상태를 뜻하는 것으로 단조, 장조가 있다. 조에 따라 음 체계가 달라지게 된다.

두번째, 음계를 뜻한다. C, D, E, F, G, A, B 는 각 음계 도, 레, 미, 파, 솔, 라, 시를 가리킨다. 각 조에 따라 중심음에 대하여 음계 (도, 레, 미, 파, 솔, 라, 시, 도) 질서를 유지하게 되는데 C 코드의 중심음은 '도', D 코드의 중심음 '레', E 코드의 중심음 '미', F 코드 중심음 '파', G 코드 중심음 '솔', A 코드 중심음 '라', B 코드 중심음 '시' 라는 규칙이 존재한다.

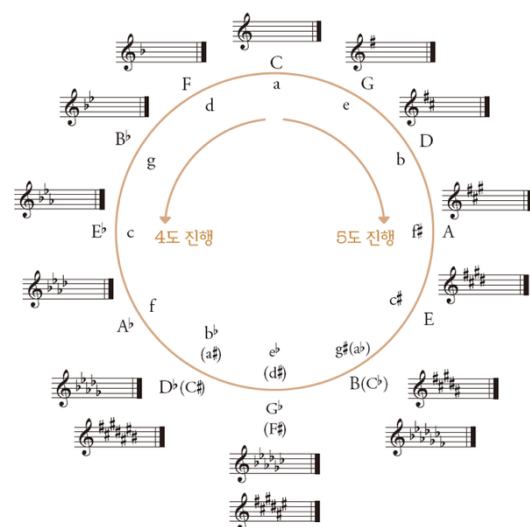


그림 52. 조 옮김 규칙

Fig 52. Rule of Chord Conversion

위의 그림 52 에서 확인할 수 있듯이 C 코드는 조표가 없고, G 코드는 파에 조표(반올림) 붙는다. 따라서 G 코드의 악보의 파 음계는 모두 조표(반올림) 이 적용 되어 반음 올려 악기를 연주해야 한다. G 코드의 경우 파에 조표한 개가 적용되지만, B 코드의 경우 파, 도, 솔, 레, 라, 미, 시 7개 음을 모두 반음 올려 연주해야 하므로, 악기를 연주하는데 어려움이 있다. 조표가 적은 C 코드나 D 코드로 조옮김하여 악기 연주를 보다 쉽게 할 수 있을 것이다.

조 옮김을 한다는 것은 중심음을 변경한다는 의미이다. 따라서 D 코드를 C 코드로 변경하는 것은 중심음을 '레' 에서 '도'로 변경하는 것이다. 따라서 D 코드의 음계 질서는 '레, 미, 파, 솔, 라, 시, 도, 레' 인데, 이것을 C 코드로 변환하면 음계 질서가 '도, 레, 미, 파, 솔, 라, 시, 도' 로 변경되는 것이다.

아래 그림 53 악보 예시를 통해 조 옮김을 실제로 적용해 볼 것이다. 첫번째 '아기 염소' 악보의 첫번째 오선이다. '아기 염소' 의 기존 코드는 F 코드이므로 중심음은 '파' 이다. 따라서 음계 질서는 '파, 솔, 라, 시, 도, 레, 미, 파' 가 된다. '아기 염소' 악보는 F 코드 이므로 시를 연주할 때 반음 (파에 조표가 붙음) 을 올려 연주해야 한다. 이를 C 코드로 변경하면 조표 없이 악기를 연주할 수 있다. C 코드의 중심음은 '도' 이기 때문에 '아기 염소' 악보의 '파' 는 '도', '솔' 은 '레', '라' 는 '미' 로 변경되어야 한다.



그림 53. 아기 염소 첫번째 오선 악보
Fig 53. '아기 염소' First 5 lines Sheet

위의 '아기 염소' 의 F 코드 음계는 '미, 솔, 라, 솔, 미, 솔, 솔, 라, 솔, 미, 솔, 파, 미, 레, 도, 시, 도' 이다. 이를 C 코드로 변환하면 음계는 '시, 레, 도, 레, 시, 레, 레, 레, 도, 레, 시, 레, 도, 시, 라, 솔, 파, 솔' 로 바뀌게 되어 파를 반올림하지 않고 (파에 조표 없어짐), 연주할 수 있다.

2.7.4 Demo

OurChord 프로그램의 개발 목표인 사용자가 원하는 조로 조 변환 된 MIDI 파일 생성의 정확도를 검증하기 위한 Demo 진행을 아래 표 39 로 정리하였다.

표 39. 데모 시나리오

Table 39. Demo Scenario

데모 시나리오	
1	로그인 확인
2	튜닝 음 출력 확인
3	PDF 파일 업로드 확인
4	MIDI 파일 생성 확인
5	조 변환 된 MIDI 파일 정확도 검증
6	공지사항, FAQ 확인

OurChord 앱에서 기존 사용자 데이터베이스에 등록된 사용자 ID와 비밀번호를 입력하여 로그인 하면, 해당 사용자 이름과 함께 "환영합니다." 라는 문구가 화면에 보여지면서 앱의 메인화면으로 이동한다.

메인 하단 튜닝음 화면에서 피아노 건반에서 듣고자 하는 음을 클릭하면 해당 음이 출력된다.

핸드폰 내부에 있는 PDF 파일을 OurChord 앱으로 불러오면 앱 화면에 '*pdfname1.pdf*', '*pdfname2.pdf*'와 같이 앱에 업로드 된 PDF 파일이 리스트 형식으로 보여진다. 앱에 업로드 된 PDF 파일 하나를 선택하고, 악보의 기존 '조' 와 변환할 '조' 를 선택한 후, 'CHORD

CONVERSION' 버튼을 클릭하면, MIDI 파일 확인 화면에서 '조옮김 악보'부분에 기존 '조' 악보에 해당하는 MIDI 파일과 변환된 '조'에 해당하는 MIDI 파일이 생성된 것을 확인한다.

생성 된 MIDI 파일의 조 변환이 정확하게 이루어진 것인지에 대한 검증을 위해, 'A' 조인 기존 악보의 MIDI 파일인 아래 그림 54 와 'B' 조로 변환된 MIDI 파일인 그림 55 를 비교하였다. 'A' 조에서 'B' 조로 변환 시, 중심음은 '라' 에서 '시' 로 변경된다. 아래 그림 54 에서 'A' 조 음계 '레, 도, 시, 라, 솔, 라, 도, 미' 가 그림 55에서 'B' 조 음계 '레, 도, 시, 라, 솔, 라, 도, 미(B 조에서 A 조로 변경 시 A# -> B 2음 올라가기 때문에 '도'와 '파'에 반올림 조표 추가)' 로 변경 된 것을 확인하였다. 이를 통해 오류 없이 조 변환 된 것을 확인하였다.



그림 54. 기존 'A'조 MIDI 파일
Fig 54. Base 'A' Chord MIDI File



그림 55. 변환 'B'조 MIDI 파일
Fig 55. Change 'B' Chord MIDI File

공지사항과 FAQ 화면에서는 중요 공지사항과 일반 공지사항과 FAQ를 리스트 형식으로 확인할 수 있다.

2.7.4.1 Demo 영상

<https://github.com/yklim1/Ourchord/DEMO>

참고 자료

References

- [1] 장경식, 박용순, 김희곤, 김인한, "악보 인식 시스템 및 이를 이용한 악보 인식 방법", 한국 기술 교육 대학교 산학 협력단, 2010.
- [2] 한우리, 이용환, 박제호, 김영섭, "얼굴 인식 통한 동적 감정 분류, Dynamic Emotion Classification through Facial Recognition", 반도체 디스플레이 기술 학회지 제 12 권 제 3호, 2013.9
- [3] 함승용. "eMusic Converting Solution 기술" 정보통신연구진흥원(Institute for Information Technology Advancement), 2002
- [4] 오타 미즈히사, 수도 코다이, 쿠로사와 타쿠마, 오다 다이ске, "실전! 딥러닝", 손민규, 위키북스(2019)
- [5] 박혜선, "Do it 정직하게 코딩하며 배우는 딥러닝 입문", 이지퍼블리싱(2019)
- [6] 이권진, "'그림 피아노'만 있으면 왕 초보도 쇼팽 부럽지 않죠", 중소기업 뉴스, 2019.12.9
- [7] 박인혜, 조성호, 박은진, "[Leisure] 엘프, 28년 동안 악기 개발에 올인" 매일경제, 2016.1.13
- [8] TensorFlow Datasets, "Tensorflow", <https://www.tensorflow.org/datasets/overview>, (2020.1.3)
- [9] NAVER CLOUD PLATFORM – AI:Application Service/TensorFlow/TensorFlow MNIST 예제, "Mnist", <https://docs.ncloud.com/ko/tensorflow/tensorflow-1-3.html>, (2020.1.3)
- [10] Musecore, "musecore", <https://github.com/muscore/MuseScore/blob/master/omr/README.md>, (2019.12.11)
- [11] 솔라리스의 인공지능 연구실 -11. 텐서플로우(TensorFlow)를 이용해서 Inception V3 모델 Retraining을 통해 나만의 데이터셋을 이미지 인식(추론) 해보기, "Tensorflow",

<http://solarisailab.com/archives/1422>,

(2020.1.3)

[12] 실제 이미지 데이터를 활용한 CNN모듈 구현하기,

"OpenCV" ,[http://www.birc.co.kr/2018/02/26/%EC%8B%A4%EC%A0%9C-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-](http://www.birc.co.kr/2018/02/26/%EC%8B%A4%EC%A0%9C-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-cnn-%EB%AA%A8%EB%8D%B8-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0/)

[cnn-%EB%AA%A8%EB%8D%B8-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0/](http://www.birc.co.kr/2018/02/26/%EC%8B%A4%EC%A0%9C-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-cnn-%EB%AA%A8%EB%8D%B8-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0/),

(2019.12.29)

[13] [Object Detection] 2. R-CNN: 딥러닝을 이용한 첫 2-stage Detector, "R-CNN",

<https://nuggy875.tistory.com/21>, (2020.1.3)

[14] Caden CV, "OpenCV" ,

<https://github.com/afikanyati/cadenCV>,

(2019.12.29)

[15] OpenCV Python 강좌 - 모폴로지 연산

(Morphological Operations), "OpenCV",

<https://webnautes.tistory.com/1257>,

(2019.12.29)

[16] 잡동사니 탐구 - 참스터디 GodGo - [30

편] 템플릿 매칭2, "템플릿 매칭",

<https://m.blog.naver.com/PostView.nhn?blogId=samsjang&logNo=220576634778&proxyReferer=https%3A%2F%2Fwww.google.com%2F>,

(2019.12.27)

[17] MYSQL , "MYSQL 연동",

<https://dev.mysql.com/doc/refman/5.7/en/gis-linestring-property-functions.html>, (2020.03.29)

[18] 케라스 훑어보기, "CNN 모델 시각화",

<https://wikidocs.net/32105>, (2020.03.31)

[19] 생활코딩,"PHP와 MySQL의 연동과

SELECT",

<https://opentutorials.org/course/3167/19586>, (20.04.09)

[20] w3schools.com, "PHP 튜토리얼",

<https://translate.google.com/translate?hl=ko&s>

[l=en&u=https://www.w3schools.com/TAGS/default.ASP&prev=search](https://www.w3schools.com/TAGS/default.ASP&prev=search), (20.04.09)

[21] CODING FACTORY, "PHP/함수

/password_hash()/비밀번호 암호화하는 함수",

<https://www.codingfactory.net/11707>, (2019.04.

17)