**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** | |
| 2020.03.05 | |
| **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 |

**- 문서 구성 -**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **진행 단계** | 프로젝트  계획서 발표 | 중간 발표1  (2월) | 중간 발표2  (4월) | 학기말 발표  (6월) | 최종 발표  (10월) |
| **기본 양식** | 계획서 양식 | | | | |
| **포함되는**  **내용** | * 1. 서론(1~6)   2. 본론(1~3)   참고자료 | I.2 본론(1~4)  참고자료 | I.2 본론(1~5)  참고자료 | X | 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. 작품 선정 배경 및 필요성 …………………………………………...… 5
   2. 기존 연구/기술 동향 분석 ……………………………………………… 5
   3. 개발 목표 ………………………………………………………………………… 7
   4. 팀 역할 분담 …………………………………………………………………… 7
   5. 개발 일정 ………………………………………………………………………… 8
   6. 개발 환경 ………………………………………………………………………… 8
2. **본론**
   1. 개발 내용 …………………………………………………………………….….. 10
   2. 문제 및 해결 방안 ………………………………………………………….. 13
   3. 시스템 및 시험 시나리오 …………………………………………...….. 14
   4. 상세 설계 …………….………………………………………………………..… 15
   5. Prototype 구현 ……………………………………………………………..… 20
   6. 시험/테스트 결과
   7. Coding & Demo
3. **결론**
   1. 연구 결과
   2. 작품 제작 소요 재료 목록

**참고자료** ………………………………………………………………………………………… 21

1. **서론(Introduction)**
   1. **작품 선정 배경 및 필요성**

음악 전공자가 아닌 사람들은 조표(샾, 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. **기존 연구/기술 동향 분석**
     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 추가하여 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 제공하였다.

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

개체, 안테나이(가) 표시된 사진

자동 생성된 설명

그림 1. 스케일 악보 예시

Fig 1. Scale score example

악보바다는 피아노 악보, 기타, 베이스, 드럼, 건반 악보, 타브(TAB) 악보, 멜로디, 코드 악보, MR 및 동영상 악보를 다양한 코드로 유료 제공하는 웹페이지이다. 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 제공할 것이고, MIDI 파일을 제공하여 악보 수정을 편리하게 할 수 있도록 할 것이다. 편리한 UI를 통해 모든 사용자가 쉽게 사용할 수 있도록 할 것이다.

* + 1. 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 사용할 것이다.

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 향상시킬 수 있다.

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 활용한다.

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 2 를 통해 상호 연결된 텐서플로우 커뮤니티를 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 2. 상호 연결된 텐서플로우 커뮤니티

Fig 2. Interconnected TensorFlow Community

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

스크린샷이(가) 표시된 사진

자동 생성된 설명

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 4 를 통해 기계 학습에 대한 관심이 해를 거듭할 수록 늘어남을 알 수 있다.

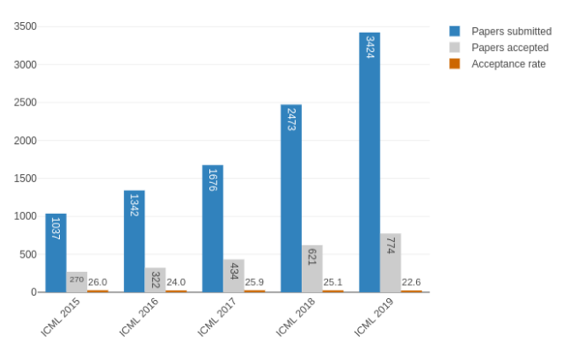


그림 4. 2015 ~ 2019 년도 국제 머신 러닝 학회의 논문 제출 수

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

* 1. **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 인식률 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. **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. **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 | 50% |
| 앱 구성도 제작 | 100% |
| 앱 구현 | 10% |
| 서버, DB 연결 | 100% |
| 앱, 서버 연결 | 0% |
| 유지보수 | 유지보수 | 2020년  4월~6월 예정 |
| 최종 검토 및 발표 | 최종 확인 | 2020년  7월 예정 |

악보 오선 인식, 악보 음계, 박자 인식, MIDI파일 변환 각각의 시스템 모듈 구현을 완료하였다. 이 후 통합을 진행하고 있다. 악보 이미지 영상 처리와 영상 처리한 결과를 통해 딥러닝(Deep Learning) 모델을 사용하도록 악보 영상처리와, 딥러닝 2가지로 나누어 통합 중이다.

앱 구성도 제작을 완료하여 앱 구현 진행 중이다. 서버에 데이터베이스(Database)를 구축 완료하였다. 하지만 시스템 모듈 구현을 파이썬(Python)으로 하였는데, 서버에서 파이썬 코드를 실행하는 것에 어려움이 있어 앱과 서버를 아직 연결하지 않았다. 계속해서 공부하여 파이썬 코드를 실행한 후 앱과 서버를 연결할 것이다.

* 1. **개발 환경**
     1. 개발 언어

OurChord 개발을 위해 영상처리와 딥러닝, MIDI 파일 변환 개발을 위해 Visual Studio Code 1.31 을 사용할 것이고, 사용자 인터페이스 구현을 위해 Java 를 사용 한다. 서버 구현을 위해 PHP 5.2.12을 사용한다.

* + 1. 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 에서 사용자 정보, 라벨링 된 음표 데이터, 사용자로부터 입력 받은 악보 데이터를 모두 저장하고, AWS(Amazon Web Server)의 기계 학습 훈련 시간을 단축시켜 줄 수 있는 p3.8xlarge 를 사용하여 데이터베이스에서 악보 파일 데이터를 송수신 한다. (상세 모듈 통합 전까지 phpMyAdmin 을 사용하여 데이터베이스를 연동하고, 기계 학습을 훈련 한다. 상세 모듈이 통합된 후 AWS(Amazon Web Server)를 사용한다.)

-> 수정 사항 있으면 아예 지우고 새로 적기

* + 1. 주요 라이브러리

악보 영상 처리를 위해 OpenCV 라이브러리를 사용한다. OpenCV 라이브러리는 이진화(Binarization), 노이즈 제거(Noise Reduction), 외곽선 검출(Edge Detection), 패턴 인식, 기계 학습(Machine Learning), ROI(Region Of Interest) 설정, 이미지 변환(Image Warping) 등의 알고리즘이 존재한다. 악보 영상 처리를 위한 알고리즘을 이용하여 음표 인식을 위한 악보 이미지 전 처리한다.

음표 인식을 위한 딥러닝 분산 처리를 위해 텐서플로우(TensorFlow) 라이브러리를 사용한다. 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 딥러닝에 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 영상처리와 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.

1. **본론(Point)**
   1. **개발 내용**

OurChord 개발을 위한 시스템 구조는 아래의 그림 5 와 같다.

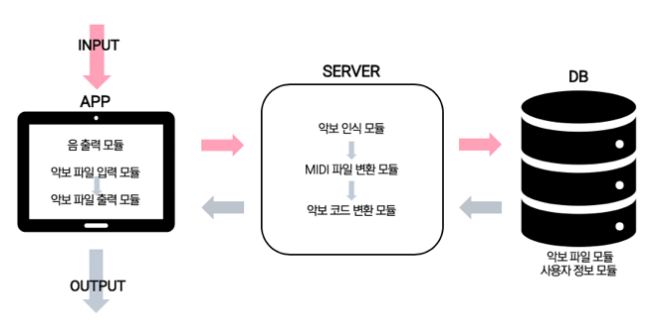


그림 5. 시스템 구성도

Fig 5. System diagram

* + 1. 악보 인식 모듈

악보의 음표 정보를 추출하기 위해 사용자가 PDF 악보 파일을 업로드 하면 그 파일을 OpenCV 라이브러리를 활용하기 위하여 PNG 파일로 변환 시킬 것이다. 이 후 음표의 음계와 박자 데이터들을 PNG 파일로 변환한 이미지 파일에서 얻을 것이다.

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 뽑힐 수 있어 1~2픽셀 정도만 차이 나는 좌표는 제거 한다.

오선의 좌표를 뽑으면 해당 오선들의 gap을 구하여 평균 gap을 구한다. 그리고 OurChord에서 탬플릿 매칭하는 이미지의 오선 gap(define 설정 값)과 비교를 하여 얼마나 차이가 나는지 비율을 알아내고 해당 이미지를 탬플릿 매칭에 알맞게 resize 한다.

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

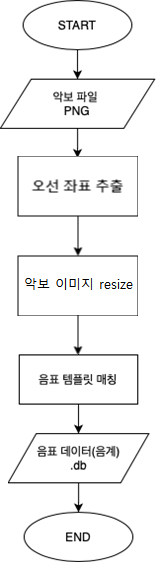


그림 6. 템플릿 매칭 통한 음계 인식 모듈

Fig 6. Melody Recognition module by Template Matching

위의 그림 6과 같이 탬플릿 매칭의 정확도를 올리기 위하여 탬플릿 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.

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

텍스트이(가) 표시된 사진

자동 생성된 설명

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 저장 된 음표 PNG 를 CNN 모델에서 반복 훈련 시킬 것이다. Conv2D, MaxPooling, Dense, Flatten 레이어 층을 추가하여 음표의 박자 인식하는데 최적화 된 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.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 8. 음표 박자 인식 모델 시각화

Fig 8. Model Visualization of Note Tempo Recognition

스크린샷이(가) 표시된 사진

자동 생성된 설명

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 레이어, MaxPooling2D 레이어, Flatten 레이어, 마지막 2 개의 Dense 레이어를 거친다는 것을 확인할 수 있다.

그림 9 에서 ‘?’는 CNN 모델에 박자를 분류할 음표 이미지 파일을 뜻한다.

그림 9 를 통해 Input 레이어의 입력 값이 224 X 224 픽셀, 입력 이미지 처리 필터 채널이 3 개임을 뜻하며, 출력 값은 입력과 마찬가지로 224 X 224 픽셀, 입력 이미지 채널 3개임을 뜻한다.

첫번째 Conv2D 레이어의 입력 값은 필터 크기 224 X 224, 필터의 갯수가 3 개임을 뜻하며, 출력 값은 필터 크기 222 X 222, 32 개의 필터임을 뜻한다. 두번째 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 갖음을 뜻한다.

Flatten 레이어의 입력 값은 필터 크기가 110 X 110, 64 개의 필터를 갖으며, 출력 값으로 1차원 필터 크기 774400 를 갖는다.

첫번째 Dense 레이어의 입력 값은 입력 뉴런의 수가 774400 개임을 뜻하며, 출력 뉴런의 수가 128 개 임을 뜻한다. 두번째 Dense 레이어의 입력 값으로 입력 뉴런의 수가 128 개이며, 출력 값으로 출력 뉴런 수가 4 개임을 확인할 수 있다.

* + 1. 악보 코드 변환 모듈

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

 개체이(가) 표시된 사진

자동 생성된 설명

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 Python의 pyknon 라이브러리를 사용하여 사용자가 원하는 조의 새로운 MIDI 파일을 제공할 것이다.

* + 1. MIDI 파일 변환 모듈

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

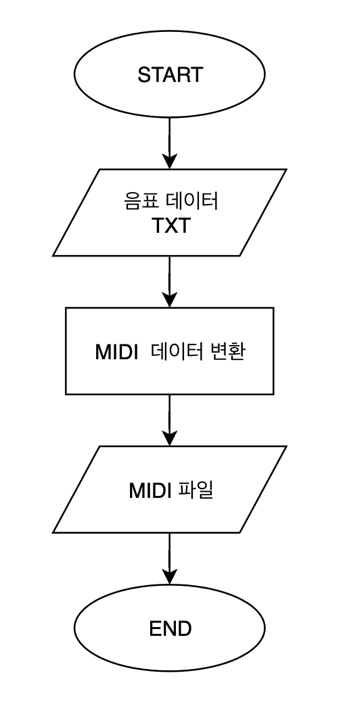


그림 11. MIDI 파일 변환 모듈 순서도

Fig 11. MIDI File Conversion Flow Chart

Python의 pyknon 라이브러리를 사용하여 MIDI 파일을 생성할 것이다. 추출한 음표 데이터를 lists로 입력 받은 후 입력 받은 음표 데이터 lists를 NoteSeq 함수를 사용하여 MIDI 데이터로 변환할 것이다.

* 1. **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 위한 해결방안은 먼저 Java 의 pdfview 라이브러리를 사용하여 PDF 파일을 PNG 파일로 변환하는 클래스를 생성하여 변환할 것이다. PNG 파일로 변환 후 파일 크기가 달라 졌는지 악보 이미지가 변한 부분이 있는지 확인(세부 크기)할 것이다.

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 추출하였다.

세번째, 음표 추출을 위해 CNN과 템플릿 매칭(Template Matching)을 이용해 정확한 음표 추출을 할 것이다. 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 할 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 12 음표 데이터 라벨링 예시

Fig 12. Note Data Labeling example

스크린샷이(가) 표시된 사진

자동 생성된 설명

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 학습시킬 것이다. 반복 된 학습을 통해 음표 인식률을 더 높일 것이다. 학습 시킨 데이터를 바탕으로 음표 인식 후 음표의 좌표를 추출하여 정확한 음표 박자 데이터를 얻을 것이다.

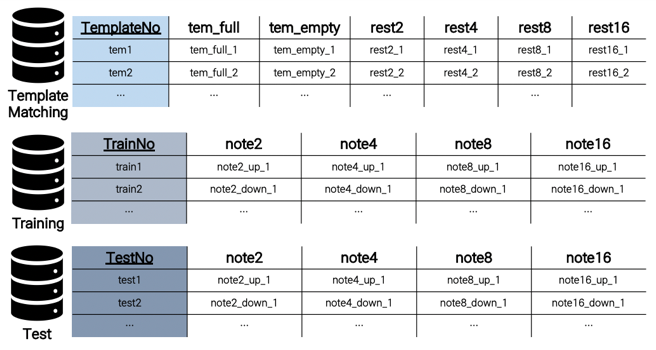


그림 14. 음표 데이터베이스 예시

Fig 14. Score Database example

템플릿 매칭을 사용하여 음표 추출 확률을 높이기 위해 많은 샘플 이미지를 음표 데이터베이스에 가지고 있을 것이다. 많은 음표 이미지를 통해 음표 추출 시 정확도를 올릴 것이다.

* 1. **시스템 및 시험 시나리오**
     1. 시스템 시나리오

OurChord 의 전체 시나리오는 아래 그림 15 와 같다.

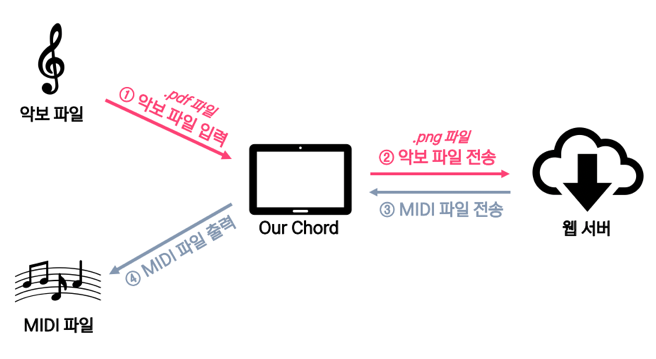


그림 15. 전체 시스템 수행 시나리오

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 웹 서버로 PNG 로 변환 된 악보 파일을 전송한다. 세 번째, 웹 서버에서 변환 된 PNG 악보의 음표를 인식하여 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 와 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 16. 사용자 악보 파일 입력 시나리오

Fig 16. User Score File Input Scenario

위의 그림 16 에서 확인할 수 있듯이 첫 번째 사용자가 OurChord 프로그램에 악보 PDF 파일을 입력한 후, 두 번째로 사용자가 원하는 변환 코드(chord)와 악보의 박자를 입력한다. 세 번째로 OurChord 프로그램에서 악보 PDF 파일을 PNG 파일로 변환한다. 네 번째로 OurChord 프로그램에서 변환 된 PNG 파일과 첫번째 단계에서 사용자가 입력한 원하는 변환 코드를 웹 서버에 전송한다. 다섯 번째, 웹 서버에서 OurChord 프로그램에서 전송 된 데이터를 통해 MIDI 파일을 생성하여 OurChord 프로그램에 전송한다. 여섯 번째로 OurChord 프로그램에서 MIDI 파일이 생성되었는지 확인한다. 마지막으로 생성 된 MIDI 파일을 OurChord 프로그램이 사용자에게 제공한다.

악보 입력 후 조옮김 수행 시나리오는 아래 그림 17 과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

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 TXT 형태로 저장한다. 세 번째 추출한 음표 TXT 데이터를 MIDI 데이터로 변환한다. 네 번째 변환 된 MIDI 데이터를 통해 사용자가 원하는 조(chord)로 MIDI 데이터를 변환한다. 다섯 번째로 변환 된 MIDI 파일을 확인할 수 있도록 한다. 마지막으로 변환 된 MIDI 파일을 OurChord 프로그램을 통해 사용자에게 출력한다.

* + 1. 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 출력한다.

MIDI 파일 변환 시 추출한 음, 박자를 MIDI파일로 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. 서버와 어플리케이션이 입력과 출력이 잘 되는지 확인한다. 그 후 서버에 구성한 알고리즘을 입력하여 어플리케이션에서 입력하면 원하는 출력 값이 나오는지 확인한다.

* 1. **상세 설계**
     1. PDF to PNG 모듈

입력받은 PDF 파일을 PNG 파일로 형식을 변환하고, PNG로 변환된 악보 파일의 배경을 투명으로 처리하여 음표의 위치와 쉼표 및 음표의 박자 구분에 대한 인식률을 높일 것이다.

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.

저장된 PNG 파일의 배경을 투명하게 처리하는 함수는 아래 표 4 와 같다.

표 4. PNG 파일의 악보 배경 투명 처리 함수

Table 4. PNG file background Transparent processing Function

|  |  |
| --- | --- |
| Transparency() | |
| 형식 | Transparency(pngID) |
| 리턴 값 | 배경이 투명한 PNG 파일 |
| 설명 | 변환된 PNG 파일 배경을 투명 변환 |
| 예시 | Newdata.append((255, 255, 255, 0)) |

표 4 에서 확인할 수 있듯이 Transparency() 함수를 통해 변환된 PNG파일 악보의 배경을 append()함수를 통해 흰색, 투명도 0으로 지정하여 PNG형식으로 저장된 악보의 모든 배경을 투명 처리하여 저장한다.

* + 1. 음 출력 모듈(튜닝 위한 음 출력)

사용자로부터 입력 받은 ‘조’에 해당하는 튜닝 음 파일의 경로에 접근하여 해당하는 MP3파일을 출력한다.

사용자가 선택한 튜닝 음을 저장하는 함수는 아래 표 5 와 같다.

표 5. 사용자가 선택한 듣고 싶은 ‘조’ 저장 함수

Table 5. User Selected ‘Chord’ Save Function

|  |  |
| --- | --- |
| getSound() | |
| 형식 | getSound() |
| 리턴 값 | 사용자가 선택한 조(chord) |
| 설명 | 사용자가 듣고 싶은 튜닝음의 ‘조’값 가져온다. |
| 예시 | Sound = getSound(sound) |

표 5 에서 확인할 수 있듯이 사용자가 선택한 튜닝 음의 ‘조’ 값(A, B, C 등)을 getSound() 함수를 통해 받아와 Sound변수에 저장하여 playSound() 함수로 전달한다.

출력할 음의 경로를 불러오는 함수는 아래 표 6 와 같다.

표 6. 선택한 ‘조’에 해당하는 소리 출력 함수

Table 6. 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 6 에서 확인할 수 있듯이 sound 변수를 통해 입력 받은 ‘조’를 확인하고 해당 ‘조’에 해당하는 MP3파일 경로로 접근하여 playSound()함수를 통해 출력한다.

* + 1. 오선지 좌표 추출

오선지에 대한 좌표 추출을 위해 첫번쨰 악보 이미지를 읽어온다. 두번째 악보 이미지를 전처리 하여 오선을 더 잘 검출할 수 있도록 한다. 세번째로 설정한 직선 검출 함수를 이용하여 직선을 검출할 것이다. 마지막으로 검출 된 직선을 필터링 하여 중복 된 직선 좌표를 제거하여 중복되지 않은 오선에 대한 좌표를 추출할 것이다.

첫번째, 이미지를 읽어오는 함수는 아래 표 7 과 같다.

표 7. 이미지를 읽어 오는 함수

Table 7. 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 7 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.

두번째, 이미지 전처리에 대한 함수는 아래 표 8 과 같다.

표 8. 이미지 전처리 함수

Table 8. Function of preprocessingPNG

|  |  |
| --- | --- |
| preprocessingPNG() | |
| 형식 | Image preeprocessingPNG() |
| 리턴 값 | 전처리한 이미지 |
| 설명 | 이미지를 흑백 변환, 이진화, 엣지 검출 등을 거쳐 전처리 된 이미지 변환 |
| 예시 | Img = preprocessingPNG(Img) |

위의 표 8 에서 확인할 수 있듯이 readPNG() 함수를 통해 읽어 온 PNG 파일을 preprocessingPNG() 함수를 통해 흑백 변환, 이진화, 엣지 검출 등 전처리 된 이미지로 변환한다.

세번째, 오선지 좌표 추출에 대한 함수는 아래 표 9 과 같다.

표 9. 오선지 직선 검출 함수

Table 9. 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 9 에서 확인할 수 있듯이 전처리 된 PNG 파일에서 detectStraightLine() 함수를 통해 xylist[] 에 검출한 직선의 x, y 좌표를 리턴한다.

마지막으로 중복 직선 필터링에 대한 함수는 아래 표 10 와 같다.

표 10. 중복 직선 필터링 함수

Table 10. Function of Duplicate Straight Line Filtering

|  |  |
| --- | --- |
| filteringStraightLine() | |
| 형식 | List filteringStraightLine(xylist) |
| 리턴 값 | 필터링 된 x, y 좌표 |
| 설명 | 입력 된 x, y 좌표들을 설정 값에 따라 필터링 후 오선지의 x, y 좌표만 리턴 |
| 예시 | staffList[] = filteringStraightLine(xylist) |

위의 표 10 에서 확인할 수 있듯이 앞서 설명한 detectStraightLine() 함수를 통해 검출 된 직선의 좌표들 중 중복 된 x, y 좌표를 filteringStraightLine() 함수를 통해 검출한다.

* + 1. 템플릿 매칭 통한 음계 분류

템플릿 매칭을 통한 음계 분류는 먼저, 악보PNG 파일을 읽어온 후 전처리 과정을 거친다. 이 후 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 7 과 같고, 이미지 전처리에 대한 함수 또한 앞서 설명한 오선지 좌표 추출에서 설명한 표 8 과 같다.

템플릿 매칭 함수는 아래 표 11 과 같다.

표 11. 템플릿 매칭 함수

Table 11. Function of Template Matching

|  |  |
| --- | --- |
| templateMatching() | |
| 형식 | List templateMatching(Image, List) |
| 리턴 값 | 매칭 된 이미지의 x, y 좌표 |
| 설명 | 설정 값과 매칭할 이미지를 가지고 템플릿 매칭하여 추출한 x, y 좌표 리턴 |
| 예시 | xynoteList[] = templateMatching(Img, List) |

위의 표 11 에서 확인할 수 있듯이 templateMatching() 함수를 통해 매칭 된 이미지의 x, y 좌표를 xynoteList[] 에 저장한다.

탬플릿 매칭 후 중복 된 x, y 좌표 필터링 함수는 아래 표 12 과 같다.

표 12. 중복 매칭 필터링 함수

Table 12. 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 12 에서 확인할 수 있듯이 filteringTem() 함수를 통해 앞서 templateMatching() 함수에서 추출한 x, y 좌표 중 중복 된 좌표들을 검출한 음표의 x, y 좌표를 리턴한다.

* + 1. 딥러닝(CNN) 통한 음표 분류

딥러닝 통한 음표 분류를 위해 훈련 데이터와 테스트할 데이터의 경로를 지정한 후 음표를 학습 시킬 모델을 생성한다. 이 후 학습 시킬 모델을 반복하여 훈련 시켜 음표 분류의 정확도를 높일 것이다.

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 에서 확인할 수 있듯이newTrainPNG() 함수에 딥러닝 시 훈련할 음표 데이터의 경로를 지정해준다.

테스트할 데이터의 경로를 불러오는 함수는 아래 표 14 와 같다.

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 있듯이newTestPNG() 함수에 딥러닝 시 검증할 음표 데이터의 경로를 지정해준다.

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 정화하게 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.

* + 1. 조 변환

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 변환 시킨다.

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.

* + 1. MIDI 데이터 변환

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 있듯이saveBaseMIDI() 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 있듯이saveChangeMIDI() 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.

* 1. **Prototype 구현**
     1. 요구 분석

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

* + 1. 프로토타입 개발/개선

OurChord 앱의 화면 흐름도는 아래 그림 18과 같다. 프로토타입의 요구 분석에 따라 주요 기능인 조옮김 기능을 사용하기 위해서는 악보 파일 업로드와 조옮김 설정이 필수이다.

조옮김 기능 외에 OurChord 는 그림 18 앱의 화면 흐름도에 따라 회원가입 후 OurChord 의 여러 기능들을 사용할 수 있다.

첫번째로, 기타같은 현악기의 각 현의 소리 튜닝 시 편하게 설정할 수 있도록 사용자가 듣고 싶은 튜닝음의 ‘조’를 선택하게 되면 해당 튜닝 음을 제공해주는 튜닝 메뉴가 제공된다.

두번째로 OurChord 앱의 공지사항, FAQ, 개인 설정을 변경할 수 있는 MY 메뉴가 제공된다. 개인 설정 변경 시 비밀번호 확인 후 회원 가입시 입력한 내용을 수정할 수 있도록 한다.

지도이(가) 표시된 사진

자동 생성된 설명

그림 18. OurChord 앱 화면 흐름도

Fig 18. OurChord Application Flowchart

위의 그림 18 에서 확인할 수 있듯이 로그인 후 메인 화면에 악보 폴더, 조 옮김, 튜닝, MY 4 개의 메뉴가 동시에 보이게 된다.

악보 폴더 메뉴는 아래 그림 19 와 같이 구성되어 있다. 사용자가 업로드한 PDF 악보 파일을 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 19. OurChord 앱의 악보 폴더 화면–목록 형

Fig 19. OurChord Application Music Sheet Screen-List type

아래 그림 20 과 같이 조옮김 된 MIDI 악보를 앱 외부의 내가 사용하는 기기 자체에 저장, 악보 파일 앱에서 삭제, 악보 파일 이름 수정이 가능하다. 또한 상단의 스위치 토글 버튼을 사용하여 스위치 토글 버튼이 왼쪽이면 목록형, 오른쪽이면 아이콘형으로 악보 파일을 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 20. OurChord 앱의 악보 폴더 화면-아이콘 형

Fig 20. OurChord Application Music Sheet Screen-Icon type

조옮김 설정 메뉴는 아래 그림 21 과 같이 구성되어 있다. 조옮김 할 PDF 악보를 선택한 후 선택한 악보의 기존 조(chord) 와 박자, 바꾸려는 조를 사용자가 선택한 뒤 ‘CHORD CONVERSION’ 버튼을 클릭하면, 사용자가 변환하고자 하는 PDF 악보 파일을 php에서 해당 경로에 접속하여 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 20과 같이 표시 후 확인 할 수 있다.

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

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 21. OurChord 조옮김 설정 화면

Fig 21. OurChord Application Chord Conversion Set Screen

OurChord 로그인 화면은 아래 그림 22 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.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 22. OurChord 로그인 화면

Fig 22. OurChord Application Login Screen

OurChord 의 ID 와 비밀번호 찾기 화면은 아래 그림 23 과 같다. ID 를 잊었을 경우 이름과 회원 가입 시 입력한 E-mail 을 입력하면 입력한 E-mail 로 ID 를 확인할 수 있다. 비밀번호를 잊었을 경우 ID 와 E-mail 을 입력하면 ID 를 찾을 때와 마찬가지로 입력한 E-mail 로 비밀번호 재설정 링크를 전송한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 23. OurChord ID, 비밀번호 찾기 화면

Fig 23. OurChord Application Find Id, Password Screen

OurChord 의 회원 가입 화면은 아래 그림 24 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.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 24. OurChord 회원 가입 화면

Fig 24. OurChord Application Member Register Screen

OurChord 의 튜닝 음 화면은 아래 그림 25 와 같다. 사용자가 원하는 튜닝 음을 선택한 후 스피커 모양 버튼을 클릭하면 튜닝 음이 스피커에서 출력된다.

시계이(가) 표시된 사진

자동 생성된 설명

그림 25. OurChord 튜닝 음 화면

Fig 25. OurChord Application Tuning Set Screen

OurChord 의 개인 정보 수정 화면은 아래 그림 26 과 같다. 비밀번호 확인 후 입력 창이 활성화되는데, 올바른 비밀번호 입력 후 개인 정보를 수정한 뒤 수정 버튼을 클릭하면 개인 정보 수정이 완료된다.



그림 26. OurChord 개인 정보 수정 화면

Fig 26. OurChord Application Individual Info Modify Screen

OurChord 의 공지사항 화면은 아래 그림 27 과 같다. 중요 공지사항은 앞에 ‘(중요)’ 가 붙으며, #F9688C 색으로 표시된다. 확인하고자 하는 공지사항 제목을 클릭하면 아코디언 메뉴 형식으로 공지사항의 상세 내용을 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 27. OurChord 공지사항 화면

Fig 27. OurChord Application Notice Screen

OurChord 의 FAQ 화면은 아래 그림 28 과 같다. 공지사항과 마찬가지로 확인하려는 FAQ 를 클릭하면 아코디언 메뉴 형식으로 FAQ 의 답을 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 28. OurChord FAQ 화면

Fig 28. OurChord Application FAQ Screen

* + 1. 프로토타입 평가
  1. **시험/테스트 결과**
     1. 템플릿 매칭 정확도
     2. 딥러닝 모델 박자 인식 정확도
     3. 서버, 데이터베이스 연동
     4. 앱, 서버 연동
  2. **Coding & Demo**
     1. 회원가입
     2. 로그인
     3. ID, 비밀번호 찾기
     4. 튜닝용 음 출력
     5. 악보 파일 업로드
     6. 조옮김 MIDI 파일 제공
     7. 개인 정보 수정
     8. 공지사항 확인
     9. FAQ 확인

**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 미쯔히사, 수도 코다이, 쿠로사와 타쿠마, 오다 다이스케, “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/musescore/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-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 - [30편] 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&sl=en&u=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)