



악보를 볼 줄 몰라도 연주가 가능하다.

머신러닝 기능을 활용한 피아노 전반 지시 어플리케이션



Intel Project

연주 도우미

TEAM : 낙원상가(Paradise store)
양승용 오태영 박철우 최은호

Contents

- 01 프로젝트 개요 및 진행도
- 02 시스템 구성 요소
- 03 개발 과정 및 결과
- 04 문제점 및 개선방안
- 05 기대효과
- 06 향후 발전 방향
- 07 고찰

프로젝트 개요

주제 선정 배경

- 문제 인식

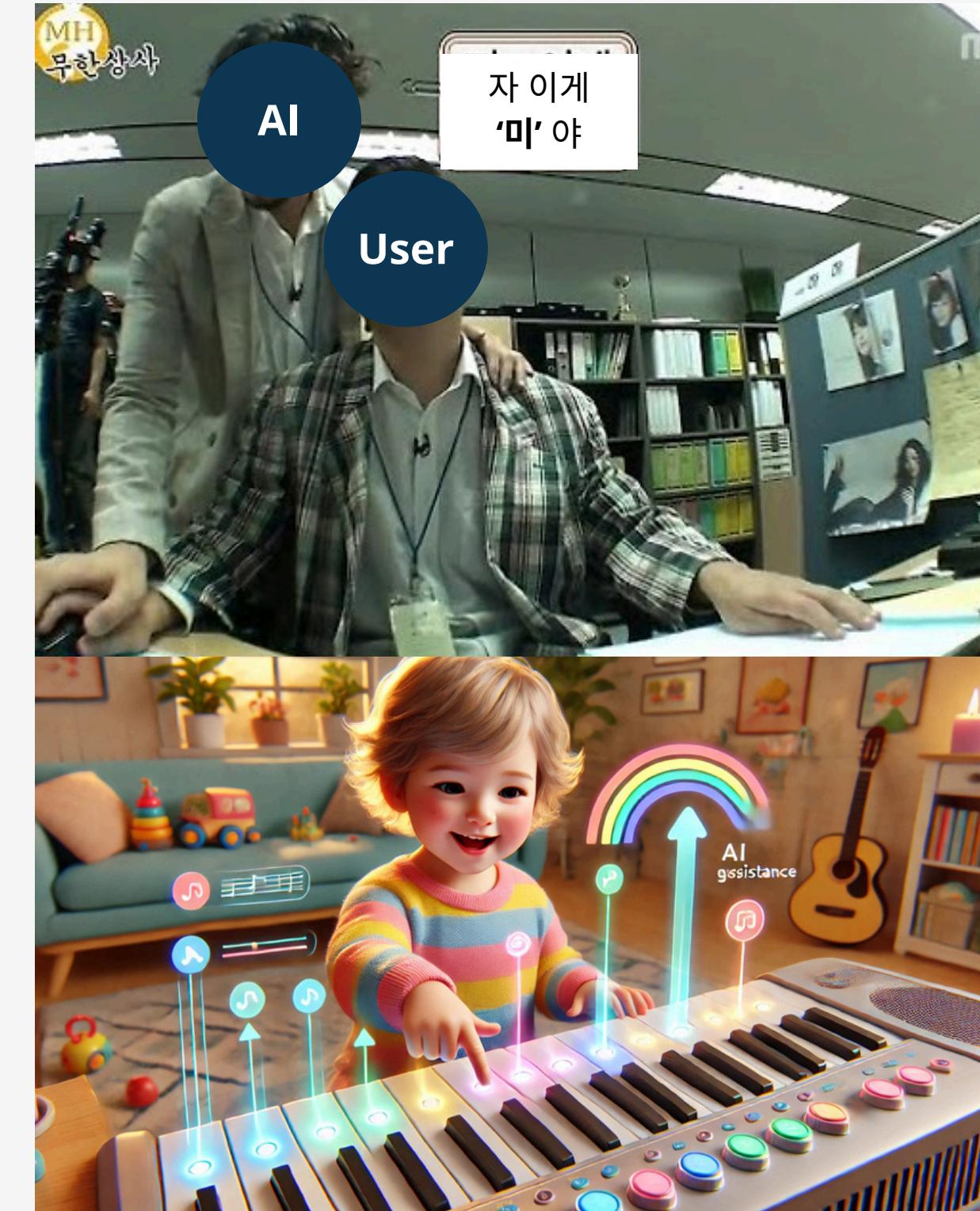
- 피아노 및 악기를 독학하면서 가장 큰 어려움 중 하나는 악보 읽기
- 악보를 이해하고 연주를 시작하는 데는 많은 시간이 걸림
- 어린이들에게는 매우 복잡하고 부담스러운 과정

- 문제 해결 및 목표

- 어린이들이 악보 없이 연주를 시작할 수 있는 환경 제작
- 학습 초기에 느끼는 좌절감과 불안감 감소
- 어린이들이 음악에 대한 흥미를 잃지 않고 학습을 지속할 수 있는 환경 제작

- 기대 효과

- 음악에 대한 흥미를 유도 및 장기적인 학습 동기 부여
- 단순히 악기를 넘어 음악을 통해 자신을 표현하고, 창의적인 사고에 기여
- 악기 학습에 대한 접근성 상승 및 음악을 즐기는 문화 확산



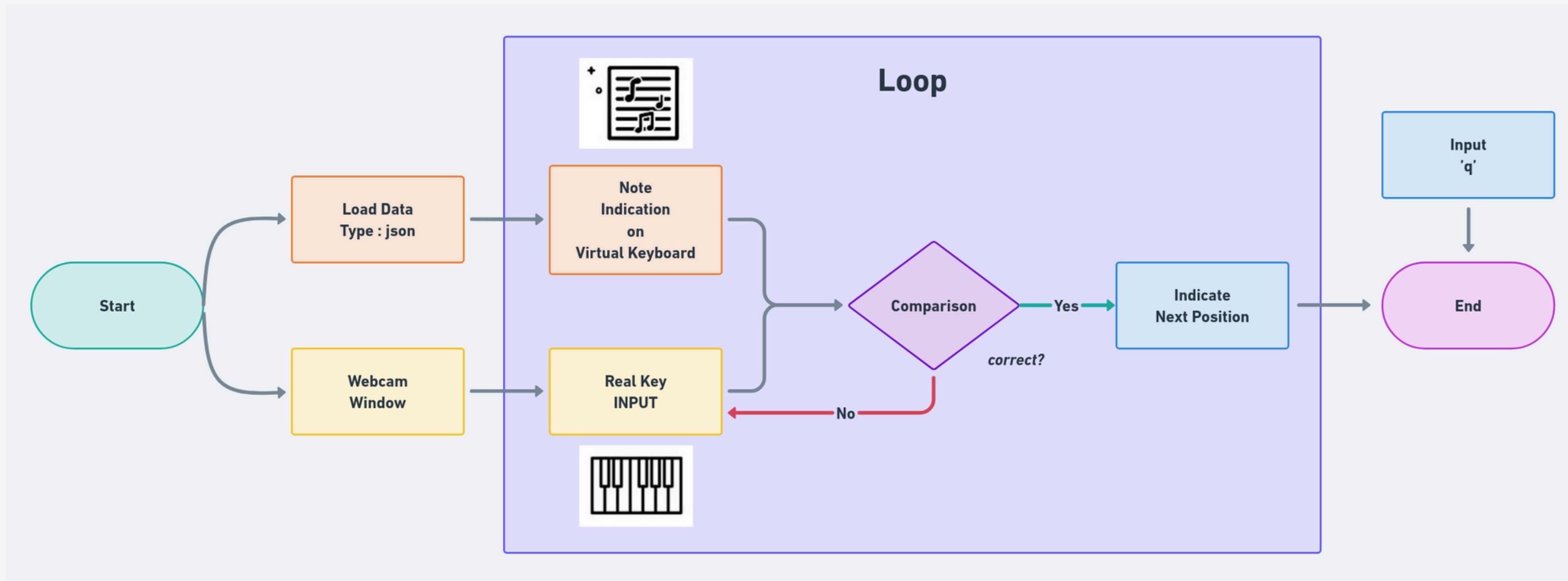
01

목표 일정 및 진행도

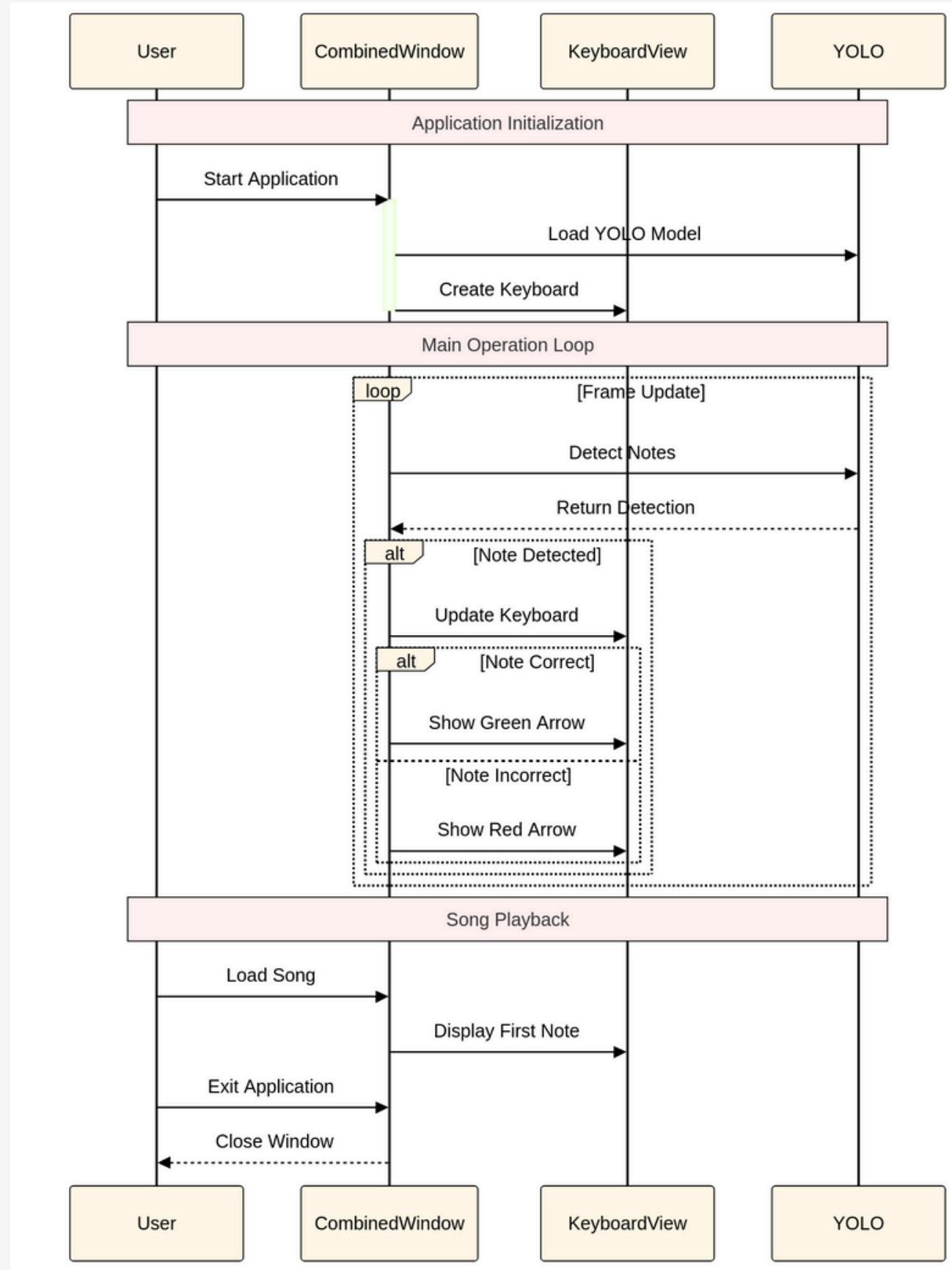
진행도 : 5/7 (12월 19일)									계획	진행	
3조 프로젝트 계획 및 진행사항											
목표	상세내용	담당자	1일차(수)	2일차(목)	3일차(금)	4일차(월)	5일차(화)	6일차(수)	7일차(목)		
프로젝트 구조 설계	기능 정의 디렉토리 구조 설정 클래스 다이어그램 작성	최은호, 박철우, 오태영, 양승용									
기본 클래스 구현	DataHandler 클래스 구현 검토 기본 예외처리 구현 패키지구조 생성	오태영,최은호									
Virtual Keyboard UI 기본 구조 (core)	메인 윈도우 레이아웃 설계 키보드 디스플레이 영역 구현 기본 컨트롤 버튼 구현	양승용									
키 인식 ML알고리즘 구현 (core)	OpenCV, Openvino 손 동작 및 키 인식 알고리즘 구현 인식 정확도 개선 보정 알고리즘 구현	오태영, 박철우									
core 기능 통합 및 동기화	웹캠 데이터와 UI 동기화 악보 데이터 실시간 로딩	오태영, 최은호									
기능 통합테스트 및 보완점 수정	컴포넌트 통합 테스트 성능 테스트 버그 수정	박철우, 양승용									
발표자료		양승용, 오태영									

02

Flow Chart



Sequence diagram



발표자용 설명 스크립트 (임시)

피아노 학습 애플리케이션 - 시퀀스 흐름 설명

1. 애플리케이션 초기화 단계

- 사용자가 애플리케이션을 시작하면서 프로세스 시작
- CombinedWindow(메인 애플리케이션)가 초기화되며 두 가지 핵심 설정 작업 수행:
 - 음표 감지를 위한 YOLO 모델 로드
 - 시각적 표현을 위한 KeyboardView 생성
- 이 초기화 과정으로 모든 핵심 구성 요소가 작동 준비 완료

3. 곡 재생 프로세스

- 사용자가 곡을 로드하면서 시작
- CombinedWindow가 곡 파일 처리
- KeyboardView가 첫 번째 음표를 표시하여 사용자 안내
- 사용자가 따라할 학습 순서 생성

4. 애플리케이션 종료

- 사용자가 종료 시작
- CombinedWindow가 정리 작업 수행
- 애플리케이션이 사용자에게 종료 확인

구현 시 주요 고려사항

- 오류 처리:**
 - 모든 구성 요소 간 상호작용에 오류 처리 포함
 - 시스템이 연결 끊김이나 실패를 원활하게 처리해야 함
- 성능 고려사항:**
 - 프레임 처리는 실시간으로 이루어져야 함
 - 시각적 피드백은 즉각적이어야 함
 - 음표 감지는 정확해야 함
- 사용자 경험:**
 - 피드백은 명확하고 즉각적이어야 함
 - 인터페이스는 작동 중에도 반응성을 유지해야 함
 - 시각적 신호는 직관적이고 도움이 되어야 함

2. 메인 작동 루프

애플리케이션이 다음과 같은 주요 작동 루프에 진입:

음표 감지 프로세스

- CombinedWindow가 지속적으로 YOLO에 프레임 전송
- YOLO가 각 프레임을 처리하고 감지 결과 반환
- 음표가 감지되면 다음 순서로 진행:
 - CombinedWindow가 감지 결과 수신
 - KeyboardView가 감지된 음표를 반영하여 업데이트

피드백 프로세스

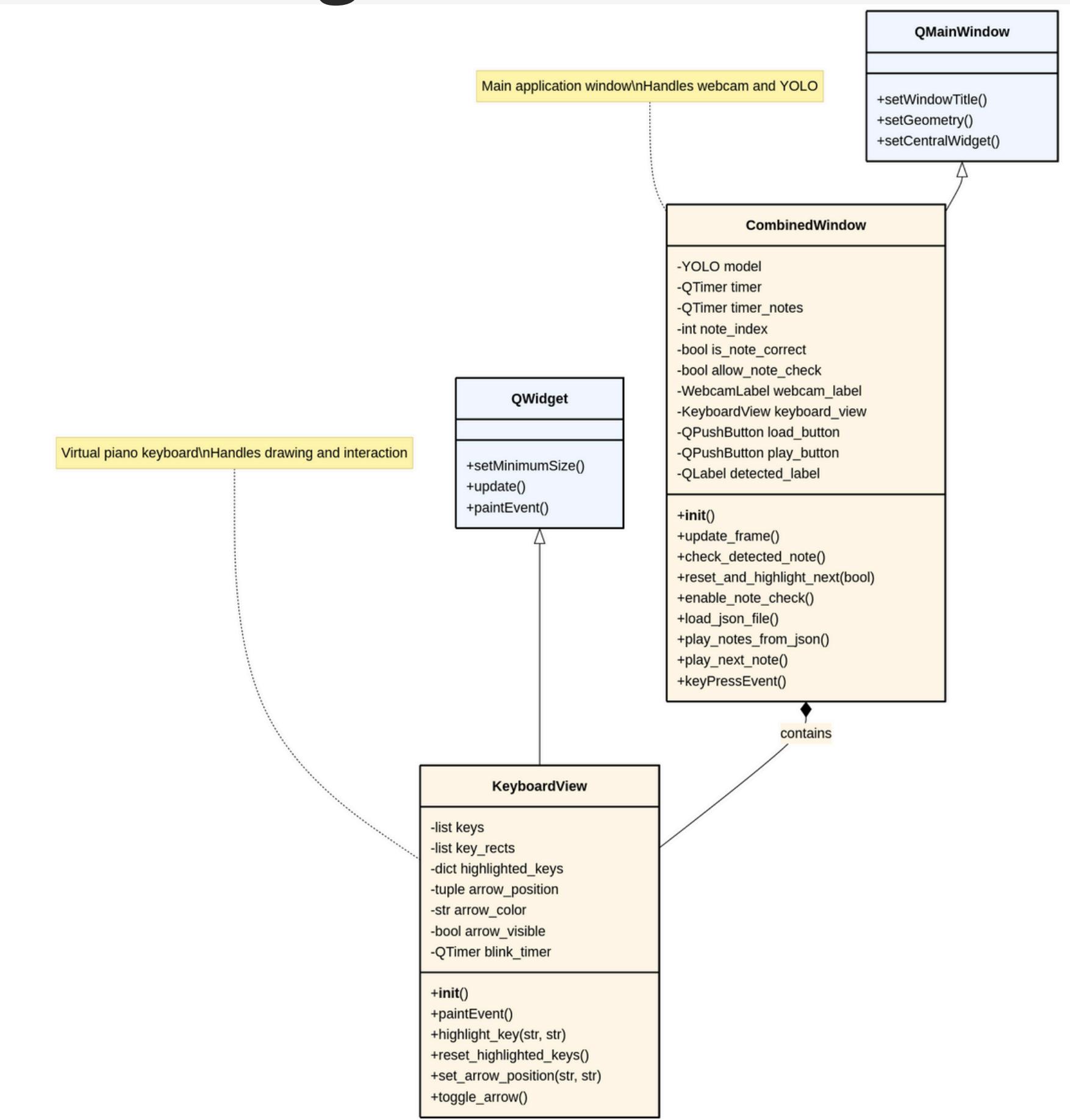
음표 감지 결과에 따라 두 가지 경로 존재:

- 정확한 음표 경로:**
 - 시스템이 올바른 음표 인식
 - KeyboardView가 초록색 화살표 표시
 - 시각적 피드백으로 사용자에게 성공 알림

- 부정확한 음표 경로:**

- 시스템이 잘못된 음표 인식
- KeyboardView가 빨간색 화살표 표시
- 시각적 피드백으로 사용자를 올바른 음표로 안내

Class diagram



발표자용 설명 스크립트 (임시)

피아노 학습 애플리케이션 - 클래스 구조 설명

1. 기본 클래스 구조

QMainWindow 클래스

- Qt 프레임워크의 기본 윈도우 클래스
- 주요 메서드:
 - setWindowTitle(): 윈도우 제목 설정
 - setGeometry(): 윈도우 크기와 위치 설정
 - setCentralWidget(): 중앙 위젯 설정
- 역할: 애플리케이션의 메인 윈도우 기능 제공

QWidget 클래스

- Qt의 기본 위젯 클래스
- 주요 메서드:
 - setMinimumSize(): 최소 크기 설정
 - update(): 위젯 갱신
 - paintEvent(): 그리기 이벤트 처리
- 역할: 기본적인 UI 구성 요소 기능 제공

2. 핵심 구현 클래스

CombinedWindow 클래스

- QMainWindow를 상속받은 메인 애플리케이션 클래스
- 주요 속성:
 - YOLO model: 음표 인식을 위한 AI 모델
 - QTimer timer: 프레임 업데이트용 타이머
 - KeyboardView keyboard_view: 가상 키보드 표시
 - 기타 UI 컴포넌트들 (버튼, 라벨 등)
- 주요 메서드:
 - update_frame(): 웹캠 프레임 업데이트
 - check_detected_note(): 감지된 음표 확인
 - play_notes_from_json(): JSON 파일에서 곡 재생
- 역할: 전체 애플리케이션 로직 제어 및 조율

3. 클래스 간 관계

상속 관계

1. CombinedWindow --> QMainWindow
 - CombinedWindow가 QMainWindow의 기능을 확장
 - 윈도우 관리 기능 상속

2. KeyboardView --> QWidget
 - KeyboardView가 QWidget의 기능을 확장
 - 기본 위젯 기능 상속

포함 관계

3. CombinedWindow *-- KeyboardView
 - CombinedWindow가 KeyboardView를 포함
 - 생명주기 관리 및 직접적인 제어 가능

4. 구현 시 고려사항

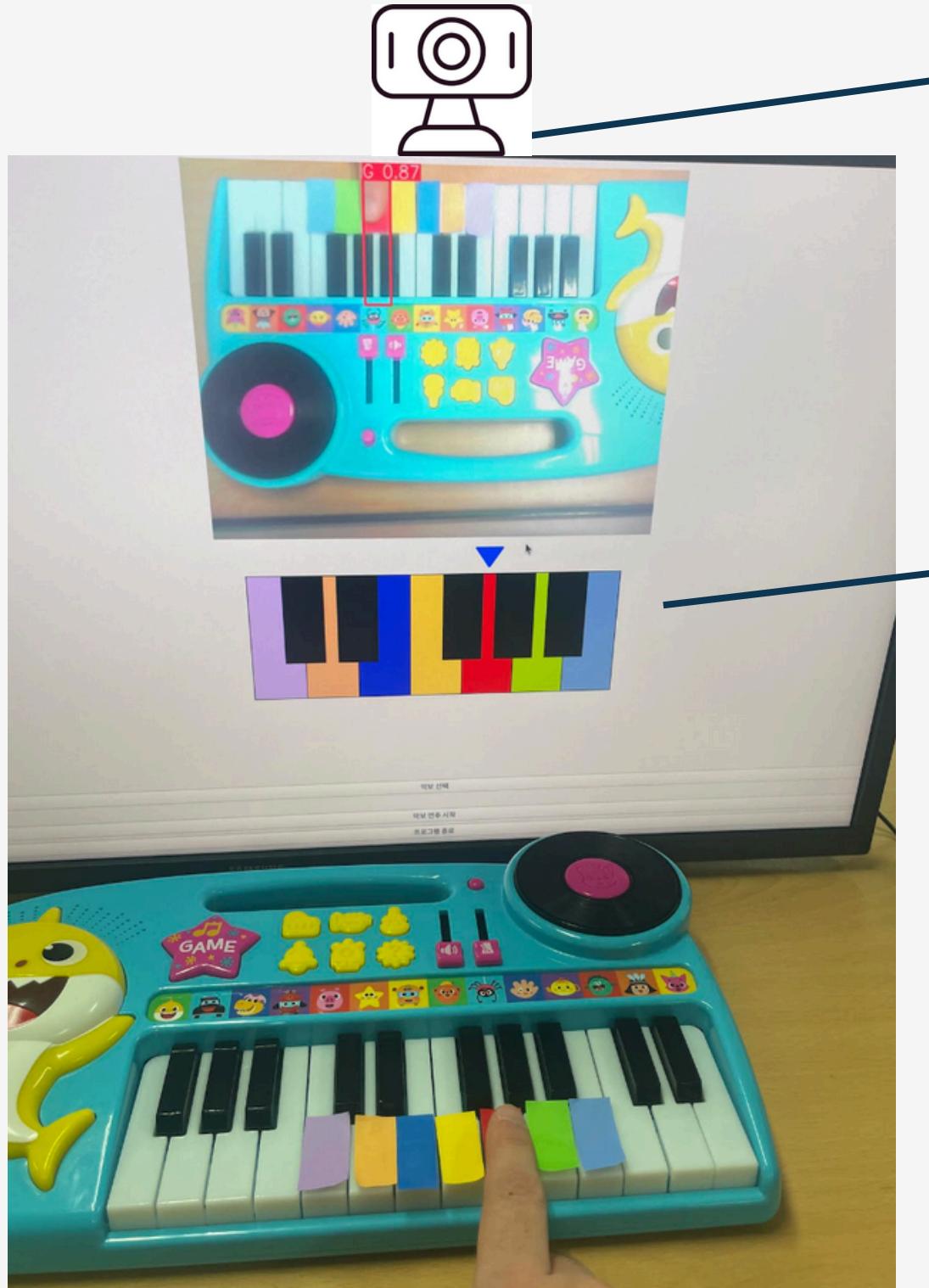
1. 확장성
 - 새로운 기능 추가가 용이한 구조
 - 각 클래스의 책임이 명확히 분리됨

2. 유지보수성
 - 클래스 간 의존성이 명확히 정의됨
 - 각 컴포넌트별 독립적인 수정 가능

3. 재사용성
 - Qt 프레임워크의 표준 클래스 활용
 - 컴포넌트별 독립적 재사용 가능

4. 성능
 - 타이머를 이용한 효율적인 업데이트 처리
 - 이벤트 기반 구조로 리소스 효율적 사용

프로그램 구성 요소



1. 머신 러닝을 통한 건반인식 프로그램 작성

Dataset 생성 및 Labeling

Augmentation으로 부족한 Dataset 보강

Yolo 11n 모델에 직접 생성한 Dataset을 학습시켜 최적화된 모델

2. 주요 패키지

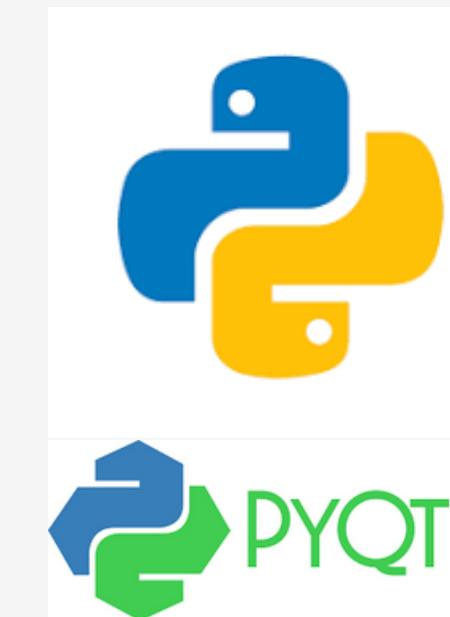
ultralytics: YOLO 모델 패키지

opencv-python: OpenCV를 사용하여 웹캠 프레임 처리

pyqt5: PyQt5 GUI 프레임워크

numpy: YOLO 모델 결과 처리

json: 기본 Python 모듈, JSON 파일을 처리 시 사용



3. 버전 요구사항

numpy>=1.21.0

opencv-python>=4.5.5

ultralytics>=8.0.0

PyQt5>=5.15.4

발표자용 설명 스크립트 (임시)

- YOLO 모델을 활용한 실시간 건반 인식
 - OpenCV와 YOLO 모델을 이용해 웹캠에서 건반을 실시간 감지
 - 감지된 건반 데이터를 UI에 업데이트 및 가상 건반 데이터와 비교
- 가상 피아노 키보드
 - PyQt5를 사용하여 가상 피아노 건반 생성(한 개의 옥타브로 제한)
- 게이름 파일을 통해 가이드 제공
 - JSON 파일을 통해 사용자가 연주할 건반을 load 및 해당 건반을 가상 피아노 건반에서 순서대로 하이라이트
- UI 상호작용
 - PyQt5의 위젯 기능들을 이용해서 사용자 상호작용 처리

1. 웹캠을 통한 실시간 음표 감지:

YOLO 모델을 사용하여 실시간으로 웹캠에서 촬영된 이미지를 분석하고, 특정 음표(예: C, D, E 등)를 감지하고 웹캠에서 받은 프레임을 OpenCV와 PyTorch 기반 YOLO 모델을 사용하여 처리합니다. YOLO 모델의 결과에서 각 객체(음표)의 클래스 ID를 확인하고, 가장 높은 신뢰도를 가진 음표를 detected_note 변수에 저장합니다.

2. 가상 피아노 키보드

KeyboardView 클래스로 가상 피아노 키보드를 표시하는 Qt 위젯으로, 키보드의 각 음표(C, D, E 등)를 흰 건반과 검은 건반으로 나누어 그려 각 건반은 QPainter를 사용하여 그리며, 마우스 클릭 이벤트가 발생하면 해당 건반을 누른 것으로 처리하여 콘솔에 출력합니다.

사용자가 감지된 음표를 따라 연주할 때, 해당 음표의 건반을 하이라이트(색상 변경)하여 사용자가 어떤 음표를 따라가야 하는지 명확히 보여줍니다.

3. 음표에 맞춰 연주

JSON 파일에는 연주할 음표들(예: {"note": "C"})이 순차적으로 정의하여 사용자가 JSON 파일을 로드하면, 해당 파일에 포함된 음표들이 순차적으로 가상 키 보드에서 하이라이트되어 실제로 플레이되는 음표를 확인할 수 있습니다. 음표는 play_notes_from_json 메서드를 통해 처리되며, 그에 맞춰 키보드에서 연주하는 음표가 자동으로 하이라이트되어 각 음표는 QTimer를 사용하여 정해진 시간 간격으로 하이라이트를 전환하며 연주됩니다.

4. 음표 감지 및 맞춤 체크:

감지된 음표는 "self.detected_note"에 저장되며, 현재 연주해야 할 음표와 비교하여 정확히 맞는지 확인합니다.

사용자가 실제로 연주하는 음표가 맞으면 self.is_note_correct = True가 설정되고, 그 후 음표 인덱스 (note_index)가 증가하여 다음 음표로 넘어갑니다. check_detected_note 메서드는 이 로직을 수행하며, 맞는 음표가 감지되면, 가상 키보드에서 하이라이트된 키를 리셋하고, 새로운 음표로 변경합니다.

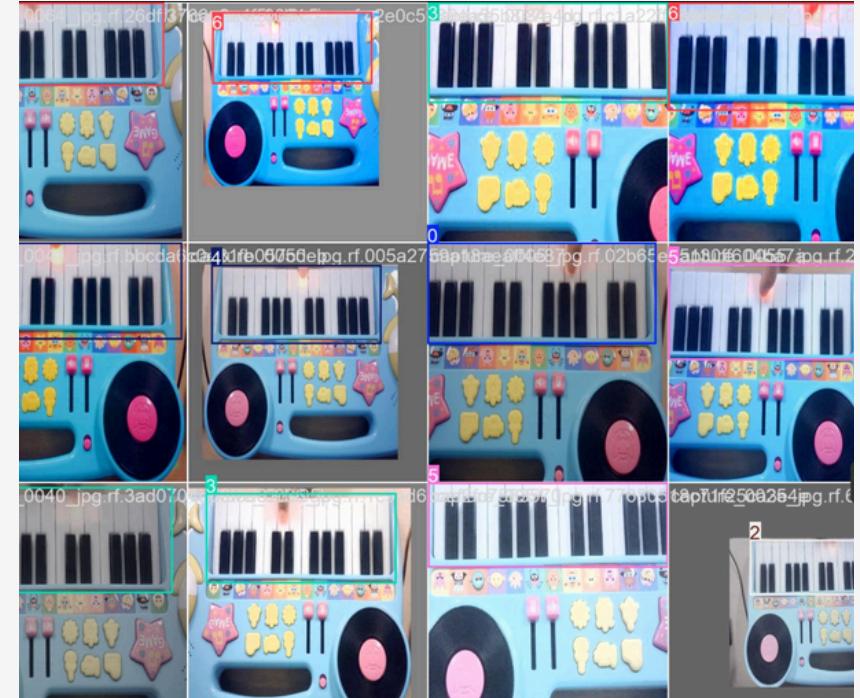
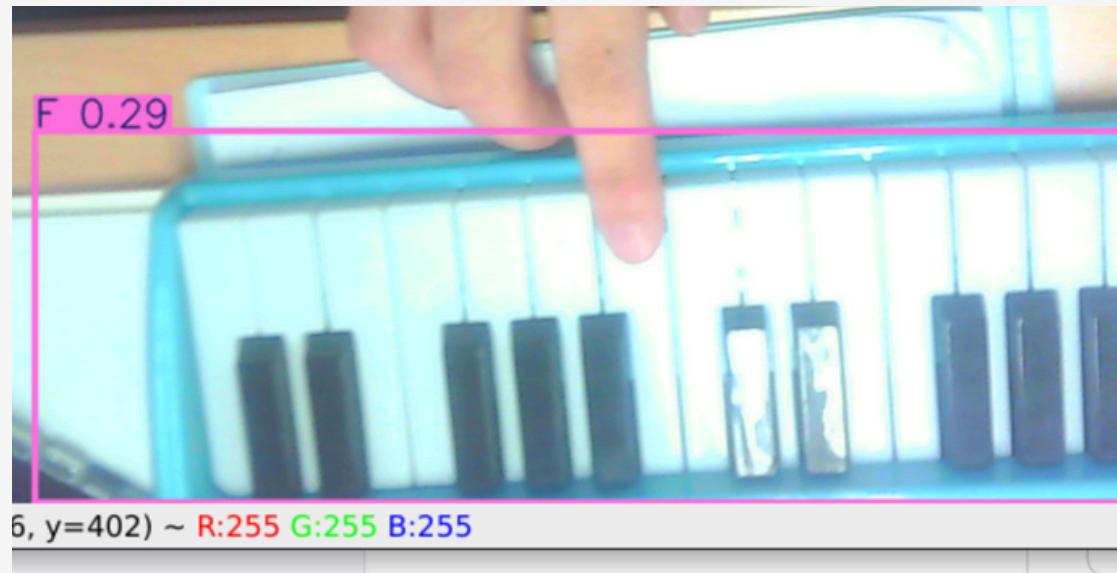
5. UI 및 버튼 상호작용

PyQt5 UI 사용하여 웹캠 화면, 가상 키보드, JSON 파일 로드 버튼, 음표 플레이 버튼, 음표 감지 상태 표시 레이블 등이 있습니다. 사용자가 JSON 파일을 로드하거나 음표를 연주하는 버튼을 클릭하면 그에 따라 동작이 수행됩니다. 각 버튼은 슬롯 메서드와 연결되어 있으며, 이벤트 기반으로 동작합니다. 예를 들어, "Play Notes" 버튼을 클릭하면 play_notes_from_json 메서드가 실행되어 연주를 시작합니다.

개발 과정 및 결과

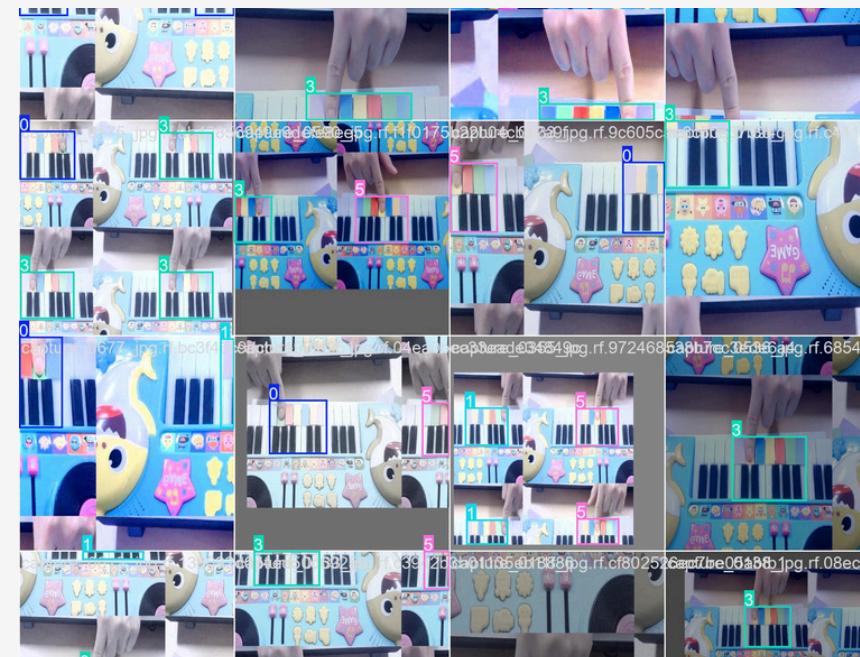
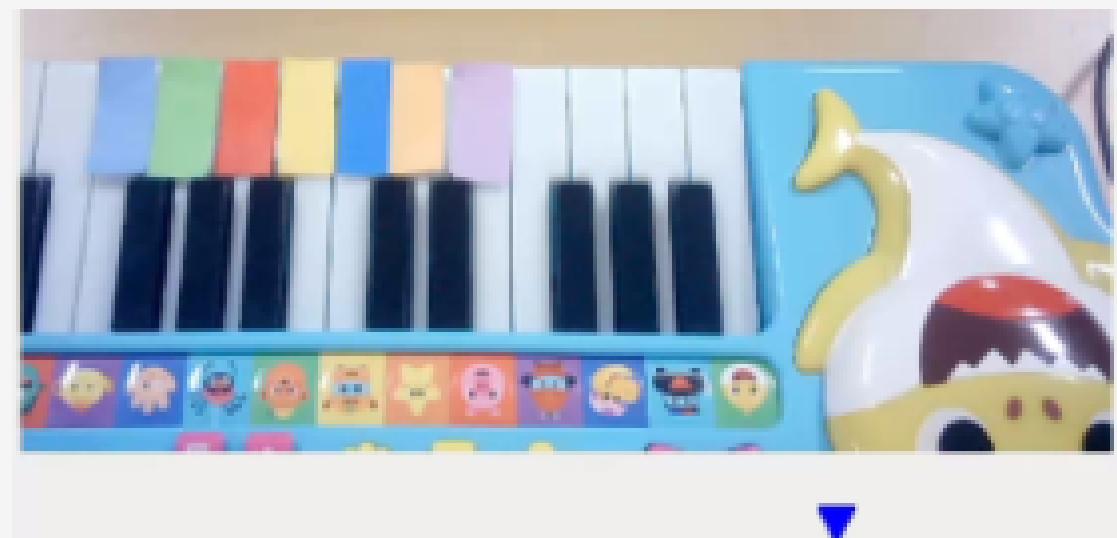
1단계 - 건반의 전체영역 라벨링 및 YOLO 모델 학습

- 피아노 건반 전체영역을 학습했으나
각 건반들의 유사한 형태로 인해 feature의 구분이 되지 않아
건반을 정확히 인식하지 못하는 결과 발생.
- 개선방안 : 색종이 부착을 통한 인식률 개선 및 라벨링 보강.



2단계 - 건반에 색종이를 부착하여 인식률 개선

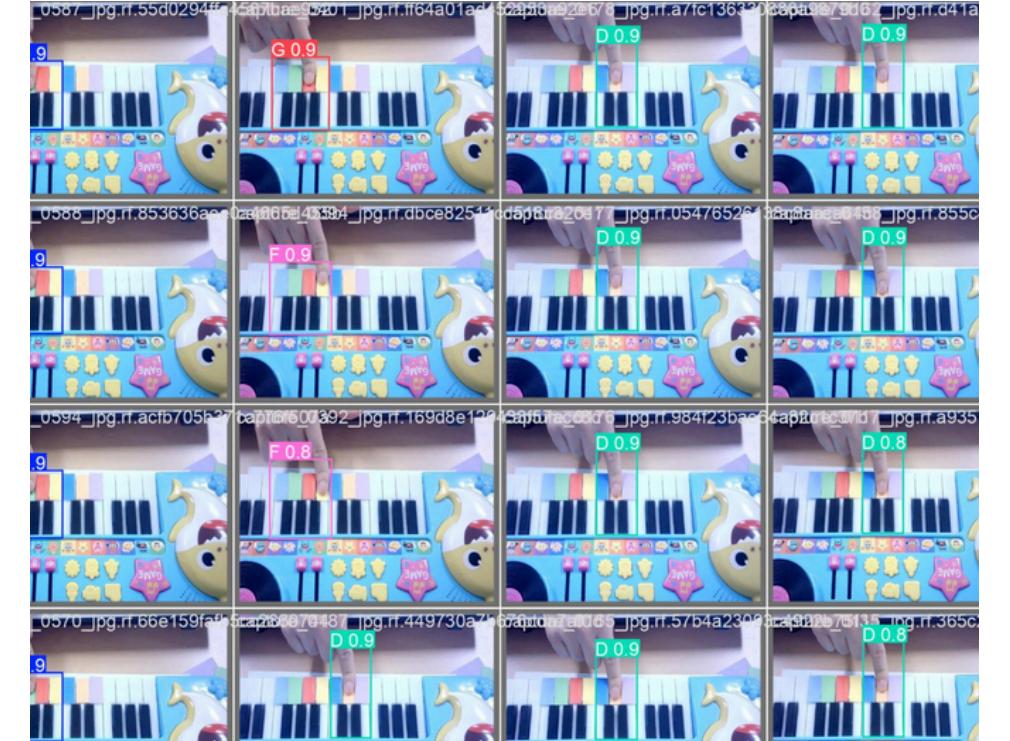
- 색종이를 각 건반에 부착하여 모델을 학습.
7개의 건반 중에 2개만 인식
- 개선방안 : 건반을 하나씩 분리하여 라벨링하고
개별적으로 인식률을 높이기 위한 시도 필요



개발 과정 및 결과

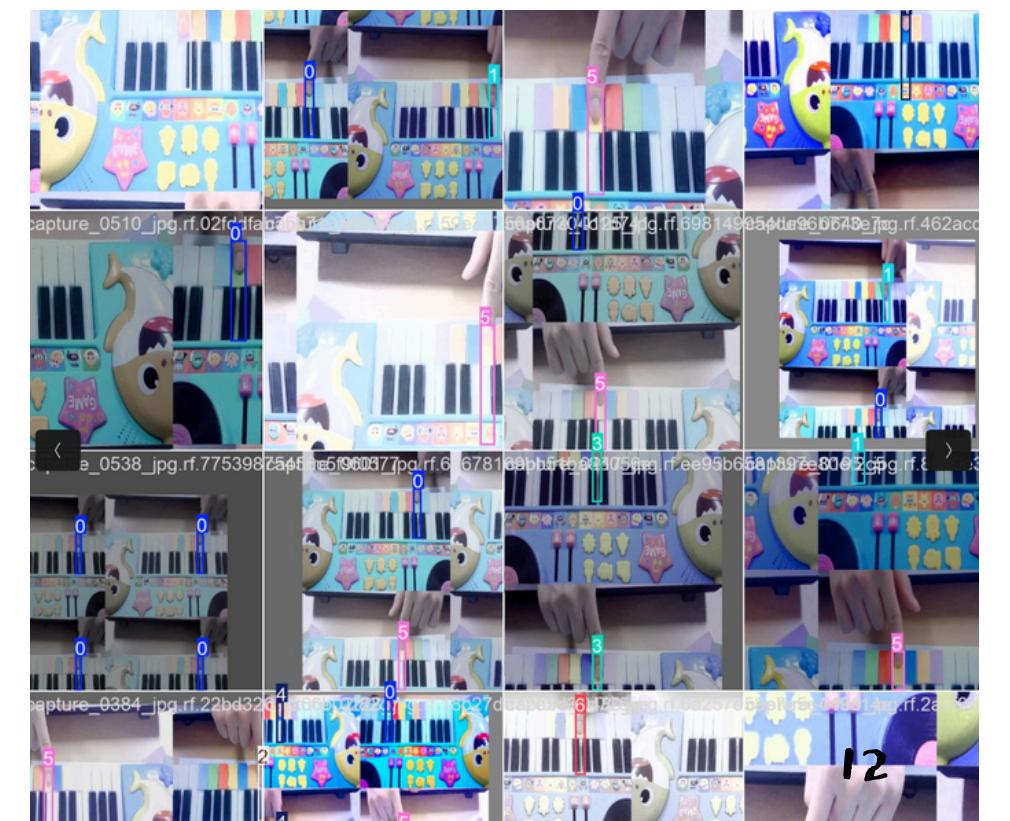
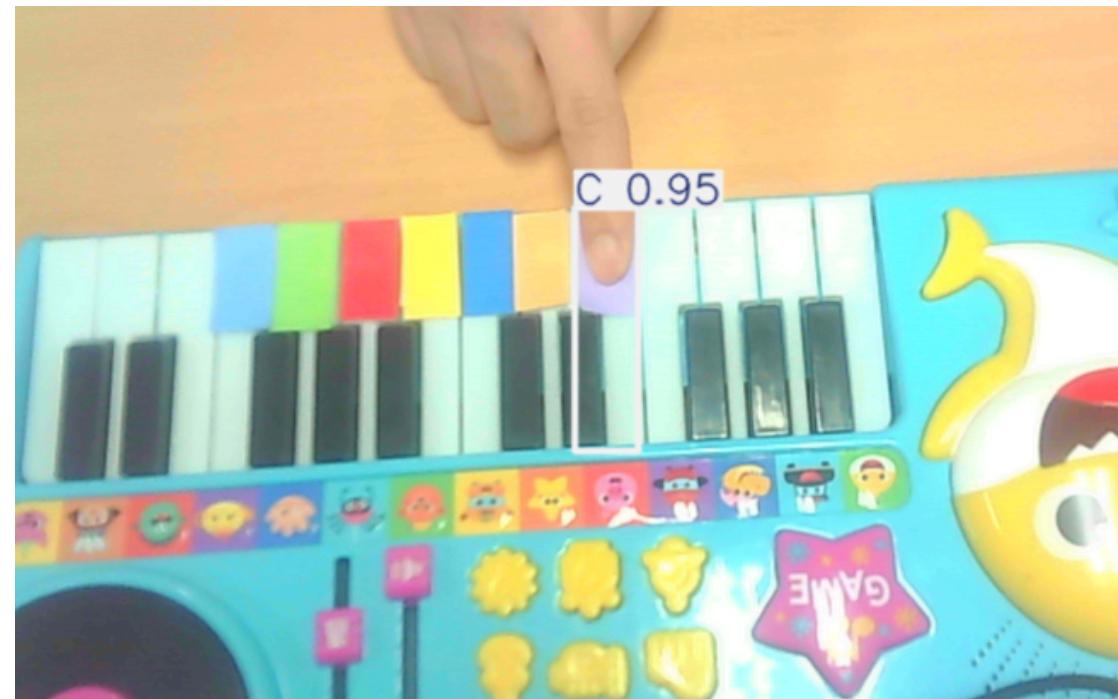
3단계 - 건반 2분할 학습

- 건반의 영역을 2개로 나누어 학습
7개 중 4개 인식
- 개선방안 : 건반을 7개 기본 음으로 나누어 학습
Augmentation을 추가 및 적용
인식 정확도 향상 기대



4단계 - 건반 7분할 학습

- C~B, 총 7개의 건반을 분할하여 학습
모델이 모든 건반을 정확하게 인식
- 모델의 일반화 성능 향상을 통해
다양한 조명 조건과 환경에서 일관된 성능 유지



04

문제점 및 개선방안

- **연속된 동일한 건반에 대한 인식 오류**

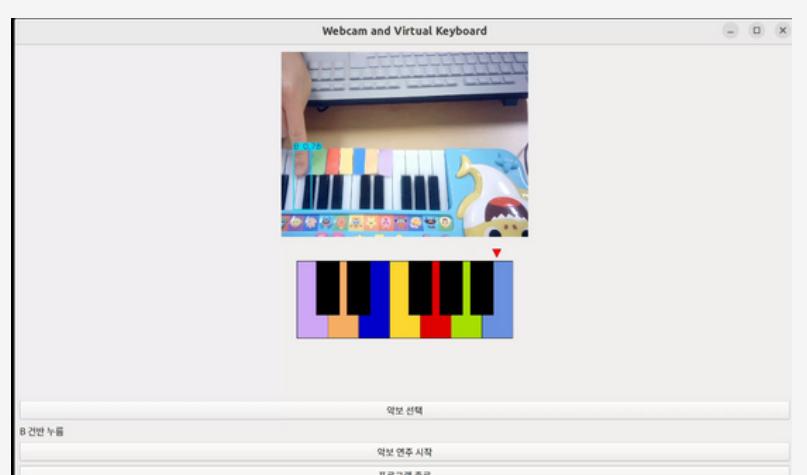
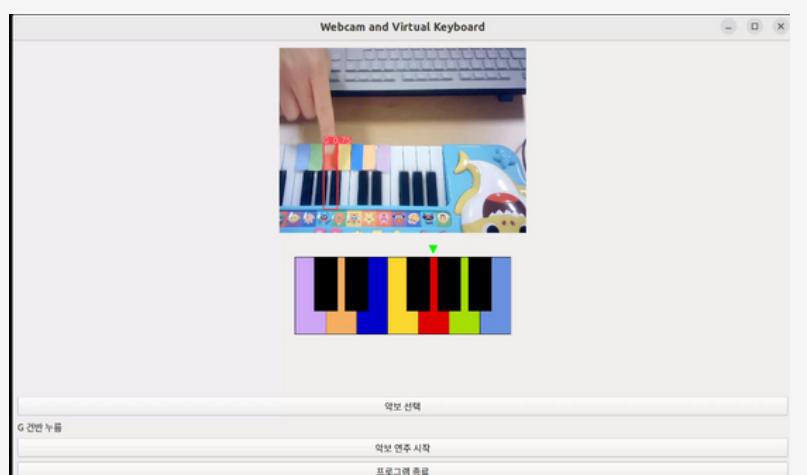
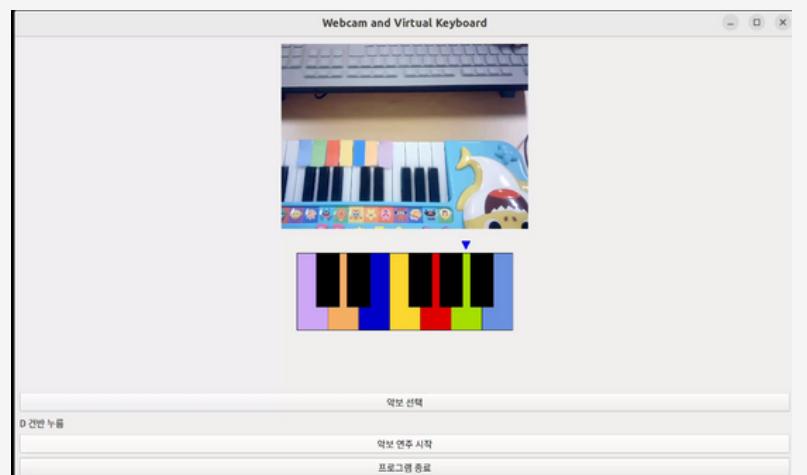
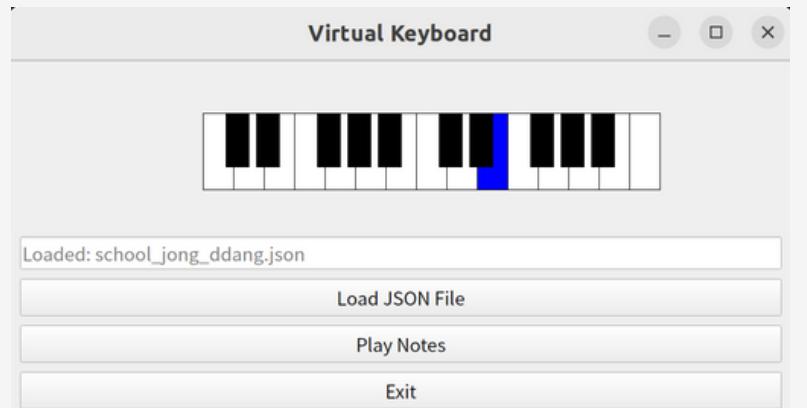
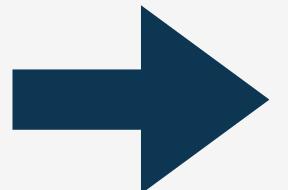
- 건반 입력 후 다음 건반 입력 사이에 딜레이를 추가
- 해당 시간 동안 데이터 처리를 하지 않으면서 문제 해결

- **가상 건반과 실제 건반 간 색상 불일치로 인해 사용자에게 시각적 혼동 초래**

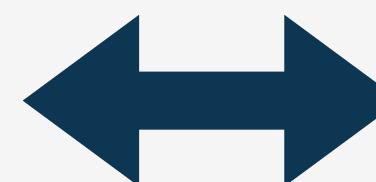
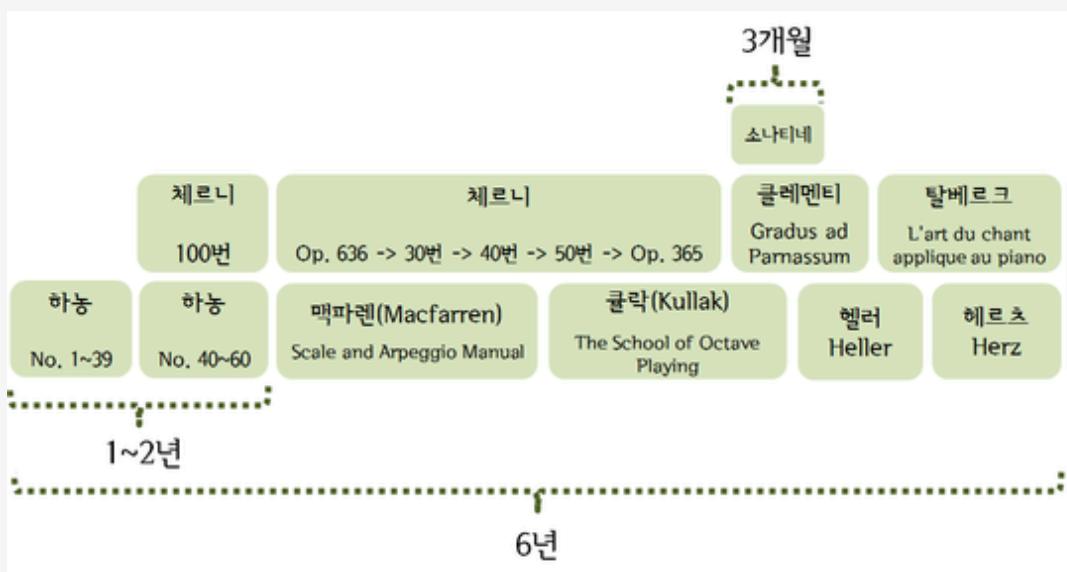
- 가상 건반의 색상을 실제 건반에 적용한 색상과 통일
- 기존에 파란색으로 건반을 하이라이트 하면서 순서를 표시
- 건반 위에 화살표로 표시하면서 혼동을 줄이고 좀 더 직관적으로 인지 가능하게 함

- **조명 또는 색 온도 변화에 따른 인식 성능 저하**

- augmentation 항목마다 range를 늘리고 학습량이 늘어난다면 개선될 것으로 예상됨



- 사용자 경험: 악보 없이 자유로운 연주, 실시간 피드백 제공
- 교육적 가치: 피아노 학습의 접근성을 높이고, 악보를 몰라도 연주 가능
- 기술적 우위: 기존 학습 시스템과 비교하여 차별화된 AI 기술 및 실시간 기능



기초 리듬 연습 (6)

하농 4번

$\langle \text{리듬 연습} \rangle^{\textcircled{4}}$

3
4

3
4



향후 발전 방향

가상 키보드



다양한 악기



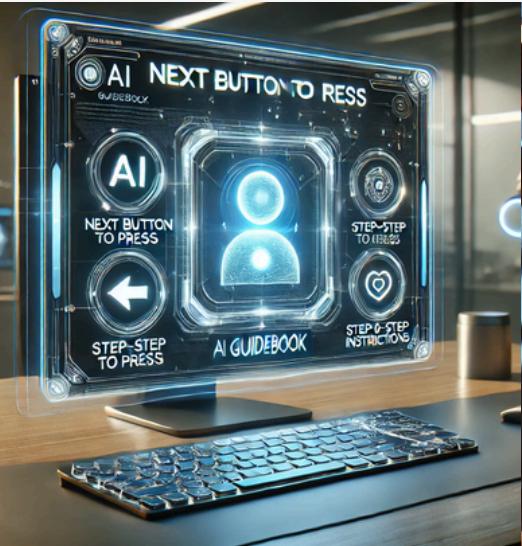
피아니스트



조립 가이드



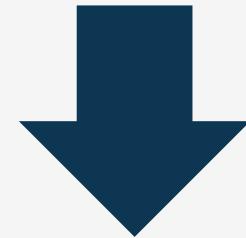
가상 가이드북



조종 어시스턴트



악보에 나열된 순서대로 동작을 수행



표준 시퀀스/설명서대로 동작을 수행할 수 있도록
지시 할 수 있는 모든 분야에 널리 적용 가능!