Data Extraction 알고리즘

박신형

Ⅰ 평가 기준

1. 기존 논문
   a. 딥러닝 모델 학습
      1) 대다수의 논문들은 LSTM, CNN (딥러닝 모델) 을 사용하여 **(Audio, Video) 데이터**를 training 시켜, 동일한 곡에 대해서 얼마나 시각적 (Visual) 또는 청각적 (Audio) 관점에서 유사한가 를 평가
      2) 피아노 연주에 대한 구체적인 평가 기준을 제시 한 것이 아님
      - Piano Tutoring System (ICPR 2018, h5-index 95)
      - Piano Performance Evaluation (Wireless Communications and Mobile Computing, 2023, IF : 2.146)
      - Piano Performance Evaluation (Applied Sciences, 2021, IF : 2.7)

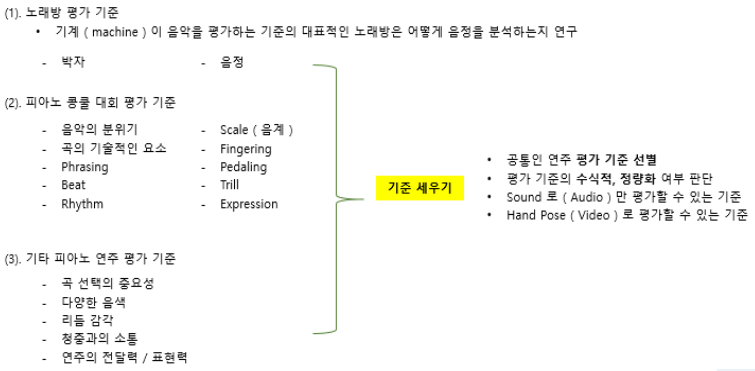
   b. 구체적인 평가 기준
      - 2가지 논문 참조

논문	데이터	평가 기준
PIANO SKILLS ASSESSMENT	• Minimum, average, and maximum performance lengths were 570, 2690, and 10038 frames 약 이 정도의 데이터 셋	(1). PL (player level) --> https://www.mtna.org/MTNA/Learn/Tips/Rhythm.aspx - 연주자의 실력 / 곡의 난이도 등을 참고 (2). SL (song level) --> https://lvmta.com/musicianship --> https://www.conbrioexams.com/exams-grade-inidications --> 음악적인 요소 (음정, 소리, 박자 등)(위 링크에 세부 내역 존재)
Observing Pianist Accuracy and Form with Computer Vision	• MFCC (Audio 데이터) • 피아노 연주 정면 데이터	• [Audio] ▪ 음악의 속도 (초당 음표수) • [Video] ▪ 아르페지오의 어려움 & 속도 ▪ 연주자의 skills (pianist 에게만 있는 unique skills) ▪ 손의 움직임

   c. 공헌
      - (1). 피아노 연주 데이터 베이스
        ① Audio 데이터 : 실제 target mid 데이터에 대응하는 연주 (test) 데이터가 오픈 소스, 인터넷에 많이 없음
        ② Video 데이터 :피아노 위에서 탑뷰로 찍은 데이터 중 오픈 된것을 찾기 어려웠음
        - 이러한 점에서, 앞으로 피아노 연주 평가와 관련된 연구를 위한 DB 에 공헌할 수 있음
      - (2). Hand pose 평가 방법 제안
        - 현재까지 피아노 연주에 관하여, 공식 가능한 평가기준과 공식 불가능한 평가 기준을 **복합적으로 정의내려** 피아노 연주에 대해 보다 detail 하게 평가하였던 연구가 없었다 (검토 필요) (**기존 연구의 한계점**)
        - **본 연구는, 따라서, 독자적인 Comparison 알고리즘 + 딥러닝 모델을 사용하여, 복합적인 요소를 고려한 피아노 연주 평가 모델을 개발하고자** 함

2. 우리 논문

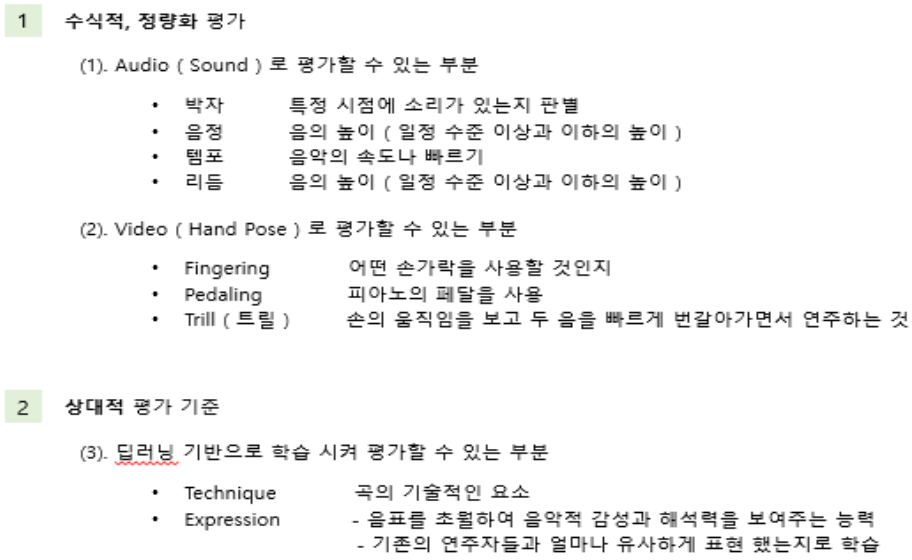
a. 평가 기준 분류



- 노래방에서 점수 평가 기준
  - ^ 박자
  - ^ 음정
- 피아노 콩쿨 대회에서 연주 평가 기준
  - ^ 음악의 분위기
  - ^ 곡의 기술적인 요소
  - ^ Phrasing (프레이징)
  - ^ Beat
  - ^ Rhythm
  - ^ Scale (음계)
  - ^ Fingering (어떤 손가락 사용 여부,)
  - ^ Pedaling
  - ^ Trill
  - ^ Expression
- 기타 피아노 연주 평가 기준
  - ^ 곡 선택의 중요성
  - ^ 다양한 음색
  - ^ 리듬 감각
  - ^ 테크닉의 완성도
  - ^ 균형 있는 양손 소리
  - ^ 능숙한 페달 처리
  - ^ 연주의 전달력 / 표현력
  - ^ 청중과의 소통

① 공통적 평가 기준 찾기

- 1차적으로 (1). 노래방 (2). 콩쿨 대회 (3). 기타 피아노 연주 평가 기준 등을 참조하여, 공통적으로 피아노 연주를 평가하기 위해 어떠한 기준을 정의하였는지 파악한 결과, “박자, 음정, 리듬, 템포, 손가락 연주 위치, 표현력, 연주자의 분위가“ 등이 있었다.

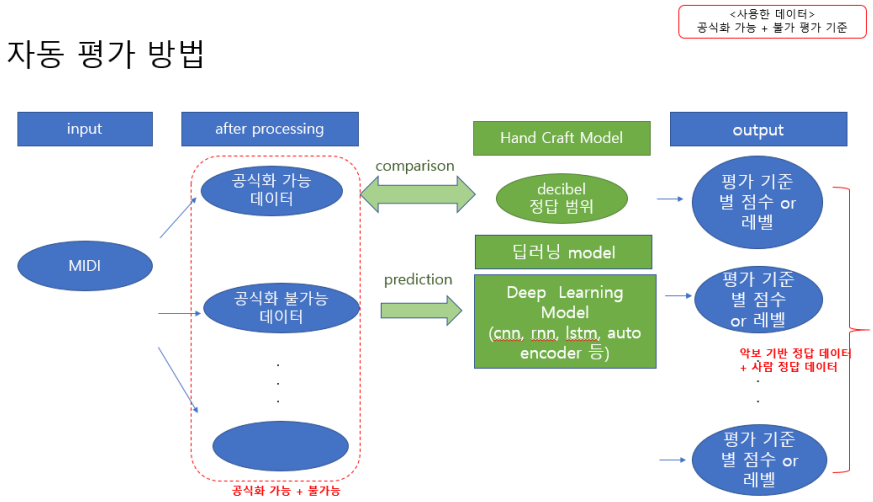


② 수식적, 정량화 평가 VS 상대적 평가

- 이후, 공통된 피아노 연주 평가 기준에서 수식적으로 정량화 할 수 있는 평가 기준과 상대적으로 평가할 수 있는 기준을 분리하였다.
- 수식적, 정량화 할 수 있는 평가 기준은 크게 ① Audio (Sound)평가 기준 ② Video (Hand Pose) 평가 기준으로 분리 할 수 있었고, Audio 평가 기준에는 “박자, 음정, 템포, 리듬 " 이 Hand Pose 평가 기준은 “ Fingering, Pedaling, Trill” 등이 있다.
  반면, 상대적 평가 기준에는 “연주자의 Technique과 Expression” 이 있다.
- 이렇게 분리한 이유는,
  (1). 수식적, 정량화하여 평가할 수 있는 기준은, **정확한 정답을 바탕으로 평가** 할 수 있지만, 상대적 평가 기준은 경험을 바탕으로 평가할 수 있기 때문이다.
- (2). 나아가, 수식적, 정량화 평가 기준은 정답을 바탕으로 하기 때문에, 객관적이지만, 상대적 평가 기준은 개인마다 그 평가 값이 다르기 때문이다.

③ 수식적, 정량화 평가 VS 상대적 평가 기준에 따른 알고리즘 (모델 설계)

- (1). 수식적, 정량화 평가 기준 : target data (정답 데이터) 와 test (input) 데이터를 비교
- (2). 상대적 평가 기준은 기존 train 데이터의 연주 평가를 바탕으로 **딥러닝 모델**을 학습시켜 test 데이터의 연주 평가를 예측하는 방향으로 알고리즘을 설계해야 할 것 같음



④ 1차 개발 (**공식 가능한 평가 기준**)

- 1차 개발 목표로, 공식 가능한 평가 기준을 위한 데이터 추출 알고리즘 설계

b. 공식 가능한 평가 기준 (세우기)

평가 기준 → **추출 데이터**

	평가 기준	매트릭	
1	음정	note	Note on / Note off 여부, note (pitch)
2	셈여림	decibel	Velocity (음의 세기)
3	셈여림의 변화	change of decibel	
4	빠르기	speed	Tempo, bpm
5	빠르기의 변화	change of speed	
6	붙임줄, 스타카토, 테누토, 늘임표	duration of note	Control 데이터 : pan, main_vol, depth
7	악센트	decibel	note (pitch)
8	옥타브	note	
9	꾸밈음, 반복 기호	note	
10	리듬	?	
11	페달링	pedaling	Pedal 데이터
12	음의 길이	duration of note	

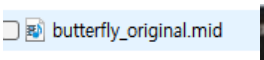
공식화 가능한 기준

① 평가 기준

- 우리 논문에서 사용할 공식 가능한 평가 기준으로는 총 12가지의 평가 기준을 세웠고 (영민/민서님께서 최종 평가 기준 세움(12가지))
- 이러한 기준을 바탕으로 피아노 연주 performance 를 측정하기 위해서는 피아노 연주 데이터로부터 **“note on//off 유무, note (pitch), velocity (음의 세기), control 데이터 (연주의 특징), pedal 데이터”**가 필요하다 .

Ⅱ 특징 추출 알고리즘 (개발)

1. Input 데이터
   - Midi 데이터 (Musical Instrument Digital Interface)
   - Midi 데이터가 무엇인지 / cubase 를 통해서 어떻게 데이터를 추출하는지
     (민서님께서 내용 보충)



2. 데이터 추출 라이브러리
   - **Midi 데이터 분석 라이브러리**
     - Mido 와 Music21 모두 음악과 관련된 작업을 수행하는 파이썬 라이브러리

     ① Mido 라이브러리
     - MIDI 파일을 입출력할 수 있는 패키지
     - ✓ 장점) Music21 에 비해 사용하기에 초보자들에게 보다 쉽다.
     - ✓ 단점). Midi 데이터만 지원한다

     ② Music21
     - 음악 이론, 분석, 악보 생성 등의 작업을 수행하는 파이썬 라이브러리
     - 음악적인 요들을 분석하고, 악보로 표현하며작곡 및 연구에 활용
     - MIDI 파일 뿐만 아니라 다양한 형식의 음악 데이터를 지원
     - ✓ 장점) MIDO 라이브러리에 비해 추출할 수 있는 음악적인 요소들이 방대하다
     - ✓ 단점) 음악을 잘 모르는 초보자들이 이해하기에 어렵다

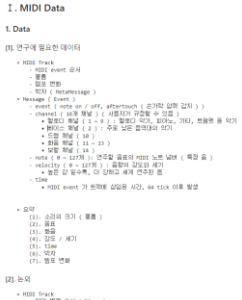
     - 우리 **논문에서는 MIDO 라이브러리를** 사용하여 개발을 진행
       - (1). Mido 라이브러리는 음악적 지식이 전무한 개발자들도 MIDI 파일을 분석하기에 유용하다
       - (2). Mido 라이브러리는 Music21에 비해 추출할 수 있는 음악적 요소가 적음에도 불구하고 우리가 원하는 데이터들을 Midi 파일로부터 충분히 추출할 수 있음
   - MIDO 라이브러리 (간략한 소개)
     - MidiFile(type=1,ticks_per_beat=480, tracks=[…])
     - 여러 track을 가질 수 있으며 각 트랙은 자체 음악 이벤트 세트를 가짐
     - Ticks_per_beat = 480 : midi 의 시간 단위인 tick 당 분해를 의미
     - 크게 [MetaMessage] 와 [Message]로 이루어져 있다.

     ① MetaMessage : track_name, tempo, time_signature (박자), 등

     ② Message
     - Message(‘note_on, ‘channe=0, note=45, velocity=78, time=64)
     - Message 는 msg.type 에 따라서 이벤트 특징 값이 달라진다.

     ➢ 관련 논문 (MIDI 데이터 비교 알고리즘) (영민님)
     - https://elib.uni-stuttgart.de/bitstream/11682/11454/1/MA_ChristianSchierle_MIDIComparisonVis

     ➢ Mido 라이브러리
     - https://mido.readthedocs.io/en/stable/message_types.html



3. Data Extraction 알고리즘 (**중요한 부분만 기술**)

➢ 코드 위치

- https://github.com/dalabdgw/Research_and_Experimental_Results/blob/main/Piano%20Performance%20Evaluation/ShinHyeong%20Park/Final_result_240212/%5BLatest%5D%20Automatic%20Assessment%20of%20Piano%20Performance%20Using%20MIDI%20Data%20extracted%20by%20mido%20V1.ipynb

[0]. Mid_test_performance (input_name, target_name)

```
def mid_test_performance(input_name, target_name):
    input_mid, target_mid = load_midi_data(input_name, target_name)  ①

    if pre_check(input_mid, target_mid):  ②
        print("===== pre_check success =====")
        input_info, target_info = extrack_mid_info(input_mid, target_mid)  ③
    else:
        print("===== pre_check failed =====")
        return
    input_msg_info, target_msg_info = pd.read_csv("midi_data/"+input_name.split('.')[0]+'.csv'),

    # 성능 테스트
    performance_evaluate(input_msg_info, target_msg_info)  ④
```

[1]. butterfly_v1 테스트

```
data_folder = "midi_data"

input_name  = "butterfly_input_data.mid"
target_name = "butterfly_target_data.mid"

mid_test_performance(input_name, target_name)
```

다음과 같이 실행

① load_midi_data

- Input data 와 target data 의 midi 파일을 불러오기

② pre_check

- Midi 데이터의 Message 부터 음악적 요소를 추출 하기 앞서,
  load_midi_data 로 불러온 Midi 정보를 통해, input 과 target 데이터를 비교
- (1). track 길이 비교 (input 과 target 의 track 길이는 같아야 한다)
- (2). track_name (곡 명, 연주 악기 등) 비교
- (3). track 시간 (연주 시간) 이 같아야 한다

```
def pre_check(input_mid, target_mid):

    # track 길이 비교
    if input_track_len != target_track_len:
        return False

    # track_name 비교
    if input_track_name != target_track_name:
        return False

    # track 시간 비교 ( time_diff_interval 이상 비교 불가)
    if (input_mid_time - target_mid_time) > time_diff_interval:
        return False


    return True
```

③ extrack_mid_info(input_mid, target_mid)

- Precheck() 결과 True 이면 실행
- (1). MetaMessage 데이터 추출
  - time (second, bpm 등) 계산을 위해, tempo 를 따로 저장
- (2). Message 데이터 추출

a. Process_msg (msg) 를 통해, msg.type 에 따른 처리를 함

- note_on, note_off
  - msg_type (note on/off)
  - Channel
  - note (음정 , pitch)
  - Velocity (음의 세기)
  - Count (0.1 초 안에 얼마나 많은 건반 (음정) 을 눌렀는지)
- program_change
- Control_change
  - msg.control 에 따른 분류
    - msg.control == 1 : 'modulation' (음향 효과)
    - msg.control == 7 : 'main_vol' (전체 볼륨)
    - msg.control == 10: 'pan' (오디오 신호의 왼쪽 오른쪽 간의 밸런스)
    - msg.control == 64 : 'pedal' (페달 신호)
    - msg.control == 91 ~ 93 : 'depth (특정 효과의 깊이)

b. 0.1 초 단위로 데이터를 저장

- Input midi 데이터와 target midi 데이터의 time 기준을 동일시 맞춤
- 0.1 초 단위로 음악적인 요소를 비교

④ performance_evalutate

- 추출된 Input, target 데이터를 바탕으로 비교 알고리즘 실행

Ⅲ 정확도

[1]. butterfly.mid 테스트

[1]. butterfly_v1 테스트

```
data_folder = "midi_data"

input_name  = "butterfly/butterfly_input_data.mid"
target_name = "butterfly/butterfly_target_data.mid"

mid_test_performance(input_name, target_name)
```

```
==================== [Input Midi Data] ====================
파일 이름:  midi_data₩butterfly/butterfly_input_data.mid
총 재생 시간:  27.00000000000003
--------------------------------------------------------------------------------
트랙 이름:  butterfly
총 트랙의 수:  2
[1], butterfly
[2], PIANO

==================== [Target Midi Data] ====================
파일 이름:  midi_data₩butterfly/butterfly_target_data.mid
총 재생 시간:  27.0
--------------------------------------------------------------------------------
트랙 이름:  butterfly
총 트랙의 수:  2
[1], butterfly
[2], PIANO
===== pre_check success =====

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

0it [00:00, ?it/s]

====================  [Input_Info[Message]]  ====================
[1], 음정 정확도: 82.0%
[2], 셈여림 유사도: 32.62%
[3], 셈여림 변화 일관성: 32.62%
[11], 페달링 일관성: 0.00%
```

[1]. 음정 정확도: 82.0%
[2]. 셈여림 유사도: 32.62%
[3]. 셈여림 변화 일관성: 32.62%
[11]. 페달링 일관성: 0.00%

[2]. archive/testt 파일 테스트

[2]. archive/testt 파일 테스트 ¶

```
data_folder = "midi_data/mid_collection/testt"

input_name  = "test.midi"
target_name = "target.midi"

mid_test_performance(input_name, target_name)
```

```
==================== [Input Midi Data] ====================
파일 이름:  midi_data/mid_collection/testt₩test.midi
총 재생 시간:  12.470833333333331
--------------------------------------------------------------------------------
트랙 이름:  test_rivers_flow
총 트랙의 수:  3
[1], test_rivers_flow
[2], Piano
[3], Yamaha MOX

==================== [Target Midi Data] ====================
파일 이름:  midi_data/mid_collection/testt₩target.midi
총 재생 시간:  12.424999999999997
--------------------------------------------------------------------------------
트랙 이름:  test_rivers_flow
총 트랙의 수:  3
[1], test_rivers_flow
[2], Piano
[3], Yamaha MOX
===== pre_check success =====

0it [00:00, ?it/s]

[1], 음정 정확도: 14.193548387096778%
[2], 셈여림 유사도: 75.61%
[3], 셈여림 변화 일관성: 75.61%
[11], 페달링 일관성: 79.35%
```

[1]. 음정 정확도: 14.193548387096778%
[2]. 셈여림 유사도: 75.61%
[3]. 셈여림 변화 일관성: 75.61%
[11]. 페달링 일관성: 79.35%

[3]. river flows in you 테스트

[3]. river flows in you 테스트

```
data_folder = "midi_data"

input_name  = "river_flows_in_you_test_v2.mid"
target_name = "river_flows_in_you_origin_v2.mid"

mid_test_performance(input_name, target_name)
```

```
==================== [Input Midi Data] ====================
파일 이름:  midi_data₩river_flows_in_you_test_v2.mid
총 재생 시간:  174.7791666666652
--------------------------------------------------------------------------------
트랙 이름:  test_rivers_flow
총 트랙의 수:  2
[1], test_rivers_flow
[2], Piano

==================== [Target Midi Data] ====================
파일 이름:  midi_data₩river_flows_in_you_origin_v2.mid
총 재생 시간:  164.77394462499913
--------------------------------------------------------------------------------
트랙 이름:  test_rivers_flow
총 트랙의 수:  2
[1], test_rivers_flow
[2], Piano

[1], 음정 정확도: 5.38675570395103%
[2], 셈여림 유사도: 22.84%
[3], 셈여림 변화 일관성: 22.84%
[11], 페달링 일관성: 0.00%
```

[3]. river flows in you 테스트

[1]. 음정 정확도: 5.38675570395103%
[2]. 셈여림 유사도: 22.84%
[3]. 셈여림 변화 일관성: 22.84%
[11]. 페달링 일관성: 0.00%

IV 기타

1. 전달 사항
   - 상세하게 작성하느라 일부러 길게 풀어서 작성
   - 요구 사항을 바탕으로 ① 평가 기준 + ② 특징 추출 한 부분에 대해서 기술
2. 현재 (24.02.29) 까지 진행 사항
   - 평가 기준 12가지 중 (5), (6), (8) 은 아직 비교 알고리즘을 개발하지 못하여, 측정 불가

	평가 기준	매트릭
1	음정	note
2	셈여림	decibel
3	셈여림의 변화	change of decibel
4	빠르기	speed
5	빠르기의 변화	change of speed
6	붙임줄, 스타카토, 테누토, 늘임표	duration of note
7	악센트	decibel
8	옥타브	note
9	꾸밈음, 반복 기호	note
10	리듬	?
11	페달링	pedaling
12	음의 길이	duration of note

공식화 가능한 기준