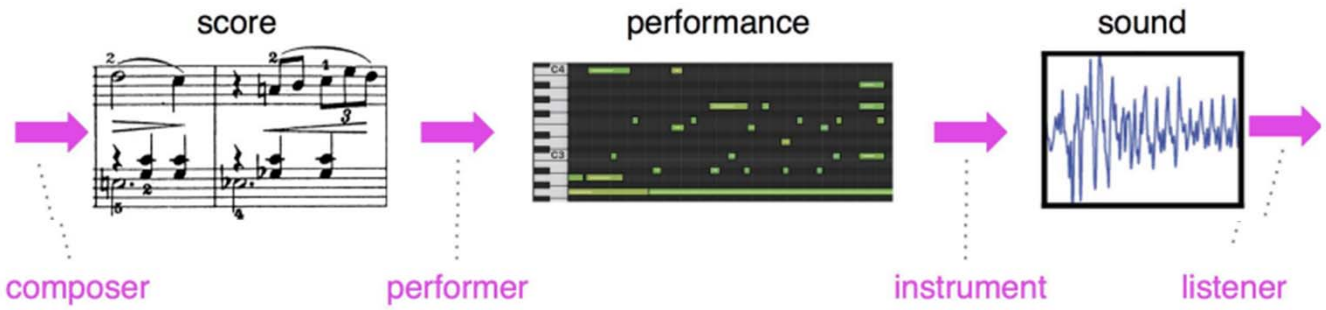
2024 edition

Deep Learning for Music Analysis and Generation

Fundamentals for Symbolic Music

Yi-Hsuan Yang Ph.D.
yhyangtw@ntu.edu.tw

Why Symbolic Music



- There are meaningful problems in symbolic music **analysis** and **generation**
- Reasons to work on symbolic music generation
  - Focus on the musical content instead of sounds
  - Care about long-range structure instead of acoustic quality or diversity
  - Less compute-demanding (compared to text-to-music generation)
  - Good first step toward generative systems for music

Outline

- **Sheet music & symbolic representations for music**
  1. Sheet music
  2. Piano roll
  3. MIDI
  4. MusicXML
  5. ABC
- Token representation of MIDI music
- Symbolic-domain MIR & evaluation metrics

Sheet Music

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1_SheetMusic.html

- A *visual* representation (*.PDF, *.PNG, etc)
  - A **guide** for performing a piece of music leaving room for different interpretations
  - Musicians may vary the **tempo**, **dynamics**, and **articulation**, thus resulting in a personal interpretation of the given musical score

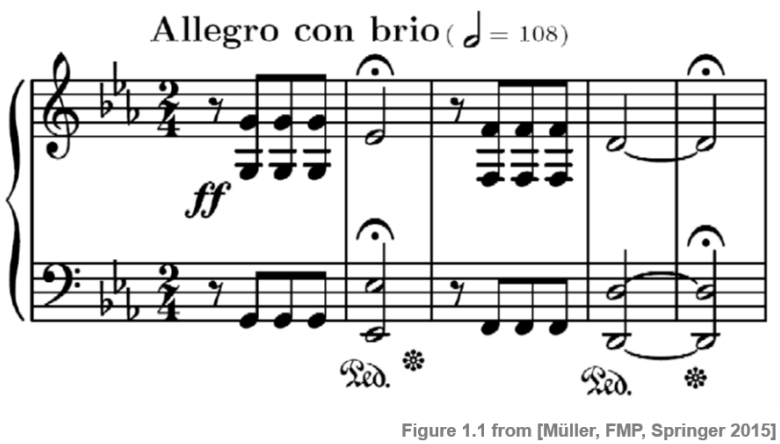
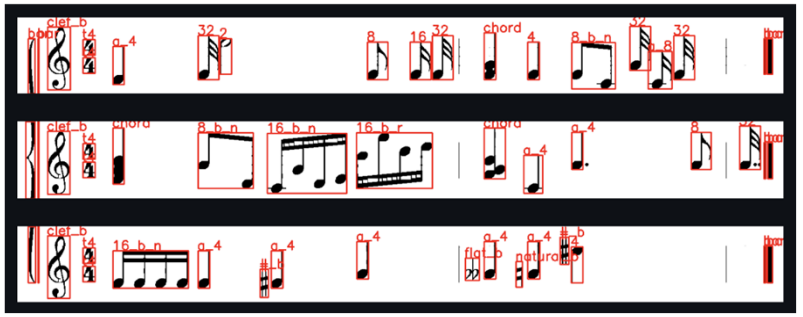


Figure 1.1 from [Müller, FMP, Springer 2015]

- ***Rarely*** directly used as input to neural network models
  - Exception: *optical music recognition* (OMR)

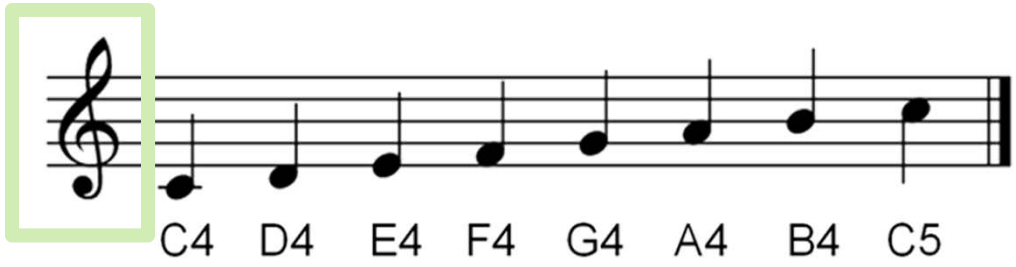


Sheet Music: Key Signature & Note Duration

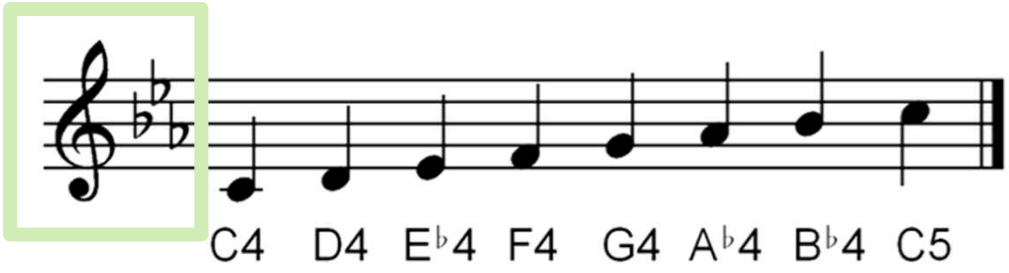
https://nicechord.com/post/major-and-minor-modes/

https://nicechord.com/post/notation-essentials/

C major: 全全半全全全半



C minor:全半全全全半全



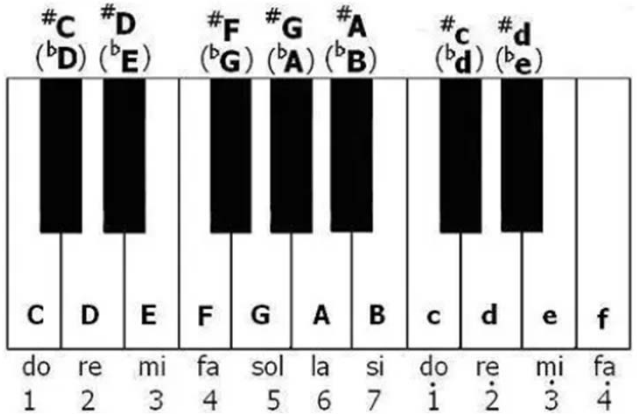
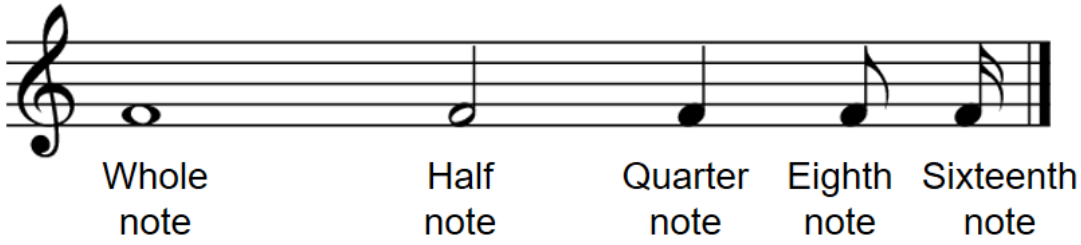


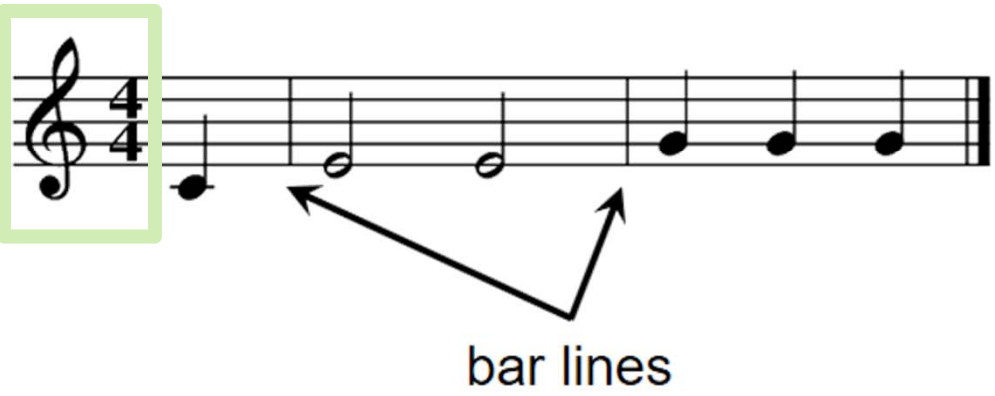
Figure 1.7 from [Müller, FMP, Springer 2015]

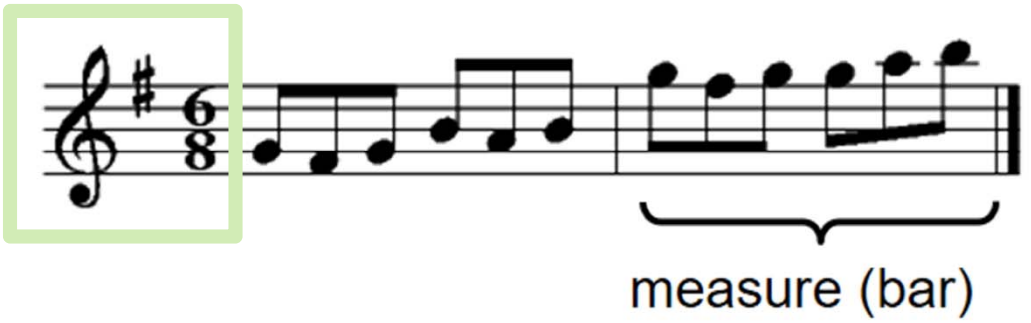


Sheet Music: Meter, Bar, Beat, Rhythm

- Dividing music into measures (bars) not only reflects its rhythmic nature, but also provides regular reference points within it

Figure 1.6 from [Müller, FMP, Springer 2015]





Sheet Music: Articulation Marks

- How certain notes are to be played

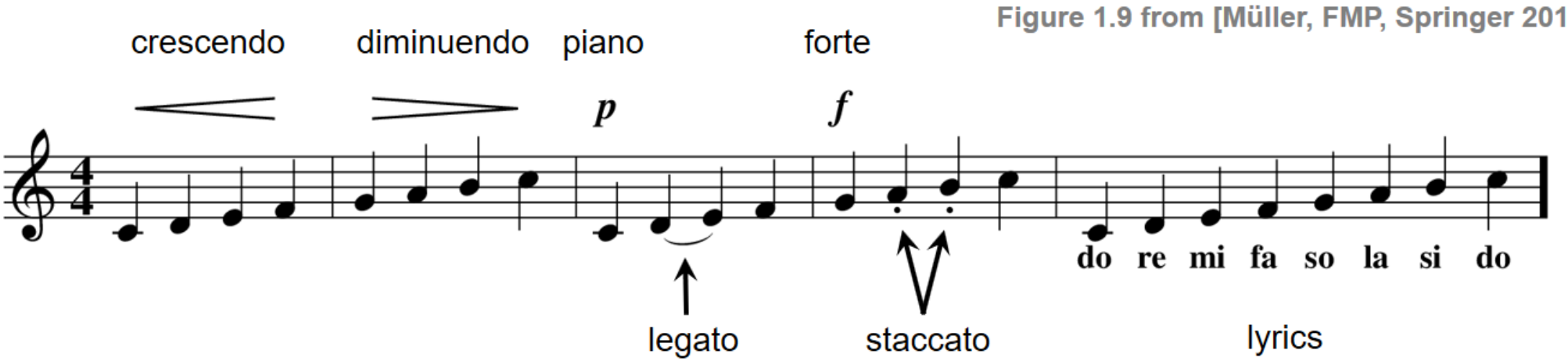
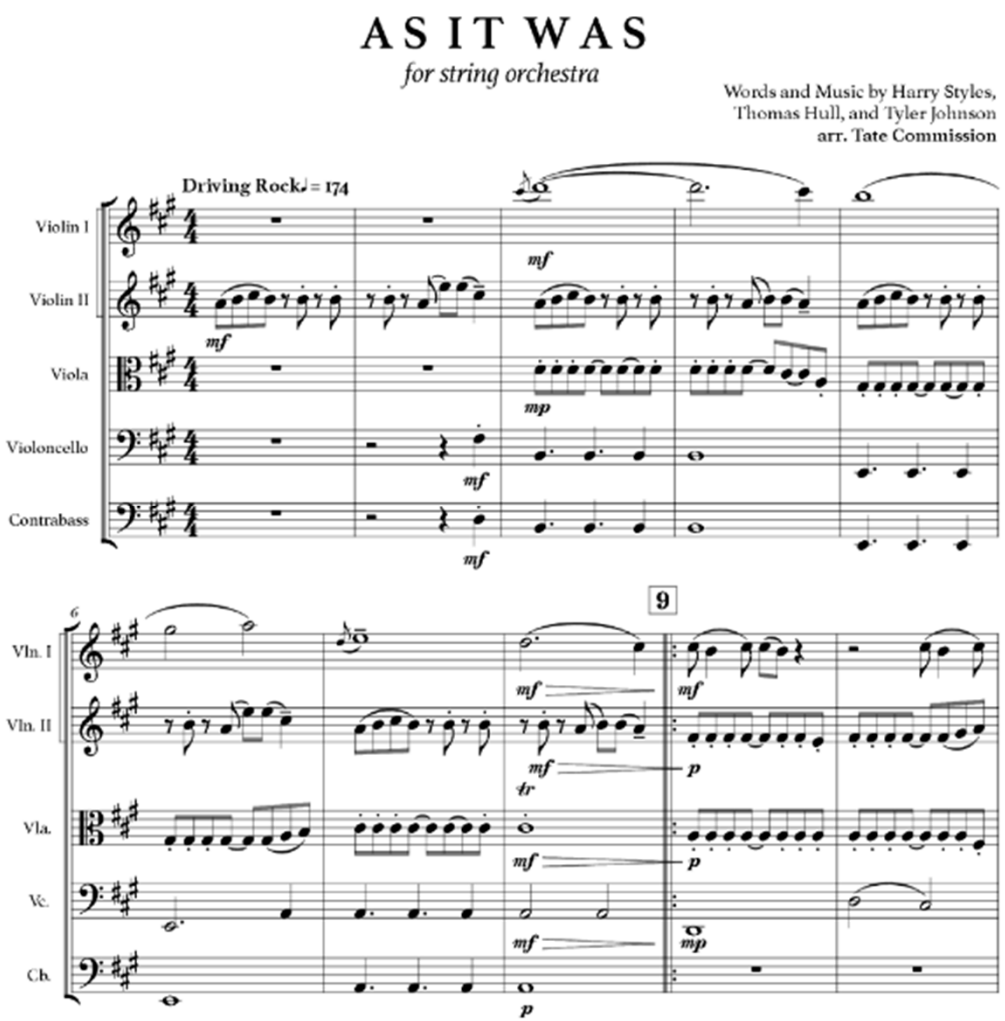
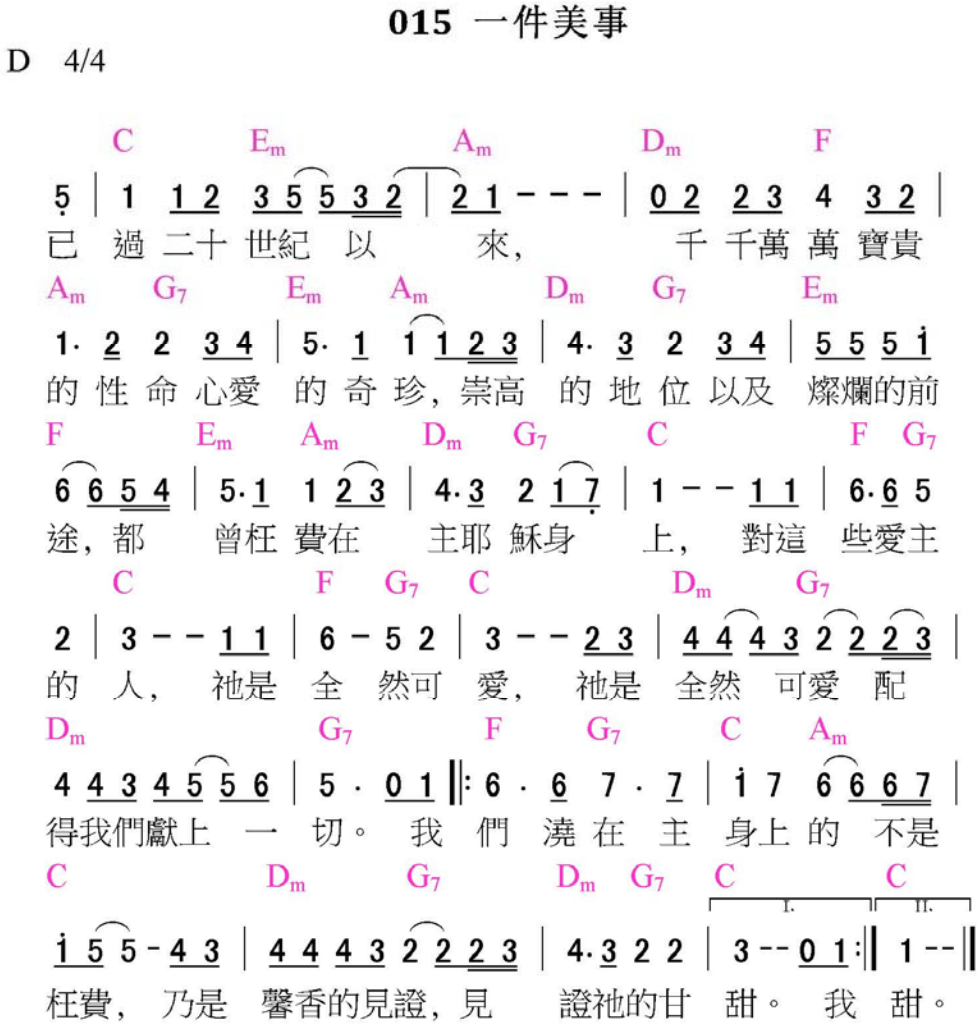


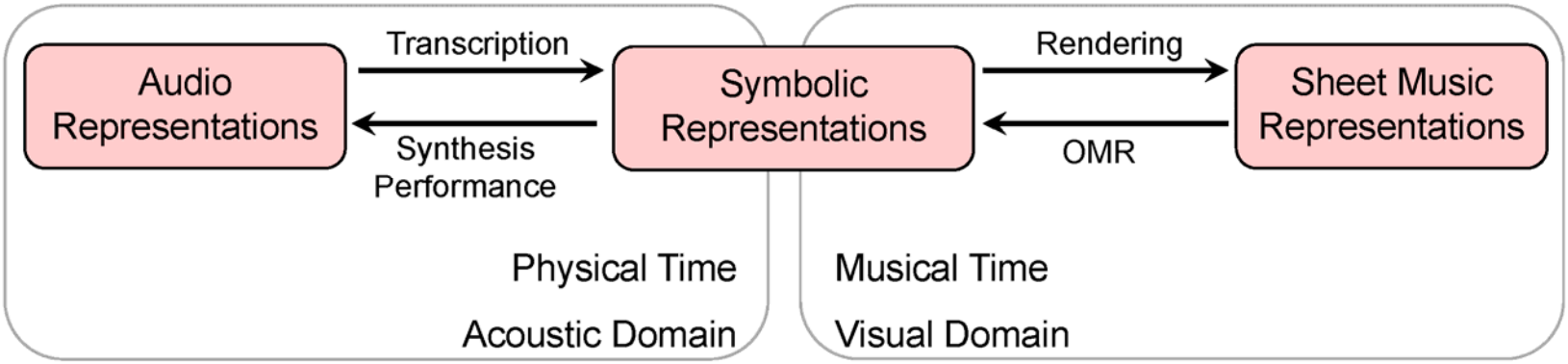
Figure 1.9 from [Müller, FMP, Springer 2015]

Sheet Music vs. Lead Sheet





Sheet Music and Symbolic-domain Representations



- **Musical time:** 1/4, 1/8, 1/16 etc, good for representation **scores**
- **Sheet music**: visual representations of a musical score either given in printed form or encoded digitally in some image format
- **Physical time:** in milliseconds or alike, good for representation **performances** (*tempo*, dynamics, and *articulation*)
- **Symbolic representations**

Ref: https://www.mdpi.com/2624-6120/2/2/18

Piano Roll Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_PianoRoll.html

- A *visual, **image-like*** representation of *.MIDI (also how DAWs visualize MIDI files)
  - Horizontal axis: **Time**
  - Vertical axis: **Pitch**
  - Rectangle: **Note**
    - Leftmost point**: Onset**
    - Lowermost point**: Pitch**
    - Width**: Duration**
    - (Can also indicate **Velocity**)

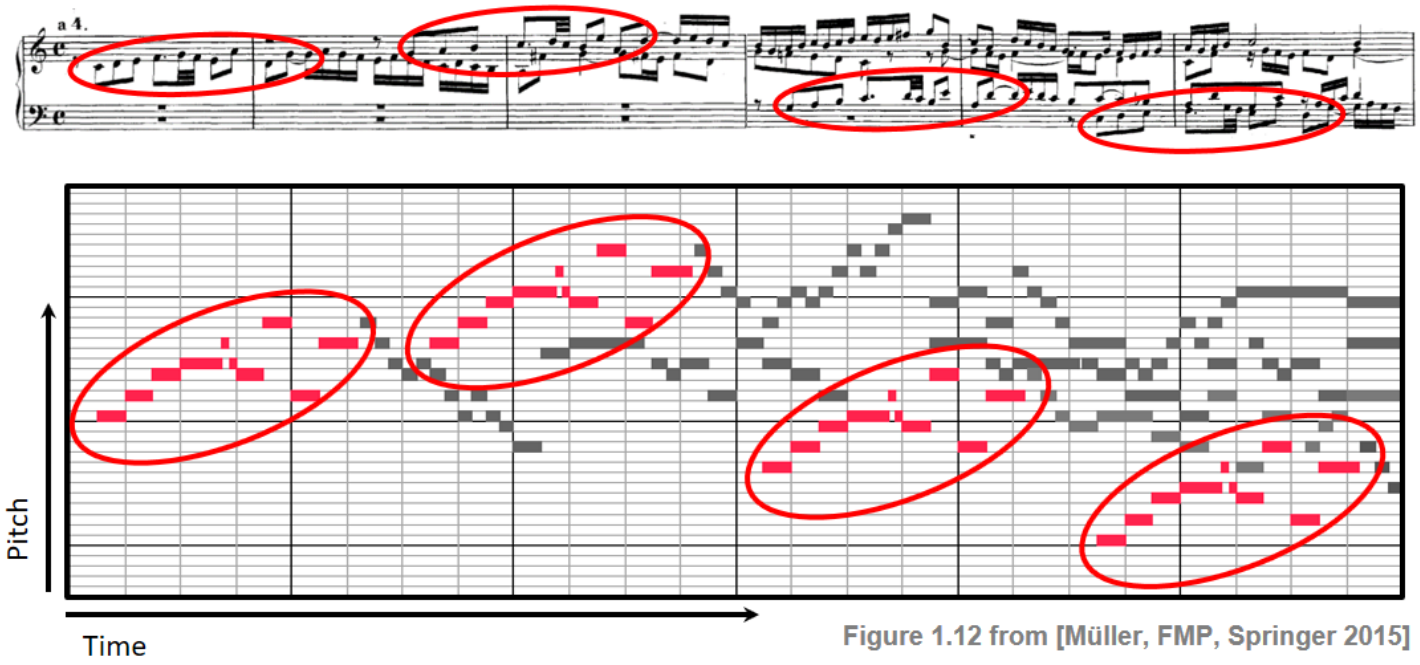
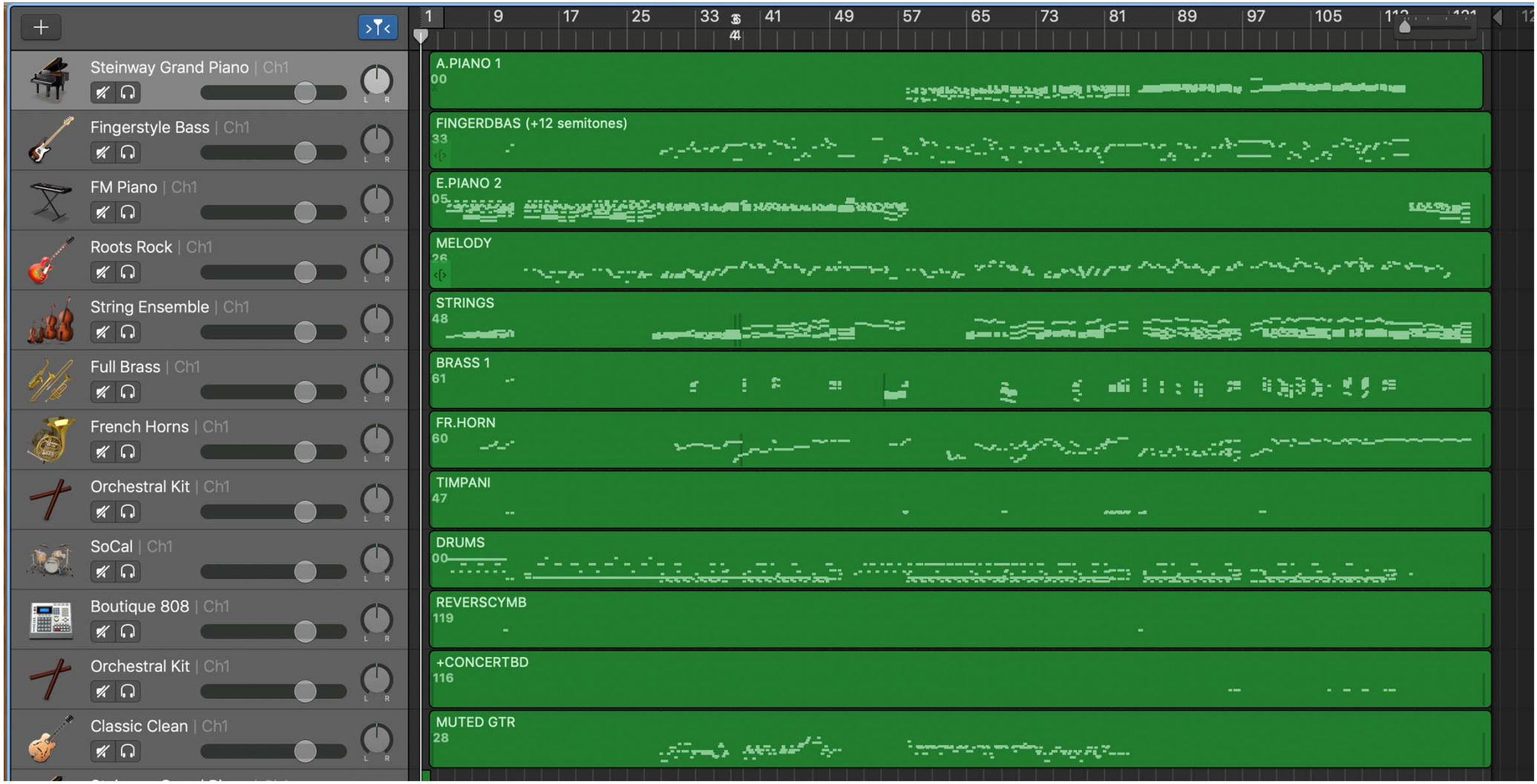


Figure 1.12 from [Müller, FMP, Springer 2015]

ps. The four occurrences of the theme of the music are highlighted

Piano Roll Representation

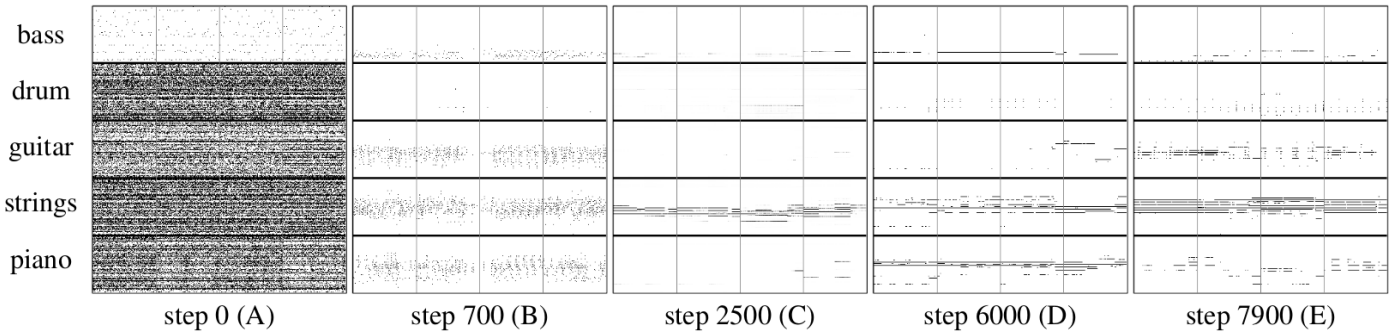
(image from the internet)



Piano Roll Representation

- A *visual,* ***image-like*** representation of *.MIDI
- Multi-track MIDI → multiple piano rolls (i.e., a tensor)
- ***Widely*** used by deep generative neural networks
  - *MidiNet* (2017) [GAN-based]
  - *MuseGAN* (2018) [GAN-based]: https://salu133445.github.io/musegan/results
  - *DiffRoll* (2023) [diffusion-based]: https://polyffusion.github.io/

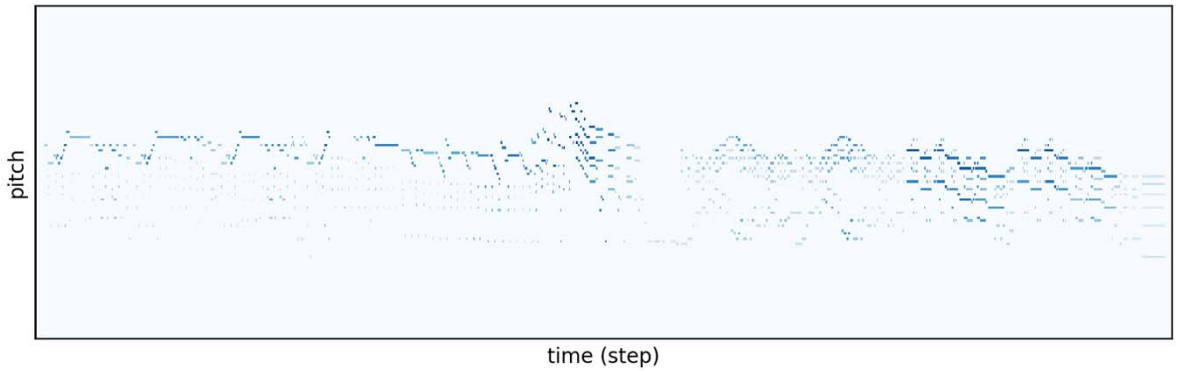
MuseGAN



DiffRoll

Library: PyPianoroll

https://salu133445.github.io/pypianoroll/



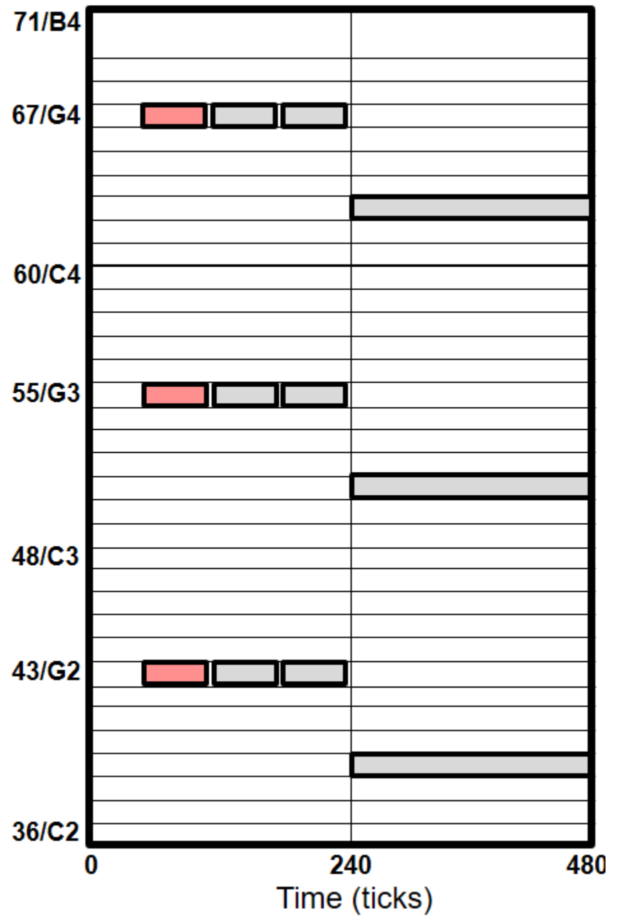
- Features
  - Manipulate multitrack piano rolls intuitively
  - Visualize multitrack piano rolls beautifully
  - Save and load multitrack piano rolls in a space-efficient format
  - Parse **MIDI** files into multitrack piano rolls
  - Write multitrack piano rolls into **MIDI** files

MIDI Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MIDI.html

- A ***text-like*** representation of *.MIDI, using **MIDI messages** and **timestamps**
  - *MIDI note number* (0-127)
  - *Key velocity* (0-127): intensity of the sound
  - *MIDI channel* (different instruments)
  - *Time stamp*: how many *clock pulses or ticks* **to wait** before the command is executed

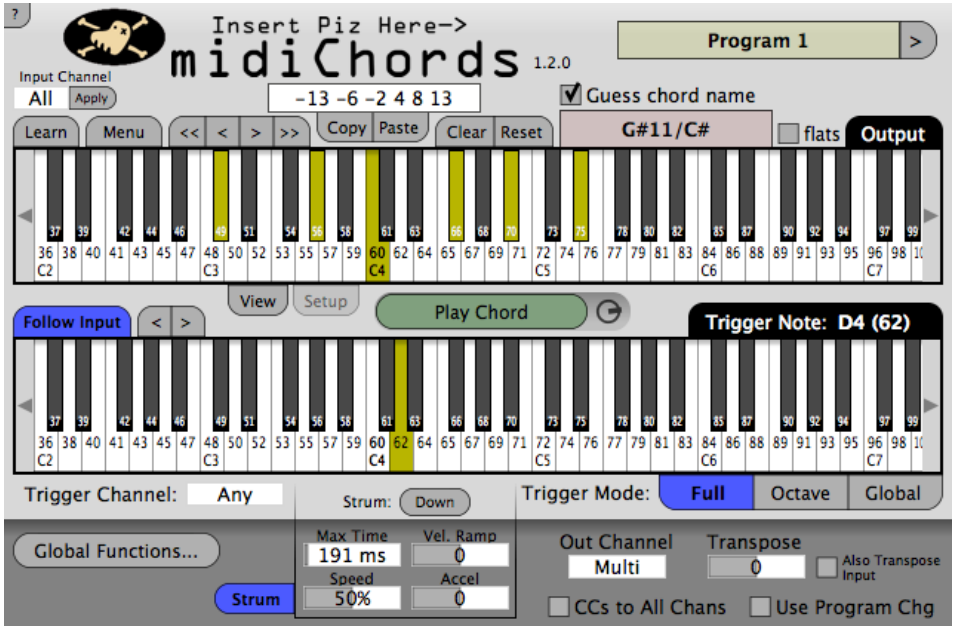
Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0



Piano Roll vs MIDI Representation

- Both represent a *.MIDI file
  - Piano roll: image-like
  - MIDI Representation: text-like

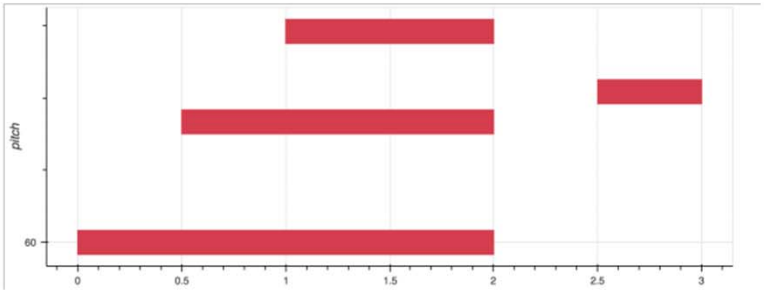
- Piano roll are like fixed-shape images
- MIDI representation is more flexible as it allows for *additional information* to be added
  - For example, **chord tokens, key tokens**, etc



Ref: Figure from https://www.kvraudio.com/product/midichords-by-insert-piz-here

MIDI Representation

- A ***text-like*** representation of *.MIDI
- ***Widely*** used by deep generative neural networks, especially Transformers
  - By viewing the MIDI messages as **"tokens"**
  - *Music Transformer* (2019) : https://magenta.tensorflow.org/music-transformer
  - *MuseNet* (2019): https://openai.com/research/musenet
  - *Pop Music Transformer* (2020): https://github.com/YatingMusic/remi
  - *MuseCoco* (2023): https://ai-muzic.github.io/musecoco/




```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```

Timing in Piano Roll & MIDI

- The time axis can either be in *absolute timing* or in *symbolic timing*
  - For **absolute** timing, the actual timing of note occurrence is used, for example by indicating the tempo for each bar or each beat
  - For **symbolic** timing, the **tempo** information is removed and thereby each beat has the same length

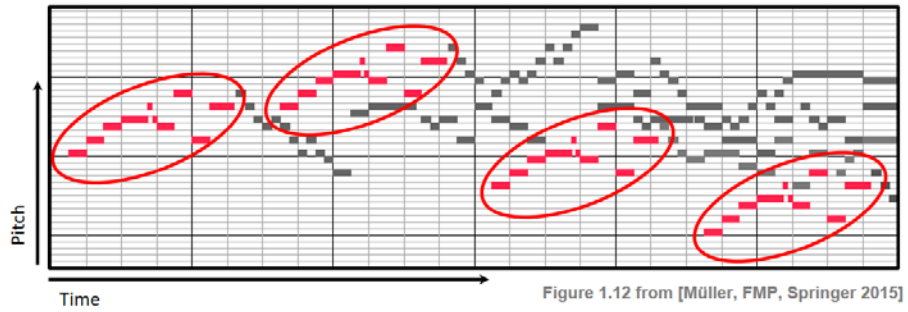
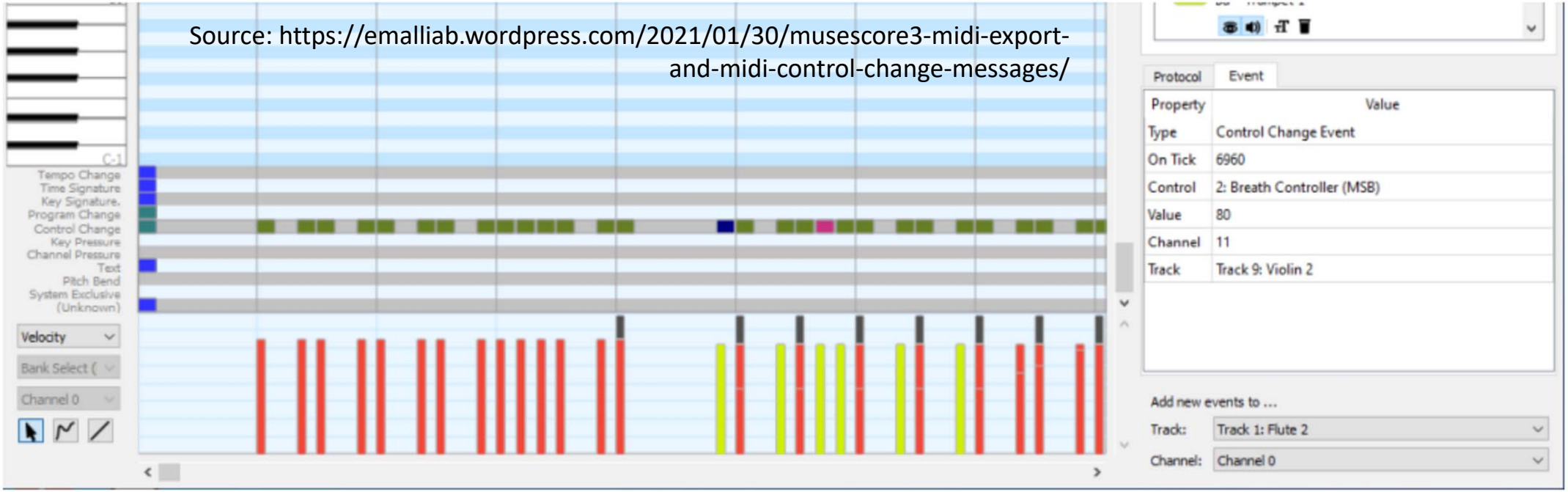


Figure 1.12 from [Müller, FMP, Springer 2015]

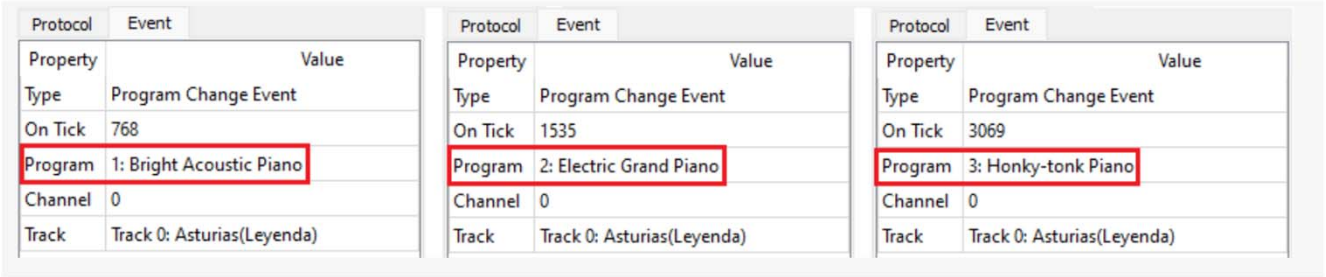
- MIDI subdivides a **quarter note** into basic time units referred to as *clock pulses* or *ticks*
  - Pulses per quarter note (PPQN): commonly 120
  - PPQN determines the resolution of the time stamps associated to note events

Ref1: https://salu133445.github.io/lakh-pianoroll-dataset/representation.html
Ref2: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MIDI.html

Program Change or Control Change (CC) Messages in MIDI



- Tempo change
- Program change
- Control change



Source: https://gigperformer.com/how-to-automate-switching-rackspace-variations-and-song-parts-using-the-midi-file-player/

MIDI Score vs MIDI Performance

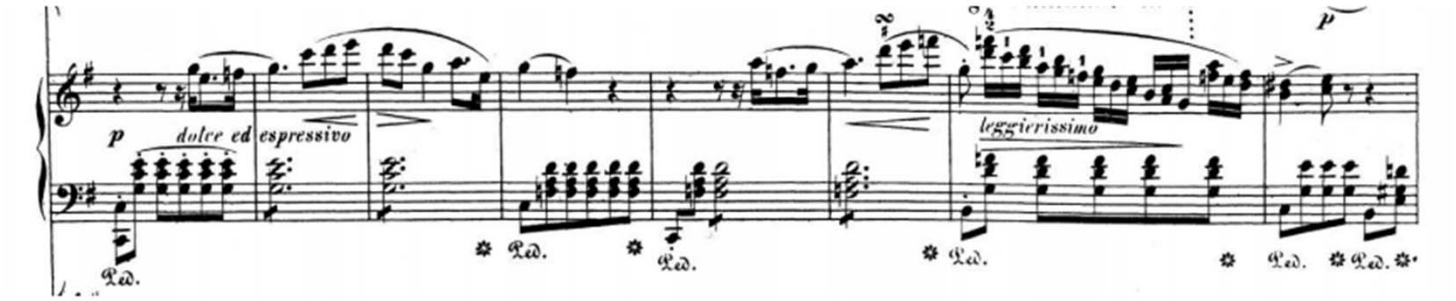
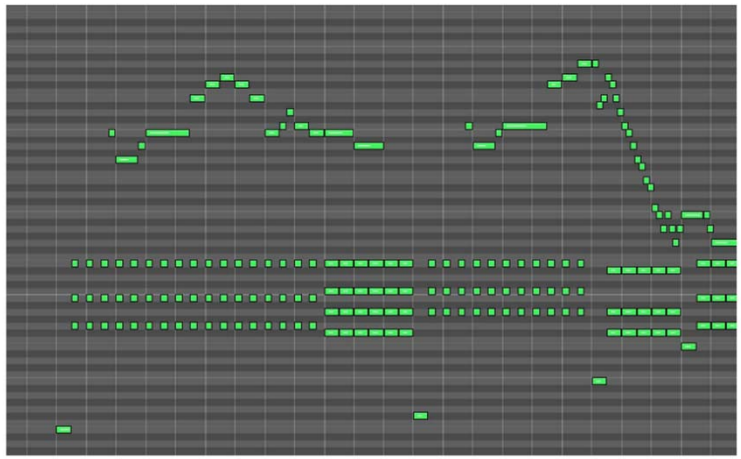


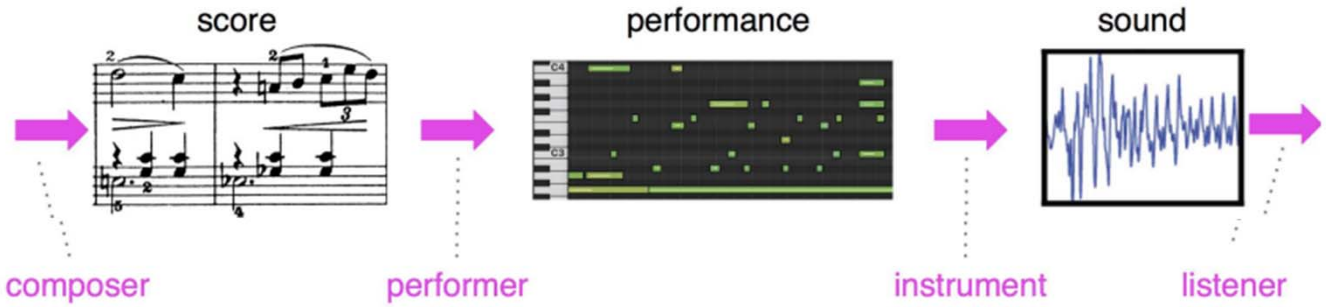
Figure 1: Excerpt from the score of Chopin's Piano Concerto No. 1.



- MIDI score
  - https://clyp.it/jhdkghso
- MIDI performance
  - https://clyp.it/x24hp1pq

Ref: Oore et al, "This time with feeling: Learning expressive musical performance," arXiv 2018

MIDI Score vs MIDI Performance



- **MIDI score**
  - A score that has been rendered **directly** into a MIDI file
  - Rendered with **NO** dynamics and **NO** expressive timing (i.e., exactly according to the written metrical grid)
- **MIDI performance**
  - A score has been performed, and that performance has been encoded into a MIDI stream, through either **MIDI keyboards** or by **automatic music transcription**
  - With expressive timing, tempo changes, dynamics, and articulation

Ref: Oore et al, "This time with feeling: Learning expressive musical performance," arXiv 2018

Library: Miditoolkit

https://github.com/YatingMusic/miditoolkit

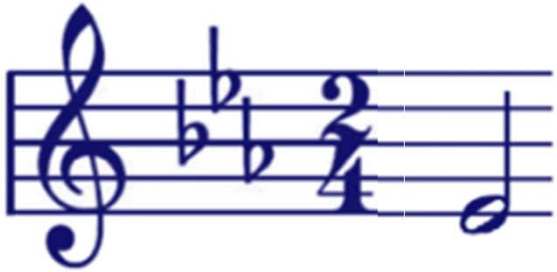
- **Miditoolkit** is designed for handling MIDI in **symbolic timing** (ticks), which is the native format of MIDI timing
- **PrettyMIDI** (https://github.com/craffel/pretty-midi) can parse MIDI files and generate pianorolls in absolute timing
- **PyPianoroll** can parse MIDI files into pianorolls in symbolic timing
- **mido** (https://github.com/mido/mido) processes MIDI files in the lower level such as messages and ports
- **music21** (https://web.mit.edu/music21/) provides analysis modules

MusicXML

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MusicXML.html

- A ***text-like*** representation of **sheet music**
  - To represent a <note>
    - <pitch>: <step>, <alter> <octave>
    - <duration>
    - <type>
- Can be **tokenized**
- Musical (not physical) onset times are specified

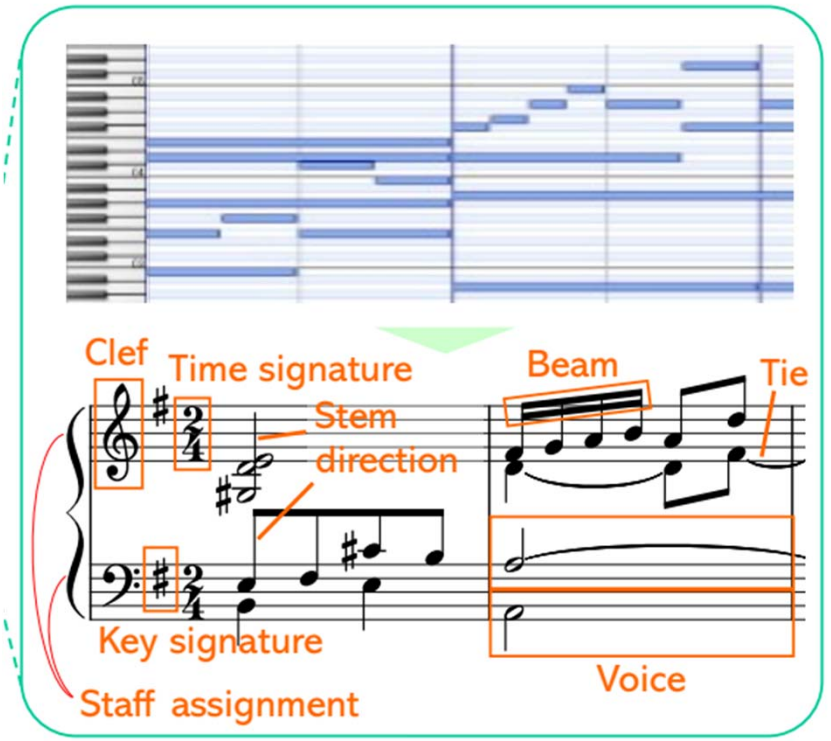
```
<note>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>2</duration>
  <type>half</type>
</note>
```



- Avoid information loss of the piano roll representation
  - Example 1, the notes **D♯4** and **E♭4** are distinguishable in MusicXML but not in MIDI

MusicXML

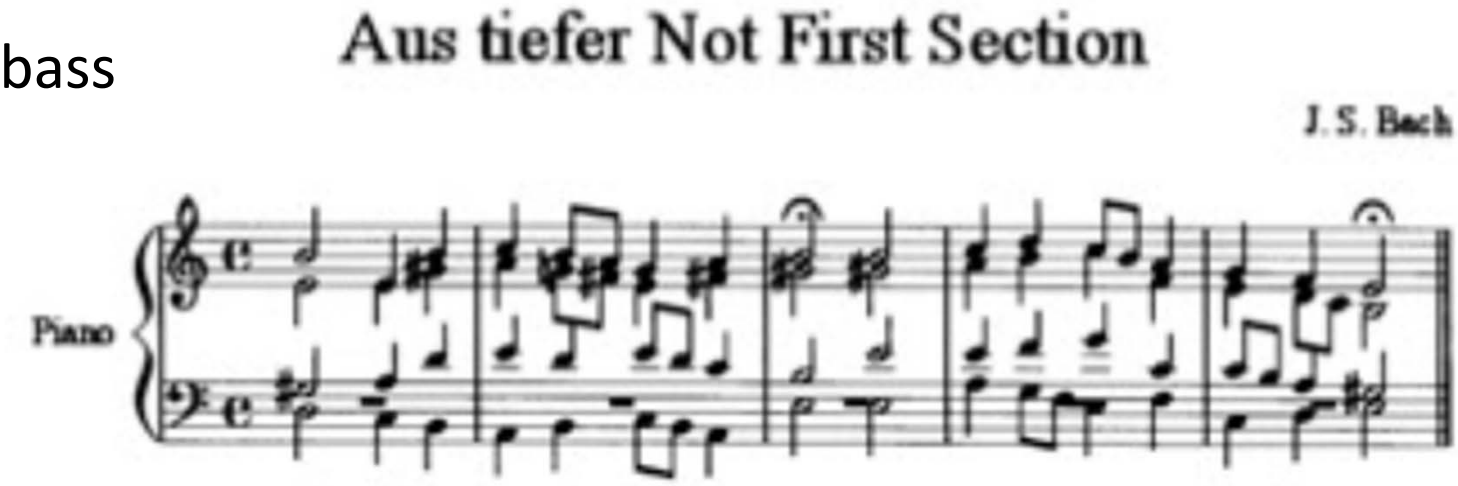
- A ***text-like*** representation of **sheet music**
- Readily tokenized
- Musical (not physical) onset times
- Avoid information loss of the piano roll representation
  - Example 1, D♯4 and E♭4 (enharmonic equivalents;同音異名)
    - https://www.youtube.com/watch?v=Sm-2e2DsdhE
  - Example 2: can describe **voices**



Ref: Suzuki, “Score Transformer: Generating musical score from note-level representation,” MMAsia 2021

Voices

- SATB
  - S: soprano
  - A: alto
  - T: tenor
  - B: bass

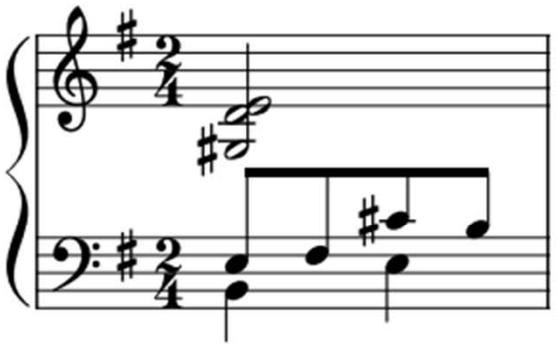


Voice type
Female
Soprano
Mezzo-soprano
Contralto
Male
Countertenor
Tenor
Baritone
Bass

https://www.youtube.com/watch?v=WD0jd1QcMQE

MusicXML

- A ***text-like*** representation of **sheet music**
- *Relatively less* used by deep generative neural networks
  - *Score Transformer* (2021)



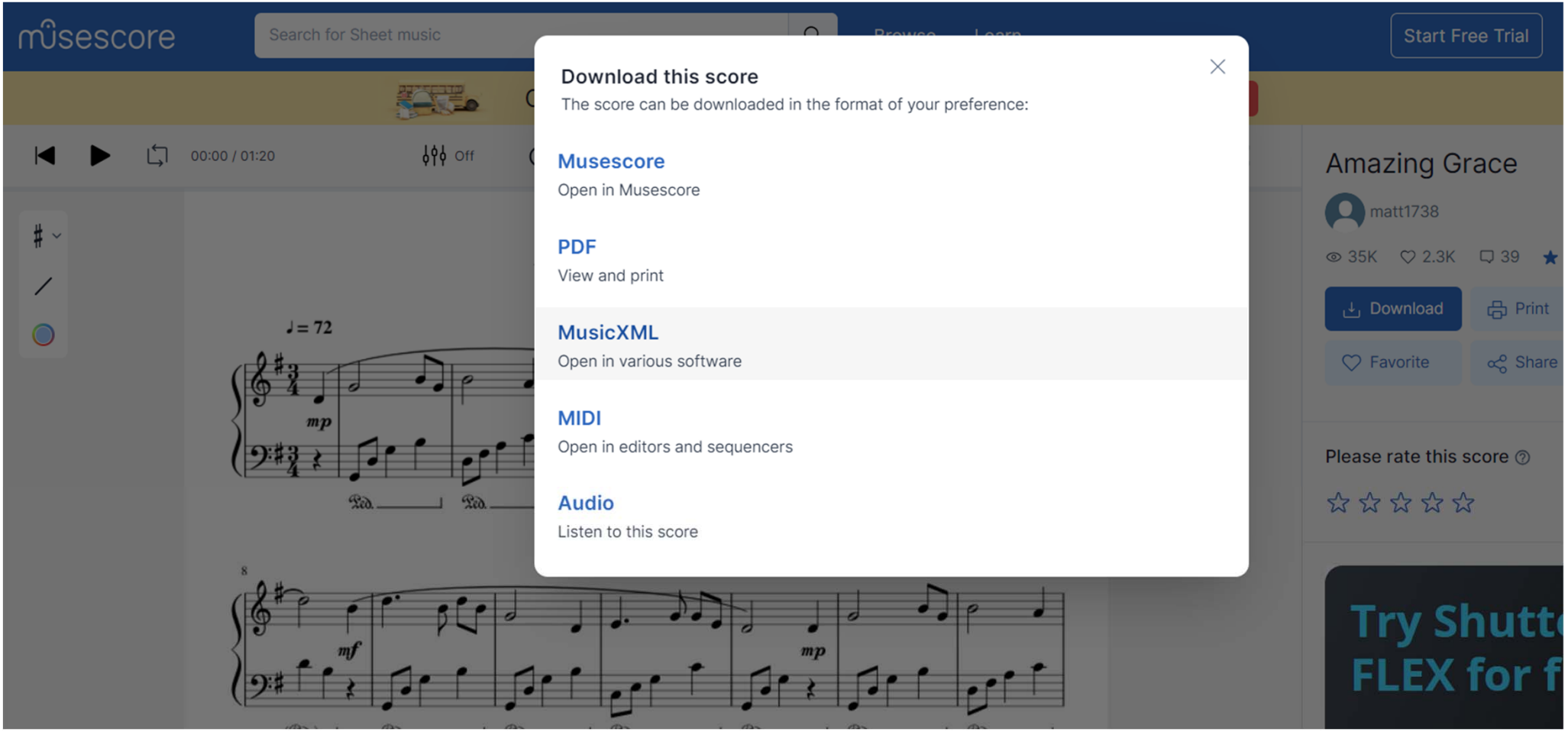
(a) Score

R bar clef_treble key_sharp_1 time_2/4 note_E4 note_D4 note_G#3 len_2 stem_up bar ... L bar clef_bass key_sharp_1 time 2/4 <voice> note_E3 len_1/2 stem_up beam_start note_F#3 len_1/2 stem_up beam_continue note_C#4 stem_up beam_continue note_B3 len_1/2 stem_up beam_stop </voice> <voice> note_B2 len_1 stem_down note_E3 len_1 stem_down </voice> bar ...

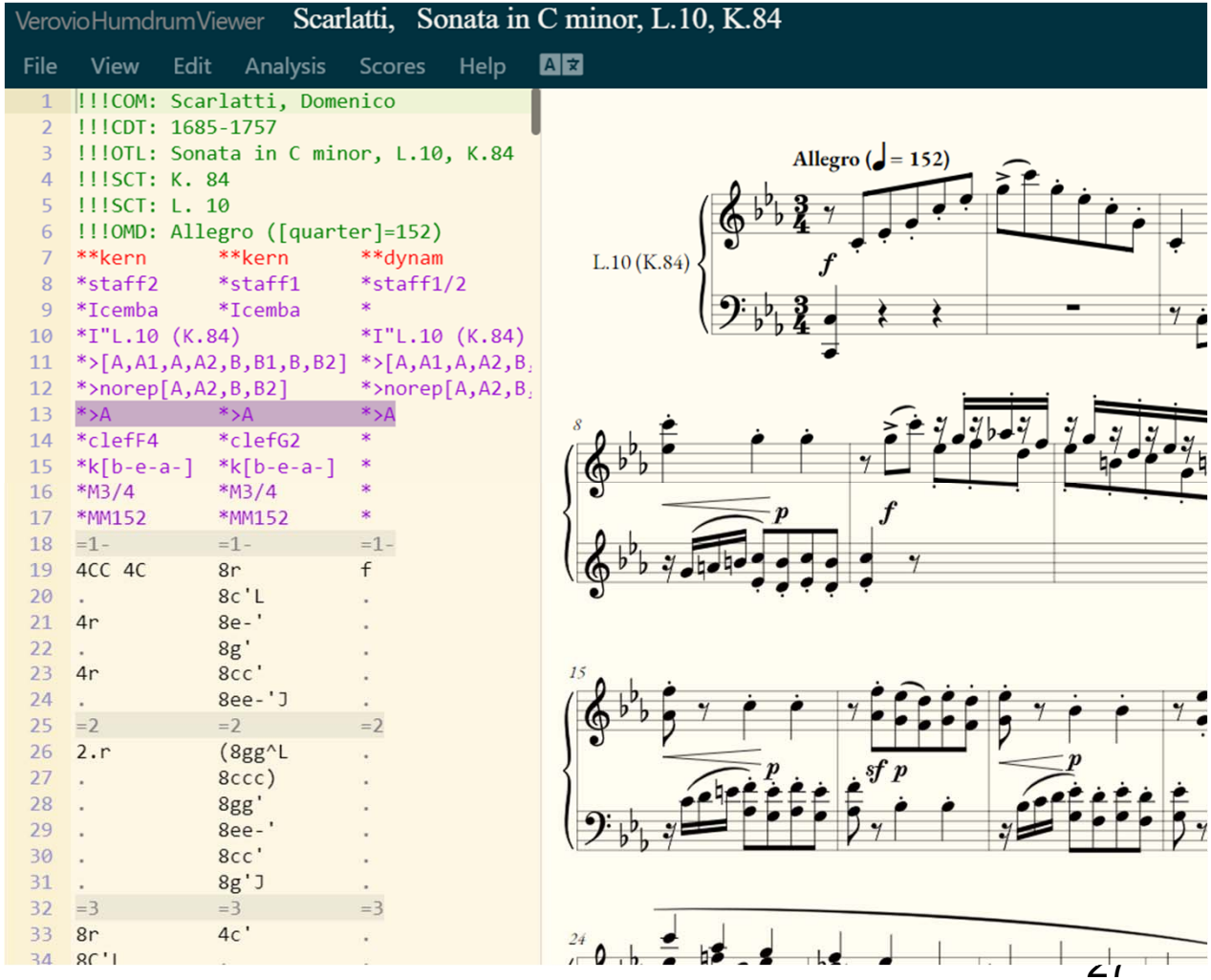
(b) Notation-level Representation

Ref: Suzuki, “Score Transformer: Generating musical score from note-level representation,” MMAsia 2021

Software: MuseScore

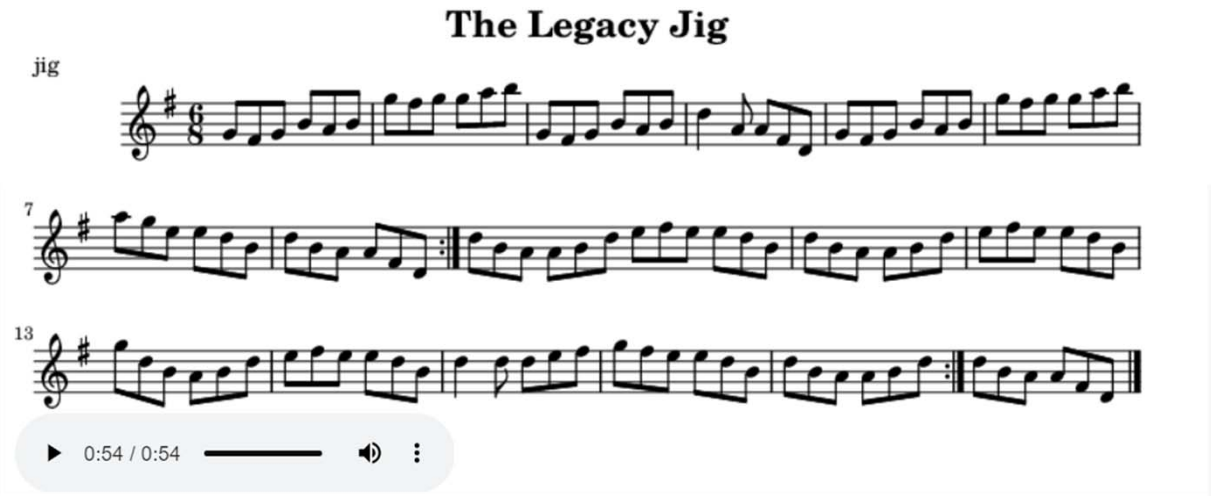


Software: KernScores



ABC

https://en.wikipedia.org/wiki/ABC_notation




```
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:G
GFG BAB | gfg gab | GFG BAB | d2A AFD |
GFG BAB | gfg gab | age edB |1 dBA AFD :|2 dBA ABd |:
efe edB | dBA ABd | efe edB | gdB ABd |
efe edB | d2d def | gfe edB |1 dBA ABd :|2 dBA AFD |]
</score>
```

ABC

https://abcnotation.com/examples

```
C, D, E, F, | G, A, B, C | D E F G | A B c d | e f g a | b c' d' e' | f' g' a' b' |
```

```
A/4 A/2 A/ A A2 A3 A4 A6 A7 A8 A12 A16
```

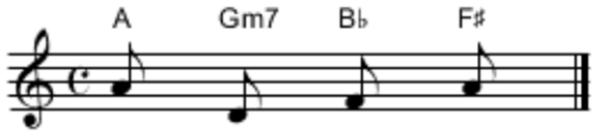
```
A>A A2>A2 | A<A A2<A2 | A>>A A2>>>A2 | A<<A A2<<<A2
```

```
(2AB (3ABA (4ABAB (5ABABA (6ABABAB (7ABABABA|
```

```
(AA) (A(A)A) ((AA)A) (A|A) A-A A-A-A A2-|A4|
```

```
__A _A =A ^A ^^A
```

```
"A"A "Gm7"D "Bb"F "F#"A
```

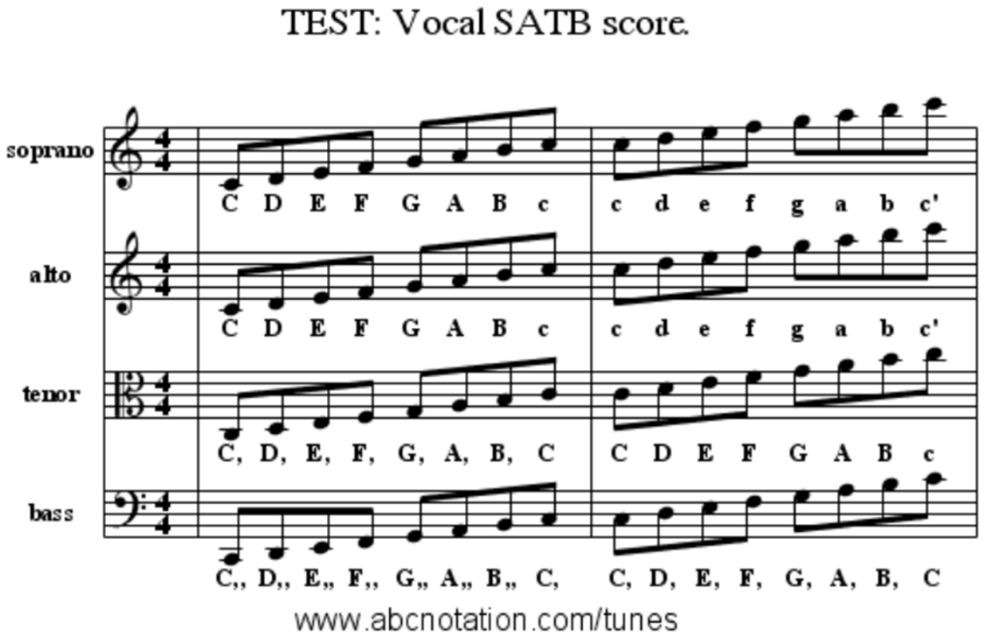



```
{g}A3 A{g}AA|{gAGAG}A3 {g}A{d}A{e}A|
```

ABC

https://abcnotation.com/tunePage?a=trillian.mit.edu/~jc/music/abc/src/jcabc2ps-20090401/abc/V_SAT1B2/0000

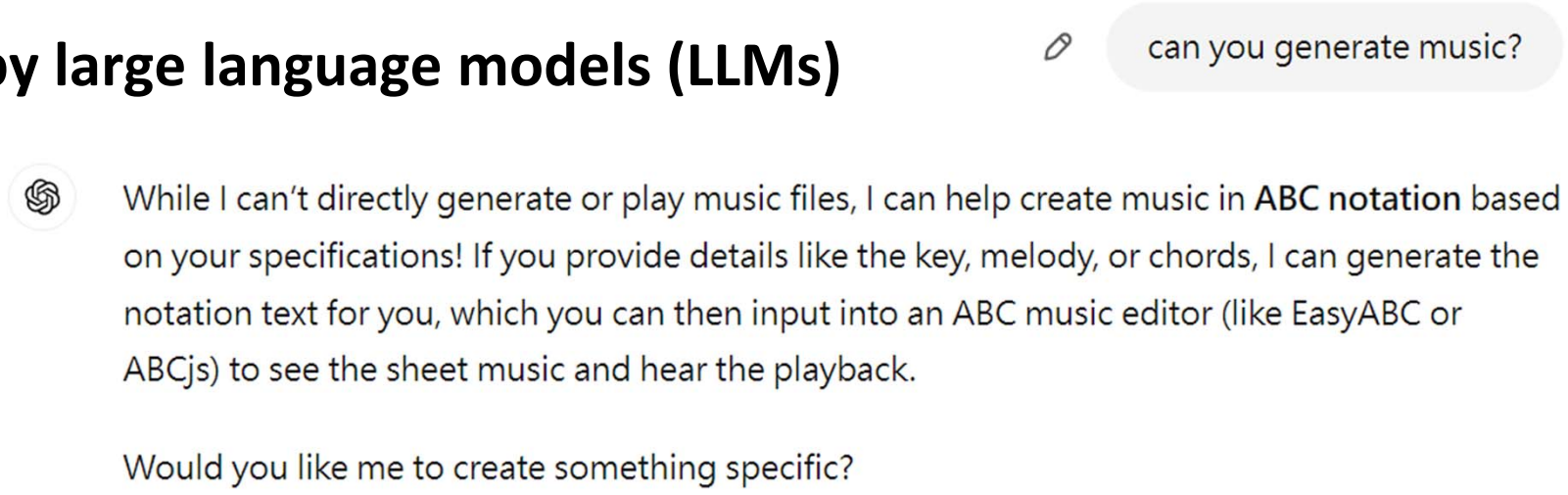
- Can support multiple voices



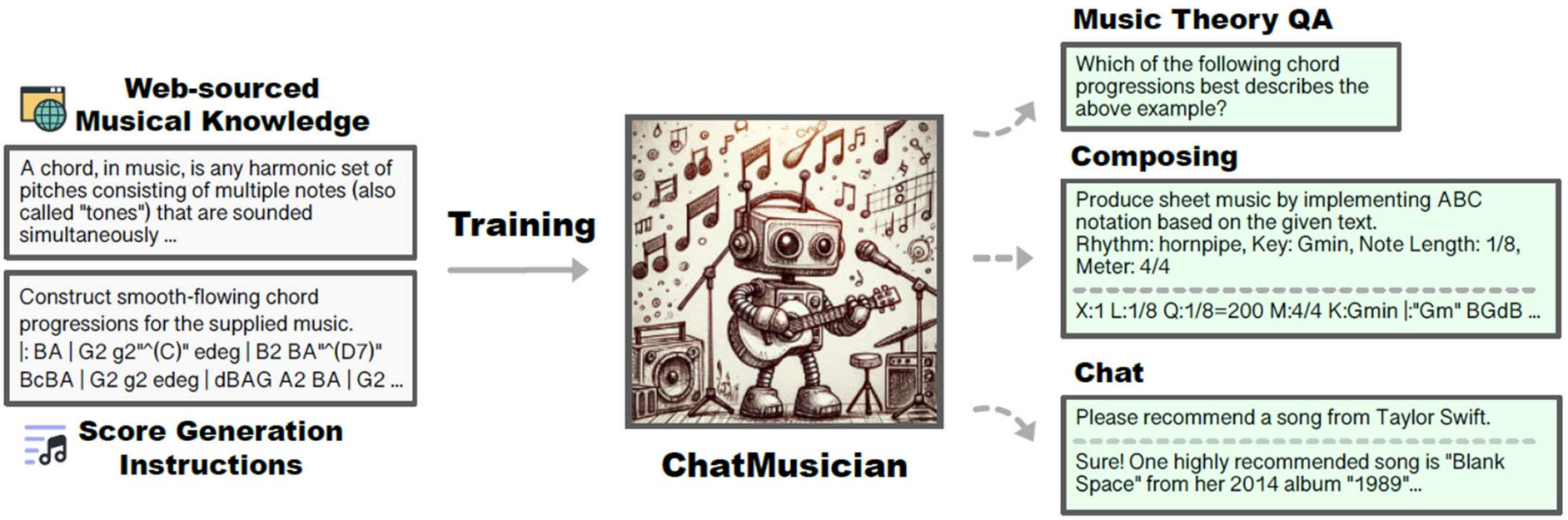

```
X:1
T: TEST: Vocal SATB score.
N: All parts written "on the staff" with commas.
N: All parts should show scale from C below to above staff.
K: C
V:S name="soprano" clef=treble   middle=B
V:A name="alto"    clef=treble   middle=B
V:T name="tenor"   clef=tenor    middle=B,
V:B name="bass"    clef=bass     middle=D,
%
V:S
| CDEF GABc | cdef gabc' |
w:  C D E F  G A B c |  c d e f  g a b c' |
V:A
| CDEF GABc | cdef gabc' |
w:  C D E F  G A B c |  c d e f  g a b c' |
V:T
| C,D,E,F, G,A,B,C | CDEF GABc |
w:  C, D, E, F,  G, A, B, C |  C D E F  G A B c |
V:B
| C,,D,,E,,F,, G,,A,,B,,C, | C,D,E,F, G,A,B,C |
w:  C,, D,, E,, F,,  G,, A,, B,, C, |  C, D, E, F,  G, A, B, C |
```

ABC

- A ***text-like*** representation of **sheet music**
- Readily tokenized
- Musical (not physical) onset times
- Can support multiple voices
- Support **score** only, not performance
- **Can be used by large language models (LLMs)**



LLM ♡ ABC



ChatMusician: Understanding and generating music intrinsically with LLM, arXiv 2024

https://github.com/hf-lin/ChatMusician

Question: "Which of the following statements about triads is correct?",
Options:
A: "A triad can only be composed of three notes.",
B: "All triads are consonant chords.",
C: "When a triad is inverted, its properties and consonance remain consistent with its original position.",
D: "A triad is defined as a chord formed by stacking three notes in a third relationship."},
Answer: D

Library: EasyABC

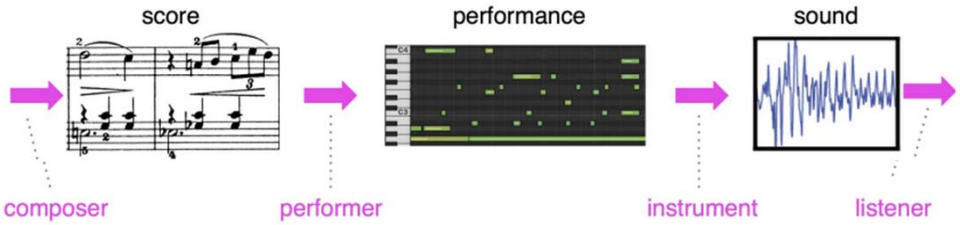
https://github.com/jwdj/EasyABC/

EasyABC / midi2abc.py

Features

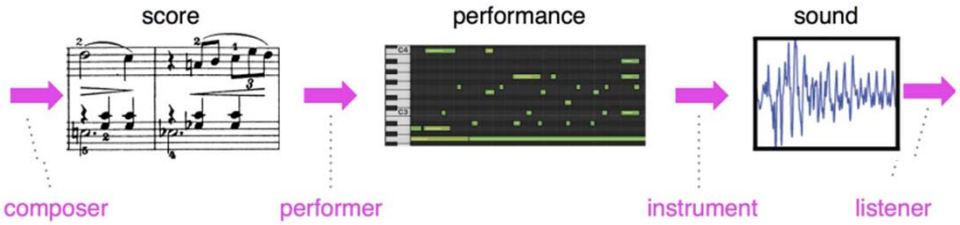
- Good ABC standard coverage thanks to internal use of abcm2ps and abc2midi
- Syntax highlighting
- Zoom support
- Import MusicXML, MIDI and Noteworthy Composer files.
- Export to MIDI, SVG, PDF (single tune or whole tune book).
- Select notes by clicking on them and add music symbols by using drop-down menus in the toolbar.
- Play the active tune as midi (using SoundFont)
- See which notes are currently playing (Follow score)

Summary



- **Sheet Music / MusicXML / ABC**
  - Support **score** only, not performance
  - Use **music timing** only, not absolute timing
  - Richer information about the score itself
- **Piano roll / MIDI**
  - Support either score or **performance**
  - Use either music timing (after quantization) or **absolute timing**
  - Rich performance information
  - Datasets more easily accessible and thereby **more widely used**

Summary (Cont')



- **ABC**
  - Good for simple melodies, chords, and basic arrangements
  - Limited for more complicated piano scores
  - Good for LLMs (but can only do composition; not performance)
- **Piano roll / MIDI / MusicXML**
  - More flexible

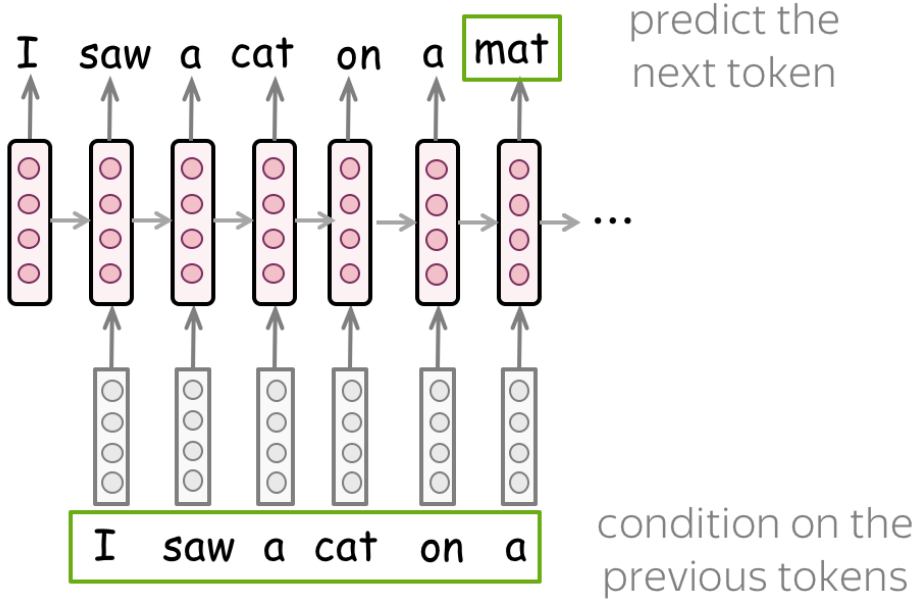
Outline

- Sheet music & symbolic representations for music
- **Token representation of MIDI music**
- Symbolic-domain MIR & evaluation metrics

Text Tokens

https://wikidocs.net/178448

- Token
  - Words, character sets, or combinations of words
  - The **smallest unit of text data**
- First step in training LMs
- Two factors
  - Number of unique tokens (vocabulary size)
  - Length of a token sequence



Text Tokens

- Different types of text tokenizers
  - **Word**-based tokenizers `["Let", "us", "learn", "tokenization."]`
    - Large vocabulary size
    - May have out-of-vocabulary (OOV; unknown) tokens
  - **Character**-based tokenizers `["L", "e", "t", "u", "s", "l", "e", "a", "r", "n", "t", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n", "."]`
    - Small vocabulary size
    - No OOV
    - Long token sequence
  - **Sub-word** tokenizers (e.g., byte-pair encoding) `["Let", "us", "learn", "token", "ization."]`
    - Middle ground

Ref 1: https://medium.com/@nimritakoul01/tokenizers-for-natural-language-transformer-models-simply-explained-ee7167a844da
Ref 2: https://blog.csdn.net/zhaohongfei_358/article/details/123379481

Text Tokens

https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

https://platform.openai.com/tokenizer

```
My favorite color is red.
```

```
[3666, 4004, 3124, 318, 2266, 13]
```

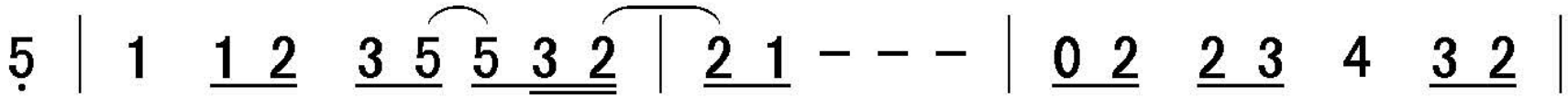
Observations:

The more probable/frequent a token is, the lower the token number assigned to it:

- The token generated for 'red' varies depending on its placement within the sentence:
  - Lowercase in the middle of a sentence: ' red' - (token: "2266")
  - Uppercase in the middle of a sentence: ' Red' - (token: "2297")
  - Uppercase at the beginning of a sentence: 'Red' - (token: "7738")

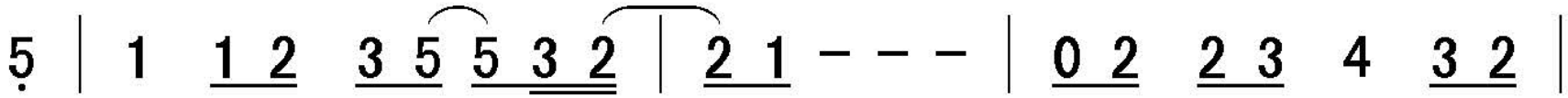
Representing Symbolic Music as Tokens

- How to represent a **melody** sequence?



Representing Symbolic Music as Tokens

- How to represent a **melody** sequence?

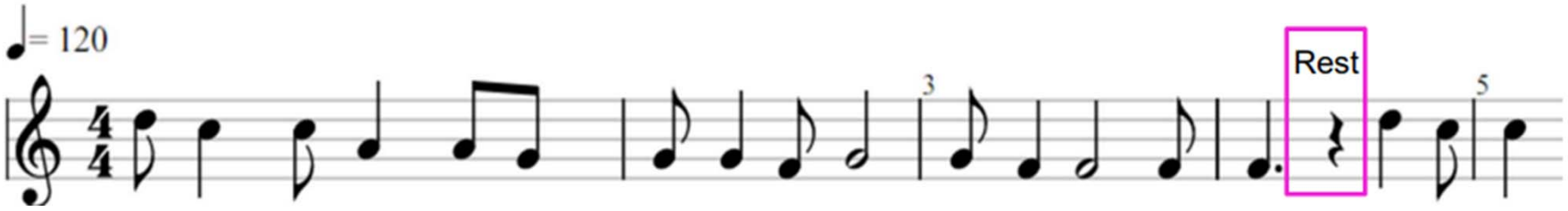


- [pitch] [duration] [pitch] [duration] [pitch] [duration] [pitch] [duration] …
- [pitch=G3] [duration=¼] [pitch=C4] [duration=¼] [pitch=C4] [duration=⅛] …
- [G3] [¼] [C4] [¼] [C4] [⅛] [D4] [⅛] …

- **Each musical note** is uniformly represented by **two tokens**
- **The ending of a note marks the beginning of the next note**

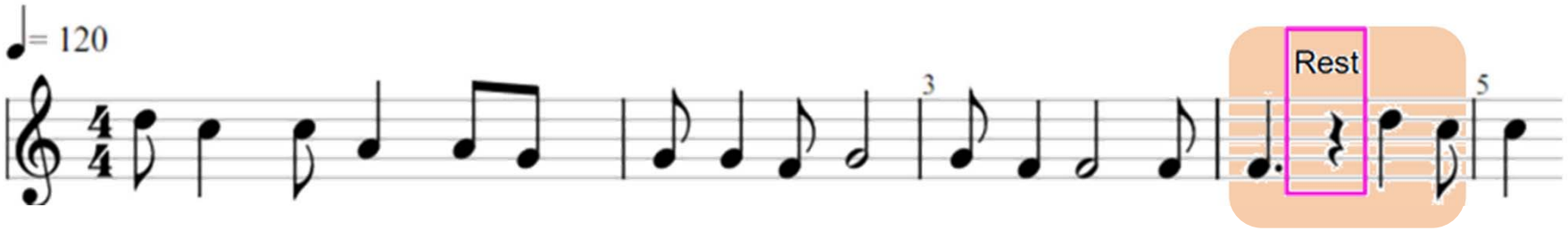
Representing Symbolic Music: Melody

- How to represent a **melody** sequence **with rest**?



Representing Symbolic Music: Melody

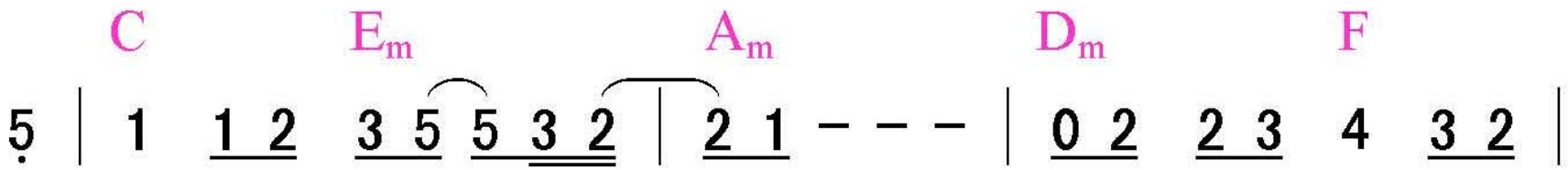
- How to represent a **melody** sequence **with rest**?



- Different methods
  1. [pitch] [duration] [rest-duration] [pitch] [duration] [pitch] [duration]
  2. [pitch] [duration] [pitch=0] [duration] [pitch] [duration] [pitch] [duration]
  3. [pitch] [duration] [rest=¼] [pitch] [duration] [rest=0] [pitch] [duration] [rest=0]

Representing Symbolic Music: Lead Sheet

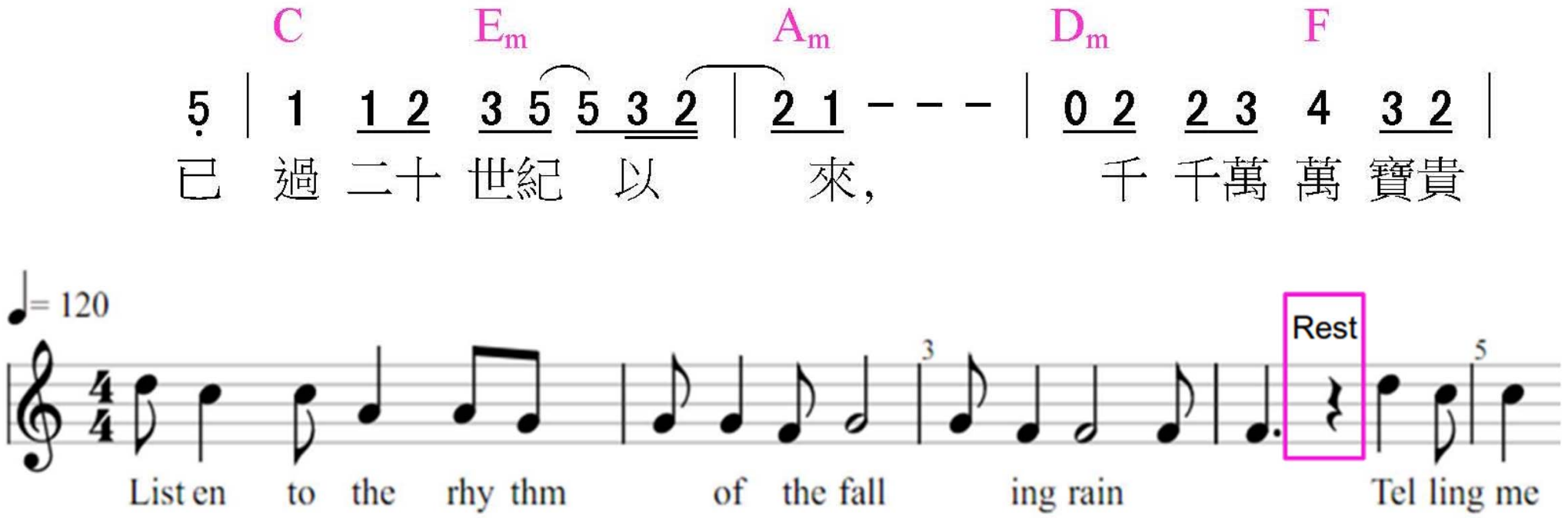
- How to represent a **lead sheet** sequence (i.e., melody+chord)?



- [pitch] [duration] [chord] [pitch] [duration] [pitch] [duration] …
  - Each musical note is still represented by two tokens, but there are *another type of tokens* (not for notes) in the token sequence
  - Getting more complicated

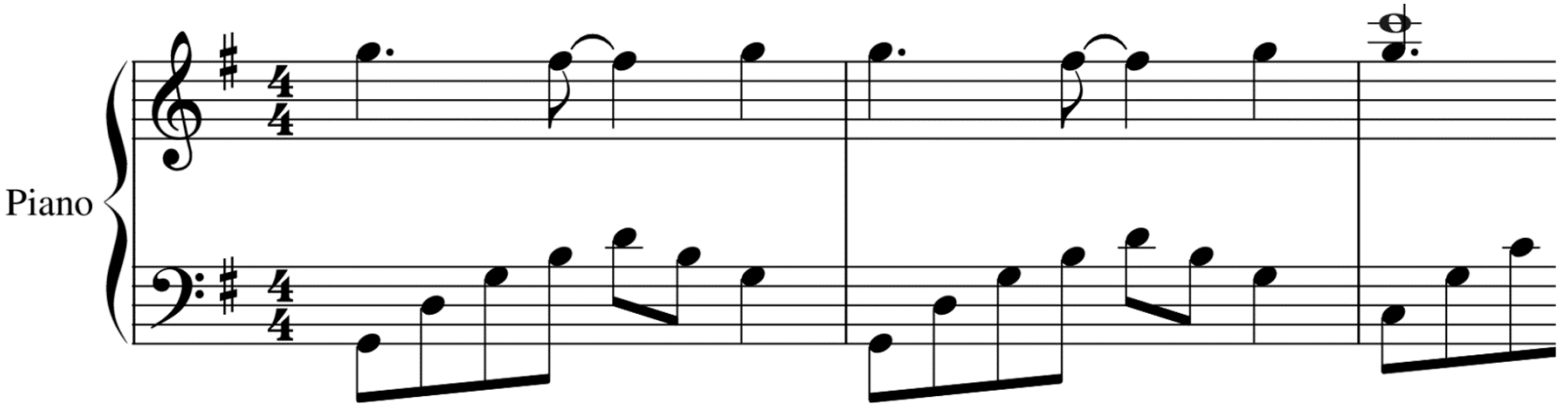
Representing Symbolic Music: Lead Sheet

- How to represent a **lead sheet** sequence (i.e., melody+chord+lyrics)?



Representing Symbolic Music: Piano Music Score

- How to represent a sequence of **piano music score**?

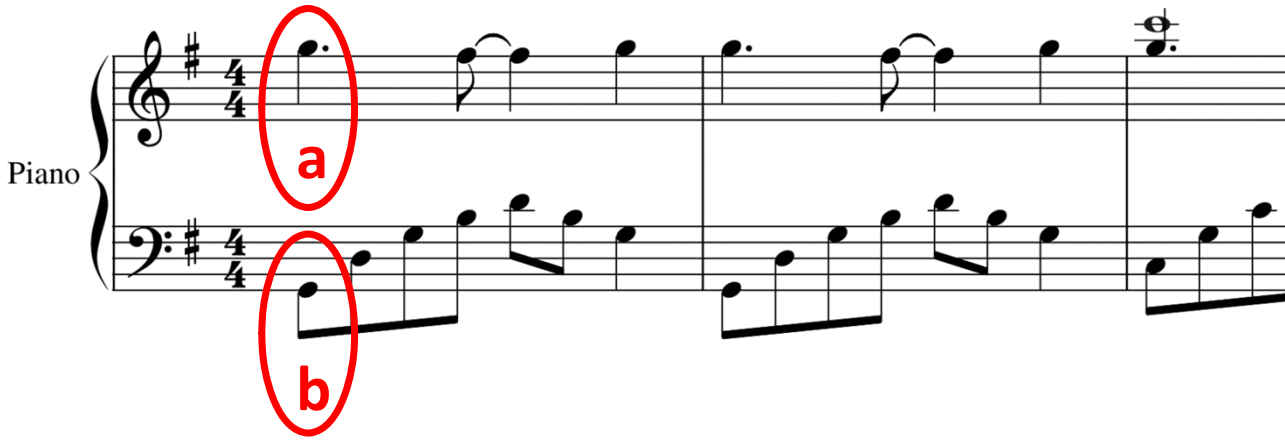


- The ending of a note is **no longer** the beginning of the next note
  - concurrent notes, temporally overlapping notes, …
- Need a way to mark the "flow of **time**"

Representing Symbolic Music: Piano Music Score

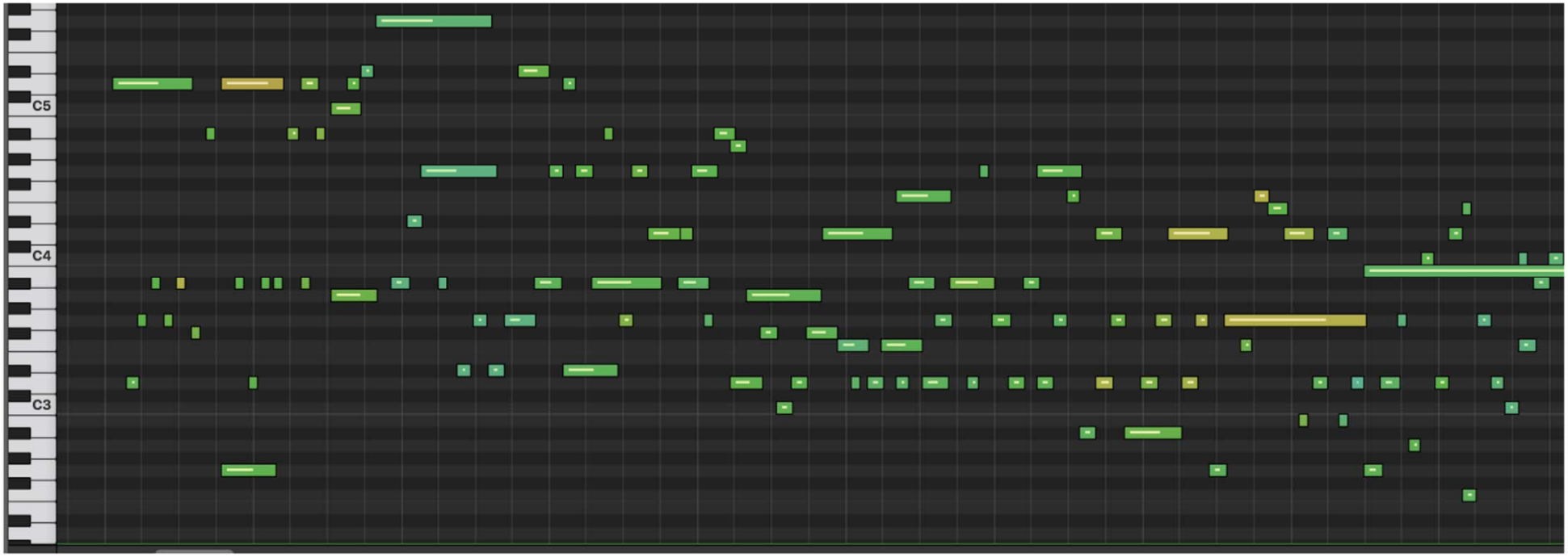
- How to represent a sequence of **piano music score**?
- Add one additional token to mark "**time-shift**" (i.e., ΔT; interval time)

- [pitch] [duration] [time-shift] [pitch] [duration] [time-shift] …



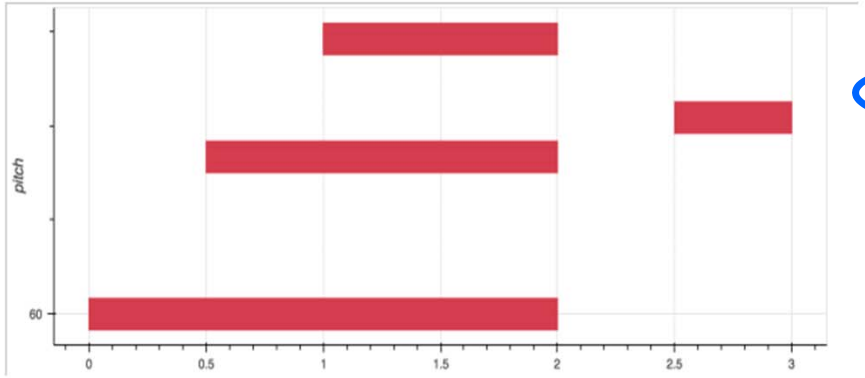
Representing Symbolic Music: Piano Music Performance

- How to represent a sequence of **piano music performance**?
  - Note **timings** are based on human performance rather than a score
  - Note **velocities** are based on human performance, i.e. with how much force did the performer strike each note



The MIDI-like Representation

- The **MIDI-like representation** for **piano music performance**
  - 128 note-on events + 128 note-off events: start or release a MIDI note
  - 100 time-shift events in increments of 10 ms up to 1 second; move forward in time to the next note event
  - 32 velocity events, corresponding to MIDI velocities quantized into 32 bins




```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```

Ref: Simon & Oore, “Performance RNN: Generating music with expressive timing and dynamics,” 2017

Representing Multi-track Music

OpenAI's **MuseNet**

https://openai.com/index/musenet/

v — **velocity** (volume), in [0, 127]
v0 — **offset** of a note
v72 — **onset** of a note
G1 — **pitch**
piano — **instrument**
wait — move the **time** marker

```
bach piano_strings start tempo90 piano:v72:G1
piano:v72:G2 piano:v72:B4 piano:v72:D4 violin:v80:G4
piano:v72:G4 piano:v72:B5 piano:v72:D5 wait:12
piano:v0:B5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:4 piano:v0:G1 piano:v0:G2 piano:v0:B4 piano:v0:D4
violin:v0:G4 piano:v0:G4 wait:1 piano:v72:G5 wait:12
piano:v0:G5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:5 piano:v72:B5 wait:12
```

(v72: onset of the piano note G1)

Representing Multi-track Music

OpenAI's **MuseNet**

https://openai.com/index/musenet/

v — **velocity** (volume), in [0, 127]
v0 — **offset** of a note
v72 — **onset** of a note
G1 — **pitch**
piano — **instrument**
wait — move the **time** marker

```
bach piano_strings start tempo90 piano:v72:G1
piano:v72:G2 piano:v72:B4 piano:v72:D4 violin:v80:G4
piano:v72:G4 piano:v72:B5 piano:v72:D5 wait:12
piano:v0:B5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:4 piano:v0:G1 piano:v0:G2 piano:v0:B4 piano:v0:D4
violin:v0:G4 piano:v0:G4 wait:1 piano:v72:G5 wait:12
piano:v0:G5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:5 piano:v72:B5 wait:12
```

(special "wait" token to indicates the delta time between events)

Representing Multi-track Music

OpenAI's **MuseNet**

https://openai.com/index/musenet/

v — **velocity** (volume), in [0, 127]
v0 — **offset** of a note
v72 — **onset** of a note
G1 — **pitch**
piano — **instrument**
wait — move the **time** marker

```
bach piano_strings start tempo90 (1)piano:v72:G1
(2)piano:v72:G2 (3)piano:v72:B4 (4)piano:v72:D4 (5)violin:v80:G4
(6)piano:v72:G4 (7)piano:v72:B5 (8)piano:v72:D5 wait:12
piano:v0:B5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:4 piano:v0:G1 piano:v0:G2 piano:v0:B4 piano:v0:D4
violin:v0:G4 piano:v0:G4 wait:1 piano:v72:G5 wait:12
piano:v0:G5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:5 piano:v72:B5 wait:12
```

(all these eight notes are played at the same time)

Representing Multi-track Music

v — **velocity** (volume), in [0, 127]
v0 — **offset** of a note
v72 — **onset** of a note
G1 — **pitch**
piano — **instrument**
wait — move the **time** marker

OpenAI's **MuseNet**

https://openai.com/index/musenet/

```
bach piano_strings start tempo90 piano:v72:G1
piano:v72:G2 piano:v72:B4 piano:v72:D4 violin:v80:G4
piano:v72:G4 piano:v72:B5 piano:v72:D5 wait:12
piano:v0:B5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:4 piano:v0:G1 piano:v0:G2 piano:v0:B4 piano:v0:D4
violin:v0:G4 piano:v0:G4 wait:1 piano:v72:G5 wait:12
piano:v0:G5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:5 piano:v72:B5 wait:12
```

(duration of the note G1: 12+5+12+4=33 ms)

Representing Multi-track Music

OpenAI's **MuseNet**

https://openai.com/index/musenet/

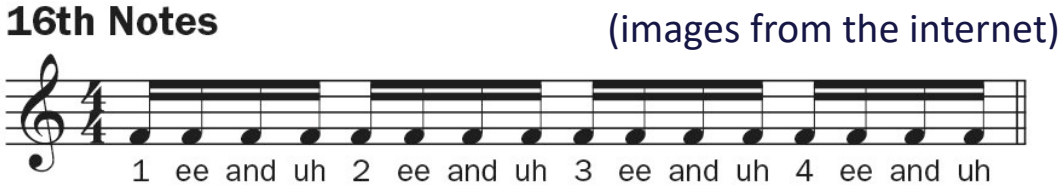
v — **velocity** (volume), in [0, 127]
v0 — **offset** of a note
v72 — **onset** of a note
G1 — **pitch**
piano — **instrument**
wait — move the **time** marker

```
bach piano_strings start tempo90 piano:v72:G1
piano:v72:G2 piano:v72:B4 piano:v72:D4 violin:v80:G4
piano:v72:G4 piano:v72:B5 piano:v72:D5 wait:12
piano:v0:B5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:4 piano:v0:G1 piano:v0:G2 piano:v0:B4 piano:v0:D4
violin:v0:G4 piano:v0:G4 wait:1 piano:v72:G5 wait:12
piano:v0:G5 wait:5 piano:v72:D5 wait:12 piano:v0:D5
wait:5 piano:v72:B5 wait:12
```

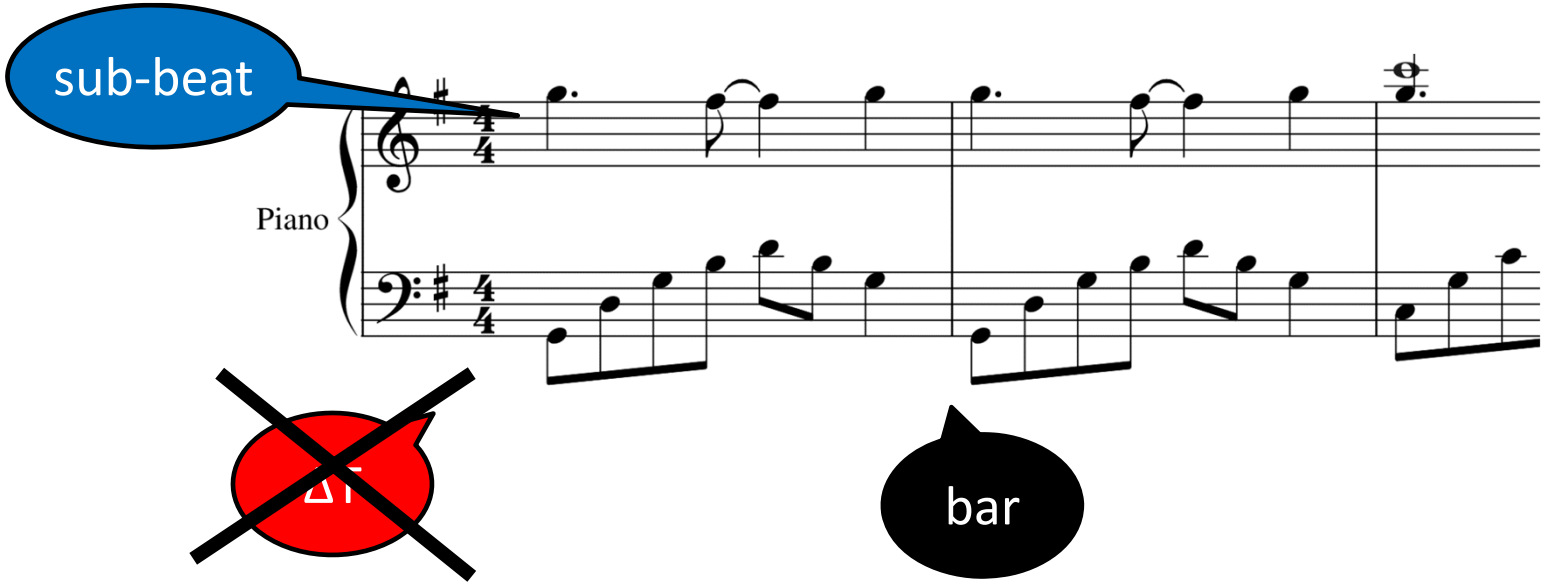
No bar lines!

REMI: Revamped MIDI Representation

- **REMI**: An alternative token representation of music
  - **Mark the bar lines** by "**bar**" tokens
  - **Indicate the position of a note within a bar** by "**sub-beat**" tokens (after timing quantization)

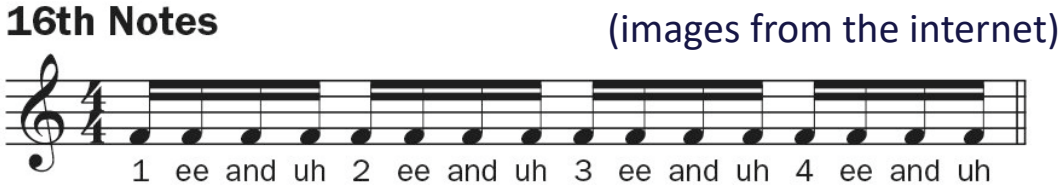


(images from the internet)

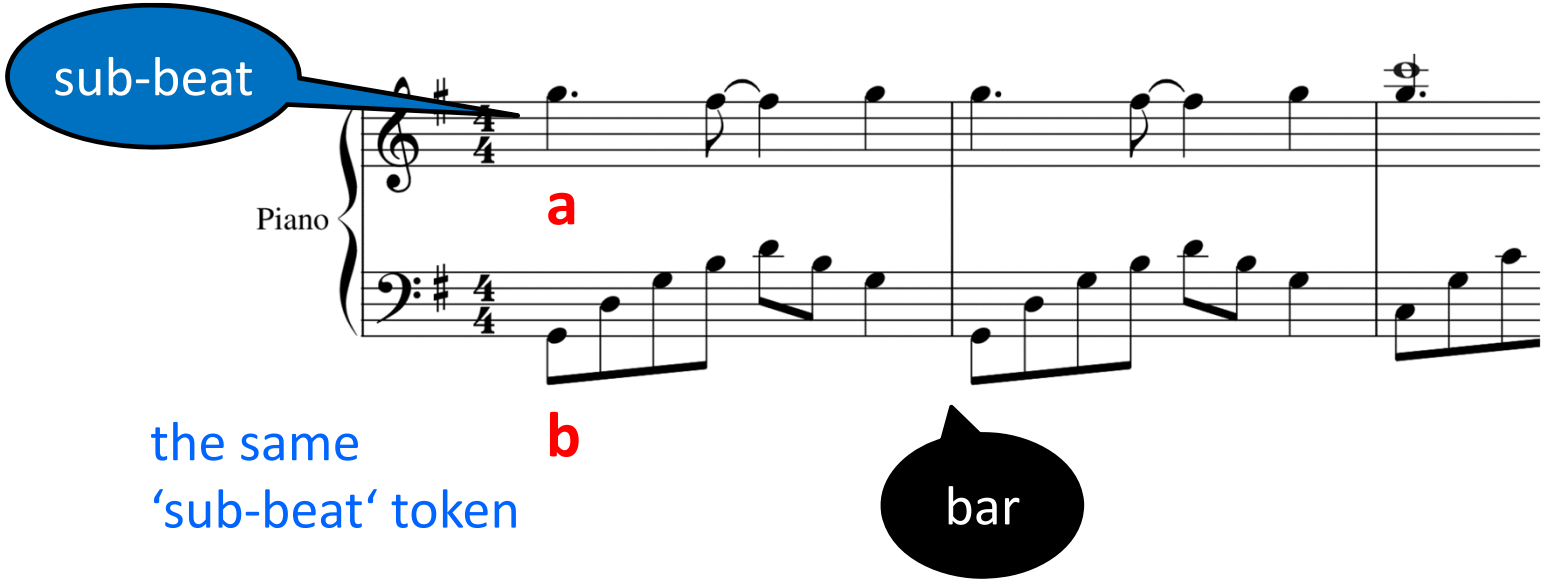


REMI: Revamped MIDI Representation

- **REMI**: An alternative token representation of music
  - **Mark the bar lines** by "**bar**" tokens
  - **Indicate the position of a note within a bar** by "**sub-beat**" tokens (after timing quantization)

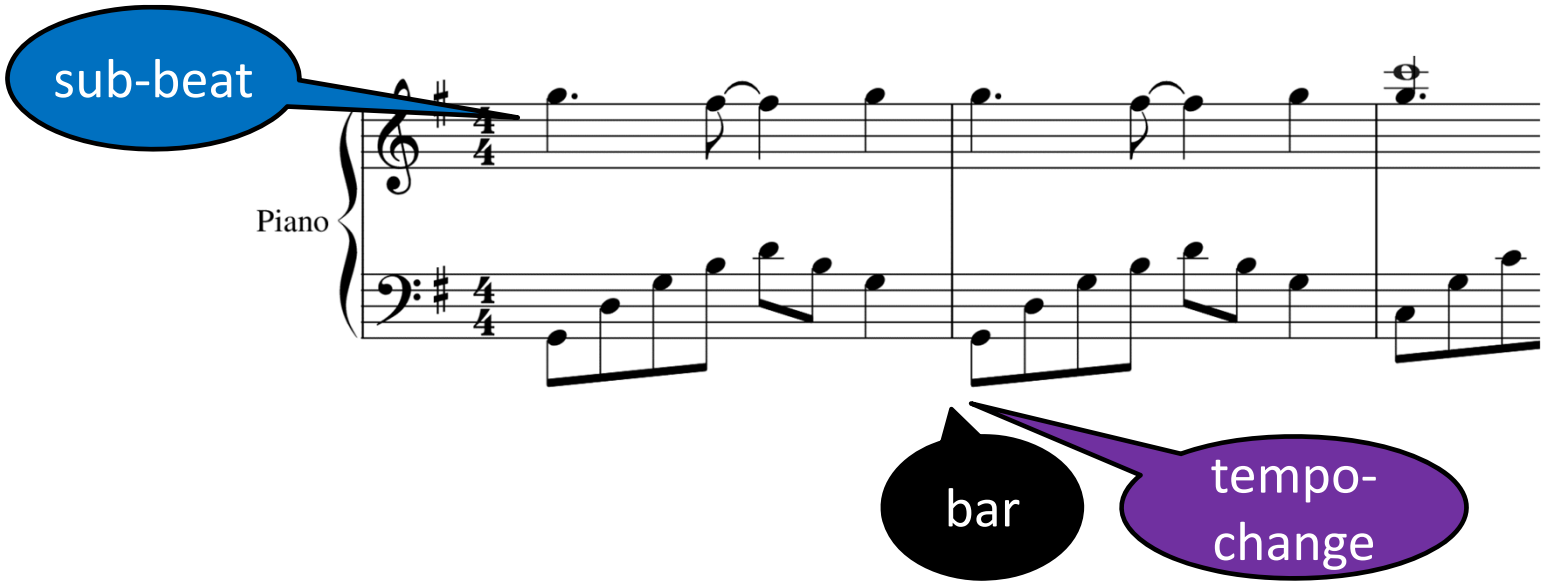


(images from the internet)



REMI: Revamped MIDI Representation

- **REMI**: An alternative token representation of music
  - Add **bar-wise tempo-change** tokens when needed to account for **tempo variations within a song** (the "tempo-change" tokens are after "bar" tokens)



REMI: Revamped MIDI Representation

- **REMI**
  - Use `**Sub-beat, Bar, and Tempo**` instead of **Time-Shift**
  - Use `**Note-Duration**` instead of **Note-Off**
    - note-on and note-off are native MIDI events
    - but using note-duration seems to have better musical meaning
  - Add `**Chord**` tokens

	MIDI-like	REMI
Note onset	NOTE-ON (0-127)	NOTE-ON (0-127)
Note offset	NOTE-OFF (0-127)	NOTE DURATION (32th note multiples)
Note velocity	NOTE VELOCITY (32 bins)	NOTE VELOCITY (32 bins)
Time grid	TIME-SHIFT (10-1000ms)	POSITION (16 bins) & BAR (1)
Tempo changes	✗	TEMPO (30-209 BPM)
Chord	✗	CHORD (60 types)

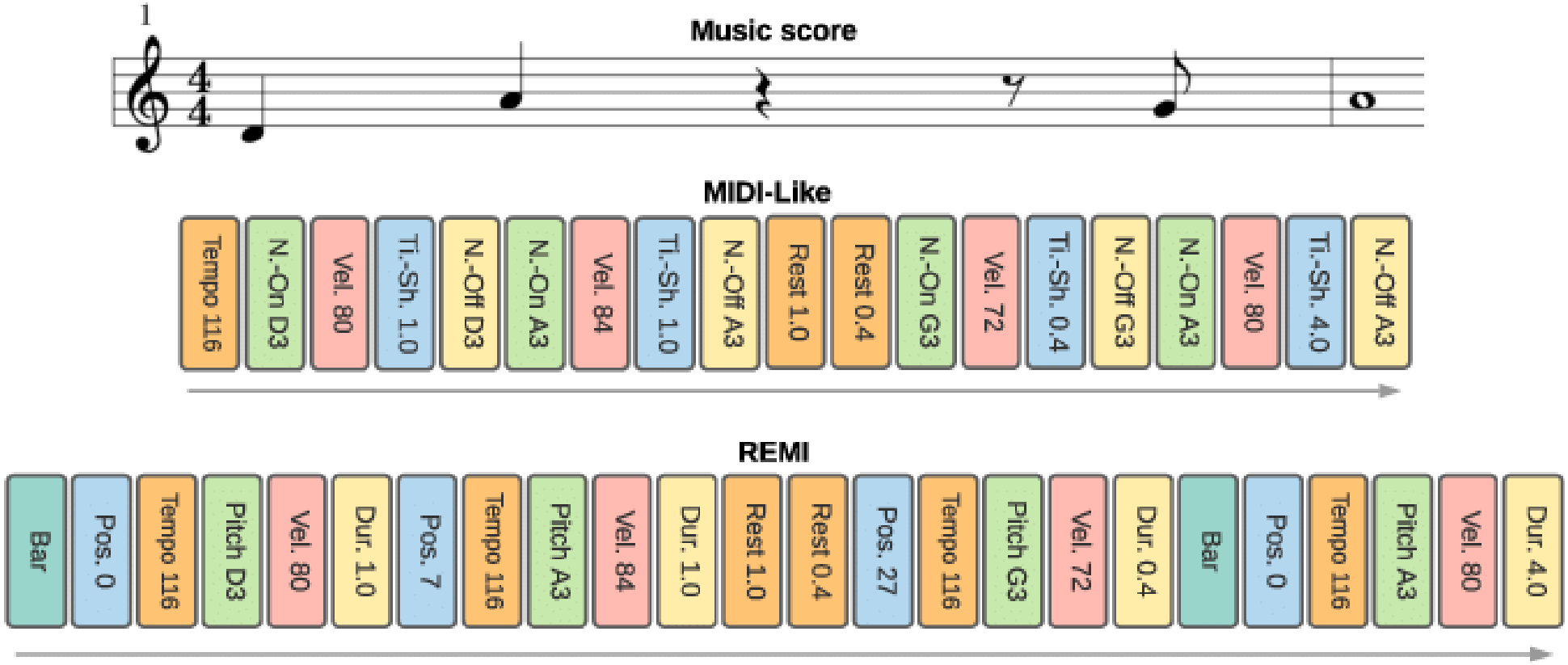
Table 1: A comparison of the commonly-used MIDI-like event representation with the proposed one, REMI. In the brackets, we show the corresponding ranges.

Ref: Huang et al, "Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions", ACM Multimedia 2020

MIDI-like vs. REMI

- **REMI**
  - Mark the bar lines **bar** tokens and divide a bar into 16 discrete **sub-beats**
  - **Symbolic timing + explicit time grid**
  - More timing quantization
  - Good for **MIDI scores**
  - Good for MIDI performances with **steady beats** like the **Pop music**
  - Good for **multi-track music**
    - The bar tokens make it easy to synchronize
- **MIDI-like**
  - Use **time-shift** tokens
  - **Physical timing & relative time intervals**
  - **Less timing quantitation**
  - Good for expressive MIDI performances such as the **Classical music**

MIDI-like vs REMI



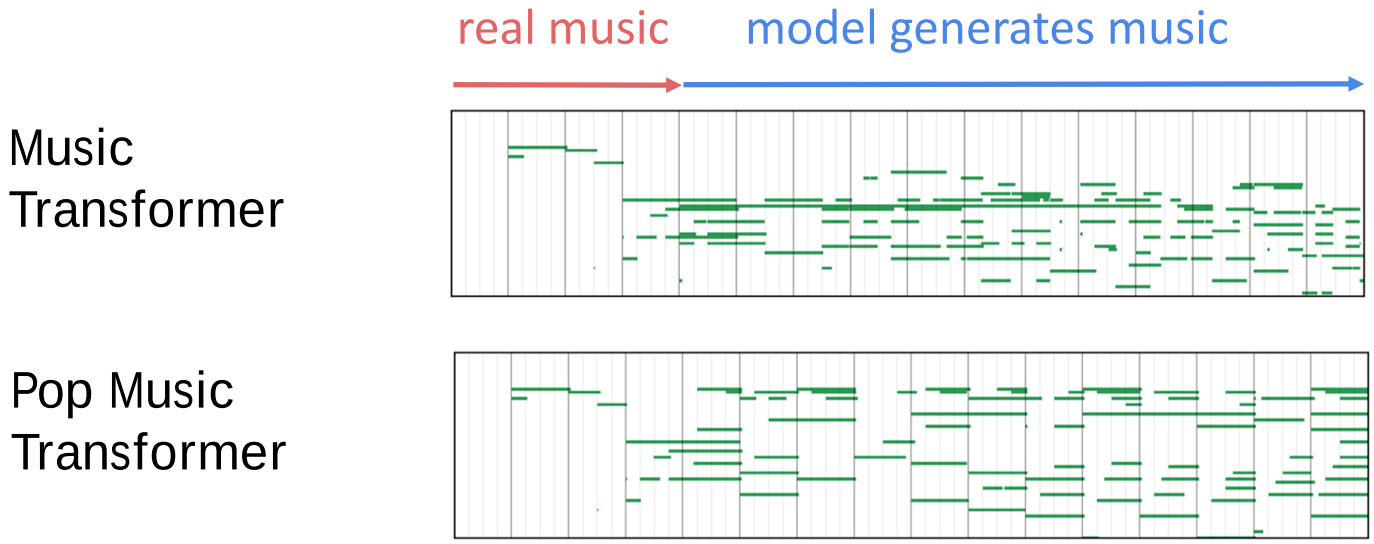
Ref: Fradet et al, “MidiTok: A Python package for MIDI file tokenization”, ISMIR LBD 2021

Pop Music Transformer [huang20mm]

piano

https://github.com/YatingMusic/remi

- Adopt the **REMI** representation
- Outperforms the Music Transformer ("MIDI-like") for generating pop piano performances



Ref: Huang et al, "Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions", ACM Multimedia 2020

Pop Music Transformer [huang20mm]

Pop Music Transformer ("REMI") outscores Music Transformer ("Baselines 1 & 3") in a user study that involves 76 raters

Pairs		Wins	Losses	p-value
REMI	Baseline 1	103	49	5.623e-5
REMI	Baseline 3	92	60	0.0187
Baseline 3	Baseline 1	90	62	0.0440

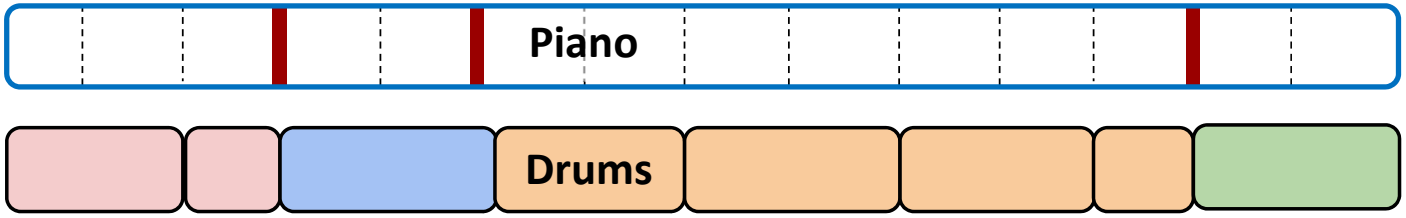
Table 5: The result of pairwise comparison from the user study, with the p-value of the Wilcoxon signed-rank test.

Method	Note offset	Time grid	TEMPO	CHORD	Beat STD	Downbeat STD	Downbeat salience
Baseline 1	NOTE-OFF	TIME-SHIFT (10-1000ms)			0.0968	0.3561	0.1033
Baseline 2	DURATION	TIME-SHIFT (10-1000ms)			0.0394	0.1372	0.1651
Baseline 3	DURATION	TIME-SHIFT (16th-note multiples)			0.0396	0.1383	0.1702
REMI	DURATION	POSITION & BAR	✓	✓	0.0386	0.1376	0.2279
	DURATION	POSITION & BAR	✓		0.0363	0.1265	0.1936
	DURATION	POSITION & BAR		✓	0.0292	0.0932	0.1742
	DURATION	POSITION & BAR			0.0199	0.0595	0.1880
Real data					0.0607	0.2163	0.2055

Pop Music Transformer

piano + drum guitar


- **Easier to add drums** (via structure analysis and grooving analysis)
  - https://soundcloud.com/yating_ai/sets/ai-piano-generation-demo-202004
  - https://soundcloud.com/yating_ai/sets/ai-pianodrum-generation-demo-202004



(Figure made by Wen-Yi Hsiao)

- Extensions can also generate **guitar tabs** [chen20ismir]
  - https://soundcloud.com/yating_ai/ai-guitar-tab-generation-202003/s-KHozfW0PTv5

Ref 1: Huang et al, “Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions”, ACM Multimedia 2020
Ref 2: Chen et al, “Automatic composition of guitar tabs by Transformers and groove modeling”, ISMIR 2020

Text vs Symbolic Music Tokens

- **Text tokenizers**
  - **Word**-based tokenizers
    - Large vocabulary size
    - OOV issue
  - **Character**-based tokenizers
    - Small vocabulary size (**tens**)
    - No OOV
    - Long token sequence
  - **Sub-word** tokenizers (e.g., byte-pair encoding)
    - Middle ground
- **Symbolic music tokenizers**
  - Need multiple tokens to describe the *various aspects* of a musical note
  - Need other auxiliary tokens to denote *time* or *musical structure*
  - It's like *character-level* tokenizers
    - Small vocabulary size (**hundreds**)
    - No OOV
    - Long token sequence
  - Can also do sub-word tokenization
    - Ref: Fradet et al, "Byte pair encoding for symbolic music," EMNLP 2023.

Library: MIDITok

https://miditok.readthedocs.io/

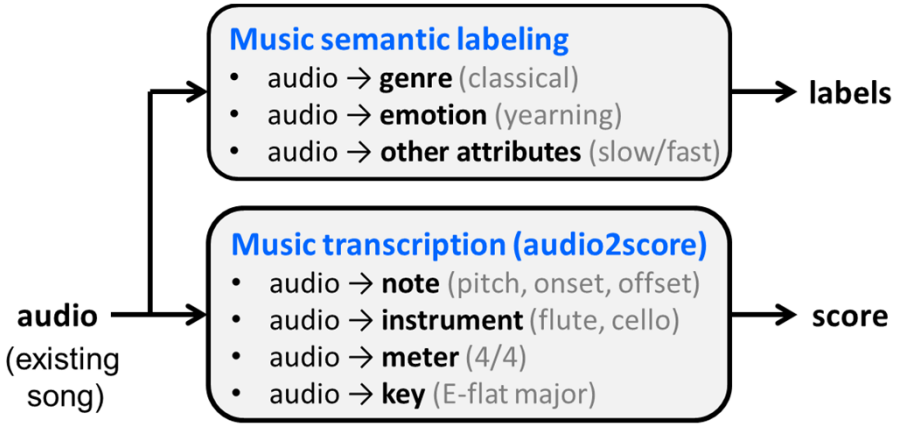
- “MidiTok can take care of converting (tokenizing) your symbolic music files (**MIDI**, **abc**) into **tokens**, ready to be fed to models such as **Transformer**, for any generation, transcription or MIR task”
  - “MidiTok features most known MIDI tokenizations (e.g. REMI, Compound Word...), and is built around the idea that they all share common parameters and methods”
  - “It supports Byte Pair Encoding (BPE) and data augmentation.”

Outline

- Sheet music & symbolic representations for music
- Token representation of MIDI music
- **Symbolic-domain MIR & evaluation metrics**

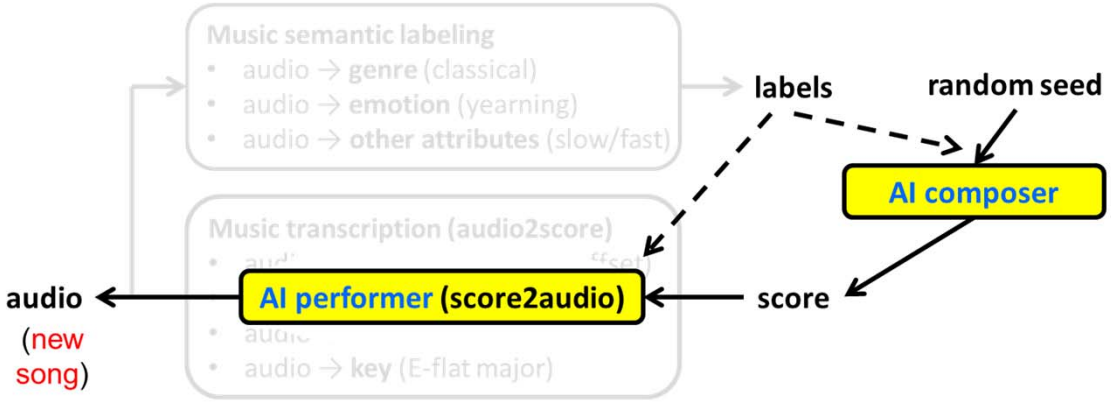
Music AI; or *Music Information Research* (MIR)

- **Music analysis**



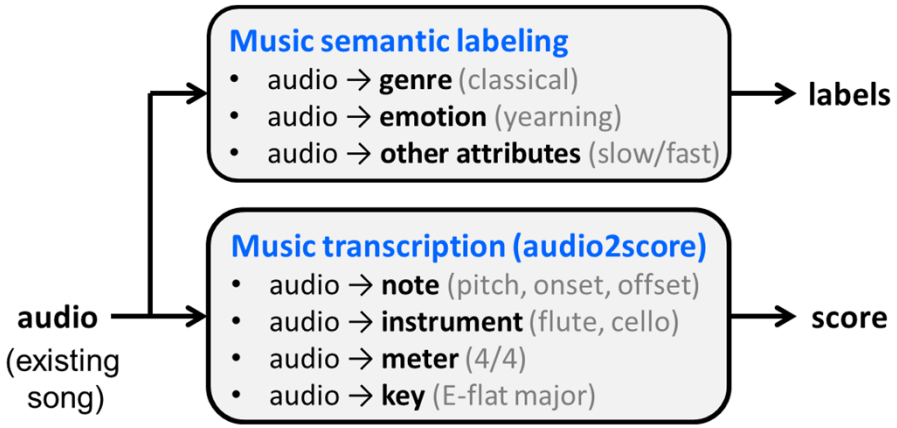
  - music understanding
  - music search
  - music recommendation

- **Music generation**



  - MIDI generation
  - audio generation
  - MIDI-to-audio generation

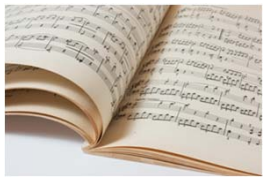
Symbolic-domain Music Analysis



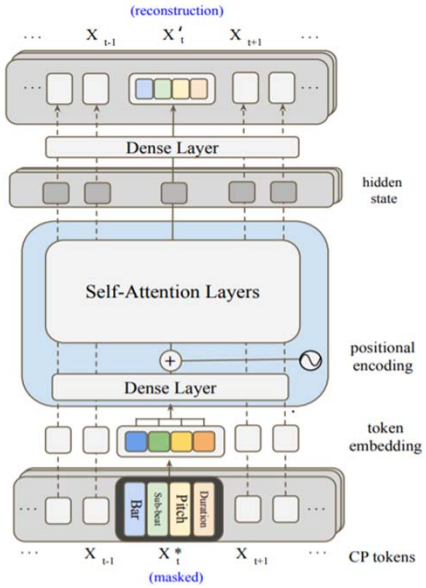
- Task
  - **Classification**
    - Genre/style classification
    - Composer/performer classification
    - Emotion classification
  - **Transcription**
    - Melody detection
    - Chord recognition
    - Key estimation
- Approach
  - Piano roll → CNN
  - Token sequence → RNN, Transformer

Symbolic-domain Music Classification

https://github.com/wazenmai/MIDI-BERT

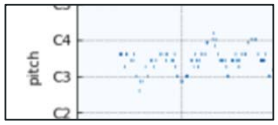


Input MIDI
(note sequence)

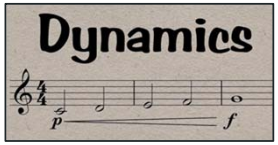


MidiBERT

Melody



Velocity



Composer

Emotion

Ref: Chou et al, "MidiBERT-Piano: BERT-like pre-training for symbolic piano music classification tasks," JCMS 2024

Symbolic-domain Music Classification

https://github.com/wazenmai/MIDI-BERT

Pianist8 dataset

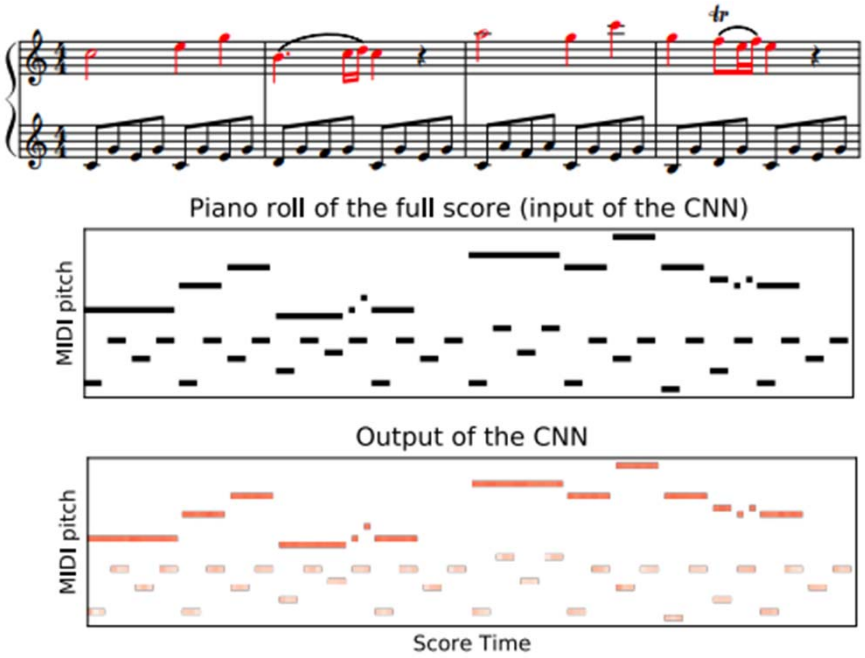
Richard Clayderman, Herbie Hancock, Ludovico Einaudi, Hisaishi Joe, Yiruma, Ryuichi Sakamoto, Bethel Music, and Hillsong Worship

Token	Model	Melody	Velocity	Style	Emotion
REMI	CNN (S. Kim *et al.*, 2020)	—	—	51.35	60.00
	RNN (Z. Lin *et al.*, 2017)	89.96	44.56	51.97	53.46
	Our model (`score`)	90.97	49.02	67.19	67.74
	Our model (`performance`)	89.23	—	75.30	66.18
CP	CNN (S. Kim *et al.*, 2020)	—	—	67.57	60.00
	RNN (Z. Lin *et al.*, 2017)	88.66	43.77	60.32	54.13
	Our model (`score`)	**96.15**	**52.11**	67.46	64.22
	Our model (`performance`)	95.83	—	**81.75**	70.64
OctupleMIDI	*MusicBERT* (Zeng *et al.*, 2021)	—	—	37.25	**77.78**

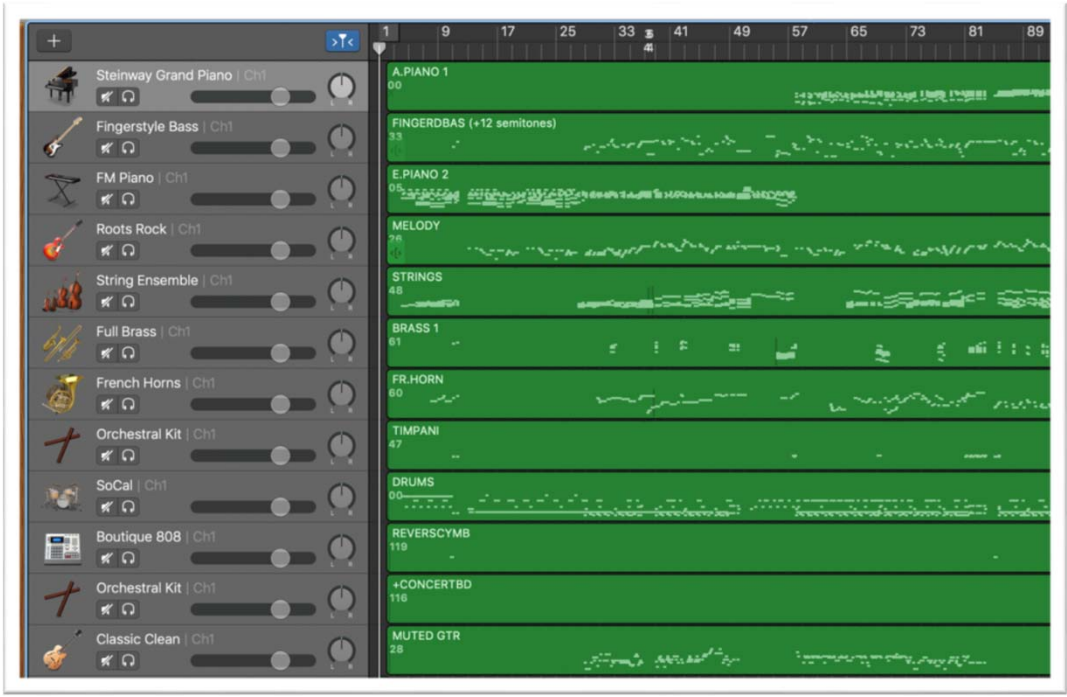
Ref: Chou et al, "MidiBERT-Piano: BERT-like pre-training for symbolic piano music classification tasks," JCMS 2024

Symbolic-domain Melody Extraction

- **Symbolic-domain melody extraction** from a *single track* of polyphonic music (e.g., piano)



- **Melody-track identification** from *multi-track* MIDI



(image from the internet)

Ref: https://archives.ismir.net/ismir2019/paper/000114.pdf

Symbolic-domain Melody Extraction

- **Symbolic-domain melody extraction** from a *single track* of polyphonic music (e.g., piano)
  - **Skyline algorithm**: https://github.com/wazenmai/MIDI-BERT/tree/CP/melody_extraction/skyline
  - **CNN**: https://github.com/sophia1488/symbolic-melody-identification
  - more: https://github.com/RetroCirce/Midi_Toolkit/tree/main
- **Melody-track identification** from *multi-track* MIDI
  - **Skyline algorithm**
    - Used in METEOR (arxiv'24)
  - **Random forest**: https://github.com/wayne391/midi-track-identification
  - **Lyrics-informed method**: https://github.com/gulnazaki/lyrics-melody
    - Used in Compose & Embellish (ICASSP'22)

Melody-track Identification from Multi-track MIDI

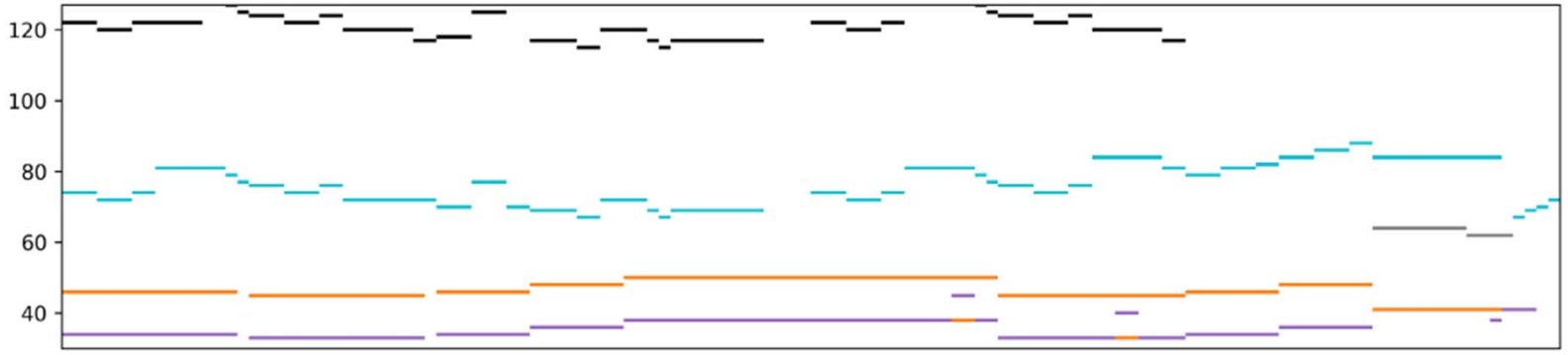
https://dinhviettoanle.github.io/meteor/

Reference

Detected melodic instrument:

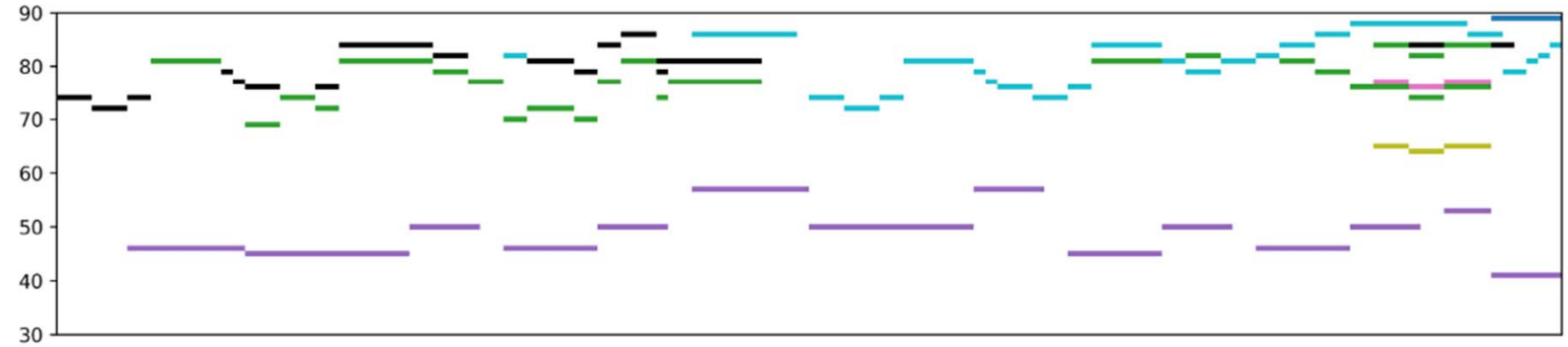
Glockenspiel, then flute

0:00 / 0:19



Melodic instrument: Trumpet

0:00 / 0:19



Symbolic-domain Chord Recognition

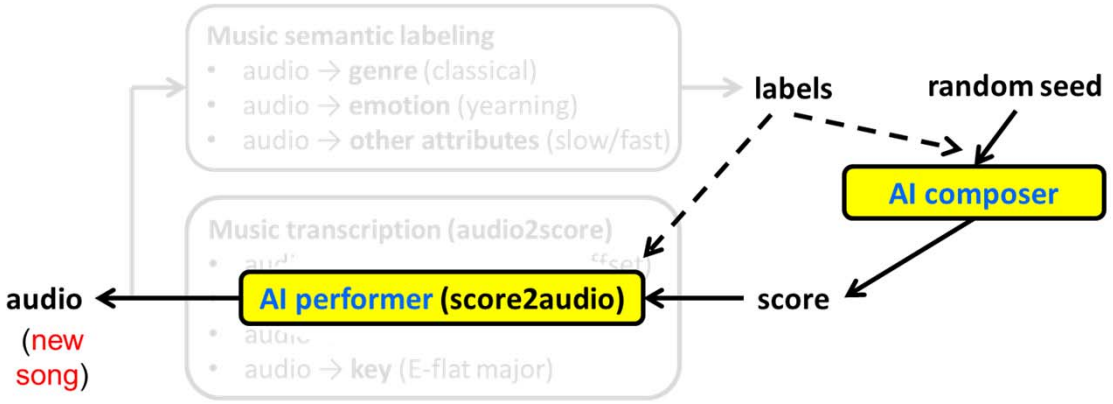
- Rule-based
  - **Chorder**: https://github.com/joshuachang2311/chorder

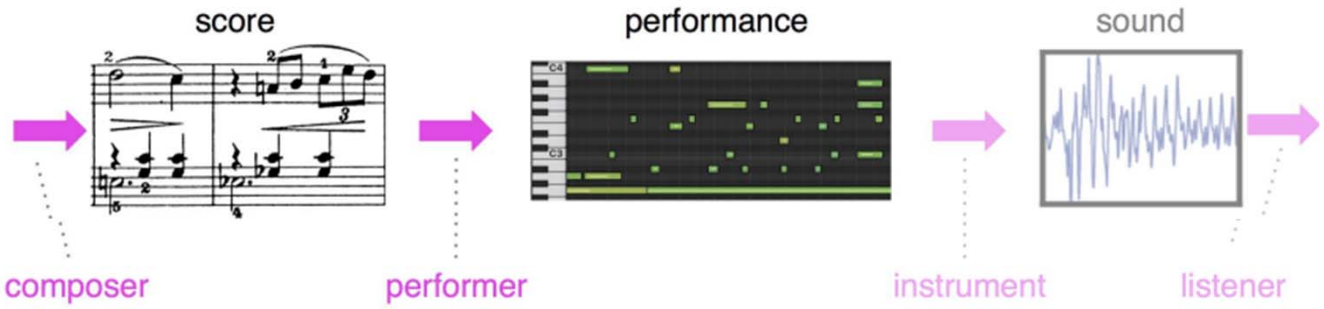
```
class Dechorder:
    major_weights = [10, -2, -1, -2, 10, -5, -2, 10, -2, -1, -2, 0]
    minor_weights = [10, -2, -1, 10, -2, -5, -2, 10, -1, -2, 0, -2]
    diminished_weights = [10, -2, -1, 10, -2, -1, 10, -2, -2, 1, -1, -2]
    augmented_weights = [10, -2, -1, -2, 10, -1, -2, -2, 10, -1, -2, 0]
    sus_2_weights = [10, -2, 5, -2, -1, -5, -2, 5, -2, -1, -2, 0]
    sus_4_weights = [10, -2, -1, -2, -5, 5, -2, 5, -2, -1, -2, 0]

    major_map = [0, 4, 7]
    minor_map = [0, 3, 7]
    diminished_map = [0, 3, 6]
    augmented_map = [0, 4, 8]
    dominant_map = [0, 4, 7, 10]
    major_seventh_map = [0, 4, 7, 11]
    minor_seventh_map = [0, 3, 7, 10]
    diminished_seventh_map = [0, 3, 6, 9]
    half_diminished_seventh_map = [0, 3, 6, 10]
    sus_2_map = [0, 2, 7]
    sus_4_map = [0, 5, 7]
```

Symbolic-domain Music Generation

- Next lecture

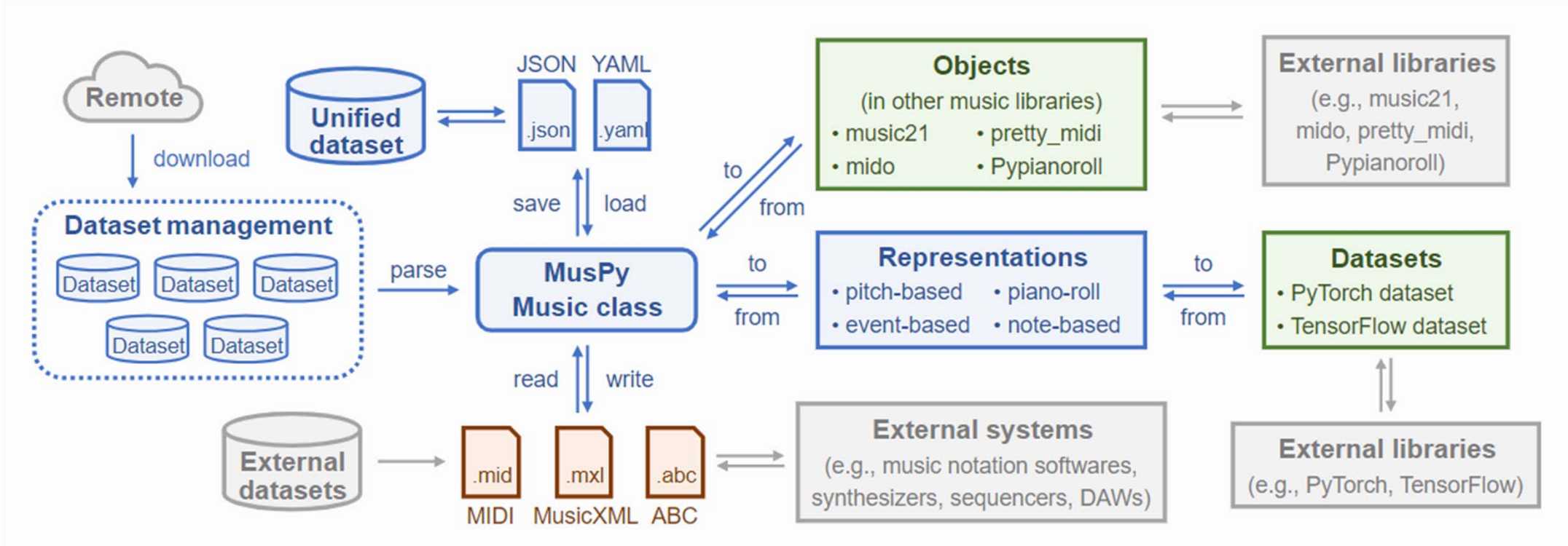
- **Music generation**





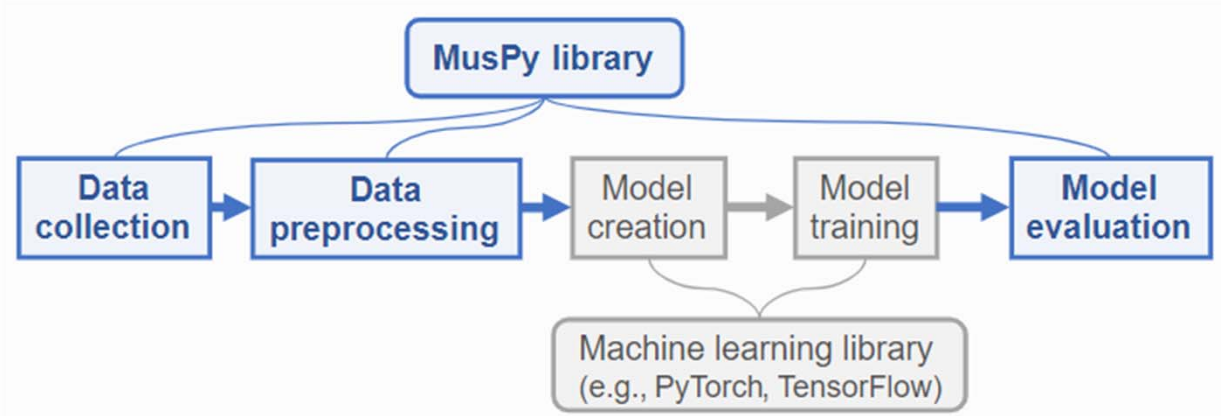
Library: MusPy

https://salu133445.github.io/muspy/index.html



Library: MusPy

https://salu133445.github.io/muspy/index.html



- **Features**
  - Dataset management system for commonly used **datasets** with interfaces to PyTorch and TensorFlow
  - Data **I/O** for common symbolic music formats and interfaces to other symbolic music libraries
  - Implementations of common **representations** for music generation, including the pitch-based, the event-based, the piano-roll and the note-based representations
  - Model **evaluation tools** for music generation systems, including audio rendering, score and piano-roll visualizations and objective metrics

Evaluation Metrics for MIDI Generation

- Subjective evaluation metrics (e.g., MOS)
  - Tend to be more convincing
  - Many details about the design of a user study, though

Table 4. User study MOS results. (**Coh**erence, **Cor**rectness, **S**tructureness, **R**ichness, **O**verall. SDs across individual data points follow ±.)

	Ch	Cr	S	R	O
CPT [4]	2.38±0.9	2.49±0.9	2.33±0.9	2.64±0.9	2.33±0.9
C&E (ours)	3.53±0.9	3.11±1.0	3.36±1.2	3.29±1.0	3.18±0.9
Real data	4.42±0.7	4.13±0.8	4.44±0.8	4.24±0.8	4.40±0.7

Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

- Objective evaluation metrics
  - May not reflect human perception
  - Good for preliminary experiments

Objective Evaluation Metrics for MIDI Generation

- Important aspects of MIDI generation
  - **Correctness**: "Is the music free of inharmonious notes, unnatural rhythms, and awkward phrasing?"
  - **Coherence**: over time & across different tracks
  - **Structureness**: "Are recurring motifs / phrases / sections, and reasonable musical development present?"
  - **Richness**: "Is the music intriguing and full of variations within?"
  - **Controllability**: Can the generation process be easily steered? Does the model allow for easy human-AI co-generation?

Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Objective Measures for Correctness and Coherence

https://salu133445.github.io/muspy/doc/metrics.html

- For general MIDI → use **MusPy**

- drum_in_pattern_rate
- drum_pattern_consistency
- empty_beat_rate
- empty_measure_rate
- groove_consistency
- n_pitch_classes_used
- n_pitches_used
- pitch_class_entropy
- pitch_entropy
- pitch_in_scale_rate
- pitch_range
- polyphony
- polyphony_rate
- scale_consistency

Ref 1: Dong et al,"MusPy: A toolkit for symbolic music generation", ISMIR 2020
Ref 2: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

Objective Measures for Correctness and Coherence

https://github.com/slSeanWU/MusDr

- For single-track MIDI (e.g., Piano) → use **MusDr**
  - Pitch-Class Histogram Entropy (**H**)
    -- measures erraticity of **pitch usage** in shorter timescales (e.g., 1 or 4 bars)
  - Grooving Pattern Similarity (**GS**)
    -- measures consistency of **rhythm** across the entire piece
  - Chord Progression Irregularity (**CPI**)
    -- measures consistency of **harmony** across the entire piece

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Pitch Class Histogram Entropy

- "If a piece's **tonality** is clear, some **pitch classes** should dominate the pitch histogram (e.g., the tonic and dominant), resulting in low-entropy"

To gain insight into the usage of different pitches, we first collect the notes appeared in a certain period (e.g., a bar) and construct the 12-dimensional pitch class histogram $\overrightarrow{\mathbf{h}}$, according to the notes' pitch classes (i.e. `C`, `C#`, ..., `A#`, `B`), normalized by the total note count in the period such that $\sum_i h_i = 1$. Then, we calculate the entropy of $\overrightarrow{\mathbf{h}}$:

$$\mathcal{H}(\overrightarrow{\mathbf{h}}) = -\sum_{i=0}^{11} h_i \log_2(h_i)\,. \tag{2}$$

The entropy, in information theory, is a measure of "uncertainty" of a probability distribution [40], hence we adopt it here as a metric to help assessing the music's quality in tonality. If a piece's tonality is clear, several pitch classes should dominate the pitch histogram (e.g., the tonic and the dominant), resulting in a low-entropy $\overrightarrow{\mathbf{h}}$; on the contrary, if the tonality is unstable, the usage of pitch classes is likely scattered, giving rise to an $\overrightarrow{\mathbf{h}}$ with high entropy.

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Grooving Pattern Similarity

- If a piece possesses a clear sense of **rhythm**, the **grooving patterns** between pairs of bars should be similar, thereby producing high GS scores

The grooving pattern represents the positions in a bar at which there is at least a note onset, denoted by $\overrightarrow{\mathbf{g}}$, a 64-dimensional binary vector in our setting.[6] We define the similarity between a pair of grooving patterns $\overrightarrow{\mathbf{g}}^a$, $\overrightarrow{\mathbf{g}}^b$ as:

$$\mathcal{GS}(\overrightarrow{\mathbf{g}}^a, \overrightarrow{\mathbf{g}}^b) = 1 - \frac{1}{Q}\sum_{i=0}^{Q-1} \mathrm{XOR}(g_i^a, g_i^b)\,, \qquad (3)$$

where Q is the dimensionality of $\overrightarrow{\mathbf{g}}^a$, $\overrightarrow{\mathbf{g}}^b$, and $\mathrm{XOR}(\cdot,\cdot)$ is the exclusive OR operation. Note that the value of $\mathcal{GS}(\cdot,\cdot)$ would always lie in between 0 and 1.

The grooving pattern similarity helps in measuring the music's rhythmicity. If a piece possesses a clear sense of rhythm, the grooving patterns between pairs of bars should be similar, thereby producing high $\mathcal{GS}$ scores; on the other hand, if the rhythm feels unsteady, the grooving patterns across bars should be erratic, resulting in low $\mathcal{GS}$ scores.

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Chord Progression Irregularity

- A well-composed Jazz piece should have a chord progression irregularity that is not too high

To measure the irregularity of a chord progression, we begin by introducing the term *chord trigram*, which is a triple composed of 3 consecutive chords in a chord progression; for example, (`Dm7`, `G7`, `CM7`). Then, *the chord progression irregularity* (*CPI*) is defined as the percentage of *unique chord trigrams* in the chord progression of an entire piece. Please note that 2 chord trigrams are considered different if any of their elements does not match.

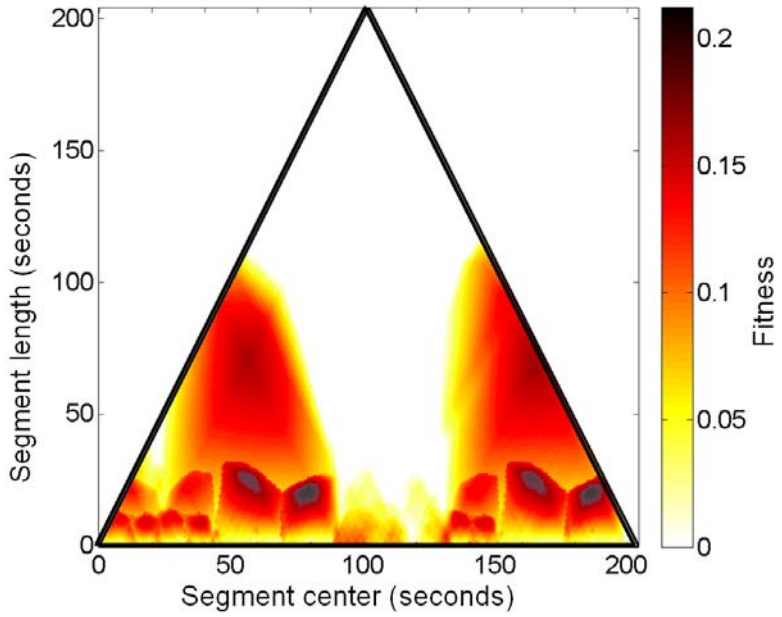
It is common for Jazz compositions to make use of 8- or 12-bar-long templates of chord progressions (known as the *8-*, or *12-bar blues*), which themselves can be broken down into similar substructures [25, 35], as the foundation of a section, and more or less "copy-paste" them to form the complete song with, say, `AABA` parts. Therefore, a well-composed Jazz piece should have a chord progression irregularity that is not too high.

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Objective Measures for Structureness

https://github.com/slSeanWU/MusDr

- **MusDr**



Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Our *structureness indicator* is based on the fitness scape plot and designed to capture the most salient repeat within a certain duration interval. For brevity of the mathematical representation, we assume the sampling frame rate of S is 1 Hz (hence N will be the piece's duration in seconds), and define the structureness indicator as follows:

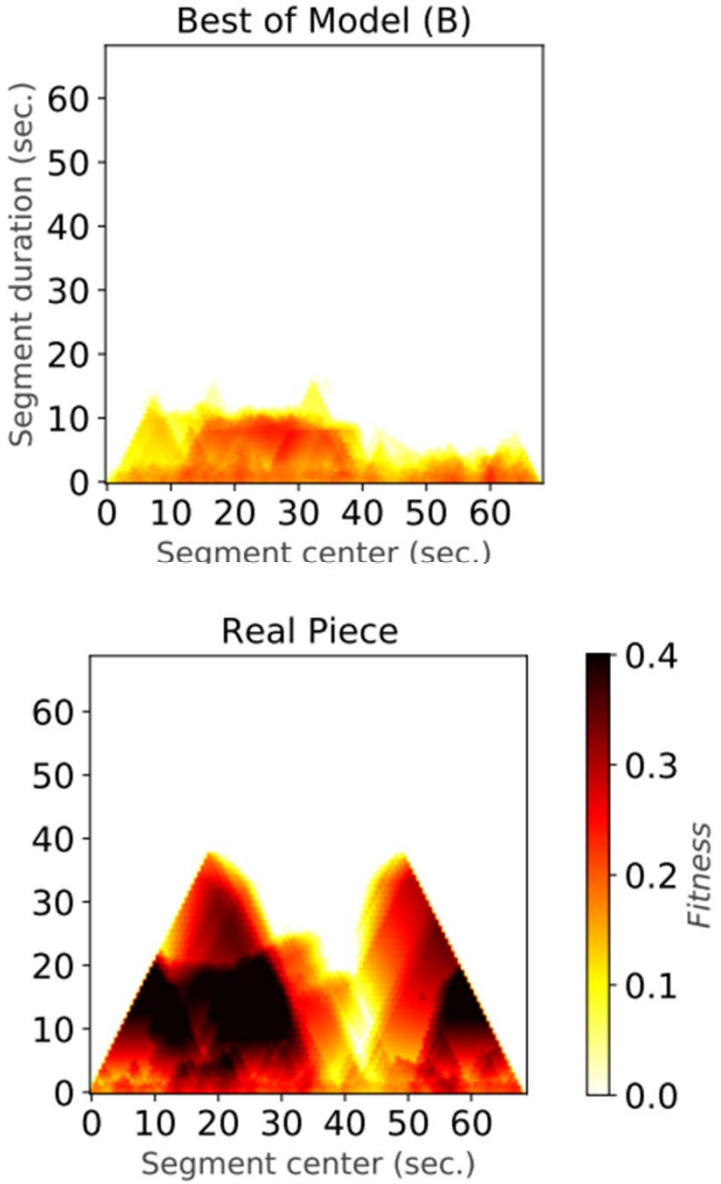
$$\mathcal{SI}_l^u(S) = \max_{\substack{l \le i \le u \\ 1 \le j \le N}} S\,, \tag{4}$$

where l, u [8] are the lower and upper bounds of the duration interval (in seconds) one is interested in. In our experiments, we choose the structureness indicators of $\mathcal{SI}_3^8$, $\mathcal{SI}_8^{15}$, and $\mathcal{SI}_{15}$ to examine the short-, medium-, and long-term structureness respectively.

Evaluation Result of a 2020 Paper

	Model (A)		Model (B)			Real
loss	*0.80*	*0.25*	*0.80*	*0.25*	*0.10*	- -
$\mathcal{H}_1$	2.29	2.45	2.26	2.20	**2.17**	1.94
$\mathcal{H}_4$	3.12	3.05	3.04	**2.91**	2.94	2.87
$\mathcal{GS}$	**0.76**	0.69	0.75	**0.76**	**0.76**	0.86
$\mathcal{CPI}$	81.2	77.6	79.2	**72.6**	75.9	40.4
$\mathcal{SI}_3^8$	0.18	0.22	0.25	**0.27**	0.26	0.36
$\mathcal{SI}_8^{15}$	0.15	0.17	**0.18**	**0.18**	0.17	0.36
$\mathcal{SI}_{15}$	0.11	**0.14**	0.10	0.12	0.11	0.35

Ref: Wu & Yang, “The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures”, ISMIR 2020



Objective Measures for Richness

Percentage of Distinct Pitch N-grams in Melody ($\mathcal{DN}$): following the popular *dist-n* [23] metric used in natural language generation to evaluate the diversity of generated content, we compute the percentage of distinct n-grams in the pitch sequence of the skyline extracted from each full performance. We regard **3∼5**, **6∼10**, and **11∼20** contiguous notes as short, medium, and long excerpts, and compute $\mathcal{DN}_{\text{short}}$, $\mathcal{DN}_{\text{mid}}$, and $\mathcal{DN}_{\text{long}}$ accordingly.

	Structureness (in %)			**Melody-line Diversity**		
	$\mathcal{SI}_{\text{short}}$	$\mathcal{SI}_{\text{mid}}$	$\mathcal{SI}_{\text{long}}$	$\mathcal{DN}_{\text{short}}$	$\mathcal{DN}_{\text{mid}}$	$\mathcal{DN}_{\text{long}}$
Naive repeats (1-bar)	83.6 ±8.6	88.3 ±5.4	75.7 ±11	1.2 ±0.6	1.2 ±0.6	1.3 ±0.6
Naive repeats (1-phrase)	75.5 ±15	86.2 ±6.8	73.9 ±11	8.2 ±4.8	10.0 ±5.8	10.8 ±6.5
CP Transformer [4]	32.5 ±3.3	29.9 ±4.4	17.9 ±6.3	82.0 ±8.9	99.6 ±0.7	100 ±0.0
Real data	43.8 ±7.1	43.1 ±8.4	34.8 ±12	50.0 ±14	74.1 ±14	88.3 ±11

Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Objective Measures for Controllability

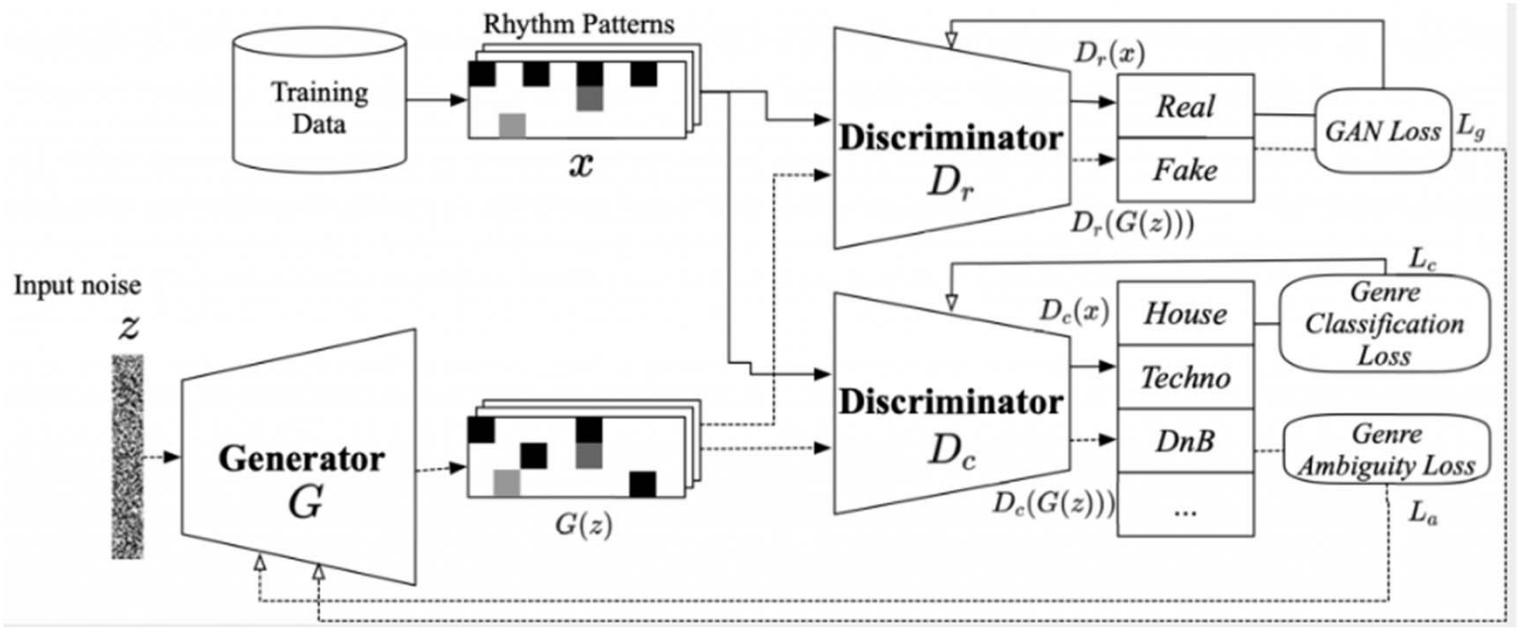
Model	Fidelity		Control			
	sim_{chr} ↑	sim_{grv} ↑	ρ_{rhym} ↑	ρ_{poly} ↑	$\lvert\rho_{\text{poly}\lvert\text{rhym}}\rvert$ ↓	$\lvert\rho_{\text{rhym}\lvert\text{poly}}\rvert$ ↓
MIDI-VAE [22]	75.6	85.4	.719	.261	.134	.056
Attr-Aware VAE [23]	85.0	76.8	**.997**	.781	.239	.040
Ours, AE objective	**98.5**	**95.7**	.181	.154	.058	.072
Ours, VAE objective	78.6	80.7	.931	.884	.038	**.003**
Ours, pre-attention	49.7	71.5	.962	**.921**	.035	.044
Ours, preferred settings	91.2	84.5	.950	.885	**.023**	.016

- The Spearman's rank correlation between an input attribute class and the attribute computed from the resulting generation
- The model should transfer an attribute without affecting other attributes

Ref: Wu & Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE", TASLP 2023

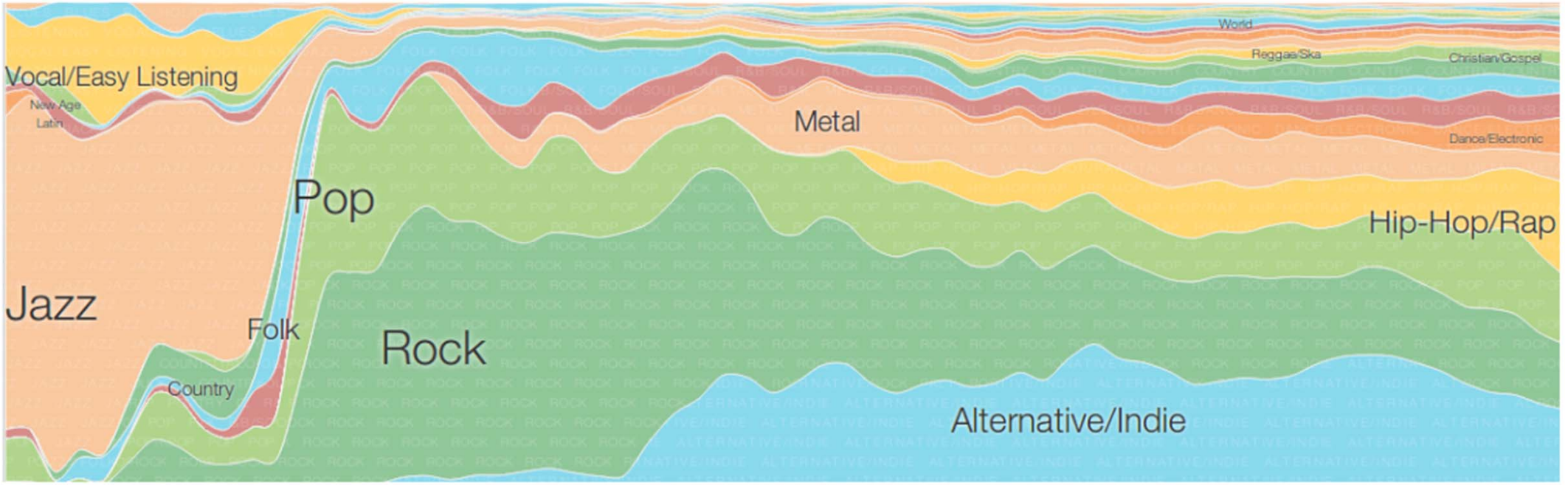
Objective Measures for “Creativity”?

- Use GAN?



Ref: https://naotokui.net/research/creative-adversarial-networks-for-rhythms/

Objective Measures for “Creativity”?



Ref: https://research.google/blog/explore-the-history-of-pop-and-punk-jazz-and-folk-with-the-music-timeline/