

2025 edition

Deep Learning for Music Analysis and Generation

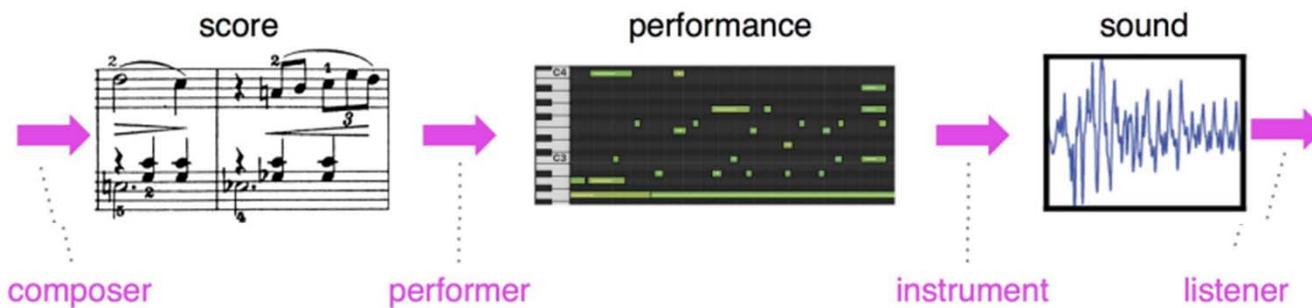
# MIDI Generation

( $x \rightarrow \text{MIDI}$ )



**Yi-Hsuan Yang** Ph.D.  
[yhyangtw@ntu.edu.tw](mailto: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
  - Still relevant (given the progress in audio) as it's interpretable and editable
  - Less compute-demanding (compared to text-to-music generation)
  - Good first step toward generative systems for music

# Outline

- **Sheet music & symbolic representations for music**
- Token representation of MIDI music
- MIDI generation
- Evaluation metrics

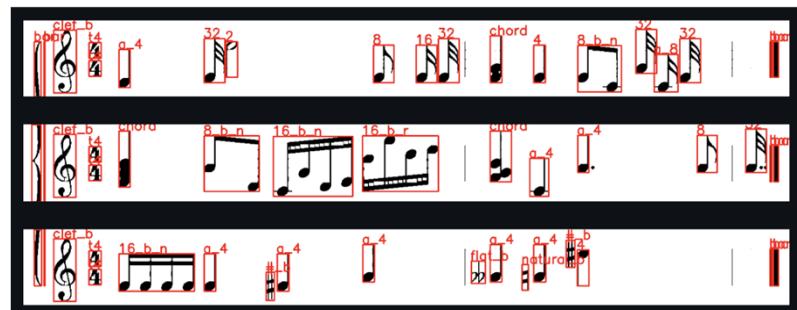
# Sheet Music

[https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S1\\_SheetMusic.html](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
- **Rarely** directly used as input to neural network models
  - Exception: *optical music recognition* (OMR)



Figure 1.1 from [Müller, FMP, Springer 2015]



# 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



A musical staff with a treble clef. It consists of eight notes: a whole note (C4), a half note (D4), a quarter note (E4), a half note (F4), a whole note (G4), a half note (A4), a whole note (B4), and a half note (C5). The notes are separated by vertical bar lines.

C minor: 半全全全全半全



A musical staff with a treble clef. It consists of eight notes: a half note (C4), a whole note (D4), a half note (E♭4), a whole note (F4), a half note (G4), a half note (A♭4), a whole note (B♭4), and a half note (C5). The notes are separated by vertical bar lines.

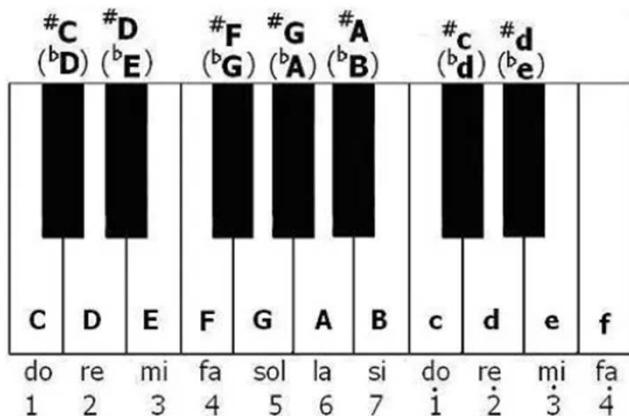
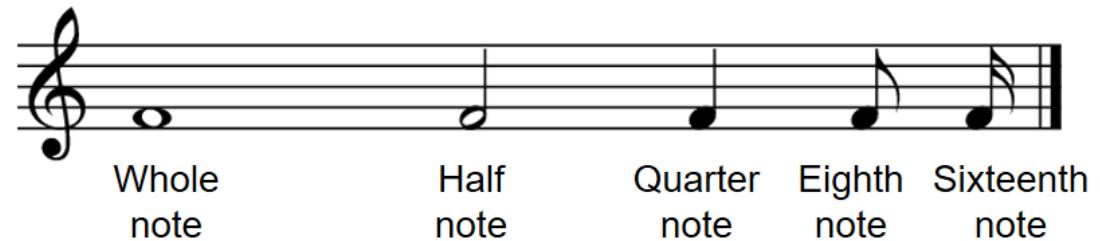


Figure 1.7 from [Müller, FMP, Springer 2015]

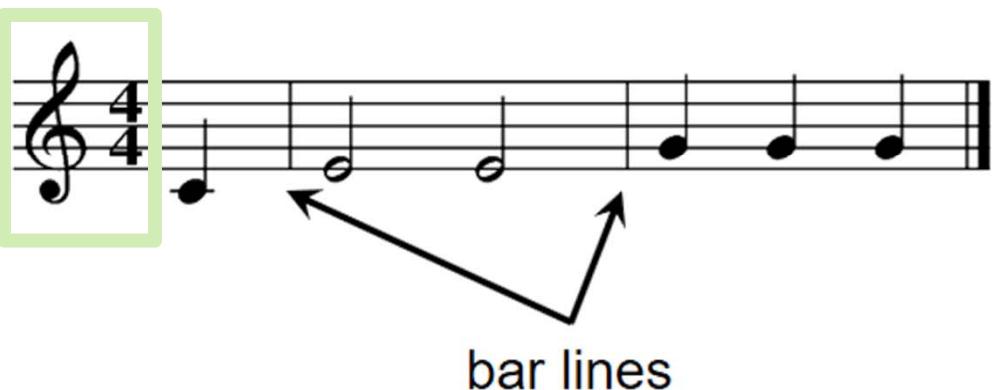


A musical staff showing five notes of decreasing size. From left to right, the notes are: Whole note, Half note, Quarter note, Eighth note, and Sixteenth note. Below each note, its name is written: Whole note, Half note, Quarter note, Eighth note, Sixteenth note.

# Sheet Music: Meter, Bar, Beat, Rhythm

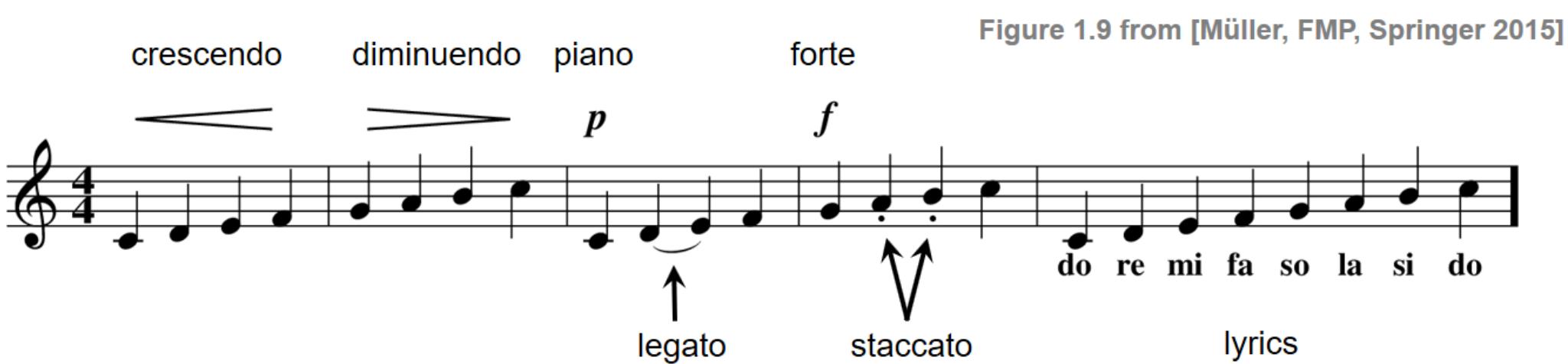
- Dividing music into measures (bars) not only reflects its rhythmic nature, but also provides regular reference points within it

Figure 1.6 from [Müller, FMP, Springer 2015]



# Sheet Music: Articulation Marks

- How certain notes are to be played



# Sheet Music vs. Lead Sheet

## A S I T W A S for string orchestra

Words and Music by Harry Styles,  
Thomas Hull, and Tyler Johnson  
arr. Tate Commission

Driving Rock ♩ = 174

Violin I

Violin II

Viola

Violoncello

Contrabass

Vln. I

Vln. II

Vla.

Vc.

Ch.

D 4/4

**015 一件美事**

C E<sub>m</sub> A<sub>m</sub> D<sub>m</sub> F

5 | 1 1 2 3 5 5 3 2 2 1 - - - | 0 2 2 3 4 3 2 |

已 過 二十 世 紀 以 來， 千 千 萬 萬 寶 貴

A<sub>m</sub> G<sub>7</sub> E<sub>m</sub> A<sub>m</sub> D<sub>m</sub> G<sub>7</sub> E<sub>m</sub>

1. 2 2 3 4 | 5. 1 1 1 2 3 | 4. 3 2 3 4 | 5 5 5 1 |

的 性 命 心 愛 的 奇 珍， 崇 高 的 地 位 以 及 燦 爛 的 前

F E<sub>m</sub> A<sub>m</sub> D<sub>m</sub> G<sub>7</sub> C F G<sub>7</sub>

6 6 5 4 | 5. 1 1 2 3 | 4. 3 2 1 7 | 1 - - 1 1 | 6. 6 5 |

途， 都 曾 枉 費 在 主 耶 穌 身 上， 對 這 些 愛 主

C F G<sub>7</sub> C D<sub>m</sub> G<sub>7</sub>

2 | 3 - - 1 1 | 6 - 5 2 | 3 - - 2 3 | 4 4 4 3 2 2 2 3 |

的 人， 祂 是 全 然 可 愛， 祂 是 全 然 可 愛 配

D<sub>m</sub> G<sub>7</sub> F G<sub>7</sub> C A<sub>m</sub>

4 4 3 4 5 5 6 | 5. 0 1 || 6. 6 7 . 7 | 1 7 6 6 6 7 |

得 我 們 獻 上 一 切。 我 們 濡 在 主 身 上 的 不 是

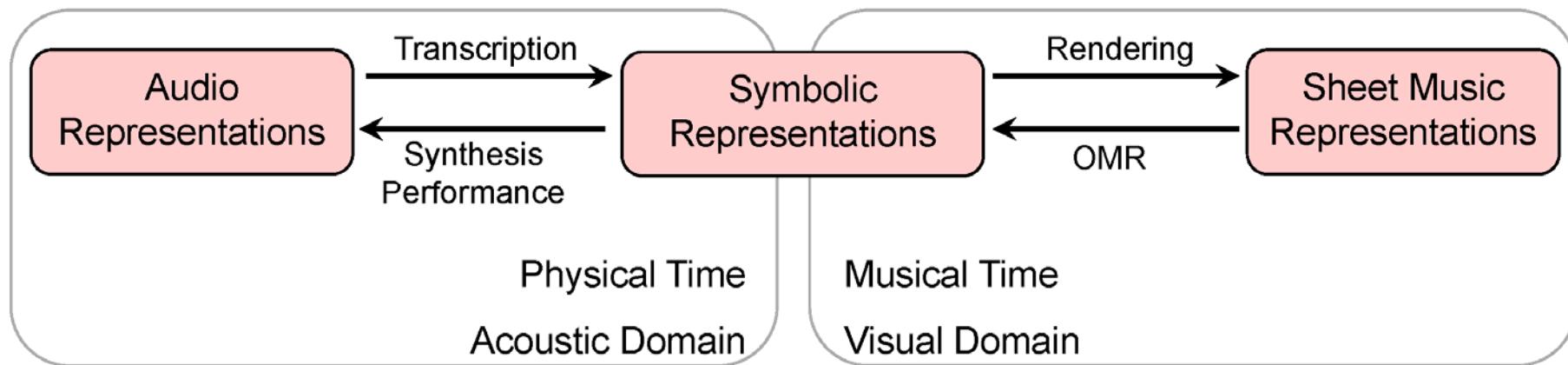
C D<sub>m</sub> G<sub>7</sub> D<sub>m</sub> G<sub>7</sub> C C

1 5 5 - 4 3 | 4 4 4 3 2 2 2 3 | 4. 3 2 2 | 3 -- 0 1 || 1 -- ||

枉 費， 乃 是 馨 香 的 見 證， 見 證 祂 的 甘 甜。 我 甜。

# Data Representation of Music

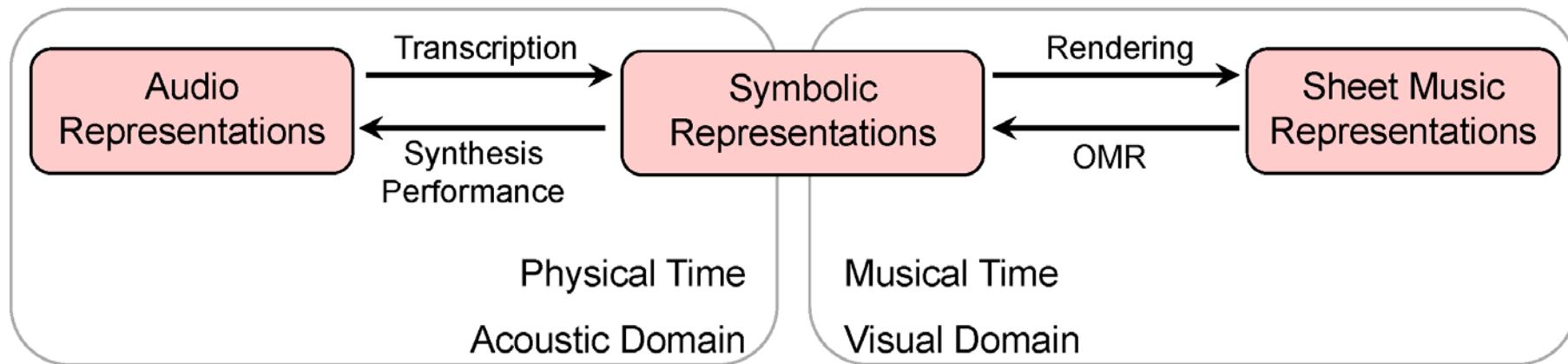
(Figure source: <https://www.mdpi.com/2624-6120/2/2/18>)



- **Musical audio** (waveforms, spectrograms)
  - what we “hear”
  - involve sounds
- **Symbolic representations** (e.g., **MIDI**, **MusicXML**)
  - the way music can be represented in a Digital Audio Workstation (DAW)
  - still relevant today as it’s *interpretable* and *editable*
- **Sheet music** (visual representations of a musical score)
  - what musicians “compose”

# Sheet Music and Symbolic-domain Representations

(Figure source: <https://www.mdpi.com/2624-6120/2/2/18>)



- **Physical time:** in milliseconds or alike, good for representation **performances** (*timing variations*)
- **Musical time:** 1/4, 1/8, 1/16 etc, good for representation **scores**
- Symbolic music can take either physical time or musical time

# Piano Roll Representation

[https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2\\_PianoRoll.html](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**)

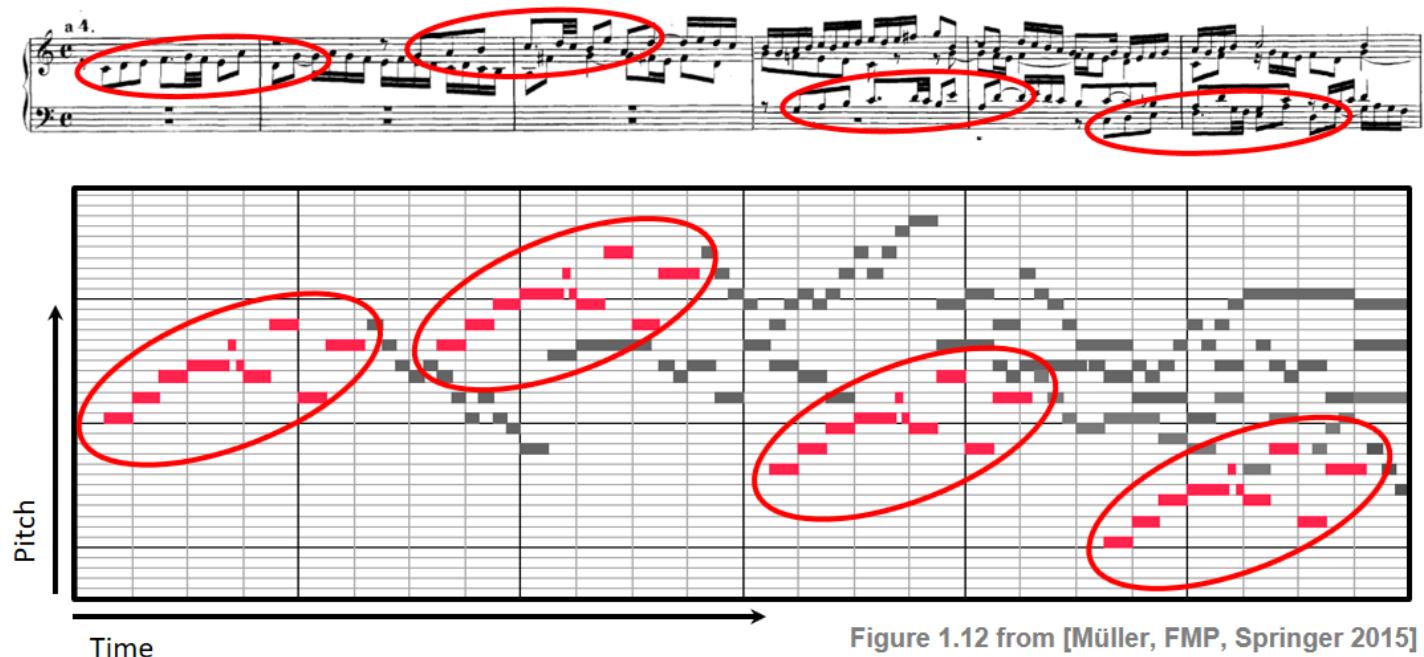
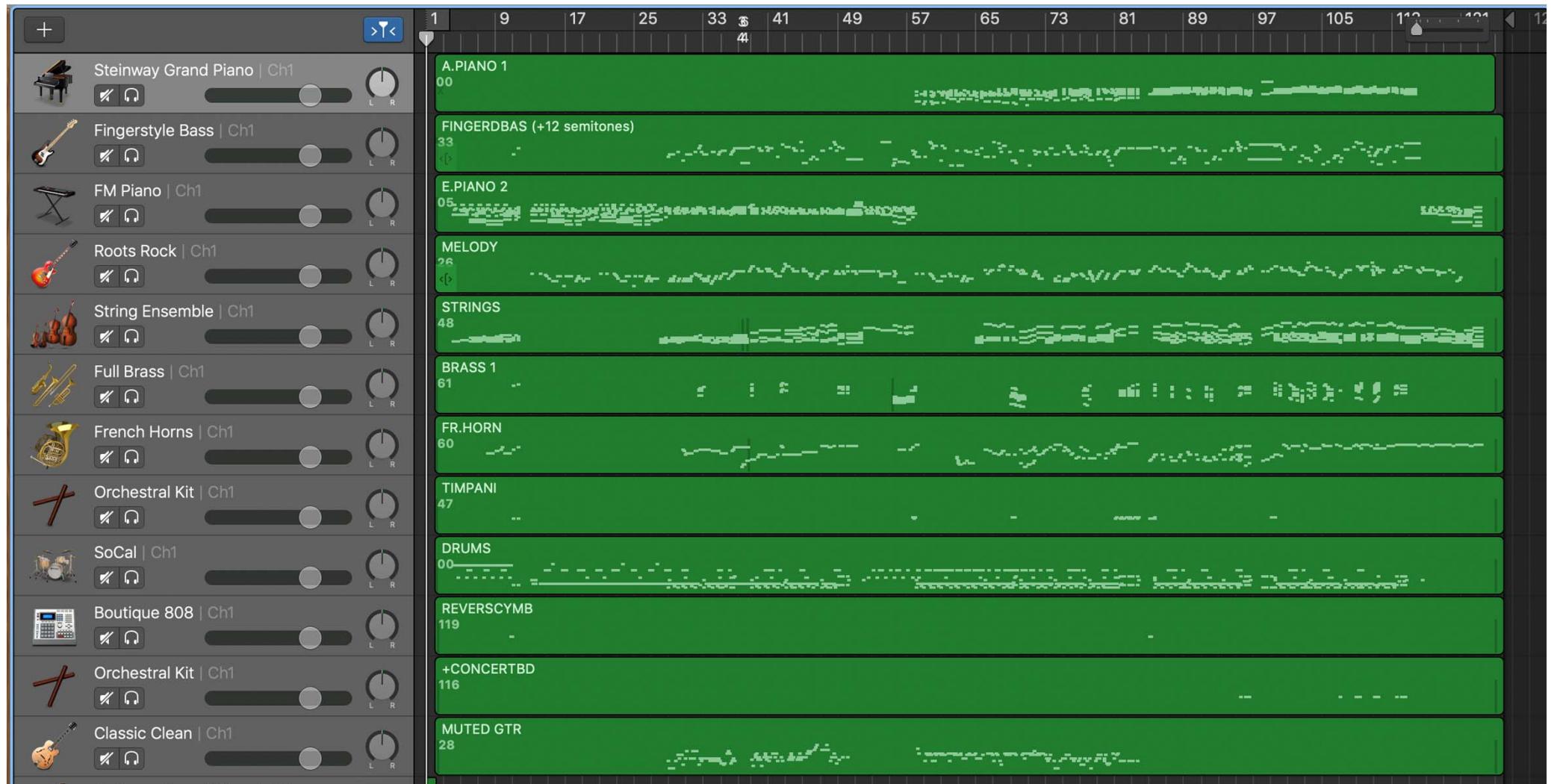


Figure 1.12 from [Müller, FMP, Springer 2015]

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

# Piano Roll Representation

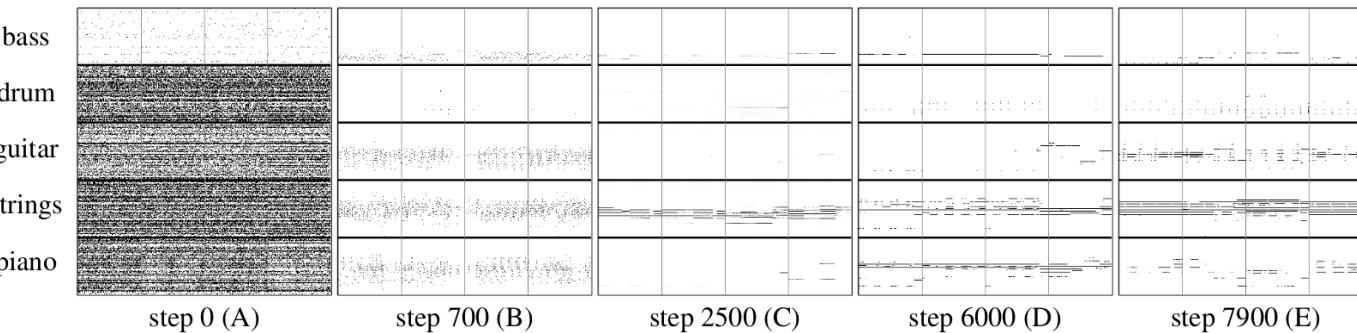
(image from the internet)



# 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

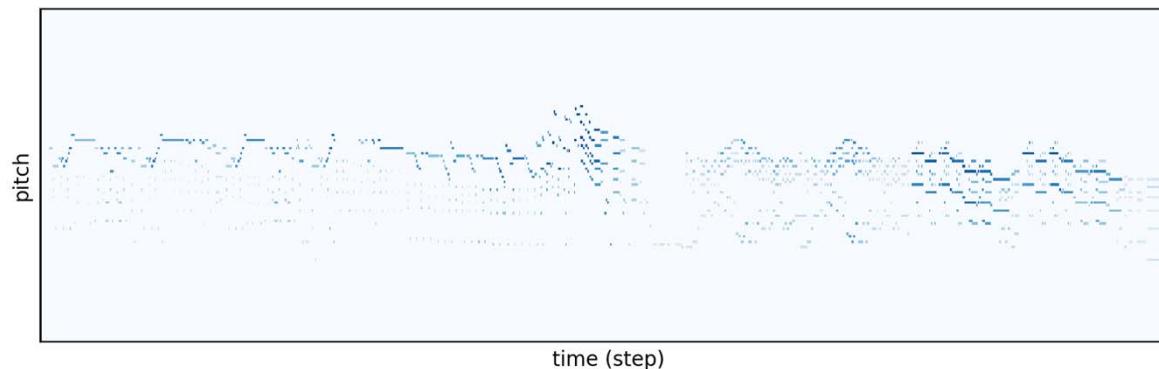


DiffRoll



# Library: PyPianoroll

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



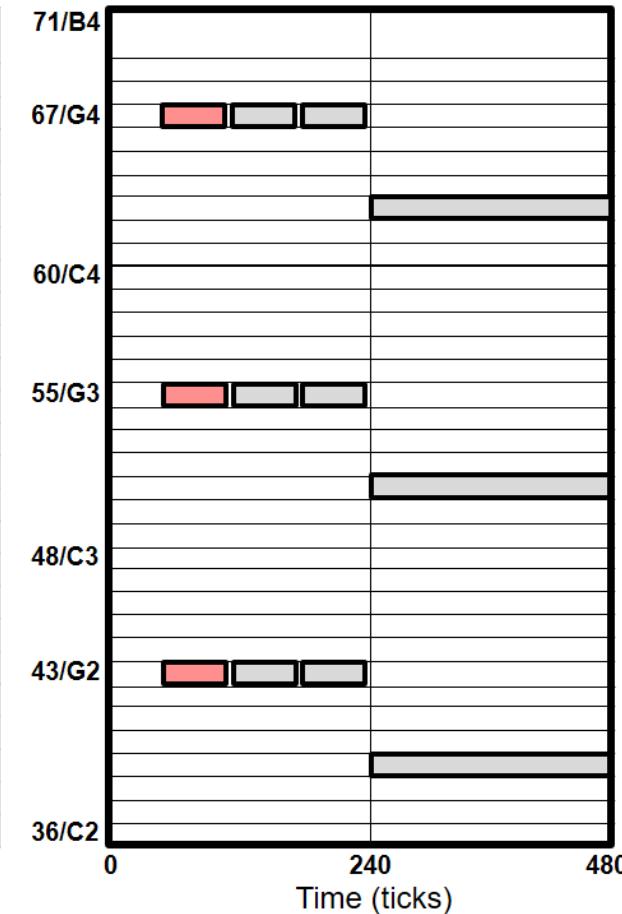
- Features
  - Manipulate multitrack piano rolls intuitively
  - Visualize multitrack piano rolls beautifully
  - Save and load multitrack piano rolls in a space-efficient format
  - Parse **MIDI** files into multitrack piano rolls
  - Write multitrack piano rolls into **MIDI** files

# MIDI Representation

[https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2\\_MIDI.html](https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_MIDI.html)

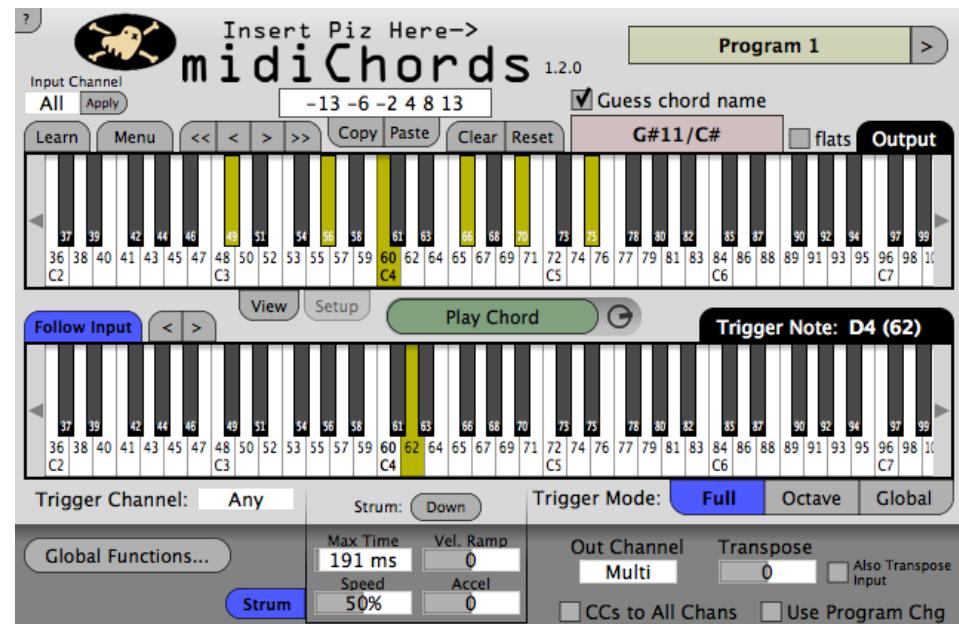
- A ***text-like*** representation of  
\*.MIDI, using **MIDI messages**  
and **timestamps**
  - *MIDI note number* (0-127)
  - *Key velocity* (0-127): intensity  
of the sound
  - *MIDI channel* (different  
instruments)
  - *Time stamp*: how many *clock  
pulses or ticks to wait* before  
the command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

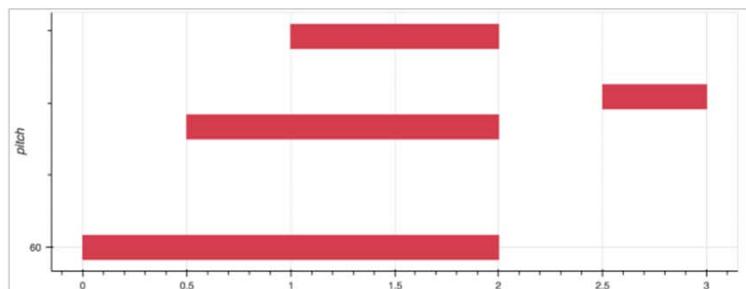
- Both represent a \*.MIDI file
  - Piano roll: image-like
  - MIDI Representation: text-like
- Piano roll are like fixed-shape images
- MIDI representation is more flexible as it allows for *additional information* to be added
  - For example, **chord tokens**, **key tokens**, etc



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
- 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

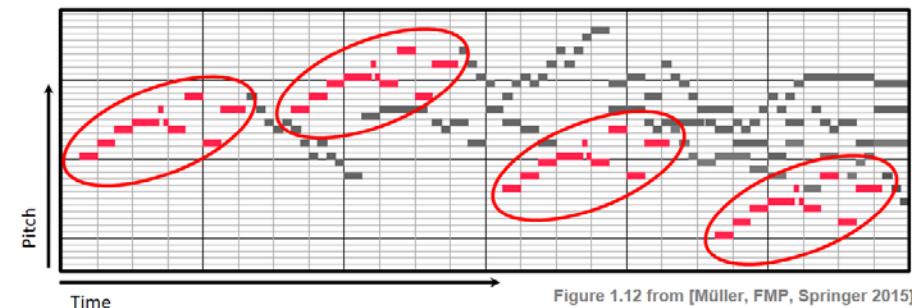
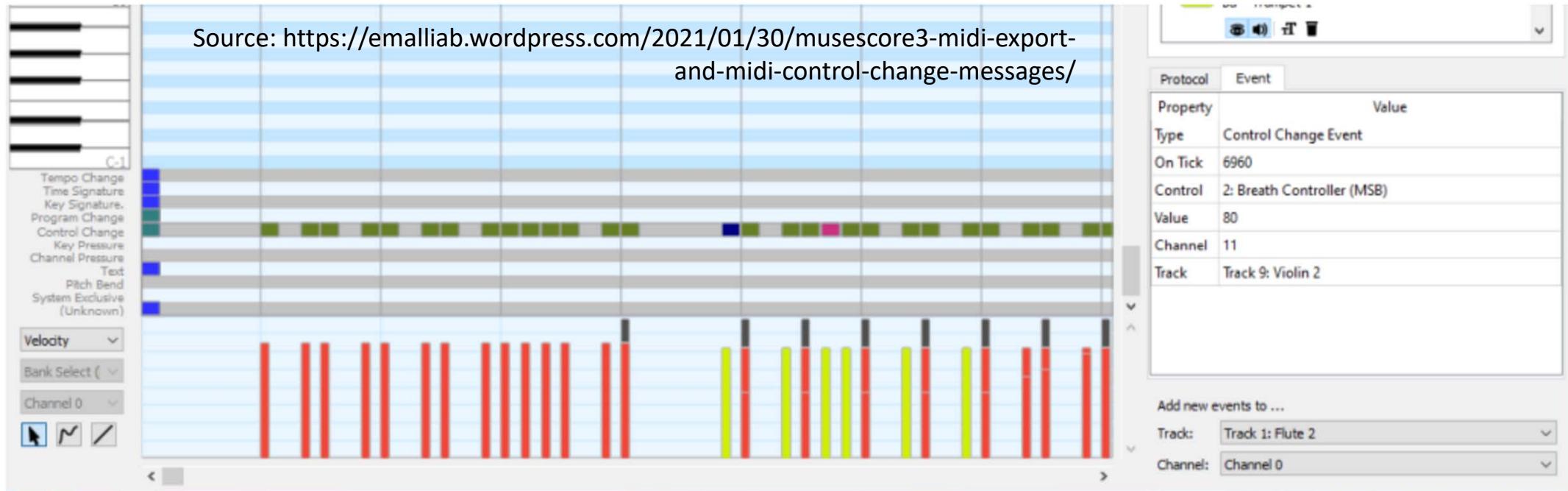


Figure 1.12 from [Müller, FMP, Springer 2015]

Ref1: <https://salu133445.github.io/lakh-pianoroll-dataset/representation.html>

Ref2: [https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2\\_MIDI.html](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

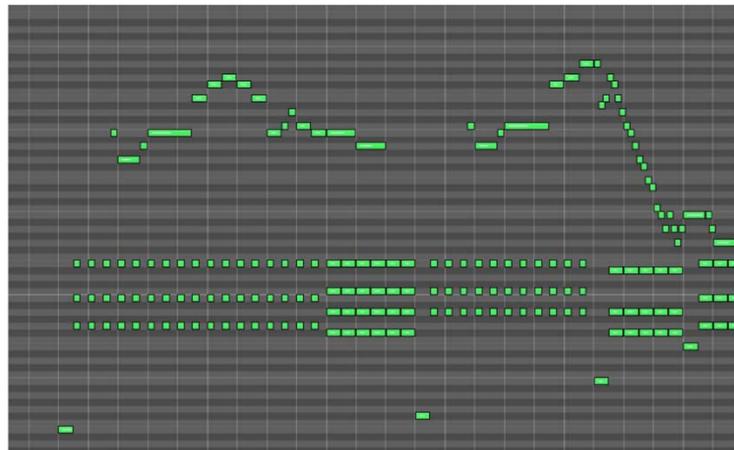
Protocol	Event	Protocol	Event	Protocol	Event
Property	Value	Property	Value	Property	Value
Type	Program Change Event	Type	Program Change Event	Type	Program Change Event
On Tick	768	On Tick	1535	On Tick	3069
Program	1: Bright Acoustic Piano	Program	2: Electric Grand Piano	Program	3: Honky-tonk Piano
Channel	0	Channel	0	Channel	0
Track	Track 0: Asturias(Leyenda)	Track	Track 0: Asturias(Leyenda)	Track	Track 0: Asturias(Leyenda)

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

# MIDI Score vs MIDI Performance

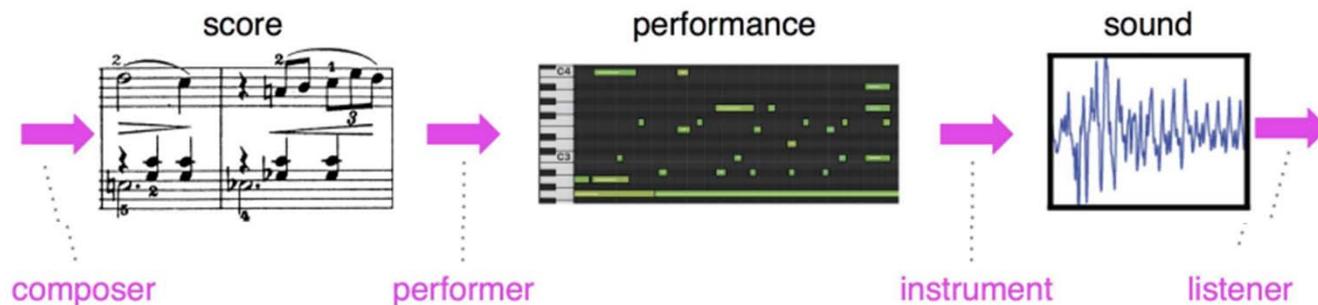


Figure 1: Excerpt from the score of Chopin's Piano Concerto No. 1.



- MIDI score
  - <https://clyp.it/jhdkgħso>
- MIDI performance
  - <https://clyp.it/x24hp1pq>

# 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

# 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

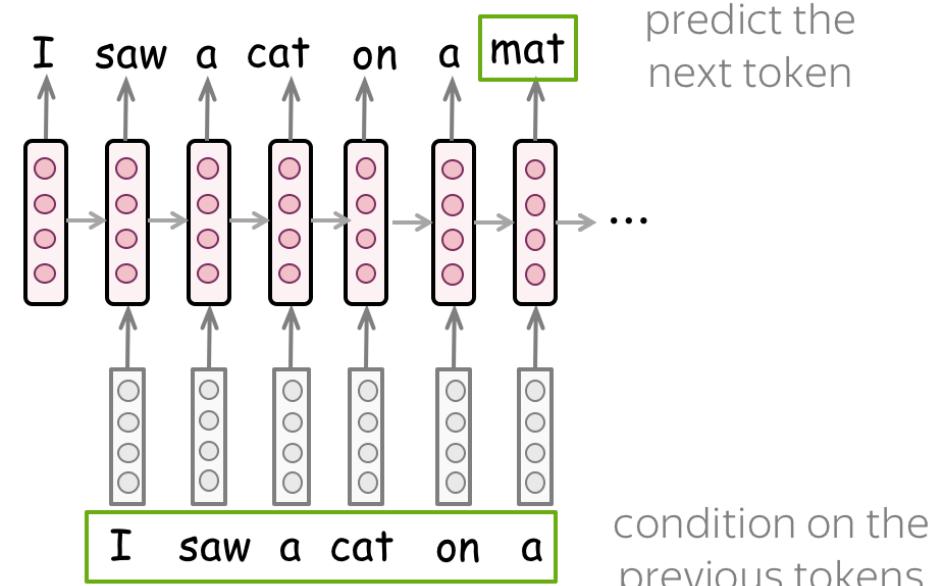
# Outline

- Sheet music & symbolic representations for music
- **Token representation of MIDI music**
- MIDI generation
- 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](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")

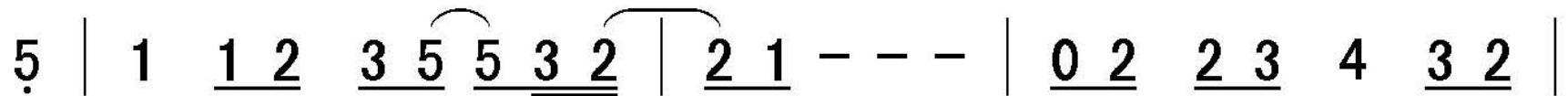
# Representing Symbolic Music as Tokens

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

5 | 1 1 2 3 5 5 3 2 | 2 1 - - - | 0 2 2 3 4 3 2 |

# 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 as Tokens

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

5 | 1 1<sub>2</sub> 3<sub>5</sub> 5<sub>3</sub><sub>2</sub> | 2<sub>1</sub> --- | 0<sub>2</sub> 2<sub>3</sub> 4 3<sub>2</sub> |

- [G3: $\frac{1}{4}$ ] [C4: $\frac{1}{4}$ ] [C4: $\frac{1}{8}$ ] [D4: $\frac{1}{8}$ ] ...
- Each musical note is uniformly represented by **one token**
- Shorter sequence length but larger vocabulary size

# 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= $\frac{1}{4}$ ] [pitch] [duration] [rest=0] [pitch] [duration] [rest=0]

# Representing Symbolic Music: Lead Sheet

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

**C**                    **E<sub>m</sub>**                    **A<sub>m</sub>**                    **D<sub>m</sub>**                    **F**  
5 | 1 1 2 3 5 5 3 2 | 2 1 - - - | 0 2 2 3 4 3 2 |

- [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)?

C                    E<sub>m</sub>                    A<sub>m</sub>                    D<sub>m</sub>                    F  
5 | 1 1 2 3 5 5 3 2 2 1 - - - | 0 2 2 3 4 3 2 |  
已 過 二十世紀 以 來， 千 千萬 萬 寶貴

$\text{♩} = 120$

The musical score consists of two parts. The first part shows a melody line with lyrics: "Listen to the rhythm of the falling rain telling me". The second part shows a continuation of the melody with lyrics: "已過二十世紀以來，千千萬萬寶貴". The music is in 4/4 time, treble clef, and includes various note values like eighth and sixteenth notes, along with rests. A pink box highlights a rest in the second part of the score.

Listen to the rhythm of the fall  
ing rain telling me

已過二十世紀以來，千千萬萬寶貴

# 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”

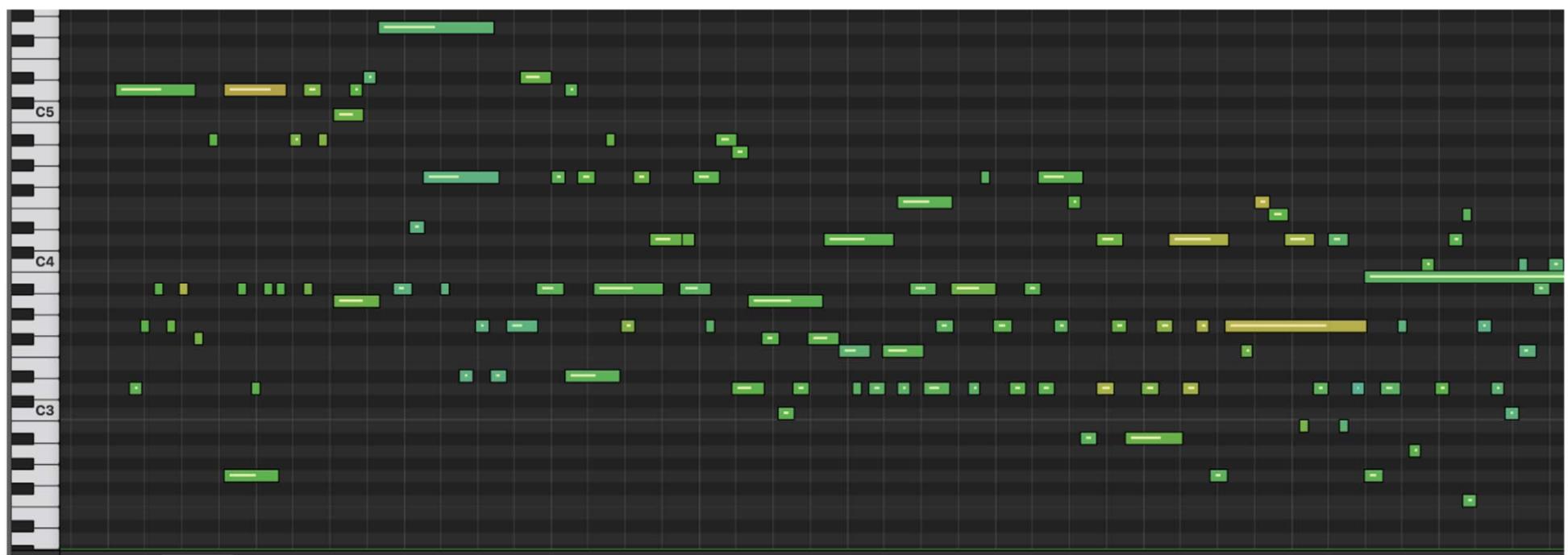
# Representing Symbolic Music: Piano Music Score

- How to represent a sequence of piano music score?
- Add one additional token to mark “**time-shift**” (i.e.,  $\Delta T$ ; interval time)
- [pitch] [duration] [time-shift] [pitch] [duration] [time-shift] ...

A piano music score is shown on a staff system. The top staff is in treble clef and has a key signature of one sharp. The bottom staff is in bass clef and also has a key signature of one sharp. Both staves have a time signature of 4/4. Two specific notes are highlighted with red circles: note 'a' is a dotted half note in the treble staff, and note 'b' is a quarter note in the bass staff. Below the staffs, a red box contains the text  $\Delta T = 0$ .

# 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



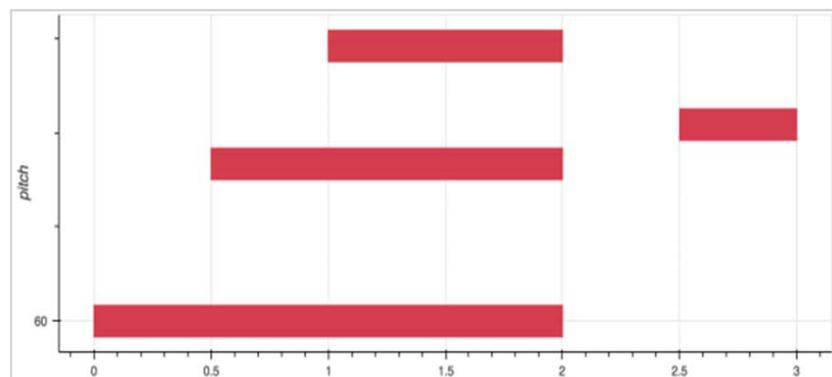
# Text vs. Symbolic-domain Music

1. **Temporally overlapping** events (notes) in music
  - Need a way to represent time
2. A musical note is associated with **multiple attributes**
  - Pitch, duration, instrument, velocity, etc
3. **Rich mid-level features**
  - Chord progression, grooving pattern, etc
4. **Music is long (>4k tokens)**
5. **Music form/structure**
  - Verse, chorus, etc
  - Like a story



# 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)
(2) piano:v72:G1 piano:v72:G2 piano:v72:B4 piano:v72:D4 violin:v80:G4
(3) (4) (5)
(6) piano:v72:G4 piano:v72:B5 piano:v72:D5 wait:12 (7) (8)
      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

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
```

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

# Representing Multi-track Music

OpenAI's MuseNet

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

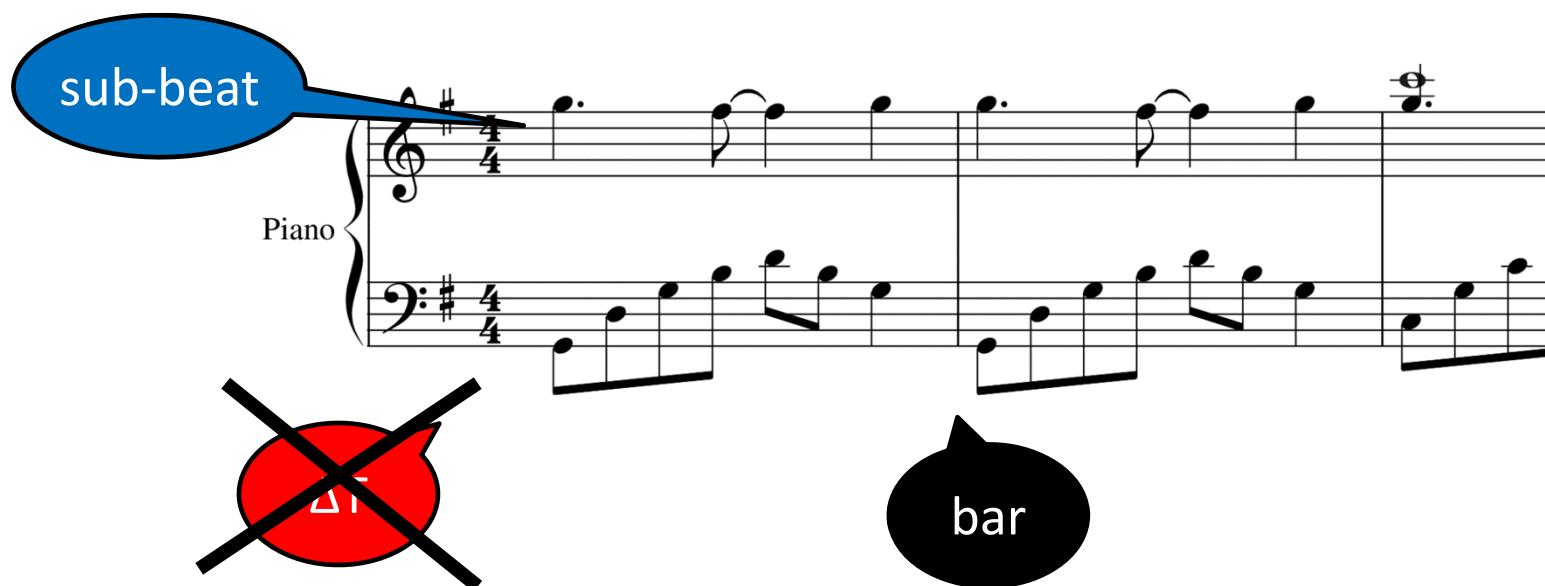
v — **velocity** (volume), in [0, 127]  
v0 — **offset** of a note  
v72 — **onset** of a note  
G1 — **pitch**  
piano — **instrument**  
wait — move the **time** marker

```
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!

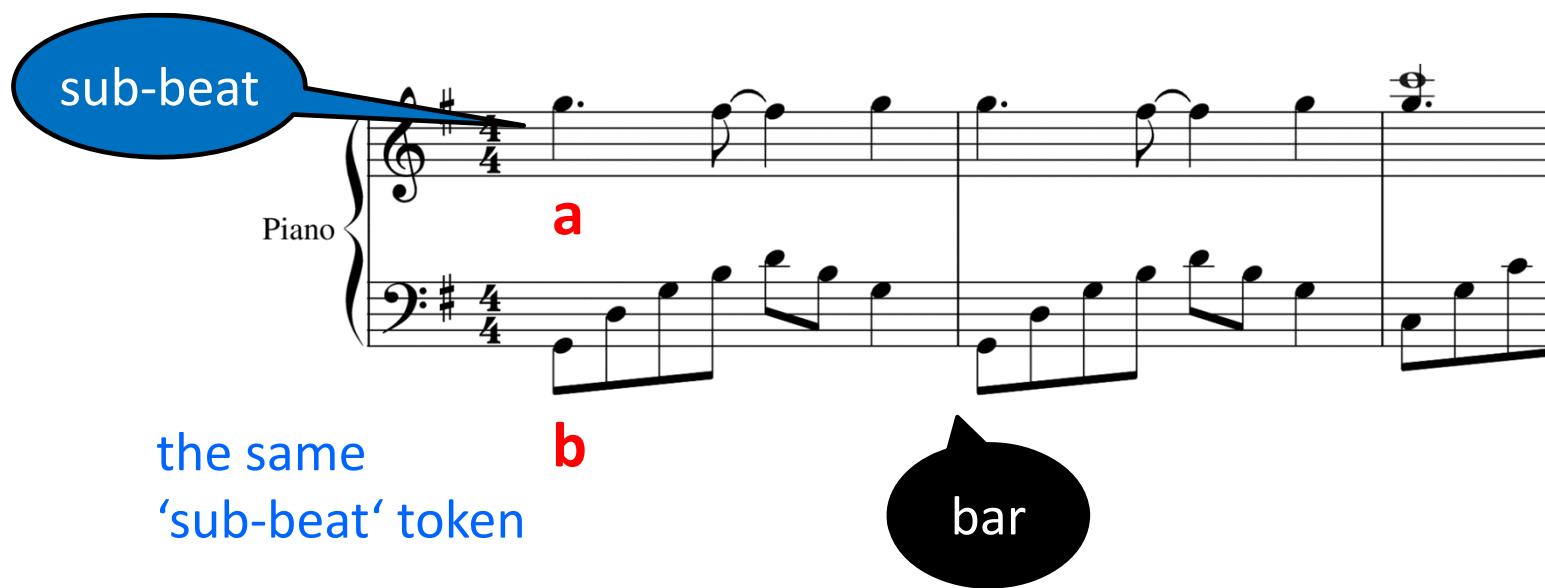
# REMI: Revamped MIDI Representation

- **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)



# 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)



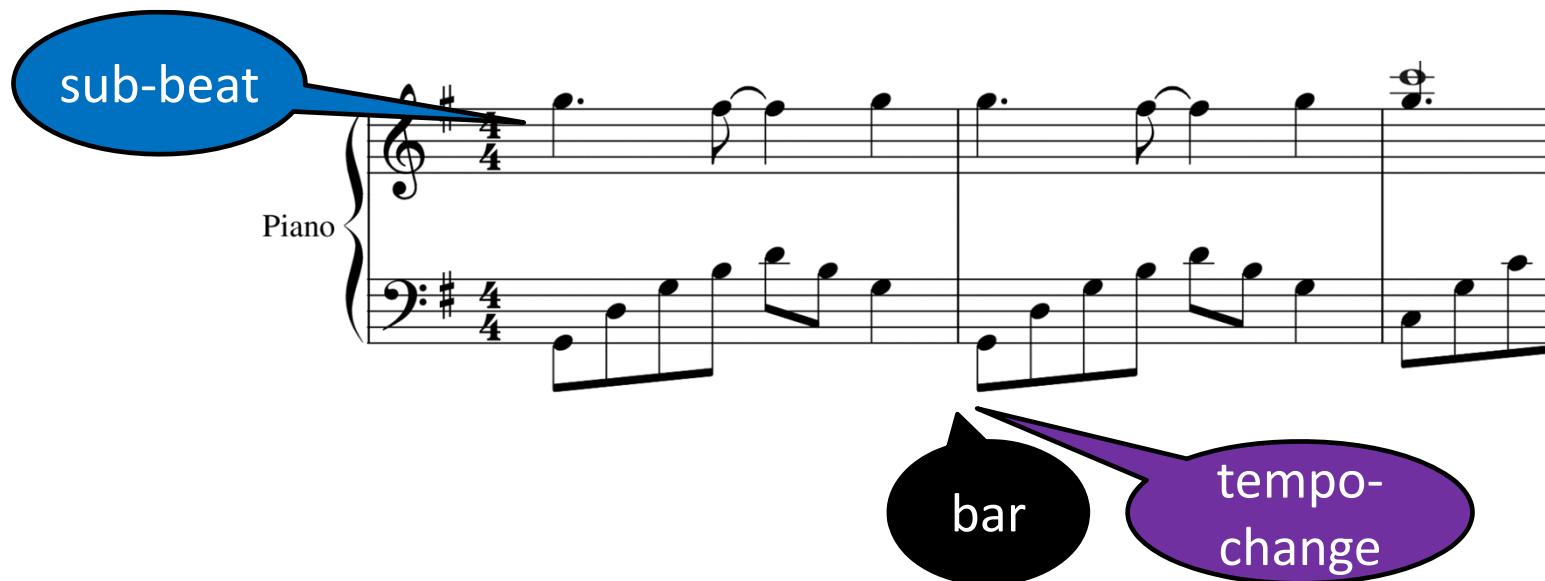
## 16th Notes

(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	x	TEMPO (30-209 BPM)
Chord	x	CHORD (60 types)

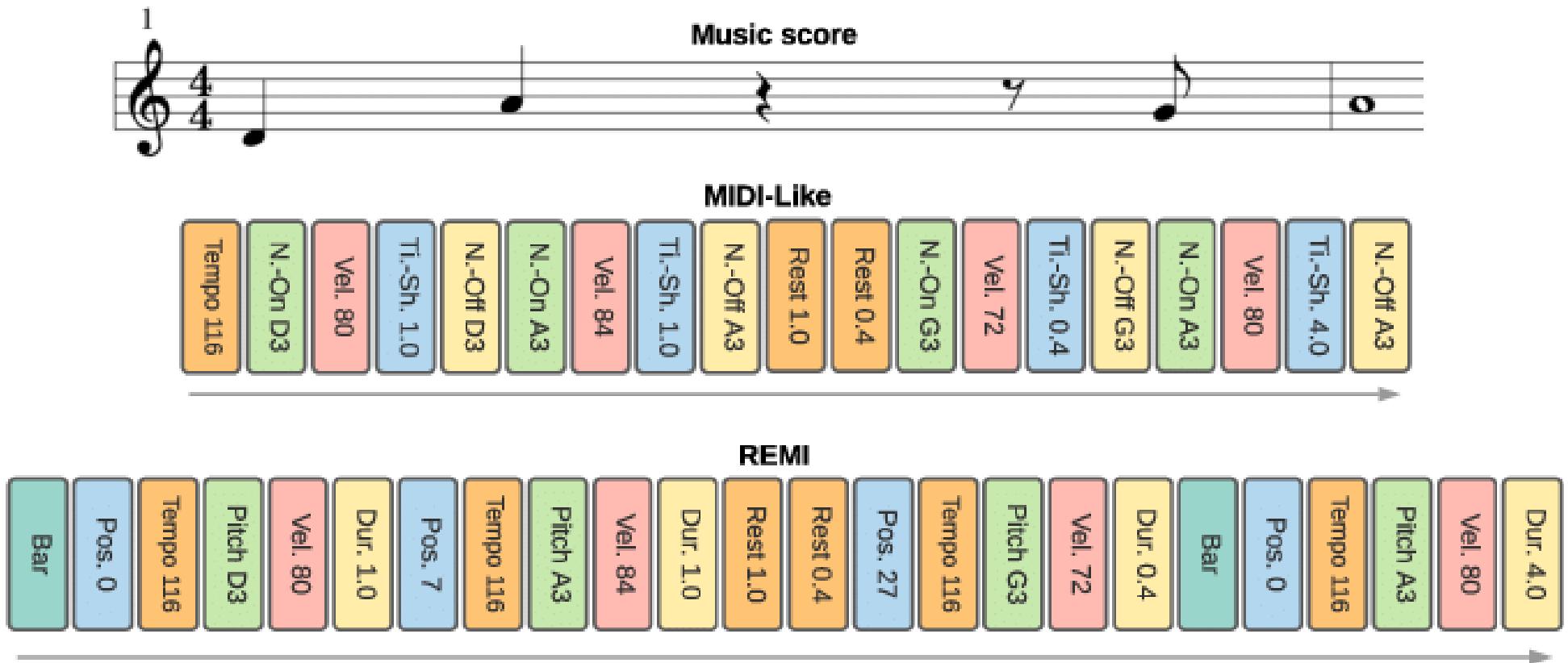
Table 1: A comparison of the commonly-used MIDI-like event representation with the proposed one, REMI. In the brackets, we show the corresponding ranges.

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

# 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/>

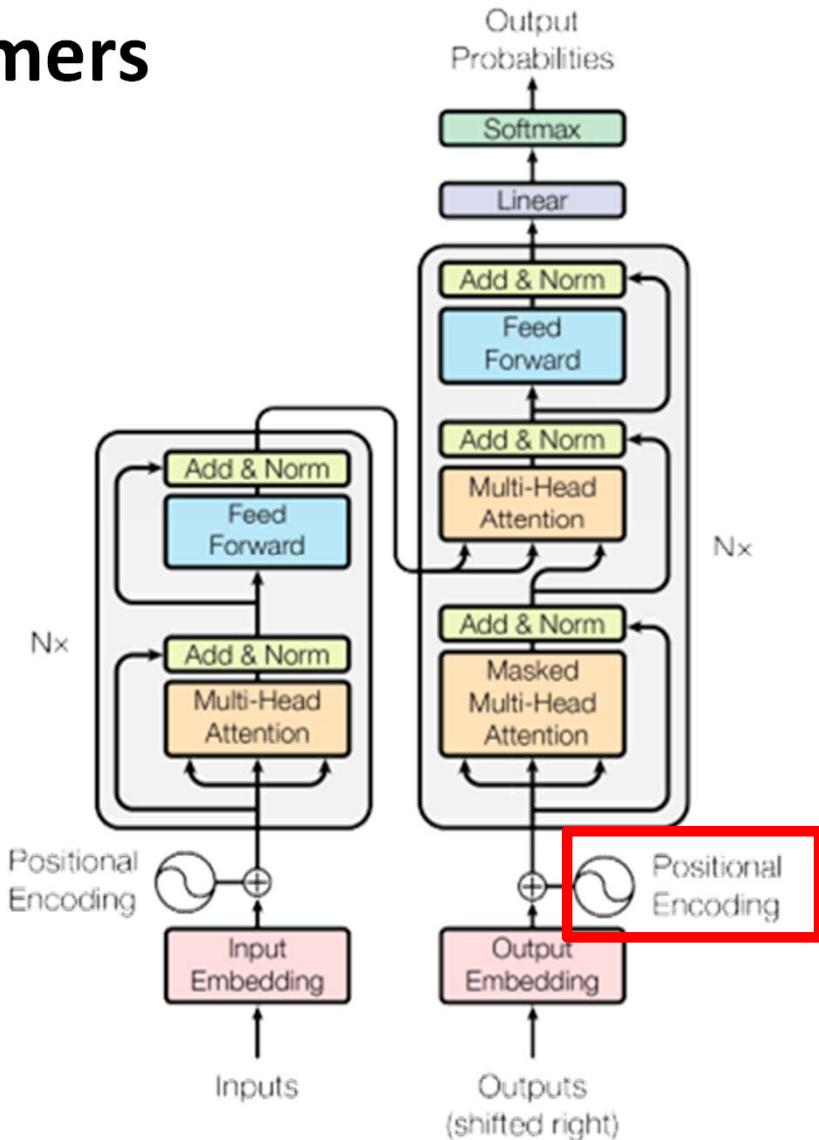
- “MidiTok can take care of converting (tokenizing) your symbolic music files (**MIDI, abc**) into **tokens**, ready to be fed to models such as **Transformer**, for any generation, transcription or MIR task”
  - “MidiTok features most known MIDI tokenizations (e.g. REMI, Compound Word...), and is built 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
- **MIDI generation**
- Evaluation metrics

# Transformers

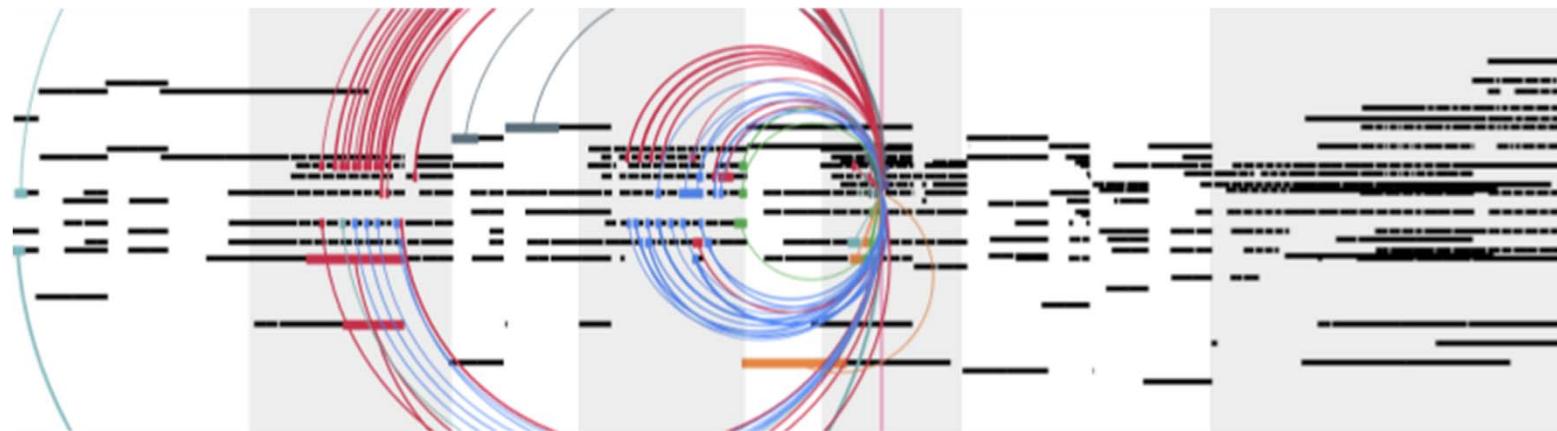
- There are encoders & decoders
  - Encoder: cross-attention
  - Decoder: self-attention
- **Decoder-only** architecture
  - Usually used for unconditional, or prompt-based generation
  - Mainly talk about this one today
- **Encoder/decoder** architecture
  - Usually used for sequence-to-sequence generation



# Transformer Example 1: Music Transformer [huang19iclr]

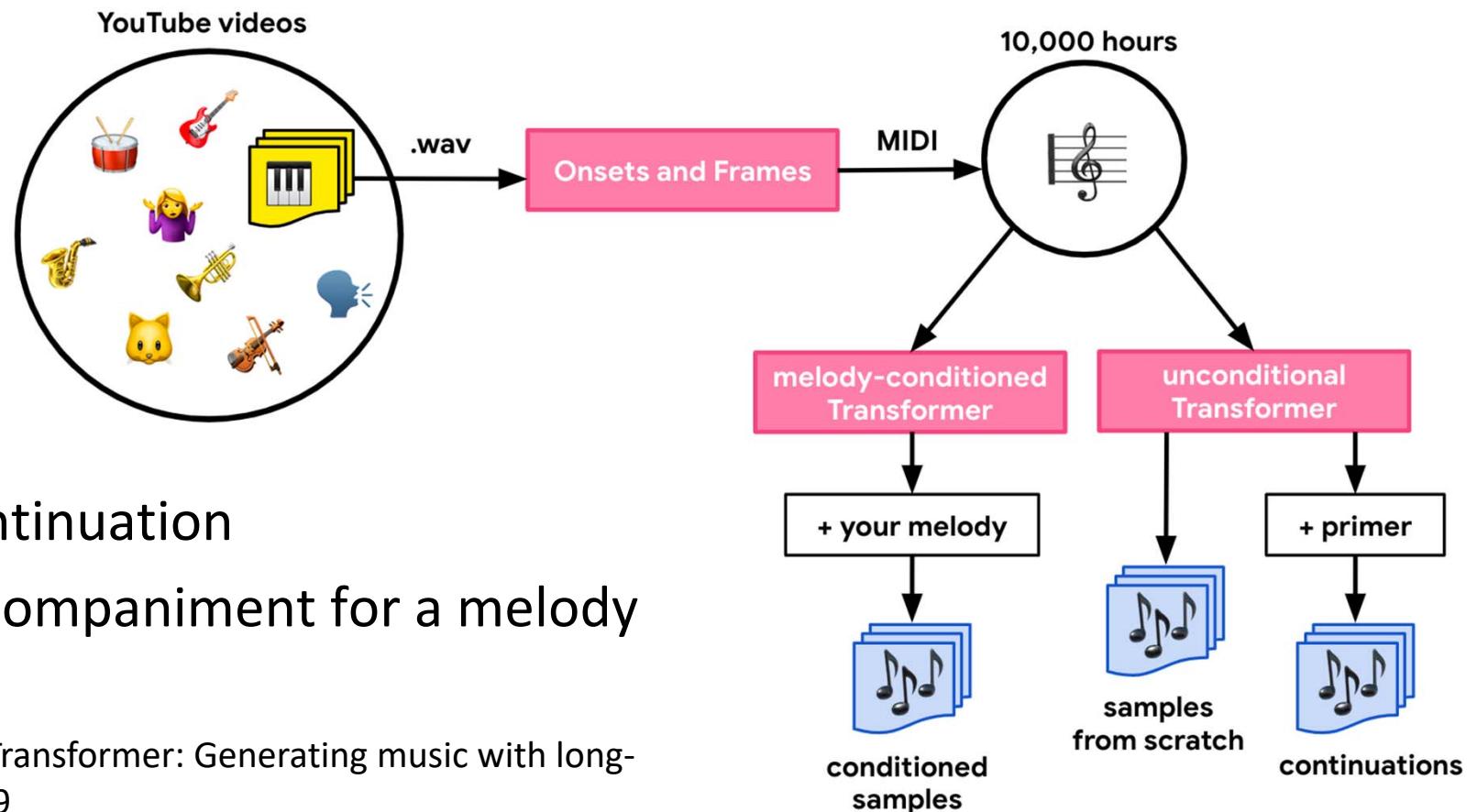
<https://magenta.github.io/listen-to-transformer/>

- Generate full piano performances (one note at a time)
  - “... can generate **minute-long** compositions (thousands of steps) with compelling structure, generate continuations that coherently elaborate on a given motif, and in a seq2seq setup generate accompaniments conditioned on melodies.”



# Music Transformer [huang19iclr]

<https://magenta.tensorflow.org/piano-transformer>



## Transformer Example 2: MuseNet [payne19]

<https://openai.com/blog/musenet/>

- Generate multi-track MIDIs by a **72-layer** sparse Transformer decoder
  - *Instrument-related* tokens

```
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
```

- Another early multitrack Transformer: LakhNES [donahue19ismir]

Ref 1: Payne et al, “MuseNet,” OpenAI blog, 2019

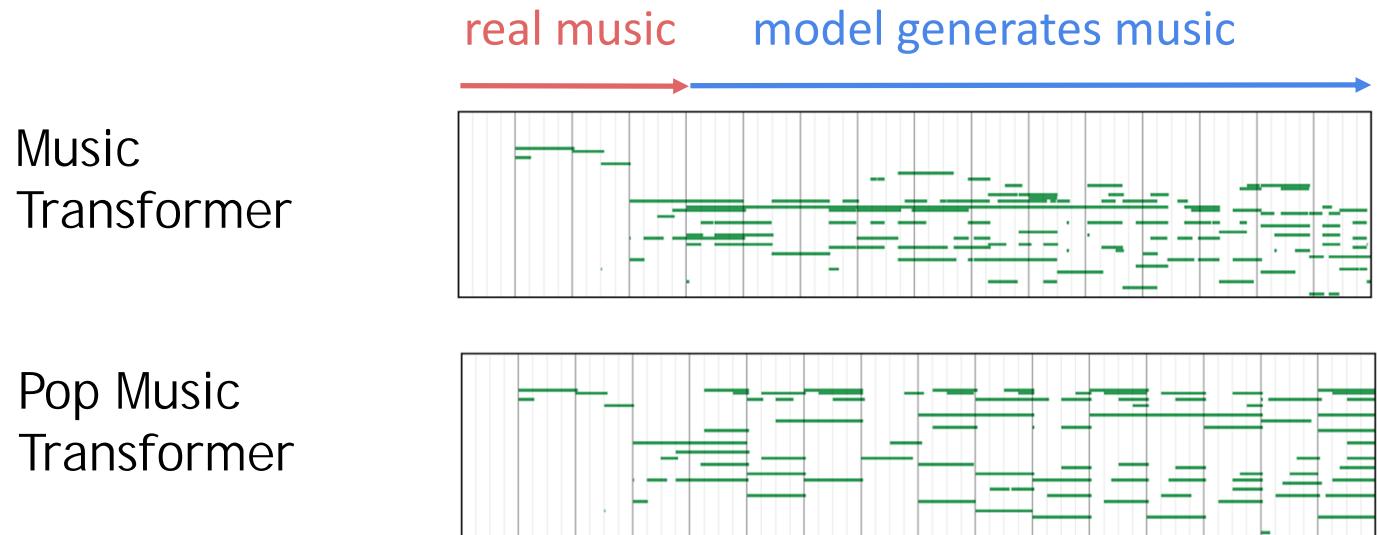
Ref 2: Donahue et al, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training”, ISMIR 2019

# Transformer Example 3: Pop Music Transformer [huang20mm]

<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	<i>p</i> -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.

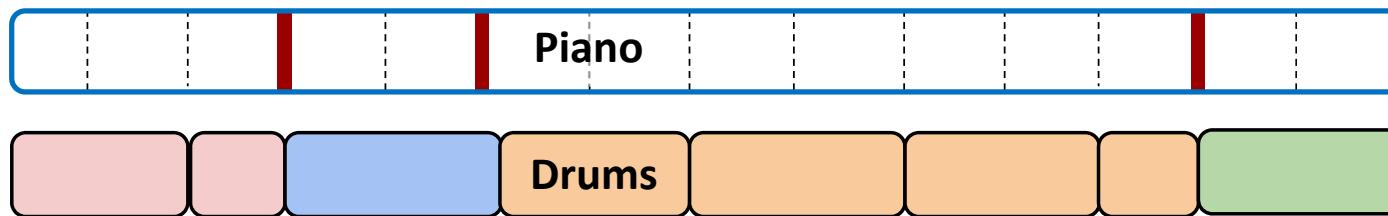
Method	Note offset	Time grid	TEMPO	CHORD	Beat STD	Downbeat STD	Downbeat salience
Baseline 1	NOTE-OFF	TIME-SHIFT (10-1000ms)			0.0968	0.3561	0.1033
Baseline 2	DURATION	TIME-SHIFT (10-1000ms)			0.0394	0.1372	0.1651
Baseline 3	DURATION	TIME-SHIFT (16th-note multiples)			0.0396	0.1383	0.1702
REMI	DURATION	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

piano + drum      guitar

# Pop Music Transformer



- **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-piano-generation-demo-202004)
  - [https://soundcloud.com/yating\\_ai/sets/ai-pianodrum-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-KHozfWOPTv5](https://soundcloud.com/yating_ai/ai-guitar-tab-generation-202003/s-KHozfWOPTv5)

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

# **Building Your First MIDI Transformer Decoder**

- Step (1): Task & Data
- Step (2): Token representation
- Step (3): Choose a model
- Step (4): Train the model till the loss is sufficiently low
- Step (5): Do inference
- Step (6): Listen to the generated music!
- Step (7): Do evaluation

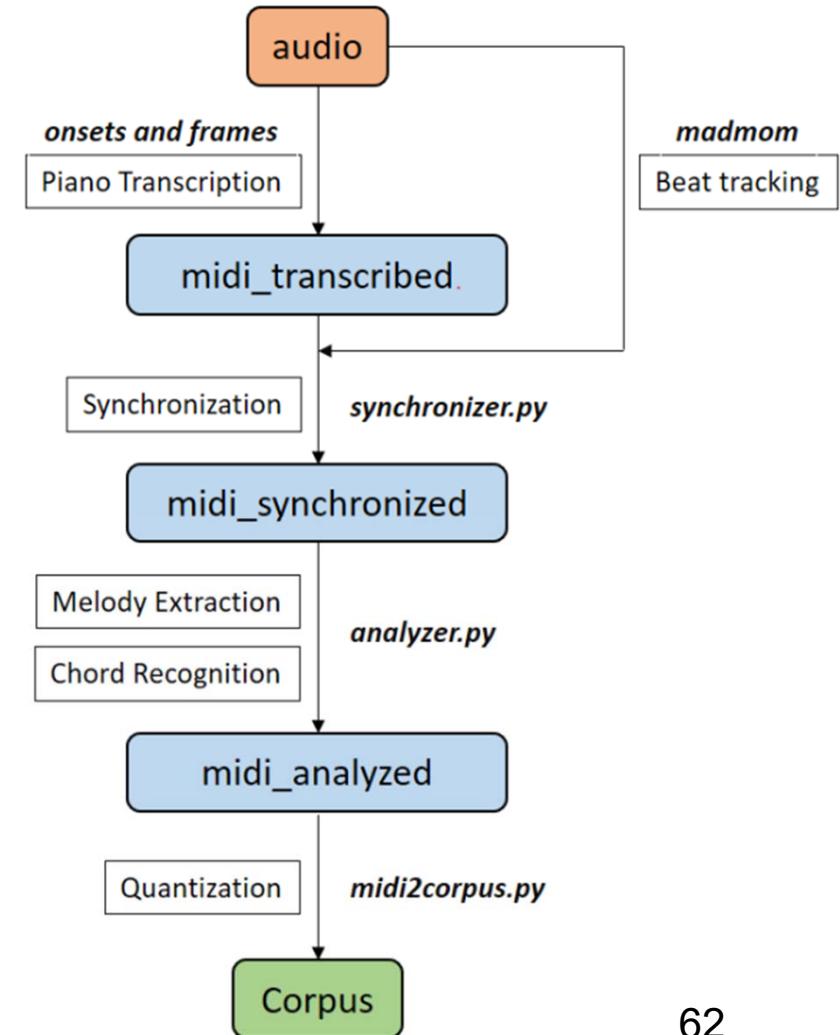
# DIY Step (1): Task & Data

- Choose a task, and a dataset
  - [https://github.com/affige/genmusic\\_demo\\_list](https://github.com/affige/genmusic_demo_list)
- List of symbolic musical datasets
  - <https://github.com/wayne391/symbolic-music-datasets>
- Three relatively more frequently studied cases
  - **Lead sheets** (MIDI score) — Jazz Transformer (ISMIR'20), Compose & Embellish (ICASSP'23)
  - **Multi-track MIDI** (MIDI score) — Multitrack Music Transformer (ICASSP'23)
  - **Piano performance** — Pop Music Transformer (MM'20), Compose & Embellish (ICASSP'23)

# DIY: To Prepare a Dataset of Piano Performances

<https://github.com/YatingMusic/compound-word-transformer/blob/main/dataset/Dataset.md>

- Piano transcription
  - (“Onsets and frames” is no longer the SOTA → lecture 7)
- Beat and downbeat tracking
- Melody extraction
- Chord recognition
- Quantization
- Additional steps
  - Auto-tagging
  - Source separation



## DIY Step (2): Choose A Token Representation

- Ref: <https://miditok.readthedocs.io/>
- Single-track
  - MIDI-like, REMI, CP, etc
- Multi-track
  - REMI+, Octuple, MMM, MuMIDI, etc
- Depend on your GPU VRAM, you may need to decide a sequence length and cut your musical pieces into segments

## DIY Step (3): Choose A Model

- PyTorch 2 Transformers

<https://pytorch.org/blog/accelerated-pytorch-2/>

- Huggingface Transformer

<https://huggingface.co/docs/transformers/index>

- x-Transformers

<https://github.com/lucidrains/x-transformers>

## DIY Step (4): Train the Model

- Train the model till the loss is *sufficiently* low
- Our experience on pop piano
  - Training loss around **0.30** gives better result
  - The model **overfits** and generates overly repeating content when loss is nearly 0
  - Keep several **checkpoints** and **listen to** them to choose one
- No need to consider validation data
  - Unless you are building a conditional generation model with, for example, an encoder/decoder architecture

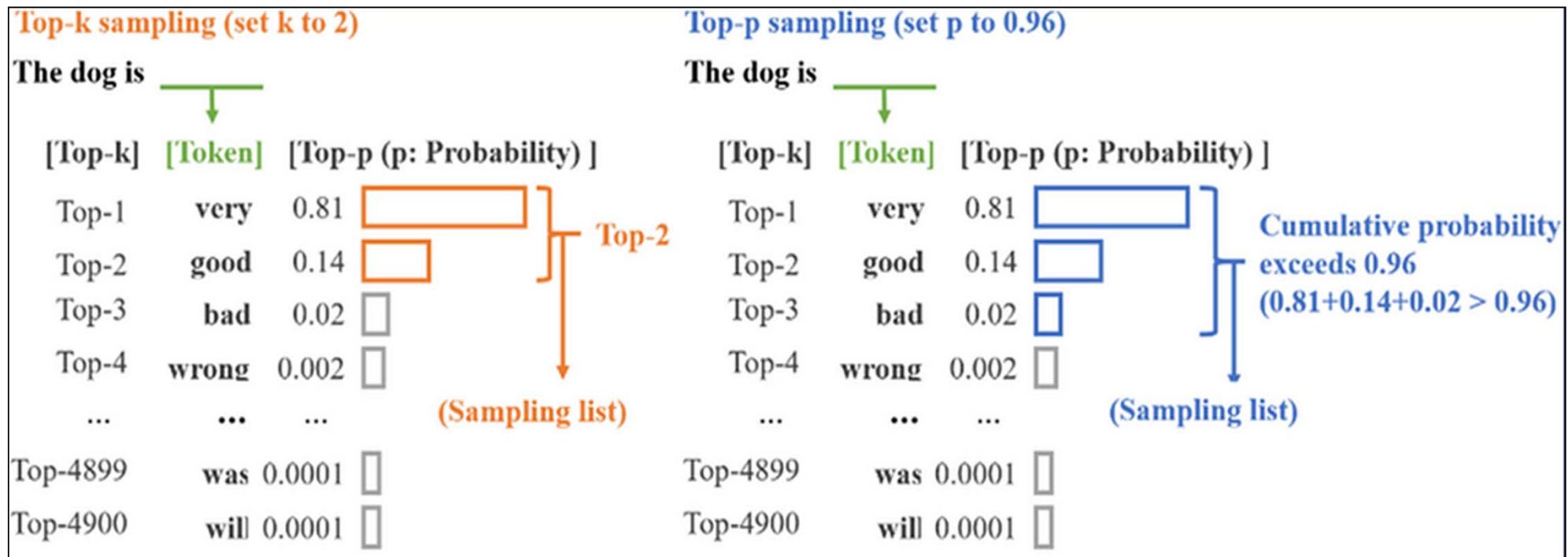
# DIY Step (5): Do Inference

[https://blog.csdn.net/qq\\_43243579/article/details/136331123](https://blog.csdn.net/qq_43243579/article/details/136331123)

[https://www.youtube.com/watch?v=\\_3DWwb96exY](https://www.youtube.com/watch?v=_3DWwb96exY)

- **Sampling strategy**
  - Greedy
  - Beam search
  - Random sampling
  - **Top-K Sampling**
    - Only top K probable tokens should be considered for a generation
  - **Nucleus Sampling**
    - Focuses on the smallest possible sets of top-V words with the sum of their probability is  $\geq p$
- **Temperature**

# Top-K Sampling & Nucleus Sampling



(image from the Internet)

- Top-K Sampling: fixed number of candidates
- Nucleus Sampling: variable number of candidates

# Temperature

- Tweak the prob. distribution *before* sampling
  - set the value **>1** for **diversity**

$$\frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

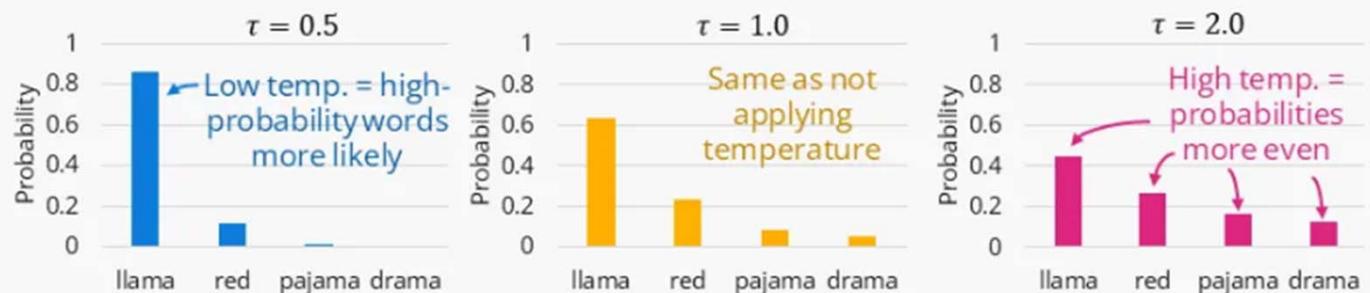
## TEMPERATURE SAMPLING

**Sampling** is a common method used by LLMs for generating output tokens: the next token is randomly chosen based on the token probabilities learned by the LLM. **Temperature sampling** affects the shape of the probability distribution of tokens by introducing a scaling factor  $\tau$  (**temperature**):

$$\text{Final normalized token probabilities} \rightarrow y = \text{softmax}\left(\frac{\text{logits}}{\tau}\right)$$

Raw model scores  
Temperature

If  $\tau = 1.0$ , the probabilities are unchanged. When  $\tau$  is closer to 0, the probability of high-probability words is increased and probability of low-probability words is decreased (i.e., randomness is reduced and the model is more likely to pick a high-probability word). Setting  $\tau$  to a value greater than 1 has the opposite effect. **Top-k** and **top-p** sampling are alternatives to and can be used in conjunction with temperature sampling.



## DIY Step (5): Do Inference

- **Tweak the sampling parameters** to strike a better balance between **diversity and quality**
  - The sampling parameters can influence the generation result **A LOT**
    - But, oftentimes people did not describe how they choose the sampling parameters for their proposed models and the evaluated baseline models, leading to unfair performance comparison
  - **A good practice in our Compose & Embellish (ICASSP'23) paper**

Nucleus sampling [21] with tempered softmax is employed during inference. We discover that the temperature  $\tau$  and probability mass truncation point  $p$  greatly affect the intra-sequence repetitiveness and diversity of generated lead sheets. Thus, we follow [22] and search within  $\tau = \{1.2, 1.3, 1.4\}$  and  $p = \{.95, .97, .98, .99\}$  to find a combination with which our lead sheet model generates outputs with the closest mean perplexity (measured by the model itself) to that of validation real data. Finally,  $\tau = 1.2$  and  $p = .97$  are cho-

# DIY Step (5): Do Inference

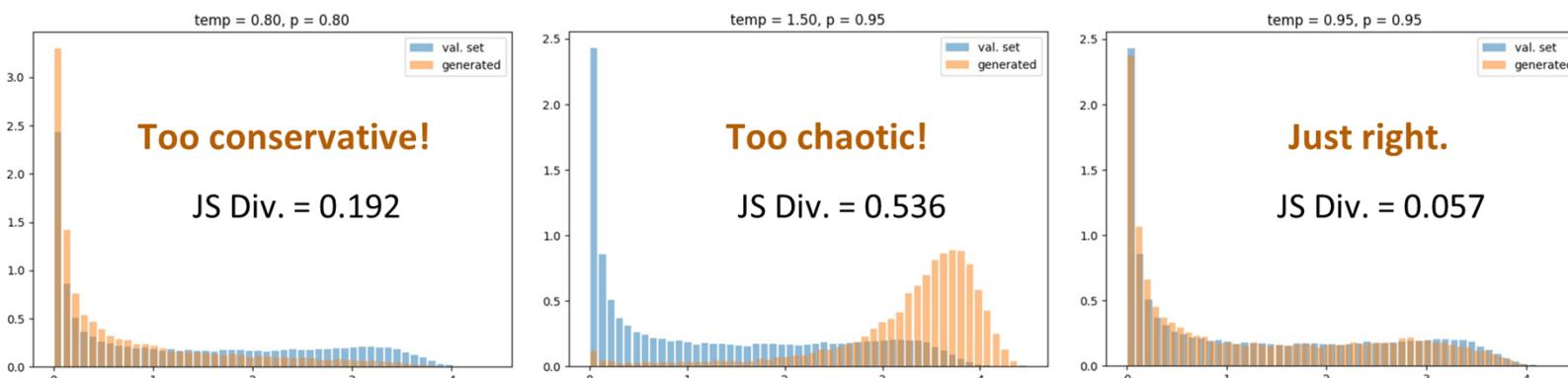
- **Nucleus Sampling w/ Temperature** [Holtzman et al., 2019]
  - Temperature                          -- Reshape distribution
  - Top- $p$                                 -- Truncate long tail

→ collectively control how **aggressive** the sampling is

$$\text{PPL}(X) = \exp \left\{ -\frac{1}{t} \sum_i^t \log p_\theta(x_i | x_{<i}) \right\}$$

<https://huggingface.co/docs/transformers/perplexity>

- Goal: match the **entropy histogram** during **sampling** to that computed on **val. set** by tuning temp. and top- $p$



(Slide made by  
Shih-Lun Wu)

Ref: Wu et al, “Compose & Embellish: Well-structured piano performance generation via a two-stage approach”, ICASSP 2023

# DIY Step (6): Listen to the Generated Music!

<https://www.fluidsynth.org/>

<https://github.com/bzamecnik/midi2audio>

- Use a synthesizer if you prefer to listen to audio

```
from midi2audio import FluidSynth
```

Play MIDI:

```
FluidSynth().play_midi('input.mid')
```

Synthesize MIDI to audio:

```
# using the default sound font in 44100 Hz sample rate
fs = FluidSynth()
fs.midi_to_audio('input.mid', 'output.wav')
```

## FluidSynth

### A SoundFont Synthesizer

**FluidSynth** is a real-time software synthesizer based on the SoundFont 2 specifications and has reached widespread distribution. FluidSynth itself does not have a graphical user interface, but due to its powerful API several applications utilize it and it has even found its way onto embedded systems and is used in some mobile apps.

## **DIY Step (7): Do Evaluation**

- Objective evaluation
- Subjective evaluation

# Outline

- Sheet music & symbolic representations for music
- Token representation of MIDI music
- MIDI generation
- **Evaluation 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. (Coherence, Correctness, Structureness, Richness, Overall. SDs across individual data points follow  $\pm$ .)

	Ch	Cr	S	R	O
CPT [4]	2.38 $\pm$ 0.9	2.49 $\pm$ 0.9	2.33 $\pm$ 0.9	2.64 $\pm$ 0.9	2.33 $\pm$ 0.9
C&E (ours)	3.53 $\pm$ 0.9	3.11 $\pm$ 1.0	3.36 $\pm$ 1.2	3.29 $\pm$ 1.0	3.18 $\pm$ 0.9
Real data	4.42 $\pm$ 0.7	4.13 $\pm$ 0.8	4.44 $\pm$ 0.8	4.24 $\pm$ 0.8	4.40 $\pm$ 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://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  $\vec{h}$ , 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  $\vec{h}$ :

$$\mathcal{H}(\vec{h}) = - \sum_{i=0}^{11} h_i \log_2(h_i). \quad (2)$$

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  $\vec{h}$ ; on the contrary, if the tonality is unstable, the usage of pitch classes is likely scattered, giving rise to an  $\vec{h}$  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  $\vec{g}$ , a 64-dimensional binary vector in our setting.<sup>6</sup> We define the similarity between a pair of grooving patterns  $\vec{g}^a, \vec{g}^b$  as:

$$\mathcal{GS}(\vec{g}^a, \vec{g}^b) = 1 - \frac{1}{Q} \sum_{i=0}^{Q-1} \text{XOR}(g_i^a, g_i^b), \quad (3)$$

where  $Q$  is the dimensionality of  $\vec{g}^a, \vec{g}^b$ , and  $\text{XOR}(\cdot, \cdot)$  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.

# 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.

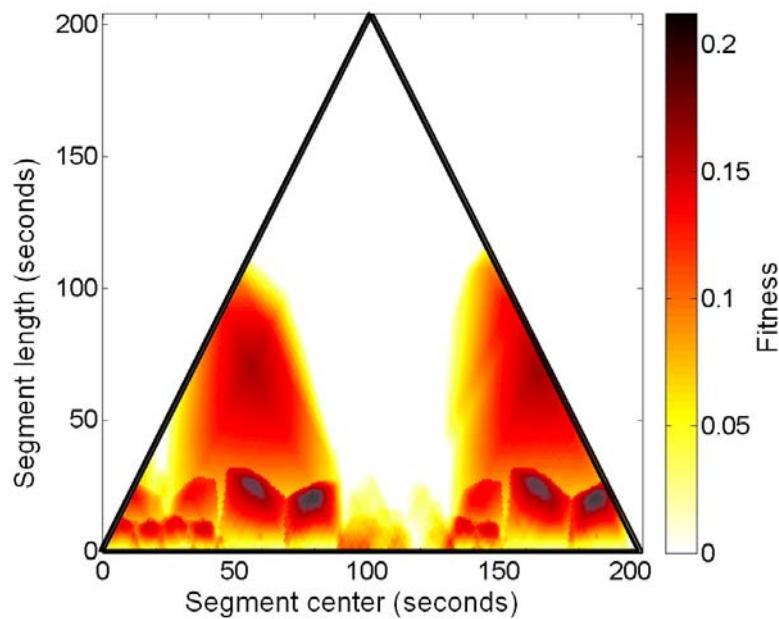
It is common for Jazz compositions to make use of 8- or 12-bar-long templates of chord progressions (known as the 8-, or *12-bar blues*), which themselves can be broken down into similar substructures [25, 35], as the foundation of a section, and more or less “copy-paste” them to form the complete song 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:

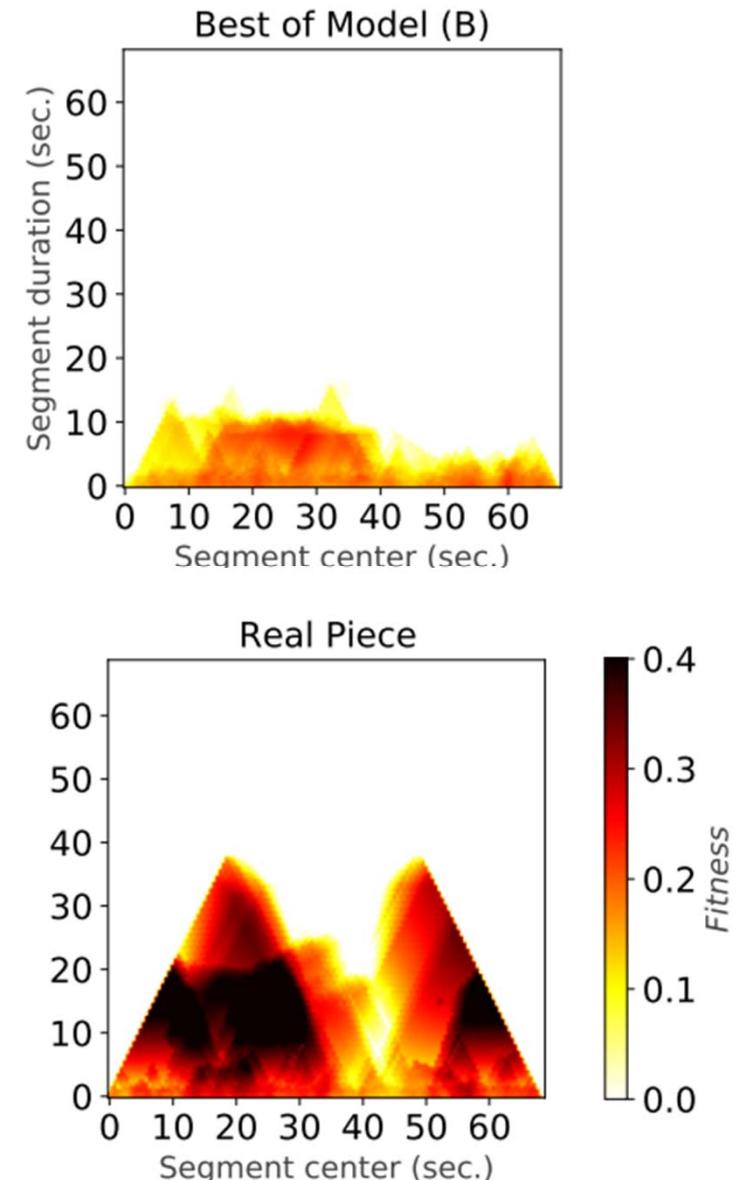
$$\mathcal{SI}_l^u(S) = \max_{\substack{l \leq i \leq u \\ 1 \leq j \leq N}} S, \quad (4)$$

where  $l, u$ <sup>8</sup> 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

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

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

- For general MIDI → use **MusPy**

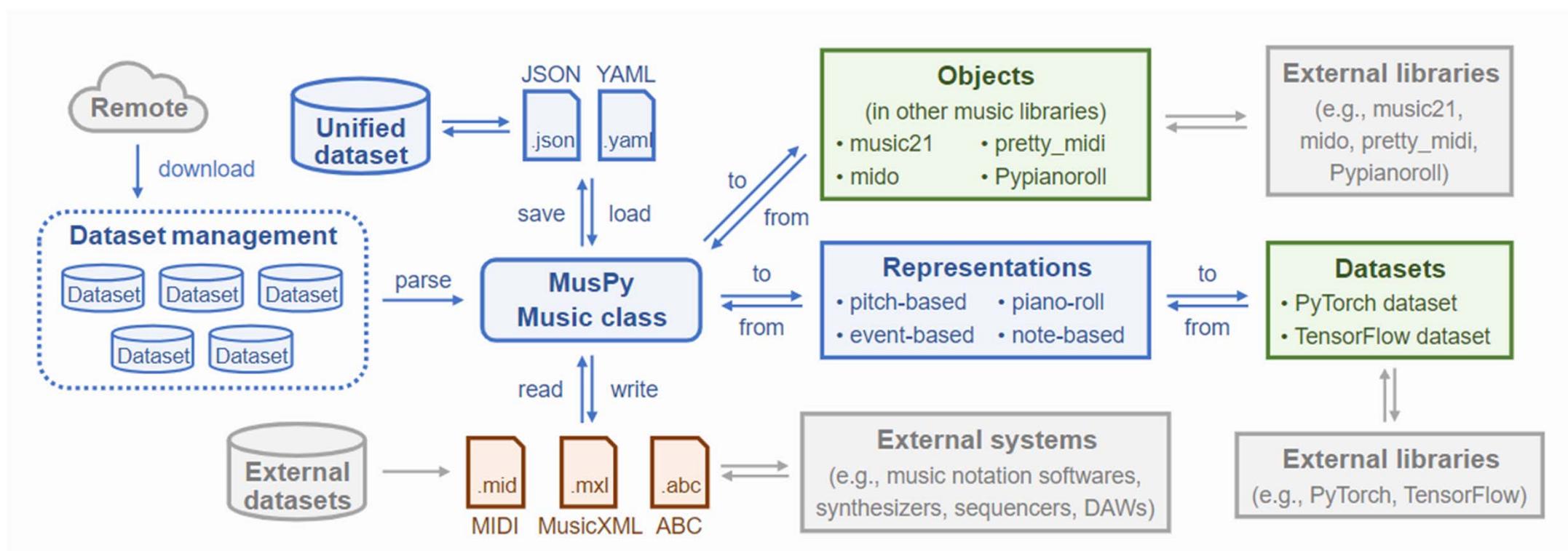
- drum\_in\_pattern\_rate
- drum\_pattern\_consistency
- empty\_beat\_rate
- empty\_measure\_rate
- groove\_consistency
- n\_pitch\_classes\_used
- n\_pitches\_used
- pitch\_class\_entropy
- pitch\_entropy
- pitch\_in\_scale\_rate
- pitch\_range
- polyphony
- polyphony\_rate
- scale\_consistency

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

# Library: MusPy

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

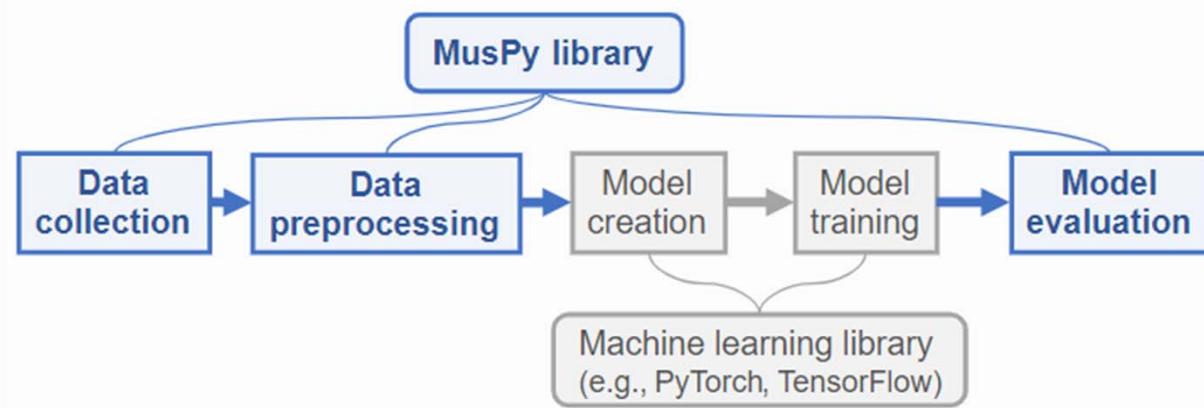


# Library: MusPy

<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



# Dataset: Piano MIDI

Dataset	Size			Artist		Modality			CER
	Performances	Hours	Compositions	Composers	Performers	Perf.	MIDI	Score	
SUPRA [18]	478	52	408	111	153	✓	✗	✗	✗
SMD [19]	50	4.7	50	11	unknown	✓	✗	✗	✗
MazurkaBL [20]	2000	110	44	1	135	✗	100%	✓	
Maestro v3.0 [21]	1276	172	864	60	205*	✓	✗	✗	
CrestMusePEDB [22]	443	unknown	35	14	12	✓	✓	✗	
GP <sup>†</sup> Curated [23]	7236	875	7236	1787	unknown	✓	✗	✗	
ASAP [24]	1068	92	222	15	unknown	✓	100%	✗	
ATEPP	11742	1007	1580	25	49	✓	43%	✓	

**Table 1.** Overview of major symbolic piano datasets. CER: composition entity resolution. \*Number obtained from crawling the Piano-e-Competition website for performer names and aligning with Maestro data. <sup>†</sup>GP stands for GiantMIDI-Piano.

# Dataset: Jazz Piano MIDI

<https://almostimplemented.github.io/PiJAMA/>

<https://zenodo.org/records/8354955>

## PiJAMA: Piano Jazz With Automatic MIDI Annotations

This notebook provides supplementary material and demonstrations of results associated with our dataset publication

The MIDI is available for download here: <https://zenodo.org/record/8354955>

### Table of Contents:

- [Transcription Videos](#)
- [Generative Model Samples](#)
- [Dataset Walkthrough](#)
  - [Metadata](#)
  - [Pitch Histograms](#)
  - [Notes per Second](#)
  - [Sliding Pitch Class Entropy](#)

# Dataset: MelodyHub

<https://huggingface.co/datasets/sander-wood/melodyhub>

**Table 1.** The MelodyHub collection statistics include the number of instances for each task along with the corresponding data sources. The JSB Chorales dataset is augmented to 15 keys due to its small size and original data in the C key.

Data Sources	Cataloging	Generation	Harmonization	Melodization	Segmentation	Transcription	Variation
<i>ABC Notation</i> [31]	184,660	184,738	31,732	31,690	—	174,779	—
<i>FolkWiki</i> [32]	6,610	6,767	1,207	1,205	—	6,218	—
<i>JSB Chorales</i> [33]	4,980	4,980	4,950	4,950	19,125	4,980	—
<i>KernScores</i> [34]	1,731	1,776	—	—	1,275	1,754	—
<i>Meertens Tune Collections</i> [35]	16,662	16,662	—	—	16,660	16,297	—
<i>Nottingham</i> [36]	1,031	1,031	1,014	1,014	—	1,021	—
<i>OpenScore Lieder</i> [37]	1,326	1,326	—	—	—	1,255	—
<i>The Session</i> [38]	44,620	44,620	3,081	3,078	—	42,838	174,104
Total	261,620	261,900	41,984	41,937	37,060	249,142	174,104