2025 edition

Deep Learning for Music Analysis and Generation

Symbolic Music Generation

(x → symbolic music)

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

Reference: Four ISMIR Tutorials

- "**Machine-Learning for Symbolic Music Generation"** by Pierre Roy (Spotify) and Jean-Pierre Briot (Sony CSL) at **ISMIR 2017**
  https://ismir2017.ismir.net/tutorials/index.html#T4
- "**Generating Music with GANs—An Overview and Case Studies**" by Hao-Wen Dong and Yi-Hsuan Yang (Academia Sinica) at **ISMIR 2019**
  https://salu133445.github.io/ismir2019tutorial/
- "**Designing Generative Models for Interactive Co-creation**" by Anna Huang (Google), Jon Gillick (UC Berkeley), Chris Donahue (Stanford) at **ISMIR 2021**
  https://github.com/chrisdonahue/music-cocreation-tutorial
- "**Transformer-based Symbolic Music Generation**" by Berker Banar (QMUL), Pedro Sarmento (QMUL), Sara Adkins (infinitealbum) at **ISMIR 2023**
  https://ismir2023.ismir.net/tutorials/

Reference: ACM Multimedia 2021 Tutorial

https://www.microsoft.com/en-us/research/uploads/prod/2021/10/Tutorial-on-AI-Music-Composition-@ACM-MM-2021.pdf

A Tutorial on AI Music Composition

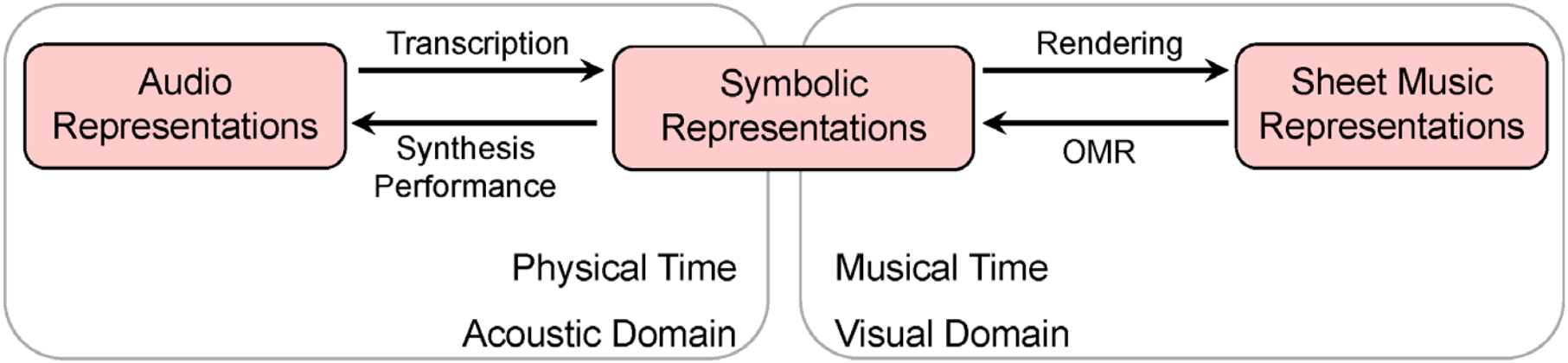
Xu Tan & Xiaobing Li
Microsoft Research Asia & Central Conservatory of Music, China

Outline

- **Symbolic representations for music: Not just MIDI**
- Image-based approach for symbolic music generation
- Text-based approach for symbolic music generation
- Advanced models for MIDI generation

Data Representation of Music

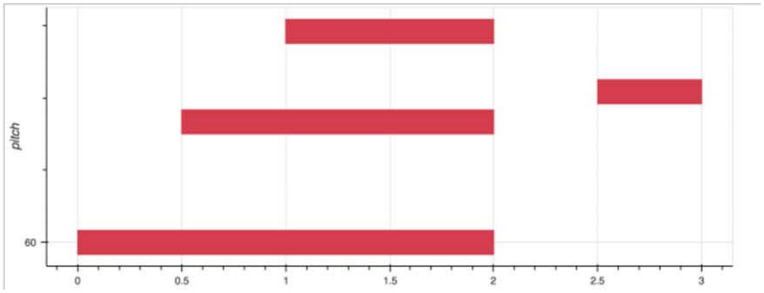
(Figure source: https://www.mdpi.com/2624-6120/2/2/18)



- **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"

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/

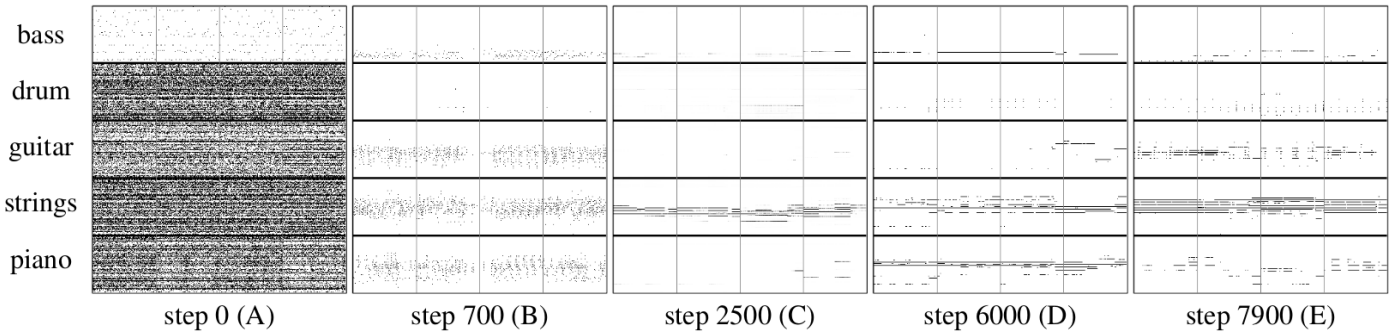



```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```

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
  - *Polyffusion* (2023) [diffusion-based]: https://polyffusion.github.io/

MuseGAN



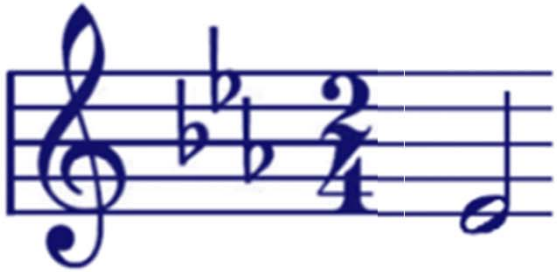
DiffRoll

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>

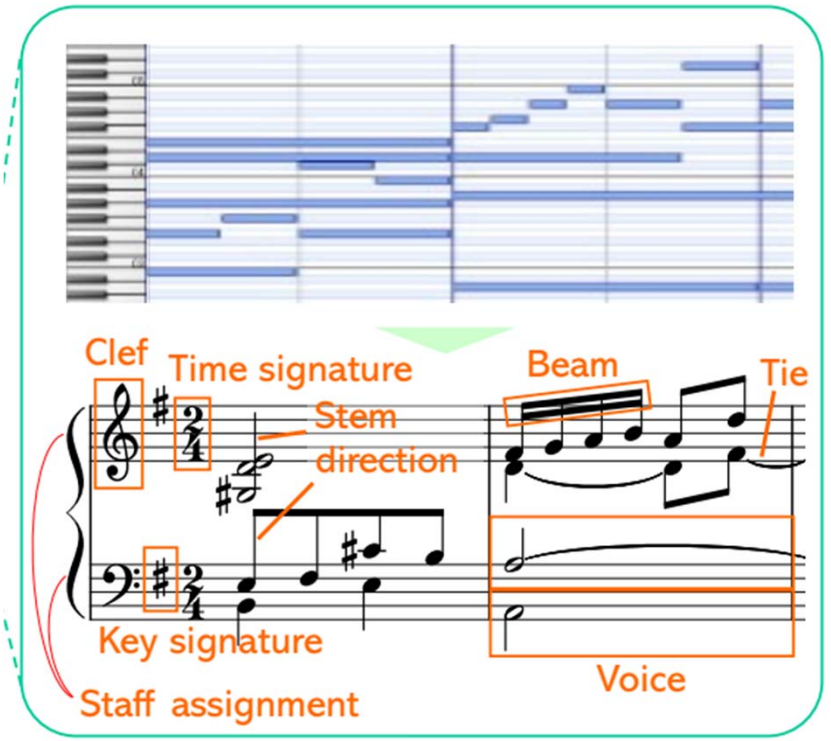
```
<note>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>2</duration>
  <type>half</type>
</note>
```



- Can be **tokenized**
- Musical (not physical) onset times are specified
- 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**

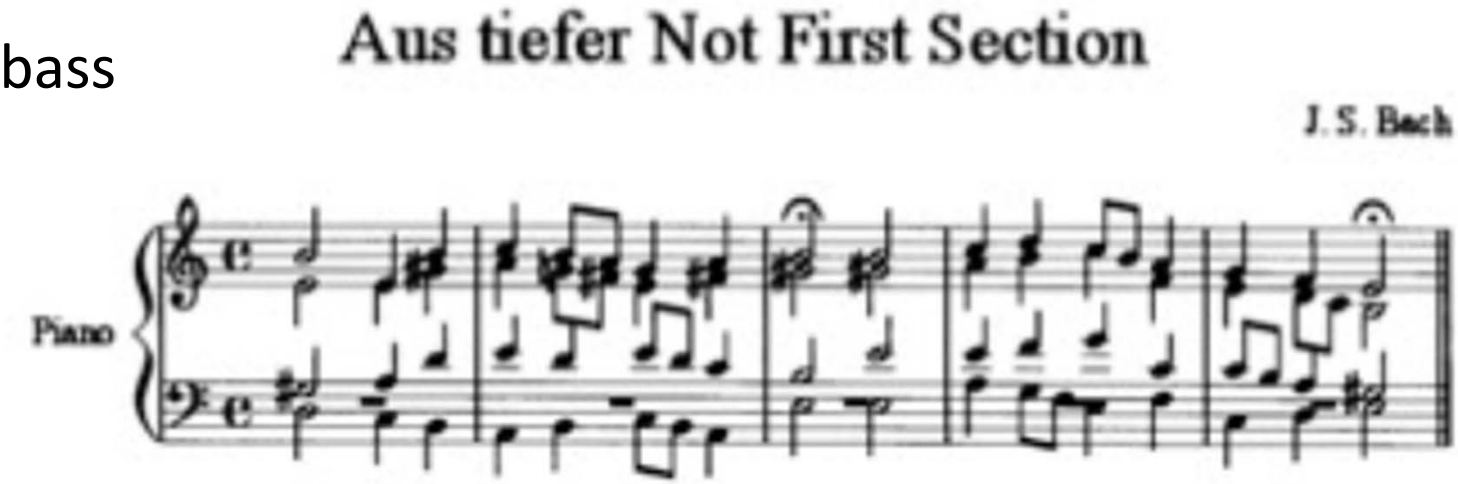


Ref: Suzuki, “Score Transformer: Generating musical score from note-level representation,” MMAsia 2021

Voices

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

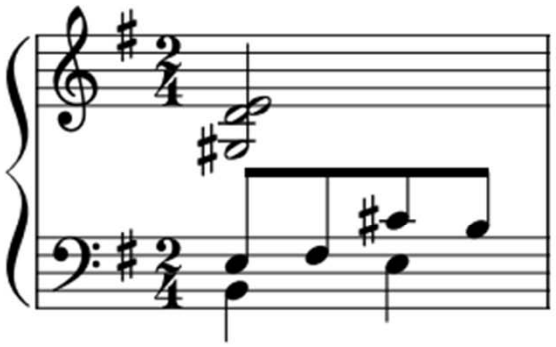
Voice type
Female
Soprano
Mezzo-soprano
Contralto
Male
Countertenor
Tenor
Baritone
Bass



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)



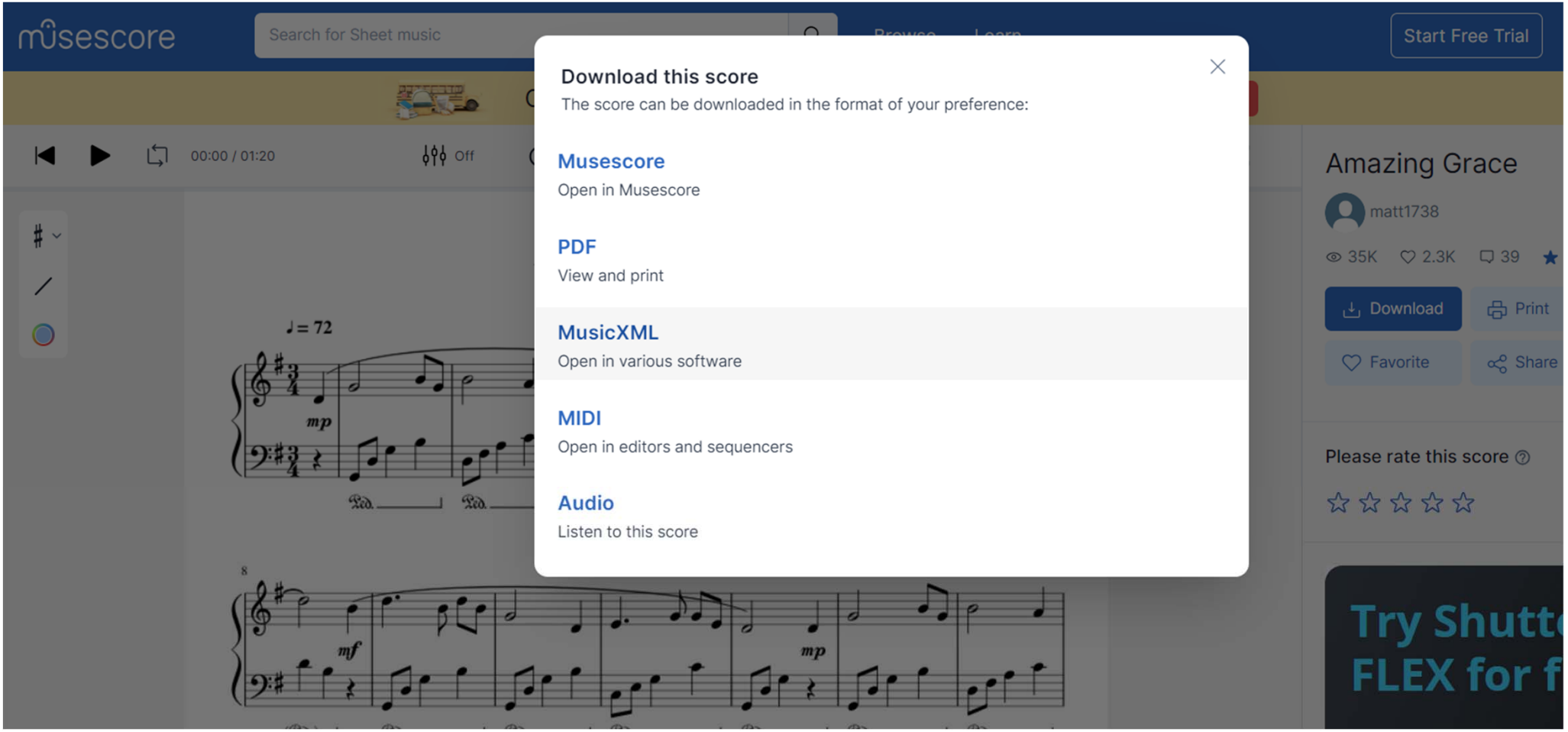
(a) Score

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

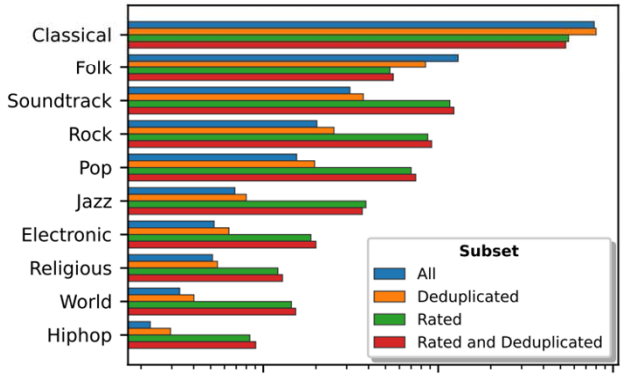


Dataset: PDMX

https://pnlong.github.io/PDMX.demo/

Dataset	Format	Hours	Size	Multitrack	Dataset License Composition
Lakh MIDI (LMD) [18]	MIDI	>9,000	174,533	✓	CC-BY 4.0†
SymphonyNet [13]	MIDI	>3,200	46,359	✓	
MAESTRO [19]	MIDI	201.21	1,282		CC BY-NC-SA 4.0
Wikifonia Lead Sheet Dataset [14]	MusicXML	198.40	6,405		
POP909 [15]	MIDI	60.00	909		
EMOPIA [20]	MIDI	11.0	1,078		CC BY-NC-SA 4.0
MMD [16]	MIDI	-	1,524,557		
MetaMidi [17]	MIDI	-	612,088	✓	
PDMX	MusicXML	6,250	254,077	✓	CC-0 / Public Domain

Subset	Hours / Size	PCE	SC (%)	GC (%)
A	6,250 / 254K	2.69 ± 0.00	97.12 ± 0.01	93.75 ± 0.01
D	3,756 / 102K	2.77 ± 0.00	95.15 ± 0.02	93.79 ± 0.01
R	1,001 / 14K	2.91 ± 0.00	91.59 ± 0.07	92.59 ± 0.05
R∩D	941 / 13K	2.90 ± 0.00	91.70 ± 0.07	92.65 ± 0.05
Fine-Tuning	595 / 6K	2.95 ± 0.00	90.54 ± 0.11	92.39 ± 0.07

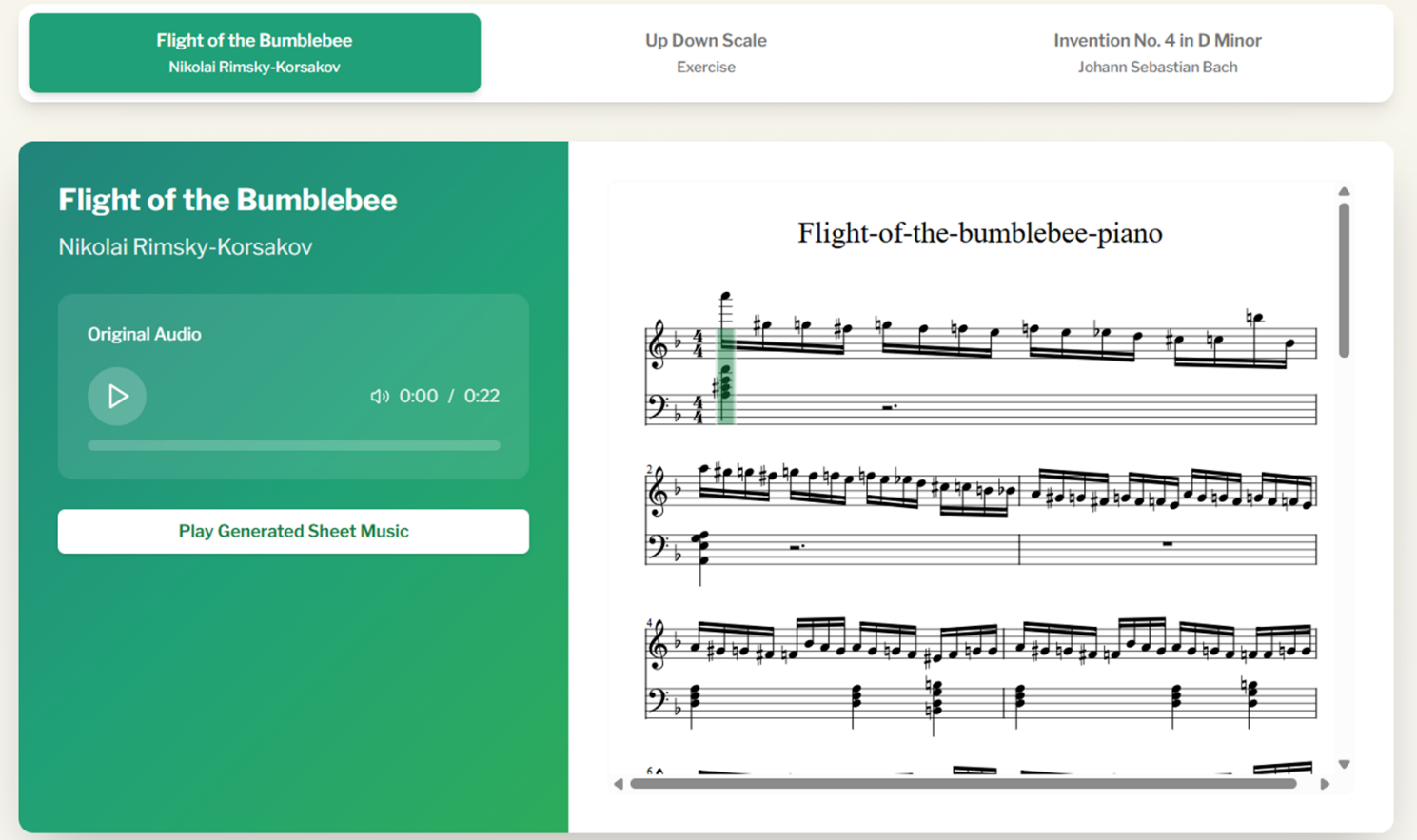


Long et al, “PDMX: A large-scale public domain MusicXML dataset for symbolic music processing,” ICASSP 2025

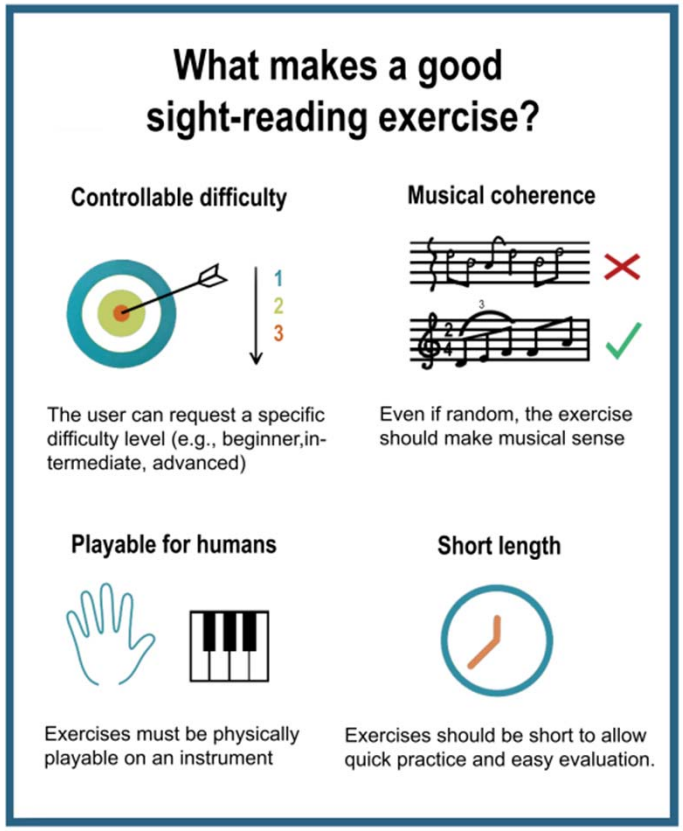
[Recap] Music Analysis Demo 1: Music Transcription

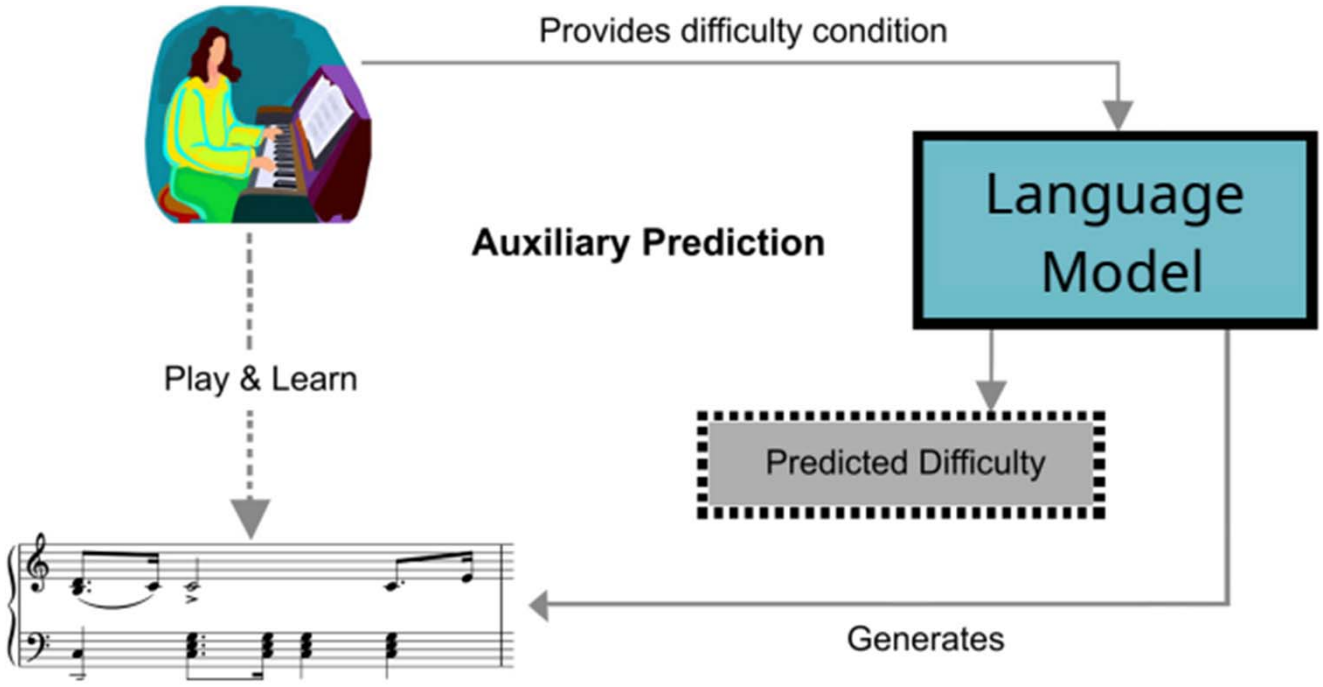
https://songscription.ai/

- songscription.ai
- AnthemScore
- Melody Scanner
- Klangio



MusicXML Generation



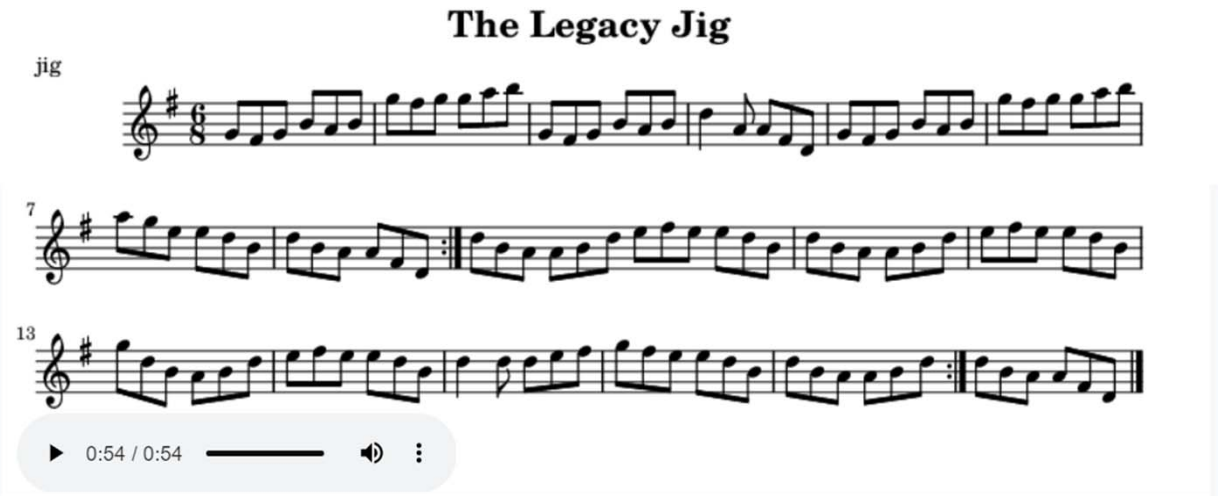


Ref: Ramoneda et al, "Difficulty-aware score generation for piano sight-reading," arXiv 2025

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
```

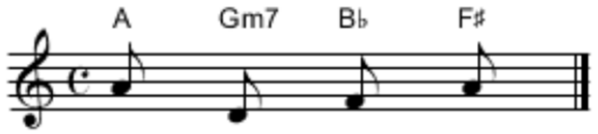
```
A>A A2>A2 | A<A A2<A2 | A>>A A2>>>A2 | A<<A A2<<<A2
```

```
(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
```

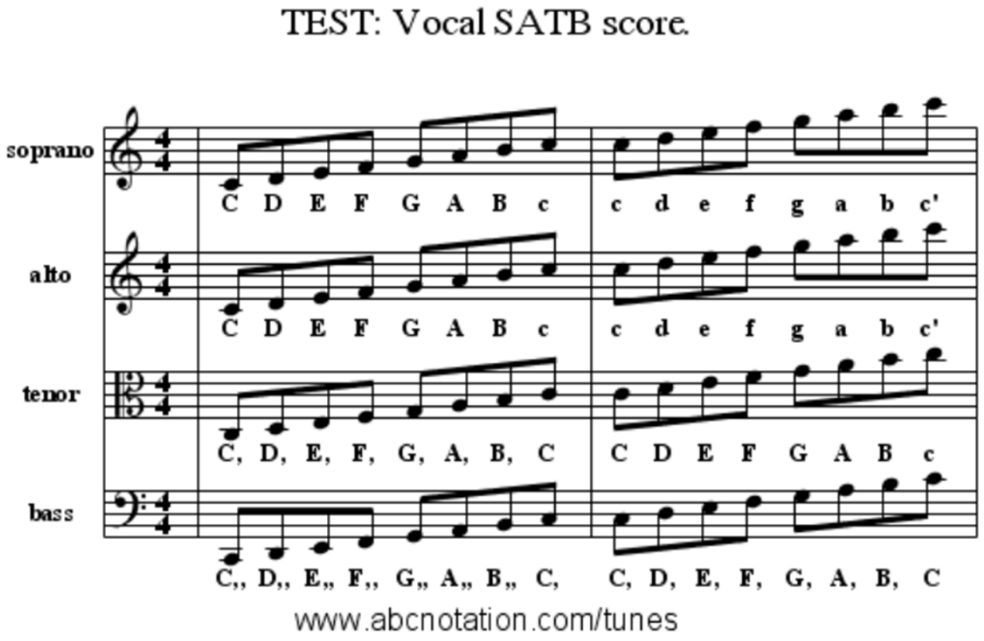



```
{g}A3 A{g}AA|{gAGAG}A3 {g}A{d}A{e}A|
```

ABC

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

- Can support multiple voices

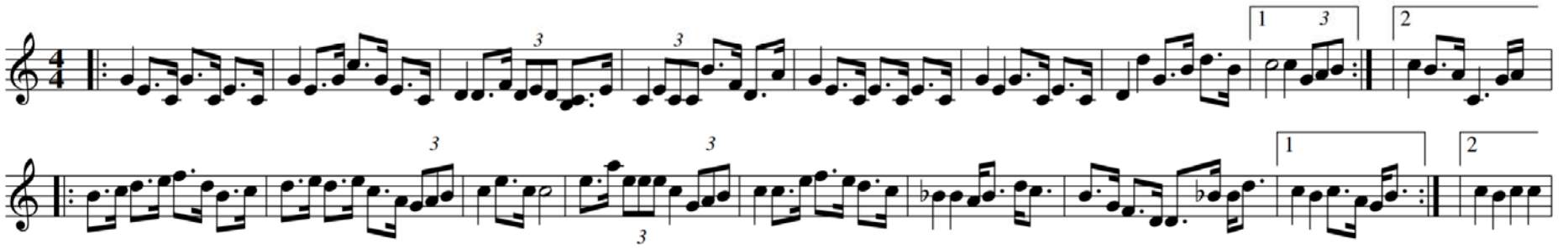



```
X:1
T: TEST: Vocal SATB score.
N: All parts written "on the staff" with commas.
N: All parts should show scale from C below to above staff.
K: C
V:S name="soprano" clef=treble   middle=B
V:A name="alto"    clef=treble   middle=B
V:T name="tenor"   clef=tenor    middle=B,
V:B name="bass"    clef=bass     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

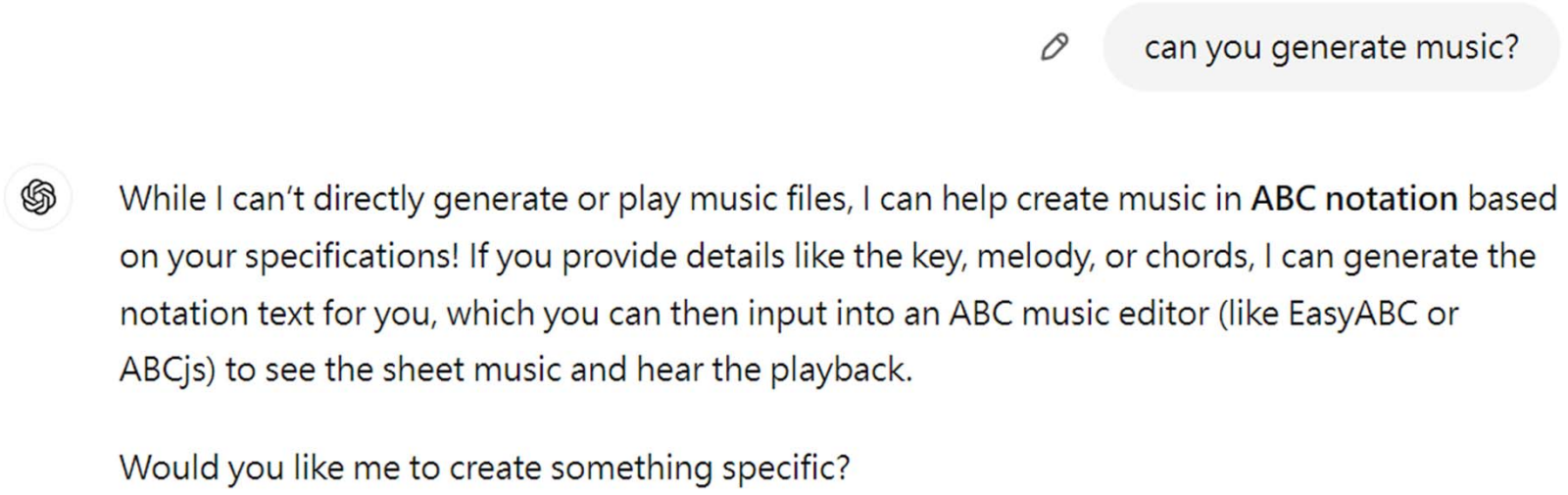
- A ***text-like*** representation of **sheet music**
- Readily tokenized
- Support **score** only, not performance
- Generative models adopting the ABC notation: *Folk RNN* [sturm15ismir-lbd]

```
M:4/4
K:Cmaj
|: G 2 E > C G > C E > C | G 2 E > G c > G E > C | D 2 D > F (3 D E D [ B, C ] > E | C 2 (3 E C C B > F D > A |
G 2 E > C E > C E > C | G 2 E 2 G > C E > C | D 2 d 2 G > B d > B |1 c 4 c 2 (3 G A B :| |2 c 2 B > A C 3 G /2 A /2
|: B > c d > e f > d B > c | d > e d > e c > A (3 G A B | c 2 e > c c 4 | e > a (3 e e e c 2 (3 G A B |
c 2 c > e f > e d > c | _B 2 B 2 A < B d < c | B > G F > D D > _B B < d |1 c 2 B 2 c > A G < B :| |2 c 2 B 2 c 2 c 2 |
```

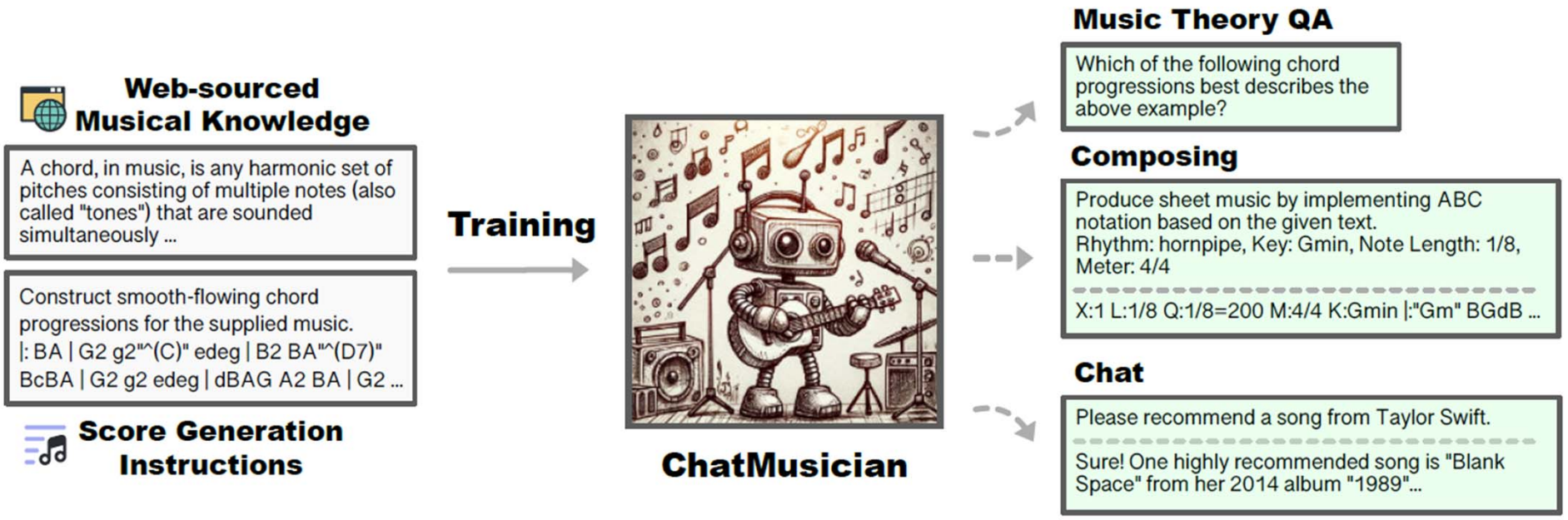


ABC

- A ***text-like*** representation of **sheet music**
- Readily tokenized
- Support **score** only, not performance
- Generative models adopting the ABC notation
- **Can be used by large language models (LLMs)**



LLM ♡ ABC



ChatMusician: Understanding and generating music intrinsically with LLM, ACL 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

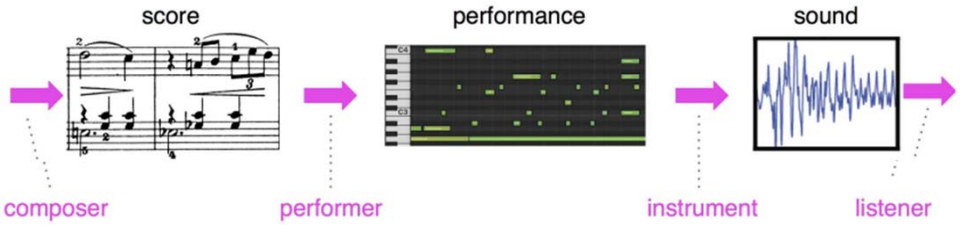
https://github.com/jwdj/EasyABC/

EasyABC / midi2abc.py

Features

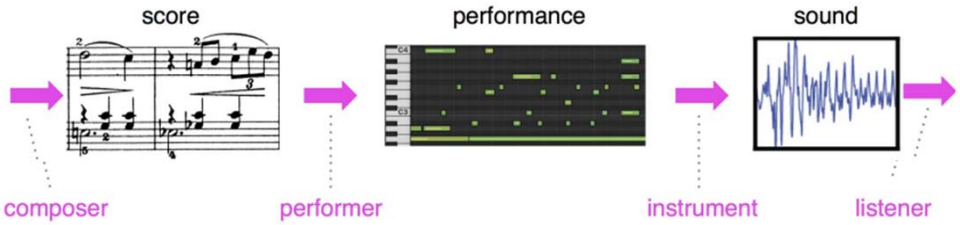
- Good ABC standard coverage thanks to internal use of abcm2ps and abc2midi
- Syntax highlighting
- Zoom support
- Import MusicXML, MIDI and Noteworthy Composer files.
- Export to MIDI, SVG, PDF (single tune or whole tune book).
- Select notes by clicking on them and add music symbols by using drop-down 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 **musical time** only, not physical time
  - Richer information about the score itself
- **Piano roll / MIDI**
  - Support both **score** and **performance**
  - Use either **musical time** (after quantization) or **physical time**
  - Rich performance information
  - Datasets more easily accessible and thereby **more widely used**

Summary (Cont')



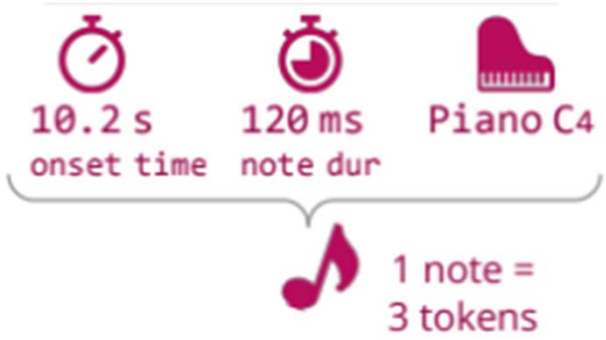
- **ABC**
  - Good for simple melodies, chords, and basic arrangements
  - Limited for more complicated scores (e.g., piano, symphony)
  - Good for LLMs (but can *only* do composition; not performance)
- **Piano roll / MIDI / MusicXML**
  - More flexible

MIDI + LLM?

- MIDI tokens are “artificial,” no standard way of tokenization
- How if we just fine-tune an LLM to expand the vocabulary to include a certain set of MIDI tokens?

Token Design in Multi-track MIDI Transformer

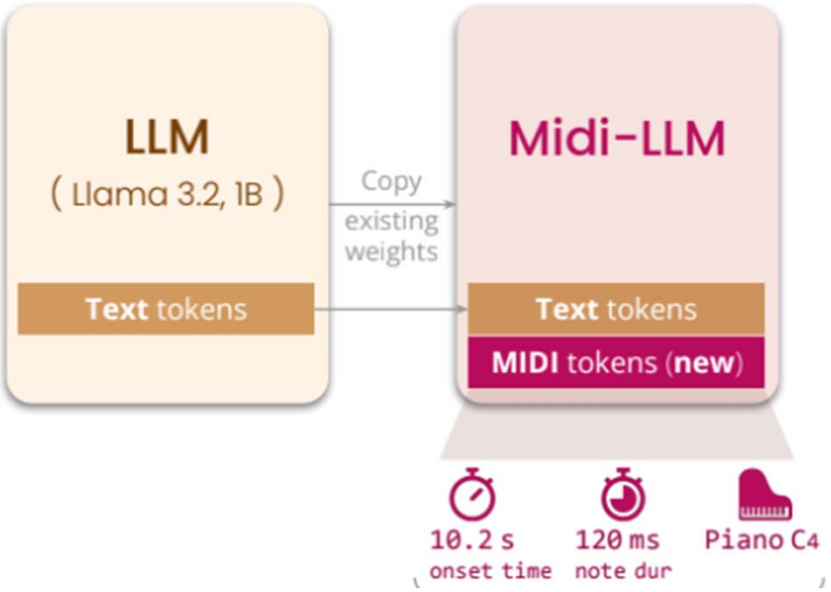
- To further account for instruments
- The tokenization used in MIDI-LLM (Wu et al 2025)
  - **Arrival (onset) time**: note's start time (0^{th} to 100^{th} second with 10 ms quantization)
  - **Note duration**: How long the note is held (0 to 10 seconds with 10 ms quantization)
  - **Instrument-pitch**: A joint token for the instrument and its pitch (129 MIDI inst. × 128 pitches)
- This amounts to **27.5K** possible tokens



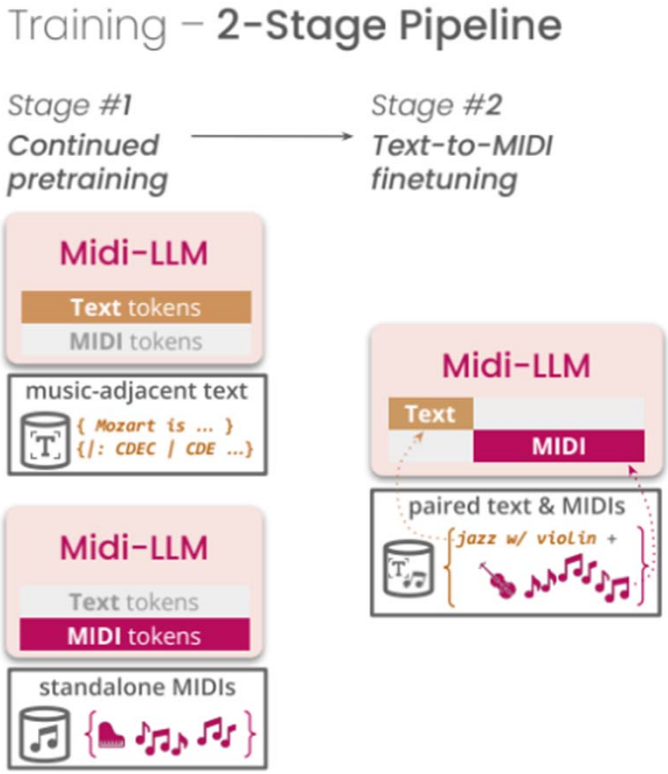
Ref: Wu et al, "MIDI-LLM: Adapting large language models for text-to-MIDI music generation", AI4Music Workshop, 2025

MIDI-LLM: An LLM for Generating Multitrack MIDI Music

https://midi-llm-demo.vercel.app/



$$\mathbf{E}_{\text{MIDI-LLM}} := \begin{bmatrix} \mathbf{E}_{\text{LLM}}^{\top} & \mathbf{E}_{\text{AMT}}^{\top} \end{bmatrix}^{\top} \in \mathbb{R}^{(|\mathcal{V}_{\text{LLM}}|+|\mathcal{V}_{\text{AMT}}|)\times D}$$



Wu et al, “MIDI-LLM: Adapting large language models for text-to-MIDI music generation”, AI4Music, 2025

Outline

- Symbolic representations for music: Not just MIDI
- **Text-based approach for symbolic music generation**
  - **RNN**
  - Transformer
- Advanced models for MIDI generation
- Image-based approach for symbolic music generation

Two Main Approaches

(2) Consider music as **text**

- Use “tokens” to represents “events” in music
- Becomes a “natural language generation” (NLG) problem

RNN-based Symbolic Music Generation

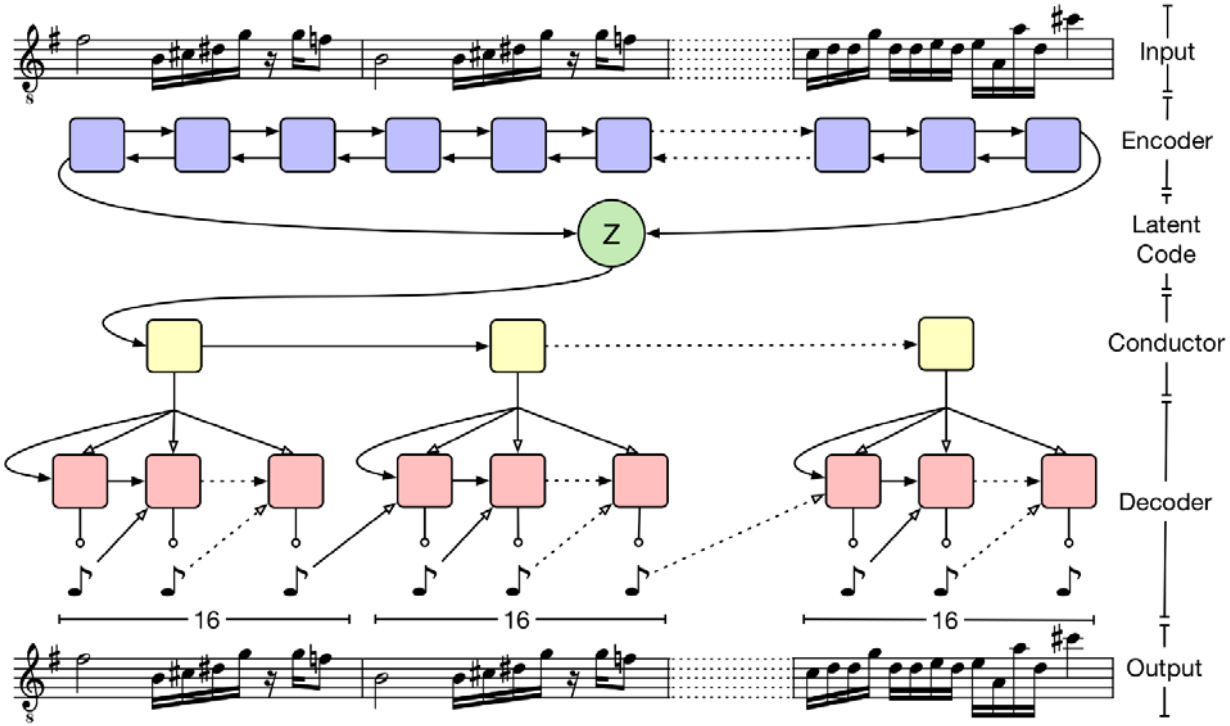
- Used to be the state-of-the-art
- Good for not-very-long sequences (e.g., 16 bars)

- Examples
  - **MelodyRNN** (2016): https://magenta.tensorflow.org/2016/06/10/recurrent-neural-network-generation-tutorial
  - **C-RNN-GAN** (2016)
  - **Song From PI** (2017)
  - **DeepBach** (ICML 2017)

RNN Example: Music VAE [roberts18icml]

https://magenta.tensorflow.org/music-vae

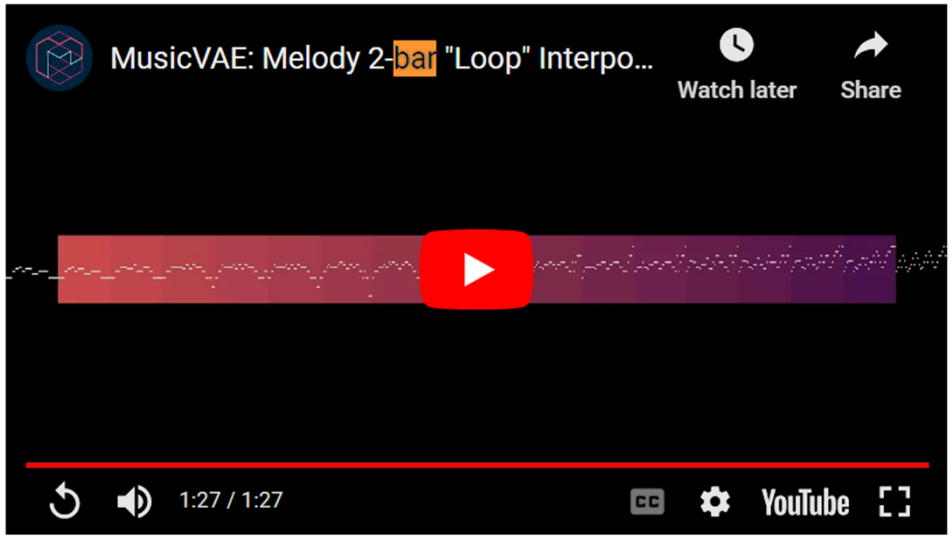
- For **melody** generation
- **Bidirectional encoder**
- **Hierarchical decoder**
  - **Bar-level** conductor
  - **Note-level** decoder
- **Recurrent VAE**
  - Two-layer LSTM for the encoder, conductor, decoder
  - Spherical Gaussian prior for the latent code *z*

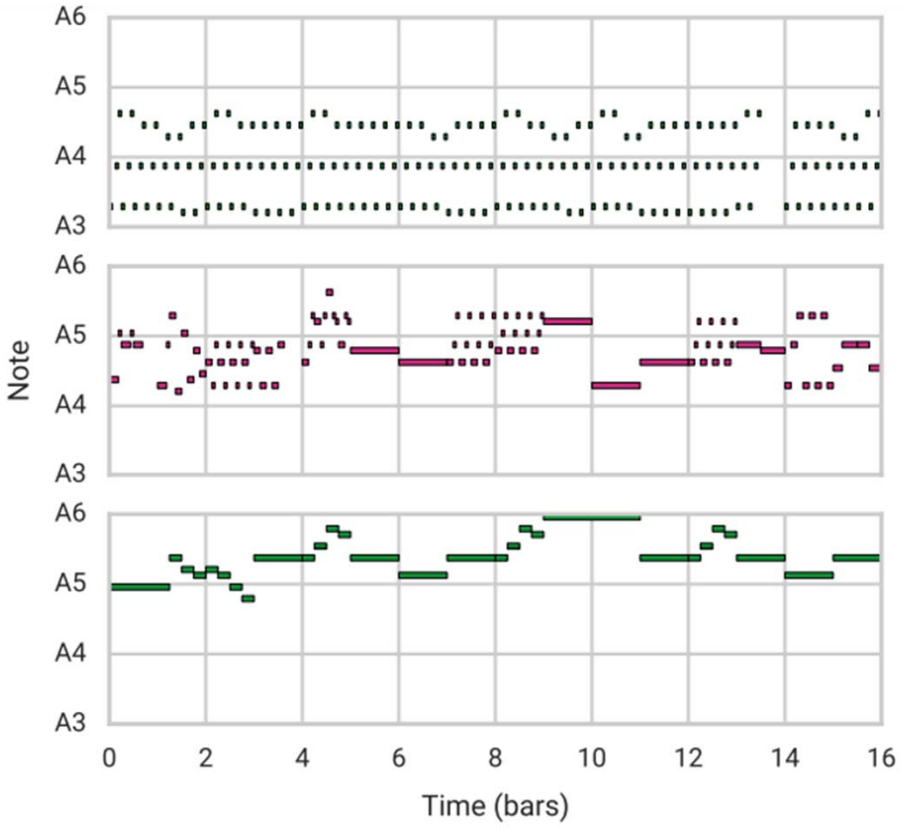


Ref: Roberts et al, "A hierarchical latent vector model for learning long-term structure in music", ICML 2018

Music VAE [roberts18icml]

- Latent space manipulation
  - Attribute vectors
  - **Spherical interpolation**



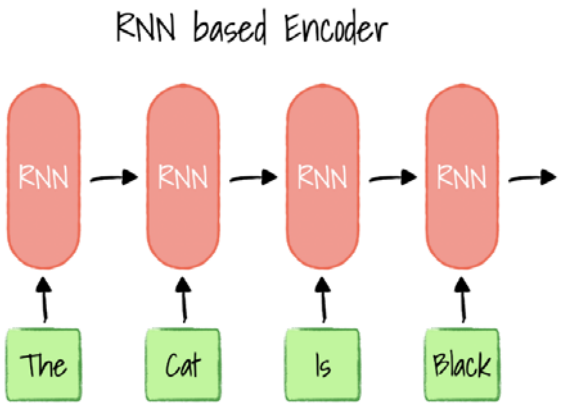


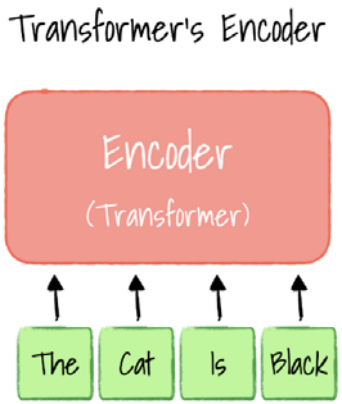
Ref: Roberts et al, "A hierarchical latent vector model for learning long-term structure in music", ICML 2018

Outline

- Symbolic representations for music: Not just MIDI
- **Text-based approach for symbolic music generation**
  - RNN
  - **Transformer**
- Advanced models for MIDI generation
- Image-based approach for symbolic music generation

RNNs vs. Transformers





- **RNNs**
  - use only a latent vector
  - $y_t = f(y_{t-1}, h_{t-1})$

current output — last output — latent representation of the "history"

- **Transformers**
  - multiple latent vectors
  - $y_t = f(y_{t-1}, [h_{t-1}, h_{t-2}, \ldots, h_{t-L}])$

latent representation of t-1 — latent representation of t-L

RNNs vs. Transformers

- **RNNs** work well when the sequence is ***not that long***
  - Melodies (e.g., Folk RNN, MusicVAE)
  - 8-bar piano performance (e.g., JazzRNN)

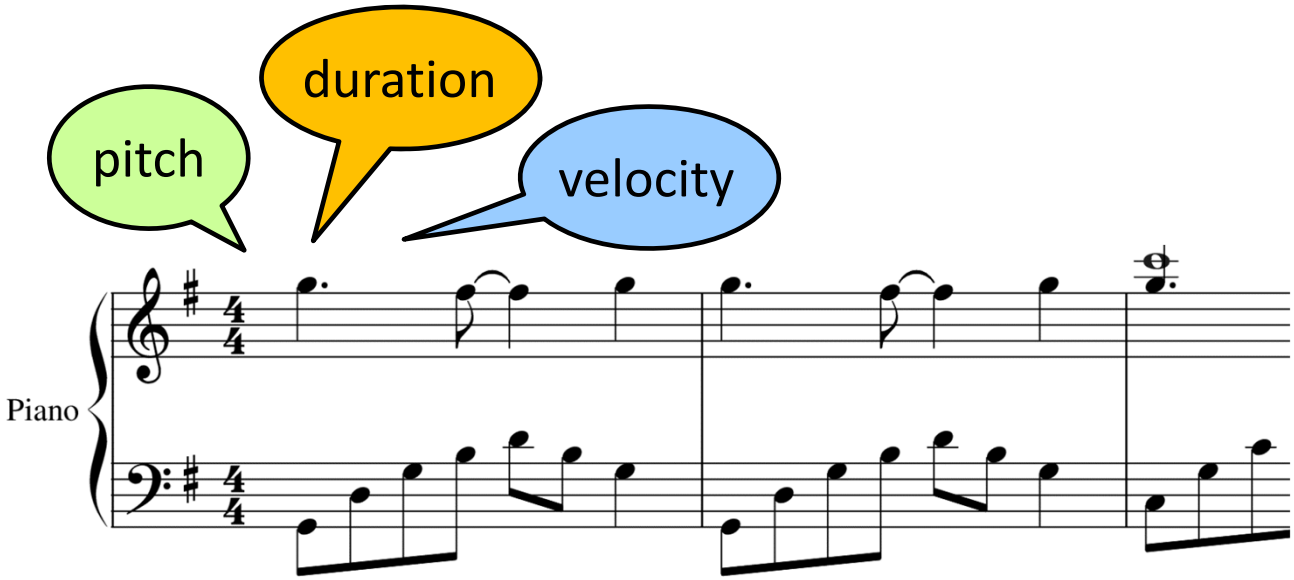
- **Transformers** work better for ***longer*** *sequences*
  - At the expense of computational power
    - An RNN-based model may contain **3** layers or so
    - A Transformer-based model may use up to >**72** layers … (e.g., MuseNet)
    - Usually, a **12**-layer Transformer decoder (with about 40M learnable parameters) works okay
  - Represent **the SOTA** approach for sequence modeling, and for building **LLMs**

Key to the Text-based Approach

- **Neural sequence model**
  - RNNs, hierarchical RNN, Transformers, state-space models (Mamba)
- **Token representation of music**
  - Token design

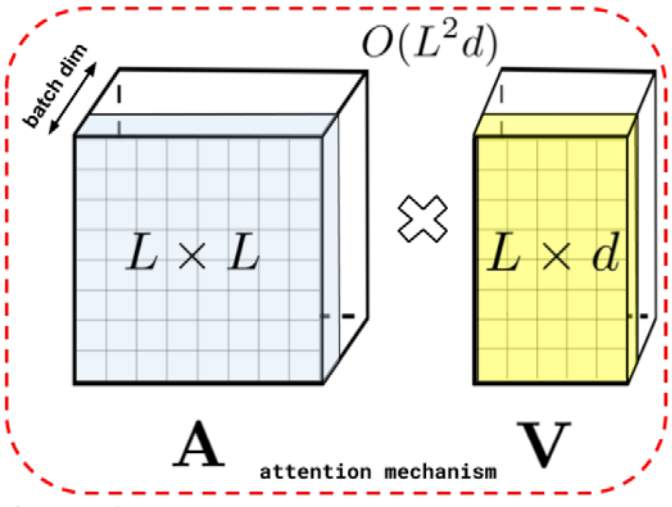
Token Design Issue 1: Long Sequence Length

- Both MIDI-like and REMI use multiple tokens to represent a musical note; isn't that inefficient?



The Problem: Transformers are Expensive

- **Transformers**
  - multiple latent vectors
  - $y_t = f(y_{t-1}, [h_{t-1}, h_{t-2}, \ldots, h_{t-L})$
  - Memory complexity: **O(L^2)**, where *L* denotes the length of the attention window (i.e., the history) because of the *L* x *L* similarity matrix
  - Need to cut a music pieces into **segments** to fit the VRAM



(Figure from the Performer paper)

The Problem: Transformers are Expensive

	Representation	Model	Attn. window
Music Transformer (Huang et al. 2019)	MIDI-like	Transformer	2,048
MuseNet (Payne 2019)	MIDI-like*	Transformer	4,096
LakhNES (Donahue et al. 2019)	MIDI-like*	Transformer-XL	512
(Choi et al. 2020)	MIDI-like	Transformer	2,048
Pop Music T. (Huang and Yang 2020)	REMI	Transformer-XL	512
Transformer VAE (Jiang et al. 2020)	MIDI-like	Transformer	128
Guitar Transformer (Chen et al. 2020)	REMI*	Transformer-XL	512
Jazz Transformer (Wu and Yang 2020)	REMI*	Transformer-XL	512
MMM (Ens and Pasquier 2020)	MIDI-like*	Transformer	2,048

- Most people use **512-token** attention window before 2020
  - But, a piano cover of a pop song can contain up to **5k** tokens
  - **Segmentation** of music pieces makes it hard to learn **long-term patterns**

Compound Word Transformer [hsiao21aaai]

https://github.com/YatingMusic/compound-word-transformer

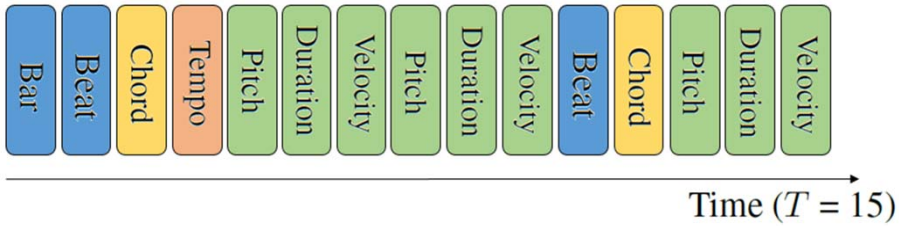
	GPU memory	Training time (single GPU)	Inference time (on GPU)	Quality MOS
Transformer-XL	4-17 GB	3-7 days	2x real time	3.0-**3.3**
CP Transformer (ours)	**4 GB**	**1 day**	**8x real time**	**3.3**

- **Compound Word**: grouping of tokens
- Shorter training time → easier to try different ideas
- Shorter inference time → good for real-time applications

Ref: Hsiao et al, "Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs," AAAI 2021

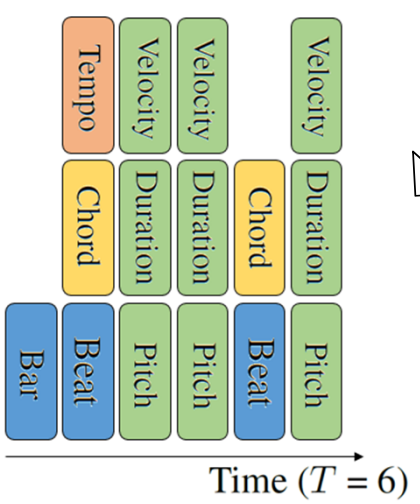
Compound Word Transformer [hsiao21aaai]

- Unlike the case in text, the vocabulary of music involves tokens of various token types (e.g., note-related, metric-related)



- Therefore, we can

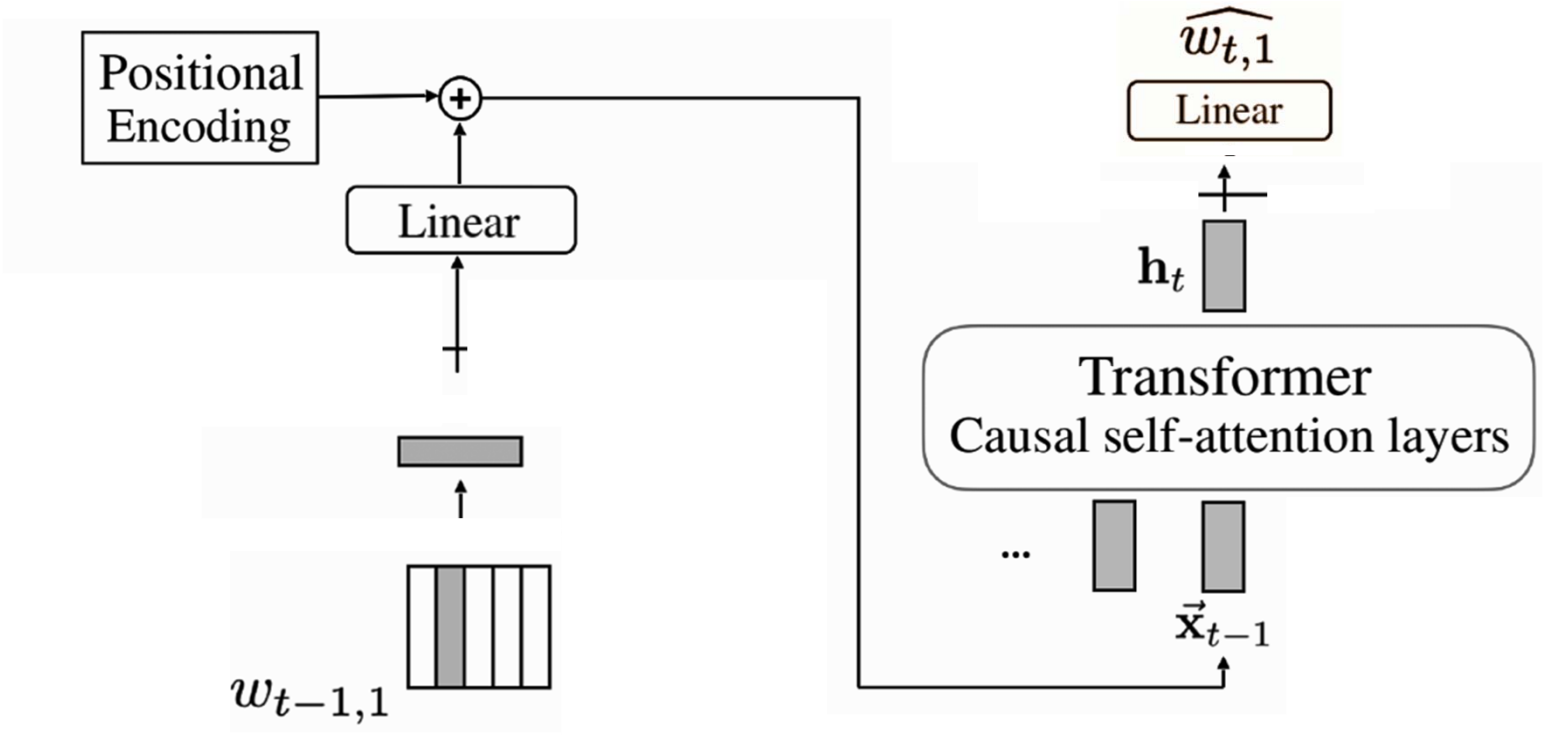
1. **group tokens** into "compound words" (e.g., pitch + duration + velocity)
2. **predict multiple tokens** of various types **at once in a single time step**
3. the embeddings of the predicted tokens are combined to be used as input to the next time step



Ref: Hsiao et al, "Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs," AAAI 2021

Original Transformer

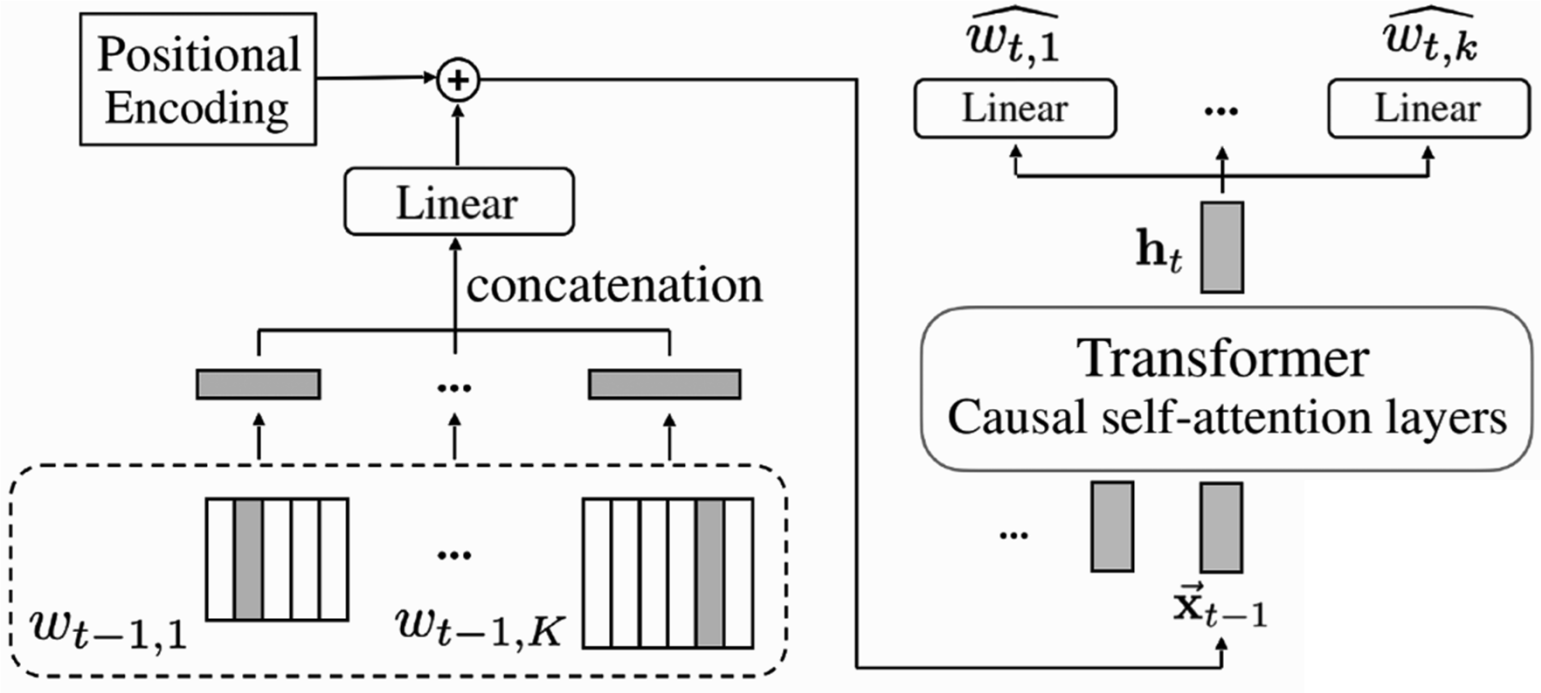
single output at each time step



single input at each time step

Compound Word Transformer

multiple output at each time step



multiple input at each time step

Transformer vs. CP Transformer

- **Transformer**
  - $y_t = f(y_{t-1}, [h_{t-1}, h_{t-2}, \ldots, h_{t-L})$
  - attention window: ***L* tokens**

- **CP Transformer**
  - multiple output: $\text{cp}_t = [y_t^{(1)}, y_t^{(2)}, \ldots, y_t^{(K)}]$
  - multiple input: $\text{cp}_{t-1} = [y_{t-1}^{(1)}, y_{t-1}^{(2)}, \ldots, y_{t-1}^{(K)}]$
  - $\text{cp}_t = f(\text{cp}_{t-1}, [h_{t-1}, h_{t-2}, \ldots, h_{t-L})$
  - `cp` is a super token
  - The latent vectors `h` are associated with each `cp`
  - Attention window: ***L* super-tokens**, or *LK* tokens

Compound Word Transformer [hsiao21aaai]

- We can now feed a **whole song** (up to 10,240 tokens) to our Transformer decoder for training on a single GPU with **11GB** VRAM
- The model converges within 1.5 days using an NVIDIA RTX 2080 Ti

Task	Representation + model@loss	Training time	GPU memory	Inference (/song) time (sec)	Inference (/song) tokens (#)
Conditional	Training data	—	—	—	—
	Training data (randomized)	—	—	—	—
	REMI + XL@0.44	3 days	4 GB	88.4	4,782
	REMI + XL@0.27	7 days	4 GB	91.5	4,890
	REMI + linear@0.50	3 days	17 GB	48.9	4,327
	CP + linear@0.27	0.6 days	10 GB	29.2	18,200
Unconditional	REMI + XL@0.50	3 days	4 GB	139.9	7,680
	CP + linear@0.25	1.3 days	4 GB	19.8	9,546

Ref: Hsiao et al, "Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs," AAAI 2021

Compound Word Transformer [hsiao21aaai]

- **Linear Transformer** [katharopoulos20icml]: **O(*L*)**
  → save GPU space
- **CP**: shorter sequence length
  → save GPU space further, and converge much faster

Representation + model@loss	Training time	GPU memory
Training data	—	—
Training data (randomized)	—	—
REMI + XL@0.44	3 days	4 GB
REMI + XL@0.27	7 days	4 GB
REMI + linear@0.50	3 days	17 GB
CP + linear@0.27	0.6 days	10 GB

Ref: Katharopoulos et al, "Transformers are RNNs: Fast autoregressive Transformers with linear attention", ICML 2020

Customed Design for Different Token Types

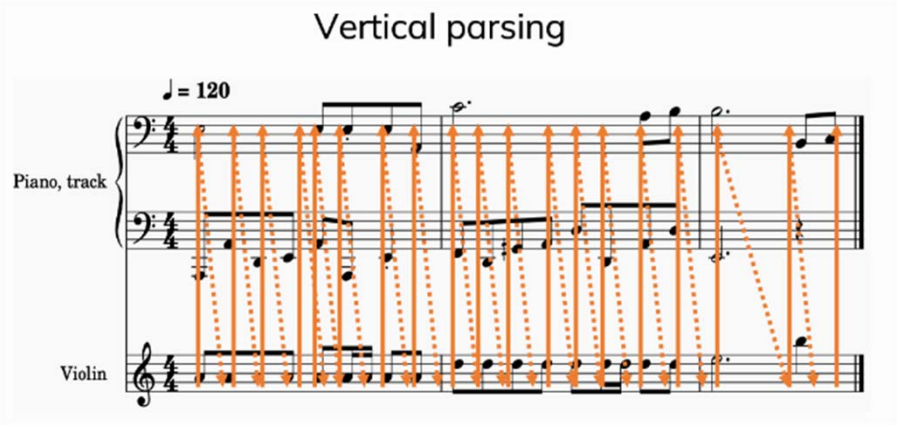
Different embedding size & sampling policy

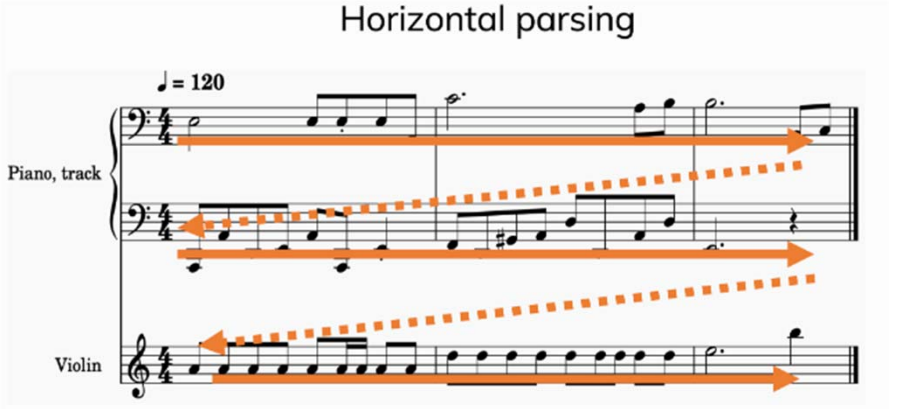
Repre.	Token type	Voc. size $\lvert\mathcal{V}_k\rvert$	Embed. size (d_k)	Sample$_k(\cdot)$ τ	ρ
CP	[track]	2 (+1)	3	1.0	0.90
	[tempo]	58 (+2)	128	1.2	0.90
	[position/bar]	17 (+1)	32	1.2	0.90
	[chord]	133 (+2)	256	1.0	0.90
	[pitch]	86 (+1)	512	1.0	0.90
	[duration]	17 (+1)	128	2.0	0.90
	[velocity]	24 (+1)	128	5.0	1.00
	[family]	4	32	1.0	0.99
	total	341 (+9)	—	—	—
REMI	total	338	512	1.2	0.90

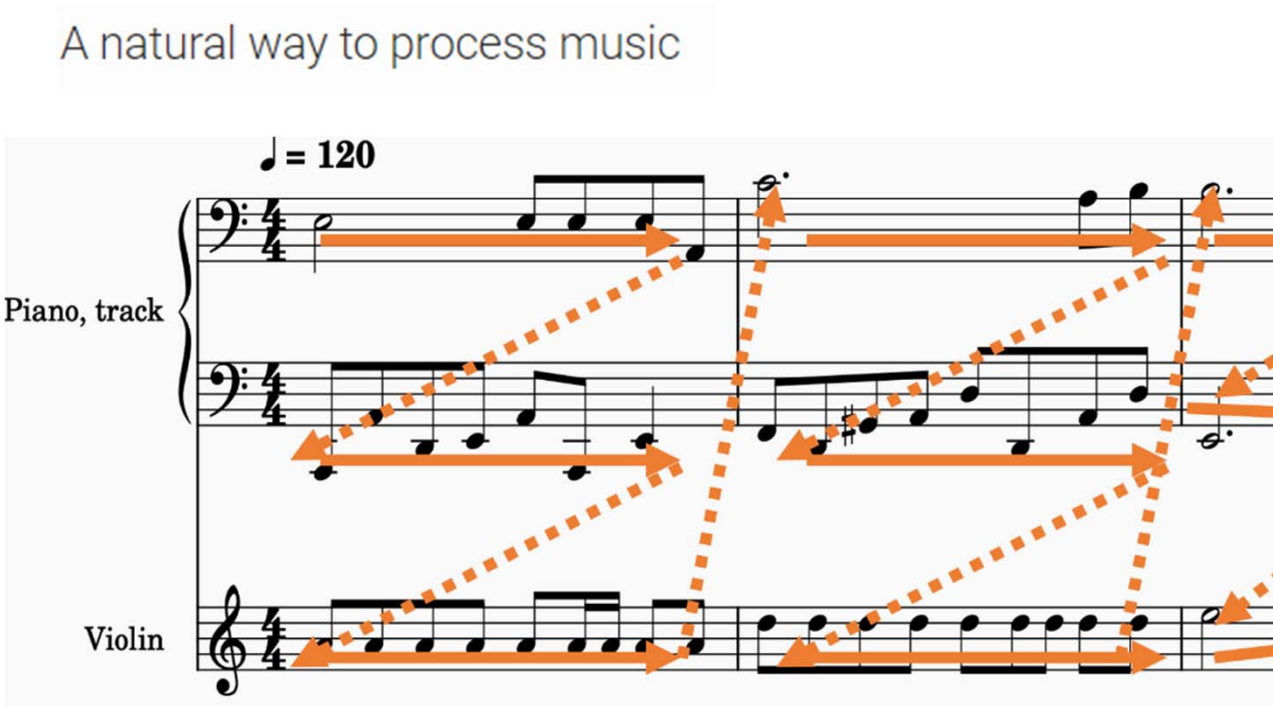
Table 3: Details of the CP representation in our implementation, including that of the sampling policy (τ-tempered top-ρ sampling). For the vocabulary size, the values in the parentheses denote the number of special tokens such as [ignore].

Issue 2: Token Order in Representing Multi-Track MIDI Music

https://musiclang.github.io/tokenizer/







Issue 3: Token Representation for Chords

https://musiclang.github.io/tokenizer/

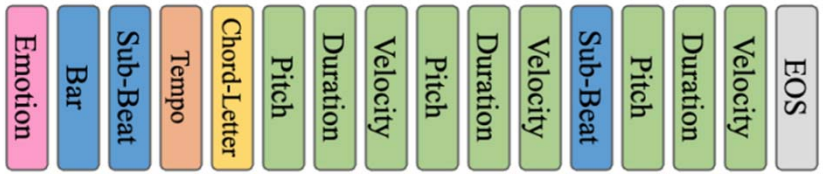
Each chord is depicted using seven tokens (six control tokens plus one "CHORD_CHANGE" token):

- **Scale degrees**: I, II, III, IV, V, VI, VII - represent the chord's degree within the scale. This does not specify the chord's type; the mode and extension are needed for that.
- **Tonality root**: 0 to 12, corresponding to each pitch class possible (C to B).
- **Tonality mode**: m for harmonic minor, M for major.
- **Chord octave**: -4 to 4, indicating the root note's octave relative to the 4th octave.
- **Extension**: Uses figured bass notation to detail the chord's bass and any extensions (e.g., '' (triad in root position), '6', '64', '7', '65', '43', '2'), including variations with sus2 or sus4 for suspended chords.
- **Duration numerator/denominator**: Numerical values representing the chord's fractional duration, providing control over the time signature (since one chord equals one bar in MusicLang).

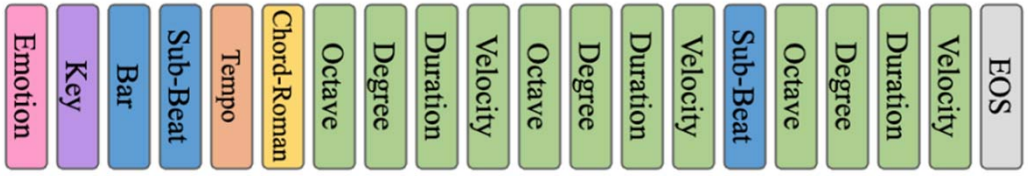
Issue 4: Token Representation for Key

https://emo-disentanger.github.io/

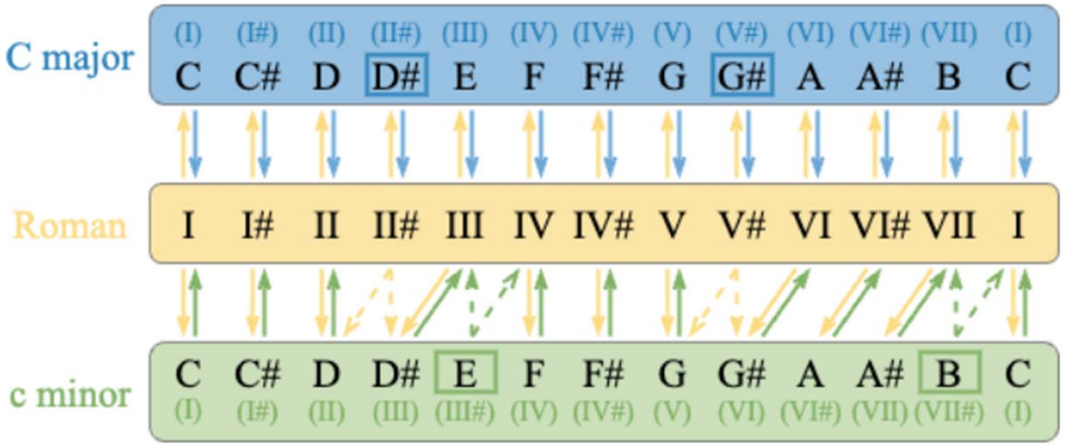
- **Functional** representation
  - Encode both melody and chords with Roman numerals relative to musical keys, to consider the interactions among notes, chords and tonalities



(a) REMI representation



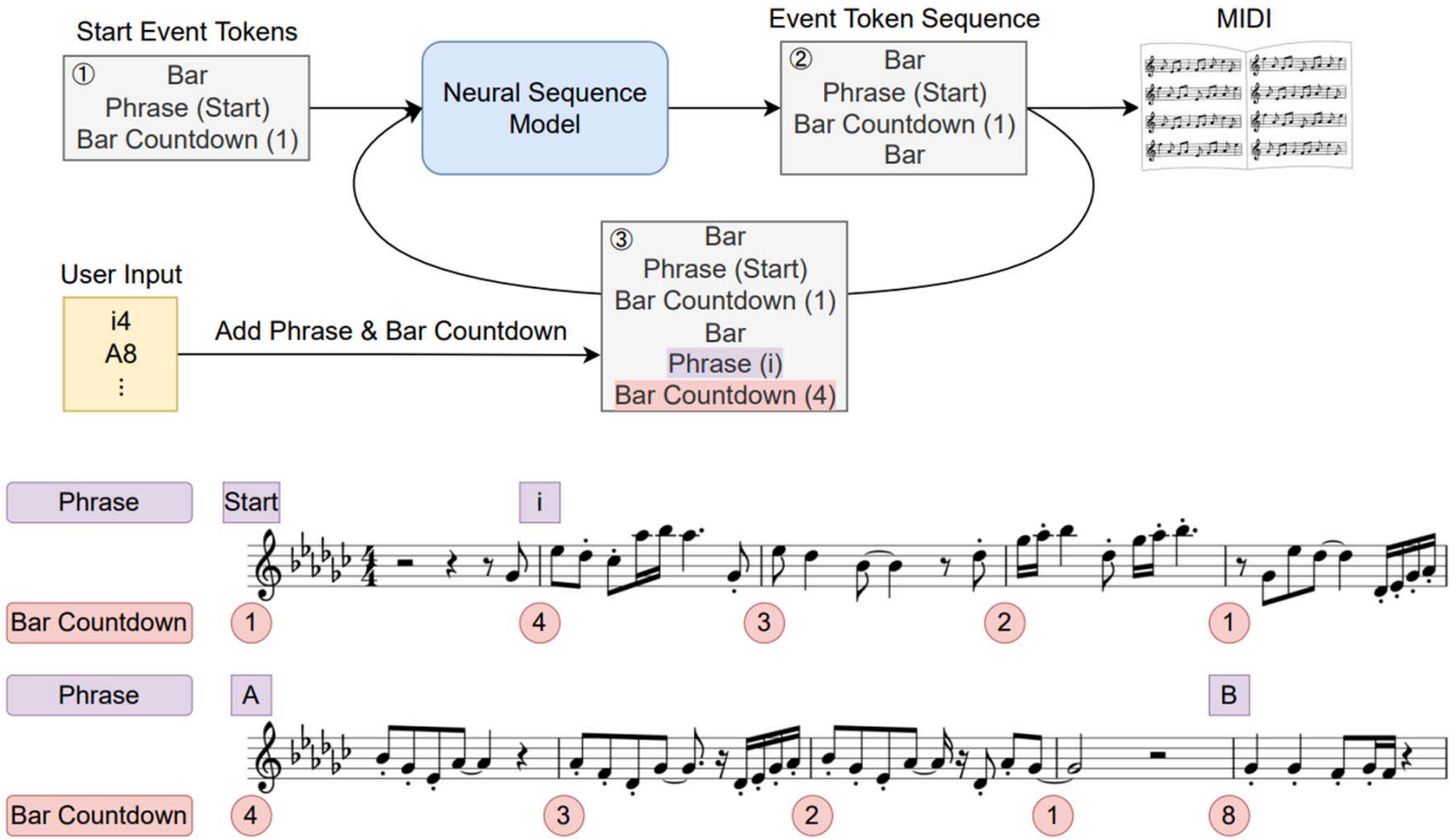
(b) Functional representation



Ref: Huang et al, “Emotion-driven piano music generation via two-stage disentanglement and functional representation”, ISMIR 2024

Issue 5: Token Representation for Musical Structure

https://mil-tokyo.github.io/phrase-length-designated-music-generation/



Outline

- Symbolic representations for music: Not just MIDI
- Text-based approach for symbolic music generation
- **Advanced models for MIDI generation**
  - Theme Transformer
  - MuseMorphose and VQVAE-Transformer
  - Compose-and-embellish
- Image-based approach for symbolic music generation

Conditional Methods for Transformers

	Condition type	Conditioning mechansim
Theme Transformer	token sequence (theme)	cross attention
MuseMorphose	single embedding (attributes)	in-attention
Compose & Embellish	token sequence (lead sheet)	interleaving prompt

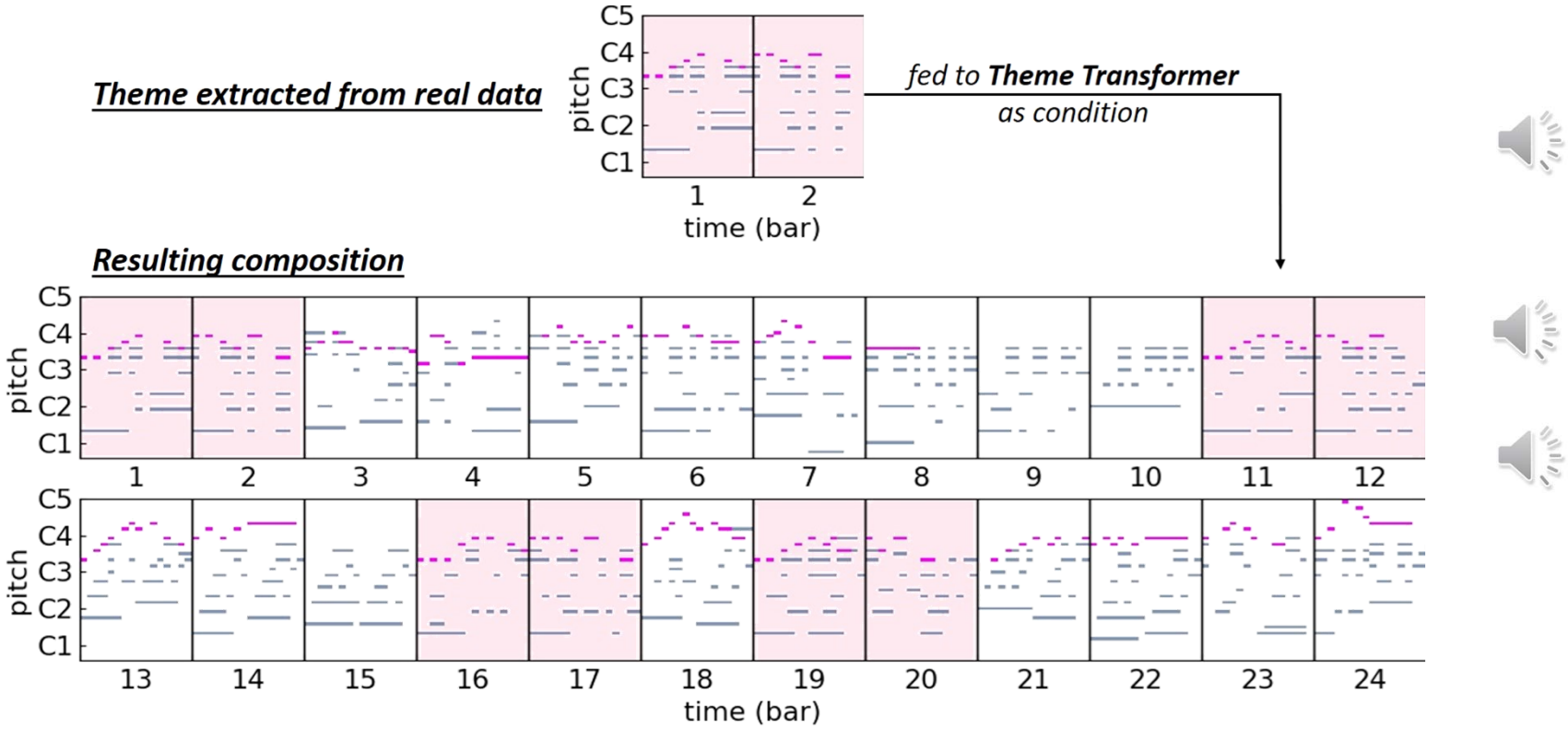
Theme Transformer: Theme-based Conditioning

- Conventional **prompt-based conditioning**
  - Cannot guarantee that the conditioning sequence would repeat or develop in the created continuation
  - Moreover, the influence of the prompt would diminish over time and the generated music *drifts away* from its beginning as the music gets longer
- Proposal: **theme-based conditioning**
  - **Automatic theme finding**: use fragments that have multiple occurrences in a music piece as the conditioning sequence at training time
  - Use **separate memory network** so that the model learns to **exploit the theme condition** when appropriate

Ref: Shih et al, “Theme Transformer: Symbolic music generation with theme-conditioned Transformer”, TMM 2023

Theme Transformer: Theme-based Generation

https://atosystem.github.io/ThemeTransformer/



Ref: Shih et al, “Theme Transformer: Symbolic music generation with theme-conditioned Transformer”, TMM 2023

Theme Transformer: Automatic Theme Finding

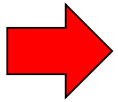
- Use **contrastive learning** to get **melody embedding**
  - **Negative pair**: fragments from different pieces
  - **Positive pair**: to be invariant to the following variations
    - Pitch shift on scale
    - Last note duration variation
    - Note splitting and combination

- Use DBSCAN to find clusters for each piece of music
- Pick the **largest cluster** as the theme for a piece of music

Theme Transformer: Token Design

- Use **THEME-START** and **THEME-END** tokens to indicate the beginning and ending of a ***theme region***

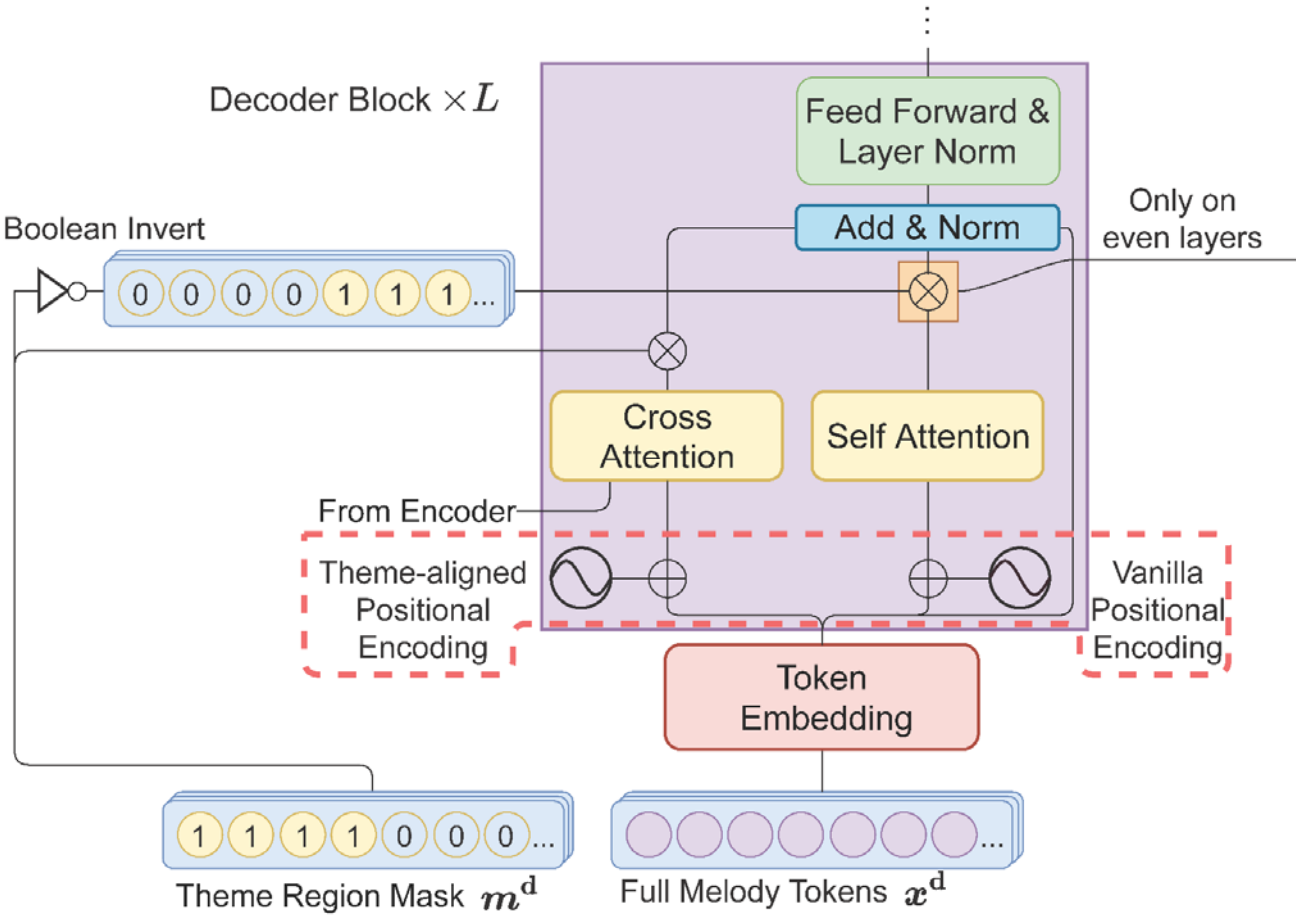
- At inference time, the model decides on its own when to enter a theme region



Token type	Piano Representation
NOTE-PITCH	127×2
NOTE-DURATION	64×2
NOTE-VELOCITY	126×2
TEMPO	76
BAR	1
SUBBEAT	16
THEME	2
PADDING	1
Total	730

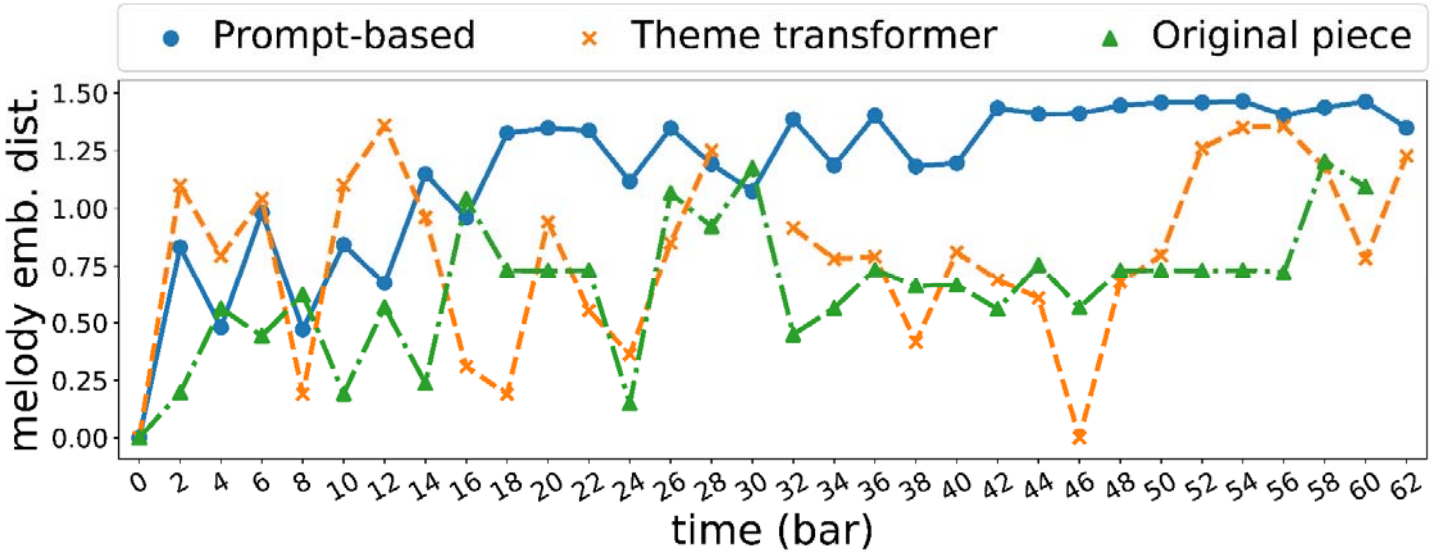
Theme Transformer: Attention Design

- **Separate memory network**
  - **Self-attention:** to attend to the past (local coherence)
  - **Cross-attention**: to attend to the theme
- **"Switch on"** cross-attention **only during theme regions**
- Dedicated **theme-aligned positional encoding**



Ref: Shih et al, "Theme Transformer: Symbolic music generation with theme-conditioned Transformer", TMM 2023

Theme Transformer: Evaluation



	Pitch class consistency↑	Melody inconsistency↓	Grooving consistency↑	Theme inconsistency↓	Theme uncontrollability↓
Baseline (Huang et al. 2019)	.59±.07	.33±.38	.84±.09	—	—
Seq2seq Transformer (proposed)	.61±.04	.46±.28	.90±.06	.22±.03	.45±.19
Theme Transformer (proposed)	.61±.06	.13±.24	.92±.07	.15±.06	.34±.13
Original pieces	.65±.05	.09±.18	.74±.10	.11±.06	.09±.06

Ref: Shih et al, “Theme Transformer: Symbolic music generation with theme-conditioned Transformer”, TMM 2023

Theme Transformer: Possible Extensions

- Extensions
  - Multiple themes
  - Application to other music genres
  - Use melody as the theme prompt (and make a UI)
  - CP token representation
  - Memory-based RNN
  - Variants of gating
  - To-copy pretrain
  - Better theme retrieval method
  - Better segmentation method

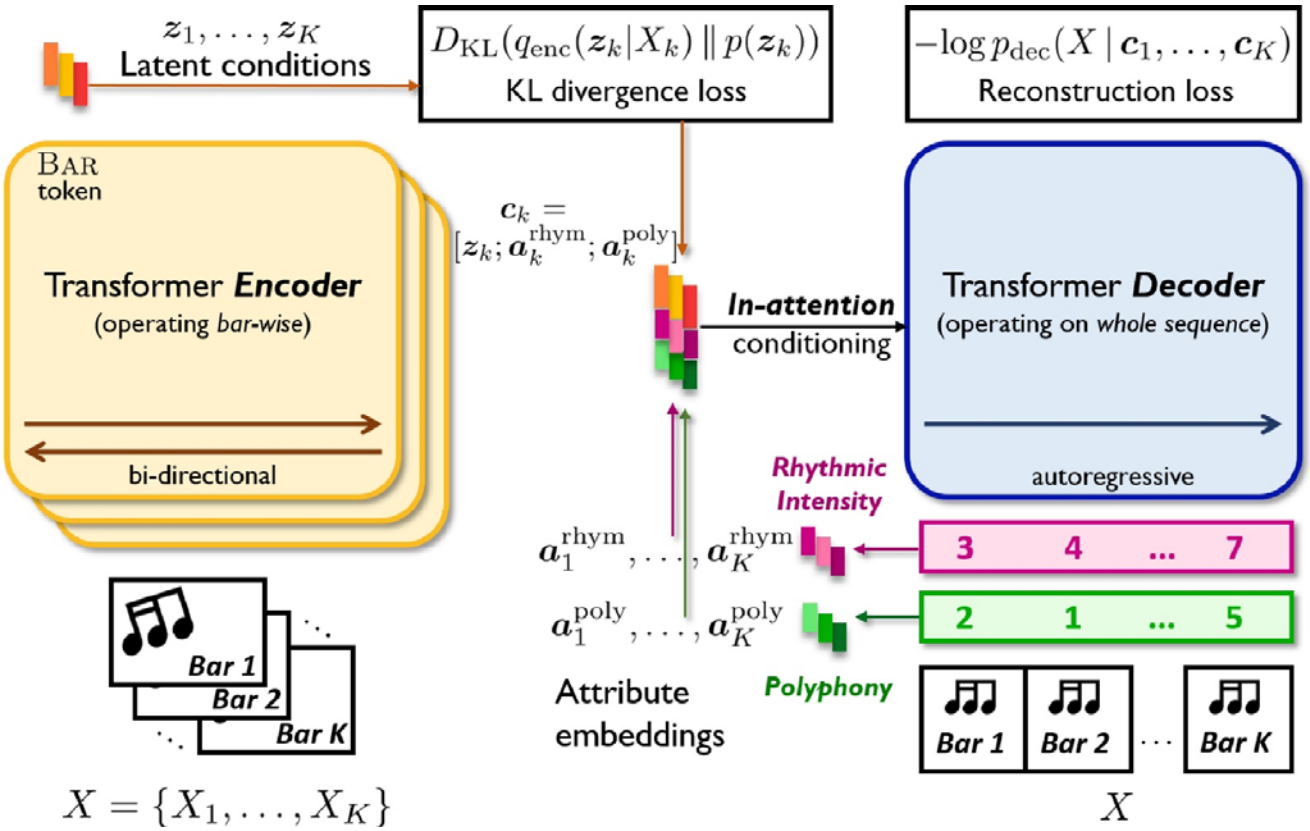
Theme Transformer: Strength and Weakness

- Allow us to repeat something over
- But
  - No development
  - Cannot control the structure

- Question: Can we generate a **skeleton/outline** of the music first, before we fill the details?

MuseMorphose: VAE-Transformer for Style Transfer

https://slseanwu.github.io/site-musemorphose/



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

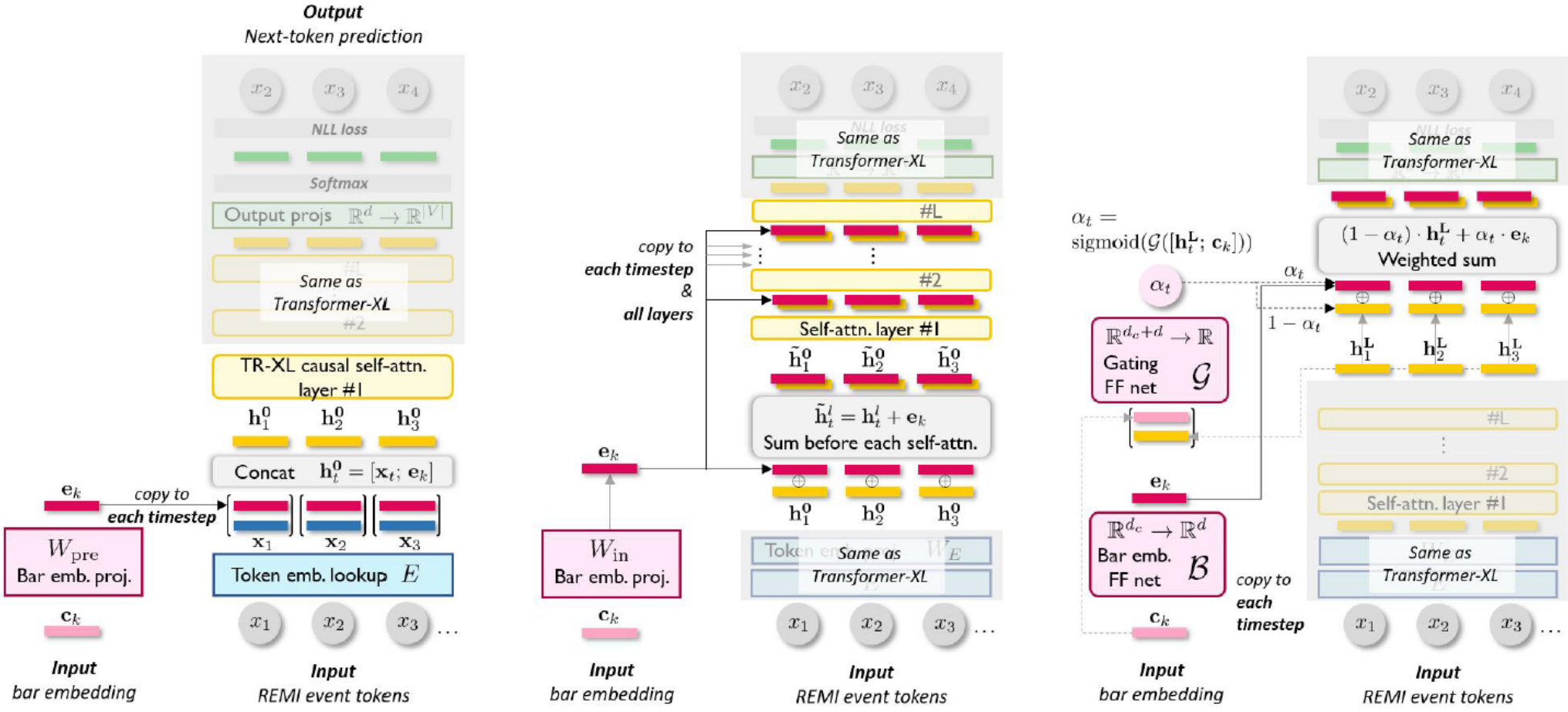
The Condition Might be Ignored!

- Happens when the decoder is powerful, or when the decoding is auto-regressive, where previous tokens in the sequence reveal strong information
  - Reduce the power of the decoder
  - Increase the relevance or understandability of the condition
  - Use a stronger conditioning mechanism
- Size of the information bottleneck
  - Having a narrow enough latent space bottleneck is a key to successful style control
  - When the latent space is unconstrained, as in the case of AE objective, the latent conditions may carry all necessary information for the decoder to reconstruct the music, rendering the other source of conditions, i.e., the attribute embeddings, which do not connect to the encoder, fruitless

$D_{\mathrm{KL}}(q_{\mathrm{enc}}(\boldsymbol{z}_k|X_k) \,\|\, p(\boldsymbol{z}_k))$
KL divergence loss

$-\log p_{\mathrm{dec}}(X \mid \boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
Reconstruction loss

MuseMorphose: Various Conditoining Mechansims



(a) Pre-attention conditioning (b) In-attention conditioning (c) Post-attention conditioning

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

MuseMorphose: VAE-Transformer for Style Transfer

Model	Best ckpt. at step	**Recons. NLL** *Train*	*Val.*	**KL divergence** *Train*	*Val.*
MIDI-VAE [19]	65K	0.676	0.894	0.697	0.682
Attr-Aware VAE [20]	190K	0.470	0.688	0.710	0.697
MuseMorphose (Ours), AE objective	90K	0.130	0.139	6.927	6.928
MuseMorphose (Ours), VAE objective	75K	0.697	0.765	0.485	0.484
MuseMorphose (Ours), preferred settings	140K	0.457	**0.584**	0.636	**0.636**

bold: leads the baselines under the same latent space constraints

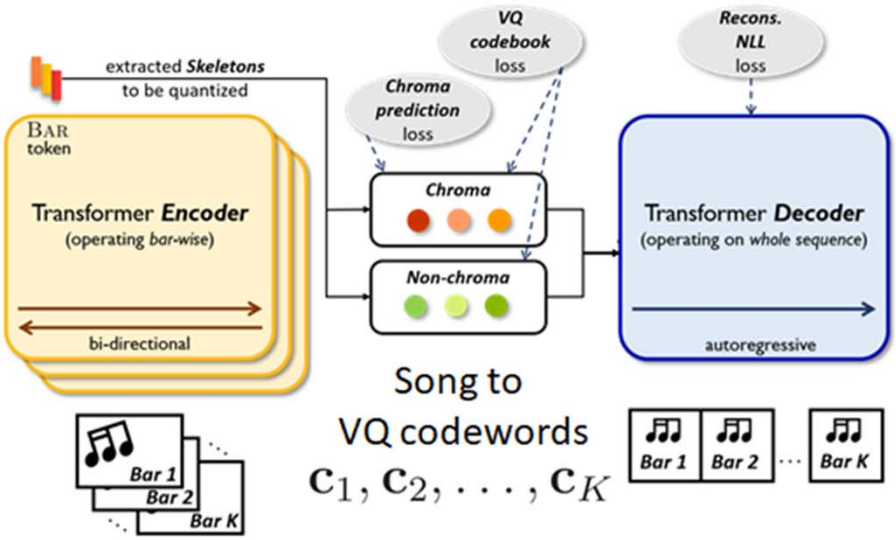
Model	**Fidelity** $sim_{\text{chr}}\uparrow$	$sim_{\text{grv}}\uparrow$	**Control** $\rho_{\text{rhym}}\uparrow$	$\rho_{\text{poly}}\uparrow$	**Fluency** PPL$\downarrow$	**Diversity** $sim_{\text{chr}}\downarrow$	$sim_{\text{grv}}\downarrow$
MIDI-VAE [19]	75.6	85.4	.719	.261	8.84	74.8	86.5
Attr-Aware VAE [20]	85.0	76.8	**.997**	.781	10.7	86.5	**84.7**
Ours, AE objective	**98.5**	**95.7**	.181	.154	**6.10**	97.9	95.4
Ours, VAE objective	78.6	80.7	.931	.884	7.89	**73.2**	84.9
Ours, preferred settings	91.2	84.5	.950	**.885**	7.39	87.1	87.6

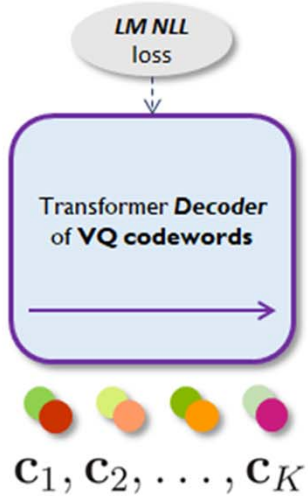
Ref: Wu & Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE", TASLP 2023

VQVAE Transformer for Structured Generation

Low-level Model:
Structure Extractor + Conditional Decoder

High-level Model:
Structure Generator





(Slide made by Shih-Lun Wu; unpublished)

VQVAE Transformer Result

- **Conditional generation:** given VQs from songs in test set
- **Better structure?** -- note n-gram repetition stats

		Short (3∼6 notes)	Mid (7∼15 notes)	Long (16∼40 notes)
Unique %	**Real data**	42.67	60.85	75.49
	VQ model	45.30	64.05	76.95
	Uncond. model	9.32	15.23	27.13
Reappearance interval (in bars)	**Real data**	16.61	23.68	29.22
	VQ model	16.33	23.74	29.08
	Uncond. model	2.74	3.56	4.70

- **Possible plagiarism?** -- len of longest common note string
  - **VQ gen** vs. **Ref real song:** 6.24 +/- 1.98 notes
  - Random real song pairs: 3.79 +/- 1.49 notes

(Slide made by Shih-Lun Wu; unpublished results)

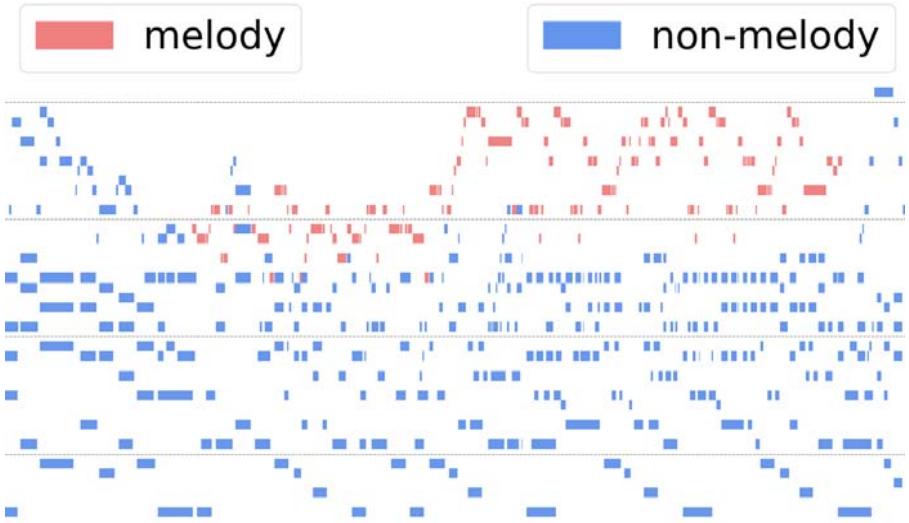
VQVAE Transformer

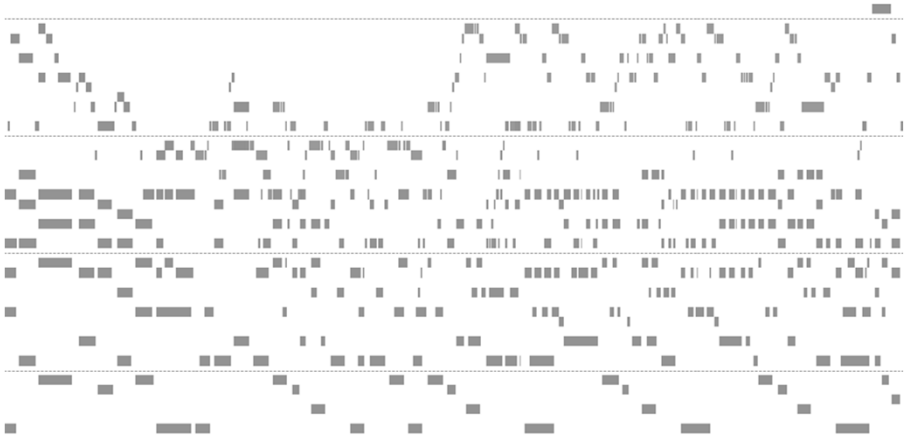
- May allow us to generate a **skeleton/outline** of the music first, before we fill the details
- But
  - Empirical result incomplete (not yet finished)

- Question: The VQVAE approach sounds quite brute-force, is there a **musically-inspired** approach?

Domain Knowledge Inspired Approach

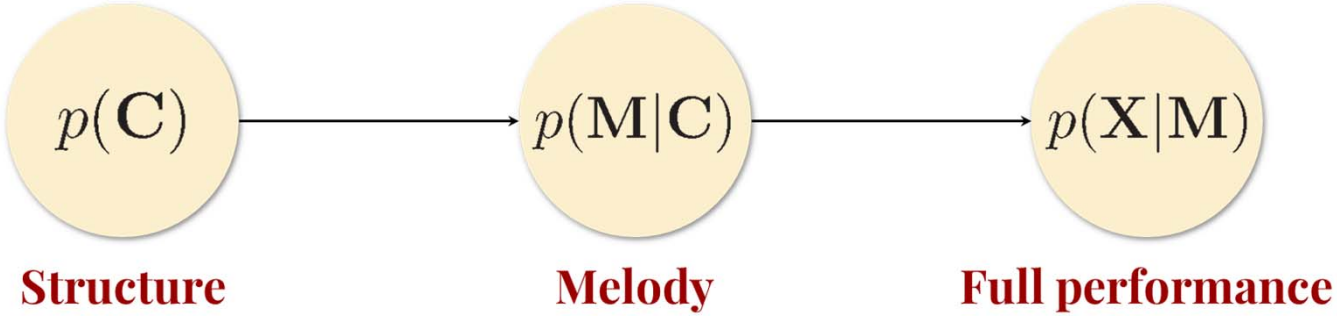
Modular approach > End-to-end approach





Generate melody first,
then the piano accompaniment

Generate Melody First, then the Piano

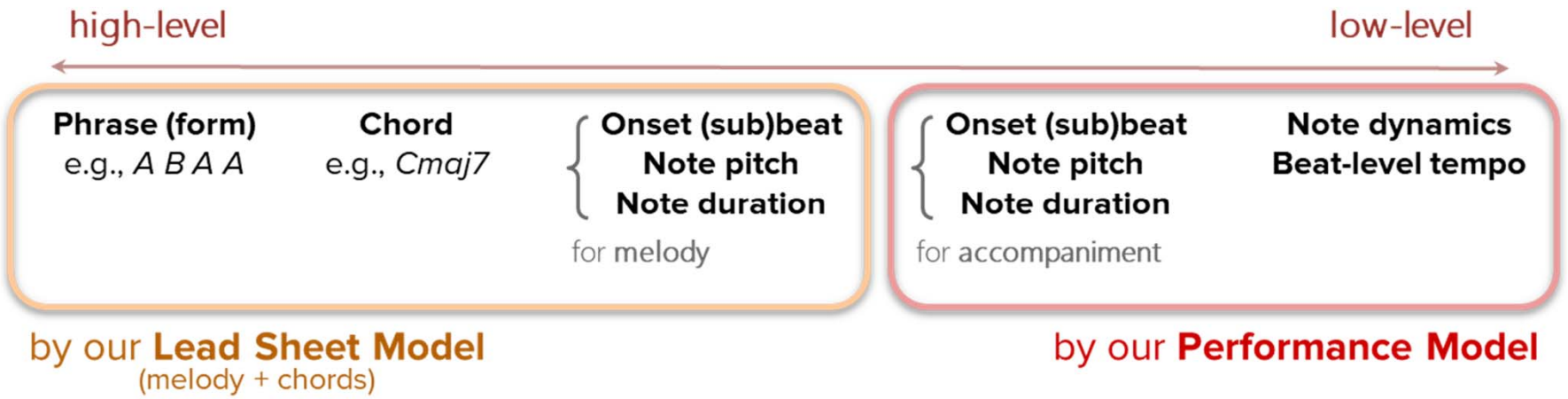


- End-to-end full-song generation is hard; will a 2-stage model make things easier? → Generate **lead sheet** (melody & chords) first, and then the **full piano performance**
- Can we pretrain the melody stage with non-piano data to do better?

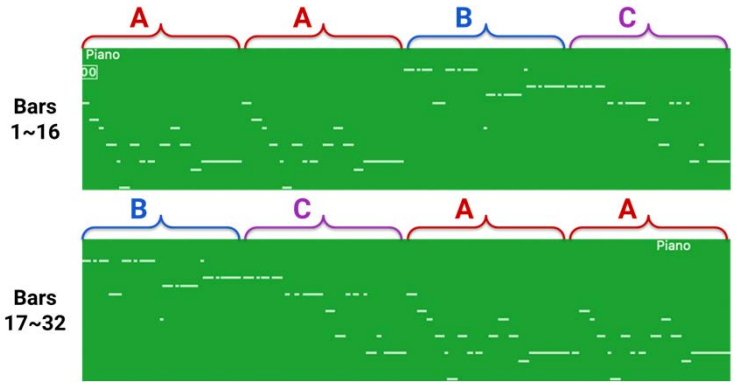
(Slide made by Shih-Lun Wu)

Compose & Embellish

(Figure made by Shih-Lun Wu)



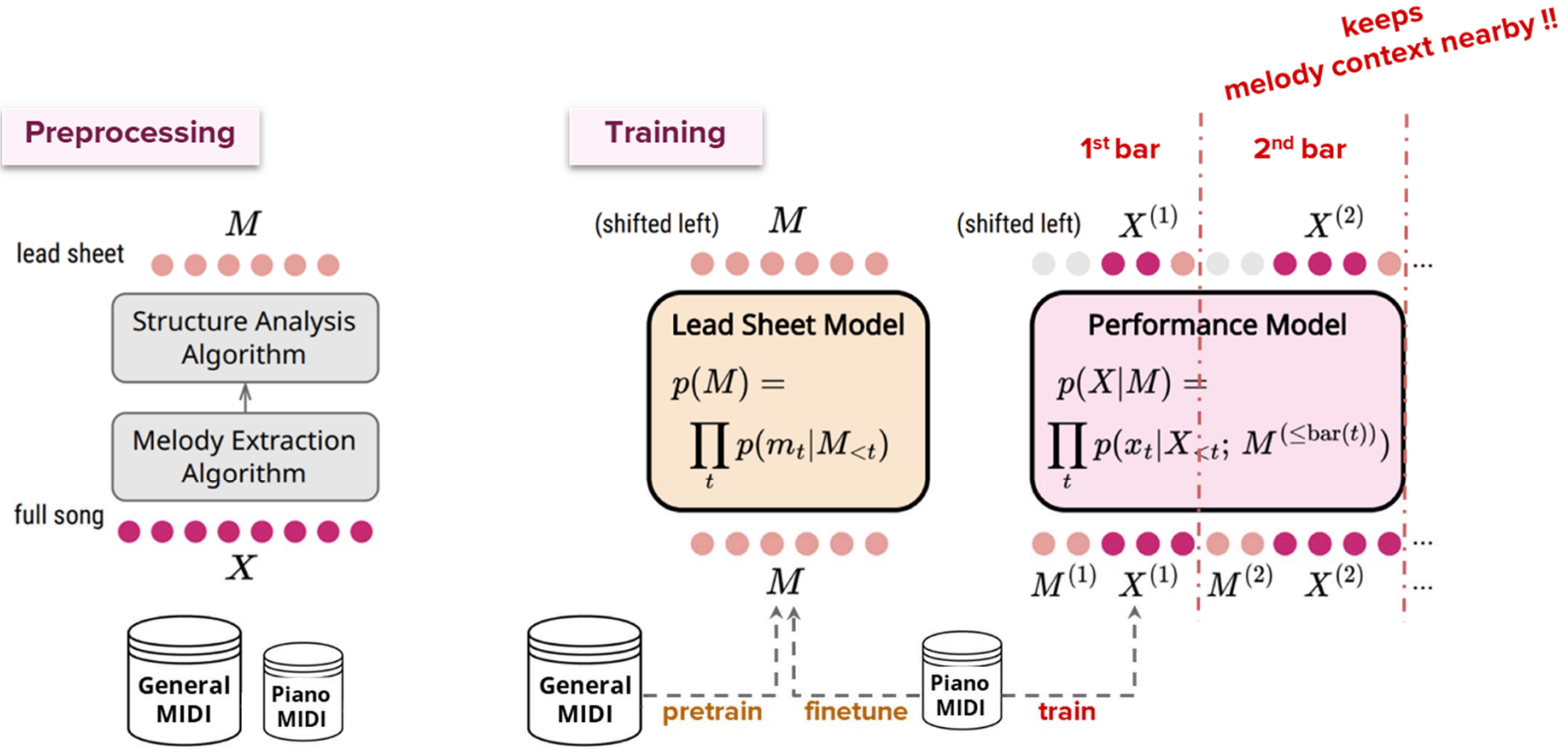
- A two-stage model
  - **Compose** lead sheet & form
  - **Embellish** it to create piano performance



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

Compose & Embellish

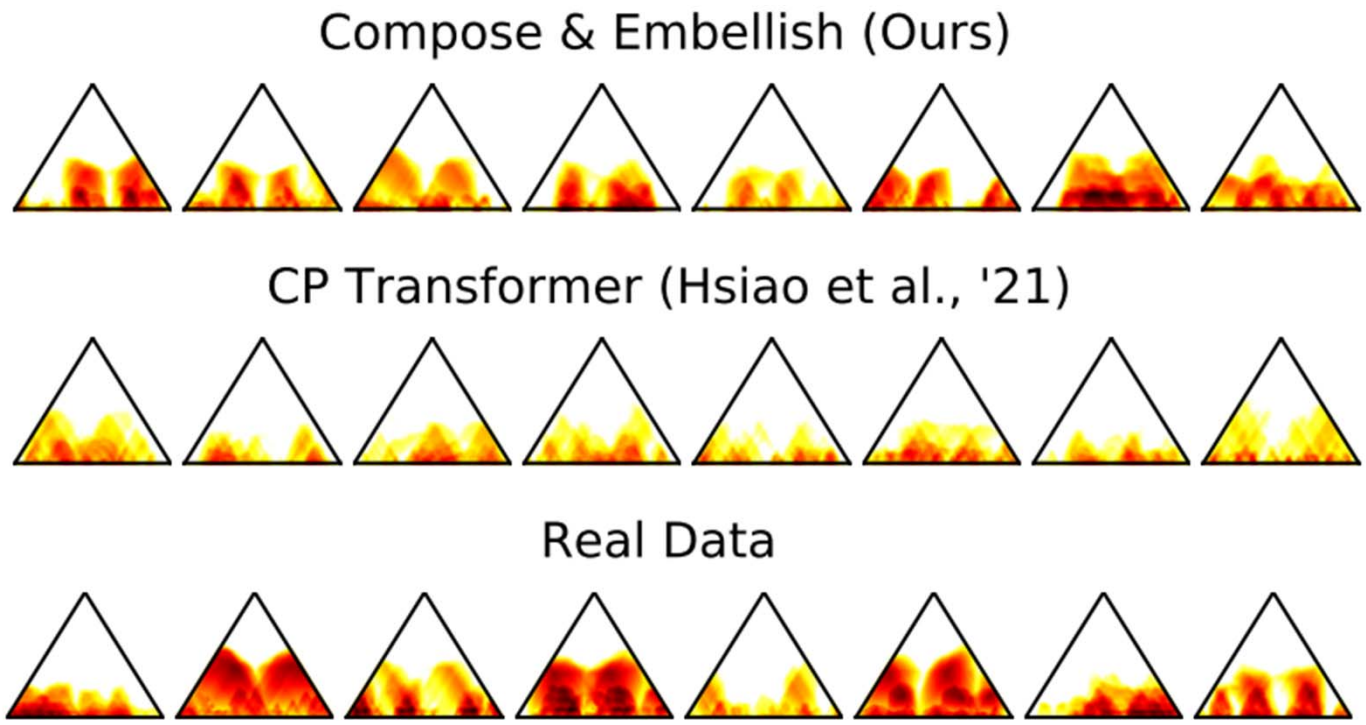
(Figure made by Shih-Lun Wu)



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

Compose & Embellish: Result

https://github.com/slSeanWU/Compose_and_Embellish



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

Compose & Embellish: Result

- Greatly outperforms CP Transformer in structureness
- Ablations show that both the use of structure tokens ("struct") and the lead sheet data pre-training help

	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
COMPOSE & EMBELLISH	**36.8** ±6.7	**35.1** ±7.7	**25.8** ±12	**49.7** ±19	69.9 ±20	81.6 ±17
w/o struct	**36.8** ±6.8	34.2 ±9.2	23.8 ±11	48.0 ±17	68.7 ±17	82.7 ±14
w/o pretrain	36.6 ±7.5	33.1 ±8.8	19.6 ±10	53.2 ±19	**74.9** ±18	**87.9** ±14
w/o struct & pretrain	36.3 ±6.0	34.1 ±6.7	23.0 ±8.4	52.5 ±18	76.1 ±16	89.0 ±11
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

Compose & Embellish: Result

- 5-point Likert scale on 5 aspects
  - coherence (Ch), correctness (Cr), structureness (S), richness (R), overall (O)

	Ch	Cr	S	R	O
CPT	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

(StDevs of *individual data points* shown after ±)

- Large (>1 point) gain over CP Transformer on coherence & structureness
- Still a long way to go to rival humans

(Figure made by Shih-Lun Wu)

Recap

- **Theme Transformer**
  - Allow for explicit **conditioning of "theme"** thru customized cross attention design
  - The idea is quite general and applicable to various types of music (melody, single-track piano, multi-track MIDI etc)

- **VAE Transformer (MuseMorphose)**
  - Allow for **time-varying condition**
  - For **style transfer**, and variation generation (as it needs a reference MIDI)
  - The idea can be extended to multi-track MIDI
  - The input can take audio files instead

Recap

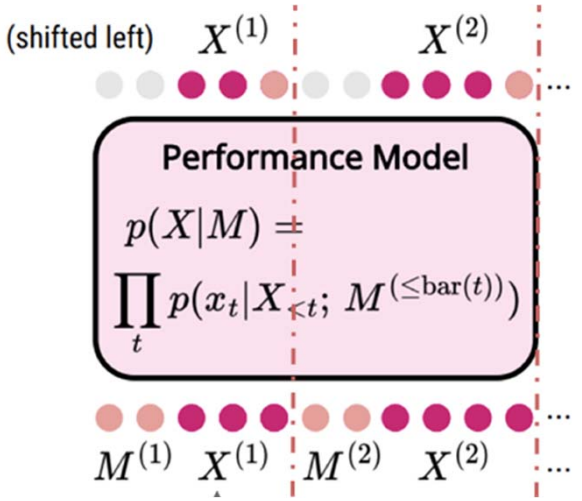
- **VQVAE Transformer**
  - A brute-force approach for unconditional MIDI generation that generates a **blueprint** first, before filling the details
  - Can be extended to multi-track MIDI

- **Compose-and-embellish**
  - A musically-inspired approach for unconditional MIDI generation that generates a ***lead sheet* as the blueprint** first, before filling the details
  - More interpretable, interactive; can use unpaired lead sheet data for pretraining
  - Can be extended to multi-track MIDI
  - Can be combined with Theme Transformer
  - The input can take audio files instead

Transformer-based Conditional Generation

	Condition type	Conditioning mechanism
Theme Transformer	token sequence (theme)	cross attention
MuseMorphose	single embedding (attributes)	in-attention
Compose & Embellish	token sequence (lead sheet)	interleaving prompt

- Decoder-only models tends to be easier to train than encoder/decoder models
- Use “bar” for segmentation tends to be useful for symbolic music generation
- Special cares might be needed to avoid the conditions to be ignored

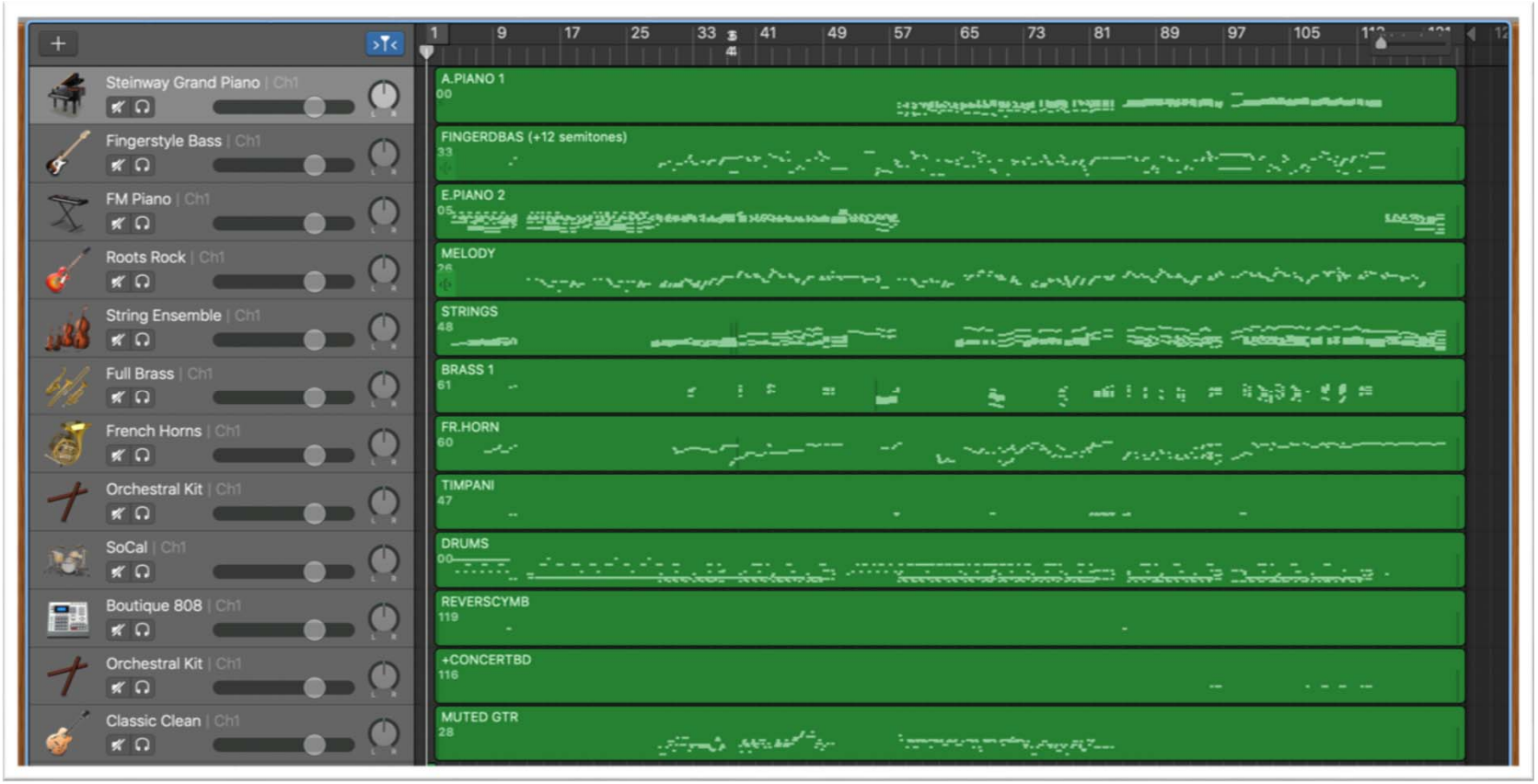


Outline

- Symbolic representations for music: Not just MIDI
- Text-based approach for symbolic music generation
- Advanced models for MIDI generation
- **Image-based approach for symbolic music generation**

Two Main Approaches

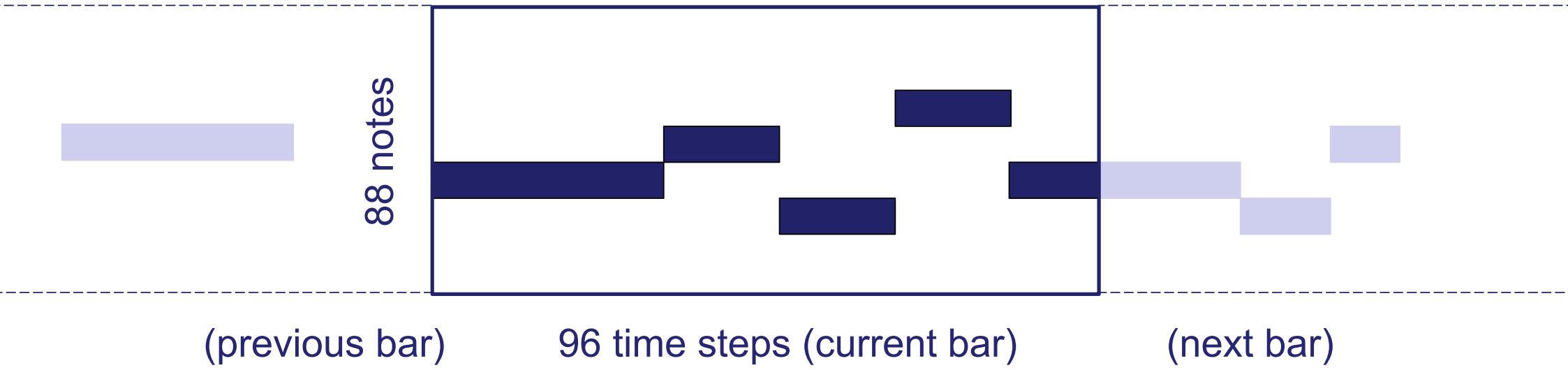
(1) Consider MIDI as **image** using the piano roll representation



(image from the internet)

Piano Roll

Represent each bar of a single-track MIDI as a fixed-size matrix (thus an image)

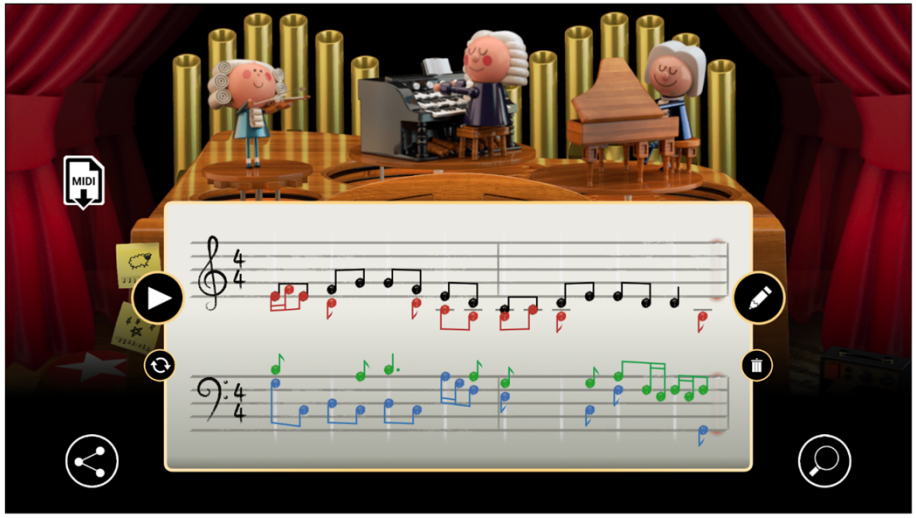


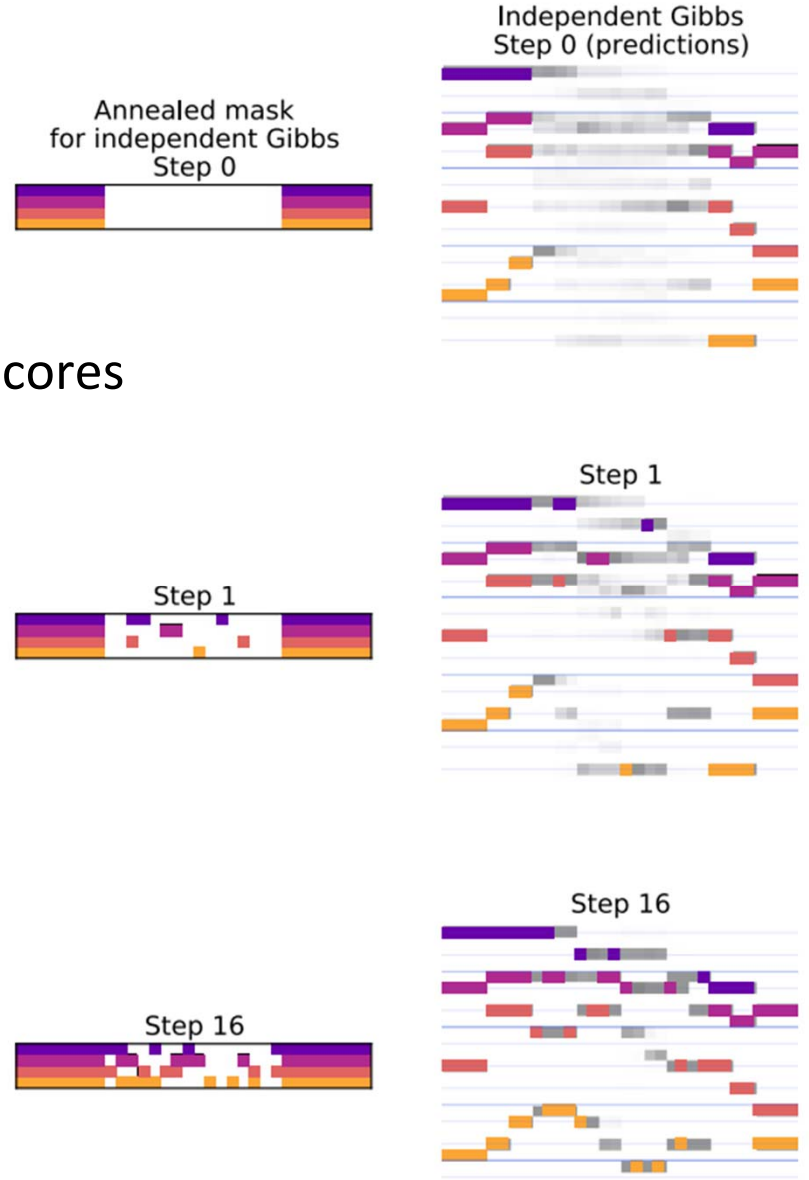
Ref: Dong et al, “Pypianoroll: Open source Python package for handling multitrack pianorolls”, ISMIR-LBD 2018

Coconet & Bach Doodle: Non-GAN CNN-based Approach

https://magenta.tensorflow.org/coconet

- For 2-bar piano roll "**infilling**": to complete *partial* scores
- Blocked-Gibbs sampling (non-autoregressive)



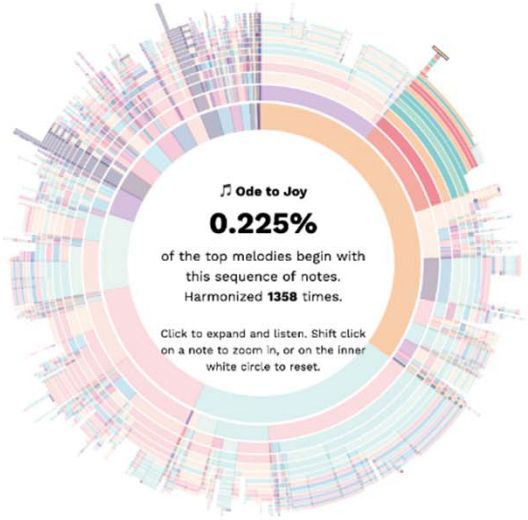


Ref: Huang et al, "Counterpoint by convolution", ISMIR 2017

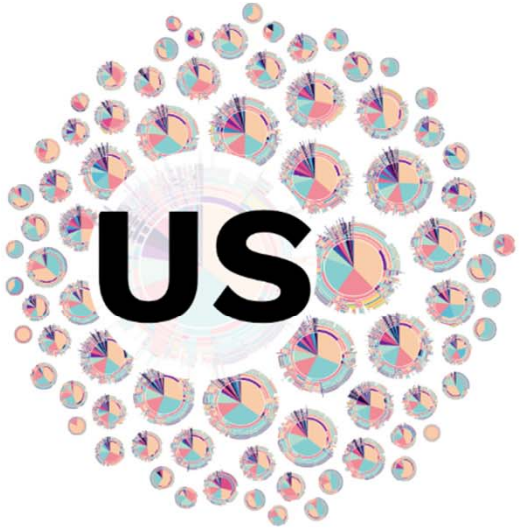
Coconet & Bach Doodle:
Non-GAN CNN-based Approach

https://magenta.tensorflow.org/datasets/bach-doodle

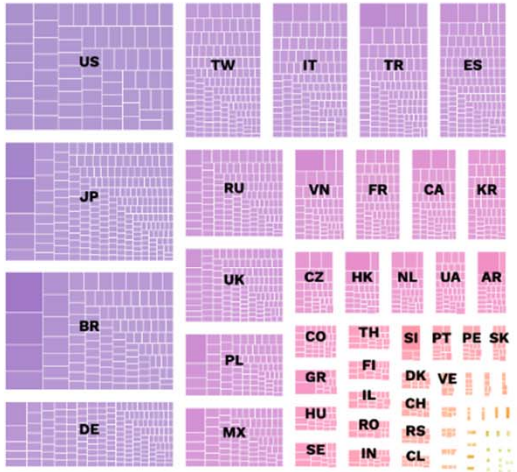
- "In three days, people spent 350 years worth of time playing with the Bach Doodle, and Coconet received more than 55 million queries"



Top overall repeated melodies



Top repeated melodies per country

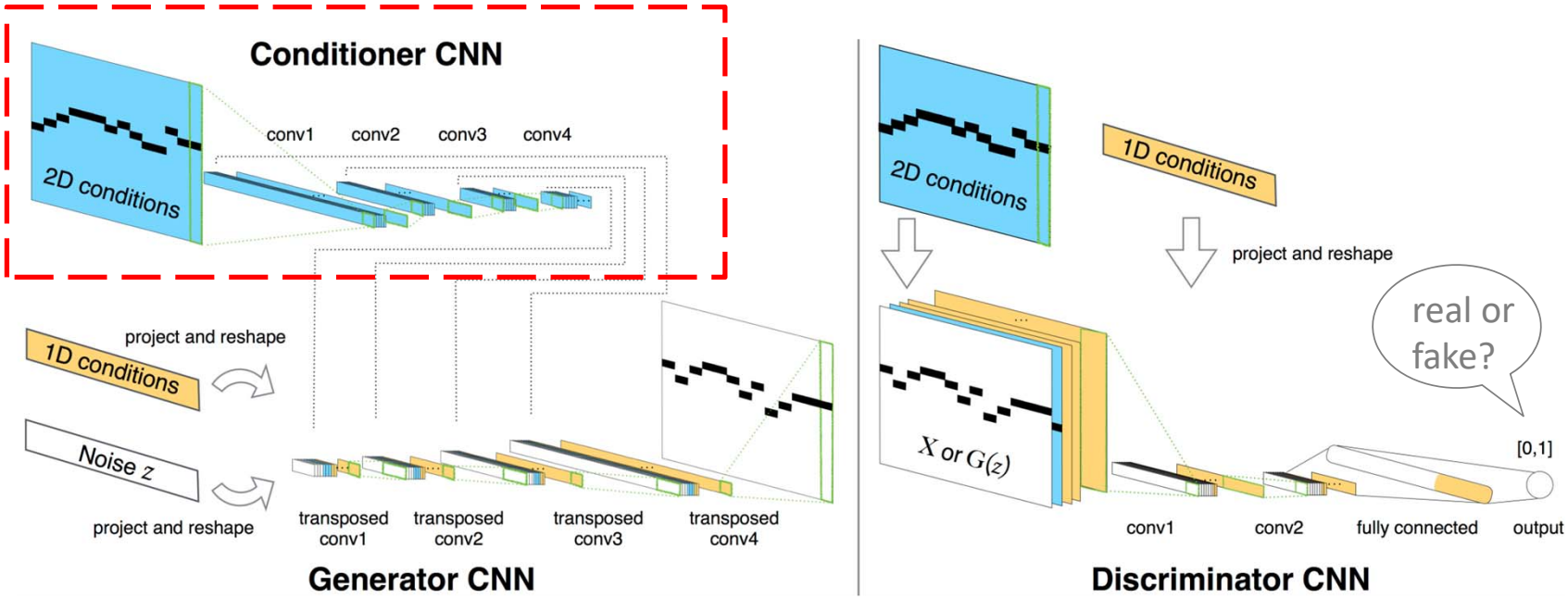


Unique regional hits

Ref: Huang et al, "The Bach Doodle: Approachable music composition with machine learning at scale", ISMIR 2019

MidiNet

- Convolutional GAN for **one-bar melody** generation
  - Turn a random vector into a piano-roll like matrix (non-autoregressive note-wise)
  - Conditioned on the previous bar (2D condition), or on the chord (1D condition)

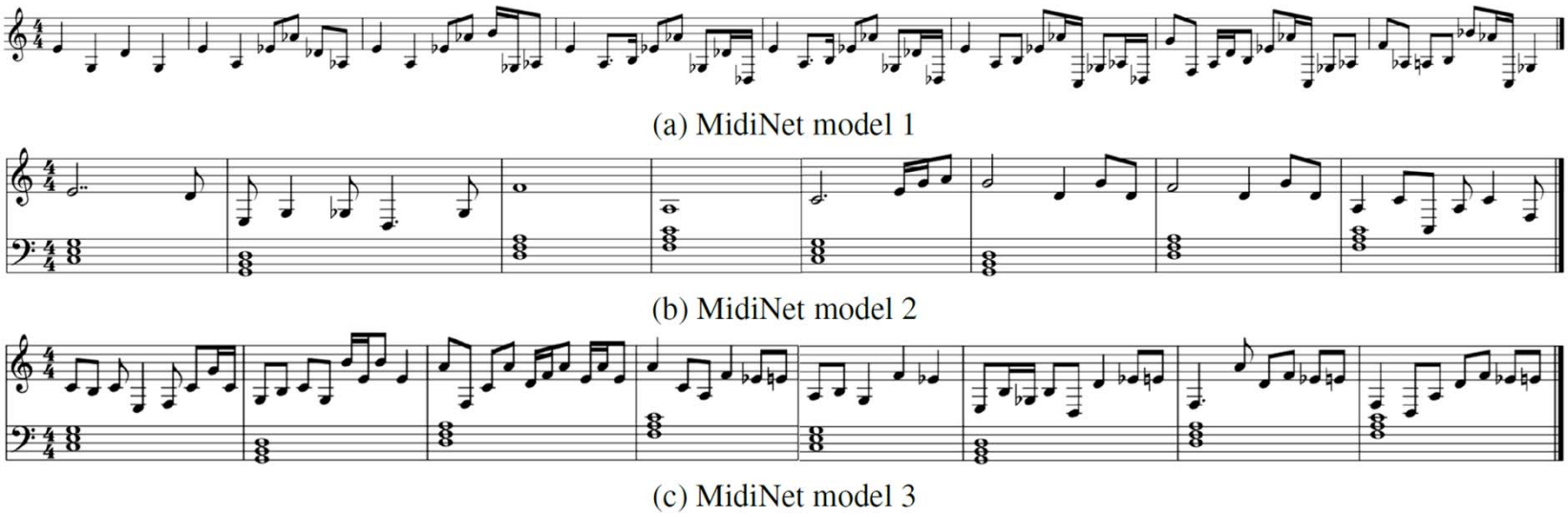


Ref: Yang et al, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation", ISMIR 2017

MidiNet: Examples

https://richardyang40148.github.io/TheBlog/midinet_arxiv_demo.html

- Variants of MidiNet 1 2 3



(a) MidiNet model 1

(b) MidiNet model 2

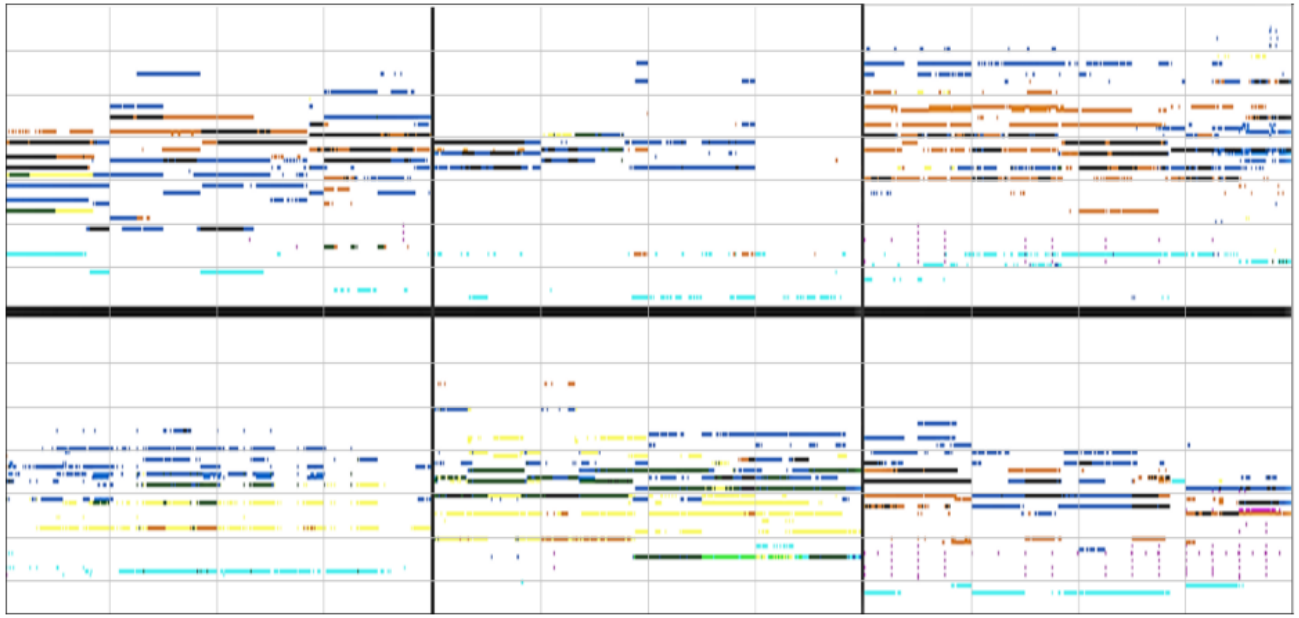
(c) MidiNet model 3

- With drums

Ref: Yang et al, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation", ISMIR 2017

Multi-track Piano Roll

- Represent difference voices (tracks) in different channels of a fixed-size tensor

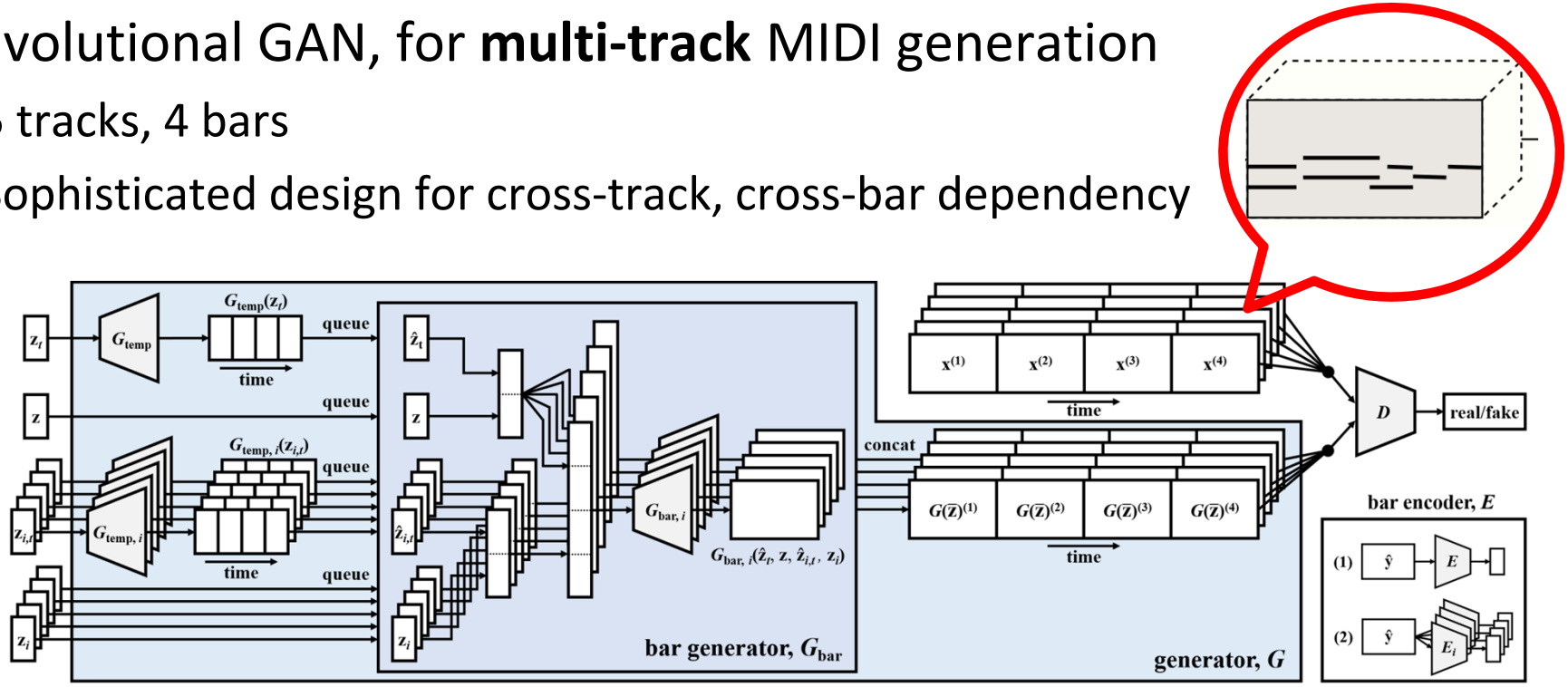


(different colors represent different tracks in this visualization)

MuseGAN

https://salu133445.github.io/musegan/

- Convolutional GAN, for **multi-track** MIDI generation
  - 5 tracks, 4 bars
  - Sophisticated design for cross-track, cross-bar dependency

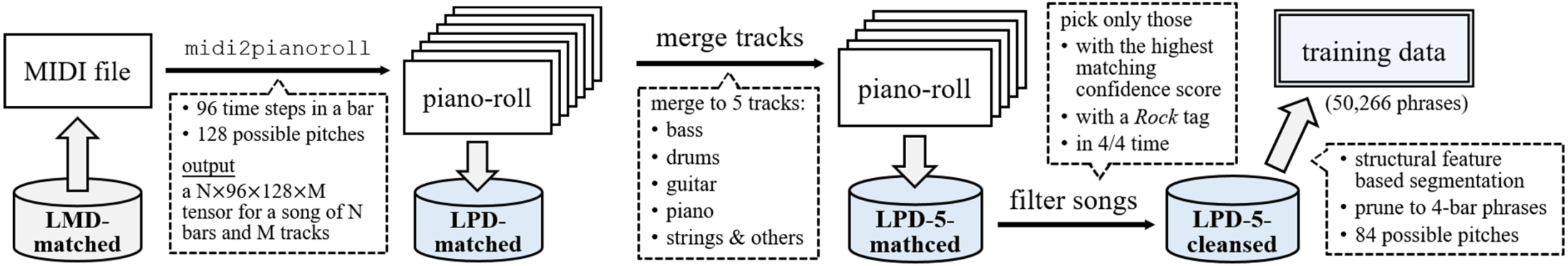


Ref: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

MuseGAN: Data

https://salu133445.github.io/musegan/dataset

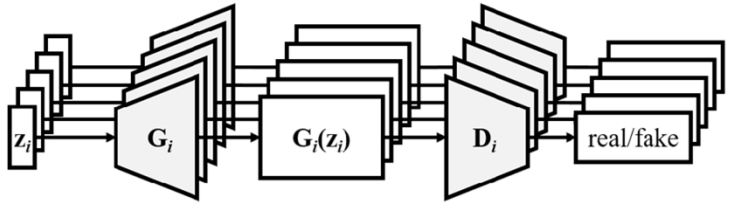
- LPD dataset: **128K MIDIs (piano-rolls)** from LMD (http://colinraffel.com/projects/lmd/)



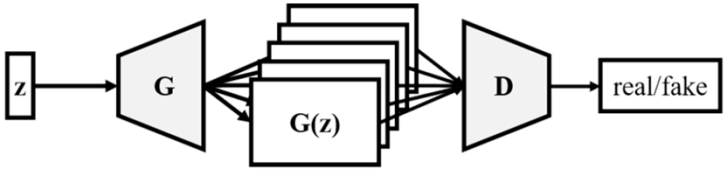
Ref: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

MuseGAN: Intra- & Inter-tracks

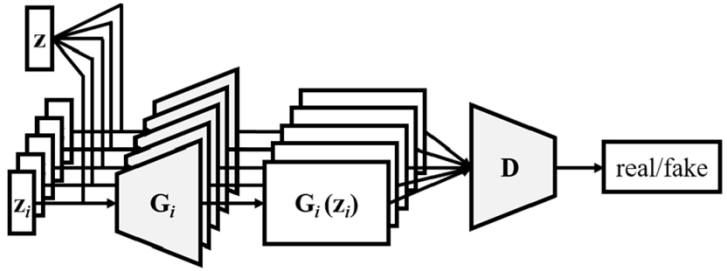
- Multi-track
  - Piano, guitar, bass, strings, drums
- **Hybrid model**
  - One "**shared**" (inter) z
  - Five "**private**" (intra) z_i
  - **Five** generators
  - **One** discriminator



(a) the jamming model



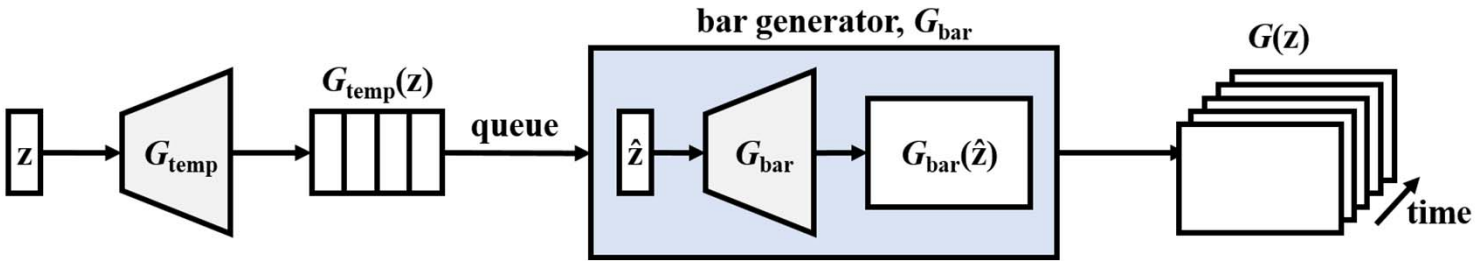
(b) the composer model



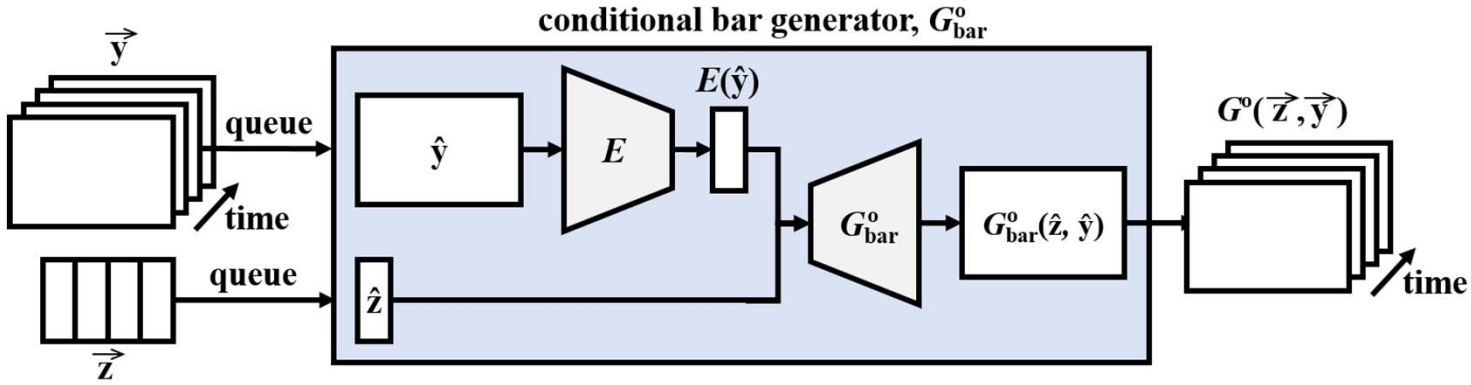
(c) the hybrid model

Ref: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

MuseGAN: Temporal Model



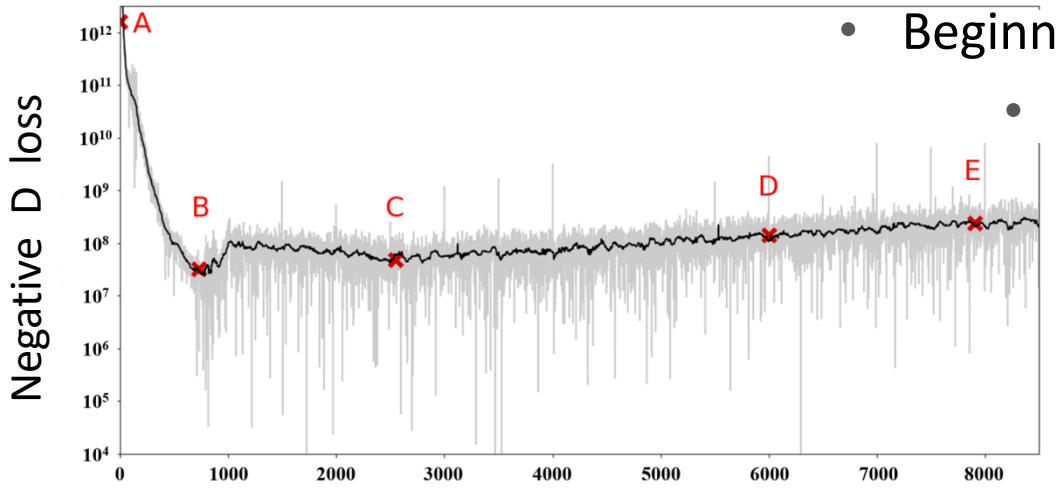
(a) generation from scratch



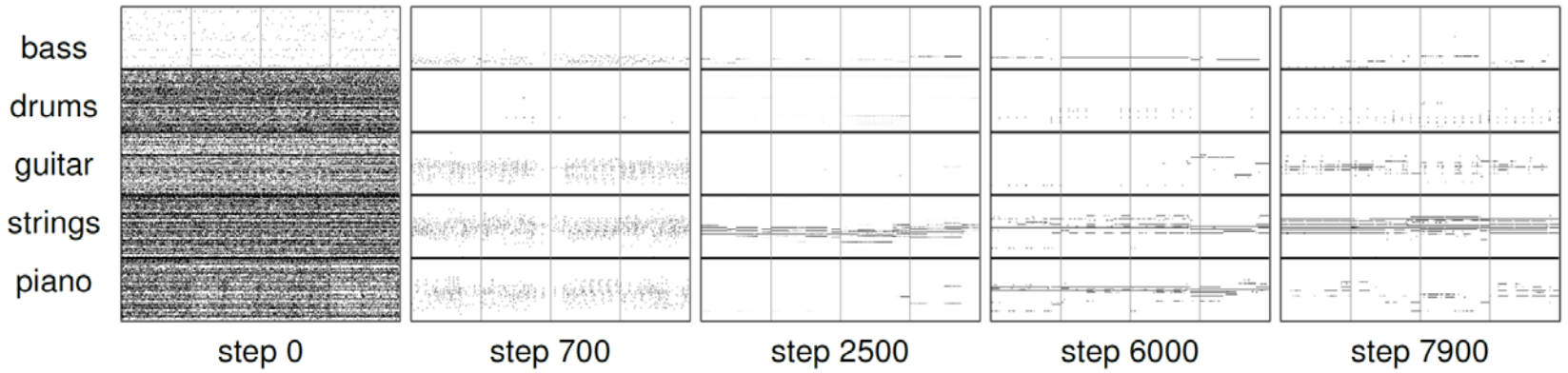
(b) track-conditional generation

Ref: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

MuseGAN: WGAN-gp



- Beginning – 500:1 D/G updates
- Later – 5:1 D/G updates
- Training time: <24 hours



Ref: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

MuseGAN: G & D

Input: $\mathbf{z} \in \Re^{128}$					
reshaped to $(1, 1) \times 128$ channels					
transconv	1024	2×1	$(2, 1)$	BN	ReLU
transconv	256	2×1	$(2, 1)$	BN	ReLU
transconv	256	2×1	$(2, 1)$	BN	ReLU
transconv	256	2×1	$(2, 1)$	BN	ReLU
transconv	128	3×1	$(3, 1)$	BN	ReLU
transconv	64	1×7	$(1, 7)$	BN	ReLU
transconv	K_{bar}	1×12	$(1, 12)$	BN	tanh
Output: $G_{\text{bar}}(\mathbf{z}) \in \Re^{96 \times 84 \times K_{\text{bar}}}$ (K_{bar}-track piano-roll)					

(b) the bar generator G_{bar}

- Grow **time steps** first
  - 1 → 2 → 4 → 8 → 16 → 32 → 96
- Then **notes** (freq)
  - octave (7)
  - then, pitch (84)

Input: $\tilde{\mathbf{x}} \in \Re^{4 \times 96 \times 84 \times 5}$ (real/fake piano-rolls of 5 tracks)				
reshaped to $(4, 96, 84) \times 5$ channels				
conv	128	$2 \times 1 \times 1$	$(1, 1, 1)$	LReLU
conv	128	$3 \times 1 \times 1$	$(1, 1, 1)$	LReLU
conv	128	$1 \times 1 \times 12$	$(1, 1, 12)$	LReLU
conv	128	$1 \times 1 \times 7$	$(1, 1, 7)$	LReLU
conv	128	$1 \times 2 \times 1$	$(1, 2, 1)$	LReLU
conv	128	$1 \times 2 \times 1$	$(1, 2, 1)$	LReLU
conv	256	$1 \times 4 \times 1$	$(1, 2, 1)$	LReLU
conv	512	$1 \times 3 \times 1$	$(1, 2, 1)$	LReLU
fully-connected	1024			LReLU
fully-connected	1			
Output: $D(\tilde{\mathbf{x}}) \in \Re$				

(c) the discriminator D

Ref: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

Polyfussion

https://polyffusion.github.io/

- Work on pixels by treating pianorolls as images
- Support a variety of tasks
  - Unconditional Generation
  - Melody generation given accompaniment
  - Accompaniment generation given melody
  - Arbitrary segment inpainting
  - Iterative inpainting for long-term generation
  - Generation with chord conditioning
  - Generation with texture conditioning

Ref: Min et al, "Polyffusion: A Diffusion model for polyphonic score generation with internal and external controls," ISMIR 2023

Two Main Approaches

(1) Consider MIDI as **image** using the piano roll representation

- Works better for generating **patterns**
- Capture **spatial-temporal** information
- May be Less effective for modeling long sequences

Whole-song Hierarchical Generation of Symbolic Music

https://wholesonggen.github.io/

Table 1: Definition of the four-level hierarchical music language. We use m for measure, b for beat, s for step, to represent the temporal resolution. M denotes the number of measures in a piece, γ denote the number of beats in a measure, and δ denotes the number of steps in a beat.

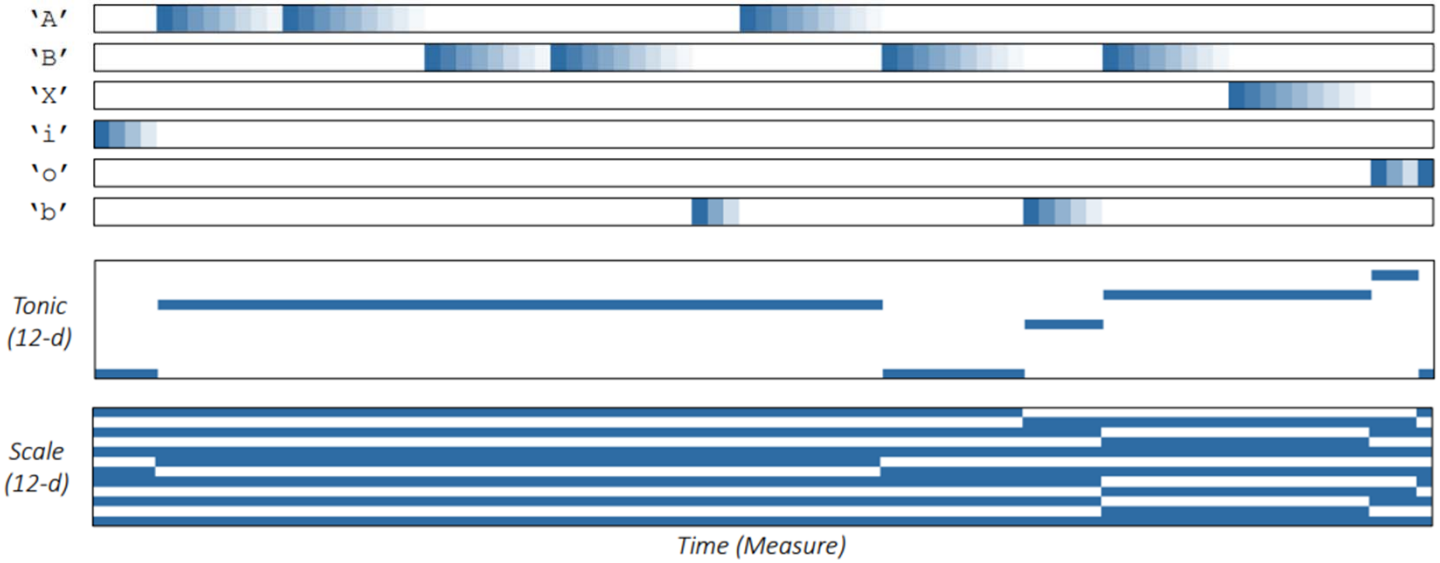
Languages (res.)	**Specification**	**Data Representation**	**Structural Focus**
Form (m)	Key changes Phrases	$\boldsymbol{X}^1 \in \mathbb{R}^{8\times M\times 12}$	Music form
Reduced Lead Sheet (b)	Melody reduction Simplified chord	$\boldsymbol{X}^2 \in \mathbb{R}^{2\times \gamma M\times 128}$	Phrase similarity, phrase development & cadence
Lead Sheet (s)	Lead melody Chord	$\boldsymbol{X}^3 \in \mathbb{R}^{2\times \delta\gamma M\times 128}$	Melodic pattern, similarity & coherence
Accompaniment (s)	Accompaniment	$\boldsymbol{X}^4 \in \mathbb{R}^{2\times \delta\gamma M\times 128}$	Acc. pattern, similarity & coherence, Mel-acc relations

Ref: Wang et al, "Whole-song hierarchical generation of symbolic music using cascaded diffusion models," ICLR 2024

Whole-song Hierarchical Generation of Symbolic Music

- Stage 1: **Form**

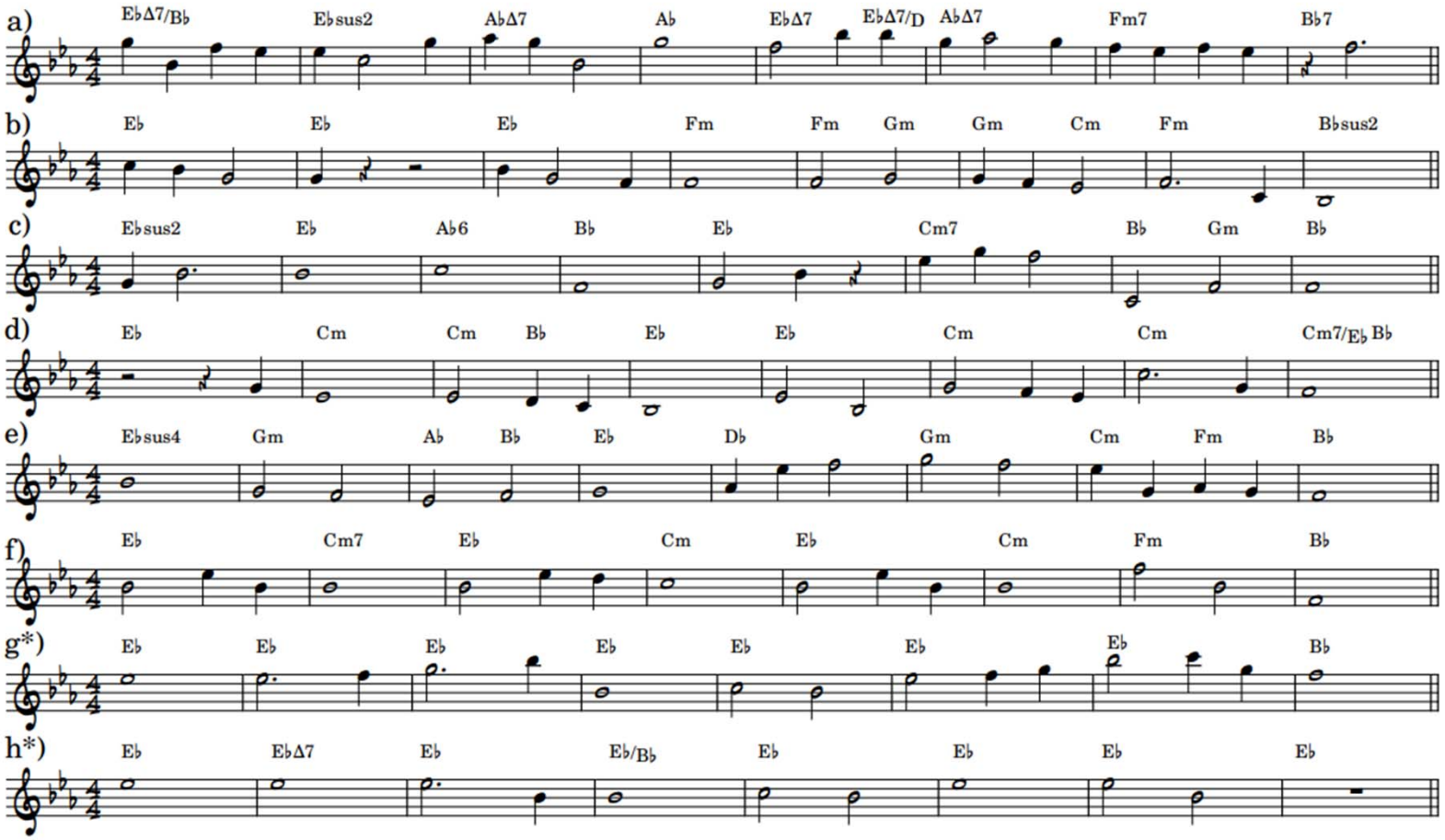
```
(i8)(A8B16A8B16)(b6)(B14)(o2)
(i12)(A4A4B12)(b4b4)(A4A4B12)(b4b4(B16)o4o1)
(i4)(A4A4B4X5)(b4)(A4b5B4X4X5)(o2)
(i4)(A8B9A8B9X18)(o2o1)
(i8)(A4A4B4B4)(b7)(A4A4B4B4B4B4)(o4)
(i8)(A4A4B4B4)(b7)(A4A4B4B4B4B4)(o4)
(i8)(A8B8X8X8)(b4b4)(A8B8X8X8X4)(o6o1)
(i4)(A4A4B9)(b3)(A4B10B9X1)(o2)
(i4)(A4A4B9)(b3)(A4B10B9X1)(o2)
(i12)(A16B16)(b4b4b4)(A16B16B16)(o10o1)
```



Ref: Wang et al, "Whole-song hierarchical generation of symbolic music using cascaded diffusion models," ICLR 2024

Whole-song Hierarchical Generation of Symbolic Music

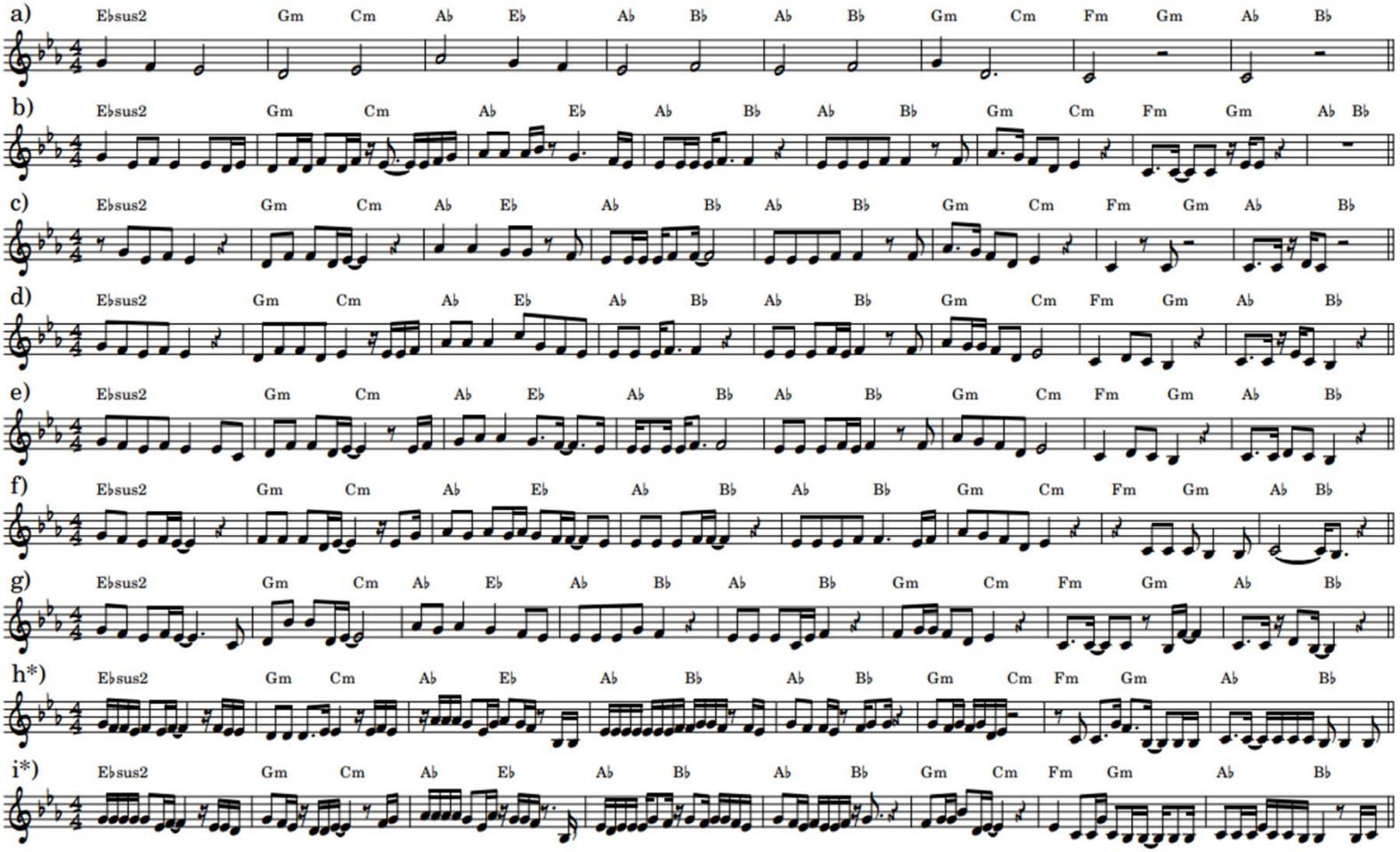
- Stage 2: **Reduced lead sheet**



Ref: Wang et al, "Whole-song hierarchical generation of symbolic music using cascaded diffusion models," ICLR 2024

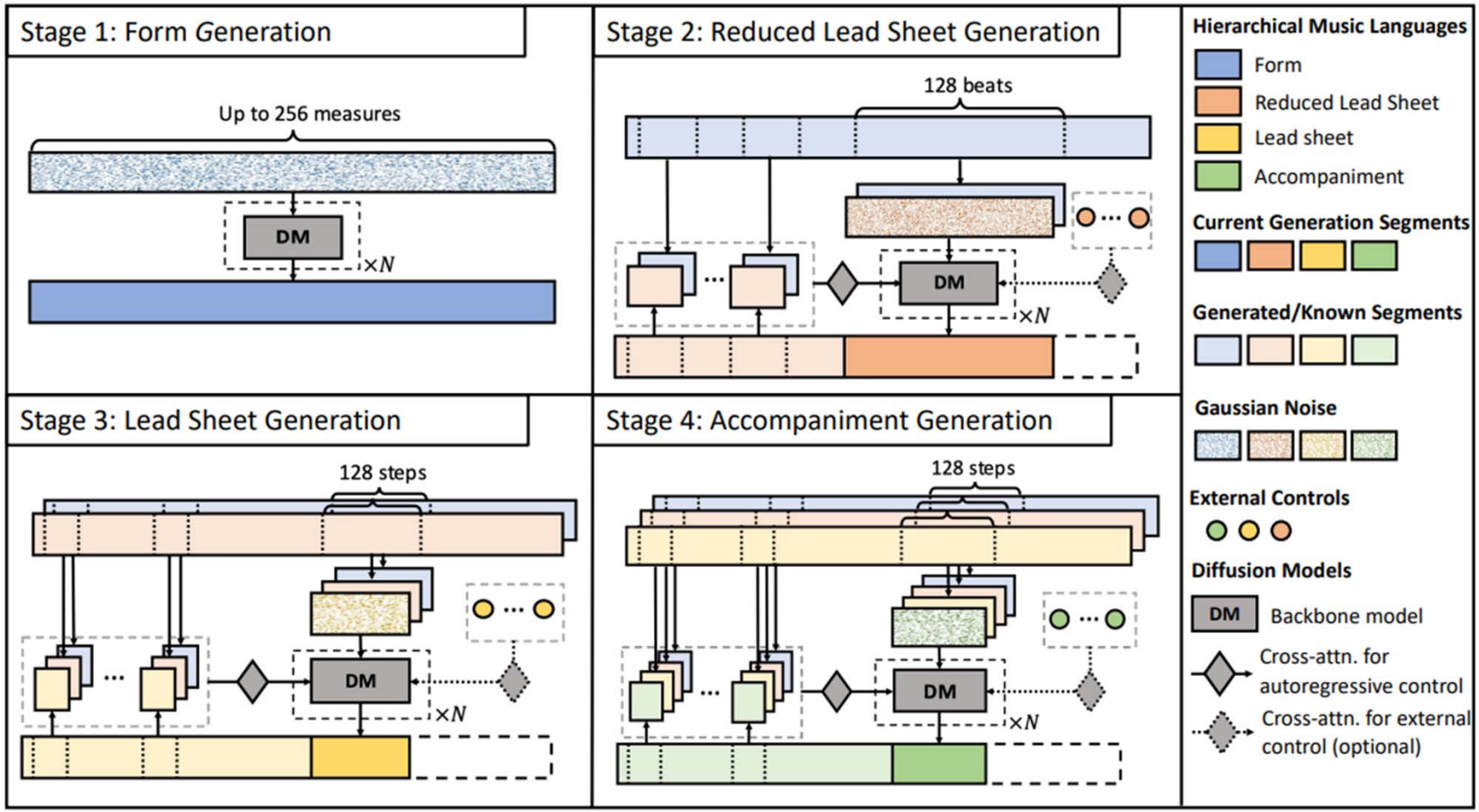
Whole-song Hierarchical Generation of Symbolic Music

- Stage 3:
  Lead sheet



Ref: Wang et al, “Whole-song hierarchical generation of symbolic music using cascaded diffusion models,” ICLR 2024

Whole-song Hierarchical Generation of Symbolic Music



Ref: Wang et al, "Whole-song hierarchical generation of symbolic music using cascaded diffusion models," ICLR 2024