AI Labs.tw
台灣人工智慧實驗室

Automatic Music Composition with Transformers

Yi-Hsuan Yang Ph.D. [1,2]

[1] Taiwan AI Labs
[2] Research Center for IT Innovation, Academia Sinica

http://mac.citi.sinica.edu.tw/~yang/
yhyang@ailabs.tw

Outline

- Image vs text-based approach for music generation
- Advances in Transformer-based music generation

Type of Music Generation Research

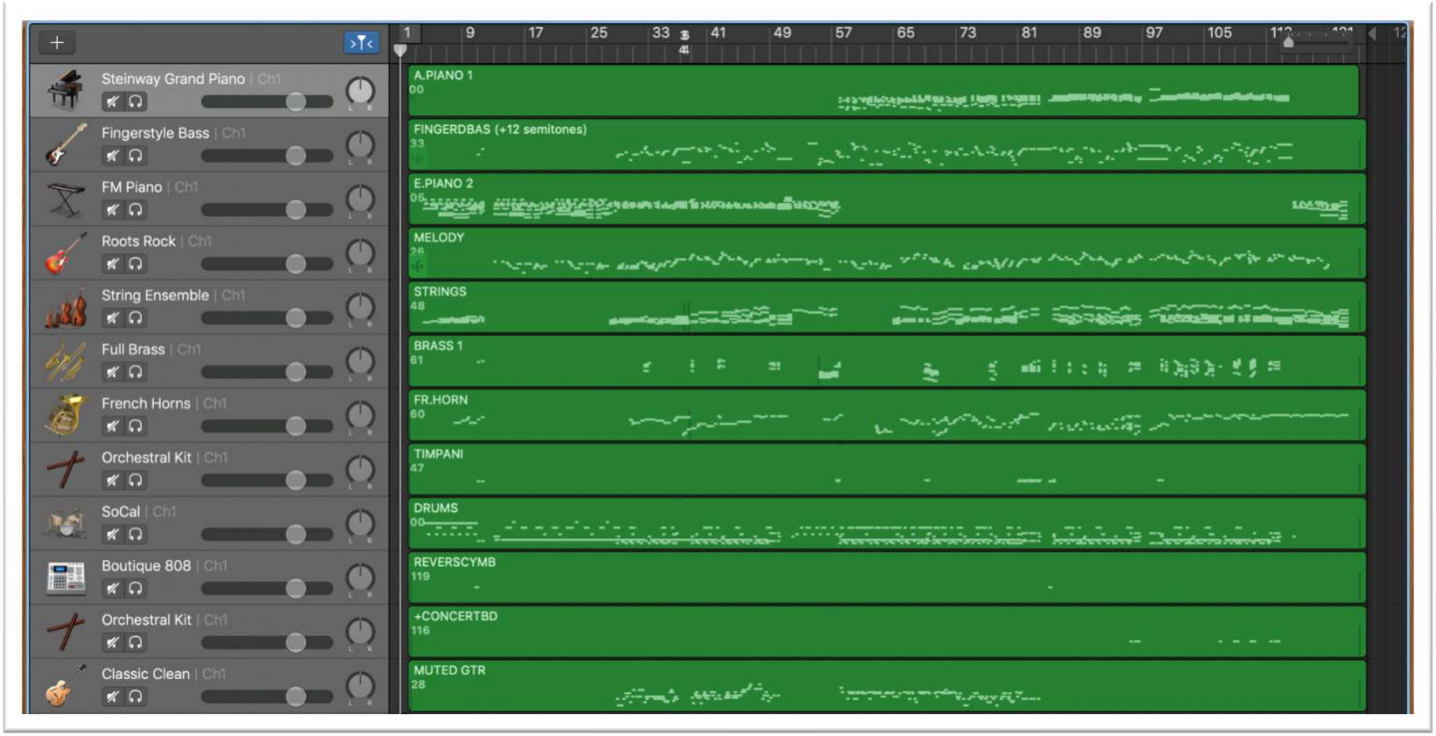
1. **Composition** (symbolic): musical scores
2. **Performance** (symbolic): musical scores + expression
    a. given score, generate performance (e.g., VirtuosoNet [jeong19ismir])
    b. generate performances *from scratch* (i.e., generate scores and their expression at the same time)
3. **Audio**

We will mainly talk about **1** and **2b**

"VirtuosoNet: A hierarchical RNN-based system for modeling expressive piano performance", ISMIR 2019

Two Main Approaches

(1) Consider music as **image**



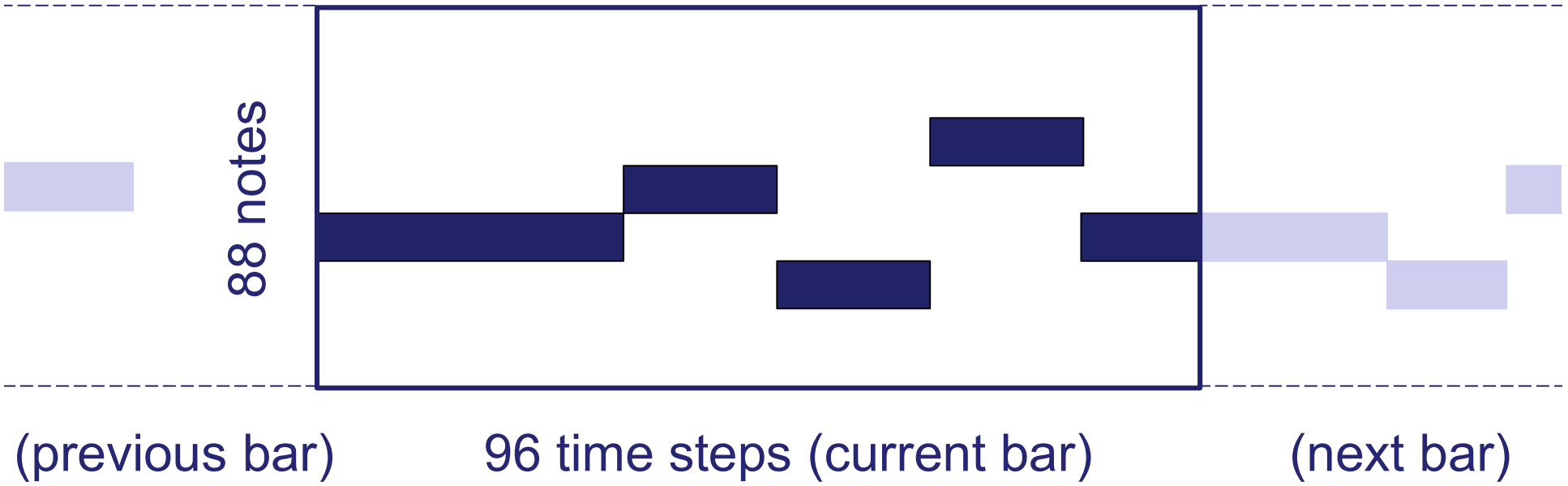
(image from the internet)

Image Generation

(result of StyleGAN 2 [karras20cvpr])

Piano-Roll

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

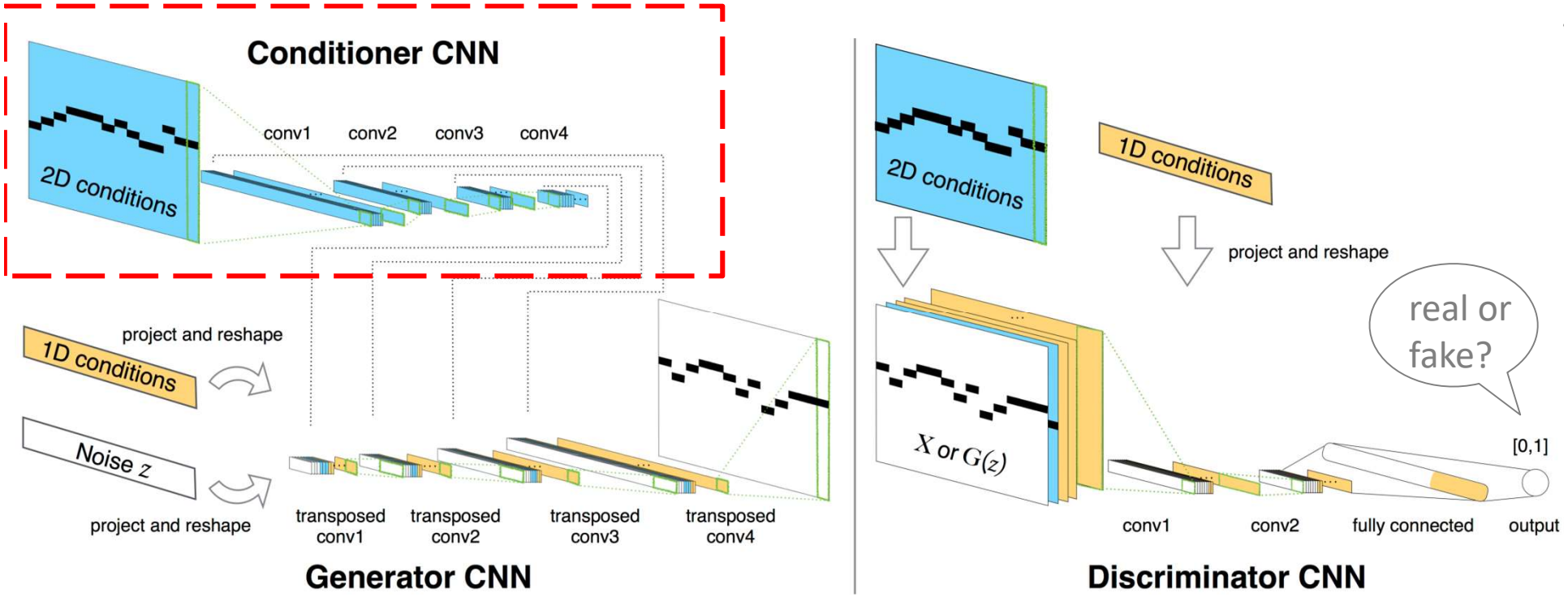


"Pypianoroll: Open source Python package for handling multitrack pianorolls", ISMIR-LBD 2018

MidiNet [yang17ismir]

Convolutional generative adversarial network (GAN)

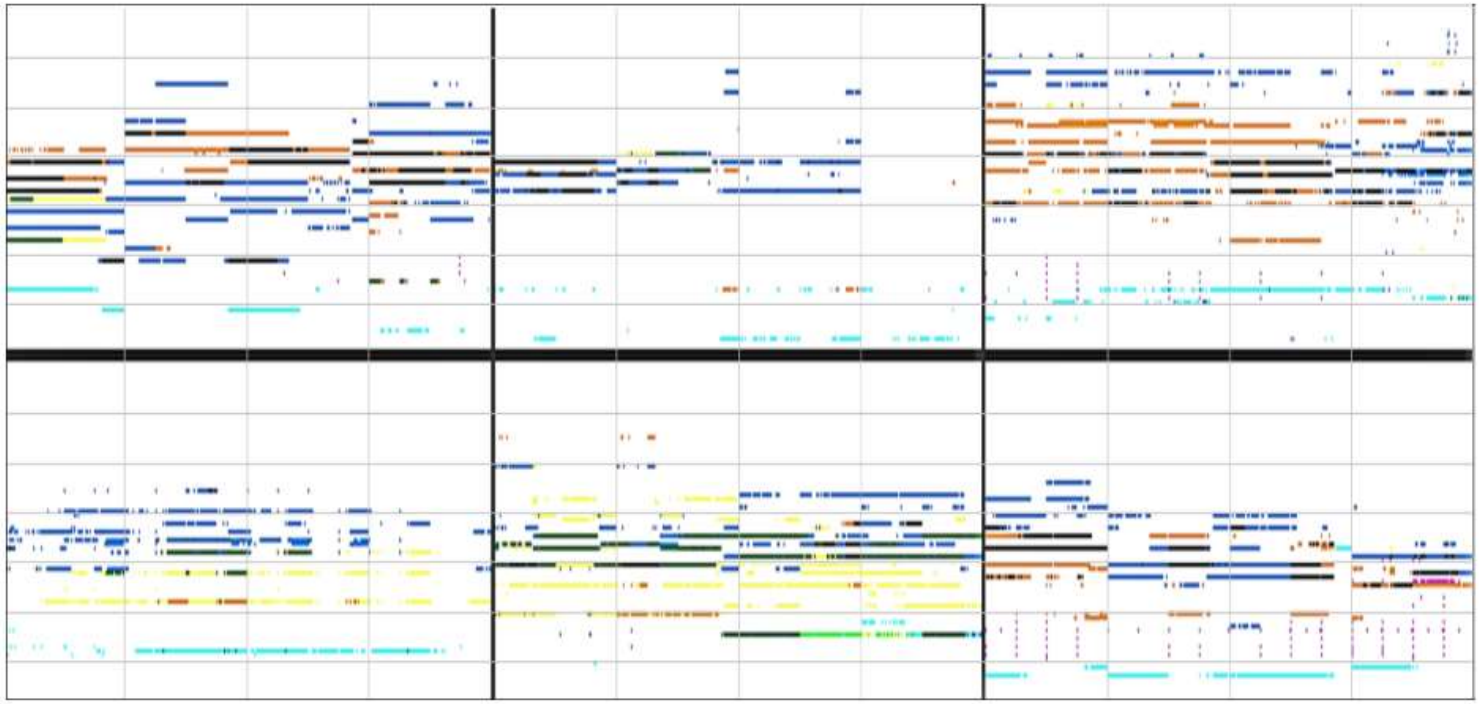
- upsample a random vector into a piano-roll like matrix



"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

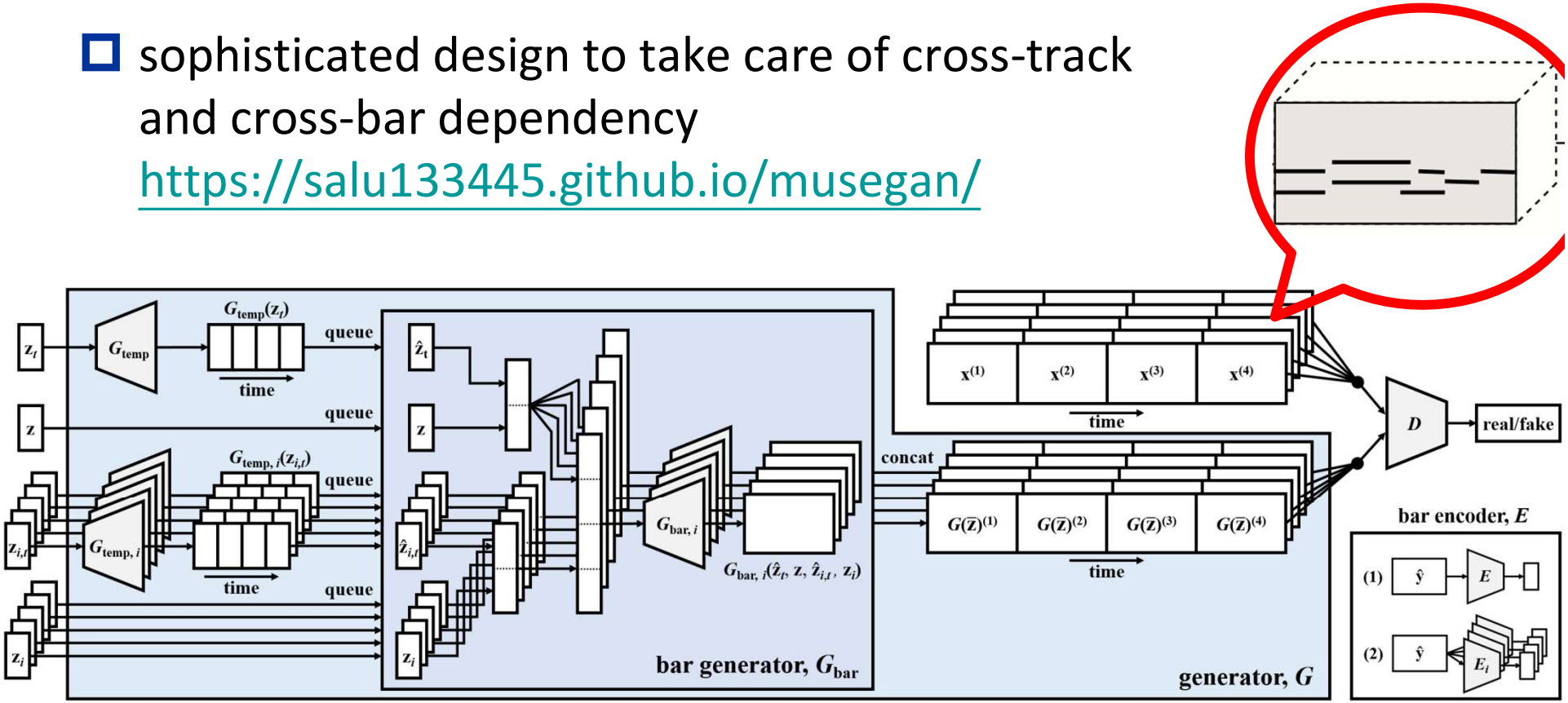


(different colors represent different tracks in this visualization)

MuseGAN [dong18aaai]

Again, convolutional GAN

- sophisticated design to take care of cross-track and cross-bar dependency
  https://salu133445.github.io/musegan/



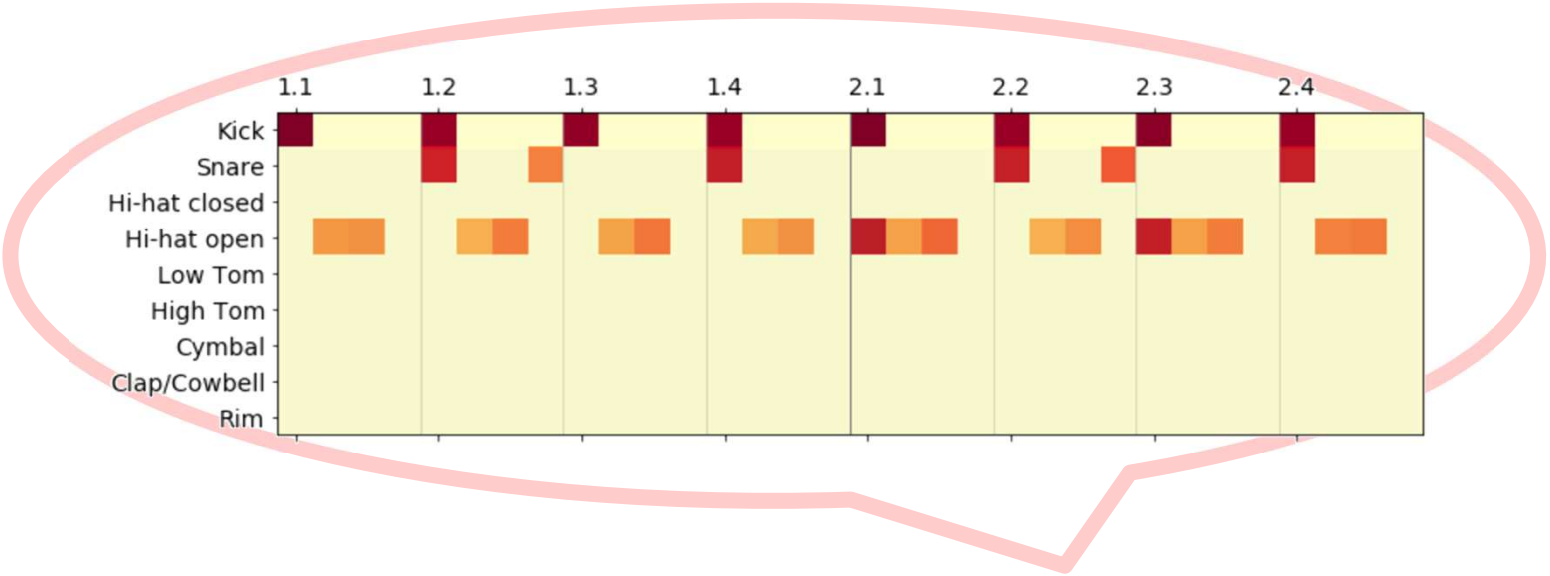
"MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

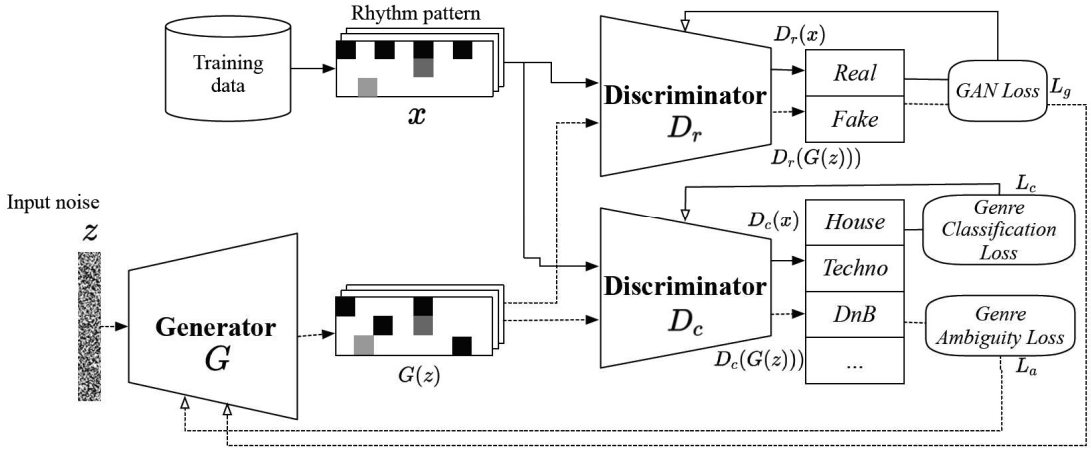
Two Main Approaches

(1) Consider music as **image**

- my personal experience is that the image-based approach works better for generating "**patterns**", and "spectrograms" (i.e. sounds), rather than "**long sequences**"

RhythmCAN [tokui20arxiv]

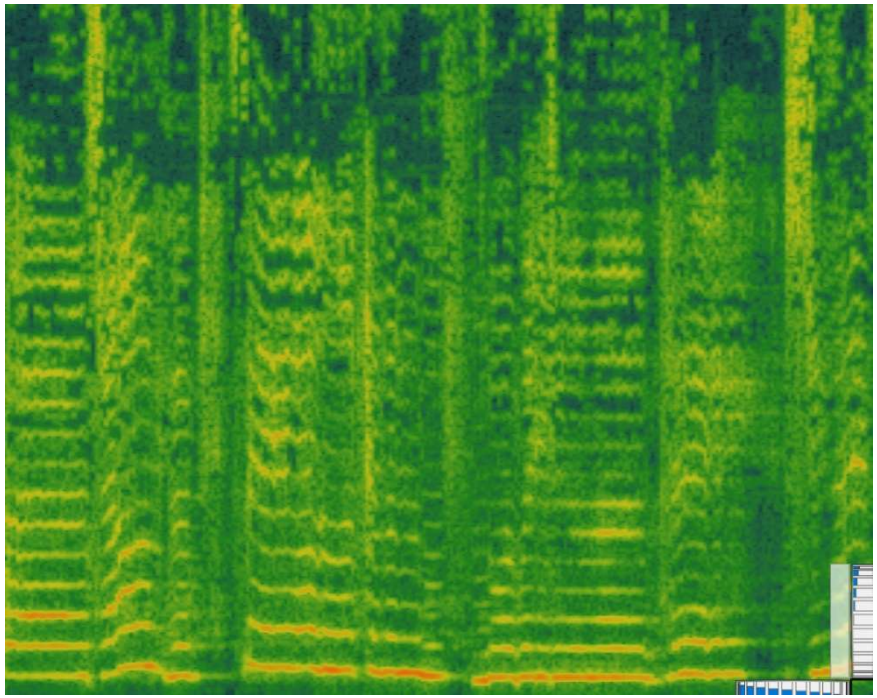




"Can GAN originate new electronic dance music genres?—Generating novel rhythm patterns using GAN with Genre Ambiguity Loss", arxiv 2020

UNAGAN [liu20interspeech]

Unconditional generation of spectrograms
https://github.com/ciaua/unagan



singing

speech

violin

piano

"Unconditional audio generation with generative adversarial networks and cycle regularization", INTERSPEECH 2020

More GANs: ISMIR 2019 Tutorial

https://salu133445.github.io/ismir2019tutorial/

Generating Music with GANs

Hao-Wen Dong, Yi-Hsuan Yang

UC San Diego & Academia Sinica

Taiwan AI Labs & Academia Sinica

Welcome to the website for the tutorial "Generating Music with GANs—An Overview and Case Studies" at ISMIR 2019 (November 4th at Delft, The Netherlands).

Slides

- ismir2019-tutorial-slides.pdf

Colab notebooks

- GAN for images (MNIST Database)
- GAN for pianorolls (Lakh Pianoroll Dataset)

Related projects

- MuseGAN
- LeadSheetGAN

Resources

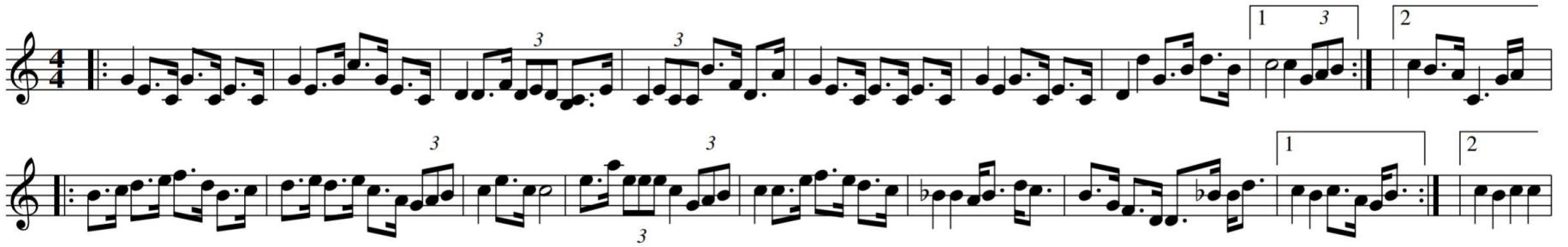
- Pypianoroll: a Python package for handling multitrack pianorolls
- Lakh Pianoroll Dataset (LPD): a collection of 174,154 multitrack pianorolls derived from the Lakh MIDI Dataset (LMD)

Two Main Approaches to Automatic Music Composition

(2) Consider music as **text**

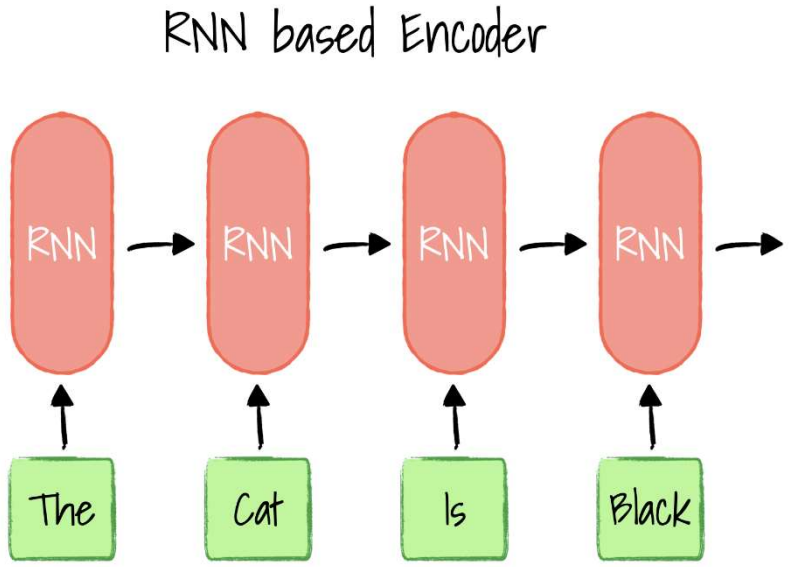
- use “tokens” to represents “events” in music
- becomes a “natural language generation” (NLG) problem
  - e.g., Folk RNN [sturm15ismir-lbd]

```
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 |
```



“Folk music style modelling by recurrent neural networks with long short term memory units”, ISMIR-LBD 2015

RNNs

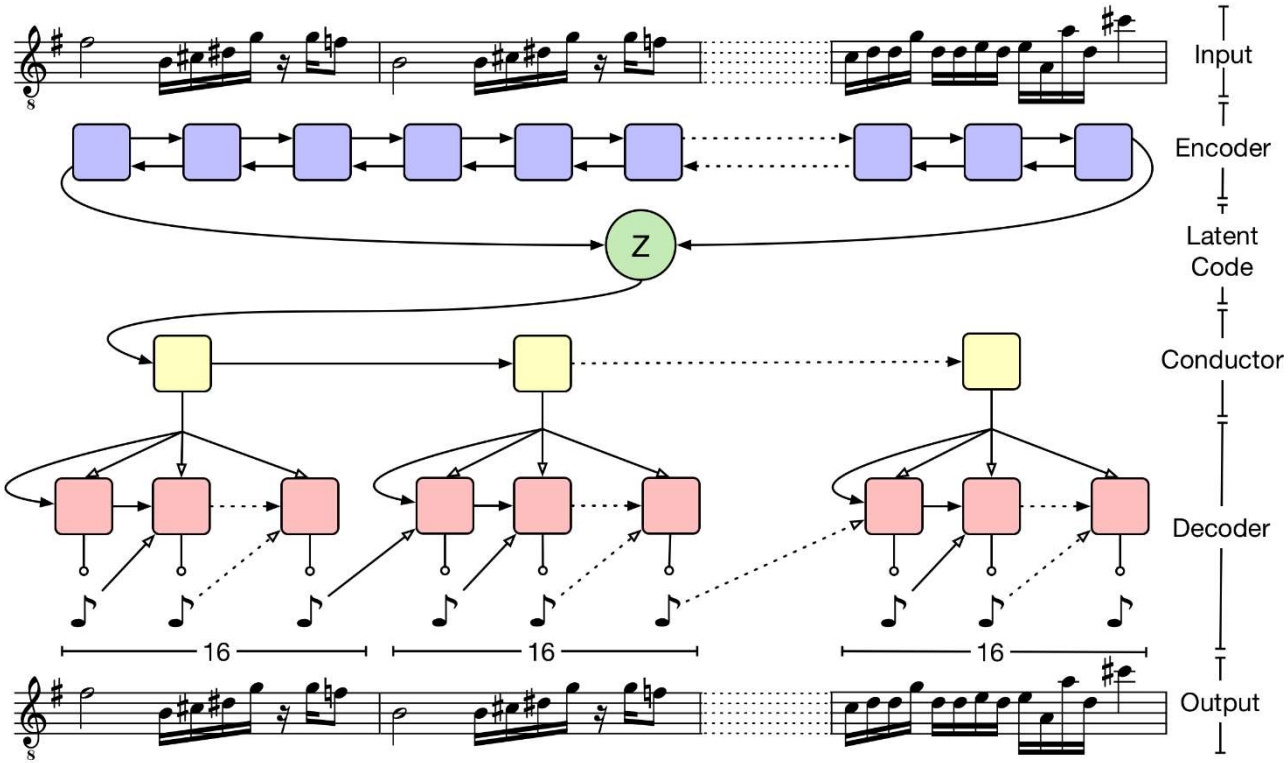


- Text tokens: “The”, “Cat”, “Is”, “Black”
- Music tokens: “C4”, “E4”, “G4”, …

Music VAE [roberts18icml]

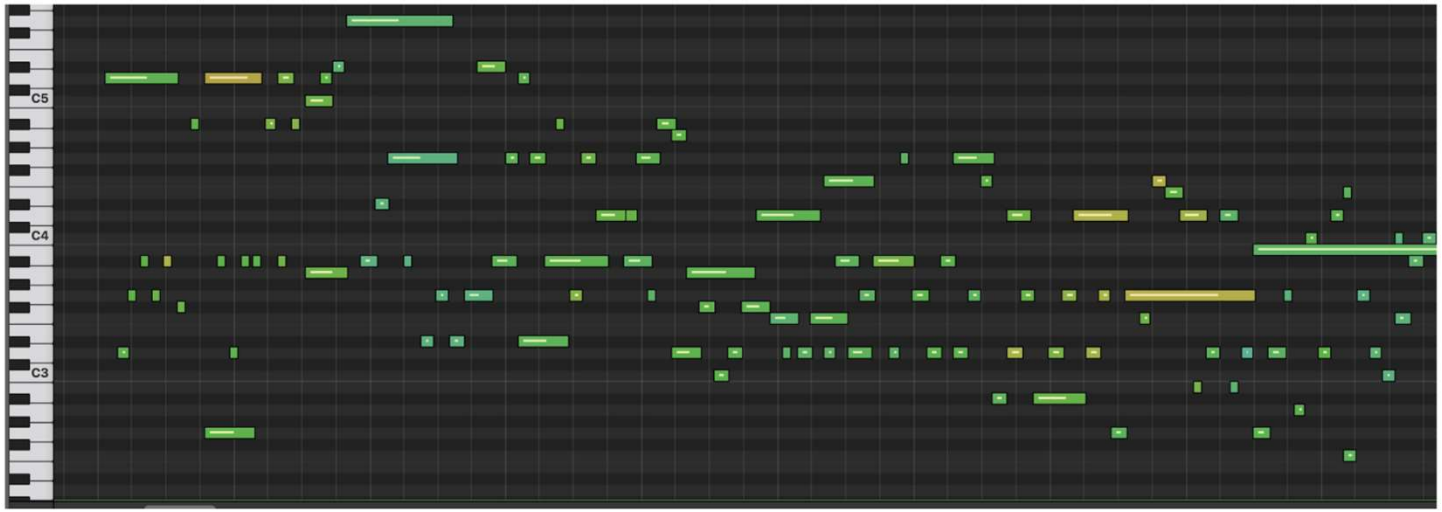
Generate melodies

- generate one note at a time



"A hierarchical latent vector model for learning long-term structure in music", ICML 2018

PerformanceRNN [simon17]

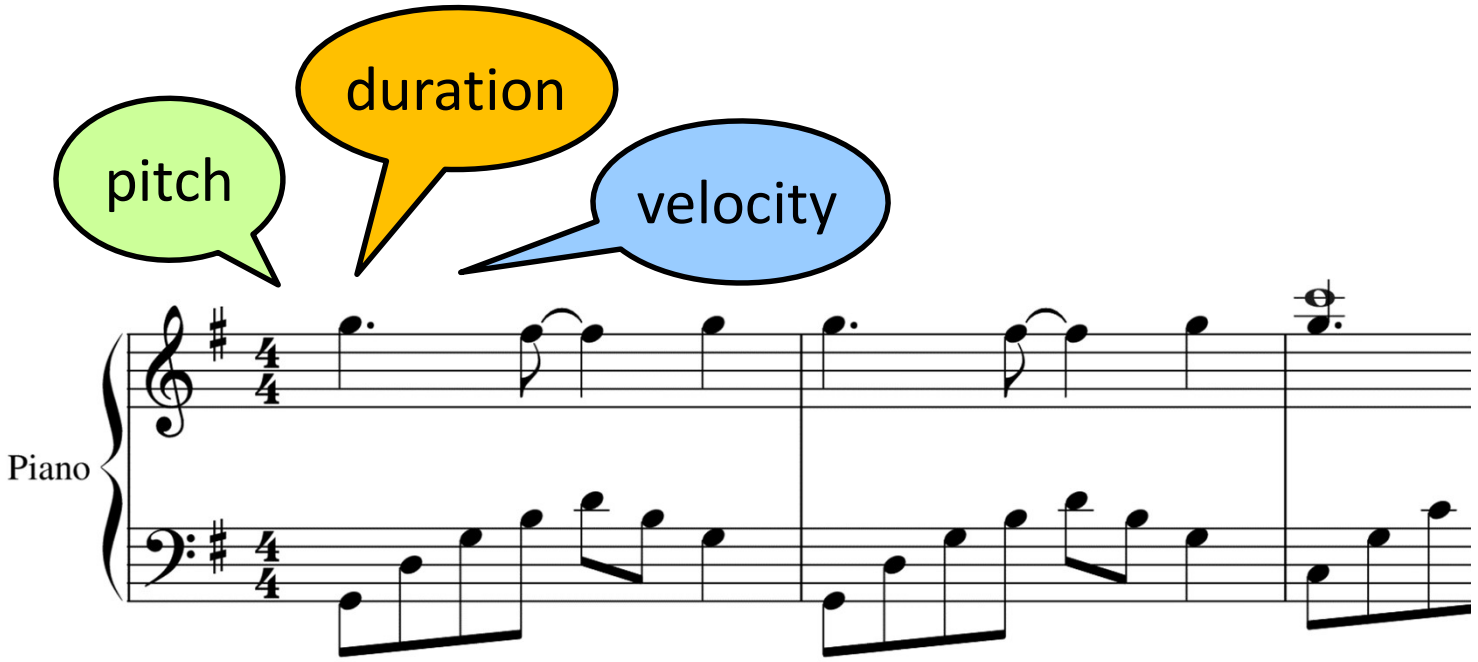


- Generate piano performances with **expressive timing + dynamics**
  - 128 **note-on** events + 128 **note-off** events: one for each of the 128 MIDI pitches; start or release a note
  - 100 **time-shift** events in increments of 10 ms up to 1 second; move forward in time to the next note event
  - 32 **velocity** events, corresponding to MIDI velocities quantized into 32 bins

"Performance RNN: Generating music with expressive timing and dynamics", 2017

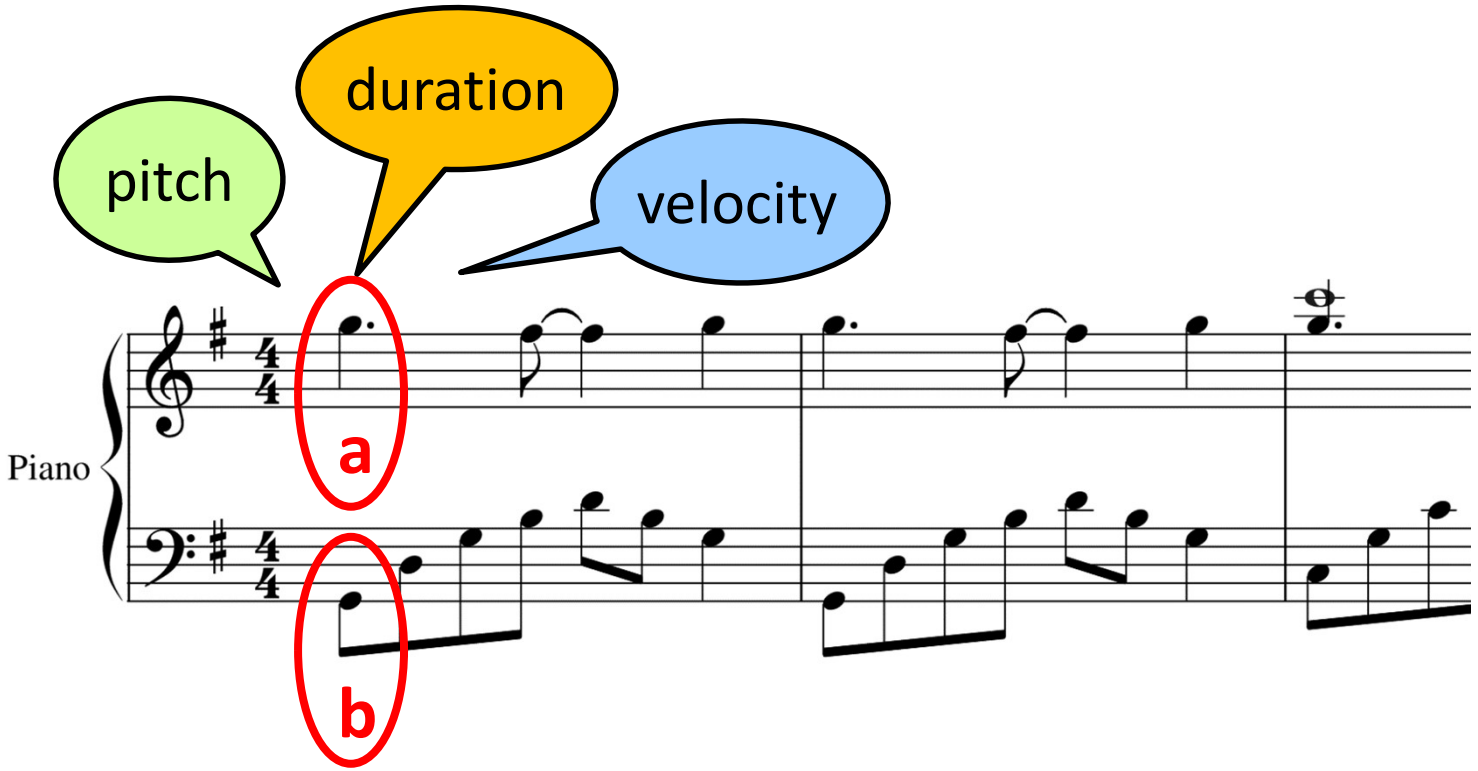
MIDI-like Representation

- Represent a music piece as a *token sequence*
- Describe a musical note by three tokens



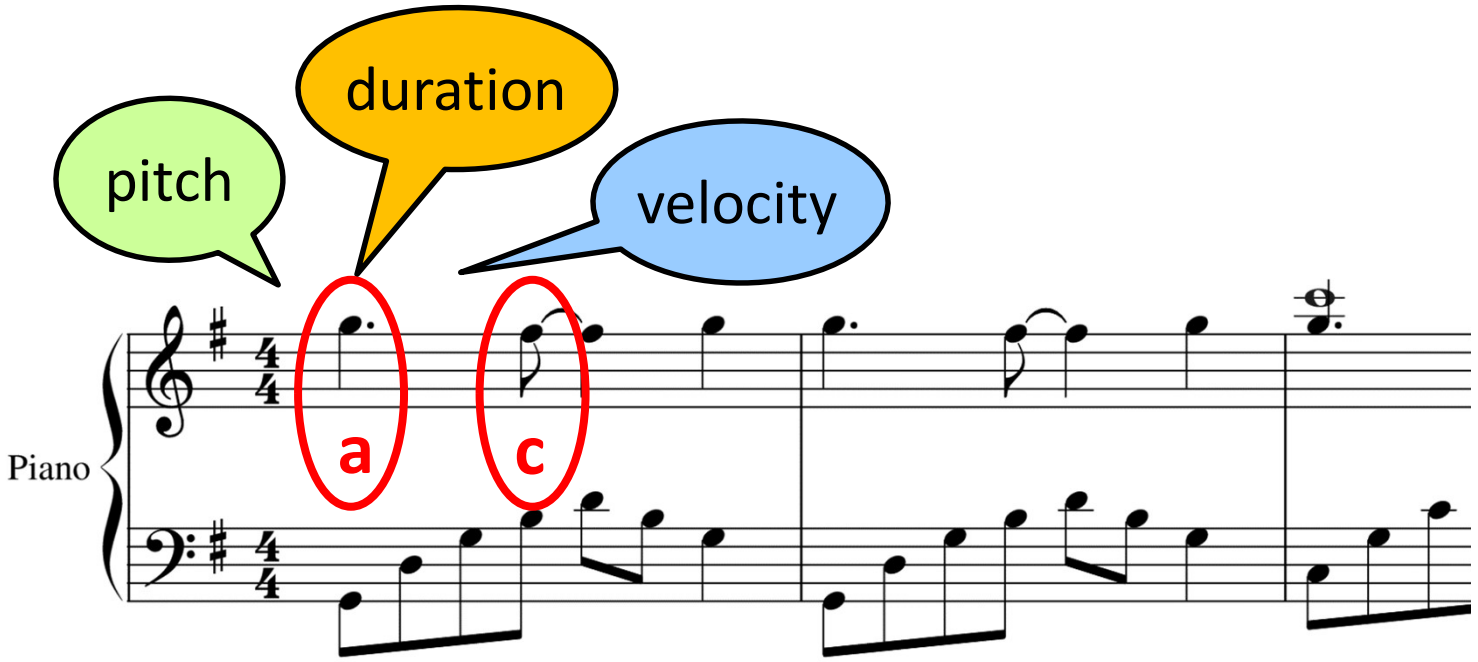
MIDI-like Representation

- And, one additional token to mark “time-shift” (i.e., ΔT; interval time)
  - for people can play multiple notes at the same time



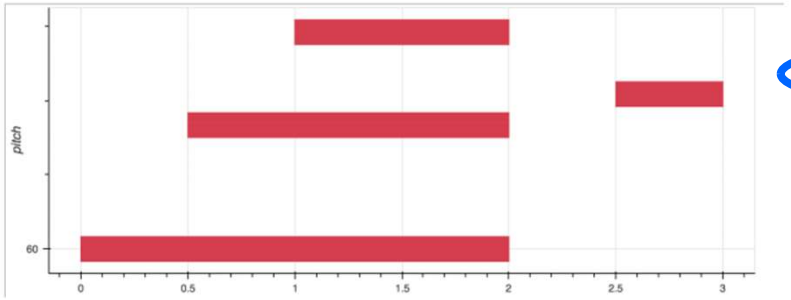
MIDI-like Representation

- And, one additional token to mark “time-shift” (i.e., ΔT; interval time)
  - for people can play multiple notes at the same time



ΔT =50ms

An Example




```
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>
```

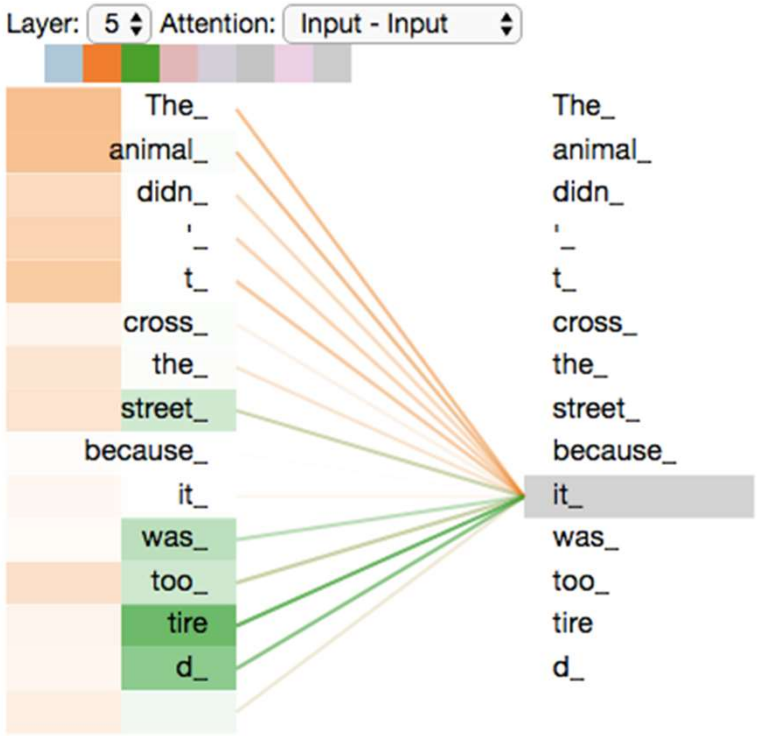
- Generate piano performances with **expressive timing + dynamics**
  - 128 **note-on** events + 128 **note-off** events: one for each of the 128 MIDI pitches; start or release a note
  - 100 **time-shift** events in increments of 10 ms up to 1 second; move forward in time to the next note event
  - 32 **velocity** events, corresponding to MIDI velocities quantized into 32 bins

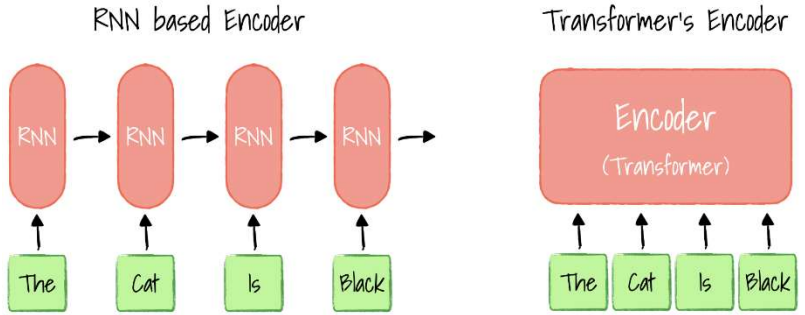
Key to the Text-based Approach

- **Neural sequence model**
  - RNNs, hierarchical RNN, Transformers
- **Token representation of music**
  - MIDI-like, …

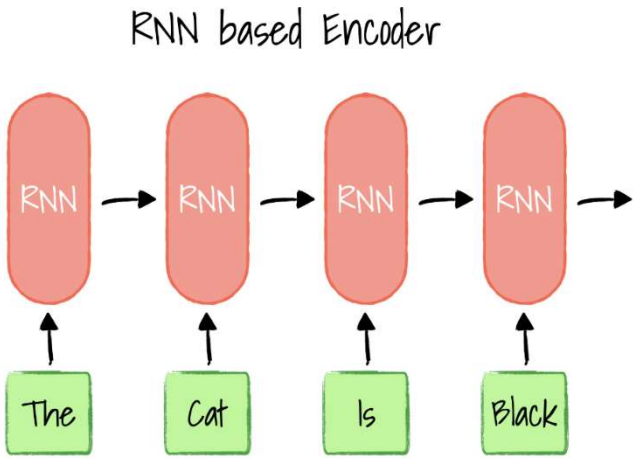
Transformers

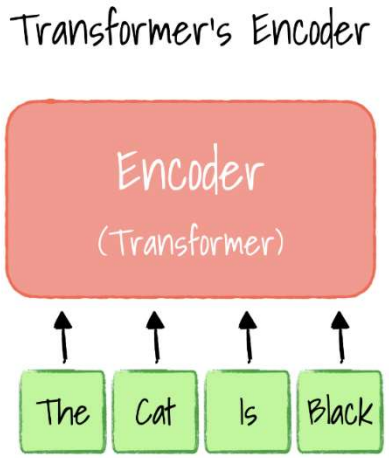
- Attention mechanism
  - maintain a latent vector for ***every*** token in the history
  - compute the correlation between the input with all the latents in the history
- In contrast, there is only one latent vector for the entire sequence in RNNs





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

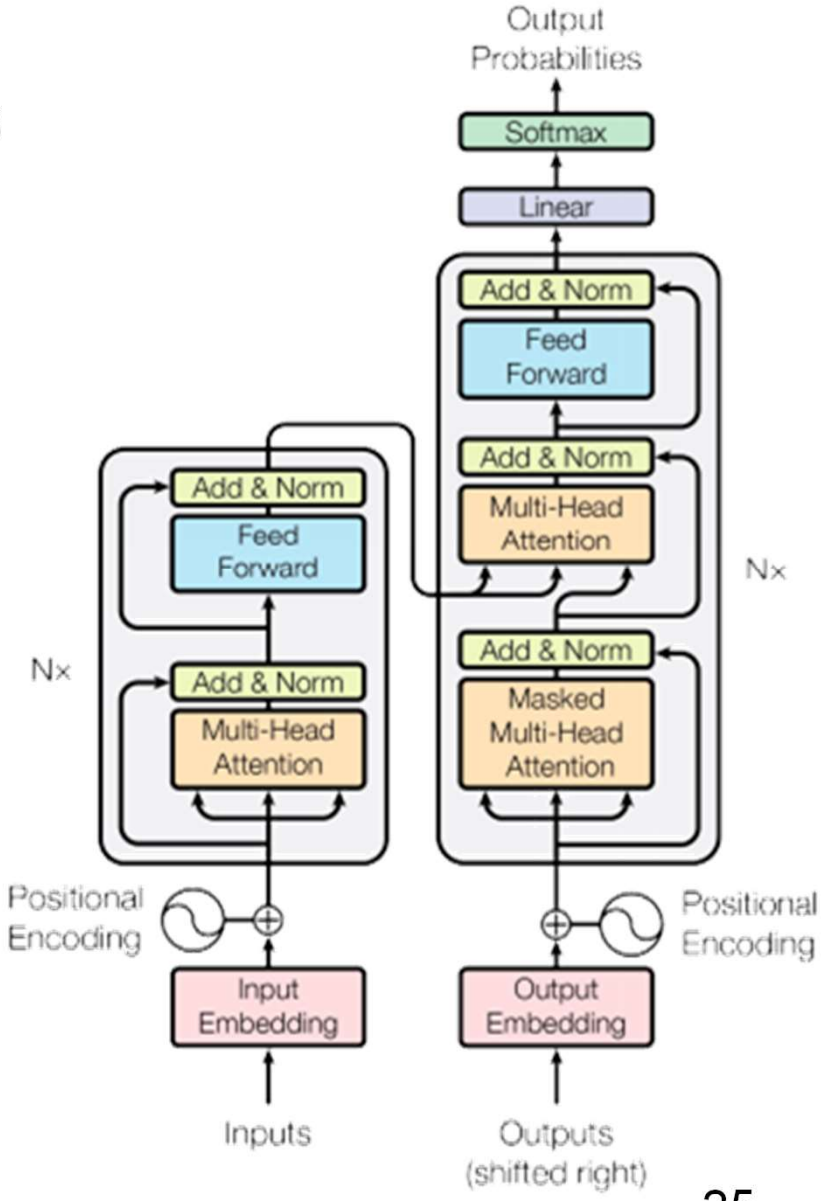
Transformers

There are encoders & decoders

We mainly talk about

only the `**decoder**` here

- use self-attention

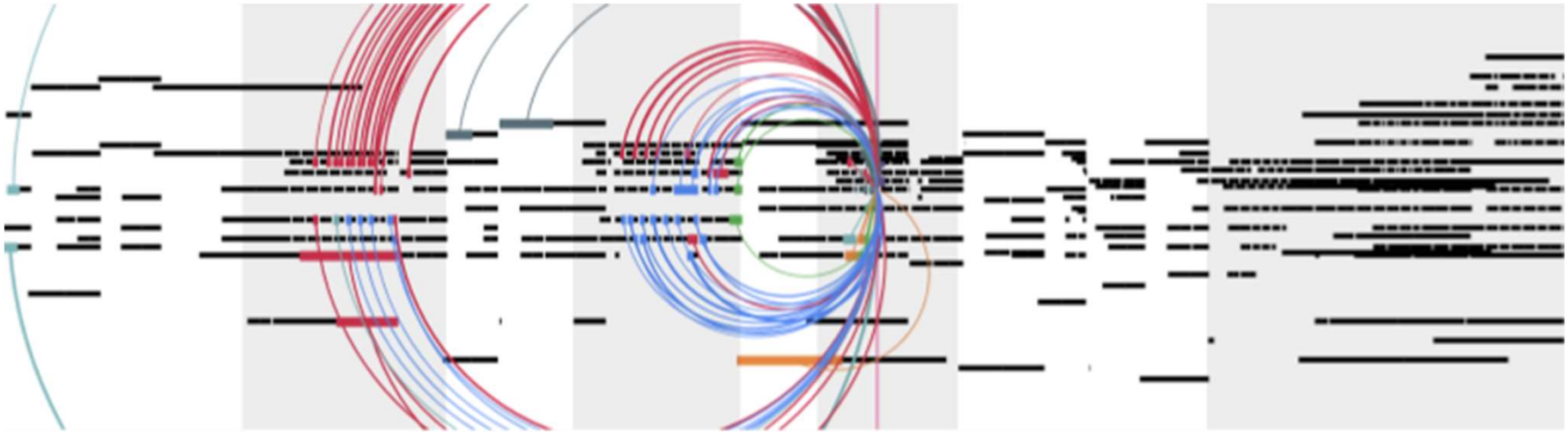


Music Transformer [huang19iclr]

Generate full piano performances

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

- again, generate one note at a time



"Music Transformer: Generating music with long-term structure", ICLR 2019

MuseNet [payne19]

Generate multi-track MIDIs

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

- *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
```

Sample encoding which combines pitch, volume, and instrument.

- another multitrack Transformer: LakhNES [donahue19ismir]

"MuseNet," OpenAI blog, 2019
"LakhNES: Improving multi-instrumental music generation with cross-domain pre-training", ISMIR 2019

RNNs vs. Transformers

- **RNNs** work well when the sequence is *short*
  - melodies (e.g., folk RNN, BebopNet [hakimi20ismir])
  - 4-bar piano passages (e.g., MusicVAE)
  - many ISMIR'20 papers on music generation still use RNNs

- **Transformers** work better for *longer sequences*
  - at the expense of computational power
    - an RNN-based model typically contains **3** layers or so
    - a Transformer-based model may use up to **72** layers … (e.g., MuseNet)
    - our own experience is that a **12**-layer Transformer (with about 41M learnable parameters) works okay

"BebopNet: Deep Neural Models for Personalized Jazz Improvisations", ISMIR 2020

How to Apply Transformers to My Research?

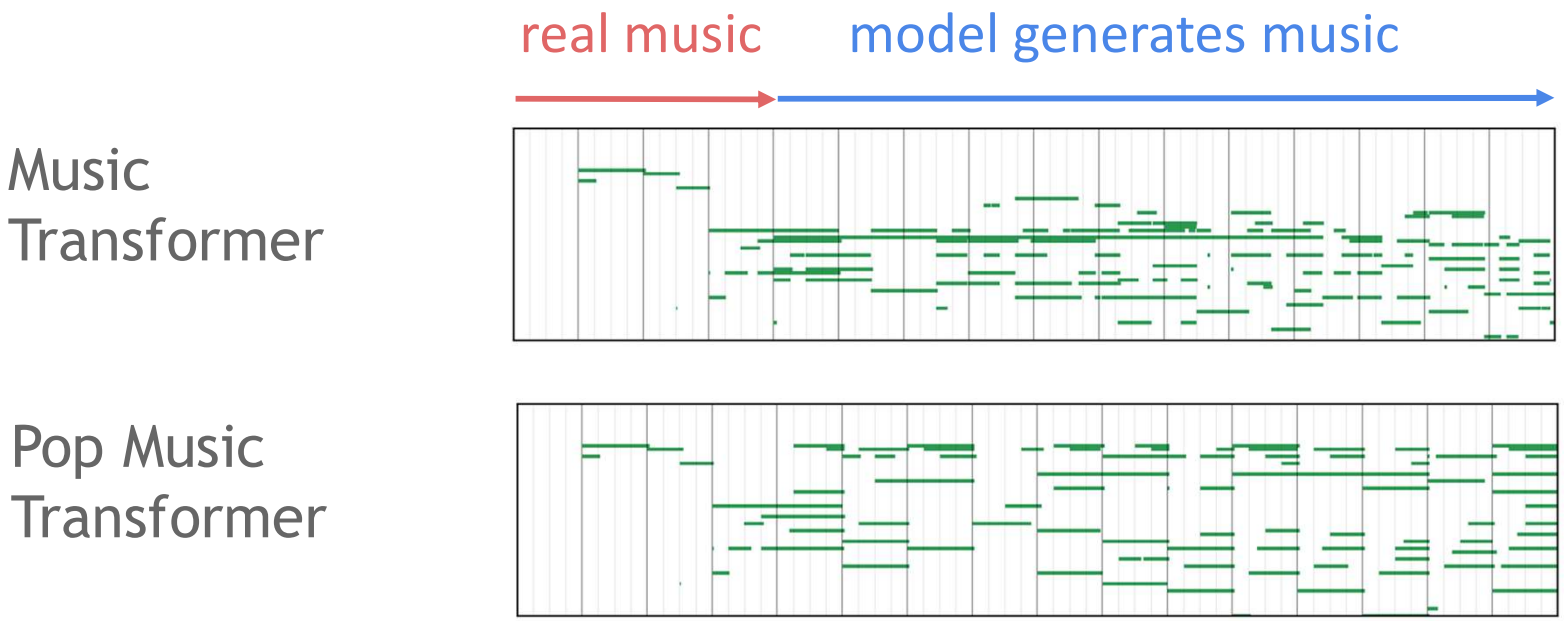
- **Common concerns/sentiments**
  - Transformers are too powerful: All you need is data
    - Reviewers may complain that “the work is not novel”
  - Transformers are too expensive to train
- **My suggestion**
  - pinpoint the weakness of your model and try to address it
  - bring domain knowledge of music
  - work on different generation tasks, or different genres https://github.com/affige/genmusic_demo_list
  - Transformers are not that expensive; see below

Outline

- Image vs text-based approach for music generation
- **Advances in Transformer-based music generation**
  - Pop Music Transformer, *ACM MM* 2020
  - Compound Word Transformer, *AAAI* 2021

Pop Music Transformer [huang20mm]

- Adapt the token representation for **pop music**, which features **steady beats**
- Use Transformer-XL instead of the vanilla Transformer



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

Pop Music Transformer [huang20mm]

We have extended it to generate guitar tabs [chen20ismir]

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/ai-guitar-tab-generation-202003/s-KHozfW0PTv5

piano

piano + drum

guitar

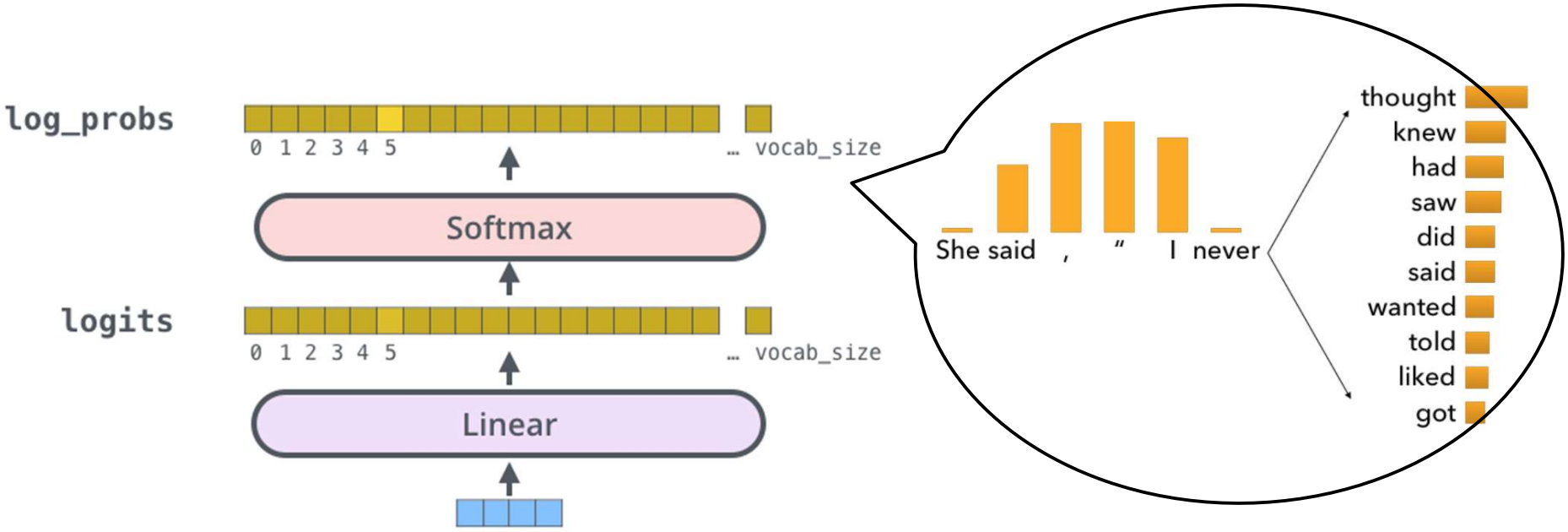
“Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions”, ACM Multimedia 2020

“Automatic composition of guitar tabs by Transformers and groove modeling”, ISMIR 2020

The Problem: “Time-Shift” does not work well for Pop Music

There is a “**sampling**” mechanism at Transformers’ output

- **errors in Time-Shift accumulate** → the generated music lacks stable rhythmic structure
- the model does not have built-in notion of beats/downbeats

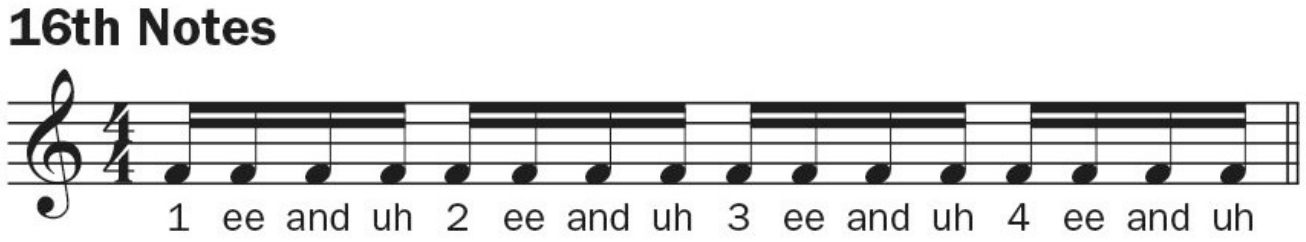


(images from the internet)

Pop Music Transformer [huang20mm]

REMI: An alternative ***token representation*** **of music**

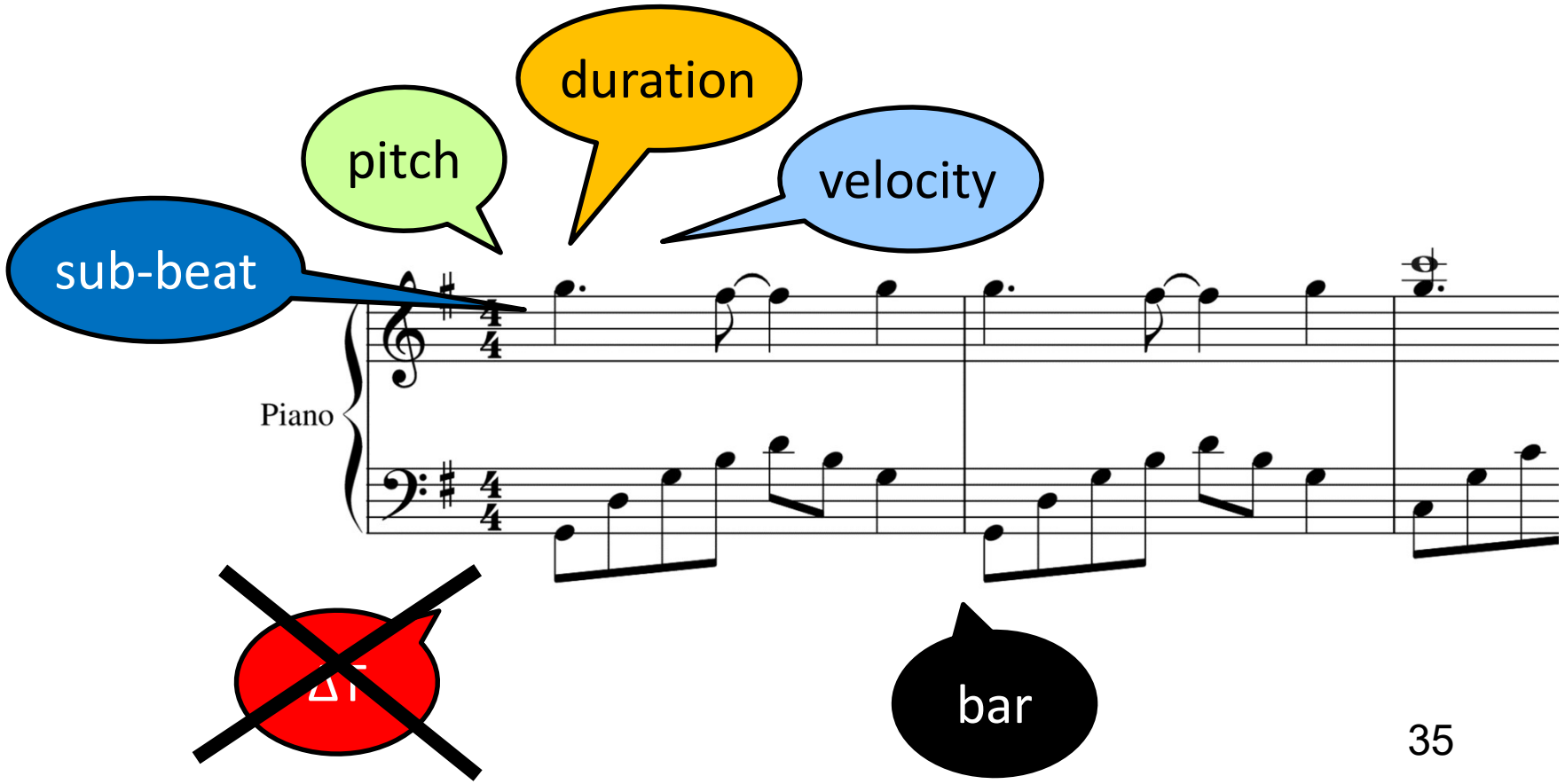
- mark the ***bar*** lines
- divide a bar into 16 discrete ***sub-beats***
- i.e., **symbolic timing** + **explicit time grid**
  (instead of relying on relative absolute time intervals)
- how to get bar/sub-beat info?
  - scores: already there
  - transcribed performances: use beat/downbeat tracking



(images from the internet)

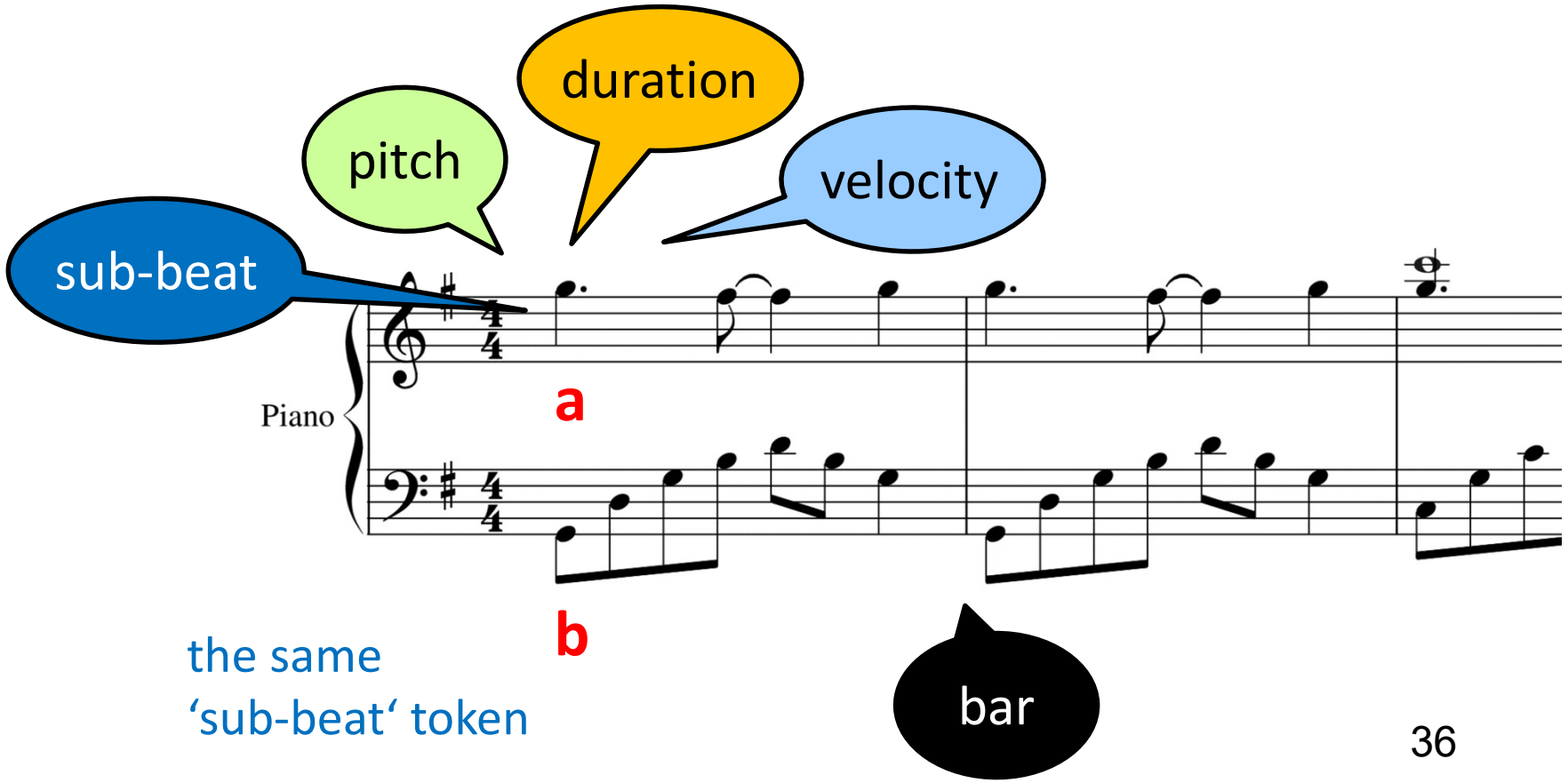
REMI

- Mark the bar lines
- Indicate the position of a note within a bar by sub-beat



REMI

- Mark the bar lines
- Indicate the position of a note within a bar by sub-beat



REMI

An alternative ***token representation*** of music

- Use `Note-Duration` instead of Note-Off
- Use `Sub-beat & Bar` instead of Time-Shift
- Add chord & tempo related tokens

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

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

Open Research

Open source code

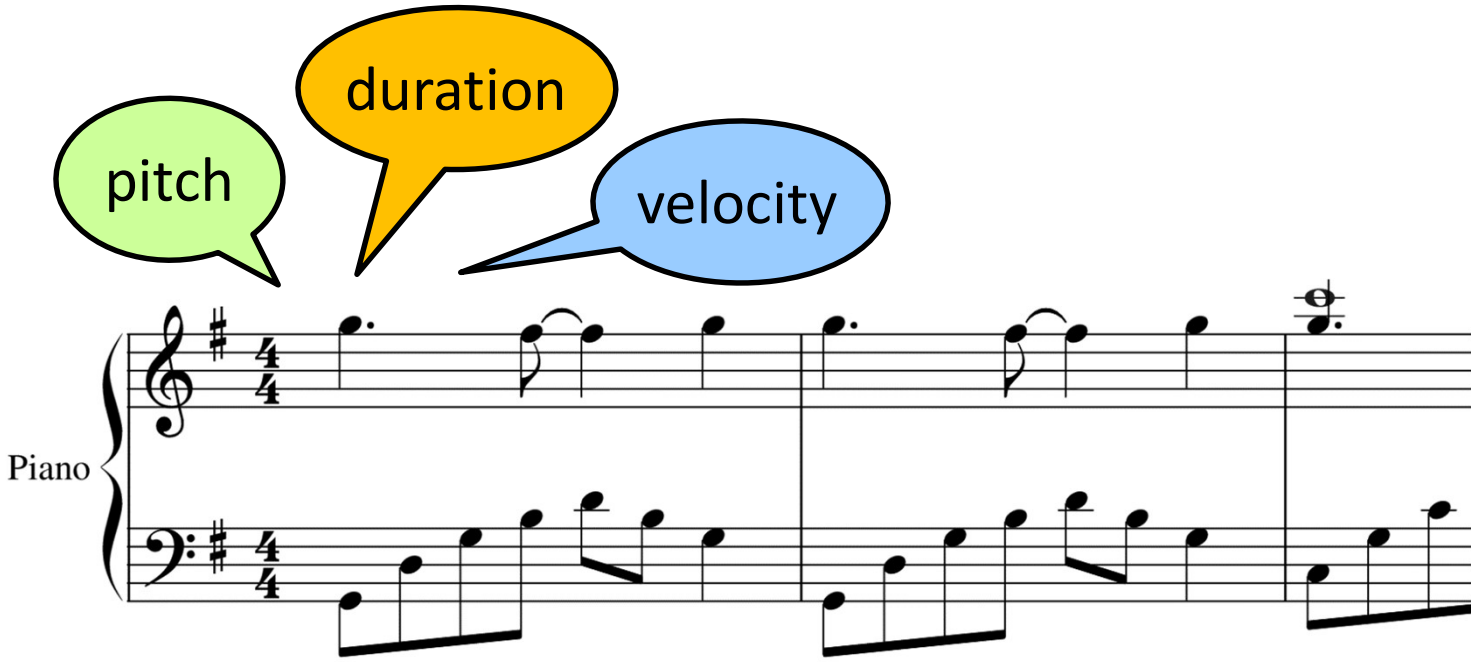
https://github.com/YatingMusic/remi (the music composing AI)

https://github.com/YatingMusic/miditoolkit (prepare MIDI data)

https://github.com/YatingMusic/ReaRender (convert MIDI to audio)

But…

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

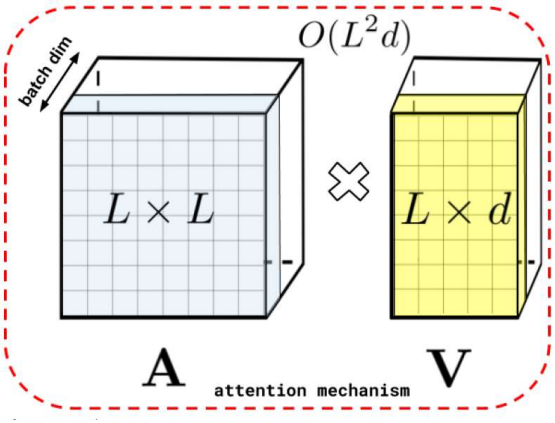


The Problem: Transformers are Expensive

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

latent representation of *t*-1

latent representation of *t*-*L*



(Figure from the Performer paper)

  - 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 memory space

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
  - 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]

	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

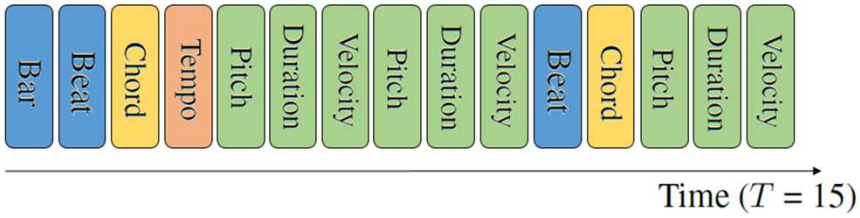
Shorter training time → easier to try different ideas

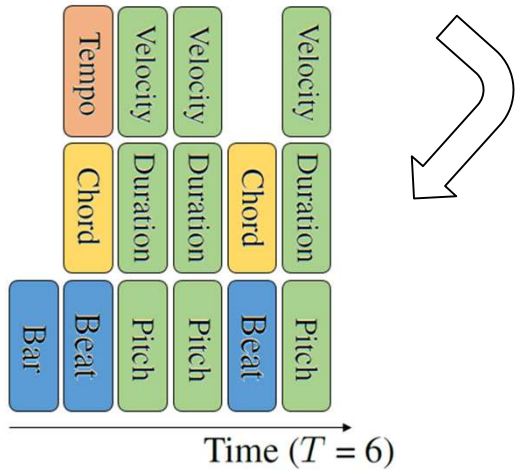
Shorter inference time → good for real-time applications

"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)



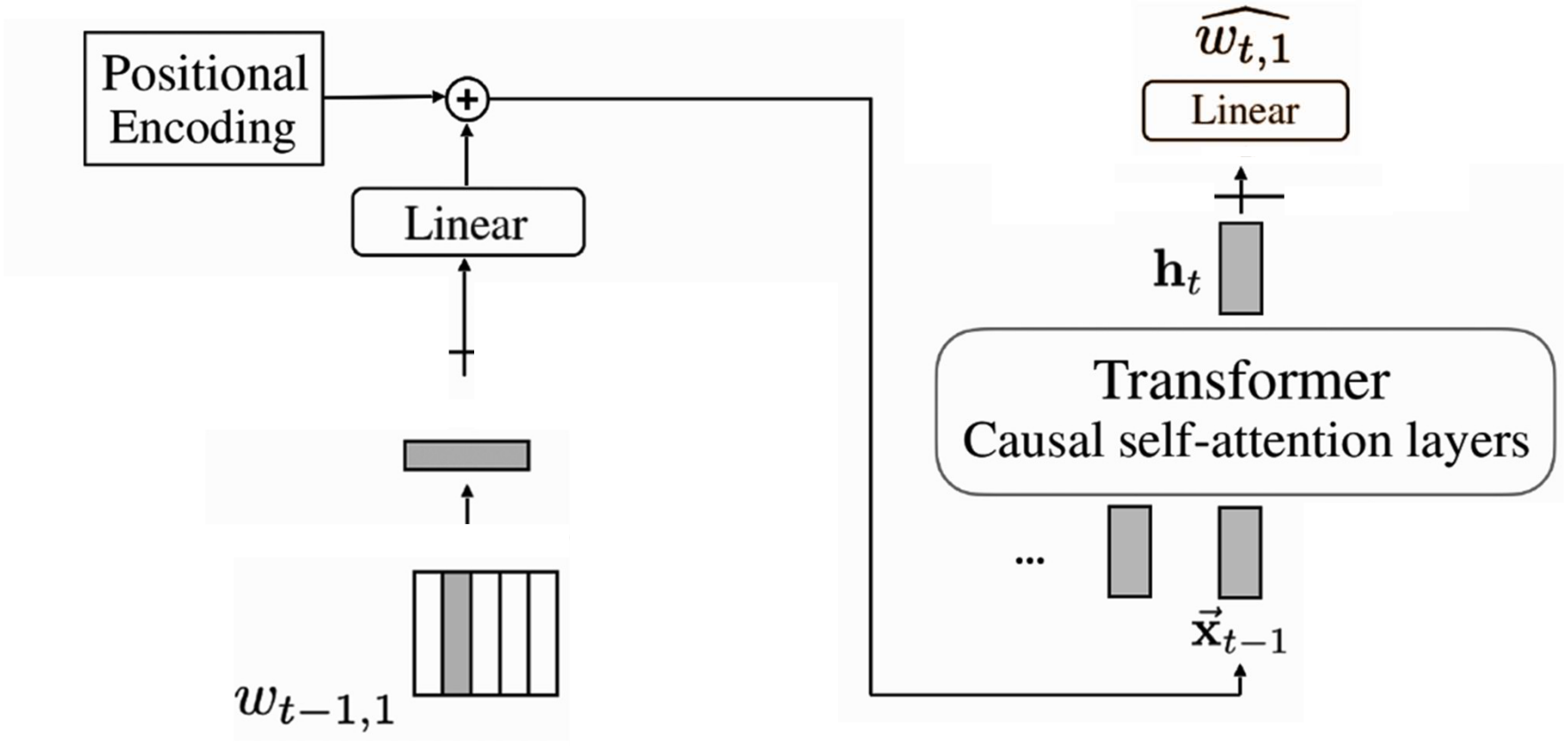


- Therefore, we

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

Original Transformer

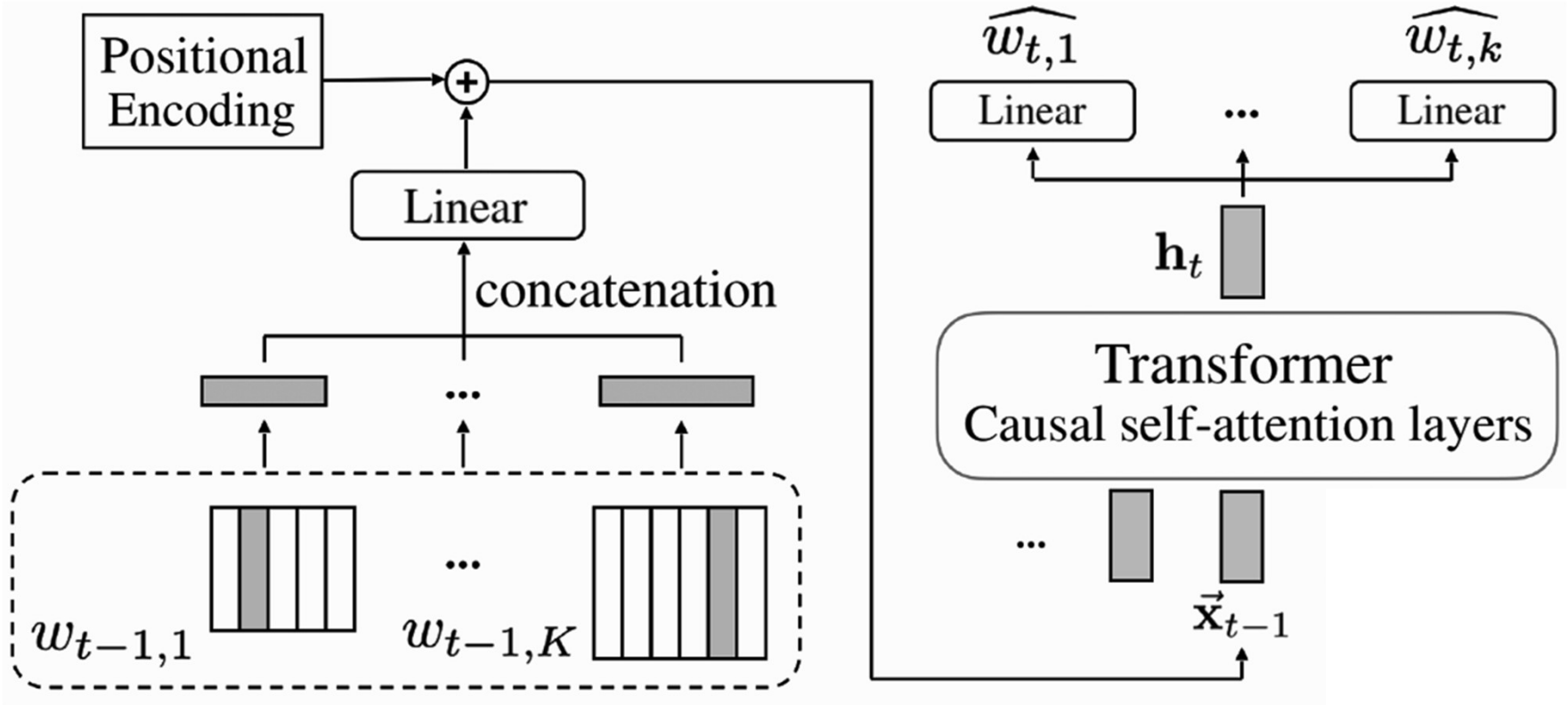
single output at each time step



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 for training on a single NVIDIA RTX 2080 Ti GPU
- And the model converges within 1.5 days

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

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

"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].

Open Research

Or, try the Pop Music Transformer first!