Homework-3
Symbolic Music Generation

HW3 TA：李維釗 (Wei-Jaw Lee)

Office hour: Mon. 13:30~15:00 @BL505

Outline

- Overview
- Detailed Explanation
- Submission
- Scoring
- Rules
- Timeline

Overview

1. Learn to manipulate MIDI file and represent symbolic music as tokens
2. Learn to train an **autoregressive model** for symbolic music generation

Detailed Explanation

- Symbolic music generation
- Dataset
- Evaluation
- Tasks
  a. Unconditional generation
  b. Continuation generation
- Optional
- How to start
- Warning & Training Tips

Symbolic music generation

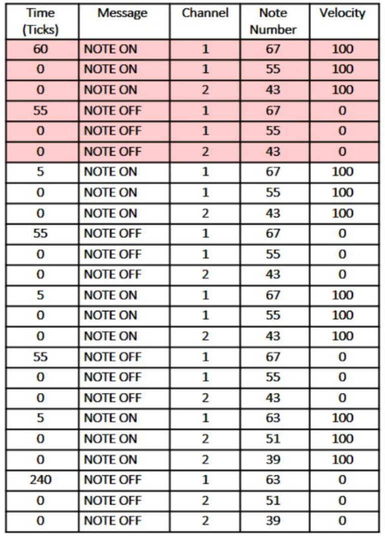
- MIDI file manipulation: represent symbolic music as tokens.
- Train a transformer-based model to generate symbolic music.

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

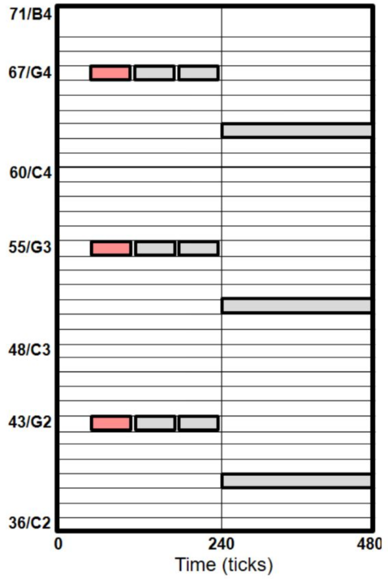
MIDI file manipulation

By viewing the MIDI messages as “tokens”.

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



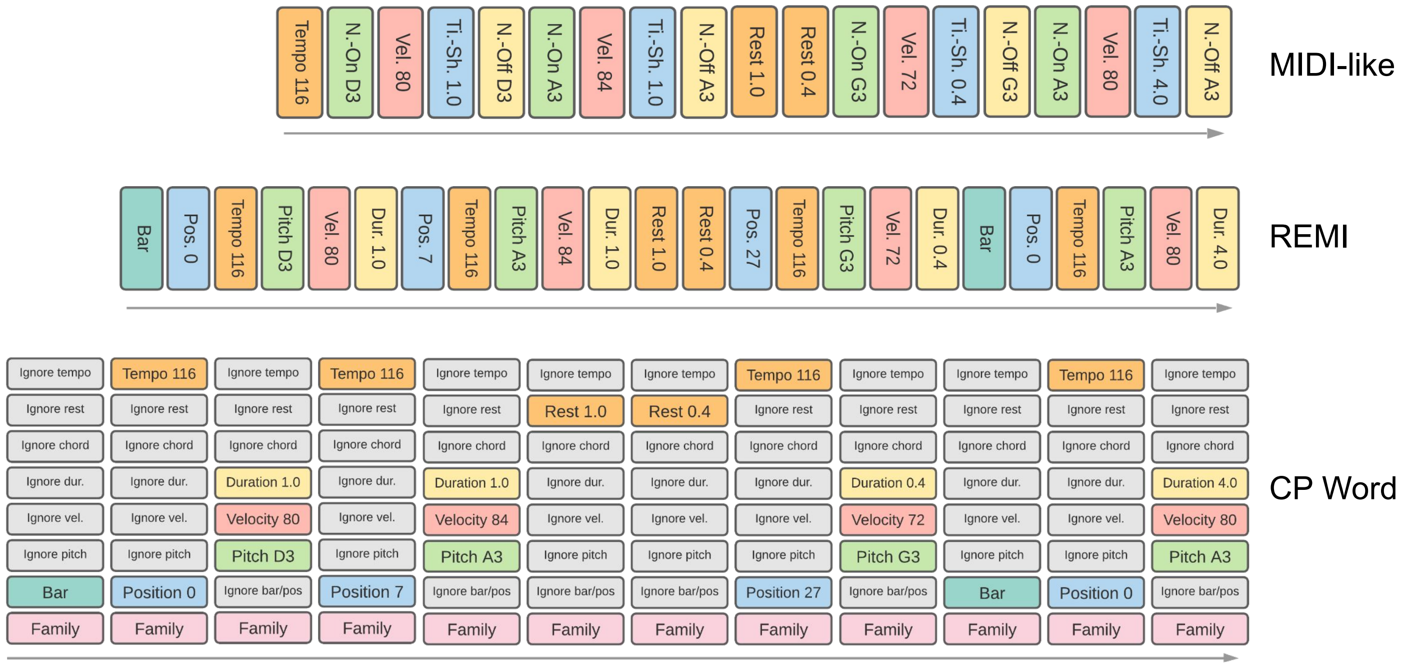
Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE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



Music piano representation

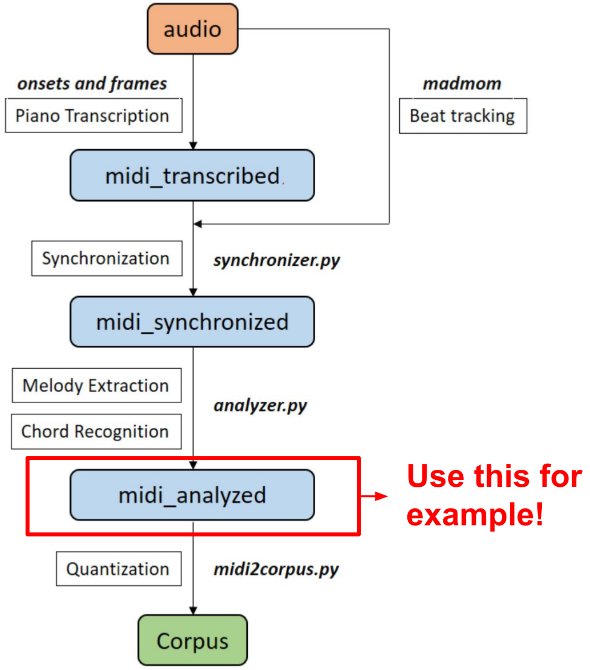
There are many representation ways for music performance.

- MIDI-like
- REMI
- REMI+
- CP word
- etc…



Dataset: Pop1K7

data processing pipeline



- 1747 pop music piano performances (mid or midi file) transcribed from youtube audio
- Single track, 4 minutes in duration, totaling 108 hours
- 4/4 time signature (four beats per bar)

➔ You can use **whole dataset** for training, there is **no need to split** train/validation set for generation task.

➔ Don't use other datasets (for the evaluation score of your generation, the closer it is to the Pop1k7 dataset, the better)

Ref: Hsiao, Wen-Yi, et al. "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 1. 2021.

Evaluation

Objective metrics [1] for generation results

- **Pitch-Class Histogram Entropy (H)** : measures erraticity of pitch usage in shorter timescales (e.g., 1 or 4 bars). (**H4 is required**)
- **Grooving Pattern Similarity (GS)** : measures consistency of rhythm across the entire piece. (**required**)
- **Structureness Indicator (SI)** : detects presence of repeated structures within a specified range of timescale. (**optional**)

➔ Use MusDr. (TA will provide sample code: *eval_metrics.py*)

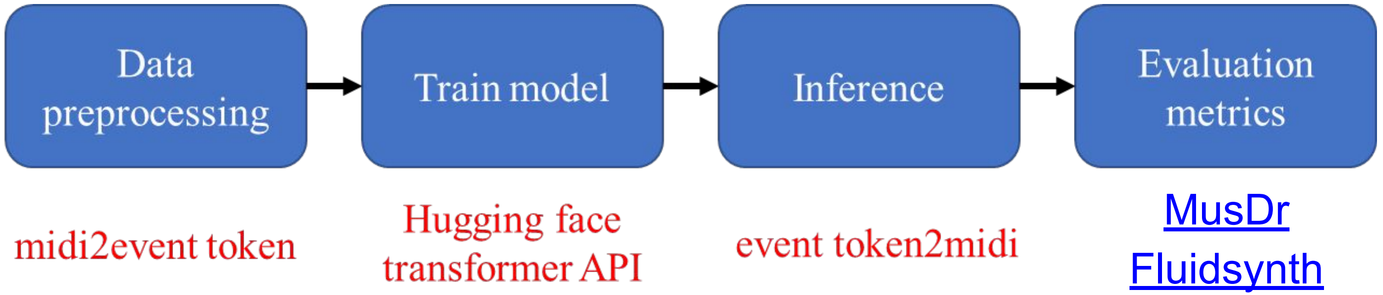
➔ The closer the score of generation results compare to the original dataset, the better.

[1] Huang, Yu-Siang, and Yi-Hsuan Yang. "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions." *Proceedings of the 28th ACM international conference on multimedia*. 2020.

Task 1: Unconditional generation

Train a transformer-based model from scratch to generate 32 bars symbolic music

- Model: You can use any autoregressive model
- Either 1-stage generation or 2-stage generation is fine



[MusDr]

[Fluidsynth]

Task 1: Unconditional generation

Report

1. What's your Token representation, model architecture
2. Implementation detail (data augmentation, hyper-parameters…)
3. Implement at least 3 combinations of your inference configurations. (it's quite influential!)
   - e.g. model loss, sampling method, top-k, temperature, etc. (see Lecture 9.)
4. For each combination, generate 20 mid/midi files (32 bars) to calaulate their average objective metrics results (H4, GS)

Submission files

5. Choose one combination, generate 20 mid/midi files (32 bars) and convert into wav files as listening samples

Task 1: Unconditional generation

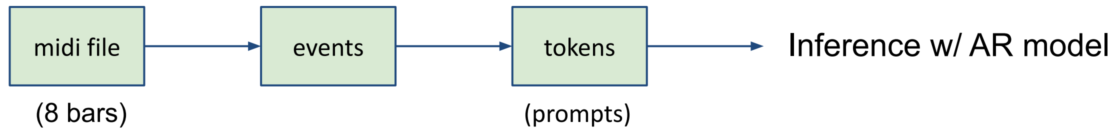
Example for objective result comparison

Model	representation	event	loss	top-k	temperature	H1	H4	GS	SI_short	SI_mid	SI_long
GPT2	REMI	w/ chord		5	1.2						
Transformer-XL	CP Word	w/o chord									
Real data											

Task 2: Continuation generation

TA will provide 3 midi files (8 bars) as prompt, you need to generate their continuation for 24 bars. (Total: 8+24 bars)

- You can use the checkpoint in Task 1; no need to train another model.
- Inference pipeline:



- You can consider this task as a「指定曲」competition. Since everyone is given the same prompts, you can compare your generation results with others.

Task 2: Continuation generation

Report

1. No need to report objective metrics results; it's not needed in Task2.
2. For each prompt song, need to generate 1 mid/midi files (8+24 bars) for 3 different inference configurations and convert into wav files as listening samples.
3. Need to specify the model, representation, inference configurations you used for each continuation.

Submission files

4. For each prompt song, need to specify the one you think that generates the best among different inference configurations.

We will choose some students to present their best results!

Optional Task

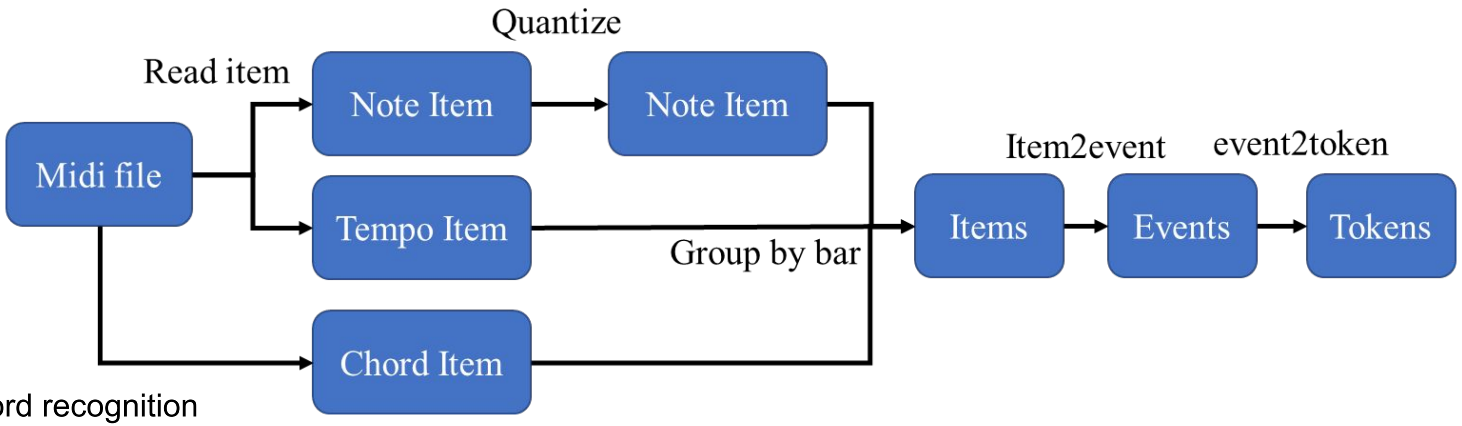
There are several optional tasks you can try. It's not required, but why not trying them if you have time?

1. **Different model**
   - 1-stage generation, 2-stage generation, different model architecture, …
2. **Different token representation**
   - functional representation, chord extraction, …
3. **Difference tokenization strategy**
   - Byte Pair Encoding, Unigram, WordPiece, …
4. **Data augmentation**
5. **Conditional generation**
   - Given key, chords, lyrics, prime melody, theme, …

How to start

Tokenization - Step-by-step tutorial

- TA will provide code (*tutorial.ipynb*) with some blank part. It may help your data pre-processing if you are not familiar with the MIDI processing pipeline.
- This tutorial is for midi file to REMI events.



Chord recognition
Ex: Chorder

How to start

It’s a powerful library but a little bit boring if you want to dig into midi file manipulation…

Tokenization - MidiTok [Docs], [Github]

- You also may use this package for MIDI file tokenization.
- It can tokenize symbolic music files (MIDI, abc), i.e. convert them into sequences of tokens ready to be fed to models such as Transformer, for any generation.
- Include: REMI, REMI+, MIDI-Like, TSD, Structured, CPWord, Octuple, MuMIDI, MMM, PerTok.
- Tokenizers can be trained with BPE, Unigram or WordPiece.
- Feature data augmentation (on MIDI level or token level).
    - e.g. increase velocities, durations of notes, shift pitches by octaves, …

A good start from MidiTok:
https://github.com/Natooz/MidiTok/blob/main/colab-notebooks/MidiTok_Full_Workflow_Tutorial.ipynb

How to start

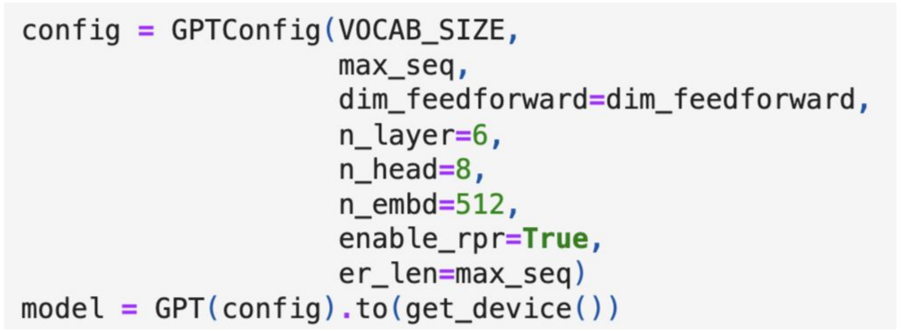
Tokenization - MidiTok [Docs], [Github]

```python
# Our parameters
pitch_range = range(21, 109)
beat_res = {(0, 4): 8, (4, 12): 4}
num_velocities = 32
additional_tokens = {'Chord': True, 'Rest': True, 'Tempo': True,
                     'rest_range': (2, 8),  # (half, 8 beats)
                     'num_tempos': 32,  # num of tempo bins
                     'tempo_range': (40, 250),  # (min, max)
                     'Program': False}

# Creates the tokenizer and loads a MIDI
tokenizer = REMI(pitch_range, beat_res, nb_velocities, additional_tokens) # REMI encoding

# tokenizer = CPWord(pitch_range, beat_res, nb_velocities, additional_tokens) # CP encoding


midi = MidiFile('Optimus-VIRTUOSO-RGA-Edition-Main-Sample.mid')

# Converts MIDI to tokens, and back to a MIDI
tokens = tokenizer.midi_to_tokens(midi)
```

How to start

Training - Hugging Face Transformer

- You can use Hugging Face Transformer API to for training and inference.
- It may be easier if you are not familiar with Transformer model implementation.
- Also it’s easier to compare different models, training and inference configurations, …
- Feel free to explore the documentation and source code to dig deeper.



```
config = GPTConfig(VOCAB_SIZE,
                   max_seq,
                   dim_feedforward=dim_feedforward,
                   n_layer=6,
                   n_head=8,
                   n_embd=512,
                   enable_rpr=True,
                   er_len=max_seq)
model = GPT(config).to(get_device())
```

How to start

Training - Open source code

You may consider using those open source code, it is helpful to learn the whole pipeline of symbolic music generation.

- 1-stage generation: Pop Music Transformer
- 2-stage generation: Compose & Embellish

TA also provides sample code for whole training & inference pipeline with some blank part.

Warning & Training Tips

Start this homework as soon as possible!

- It takes time to train a Transformer with a large dataset. (maybe several days for training to generate beautiful music)
- TA's experience (REMI + Transformer-XL)
  - 15~40 mins / epoch (GTX 1080 Ti) depending on training config
  - Model starts to converge when over about 100 epoch
- For those don't have good GPU, you may consider using smaller subset, or use smaller x_len (like 512) for model input, or use CP word to reduce sequence length.

Warning & Training Tips

- At least **12-layers** Transformer is recommended.
- For SI score, it is time-consuming to compute scape plot.
  Start early if you want to report it.
- **Don’t use other dataset**, as it may cause the result tokens distribution far from Pop1K7.

Submission file and details

1. Report (to NTU Cool)
2. Readme file and requirements.txt (to your cloud drive)
3. Code and one model checkpoint for inference (to your cloud drive)
4. Generation results (to your cloud drive)
    a. Task 1: 20 midi files + 20 wav files for the best inference configuration
    b. Task 2: (3 midi files + 3 wav)*3 files for each prompt song
    (specify the best one for each prompt song!)

- We will randomly select several classmates' code to run inference on your model and run the score on your results, so please ensure that the files you upload include trained model which can successfully execute the entire inference process.
- Don’t upload: training data, preprocessed data, others model, cache file

Report

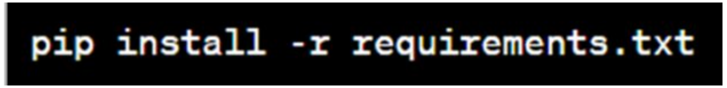
- Write with PPT or PPT-like format (16:9)
- Upload studentID_report.pdf (ex: r12345678_report.pdf)
- Please create a report that is clear and can be understood without the need for oral explanations.
- There is no specific length requirement, but it should clearly communicate the experiments conducted and their results. Approximately 10 pages is a suggested standard, but not a strict limitation.

Report template

- **Cover page**: your name, student ID etc
- **Novelty highlight** (one page; optional): what's special about your work?
- **Methodology highlight** (one page): how did you make it? Or, list the attempts you have made
- **Details of your approach** (multi-pages): If you use open source code, you may want to read some of the associated paper(s) and summarize your understanding of the paper(s) (e.g., why it works)
- **Result analysis & discussion** (multi-pages)

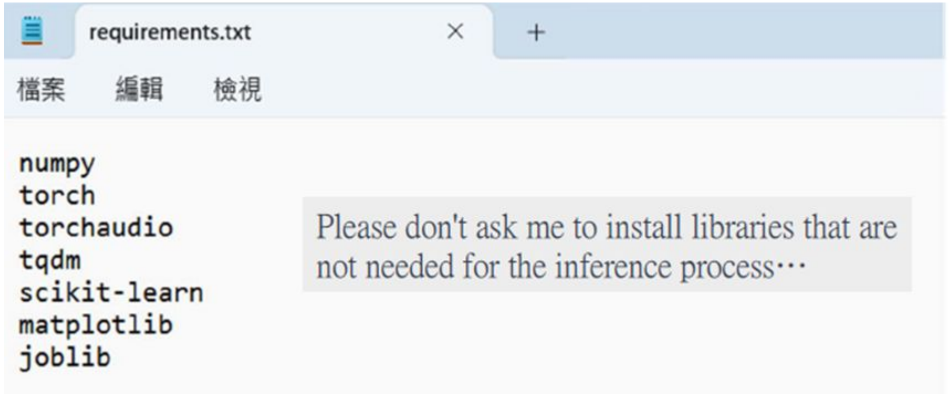
Code

- Upload **all your source code and model** to a cloud drive, open access permissions, and then upload the link to the NTU Cool assignment HW3_report in comments, as well as include it on the first page of the report
- You will need to upload requirements.txt
- I'll run :



```
pip install -r requirements.txt
```

If you have used third-party programs that cannot be installed directly via 'pip install,' please write the URL and install method command by command on **your readme file.**



```
numpy
torch
torchaudio
tqdm
scikit-learn
matplotlib
joblib
```

Please don't ask me to install libraries that are not needed for the inference process…

Code

- You will also need to upload readme file to guide me on how to perform inference on your model. I will inference the generation from scratch, which is to make sure your generation results is not from others.
- The inference code should generate some listening samples.
- The inference process should allow me to set the output file path.

Scoring

- HW3 accounts for 15% of the total grade
- Report: 100% (You'll get an A- if you only achieve the minimum requirements. Adding another experiment results, analysis, discussion will help you to get higher score.)

When you encounter problem

1. Check out all course materials and announcement documents
2. Use the power of the internet and AI
3. Use **Discussions** on NTU COOL
4. Email me weijaw2000@gmail.com or come to office hour

Rules

- Don’t cheat
- Use transformer-based generation model
- Can use pretrained model, but not suggested
- Can use public codes with citation in report
- Don’t use extra data

Timeline

- W9 – 10/30 (Thursday): Announcement of HW3
- W12 – 11/19 (Wednesday 23:59pm): Deadline

Late submission: 1 day (-20%), 2 days (-40%), 3 days (-60%), after that (no score)

ALL things you need to do before 11/19 23:59

- HW3_report
  - StudentID_report.pdf
- Cloud drive link
  - generation results
  - README file
  - requirements.txt
  - Codes and model to run inference
  - Others codes