SYMPAC: SCALABLE SYMBOLIC MUSIC GENERATION WITH PROMPTS AND CONSTRAINTS

Haonan Chen[1]   **Jordan B. L. Smith**[2]   **Janne Spijkervet**[1]   **Ju-Chiang Wang**[1]
Pei Zou[1]   **Bochen Li**[1]   **Qiuqiang Kong**[3]   **Xingjian Du**[1]

[1] ByteDance Inc. San Jose, USA
[2] Queen Mary University of London
[3] Department of Electronic Engineering, The Chinese University of Hong Kong

haonanchen@bytedance.com

ABSTRACT

Progress in the task of symbolic music generation may be lagging behind other tasks like audio and text generation, in part because of the scarcity of symbolic training data. In this paper, we leverage the greater scale of audio music data by applying pre-trained MIR models (for transcription, beat tracking, structure analysis, etc.) to extract symbolic events and encode them into token sequences. To the best of our knowledge, this work is the first to demonstrate the feasibility of training symbolic generation models solely from auto-transcribed audio data. Furthermore, to enhance the controllability of the trained model, we introduce SymPAC (Symbolic Music Language Model with Prompting and Constrained Generation), which is distinguished by using (a) *prompt bars* in encoding and (b) a technique called *Constrained Generation via Finite State Machines (FSMs)* during inference time. We show the flexibility and controllability of this approach, which may be critical in making music AI useful to creators and users.

1. INTRODUCTION

The success of language models — especially large ones — has demonstrated that with more data and larger models, using a simple language model objective can endow a model with powerful natural language generation capabilities. On the other hand, although symbolic music and natural language share many similarities, no music model has yet seemed to match the capabilities of generative text models. One reason for this gap is the insufficient amount of symbolic music data.

To address this, previous efforts in symbolic music generation have involved combining limited manually annotated data with data obtained by automatic transcription [1], or collecting private symbolic training datasets [2]. By contrast, in this work, we demonstrate that a high-quality, multi-track symbolic music generation model can be trained just using results from running Music Information Retrieval (MIR) models on audio music data. In this way, our framework eliminates the need for manually annotated symbolic music data, allowing for expansion purely through audio datasets.

On the other hand, there has been a recent surge of efforts that directly generate the auditory modality of music [3–5]. This is useful for some applications, but typically precludes fine-grained control and editing the outcome, which is crucial for composers who wish to shape their musical ideas precisely. In contrast, outputting symbolic data gives composers the ability to interactively shape and modify their musical ideas.

Considering such advantages, the problem of how to integrate user input to control the generation of symbolic music has been a popular research topic. In previous works, two methods for incorporating control signals are usually used. The first approach is based on a Variational Autoencoder (VAE) [6, 7], wherein the control is exerted within the VAE's latent space. The second approach is to embed control information directly into the encoding of symbolic music and implant control inputs during inference [8–11].

In this work, we introduce the **SymPAC** framework (**Sym**bolic Music Language Model with **P**rompting **A**nd **C**onstrained Generation), designed to work with decoder-only language models to enable user input controls. The SymPAC framework consists of the following two parts. First, inspired by the prompting mechanism used in the natural language domain [12, 13], we introduce *prompt bars* in our symbolic music encoding, which consolidates all control signals into a separate prompt section before encoding the actual musical notes. This design is essential for a decoder-only language model to have the full context of control signals during the generation of music. Second, in the controlled symbolic music generation setting, the generated tokens should not only comply with the encoding grammar but also adhere to user inputs. Thus we propose to use *Constrained Generation via Finite State Machines (FSMs)*, which constrains the sampling of tokens at each time step to a subspace. We will discuss the advantages of SymPAC over previous methods in Section 2, and provide more details of how SymPAC can be used for various types of user inputs in Sections 3 and 4.

We collected roughly one million in-house audio samples and extracted MIR information for each, using pre-trained models for beat tracking [14], chord detection [15], section detection [16, 17], multi-track transcription [18], and music tagging [19]. The MIR results were transformed into various tokens, and then integrated into an extended REMI [10, 20] encoding to train a language model based on Llama [21] architecture. To summarize, our main contributions are:

Scalability: We demonstrate that a high-quality symbolic music generation model can be trained solely with transcribed data, without the need of manually annotated symbolic music, and can be scaled by amassing more audios.

Controllability: We propose the SymPAC framework, which enables flexible user input controls on a decoder-only language model while retaining good quality.

2. RELATED WORK

2.1 Training Data For Symbolic Music

In Table 1, we summarize some popular music datasets in the symbolic and audio domains, together with our in-house audio dataset, and compare their sizes. The Lakh MIDI Dataset [24] is one of the biggest public datasets, containing 170K multitrack pieces in MIDI format. Many researchers use publicly available symbolic music datasets for training, but some collect and use large-scale ones that are not disclosed; e.g., MusicBERT [2] was trained on the Million-MIDI Dataset (MMD).

Although the combined size of the public datasets in Table 1 is large, combining them is not straightforward since they vary in format. For example, the Maestro dataset consists of transcriptions of piano performances where note timings reflect actual performance timings, whereas datasets like Lakh are quantized to metrical time with alignment to beats. The inclusion of instrument tracks and additional information (e.g., chords, sections) also differs between datasets. To expand the scale of training data by combining these datasets, it is necessary to unify their formats first, which may be tedious and introduce errors.

On the other hand, publicly available audio datasets are much larger in scale. The Million Song Dataset (MSD) [26], for example, contains 1M songs, or 709M notes in total after being run through a 5-track transcription model [18]. The recently published DISCO-10M [27] is of an even larger scale. Furthermore, by using a single set of MIR models to annotate all the audio data, we do not need to be concerned about the issue of inconsistent data formats. This makes it easier to scale up the training dataset.

2.2 Encoding For Symbolic Music

Since the introduction of the Music Transformer [28], language models based on the transformer architecture have become a popular choice for symbolic music generation. One of the most critical research questions has been how to encode symbolic music that is amenable to processing by such a model, which, in the context of language models, involves converting the piece into a sequence of tokens.

Early transformer-based models for symbolic music predominantly employed a MIDI-like encoding scheme, by treating MIDI event sequences almost identically as input token sequences [8, 9, 29]. Later, the Revamped MIDI (REMI) encoding [20] was proposed, which modified the MIDI encoding by replacing time shift events with duration events for each note and introducing bar and beat concepts to adopt metrical time instead of absolute time. These modifications facilitated the model's learning of rhythmic patterns within the music, improving the quality of the output. Building upon REMI, several extensions have been proposed to support encoding multitrack [9] and various control tokens [10]. Our work is based on the multitrack REMI encoding, and given the MIR models we have, it incorporates control tokens such as genre, chord, and section tokens to the encoding.

2.3 Controllable Symbolic Music Generation

Previous methods for controlling symbolic music generation have typically fallen into two categories. The first is based on Variational Autoencoders (VAEs) [6, 7]. VAEs aim to find a latent space for representing music that encodes distinct musical attributes in independent dimensions. This disentanglement allows for specific attributes of generated music (e.g., rhythm, genre, or timbre) to be individually manipulated by altering corresponding dimensions in the latent space without affecting other attributes, thereby enhancing the controllability of music generation.

The second approach is to include control tokens in the encoding of symbolic music. For example, MMM [9] includes instruments and note density tokens in the encoding, which can be specified at inference. Similarly, FIGARO [10] uses "expert descriptions" indicating time signature, note density, mean pitch, mean velocity and mean duration as well as instruments and chords. It then uses an encoder-decoder model to learn a mapping from descriptions to sequences of a piece of music. Driven by the development of Large Language Models (LLMs), recent work has also explored using natural language to control symbolic music generation [30–33]. Natural language text can also be treated as control tokens, with the key distinction that it usually requires pre-training the LLM on text.

In our work, the proposed SymPAC framework is designed to work with a decoder-only language model. In a controlled generation setting, prompt bars that conform with user input control signals are generated first. The generation of musical part comes after that, in which the model will have full context of control signals from prompt bars. These two generation stages are both controlled by an FSM, which takes into consideration the grammar of the encoding and user inputs. There are two main differences between SymPAC and previous works

1. We encode control signals as tokens and use FSM to enforce input control signals during inference. In contrast, for VAE-based control methods, control signals are converted into latent embeddings, and the model is not guaranteed to follow these control signals.

2. Since we use a decoder-only language model, the to-

Dataset	#Songs	#Notes	Format	Multitrack	Public
Maestro [22]	1.1K	6M	MIDI	N	Y
GiantMIDI-Piano [23]	10.9K	39M	MIDI	N	Y
Lakh [24]	170K	910M	MIDI	Y	Y
MMD [2]	1.5M	2,075M	MIDI	Y	N
FMA [25]	100K	N/A *	Audio	Y	Y
MSD [26]	1M	709M	Audio	Y	Y
DISCO-10M [27]	15M	N/A *	Audio	Y	Y
In-House Dataset (IHD)	1M	3,688M	Audio	Y	N

Table 1: Comparison of different symbolic and audio music datasets. * Since we did not run transcription on FMA or DISCO-10M, we don't have the number of notes information for them.

kens in prompt bars are also learned simultaneously. Consequently, the user is only required to input a portion of the control information, with the model being able to automatically generate missing controls. In contrast, an encoder-decoder framework like the one described in [10] would require a complete encoder input during inference, which lacks flexibility.

3. METHOD

3.1 Symbolic Music Encoding And Prompt Bars

Our data representation is based on the REMI+ [10] representation, an extension of REMI [20] that supports multitrack data. An illustration of our encoding is shown in Fig. 1. The fundamental unit of our encoding is a bar, of which there are two types: *prompt bar* and *song bar*. The token sequence of a song bar can be divided into four parts:

- The *meta* part includes four tokens for the `bar`, `genre`, `sec` (for section type name), and `bpm_level` (which indicates the tempo range).

- The *chord* part consists of alternating `position` and `chord` tokens.

- Each *instrument track* part consists of a `track` token, followed by one or more groups of `position`, `duration` and `pitch` tokens.

- The *drum track* part consists of a `track<drum>` token, followed by one or more groups of `position` and `drum` (drum MIDI) tokens.

Here are further explanations of `position`, `duration` and `track` tokens [1]:

- `position`: Represents the starting position of subsequent `chord`, `pitch` or `drum` token within a bar. Each bar is divided into 16 steps, so that position ranges from 0/16 to 15/16.

- `duration`: Ranges from the minimum time division of 1/16 bar to a maximum of 2 bars, or 32/16.

- `track`: A track token will only exist if there is at least one note in the bar for the corresponding instrument. This allows the user to control which instruments are used within a bar.

---

[1] Details of all token types are provided in supplementary materials

Prompt bars contain a subset of tokens in song bars, retaining only tokens that represent *control signals*. In our case, these include genre, section, tempo, chords and tracks. As future work, this encoding could be extended to include more control signals (e.g. note density for a track). The encoding of a full piece of music will consist of: all prompt bars in the piece; then, a special `end_of_prompt` token; then, all song bars in the piece; and finally a special `end_of_song` token.

During training stage, the model is trained to predict tokens in prompt bars as well, not distinguishing them from tokens in song bars. As mentioned previously, this design enables the user to input partial control signals (or no input at all), and the model is able to infer the missing ones.

Algorithm 1 Constrained Generation via FSM

1: **procedure** CONSTRAINEDSAMPLING($\mathcal{M}$, $\mathcal{V}$, $\mathcal{R}$)
2: $\quad$ $\mathbf{s}_0 \leftarrow x_0$ start token (`bar` in our encoding)
3: $\quad$ $q_0 \leftarrow$ initial state
4: $\quad$ $t \leftarrow 0$
5: $\quad$ **while** not end of sequence **do**
6: $\quad\quad$ $\mathcal{V}_{t+1} \leftarrow$ GETSUBVOCAB($\mathcal{R}$, q_t, x_t)
7: $\quad\quad$ $q_{t+1} \leftarrow$ UPDATESTATE($\mathcal{R}$, q_t, x_t)
8: $\quad\quad$ $x_{t+1} \leftarrow$ SAMPLE($\mathcal{M}$, $\mathcal{V}_{t+1}$)
9: $\quad\quad$ $\mathbf{s}_{t+1} \leftarrow \mathbf{s}_t \circ x_{t+1}$
10: $\quad\quad$ $t \leftarrow t + 1$
11: $\quad$ **end while**
12: $\quad$ **return** $\mathbf{s}_t$
13: **end procedure**

3.2 Constrained Generation via FSM

In the controlled symbolic music generation setting, there are two types of constraints:

Grammar constraint: The encoding of symbolic music follows a specific format. For example, for our proposed encoding shown in Fig. 1, a `bar` token will always be followed by a `genre` token.

User input constraint: Generated token sequence should conform with user inputs. For example, if the user wants to generate "rock" style music, the `genre` token can only be `genre<rock>`.

Since we are already aware of these constraints in advance, there is no need to sample from the entire vocabulary space during inference. Instead, we can sample from
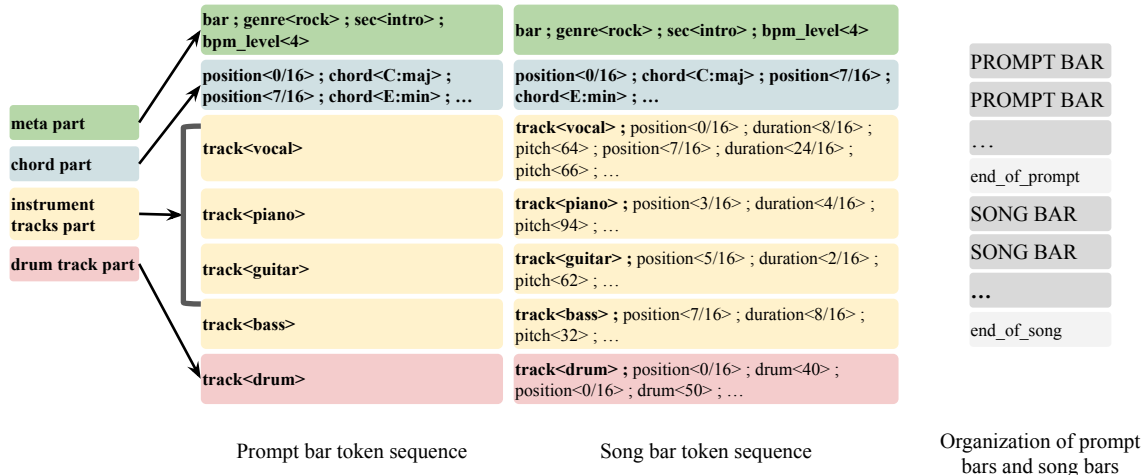
Figure 1: Illustration of our symbolic music encoding.

a subspace that is in accordance with the constraints.

To achieve this, we employ a Finite State Machine (FSM) to interact with the language model $\mathcal{M}$ during inference. Let x_t denote the token generated by $\mathcal{M}$ at time step t. The FSM takes x_t, the current state q_t and the predetermined rule set $\mathcal{R}$, and outputs a subset of the vocabulary $\mathcal{V}_{t+1}$, from which the language model $\mathcal{M}$ can sample at time $t + 1$. We call this procedure Constrained Generation via FSM, which is formally defined in Algorithm 1. This algorithm is analogous to regular expression matching, where it checks if a given input string conforms to a specified pattern. Here the pattern and input string are equivalent to rule set $\mathcal{R}$ and token sequence s_t respectively.

4. EXPERIMENTS AND RESULTS

To validate our contributions, we conduct experiments to assess whether the system is scalable (i.e., improves when scaling up training data) and controllable (i.e., there is consistency between generation output and user inputs).

In Sec. 4.3, we conduct a quantitative analysis to compare models trained on different amounts of training data, in order to assess scalability. In Sec. 4.4, we examine two common types of control inputs: chord progression and section structure. The impact of these control inputs is tested through both quantitative metrics and qualitative examples. Lastly, in Sec. 4.5, we compare our models trained on different datasets with other baseline symbolic music generation systems through subjective evaluation.

4.1 Datasets

We use three datasets in our experiments. We always use each dataset individually; i.e., we never merge the datasets to train a single model. The datasets are:

Lakh MIDI Dataset (LMD) [24]. A dataset in MIDI format, containing around 170K songs. We use this to compare with models trained on transcribed audio data.

Million Song Dataset (MSD) [26]. A public dataset used extensively by MIR researchers. We use the 30–60s preview audio clips, representing the highlight of the song.

In-House Dataset (IHD). We use a licensed internal collection with about 1M full songs in audio format, covering a wide range of Western modern genres.

4.2 Training Settings

We train a decoder language model with the Llama [21] architecture. We set the number of layers, number of attention heads and embedding dimensions to be 12, 12 and 768 respectively, resulting in a model with about 86M trainable parameters. We concatenate token sequences of all pieces into a 1-D array, and randomly pick a window of size 10,240 as one training sample. As the average sequence lengths of LMD, MSD and IHD are 900, 1500 and 8000 respectively, this window size would contain 11.4, 6.8 and 1.3 pieces on average for each dataset.

When training data are limited, data augmentation and data filtering (to ensure that unusual data do not pollute the training) are commonly used. However, we adopt neither approach, for two reasons. First, since we have a large dataset of audio samples, the training data are likely to cover a broad spectrum of examples already, reducing the need to filter out unusual data points. Second, augmentation may alter the training data in unwanted ways. For example, a common augmentation approach is to transpose all the pitches in a piece [11, 28]. However, this may distort the pitch ranges of each instrument: e.g., if the input bass parts are transposed up and down, the model will not learn the correct range of realistic bass notes.

Metric Class	IHD 100%	IHD 10%	IHD 1%
Chord	**0.112**	0.119	0.347
Structure	0.348	**0.220**	0.786
Vocal Note	**0.416**	0.892	1.086
Guitar Note	**0.222**	0.257	0.397
Piano Note	**0.178**	0.403	0.686
Bass Note	**0.180**	0.867	1.038
Drum Note	**0.650**	2.902	1.248

Table 2: Average Kullback-Leibler Divergence (KLD) of metrics in different metric classes for models trained on different dataset against a held-out validation set.

4.3 Unconditioned Generation

Intuitively, increasing the amount of data should enhance the performance of the model. In this experiment, we use objective metrics to validate this. Designing objective metrics to evaluate symbolic music remains an open question. A common approach is to prepare a reference dataset, calculate embeddings or metrics of the generated samples and reference set, and then compare these using distance metrics such as the Fréchet Distance or Kullback-Leibler Divergence (KLD). For a detailed review on evaluation methods for symbolic music, see [34, 35].

In our experiment, we prepare a held-out validation set with 3000 samples. We use a range of metrics that can be categorized into the following classes: chord, structure, instrument note (including vocals, guitar, piano and bass) and drum note. Detailed definitions are provided in supplements. In general, the metrics in each class are as follows:

- **Chord**: chord label, chord root, chord transition;

- **Structure**: section label, section label bigram, instrument labels per bar;

- **Instrument Note**: note pitch, note duration, pitch class, min/max pitch per bar, max number of notes per bar, uniformity of number of notes per bar;

- **Drum Note**: drum key, max number of notes per bar, uniformity of number of notes per bar, and unique drums per bar.

We compare models trained on 100%, 10% and 1% of the IHD data, and do generation in an unconditioned setting. For each model, we generate 800 samples to compute metric distributions. KLD values are then computed between distributions of generated samples and distribution of the validation set for each metric. Lower KLD indicates that two distributions are closer, suggesting the generated samples sound more similar to the validation set. We report the average KLD values for the same class, and provide a full list of KLDs for each metric in supplements.

The results are shown in Table 2. We can see that the model trained with 100% IHD data has the lowest KLD against the validation set on 6 out of 7 classes, and the model trained on only 1% data has the highest KLD on 6 out of 7 classes. The results confirm that a model trained on more data can generate samples closer to the training data. Furthermore, we observed that the benefit of using more data is greater for the 'Note' metrics than for the 'Chord' or 'Structure' ones. This is likely because note tokens are more numerous and have complex distributions, which needs larger scale of data to learn. Counterintuitively, the KLD for 'Structure' was better when using 10% of the data instead of 100%. We speculate that since the structure tokens are scarcest, this could be the result of a lucky alignment between the validation set at the 10% of the data used, but this deserves more study.

4.4 Controlled Generation

The SymPAC framework aims to give users flexible control over the music generation process. However, we need to verify that this control is effective: do the notes generated agree with the control inputs? To this end, we conduct controlled generation experiments on two input scenarios: chord progression inputs and section structure inputs.

Chord Progression Inputs. In this experiment, we randomly pick 20 top trending chord progressions from *HookTheory* [2] as the chord progression inputs. We only include major and minor triad chords. We then let the model generate 64 bars of music by looping the chord progressions. To evaluate the match between the input chord progression and the output, we apply a symbolic chord detection method on the generated samples. Details about the method can be referred in the supplementary materials.

The accuracy of detected chord from the input chord progression is shown in Table 4. As shown in the result, the models trained on MSD, IHD 100% and IHD 10% all have similar overall accuracy, with MSD slightly outperforming the others. But the model trained on IHD 1% (just 10K songs) is much worse than the other three. This suggests that a dataset at the scale of 100K songs is sufficient to model low-level control signals like chord, given the model and encoding we are using here. We also provide examples in supplementary audios of outputs when given unusual chord progressions.

Section Structure Inputs. In this experiment, we take 10 typical section sequences as inputs (listed in supplements), ranging in length from 4 to 13 sections (16 to 68 bars), and use each model to generate 100 outputs per prompt. We compare the same 4 models from the previous section. For each generated output, we leverage a Music Structure Analysis (MSA) algorithm [36] to predict its structure, and compare this to the input structure. The MSA algorithm's predictions may be inaccurate, but we still expect that a greater match between the intended and estimated structure indicates more success at controlling the structure. We use Foote's algorithm [37] for segmentation and the 2D-Fourier magnitude algorithm [38] for section labeling, with a beat-wise feature embedding that averages the pitchwise MIDI piano rolls within a beat interval. We evaluate the results using `mir_eval` [39], and report three metrics: boundary prediction f-measure with a 3-second tolerance (HR3F); pairwise clustering f-measure (PWF); and the normalized entropy score f-measure (Sf). To test directly how similar the repeated sections are, we also report PWF and Sf when the ground-truth segmentation is used.

We find that all metrics are worse (lower) when the system is trained on MSD or IHD 1%, and improve substantially when at least 10% of the data are used (Table 5). This is expected, since the audio clips in MSD are only excerpts and thus not instructive for modelling full-song structure.

Fig. 2 shows the piano roll of a typical output, where the match between the intended and predicted structure was average (Sf = 0.508). Even so, the match between the intended and realized structure is evident in the piano roll: the chorus sections are similar but not identical to each other, and so are the verse sections.

In both controlled generation experiments, the gap be-

---

[2] https://www.hooktheory.com

Model	Coherence	Richness	Arrangement	Structure	Overall
FIGARO	3.12 ± 0.82	2.73 ± 0.92	2.85 ± 0.96	2.62 ± 0.80	2.74 ± 0.89
MMT	2.37 ± 0.35	2.27 ± 0.36	2.37 ± 0.34	2.08 ± 0.30	2.16 ± 0.35
Ours (IHD)	**3.55 ± 0.53**	**3.58 ± 0.38**	**3.45 ± 0.49**	**3.73 ± 0.32**	**3.60 ± 0.39**
Ours (LMD)	3.25 ± 0.34	3.25 ± 0.35	3.28 ± 0.30	3.20 ± 0.61	3.25 ± 0.46
Ours (MSD)	3.16 ± 0.27	3.17 ± 0.33	3.09 ± 0.32	3.15 ± 0.29	3.07 ± 0.28

Table 3: Results of subjective evaluation, mean opinion score (MOS)

Training Dataset	Accuracy
IHD 100%	87.2%
IHD 10%	86.9%
IHD 1%	74.0%
MSD	**87.6%**

Table 4: Accuracy of chord progressions in controlled generation with chord input.

Dataset	Regular			Oracle	
	HR3F	PWF	Sf	PWF	Sf
IHD 100%	**0.60**	**0.50**	**0.50**	**0.72**	**0.80**
IHD 10%	**0.60**	0.49	0.49	0.70	0.79
IHD 1%	0.54	0.47	0.47	0.62	0.73
MSD	0.57	0.47	0.47	0.63	0.74

Table 5: Accuracy of structure predicted from generated songs with no guidance (left) and with ground truth segmentation (right).

tween 100% and 10% IHD is very small, indicating that 10% IHD data combined with SymPAC is sufficient for achieving good adherence to control inputs. However, it is important to remember that the metrics of these two experiments only reflect whether the control signals are well-followed, not the overall quality of the generated pieces.

4.5 Subjective Evaluation

The models tested so far were all trained on transcribed audio data, so it is worth comparing with models trained directly on MIDI data. In this experiment, we compare our model trained on LMD, MSD and IHD, and also two baselines, FIGARO [10] and MMT [11], in a subjective listening test. We recruited 12 participants with the background of MIR researchers or music producers. Similar to [11], we asked each participant to rate 10 audio samples generated by each model on a 5-point Likert scale on five criteria: coherence, richness, arrangement, structure and overall [3].

The result is summarized in Table 3. All of our proposed models outperform the baselines in all dimensions. Our model trained on IHD has higher performance than the other two training data setups, which attests to the viability of leveraging audio data by running MIR models
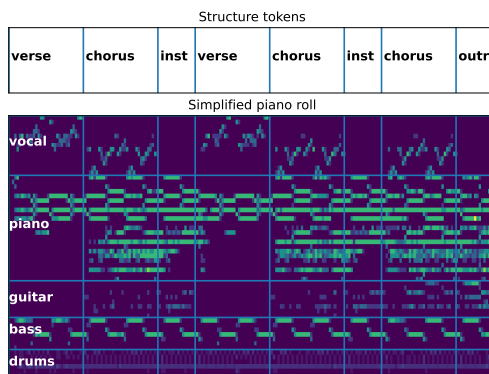

Figure 2: Constrained generation output with user-defined structure using IHD 100% model. The simplified piano roll gives beat-averaged values and excludes empty lines.

at scale. The result using LMD was better than MSD, despite having fewer songs; this could be due to LSD having more notes than MSD (see Tab 1), or due to it containing full songs instead of only excerpts. We only compare FIGARO and Ours (LMD) with a statistical test, since these were trained on the same dataset. Mann-Whitney U tests found significant differences in Richness (p = .005), Structure (p = .0005), and Overall (p = .027) ratings, but not in Coherence (p = .85) or Arrangement (p = .122).

5. CONCLUSIONS AND FUTURE WORK

We trained a language model for symbolic music generation leveraging audio data and pre-trained MIR models. We proposed the SymPAC framework, which includes prompt bars in encoding and Constrained Generation via FSM during inference time. We showed how combining these two components enables a user to control the generation process, and we evaluated the results through quantitative and qualitative analysis.

Future work could improve at least two aspects of this system: (1) We quantified position and duration to 1/16 per bar, which does not support 3/4 or 6/8 time signatures well. Also, the chord detection model we used only supports 12 major and minor chords, limiting the user input options. We can expand the encoding to support finer-grained quantization and more advanced chords. (2) Our token sequence length is long: 8000 on average for samples in IHD. We could use tokenization methods such as Byte Pair Encoding [40] or use compound word tokens [41] to compress sequences and improve training efficiency.

---

[3] These criteria are described as: (1) Coherence: The rhythm is stable; The chord progression develops logically; Dissonant notes are not excessive. (2) Richness: The melody and acccompaniment are interesting and diverse. (3) Arrangement: Collaboration among multiple instruments is harmonious and natural; Arrangements for different instruments are diverse and reasonable. (4) Structure: The piece includes a clear and engaging structure with appropriate repetitions and variations; The piece has obvious connections and reasonable developments between sections. (5) Overall: I like this piece in general.

6. REFERENCES

[1] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1899–1908.

[2] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic music understanding with large-scale pre-training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 791–800.

[3] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi *et al.*, "MusicLM: Generating music from text," *arXiv preprint arXiv:2301.11325*, 2023.

[4] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank *et al.*, "Noise2Music: Text-conditioned music generation with diffusion models," *arXiv preprint arXiv:2302.03917*, 2023.

[5] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[6] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4364–4373.

[7] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one transformer VAE," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1953–1967, 2023.

[8] C. Payne, "MuseNet," OpenAI, 25 Apr. 2019. [Online]. Available: openai.com/blog/musenet

[9] J. Ens and P. Pasquier, "MMM: Exploring conditional multi-track music generation with the transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[10] D. von Rütte, L. Biggio, Y. Kilcher, and T. Hofmann, "FIGARO: Generating symbolic music with fine-grained artistic control," *arXiv preprint arXiv:2201.10936*, 2022.

[11] H.-W. Dong, K. Chen, S. Dubnov, J. McAuley, and T. Berg-Kirkpatrick, "Multitrack music transformer," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, 2019.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[14] Y.-N. Hung, J.-C. Wang, X. Song, W.-T. Lu, and M. Won, "Modeling beats and downbeats with a time-frequency transformer," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 401–405.

[15] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, "SpecTNT: A time-frequency transformer for music audio," in *Proc. ISMIR*, 2021.

[16] J.-C. Wang, Y.-N. Hung, and J. B. L. Smith, "To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 416–420.

[17] J.-C. Wang, J. B. L. Smith, and Y.-N. Hung, "MuSFA: Improving music structural function analysis with partially labeled data," *arXiv preprint arXiv:2211.15787*, 2022.

[18] W.-T. Lu, J.-C. Wang, and Y.-N. Hung, "Multitrack music transcription with a time-frequency perceiver," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[19] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," in *Proc. ISMIR*, 2021, pp. 768–776.

[20] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[21] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[22] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and

D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," in *International Conference on Learning Representations*, 2018.

[23] Q. Kong, B. Li, J. Chen, and Y. Wang, "GiantMIDI-Piano: A large-scale MIDI dataset for classical piano music," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 87–98, 2022.

[24] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[25] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proc. ISMIR*, 2017. [Online]. Available: https://arxiv.org/abs/1612.01840

[26] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. ISMIR*, 2011.

[27] L. Lanzendörfer, F. Grötschla, E. Funke, and R. Wattenhofer, "DISCO-10M: A large-scale music dataset," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[28] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[29] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," *arXiv preprint arXiv:1907.04868*, 2019.

[30] Z. Sheng, K. Song, X. Tan, Y. Ren, W. Ye, S. Zhang, and T. Qin, "SongMASS: Automatic song writing with pre-training and alignment constraint," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 798–13 805.

[31] A. S. Hussain, S. Liu, C. Sun, and Y. Shan, "M^{2}UGen: Multi-modal music understanding and generation with the power of large language models," *arXiv preprint arXiv:2311.11255*, 2023.

[32] P. Lu, X. Xu, C. Kang, B. Yu, C. Xing, X. Tan, and J. Bian, "MuseCoco: Generating symbolic music from text," *arXiv preprint arXiv:2306.00110*, 2023.

[33] R. Yuan, H. Lin, Y. Wang, Z. Tian, S. Wu, T. Shen, G. Zhang, Y. Wu, C. Liu, Z. Zhou *et al.*, "ChatMusician: Understanding and generating music intrinsically with llm," *arXiv preprint arXiv:2402.16153*, 2024.

[34] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

[35] S. Ji, X. Yang, and J. Luo, "A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–39, 2023.

[36] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-based music structure analysis: Current trends, open challenges, and applications," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 246–263, 2020.

[37] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2000, pp. 452–455.

[38] O. Nieto and J. P. Bello, "Music segment similarity using 2D-Fourier magnitude coefficients," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 664–668.

[39] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. ISMIR*. Curitiba, Brazil: Citeseer, 2014, pp. 367—372.

[40] N. Fradet, N. Gutowski, F. Chhel, and J.-P. Briot, "Byte pair encoding for symbolic music," *arXiv preprint arXiv:2301.11975*, 2023.

[41] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 178–186.