MUSICONGEN: RHYTHM AND CHORD CONTROL FOR TRANSFORMER-BASED TEXT-TO-MUSIC GENERATION

Yun-Han Lan[1,2]   **Wen-Yi Hsiao**[1]   **Hao-Chung Cheng**[2]   **Yi-Hsuan Yang**[1,2]

[1] Taiwan AI Labs   [2] National Taiwan University

cyan0731@gmail.com, wayne391@ailabs.tw, {haochung,yhyangtw}@ntu.edu.tw

ABSTRACT

Existing text-to-music models can produce high-quality audio with great diversity. However, textual prompts alone cannot precisely control temporal musical features such as chords and rhythm of the generated music. To address this challenge, we introduce MusiConGen, a temporally-conditioned Transformer-based text-to-music model that builds upon the pretrained MusicGen framework. Our innovation lies in an efficient finetuning mechanism, tailored for consumer-grade GPUs, that integrates automatically-extracted rhythm and chords as the condition signal. During inference, the condition can either be musical features extracted from a reference audio signal, or be user-defined symbolic chord sequence, BPM, and textual prompts. Our performance evaluation on two datasets—one derived from extracted features and the other from user-created inputs—demonstrates that MusiConGen can generate realistic backing track music that aligns well with the specified conditions. We open-source the code and model checkpoints, and provide audio examples online, https://musicongen.github.io/musicongen_demo/.

1. INTRODUCTION

The realm of text-to-music generation has seen significant progress over the recent years [1–11]. These models span various genres and styles, largely leveraging textual prompts to guide the creative process. There have been two primary methodological frameworks so far. The first employs *Transformer* architectures to model audio tokens [12] derived from pre-trained audio codec models [13–15]; noted examples include MusicLM [1] and MusicGen [2]. The second employs *diffusion* models to represent audio through spectrograms or audio features, such as AudioLDM 2 [4] and JEN-1 [5].

Text-to-music generation model generally relies on the global textual conditions to guide the music generation process. Textual prompts serving as high-level conceptual guides, however, introduce a degree of ambiguity and verboseness into the music generation for describing the musi-

Model	Chord control	Rhythm control	Do not need reference audio
Coco-Mulla [6]	√	√	
Music ControlNet [7]		√	√
Ours	√	√	√

Table 1. The comparison for conditions and condition type of related temporally-conditioned text-to-music models.

cal features [7]. This inherent vagueness poses a challenge in precisely controlling temporal musical features such as melody, chords and rhythm, which are crucial for music creation. Building on the success of MusicGen-melody [2] in melody control, our focus now shifts to enhancing chord and rhythm control, aiming to create a more integrated approach to music generation that captures the full spectrum of musical elements.

Table 1 tabulates two existing studies that have explored the incorporation of time-varying chord- and rhythm-related attributes in text-to-music generation. Coco-Mulla [6] is a Transformer-based model that employs a large-scale, 3.3B-parameter MusicGen model, finetuned with an adapted LLaMA-adapter [16] for chord and rhythm control. For rhythm control in particular, Coco-Mulla uses drum audio codec tokens extracted from a reference drum audio signal as a condition for guiding the music generation, thereby demanding *reference audio* for control. While it is appropriate to assume the availability of such reference *audio* in some scenarios, for broader use cases we desire to have a model that can take user-provided *text*-like inputs as well, such as the intended beats-per-minute (BPM) value (for rhythm) and the chord progression as a series of chord symbols (for chords). This function is not supported by Coco-Mulla.

The other model, Music ControlNet [7], leverages a diffusion model architecture and the adapter-based conditioning mechanism of ControlNet [17] to manipulate text-like, symbolic melody, dynamics, and rhythm conditions. This diffusion model creates a spectrogram based on the provided conditions, which is then transformed into audio using their pretrained vocoder. For musical conditions, a 12-pitch-class chromagram representation is used for the melody, combined with beat and downbeat probability curves concatenation for rhythm control, and an energy curve to adjust the dynamic volume. However, Music ControlNet does not deal with chord conditions.

In view of the limits of the prior works, we introduce in this paper MusiConGen, a Transformer-based text-to-music model that applies temporal conditioning to enhance control over rhythm and chord. MusiConGen is finetuned from the pretrained MusicGen framework [2]. We design our temporal condition controls in a way that it supports not only musical features extracted from reference *audio* signals, but also the aforementioned user-provided *text-like* symbolic inputs such as BPM value and chord progression. For effective conditioning of such time-varying features, we propose "adaptive in-attention" conditioning by extending the in-attention mechanism proposed in the MuseMorphose model [18]. Table 1 includes a conceptual comparison of MusiConGen with existing models in terms of the conditions and their types.

In our implementation, we train MusiConGen on a dataset of *backing track music* comprising 5,000 text-audio pairs obtained from YouTube. This training utilizes beat tracking and chord recognition models to extract necessary condition signals without the need for manual labeling. We note that rhythm and chord controls are inherently critical for backing tracks, for backing tracks often do not include the primary melody and their purpose is mainly to provide accompaniment for a lead performer.

Moreover, instead of using the adapter-based finetuning methods [16, 17, 19], we apply the straightforward "direct finetuning" approach to accommodate the domain shift from general instrumental music (on which Music-Gen was trained) to the intended backing track music. We leave the use of adapter-based finetuning as future work. To make our approach suited for operations on consumer-grade GPUs, we propose a mechanism referred to as "jump finetuning" instead of finetuning the full MusicGen model.

We present a comprehensive performance study involving objective and subjective evaluation using two public-domain datasets, MUSDB18 [20] and RWC-pop-100 [21]. Our evaluation demonstrates MusiConGen's enhanced ability to offer nuanced temporal control, surpassing the original MusicGen model in producing music that aligns more faithfully with the given conditions.

The contributions of this work are two-fold. First, to our best knowledge, this work presents the first Transformer-based text-to-music generation model that follows user-provided rhythm and chords conditions, requiring no reference audio signals. Second, we present efficient training configuration allowing such a model to be built by finetuning the publicly-available MusicGen model with customer-level GPU, specifically 4x RTX-3090 in all our experiments. We open-source the code, checkpoint, and information about the training data of MusiConGen on GitHub. [1]

2. BACKGROUND

2.1 Codec Models for Audio Representation

In contemporary music generation tasks, audio signals are typically compressed into more compact representations

using two main methods: Mel spectrograms and codec tokens. Mel spectrograms provide a two-dimensional time-frequency representation, adjusting the frequency axis to the Mel scale to better align with human auditory perception. Codec tokens, on the other hand, are often residual vector quantization (RVQ) tokens that are encoded from audio signals by a codec model [13–15]. Following MusicGen, we employ in our work the Encodec (32k) [14] as the pretrained codec model to encode audio data at a sample rate of 32,000 Hz. This Encodec model comprises 4 codebooks, each containing 2,048 codes, and operates at a code frame rate f_s of 50 Hz.

2.2 Classifier-Free Guidance

Classifier-free guidance [22] is a technique initially developed for diffusion models in generative modeling to enhance the quality and relevance of the outputs without the need for an external classifier. This approach involves training the generative model in both a conditional and an unconditional manner, combining the output score estimates from both methods during the inference stage. The mathematical expression is as $\nabla_x \log \tilde{p}_\theta(x|c) = (1-\gamma)\nabla_x \log p_\theta(x) + \gamma \nabla_x \log p_\theta(x|c)$. Here, γ represents the guidance scale, which adjusts the influence of the conditioning information. We perform a weighted average of $f_\theta(x,c)$ and $f_\theta(x)$ when sampling from the output logits.

2.3 Pretrained MusicGen Model

The pretrained model used in our study is a MusicGen model with 1.5B parameters, equipped with melody control (i.e., MusicGen-Melody). The melody condition employs a chromagram of 12 pitch classes at a frame rate $f_\mathcal{M}$, denoted as $\mathcal{M} \in \mathbb{R}^{T_{f_\mathcal{M}} \times 12 \times 1}$, derived from the linear spectrogram of the provided reference audio. For text encoding, the model leverages the FLAN-T5 [23] as a text encoder to generate conditioning text embeddings, represented as $\mathcal{T} \in \mathbb{R}^{T_{t5} \times d_{t5} \times 1}$. Both the melody and text conditions undergo linear projection into a D-dimensional space before being prepended to the input audio embedding. Regarding the input audio for training, audio signals are initially encoded into RVQ tokens, $X_{rvq} \in \mathbb{R}^{T_{f_s} \times 1 \times 4}$, using the pretrained Encodec model. These tokens are then formatted into a "delay pattern" [2], maintaining the same sequence length. Subsequently, an embedding lookup table, $W_{emb} \in \mathbb{R}^{N \times D \times 4}$, where N represents for numbers of codes in a codebook, is used to represent the associated codes, summing contributions from each codebook of X_{rvq} to form the audio embedding $X_{emb} \in \mathbb{R}^{T_{f_s} \times D \times 1}$. The input representation is then fed to the self-attention layers via additive sinusoidal encoding.

3. METHODOLOGY

Our method seeks to efficiently finetune the foundational MusicGen model using time-varying symbolic rhythm and chord conditions as guiding conditions. To achieve this, we must carefully consider both the representation of these conditions and the finetuning mechanism as follows:

---

[1] `https://github.com/Cyan0731/MusiConGen`
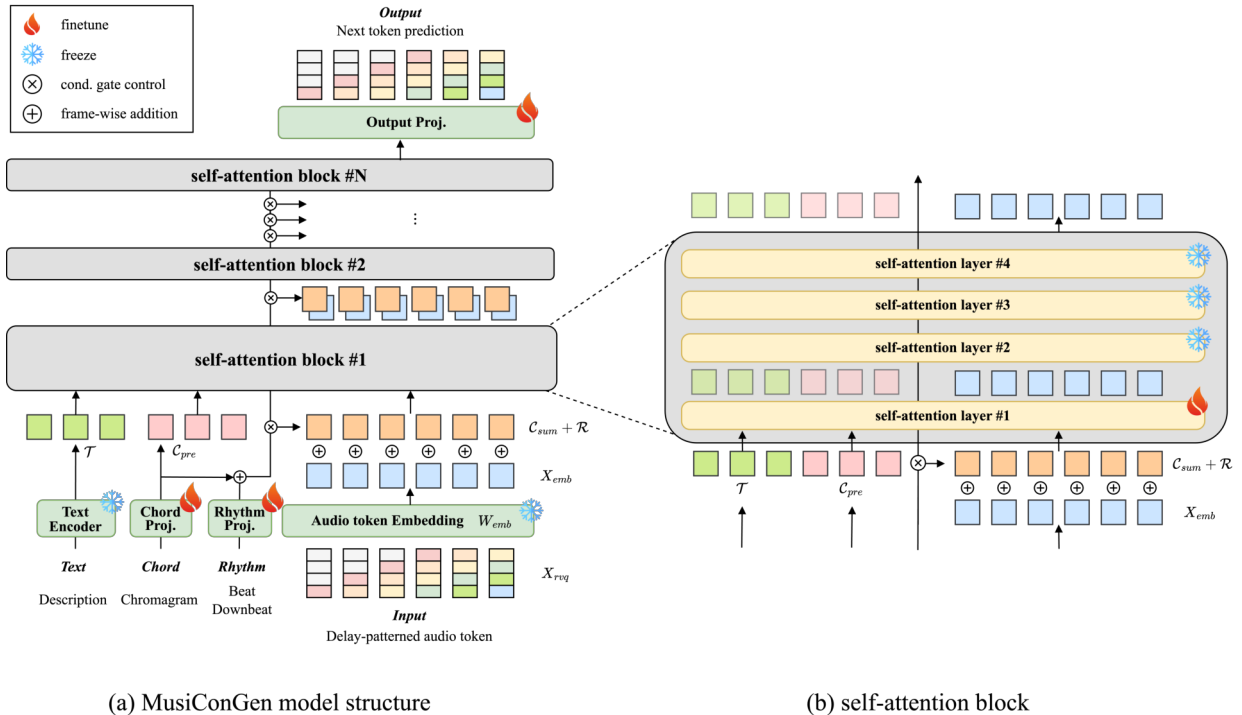
(a) MusiConGen model structure

(b) self-attention block

Figure 1. The model structure of MusiConGen and the self-attention block. a) MusiConGen takes text $\mathcal{T}$, downsampled chord $\mathcal{C}_{pre}$ as prepended condition and frame-wise chord $\mathcal{C}_{sum}$ and rhythm $\mathcal{R}$ as additive condition. The addition operation of frame-wise conditions to each self-attention block is regulated by the condition gate control ($\otimes$). b) Each self-attention block consists of four layers. In our proposed model, only the first layer is finetuned, which is also called jump finetuning.

3.1 Representing Temporal & Symbolic Conditions

Chords. For chord condition, we employ two methods. The first **prepend** method is similar to the melody control method of MusicGen, denoted as $\mathcal{C}_{pre} \in \mathbb{R}^{T_{f_{\mathcal{M}}} \times 12}$ where $\mathcal{C}_{pre}$ maintains the same resolution (i.e. frame rate $f_{\mathcal{M}}$ and sequence length) as MusicGen's melody condition $\mathcal{M}$. This allows us to utilize the pretrained melody projection weights from MusicGen as initial weights. Furthermore, we have noted that chord transitions can lead to asynchronization issues. To address this, we introduce a second **frame-wise** chord condition, $\mathcal{C}_{sum} \in \mathbb{R}^{T_{f_s} \times 12 \times 1}$, which matches the resolution of the audio codec tokens, thus providing a solution for the synchronization problem.

Rhythm. To control rhythm, we derive conditions from both the beat and the downbeat. The beat represents the consistent pulse within a piece of music, and the downbeat signifies the first and most emphasized beat of each measure, forming the piece's rhythmic backbone. We encode beat and downbeat information into one-hot embedding each at a frame rate of f_s. For the beat embedding, a soft kernel is applied to allow for a tolerance of 70ms. Subsequently, the beat and downbeat arrays are summed to yield the **frame-wise** rhythm condition $\mathcal{R} \in \mathbb{R}^{T_{f_s} \times 1}$.

3.2 Finetuning Mechanisms

The finetuning mechanism we employ consists of two parts: 1) jump finetuning, and 2) an adaptive in-attention mechanism. As illustrated in Figure 1, our proposed model activates condition gates at the "block" level, treating four consecutive self-attention layers as a block.

Jump finetuning is designed to specifically target the *first* self-attention layer *within each block* for finetuning, while freezing the remaining three self-attention layers of the same block, as shown in Figure 1 (b). Doing so reduces the number of parameters of finetuning while maintaining the flexibility to learn to respond to the new conditions by refining the first self-attention layer per block.

The adaptive in-attention mechanism is designed to improve control over chords and rhythm. It is an adaptation of the in-attention technique of MuseMorphose [18], whose main idea is to augment every intermediate output of the self-attention layers with copies of the condition. Unlike the original implementation that augment all the self-attention layers, we selectively apply it to the first three-quarters of self-attention blocks (e.g., for a model with 12 blocks, in-attention is applied to first 9 blocks) to relax the control in the last few blocks for better balancing on rhythm and chords. This leads to better result empirically, as will be shown in Section 5.2 and Table 3.

4. EXPERIMENTAL SETUP

4.1 Datasets

We finetuned the model using a dataset of ~250 hours backing track music sourced from YouTube, comprising 5K songs across five genres: Rock, Funk, Jazz, Blues, and Metal, with 1K songs per genre. After preprocessing (see Section 4.2), the training data contained 80,871 clips.

For evaluation, we used the rhythm and chords from two public-domain datasets—MUSDB18 [20] and RWC-pop-100 [21]. For MUSDB18, the rhythm and chords are *extracted* from the audio signals, so this dataset reflects the case where the condition signals are from a *reference audio*. There are 150 songs with four isolated stems: vocal, bass, drum, and others. For each song, we dropped the vocals and divided the mix of the remaining tracks into 30-second clips, resulting in a total of 1,089 clips.

The RWC comprises 100 Japanese pop songs with human annotated chord progressions and BPM labels. We simply use the *human labels* as the conditions here, reflecting the case where the condition signals are user provided *in a text-like format*. We similarly divided each song into 30-second clips, leading to 755 clips in total.

4.2 Dataset Pre-processing Details

The training and evaluation datasets consist of full-song data, with durations ranging from 2 to 5 minutes per song. Below are the preprocessing details for each type of input:

Audios: All audio data have vocals removed. For the training and RWC dataset, we employed the source separation model Demucs [24, 25] to eliminate the vocal stem. In the MUSDB18 dataset, which already features isolated stems, we combined the bass, drum, and others stems to form the dataset. Each song was segmented into 30-second clips, ensuring each clip starts at a downbeat.

Descriptions: For the training set, the text prompts were simply extracted from the titles of the corresponding YouTube videos. For the two evaluation datasets, we tasked ChatGPT [26] to generate 16 distinct text prompts, covering the five genres included by the training set. Here is an example—"A smooth acid Jazz track with a laid-back groove, silky electric piano, and a cool bass, providing a modern take on Jazz. Instruments: electric piano, bass, drums." At inference time, we randomly selected one of the 16 text prompts in a uniform distribution.

Chords: The RWC dataset comes with ground truth labeled chords. For both the training set and MUSDB18, we used the BTC model [27] as the chord extraction model to predict symbolic chords with time tags for each clip. The detailed chord quality extends to the seventh note. We then translated the extracted chord symbols with time tags into a 12-pitch chromagram in the order of C, C#, ..., B. The chromagram's frame rate for the frame-wise condition $\mathcal{C}_{sum}$ is f_s, and for the prepend condition $\mathcal{C}_{pre}$ it is $f_{\mathcal{M}}$.

Rhythm: Except for RWC, beat and downbeat were extracted using the RNN+HMM model [28] from the Madmom library [29]. The timing format for beats and downbeats was transformed into a one-hot representation matching the audio token frame rate f_s. A soft kernel was applied to the one-hot beat array to create a softened beat array. The rhythm representation $\mathcal{R}$ was the frame-wise summation of the softened beat array and downbeat array.

4.3 Training Configuration

The proposed rhythm and chord-conditioned Transformer was built upon the architecture of the medium-sized (1.5B)

MusicGen-melody, featuring $L = 48$ self-attention layers with dimension $D = 1,536$ and 24 multi-head attention units. The condition dropout rate is 0.5 and guidance scale is set to be $\gamma = 3$ for classifier-free guidance. We finetuned only a quarter of the full model, which corresponds to 352 million parameters, while keeping both the audio token embedding lookup table and the FLAN-T5 text encoder frozen. The training involved 100K finetuning steps, carried out over approximately 2 days on 4 RTX-3090 GPUs, with a batch size of 2 per GPU for each experiment.

4.4 Objective Evaluation Metrics

We employed metrics to evaluate controllability of chords and rhythm, textual adherence and audio fidelity. For the first two metrics, we used the rhythm and chord conditions from a clip in a evaluation dataset to generate music (along with a text prompt generated by ChatGPT; see Section 4.2), applied the Madmom and BTC models on the generated audio to estimate beats and chords, and evaluated how they reflect the given conditions. See Figure 2 for examples.

Chord. We used the *mir_eval* [30] package to measure 3 different degrees of frame-wise chord correctness: **majmin**, **triads** and **tetrads**. The majmin function compares chords in major-minor rule ignoring chord qualities outside major/minor/no-chord. The triads function compares chords along triad (root & qulaity to #5), while the tetrads compares chords along tetrad (root & full quality).

Rhythm F1 measurement follows the standard methodology for beat evaluation. We measured the beat accuracy also via *mir_eval*, assessing the alignment between the beat timestamps of the generated music and the reference rhythm music data, with a tolerance window of 70ms.

CLAP [31, 32] score examines the textual adherence by the cosine similarity between the embedding of the text prompt and that of the generated audio in a text-audio joint embedding space learned by contrastive learning. Here, we used the LAION CLAP model trained for music [33], `music_audioset_epoch_15_esc_90.14.pt`.

FAD is the Fréchet distance between the embeddings distribution from a set of reference audios and that from the generated audios [34, 35]. The metric represent how realistic the generated audios are compared to the given reference audios. The audio encoder of FAD we used is VGGish [36] model which trained on an audio classification task. The reference set of audios was from MUSDB18 or RWC depending on the evaluation set.

4.5 Subjective Evaluation Metrics

We also did a listening test to evaluate the followings aspects: text relevance, rhythm consistency, and chord relevance. Text relevance concerns how the generated audio clips reflect the given text prompts. Rhythm consistency is about how steady the beats is within an audio clip. (We found that, unlike the case of objective evaluations, minor out-of-sync beats at the beginning of a clip were deemed acceptable here perceptually.) Chord relevance concerns how a generated clip follows the given chord progressions.

Model	Evaluation dataset	Rhythm F-measure(%)	Chord			FAD	CLAP
			majmin(%)	triads(%)	tetrads(%)		
proposed	MUSDB18	**69.76**	67.03	66.19	56.91	**1.29**	0.34
($\mathcal{C}_{pre}+\mathcal{C}_{sum}+\mathcal{R}$)	RWC	**79.40**	**73.03**	**68.42**	**54.12**	**0.96**	0.34
chords only	MUSDB18	39.47	**73.25**	**72.29**	**60.89**	1.91	0.34
($\mathcal{C}_{pre}+\mathcal{C}_{sum}$)	RWC	49.85	**73.30**	**68.50**	50.66	2.18	0.34
rhythm only	MUSDB18	61.37	5.84	5.76	3.84	1.95	0.32
($\mathcal{R}$)	RWC	58.39	5.40	5.08	2.90	2.67	0.32
no frame-wise chords	MUSDB18	61.68	57.39	56.65	47.17	1.44	0.35
($\mathcal{C}_{pre}+\mathcal{R}$)	RWC	69.30	60.95	57.19	44.21	1.29	0.35
baseline	MUSDB18	26.14	53.13	52.31	44.83	2.01	0.34
(no finetuning; $\mathcal{M}$ for $\mathcal{C}_{pre}$)	RWC	30.67	51.90	48.54	35.81	2.30	0.35

Table 2. Objective evaluation results for models with different conditions on two different test sets MUSDB18 and RWC. With the proposed condition representation, we can achieve better performance both in rhythm and chord controls.

Model	Evaluation dataset	Rhythm F-measure(%)	Chord			FAD	CLAP
			majmin(%)	triads(%)	tetrads(%)		
proposed	MUSDB18	**69.76**	67.03	66.19	56.91	**1.29**	0.34
(jump+adaptive in-attn)	RWC	**79.40**	**73.03**	**68.42**	**54.12**	**0.96**	0.34
ablation 1	MUSDB18	42.28	**71.06**	**70.21**	**61.58**	1.39	0.36
(jump finetuning only)	RWC	53.14	**76.04**	**71.33**	**57.52**	1.27	0.36
ablation 2	MUSDB18	67.23	66.47	65.60	56.37	1.59	0.35
(jump+full in-attn)	RWC	71.13	64.82	60.77	48.07	1.47	0.35
finetuned baseline	MUSDB18	40.15	55.65	54.88	45.52	1.94	0.36
(jump only; no $\mathcal{C}_{sum}$ no $\mathcal{R}$)	RWC	49.25	56.49	52.66	38.07	2.24	0.36

Table 3. Objective evaluation results for models trained with different finetuning mechanisms. We see that the proposed jump finetuning with adaptive (partial) in-attention achieves better result on rhythm and chord controls.

5. EXPERIMENTAL RESULTS

5.1 Objective Evaluation: Temporal Conditions

We assessed the audio generated under various condition combinations applied to the training model, including the proposed method and its **ablations** with either chord- or rhythm-only as the temporal condition, or using both but without the frame-wise chord condition. The finetuning configurations and mechanisms for these models were the same. Moreover, we considered the **baseline** as follows. The pretrained MusicGen-melody model originally processes text and melody conditions $\mathcal{T}, \mathcal{M}$. We simply used the prepend chord condition $\mathcal{C}_{pre}$ as input to the linear projection layers originally pretrained to take the melody condition, without finetuning the entire model at all. In addition, we appended to the end of the text prompt BPM information (e.g., "at BPM 90") as the rhythm condition.

Result shown in Table 2 leads to many findings. Firstly, a comparison between the result of the proposed model (first row) and the baseline (last row) demonstrates nicely the effectiveness of the proposed design. The proposed model leads to much higher scores in almost all the met-

rics. Moreover, it performs similarly well for the two evaluation datasets, suggesting that MusiConGen can deal with both conditions extracted from a reference audio signals or provided by creators in a symbolic text-like format.

Secondly, although the baseline model does not perform well, it still exhibits some level of chord control, showing the knowledge of melody can be transferred to chords.

Finally, from the ablations (middle three rows), chord-only and rhythm-only did not work well for rhythm and chord control respectively, which is expected. Compared to the proposed model, excluding per-frame chord condition degrades both chord and rhythm controllability, showing that chord and rhythm are interrelated.

5.2 Objective Evaluation: Finetuning Mechanisms

Besides the proposed finetuning method, we evaluated the following alternatives. **Finetuned baseline** is a baseline model that was finetuned using the prepended chords ($\mathcal{C}_{pre}$) instead of melody $\mathcal{M}$ the frame-level conditions, employing the jump finetuning mechanism but no in-attention. **Jump finetuning without in-attention** (ablation 1) and **jump finetuning with full in-attention** (abal-
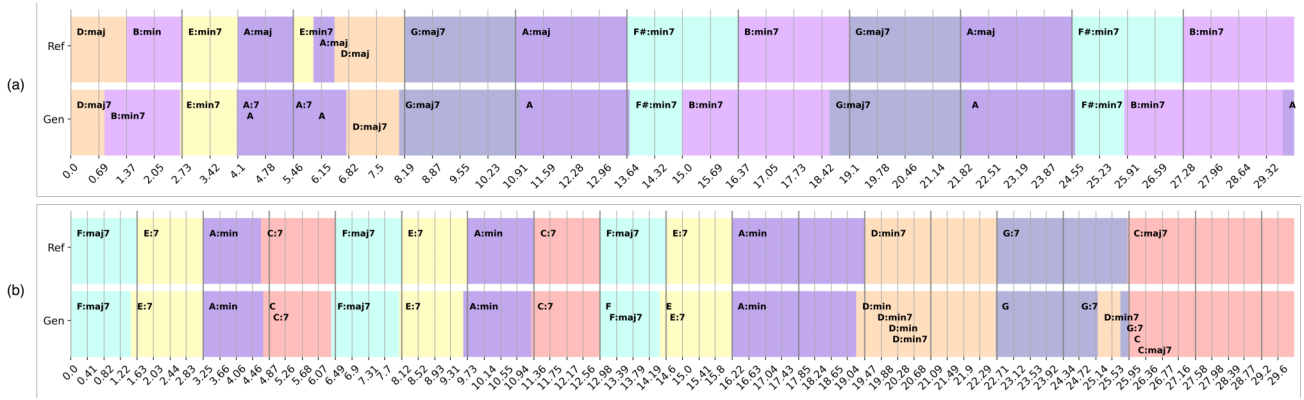
Figure 2. Comparison on chord progression and beats of ground truth and generated samples, using the conditions from RWC. For each example (a) or (b), the top row is ground truth chords and the bottom row is extracted chords from generated samples. The thick and light gray lines indicate the times of the downbeat and the beat, respectively.
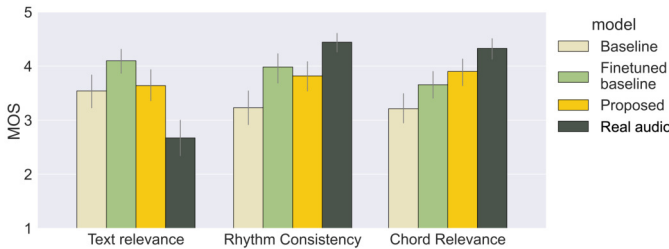

Figure 3. Subjective evaluation of condition controls—5-scale mean opinion score with 95% confidence interval.

ation 2) are ablations which use full conditions (prepended chord $\mathcal{C}_{pre}$, frame-wise chord $\mathcal{C}_{sum}$, and rhythm $\mathcal{R}$), but we either dropped in-attention entirely, or employed in-attention to every self-attention block, instead of only the first three-quarter blocks as done by the proposed method.

The result is tabulated in Table 3. Among the four methods, the proposed method leads to the best rhythm control and very competitive chord control. Comparing the results of the proposed method and the two ablations reveals a trade-off in rhythm and chord control when we go from no in-attention, adaptive (partial) in-attention, to full in-attention. The proposed method strikes an effective balance between rhythm and chord controls.

Comparing the last row of Table 2 and that of Table 3 shows that the finetuned baseline outperforms the baseline (with no finetuning at all) mainly in the rhythm control. This is notable as the finetuned baseline is actually trained with only the prepend chord condition $\mathcal{C}_{pre}$, not using the rhythm condition $\mathcal{R}$, suggesting again the interrelation of chord and rhythm. Moreover, although the finetuned baseline is better than the baseline, it is still much inferior to the proposed method in both chord and rhythm controls.

5.3 Subjective Evaluation

We evaluated three models in the listening test: the baseline, the finetuned baseline, and the proposed model. Each model generates a music clip using the ChatGPT-generated text prompts, along with the BPM and chords

from the RWC dataset, namely considering text-like symbolic rhythm and chord conditions. Besides the audios generated by the three models, we also included real audios from the RWC dataset as the **real audio**. We note that the real audios would have perfect rhythm and chord controllability (for they are where the conditions are from), but the textual adherence would be bad because RWC songs are J-Pop rather than any of the five genres (i.e., Rock, Funk, Jazz, Blues, and Metal) described by the text prompts.

We had 23 participants in the user study, 85% of whom have over three years of musical training. Each time, we displayed the given text, rhythm and chord conditions, and asked a participant to rate the generated audio and the real audio (anonymized and in random order) on a five-point Likert scale. The result is shown in Figure 3.

Several findings emerged. Firstly, the proposed model demonstrated superior chord control compared to the other two models, although it still fell short of matching the real audio. Secondly, the proposed model has no significant advantage on rhythm consistency against the finetuned baseline. As suggested by the examples on our demo page, we found that being on the precise beat onset does not significantly impact rhythm perception. Thirdly, our model had lower text relevance than the finetuned baseline, suggesting that our model may have traded text control for increased temporal control of rhythm and chords.

6. CONCLUSION AND FUTURE WORK

This paper has presented conditioning mechanisms and finetuning techniques to adapt MusicGen for better rhythm and chord control. Our evaluation on backing track generation shows that the model can take condition signals from either a reference audio or a symbolic input. For future work, our user study shows room to further improve the rhythm and chord controllability while keeping the text relevance. This might be done by scaling up the model size, better language model, or audio codecs. It is also interesting to incorporate additional conditions, such as symbolic melody, instrumentation, vocal audio, and video clips.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] A. Agostinelli, T. I. Denk, Z. Borsos, J. H. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. H. Frank, "MusicLM: Generating music from text." *arXiv preprint arXiv:2301.11325*, 2023.

[2] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," in *Proc. NeurIPS*, 2023.

[3] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank, J. Engel, Q. V. Le, W. Chan, Z. Chen, and W. Han, "Noise2Music: Text-conditioned music generation with diffusion models," *arXiv preprint arXiv:2302.03917*, 2023.

[4] H. Liu, Q. Tian, Y. Yuan, X. Liu, X. Mei, Q. Kong, Y. Wang, W. Wang, Y. Wang, and M. D. Plumbley, "AudioLDM 2: Learning holistic audio generation with self-supervised pretraining," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2871–2883, 2024.

[5] P. Li, B. Chen, Y. Yao, Y. Wang, A. Wang, and A. Wang, "JEN-1: Text-guided universal music generation with omnidirectional diffusion models," *arXiv preprint arXiv:2308.04729*, 2024.

[6] L. Lin, G. Xia, J. Jiang, and Y. Zhang, "Content-based controls for music large language modeling," *arXiv preprint arXiv:2310.17162*, 2023.

[7] S.-L. Wu, C. Donahue, S. Watanabe, and N. J. Bryan, "Music ControlNet: Multiple time-varying controls for music generation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2692–2703, 2024.

[8] K. Chen, Y. Wu, H. Liu, M. Nezhurina, T. Berg-Kirkpatrick, and S. Dubnov, "MusicLDM: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies," in *Proc. ICASSP*, 2024.

[9] J. Melechovsky, Z. Guo, D. Ghosal, N. Majumder, D. Herremans, and S. Poria, "Mustango: Toward controllable text-to-music generation," in *Proc. NAACL*, 2024.

[10] Y. Zhang, Y. Ikemiya, G. Xia, N. Murata, M. Martínez, W.-H. Liao, Y. Mitsufuji, and S. Dixon, "MusicMagus: Zero-shot text-to-music editing via diffusion models," in *Proc. IJCAI*, 2024.

[11] F.-D. Tsai, S.-L. Wu, H. Kim, B.-Y. Chen, H.-C. Cheng, and Y.-H. Yang, "Audio Prompt Adapter: Unleashing music editing abilities for text-to-music with lightweight finetuning," in *Proc. ISMIR*, 2024.

[12] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[13] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 30, p. 495–507, 2021.

[14] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[15] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, "High-fidelity audio compression with improved RVQGAN," in *Proc. NeurIPS*, 2023.

[16] R. Zhang, J. Han, C. Liu, P. Gao, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, and Y. Qiao, "LLaMA-Adapter: Efficient fine-tuning of language models with zero-init attention," *arXiv preprint arXiv:2303.16199*, 2023.

[17] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proc. ICCV*, 2023.

[18] 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.

[19] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "PEFT: State-of-the-art parameter-efficient fine-tuning methods," 2022. [Online]. Available: https://github.com/huggingface/peft

[20] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," 2017. [Online]. Available: https://sigsep.github.io/datasets/musdb.html

[21] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, classical, and jazz music databases," in *Proc. ISMIR*, 2002. [Online]. Available: https://staff.aist.go.jp/m.goto/RWC-MDB/

[22] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *Proc. NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[23] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *arXiv preprint arXiv:2210.11416*, 2022.

[24] S. Rouard, F. Massa, and A. Défossez, "Hybrid Transformers for music source separation," in *Proc. ICASSP*, 2023. [Online]. Available: https://github.com/facebookresearch/demucs

[25] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proc. ISMIR 2021 Workshop on Music Source Separation*, 2021.

[26] J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng *et al.*, "Introducing ChatGPT," 2022.

[27] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, "A bi-directional Transformer for musical chord recognition," in *Proc. ISMIR*, 2019. [Online]. Available: https://github.com/jayg996/BTC-ISMIR19

[28] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proc. ISMIR*, 2016, pp. 255–261.

[29] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python audio and music signal processing library," in *Proc. ACM Multimedia*, 2016, pp. 1174–1178. [Online]. Available: https://github.com/CPJKU/madmom

[30] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "Mir_eval: A transparent implementation of common MIR metrics," in *Proc. ISMIR*, 2014, pp. 367–372. [Online]. Available: https://github.com/craffel/mir_eval

[31] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *Proc. ICASSP*, 2023.

[32] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token-semantic audio Transformer for sound classification and detection," in *Proc. ICASSP*, 2022.

[33] "LAION CLAP." [Online]. Available: https://github.com/LAION-AI/CLAP

[34] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv:1812.08466*, 2018.

[35] A. Gui, H. Gamper, S. Braun, and D. Emmanouilidou, "Adapting Fréchet audio distance for generative music evaluation," in *Proc. ICASSP*, 2024, pp. 1331–1335.

[36] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *Proc. ICASSP*, 2017.