VERSATILE SYMBOLIC MUSIC-FOR-MUSIC MODELING VIA FUNCTION ALIGNMENT

Anonymous Authors
Anonymous Affiliations
anonymous@ismir.net

ABSTRACT

Many music AI models learn a map between music content and human-defined labels. However, many annotations, such as chords, can be naturally expressed within the music modality itself, e.g., as sequences of symbolic notes. This observation enables both understanding tasks (e.g., chord recognition) and conditional generation tasks (e.g., chord-conditioned melody generation) to be unified under a music-for-music sequence modeling paradigm. In this work, we propose parameter-efficient solutions for a variety of symbolic music-for-music tasks. The high-level idea is that (1) we utilize a pretrained Language Model (LM) for both the reference and the target sequence and (2) we link these two LMs via a lightweight adapter. Experiments show that our method achieves superior performance among different tasks such as chord recognition, melody generation, and drum track generation. All demos, code and model weights are publicly available [1] .

1. INTRODUCTION

Many foundational tasks in music AI, such as music information retrieval (MIR) and conditional music generation, have traditionally been formulated as mappings between music and labels: either from music to task-specific annotations (e.g., chord recognition), or from descriptive conditions to music (e.g., chord-conditioned melody generation). While these tasks have long been treated separately, a key observation is that in many cases, the "labels" themselves can also be represented in the *same music modality*—for example, as note sequences. This suggests a unifying perspective: a wide range of MIR and generation tasks can be reformulated as sequence-to-sequence problems within the music domain. We refer to this formulation as *music-for-music* modeling.

To achieve versatile music-for-music modeling in a sample-efficient way, we apply *knowledge transfer* to pretrained foundational Language Models (LMs) using a light-parameterized adaptor. As illustrated in Fig. 1(a)-(b), many existing methods such as probing [1–3] and prefix tuning [4–6] transfer knowledge of foundation models to downstream tasks by adapting them to new input or output, but the knowledge resides in only one language—either the LM of source $\mathbf{x}$ or the LM of target $\mathbf{y}$. In contrast, our method distills knowledge from *both* LMs via aligning them in a layer-wise manner, as shown in Fig. 1(c).

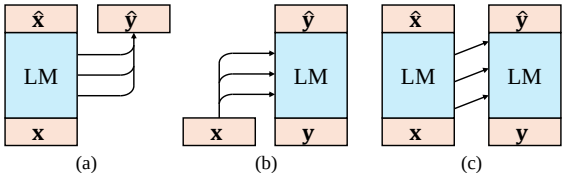

Figure 1. Three types of sequence-to-sequence models by knowledge transfer from pretrained LMs. $\mathbf{x}$ and $\mathbf{y}$ are input sequences and $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are predictions, possibly right-shifted due to the autoregressive targets. (a) Probing; (b) Prefix tuning; (c) Function alignment ($\mathbf{x} \rightarrow \mathbf{y}$).

At the methodology level, our approach is inspired by *function alignment* [7], a recently proposed theory of mind that attributes the emergence of intelligence to the dynamic synergy among interacting agents. In our work, we contribute two concrete implementations of this idea—treating two language models (LMs) as agents and creating synergy through Parameter-Efficient Fine-Tuning (PEFT).

The first approach introduces a trainable cross-attention layer between two separately pretrained LMs. The second, more concise solution, uses a lightweight self-attentive adapter applied to concatenated input-output sequences within a *single shared LM*—a strategy applicable when both input and output share the same vocabulary. We show the effectiveness of both implementations using experiments on both generative and analysis tasks, including: (1) chord-conditioned melody generation, (2) melody-conditioned chord generation, (3) drum-conditioned song generation, (4) song-conditioned drum generation and (5) few-shot symbolic music analysis.

The main contribution of this paper is as follows:

1. We achieve versatile *music-for-music* modeling, unifying a broad range of music understanding and controllable generation tasks under a shared framework.
2. At the methodological level, we are the first to introduce function alignment—a recently proposed theory of mind emphasizing the synergy among agents—into the domain of music AI, offering a

[1] https://ismir2025submission25.github.io/function-alignment/

novel perspective on modeling music sequence-to-sequence tasks.

3. While function alignment remains at a theoretical level, we present two concrete, *parameter-efficient* implementations in the context of modern language models: one via cross-attentive adapters across two LMs, and another via a self-attentive adapter within a shared LM. We demonstrate the effectiveness of both approaches through theoretical analysis and empirical validation.

2. RELATED WORKS

2.1 Music Foundation Models

Since the invention of the Transformer architecture [8], transformer-based language models have become the mainstream of music foundation models on multiple modalities, including audio [9–12], symbolic [13–21] and text-based music representation [22]. In addition to autoregressive models, masked language models [2, 23] and diffusion models [24–29] and flow-based models [30] can also be used as foundation models, but we focus on autoregressive models in the literature review.

For symbolic music, music transformer [13] is an early work to adopt the transformer architecture to music. Some follow-up works try to design a better representation of the music content. For example, pop music transformer imposes a metrical structure in the data representation [15]. MuPT trains transformers on their proposed synchronized multi-track ABC notation [20]. Other works aim to introduce controllability to the generative model. Musecoco generates the music score from text [14]. METEOR performs melody-aware orchestral music generation with texture control [16]. SymPAC trains symbolic generation models from transcribed audio data with chord, section, and instrument controls [17]. Zhang *et al.* improve generation discriminators to better follow rhythm and melody conditions [18]. The Theme Transformer [19] uses a short theme condition for generation. MuseBarControl generates music with fine-grained control to the bar level [21].

2.2 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) methods add lightly parameterized adapters to large pretrained models. Compared to full-parameter fine-tuning, PEFT requires significantly less computation and training data. Existing methods include appending task-specific prefixes to input sequences [4,31], injecting low-rank adaptation (LoRA) to linear layers [32], and adding learnable hidden states to the self-attention blocks [5, 33].

PEFT has been applied to music foundation models to support new tasks. Coco-Mulla [6] and MusiConGen [34] both adapt MusicGen to follow content controls such as chord and rhythm. Additionally, AirGen enables MusicGen to infill segments based on content controls [35]. Instruct-MusicGen extends MusicGen for music editing by text instructions [36]. Audio Prompt Adapter extends AudioLDM2 for music editing following controls such as genre, timbre, and melody [37]. Ou *et al.* finetunes a symbolic language model for band arrangement, piano reduction, drum arrangement, voice separation, and more [38].

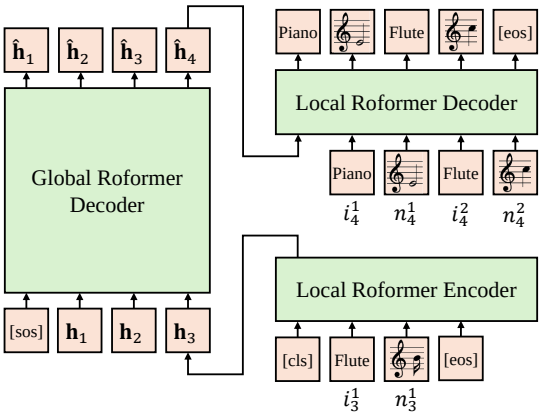

Figure 2. The architecture of the foundation model. The left side shows the global decoder. The right side shows the encoding of a single time step $\mathbf{x}_3 = \{i_3^1, n_3^1, [\text{eos}]\}$ and the decoding of the next step $\mathbf{x}_4 = \{i_4^1, n_4^1, i_4^2, n_4^2, [\text{eos}]\}$.

3. METHODOLOGY

3.1 Base Model

For this study, we choose the base model (the pretrained symbolic LM) with two main considerations. First, we do not wish to introduce *any* control in the pretraining stage, since we want to demonstrate the controllability using PEFT. We refain from using any annotation or metadata (i.e., chord, bar or text annotations) to pretrain the base model. Second, we want to adopt a data representation that can help the model align multiple sequences in time easily. Instead of using a MIDI event-like representation [13, 15, 39] where two time-aligned sequences might have a significant length difference, we use a fixed time step (a 16$^{\text{th}}$ note unit) for the input sequence.

Since multiple notes can occur at the same time step, we use a hierarchical scheme to compress (decompress) the note lists on the same time step with a local encoder (decoder), as shown in Fig. 2.

3.1.1 Data Representation

Formally, we represent a score sequence $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$ with a fixed time step of a 16$^{\text{th}}$ note. Since each time step may contain multiple note onsets, each $\mathbf{x}_t$ represents a list of N_t notes whose quantized onset time is the t-th 16$^{\text{th}}$ note (i.e., $\mathbf{x}_t$ is a simu-note [40] at time step t). We define

$$\mathbf{x}_t = \{i_t^1, n_t^1, i_t^2, n_t^2, ..., i_t^{N_t}, n_t^{N_t}, [\text{eos}]\} \quad (1)$$

where i_t^k is the instrument ID for the k-th note. We use the MIDI program number 0...127 for pitched instruments and $i_t^k = 128$ for drums. $n_t^k = 24p_t^k + d_t^k$ is a flattened representation of the k-th note's pitch p_t^k and duration d_t^k. p_t^k denotes the MIDI pitch from 0 to 127. $d_t^k \in \{0, ..., 23\}$ is the note duration quantized into 24 possible bins, $d_t^k = j$ corrsponds to a duration of b_j sixteenth

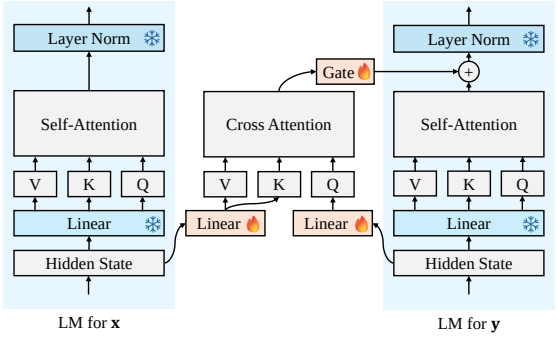

Figure 3. The architecture of a cross-attentive function alignment adapter. The fire icon denotes trainable parameters, and the snowflake icon denotes frozen parameters.

notes where $\mathbf{b} = [1, 2, 3, 4, 6, 8, 12, 16, 24, ..., 4096]$. [eos] is a special token marking the end of the list. All notes in $\mathbf{x}_t$ are sorted primarily by i_t^k and secondarily by n_t^k.

3.1.2 *Model Design*

We use a Roformer [41], a popular transformer architecture as the backbone model. The model architecture is shown in Fig. 2. Since our input sequence contains nested lists, we first encode each $\mathbf{x}_t$ with a local Roformer encoder:

$$[\mathbf{h}_t, \_] = \text{LocalEncoder}([\text{cls}], \mathbf{x}_t) \tag{2}$$

for all $t = 1...T$. Specifically, we prepend a [cls] token at the beginning of $\mathbf{x}_t$ and pass the sequence to the encoder. $\mathbf{h}_t$ is acquired from the output representation of the [cls] token. We then use a global Roformer decoder to autoregressively model the symbolic score:

$$\hat{\mathbf{h}}_t = \text{GlobalDecoder}(\mathbf{e}_{\text{sos}}, \mathbf{h}_{1...t-1}) \tag{3}$$

where $\mathbf{e}_{\text{sos}}$ is a learnable start-of-sentence (sos) embedding. Finally, a local Roformer decoder generates each note by

$$\hat{\mathbf{x}}_{t,j} = \text{LocalDecoder}([\text{sos}]_t, \mathbf{x}_{t,1...j-1}) \tag{4}$$

for all $t = 1...T$. Here, the embedding result of $\text{Emb}([\text{sos}]_t) := \hat{\mathbf{h}}_t$ passes the global state $\hat{\mathbf{h}}_t$ to the local decoder. $\mathbf{x}_{t,j}$ denotes the j-th token of list $\mathbf{x}_t$ (see Eqn. 1). The local decoder terminates when an end-of-sentence (eos) token is generated.

We will use $\mathbf{x}_t = \text{LM}(\mathbf{x}_{0...t-1})$ (or simply $\text{LM}(\mathbf{x})$) as a shorthand for the autoregressive model of sequence $\mathbf{x}$ through Eqs. 2-4. Here, $\mathbf{x}_0$ denotes the global start-of-sentence embedding $\mathbf{e}_{\text{sos}}$.

3.2 Parameter-Efficient Fine-Tuning

Our fine-tuning strategy leverages pretrained LMs for $\mathbf{x}$ and $\mathbf{y}$, connected via a parameter-efficient module. We present two variants: cross-attentive adapters for separate LMs, and self-attentive adapters for a shared LM. We apply both adapters to the backbone of the foundation model (the global decoder in Eqn. 3) only.

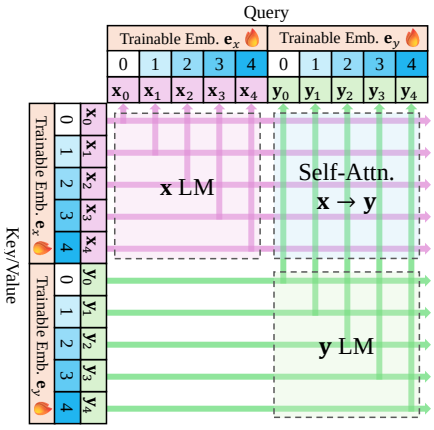

Figure 4. The architecture of a self-attentive function alignment adapter. Crossed vertical and horizontal arrows indicate the flow of information between the corresponding query and key/value pairs, while all other connections are masked by the autoregressive self-attention mechanism. The indices 0 through 4 represent the proposed positional embeddings for the concatenated sequence.

3.2.1 *Cross-attentive Function Alignment*

Our first approach is to use a cross-attention layer between the hidden layers of two LMs. A similar architecture has been adopted in language processing [42] and speech processing [43]. We refer to the design of [42] and show an adapted version in Fig. 3. For the l-th attention layer of $\text{LM}(\mathbf{y})$, the original self-attention is defined as:

$$\mathbf{h}_{\text{p}}^l = \text{SelfAttn}(\mathbf{W}_q^l \mathbf{z}_y^l, \mathbf{W}_k^l \mathbf{z}_y^l, \mathbf{W}_v^l \mathbf{z}_y^l) \tag{5}$$

where $\mathbf{z}_y^l$ denote the l-th layer hidden states for $\text{LM}(\mathbf{y})$ and $\mathbf{W}^l$ denotes pretrained weights. The adapted version can be written as:

$$\mathbf{h}_{\text{a}}^l = \mathbf{h}_{\text{p}}^l + g \cdot \text{CrossAttn}(\mathbf{U}_q^l \mathbf{z}_y^l, \mathbf{U}_k^l \mathbf{z}_x^l, \mathbf{U}_v^l \mathbf{z}_x^l) \tag{6}$$

where g is a zero-initialized trainable gate scaler. $\mathbf{U}^l$ are trainable parameters. Intuitively, this allows the query from $\text{LM}(\mathbf{y})$ to attend both to itself (self-attention) and to the condition from $\text{LM}(\mathbf{x})$ (cross-attention).

Besides the trainable cross-attention module, we also append LoRAs [32] to all $\mathbf{W}_q^l$ and $\mathbf{W}_v^l$ of both pretrained models $\text{LM}(\mathbf{x})$ and $\text{LM}(\mathbf{y})$, allowing the model to learn distinctive features of sequences $\mathbf{x}$ and $\mathbf{y}$.

3.2.2 *Self-attentive Function Alignment*

When $\mathbf{x}$ and $\mathbf{y}$ share the same pretrained LM, alignment becomes a special case: it can be achieved by concatenating their sequences and feeding them into a single model. The LM will first model $\mathbf{x}$ and predict $\mathbf{y}$ given $\mathbf{x}$ as a prefix.

This implies that prior PEFT methods [35, 38], which structure the condition and generated sequence within a single language model, can be viewed as broader forms of function alignment. We show that a simpler configuration is also effective and explain why it realizes function alignment.

When we directly concatenate two sequences $[\mathbf{x}, \mathbf{y}]$ and feed them to the decoder self-attention layer, we can decompose it into the self-attention of $\mathbf{x}$ and $\mathbf{y}$, and an extra component influencing $\mathbf{y}$ from $\mathbf{x}$, as shown in Fig. 4. Specifically, we have

$$\begin{aligned}[\mathbf{h}^l_{x\mathrm{a}}, \mathbf{h}^l_{y\mathrm{a}}] &= \text{SelfAttn}(\mathbf{W}^l_q[\mathbf{z}^l_x, \mathbf{z}^l_y], \mathbf{W}^l_k[\mathbf{z}^l_x, \mathbf{z}^l_y], \mathbf{W}^l_v[\mathbf{z}^l_x, \mathbf{z}^l_y]) \\ &= \text{SelfAttn}([\mathbf{Q}^l_x, \mathbf{Q}^l_y], [\mathbf{K}^l_x, \mathbf{K}^l_y], [\mathbf{V}^l_x, \mathbf{V}^l_y])\end{aligned} \tag{7}$$

for every layer l. In a single-head setting, we have $\text{SelfAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) := \text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d}+\mathbf{M})\mathbf{V}$, where d is the dimension of the key vectors and $\mathbf{M}$ is the autoregressive mask. We can rewrite Eqn. 7 by

$$\mathbf{h}^l_{x\mathrm{a}} = \text{SelfAttn}(\mathbf{Q}^l_x, \mathbf{K}^l_x, \mathbf{V}^l_x) \tag{8}$$

$$\begin{aligned}\mathbf{h}^l_{y\mathrm{a}} = &\frac{\mathbf{a}}{\mathbf{a}+\mathbf{b}}\text{SelfAttn}(\mathbf{Q}^l_y, \mathbf{K}^l_y, \mathbf{V}^l_y) \\ &+ \frac{\mathbf{b}}{\mathbf{a}+\mathbf{b}}\text{CrossAttn}(\mathbf{Q}^l_y, \mathbf{K}^l_x, \mathbf{V}^l_x)\end{aligned} \tag{9}$$

where $\mathbf{a} = \sum_j \exp\left[(\mathbf{Q}^l_y\mathbf{K}^l_y)_j/\sqrt{d}+\mathbf{M}\right]$ and $\mathbf{b} = \sum_j \exp\left[(\mathbf{Q}^l_y\mathbf{K}^l_x)_j/\sqrt{d}\right]$. Note that Eqn. 9 closely mirrors Eqn. 6 in form. Although the gating vectors $\mathbf{a}$ and $\mathbf{b}$ are not explicitly parameterized, we hypothesize that this design remains effective.

After concatenating $\mathbf{x}$ and $\mathbf{y}$, we reset $\mathbf{y}$'s positional embeddings to start from 0 to better preserve the pretrained behavior of $\text{LM}(\mathbf{y})$. To avoid token indistinguishability due to overlapping positions, we also add zero-initialized, learnable sentence embeddings $\mathbf{e}_x$ and $\mathbf{e}_y$ to their respective positional encodings, as shwon in Fig. 4.

Similar to cross-attentive adapters, a trainable LoRA module is also appended to the pretrained LM.

4. EXPERIMENTS

In the experiments, we first describe the hyperparameters and the pretraining scheme of our foundation model (Sec. 4.1). We evaluate our adapters on both generative and analysis tasks. We describe the tasks in Sec. 4.2 and models in Sec. 4.3. We then show the setting for subjective evaluation (Sec. 4.4) and objective evaluation (Sec. 4.5), and analyze the results in Sec. 4.6.

4.1 Model Pretraining

We use a Roformer with a 12-layer global decoder (hidden size 768, intermediate size 3072, 12 heads). The local encoder and decoder are smaller 3-layer Roformers (hidden size 768, intermediate size 768, 8 heads).

We pretrain our foundation model on the Los Angeles MIDI dataset [44], which contains approximately 405,000 MIDI files. As a score-based model, it relies on accurate beat annotations (inferred from tempo change events) for correct quantization. However, many files in the pretraining dataset contain incorrect tempo information.

To address this, we apply a rule-based filter. Normally, note onsets are not uniformly distributed across odd and even time steps. We compute the ratio of notes quantized to odd vs. even time steps. If the ratio falls within 0.5 ± 0.15 for every track, we assume it is poorly quantized and discard the song. This yields a cleaned subset of 357,279 files. During pretraining, we also apply a random pitch shift within $[-5, 6]$ semitones for data augmentation.

We set the global sequence length to $T = 384$ and cap the maximal polyphony by $N_t \leq 16$, clipping excess notes per time step. A batch size of 48 is used for pretraining. We train the model for 2,000,000 iterations using AdamW [45] with β=$(0.9, 0.999)$ and weight decay 0.01. We use a OneCycleLR [46] scheduler with a maximum LR 10^{-4} and 10,000 warm-up steps. Pretraining takes around 12 days on 4×A100 (40GB) GPUs.

4.2 Downstream Tasks

We evaluate the adaptor on different music generation and understanding task. Specifically, we have 3 sets of tasks:

- **Melody to chord** and **chord to melody**: we fine-tune the model on the Nottingham dataset [47] with a total of 1,020 songs. The model is asked to generate chords from a given melody or to generate a melody given a chord progression.
- **Drum to others** and **others to drum**: we fine-tune the model on a subset of 31,000 songs in the Los Angeles dataset with a drum track. The model is asked to generate the drum track given the full score of non-percussive instruments, or to generate other instruments given a drum track.
- **Few-shot symbolic music analysis**: we fine-tune the model on 93 songs in the RWC Pop dataset [48]. The model is asked to transcribe the chords and metrical structure given a symbolic pop music. We evaluate the results on symbolic chord recognition.

In each task, we perform a random 8:1:1 split for training, validation, and testing. For the drum-to-others and others-to-drum tasks, RWC Pop is used as an external test set.

4.3 Compared Models

We compare the performance of the following models, with slight hyperparameter adjustments to ensure comparable numbers of trainable parameters.

- **FA-Cross**: The base model is fine-tuned with a cross-attentive adapter (4 heads, hidden size 256), inserted every two layers of the global decoder. A LoRA with $r = 16, \alpha = 32$ is used on the query and value projectors of both LMs.
- **FA-Self**: The base model fine-tuned with a self-attentive adapter. A LoRA with $r = 64, \alpha = 128$ is used on the query and value projectors of both LMs.
- **Coco-Mulla**: The Cocomulla [6] adapter applied on the Roformer model. The adapter has a trainable positional encoding size of 384.

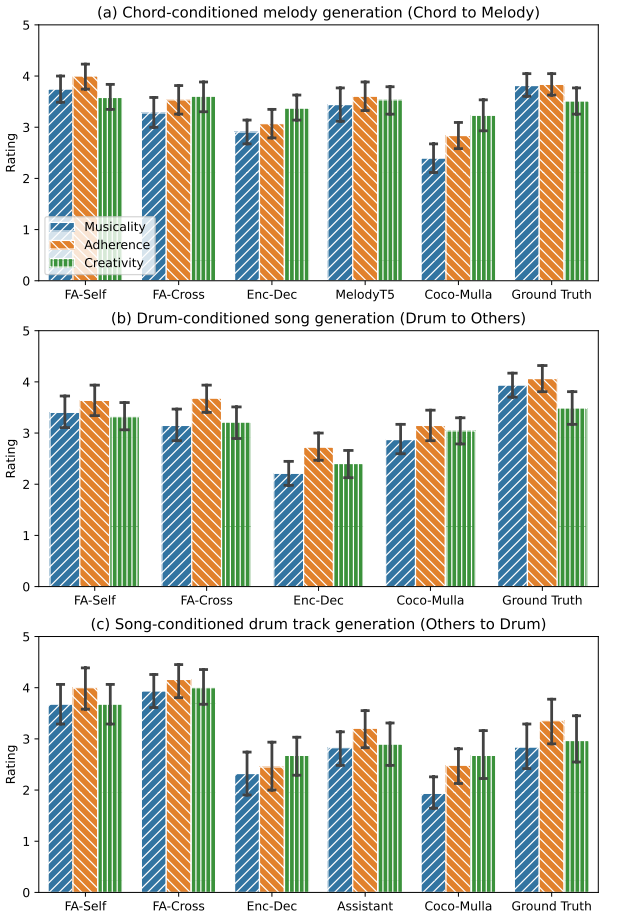

Figure 5. Subjective evaluation results. The error bars show the 95% confidence intervals of the true mean.

- **Prober**: A 2-layer Multilayer Perceptron (MLP) prober as used in [2]. The MLP layer uses a weighted sum of all layers' hidden states and has a hidden dimension of 768.
- **Enc-Dec**: A baseline trained from scratch with a small Roformer encoder-decoder (3 layers, hidden size 256, intermediate size 512, 4 heads for both encoder and decoder).
- **MelodyT5** [49]: an external baseline for the *melody to chord* and *chord to melody* tasks. The model is trained on 261K songs represented by ABC notations. We do not retrain this baseline.
- **Assistant** (Composers Assistant V2) [50]: an external baseline for the *others to drum* task. We do not retrain the baseline.

All modules are trained for up to 60,000 iterations with a fixed learning rate of 10^{-4} and a batch size of 8 on a single A100 GPU. Early stopping is applied if validation loss doesn't improve for 10 rounds (5,000 iterations).

4.4 Subjective Evaluation

For the three generative tasks (chord-to-melody, drum-to-others, and others-to-drums), we conducted a subjective evaluation via a user survey. We selected 8 songs from the test set (2 for chord-to-melody, 4 for drum-to-others, and 2 for others-to-drums). We asked participants to rate both the generated outputs and ground truth on a 5-point scale across the following metrics:

- **Musicality**: Does it sound good as music?
- **Adherence**: Does it respect and follow the input condition?
- **Creativity**: Given the input conditions, is it creative in its musical decisions?

We received a total of 54 answers, and the results are shown in Fig. 5.

	Chord to melody	**Melody to chord**	**Drum to others**	**Others to drum**
FA-Cross	1.4204 ±0.0986	1.4177 ±0.1042	2.0459 ±0.5598	1.8619 ±0.5633
FA-Self	**1.4116 ±0.1165**	**1.4104 ±0.0994**	**2.0222 ±0.6322**	**1.8402 ±0.5676**
Coco-Mulla	1.8016 ±0.1702	1.5996 ±0.1437	2.2027 ±0.6495	1.9860 ±0.6818
Enc-Dec	1.6113 ±0.1780	1.5067 ±0.1201	2.5830 ±0.9094	1.8765 ±0.5352
Ground Truth	1.3917 ±0.0982	1.3917 ±0.0982	2.0730 ±0.7118	2.0730 ±0.7118

Table 1. Test set perplexity on different downstream tasks.

4.5 Objective Evaluation

For the generative tasks by fine-tuned models, we report the generated results' perplexity on the Roformer base model on the test set. Since perplexity is inaccurate on long repetitive generations [51], we only calculate the perplexity using 8-bar generative results (128 steps) conditioned on 2-bar prompts (32 steps). The results are shown in Tab. 1.

For the melody to chord task, we report two additional metrics to compare with MelodT5. We first calculate the L_1 distance between the chromagram (chroma) of the predicted chords and the ground-truth chords. We also report the CTnCTR [52] metric between the melody and the generated chords. Since the test part of the Nottingham dataset has significant overlap with MelodyT5's training set, we perform a small pitch shift (up to 2 semitones) for all test songs to another commonly used key in the Nottingham dataset (e.g., C major to D major, A major to G major etc). The results are shown in Tab. 2.

For the music analysis task, we represent both the chord and the metrical labels by MIDI notes. The chord notes are represented by block notes using String Ensemble 1 (MIDI program 48). The bass note is placed in the range C3 to B3 (MIDI pitch 36-41), and other chord notes are stacked above them. We use a drum track to represent metrical labels. We use a bass drum note (MIDI pitch 35) to represent a downbeat and a snare drum note (MIDI pitch 38) for subsidiary strong beats. An 8-note infilling by closed hi-hat note (MIDI pitch 42) is also used.

For sequence-to-sequence modeling, the model predicts both tracks from the full MIDI input, and final chord labels are derived via template matching on the average of

	Chroma ↓	CTnCTR ↑
Ground Truth	0.0000±0.0000	0.9675±0.0324
FA-Cross	1.5690±0.7087	0.9113±0.0750
FA-Self	**1.2685±0.5024**	**0.9484±0.0415**
Coco-Mulla [6]	3.4613±0.5854	0.6647±0.1219
Seq2Seq	3.0044±0.5613	0.8387±0.0749
MelodyT5 [54]	3.0428±0.8694	0.8463±0.1036

Table 2. Objective evaluation results on unprompted melody to chord generation on the test split of the Nottingham dataset.

Model	Root ↑	Majmin ↑	Seventh ↑
Chorder [39]	0.7244	0.6760	0.3374
HMM [55]	**0.8386**	0.8169	0.6930
FA-Cross	0.8203	0.8455	0.6761
FA-Self	0.8275	**0.8693**	**0.6986**
Prober	0.8231	0.8370	0.6191
Seq2Seq	0.1786	0.1500	0.0378

Table 3. Evaluation results on symbolic chord recognition. The table shows the median result among the test split of the RWC Pop dataset.

16 generations. The exception is the prober, trained as a 25-class classifier (12 major, 12 minor, 1 no-chord). We evaluate using chord metrics (root, majmin, seventh) from the mir_eval package [53]. Results are shown in Table 3.

4.6 Evaluation Results

In this subsection, we analyze the results for each downstream task.

4.6.1 Few-shot Symbolic Music Analysis

With only 74 training songs, our adapters outperform rule-based baselines on both majmin and seventh categories. By comparing function alignment models (FA) with the prober, we see that using a pretrained LM for the target sequence $\mathbf{y}$ (chord+drums) improves performance on the music understanding task.

Between the function alignment models, the self-attentive adapters achieve better performance compared to cross-attentive implementation. Such trend is also observed in other tasks.

4.6.2 Chord to Melody

The results in subjective evaluation (Fig. 5a) shows that the our proposed adapters (FA-Self, FA-Cross) achieve comparable performance compared to Melody T5. Cocomulla is not effective on the task, achieving even lower performance compared to a encoder-decoder training. is also demonstrated in objective evaluation results (Tab. 1).

4.6.3 Melody to Chord

Both the perplexity results (Tab. 1) and chord consistency results (Tab. 2) demonstrate the effectiveness of function alignment, especially self-attentive adapters against other baselines. We note that MelodyT5 shows low chroma consistency. MelodyT5 often fails to generate music that meets the constraints of the condition melody (e.g., replaced by an improvised melody or inconsistent structures). This results in a misalignment between the generated chords and the ground truth.

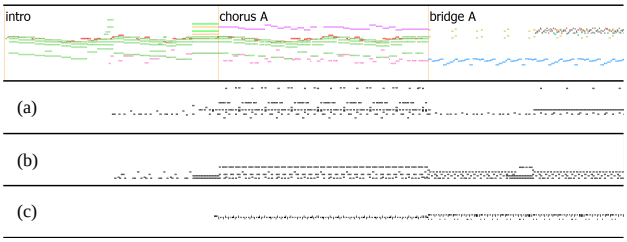

Figure 6. Case study of an others-to-drum example on RWC-Pop-003. The top displays the non-drum condition inputs with a piano roll (structure labels are shown for reference but not used by the model). The bottom shows the drum track by (a) FA-Cross; (b) FA-Self; (c) Ground truth.

4.6.4 Drum to Others

FA models demonstrate strong performance in this category. Also, drum-to-others is the only task where Coco-Mulla outperforms Enc-Dec, highlighting the value of the pretrained LM for the target sequence $\mathbf{y}$. However, Coco-Mulla does not utilize the knowledge stored in LM($\mathbf{x}$), leading to a worse performance compared to function alignment adapters.

4.6.5 Others to Drum

The others-to-drum task yields the interesting results: function alignment models outperform even the ground truth both subjectively (Fig. 5(c)) and objectively (Tab. 1). This is likely because RWC-Pop uses a limited drum set and regular patterns, while our training data (Los Angeles MIDI) includes diverse textures and instruments (e.g., Cuica). Function alignment models generate rich, varied drum patterns aligned with long-term structure, showing strong creativity and musicality (see Fig. 6 for an example). The baseline model Composer Assistant V2 [50] also produces less variation.

5. CONCLUSION AND FUTURE WORKS

In this paper, we address the problem of versatile music-for-music modeling that unifies a broad range of music understanding and controllable generation tasks. Inspired by function alignment, we adopt a parameter-efficient approach by knowledge transfer from the pretrained LM of both the input and the output sequence. We introduce two implementations, the cross-attentive adapter and the self-attentive adapter. Both adapters show competitive results on analysis and generation tasks, with self-attentive adapters relatively outperforming.

There are mainly two future works. First, we need to refine the representations for different music-for-music tasks. We also plan to extend the framework to cross-modal adapters, such as text-to-music tasks.

6. REFERENCES

[1] C. Donahue, J. Thickstun, and P. Liang, "Melody transcription via generative pre-training," *arXiv preprint arXiv:2212.01884*, 2022.

[2] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Xiao, C. Lin, A. Ragni, E. Benetos *et al.*, "Mert: Acoustic music understanding model with large-scale self-supervised training," *arXiv preprint arXiv:2306.00107*, 2023.

[3] D. Li, Y. Ma, W. Wei, Q. Kong, Y. Wu, M. Che, F. Xia, E. Benetos, and W. Li, "Mertech: Instrument playing technique detection using self-supervised pre-trained model with multi-task finetuning," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 521–525.

[4] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

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

[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] G. G. Xia, "Function alignment: A new theory for mind and intelligence, part i: Foundations," 2025. [Online]. Available: https://arxiv.org/abs/2503.21106

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] 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.

[10] 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, pp. 47 704–47 720, 2023.

[11] 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.

[12] C. Zhang, Y. Ma, Q. Chen, W. Wang, S. Zhao, Z. Pan, H. Wang, C. Ni, T. H. Nguyen, K. Zhou *et al.*, "Inspiremusic: Integrating super resolution and large language model for high-fidelity long-form music generation," *arXiv preprint arXiv:2503.00084*, 2025.

[13] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. M. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:54477714

[14] 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.

[15] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM international conference on multimedia*, 2020, pp. 1180–1188.

[16] D.-V.-T. Le and Y.-H. Yang, "Meteor: Melody-aware texture-controllable symbolic orchestral music generation," *arXiv preprint arXiv:2409.11753*, 2024.

[17] H. Chen, J. B. L. Smith, J. Spijkervet, J. Wang, P. Zou, B. Li, Q. Kong, and X. Du, "Sympac: Scalable symbolic music generation with prompts and constraints," pp. 1029–1036, 2024.

[18] Z. Zhang, L. Li, J. Zhang, Z. Hu, H. Wang, C. Yan, J. Yang, and Y. Qi, "Generating high-quality symbolic music using fine-grained discriminators," in *International Conference on Pattern Recognition*. Springer, 2025, pp. 332–344.

[19] Y.-J. Shih, S.-L. Wu, F. Zalkow, M. Müller, and Y.-H. Yang, "Theme transformer: Symbolic music generation with theme-conditioned transformer," *IEEE Transactions on Multimedia*, vol. 25, pp. 3495–3508, 2022.

[20] X. Qu, Y. Bai, Y. Ma, Z. Zhou, K. M. Lo, J. Liu, R. Yuan, L. Min, X. Liu, T. Zhang *et al.*, "Mupt: A generative symbolic music pretrained transformer," *arXiv preprint arXiv:2404.06393*, 2024.

[21] Y. Shu, H. Xu, Z. Zhou, A. v. d. Hengel, and L. Liu, "Musebarcontrol: Enhancing fine-grained control in symbolic music generation through pre-training and counterfactual loss," *arXiv preprint arXiv:2407.04331*, 2024.

[22] 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.

[23] H. F. García, P. Seetharaman, R. Kumar, and B. Pardo, "Vampnet: Music generation via masked acoustic token modeling," in *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*, 2023, pp. 359–366.

[24] 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 *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 1206–1210.

[25] 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.

[26] M. W. Lam, Q. Tian, T. Li, Z. Yin, S. Feng, M. Tu, Y. Ji, R. Xia, M. Ma, X. Song *et al.*, "Efficient neural music generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 17 450–17 463, 2023.

[27] F. Schneider, O. Kamal, Z. Jin, and B. Schölkopf, "Moûsai: Efficient text-to-music diffusion models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 8050–8068.

[28] S. Hou, S. Liu, R. Yuan, W. Xue, Y. Shan, M. Zhao, and C. Zhang, "Editing music with melody and text: Using controlnet for diffusion transformer," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.

[29] Z. Wang, L. Min, and G. Xia, "Whole-song hierarchical generation of symbolic music using cascaded diffusion models," in *The Twelfth International Conference on Learning Representations*, 2024.

[30] O. Tal, A. Ziv, I. Gat, F. Kreuk, and Y. Adi, "Joint audio and symbolic conditioning for temporally controlled text-to-music generation," *arXiv preprint arXiv:2406.10970*, 2024.

[31] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2022, pp. 61–68.

[32] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[33] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue *et al.*, "Llama-adapter v2: Parameter-efficient visual instruction model," *arXiv preprint arXiv:2304.15010*, 2023.

[34] Y. Lan, W. Hsiao, H. Cheng, and Y. Yang, "Musicongen: Rhythm and chord control for transformer-based text-to-music generation," in *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024, San Francisco, California, USA and Online, November 10-14, 2024*, 2024, pp. 311–318.

[35] L. Lin, G. Xia, Y. Zhang, and J. Jiang, "Arrange, inpaint, and refine: Steerable long-term music audio generation and editing via content-based controls," *arXiv preprint arXiv:2402.09508*, 2024.

[36] Y. Zhang, Y. Ikemiya, W. Choi, N. Murata, M. A. Martínez-Ramírez, L. Lin, G. Xia, W.-H. Liao, Y. Mitsufuji, and S. Dixon, "Instruct-musicgen: Unlocking text-to-music editing for music language models via instruction tuning," *arXiv preprint arXiv:2405.18386*, 2024.

[37] F. Tsai, S. Wu, H. Kim, B. Chen, H. Cheng, and Y. Yang, "Audio prompt adapter: Unleashing music editing abilities for text-to-music with lightweight finetuning," in *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024, San Francisco, California, USA and Online, November 10-14, 2024*, 2024, pp. 634–641.

[38] L. Ou, J. Zhao, Z. Wang, G. Xia, and Y. Wang, "Unlocking potential in pre-trained music language models for versatile multi-track music arrangement," *arXiv preprint arXiv:2408.15176*, 2024.

[39] 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.

[40] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "Pianotree vae: Structured representation learning for polyphonic music," *arXiv preprint arXiv:2008.07118*, 2020.

[41] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[42] R. Bansal, B. Samanta, S. Dalmia, N. Gupta, S. Vashishth, S. Ganapathy, A. Bapna, P. Jain, and P. Talukdar, "Llm augmented llms: Expanding capabilities through composition," *arXiv preprint arXiv:2401.02412*, 2024.

[43] V. Zayats, P. Chen, M. Ferrari, and D. Padfield, "Zipper: A multi-tower decoder architecture for fusing modalities," *arXiv preprint arXiv:2405.18669*, 2024.

[44] A. Lev, "Los angeles midi dataset: Sota kilo-scale midi dataset for mir and music ai purposes," in *GitHub*, 2024.

[45] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[46] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. SPIE, 2019, pp. 369–386.

[47] "Nottingham database," http://ifdo.ca/~seymour/nottingham/nottingham.html, accessed: 2025-03-26.

[48] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical and jazz music databases." in *ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002, Proceedings*, vol. 2, 2002, pp. 287–288.

[49] S. Wu, Y. Wang, X. Li, F. Yu, and M. Sun, "Melodyt5: A unified score-to-score transformer for symbolic music processing," *arXiv preprint arXiv:2407.02277*, 2024.

[50] M. Malandro, "Composer's Assistant 2: Interactive Multi-Track MIDI Infilling with Fine-Grained User Control," in *Proc. 25th Int. Society for Music Information Retrieval Conf.*, San Francisco, CA, USA, 2024, pp. 438–445.

[51] Y. Wang, J. Deng, A. Sun, and X. Meng, "Perplexity from plm is unreliable for evaluating text quality," *arXiv preprint arXiv:2210.05892*, 2022.

[52] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research*, vol. 50, no. 1, pp. 37–51, 2021.

[53] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "Mir_eval: A transparent implementation of common mir metrics." in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, vol. 10, 2014, p. 2014.

[54] S. Wu, Y. Wang, X. Li, F. Yu, and M. Sun, "Melodyt5: A unified score-to-score transformer for symbolic music processing," *arXiv preprint arXiv:2407.02277*, 2024.

[55] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," *arXiv preprint arXiv:2008.07122*, 2020.