HIERARCHICAL AUTOREGRESSIVE MUSIC MODELING WITH TEMPORAL REALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models for symbolic music typically flatten events into a single sequence. When used for conditional generation, this creates a conflict between autoregressive sequence order and musical time: future conditioning information must appear early in the sequence, which can disrupt the underlying temporal relationships. We address this mismatch with *model-level temporal realignment*. A hierarchical grouping architecture first organizes tokens into timing-aware groups (notes and concurrent onsets). Rotary time embeddings then reassign temporal coordinates to the original musical times while leaving the input sequence and causal factorization unchanged. The resulting model can still be trained with the standard next-token objective. On both symbolic and audio-conditioned generation tasks, temporal realignment substantially improves timing prediction, yields higher coherence scores, and is preferred by human listeners over an otherwise identical model without realignment under random conditioning.

1 INTRODUCTION

Music is composed of organized sound events whose structure can be understood through a set of abstract perceptual concepts. Unlike general acoustic events, much of the perceptual and emotional impact of music arises from how sounds are structured. These relationships give rise to recurring patterns over time, forming rhythm, while relationships in pitch contribute to melody and harmony. More broadly, musical structure emerges from the intricate interplay between horizontal temporal dependencies and vertical synchronous relationships (Lerdahl & Jackendoff, 1996).

Humans express musical ideas through symbolic representations, much like language conveys meaning through discrete symbols. In computational modeling, symbols are encoded as tokens (Huang et al., 2019). Contiguous tokens can be naturally organized into *groups*, representing musical events or temporal structures such as notes or sets of notes sharing the same onset (Hsiao et al., 2021; Zeng et al., 2021; Dong et al., 2023). *Event-based representations* therefore naturally span multiple levels, from individual tokens to note groups and higher-level temporal groups. Autoregressive models, however, typically operate on the flattened token sequence and process it according to sequence order. For conditional generation, existing *in-sequence* decoder-only methods place conditioning and target tokens within the same sequence (Thickstun et al., 2024). For example, Tan et al. (2024a) use melody and chords as conditions to generate a piano performance realization as the target.

A fundamental challenge in in-sequence conditioning arises from an inherent conflict between **sequence order** and **musical-time order**. To satisfy autoregressive causality, conditioning tokens must be placed before the target tokens that depend on them, inevitably disrupting their original temporal ordering, as illustrated in Figure 1. This mismatch is particularly important in event-based music representations, where sequence position does not itself determine musical time. Although time-related tokens can explicitly encode the musical time associated with each group, the model still processes a flattened token sequence whose ordering may be temporally misaligned. It must therefore recover temporal relationships from a sequence whose ordering no longer follows the underlying musical timeline. The mismatch also raises a question of representation granularity. Individual tokens encode musical attributes, whereas musical time is associated with musical events and higher-level temporal groups rather than isolated token positions. Temporal ordering must therefore be considered at the level of these groups. Resolving the mismatch consequently requires modeling the groups explicitly and performing realignment at the corresponding level.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

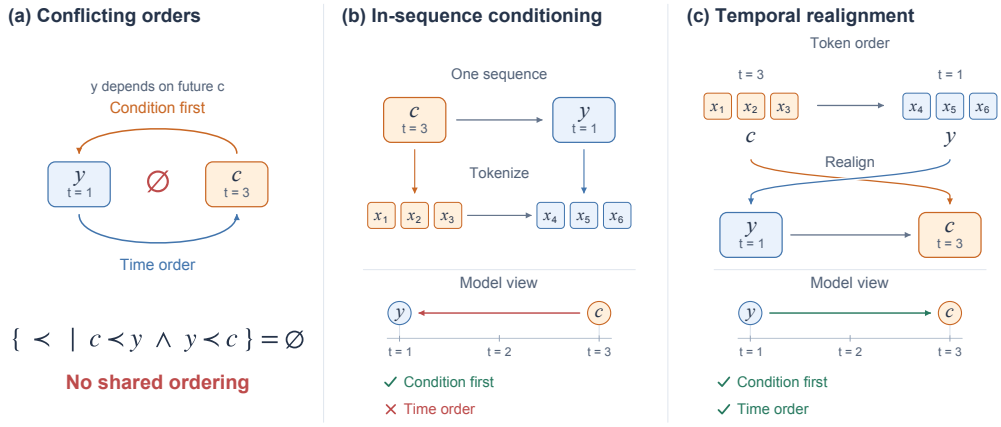

Figure 1: Sequence order versus musical-time order. (a) A target group y at $t = 1$ depends on a conditioning group c at $t = 3$. Conditioning availability requires $c \prec y$, whereas musical chronology requires $y \prec c$; a shared ordering cannot satisfy both ($\varnothing$ denotes this contradiction). (b) In-sequence conditioning places the tokens of c before those of y, reversing their musical-time order. (c) Temporal realignment restores musical-time coordinates at the group level while preserving the condition-first token factorization. The rearranged groups and model-view inset depict musical-time relationships; the input sequence remains unchanged.

In this paper, we propose *temporal realignment*, which allows the model to process musical groups according to their original musical-time order without explicitly reordering the input sequence, while preserving autoregressive modeling. We first formalize two structures underlying this problem in Section 2. We introduce *delta conditioning*, a general formulation of lookahead conditioning that characterizes how conditioning groups are positioned relative to target groups in the autoregressive order, and *laminar group representation*, which characterizes hierarchical token groupings whose boundaries remain identifiable in autoregressive generation. Building on this framework, Section 3 introduces temporal realignment. Rather than encoding timing only at the representation level, our method realigns how groups are processed so that target and conditioning groups are viewed according to their original musical-time order. This allows the model to learn their temporal relationships while retaining the autoregressive formulation. Specifically, flattened token representations are mapped into timing-aware groups through hierarchical grouping, temporal realignment is applied at the corresponding level, and the representations are mapped back for autoregressive prediction. The resulting end-to-end model remains compatible with the standard autoregressive objective.

We evaluate our approach on symbolic music generation under two conditioning modalities: symbolic and audio. Temporal realignment consistently outperforms non-realigned baselines for both settings, suggesting its effectiveness and broader applicability across different conditioning sources.

2 Formal Framework

Symbolic music is represented as individual tokens, but these tokens naturally form larger musical structures, from events to higher-level temporal groups. In this paper, we use *group* to refer generally to a contiguous collection of tokens representing such a structure. Autoregressive models operate on the resulting flattened token sequence, where each group occupies a contiguous span. In conditional generation, conditioning groups are placed earlier in the sequence to make them available to the target groups. We establish the structural foundation for this setting here.

2.1 Musical Time and Ordering Time

For a group g, we distinguish between two notions of time. Its *musical time*, denoted by $t(g)$, specifies the time represented by the group itself, while its *ordering time*, denoted by $\tilde{t}(g)$, determines where the group appears in the sequence. When groups are ordered by their musical times, we set

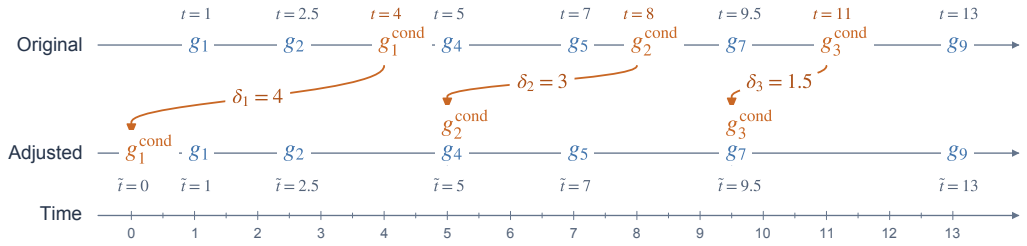

Figure 2: Mono-constrained delta conditioning. Target groups g_i (blue; shorthand for g_i^{tar}) and conditioning groups g_n^{cond} (orange) share a linear time axis. Offsets $(4, 3, 1.5)$ shift conditioning ordering times from $(4, 8, 11)$ to $(0, 5, 9.5)$ while preserving musical timestamps and the relative order of conditioning groups. Target ordering times remain unchanged. Vertically aligned groups share an ordering time; a tie-breaking rule determines their sequence ordering.

the ordering time equal to the musical time,

$$\tilde{t}(g) = t(g). \tag{1}$$

Changing $\tilde{t}$ may alter the sequence order of groups without changing the musical time t they represent. This allows the autoregressive ordering of groups to be modified independently of the underlying musical timeline. For example, when generating an accompaniment, the model may be given access to future melody groups before generating the current accompaniment. These melody groups are assigned earlier ordering times $\tilde{t}$ so that they appear earlier in the sequence, without changing t.

2.2 Delta Conditioning

For conditional generation, a sequence $\mathbf{g} = (g_1, \ldots, g_L)$ consists of conditioning groups $\mathbf{g}^{\text{cond}}$ and target groups $\mathbf{g}^{\text{tar}}$, which may be interleaved in different ways. For example, the conditioning sequence may be placed as a prefix, $\mathbf{g} = (\mathbf{g}^{\text{cond}}, \mathbf{g}^{\text{tar}})$. For autoregressive generation, we consider permutations that place each conditioning group before the target groups that depend on it, making the required conditioning information available when those targets are generated. To formally decouple conditioning availability from musical chronology, we refer to placing conditioning and target groups according to such permutations as performing *delta conditioning*. Many conditioning approaches (Ou et al., 2025; Wu & Yang, 2023a; Thickstun et al., 2024) can be viewed as special cases of delta conditioning, as discussed in Appendix A.

Let $\mathcal{Z}_\Delta(\mathbf{g})$ denote the set of delta-conditioning permutations for the group sequence $\mathbf{g}$. For $\mathbf{z} = (z_1, \ldots, z_L) \in \mathcal{Z}_\Delta(\mathbf{g})$, let $\mathbf{g}_{\mathbf{z}_{<i}} = (g_{z_1}, \ldots, g_{z_{i-1}})$ denote the groups preceding a specific group g_{z_i}. The corresponding autoregressive factorization is

$$p_\theta(\mathbf{g} \mid \mathbf{z}) = \prod_{i=1}^{L} p_\theta(g_{z_i} \mid \mathbf{g}_{\mathbf{z}_{<i}}). \tag{2}$$

Further background on permutation-based autoregressive factorization (Yang et al., 2019; Bautista & Atienza, 2022) is provided in Appendix B.

Delta conditioning is parameterized by shifting the ordering times $\tilde{t}$ of the conditioning groups while leaving their musical times t unchanged. As illustrated in Figure 2, each conditioning group g_n^{cond} is assigned an offset δ_n, which moves it earlier in the factorization order without changing its musical time $t(g_n^{\text{cond}})$. Formally, the ordering time $\tilde{t}(g_\ell)$ is given by

$$\tilde{t}(g_\ell) = \begin{cases} t(g_n^{\text{cond}}) - \delta_n, & \text{if } g_\ell = g_n^{\text{cond}}, \\ t(g_m^{\text{tar}}), & \text{if } g_\ell = g_m^{\text{tar}}. \end{cases} \tag{3}$$

where the offsets are chosen such that $\tilde{t}(g_n^{\text{cond}}) \leq \tilde{t}(g_m^{\text{tar}})$ whenever g_m^{tar} depends on g_n^{cond}, with ties resolved in favor of the conditioning group.

A delta-conditioning permutation $\mathbf{z}$ is a factorization order consistent with the ordering times:

$$\tilde{t}(g_{z_i}) \leq \tilde{t}(g_{z_{i+1}}), \qquad i = 1, \ldots, L-1. \tag{4}$$

Equivalently, one such permutation can be obtained by

$$\mathbf{z} = \pi_{\boldsymbol{\delta}}(\mathbf{g}) = \mathrm{argsort}_{\ell}\, \tilde{t}(g_{\ell}), \tag{5}$$

with an appropriate tie-breaking rule, where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_N)$ denotes the conditioning offsets.

Assigning arbitrary values of δ_n may disrupt the temporal ordering among conditioning groups. To preserve this ordering, we restrict the offsets such that their ordering times remain monotonic:

$$t(g_{n-1}^{\mathrm{cond}}) - \delta_{n-1} < t(g_n^{\mathrm{cond}}) - \delta_n, \qquad n = 2, \ldots, N. \tag{6}$$

We refer to this setting as *mono-constrained delta conditioning*, which we assume throughout the rest of the paper. We consider in-sequence conditional generation under this setting, where target groups are generated from both conditioning groups and preceding target history while the relative ordering of conditioning groups is preserved. A given set of conditioning offsets does not necessarily determine a unique sequence order, since multiple groups may share the same ordering time.

2.3 Laminar Group Representation

With the conditioning framework in place, we next formalize the hierarchical organization of tokens into groups and their associated timing, a structure that has not been made explicit in prior work.

Specifically, we introduce the notion of a *laminar group representation*, in which token groups are arranged hierarchically. Groups may recursively contain nested subgroups, while any two groups are either nested or disjoint but never partially overlapping, reflecting the meaning of laminar.

Formally, let $\mathcal{G}$ denote the set of groups in a token sequence $\mathbf{x} = (x_1, \ldots, x_M)$. Each group $g \in \mathcal{G}$ occupies a contiguous token span $I(g) = \{a_g, \ldots, b_g\}$, where $1 \le a_g \le b_g \le M$.

First, the group spans form a laminar family:

$$I(g) \cap I(g') = \varnothing \quad \text{or} \quad I(g) \subseteq I(g') \quad \text{or} \quad I(g') \subseteq I(g), \qquad \forall g, g' \in \mathcal{G}. \tag{7}$$

This laminar structure naturally supports recursive grouping across multiple levels, allowing a group to contain finer-grained subgroups without introducing partial overlap.

For two non-overlapping groups $g, g' \in \mathcal{G}$ with $b_g < a_{g'}$, possibly at different hierarchy levels, we say that g and g' are *adjacent* if there exists no group $h \in \mathcal{G}$ whose span lies strictly between them:

$$\nexists\, h \in \mathcal{G} \quad \text{such that} \quad b_g < a_h \le b_h < a_{g'}. \tag{8}$$

Second, the boundary between every pair of adjacent groups must be deterministically identifiable during autoregressive inference. For adjacent groups g and g', there must exist a boundary position j satisfying

$$b_g \le j \le a_{g'}, \tag{9}$$

such that the transition from g to g' can be determined from the generated prefix $\mathbf{x}_{\le j}$ without relying on future tokens. The boundary may be indicated by the end of g, by a separator between the two groups, or by the start of g'.

This structure supports recursive grouping across multiple levels. Figure 5 in Appendix C illustrates an example. Many existing symbolic music representations satisfy this definition (Huang & Yang, 2020; Hsiao et al., 2021; Thickstun et al., 2024), making laminar group representation a practical assumption for our method. Further discussion and properties of laminar group representations, including their compatibility with autoregressive inference, are provided in Appendix C.

3 Methodology

We begin by formalizing the temporal misalignment introduced by delta conditioning through the distinction between ordering time and musical time.

Proposition 1 (Temporal misalignment under delta conditioning)**.** *If a target group depends on a conditioning group occurring later in musical time, making the conditioning group available before generating the target necessarily introduces a misalignment between ordering and musical time.*

Consider a target group g_i^{tar} that depends on a conditioning group g_n^{cond} occurring later in musical time, such that $t(g_i^{\mathrm{tar}}) < t(g_n^{\mathrm{cond}})$. Making the conditioning group available before generating the target instead requires $\tilde{t}(g_n^{\mathrm{cond}}) \leq \tilde{t}(g_i^{\mathrm{tar}})$, reversing their relative order. The ordering-time relation is therefore misaligned with the musical-time relation. A proof is provided in Appendix D.

3.1 Temporal Realignment

The temporal misalignment cannot be resolved by restoring musical-time order at the token sequence level, since doing so would remove the causal availability of future conditioning information. Instead, we preserve the ordering-time sequence required for autoregressive generation and realign how groups are processed within the model. We refer to this operation as *temporal realignment*. It allows groups to be processed according to their musical times while leaving the input sequence and autoregressive factorization unchanged. Conditioning groups therefore remain causally available to their targets, while the model operates on the original musical-time relationships among groups.

To apply realignment at the level where musical time is defined, we exploit the hierarchical structure of a laminar group representation. Token-level representations are mapped into groups at the appropriate level, and their temporal relationships are modeled according to musical time rather than their positions in the flattened token sequence.

Let $\mathbf{z} = (z_1, \ldots, z_L)$ denote a delta-conditioning permutation, and let

$$\mathbf{g}_{\mathbf{z}} = (g_{z_1}, \ldots, g_{z_L}) \tag{10}$$

denote the resulting group sequence in factorization order. Each group g_{z_i} has an ordering time $\tilde{t}(g_{z_i})$ that determines its position in the sequence, and a musical time $t(g_{z_i})$ that remains unchanged under delta conditioning.

Because the offsets used by delta conditioning are deterministic and known, the mapping from ordering time back to musical time can be constructed directly. Define the group-wise offset

$$\delta(g_\ell) = \begin{cases} \delta_n, & \text{if } g_\ell = g_n^{\mathrm{cond}}, \\ 0, & \text{if } g_\ell = g_m^{\mathrm{tar}}. \end{cases} \tag{11}$$

By definition of delta conditioning,

$$t(g_\ell) = \tilde{t}(g_\ell) + \delta(g_\ell). \tag{12}$$

We define *temporal realignment* as restoring this temporal coordinate while preserving the factorization order:

$$\mathcal{R}_{\boldsymbol{\delta}}\left(g_\ell, \tilde{t}(g_\ell)\right) = \left(g_\ell, \tilde{t}(g_\ell) + \delta(g_\ell)\right) = \left(g_\ell, t(g_\ell)\right). \tag{13}$$

Applied to the full sequence, $\mathcal{R}_{\boldsymbol{\delta}}$ leaves the group order unchanged and realigns only the temporal coordinates used for modeling. The model therefore receives groups in the delta-conditioned factorization order required for autoregressive causality, while their temporal relationships are determined by the original musical times.

3.2 Structural Grouping

Temporal realignment cannot be applied directly at the token level, since musical time is defined over groups rather than individual token positions. Therefore, we map tokens into groups at the appropriate level, apply temporal realignment to the grouped representations, and map the resulting representations back to the token level for autoregressive prediction. Let $X = (h_1, \ldots, h_M)$ denote the input token representations. We apply a grouping function $\mathcal{G}$ to obtain group-level representations,

$$H = \mathcal{G}(X) = (u_1, \ldots, u_L), \tag{14}$$

where each u_i corresponds to a group g_i in the input sequence.

Because the groups follow the delta-conditioned factorization order, their ordering times may not align with their musical times. We hence apply temporal realignment to the grouped representations,

$$\widetilde{H} = \mathcal{R}_{\boldsymbol{\delta}}(H), \tag{15}$$

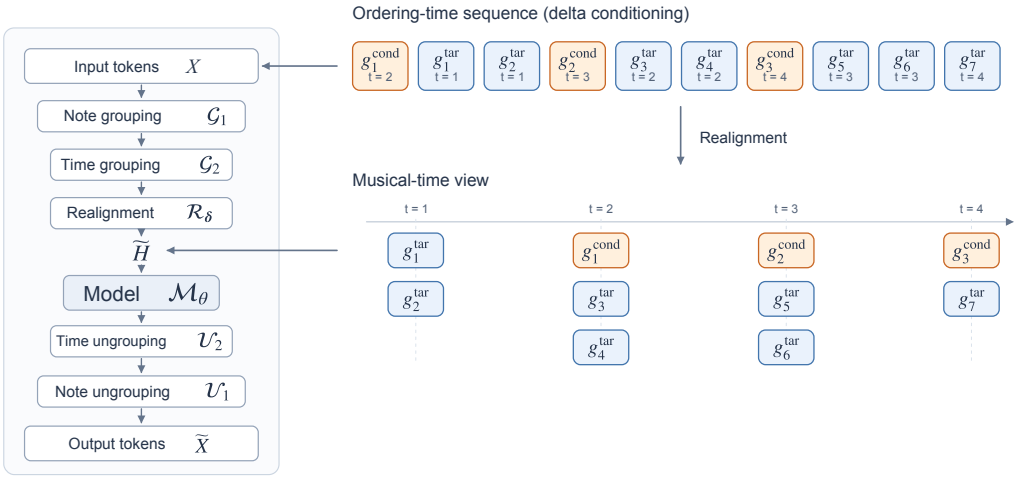

Figure 3: Temporal realignment through structural grouping. Left: token representations X are grouped into "notes" by $\mathcal{G}_1$ and subsequently "time groups" by $\mathcal{G}_2$. Temporal realignment $\mathcal{R}_{\boldsymbol{\delta}}$ assigns musical-time coordinates before modeling by the language model $\mathcal{M}_\theta$. Time and note ungrouping $(\mathcal{U}_2, \mathcal{U}_1)$ map the results back to the token-level representations $\widetilde{X}$. Block widths schematically indicate compression and expansion. Right: conditioning groups g_n^{cond} (orange) and target groups g_i^{tar} (blue) are ordered by ordering time under delta conditioning (top). The musical-time view (bottom) visualizes the temporal coordinates associated with the realigned representations $\widetilde{H}$, aligning groups with equal timestamps in the same column. This view illustrates temporal relationships; the input sequence and autoregressive factorization remain unchanged.

where $\mathcal{R}_{\boldsymbol{\delta}}$ replaces the ordering-time coordinates associated with each group by their corresponding musical-time coordinates while leaving the representation sequence unchanged.

The realigned representations $\widetilde{H}$ are then processed by a decoder-only Transformer $\mathcal{M}_\theta$. In this way, the model operates on group relationships in musical time while preserving the delta-conditioned sequence order required for autoregressive generation. After language modeling, an ungrouping function $\mathcal{U}$ maps the group-level representations back to the token level,

$$\widetilde{X} = \mathcal{U}(\widetilde{H}). \tag{16}$$

This allows temporal realignment to operate at the group level while preserving the original token-level autoregressive prediction order. The overall concept is depicted in Figure 3.

3.3 Implementation

Our representation organizes symbolic music into three hierarchical levels: tokens, note groups, and time groups. Each *note group* contains attribute tokens (instrument, pitch, and duration) corresponding to a single musical note event, while each *time group* contains the note groups that share the same musical onset (i.e., concurrent notes). Further representation details are provided in Appendix E.

We implement grouping and ungrouping using a hierarchical architecture inspired by H-Net (Hwang et al., 2026). Starting from token-level representations $X^{(0)}$, the model applies two grouping stages,

$$X^{(1)} = \mathcal{G}_1(X^{(0)}), \qquad X^{(2)} = \mathcal{G}_2(X^{(1)}), \tag{17}$$

where $X^{(1)}$ and $X^{(2)}$ denote the representations at the level of note groups (i.e., "note-level") and that of the time groups ("time-level"), respectively. Corresponding ungrouping functions $\mathcal{U}_2$ and $\mathcal{U}_1$ map the representations back through the hierarchy for token-level autoregressive prediction.

Temporal realignment is applied at the time level $X^{(2)}$, where musical time is explicitly defined. We realize $\mathcal{R}_{\boldsymbol{\delta}}$ through the rotary time embedding (RoTE) (Goel et al., 2024; Ghosh et al., 2026). Each

time-level representation remains at the same position in the delta-conditioned sequence, while its temporal coordinate is reassigned from ordering time to musical time. After modeling, $\mathcal{R}_{\boldsymbol{\delta}}^{-1}$ restores the ordering-time assignment before ungrouping. The overall pipeline can be summarized as

$$X^{(0)} \xrightarrow{\mathcal{G}_1} X^{(1)} \xrightarrow{\mathcal{G}_2} X^{(2)} \xrightarrow{\mathcal{R}_{\boldsymbol{\delta}}} \widetilde{X}^{(2)} \xrightarrow{\mathcal{M}_\theta} \widetilde{Z}^{(2)} \xrightarrow{\mathcal{R}_{\boldsymbol{\delta}}^{-1}} Z^{(2)} \xrightarrow{\mathcal{U}_2} Z^{(1)} \xrightarrow{\mathcal{U}_1} Z^{(0)}. \tag{18}$$

Both $\mathcal{R}_{\boldsymbol{\delta}}$ and $\mathcal{R}_{\boldsymbol{\delta}}^{-1}$ are realized through RoTE position assignments without explicitly reordering the representation sequence. Additional model details are in Appendix F.

4 Evaluation

We evaluate temporal realignment under two conditioning modalities, symbolic and audio, using symbolic MIDI generation as the target task. We evaluate teacher-forced prediction using *perplexity* and generation quality using *PHALAR* (Marincione et al., 2026), along with a listening study. Our experiments use the Lakh MIDI Dataset (Raffel, 2016) with the deduplicated split of Choi et al. (2025). PHALAR measures the coherence between a conditioning signal and the generated target by comparing their synthesized audio representations. Since PHALAR is trained to match paired audio clips, MIDI is synthesized to audio before evaluation. Due to space limit, we provide description of the full model, training process, representation and evaluation details in Appendix G.

4.1 Symbolic Conditioning

For symbolic conditioning, we implement anticipation conditioning (Thickstun et al., 2024) with a 10-second lookahead window, a family of delta conditioning, and evaluate two tasks. In *accompaniment*, we use the melody track to condition the generation of the remaining MIDI tracks (instruments). In *random conditioning*, 25% of the notes across all the MIDI tracks are randomly selected as conditioning and the model generates the remainder. Both tasks use a 5-second prompt.

We implement three variants of the proposed H-Net based model: `Balanced`, `Inner`, and `Outer`. As detailed in Appendix G, the names reflect how parameters are distributed across different stages of the model. Compared with `Balanced`, `Inner` allocates more parameters to $\mathcal{M}_\theta$, whereas `Outer` allocates more parameters to $\mathcal{G}_2$. Moreover, we implement ablated versions of them without temporal realignment (i.e., no $\mathcal{R}_{\boldsymbol{\delta}}$). As for the baseline, we implement the state-of-the-art MIDI generation model Anticipatory Music Transformer (`AMT`) (Thickstun et al., 2024). All these models employ anticipation conditioning. For fair comparison, we ensure that they have the same training data, training setup and underlying token representation, and similar total model size ($\sim$134M).

Table 1 shows that temporal realignment most strongly improves timing prediction, as reflected by PPLt. The realigned variants substantially reduce time-token perplexity relative to their no-realignment counterparts (e.g., `Balanced` vs. '`Balanced` w/o realignment'), while differences in instrument, pitch, and duration are smaller. The best realigned model also outperforms the `AMT` baseline in generation quality. This suggests that group-level realignment retains the efficiency of grouped modeling (Hsiao et al., 2021; Ryu et al., 2024) while improving generation performance.

Table 2 shows the result of the subjective evaluation comparing the human-composed ground truth, the `Balanced` model, and its no-realignment ablation (see Appendix G.5 for the listening protocol). For random conditioning,the one with temporal realignment significantly outperforms that without, with 67 wins, 19 ties, and 14 losses (p-value < 0.05) with exact paired permutation test. For accompaniment conditioning, the difference is smaller and not significant. Human compositions remain significantly preferred in both tasks. We note that, under random conditioning, conditioning notes are distributed throughout the generation span, requiring the model to account more precisely for their musical times relative to the generated target. This suggests that temporal realignment becomes increasingly beneficial as conditioning places stronger demands on temporal alignment.

4.2 Audio Conditioning

We evaluate temporal realignment under audio conditioning using drum generation as the exemplary target task. Given accompaniment audio synthesized from MIDI, the model generates the missing drum track. The accompaniment reaches the model only as audio latents, which bypass the grouping stages $\mathcal{G}_1$ and $\mathcal{G}_2$ and enter the innermost network $\mathcal{M}_\theta$ directly. We place them at this level because

Table 1: Objective evaluation results for symbolic- and audio- conditioned MIDI generation, using perplexity ('PPL'; the lower the better) and PHALAR ('PHA.'; the higher the better). PPLt, PPLi, PPLp, PPLd denote the perplexity for time, instrument, pitch, and duration tokens, respectively.

Condition	Model	PPL ↓	PPL$_t$ ↓	PPL$_i$ ↓	PPL$_p$ ↓	PPL$_d$ ↓	PHA. ↑
Symbolic	Human compositions	–	–	–	–	–	5.98
	`Balanced`	1.78	**1.80**	1.36	1.56	2.66	**5.57**
	`Inner`	1.84	1.88	1.39	1.63	2.75	5.51
	`Outer`	1.79	1.85	1.36	1.57	2.67	5.15
	`Balanced` w/o realignment	1.95	2.57	1.45	1.66	2.81	3.25
	`Inner` w/o realignment	2.05	2.82	1.51	1.76	2.91	2.63
	`Outer` w/o realignment	1.92	2.58	1.42	1.64	2.74	2.71
	`AMT`	**1.71**	1.83	**1.35**	**1.46**	**2.49**	4.27
Audio	Human compositions	–	–	–	–	–	6.51
	`Inner`	**1.54**	**1.87**	**1.07**	1.65	**1.84**	**6.35**
	`Inner` w/o realignment	1.54	1.91	1.07	**1.62**	1.87	5.40

Table 2: Subjective evaluation result of pairwise preferences comparing 'A' vs 'B'. We use "Win" to refer to the case "A wins (is better than) B," and similarly for "Tie" and "Lose."

Task	**Comparison**	**Win**	**Tie**	**Lose**
Accompaniment	`Balanced` vs. Human compositions	16	20	64
	`Balanced` vs. `Balanced w/o realignment`	41	25	34
Random	`Balanced` vs. Human compositions	27	27	46
	`Balanced` vs. `Balanced w/o realignment`	67	19	14

each latent summarizes a short window of audio anchored at a known onset time, which makes it semantically closer to a time group, i.e., everything sounding at one musical time, than to an individual attribute token or note group. Since no symbolic conditioning tokens are present, the timing of the accompaniment cannot be read off `TIME` tokens; any alignment between the generated drums and the accompaniment must be inferred by $\mathcal{M}_\theta$ from the audio latents alone. This makes the setting a direct test of whether temporal realignment helps $\mathcal{M}_\theta$ relate a time group to the audio at the same musical time. We compare the same frame-level ConvNeXt audio encoder with and without temporal realignment, using the `Inner` variant. Both models are evaluated on 100 songs from its test split, with a 5 s human drum prompt per 30 s clip. Generation quality is measured with PHALAR, which scores the coherence between the generated drums and the conditioning accompaniment. The encoder and the full setup are described in Appendix H.

As shown in Table 1, temporal realignment substantially improves PHALAR, from 5.40 to 6.35. The realigned model also exhibits a smaller spread (standard deviation 1.76 versus 2.59), suggesting fewer windows in which the generated drums fail to align coherently with the accompaniment. Although the frame-level encoder retains fine timing within each latent, it does not replace realignment: without a shared musical-time coordinate, the model must infer the time of each audio token from its content, and it does so only loosely.

Figure 4 provides a qualitative view of this effect. With temporal realignment, attention from MIDI queries to audio keys forms a sharp, high-intensity diagonal: the peak attention weight is roughly $2.5\times$ that of a uniform distribution, compared with only slightly above uniform without realignment. The diagonal is also asymmetric, with most of its mass on audio slightly *after* the query time and earlier audio sharply suppressed. This matches the autoregressive setting, in which a MIDI token anticipates the notes that follow and attends to the audio about to be transcribed. Without realignment, the diagonal becomes broad and weak, and vertical stripes appear: a small set of audio frames act as attention sinks and absorb attention regardless of the query time. These patterns are consistent with the PHALAR results and suggest that temporal realignment helps the model establish precise temporal correspondences between symbolic and audio representations.

A per-song comparison of PHALAR shows that the model with temporal realignment wins more often and by larger margins. Listening confirms this difference, especially on samples with large PHALAR gaps: with realignment, the generated drums lock onto the beat of the accompaniment, whereas without realignment they frequently land on the wrong beat. Because perplexity averages

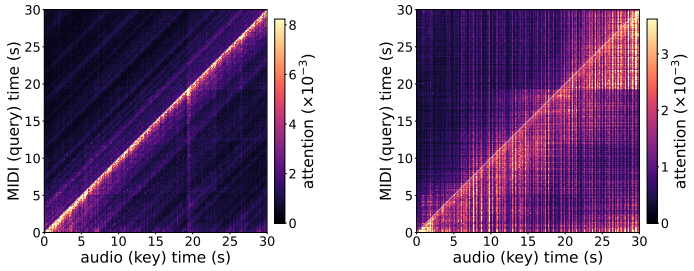

Figure 4: Attention inside the main network $\mathcal{M}_\theta$, i.e., at the time-group level, from time-group queries (vertical axis, query time) to audio keys (horizontal axis, key time) over one 30 s clip. Left: with RoTE. Right: without RoTE. Both maps use the same clip from the Lakh test split. The queries are the time-level states of the song's ground-truth MIDI, fed with teacher forcing, not generated ones. Each map is averaged over all layers and all heads. With realignment, attention concentrates on the diagonal, that is, on the audio sounding at the same moment.

the likelihood over all tokens, it does not capture this misalignment as the timing tokens that determine where each hit falls make up only a small fraction of the sequence.

5 Related Work

Related time-series studies have addressed irregular event modeling and temporal alignment. Event Stream GPT models irregularly timed event streams autoregressively as temporal point processes (McDermott et al., 2023). For regularly sampled time series, CITRAS, CITRAS-FM, and TAT align contextual variables with prediction horizons through shifted attention or related mechanisms (Yamaguchi et al., 2026b;a; Zhao et al., 2026). In contrast, our misalignment arises from placing conditioning and target events within a single autoregressive sequence, causing causal sequence order to deviate from irregular musical-time order. Permutation-based autoregressive modeling has also been explored in XLNet (Yang et al., 2019) and PARSeq (Bautista & Atienza, 2022). These methods alter factorization order over discrete sequence positions, whereas our approach uses rotary time embedding (RoTE) to align model timing directly with musical time. Conditional generation is widely studied in symbolic music, using high-level attributes (von Rütte et al., 2023; Wu & Yang, 2023b), thematic material (Shih et al., 2022), and melodies or lead sheets (Ren et al., 2020; Wu & Yang, 2023a; Tan et al., 2024b). The closest work to ours is the Anticipatory Music Transformer (AMT) (Thickstun et al., 2024), which formalizes asynchronous conditioning by interleaving control and target events in a single autoregressive sequence. However, it does not explicitly distinguish autoregressive order from musical-time order or address the conflicts between them. AMT uses a fixed five-second anticipation window, where such discrepancies may be less pronounced than under longer or varying displacements. Prior work has also explored hierarchical symbolic music modeling across attribute, event, beat, bar, phrase, and section levels (Roberts et al., 2018; Wu et al., 2020; 2023; Wei et al., 2023; Wang et al., 2024; Ryu et al., 2024; Huang et al., 2026). These hierarchies are mainly used for efficient representation, long-range modeling, or coarse-to-fine generation. We instead use hierarchical grouping to define the temporal units on which realignment is performed.

6 Conclusion

We introduced a general framework for temporal realignment in autoregressive conditional generation, combining hierarchical grouping with model-level realignment to reconcile autoregressive order with musical time. Using a new representation for general MIDI, we showed consistent improvements under both symbolic and audio conditioning. Moreover, our results highlight the potential of multi-level tokenization, where grouped musical units can be modeled efficiently while retaining fine-grained autoregressive prediction. Although currently evaluated on symbolic music, the framework itself is general and may apply to other tasks involving irregular time-series or event sequences. Its effectiveness on mixed audio–MIDI sequences further suggests potential for multimodal sequence modeling with shared temporal structure.

AI USE DISCLOSURE

In this work, we used generative AI tools to assist with the mathematical formalization and presentation of ideas developed by the authors, as well as with literature search and manuscript editing. All AI-assisted content was reviewed and verified by the authors. The research ideas and proposed methods were developed by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work involves no sensitive personal data or high-risk applications. For the human-subject evaluation, participation was voluntary and responses were collected anonymously without collecting personally identifiable information. We followed the applicable institutional and conference research ethics requirements.

REPRODUCIBILITY STATEMENT.

We provide implementation and training details, data preprocessing procedures, model configurations, and evaluation protocols in the main paper and appendix. Additional experimental details required to reproduce the reported results are included in the supplementary material.

REFERENCES

Darwin Bautista and Rowel Atienza. Scene text recognition with permuted autoregressive sequence models. In *European Conference on Computer Vision*, pp. 178–196. Springer Nature Switzerland, 2022. doi: 10.1007/978-3-031-19815-1_11.

Eunjin Choi, Hyerin Kim, Jiwoo Ryu, Juhan Nam, and Dasaem Jeong. On the de-duplication of the lakh MIDI dataset. In *Proceedings of the 26th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 44–51, 2025. doi: 10.5281/zenodo.17811316.

Ruben Ciranni, Giorgio Mariani, Michele Mancusi, Emilian Postolache, Giorgio Fabbro, Emanuele Rodolà, and Luca Cosmo. Cocola: Coherence-oriented contrastive learning of musical audio representations, 2025. URL https://arxiv.org/abs/2404.16969.

Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley, and Taylor Berg-Kirkpatrick. Multitrack music transformer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Sreyan Ghosh, Arushi Goel, Lasha Koroshinadze, Sang-gil Lee, Zhifeng Kong, Joao Felipe Santos, Ramani Duraiswami, Dinesh Manocha, Wei Ping, Mohammad Shoeybi, and Bryan Catanzaro. Music flamingo: Scaling music understanding in audio language models. In *International Conference on Learning Representations*, 2026.

Arushi Goel, Karan Sapra, Matthieu Le, Rafael Valle, Andrew Tao, and Bryan Catanzaro. Omcat: Omni context aware transformer. *arXiv preprint arXiv:2410.12109*, 2024.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 178–186, 2021. doi: 10.1609/aaai.v35i1.16091.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJe4ShAcF7.

Jingyue Huang, Zachary Novack, Phillip Long, Yupeng Hou, Ke Chen, Taylor Berg-Kirkpatrick, and Julian McAuley. MuseTok: Symbolic music tokenization for generation and semantic understanding. In *2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2026.

Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1180–1188. Association for Computing Machinery, 2020. doi: 10.1145/3394171.3413671.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. In *International Conference on Learning Representations*, 2026.

Fred Lerdahl and Ray S. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, 1996. ISBN 9780262278164. doi: 10.7551/mitpress/12513.001.0001. URL https://direct.mit.edu/books/monograph/4673/A-Generative-Theory-of-Tonal-Music.

Davide Marincione, Michele Mancusi, Giorgio Strano, Luca Cerovaz, Donato Crisostomi, Roberto Ribuoli, and Emanuele Rodolà. Phalar: Phasors for learned musical audio representations, 2026. URL https://arxiv.org/abs/2605.03929.

Matthew B. A. McDermott, Bret Nestor, Peniel Argaw, and Isaac S. Kohane. Event stream gpt: A data pre-processing and modeling library for generative, pre-trained transformers over continuous-time sequences of complex events. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Longshen Ou, Jingwei Zhao, Ziyu Wang, Gus Xia, Qihao Liang, Torin Hopkins, and Ye Wang. Unifying symbolic music arrangement: Track-aware reconstruction and structured tokenization. In *Advances in Neural Information Processing Systems*, volume 38, pp. 16491–16520, 2025. doi: 10.52202/085713-0496. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/15a5d4408b2bb41b9f923d724bb943f3-Abstract-Conference.html.

Colin Raffel. The lakh midi dataset v0.1. Online dataset, 2016. URL https://colinraffel.com/projects/lmd/.

Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. PopMAG: Pop music accompaniment generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1198–1206, 2020. doi: 10.1145/3394171.3413721.

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4364–4373. PMLR, 2018.

Jiwoo Ryu, Hao-Wen Dong, Jongmin Jung, and Dasaem Jeong. Nested music transformer: Sequentially decoding compound tokens in symbolic music and audio generation. In *Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

Yi-Jen Shih, Shih-Lun Wu, Frank Zalkow, Meinard Müller, and Yi-Hsuan Yang. Theme transformer: Symbolic music generation with theme-conditioned transformer. *IEEE Transactions on Multimedia*, 25:3495–3508, 2022. doi: 10.1109/TMM.2022.3161851.

Chih-Pin Tan, Hsin Ai, Yi-Hsin Chang, Shuen-Huei Guan, and Yi-Hsuan Yang. PiCoGen2: Piano cover generation with transfer learning approach and weakly aligned data. In *Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 555–562, 2024a. doi: 10.5281/zenodo.14877397.

Chih-Pin Tan, Shuen-Huei Guan, and Yi-Hsuan Yang. PiCoGen: Generate piano covers with a two-stage approach. In *Proceedings of the 2024 International Conference on Multimedia Retrieval (ICMR)*, pp. 1180–1184, 2024b. doi: 10.1145/3652583.3657626.

John Thickstun, David Hall, Chris Donahue, and Percy Liang. Anticipatory music transformer. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=EBNJ33Fcrl.

Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann. FIGARO: Controllable music generation using learned and expert features. In *International Conference on Learning Representations (ICLR)*, 2023.

Ziyu Wang, Lejun Min, and Gus Xia. Whole-song hierarchical generation of symbolic music using cascaded diffusion models. In *International Conference on Learning Representations*, 2024.

Xipin Wei, Junhui Chen, Zirui Zheng, Li Guo, Lantian Li, and Dong Wang. A multi-scale attentive transformer for multi-instrument symbolic music generation. In *Proceedings of Interspeech 2023*, pp. 5391–5395, 2023. doi: 10.21437/Interspeech.2023-1626.

Guowei Wu, Shipei Liu, and Xiaoya Fan. The power of fragmentation: A hierarchical transformer model for structural segmentation in symbolic music generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1409–1420, 2023. doi: 10.1109/TASLP.2023.3263797.

Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu. A hierarchical recurrent neural network for symbolic melody generation. *IEEE Transactions on Cybernetics*, 50(6):2749–2757, 2020. doi: 10.1109/TCYB.2019.2953194.

Shih-Lun Wu and Yi-Hsuan Yang. Compose & Embellish: Well-structured piano performance generation via a two-stage approach. In *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023a. doi: 10.1109/ICASSP49357.2023.10095098.

Shih-Lun Wu and Yi-Hsuan Yang. MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1953–1967, 2023b. doi: 10.1109/TASLP.2023.3270726.

Yosuke Yamaguchi, Issei Suemitsu, Yuki Kajihara, and Wenpeng Wei. CITRAS-FM: Tiny time series foundation model for covariate-informed zero-shot forecasting. *arXiv preprint arXiv:2606.10798*, 2026a.

Yosuke Yamaguchi, Issei Suemitsu, and Wenpeng Wei. CITRAS: Covariate-informed transformer for time series forecasting. *IEEE Access*, 14:77983–77998, 2026b. doi: 10.1109/ACCESS.2026.3695717.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu. MusicBERT: Symbolic music understanding with large-scale pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 791–800. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.70.

Zhiyuan Zhao, Sitan Yang, Kin G. Olivares, Boris N. Oreshkin, Stan Vitebsky, Michael W. Mahoney, B. Aditya Prakash, and Dmitry Efimov. TAT: Temporal-aligned transformer for multi-horizon peak demand forecasting. In *2026 IEEE 42nd International Conference on Data Engineering (ICDE)*, pp. 3722–3736, 2026. doi: 10.1109/ICDE65706.2026.00276.

APPENDIX

A VARIANTS OF DELTA CONDITIONING

Delta conditioning provides a general description of how conditioning events are placed relative to target events in an autoregressive sequence. In fact, a broad class of existing conditioning schemes can be expressed as particular choices of the conditioning offsets and tie-breaking rule.

General interleaving. Consider target events $\mathbf{e}^{\mathrm{tar}} = (e_1^{\mathrm{tar}}, \ldots, e_N^{\mathrm{tar}})$ ordered by musical time, and an arbitrary merged ordering that preserves the relative order of the target events. Each conditioning event must then occupy one of the $N + 1$ insertion positions: before the first target, between two consecutive targets, or after the last target.

For a conditioning event e_n^{cond} inserted between e_j^{tar} and e_{j+1}^{tar}, choose an adjusted time $\tilde{t}(e_n^{\mathrm{cond}})$ satisfying

$$t(e_j^{\mathrm{tar}}) \leq \tilde{t}(e_n^{\mathrm{cond}}) \leq t(e_{j+1}^{\mathrm{tar}}), \tag{19}$$

with the desired position within a tie determined by the tie-breaking rule. For insertion before the first or after the last target, an adjusted time outside the corresponding boundary can be chosen. The required delta-conditioning offset is then simply

$$\delta_n = t(e_n^{\mathrm{cond}}) - \tilde{t}(e_n^{\mathrm{cond}}). \tag{20}$$

Sorting by adjusted time therefore reproduces the desired merged ordering. If multiple conditioning events occupy the same insertion position, their relative order can likewise be specified by tie-breaking, or equivalently by arbitrarily small perturbations of their adjusted times. Thus, any interleaving that preserves target-event order can be represented by delta conditioning. If the conditioning order is also preserved, the construction can be chosen to satisfy the mono-constraint.

This observation makes several common conditioning schemes special cases of delta conditioning.

Prefix and sequence-to-sequence conditioning. In prefix-style conditioning, all conditioning information is made available before target generation, as in common sequence-to-sequence formulations (Shih et al., 2022; von Rütte et al., 2023). This corresponds to choosing

$$\tilde{t}(e_n^{\mathrm{cond}}) < t(e_1^{\mathrm{tar}}) \qquad \forall n, \tag{21}$$

with the conditioning events ordered by the tie-breaking rule or by distinct adjusted times. Hence, prefix conditioning is the special case in which all conditioning events are assigned to the insertion position preceding the first target event.

Block-wise interleaving. Block-wise conditioning interleaves conditioning and target content at a coarser temporal granularity, for example by placing the conditioning content for each bar before the target content of the same bar (Tan et al., 2024b;a). Let j_b denote the first target event in block b. Conditioning events belonging to block b can be assigned

$$\tilde{t}(e_n^{\mathrm{cond}}) = t(e_{j_b}^{\mathrm{tar}}), \tag{22}$$

with conditioning events ordered before the target events under the tie-breaking rule. The resulting delta offsets therefore reproduce the block-wise interleaving.

Anticipation conditioning. Anticipation conditioning (Thickstun et al., 2024) requires a slightly different interpretation. Let $s_n = t(e_n^{\mathrm{cond}})$ denote the musical time of a conditioning event, and let Δ denote the anticipation interval used by AMT. Anticipation first uses $s_n - \Delta$ to determine when the conditioning event should become available. In particular, let

$$j_n = \min\left\{j : t(e_j^{\mathrm{tar}}) \geq s_n - \Delta\right\}. \tag{23}$$

The conditioning event is inserted immediately after $e_{j_n}^{\mathrm{tar}}$. Thus, Δ does not directly correspond to the offset δ_n in delta conditioning; rather, it determines an insertion position through the realized target-event sequence.

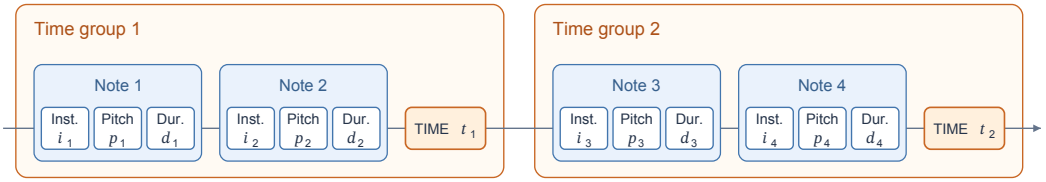

Figure 5: Laminar group representation. Each note group contains instrument, pitch, and duration tokens, with the duration token closing the group. Notes sharing an onset form a time group, closed by an absolute `TIME` token specifying that onset. Group spans are nested or disjoint, never partially overlapping. Closing duration and time tokens make group boundaries identifiable from the generated prefix.

The same ordering is obtained under delta conditioning by assigning

$$\tilde{t}(e_n^{\text{cond}}) = t(e_{j_n}^{\text{tar}}) \tag{24}$$

and breaking the tie such that $e_{j_n}^{\text{tar}}$ precedes e_n^{cond}. The corresponding delta-conditioning offset is therefore

$$\delta_n = s_n - t(e_{j_n}^{\text{tar}}). \tag{25}$$

Consequently, even when AMT uses a fixed anticipation interval Δ, the equivalent delta-conditioning offsets δ_n generally vary across conditioning events and depend on the surrounding target events. In this sense, anticipation first determines *where* each conditioning event is inserted, while delta conditioning directly characterizes the resulting permutation through event-specific offsets.

Prefix conditioning, block-wise interleaving, and anticipation therefore differ primarily in how their insertion positions—and equivalently their delta offsets—are chosen. Delta conditioning provides a common formulation of these otherwise different conditioning strategies.

B Permutation Language Modeling

Permutation language modeling (Yang et al., 2019) trains an autoregressive model over multiple factorization orders. Consider a token sequence $\mathbf{x} = [x_1, \ldots, x_T]$. Its T positions admit $T!$ autoregressive factorization orders. Write $\mathcal{Z}_T$ for these permutations of $[1, 2, \ldots, T]$. For $\mathbf{z} \in \mathcal{Z}_T$, z_t identifies the position predicted at step t, and $\mathbf{z}_{<t}$ lists the positions preceding it in that order. The training objective maximizes the expected log-likelihood over these orders:

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T} \log p_\theta\left(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}\right)\right]. \tag{26}$$

Training samples factorization orders while sharing θ across them, allowing a prediction to use context from either side of its position in the original sequence.

XLNet preserves the original token positions and their positional encodings. Attention masks determine which positions are available under each sampled factorization order. The permutation therefore specifies prediction dependencies while the input retains its original order.

C Properties of Laminar Group Representations

The main text defines a laminar group representation through two properties: group spans form a laminar family, and boundaries between adjacent groups are identifiable from the generated prefix. Here, we further discuss the implications of these properties for hierarchical grouping and autoregressive inference.

Recursive grouping. Because group spans are either nested or disjoint, the groups naturally form a hierarchy. A group may contain finer-grained subgroups, which may themselves contain further subgroups, without introducing partial overlap between branches of the hierarchy. Importantly, the definition does not require the subgroups of a group to fully partition its span. Tokens may therefore belong directly to a higher-level group without forming an additional subgroup. This allows representations such as a time group containing note groups together with its own boundary tokens.

Boundary identifiability. Autoregressive generation proceeds token by token and therefore cannot rely on future tokens to determine the structure of the generated prefix. The boundary condition in a laminar group representation ensures that the transition between adjacent groups can be identified once a sufficient prefix has been generated. An explicit closing token is not required. For example, a boundary may be identified by a token that closes the current group, by a separator, or by a token that opens the next group.

Proposition 2 (Compatibility with autoregressive inference)**.** *Let $\mathbf{x} = (x_1, \ldots, x_M)$ admit a laminar group representation. For any pair of adjacent groups g and g', the transition from g to g' can be determined during autoregressive generation without observing tokens beyond the corresponding boundary position. Consequently, group-level representations can be constructed causally once their boundaries become identifiable.*

Proof. By definition, for every pair of adjacent groups g and g', there exists a boundary position j with

$$b_g \leq j \leq a_{g'}$$

such that the transition from g to g' can be determined from the generated prefix $\mathbf{x}_{\leq j}$. Since $\mathbf{x}_{\leq j}$ contains only tokens already available at autoregressive step j, identifying the boundary does not require access to future tokens. A group can therefore be closed and mapped to a group-level representation once this boundary is identified. □

The laminar property further allows this procedure to be applied recursively across multiple levels of the hierarchy. Lower-level groups can be formed first and subsequently grouped into higher-level representations, while the absence of partial overlap prevents incompatible group structures from arising. Thus, laminar group representations provide a natural structural basis for hierarchical grouping within an autoregressive model.

D Proof of Temporal Misalignment under Delta Conditioning

Proof. Consider a target group g_i^{tar} and a conditioning group g_n^{cond} such that

$$t(g_i^{\mathrm{tar}}) < t(g_n^{\mathrm{cond}}),$$

where g_i^{tar} depends on g_n^{cond}.

Their musical-time order therefore places g_i^{tar} before g_n^{cond}. However, for g_n^{cond} to be available when generating g_i^{tar}, the factorization order must place g_n^{cond} before g_i^{tar}. Equivalently, their ordering times must satisfy

$$\tilde{t}(g_n^{\mathrm{cond}}) \leq \tilde{t}(g_i^{\mathrm{tar}}),$$

with ties resolved in favor of the conditioning group.

Under delta conditioning,

$$\tilde{t}(g_n^{\mathrm{cond}}) = t(g_n^{\mathrm{cond}}) - \delta_n, \qquad \tilde{t}(g_i^{\mathrm{tar}}) = t(g_i^{\mathrm{tar}}).$$

Hence, making the conditioning group available requires

$$\delta_n \geq t(g_n^{\mathrm{cond}}) - t(g_i^{\mathrm{tar}}),$$

up to the tie-breaking rule.

Thus, the ordering-time relation required by autoregressive conditioning cannot preserve the original musical-time order of the two groups. This establishes the temporal misalignment. □

Table 3: Symbolic music vocabulary with a 10 ms time step. Input time tokens encode absolute onsets, while output time classes encode increments from the preceding target time. All integer ranges are inclusive; a dash indicates absence from that space.

Token family	Values	Input	Output	Meaning
TIME_t	0–60,000	60,001	—	Absolute onset; closes a time group
INSTRUMENT_i	0–127	128	128	Melodic instrument
INSTRUMENT_DRUM	—	1	1	Drum instrument
INSTRUMENT_EXTEND_i	0–127	128	128	Melodic continuation
INSTRUMENT_EXTEND_DRUM	—	1	1	Drum continuation
CONDITION_w	As above	258	—	Conditioning note-group opener
PITCH_p	0–127	128	128	Melodic pitch
PITCH_DRUM_p	0–127	128	128	Drum pitch
DURATION_d	1–500	500	500	Duration; closes a note group
SS	—	1	—	Sample start
TARGET_START	—	1	—	Target declaration start
TARGET_END	—	1	1	Target declaration end
BOS	—	1	—	Song start
EOS	—	1	1	Song end
CONDITION_EOS	—	1	—	Conditioning input end
PAD	—	1	—	Padding
Relative time Δt	0–100	—	101	Increment from preceding target time
Total		61,280	1,117	

E MUSIC TOKEN REPRESENTATION

We serialize MIDI into three nested levels: tokens, note groups, and time groups. Onsets and durations are quantized to a 10 ms grid. Each note segment forms a three-token group, INSTRUMENT_i PITCH_p DURATION_d. Segments sharing an onset are collected into a time group followed by TIME_t. Thus, DURATION_d and TIME_t close note and time groups, respectively, providing explicit causal boundaries for hierarchical compression. This representation follows the laminar grouping defined in Section 2.

Melodic instruments use MIDI program numbers, while drums use INSTRUMENT_DRUM with a separate PITCH_DRUM_p namespace. Notes longer than 5 s are split into consecutive segments, with continuation segments marked by INSTRUMENT_EXTEND_i or INSTRUMENT_EXTEND_DRUM. Velocity is omitted.

A sample begins with SS and a target-instrument declaration between TARGET_START and TARGET_END; BOS and EOS mark song boundaries when present. Target and conditioning groups are interleaved using mono-constrained delta conditioning. Conditioning note groups replace their instrument token with the corresponding CONDITION_ token, while sharing pitch, duration, and time tokens with target groups. CONDITION_EOS marks the end of conditioning input. Temporal realignment operates on musical time at the time-group level without changing the serialized order.

The model uses distinct input and output time representations. Input TIME_t tokens encode absolute onset time, $0.01t$ seconds, while output time classes encode target-time increments

$$\Delta t_k = t_k - t_{k-1}. \tag{27}$$

The target clock therefore advances only on target events; intervening conditioning times do not affect it. Empty time groups bridge long gaps so that $\Delta t_k \leq 100$ steps (1 s). During generation, each predicted increment is added to the target clock and converted back to an absolute-time input token. This preserves a shared timeline under reordered conditioning while keeping output timing relative.

Table 3 summarizes the vocabulary. The model uses 61,280 input tokens and 1,117 output classes, including 60,001 absolute-time input tokens and 101 relative-time output classes.

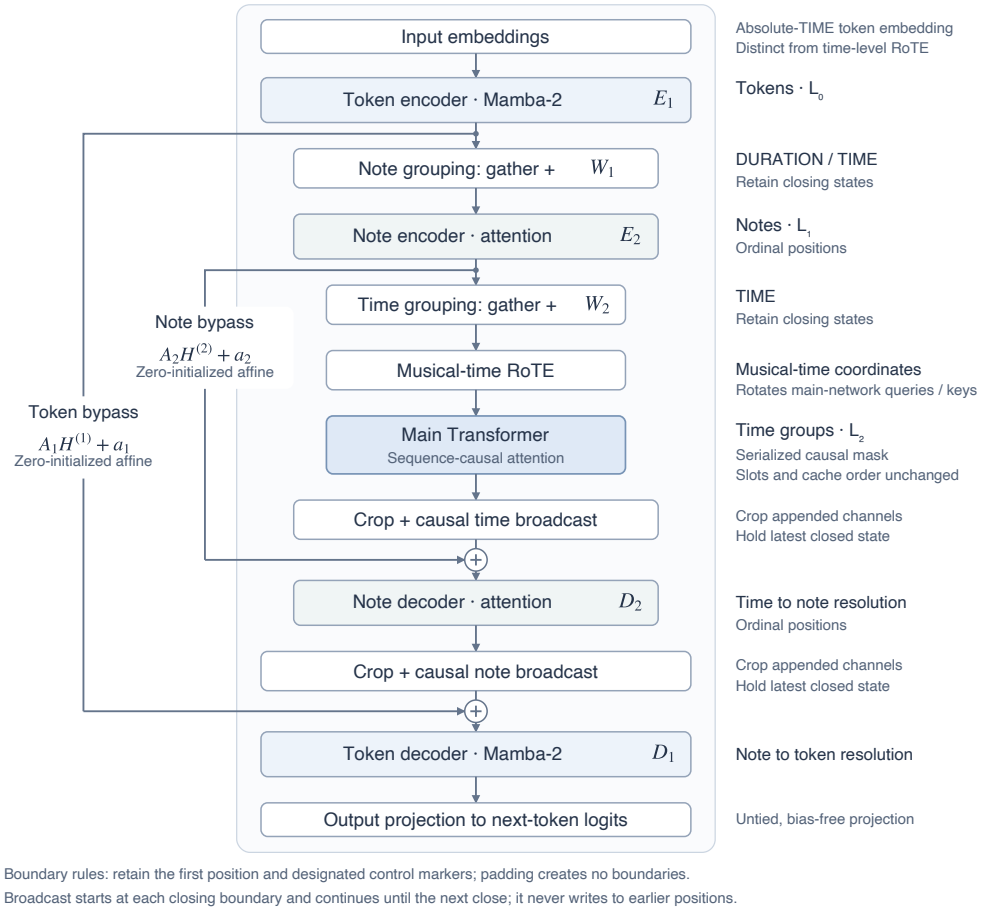

Figure 6: Two-stage H-Net adaptation for symbolic music. Box widths indicate sequence resolution schematically. Each grouping operation retains causal encoder states at closing boundaries; each ungrouping operation broadcasts the latest available processed state and applies a decoder after adding an affine encoder bypass. Both bypasses are initialized to zero. Time and control markers are retained as required by the hierarchy; the boundary labels show the musical delimiters. Musical-time RoTE changes the main Transformer's attention coordinates while outputs remain in the sequence slots used for ungrouping.

F Model architecture

Figure 6 expands the grouping, realignment, and ungrouping flow of Figure 3 into its constituent networks. The diagram follows the layout of the conceptual overview and adds the causal encoders, decoders, and bypass connections.

H-Net and our adaptation. H-Net (Hwang et al., 2026) organizes an autoregressive model as an encoder, a chunking operation, a main network at reduced sequence resolution, a dechunking operation, and a decoder at the original resolution. Replacing the main network with another such stage produces a recursive hierarchy. The original H-Net learns content-dependent boundaries with a router and uses boundary-dependent smoothing and a compression-ratio objective to train this discrete segmentation. We adopt the nested architecture and encoder bypasses, but obtain boundaries directly from the music representation. Consequently, our model requires neither a learned router nor a routing or compression-ratio loss. Its hierarchy is token $\rightarrow$ note $\rightarrow$ time, with two compression stages and three sequence resolutions. Here, "encoder" and "decoder" name causal networks before and after compression; every network processes only the available prefix.

Hierarchical processing. Our model is a causal autoregressive network with token, note, and time resolutions. Two nested encoder–decoder stages, inspired by H-Net (Hwang et al., 2026), implement the grouping and ungrouping operations in Section 3. A Mamba-2 encoder processes token embeddings and groups them into note-level states. An attention encoder then groups these states into time-level representations, which are processed by the main Transformer. Attention and Mamba-2 decoders restore note and token resolution, respectively. Target and conditioning tokens share this network and remain in their delta-conditioned sequence order.

The attention encoder and decoder use ordinal positions in the note-level sequence. The main network alone uses musical-time attention coordinates. Encoder states are contextual summaries: a state retained at a note or time boundary can depend on earlier groups as well as the group it closes. Grouping therefore selects states from a causal sequence model rather than independently encoding each group. Mamba-2 handles the longest, token-level sequence, while attention operates on the shorter sequences. If their lengths are L_0, L_1, and L_2, attention pair counts scale as $O(L_1^2)$ at note resolution and $O(L_2^2)$ in the main network. Actual compression depends on the number of notes per onset and retained markers; sequence-length savings alone do not determine end-to-end runtime.

Grouping at closing boundaries. Let $X^{(g-1)}$ be the input to stage $g \in \{1, 2\}$ and let $b_{g,1} < \cdots < b_{g,L_g}$ be its closing positions, indexed at that stage's input resolution. The encoder and grouping operations are

$$H^{(g)} = E_g(X^{(g-1)}), \qquad X_j^{(g)} = W_g\left(H_{b_{g,j}}^{(g)}\right), \tag{28}$$

where W_g matches the width expected by the inner network. When the inner width is larger, it appends learned channels to the retained vector; the original channels are preserved. On the return path, extra channels are removed before expansion to the outer resolution.

At the first stage, `DURATION` closes a note group and `TIME` is also retained. At the second stage, the retained `TIME` state closes the time group. Control markers designated by the representation survive the relevant stages. The first sequence position is always retained, ensuring an available initial state, whereas padding creates no boundaries. For example, a two-note onset has the form

$$\underbrace{I_1\ P_1\ D_1}_{\text{note 1}} \quad \underbrace{I_2\ P_2\ D_2}_{\text{note 2}} \quad T_t. \tag{29}$$

Here I, P, D, and T abbreviate instrument, pitch, duration, and time tokens. The first stage retains states at D_1, D_2, and T_t; the second retains the state at T_t. The latter has observed both notes and the token specifying their onset. An empty time group consisting only of T_t is valid and survives both stages.

Causal ungrouping and bypasses. Let $Z^{(g)}$ denote the processed sequence returned by the inner network, cropped to the current stage's width. For an input position i, define

$$j_g(i) = \max\{j : b_{g,j} \le i\}, \qquad V_i^{(g)} = Z_{j_g(i)}^{(g)}. \tag{30}$$

Thus a processed group state becomes available at its closing position and is held until the next close. It is never copied into positions before its boundary. The decoder receives this state together with an affine bypass of the encoder output at the same resolution:

$$Z^{(g-1)} = D_g\left(V^{(g)} + A_g H^{(g)} + a_g\right), \qquad A_g = 0,\ a_g = 0 \quad \text{at initialization.} \tag{31}$$

The learned bypass supplies position-specific information that would otherwise be lost when several positions share one group state. Its zero initialization starts the model with the compressed path and lets training learn the contribution of the bypass.

In the example above, the state at D_1 can use the completed first note to predict the next token, I_2. Positions I_2 and P_2 still receive the inner-network output from the most recent close, while their own encoder states contribute through the token bypass. The second note's completed summary first becomes available at D_2, and the current time group's main-network output first becomes available at T_t. At every position i, the final state predicts token $i + 1$ using only tokens through i. A trailing incomplete group continues to use the preceding completed state. The same rule supports cached generation: the outer stacks advance at every token, the note-level stacks advance at outer boundaries, and the main network advances only at inner boundaries.

Embeddings and temporal realignment. Non-time tokens use learned embeddings. Each absolute-time input token `TIME_`t is embedded using learned seed vectors rotated by t. This input embedding is distinct from time-level RoTE: the former encodes a token's value, while the latter determines temporal relationships in the main Transformer's attention. Let τ_j be the musical-time index specified by the `TIME` token closing time group j. For query and key vectors q_j and k_ℓ, time-level attention uses

$$\widehat{q}_j = R(\tau_j)q_j, \qquad \widehat{k}_\ell = R(\tau_\ell)k_\ell, \qquad \widehat{q}_j^{\mathsf{T}}\widehat{k}_\ell = q_j^{\mathsf{T}}R(\tau_\ell - \tau_j)k_\ell, \tag{32}$$

where R is the rotary transform. Temporal relationships therefore use musical-time differences. The causal mask still permits only $\ell \leq j$ in serialized sequence order. A future-sounding conditioning group can be attended to if it has already appeared in that order; a later sequence position remains masked regardless of its musical timestamp. Distinct conditioning and target groups may share a timestamp without being merged. Retained markers without an associated musical time use a reserved prefix coordinate.

The realignment operators in Eq. 18 describe this change of temporal coordinates. The implementation leaves sequence slots and cache order intact throughout. Consequently, returning to ungrouping requires no sorting or inverse rotation of output states: the same boundary indices already identify their destinations. Replacing τ_j with ordinal main-stack positions gives the no-realignment comparison in Section G.2.

Autoregressive output and training. An untied, bias-free projection maps $Z_i^{(0)}$ to logits for the next token's output class. Time classes encode increments from the preceding target time; conditioning timestamps do not advance that target clock. During generation, each sampled increment is converted to an absolute-time input token before the next model step, so both the boundary decision and RoTE coordinate use the reconstructed input time. Representation and vocabulary details are given in Appendix E. All encoders, decoders, embeddings, bypasses, and the main network are trained jointly with next-token cross-entropy on valid prediction labels. The deterministic gathers and broadcasts propagate gradients to their selected states without learning the boundary indices. Parameter settings and model comparisons are given in Section G.2.

G Experimental details

G.1 Lakh MIDI Dataset Splits and Duplicate Filtering

We use the LMD-full subset of the Lakh MIDI Dataset, containing 178,561 MIDI files. Following the split convention of the Anticipatory Music Transformer, we assign files to splits according to the first hexadecimal character of their MD5 filenames: `0`–`d` for training, `e` for validation, and `f` for testing.

To reduce cross-split duplication, we use the precomputed duplicate clusters released by Choi et al. (2025), with similarity thresholds of 0.93 for CAugBERT and 0.87 for CLaMP-1024. This is the most aggressive of their three released filtering lists. The clusters are connected components formed from the union of the two models' duplicate detections. We treat each JSON key and its associated list as one cluster, rather than directly applying the authors' representative-selection policy.

Our filtering policy gives priority to training, followed by testing, and then validation. If a cluster contains a training file, we exclude all of its validation and test files. Otherwise, if a cluster contains both validation and test files, we exclude its validation files and retain its test files. Only cluster members present under the configured dataset root are considered. Training files remain unchanged, duplicates within the same split are retained, and files absent from the detection list are retained.

Table 4: LMD-full split sizes before and after cross-split duplicate filtering.

Split	Original	Excluded	Retained
Training	156,256	0	156,256
Validation	11,076	5,552	5,524
Test	11,229	5,524	5,705

Table 5: Architecture variants of our model. Each tuple is $(L, d, d_{\text{ff}}, H_q)$: number of layers, hidden width, feed-forward width, and number of query heads. A dash indicates that the Mamba-2 stack has no separate feed-forward sublayer or attention heads. Attention uses four key/value heads and head dimension 64 throughout. Encoder and decoder entries give the settings for each stack separately.

Stack	Resolution / mixer	Balanced	Inner	Outer
Outer encoder	Token / Mamba-2	$(2, 512, -, -)$	$(2, 512, -, -)$	$(2, 512, -, -)$
Inner encoder	Note / attention	$(6, 512, 2048, 8)$	$(2, 512, 2048, 8)$	$(6, 768, 3072, 12)$
Main	Time / attention	$(8, 768, 3072, 12)$	$(12, 768, 3072, 12)$	$(2, 768, 3072, 12)$
Inner decoder	Note / attention	$(6, 512, 2048, 8)$	$(2, 512, 2048, 8)$	$(6, 768, 3072, 12)$
Outer decoder	Token / Mamba-2	$(2, 512, -, -)$	$(2, 512, -, -)$	$(2, 512, -, -)$
Total mixer blocks		24	20	18

Of the 5,552 excluded validation files, 5,439 belong to clusters containing training files, and an additional 113 are excluded solely because their clusters contain test files. Filtering is reapplied to the in-memory file index at every validation or test dataset initialization; the original MIDI files are never modified or deleted. The detection list is stored locally and downloaded from a pinned upstream revision if missing. This procedure removes cross-split overlap identified by the supplied clusters; it does not guarantee the absence of undetected duplicates.

G.2 Model configurations and comparisons

The hierarchical models share the architecture described in Appendix F. We compare their allocation of capacity across resolutions and the effect of temporal realignment.

Shared settings. Each token-level stack contains two Mamba-2 blocks of width 512, with state size 128, convolution width 4, expansion factor 2, and head dimension 64. The note-level and main stacks use Qwen3.5 full-attention blocks. Attention dropout and projection biases are disabled, and RMS normalization uses $\epsilon = 10^{-6}$. Absolute-time input embeddings concatenate four learned 128-dimensional seeds with rotary base 10^6. Note-level attention uses rotary base 5×10^5, and main-stack attention uses base 10^6.

Realignment comparison. For each hierarchical configuration, we train a counterpart that uses ordinal main-stack positions in place of musical-time coordinates. The architecture, absolute-time input embeddings, conditioning order, and training settings are otherwise shared, isolating the effect of time-level temporal realignment.

Model variants. Table 5 gives the three parameter allocations. Balanced uses $6/8/6$ attention layers across the note-level encoder, main stack, and note-level decoder. Inner shifts capacity to the main stack ($2/12/2$), while Outer uses $6/2/6$ and widens the note-level stacks to 768. All three use an untied, bias-free $512 \times 1{,}117$ output projection and a maximum context of $4{,}096$ tokens. Vocabulary details are given in Appendix E.

Vanilla operates directly on tokens with 14 Transformer layers, hidden width 768, feed-forward width $3{,}072$, 12 query heads, and 6 key/value heads of dimension 64. It uses ordinal token positions, without grouping or temporal realignment, and shares the symbolic vocabulary and training protocol.

Audio conditioning. For drum generation, a frame-level ConvNeXt encoder supplies accompaniment audio latents to the main network. The same encoder design is used with and without temporal realignment; details are in Appendix H.

G.3 Training and inference settings

Training data and preprocessing. For symbolic conditioning, we train Balanced, Inner, Outer, their no-realignment counterparts, and the Vanilla Transformer from scratch on the Lakh MIDI Dataset (Raffel, 2016) (`lmd_full`), using the deduplication information of Choi et al. (2025). We partition files by the first hexadecimal character of their filename: `0`–`d` for training, `e` for validation, and `f` for testing. Cross-split duplicate filtering removes held-out counterparts of training files; the

Table 6: Shared training settings for the symbolic models and the Vanilla baseline.

Setting	Value
Maximum input sequence length	4,096 tokens
Global batch size	16 sequences
Gradient accumulation	1 batch
Optimizer updates	200,000
Optimizer	AdamW
AdamW (β_1, β_2)	$(0.9, 0.99)$
Weight decay	0.1
Peak learning rate	10^{-4}
Minimum learning rate	10^{-5}
Warmup updates	2,000
Decay schedule	Cosine
Gradient norm limit	1.0
Precision	bfloat16 mixed
Random seed	1234

training split is retained. Training files are shuffled, and training sequences contain at most $4{,}096$ input tokens, including the target-instrument prefix. Longer sequences are randomly cropped and shorter sequences are padded. We enable crop eviction and refill so that sampled contexts also expose the model to the removal of earlier events. After this selection, we subtract the earliest retained time across the target and conditioning streams from all time tokens.

Before tokenization, we transpose melodic notes by a shared random integer offset of up to six semitones in either direction, restricted to keep pitches within the MIDI range; drum pitches are unchanged. We also add a shared random onset offset between 0 and 10 ms before quantizing to a 10 ms grid. Additional time shifting after rebasing is disabled. Note segments are limited to 5 s, and relative target-time increments to 1 s, with extension segments and empty time groups representing longer durations and gaps, respectively.

Conditioning and training objective. For each sampled sequence, we draw an anticipation horizon uniformly from 500 to $2{,}000$ quantization steps (5–20 s) and interleave the conditioning events ahead of their sounding times. We draw one of four training tasks: unconditional generation, instrument conditioning, span conditioning, or random-note conditioning. For instrument conditioning, we draw the number of target instruments uniformly from 1 to $n-1$, where n is the number of available instruments, then sample a subset of that size; the remaining instruments provide conditioning. Span conditioning supplies passages whose lengths are drawn uniformly from 5 to 10 s, separated by exponentially distributed waits with a mean of 20 s. Notes outside those supplied passages form the target. Random-note conditioning draws a conditioning probability uniformly from $\{0.1, 0.2, \ldots, 0.9\}$ and samples individual notes at that probability. An inapplicable instrument split or a split leaving no target notes falls back to unconditional generation.

The symbolic models reported in Table 1 assign probabilities 0.1, 0.4, 0.1, and 0.4 to these four tasks, respectively. We minimize autoregressive cross-entropy, averaged over valid target labels. Conditioning and padding labels are excluded from the loss. If cropping or eviction removes the preceding target-time context, the first target-time label is also masked.

Optimization. All symbolic model variants and the Vanilla baseline share the optimization settings in Table 6. We train for $200{,}000$ optimizer updates with a global batch size of 16 sequences and no gradient accumulation. AdamW applies weight decay only to parameters with two or more dimensions, excluding biases and normalization gains. The learning rate increases linearly during the first $2{,}000$ updates and then follows a cosine decay toward 10^{-5}. We use bfloat16 mixed precision and clip the gradient norm to 1.0. The reported comparisons use the fixed $200{,}000$-step checkpoints; training-time validation sampling is disabled.

Inference data and conditioning. For automatic evaluation under symbolic conditioning, we prepare one randomly selected eligible 30-second window per selected test song and use the same prepared windows for all systems. The first 5 s provide the prompt, and the generation interval is $[5, 30)$ s. In accompaniment generation, the selected melody provides conditioning throughout

the window, while the first 5 s of the remaining parts provide the target history. In random-note conditioning, all notes in the first 5 s form the prompt; approximately 25% of the notes in the remaining interval are sampled without replacement as conditioning, and the remaining notes form the reference target.

Decoding. We use the frozen $200{,}000$-step checkpoints with bfloat16 mixed precision and a maximum context of $4{,}096$ tokens. Tokens are sampled autoregressively with temperature 1.0 and nucleus sampling at $p = 0.95$. The initial sampling seed is 1234 for each item. Anticipation conditioning uses a fixed 10-second lookahead, a special case of delta conditioning; the horizon is held constant throughout generation. The sampler enforces the note grammar and, for accompaniment, restricts generated instruments to the declared target set. Conditioning events are supplied as generation advances, and the target timeline is limited to the prepared window's endpoint.

Context management and completion. When the context fills, the sampler evicts earlier events using an eviction share of $1/16$ and rebuilds its caches from the retained context. Input times are rebased after eviction while preserving the output timeline. Each generation attempt has a safety limit of $1{,}000{,}000$ new tokens; completion is determined by reaching the target time horizon. If an attempt stops before completing the target interval, generation restarts from the same prompt and conditioning with the next seed, for at most three attempts in total (seeds 1234, 1235, and 1236). We retain the first completed attempt, or the final incomplete attempt if all three stop early, and record the completion status and attempt metadata.

Audio-conditioned evaluation. For audio conditioning, the target is the drum track and the conditioning signal is accompaniment audio synthesized from MIDI. A frame-level ConvNeXt encoder supplies audio latents directly to the main network. The comparison uses the same encoder with and without temporal realignment and evaluates 100 Lakh test songs, with a 5-second human drum prompt per 30-second clip. PHALAR measures coherence between generated drums and the conditioning accompaniment. The audio encoder and its training setup are described separately in Appendix H.

G.4 Metrics

Perplexity. We evaluate next-token prediction by teacher forcing the human reference. Let $\mathcal{I}$ contain the scored target positions across the evaluation set, and let y_k denote the reference output class at position k. Perplexity is

$$\mathrm{PPL} = \exp\left(-\frac{1}{|\mathcal{I}|}\sum_{k\in\mathcal{I}} \log p_\theta(y_k \mid \mathbf{x}_{<k})\right), \tag{33}$$

where $\mathbf{x}_{<k}$ includes the available reference history and supplied conditioning. Lower values are better. Losses and token counts are summed across examples before exponentiation, giving a token-weighted corpus score. Sampling temperature, nucleus truncation, and generation-time grammar constraints are not applied to the likelihood logits.

For the symbolic models, prompt, conditioning, padding, and positions outside the target interval are excluded from the loss. Relative target-time classes, including the clocks that bridge silent intervals, are scored. Reference sequences use the same fixed 10-second lookahead as generation. Long sequences are evaluated in overlapping contexts of at most $4{,}096$ tokens with stride 256; input times are rebased within each context and each selected label is scored exactly once. We also report PPL_t, PPL_i, PPL_p, and PPL_d by restricting $\mathcal{I}$ to time, instrument, pitch, and duration labels, respectively. Each loss uses the full output vocabulary. Drum and extension tokens belong to their corresponding attribute families; any scored special markers contribute to overall PPL. PPL comparisons depend on the representation and output vocabulary.

PHALAR. PHALAR (Marincione et al., 2026) measures coherence between paired audio stems, with higher scores indicating greater coherence. For symbolic accompaniment, we pair the generated accompaniment with the supplied melody; the human accompaniment paired with that same melody provides the reference. For audio-conditioned drum generation, the pair consists of generated drums and conditioning accompaniment audio.

The symbolic evaluation uses the pretrained `PHALAR_best.ckpt` encoder and symmetrizes its learned pair similarity over the two stem orders. MIDI stems are synthesized with FluidSynth and a shared SoundFont at velocity 90, then represented as mono audio at 16 kHz. We score non-overlapping 5-second windows within the target interval, excluding the initial 5-second prompt and incomplete final windows. A window is retained only when generated target, conditioning, and human target are all non-silent; silence is defined by peak absolute waveform amplitude below 10^{-3}. Scores are averaged over retained windows within each excerpt and then equally across scored excerpts. Silent windows receive no score, and excerpts without a valid window are excluded. Generation failures, incomplete generations, and invalid MIDI artifacts are also excluded from PHALAR scoring.

We report raw PHALAR scores, with human-reference scores shown alongside the generated systems for comparison.

Human preference. Listening judgments are summarized as wins, ties, and losses for each system pair. Significance tests use dense-rank differences with participant-level label swaps and Holm correction, as specified in Section G.6.

G.5 Listening Test Protocol

Listening-test design. We use a web-based, system-blinded preference-ranking study to compare music generation under random conditioning and melody-conditioned accompaniment. The three compared versions are human-composed ground truth, the Balanced model, and the Balanced model without temporal realignment. The labels "Original" and "No-reorder" in Table 2 refer to these two model versions, respectively. Each page presents two separate comparison groups from different source songs: random-note conditioning first and accompaniment second. Within each group, listeners compare the versions of the same excerpt and rank them from most to least preferred according to overall musical quality. The instructions suggest rhythmic stability and harmony as considerations, without collecting separate ratings for these dimensions. Versions judged approximately equally preferable may share a rank, including a tie among all versions.

Source excerpts and conditioning tasks. The preparation procedure samples one random 20-second excerpt from each of 150 distinct eligible songs in our deduplicated Lakh test split, using seed 0. Onsets and durations are quantized to a 10 ms grid, note velocities are fixed to 80, and notes extending beyond the excerpt are truncated at its endpoint. The same prepared excerpt is used for both conditioning tasks and for all compared systems.

Each excerpt must contain 4–10 non-drum instruments, counted by MIDI program. Among eligible non-drum parts, we select the part with the highest mean note pitch as the melody, excluding zero-based programs 0, 9, and 112–127. The selected melody must contain at least 20 notes, begin by 5 s, and have its last onset at or after 18 s; otherwise the excerpt is rejected. In accompaniment, the melody provides conditioning throughout the excerpt, while the first 5 s of the other parts provide the prompt. The model generates the remaining parts over $[5, 20)$ s. In random-note conditioning, all notes in the first 5 s form the prompt; approximately 25% of the subsequent notes are sampled without replacement as conditioning, and the remaining notes form the reference target.

The listening-study generation configuration uses a fixed anticipation horizon of 10 s and initial sampling seed 1234. An attempt that stops before completing the target interval is retried with the next seed, for at most three attempts in total. Model stimuli use the 200,000-step checkpoints, temperature 1.0, and nucleus sampling at $p = 0.95$, consistent with the symbolic generation settings in Section G.3.

Stimulus catalog and rendering. Only excerpts with completed generations from every selected system in both tasks are eligible for the listening catalog. The generated results must match the prepared excerpt metadata. From this common pool, the website selects up to 100 songs with fixed seed 0 and holds the catalog fixed for the study session. The default assignment presents each selected song once per task, giving up to 100 pages and 200 separate rankings per listener. Completion of the entire assignment is optional.

Each model stimulus is the full arrangement comprising its generated target, the shared prompt, and the supplied conditioning notes. The prepared human source is included as an additional, unlabeled reference version wherever its source MIDI is available. The conditioning-only input is not a ranked candidate. All versions are synthesized through the same FluidSynth renderer and SoundFont and delivered as OGG audio. A synchronized piano-roll visualization accompanies each version, with identifying MIDI track text removed.

Playback exposes the full prepared 20-second excerpt, including the shared 5-second prompt. The on-screen instructions specifically ask listeners to focus their comparison on 5–15 s. This is an instructed comparison interval; playback is not cropped to that interval. Listeners may replay the versions freely.

Blinding and assignment. The three versions are displayed only as anonymous labels A, B, and C; model identities and the human-reference identity are withheld from the participant interface. Version order is randomized separately for each participant, source excerpt, and task. Source orders are randomized separately for the two tasks and paired subject to the constraint that the two groups on a page come from different songs. The task display order remains fixed. Assignments persist when a listener resumes the study, and anonymous labels do not denote a consistent system across groups.

Participants and procedure. The saved study results contain 15 registered participants, of whom 12 submitted rankings. These participants contributed 100 paired pages, yielding 100 rankings per task and 200 task rankings in total. The submitted rankings cover 66 distinct excerpts for accompaniment and 68 for random conditioning. The three participants without submissions are excluded from the statistical analysis.

Before listening, participants explicitly confirm that they are at least 18 years old and consent to participation. They then report their years of musical study or training, including formal lessons and self-study without double-counting overlapping years. The five response categories are none, less than one year, one to less than five years, five to less than ten years, and ten years or more. The interface supports Traditional Chinese and English. Participants are advised to use headphones in a quiet environment at a comfortable volume.

All versions initially appear unranked. Listeners construct an ordered list of rank groups using drag-and-drop or equivalent selection controls. For a new page, the interface requires playback to have started for every version and a complete ranking in both tasks before submission; it imposes no minimum listening duration. Both task rankings are saved together. Participants may stop at any time, resume later, or revise previously submitted rankings. The latest submitted ranking replaces the earlier answer and contributes only one response. An email-derived hashed identifier links assignments, consent, and responses across visits; the response database does not store the raw email address.

Response representation and descriptive analysis. Each submitted task ranking is stored as consecutive integer ranks, with rank 1 indicating the most preferred group and equal integers indicating a tie. For example, $(1, 2, 2)$ records one preferred version followed by two tied versions. Analysis is performed separately for random-note conditioning and accompaniment. For every pair of systems appearing in the same submitted ranking, the response contributes one win, tie, or loss according to their relative ranks. We also summarize each system's mean dense rank and number of first-place appearances, including tied first places. Only submitted responses from the current study catalog are included, including those from participants who complete only part of the assignment. The website produces these descriptive summaries; significance testing for the paper is a separate analysis.

Reported preferences. Table 7 reproduces the pairwise counts in Table 2, with model names expanded for clarity. Each comparison contains 100 judgments; this is a count of judgments, not a participant count. Under random conditioning, Balanced is preferred to its counterpart without realignment in 67 judgments, tied in 19, and less preferred in 14. For accompaniment, the corresponding counts are 41, 25, and 34. These counts summarize individual judgments; the participant-level significance analysis is reported in Section G.6.

Table 7: Listening preferences from the perspective of Balanced. Counts match the main manuscript's Original comparisons.

Task	Compared against	Win	Tie	Loss
Accompaniment	Human composition	16	20	64
	Balanced w/o realignment	41	25	34
Random	Human composition	27	27	46
	Balanced w/o realignment	67	19	14

Table 8: Exact paired permutation tests with participant-level label swaps. Negative mean rank differences favor Balanced. K counts assignments at least as extreme as the observation, out of 4,096. Bold denotes Holm-adjusted $p < 0.05$ across four tests.

Task	Compared against	Mean	K	Raw p	Adjusted p
Accompaniment	Human composition	0.62	6	0.001465	**0.005859**
	Without realignment	−0.09	1120	0.273438	0.273438
Random	Human composition	0.26	12	0.002930	**0.008789**
	Without realignment	−0.70	24	0.005859	**0.011719**

G.6 Statistical tests

We use a two-sided exact paired permutation test, exchanging system labels at the participant level. Rankings use dense ranks: $A > B > C$ receives $(1, 2, 3)$, while $A = B > C$ receives $(1, 1, 2)$. For each task and system pair, we compute the rank difference for judgment j from participant i:

$$d_{ij} = r_{ij,\text{Balanced}} - r_{ij,\text{comparison}}, \qquad T_{\text{obs}} = \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{n_i} d_{ij}, \tag{34}$$

where $m = 12$, n_i is the number of judgments from participant i, and $N = \sum_i n_i = 100$ for each task. Negative differences favor Balanced. Each judgment has equal weight, so participants who submit more judgments contribute more to the statistic. Ties contribute zero and remain in the denominator. Rank differences distinguish a one-rank advantage from a two-rank advantage; the win/tie/loss counts are descriptive summaries and are not the input to the test.

Under the null hypothesis, the two system labels are exchangeable across all responses within each participant. Assuming independent participants, we enumerate all $2^{12} = 4{,}096$ sign assignments $\boldsymbol{\epsilon} \in \{-1, +1\}^m$ and compute

$$p = \frac{1}{2^m} \sum_{\boldsymbol{\epsilon} \in \{-1,+1\}^m} \mathbf{1} \left\{ \left| \frac{1}{N} \sum_{i=1}^{m} \epsilon_i \sum_{j=1}^{n_i} d_{ij} \right| \geq |T_{\text{obs}}| \right\}. \tag{35}$$

The same sign is applied to all rank differences from a participant, preserving the dependence among that participant's repeated judgments. Participants with a zero sum remain in the enumeration. This is an exact enumeration, with no Monte Carlo sampling. We apply Holm correction across the four comparisons reported in the manuscript: Balanced against human composition and against its no-realignment counterpart, for each of the two tasks. Statistical significance is assessed at adjusted $p < 0.05$.

After correction, human composition is significantly preferred to Balanced in both tasks. Balanced is significantly preferred to its no-realignment counterpart under random conditioning (adjusted $p = 0.011719$), while the accompaniment comparison does not reach significance (adjusted $p = 0.273438$).

This analysis conditions on the study's excerpt catalog and does not fit a song random effect or establish generalization to a population of songs. Perplexity and PHALAR comparisons are reported descriptively, without significance tests here.

Table 9: Audio encoder layers. Shapes are per clip, with $F = 84$ CQT bins and $T = 3000$ frames for a 30-second clip at 100 Hz. The input is normalized to peak-relative dB, floored at -80 dB and scaled to $[-1, 1]$, before the learned per-bin affine. The projector maps the latents to the main network's width and is trained with the encoder.

Layer	Configuration	Output	Params
Per-bin affine	scale and bias per bin	84×3000	168
Spectral flux	$\max(x_t - x_{t-1}, 0)$, concatenated	168×3000	0
Frame projection	1×1 conv, $168 \to 256$	256×3000	43K
ConvNeXt blocks	$4\times$ (depthwise k=7, LN, MLP $256 \to 1024 \to 256$)	256×3000	2.11M
Window pooling	LN; conv $256 \to 768$, kernel 20, stride 10	768×300	3.93M
Output norm	LN	768×300	1.5K
Projector	linear $768 \to 768$, GELU, linear $768 \to 768$, RMSNorm	300×768	1.18M
Total			7.27M

H Audio Conditioning Evaluation Details

H.1 Audio Encoder

The accompaniment audio is synthesized from the clip's MIDI with all drum tracks removed, using the GeneralUser GS SoundFont[1] at 16 kHz, and converted to a log-magnitude CQT with 84 semitone bins (C1–B7) and hop size 160, i.e., 100 frames per second.

The audio encoder is a frame-level ConvNeXt encoder. It applies four ConvNeXt blocks with kernel size 7 at the full 100 Hz CQT frame rate. Each 10 Hz latent of width 768 is then formed from the corresponding 10 frames together with 5 neighboring frames on each side, using separate weights for individual frames, so that each latent retains frame-level timing within its window. A 30-second clip therefore yields 300 audio latents. Table 9 lists the layers.

Within $\mathcal{M}_\theta$, the audio latents are inserted into the time-level sequence immediately after the time-level state of the start-of-sequence (`SS`) token, preceding the first time group. They serve purely as conditioning: the outputs of $\mathcal{M}_\theta$ at audio positions are discarded before ungrouping ($\mathcal{U}_2$, $\mathcal{U}_1$), so no prediction or loss is computed at those positions. They serve purely as conditioning: the outputs of $\mathcal{M}_\theta$ at audio positions are discarded before ungrouping ($\mathcal{U}_2, \mathcal{U}_1$), so no prediction or loss is computed at those positions.

H.2 Temporal Realignment with Audio

With temporal realignment enabled, the main-network attention measures relative distance in musical time rather than sequence position. Each audio latent is assigned its onset time, and a time-group query and an audio key corresponding to the same musical time therefore have zero relative temporal distance. The audio latents themselves are not rotated; their timing reaches the main network only through these attention positions.

In the *no-RoTE* variant, temporal realignment is disabled and the main-network attention instead follows sequence position. The two variants therefore differ only in the attention positions.

H.3 Training and Evaluation Setup

Both models use the `Inner` variant in Table 5, described in Appendix F.

Each training example is a 30-second clip cropped from a song at a random position: its start is drawn uniformly between the song's first onset and the latest point that still leaves a full 30 seconds of music, and is redrawn every time the song is visited. Notes sounding across the start of the clip are dropped. A clip is kept only if it contains at least one drum note and at least one non-drum note; otherwise a new start is drawn once more before the song is skipped. Training uses batch size 64, and AdamW with an initial learning rate of 10^{-4}, cosine decay to 10^{-5}, 2,000 warm-up steps, weight decay 0.1, and bf16 mixed precision. Both models are trained for 200,000 steps.

[1] `https://schristiancollins.com/generaluser.php`

For evaluation, we randomly sample 100 songs from the Lakh test split and divide each into non-overlapping 30-second clips starting from its first note. A clip is kept only if it contains both drum and non-drum notes, and songs without such a clip are skipped, yielding 667 clips in total. For each clip, the model is given the first 5 seconds of the human drums as a prompt and generates the remaining 25 seconds. Sampling uses top-$k = 50$, top-$p = 0.95$, and temperature 1.0.

We evaluate musical coherence using PHALAR (Marincione et al., 2026), the successor of CO-COLA (Ciranni et al., 2025). PHALAR independently embeds the drum stem and accompaniment and measures their agreement. All MIDI outputs are rendered with the same FluidSynth setup and SoundFont as the conditioning audio to minimize timbral variation.

PHALAR is computed over 5-second windows. The first 5-second window of each clip contains the ground-truth prompt and is excluded from scoring. A window is also excluded if any of the generated drums, accompaniment, or human drum stems has peak amplitude below the silence threshold of 10^{-3}. We report the mean and standard deviation over the windows that are valid for both models (2,454 five-second windows from 629 clips). The ground-truth score is obtained from the corresponding human drum performances and serves as a reference.