PhraseVAE and PhraseLDM: Latent Diffusion for Full-Song Multitrack Symbolic Music Generation

Longshen Ou
School of Computing
National University of Singapore
oulongshen@u.nus.edu

Ye Wang
School of Computing
National University of Singapore
wangye@comp.nus.edu.sg

Abstract

This technical report presents a new paradigm for full-song symbolic music generation. Existing symbolic models operate on note-attribute tokens and suffer from extremely long sequences, limited context length, and weak support for long-range structure. We address these issues by introducing PhraseVAE and PhraseLDM, the first latent diffusion framework designed for multitrack symbolic music. PhraseVAE compresses variable-length note sequences into compact 64-dimensional phrase-level representations with high reconstruction fidelity, allowing efficient training and a well-structured latent space. Built on this latent space, PhraseLDM generates an entire multi-track song in a single pass without any autoregressive components. The system eliminates bar-wise sequential modeling, supports up to 128 bars of music (8 minutes in 64 bpm), and produces complete songs with coherent local texture, idiomatic instrument patterns, and clear global structure. With only 45M parameters, our framework generates a full song within seconds while maintaining competitive musical quality and generation diversity. Together, these results show that phrase-level latent diffusion provides an effective and scalable solution to long-sequence modeling in symbolic music generation. We hope this work encourages future symbolic music research to move beyond note-attribute tokens and to consider phrase-level units as a more effective and musically meaningful modeling target.[1]

1 Introduction

The significance of the song is paramount. When engaging with music, one interacts with complete compositions or pieces, not fragments. In musical performance, music is executed at the song level. For educational purposes, music is learned through individual songs. Composers and arrangers create music in the form of songs. Artists market their compositions as songs. The rationale is straightforward: analogous to how written content is structured into articles and visual media into complete films or television episodes, the song represents the most prevalent form of auditory art that offers the audience a full, cohesive musical experience.

Despite the centrality of songs in musical practice, most music generation systems still operate at the segment level, producing short excerpts rather than complete musical works. This disconnect arises not from a lack of demand, but from the inherent difficulty of modeling long-form symbolic music. A full song typically spans tens or hundreds of bars, involves multiple sections with distinct structural functions, and expresses musical ideas through long-range development rather than isolated moments. Consequently, generating a song requires capturing not only local note-to-note coherence but also global structure, such as sectional form, thematic recurrence, harmonic pacing, texture arrangement, and long-term melodic evolution. Bridging this gap between short-form generative

[1] Demos and code: https://www.oulongshen.xyz/midi_ldm.

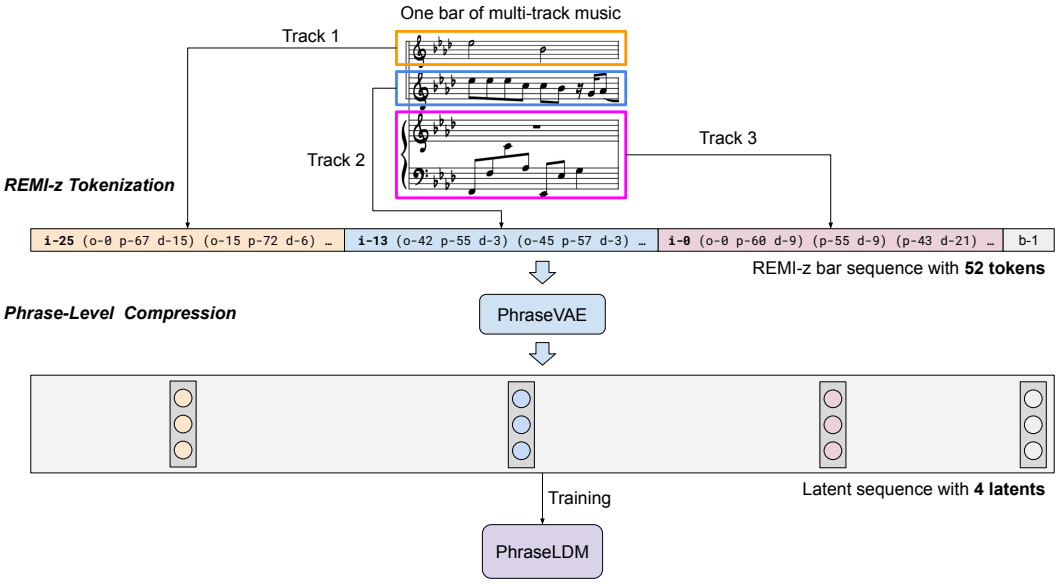


Figure 1: Data representation and framework overview.

models and real-world musical artifacts motivates the need for more effective full-song symbolic music generation.

Modeling music is an important topic in generative AI, yet full-song symbolic music generation remains relatively underexplored. Most existing approaches rely on sub-note-level tokenization, which we call *note-attribute token* in the paper—where a single note event is decomposed into multiple tokens—and autoregressive decoding. This design leads to extremely long token sequences, which introduces several inherent limitations. First, the computational cost becomes prohibitive: for instance, generating a 32-bar sample with MuseCoco can take nearly ten minutes, whereas typical users may expect results within seconds. Second, these systems generally support only segment-level generation because their context windows are insufficient to model an entire song. Third, the excessive sequence length makes it difficult to learn long-range musical structure, such as maintaining coherent melodic variation across repeated sections, especially under limited training data.

There are some initial explorations on full-song generation. These works are good early attempts, but they have two major limitations: (1) they focus on single-track music, and (2) they rely on song-level autoregression. In other words, the bars are still generated autoregressively, even though the generation of each bar is done through a diffusion process. This makes inference slow. Moreover, the authors believe that bar-wise autoregressiveness is not a good inductive bias for music. We argue that when observing similar bars or sections in music, we understand them as mutually related rather than strictly dependent in one direction. Even in human composition, composers and arrangers do not always create music in a monotonic left-to-right manner.

To address the challenges discussed above, we present a new hierarchical framework for full-song symbolic music generation. The system removes bar-wise autoregressive decoding so that it has high inference efficiency. It can be trained on a relatively small dataset, and its memory footprint is low enough that the entire 800-song corpus can be loaded in a few batches. The framework operates in a compact 64-dimensional phrase-level latent space, where each latent vector represents both the note content and timbre of one bar for a single instrument. While the latent diffusion model alone is sufficient for unconditional generation, placing additional length and structural conditions further improves its usability and long-range coherence.

Our main contributions are as follows:

- **A Phrase-Based Generation Paradigm.** We propose generating multitrack symbolic music at the phrase level—musically meaningful spans of note sequences—rather than using note-attribute tokens. This paradigm reduces the context length required by the full-song generation model from over 10k tokens to 512 latents, making full-song generation feasible.

- **PhraseVAE.** We introduce a phrase-level sequence compression model that maps variable-length note sequences into compact 64-dimensional latent vectors with near-perfect reconstruction quality (99.0% $F1_{op}$). PhraseVAE employs a multi-query compression mechanism that outperforms query-based and pooling-based alternatives, and a multi-stage training strategy that accelerates convergence and produces a well-regularized latent space.
- **PhraseLDM.** Built upon the latent space provided by PhraseVAE, we develop PhraseLDM, a latent diffusion model capable of generating an entire song in a single pass without any autoregressive components. PhraseLDM generates music at the phrase level, producing coherent local texture and harmony, together with decent structure planning ability. It demonstrates competitive unconditional generation quality, further improved by conditioning on sequence length and structure.
- **A Lightweight Full-Song Generation Framework.** Together, PhraseVAE and PhraseLDM form a lightweight yet capable full-song symbolic generation framework. The system contains around 45M parameters in total and can generate a complete multi-track song (up to 128 bars; roughly eight minutes at 64 BPM) within 3.3 seconds on an NVIDIA A40 GPU. This design enables fast, scalable, and practical full-song generation, supporting long-form multi-track compositions with realistic instrument usage and clear global structure.

2 Motivation

2.1 Semantics of Music

In natural language, a single word can convey rich and highly abstract meaning. A word functions as a compact pointer to real-world concepts, such as objects, actions, emotions, or events. For example, the word "encore" immediately evokes a complete situational context: an enthusiastic audience, repeated applause, the anticipation of an additional performance, and the emotional atmosphere surrounding it. In other words, a single lexical token can trigger a complex semantic representation in the human mind. This semantic density is one of the reasons why modern NLP builds on learning expressive representations for words and subwords, enabling models to encode the nuanced meaning they carry [9, 1].

In symbolic music, this semantic richness does not exist at the note or sub-note level. Most symbolic music models—including nearly all autoregressive ones—encode each note event by decomposing it into multiple note-attribute tokens, where each token represents only a single aspect of the event [5, 14, 17, 21, 13, 12, 16]. For example, in the REMI-z tokenization scheme [12], a token such as `o-12` merely indicates that an onset occurs at the first beat of a bar. It provides no information about the pitch, instrument, timbre, duration, expressive intent, or the musical role of the note (e.g., whether it belongs to a melodic line, comping pattern, padding texture, or arpeggio). In effect, the model in [12] learns a 768-dimensional embedding for that token whose semantic content is simply "something begins here." While the efficiency issues of such representations will be discussed later, the key point is that an individual note-attribute token carries almost no semantic meaning on its own, making it a poor unit for modeling music.

We hypothesize that the meaning of music is carried by note sequences, not by single note-attribute tokens. A note sequence can involve multiple instruments: when all the strings play together, it often gives a solemn feeling; when the drum fills the empty space before a new section, it suggests the music is moving forward. Even a single-track sequence can be expressive. A loose guitar strumming pattern can sound relaxed and casual, and a fast upward bass line can feel like the player is showing off. A solo guitar with an alternating-bass pattern can create a countryside atmosphere, while a slow free-style erhu line often suggests sadness or mourning. Listeners may not agree on the exact meaning of each example, but most would agree that these sequences do convey something or trigger some emotion [6, 15]. This is where the semantic meaning of music appears. These are the things that need to be captured in composition and in generative music models.

2.2 Previous Paradigms

There are two major paradigms of symbolic music generation.

2.2.1 Autoregressive Models

The first and most widely studied paradigm is autoregressive modeling. Many works follow this direction. Some explore large-scale training [14, 8, 17], some use attribute-based conditioning [8], some adapt instrumentation [12, 22], and some incorporate human preference into the training loop [17]. These are all valuable studies, and many of them can generate high quality music segments. However, listeners do not usually consume music in segments. Generating a full piece of music remains a major challenge for most autoregressive models.

To illustrate the difficulty of full-song generation problem, consider a typical multitrack dataset such as Slakh2100. A song in this dataset contains about 10,874 notes at the 95th percentile. Even with an efficient tokenization scheme that converts each note into only three attribute tokens, the sequence still exceeds 30k tokens—far longer than the context length supported by most existing symbolic music generation models. In practice, many autoregressive models are trained only on short segments instead, and they are simply not designed for full-song generation.

Many previous works have discussed this long sequence issue. Several attempts have been made to reduce sequence length [4, 2, 12], but the problem is still not fully solved. The most efficient tokenization schemes [4] use a single compound vector to represent a note event, but even this is not efficient enough (10k sequence length). This raises a natural question: is there a way to represent note sequences, which we believe are the basic units of composition, at a more abstract level while still preserving enough detail? If such a method exists, it could solve both the efficiency and context length issues, and it may also make long-range modeling easier, such as capturing relationships between different sections of a song.

2.2.2 Diffusion Models

The second paradigm is the use of diffusion models, with a focus on discrete diffusion models [10, 19] in recent years. In this setting, diffusion is applied directly in the data space of the piano roll, which is a matrix-like representation of short single-track music segment. The piano roll usually covers one or a few bars along the time axis and uses the pitch axis as the other dimension, where each cell stores duration and possibly velocity. The model generates one segment at a time. After finishing the diffusion process for that segment, it moves on to the next one, and continues in this way until the song is complete.

This paradigm has several issues. First, it is slow: each segment must go through a full diffusion process, and a full song requires repeating this many times. Second, the data dimension is very high. With a 48th-note resolution, one bar contains 6,144 channels of information in float numbers that the diffusion model must handle (in comparison, ours uses 152.3, as shown later). Third, current attempts are limited to single-track music, and it is not clear how this design can be extended to multitrack composition. Fourth, bar-wise sequential modeling is a questionable inductive bias. At the song level, musical ideas are often interdependent rather than strictly dependent in one direction. This design also causes a cadence problem. Music needs to end in specific ways at both the section level and the song level, but a one-directional model may fail to implement this correctly when a stop signal is given.

There is also an early attempt at applying latent diffusion to symbolic music [11]. It is a valuable demonstration that the LDM framework is, in principle, applicable to symbolic music modeling. However, many issues remain unresolved.

(1) **Insufficient VAE reconstruction quality.** The VAE reports only around 90% token-level accuracy under teacher forcing and around 80% accuracy when reconstructing directly from latents. This level of fidelity is far from enough to produce natural music with idiomatic playing patterns. A later VAE work [20] improves reconstruction quality, but we will show later that it does not handle 48th-note quantization well—a temporal resolution widely adopted in symbolic generation tasks.

(2) **Restricted to monophonic sequences.** The model is designed for monophonic note sequences only. It cannot represent complex polyphonic instruments such as piano or guitar, and the method does not generalize to multitrack music in a straightforward manner.

(3) **Limited long-term modeling support.** It supports sequences up to 32 bars, which is insufficient for most contemporary full-song compositions.

(4) **Limitations in music quality.** The LDM outputs are musically weak even for short monophonic segments. The generated samples lack naturalness, coherence, musicality, and structural organization. Overall, while the work is an early step toward applying LDM to symbolic music, the resulting music is far from sounding like real compositions, leaving substantial space for improvement.

3 Method

As discussed earlier, full-song generation becomes difficult when the model operates directly on note-attribute tokens, since each token encodes only a small part of a note and leads to extremely long sequences. Our approach is to compress music into a higher-level granularity that carries meaningful musical semantics, which greatly reduces the sequence length required for full-song modeling. The generation model then operates on this compact and abstract representation rather than on raw tokens.

3.1 Phrase and REMI-z Grammar

We now formally define the meaning of a *phrase* used in this paper. A phrase can have different meanings in different music research settings, but here it has a precise definition. A phrase refers to all information of what one instrument plays within a single bar. A phrase must contain the following: the instrument identity, and the set of notes played by that instrument in that bar. Each note includes its onset and duration, as well as its pitch. This is the minimum information required for a phrase, and it is also the level of representation that our generation model operates on.

When we prepare symbolic music using the REMI-z tokenization scheme [12], the concept of a phrase directly corresponds to the *track sequence* defined in the REMI-z paper. A phrase (or track sequence) contains the following tokens: it begins with an instrument token, followed by a sequence of note-related tokens. Each note is represented by three parts: an onset token that marks the relative starting position within the bar, a duration token, and a pitch token represented as an integer between 1 and 128. Both onset and duration are quantized with 48-note time resolution. The order of notes follows the chronological order of their onsets. Notes sharing the same onset position use only a single onset token, and their relative order is sorted by pitch, with higher pitch appearing earlier.

A bar contains multiple phrases, one for each instrument that plays in that bar. In REMI-z tokenization, a bar sequence is simply a list of phrase, and the order of the phrases is determined by the average pitch of each corresponding phrase, phrases with higher pitches appears earlier. A song is then formed by a sequence of bar sequences. Viewed this way, compressing music at the phrase level is intuitive and greatly reduces the complexity of the information that the generation model needs to handle.

3.2 PhraseVAE

We look for a unit of representation that carries meaningful semantics, is relatively independent from its surrounding context, interacts with other units to form musical patterns, and is not so complex that it requires a very large latent dimension to encode. We find that phrase-level compression fits this purpose well. Our goal is to learn a vector representation for a sequence of note events that contains all the essential information of that phrase: the instrument identity, the onset times, the pitches, and the durations. The method must also handle sequences of different lengths, since the number of notes played by different instruments, or even by the same instrument across different bars, can vary widely.

Based on this idea, we build a variational auto-encoder (VAE) that compresses a variable-length note sequence into a 64-dimensional vector with high reconstruction quality (99% F1 when measuring onset and pitch). As in Figure 2, the training procedure follows a three-step recipe. First, we train a sequence-to-sequence model using a span-infilling objective (§ 3.2.2). Second, we attach a bottleneck to this model and train it as an autoencoder for sequence compression (§ 3.2.1). Third, we further tighten the bottleneck and fine-tune the model so that it becomes a VAE (§ 3.2.3). Below we provide details of the training procedure, beginning with the second stage.

3.2.1 Multi-Query Compression

There have been explorations of Transformer-based VAEs in NLP and symbolic music [cite:Jiang], but these methods focus on token-level latent representations. In our case, we need a latent that represents the entire sequence. The question is how to obtain such a sequence-level latent.

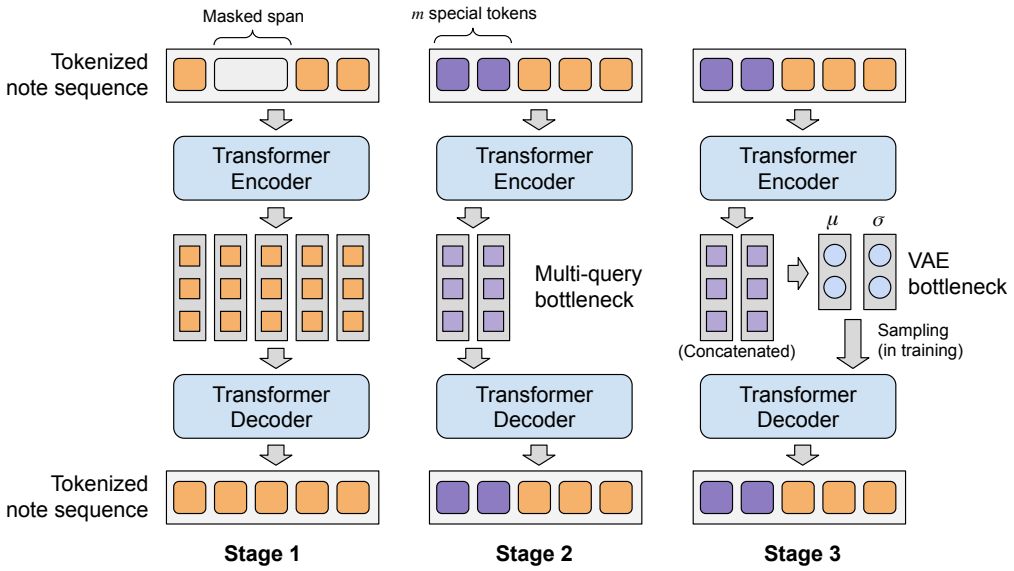


Figure 2: PhraseVAE training stages.

Our autoencoder (AE) is a sequence-to-sequence model implemented with an encoder–decoder Transformer. The input and output of the model are the same sequence, since the model learns to compress and reconstruct it. As in standard encoder–decoder setups, the encoder produces a sequence of hidden states, and the decoder autoregressively reconstructs the sequence during inference. The difference is that the decoder does not attend to the full encoder states. Instead, it receives only a compressed set of latent vectors through a bottleneck.

Previous work in NLP has tried to obtain sequence-level information using methods such as the CLS token in BERT or sequence-level pooling for classification tasks. These methods do capture some high-level semantics, but they are not sufficient for accurate sequence reconstruction. Compressing all information into a single latent vector is simply too difficult for our task, since full reconstruction requires complete detailed information.

Our solution is straightforward: instead of compressing the entire sequence into a single vector, we compress it into m vectors. This gives us a way to balance completeness and compactness. A larger m preserves more information but is less compact, and a smaller m is more compact but loses detail. In our experiments, we set $m = 4$. This is implemented by introducing multiple special queries that attend to the encoder hidden states. Each query performs dot-product attention over the sequence and extracts a different aspect of the information. The decoder then uses these m queried outputs for cross-attention.

3.2.2 Span Infilling Pretraining

There is an important component of music sequence modeling that is not directly related to compression, namely the sequence representation itself, i.e., what a valid music sequence looks like. Since we adopt the REMI-z tokenization, both the input sequence to be compressed and the output sequence to be reconstructed must follow its grammar. For example, each phrase sequence must start with an instrument token, and a duration token must always appear after the corresponding pitch token. Learning this grammar does not require compression training, and we hypothesize that learning to generate sequences that follow this grammar is helpful for later compression, because the model can build a strong contextual representation first.

This idea is implemented as a pretraining stage before the multi-query compression training. We use a denoising autoencoder setup similar to the BART training framework [7], where the objective is span infilling. During training, several spans of random length are masked out from the input sequence and fed into the encoder. The decoder is then asked to autoregressively generate the full sequence while correctly recovering the masked regions. Through this process, the model learns robust and contextual token representations and becomes able to generate grammatically correct

sequences under the REMI-z rules. As noted in the original BART paper, such pretraining is very effective for sequence generation tasks, which is why we adopt it here. This pretraining stage is performed without any bottleneck, which means the decoder can see the entire hidden states sequence produced by the encoder. After the pretraining, the model is later fine-tuned with the multi-query bottleneck for compression.

3.2.3 From AE to VAE

The autoencoder described above is not the final model, because its latent space is not well regularized. If we plot the latent codes using 2D or 3D PCA, the space does not appear symmetric, and the values of different dimensions vary widely. This is not ideal for training the latent diffusion model, which assumes that the data to be modeled follows a standard normal distribution. In other domains, especially image generation, VAE-based compression is commonly used to make the latent space smoother and closer to a normal distribution. This motivates us to convert our autoencoder into a VAE.

The next step is to fine-tune the autoencoder into a VAE. We continue training from the autoencoder checkpoint while adding a tighter bottleneck for more aggressive compression. The implementation follows the standard VAE setup. The four query outputs are concatenated into one long vector (2048-dim) and projected into a 128-dimensional vector. This vector is then split into two halves: one representing the mean and the other representing the standard deviation. During training, the latent is sampled using the reparameterization trick, and the KL divergence loss is used to regularize the latent distribution toward a standard normal distribution. We adopt a loss function similar to the one used in *beta*-VAE, where the total loss is a weighted sum of the reconstruction loss and the KL divergence loss. In our training, we set the weight for the KL term to 0.01. (This does not mean $\beta < 1$. See Discussion for details.) As a result, we obtain 64-dimension well regularized and noise-robust lantent thats represents phrase sequences.

3.3 PhraseLDM

After obtaining PhraseVAE, we use it to compute the latent representations for the entire dataset and train PhraseLDM on these latents. During training, the latent diffusion model still respects the REMI-z grammar at a higher level, i.e. each bar contains multiple phrases ordered by instrument pitch, end with a special end-of-bar token, and then the phrases of the next bar follow. This defines the data structure of the phrase-level latent sequence. Note that this ordering is only a structural arrangement of the latent sequence; the diffusion model itself generates the entire latent sequence for the full song in one shot.

3.3.1 Model Structure

The model structure follows the design of Stable Audio, which is a diffusion Transformer (DiT) operating on a 1D latent sequence. The Transformer network has six layers and uses an encoder-only format. Each Transformer block contains both self-attention and cross-attention. Cross-attention allows the model to attend to external information and is used for conditional generation. For both training and inference, we use a standard DDPM scheduler. During training, the model predicts the noise added at each time step and the loss is computed using mean squared error (MSE). Figure 3 shows the PhraseLDM model structure.

There are several important implementation details for handling variable-length songs. During training, all latent sequences are padded to the maximum supported length, which is 128 bars. This allows the model to generate a fixed-length latent sequence, while different songs may have different actual lengths. To let the model learn where a song should end, we introduce a special end-of-song latent in the VAE training stage, represented by a dedicated token. When training the diffusion model, the latent corresponding to this end-of-song phrase serves as the boundary indicator. If a song is shorter than the maximum length, we pad the remaining positions using this special latent as well. During decoding, the output token sequence from the VAE is truncated at the first occurrence of the end-of-song token. We do not apply attention masks inside the LDM's self attention. Positional encoding is crucial for indicating the location and ordering of latents, and we adopt rotary position embeddings similar to Stable Audio.

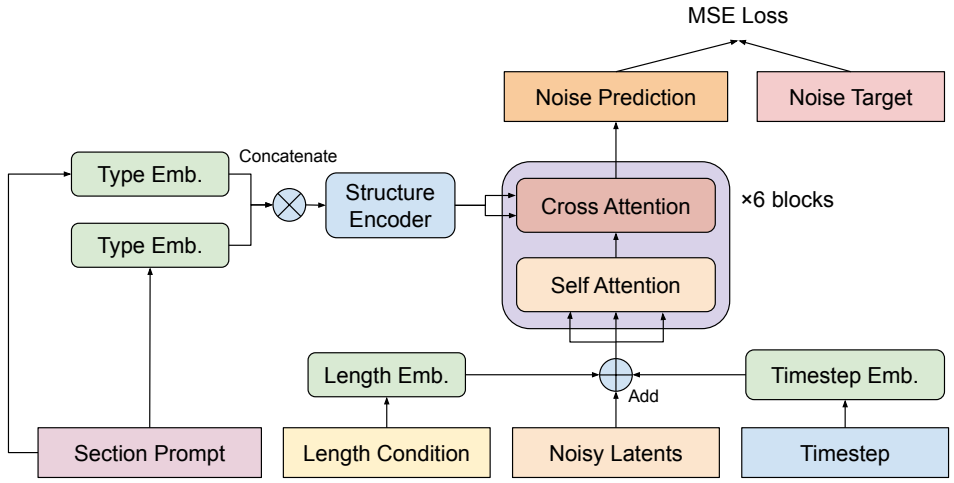


Figure 3: PhraseLDM model structure.

The latent diffusion model receives three types of conditions during training and inference. The first is the diffusion time step, necessary for all diffusion models, which is mapped into an embedding and added to all hidden states. The other two conditions—length condition and structure condition—are optional and will be described in the following subsections.

3.3.2 Length Condition

At the initial stage of training, we found that the unconditional model sometimes struggles to maintain consistent phrases across bars. We hypothesize that this is because the distribution of phrase latents may differ across songs of different lengths. For example, longer songs may contain more repetitions or recurring patterns. Hence let the model to learn length-conditioned distribution might alleviate this issue. Adding a length condition also improves usability, since users may want to generate songs of specific lengths for different use cases.

We provide the length information using the number of bars in the song both for training and inference. This value is mapped to an embedding and added to all latent vectors in the sequence. To avoid overfitting to specific bar counts, we use a length bucket instead of the exact number. The condition specifies a target range of bars, where the bucket size is B, and each range is a multiple of B. In our experiments, we set $B = 10$. For example, a length condition may look like "generate a song that contains bars in the range $[40, 50)$."

3.3.3 Structure Condition

There are two motivations for adding structure conditioning. The first is to make the system more usable, since some users may want the generation to follow a predefined structure. The second is to enable a more clear form structure in the generation. We observe although the length-conditioned model already shows a certain degree of section structure on its own, the section boundaries and recurrence patterns are still not as clear as in real songs. Therefore, we think that providing external section information as a prompt may further improve structure-related generation quality.

In our experiment, we use the "section layout" as the structure prompt. This layout comes from human annotations, which is a list of structure descriptor and each element describe the type and length of one section in the song, in sequential order, such as: [intro - 8 bars, verse - 10 bars, chorus - 8 bars, outro - 4 bars]". The section type and the section length are tokenized and embedded separately, then concatenated, and sent to a small Transformer encoder to make the prompt more contextualized. The resulting embedding is then fed into the cross-attention of the DiT.

4 Experiments

4.1 Dataset

We adopt the POP909 dataset [18] for both training and evaluation. We randomly select 5% of the songs as the validation set and use the remaining pieces for training. POP909 provides a convenient multitrack setting with moderate musical complexity: each song contains three tracks—a melody line (monophonic), a lead instrument line (sometimes polyphonic), and a piano accompaniment (polyphonic)—which makes it well-suited for testing our proposed framework.

One limitation of POP909 is its relatively small size, containing only about 800 songs after removing songs with annotation errors. Nevertheless, we will later show that our approach is capable of achieving high-quality, diverse, and musically coherent full-song generation even under such limited data conditions, with minimal signs of overfitting.

4.2 PhraseVAE

4.2.1 Implementation Details

PhraseVAE is implemented as an encoder–decoder Transformer following the T5 architecture provided by HuggingFace Transformers. Both the encoder and decoder contain 3 layers, with a hidden size of 512, 6 attention heads, and a positionwise feed-forward inner dimension of 1024. The resulting model has approximately 15M parameters, comparable in scale to our baseline [20]. For training, we use a batch size of 128 and a learning rate of 1×10^{-4}. Across all training stages, models are optimized until the validation loss plateaus, with early stopping (patience 20 epochs) and checkpoint selection based on the lowest validation loss.

During both AE and VAE training stages, we prepend four special tokens (`Q1`–`Q4`) to the encoder input. The corresponding encoder outputs of these tokens serve as the compressed latent vectors and are provided to the decoder as its cross-attention inputs.

4.2.2 AE for Phrase Sequence Compression

This part of experiments evaluate models undergone the span infilling pretrain and the AE training stages, before the VAE finetuning.

F1-based Metrics. When evaluating reconstruction quality for both the AE and VAE, our primary concern is fidelity: the reconstructed sequence should match the original phrase as closely as possible. Many similarity metrics could be used, but we choose one that aligns naturally with musical interpretation. We adopt the F1 score as our main metric, where both missing notes and spurious notes are penalized. Concretely, the reconstructed REMI-z track sequence is converted into a bar-level piano roll with the same 48th-note quantization, and F1 scores are computed from this representation.

We report three levels of F1 score with progressively stricter correctness criteria: $\mathbf{F1}_{\mathrm{op}}$, $\mathbf{F1}_{\mathrm{opd}}$, and $\mathbf{F1}_{\mathrm{iopd}}$. A predicted note is counted as correct if (1) onset and pitch match ($\mathrm{F1}_{op}$), (2) onset, pitch, and duration match ($\mathrm{F1}_{opd}$), or (3) instrument, onset, pitch, and duration all match ($\mathrm{F1}_{iopd}$).

Most prior works evaluate reconstruction using only $\mathrm{F1}_{op}$ or $\mathrm{F1}_{opd}$, as these two components directly reflect musical timing and pitch accuracy, which are the dominant factors affecting perceptual similarity. We follow this common practice when comparing with existing methods. Nevertheless, we additionally report $\mathrm{F1}_{iopd}$ because PhraseVAE explicitly models instrument labels within each phrase, and this stricter metric gives a more complete view of how well the latent representation preserves multitrack information.

AE Performance On the phrase compression task, our multi-query compression AE model achieves **99.9%** $\mathrm{F1}_{op}$ and **99.9%** $\mathrm{F1}_{opd}$.

Comparison of Compression Strategies for AE. In a more challenging setting, we further compare different compression strategies to understand their effectiveness. We consider two commonly used baselines: (1) *sequence pooling*, which applies average pooling over all encoder hidden states to obtain a single sequence embedding, and (2) *single-query compression*, which uses one learned query

Table 1: Comparison of different AE compression strategies (bottlenecks).

Model name	$\mathbf{F1}_{\mathrm{op}}$	$\mathbf{F1}_{\mathrm{opd}}$	Best step
Pooling-based	98.5%	97.3%	-
Query-based (m=1)	93.1%	89.1%	-
Multi-query (m=4)	**99.7%**	**99.5%**	**119**
w/o pretrain	99.6%	99.1%	143

Table 2: Comparison of VAE bottleneck size.

Latent dim	F1_op	F1_opd
128	99.9%	99.7%
64	**99.0%**	**98.4%**
32	95.1%	92.6%

token to attend over encoder states and aggregate information into a single vector. We also include an ablation baseline that removes the span-infilling pretraining stage and trains the autoencoder directly. Importantly, this comparison is conducted on the more difficult *bar-compression* task, where the model must reconstruct the full REMI-z bar sequence, including multiple instrument-specific phrase sequences arranged in a specific order. This is significantly harder than phrase compression, making performance differences more observable.

As in Table 1, single-query compression performs the worst, while pooling-based compression yields substantially better reconstruction, consistent with observations in prior work. However, pooling still loses nontrivial information, achieving only 97.3% $\mathrm{F1}_{opd}$, leaving room for improvement. Simply increasing the number of query vectors from one to four—our proposed multi-query compression—significantly improves reconstruction quality and surpasses the pooling baseline, reaching near-perfect performance (99.5% $\mathrm{F1}_{opd}$). Removing the span-infilling pretraining also degrades performance and slows convergence, confirming that learning the REMI-z grammar beforehand meaningfully benefits downstream compression.

4.2.3 VAE with Bottleneck

Latent Size. Recall that in the VAE finetuning, we add a tighter bottleneck on top of the multi-query bottleneck (2048-dim per phrase sequence) to further tighten the latent space. We trained PhraseVAEs with different VAE bottleneck dimensions to produce latents of various sizes, and here we present the comparison in Table 2.

Models with 128-dimensional latents or larger can almost perfectly reconstruct the music. (We will show later that such large latents are also suboptimal due to memorization issues.) A 32-dimensional bottleneck is too restrictive, achieving only 95.1% $\mathbf{F1}_{\mathrm{op}}$. In symbolic music, onset or pitch errors are highly perceptible, often causing reconstructed notes to shift to incorrect timing or incorrect frequencies, which is musically unacceptable. For this reason we did not adopt 32 dimensions. Instead, we choose 64 dimensions, which is compact while still providing high reconstruction fidelity.

Representation Granularity. This comparison shows that phrase-level compression is not only conceptually meaningful for downstream generation, as discussed earlier, but also empirically the most efficient granularity for LDM. It strikes a balance between semantic completeness and latent compactness, making it an effective "sweet spot" for symbolic music representation.

We compare our PhraseVAE against two alternative granularities:

- **BarVAE.** The VAE compresses the entire REMI-z bar sequence into a single latent vector. One latent encodes all instruments and all notes within the bar. This representation is highly expressive because it preserves complete bar-level musical information.
- **PositionVAE.** The VAE compresses each active temporal position inside a bar. For every non-empty 48th-note slot that contains one or more note onsets (across all instruments), the model produces one latent. This representation is more fine-grained than PhraseVAE.

Table 3: Comparison of latent granularity.

Model	Latent size per unit	Avg. #units per bar	Avg. latent dim per bar	F1_opd
BarVAE	512	1	512.0	98.3%
PositionVAE	32	9.54	305.3	98.7%
PhraseVAE	64	2.38	**152.3**	98.4%

Table 4: Comparision between our VAE and PianoTreeVAE.

Model	F1_op	F1_opd	Training Time
PianoTreeVAE	92.2%	88.7%	30 hours
Ours (512-dim BarVAE)	98.9%	98.3%	14 hours

As shown earlier, more compact latents make accurate reconstruction harder. Conversely, by increasing the latent dimensionality sufficiently, one can always obtain a VAE that reconstructs nearly perfectly. However, this is not desirable: large latent vectors directly increase the diffusion model's channel size, and prior work in latent diffusion suggests that high-dimensional latents make LDM training more difficult. Therefore, for each representation granularity, we search for the *smallest* bottleneck size within {16, 32, 64, 128, 256, 512, 1024} that achieves $\mathbf{F1}_{\mathrm{opd}} > 98\%$, and report the resulting information footprint in Table 3.

Although the three models achieve similar reconstruction quality, they differ substantially in efficiency for LDM training. This is because the LDM must generate a different number of latent vectors per bar depending on the granularity:

- **BarVAE:** one latent per bar, resulting in 512 floating-point values.
- **PositionVAE:** roughly 10 non-empty positions per bar, leading to around 300 floating-point values.
- **PhraseVAE:** on average 2.38 phrases per bar, resulting in about 150 floating-point values.

PhraseVAE requires the fewest floating-point values to represent the same bar of music, making it the most efficient design for latent diffusion among the three granularities, which potentially facilitate LDM learning. For this reason, we adopt phrase-level latents for our LDM.

Comparison with Existing Models. We evaluate the reconstruction capability of our model by comparing it against a previous state-of-the-art symbolic music VAE, PianoTreeVAE [20]. This comparison is performed on the bar-level compression task, which is more challenging than phrase-level compression. We retrained PianoTreeVAE on our bar-level dataset after flattening all instruments (i.e., treating all notes as belonging to a single instrument), and adopted the same temporal quantization as our system, namely 48th-note resolution for onset and duration. For a fair comparison, we used a 512-dimensional latent vector, matching the latent size of PianoTreeVAE.

As in Table 4, our model substantially outperforms PianoTreeVAE, achieving roughly a 10% improvement in $\mathbf{F1}_{\mathrm{opd}}$. Additionally, due to the parallel training capability of the Transformer architecture (as opposed to the autoregressive nature of PianoTreeVAE), the total training time of our method—including all three stages—is much shorter than that of PianoTreeVAE.

4.3 PhraseLDM

4.3.1 Implementation Details

PhraseLDM is implemented as an encoder-only Transformer and adopts the implementation from Stable Audio DiT. The model uses a hidden dimension of 512 ($d_{\mathrm{model}} = 512$), 64 input and output channels, 6 Transformer layers, and 16 attention heads, resulting in a model of approximately 30M parameters. The context length is set to 512 latents, supporting up to 128 bars of music for the adopted dataset (each bar contains three phrase latents plus an end-of-bar latent). The cross-attention module operates at 128 dimensions, matching the output size of the structure encoder. Diffusion timesteps are encoded via a 128-dimensional time projection. Length conditioning is provided through a simple

embedding layer that outputs 128-dimensional vectors. The structure encoder is implemented as a 3-layer Transformer encoder. For training, we use a batch size of 128 and a learning rate of 5×10^{-4}. All models are trained for 200k steps.

4.3.2 Metrics for Generation Quality

Since full-song symbolic music generation is a relatively new task, there is no established standard for evaluation. We therefore propose a set of metrics that we believe are reasonable and informative.

PhraseFID. This metric evaluates phrase-level generation quality. We directly compute FID directly in the PhraseVAE latent space between generated songs and training set. It measures (1) how well the model generates phrases that resemble those in the training data, and (2) the phrase-level musical quality, with the assumption that phrases closer to human created phrase have higher quality. A lower PhraseFID indicates that generated phrases follow a distribution closer to real data and are hence better in quality.

4.3.3 Metrics for Section Structures

We adopt a bar-level latent Self-Similarity Matrix (**BarSSM**) to evaluate structural quality. Existing structure-related metrics (e.g., those in [**?**]) apply only to structure-conditioned generation. In contrast, BarSSM measures structure coherence for any model. The method is inspired by the long-standing use of SSMs in the music structural segmentation task. Since repetition is a key characteristic of contemporary music and a direct indicator of section recurrence, SSMs of real songs typically display clear patterns. To compute BarSSM, we first average-pool the phrase latents within each bar to obtain a single "bar latent." We then compute a self-similarity matrix using pairwise cosine distances between all bar latents, and visualize it as a two-dimensional image.

SSMs on symbolic music latents reveal structure almost directly. As illustrated in Figure **??**, genuine songs show distinctive non-major-diagonal lines formed by "bright points," indicating the presence of similar sections that recur in the composition. From such a pattern, one can infer the section layout of a real song—e.g., *intro – A + B – interlude – A + B – bridge – B × 2*— without even listening to the audio.

Thus, we visually inspect SSMs of generated samples to determine whether clear section-level concepts appear. If the SSM lacks meaningful non-major-diagonal structure, the song is likely a stream of unrelated bars with no recurrence. In contrast, a well-trained model produces SSMs that resemble those of real songs.

However, manual inspection is slow. Therefore, we propose a simple but intuitive metric to quantify section recurrence:

SRS (Section Recurrence Score). We count the total lengths of all non-major-diagonal lines in the *upper triangular* region of the SSM, subject to two rules: (1) a diagonal must have length ≥ 4 bars, (2) clustered diagonals (diagonals adjacent to each other) are excluded (these correspond to similar consecutive bars rather than recurring sections). After summing the lengths, we divide by the total number of bars in the song. SRS has a clear physical interpretation: on average, how many times a musical idea reappears in sections of length ≥ 4 bars, beyond its original occurrence. (Do not laugh at its simplicity.)

The SRS depends on the threshold used to define "bright points" in the SSM. With a threshold of 0.5, real POP909 songs typically achieve an SRS of around 0.6 (to be confirmed). A model that generates high-quality local texture but has no notion of section structure will score near 0. Higher SRS values indicate more repetition, which is generally desirable because most models we trained struggle with structural recurrence. Conversely, models with excessively high SRS (even higher than real data) often show degraded PhraseFID, indicating a trade-off between repetition and phrase-level diversity.

4.3.4 Metrics for Memorization

While generated samples that are close to the training set often appear high-quality, it is equally important to ensure sufficient diversity. If a model simply replicates training examples, quality metrics may still look perfect, but the model effectively degenerates into a lookup table. Since the purpose of a generative model is to produce novel outputs, we must explicitly measure and control memorization.

Table 5: PhraseLDM objective metrics. Len. Acc. refers to length accuracy.

Model	PhraseFID	SRS	T2R	MR	Len. Acc.
Real song	4.24	0.444	0.937	5%	-
Unconditioned	3.89	0.161	0.984	0	15%
Length conditioned	3.81	0.165	0.989	0	95%
Length and section conditioned	(7.65)	(0.024)	0.983	0	100%
Length conditioned (128-dim, 170k step)	2.32	0.273	0.466	55%	-
Length conditioned (128-dim, 140k step)	4.44	0.075	0.872	20%	-

We follow the idea of melody-based similarity evaluation from [19]. A higher melody similarity between a generated song and the training set indicates stronger memorization. But unlike [19], which computes segment-level similarity, we evaluate similarity at the full-song level and therefore introduce new metrics.

We extract the melody track from each song using REMI-z grammar and obtain two melody strings, $\mathrm{mel\_str}_A$ and $\mathrm{mel\_str}_B$. Melody similarity is defined as:

$$\mathrm{Sim}_{\mathrm{mel}} = 1 - \mathrm{WER}(\mathrm{mel\_str}_A, \mathrm{mel\_str}_B).$$

MMR (Melody Memorization Rate). For each generated song, we compute its melody similarity with all songs in the training set and take the maximum similarity, which indicates how similar generated melodies are to their closest training melodies.

T2R (Top-2 Distance Ratio). To determine whether a generated sample is "memorized," we examine the distances between the generated melody and its top two closest training melodies. Let SIM_1 and SIM_2 denote the most similar and second-most similar training melodies, and let their WER distances be d_1 and d_2. We compute:

$$\mathrm{T2R} = \frac{d_1}{d_2}.$$

Following [3], we classify a sample as memorized if $\mathrm{T2R} < \frac{1}{3}$. The intuition is that if a generated song is extremely close to one specific training sample yet noticeably different from all others, the model has likely reproduced that example.

MR (Memorization Ratio). For a batch of generated samples, we compute the proportion of songs classified as memorized under the above rule.

4.3.5 Main Observations

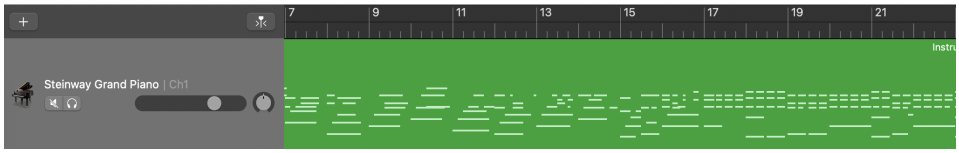
We randomly select 100 songs from the training set as the evaluation set. Each model generates 20 songs, and we compute all metrics between these two groups. For the "real song" baseline, we randomly select another 20 songs from the training set that are not part of the evaluation set.

LDM successfully learns the phrase-level distribution. As in Table 5, both the unconditioned and length-conditioned versions of our 64-dimensional model achieve lower PhraseFID scores than the "real song" baseline. This indicates that the model captures the phrase-level latent distribution effectively.

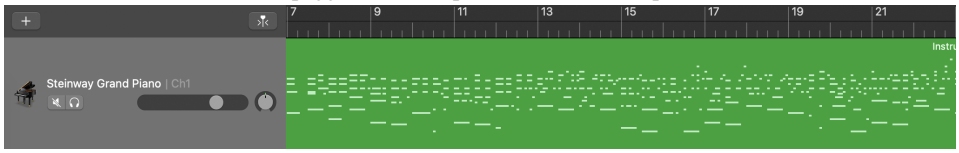
Since all structure-conditioned samples are generated using the same structure prompt, their PhraseFID and SRS naturally differ from those of real data. Therefore, we do not compare their scores directly against other models.

Models without structure conditioning still learn some degree of structure planning. Both the unconditional model and the length-only model achieve an SRS of around 16%, indicating that they have learned to reuse musical material across sections. From the generated examples on our demo site, one can observe recurring themes appearing in different parts of the song. But they still have some distance to the real song's SRS level.

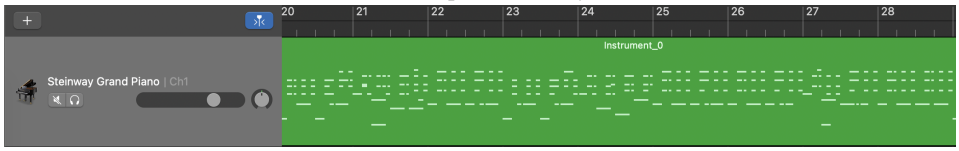
64-dim LDM generation exhibits high diversity. Up to 200k training steps, none of the generated samples are classified as memorized, yielding a 0% memorization rate. Combined with consistently high Top-2 Ratios, this shows that generated samples are distinct from all training songs and do not



(a) Transition from arpeggiated accompaniment to motoric pattern across sections.

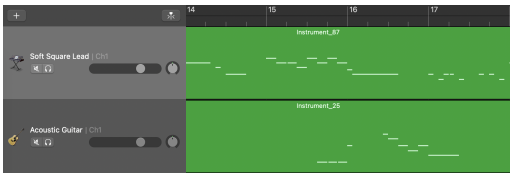


(b) A texture with sophisticated rhythmic detail.

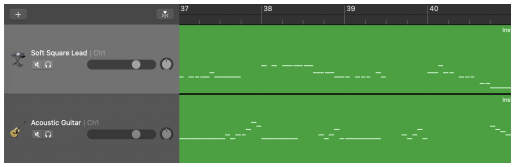


(c) A bossa nova accompaniment pattern.

Figure 4: Case studies of generated piano textures. The examples demonstrate idiomatic phrasing, structural coherence, stylistic diversity, and practical playability.



(a) Lead instrument filling the spaces between melody phrases.



(b) Lead instrument playing a countermelody that forms a rich composite texture with the melody.

Figure 5: Examples of interactions between the lead instrument (the guitar track, 2nd row) and the melody track (synth lead, 1st row) in generated music.

overly resemble any specific one. Interestingly, this diversity is even higher than that of real data under our metric, where one real song is flagged as "memorized."

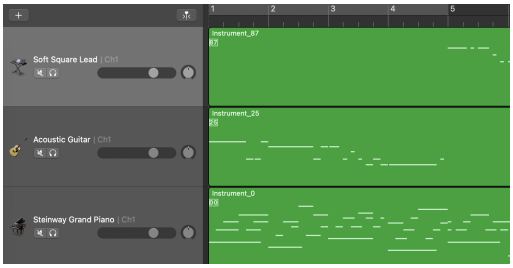
128-dim latents introduce notable memorization issues. We also trained an LDM using a 128-dimensional PhraseVAE to compare with the 64-dim model. While the larger latent dimension reduces PhraseFID and increases SRS, this comes at the expense of memorization. By 140k steps, 20% of generated samples are nearly identical to a specific training song. The memorization rate continues to rise during training and reaches 50% by 170k steps.

Length conditioning is effective, and further strengthened by structure conditioning. Length accuracy is computed as follows: for length-conditioned models, we set the target to 64 bars (a near-median length in POP909) and consider outputs correct if their length falls within the soft constraint interval $[60, 70)$. Without length conditioning, only 15% of generated songs happen to fall within this range. With length conditioning, accuracy increases dramatically to 95%. Adding structure conditioning further improves this to 100%.

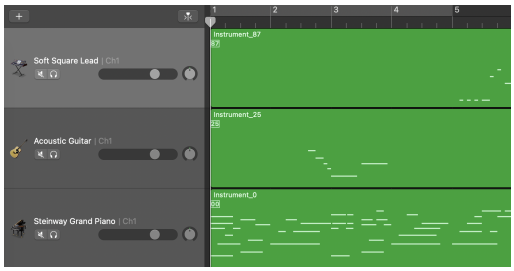
4.3.6 Qualitative Results

Due to time limitations, we did not conduct a systematic human evaluation. Instead, we report several qualitative observations based on representative, non–cherry-picked generated samples.

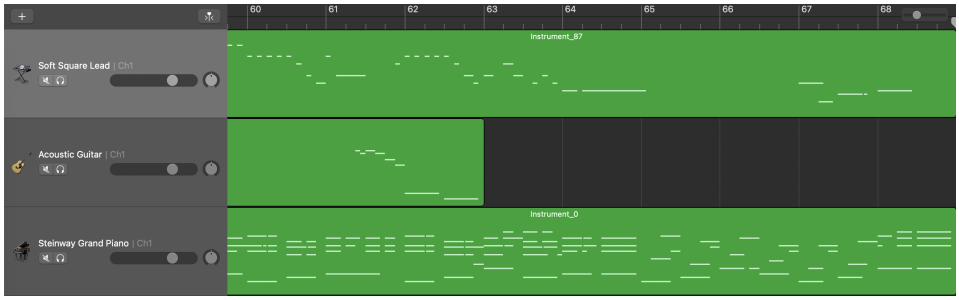
High Local Quality. Segments generated by the length-conditioned model exhibit strong phrase-level musicality. The piano accompaniment patterns are idiomatic and appear perfectly playable. Chord progressions follow common POP909 conventions. Melodies sound natural and blend well with the



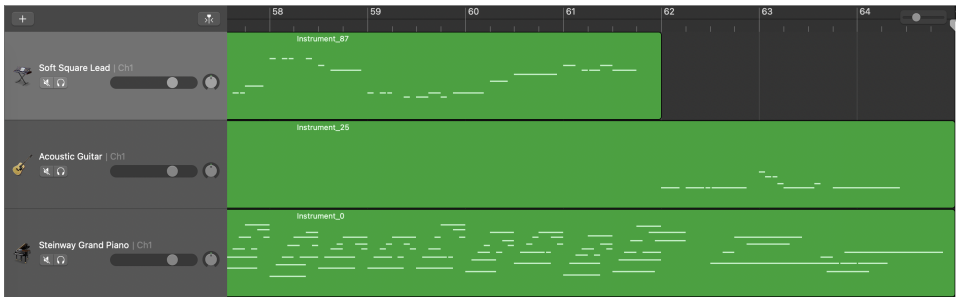
(a) Intro segment with sparse texture and no melody content, reflecting a typical lead-in arrangement.



(b) Another example of an intro region where the model avoids melody content in the intro part.



(c) Outro segment demonstrating reduced texture density and simplified accompaniment, creating a natural sense of closure.



(d) Another outro example where the rhythmic activity diminishes toward the end, reinforcing the ending gesture.

Figure 6: Examples of intro and outro regions generated by PHRASELDM. Intro segments consistently avoid premature melodic content, while outro segments naturally taper in texture and density, producing a convincing ending.

accompaniment, and the lead instrument track often demonstrates musically meaningful behavior, such as countermelodies or interplays with the main melody.

Unconditional Model Learns Variable-Length Generation. Surprisingly, the unconditional model is capable of producing full songs of widely varying lengths. For example, three randomly selected samples contain 51, 89, and 103 bars. This suggests that the model has learned the distribution of song-level latent lengths without explicit supervision.

Structure-Unconditioned Models Often Produce Coherent Song Structures. Even without structure conditioning, the model frequently generates musically meaningful large-scale structures. For instance, many samples contain clear intros and outros with characteristic sparse textures or cadential gestures. Between melodic sections, the model often inserts lead-instrument fills that function as transitions. Although chorus/verse boundaries are not always as distinct as in real songs, we still observe repeated sections that indicate implicit structure planning.

Section conditioning is partially effective. When conditioned on the same length and section specification, the model consistently produces songs with similar structural layouts. As shown in

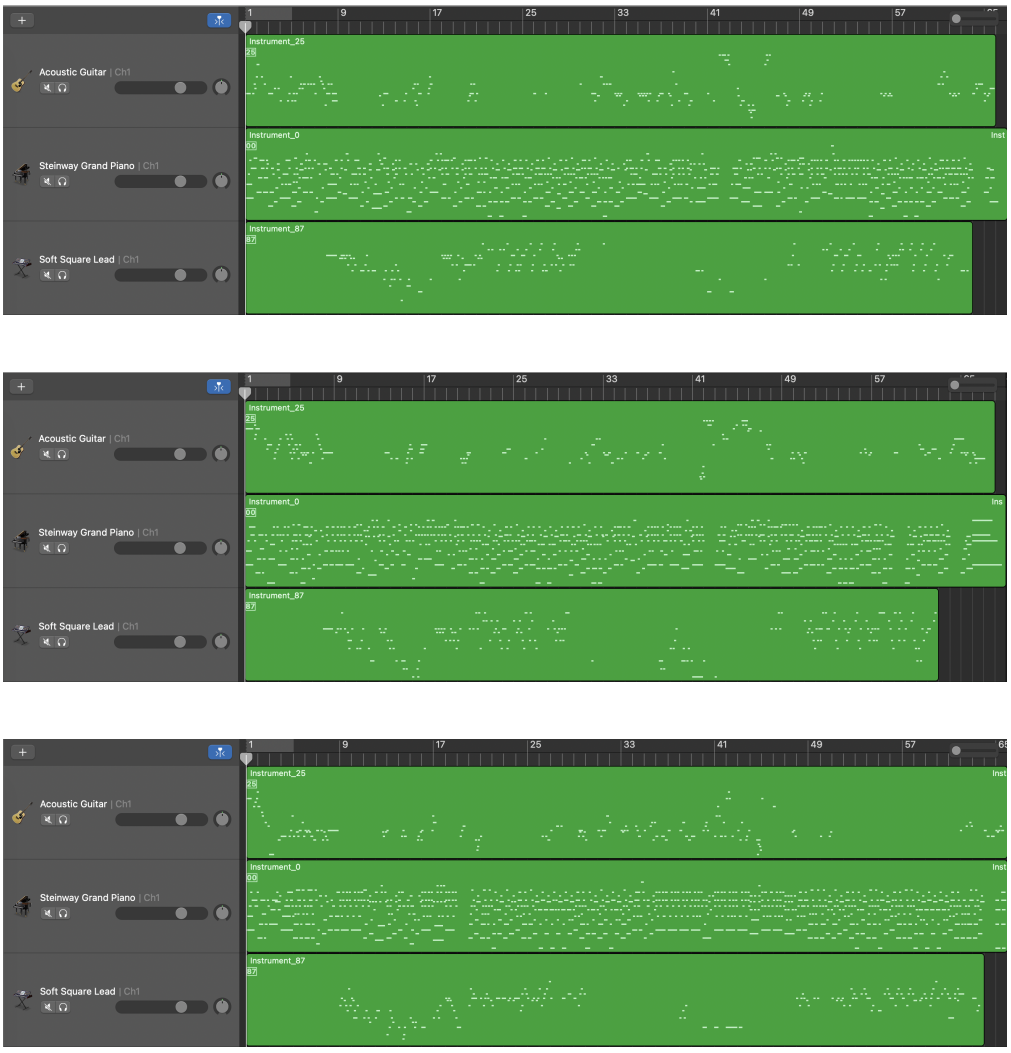


Figure 7: Three randomly selected generations from the section-conditioned model under the same length and section conditions. The length condition is [60,70) bars. The model is conditioned on the following structure sequence: [i-8, A-8, A-8, B-4, B-4, x-4, A-8, B-4, B-4, B-4, B-4, X-4].

Figure 7, in the three randomly selected examples, the distributions of instrument activity exhibit clear alignment among different generated songs, indicating that the conditioning signal is indeed utilized. However, notable limitations remain. In particular, when the condition requires the second occurrence of the verse section (A-8 after x-4), all outputs fail to generate coherent melodic content. This suggests that while the model learns coarse structural placement, it does not fully capture the functional meaning of each section token. Further refinement of the section representation and conditioning mechanism is left for future work.

Overall, qualitative inspection aligns well with our quantitative metrics and indicates that PhraseLDM produces musically coherent, diverse, and structurally meaningful full-song generations.

5 Discussion

5.1 Beta Value for PhraseVAE

In PhraseVAE, our use of a KL weight of 0.01 should not be interpreted as weak regularization. Because reconstruction loss is computed at the *token* level while KL divergence is applied once per

sequence, setting the KL weight to 1 would yield an excessively large effective β-value and prevent the VAE from converging. This mismatch arises because a single KL term must implicitly balance against hundreds of token-level reconstruction terms; in other words, the sequence-level KL must effectively be "distributed" across all tokens to match their scale.

Despite the seemingly small weight, the resulting latent space is in fact strongly regularized: the empirical standard deviation is approximately 0.76, which is characteristic of a β-VAE with an effective $\beta > 1$. The exact effective β is not explicitly defined, as the compressed sequences vary in length and therefore in the number of reconstruction terms.

5.2 Validation Loss

Validation loss is a meaningful signal at the segment level (phrase or bar) and is used throughout all stages of PhraseVAE training.

However, validation loss at the *song level* is far less informative. Melodies, harmonic progressions, and section layouts differ substantially across songs, so the distribution of phrase-level latents varies widely between the songs in the training set and that in the validaion set. During PhraseLDM training, we consistently observe that song-level validation loss increases steadily, yet this bears no correlation with generation quality. We therefore recommend using song-level validation loss sparingly.

5.3 Latent Dimension

Initially, we attempted to train the LDM using an embedding dimension equal to the VAE latent dimension. Under this configuration, the model was unable to fit even a single-song toy dataset. Increasing the `d_model` size significantly alleviated this issue, whereas increasing depth alone did not help. For future work, we recommend setting the LDM model dimension to at least twice the latent dimensionality to ensure stable training.

5.4 Other Evaluation Metrics

Several other evaluation metrics exist but were not adopted here.

FMD. Proposed in prior work using the CLAMP-2 model as the feature extractor for FID-style evaluation. Since CLAMP-2 has a 32k-token context limit—insufficient for full multitrack songs—FMD functions more as a segment-level metric. Given that our study focuses on local and full-song evaluation, we elected not to use it. Future work may incorporate FMD for broader comparison.

Latent-Based Structure Metrics from [?]. This metric evaluates intra-segment coherence and inter-segment distinction, useful for analyzing controllability under structure-conditioning. Because structure conditioning is not the central focus of this work, we did not include it, but it remains a strong candidate for future studies.

5.5 Scalability of Structure Condition

The current structural conditioning scheme is not scalable: it requires manually annotated section types and lengths for every song. While feasible for a dataset of roughly 1,000 songs, it is entirely impractical at the scale of millions or billions. Further, the concept of "section" itself—e.g., verse, chorus—varies widely across genres such as jazz, EDM, or ambient music. A rigid, universal taxonomy would introduce substantial noise.

To address these limitations, we outline two directions for future research.

1. Extracting Section Layouts Directly from the SSM. Self-Similarity Matrices can be computed automatically from MIDI or latent sequences and inherently encode sectional recurrence. Using SSM-based segmentation could bypass the need for human annotation.

2. Replacing Section Labels with Automatically Extractable Musical Features. We observe a strong correlation between instrumentation layout and sectional boundaries in POP909. For example, bar-to-bar instrumentation changes occur at section boundaries with probability 59.2%, compared to 33.8% within the same section. When considering only instrument combinations (ignoring voice order), the difference becomes even more pronounced (49.2% vs. 22.9%). This suggests that instrumentation implicitly encodes section-level structure.

We attempted to use instrumentation layout as a replacement structural condition. However, this produces very long conditioning sequences (e.g., four instrument tokens per bar; 400 tokens for a 100-bar song). On a small dataset of ~ 800 songs, this proved too difficult for the model to learn effectively. With substantially larger datasets, this direction may become viable.

6 Summary

This report presents a complete phrase-level latent diffusion framework for full-song symbolic music generation. By departing from the conventional note-attribute representation and instead modeling music at the phrase level, we transform the full-song generation problem from an extremely long-sequence modeling task into a compact latent modeling problem. PhraseVAE provides high-fidelity, well regularized phrase representations that preserve musical semantics while greatly reducing data dimensionality. PhraseLDM then operates entirely in this latent space, enabling efficient, non-autoregressive full-song generation with coherent local texture, idiomatic multitrack interaction, and meaningful global structure.

Through extensive analysis, we demonstrate that phrase-level latents strike a favorable balance between expressiveness, compactness, and trainability. The latent space supports diffusion modeling at a scale (128 bars per sequence) previously impractical for symbolic music. We further show that length conditioning and structure conditioning can reliably guide global form, while careful bottleneck design is essential for preventing memorization. Our evaluation also highlights the importance of new metrics—such as PhraseFID, SSM-based structural measures, and melody-level memorization tests—for assessing full-song symbolic generation models.

Overall, our results suggest that phrase-level latent diffusion is a promising and scalable paradigm for symbolic music generation. It removes the core bottlenecks associated with note-level modeling, enables fast and lightweight full-song generation, and produces musically convincing outputs even on modest datasets. We hope this work encourages future research to develop richer phrase representations, more scalable structural conditioning, and larger training corpora, ultimately advancing symbolic music generation toward more expressive, reliable, and globally coherent long-form composition.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[2] Nathan Fradet, Nicolas Gutowski, Fabien Chhel, and Jean-Pierre Briot. Byte pair encoding for symbolic music. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2020, Singapore, December 2023. Association for Computational Linguistics.

[3] Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

[4] 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 *AAAI Conference on Artificial Intelligence*, 2021.

[5] 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*, pages 1180–1188, 2020.

[6] David Brian Huron. *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.

[7] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[8] Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, and Jiang Bian. Musecoco: Generating symbolic music from text. *arXiv preprint arXiv:2306.00110*, 2023.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] Lejun Min, Junyan Jiang, Gus Xia, and Jingwei Zhao. Polyffusion: A diffusion model for polyphonic score generation with internal and external controls. *arXiv preprint arXiv:2307.10304*, 2023.

[11] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021.

[12] 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[13] Philippe Pasquier, Jeff Ens, Nathan Fradet, Paul Triana, Davide Rizzotti, Jean-Baptiste Rolland, and Maryam Safi. Midi-gpt: A controllable generative model for computer-assisted multitrack music composition. *arXiv preprint arXiv:2501.17011*, 2025.

[14] Xingwei Qu, Yuelin Bai, Yinghao Ma, Ziya Zhou, Ka Man Lo, Jiaheng Liu, Ruibin Yuan, Lejun Min, Xueling Liu, Tianyu Zhang, et al. Mupt: A generative symbolic music pretrained transformer. *arXiv preprint arXiv:2404.06393*, 2024.

[15] David Temperley. *The cognition of basic musical structures*. MIT press, 2004.

[16] Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann. Figaro: Controllable music generation using learned and expert features. In *The Eleventh International Conference on Learning Representations*, 2023.

[17] Yashan Wang, Shangda Wu, Jianhuai Hu, Xingjian Du, Yueqi Peng, Yongxin Huang, Shuai Fan, Xiaobing Li, Feng Yu, and Maosong Sun. Notagen: Advancing musicality in symbolic music generation with large language model training paradigms. *arXiv preprint arXiv:2502.18008*, 2025.

[18] Ziyu Wang, Ke Chen, Junyan Jiang, Yiyi Zhang, Maoran Xu, Shuqi Dai, Xianbin Gu, and Gus Xia. Pop909: A pop-song dataset for music arrangement generation. *arXiv preprint arXiv:2008.07142*, 2020.

[19] Ziyu Wang, Lejun Min, and Gus Xia. Whole-song hierarchical generation of symbolic music using cascaded diffusion models. *arXiv preprint arXiv:2405.09901*, 2024.

[20] Ziyu Wang, Yiyi Zhang, Yixiao Zhang, Junyan Jiang, Ruihan Yang, Junbo Zhao, and Gus Xia. Pianotree vae: Structured representation learning for polyphonic music. *arXiv preprint arXiv:2008.07118*, 2020.

[21] Shangda Wu, Yashan Wang, Xiaobing Li, Feng Yu, and Maosong Sun. Melodyt5: A unified score-to-score transformer for symbolic music processing. *arXiv preprint arXiv:2407.02277*, 2024.

[22] Yi-Hsuan Yang et al. Meteor: Melody-aware texture-controllable symbolic orchestral music generation via transformer vae. In *Thirty-Fourth International Joint Conference on Artificial Intelligence AI, Arts & Creativity*, 2025.