

# UNIComposer: BAND-LEVEL MUSIC COMPOSITION WITH SYMBOLIC AND AUDIO UNIFICATION

Anonymous authors

Paper under double-blind review

## ABSTRACT

Multi-track deep music generation has largely focused on pre-specified structures and instruments. However, it remains a challenge to generate “band-level” full-length music that is capable of allocating instruments based on musical features, their expressive potential, and their performance characteristics differences. Moreover, the representations of symbolic and audio music have been treated as distinct sub-areas, without a unified architecture to join their own advantages. In this work, we introduce **UniComposer**<sup>1</sup>, a novel music generation pipeline that composes at the band level, utilizing a hierarchical multi-track music representation complemented by four cascaded diffusion models which progressively generate rhythm features, and unified features extracted from both symbolic and audio music by autoencoders. Experiments and analysis demonstrate that UniComposer achieves a unified latent space for symbolic and audio music, and is capable of generating band-level compositions with well structured multi-track arrangements, surpassing previous methods in performances.

## 1 INTRODUCTION

The area of deep generative models for symbolic music (Wang et al. (2024), Hsiao et al. (2021), Huang & Yang (2020), among others) have witnessed advancements in a range of fronts. However, the approaches in the literature do not yet achieve band-level music generation. Band-level music has two critical features: 1) Instruments in a band collaborate to fulfill specific roles, such as harmony, rhythm, or accompaniment, which can differ from their roles in solo performances—a nuance overlooked in current multi-track generation. For instance, the piano often provides repetitive harmonies in a band, contrasting with its more varied melodies in solo settings. 2) Composers carefully select instruments to match the melody’s characteristics, ensuring timbre, range, and expressiveness align with the intended emotion. They also arrange instruments dynamically, adjusting prominence based on sections or rhythmic changes. Current methods either replicate input instruments (Liu et al. (2022), Dong et al. (2018)) or rely on pre-defined user input (Dong et al. (2023), von Rütte et al. (2023), OpenAI (2024)).

For music data that take Audio formats such as MP3 (mpgedit (2003)) and WAV (bitsforbyte (2021)), there are billions of publicly accessible music tracks spanning a wide range of styles, instruments, and time periods. However, the situation is different for symbolic music. On one hand, symbolic-format music data are not as ample as the audio-format. On the other hand, methods for analysis and generation in symbolic music are often more structured and fine-grained, allowing explicit incorporation of music theory into the model architecture (Wang et al. (2024), von Rütte et al. (2023)). This enhances the model’s ability to implicitly capture complex musical relationships. In contrast, methodologies for analyzing and generating audio music tend to rely solely on the model’s inherent capacity to learn these relationships (Ji et al. (2023), Huang et al. (2023)), without explicitly incorporating music structure into the architecture design. However, no public work has yet developed approaches to organically combine the two forms of music, leaving this area largely unexplored.

Our work, UniComposer, simplifies band-level music modeling by introducing a hierarchical structure that groups instruments into monophonic, polyphonic, and percussion categories. Monophonic instruments provide chordal foundations, polyphonic instruments add melodic complexity, and percussion drives the rhythm. We use a two-step generation process: first, a basic rhythm is created

<sup>1</sup>Demo page is on: <https://sites.google.com/view/unicomposer>

for each type, then details and variations are added. This method is powered by four cascaded Transformer-based diffusion models, operating at the bar level unified feature space for symbolic music and corresponding time for single bar for audio music.

UniComposer combines the strengths of symbolic and audio music formats through two key methods. First, it converts audio into symbolic format, utilizing the structured nature of symbolic music for generation, grounded in music theory. Second, for symbolic music generation, it uses a bar decoder to extract notes from audio data, enhancing the model’s performance through data augmentation. This is achieved via encoders respectively for symbolic and audio inputs, sharing a unified bar decoder. Both inputs are mapped to a common latent space, seamlessly integrating audio and symbolic music generation. Figure 1 illustrates the architecture.

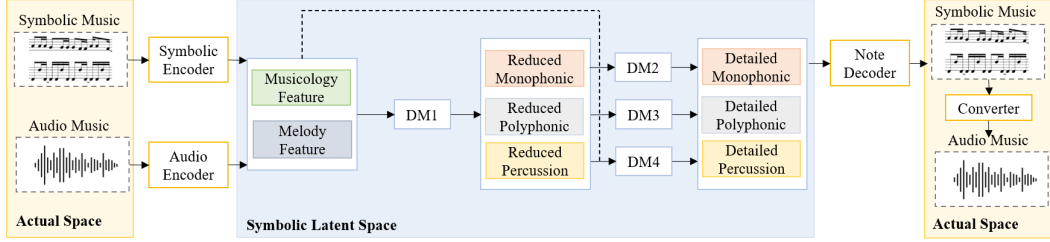


Figure 1: Overall Architecture of UniComposer.

In summary, the contribution of the paper is as follows:

1. We introduce UniComposer, the first band-level music generation system that involves collaborative roles of instruments, tailored to harmonize, provide rhythm, or accompaniment, with careful selection of instruments to match the melody’s expressive qualities. UniComposer uses a hierarchical representation across three instrument types.
2. We propose an approach that integrates the advantages of both symbolic and audio music, using separate encoders and a shared decoder to bridge both formats within a common feature space. This approach can leverage the rich audio music to facilitate data augmentation for symbolic music.
3. UniComposer is capable of mapping inputs into a unified feature space for generation through cascaded bar-based diffusion models. The composer can accept both symbolic and audio music as input, producing well-structured band-level music.

## 2 RELATED WORK

In this section, we first review multi-track music generation, followed by integrated processing for symbolic and audio music. Then, we review structured music modeling. Finally, we review the applications of autoencoders and diffusion models in music generation.

**Multi-Track Music Generation.** Addressing the complexity of generating music consisting of multiple interrelated tracks, multi-track music generation has become a significant area of research. Some works focus on developing multi-track representations of music which facilitate the generation process, while others aim to utilize multi-track approaches to improve music resolution and achieve fine-grained control. Guo (2024) integrates the AC algorithm to enhance diversity in multi-track music generation. MMBert (Zhu et al. (2024)) is a multi-track generation method focusing on the chord aspects of music. Jen-1 Composer (Yao et al. (2023)) generates music that contains four tracks from text prompts. Multitrack Music LDM (Karchkhadze et al. (2024)) leverages multi-track techniques to generate high-fidelity audio music. Cosenza et al. (2023) utilizes a graph-based generation approach to produce polyphonic music, with the instruments to be played specified by the user. MMMachine (Ens & Pasquier (2020)) mainly concentrates on multi-track inputs for music representation to achieve a better understanding of music, similar to SymphonyNet (Liu et al. (2022)) that copies input tracks for generation.

**Integrated Processing of Symbolic and Audio Music.** Research that combines audio and symbolic music is currently divided into two interconnected areas: audio rendering of symbolic music and automatic music transcription that translates audio into symbolic notes. For audio rendering of

symbolic music, traditional audio renderers use DSP-based or sample-based synthesizers to produce instrumental sounds. A parallel line of research has applied data-driven approaches to audio synthesis, often referred to as "neural audio synthesis." Notable examples include NSynth (Engel et al. (2017)), GANSynth (Engel et al. (2019)), DDSP (Engel et al. (2020)), and MIDI-DDSP (Wu et al. (2021)). These approaches share a similar concept of converting notes to audio waveforms using neural networks that perform an upsampling process.

For automatic music transcription, the outputs are typically frame-level multi-pitch estimation (MPE) or note-level estimation. These methods commonly first estimate a pitch posteriorgram, where each time-frequency bin is assigned an estimate of the likelihood of a fundamental frequency being active at a given time (Duan et al. (2010); Bittner et al. (2017)). Multiple methods have been proposed for estimating notes from pitch posteriorgrams, such as using median filtering (Klapuri & Davy (2007)), Hidden Markov Models (Benetos (2017)), or neural networks (Ewert & Sandler (2017); Nishikimi et al. (2021)). Gardner et al. (2021) is the first lighting work of leveraging a unified framework for transcription.

**Structured Music Modeling.** The local musical structure can be effectively modeled in two distinct ways: through approaches that either learn structure indirectly or through methods that directly define or extract musical elements, both yielding good results. Approaches like Music Transformer (Huang et al. (2018)), JukeBox (Dhariwal et al. (2020)), MuseFormer (Yu et al. (2022)), MERT (Li et al. (2023)), LLark (Gardner et al. (2023)) and Lemerrier et al. (2024) represent the former, where models predict and generate musical events by capturing dependencies within the music. On the other hand, methods that rely on musical domain knowledge define specific features or extract interpretable musical representations, enabling the model to learn structures such as pitch contours at the measure level and accompaniment patterns. Some studies also aim to combine them, yielding high quality output (Mariani et al. (2023), Wang et al. (2024)).

**Autoencoders and Diffusion Models for Music Generation.** (Variational) Autoencoders are widely utilized to extract musical representations from pre-trained models, encapsulating both theoretical and perceptual aspects of music (Brunner et al. (2018); Jiang et al. (2020); Wu & Yang (2023)). These extracted features serve as essential tools for modeling cascading musical events due to their rich informational content. In the domain of music generation using diffusion models, two principal approaches have emerged. The first approach represents music in a piano-roll format, enabling diffusion models to directly generate note pitches and durations without the need for intermediate representations (Mittal et al. (2021), Atassi (2023), Wang et al. (2024)). The second approach employs diffusion models as feature learners; these models generate music by manipulating features encoded from preceding stages and executing reverse diffusion processes to synthesize the final output (Zhu et al. (2022), Zhang et al. (2023), Huang et al. (2024)).

### 3 METHODOLOGY

Given an input melody, our proposed UniComposer extracts features from each bar in symbolic music or the relative duration in audio. The composition process operates within the latent space of these features, categorized into three main functions: monophonic, polyphonic, and percussion. Feature extraction is managed by symbolic and audio encoders, while a shared decoder reconstructs bars from these extracted features. Using bars as the fundamental unit, the system progressively generates detailed features for each category through four cascaded diffusion models. A converter can subsequently translate symbolic output into audio. Section 3.1 outlines the hierarchical representation of band-level music, followed by the feature extraction together with the training strategy in Section 3.2, 3.3 and 3.4. Finally, the overall generation process is discussed in Section 3.5.

#### 3.1 HIERARCHICAL REPRESENTATION OF BAND-LEVEL MUSIC

To streamline and simplify the modeling of band-level music, we adopt a functional separation approach to represent the three components of the musical ensemble, with a hierarchical reduction process transitioning from detailed to simplified rhythmic structures inspired by Prajwal et al. (2024) and Wang et al. (2024) to capture the overarching compositional framework. This approach enhances the efficiency of the learning process for UniComposer, and is illustrated in Figure 2.

**Functional Separation.** Monophonic instruments play one note at a time, while polyphonic instruments can produce multiple notes simultaneously, enabling harmonic richness. Percussion instruments create sound by actions like striking or shaking. Monophonic instruments focus on melody, contributing to musical expression by adding emotional depth through greater variation, while polyphonic ones often emphasize harmony and accompaniment with repeated or transposed notes, supporting the music’s structure. Percussion maintains rhythm, articulating the beats and tension that drive the progression of a piece. Recognizing these functional distinctions, UniComposer separates these musical elements to improve data representation for band-level music. This division is facilitated by referencing the program IDs on the MIDI standard (MIDI-Association (2024)).

**Hierarchical Reduction.** On one hand, as is typical for composers, the initial step often involves drafting each section of the ensemble, defining key structural elements such as the primary modulation points and the bassline pitches. On the other hand, the process of function separation can result in superimposition of different instrumental parts, leading to redundant and overly intricate note sequences, which makes it challenging for UniComposer to learn. Inspired by Wang et al. (2024), we adopt a similar reduction approach based on the detailed rhythmic data for each instrumental category. The rhythmic structure of both monophonic and polyphonic sections is largely influenced by the harmonic chroma, while the rhythm of percussive instruments is primarily determined by the time signature, as it governs the temporal shifts within a composition.

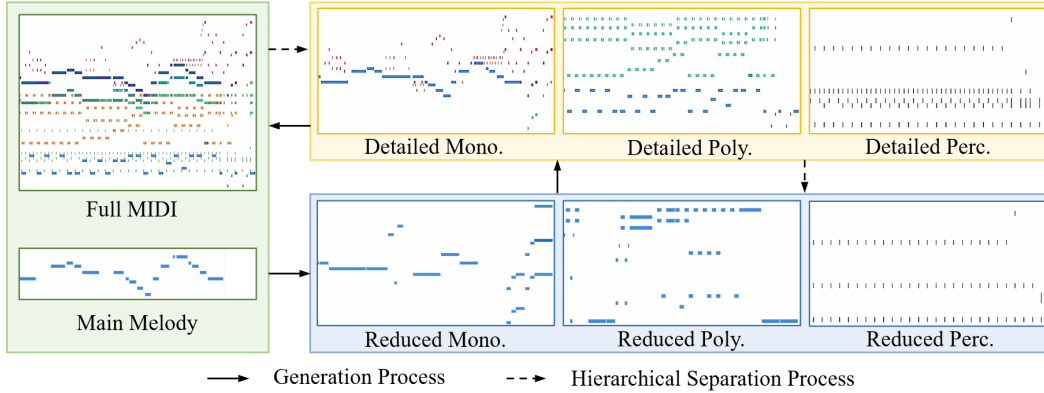


Figure 2: Band-Level Hierarchical Music Representation. Green box shows music with pre-labeled main melody. Mono., Poly. and Perc. represent monophonic, polyphonic and percussion, respectively. Hierarchical separation process breaks down band-level music into three parts, from detailed to reduced. Generation process starts with the main melody and adds reduced and detailed components based on instrument separation.

### 3.2 BAR-LEVEL FEATURE EXTRACTION FOR SYMBOLIC MUSIC

To analyze a sequence of notes within a bar, UniComposer extracts two key features: one representing the notes themselves (note feature) and the other capturing the overall musical characteristics of the bar (musical feature). This process is facilitated by the joint training of a symbolic encoder and a bar decoder, inspired by the approach outlined in Roberts et al. (2018), but with a more lightweight model that utilizes Transformer architecture (Vaswani (2017)) instead of RNNs.

**Data Representation.** UniComposer employs a modified version of the REMI representation (Huang & Yang (2020)), specifically adapted to handle bar-level features in symbolic music. Four key attributes are extracted for each bar: chord, time signature, dynamics, and tonality. Chords are represented by their root, chroma, and bass, while time signatures are restricted to three discrete values: 3/4, 6/8, and 4/4. Dynamics are categorized into six levels, ranging from pianissimo (pp) to fortissimo (ff). Tonality, indicating the organization of chords, can be either major or minor. Each note is characterized by five attributes: onset, duration, pitch, velocity, and instrument. To achieve a fine-grained temporal resolution, each bar is divided into 24 time bins. Velocity is simplified to four levels (von Rütte et al. (2023)), preserving essential nuances while reducing complexity. Pitch is encoded using 128 discrete values (MIDI-Association (2024)), and We group the 128 MIDI instruments, along with drums, into 13 categories, such as classifying all types of pianos (e.g., acoustic

piano, electric piano) under the piano category. UniComposer compiles all the notes present in its dataset and uses them (which is approximately  $10^6$ ) as its vocabulary.

**Addressing Embedding Challenge.** Embedding layers trained alongside the model (e.g., Liu et al. (2022), von Rütte et al. (2023)) face instability due to the excessively large vocabulary size of UniComposer. Additionally, for this five-attribute note representation, the meaning of each note is inherently determined by its own attributes, contrasting with the word embedding approach (e.g., Mikolov (2013)), where meaning is derived from contextual co-occurrence. To tackle these challenges, UniComposer proposes an innovative solution: a simple multi-layer perceptron (E-MLP) is trained to encode each note by applying basic addition and concatenation operations to the traditional embedding vectors of the five attributes of a note. This process is followed by feed-forward layers, which produce an output embedding vector with a fixed dimensionality of 16.

**Symbolic Encoder.** The symbolic encoder employs the E-MLP mentioned above to generate a sequence of embedding vectors for the notes within a bar. Two special learnable vectors, representing the start  $[BOS]$  and end  $[EOS]$  of the sequence, are added to the embedding sequence. This sequence is then processed by a bi-directional model based on the classical multi-head transformer architecture proposed by Vaswani (2017), consisting of 4 layers and 4 attention heads is used to handle this embedding sequence. The transformer’s final hidden state is passed through two separate linear projection layers, producing outputs for both note and musical feature representations.

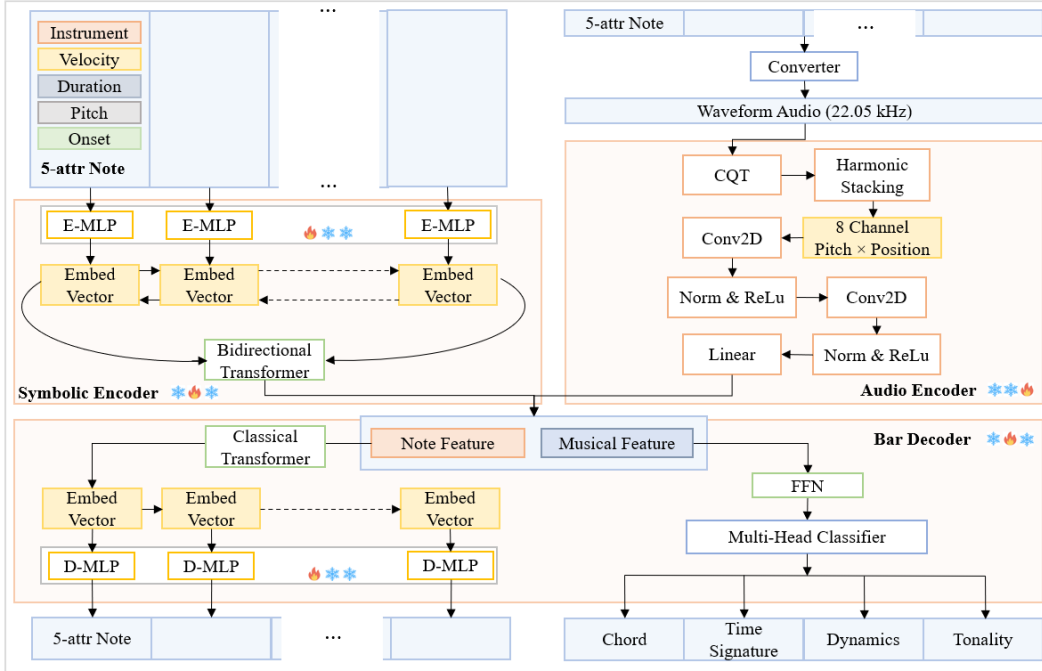


Figure 3: Overall Architecture for feature extraction by the encoders. The symbolic encoder and audio encoder process these two types of input, respectively, while a bar decoder is employed to re-construct notes and predict musical features. The flame and snowflake icons in the image correspond to Table 1, indicating the three-step training schedule for each module.

### 3.3 UNIFICATION OF AUDIO INTO SYMBOLIC MUSIC

**Pre-processing.** Since general audio formats such as MP3 and WAV are dense time-sequence data, UniComposer applies the Harmonic Constant-Q Transform (H-CQT) (Bittner et al. (2017)) to process them. H-CQT takes an audio signal as input, and the output is a three-dimensional tensor representing the time-frequency representation at different harmonic frequencies. Its main function is to capture both the fundamental frequency and its harmonic components, providing a richer spectral representation. UniComposer computes H-CQT with 3 bins per semitone and a hop size of 11 ms, operating with a sample rate of 22.05 kHz and utilizes 8 harmonics. For an input of shape  $(1, t \times 22, 050)$ , this H-CQT produces an output with the shape  $(8, t \times 172, 128)$ .

**Audio Encoder.** The audio encoder aims to map audio segmentation to the same note and musical feature space of the symbolic encoder. After pre-processing, a fully convolutional model followed by an output layer is applied to produce the note and musical feature. The architecture is similar to that proposed by Bittner et al. (2022), but with fewer layers, as the audio encoder processes shorter input segments. Since H-CQT captures all harmonic waves, the convolutional layers with deep kernels are able to detect notes with correct pitch and duration. Finally, using a series of linear and activation layers, the audio encoder generates the expected two features.

### 3.4 REFACTORING AND TRAINING PROCESS FOR FEATURE EXTRACTION

**Bar Decoder.** The task of the bar decoder is to reconstruct all the notes from the input based on the note feature, and predict four musical attributes of the bar based on the musical feature. For note reconstruction, the bar decoder uses a classical transformer decoder with 4 layers and 4 attention heads. The note feature is replicated  $n$  times to serve as the input hidden state of the decoder. The transformer generates an embedding vector for each position auto-regressively, with a dimension of 16. For musical feature prediction, a multi-layer perceptron is applied to the musical feature, followed by four classifiers to predict the four musical attribute of a bar.

**Three-Step Training Process.** Firstly, the E-MLP is optimized to establish a stable embedding space for the notes. A separate MLP architecture (D-MLP), which includes five distinct classifiers, is employed to decode specific attributes from the E-MLP’s output. Secondly, the symbolic encoder and bar decoder are trained jointly in a self-reconstruction manner for symbolic music. Specifically, the symbolic encoder extracts both the note and feature from a musical bar, while the bar decoder attempts to reconstruct the bar and predict its musical attributes. Thirdly, the bar decoder is frozen, and a similar self-reconstruction training approach is applied to the audio encoder, in conjunction with the bar decoder. The key difference here is that a converter, based on the open-source tool Fluidsynth (Tom Moebert (2024)), is used to convert symbolic music into audio waveforms, which serve as the input for the audio encoder. These three steps can be found in Table 1.

Table 1: Three step training process for the encoders

Step	Trainable Part	Target
1	E-MLP, D-MLP	Stable Embedding Vector
2	Symbolic Encoder, Bar Decoder	Stable Symbolic Feature Space
3	Audio Encoder	Unified Audio and Symbolic Feature

### 3.5 HIERARCHICAL GENERATION IN FEATURE SPACE

UniComposer operates following the process depicted by the solid lines in Figure 2, but within a unified feature space. The generation process begins with the main melody’s features, progressing from reduced to detailed. Finally, the feature are decoded and merged together.

**Cascaded Diffusion Models.** UniComposer utilizes four Transformer-based diffusion models (denoted as DM1 to DM4), all of which share the same architecture. They work within the bar-level unified feature space, using background conditions to generate target features from Gaussian noise. Starting with the note and musical features extracted from a given melody, DM1 generates the reduced monophonic, polyphonic, and percussion features for each bar. Then, with the note and musical features, combined with one of the reduced features as part of the background condition, DM2–DM4 generate detailed features for each bar. Detailed configurations for every diffusion model can be found in Table 2.

Table 2: Configuration of the four diffusion models. ‘R-’ stands for ‘Reduced-’.  $n$  represents the number of bars, and each channel has the shape  $(n, 256)$ , as each type of feature is represented by 256 dimensions.

	Background Cond.		Output	
	Shape	Interpretation	Shape	Interpretation
DM1	$(2, n, 256)$	Note, Musical	$(3, n, 256)$	R-Mono., R-Poly., R-Perc.
DM2	$(3, n, 256)$	Note, Musical, R-Mono.	$(1, n, 256)$	Detailed Mono.
DM3	$(3, n, 256)$	Note, Musical, R-Poly.	$(1, n, 256)$	Detailed Poly.
DM4	$(3, n, 256)$	Note, Musical, R-Perc.	$(1, n, 256)$	Detailed Perc.

**Attention Mechanism.** Inspired by Shabani et al. (2023), we design three types of attention mechanisms. 1) Self Attention, which restricts the attention to within a single bar to enhance local information; 2) Global Attention, a standard self-attention applied across all bars in a song; and 3) Local Attention, which operates at a 4-bar level to help the model capture short-term dependencies within the song. In each attention layer, we learn three separate sets of key, query and value matrices, using four multi-heads. The outputs of the three attention mechanisms are summed and then passed through a standard Add & Norm layer. The denoising process repeats this attention block six times. The masks used to implement these three types of attention mechanisms are shown in Figure 4.

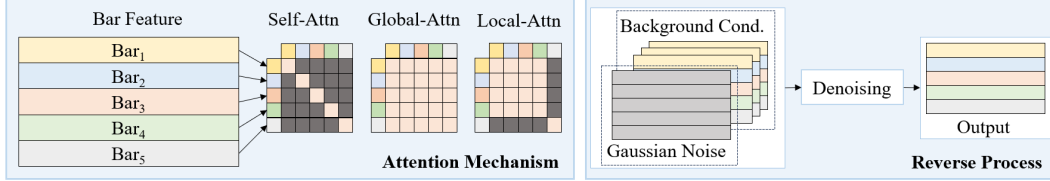


Figure 4: Attention mechanism and diffusion details. Three types of attention are applied using masks and separate query, key and value sets. The Transformer-based diffusion model learns to reconstruct the features of each bar from Gaussian noise, based on specific background conditions.

**Data Augmentation Derived from Audio.** Due to the richness of audio data, this can be achieved in two ways: 1) For audio tracks that have already been separated by instrument type, which is common in DJ music, both the input and output of DM1 can be derived by combining tracks based on their track names, utilizing features extracted from the audio encoder. 2) For general audio music, the combination of the audio encoder and bar decoder can be used as an automatic music transcription system, converting audio into symbolic music, thereby enhancing data diversity for training.

## 4 EXPERIMENTS

We implemented UniComposer using PyTorch, with the diffusion models built upon a public implementation of Guided-Diffusion (Dhariwal & Nichol (2021)). All experiments were conducted on a single NVIDIA RTX 4090 GPU. In the following sections, we first introduce the composition of the dataset used to train each part of UniComposer, and then evaluate its performance in terms of the unified feature and the generated quality. We also conduct ablation studies on the attention mechanism and the cascaded diffusion models, demonstrating their necessity.

### 4.1 DATASET

We use notes, bars, and songs from the Lakh MIDI Dataset (LMD, Raffel (2016)) to train UniComposer. The LMD dataset contains over 170,000 unique multi-track MIDI files, with each track labeled by instrument name. We reserve 1,000 songs as evaluation set.

**Dataset for encoders.** In Step 1, we collect unique notes from LMD to train the E-MLP and D-MLP. In Step 2, we use non-empty bars with varying note densities, instruments, and durations to train the symbolic encoder and bar decoder. In Step 3, we first synthesize entire songs from MIDI files using Fluidsynth (Tom Moebert (2024)), then extract the corresponding segments of bars filtered in Step 2 from the audio. This process results in MIDI-audio bar pairs, which are used to train the audio encoder while keeping the bar decoder frozen.

**Dataset for Cascaded Diffusion Models.** For training the four diffusion models, we filter out songs in LMD that are either too short or too long, selecting only those that contain a pre-labeled “melody” or “main melody” along with at least one monophonic instrument, one polyphonic instrument, and one percussion instrument. A detailed breakdown of the training dataset is provided in Table 3.

Table 3: Dataset composition for encoder, decoder and cascaded diffusion models of UniComposer

Part	Encoder and Decoders			Cascaded Diffusion Models			
	Step1	Step2	Step3	DM1	DM2	DM3	DM4
Unit	Note	Bar	MIDI-Audio Bar Pair	Song	Song	Song	Song
Count	608,020	915,644	915,644	13,232	13,232	13,232	13,232



## 4.2 EVALUATION ON UNIFIED FEATURE

**Method and Metrics.** To assess the performance of our encoders, we focused on the note estimation task by integrating the audio encoder and bar decoder (A&B). This system takes audio input and estimates notes, reflecting the core capabilities of the model. Performance is measured using the note-level F-measure ( $F$ ), where a note is considered correct if its pitch, onset, and offset fall within a defined threshold compared to the ground truth. Additionally, we report the note-level F-measure without considering offsets ( $F_{no}$ ), following the same criteria as the F-measure but ignoring offsets, as well as the bar-level note accuracy ( $Acc$ ). These metrics are computed using the `mir_eval` library (Liang et al. (2015)).

**Baseline Settings.** As a baseline, we compare the performance of A&B on note onset and duration with other note estimation pipelines, including Basic-Pitch (Bittner et al. (2022)) and MI-AMT (Wu et al. (2020)). To assess the necessity of E-MLP and D-MLP, we also create a variation of Uni-Composer where these components are replaced by a vocabulary of 608,020 tokens, followed by a standard embedding layer. This model is denoted as Vocab-AE. The reserved evaluation set are separated into main melody, monophonic and polyphonic tracks, and we report the note reconstruction accuracy in Table 4.

**Analysis.** It is important to note that Basic-Pitch and MI-AMT only estimate pitch, onset, and duration, whereas A&B and Vocab-AE also predicts velocity and instrument. Vocab-AE performs worse, emphasizing the value of the stable note embeddings produced by E-MLP. A&B performs on par with Basic-Pitch in most cases. Given that A&B operates in a unified feature space while Basic-Pitch is specialized for note estimation, A&B demonstrates strong performance.

Table 4: Note estimation evaluation of Audio Encoder & Bar Decoder (A&B) with other baselines

	Melody			Mono.			Poly.		
	$Acc$	$F_{no}$	$F$	$Acc$	$F_{no}$	$F$	$Acc$	$F_{no}$	$F$
<b>Vocab-AE</b>	0.44	0.36	0.31	0.12	0.08	0.04	0.11	0.06	0.02
<b>Basic-Pitch</b>	0.91	0.79	0.71	0.76	0.69	0.61	0.70	0.65	0.63
<b>MI-AMT</b>	0.80	0.68	0.62	0.64	0.56	0.50	0.42	0.34	0.25
<b>A&amp;B (Ours)</b>	0.94	0.82	0.74	0.72	0.63	0.57	0.77	0.62	0.59

## 4.3 EVALUATION ON GENERATION QUALITY

**Metrics.** A variety of metrics have been proposed to evaluate the harmony, quality and similarity of music generation. We borrowed the metrics from Ji et al. (2023) and Ren et al. (2020) for evaluation: chord accuracy (CA) that measures the harmony, and averaging overlapped area (OA) of distributions ( $\mathcal{D}_A$ ,  $A$  can be one of  $P(itch)$ ,  $V(elocity)$ ,  $D(uration)$ , and  $OI(onsetInterval)$ ) to measure the difference between generated musical piece and ground-truth musical piece.

**Baseline Settings.** We introduce MuseGAN (M-G, Dong et al. (2018)), PopMAG (P-M, Ren et al. (2020)) and Figaro (F-G, von Rütte et al. (2023)). To make UniComposer (U-C) comparable with the three model: 1) For MuseGAN, we take 4 bars as input to generate 4 tracks (guitar, drum, string and bass) conditioned on a main melody of piano track, and free of musical information since MuseGAN doesn't use chord and other features. 2) For PopMAG, we use the same task, while extend the generation length into 64 bars. 3) For Figaro, we use only features including chord, beat and target instrument as input, as Figaro just take them as input to generate new rhythm. We compare the ability of UniComposer with them separately on evaluation set. Results are in Table 5.

Table 5: Music Generation Quality Evaluation. The three groups carry out different tasks.

	$CA$	$\mathcal{D}_P$	$\mathcal{D}_V$	$\mathcal{D}_D$	$\mathcal{D}_{OI}$
<b>M-G</b>	$0.334 \pm 0.011$	$0.294 \pm 0.013$	$0.342 \pm 0.008$	$0.311 \pm 0.016$	$0.336 \pm 0.010$
<b>U-C</b>	$0.397 \pm 0.010$	$0.358 \pm 0.012$	$0.407 \pm 0.016$	$0.301 \pm 0.009$	$0.398 \pm 0.013$
<b>P-M</b>	$0.589 \pm 0.011$	$0.601 \pm 0.013$	$0.492 \pm 0.009$	$0.523 \pm 0.007$	$0.655 \pm 0.007$
<b>U-C</b>	$0.566 \pm 0.010$	$0.625 \pm 0.018$	$0.511 \pm 0.014$	$0.507 \pm 0.005$	$0.628 \pm 0.014$
<b>F-G</b>	$0.541 \pm 0.011$	$0.356 \pm 0.006$	$0.642 \pm 0.008$	$0.477 \pm 0.008$	$0.433 \pm 0.004$
<b>U-C</b>	$0.445 \pm 0.012$	$0.451 \pm 0.016$	$0.592 \pm 0.011$	$0.504 \pm 0.011$	$0.507 \pm 0.008$



**Instrument Assignment Capability.** Since UniComposer is the first system to automatically select instruments, metrics are limited. To assess its effectiveness, we analyzed the instrument distribution across 10 melodies in four emotional categories: happy, sad, soothing, and stirring. The occurrence of guitar, violin, trumpet, and flute in these melodies was recorded and summarized in Table 6.

Table 6: Instrument occurrence in different emotional melodies, each category containing 10 pieces.

	Guitar	Violin	Trumpet	Flute		Guitar	Violin	Trumpet	Flute
<b>Happy</b>	8	5	3	0	<b>Soothing</b>	7	7	2	1
<b>Sad</b>	8	7	0	0	<b>Stirring</b>	8	6	3	0

#### 4.4 ABLATION STUDY

**Attention Mechanisms.** We begin by experimenting with the global attention mask (U-GA), a common approach in natural language processing. Next, we test the addition of either the self-attention mask (U-GSA) or the local attention mask (U-GLA) to global attention individually. Finally, we compare these three attention mechanisms with UniComposer (UniComp.). As shown in Table 7, global attention establishes a foundation by providing cross-bar focus within a music piece. Self-attention refines this by encouraging the model to emphasize pitch and velocity, yielding slight improvements. Local attention, which mimics a composer’s focus on musical sub-structures, further enhances overall quality.

**Cascaded Diffusion Models.** To evaluate the hierarchical generation process, we first removed diffusion models DM2 through DM4, leaving only DM1 (U-DMa). In this configuration, DM1 is trained to generate the entire MIDI based solely on the melody and musical features. We also implemented a second structure in which DM2 through DM4 were compressed into a single diffusion model (U-DMb), while DM1 remained unchanged. In this variant, the monophonic, polyphonic, and percussion features were combined, and a single diffusion model was trained to generate the complete raw MIDI. As shown in Table 7, the results demonstrate that it is challenging for a single diffusion model to capture the full data distribution, from the main melody to the detailed instrumentation. Due to the complexity of each individual instrument, integrating them into a single diffusion model proves difficult.

Table 7: Ablation study on attention mechanisms and cascaded diffusion models.

	$\mathcal{C}\mathcal{A}$	$\mathcal{D}_P$	$\mathcal{D}_V$	$\mathcal{D}_D$	$\mathcal{D}_{OI}$
<b>U-GA</b>	$0.549 \pm 0.008$	$0.501 \pm 0.006$	$0.405 \pm 0.004$	$0.492 \pm 0.007$	$0.540 \pm 0.006$
<b>U-GSA</b>	$0.532 \pm 0.009$	$0.530 \pm 0.006$	$0.421 \pm 0.010$	$0.493 \pm 0.006$	$0.559 \pm 0.010$
<b>U-GLA</b>	$0.601 \pm 0.008$	$0.627 \pm 0.011$	$0.537 \pm 0.012$	$0.507 \pm 0.006$	$0.629 \pm 0.015$
<b>U-DMa</b>	$0.187 \pm 0.006$	$0.156 \pm 0.002$	$0.174 \pm 0.004$	$0.102 \pm 0.001$	$0.168 \pm 0.004$
<b>U-DMb</b>	$0.364 \pm 0.008$	$0.420 \pm 0.007$	$0.386 \pm 0.009$	$0.366 \pm 0.005$	$0.309 \pm 0.004$
<b>UniComp.</b>	<b><math>0.607 \pm 0.008</math></b>	<b><math>0.644 \pm 0.013</math></b>	<b><math>0.549 \pm 0.011</math></b>	<b><math>0.511 \pm 0.009</math></b>	<b><math>0.643 \pm 0.010</math></b>

## 5 CONCLUSION

In this work, we propose UniComposer, a band-level music generation pipeline that involves collaborative roles of instruments and bridges the gap between symbolic and audio music. UniComposer first extracts features from both symbolic bars and audio segments, and then applies four cascaded diffusion models to generate the corresponding features for each bar. Experiments on the feature space demonstrate the effectiveness of the unified latent space, while comparisons with other music generation pipelines highlight UniComposer’s ability to generate band-level music. Additionally, the ablation studies confirm the effectiveness of the various components within UniComposer.

**Discussion and Future work.** There are potential discrepancies in data distribution between synthesized audio generated from MIDI, background noise, and real-world recordings. However, a limitation is the lack of extensive datasets containing paired raw audio and corresponding MIDI for comprehensive training. Meanwhile, UniComposer’s capacity to process human vocals remains limited due to insufficient data. Another constraint is the requirement for prior BPM (Beats Per Minute) information, which can restrict the data. Addressing these issues may be a focus of future research.

## REFERENCES

- Lilac Atassi. Generating symbolic music using diffusion models. *arXiv preprint arXiv:2303.08385*, 2023.
- Emmanouil Benetos. Polyphonic note and instrument tracking using linear dynamical systems. 2017.
- bitsforbyte. Wave file format specification. <https://www.bitsforbyte.com/2021/04/09/wave-file-format-specification/>, 2021. Accessed: 2024-09-26.
- Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep salience representations for f0 estimation in polyphonic music. In *ISMIR*, pp. 63–70. Suzhou (China), 2017.
- Rachel M Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 781–785. IEEE, 2022.
- Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. *arXiv preprint arXiv:1809.07600*, 2018.
- Emanuele Cosenza, Andrea Valenti, and Davide Bacciu. Graph-based polyphonic multitrack music generation. *arXiv preprint arXiv:2307.14928*, 2023.
- Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.
- Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley, and Taylor Berg-Kirkpatrick. Multitrack music transformer. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.
- Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(8):2121–2133, 2010.
- Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pp. 1068–1077. PMLR, 2017.
- Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.
- Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.
- Jeff Ens and Philippe Pasquier. Mmm: Exploring conditional multi-track music generation with the transformer. *arXiv preprint arXiv:2008.06048*, 2020.
- Sebastian Ewert and Mark B Sandler. An augmented lagrangian method for piano transcription using equal loudness thresholding and lstm-based decoding. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 146–150. IEEE, 2017.
- Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse Engel. Mt3: Multi-task multitrack music transcription. *arXiv preprint arXiv:2111.03017*, 2021.

- Joshua P Gardner, Simon Durand, Daniel Stoller, and Rachel M Bittner. Llark: A multimodal instruction-following language model for music. In *Forty-first International Conference on Machine Learning*, 2023.
- Wei Guo. Multi-track music generation based on the ac algorithm and global value return network. *International Journal of Advanced Computer Science & Applications*, 15(3), 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.
- 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. *arXiv preprint arXiv:1809.04281*, 2018.
- Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.
- 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, 2020.
- Yujia Huang, Adishree Ghatare, Yuanzhe Liu, Ziniu Hu, Qinsheng Zhang, Chandramouli S Sastri, Siddharth Gururani, Sageev Oore, and Yisong Yue. Symbolic music generation with non-differentiable rule guided diffusion. *arXiv preprint arXiv:2402.14285*, 2024.
- Shulei Ji, Xinyu Yang, and Jing Luo. A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges. *ACM Computing Surveys*, 56(1):1–39, 2023.
- Junyan Jiang, Gus G Xia, Dave B Carlton, Chris N Anderson, and Ryan H Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 516–520. IEEE, 2020.
- Tornike Karchkhadze, Mohammad Rasool Izadi, Ke Chen, Gerard Assayag, and Shlomo Dubnov. Multi-track musicldm: Towards versatile music generation with latent diffusion model. *arXiv preprint arXiv:2409.02845*, 2024.
- Anssi Klapuri and Manuel Davy. Signal processing methods for music transcription. 2007.
- Jean-Marie Lemerrier, Simon Rouard, Jade Copet, Yossi Adi, and Alexandre Déffosez. An independence-promoting loss for music generation with language models. *arXiv preprint arXiv:2406.02315*, 2024.
- Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, et al. Mert: Acoustic music understanding model with large-scale self-supervised training. *arXiv preprint arXiv:2306.00107*, 2023.
- Che-Yuan Liang, Li Su, Yi-Hsuan Yang, and Hsin-Ming Lin. Musical offset detection of pitched instruments: The case of violin. In *ISMIR*, pp. 281–287, 2015.
- Jiafeng Liu, Yuanliang Dong, Zehua Cheng, Xinran Zhang, Xiaobing Li, Feng Yu, and Maosong Sun. Symphony generation with permutation invariant language model. *arXiv preprint arXiv:2205.05448*, 2022.
- Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation. *arXiv preprint arXiv:2302.02257*, 2023.
- MIDI-Association. Midi-2.0. <https://midi.org/midi-2-0>, 2024. Accessed: 2023-06.

- Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021.
- mpgedit. Mp3 file format specification. [http://mpgedit.org/mpgedit/mpeg\\_format/MP3Format.html](http://mpgedit.org/mpgedit/mpeg_format/MP3Format.html), 2003. Accessed: 2024-09-26.
- Ryo Nishikimi, Eita Nakamura, Masataka Goto, and Kazuyoshi Yoshii. Audio-to-score singing transcription based on a crnn-hsmm hybrid model. *APSIPA Transactions on Signal and Information Processing*, 10:e7, 2021.
- OpenAI. Musenet. <https://openai.com/index/musenet/>, 2024. Accessed: 2019-04-25.
- KR Prajwal, Bowen Shi, Matthew Le, Apoorv Vyas, Andros Tjandra, Mahi Luthra, Baishan Guo, Huiyu Wang, Triantafyllos Afouras, David Kant, et al. Musicflow: Cascaded flow matching for text guided music generation. In *Forty-first International Conference on Machine Learning*, 2024.
- Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.
- 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.
- Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *International conference on machine learning*, pp. 4364–4373. PMLR, 2018.
- Mohammad Amin Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5466–5475, 2023.
- Marcus Weseloh Tom Moebert, Jean-Jacques Ceresa. Midi-2.0. <https://midi.org/midi-2-0>, 2024. Accessed: 2023-06.
- A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- 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.
- Ziyu Wang, Lejun Min, and Gus Xia. Whole-song hierarchical generation of symbolic music using cascaded diffusion models. *arXiv preprint arXiv:2405.09901*, 2024.
- 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, 2023.
- Yu-Te Wu, Berlin Chen, and Li Su. Multi-instrument automatic music transcription with self-attention-based instance segmentation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2796–2809, 2020.
- Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel. Midi-ddsp: Detailed control of musical performance via hierarchical modeling. *arXiv preprint arXiv:2112.09312*, 2021.
- Yao Yao, Peike Li, Boyu Chen, and Alex Wang. Jen-1 composer: A unified framework for high-fidelity multi-track music generation. *arXiv preprint arXiv:2310.19180*, 2023.
- Botao Yu, Peiling Lu, Rui Wang, Wei Hu, Xu Tan, Wei Ye, Shikun Zhang, Tao Qin, and Tie-Yan Liu. Museformer: Transformer with fine-and coarse-grained attention for music generation. *Advances in Neural Information Processing Systems*, 35:1376–1388, 2022.

Jincheng Zhang, Jingjing Tang, Charalampos Saitis, and György Fazekas. Composer style-specific symbolic music generation using vector quantized discrete diffusion models. *arXiv preprint arXiv:2310.14044*, 2023.

Jinlong Zhu, Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Mmt-bert: Chord-aware symbolic music generation based on multitrack music transformer and musicbert. *arXiv preprint arXiv:2409.00919*, 2024.

Ye Zhu, Yu Wu, Kyle Olszewski, Jian Ren, Sergey Tulyakov, and Yan Yan. Discrete contrastive diffusion for cross-modal music and image generation. *arXiv preprint arXiv:2206.07771*, 2022.

## APPENDICES

This section aims to offer a comprehensive explanation of the technical concepts and mathematical formulations discussed in the paper, along with additional examples and case studies to better illustrate their practical applications.

## A DETAILS FOR ENCODER, DECODER AND TRAINING

### A.1 E-MLP AND D-MLP STRUCTURE

We denote a five-attribute note as  $N = [O, D, P, V, I]$ , where  $O$  represents the onset time,  $D$  stands for duration,  $P$  denotes pitch,  $V$  refers to velocity, and  $I$  indicates instrument. A vocabulary is constructed to encompass all possible tokens corresponding to these five attributes.

**E-MLP (Encoder MLP).** The E-MLP model is responsible for encoding the input tokens of the note. It consists of a token embedding layer that maps each input token to a dense vector representation. Specifically, each token is passed through an embedding layer, followed by a series of linear transformations and activation functions. The output is a 16-dimensional vector for each note. Mathematically, the process can be represented as:

$$E_{\text{MLP}}(N) = \sigma(W_2(\sigma(W_1 T_N + b_1)) + b_2)$$

where:  $T_N$  represents the input token embedding for note  $N$ ,  $W_1$  and  $W_2$  are the weight matrices for the two linear layers,  $b_1$  and  $b_2$  are the bias terms, and  $\sigma$  denotes the activation function ReLU. The final output of the E-MLP is a fixed 16-dimensional latent representation of the note  $N$ .

**D-MLP (Decoder MLP).** The D-MLP model aims to reconstruct or predict the five discrete attributes ( $O, D, P, V, I$ ) from the latent representation generated by the E-MLP. The D-MLP consists of two linear-activation layers that are stacked together. For each attribute, a separate classifier is used, as each attribute takes discrete values from predefined categories. The process for predicting each attribute can be described as follows:

$$\hat{A}_i = \text{softmax}(W_i \cdot \sigma(W_{i-1} \cdot Z + b_{i-1}) + b_i)$$

where:  $Z$  is the latent 16-dimensional vector obtained from the E-MLP,  $W_{i-1}$  and  $W_i$  are weight matrices, with  $W_{i-1}$  belonging to the first linear layer and  $W_i$  to the classifier for attribute  $A_i$  (where  $i \in \{O, D, P, V, I\}$ ),  $b_{i-1}$  and  $b_i$  are the bias terms, and softmax is applied to ensure that the output is a valid probability distribution over the possible discrete values of each attribute. Thus, the D-MLP outputs five probability distributions, one for each attribute of the note.

**Overall Process.** In summary, the E-MLP encodes the note into a fixed-length latent representation, while the D-MLP decodes this representation into the predicted values of the five attributes. The entire process can be formulated as:

$$\hat{N} = D_{\text{MLP}}(E_{\text{MLP}}(N))$$

where  $\hat{N}$  represents the predicted note attributes.

### A.2 FORMULATION OF THE NOTE ENCODER

As stated in the paper, the note encoder takes a sequence of embedding vectors as input and utilizes a 4-head, 6-layer Transformer architecture. Denote each note as  $N$ , and let the embedding of the note be processed by the E-MLP, denoted as  $E_{\text{MLP}}$ . We use the final hidden state of the Transformer as the input to two separate output layers. Each of these output layers consists of a linear transformation followed by a non-linear activation function. These layers output two distinct features: note features and musical features.

The input to the Transformer is a sequence of embedding vectors, denoted as  $X = [E_{\text{MLP}}(N_1), E_{\text{MLP}}(N_2), \dots, E_{\text{MLP}}(N_T)]$ , where  $T$  is the length of the sequence, and each  $E_{\text{MLP}}(N_i)$

is the 16-dimensional embedding of note  $N_i$  obtained from the E-MLP. The Transformer first maps these to its feature space, and then processes this sequence through its multi-head self-attention mechanism and a series of feed-forward layers. The final hidden state  $H_T$  at time step  $T$  (corresponding to the last note in the sequence) is used as the input to the subsequent output layers.

The Transformer-based encoding process can be formulated as:

$$H_T = \text{Transformer}(X)$$

where  $H_T$  is the hidden state vector at the last layer and the final time step of the Transformer.

We then apply two separate output layers to  $H_T$ , one for extracting note features and another for extracting musical features. Each output layer consists of a linear transformation followed by a non-linear activation function. The outputs are defined as follows:

$$F_{\text{note}} = \sigma(W_{\text{note}}H_T + b_{\text{note}}), F_{\text{music}} = \sigma(W_{\text{music}}H_T + b_{\text{music}})$$

Thus, the final formulation can be expressed as:

$$F_{\text{note}}, F_{\text{music}} = \text{Output Layers}(\text{Transformer}(E_{\text{MLP}}(N_1), E_{\text{MLP}}(N_2), \dots, E_{\text{MLP}}(N_T)))$$

### A.3 LOSS FOR THREE-STEP TRAINING OF FEATURE EXTRACTION

**Step 1: E-MLP and D-MLP.** Given a note  $N = [O, D, P, V, I]$  representing its five attributes (onset, duration, pitch, velocity, instrument), the task is to embed this note into a 16-dimensional vector using the E-MLP. The D-MLP then reconstructs the five attributes from this 16-dimensional representation. The loss function used to minimize the reconstruction error across the five attributes is formulated as:

$$\mathcal{L} = \sum_{i=1}^5 \text{CrossEntropy}(A_i, \hat{A}_i)$$

where  $A_i$  are the true attributes and  $\hat{A}_i$  are the predicted attributes. Each attribute is treated as a classification task using cross-entropy loss.

**Step 2: Note Encoder and Bar Decoder.** The note encoder processes a sequence of 16-dimensional embeddings, generating both note-level and musical-level features. The bar decoder takes the note features as input and auto-regressively generates the embedding vector at each position.

The loss function for the note embeddings is the Mean Squared Error (MSE) between the predicted and target sequences, both of which consist of 16-dimensional embedding vectors. This can be formulated as:

$$\mathcal{L}_{\text{note}} = \frac{1}{T} \sum_{t=1}^T \text{MSE}(E_t, \hat{E}_t)$$

where  $E_t$  and  $\hat{E}_t$  represent the true and predicted 16-dimensional embedding vectors at time step  $t$ , and  $T$  is the length of the sequence.

For the musical attributes (chord, time signature, tonality and dynamics), the loss is computed using a classifier-based loss function, typically cross-entropy, for each attribute:

$$\mathcal{L}_{\text{attr}} = \sum_{i=1}^4 \text{CrossEntropy}(A_i, \hat{A}_i)$$

where  $A_i$  and  $\hat{A}_i$  are the true and predicted values for the four musical attributes, respectively.



**Step 3: Audio Encoder.** As described in Section 3, the training data for the audio encoder is derived from MIDI files. Given a MIDI song  $S$ , we convert it to an audio waveform  $W$ . For each bar  $B_i$  in  $S$  and its corresponding waveform segment in  $W$  (denoted as  $B_w$ ), we extract the note sequence from  $S$  and use the E-MLP to obtain an embedding sequence. Next,  $B_w$  is fed into the audio encoder, which outputs the note features and musical features. The bar decoder then uses the note features to reconstruct the embedding vector list, while the musical features are used to predict the four attributes (onset, duration, pitch, and velocity).

The loss function in this process consists of two parts: 1) MSE Loss: The Mean Squared Error (MSE) between the input embedding sequence from the E-MLP and the output embedding sequence generated by the bar decoder; 2) Classification Loss: A cross-entropy loss for the four musical attributes predicted from the musical features.

$$\mathcal{L}_{\text{audio}} = \frac{1}{T} \sum_{t=1}^T \text{MSE}(E_t, \hat{E}_t) + \sum_{i=1}^4 \text{CrossEntropy}(A_i, \hat{A}_i)$$

where  $E_t$  is the input embedding at time step  $t$ ,  $\hat{E}_t$  is the reconstructed embedding, and  $T$  is the length of the sequence. In this training process, the bar decoder is frozen, and only the audio encoder is updated.

## B DEALING WITH AUDIO INPUT

### B.1 HARMONIC CONSTANT-Q TRANSFORMATION

The Harmonic Constant-Q Transform (H-CQT) extends the traditional Constant-Q Transform (CQT) by introducing a third dimension, designed to align harmonically related frequencies. This alignment allows the transformation to more effectively capture harmonic information, which is crucial in many audio analysis tasks, such as music transcription, pitch tracking, and timbre analysis. By analyzing harmonics across multiple frequency bins, the H-CQT emphasizes both the fundamental frequencies and their harmonics, improving the representation of harmonic structures.

The H-CQT is mathematically defined as follows:

$$X_{HCQT}(h, t, k) = \sum_{n=0}^{N-1} x(n) w(n-t) e^{-2\pi i h f_k n}$$

where:

- $X_{HCQT}(h, t, k)$  represents the H-CQT coefficient at harmonic index  $h$ , time frame  $t$ , and frequency bin  $k$ .
- $x(n)$  is the input signal in the time domain, typically a sampled audio signal of length  $N$ .

- $w(n-t)$  is a window function (such as a Hann or Hamming window) centered around time frame  $t$ , which serves to localize the analysis in both time and frequency domains. The window function is crucial in minimizing spectral leakage and ensuring that the signal is properly segmented in time.

- $f_k$  denotes the center frequency of the  $k$ -th bin in the Constant-Q Transform, which is characterized by a logarithmic frequency spacing, making it well-suited for processing audio signals where perception of pitch is often logarithmic in nature.

- $h$  is the harmonic index, which allows for the computation of higher harmonics of the fundamental frequencies present in the signal. Specifically, when  $h = 1$ , the transform focuses on the fundamental frequencies, while  $h > 1$  captures the harmonics.

In this formulation, the summation operates over all signal samples  $x(n)$ , where  $n = 0, 1, \dots, N-1$ . The term  $e^{-2\pi i h f_k n}$  is a complex exponential that shifts the signal by a frequency determined by both the harmonic index  $h$  and the center frequency  $f_k$ . This process is analogous to taking a short-time Fourier transform, but in the context of CQT, it scales the analysis resolution logarithmically across frequencies, ensuring that each frequency bin captures a musically relevant pitch range.

By introducing the harmonic index  $h$ , the H-CQT enables the simultaneous representation of multiple harmonics for each fundamental frequency. This multidimensional transform offers a richer description of harmonic content compared to the traditional CQT, making it a powerful tool for analyzing signals where harmonic structure is important, such as music.

## B.2 TIME ALIGNMENT FOR THE SECOND OF A BAR

As discussed in the conclusion section, obtaining the BPM (beats per minute) information for a piece of audio is a crucial step. After acquiring this information, we extract a 2-second audio segment from the raw input and process it using the audio encoder and bar decoder to predict the time signature of the audio.

Let us assume the time signature is represented as  $\frac{n}{m}$ , where  $n$  is the number of beats per measure (or bar), and  $m$  defines the note value that constitutes one beat (e.g., quarter note, eighth note). Additionally, let the BPM of the audio be denoted as BPM.

The duration of one bar in seconds, which corresponds to the time it takes to complete a single measure, can be calculated using the following formula:

$$\text{Duration of one bar (in seconds)} = \frac{n}{\text{BPM}} \times 60$$

Where: -  $n$  is the number of beats per bar (from the time signature), - BPM is the beats per minute, - 60 is the number of seconds in a minute.

Using this calculated duration, we define the new segment length for processing the entire audio. The audio will now be divided into segments corresponding to the duration of one bar, allowing for more precise temporal alignment. This adjustment ensures that each segment represents a musically coherent unit, aligning with the beats and measures of the audio.

## C INSTRUMENT SEPARATION

To simplify the classification of instruments, we propose a streamlined categorization based on their corresponding MIDI program IDs, as shown in Table 8. This approach consolidates multiple MIDI instrument categories into broader groups, facilitating more efficient handling in instrument separation tasks. For example, various types of pianos (e.g., acoustic grand piano, electric piano) are grouped under a single "Pianos" category. Similar consolidations are applied across other instrument families, such as chromatic percussion, guitars, and basses.

The primary objective of this reduction is to enhance the efficiency of instrument separation models by lowering the complexity of the classification process while preserving the key distinctions necessary for accurate audio representation and processing. Each generalized instrument group corresponds to a specific range of MIDI program IDs, with the exception of standard drums, which are categorized separately.

We categorize instruments into three main groups: Monophonic, Polyphonic, and Percussion. Below is the detailed classification for each category, as shown in Table 9.

**Monophonic Instruments.** Monophonic instruments are those that generally produce only one note at a time. These instruments are commonly used in melodic lines and solos, as their tonal clarity allows for strong emphasis on individual notes including: 1) Violin: A bowed string instrument known for its wide range and expressive dynamics. 2) Trumpet: A brass instrument with a bright and powerful sound. 3) Saxophone: A woodwind instrument with a reedy sound, common in jazz and classical genres. 4) Bassoon: A large woodwind instrument, characterized by its deep, rich sound. 5) Flute: A woodwind instrument producing sound by air flow across an opening, known for its bright and airy timbre.

**Polyphonic Instruments.** Polyphonic instruments can play multiple notes simultaneously, making them suitable for harmonic and chordal accompaniment. These instruments are essential in creating complex textures in music including: 1) Piano: A keyboard instrument capable of producing rich harmonies and complex melodies. 2) Guitar: A string instrument often used in various musical genres, capable of playing chords and melodic lines. 3) Organs: A keyboard instrument with

Table 8: Instrument reduction according to MIDI program IDs

#	Program ID	Instrument in UniComposer
1	0-7	Pianos
2	8-15	Chromatic Percussion
3	16-23	Organs
4	24-31	Guitar
5	32-39	Bass
6	40-43	Violin
7	54-58	Trumpet
8	62-65	Saxophone
9	66-69	Bassoon
10	70-77	Flute
11	102-109	Ethnic
12	110-117	Melodic Percussion
13	/	Standard Drums

multiple sound-producing pipes, known for its grandeur in churches and large venues. 4) Bass (Electric/Acoustic): A low-pitched instrument that plays a foundational role in harmonic structures, often contributing to both rhythm and harmony.

**Percussion Instruments.** Percussion instruments are classified based on their ability to produce sound through striking or hitting. This category includes both rhythmic and melodic percussion instruments including: 1) Melodic Percussion: Instruments such as xylophones and vibraphones that can produce pitched sounds, allowing for melodic content. 2) Chromatic Percussion: Instruments capable of producing all the notes in the chromatic scale, often used for melodic and harmonic roles in orchestras. 3) Standard Drums: Drums typically found in drum kits, primarily used for rhythmic purposes in various musical genres.

Table 9: Three category instrument separation.

Category	Instruments
Monophonic	Violin, Trumpet, Saxophone, Bassoon, Flute
Polyphonic	Piano, Guitar, Organs, Bass
Percussion	Melodic Percussion, Chromatic Percussion, Standard Drums

## D DATASETS

### D.1 NOTE VOCABULARY SELECTION

We collect 608,020 unique notes from LMD dataset. The rationale behind collecting all notes that occur in the LMD dataset is outlined as follows:

1) Practicality of Limiting the Scope: It is computationally impractical to account for every possible combination of notes, as the total number of potential notes within the five-attribute representation is exceedingly large. Specifically, this involves combinations of the following five attributes: instrument type, pitch, duration, velocity, and time signature. The total number of potential combinations is the product of the ranges of these attributes, calculated as 24 (onset) \* 24 (duration) \* 4 (velocity bin) \* 128 (pitch) \* 13 (reduced instrument), resulting in an immense number of possible notes. Given this, it is more feasible to focus on notes that are actually observed in the dataset, as they represent a more manageable subset of possibilities.

2) Infrequency of Certain Combinations: Many attribute combinations are highly unlikely or do not occur frequently in real-world compositions. For instance, specific instruments such as the violin rarely play pitches lower than G3 (pitch number 55). Therefore, it is unnecessary to consider these improbable combinations, further reducing the computational complexity by excluding unrealistic or rare note occurrences. This selective approach helps in focusing on the most relevant data.

## D.2 BAR-LEVEL FEATURE EXTRACTION

The four attributes of a bar mentioned in this paper are chord, time signature, dynamics, and tonality. These attributes are extracted from a bar, given the five-attribute notes contained within it.

**Chord.** We adopt the chord calculation algorithm from the REMI representation. A chord is defined by three attributes: root, chroma, and bass. These can be extracted using the library `mir_eval` by analyzing the chord name. Given the sequence of note pitches in a bar, a similarity metric is computed between this pitch sequence and all possible chord chromas. The chroma with the highest similarity is selected as the most likely chroma for the current bar, and the corresponding chord is then determined. This method ensures the chord assigned to a bar accurately reflects the harmonic content of that segment.

**Time Signature.** In the MIDI standard, the time signature is embedded in the program change channel, where multiple time signatures can exist within a single piece of music. To determine the time signature of a specific bar, we record the index of each bar and its corresponding program change. The time signature of a bar is assigned based on the nearest preceding program change. This method ensures that each bar’s time signature is correctly aligned with the overall structure of the piece.

**Dynamics.** We categorize dynamics into six levels: `pp`, `p`, `mp`, `mf`, `f`, and `ff`, which correspond to very soft to very loud dynamic ranges. To determine the dynamics of a bar, we extract the velocity and pitch information from the note sequence in the bar. The average velocity of all notes in the bar is computed, and based on this value, we map the velocity to one of the six dynamic levels. For instance, very low velocities map to `pp`, while very high velocities map to `ff`. This approach provides a clear dynamic profile for each bar, reflecting the performance intensity.

**Tonality.** We adopt a simple binary classification of tonality: major and minor. Major tonality often expresses emotions such as happiness, brightness, and triumph, while minor tonality conveys sadness, tension, or somberness. In this paper, tonality is derived together with the chord extraction process. A chord label, such as ‘C:maj7’ or ‘D:min7’, directly indicates whether the tonality is major or minor. The label ‘maj’ (major) or ‘min’ (minor) is used to classify the tonality of the bar. This binary tonality annotation simplifies the tonal analysis of the music while still capturing essential emotional qualities.

## D.3 HANDLING NOTE DURATION ACROSS BARS

In our approach, each bar is subdivided into 24 bins to represent discrete time intervals within the bar. However, a challenge arises when dealing with notes that extend beyond a single bar. Specifically, a note may begin in one bar but continue into subsequent bars, resulting in a duration that spans multiple bars.

To address this, we segment the duration of such notes into discrete parts. Each segment corresponds to a note that starts within a specific bar and, in most cases, extends to the end of that bar. The remaining portion of the note continues into the next bar in a similar manner. By this method, we ensure that each bar is represented by a set of notes with a maximum duration of 24 bins, effectively standardizing the duration representation within each bar.

This approach allows for seamless integration of note durations across multiple bars while maintaining the temporal structure imposed by the 24-bin division. It also facilitates the uniform handling of note events across bars, enabling consistent analysis and processing of musical sequences.

## E METRICS DETAILS

We evaluate performance using the note-level F-measure ( $F$ ), where a note is considered correctly identified if its pitch is within a quarter tone, the onset is within 50 ms, and the offset is within 20% of the note’s duration.

**Chord Accuracy (CA).** Chord Accuracy assesses whether the chords of the generated tracks align with the conditional chord sequence, which directly impacts the harmony of the generated music.

Chord accuracy is defined as:

$$CA = \frac{1}{N_{\text{tracks}} \times N_{\text{chords}}} \sum_{i=1}^{N_{\text{tracks}}} \sum_{j=1}^{N_{\text{chords}}} \mathbb{I}\{C_{i,j} = \hat{C}_{i,j}\}$$

where  $N_{\text{tracks}}$  represents the number of tracks,  $N_{\text{chords}}$  the number of bars,  $C_{i,j}$  denotes the  $j$ -th ground-truth chord in the  $i$ -th track, and  $\hat{C}_{i,j}$  refers to the corresponding generated chord in the same position.

**Pitch, Velocity, Duration, Onset Interval.** To provide a more comprehensive evaluation of the harmony, dynamics, and expressiveness of musical compositions, we analyze the distributions of various features (e.g., pitch and velocity) and compare the distances between the distributions of generated and ground-truth musical pieces. First, histograms are computed for each feature, followed by kernel density estimation to convert the histograms into continuous probability density functions. This smoothing process offers a more generalizable representation of the data.

**Averaging Overlapped Area (OA).** The overlapping area (OA) of distributions ( $\mathcal{D}_{\mathcal{A}}$ , where  $\mathcal{A}$  can be one of Pitch, Velocity, Duration, or Onset Interval) is used to quantify the difference between the generated and ground-truth musical pieces. This is represented by the following formula:

$$\mathcal{D}_{\mathcal{A}} = \frac{1}{N_{\text{tracks}} \times N_{\text{bars}}} \sum_{i=1}^{N_{\text{tracks}}} \sum_{j=1}^{N_{\text{bars}}} \text{OA}(\mathcal{P}_{i,j}^{\mathcal{A}}, \hat{\mathcal{P}}_{i,j}^{\mathcal{A}})$$

where OA denotes the averaged overlapping area between two distributions.  $\mathcal{P}_{i,j}^{\mathcal{A}}$  represents the distribution of feature  $\mathcal{A}$  in the  $i$ -th track and  $j$ -th bar of the ground-truth musical piece, while  $\hat{\mathcal{P}}_{i,j}^{\mathcal{A}}$  represents the same in the generated piece.

## F MORE SHOWCASES AND DISCUSSIONS

### F.1 FORMULATION OF THE CASES

We denote MIDI music as images, where the height of the image represents the pitch of each note, and the length corresponds to the time. In this representation, each note is mapped to a specific position in the image based on its pitch and onset time. The color of each pixel represents instrument, providing a visual representation of the musical structure. More showcases are on <https://sites.google.com/view/unicomposer>

### F.2 BENEFITS OF USING BARS AS THE BASIC UNIT

Since UniComposer uses bars as the fundamental unit for music generation, the resulting compositions are strongly aligned by bars, which helps to maintain a coherent and well-structured musical form. This bar-level alignment ensures that the generated music follows a clear rhythmic structure, improving both the musicality and organization of the output.

### F.3 BENEFITS OF INSTRUMENT SEPARATION

UniComposer distinguishes between monophonic, polyphonic, and percussion instruments, allowing for a more refined focus on their individual roles within band-level music compositions. Monophonic instruments primarily contribute to melodic variation and dynamic changes, playing a central role in driving the musical narrative. Polyphonic instruments provide harmonic support and stability, often through repetitive chord progressions or sustained notes that enhance the overall harmonic texture. Percussion instruments, on the other hand, serve as the rhythmic foundation, dictating the tempo and guiding the progression of the music. This separation of instrumental roles contributes to a clearer structural organization in band-level compositions, ensuring that each instrument type complements the others in a cohesive and balanced manner.

