
Efficient Neural Music Generation

Max W. Y. Lam, Qiao Tian, Tang Li, Zongyu Yin, Siyuan Feng, Ming Tu, Yuliang Ji,
Rui Xia, Mingbo Ma, Xuchen Song, Jitong Chen, Yuping Wang, Yuxuan Wang
Speech, Audio & Music Intelligence (SAMI), ByteDance

Abstract

Recent progress in music generation has been remarkably advanced by the state-of-the-art MusicLM, which comprises a hierarchy of three LMs, respectively, for semantic, coarse acoustic, and fine acoustic modelings. Yet, sampling with the MusicLM requires processing through these LMs one by one to obtain the fine-grained acoustic tokens, making it computationally expensive and prohibitive for a real-time generation. Efficient music generation with a quality on par with MusicLM remains a significant challenge. In this paper, we present **MeLoDy** (**M** for music; **L** for LM; **D** for diffusion), an LM-guided diffusion model that generates music audios of state-of-the-art quality meanwhile reducing 95.7% or 99.6% forward passes in MusicLM, respectively, for sampling 10s or 30s music. MeLoDy inherits the highest-level LM from MusicLM for semantic modeling, and applies a novel dual-path diffusion (DPD) model and an audio VAE-GAN to efficiently decode the conditioning semantic tokens into waveform. DPD is proposed to simultaneously model the coarse and fine acoustics by incorporating the semantic information into segments of latents effectively via cross-attention at each denoising step. Our experimental results suggest the superiority of MeLoDy, not only in its practical advantages on sampling speed and infinitely continuable generation, but also in its state-of-the-art musicality, audio quality, and text correlation.

Our samples are available at <https://Efficient-MeLoDy.github.io/>.

1 Introduction

Music is an art composed of harmony, melody, and rhythm that permeates every aspect of human life. With the blossoming of deep generative models [1–3], music generation has drawn much attention in recent years [4–6]. As a prominent class of generative models, language models (LMs) [7, 8] showed extraordinary modeling capability in modeling complex relationships across long-term contexts [9–11]. In light of this, AudioLM [3] and many follow-up works [5, 12–14] successfully applied LMs to audio synthesis. Concurrent to the LM-based approaches, diffusion probabilistic models (DPMs) [1, 15, 16], as another competitive class of generative models [2, 17], have also demonstrated exceptional abilities in synthesizing speech [18–20], sounds [21, 22] and music [6, 23].

However, generating music from free-form text remains challenging as the permissible music descriptions can be very diverse and relate to any of the genres, instruments, tempo, scenarios, or even some subjective feelings. Conventional text-to-music generation models are listed in Table 1, where both MusicLM [5] and Noise2Music [6] were trained on large-scale music datasets and demonstrated the state-of-the-art (SOTA) generative performances with high fidelity and adherence to various aspects of text prompts. Yet, the success of these two methods comes with large computational costs, which would be a serious impediment to their practicalities. In comparison, Moûsai [23] building upon DPMs made efficient samplings of high-quality music possible. Nevertheless, the number of their demonstrated cases was comparatively small and showed limited in-sample dynamics. Aiming for a feasible music creation tool, a high efficiency of the generative model is essential since it facilitates interactive creation with human feedback being taken into account as in [24].

Table 1: A comparison of MeLoDy with conventional text-to-music generation models in the literature. We use **AC** to denote whether audio continuation is supported, **FR** to denote whether the sampling is faster than real-time on a V100 GPU, **VT** to denote whether the model has been tested and demonstrated using various types of text prompts including instruments, genres, and long-form rich descriptions, and **MP** to denote whether the evaluation was done by music producers.

Model	Prompts	Training Data	AC	FR	VT	MP
Moúsai [23]	Text	2.5k hours of music	✓	✓	✗	✗
MusicLM [5]	Text, Melody	280k hours of music	✓	✗	✓	✗
Noise2Music [6]	Text	340k hours of music	✗	✗	✓	✗
MeLoDy (Ours)	Text, Audio	257k hours of music ¹	✓	✓	✓	✓

While LMs and DPMs both showed promising results, we believe the relevant question is not whether one should be preferred over another but whether we can leverage both approaches with respect to their individual advantages, e.g., [25]. After analyzing the success of MusicLM, we leverage the highest-level LM in MusicLM, termed as *semantic LM*, to model the semantic structure of music, determining the overall arrangement of melody, rhythm, dynamics, timbre, and tempo. Conditional on this semantic LM, we exploit the non-autoregressive nature of DPMs to model the acoustics efficiently and effectively with the help of a successful sampling acceleration technique [26]. All in all, in this paper, we introduce several novelties that constitute our main contributions:

1. We present **MeLoDy** (**M** for music; **L** for LM; **D** for diffusion), an LM-guided diffusion model that generates music of competitive quality while reducing 95.7% and 99.6% iterations of MusicLM to sample 10s and 30s music, being faster than real-time on a V100 GPU.
2. We propose the novel dual-path diffusion (DPD) models to efficiently model coarse and fine acoustic information simultaneously with a particular semantic conditioning strategy.
3. We design an effective sampling scheme for DPD, which improves the generation quality over the previous sampling method in [23] proposed for this class of LDMs.
4. We reveal a successful audio VAE-GAN that effectively learns continuous latent representations, and is capable of synthesizing audios of competitive quality together with DPD.

2 Related Work

Audio Generation Apart from the generation models shown in Table 1, there are also music generation models [28, 29] that can generate high-quality music samples at high speed, yet they cannot accept free-form text conditions and can only be trained to specialize in single-genre music, e.g., techno music in [29]. There also are some successful music generators in the industry, e.g. Mubert [30] and Riffusion [31], yet, as analyzed in [5], they struggled to compete with MusicLM in handling free-form text prompts. In a more general scope of audio synthesis, some promising text-to-audio synthesizers [12, 21, 22] trained with AudioSet [32] also demonstrated to be able to generate music from free-form text, but the musicality is limited. AudioLM [3] unconditionally continued piano audios with promising fidelity. Parallel to this work, SoundStorm [33] exceedingly accelerated the AudioLM with a non-autoregressive decoding scheme [34], such that the acoustic LM can be decoded in 27 forward passes. In comparison, neglecting the individual cost of networks, MeLoDy takes 5 to 20 forward passes to generate acoustics of high fidelity, as discussed in Section 5.

Network Architecture The architecture designed for our proposed DPD was inspired by the dual-path networks used in the context of audio separation, where Luo et al. [35] initiated the idea of segmentation-based dual-path processing, and triggered a number of follow-up works achieving the state-of-the-art results [36–40]. Noticing that the objective in diffusion models indeed can be viewed as a special case of source separation, this kind of dual-path architecture effectually provides us a basis for simultaneous coarse-and-fine acoustic modeling.

¹We focus on non-vocal music data by using an audio classifier [27] to filter out in-house music data with vocals. Noticeably, generating vocals and instrumental music simultaneously in one model is defective even in the SOTA works [5, 6] because of the unnaturally sound vocals. While this work aims for generating production-level music, we improve the fidelity by reducing the tendency of generating vocals.

3 Background on Audio Language Modeling

This section provides the preliminaries that serve as the basis for our model. In particular, we briefly describe the audio language modeling framework used in MusicLM.

3.1 Audio Language Modeling with MusicLM

MusicLM [5] mainly follows the audio language modeling framework presented in AudioLM [3], where audio synthesis is viewed as a language modeling task over a hierarchy of coarse-to-fine audio tokens. In AudioLM, there are two kinds of tokenization for representing different scopes of audio:

- **Semantic Tokenization:** K-means over representations from SSL, e.g., w2v-BERT [41];
- **Acoustic Tokenization:** Neural audio codec, e.g., SoundStream [42].

To better handle the hierarchical structure of the acoustic tokens, AudioLM further separates the modeling of acoustic tokens into coarse and fine stages. In total, AudioLM defines three LM tasks: (1) semantic modeling, (2) coarse acoustic modeling, and (3) fine acoustic modeling.

We generally define the sequence of conditioning tokens as $\mathbf{c}_{1:T_{\text{end}}} := [\mathbf{c}_1, \dots, \mathbf{c}_{T_{\text{end}}}]$ and the sequence of target tokens as $\mathbf{u}_{1:T_{\text{tgt}}} := [\mathbf{u}_1, \dots, \mathbf{u}_{T_{\text{tgt}}}]$. In each modeling task, a Transformer-decoder language model parameterized by θ is tasked to solve the following autoregressive modeling problem:

$$p_{\theta}(\mathbf{u}_{1:T_{\text{tgt}}} | \mathbf{c}_{1:T_{\text{end}}}) = \prod_{j=1}^{T_{\text{tgt}}} p_{\theta}(\mathbf{u}_j | [\mathbf{c}_1, \dots, \mathbf{c}_{T_{\text{end}}}, \mathbf{u}_1, \dots, \mathbf{u}_{j-1}]), \quad (1)$$

where the conditioning tokens are concatenated to the target tokens as prefixes. In AudioLM, semantic modeling takes no condition; coarse acoustic modeling takes the semantic tokens as conditions; fine acoustic modeling takes the coarse acoustic tokens as conditions. The three corresponding LMs can be trained in parallel with the ground-truth tokens, but need to be sampled sequentially for inference.

3.1.1 Joint Tokenization of Music and Text with MuLan and RVQ

To maintain the merit of audio-only training, MusicLM relies on MuLan [43], which is a two-tower, joint audio-text embedding model that can be individually trained with large-scale music data and weakly-associated, free-form text annotations. The MuLan model is pre-trained to project the music audio and its corresponding text description into the same embedding space such that the associated embeddings can be close to each other. In MusicLM, the MuLan embeddings of music and text are tokenized using a separately learned residual vector quantization (RVQ) [42].

Different from AudioLM, MusicLM employs the MuLan tokens as the additional prefixing tokens, as in Eq. (1), for the semantic modeling and the coarse acoustic modeling. During training, the audio is first fed to the MuLan music tower to obtain the music embedding. Then, an RVQ is applied to the music embedding, resulting in the ground-truth MuLan tokens for conditioning the semantic LM and the coarse acoustic LM. To generate music from a text prompt, the text embedding obtained from the MuLan text tower is passed to the same RVQ and is discretized into the inference-time MuLan tokens. Based on the prefixing MuLan tokens, the semantic tokens, coarse acoustic tokens, and fine acoustic tokens are subsequently computed to generate high-fidelity music audio adhering to the text prompt.

4 Model Description

The overall training and sampling pipelines of MeLoDy are shown in Figure 1, where, we have three modules for representation learning: (1) MuLan, (2) Wav2Vec2-Conformer, and (3) audio VAE, and two generative models: a language model (LM) and a dual-path diffusion (DPD) model, respectively, for semantic modeling and acoustic modeling. In the same spirit as MusicLM, we leverage LM to model the semantic structure of music for its promising capability of modeling complex relationships across long-term contexts [9–11]. Similar to MusicLM, we pre-train a MuLan model to obtain the conditioning tokens. For semantic tokenization, we opt to use the Wav2Vec2-Conformer model, which follows the same architecture as Wav2Vec2 [44] but employs the Conformer blocks [45] in place of the Transformer blocks. The remainder of this section presents our newly proposed DPD model and the audio VAE-GAN used for DPD model, while other modules overlapped with MusicLM are described in Appendix B regarding the training and implementation details.

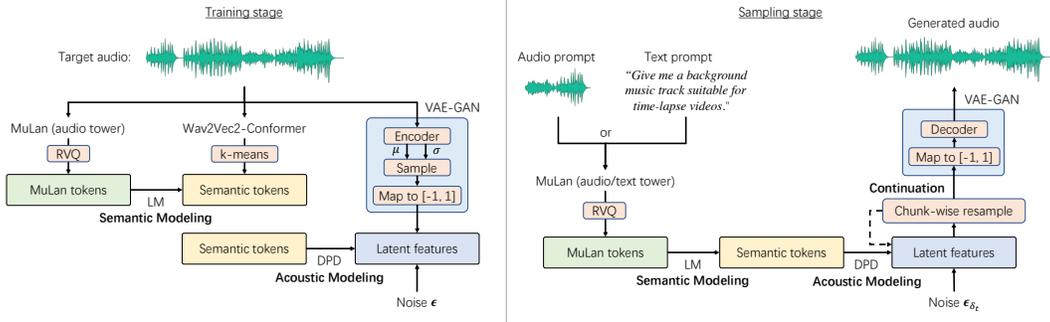


Figure 1: The training and sampling pipelines of MeLoDy

4.1 Dual-Dath Diffusion: Angle-Parameterized Continuous-Time Latent Diffusion Models

The proposed dual-path diffusion (DPD) model is a variant of diffusion probabilistic models (DPMs) [1, 15, 46] in continuous-time [16, 47–49]. Instead of directly operating on the raw data $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$, with reference to the latent diffusion models (LDMs) [2], we consider a low-dimensional latent representation $\mathbf{z} = \mathcal{E}_\phi(\mathbf{x})$, where ϕ is a pre-trained autoencoder that enables reconstruction of the raw data from the latent: $\mathbf{x} \approx \mathcal{D}_\phi(\mathbf{z})$. Here, we use \mathcal{E}_ϕ to denote the encoder, and \mathcal{D}_ϕ to denote the decoder. By working on a low-dimensional latent space, the computational burden of DPMs can be significantly relieved [2]. We present our audio autoencoder in Section 4.2, which is tailored for DPMs and performed the stablest in our experiments.

In DPD, we consider a Gaussian diffusion process \mathbf{z}_t that is fully specified by two strictly positive scalar-valued, continuously differentiable functions α_t, σ_t [16]: $q(\mathbf{z}_t | \mathbf{z}) = \mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{z}, \sigma_t^2 \mathbf{I})$ for any $t \in [0, 1]$. In the light of [48], we define $\alpha_t := \cos(\pi t/2)$ and $\sigma_t := \sin(\pi t/2)$ to benefit from some nice trigonometric properties, i.e., $\sigma_t = \sqrt{1 - \alpha_t^2}$ (a.k.a. variance-preserving [16]). By this definition, \mathbf{z}_t can be elegantly re-parameterized in terms of angles δ :

$$\mathbf{z}_\delta = \cos(\delta)\mathbf{z} + \sin(\delta)\boldsymbol{\epsilon} \quad \text{for any } \delta \in [0, \pi/2], \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (2)$$

Note that \mathbf{z}_δ gets noisier as δ increases from 0 to $\pi/2$, which defines the forward diffusion process.

To generate samples, we use a θ -parameterized variational model $p_\theta(\mathbf{z}_{\delta-\omega} | \mathbf{z}_\delta)$ to invert the diffusion process by enabling running backward in angle with $0 < \omega \leq \delta$. Based on this model, we can sample \mathbf{z} from $\mathbf{z}_{\pi/2} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with T sampling steps, by discretizing $\pi/2$ into T segments as follows:

$$p_\theta(\mathbf{z} | \mathbf{z}_{\pi/2}) = \int_{\mathbf{z}_{\delta_1, T-1}} \prod_{t=1}^T p_\theta(\mathbf{z}_{\delta_t - \omega_t} | \mathbf{z}_{\delta_t}) d\mathbf{z}_{\delta_1, T-1}, \quad \delta_t = \begin{cases} \frac{\pi}{2} - \sum_{i=t+1}^T \omega_i, & 1 \leq t < T; \\ \frac{\pi}{2}, & t = T, \end{cases} \quad (3)$$

where the *angle schedule*, denoted by $\omega_1, \dots, \omega_T$, satisfies $\sum_{t=1}^T \omega_t = \pi/2$. Schneider et al. [23] proposed a uniform angle schedule: $\omega_t = \frac{\pi}{2T}$ for all t . As revealed in previous scheduling methods [50, 51] for DPMs, taking larger steps at the beginning of the sampling followed by smaller steps could improve the quality of samples. Following this strategy, we design a new linear angle schedule, which empirically gives more stable and higher-quality results, and is written as

$$\omega_t = \frac{\pi}{6T} + \frac{2\pi t}{3T(T+1)}. \quad (4)$$

We extensively compare this linear angle schedule with the uniform one in [23] in Appendix D.

4.1.1 Multi-Chunk Velocity Prediction for Long-Context Generation

For model training, similar to the setting in [23] for long-context generation, the neural network is tasked to predict a multi-chunk target \mathbf{v}_{tgt} that comprises M chunks of velocities, each having a different noise scale. Formally speaking, given that $\mathbf{z}, \mathbf{z}_\delta, \boldsymbol{\epsilon} \in \mathbb{R}^{L \times D}$ with L representing the length of audio latents and D representing the latent dimensions, we define $\mathbf{v}_{\text{tgt}} := \mathbf{v}_1 \oplus \dots \oplus \mathbf{v}_M$, where

$$\mathbf{v}_m := \cos(\delta_m)\boldsymbol{\epsilon}[L_{m-1} : L_m, :] - \sin(\delta_m)\mathbf{z}[L_{m-1} : L_m, :], \quad L_m := \left\lfloor \frac{mL}{M} \right\rfloor. \quad (5)$$

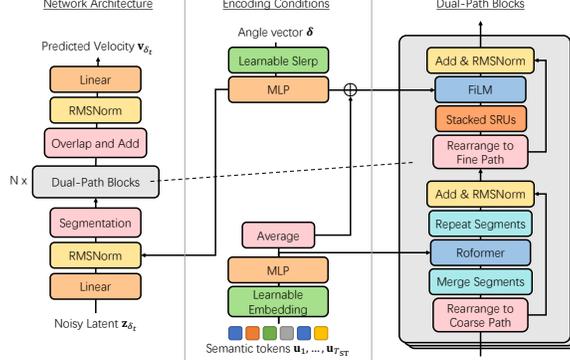


Figure 2: The proposed dual-path diffusion (DPD) model

Here, we use the NumPy slicing syntax (0 as the first index) to locate the m -th chunk, and we draw $\delta_m \sim \text{Uniform}[0, \pi/2]$ for each chunk at each training step to determine the noise scale. To learn θ , we use the mean squared error (MSE) loss in [1, 48]:

$$\mathcal{L}_{\text{diff}} := \mathbb{E}_{\mathbf{z}, \epsilon, \delta_1, \dots, \delta_M} \left[\|\mathbf{v}_{\text{tgt}} - \hat{\mathbf{v}}_{\theta}(\mathbf{z}_{\text{noisy}}; \mathbf{c})\|_2^2 \right], \quad (6)$$

$$\mathbf{z}_{\text{noisy}} := \cos(\delta_m) \mathbf{z}[L_{m-1} : L_m, :] + \sin(\delta_m) \epsilon[L_{m-1} : L_m, :], \quad (7)$$

where \mathbf{c} generally denotes the collection of conditions used for the velocity prediction. In MeLoDy, as illustrated in Figure 1, we propose to use the semantic tokens $\mathbf{u}_1, \dots, \mathbf{u}_{T_{\text{ST}}}$, which are obtained from the SSL model during training and generated by the LM at inference time, to condition the DPD model. In our experiments, we find that the stability of generation can be significantly improved if we use token-based discrete conditions to control the semantics of the music and let the diffusion model learn the embedding vector for each token itself. Additionally, to assist the multi-chunk prediction, we append an angle vector to the condition that represents the angles drawn in the M chunks:

$$\mathbf{c} := \{\mathbf{u}_1, \dots, \mathbf{u}_{T_{\text{ST}}}, \boldsymbol{\delta}\}, \quad \boldsymbol{\delta} := [\delta_1]_{r=1}^{L_1} \oplus \dots \oplus [\delta_M]_{r=1}^{L_M} \in \mathbb{R}^L \quad (8)$$

where $[a]_{r=1}^B$ denotes the operation of repeating a scalar a for B times to make a B -length vector. Suppose we have a well-trained velocity model, for sampling, we apply the trigonometric identities to the DDIM sampling algorithm [26] (see Appendix A) and obtain a simplified update rule:

$$\mathbf{z}_{\delta_t - \omega_t} = \cos(\omega_t) \mathbf{z}_{\delta_t} - \sin(\omega_t) \hat{\mathbf{v}}_{\theta}(\mathbf{z}_{\delta_t}; \mathbf{c}), \quad (9)$$

by which, using the angle schedule in Eq. (4) and running from $t = T$ to $t = 1$, we get a sample of \mathbf{z} .

4.1.2 Dual-Path Modeling for Efficient and Effective Velocity Prediction

Next, we present how $\hat{\mathbf{v}}_{\theta}$ takes in the noisy latent and the conditions and efficiently incorporates the semantic tokens into the coarse processing path for effective velocity prediction. As a highlight of this work, we modify the dual-path technique borrowed from audio separation [35, 37, 38], and propose a novel architecture for efficient, simultaneous coarse and fine acoustic modeling, as shown in Figure 2. This architecture comprises several critical modules, which we present one by one below.

To begin with, we describe how the conditions are processed in DPD (the middle part in Figure 2):

Encoding Angle Vector First, we encode $\boldsymbol{\delta} \in \mathbb{R}^L$, which records the frame-level noise scales of latents. Instead of using the classical positional encoding [1], we use a Slerp-alike spherical interpolation [52] to two learnable vectors $\mathbf{e}_{\text{start}}, \mathbf{e}_{\text{end}} \in \mathbb{R}^{256}$ based on broadcast multiplications \otimes :

$$\mathbf{E}_{\boldsymbol{\delta}} := \text{MLP}(\sin(\boldsymbol{\delta}) \otimes \mathbf{e}_{\text{start}} + \sin(\boldsymbol{\delta}) \otimes \mathbf{e}_{\text{end}}) \in \mathbb{R}^{L \times D_{\text{hid}}}, \quad (10)$$

where $\text{MLP}(\mathbf{x}) := \text{RMSNorm}(\mathbf{W}_2 \text{GELU}(\mathbf{x} \mathbf{W}_1 + \mathbf{b}_1) + \mathbf{b}_2)$ projects an arbitrary input $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$ to $\mathbb{R}^{D_{\text{hid}}}$ using RMSNorm [53] and GELU activation [54]. Here, D_{hid} is hidden dimension defined for the model, and $\mathbf{W}_1 \in \mathbb{R}^{D_{\text{in}} \times D_{\text{hid}}}$, $\mathbf{W}_2 \in \mathbb{R}^{D_{\text{hid}} \times D_{\text{hid}}}$, $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^{D_{\text{hid}}}$ are the learnable parameters.

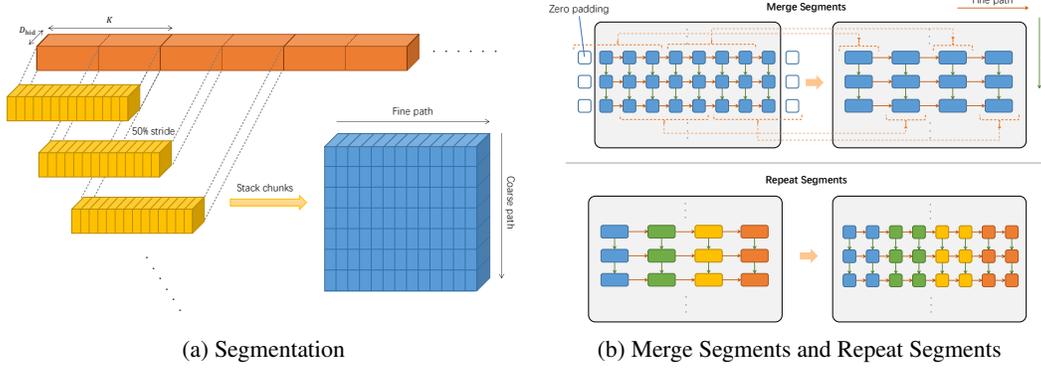


Figure 3: Diagrams for visually understanding the operations over the 3-D segments

Encoding Semantic Tokens The remaining conditions are the discrete tokens representing semantic information $\mathbf{u}_1, \dots, \mathbf{u}_{T_{\text{ST}}}$. Following the typical approach for embedding natural languages [8], we directly use a lookup table of vectors to map any token $\mathbf{u}_t \in \{1, \dots, V_{\text{ST}}\}$ into a real-valued vector $E(\mathbf{u}_t) \in \mathbb{R}^{D_{\text{hid}}}$, where V_{ST} denotes the vocabulary size of the semantic tokens, i.e., the number of clusters in k-means for Wav2Vec2-Conformer. By stacking the vectors along the time axis and applying an MLP block, we obtain $\mathbf{E}_{\text{ST}} := \text{MLP}([E(\mathbf{u}_1), \dots, E(\mathbf{u}_{T_{\text{ST}}})]) \in \mathbb{R}^{T_{\text{ST}} \times D_{\text{hid}}}$.

Next, we show how the network input (i.e., $\mathbf{z}_{\text{noisy}}$ at training time, or \mathbf{z}_{δ_t} at inference time) is processed given the condition embeddings. We use $\mathbf{z}_{\text{noisy}}$ as input for our explanation below, since \mathbf{z}_{δ_t} is only its special case with all chunks having the same noise scale. The input $\mathbf{z}_{\text{noisy}}$ is first linearly transformed and added up with the angle embedding of the same shape: $\mathbf{H} := \text{RMSNorm}(\mathbf{z}_{\text{noisy}} \mathbf{W}_{\text{in}} + \mathbf{E}_{\delta})$, where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{D \times D_{\text{hid}}}$ is learnable. We then perform segmentation for dual-path processing.

Segmentation As shown in Figure 3a, the segmentation module divides a 2-D input into S half-overlapping segments each of length K , represented by a 3-D tensor $\mathbb{H} := [\mathbf{0}, \mathbf{H}_1, \dots, \mathbf{H}_S, \mathbf{0}] \in \mathbb{R}^{S \times K \times D_{\text{hid}}}$, where $\mathbf{H}_s := \mathbf{H} \left[\frac{(s-1)K}{2} : \frac{(s-1)K}{2} + K, : \right]$, and \mathbb{H} is zero-padded such that we have $S = \lceil \frac{2L}{K} \rceil + 1$. With a segment size $K \approx \sqrt{L}$, the length for sequence processing becomes sub-linear ($\mathcal{O}(\sqrt{L})$) as opposed to tackling the whole sequence ($\mathcal{O}(L)$). This greatly reduces the difficulty of learning a very long sequence and permits MeLoDy to use higher-frequency latents.

Dual-Path Blocks After the segmentation, we obtain a 3-D tensor input for N dual-path blocks, each block exhibits an architecture shown on the rightmost of Figure 2. The input to the i -th dual-path block is denoted as $\mathbb{H}^{(i)}$, and we have $\mathbb{H}^{(1)} := \mathbb{H}$. Each block contains two stages corresponding to coarse-path (i.e., inter-segment) and fine-path (i.e., intra-segment) processing, respectively. Similar to the observations in [37, 38], we find it superior to use an attention-based network for coarse-path processing and to use a bi-directional RNN for fine-path processing. The goal of fine acoustic modeling is to better reconstruct the fine details from the roughly determined audio structure [3]. At a finer scope, only the nearby elements matter and contain the most information needed for refinement, as supported by the modeling perspectives in neural vocoding [55, 56]. Specifically, we employ the Roformer network [57] for coarse-path processing, where we use a self-attention layer followed by a cross-attention layer to be conditional on \mathbf{E}_{ST} with rotary positional embedding. On the other hand, we use a stack of 2-layer simple recurrent units (SRUs) [58] for fine-path processing. The feature-wise linear modulation (FiLM) [59] is applied to the output of SRUs to assist the denoising with the angle embedding \mathbf{E}_{δ} and the pooled \mathbf{E}_{ST} . Each of these processing stages is detailed below.

Coarse-Path Processing In a dual-path block, we first process the coarse path corresponding to the vertical axis shown in Figure 3a, in which the columns are processed in parallel:

$$\mathbb{H}_{\text{c-out}}^{(i)} := \text{RepeatSegments} \left(\left[\text{Roformer} \left(\text{MergeSegments} \left(\mathbb{H}^{(i)} \right) [:, k, :] \right), k = 0, \dots, K_{\text{MS}}^{(i)} - 1 \right] \right), \quad (11)$$

where the coarse-path output $\mathbb{H}_{\text{c-out}}^{(i)} \in \mathbb{R}^{S \times K \times D_{\text{hid}}}$ has the same shape as $\mathbb{H}^{(i)}$, and $\text{MergeSegments}(\cdot)$ and $\text{RepeatSegments}(\cdot)$ are the operations that, respectively, compress and expand the segments

horizontally to aggregate the information within a segment for a coarser scale of inter-segment processing. Note that, without taking the merging and repeating operations, the vertical axis is simply a sequence formed by skipping $K/2$ elements in \mathbf{H} , which does not really capture the desired coarse information. The merging is done by averaging every pair of $2^{\min\{i, N-i+1\}}$ columns with zero paddings and a half stride such that $K_{\text{MS}}^{(i)} = \lceil \frac{K}{2^{\min\{i, N-i+1\}-1}} \rceil$. The upper part of Figure 3b illustrates the case of $i = 2$. Similar to [38], our definition of $K_{\text{MS}}^{(i)}$ changes the width of the 3d tensor with the block index i in a sandglass style, as we have the shortest segment at the middle block and the longest segment at the first and the last block. To match with the original length, a repeating operation following from the Roformer is performed, as shown in the lower part of Figure 3b.

Fine-Path Processing We then obtain the fine-path input: $\mathbb{H}_{\text{f-in}}^{(i)} := \text{RMSNorm} \left(\mathbb{H}^{(i)} + \mathbb{H}_{\text{c-out}}^{(i)} \right)$, which is fed to a two-layer SRU by parallelly processing the rows illustrated in Figure 3a:

$$\mathbb{H}_{\text{f-out}}^{(i)} := \left[\text{FiLM} \left(\text{SRU} \left(\mathbb{H}_{\text{f-in}}^{(i)}[s, :, :] \right), \mathbf{E}_\delta \left[\frac{sL}{S}, : \right] + \frac{1}{T_{\text{ST}}} \sum_{t=0}^{T_{\text{ST}}-1} \mathbf{E}_{\text{ST}}[t, :] \right), s = 0, \dots, S-1 \right], \quad (12)$$

where $\text{FiLM}(\mathbf{x}, \mathbf{m}) := \text{MLP}_3((\mathbf{x} \otimes \text{MLP}_1(\mathbf{m})) + \text{MLP}_2(\mathbf{m}))$ for an arbitrary input \mathbf{x} and modulation condition \mathbf{m} , and \otimes is the operations of broadcast multiplication. Followed from this, we have the input for the next dual-path block: $\mathbb{H}^{(i+1)} := \text{RMSNorm} \left(\mathbb{H}_{\text{f-in}}^{(i)} + \mathbb{H}_{\text{f-out}}^{(i)} \right)$. After recursively processing through N dual-path blocks, the 3-D tensor is transformed back to a 2-D matrix using an overlap-and-add method [35]. Finally, the predicted velocity is obtained as follows:

$$\hat{\mathbf{v}}_\theta(\mathbf{z}_{\text{noisy}}; \mathbf{c}) := \text{RMSNorm} \left(\text{OverlapAdd} \left(\mathbb{H}^{(N+1)} \right) \right) \mathbf{W}_{\text{out}}, \quad (13)$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D_{\text{hid}} \times D}$ is learnable. We present more details of our implementation in Appendix B.

4.2 Audio VAE-GANs for Latent Representation Learning

To avoid learning arbitrarily high-variance latent representations, Rombach et al. [2] examined a KL-regularized image autoencoder for latent diffusion models (LDMs) and demonstrated extraordinary stability in generating high-quality image [60], igniting a series of follow-up works [61]. Such an autoencoder imposes a KL penalty on the encoder outputs in a way similar to VAEs [62, 63], but, different from the classical VAEs, it is adversarially trained as in the generative adversarial networks (GANs) [64]. In this paper, this class of autoencoders is referred to as the VAE-GAN. Although VAE-GANs are promisingly applied to image generation, there is still a lack of comparable successful methods for the autoencoding of audio waveforms. In this work, we propose a similarly trained audio VAE-GAN, which empirically showed remarkable stability when applied to our DPD model in comparison to other commonly used VQ-VAE used in [12, 21, 65].

Specifically, the audio VAE-GAN is trained to reconstruct 24kHz audio with a striding factor of 96, resulting in a 250Hz latent sequence. The architecture of the decoder is the same as that in HiFi-GAN [66]. For the encoder, we basically replace the up-sampling modules in the decoder with convolution-based down-sampling modules while other modules stay the same. For adversarial training, we use the multi-period discriminators in [66] and the multi-resolution spectrogram discriminators in [67]. The training details are further discussed in Appendix B. To match the normal range of targets for diffusion models [1, 2], we map the encoder outputs to $[-1, 1]$ by $\mathbf{z}_{(i,j)} := \min \left\{ \max \left\{ \bar{\mathbf{z}}_{(i,j)}/3, -1 \right\}, 1 \right\} \forall i, j$, where the subscript (i, j) denotes the value on the i -th row and j -th column, and the choice of 3 in practice would sieve extreme values occupying $< 0.1\%$.

4.3 Music Inpainting, Music Continuation and Music Prompting with MeLoDy

We show that the proposed MeLoDy supports interpolation (i.e., audio inpainting) and extrapolation (i.e., audio continuation) with tricks of manipulating random noises. Noticeably, diffusion models have been successfully used for effective audio inpainting [21, 22]. Yet, audio continuation has been an obstacle for diffusion models due to their non-autoregressive nature. Besides audio continuation, based on MuLan, MeLoDy also supports music prompts to generate music of a similar style, as shown in Figure 1. Examples of music inpainting, music continuation, and music prompting are shown on our demo page. We present the algorithms of these functionalities in Appendix C.

Table 2: The speed and the quality of our proposed MeLoDy on a CPU (Intel Xeon Platinum 8260 CPU @ 2.40GHz) or a GPU (NVIDIA Tesla V100) using different numbers of sampling steps.

Steps (T)	Speed on CPU (\uparrow)	Speed on GPU (\uparrow)	FAD (\downarrow)	MCC (\uparrow)
(MusicCaps)	-	-	-	0.43
5	1472Hz (0.06\times)	181.1kHz (7.5\times)	7.23	0.49
10	893Hz (0.04 \times)	104.8kHz (4.4 \times)	5.93	0.52
20	498Hz (0.02 \times)	56.9kHz (2.4 \times)	5.41	0.53

5 Experiments

5.1 Experimental Setup

Data Preparation As shown in Table 1, MeLoDy was trained on 257k hours of music data (6.4M 24kHz audios), which were filtered with [27] to focus on non-vocal music. Additionally, inspired by the text augmentation in [6], we enriched the tag-based texts to generate music captions by asking ChatGPT [68]. This music description pool is used for the training of our 195.3M MuLan, where we randomly paired each audio with either the generated caption or its respective tags. In this way, we robustly improve the model’s capability of handling free-form text.

Semantic LM For semantic modeling, we trained a 429.5M LLaMA [69] with 24 layers, 8 heads, and 2048 hidden dimensions, which has a comparable number of parameters to that of the MusicLM [5]. For conditioning, we set up the MuLan RVQ using 12 1024-sized codebooks, resulting in 12 prefixing tokens. The training targets were 10s semantic tokens, which are obtained from discretizing the 25Hz embeddings from a 199.5M Wav2Vec2-Conformer with 1024-center k-means.

Dual-Path Diffusion For the DPD model, we set the hidden dimension to $D_{\text{hid}} = 768$, and block number to $N = 8$, resulting in 296.6M parameters. For the input chunking strategy, we divide the 10s training inputs in a fixed length of $L = 2500$ into $M = 4$ parts. For segmentation, we used a segment size of $K = 64$ (i.e., each segment is 256ms long), leading to $S = 80$ segments. In addition, we applied the classifier-free guidance [70] to DPD for improving the correspondence between samples and conditions. During training, the cross-attention to semantic tokens is randomly replaced by self-attention with a probability of 0.1. For sampling, the predicted velocity is linearly combined as . For all of our generations, a scale of 2.5 was used for classifier-free guidance.

Audio VAE-GAN For audio VAE-GAN, we used a hop size of 96, resulting in 250Hz latent sequences for encoding 24kHz music audio. The latent dimension $D = 16$, thus we have a total compression rate of 6 \times . The hidden channels used in the encoder were 256, whereas that used in the decoder were 768. The audio VAE-GAN in total contains 100.1M parameters.

5.2 Performance Analysis

Objective Metrics We use the VGGish-based [71] Fréchet audio distance (FAD) [72] between the generated audios and the reference audios from MusicCaps [5] as a rough measure of generation fidelity.² To measure text correlation, we use the MuLan cycle consistency (MCC) [5], which calculates the cosine similarity between text and audio embeddings using a pre-trained MuLan.³

Inference Speed We first evaluate the sampling efficiency of our proposed MeLoDy. As DPD permits using different numbers of sampling steps depending on our needs, we report its generation speed in Table 2. Surprisingly, MeLoDy steadily achieved a higher MCC score than that of the reference set, even taking only 5 sampling steps. This means that (i) the MuLan model determined that our generated samples were more correlated to MusicCaps captions than reference audios, and (ii) the proposed DPD is capable of consistently completing the MuLan cycle at significantly lower costs than the nested LMs in [5].

²Note that MeLoDy was mainly trained with non-vocal music data, its sample distribution could not fit the reference one as well as in [5, 6], since about 76% audios in MusicCaps contain either vocals or speech.

³Since our MuLan model was trained with a different dataset, our MCC results cannot be compared to [5, 6].

Table 3: The comparison of MeLoDy with the SOTA text-to-music generation models. **NFE** is the number of function evaluations [48] for generating T -second audio.⁵ **Musicality**, **Quality**, and **Text Corr.** are the winning proportions in terms of musicality, quality, and text correlation, respectively.

Model	NFE (\downarrow)	Musicality (\uparrow)		Quality (\uparrow)		Text Corr. (\uparrow)	
		MLM	N2M	MLM	N2M	MLM	N2M
MusicLM [5]	$(25 + 200 + 400)T$	0.541	-	0.465	-	0.548	-
Noise2Music [6]	$1000 + 800 + 800$	-	0.555	-	0.436	-	0.572
MeLoDy (20 steps)	$25T + 20$	0.459	0.445	0.535	0.564	0.452	0.428

Comparisons with SOTA models We evaluate the performance of MeLoDy by comparing it to MusicLM [5] and Noise2Music [6], which both were trained large-scale music datasets and demonstrated SOTA results for a wide range of text prompts. To conduct fair comparisons, we used the same text prompts in their demos (70 samples from MusicLM; 41 samples from Noise2Music),⁴ and asked seven music producers to select the best out of a pair of samples or voting for a tie (both win) in terms of musicality, audio quality, and text correlation. In total, we conducted 777 comparisons and collected 1,554 ratings. We detail the evaluation protocol in Appendix F. Table 3 shows the comparison results, where each category of ratings is separated into two columns, representing the comparison against MusicLM (MLM) or Noise2Music (N2M), respectively. Finally, MeLoDy consistently achieved comparable performances (all winning proportions fall into [0.4, 0.6]) in musicality and text correlation to MusicLM and Noise2Music. Regarding audio quality, MeLoDy outperformed MusicLM ($p < 0.05$) and Noise2Music ($p < 0.01$), where the p -values were calculated using the Wilcoxon signed-rank test. We note that, to sample 10s and 30s music, MeLoDy only takes 4.32% and 0.41% NFEs of MusicLM, and 10.4% and 29.6% NFEs of Noise2Music, respectively.

Diversity Analysis Diffusion models are distinguished for its high diversity [25]. We conduct an additional experiment to study the diversity and validity of MeLoDy’s generation given the same text prompt of open description, e.g., feelings or scenarios. The sampled results were shown on our demo page, in which we obtained samples with diverse combinations of instruments and textures.

Ablation Studies We also study the ablation on two aspects of the proposed method. In Appendix D, we compared the uniform angle schedule in [23] and the linear one proposed in DPD using the MCC metric and case-by-case qualitative analysis. It turns out that our proposed schedule tends to induce fewer acoustic issues when taking a small number of sampling steps. In Appendix E, we showed that the proposed dual-path architecture outperformed other architectures [23, 31] used for LDMs in terms of the signal-to-noise ratio (SNR) improvements using a subset of the training data.

6 Discussion

Limitation We acknowledge the limitations of our proposed MeLoDy. To prevent from having any disruption caused by unnaturally sound vocals, our training data was prepared to mostly contain non-vocal music only, which may limit the range of effective prompts for MeLoDy. Besides, the training corpus we used was unbalanced and slightly biased towards pop and classical music. Lastly, as we trained the LM and DPD on 10s segments, the dynamics of a long generation may be limited.

Broader Impact We believe our work has a huge potential to grow into a music creation tool for music producers, content creators, or even normal users to seamlessly express their creative pursuits with a low entry barrier. MeLoDy also facilitates an interactive creation process, as in Midjourney [24], to take human feedback into account. For a more precise tune of MeLoDy on a musical style, the LoRA technique [73] can be potentially applied to MeLoDy, as in Stable Diffusion [60].

⁴All samples for evaluation are available at <https://Efficient-MeLoDy.github.io/>. Note that our samples were not cherry-picked, whereas the samples we compared were cherry-picked [6], constituting very strong baselines.

⁵We use + to separate the counts for the iterative modules, i.e., LM or DPM. Suppose the cost of each module is comparable, then the time steps taken by LM and the diffusion steps taken by DPM can be fairly compared.

References

- [1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- [2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.
- [3] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audioldm: a language modeling approach to audio generation. *arXiv preprint arXiv:2209.03143*, 2022.
- [4] 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.
- [5] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.
- [6] 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.
- [7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [10] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
- [11] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [12] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.
- [13] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.
- [14] Eugene Kharitonov, Damien Vincent, Zalán Borsos, Raphaël Marinier, Sertan Girgin, Olivier Pietquin, Matt Sharifi, Marco Tagliasacchi, and Neil Zeghidour. Speak, read and prompt: High-fidelity text-to-speech with minimal supervision. *arXiv preprint arXiv:2302.03540*, 2023.
- [15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.
- [16] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

- [17] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34, 2021.
- [18] Rongjie Huang, Max WY Lam, Jun Wang, Dan Su, Dong Yu, Yi Ren, and Zhou Zhao. Fast-diff: A fast conditional diffusion model for high-quality speech synthesis. *arXiv preprint arXiv:2204.09934*, 2022.
- [19] Sungwon Kim, Heeseung Kim, and Sungroh Yoon. Guided-tts 2: A diffusion model for high-quality adaptive text-to-speech with untranscribed data. *arXiv preprint arXiv:2205.15370*, 2022.
- [20] Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. *arXiv preprint arXiv:2304.09116*, 2023.
- [21] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.
- [22] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. *arXiv preprint arXiv:2301.12661*, 2023.
- [23] Flavio Schneider, Zhijing Jin, and Bernhard Schölkopf. Moûsai: Text-to-music generation with long-context latent diffusion. *arXiv preprint arXiv:2301.11757*, 2023.
- [24] David Holz et al. Midjourney. *Artificial Intelligence platform*. Accessible at <https://www.midjourney.com/> Accessed November 1st, 2023.
- [25] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.
- [26] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [27] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.
- [28] Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis. *arXiv preprint arXiv:2111.05011*, 2021.
- [29] Marco Pasini and Jan Schlüter. Musika! fast infinite waveform music generation. *arXiv preprint arXiv:2208.08706*, 2022.
- [30] Mubert Inc. Mubert. URL <https://mubert.com/>, 2023.
- [31] S Forsgren and H Martiros. Riffusion - stable diffusion for real-time music generation. URL <https://riffusion.com/>, 2023.
- [32] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 776–780. IEEE, 2017.
- [33] Zalân Borsos, Matt Sharifi, Damien Vincent, Eugene Kharitonov, Neil Zeghidour, and Marco Tagliasacchi. Soundstorm: Efficient parallel audio generation. *arXiv preprint arXiv:2305.09636*, 2023.
- [34] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

- [35] Yi Luo, Zhuo Chen, and Takuya Yoshioka. Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 46–50. IEEE, 2020.
- [36] Jingjing Chen, Qirong Mao, and Dong Liu. Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation. *arXiv preprint arXiv:2007.13975*, 2020.
- [37] Max WY Lam, Jun Wang, Dan Su, and Dong Yu. Effective low-cost time-domain audio separation using globally attentive locally recurrent networks. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 801–808. IEEE, 2021.
- [38] Max WY Lam, Jun Wang, Dan Su, and Dong Yu. Sandglassnet: A light multi-granularity self-attentive network for time-domain speech separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5759–5763. IEEE, 2021.
- [39] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. Attention is all you need in speech separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–25. IEEE, 2021.
- [40] Shengkui Zhao and Bin Ma. Mossformer: Pushing the performance limit of monaural speech separation using gated single-head transformer with convolution-augmented joint self-attentions. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [41] Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 244–250. IEEE, 2021.
- [42] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.
- [43] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel PW Ellis. Mulan: A joint embedding of music audio and natural language. *arXiv preprint arXiv:2208.12415*, 2022.
- [44] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.
- [45] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.
- [46] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.
- [47] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- [48] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.
- [49] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.
- [50] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International conference on learning representations*, 2020.

- [51] Max WY Lam, Jun Wang, Dan Su, and Dong Yu. Bddm: Bilateral denoising diffusion models for fast and high-quality speech synthesis. In *International Conference on Learning Representations*, 2022.
- [52] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985.
- [53] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.
- [54] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [55] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [56] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.
- [57] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.
- [58] Tao Lei, Yu Zhang, Sida I Wang, Hui Dai, and Yoav Artzi. Simple recurrent units for highly parallelizable recurrence. *arXiv preprint arXiv:1709.02755*, 2017.
- [59] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [60] Robin Rombach and Patrick Esser. Stable diffusion v2-1. *URL <https://huggingface.co/stabilityai/stable-diffusion-2-1>*, 2023.
- [61] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.
- [62] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [63] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.
- [64] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.
- [65] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [66] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.
- [67] Won Jang, Dan Lim, Jaesam Yoon, Bongwan Kim, and Juntae Kim. Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation. *arXiv preprint arXiv:2106.07889*, 2021.
- [68] OpenAI. Chatgpt. *URL <https://chat.openai.com/>*, 2023.

- [69] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [70] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.
- [71] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.
- [72] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *INTERSPEECH*, pages 2350–2354, 2019.
- [73] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [74] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

A Mathematical Background for Dual-Path Diffusion

A.1 Forward Diffusion Process

In dual-path diffusion (DPD), we consider a Gaussian diffusion process [16] that continuously diffuses our generation target \mathbf{z} into increasingly noisy versions of \mathbf{z} , denoted as \mathbf{z}_t with $t \in [0, 1]$ running from $t = 0$ (least noisy) to $t = 1$ (most noisy). This forward diffusion process is formally defined as

$$q(\mathbf{z}_t|\mathbf{z}) = \mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{z}, \sigma_t^2 \mathbf{I}), \quad (14)$$

where two strictly positive scalar-valued, continuously differentiable functions α_t, σ_t define the noise schedule [1] of this forward diffusion process. Building upon the nice properties of Gaussian distributions, we can express $q(\mathbf{z}_t|\mathbf{z}_s)$, for any $0 \leq s < t \leq 1$, as another Gaussian distribution:

$$q(\mathbf{z}_t|\mathbf{z}_s) = \mathcal{N}\left(\mathbf{z}_t; \frac{\alpha_t}{\alpha_s} \mathbf{z}_s, \left(\sigma_t^2 - \frac{\alpha_t}{\alpha_s} \sigma_s^2\right) \mathbf{I}\right). \quad (15)$$

Regarding the choice of noise scheduling functions, we consider the typical setting used in [1, 15]: $\alpha_t = \sqrt{1 - \sigma_t^2}$, which gives rise to a *variance-preserving* diffusion process [16]. Specifically, we employ the trigonometric functions in [48], defined as follows:

$$\alpha_t := \cos(\pi t/2) \quad \sigma_t := \sin(\pi t/2) \quad \forall t \in [0, 1] \quad (16)$$

$$\Leftrightarrow \alpha_\delta := \cos(\delta) \quad \sigma_\delta := \sin(\delta) \quad \forall \delta \in [0, \pi/2]. \quad (17)$$

With this re-parameterization, the diffusion process can now be defined in terms of angle $\delta \in [0, \pi/2]$:

$$\mathbf{z}_\delta = \cos(\delta) \mathbf{z} + \sin(\delta) \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (18)$$

where \mathbf{z}_δ gets noisier as δ increases from 0 to $\pi/2$.

A.2 Prediction of Diffusion Velocity

The diffusion velocity of \mathbf{z}_δ at δ [48] is defined as:

$$\mathbf{v}_\delta := \frac{d\mathbf{z}_\delta}{d\delta} = \frac{d \cos(\delta)}{d\delta} \mathbf{z} + \frac{d \sin(\delta)}{d\delta} \boldsymbol{\epsilon} = \cos(\delta) \boldsymbol{\epsilon} - \sin(\delta) \mathbf{z}. \quad (19)$$

Based on \mathbf{v}_δ , we can compute \mathbf{z} and $\boldsymbol{\epsilon}$ from a noisy latent \mathbf{z}_δ :

$$\mathbf{z} = \cos(\delta) \mathbf{z}_\delta - \sin(\delta) \mathbf{v}_\delta = \alpha_\delta \mathbf{z}_\delta - \sigma_\delta \mathbf{v}_\delta; \quad (20)$$

$$\boldsymbol{\epsilon} = \sin(\delta) \mathbf{z}_\delta + \cos(\delta) \mathbf{v}_\delta = \sigma_\delta \mathbf{z}_\delta + \alpha_\delta \mathbf{v}_\delta, \quad (21)$$

which suggests \mathbf{v}_δ a feasible target for network prediction $\hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})$ given a collection of conditions \mathbf{c} , as an alternative to the \mathbf{z} prediction ($\hat{\mathbf{z}}_\theta(\mathbf{z}_\delta; \mathbf{c})$), e.g., in [16], and the $\boldsymbol{\epsilon}$ prediction ($\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_\delta; \mathbf{c})$), e.g., in [1, 50, 74]. As reported by Salimans and Ho [48] and Schneider et al. [23], training the neural network θ with a mean squared error (MSE) loss as in the pioneering work [1] remains effective:

$$\mathcal{L} := \mathbb{E}_{\mathbf{z} \sim p_{\text{data}}(\mathbf{z}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \delta \sim \text{Uniform}[0, 1]} \left[\|\cos(\delta) \boldsymbol{\epsilon} - \sin(\delta) \mathbf{z} - \hat{\mathbf{v}}_\theta(\cos(\delta) \mathbf{z} + \sin(\delta) \boldsymbol{\epsilon}; \mathbf{c})\|_2^2 \right], \quad (22)$$

which forms the basis of DPD's training loss, i.e., the simplest case of considering only a single chunk per input ($M = 1$) in Eq. (7). We can easily extend this to a multi-chunk version by sampling M different angles $\delta_1, \dots, \delta_M \sim \text{Uniform}[0, 1]$, where the m -th sampled angle is applied to the corresponding chunk of the latent, i.e., $\mathbf{z}[(m-1)L/M : mL/M]$.

A.3 Generative Diffusion Process

Generation is done by inverting the forward process from a noise vector randomly drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. One efficient way to accomplish this is to take advantage of DDIM [26], which enables running backward from angle δ to angle $\delta - \omega$, for any step size $0 < \omega < \delta$:

$$p_\theta(\mathbf{z}_{\delta-\omega}|\mathbf{z}_\delta) := q\left(\mathbf{z}_{\delta-\omega} \left| \mathbf{z} = \frac{\mathbf{z}_\delta - \sigma_\delta \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_\delta; \mathbf{c})}{\alpha_\delta} \right.\right) = \alpha_{\delta-\omega} \left(\frac{\mathbf{z}_\delta - \sigma_\delta \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_\delta; \mathbf{c})}{\alpha_\delta} \right) + \sigma_{\delta-\omega} \boldsymbol{\epsilon}, \quad (23)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Song et al. [26] considered $\epsilon \equiv \hat{\epsilon}_\theta(\mathbf{z}_\delta; \mathbf{c})$, leading to a deterministic update rule:

$$\mathbf{z}_{\delta-\omega} = \frac{\alpha_{\delta-\omega}}{\alpha_\delta} \mathbf{z}_\delta + \left(\sigma_{\delta-\omega} - \frac{\alpha_{\delta-\omega} \sigma_\delta}{\alpha_\delta} \right) \hat{\epsilon}_\theta(\mathbf{z}_\delta; \mathbf{c}). \quad (24)$$

Building upon the diffusion velocity, Salimans and Ho [48] re-parameterized DDIM as

$$p_\theta(\mathbf{z}_{\delta-\omega} | \mathbf{z}_\delta) := q(\mathbf{z}_{\delta-\omega} | \mathbf{z} = \alpha_\delta \mathbf{z}_\delta - \sigma_\delta \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})) \quad (25)$$

$$= \alpha_{\delta-\omega} (\alpha_\delta \mathbf{z}_\delta - \sigma_\delta \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})) + \sigma_{\delta-\omega} \epsilon, \quad (26)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here, we can similarly consider a parameterized noise vector $\epsilon \equiv \sigma_\delta \mathbf{z}_\delta + \alpha_\delta \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})$ based on Eq. (21), yielding a simplified deterministic update rule:

$$\mathbf{z}_{\delta-\omega} = \alpha_{\delta-\omega} (\alpha_\delta \mathbf{z}_\delta - \sigma_\delta \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})) + \sigma_{\delta-\omega} (\sigma_\delta \mathbf{z}_\delta + \alpha_\delta \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})) \quad (27)$$

$$= (\alpha_{\delta-\omega} \alpha_\delta - \sigma_{\delta-\omega} \sigma_\delta) \mathbf{z}_\delta + (\sigma_{\delta-\omega} \alpha_\delta - \alpha_{\delta-\omega} \sigma_\delta) \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c}) \quad (28)$$

$$= \cos(\omega) \mathbf{z}_\delta - \sin(\omega) \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c}) \quad (29)$$

where the last equation is obtained by applying the trigonometric identities:

$$\alpha_{\delta-\omega} \alpha_\delta - \sigma_{\delta-\omega} \sigma_\delta = \cos(\delta - \omega) \cos(\delta) - \sin(\delta - \omega) \sin(\delta) = \cos(\omega); \quad (30)$$

$$\sigma_{\delta-\omega} \alpha_\delta - \alpha_{\delta-\omega} \sigma_\delta = \sin(\delta - \omega) \cos(\delta) - \cos(\delta - \omega) \sin(\delta) = \sin(\omega). \quad (31)$$

Building upon this angular update rule and having specified the angle step sizes $\omega_1, \dots, \omega_T$ with $\sum_{t=1}^T \omega_t = \pi/2$, we can generate samples from $\mathbf{z}_{\pi/2} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ after T steps of sampling:

$$\mathbf{z}_{\delta_t - \omega_t} = \cos(\omega_t) \mathbf{z}_{\delta_t} - \sin(\omega_t) \hat{\mathbf{v}}_\theta(\mathbf{z}_{\delta_t}; \mathbf{c}), \quad \delta_t = \begin{cases} \frac{\pi}{2} - \sum_{i=t+1}^T \omega_i, & 1 \leq t < T; \\ \frac{\pi}{2}, & t = T, \end{cases} \quad (32)$$

running from $t = T$ to $t = 1$.

B Training and Implementation Details

B.1 Audio VAE-GAN

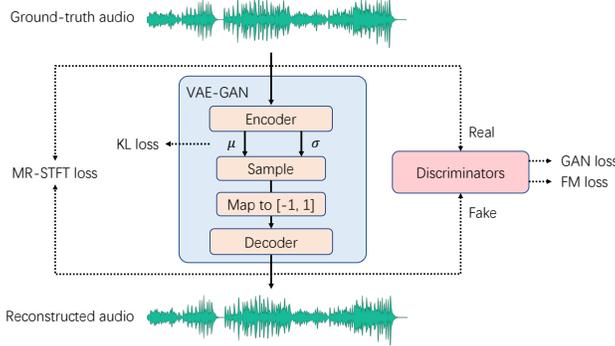


Figure 4: The audio VAE-GAN trained for dual-path diffusion models

As shown in Figure 4, we train a VAE-GAN to extract 250Hz 16-dimensional latent $\mathbf{z} \in \mathbb{R}^{L \times 16}$ from a 24kHz audio $\mathbf{x} \in \mathbb{R}^{T_{\text{wav}}}$ with $L = \lceil T_{\text{wav}}/96 \rceil$. The audio VAE-GAN mainly comprises three trainable modules: (i) a variational Gaussian encoder $\mathcal{E}_\phi(\mathbf{x}) \equiv \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x})\mathbf{I})$, (ii) a decoder $\mathcal{D}_\phi(\mathbf{z})$, and (iii) a set of n discriminators $\{D^{(i)}\}_{i=1}^n$.

Regarding network architecture, we use the ResNet-style convolutional neural networks (CNNs) in HiFi-GAN [66] as the backbone.⁶ For the encoder, we replace the up-sampling blocks in HiFi-GAN with convolution-based down-sampling blocks, with down-sampling rates of [2, 3, 4, 4], output dimensions of [32, 64, 128, 256] and kernel sizes of [5, 7, 9, 9] in four down-sampling blocks, giving 40M

⁶Our implementation is similar to that in <https://github.com/jik876/hifi-gan>.

parameters. The final layer of the encoder maps the 256-dimensional output to two 16-dimensional latent sequences, respectively for the mean and variance of diagonal Gaussian sampling.⁷ As shown in Figure 4, to match the normal range of targets for diffusion models [1, 2], we map the sampled outputs to $[-1, 1]$ by $\mathbf{z}_{(i,j)} := \min \{ \max \{ \bar{\mathbf{z}}_{(i,j)}/3, -1 \}, 1 \} \forall i, j$, where the subscript (i, j) denotes the value on the i -th row and j -th column, and the choice of 3 in practice would sieve extreme values occupying $< 0.1\%$. For the architecture setting of the decoder, it inherits the same architecture of HiFi-GAN, and uses up-sampling rates of [4, 4, 3, 2], kernel sizes of [9, 9, 5, 7] and larger number of output channels ([768, 384, 192, 96]) for four up-sampling blocks, taking 60.1M parameters.

For adversarial training, we use the multi-period discriminators in [66] and the multi-resolution spectrogram discriminators in [67]. The training scheme is similar to that in [66]. The training loss for the encoder and the decoder comprises four components:

$$\mathcal{L}_{\text{vae-gan}}(\phi) := \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\mathbb{E}_{\mathbf{z} \sim \mathcal{E}_\phi(\mathbf{x})} [\lambda_{\text{mr-stft}} \mathcal{L}_{\text{mr-stft}} + \lambda_{\text{fm}} \mathcal{L}_{\text{fm}} + \lambda_{\text{gan}} \mathcal{L}_{\text{gan}} + \lambda_{\text{kl}} \mathcal{L}_{\text{kl}}]] \quad (33)$$

$$\mathcal{L}_{\text{mr-stft}} := \sum_{r=1}^R \|\text{STFT}_r(\mathbf{x}) - \text{STFT}_r(\mathcal{D}_\phi(\mathbf{z}))\|_1 \quad (34)$$

$$\mathcal{L}_{\text{fm}} := \sum_{i=1}^n \frac{1}{|D^{(i)}|} \sum_{l=1}^{|D^{(i)}|} \|D_l^{(i)}(\mathbf{x}) - D_l^{(i)}(\mathcal{D}_\phi(\mathbf{z}))\|_1 \quad (35)$$

$$\mathcal{L}_{\text{gan}} := \sum_{i=1}^n \left(D^{(i)}(\mathcal{D}_\phi(\mathbf{z})) - 1 \right)^2 \quad (36)$$

$$\mathcal{L}_{\text{kl}} := \text{KL}(\mathcal{E}_\phi(\mathbf{x}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})), \quad (37)$$

where STFT_r computes the magnitudes after the r -th short-time Fourier transform (STFT) out of $R = 7$ STFTs (the number of FFTs = [8192, 4096, 2048, 512, 128, 64, 32]; the window sizes = [4096, 2048, 1024, 256, 64, 32, 16]; the hop sizes = [2048, 1024, 512, 128, 32, 16, 8]), $|D^{(i)}|$ denotes the number of hidden layers used for feature matching in discriminator $D^{(i)}$, $D_l^{(i)}$ denotes the outputs of the l -th hidden layers in discriminator $D^{(i)}$, and $\lambda_{\text{mr-stft}}$, λ_{fm} , λ_{gan} , λ_{kl} are the weights, respectively, for the multi-resolution STFT loss $\mathcal{L}_{\text{mr-stft}}$, the feature matching loss \mathcal{L}_{fm} , the GAN’s generator loss \mathcal{L}_{gan} , and the Kullback–Leibler divergence based regularization loss \mathcal{L}_{kl} . To balance the scale of different losses, we set $\lambda_{\text{mr-stft}} = 50$, $\lambda_{\text{fm}} = 20$, $\mathcal{L}_{\text{gan}} = 1$, and $\lambda_{\text{kl}} = 5 \times 10^{-3}$ in our training. In practice, we find it critical to lower the scale of the KL loss for a better reconstruction, though the distribution of the latents can still be close to zero mean and unit variance.

B.2 Wav2Vec2-Conformer

Our implementation of Wav2Vec2-Conformer was based on an open-source library.⁸ In particular, Wav2Vec2-Conformer follows the same architecture as Wav2Vec2 [44], but replaces the Transformer structure with the Conformer [45]. This model with 199.5M parameters was trained in self-supervised learning (SSL) manner similar to [44] using our prepared 257k hours of music data.

B.3 MuLan

Our reproduced MuLan [43] is composed of a music encoder and a text encoder. For music encoding, we rely on a publicly accessible Audio Spectrogram Transformer (AST) model pre-trained on AudioSet,⁹ which gives promising results on various audio classification benchmarks. For text encoding, we employ the BERT [8] base model pre-trained on a large corpus of English data using a masked language modeling (MLM) objective.¹⁰ These two pre-trained encoders, together having 195.3M parameters, were subsequently fine-tuned on the 257k hours of music data with a text augmentation technique similar to [6]. In particular, we enriched the tag-based texts to generate music captions by asking ChatGPT [68]. At training time, we randomly paired each audio with either

⁷The Gaussian sampling is referred to LDMS’ implementation at <https://github.com/CompVis/latent-diffusion/blob/main/ldm/modules/distributions/distributions.py>

⁸https://huggingface.co/docs/transformers/model_doc/wav2vec2-conformer

⁹<https://huggingface.co/MIT/ast-finetuned-audioset-10-10-0.4593>

¹⁰<https://huggingface.co/bert-base-uncased>

Algorithm 1 Music Generation

```

1: given  $\mathcal{D}_\phi, \hat{\mathbf{v}}_\theta, T, \omega_1, \dots, \omega_T$ 
2: input Music/text prompt  $\mathcal{P}$ 
3:
4: Initialize  $\delta_T = \pi/2$ 
5: Compute the MuLan tokens for  $\mathcal{P}$ :  $\mathbf{c}_{1:T_{\text{cnd}}}$ 
6: Generate  $\mathbf{u}_{1:T_{\text{ST}}}$  from  $\mathbf{c}_{1:T_{\text{cnd}}}$  with LM  $\triangleright$  (1)
7: Sample  $\mathbf{z}_{\delta_T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8: for  $t = T$  to 1 do
9:   Prepare condition:  $\mathbf{c} = \{\mathbf{u}_{1:T_{\text{ST}}}, [\delta_t]_{r=1}^L\}$   $\triangleright$  (8)
10:  Update angle:  $\delta_{t-1} = \delta_t - \omega_t$   $\triangleright$  (3)
11:   $\mathbf{z}_{\delta_{t-1}} = \cos(\omega_t)\mathbf{z}_{\delta_t} - \sin(\omega_t)\hat{\mathbf{v}}_\theta(\mathbf{z}_{\delta_t}; \mathbf{c})$   $\triangleright$  (9)
12: end for
13: repeat
14:   pass  $\mathbf{c}_{1:T_{\text{cnd}}}, \mathbf{u}_{1:T_{\text{ST}}}$  and  $\mathbf{z}_0$  to Algorithm 2
15: until  $\mathbf{z}_0$  reaches the desired length
16: return  $\mathcal{D}_\phi(\mathbf{z}_0)$ 

```

Algorithm 2 Music Continuation

```

1: given  $\mathcal{D}_\phi, \hat{\mathbf{v}}_\theta, T, M, \omega_1, \dots, \omega_T$ 
2: input Music  $\mathbf{z}_0$  and  $\mathbf{c}_{1:T_{\text{cnd}}}, \mathbf{u}_{1:T_{\text{ST}}}$  (if provided)
3:
4: Denote  $M_{\text{ST}} = \lceil T_{\text{ST}}/M \rceil, L_M = \lceil L/M \rceil$ 
5: Initialize  $\delta_T = \pi/2$ 
6: Generate  $\mathbf{u}_{T_{\text{ST}}:T_{\text{ST}}+M_{\text{ST}}}$  from  $\mathbf{c}_{1:T_{\text{cnd}}} \oplus \mathbf{u}_{M_{\text{ST}}:T_{\text{ST}}}$ 
7: Sample  $\mathbf{z}_{\text{new}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^{L_M}$ 
8: Save first chunk:  $\mathbf{z}_{\text{save}} = \mathbf{z}_0[:L_M]$ 
9:  $\mathbf{z}_{\delta_T} = \mathbf{z}_0[L_M:] \oplus \mathbf{z}_{\text{new}}$ 
10: for  $t = T$  to 1 do
11:  Update  $\delta_{\text{new}} = [0]_{r=1}^{L-L_M} \oplus [\delta_t]_{r=1}^{L_M}$ 
12:  Prepare condition:  $\mathbf{c} = \{\mathbf{u}_{M_{\text{ST}}:T_{\text{ST}}+M_{\text{ST}}}, \delta_{\text{new}}\}$ 
13:  Update angle:  $\delta_{t-1} = \delta_t - \omega_t$ 
14:   $\mathbf{z}_{\delta_{t-1}} = \cos(\omega_t)\mathbf{z}_{\delta_t} - \sin(\omega_t)\hat{\mathbf{v}}_\theta(\mathbf{z}_{\delta_t}; \mathbf{c})$ 
15: end for
16: return  $\mathbf{z}_{\text{save}} \oplus \mathbf{z}_0$ 

```

the generated caption or its respective tags. In practice, this could robustly improve the model’s capability of handling free-form text.

C Algorithms for MeLoDy

MeLoDy supports music or text prompting for music generation, as illustrated in Figure 1. We concretely detail the sampling procedures in Algorithm 1, where the algorithm starts by generating the latent sequence of length L and then recursively prolongs the latent sequence using Algorithm 2 until it reaches the desired length.

We further explain how music continuation can be effectively done in DPD. Recall that the inputs for training DPD are the concatenated chunks of noisy latents in different noise scales. To continue a given music audio, we can add a new chunk composed of random noises and drop the first chunk. This is feasible since the conditions (i.e., the semantic tokens and the angles) defined for DPD have an autoregressive nature. Based on the semantic LM, we can continue the generation of $\lceil T_{\text{ST}}/M \rceil$ semantic tokens for the new chunk. Besides, it is sensible to keep the chunks other than the new chunk to have zero angles: $\delta_{\text{new}} := [0]_{r=1}^{L-\lceil L/M \rceil} \oplus [\delta_t]_{r=1}^{\lceil L/M \rceil}$, as shown in Algorithm 2.

In addition, music inpainting can be done in a similar way. We replace the inpainting partition of the input audio with random noise and partially set the angle vector to zeros to mark the positions where the denoising operations are not needed. Yet, in this case, the semantic tokens can only be roughly estimated using the remaining part of the music audio.

Table 4: The objective measures for the ablation study on angle schedules.

Angle schedule (ω_t)	Steps (T)	FAD (\downarrow)	MCC (\uparrow)
Uniform [23]: $\omega_t = \frac{\pi}{2T}$	10	8.52	0.45
	20	6.31	0.49
Ours proposed in Eq. (4): $\omega_t = \frac{\pi}{6T} + \frac{2\pi t}{3T(T+1)}$	10	5.93	0.52
	20	5.41	0.53

D Ablation Study on Angle Schedules

We conduct an ablation study on angle schedules to validate the effectiveness of our proposed angle schedule $\omega_1, \dots, \omega_T$ in Eq. (4) in comparison to the previous uniform angle schedule [23] also used for angle-parameterized continuous-time diffusion models. In particular, the same pre-trained DPD model $\hat{\mathbf{v}}_\theta$ and was used to sample with two different angle schedules with 10 steps and 20 steps,

respectively, conditional on the same semantic tokens generated for the text prompts in MusicCaps. Table 4 shows their corresponding objective measures in terms of FAD and MCC. We observe a significant improvement, especially when taking a small number of sampling steps, by using the proposed sampling method. This is aligned with our expectations that taking larger steps at the beginning of the sampling followed by smaller steps could improve the quality of samples, similar to the findings in previous diffusion scheduling methods [50, 51].

We further qualitatively analyze the quality of the generated samples using some simple text prompts of instruments, i.e., flute, saxophone, and acoustic guitar, by pair-wise comparing their spectrograms as illustrated in Figure 5. In the case of “flute”, sampling with the proposed angle schedule results in a piece of naturally sound music, being more saturated in high-frequency bands and even remedying the breathiness of flute playing, as shown in Figure 5b. On the contrary, we can observe from the spectrogram in Figure 5a that the sample generated with a uniform angle schedule is comparatively monotonous. In the case of “saxophone”, the uniform angle schedule leads to metallic sounds that are dissonant, as revealed by the higher energy in 3kHz to 6kHz frequency bands shown in Figure 5c. In comparison, the frequency bands are more consistent in Figure 5d, when the proposed schedule is used. While the comparatively poorer sample quality using the uniform schedule could be caused by the limited number of sampling steps, we also show the spectrograms after increasing the sampling steps from 10 to 20. In the case of “acoustic guitar”, when taking 20 sampling steps, the samples generated with both angle schedules sound more natural. However, in Figure 5e, we witness a horizontal line around the 4.4kHz frequency band, which is unpleasant to hear. Whereas, the sample generated by our proposed schedule escaped such an acoustic issue, as presented in Figure 5f.

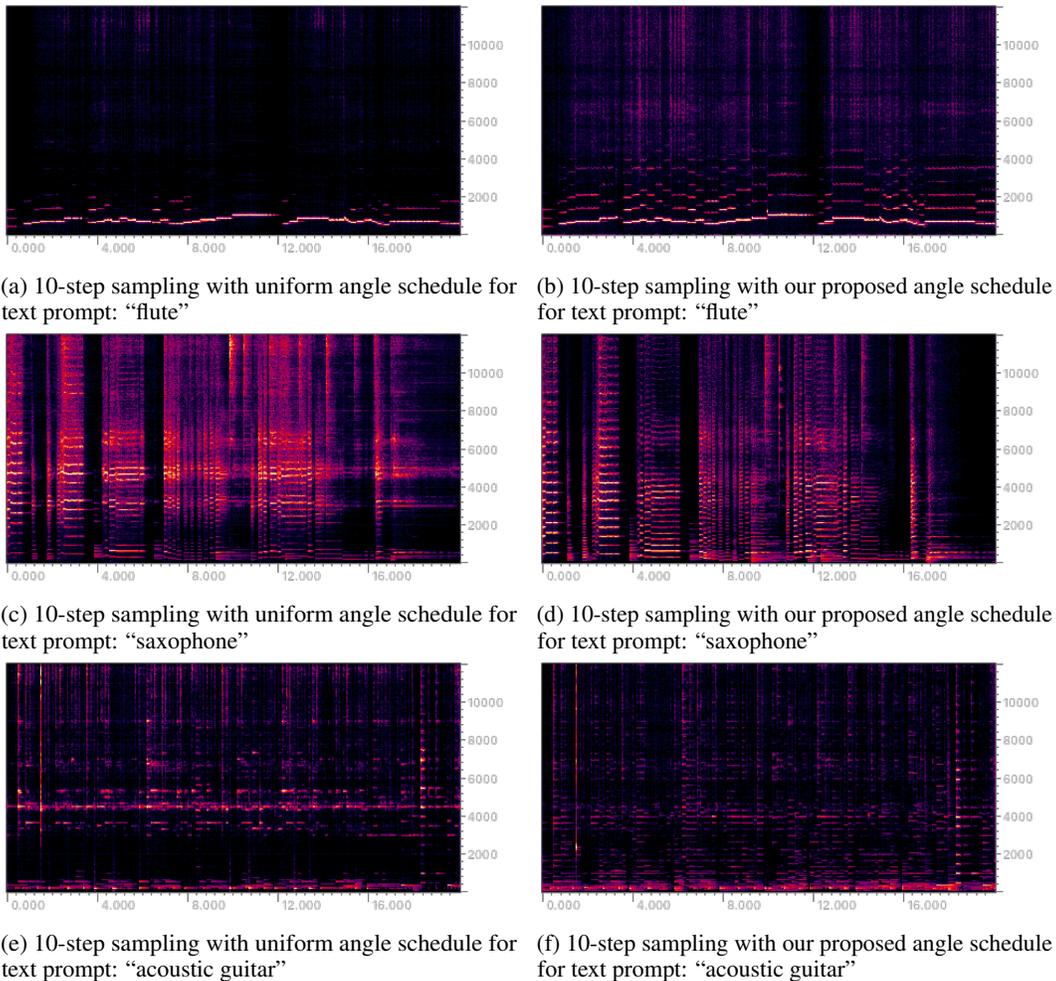


Figure 5: Spectrograms of generated samples with uniform (left) and our proposed (right) angle schedules

Table 5: The objective measures for the ablation study on architectures.

Architecture	Velocity MSE (\downarrow)	SI-SNRi (\uparrow)
UNet-1D [23]	0.13	5.33
UNet-2D [31]	0.15	4.96
DPD (Ours)	0.12	6.15

E Ablation Study on Architectures

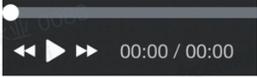
To examine the superiority of our proposed dual-path model in Figure 2, we also study the ablation of network architectures. In particular, to focus on the denoising capability of different architectures, we only take a subset of the training data (approximately 5k hours of music data) to train different networks with the same optimization configurations – 100k training steps using AdamW optimizer with a learning rate of 5×10^{-4} and a batch size of 96 on 8 NVIDIA V100 GPUs. For a fair comparison, we train the UNet-1D¹¹ and the UNet-2D¹² with comparable numbers of parameters (approximately 300M). Note that the FAD and MCC measures are not suitable for evaluating the performance of each forward pass of the trained network for denoising. In addition to the training objective, i.e., the velocity MSE, we use the scale-invariant signal-to-noise ratio (SNR) improvements (SI-SNRi) [35, 37] between the true latent \mathbf{z} and the predicted latent $\hat{\mathbf{z}} = \alpha_\delta \mathbf{z}_\delta - \sigma_\delta \hat{\mathbf{v}}_\theta(\mathbf{z}_\delta; \mathbf{c})$. The results are shown in Table 5, where our proposed dual-path architecture outperforms the other two widely used UNet-style architectures in terms of both the velocity MSE and SI-SNRi.

F Qualitative Evaluation

To conduct a pair-wise comparison, each music producer is asked to fill in the form composed of three questions. Specifically, we present the user interface for each pair-wise comparison in Figure 6.

Text Prompt
acoustic guitar

Audio 1



Audio 2



Questions: (all single-selection)

Q1. Which one is better in terms of Musicality?

Audio 1 Audio 2 Similar

Q2. Which one is better in terms of Audio Quality?

Audio 1 Audio 2 Similar

Q3. Which one is better in terms of Text Correlation?

Audio 1 Audio 2 Similar

Figure 6: The user interface for music producers in each pair-wise comparison

¹¹Our implementation of UNet-1D relied on <https://github.com/archinetai/a-unet>.

¹²Our implementation of UNet-2D relied on <https://huggingface.co/riffusion/riffusion-model-v1>.