MusicFlow: Cascaded Flow Matching for Text Guided Music Generation

K R Prajwal [*1] **Bowen Shi** [*2] **Matthew Lee** [2] **Apoorv Vyas** [2] **Andros Tjandra** [2] **Mahi Luthra** [2] **Baishan Guo** [2] **Huiyu Wang** [2] **Triantafyllos Afouras** [2] **David Kant** [2] **Wei-Ning Hsu** [2]

Abstract

We introduce MusicFlow, a cascaded text-to-music generation model based on flow matching. Based on self-supervised representations to bridge between text descriptions and music audios, we construct two flow matching networks to model the conditional distribution of semantic and acoustic features. Additionally, we leverage masked prediction as the training objective, enabling the model to generalize to other tasks such as music infilling and continuation in a zero-shot manner. Experiments on MusicCaps reveal that the music generated by MusicFlow exhibits superior quality and text coherence despite being over $2 \sim 5$ times smaller and requiring 5 times fewer iterative steps. Simultaneously, the model can perform other music generation tasks and achieves competitive performance in music infilling and continuation.

1. Introduction

Audio generation has recently received a lot of attention from the research community as well as the general public. Making sound automatically has a lot of practical applications, including voice acting, podcast making, creating foley sound effects (Luo et al., 2023), making background music for movies (Liu et al., 2023c), and can greatly reduce the barrier for audio content creation. In terms of research, audio generation poses a few challenges due to its long-term structure and complex interaction between channels (e.g., multiple events may appear at the same time), thus being a suitable testbed for generative models.

Modeling approaches for audio generation has rapidly progressed over the past few years due to the development of sophisticated generative methods such as autoregressive language models (Kreuk et al., 2022; Wang et al., 2023) and non-autoregressive approaches (Le et al., 2023; Vyas et al., 2023; Liu et al., 2023a). A significant portion of the generative models are focused on speech and general sound, where state-of-the-art (SOTA) models (Vyas et al., 2023) are able to generate speech in diverse styles or general sound events in highly realistic manner. Compared to these two common modalities, music generation is a particularly challenging problem as it requires modeling long-term temporal structures (Agostinelli et al., 2023) and full frequency spectrum (Müller, 2015). Compared to typical sound events (e.g., dog barking), it contains harmonies and melodies from different instruments. Music pieces often consist of multiple tracks, which can be intricately woven together and may involve significant interference.

With the improvement of audio tokenizers (Zeghidour et al., 2021; Défossez et al., 2022) and generative models, the quality of generated music has been greatly improved in recent works (Agostinelli et al., 2023; Copet et al., 2023). However, many prior works are built upon language models (Agostinelli et al., 2023; Copet et al., 2023; Yang et al., 2023), which requires a computationally expensive autoregressive inference procedure with number of forward passes proportional to the sequence length. This is worsened because many such models are based on a hierarchical set of units (e.g., Encodec tokens (Copet et al., 2023)), which brings another factor up to the computation. Despite the usage of non-autoregressive models such as diffusion models (Liu et al., 2023b; Huang et al., 2023; Forsgren & Martiros, 2022; Schneider et al., 2023), these approaches require hundreds of denoising steps during inference to achieve high performance. On the other hand, most of the existing models perform generation in a single stage, which models the audio waveform (Huang et al., 2023) or its low-level representation such as VAE features (Liu et al., 2023b) conditioned on text description directly. As music audios contains rich structural information and its text description can be very detailed (e.g., *This is a live recording of a keyboardist playing a twelve bar blues progression on an electric keyboard. The player adds embellishments between chord changes and the piece sounds groovy, bluesy and soulful.*), such approaches commonly fail to capture the intriguing dependency between text description and music

[*]Equal contribution [1]VGG, University of Oxford, UK. Work done while at Meta. [2]Meta, USA. Correspondence to: K R Prajwal <prajwal@robots.ox.ac.uk>, Bowen Shi <bshi@meta.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

pieces. Finally, most existing work focuses on text-to-music (TTM) generation, while lacking the ability to perform other practically useful generative tasks such as music infilling.

In this paper, we present MusicFlow, a cascaded text-to-music generation model based on flow matching. Our model is composed of two flow-matching networks, which transform text description into a sequence of semantic features and semantics into decodable acoustic features in non-autoregressive fashion. The flow matching objective equips the model with high efficiency in both training and inference, outperforming prior works with smaller model size and faster inference speed. Furthermore, by training with a masked prediction objective, MusicFlow is able to perform multiple music generation tasks, including TTM, music continuation and music infilling in a unified fashion.

2. Related Work

Early works on music generation are mostly on constrained scenarios, such as generating audios for a specific style (e.g., Jazz (Hung et al., 2019)) or a specific instrument (e.g., piano (Hawthorne et al., 2018)). More recent works shift the focus to generating music from free-form natural language descriptions. Typically, the language description is encoded by a pre-trained text encoder, which is then used for conditioning the model. One big class of the generation backbone falls into the category of language models (Agostinelli et al., 2023; Copet et al., 2023). In this type of model, an audio is quantized into discrete units through an auto-encoder (e.g., SoundStorm (Zeghidour et al., 2021), Encodec (Défossez et al., 2022)). The language model is built to model the distribution of these units. During inference, the units sampled from the language model is decoded back into raw waveforms with the decoder directly without an explicit vocoder. The units are sampled either autoregressively (Copet et al., 2023; Agostinelli et al., 2023; Yang et al., 2023) or in conjunction with non-autoregressive unit decoding (Ziv et al., 2024). Diffusion-based music generation is typically built on top of the audio spectrogram. AudioLDM2 (Liu et al., 2023b) employs a variational auto-encoder to compress the spectrogram, where a DDIM (Song et al., 2020) model is trained with the compressed features. During inference, the generation is first decoded with the VAE decoder and transformed to waveform with a vocoder. Similar approaches include Riffusion (Forsgren & Martiros, 2022), which directly fine-tunes a stable diffusion model with spectrograms; MeLoDy (Lam et al., 2024) which proposes a LM-guided Diffusion with a focus on fast sampling speed; and Noise2Music (Huang et al., 2023), which also builds a diffusion-based vocoder; and StableAudio (Evans et al., 2024) which takes a latent diffusion approach, again with a focus on fast inference.

Most of the existing methods directly learns the music distribution conditioned on text , which models the low-level audio features directly. In this work, our cascaded model is bridged by semantic features, which are learned separately with a self-supervised model. A similar approach to ours is MusicLM (Agostinelli et al., 2023), which learns two language models generating semantic and acoustic units respectively. However, our model relies on flow matching, which offers improved efficiency. Its non-autoregressive nature also enables the model to better leverage context and generalize to other tasks.

3. Method

3.1. Background: Flow matching

Introduced in (Lipman et al., 2023), flow matching is a method addressing continuous transformation of probability densities. Specifically, it studies flow, a time-dependent diffeomorphic mapping $\phi_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ordinary differential equation (ODE):

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \tag{1}$$

$v_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, namely a vector field, is parameterized by a neural network θ and learned by minimizing the flow matching objective: $L_{FM} = \mathbb{E}_{t,p_t(x)}||v_t(x;\theta) - u_t(x)||^2$, where $p_t(x)$ is a probability density path and $u_t(x)$ is the corresponding vector field. As both $p_t(x)$ and $u_t(x)$ are generally unknown, Lipman et al. (2023) proposes minimizing the following conditional flow matching objective, which is equivalent to minimizing L_{FM}:

$$L_{\text{CFM}} = \mathbb{E}_{t,p(x|x_1),q(x_1)}||v_t(x;\theta) - u_t(x|x_1)||^2 \tag{2}$$

Considering Gaussian distributions for $p_t(x|x_1) = \mathcal{N}(x|\mu_t(x_1), \sigma_t(x_1)^2 I)$, the target vector field for Equation 2 can be solved in closed form: $u_t(x|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}(x - \mu_t(x_1)) + \mu_t'(x_1)$. Several diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) can be described under the same framework with specific conditional probability paths of $\sigma_t(x_1)$ and $\mu_t(x_1)$. Specifically, Lipman et al. (2023) considers a conditional probability path with Gaussian mean and standard deviation changing linearly in time with $\mu_t(x) = tx$ and $\sigma_t(x) = 1 - (1 - \sigma_{min})t$, which produces an optimal transport displacement mapping between conditional distributions. Due to its efficiency in both training and inference (Lipman et al., 2023; Le et al., 2023), we always stick to this conditional probability path as the default setting throughout the paper.

3.2. Problem Formulation

We now describe the music generation task and the general methodology based on flow matching that we employ. Given a dataset consisting of audio-text pairs (x, w),

where $x \in \mathbb{R}^{T \times C}$ (T: number of timesteps, C: number of channels) is the music audio and $w = \{w_1, w_2, .., w_n\}$ (w: words) is the corresponding textual description represented as a sequence of words, the goal is to build a text-conditioned music generation model $p(x|w)$. In addition to generating music from scratch, we further consider two practical tasks: music continuation $p(x_{t_1:T}|x_{1:t_1}, w)$ and music infilling $p(x_{t_1:t_2}|x_{1:t_1}, w, x_{t_2:T})$, with t_1, $t_2 \in [0, T]$. In order to allow the model to perform all the text-guided music generation, we formulate our approach as an in-context learning task following (Le et al., 2023). Specifically, given a binary temporal mask m for a music track x, we train a conditional flow matching model predicting the vector field in the masked regions of the music track $x_m = x \odot m$ while conditioning on the unmasked regions of the music track $x_{ctx} = x \odot (1 - m)$ and the text caption w about the music piece. Formally, we train with the following flow matching loss: $L_{\text{CFM}} = \mathbb{E}_{t,m,p(x|x_1),q(x_1,w)}||m \odot (v_t(x, x_{ctx}, w; \theta) - u_t(x|x_{ctx}, w))||^2$.

In addition to increasing the model capacity, such masked prediction objective also benefits generative modeling in general, as is shown in (Li et al., 2023a; Le et al., 2023). Within this framework, the three tasks of TTM, music continuation and music infilling can be conceptualized as setting specific mask values for $p(x_m|(1 - m) \odot x, w)$, where m is set to be $\mathbf{1}_{1:T}$, $[\mathbf{0}_{1:t_1}, \mathbf{1}_{t_1:T}]$ and $[\mathbf{0}_{1:t_1}, \mathbf{1}_{t_1:t_2}, \mathbf{0}_{t_2:T}]$ respectively.

3.3. A Cascaded Flow-matching Approach

Training flow matching to directly generate music conditioned on text captions is difficult (Liu et al., 2023b) given the vast number of potential music tracks corresponding to a single text caption. As the text caption lacks the fine-grained information to adequately describe a music track, we propose to condition on a latent music representation that describes the music at the frame level.

MusicFlow is thus divided into two stages: semantic modeling and acoustic modeling. The first stage outputs latent representations $h = (h_1, h_2, ...h_M) \in \mathcal{R}^{M \times D}$ conditioned on a text caption w. In the second stage, we condition on the latent representations from the first-stage model, and a text caption w to output low-level acoustic features of N frames, $x = (x_1, x_2, ..., x_N) \in \mathcal{R}^{N \times C}$. Note h and x are monotonically aligned. Both stages are inherently stochastic, meaning there are multiple potential (h, x) pairs for a given caption w. Therefore, we model these two stages separately with flow matching. In both stages, we predict masked vector fields as discussed before and provide detailed descriptions on the two stages below.

3.4. Stage 1: Music Semantic Flow Matching from Text

Our first-stage model consists of generating the semantics of the music piece conditioned on the text description. Here the semantics refer to high-level musical information instead of fine-grained details such as the general audio quality, which are inferred from the text description. For music, the semantics can refer to the melody and rhythm or harmony in a piano piece, analogous to the linguistic content in speech.

Semantic latent representation One natural way of representing music is through music transcription. Transcripts in music typically refer to some notation system (e.g., music scores) that indicates the pitches, rhythms, or chords of a musical piece. A notable advantage of music transcript is its interpretability as it is human-readable and thus poses easy alignment with humans. However, for large-scale audio datasets, the associated music transcripts are usually not readily available, while manual annotation involves a non-trivial amount of labeling efforts. Automatic music transcription is a challenging task (Benetos et al., 2019) and the existing approaches (Bittner et al., 2022; Hawthorne et al., 2021; Hsu & Su, 2021; Su et al., 2019; Hawthorne et al., 2018) are heavily restricted to a single-instrument setting (e.g. piano, solo vocals, etc.).

To address the challenge of acquiring music transcriptions, we adopt HuBERT (Hsu et al., 2021), a popular self-supervised speech representation learning framework, to obtain frame-level semantic features, which can be regarded as a form of pseudo transcription. In essence, a HuBERT model consists of masked prediction of hidden units from the raw audio, which are inferred initially from MFCC and iteratively refined with layerwise features. For speech, HuBERT units have shown to correlate well with phonemes (Hsu et al., 2021) and its intermediate features entails rich semantic information (Pasad et al., 2023). In music understanding tasks, HuBERT has been successfully applied in source separation (Pasini et al., 2023), shedding light on its potential for capturing musical characteristics. As the original HuBERT model is pre-trained with speech only, we re-train HuBERT using music data following the original recipe. Training details are given in Section 4.

Semantic flow matching Given a HuBERT model $\mathcal{H}$, one can extract the semantic features from its lth layer l: $h = \mathcal{H}(x) \in \mathcal{R}^{M \times C_h}$, where C_h is the HuBERT feature dimension. The layer index l is tuned in practice. A text-conditioned semantic flow-matching model $p(h|w)$ can be trained given text-feature pairs (h, w). As described in Section 3.2, we adopt the masked prediction objective by conditioning on the context $h_{ctx} = m \odot h$, where m is a span mask of length M. More formally, we adopt the following training objective for the semantic modeling stage: $L_{\text{H-CFM}} = \mathbb{E}_{t,m,p(h|h_1),q(h_1,w)}||m \odot (v_t(h, h_{ctx}, w; \theta) - u_t(h|h_{ctx}, w))||^2$. Cross-attention layers are integrated into

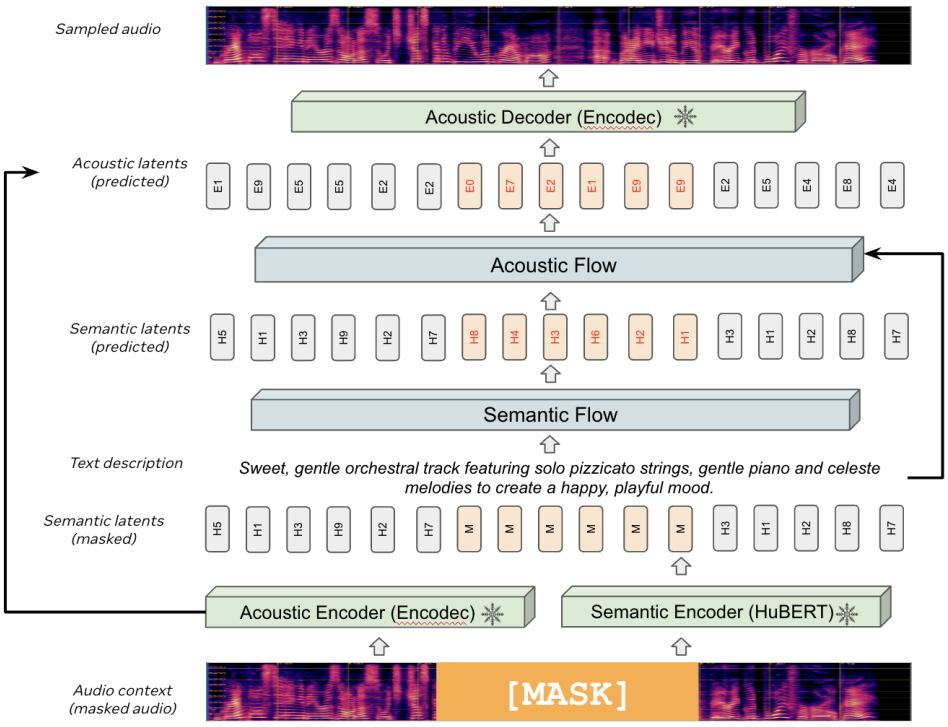

Figure 1. MusicFlow Diagram. Note the acoustic encoder, acoustic decoder and semantic encoder are pre-trained and frozen during generative model training. For text-to-music generation (i.e., 100% masking), both acoustic and semantic encoder are discarded in inference.

the backbone model, enabling it to attend to the text description w, akin to (Rombach et al., 2021).

As an alternative to modeling the distribution of dense features, one can quantize the layerwise features h into units u and model the unit distribution $p(u|w)$ instead. For this case, a straightforward method is to build an autoregressive language model, which factorizes $p(u|w) = \prod_{n=1}^{M} p(u_n|u_{1:n-1}, w)$. Using a semantic LM has been explored in (Agostinelli et al., 2023) in hierarchical LMs for music generation. We also noticed its effectiveness when combined with flow matching, as will be shown in Section 4. However, this hybrid model is unsuitable for music infilling task due to its left-to-right nature.

3.5. Stage 2: Music Acoustic Flow Matching from text and semantics

Acoustic latent representation The second-stage model aims to infer the low-level acoustic information (e.g., volume, recording quality) implied by the semantic tokens. Directly predicting raw waveforms ensures the completeness of information while imposes the challenge of modeling the long sequences. To balance between quality and sequence length, we use Encodec (Défossez et al., 2022) to map raw waveforms into dense feature sequences. In a nutshell, Encodec (Défossez et al., 2022), an auto-encoder based on residual vector quantization, comprises of an encoder $\mathcal{E}$ and decoder $\mathcal{D}$. During training, we map raw waveforms into acoustic features with the encoder $\mathcal{E}$: $e = \mathcal{E}(x) \in \mathbb{R}^{N \times C_e}$, where C_e is the feature dimension of encodec.

Acoustic flow matching The second-stage flow matching aims to model the following conditional distribution: $p(e|h, w)$. Similar to semantic flow matching, we apply masked prediction and the corresponding training objective is formulated as: $L_{\text{E-CFM}} = \mathbb{E}_{t,m,p(e|h_1,e_1),q(e_1,h_1,w)} ||m \odot (v_t(e, e_{ctx}, h_1, w; \theta) - u_t(e|e_1, h_1, e_{ctx}, w))||^2$. As the semantic and acoustic features are aligned ($N/M \approx sr_{\mathcal{E}}/sr_{\mathcal{H}}$, sr: sample rate), we simply linearly interpolate the HuBERT feature sequence h to length N before feeding it into encoding.

Note though Encodec includes multiple different codebooks to quantize the latent features, we directly model the dense feature sequence from the encoder $\mathcal{E}$ without any quantization. This avoids the length increase brought by using multiple codebooks, where the total number of discrete tokens is $K-1$ times more than the dense feature length. Thus, it eliminates the necessity of carefully designing interleaving pattern of discrete tokens to account for dependencies

between multiple codebooks (Copet et al., 2023; Wang et al., 2023).

3.6. Classifier-free guidance

During inference, we sequentially sample the HuBERT features $\hat{h}$ and encodec features $\hat{e}$ using the estimated vector field $v_t(h, h_{ctx}, w; \theta_h)$ and $v_t(e, e_{ctx}, \hat{h}, w; \theta_e)$ following the ODE equation 1. The acoustic features are decoded into waveforms via the decoder $\mathcal{D}$ of the Encodec.

As is common in diffusion models, classifier-free guidance is a widely used technique to balance sample diversity and text coherence. Thus we also adopt it in our cascaded generation framework. For flow matching, using classifier-free guidance (Zheng et al., 2023) consists of computing a linear combination between conditional and unconditional vector field: $\tilde{v}_t^H(h, w, h^{ctx}; \theta_h) = (1 + \alpha_h)v_t^H(h, w, h^{ctx}; \theta_h) - \alpha_h v_t^{H,uncond}(h; \theta_h)$ and $\tilde{v}_t^E(e, e^{ctx}, \hat{h}, w; \theta_e) = (1 + \alpha_e)v_t^e(e, e^{ctx}, \hat{h}, w; \theta_e) - \alpha_e v_t^{e,uncond}(e; \theta_e)$.

In order to model the unconditional vector field $v_t^{H,uncond}$ and $v_t^{E,uncond}$ with v_t^H and v_t^E , we randomly drop the conditions (e.g., text, the contextual features) in both flow models with probability p^H and p^E in training, whose values are also tuned.

4. Experiments

4.1. Experimental Setup

Data We use 20K hours of proprietary music data (∼400K tracks) to train our model. We follow the original recipe in (Hsu et al., 2021) to train the music HuBERT model with music data. For data preprocessing, we filter out all the vocal tracks and resample all the data to 32kHz and perform channel-wise averaging to downmix all multi-channel music into mono. Only text descriptions are retained for training, while the other metadata such as genre, BPM and music tags are discarded. We evaluate our model on MusicCaps (Agostinelli et al., 2023), which incorporates 5.5K 10s-long audio samples annotated by expert musicians in total. For subjective evaluation, we use the 1K genre-balanced subset following (Agostinelli et al., 2023).

Implementation details We follow (Le et al., 2023) for backbone architectures in both stages, which are Transformers (Vaswani et al., 2017) with convolutional position embeddings (Baevski et al., 2020), symmetric bi-directional ALiBi self-attention bias (Press et al., 2021) and UNet-style skip connections. Specifically, the transformers of the first and second stage include 8 and 24 layers of 12 attention heads with 768/3072 embedding/feed-forward network (FFN) dimension, leading to 84M and 246M parameters (see Section 4.4.2 for ablation on model size). The models are trained with an effective batch size of 480K frames, for 300K/600K updates in two stages respectively. For efficiency, audios are randomly chunked to 10s during training. For masking, we adopt the span masking strategy and the masking ratio is randomly chosen between $70-100\%$. Condition dropping probabilities (i.e., p^H and p^E) are 0.3 for both stages. We use the Adam optimizer (Kingma & Ba, 2014) with learning rate 2e-4, linearly warmed up for 4k steps and decayed over the rest of training.

Objective evaluation We evaluate the model using the standard Frechet Audio Distance (FAD) (Kilgour et al., 2019), Frechet Distance (FD) and KL divergence (KLD) based on the pre-trained audio event tagger PANN (Kong et al., 2019), and Inception score (ISc) (Salimans et al., 2016), which are adapted from sound generation and has been widely used in prior works for text-to-music generation (Agostinelli et al., 2023; Huang et al., 2023; Copet et al., 2023; Li et al., 2023a). Specifically, FAD and FD measure distribution-level similarity between reference samples and generated samples. KLD is an instance level metric computing the divergence of the acoustic event posterior between the reference and the generated sample for a given description. The metrics are calculated using the `audioldm_eval` toolkit.[1]. To measure how well the generated music matches the text description, we use CLAP [2] similarity, defined as the cosine similarity between audio and text embeddings.

Subjective evaluation In addition to the objective metrics mentioned above, we further conduct subjective evaluation with human annotators. The study consists of multiple pairwise studies following the evaluation protocol of Agostinelli et al. (2023). Specifically, each human annotator is presented with pairs of audio clips generated by two different systems and is required to give their preference based on how well the generated music captures the elements in the text description.

4.2. Main Results

Table 1 compares our model to prior works in text-to-music generation on MusicCaps in terms of objective metrics. Given the variation in models used for evaluation in prior works, we primarily rely on FAD, which is computed using the vggish feature (Kilgour et al., 2019) and serves as a unified benchmark across different studies. Specifically, when evaluating MusicGen, we opt for its medium version due to its overall superior performance compared to other variants (Copet et al., 2023). For MusicGen and AudioLDM2, we use the public model checkpoints in order to get FD, ISc and CLAP similarity since these metrics were not reported in the paper.

[1] https://github.com/haoheliu/audioldm_eval

[2] We use the `music_speech_epoch_15_esc_89.25.pt` checkpoint, trained on both speech and music data.

Table 1. Comparisons between MusicFlow with previous works in text-to-music generation on the MusicCaps dataset.

MODEL	# PARAMS	FAD(↓)	FD(↓)	KL-DIV(↓)	ISC.(↑)	CLAP-TEXT(↑)
MUSICLM (AGOSTINELLI ET AL., 2023)	860M	4.00	-	-	-	-
MUSICGEN (COPET ET AL., 2023)	1.5B	3.40	24.1	1.23	2.29	0.37
UNIAUDIO (YANG ET AL., 2023)	1B	3.65	-	1.90	-	-
AUDIOLDM-2 (LIU ET AL., 2023B)	746M	3.13	18.8	**1.20**	2.77	0.43
NOISE2MUSIC (HUANG ET AL., 2023)	1.3B	2.10	-	-	-	-
JEN-1 (LI ET AL., 2023A)	746M	**2.00**	-	1.29	-	-
MUSICFLOW (UNIDIRECTIONAL LM + FM)	546M	2.69	**13.2**	1.23	2.69	0.52
MUSICFLOW (BIDIRECTIONAL FM + FM)	330M	2.82	14.2	1.23	**2.78**	**0.56**

In MusicFlow, we additionally present the results of a model with the first stage functioning as a language model, predicting HuBERT units as detailed in Section 3. The language model includes 24 transformer layers, 16 attention heads, and a hidden dimension of 1024, leading to $\sim 300M$ parameters in total.

In comparison to all prior works, our model exhibits a significant reduction in size, with parameter reduction ranging from 50% to 80%, while remaining competitive in terms of generation quality. Compared with a standard diffusion model - AudioLDM-2, MusicFlow achieves a 10% lower FAD ($3.13 \rightarrow 2.82$) with approximately 50% fewer parameters. Similarly, compared to the language-model-based MusicGen, our approach shows a 20% improvement in FAD ($3.40 \rightarrow 2.82$) while using only 20% of the parameters. These results highlight the efficiency of our approach.

It's noteworthy that MusicLM (Agostinelli et al., 2023) shares similarities with ours, incorporating semantic and acoustic modeling stages based on language models. However, we surpass this approach by roughly 30% in FAD with less than 65% of the parameters. Additionally, in contrast to the current state-of-the-art model on MusicCaps, Jen-1 (Li et al., 2023a), our results shows a mixture of results. While falling behind in FAD, we outperform it in KL divergence with only half of the parameters.

LM vs. FM for first stage In addition to our main approach, we investigate the integration of a language model for first-stage modeling. Both approaches share the second-stage model. According to the last two rows of table 1, using a first-stage LM yields marginally superior results compared to using a flow matching model. This implies that semantic features in music audios possess discrete structures, which can be well captured by an auto-regressive language model. Nonetheless, for the sake of model efficiency and task generalization, we adhere to using the flow matching cascade moving forward.

Subjective evaluation Figure 4.2 shows the pairwise comparison between our model and prior works. In particular, we compare MusicFlow to AudioLDM2 and MusicGen, which are the only two publicly available models in Table 1. For our model, we use the `bidirectional FM+FM` configuration in Table 1. Our model surpasses both AudioLDM2 and MusicGen. This observation aligns with the objective metrics presented in Table 1. However, it's worth noting that there is still a gap between our model and the ground-truth.

Inference Efficiency In Table 1, we only lists the model size, which is one aspect of model efficiency. In Figure 3, we plot how FAD changes when we vary the number of function evaluations (NFE) during inference. For flow matching and AudioLDM2, this is achieved by adjusting the number of iterative steps in the ODE solver[3] and DDIM steps, respectively. Since MusicFlow involves two flow matching models, we simply aggregate the NFE of the two modules as the final NFE we plot. For comparison, we further show the MusicGen, which runs a fixed number of auto-regressive steps. As shown in Figure 3, MusicFlow outperforms MusicGen (FAD: 3.13 vs. 3.40) by using 20% of inference steps. Running with longer steps further improves the performance. The final model takes only 50% the network forward passes of MusicGen. AudioLDM2 exhibits a similar trend to ours, although its generation quality consistently lags behind with the same number of inference steps.

4.3. Infilling and Continuation

One advantage of MusicFlow is its ability to handle multiple audio-conditioned generative tasks, such as infilling and continuation, with a single model. These tasks have also been explored in (Li et al., 2023a), albeit without reported quantitative metrics. Due to lack of baselines, we compare the model performance to the our own text-to-music model, as detailed in Table 1. For the infilling task, we infill the middle 70% of the audio segment. For the continuation task, given the beginning 30% of the audio clip, the model generates the remaining 70%.

[3]We use the `midpoint` solver for this analysis

Table 2. Performance of MusicFlow on various music generation tasks on MusicCaps dataset. We compare with AudioLDM-2 (Liu et al., 2023b) for text-to-music and AudioLDM for music infilling and continuation.

TASK // MODEL	FAD(↓)	FD(↓)	KL-DIV(↓)	ISC.(↑)	CLAP-SIM(↑)	CLAP-AUDIO(↑)	CLAP-TEXT(↑)
TEXT-TO-MUSIC (100%)							
AUDIOLDM-2 (LIU ET AL., 2023B)	3.13	18.8	**1.20**	2.77	-	0.44	0.43
MUSICFLOW	**2.82**	**14.2**	1.23	**2.78**	-	**0.48**	**0.56**
CONTINUATION (LAST 70%)							
AUDIOLDM (LIU ET AL., 2023A)	2.08	25.08	0.66	2.80	0.61	0.61	0.53
MUSICFLOW	**1.63**	**6.50**	**0.49**	**3.37**	**0.88**	**0.77**	**0.56**
INFILLING (MIDDLE 70%)							
AUDIOLDM (LIU ET AL., 2023A)	2.09	45.93	0.76	2.39	0.59	0.61	0.54
MUSICFLOW	**1.71**	**6.5**	**0.38**	**3.18**	**0.89**	**0.79**	**0.57**

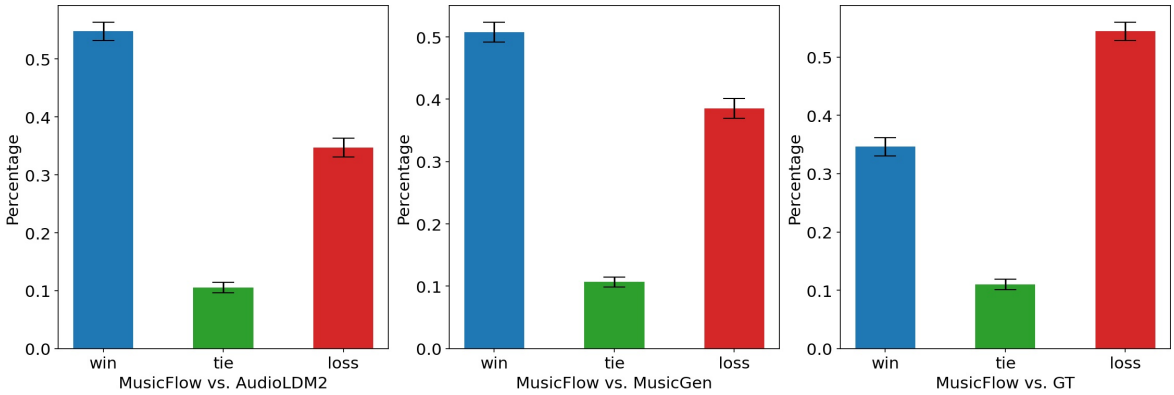

Figure 2. Pairwise comparison between MusicFlow, AudioLDM2, MusicGen and ground-truth

As is shown in Table 4.2, our model effectively uses the context to enhance audio generation. In both settings, using a 3s audio context enables a nearly 50% reduction in FAD. The text-to-audio similarity is slightly increased in infilling ($0.44 \rightarrow 0.45$). We hypothesize this may be because the CLAP model struggles to discern fine-grained details in the text description. Hence, we conduct a subjective study to measure text faithfulness. The MOS scores of text-to-music, conditnuation and infilling are respectively 3.34 ± 0.18, 3.47 ± 0.18, 3.42 ± 0.19 with 95% confidence interval. This confirms an improvement in text faithfulness through context utilization.

Additionally, among other metrics, we compute the CLAP-Audio score, defined as the cosine similarity between the embeddings of the generated and ground-truth audios. Compared to text-only generation, the generated audio achieved higher scores, suggesting better acoustic matching through context conditioning. Finally, we measure the CLAP similarity between the generated segment and the original context (CLAP-SIM). Both settings achieve scores close to 1, implying coherence between the generation and context.

4.4. Ablation Study

Below we analyze the impact of different design choices in MusicFlow, particularly focusing on the necessity of a two-stage cascade and how model scales differently in each stage.

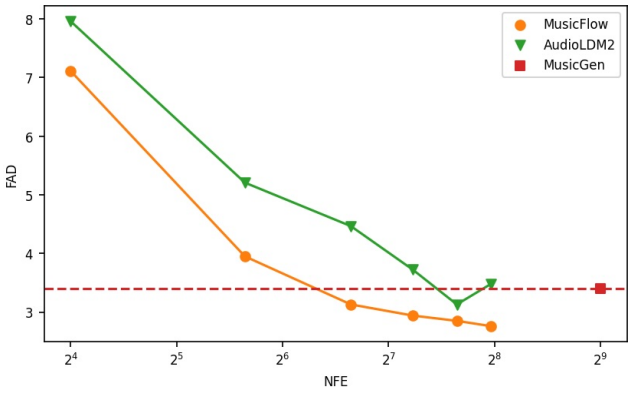

Figure 3. Comparison between MusicFlow and prior works in FAD-NFE in terms of inference efficiency.

4.4.1. SINGLE-STAGE VS. TWO-STAGE MODEL

We compare MusicFlow to a simple flow matching baseline of directly generating music based on text descriptions with-

out the intermediate HuBERT features in Table 3. Including a stage of HuBERT prediction consistently improves the performance across various metrics regardless of model size. HuBERT-based flow matching brings a $\sim 30\%$ relative improvement in terms of FAD. Note while we increased the size of the single-stage model to 431M, it did not yield additional gains, despite having more parameters.

Table 3. Comparison between single-stage and multi-stage flow matching in different model sizes

MODEL	FAD	FD	KL-DIV	ISC.
SINGLE-STAGE (123M)	4.58	25.5	1.62	2.52
SINGLE-STAGE (246M)	4.52	22.9	1.57	2.66
SINGLE-STAGE (431M)	5.11	27.5	1.64	2.68
TWO-STAGE (84M+123M)	3.37	20.6	1.50	2.59
TWO-STAGE (84M+246M)	**2.82**	**14.2**	**1.23**	**2.78**

4.4.2. EFFECT OF MODEL SIZE

Empirically, we observed the performance of our models is heavily influenced by the model size. In this analysis, we delve into the impact of model size in each stage.

Second-stage: Text + HuBERT to Music. We first examine how the size of the second-stage model, specifically the Text + HuBERT features $\rightarrow$ music, affects the overall performance. We keep the best first-stage model and scale the second-stage model by altering the number of transformer layers and the hidden dimension of each layer (see Table 4). The performance improves as we increase the number of layers until reaching 24 layers. Beyond this point, increasing the number of layers or feature dimensions results in degradation, suggesting a potential overfitting issue of the model.

First-stage: Text to HuBERT. We fix the configuration for our second-stage model based on the above findings and vary only the first-stage configuration (see Table 5). Unlike the second-stage, where the best model is with 24 transformer layers, our best first-stage model for Text $\rightarrow$ HuBERT feature prediction is notably smaller with an optimal configuration of only 8 layers. According to Table 5, smaller models typically perform equally well or even better than their larger counterparts in the first-stage model. We hypothesize that predicting HuBERT features is simpler than predicting the low-level Encodec features, particularly for shorter music pieces with standard music structures, as the former consists of learning only the coarse-grained semantics. Consequently, a larger variant is more susceptible to overfitting compared to the second-stage scenario.

Table 4. Effect of the *second-stage* model size on performance. In each row we specify the number of layers, the hidden dimension of the transformer and the total number of trainable parameters.

MODEL CONFIGURATION	FAD	FD	KL-DIV	ISC.
12L, 768D (123M)	3.37	20.6	1.50	2.59
18L, 768D (123M)	3.22	18.2	1.42	2.62
24L, 768D (246M)	**2.82**	**14.2**	**1.23**	**2.78**
32L, 768D, (323M)	3.12	17.9	1.42	2.64
12L, 1024D, (217M)	3.56	18.7	1.43	2.67
18L, 1024D, (324M)	3.26	18.4	1.42	2.67
24L, 1024D, (441M)	3.40	17.8	1.43	2.71

Table 5. Effect of the *first-stage* model size on performance. In each row we specify the number of layers, the hidden dimension of the transformer and the total number of trainable parameters.

MODEL CONFIGURATION	FAD	FD	KL-DIV	ISC.
12L, 1024D (217M)	3.18	18.2	1.44	2.74
12L, 768D (123M)	3.09	17.1	1.42	2.73
8L, 768D (84M)	**2.82**	**14.2**	**1.23**	**2.78**
6L, 768D, (64M)	3.30	18.1	1.47	2.69
8L, 512D, (38M)	3.20	17.6	1.41	2.76

4.4.3. NUMBER OF TRAINING ITERATIONS

We notice the performance of both stages in MusicFlow is sensitive to the number of training iterations. Generally, longer training boosts performance, as can be seen from Table 6. While varying the number of training iterations, we maintains the sizes of best models from Table 5 and 4. Comparing the two stages, longer training consistently enhances performance in the second stage, while there is a degradation in performance with further increases in training iterations in the first stage. This aligns with our observations in model scaling, which highlight the different tendencies of model overfitting in both stages.

Table 6. Impact of training steps on the model performance

STAGE 1	STAGE 2	FAD	FD	KL-DIV	ISC.
100K	600K	3.60	18.1	1.42	2.54
200K		3.00	17.7	1.45	2.79
300K		**2.82**	**14.2**	**1.23**	**2.78**
400K		2.90	16.3	1.39	2.85
300K	200K	3.19	19.3	1.42	2.51
	400K	2.84	16.6	*1.39*	2.71
	600K	**2.82**	**14.2**	**1.23**	**2.78**

4.4.4. Choice of Semantic Latent Representation

The first stage model predicts semantic latent representations conditioned on text tokens. The choice of the semantic latents has an impact on the final performance. In addition to HuBERT units, we also experiment with MERT units (Li et al., 2023b) using the officially released pre-trained music model. In Table 7, we can see that it is clearly worse compared to using HuBERT units.

Table 7. Impact of using a different semantic latent representation instead of HuBERT. We compare with MERT (Li et al., 2023b) units below.

Semantic Latent	FAD	FD	KL-div	ISc.
MERT	3.43	18.3	1.47	2.54
HuBERT	**2.82**	**14.2**	**1.23**	**2.78**

4.4.5. Choice of Acoustic Latent Representation

The second stage model predicts acoustic latent representations from the semantic latent features. The choice of the acoustic latents also affects the final performance. In addition to Encodec, we also experiment with the recently proposed UniAudio tokenizer (Dongchao et al., 2023) and DAC (Kumar et al., 2024). We could not achieve convergence with DAC, and found UniAudio to perform slightly worse compared to Encodec in terms of all quantitative metrics. We report the results in Table 8.

Table 8. Impact of using a different acoustic latent representation instead of Encodec. We compare with UniAudio (Dongchao et al., 2023) below. We could not achieve convergence with DAC (Kumar et al., 2024)

Acoustic Latent	FAD	FD	KL-div	ISc.
DAC	-	-	-	-
UniAudio	3.18	18.2	1.44	2.74
Encodec	**2.82**	**14.2**	**1.23**	**2.78**

5. Conclusion

We present MusicFlow, a cascaded flow-matching network for text-guided music generation. Our model leverages a self-supervised model to capture semantic information within music audio. Comprising two flow matching networks that predict semantic and acoustic features in a cascaded manner, MusicFlow consistently outperforms all public text-to-music models in both subjective and objective metrics, with only a fraction of model parameters and inference steps. Overall, MusicFlow achieves performance on par with the state-of-the-art models while being significantly smaller. Additionally, our model allows text-guided music continuation and infilling through in-context learning, eliminating the need for task-specific training. Our future work includes further improving model efficiency by using sophisticated ODE solvers such as (Shaul et al., 2023).

Impact Statement

While music generation technologies make music creation more accessible to amateur creators, they also pose potential societal challenges. Given that modern music generation models often require substantial data, preventing copyright infringement deserves careful attention. In this work, we ensure the use of music data for model training adheres to legal terms. For future data scaling, it's essential to inform artists of data usage and provide opt-out options, as commonly practiced in concurrent music generation works. Furthermore, we acknowledge the lack of diversity in our model generations, potentially stemming from the predominantly stock music training data with limited world music. Our future objective is to ensure high-quality music generation across diverse genres.

References

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

Baevski, A., Zhou, H., rahman Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *ArXiv*, abs/2006.11477, 2020. URL https://api.semanticscholar.org/CorpusID:219966759.

Benetos, E., Dixon, S., Duan, Z., and Ewert, S. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36:20–30, 2019. URL https://api.semanticscholar.org/CorpusID:57191022.

Bittner, R. M., Bosch, J. J., Rubinstein, D., Meseguer-Brocal, G., and Ewert, S. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 781–785, 2022. URL https://api.semanticscholar.org/CorpusID:247595162.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. Simple and controllable music generation. *arXiv preprint arXiv:2306.05284*, 2023.

Dongchao, Y., Jinchuan, T., Xu, T., Rongjie, H., Songxiang, L., Xuankai, C., Jiatong, S., Sheng, Z., Jiang, B., Xixin,

W., Zhou, Z., and Helen, M. Uniaudio: An audio foundation model toward universal audio generation. *arXiv preprint arXiv:2310.00704*, 2023.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Evans, Z., Carr, C., Taylor, J., Hawley, S. H., and Pons, J. Fast timing-conditioned latent audio diffusion. *arXiv preprint arXiv:2402.04825*, 2024.

Forsgren, S. and Martiros, H. Riffusion-stable diffusion for real-time music generation. 2022. URL https://riffusion.com/about.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. Enabling factorized piano music modeling and generation with the maestro dataset. In *ICLR*, volume abs/1810.12247, 2018. URL https://api.semanticscholar.org/CorpusID:53094405.

Hawthorne, C., Simon, I., Swavely, R., Manilow, E., and Engel, J. Sequence-to-sequence piano transcription with transformers. In *International Society for Music Information Retrieval Conference*, 2021. URL https://api.semanticscholar.org/CorpusID:236134377.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Hsu, J.-Y. and Su, L. VOCANO: A note transcription framework for singing voice in polyphonic music. In *Proc. International Society of Music Information Retrieval Conference (ISMIR)*, 2021.

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and rahman Mohamed, A. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021. URL https://api.semanticscholar.org/CorpusID:235421619.

Huang, Q., Park, D. S., Wang, T., Denk, T. I., Ly, A., Chen, N., Zhang, Z., Zhang, Z., Yu, J., Frank, C., et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.

Hung, H.-T., Wang, C.-Y., Yang, Y.-H., and Wang, H.-M. Improving automatic jazz melody generation by transfer learning techniques. *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 339–346, 2019. URL https://api.semanticscholar.org/CorpusID:201666995.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Interspeech*, 2019.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., and Plumbley, M. D. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 2019.

Kreuk, F., Synnaeve, G., Polyak, A., Singer, U., D'efossez, A., Copet, J., Parikh, D., Taigman, Y., and Adi, Y. Audiogen: Textually guided audio generation. *ArXiv*, abs/2209.15352, 2022. URL https://api.semanticscholar.org/CorpusID:252668761.

Kumar, R., Seetharaman, P., Luebs, A., Kumar, I., and Kumar, K. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36, 2024.

Lam, M. W., Tian, Q., Li, T., Yin, Z., Feng, S., Tu, M., Ji, Y., Xia, R., Ma, M., Song, X., et al. Efficient neural music generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Le, M., Vyas, A., Shi, B., Karrer, B., Sari, L., Moritz, R., Williamson, M., Manohar, V., Adi, Y., Mahadeokar, J., and Hsu, W.-N. Voicebox: Text-guided multilingual universal speech generation at scale. In *NeurIPS*, 2023.

Li, P., Chen, B., Yao, Y., Wang, Y., Wang, A., and Wang, A. Jen-1: Text-guided universal music generation with omnidirectional diffusion models. *arXiv preprint arXiv:2308.04729*, 2023a.

Li, Y., Yuan, R., Zhang, G., Ma, Y., Chen, X., Yin, H., Lin, C., Ragni, A., Benetos, E., Gyenge, N., Dannenberg, R., Liu, R., Chen, W., Xia, G., Shi, Y., Huang, W., Guo, Y., and Fu, J. Mert: Acoustic music understanding model with large-scale self-supervised training, 2023b.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *ICLR*, 2023.

Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D. AudioLDM: Text-to-audio generation with latent diffusion models. *Proceedings of the International Conference on Machine Learning*, 2023a.

Liu, H., Tian, Q., Yuan, Y., Liu, X., Mei, X., Kong, Q., Wang, Y., Wang, W., Wang, Y., and Plumbley, M. D. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *arXiv preprint arXiv:2308.05734*, 2023b.

Liu, X., Zhu, Z., Liu, H., Yuan, Y., Huang, Q., Liang, J., Cao, Y., Kong, Q., Plumbley, M. D., and Wang, W. Wavjourney: Compositional audio creation with large language models. *arXiv preprint arXiv:2307.14335*, 2023c.

Luo, S., Yan, C., Hu, C., and Zhao, H. Diff-foley: Synchronized video-to-audio synthesis with latent diffusion models. In *CVPR*, volume abs/2306.17203, 2023. URL https://api.semanticscholar.org/CorpusID:259309037.

Müller, M. *Fundamentals of Music Processing*. 01 2015. ISBN 978-3-319-21944-8. doi: 10.1007/978-3-319-21945-5.

Pasad, A., Shi, B., and Livescu, K. Comparative layer-wise analysis of self-supervised speech models. In *ICASSP*, 2023.

Pasini, M., Lattner, S., and Fazekas, G. Self-supervised music source separation using vector-quantized source category estimates. *ArXiv*, abs/2311.13058, 2023. URL https://api.semanticscholar.org/CorpusID:265351916.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *ArXiv*, abs/2108.12409, 2021. URL https://api.semanticscholar.org/CorpusID:237347130.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *ArXiv*, abs/1606.03498, 2016. URL https://api.semanticscholar.org/CorpusID:1687220.

Schneider, F., Jin, Z., and Schölkopf, B. Moûsai: Text-to-music generation with long-context latent diffusion. 01 2023. doi: 10.48550/arXiv.2301.11757.

Shaul, N., Perez, J., Chen, R. T. Q., Thabet, A., Pumarola, A., and Lipman, Y. Bespoke solvers for generative flow models. *arXiv:2310.19075*, 2023. URL https://arxiv.org/abs/2310.19075.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *JMLR*, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv:2010.02502*, October 2020. URL https://arxiv.org/abs/2010.02502.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Su, T.-W., Chen, Y.-P., Su, L., and Yang, y.-h. Tent: Technique-embedded note tracking for real-world guitar solo recordings. *Transactions of the International Society for Music Information Retrieval*, 2:15–28, 07 2019. doi: 10.5334/tismir.23.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

Vyas, A., Shi, B., Le, M., Tjandra, A., Wu, Y.-C., Guo, B., Zhang, J., Zhang, X., Adkins, R., Ngan, W., Wang, J., Cruz, I., Akula, B., Akinyemi, A. T., Ellis, B., Moritz, R., Yungster, Y., Rakotoarison, A., Tan, L., Summers, C., Wood, C., Lane, J., Williamson, M., and Hsu, W.-N. Audiobox: Unified audio generation with natural language prompts. 2023. URL https://api.semanticscholar.org/CorpusID:266551778.

Wang, C., Chen, S., Wu, Y., Zhang, Z.-H., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang, H., Li, J., He, L., Zhao, S., and Wei, F. Neural codec language models are zero-shot text to speech synthesizers. *ArXiv*, abs/2301.02111, 2023. URL https://api.semanticscholar.org/CorpusID:255440307.

Yang, D., Tian, J., Tan, X., Huang, R., Liu, S., Chang, X., Shi, J., Zhao, S., Bian, J., Wu, X., Zhao, Z., and Meng, H. Uniaudio: An audio foundation model toward universal audio generation. *arXiv preprint arXiv:2310.00704*, 2023.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. Soundstream: An end-to-end neural audio codec. 2021.

Zheng, Q., Le, M., Shaul, N., Lipman, Y., Grover, A., and Chen, R. T. Q. Guided flows for generative modeling and decision making. *ArXiv*, abs/2311.13443, 2023. URL https://api.semanticscholar.org/CorpusID:265351587.

Ziv, A., Gat, I., Lan, G. L., Remez, T., Kreuk, F., Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. Masked audio generation using a single non-autoregressive transformer. *arXiv:2401.04577*, 2024. URL https://arxiv.org/abs/2401.04577.