MusicGen-Stem: Multi-stem music generation and edition through autoregressive modeling

Simon Rouard, Robin San Roman, Yossi Adi, Axel Roebel

MusicGen-Stem: Multi-stem music generation and edition through autoregressive modeling

Simon Rouard*
Meta & UMR STMS
IRCAM-CNRS-Sorbonne Univ.

Robin San Roman*
Meta, FAIR, Univ. de
Lorraine, CNRS, Inria, Loria

Yossi Adi
Meta & Hebrew
Univ. of Jerusalem

Axel Roebel
UMR STMS, IRCAM-CNRS
Sorbonne Univ.

***Abstract*—While most music generation models generate a mixture of stems (in mono or stereo), we propose to train a multi-stem generative model with 3 stems (bass, drums and other) that learn the musical dependencies between them. To do so, we train one specialized compression algorithm per stem to tokenize the music into parallel streams of tokens. Then, we leverage recent improvements in the task of music source separation to train a multi-stream text-to-music language model on a large dataset. Finally, thanks to a particular conditioning method, our model is able to edit bass, drums or other stems on existing or generated songs as well as doing iterative composition (e.g. generating bass on top of existing drums). This gives more flexibility in music generation algorithms and it is to the best of our knowledge the first open-source multi-stem autoregressive music generation model that can perform good quality generation and coherent source editing. Code and model weights will be released and samples are available on simonrouard.github.io/musicgenstem.**

***Index Terms*—Music editing, Generative models**

I. Introduction

Recent models for music generation [1]–[4] allow generating long and coherent audio sequences of up to several minutes with reasonable audio quality. Although recent studies provide rich conditions for controlling the generated music [4], [5], the dominant approach is still text instructions. While these text prompts have the benefit of allowing extensive control over high-level musical characteristics of the generated music (like style, instrumentation, or mood), they remain limited with respect to precise control, specifically editing. For example, the production of a drum track for a given music piece can not be achieved using only a textual description of the result.

These limitations have led to research activities that aim extending the use cases for music generation models. One line of research addresses the generation of musically coherent stems (or tracks) conditioned on audio input [6]–[9]. If successful, these approaches would provide innovative means for the generation of musical accompaniment, for the iterative, stem-wise creation of a musical piece, and in combination with source separation, may allow replacing a stem in a given musical piece.

In line with these recent research activities the present work introduces MusicGen-Stem, an extension of [1] towards multi-stem music generation that is able to perform at the same time text-to-music generation, text and audio conditioned stem generation, as well as iterative stem-by-stem generation. One of the main problems for training music generation for individual stems is the training data. Here we follow [6] and use one of the state-of-the-art music source separation models [10] to produce *bass*, *drum* and *other* stems. Note that we intentionally exclude vocal stems from the present study, on one hand to avoid the complexity of the generation of coherent lyrics, and on the other hand due to legal constraints.

The proposed methods allow for several use-cases including: (i) Generate music given a textual prompt and directly obtain 3 separated stems (*bass*, *drums* and *other*); (ii) Generate the complementary stems (e.g. add the *drums* and *other* instruments on top of the *bass*) given one or multiple stems (e.g. a *bass*). Here again the generation may be controlled by means of an additional text prompt; (iii) Remove and regenerate one or more of the three stems in an existing song; (iv) Modify the sound texture of the *other* stem by regenerating its RVQ residuals while keeping its first stream fixed (see III-C).

Our contributions are as follows: (i) We introduce MusicGen-Stem a variant of the autoregressive text-to-music model MusicGen [1] that allows generating the three different stems that are used in the present study. The proposed method can generate all stems at once, or individual stems conditioned on a given musical sample; (ii) To prevent the possibility of cross-talk across the stems, we propose to use specialized compression models that are used to tokenize the individual stems; (iii) We evaluate on the text-to-music task and despite the additional complexity of the parallel generation of multiple stems, the proposed model is on par with its predecessor [1]. Additionally, on the unconditional generation task, it outperforms all the previous multi-stem generative models on objective and subjective metrics; (iv) We introduce a particular conditioning approach that allows our model to perform stem editing (replace an existing stem) and stem by stem generation. Our evaluations show that our approach compares favorably to the previous open-source models that have been proposed for the task of stem editing.

II. Related Work

A. Music generation models

One of the pioneering works in music generation was Wavenet [11], which introduced an autoregressive model for predicting the next sample in a quantized signal. This approach was initially inefficient during the sampling stage due to

*Equal contribution

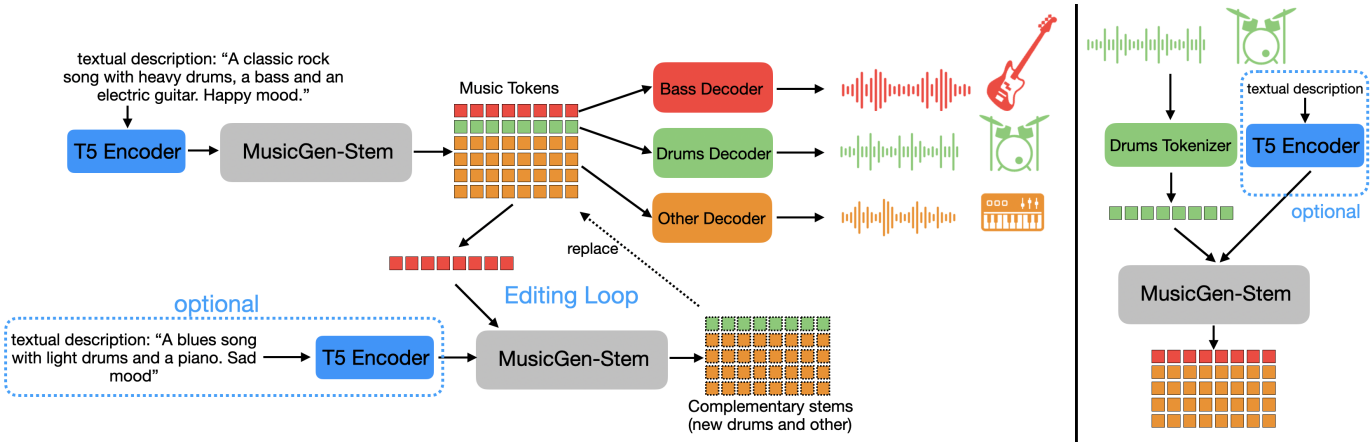

Fig. 1: Three use-cases of our model: **(up)** MusicGen-Stem can perform text-to-music generation and generates parallel streams of tokens representing the 3 stems (*bass, drums and other*). **(down)** MusicGen-Stem can also perform stem editing: given a subgroup of stems, the model can generate the complementary ones with an optional text prompt. **(right)** Given the waveform of one or multiple stems (that can be extracted from an existing song with Demucs), we tokenize them and MusicGen-Stem can generate the missing stems with an optional text prompt. We can then decode them.

the high dimensionality of audio. The development of neural compression models [12]–[14] addressed this issue by representing one second of audio with a few hundreds audio tokens. Models such as MusicLM [3] and MusicGen [1] leveraged this technique to build autoregressive models capable of generating coherent long-form music conditioned on text, audio, or even melodies. The authors in [15], propose an improved optimization process in the training of the compression model to obtain better tokenization for autoregressive modeling. Other approaches [16], [17] have employed these compression models in a non-autoregressive manner, reducing latency but often at the expense of less convincing results.

Concurrently, diffusion models have established new standards in image generation [18], with the advent of latent diffusion models [19] specifically tackling the challenge of high-dimensional data. In the audio domain, methods like MusicLDM [20], AudioLDM [21] and Stable Audio [2] exemplify these advancements, with the latter being capable of generating up to 95 seconds of music. These methods operate on the intermediate representations of autoencoders, similar to compression models but without a quantization stage, thereby achieving higher fidelity.

B. Music editing models

Techniques of **zero-shot editing** such as DDPM inversion [22], [23] and DDIM inversion [24] illustrate the potential of diffusion models to provide more flexibility and control in music generation. However, when attempting fine editing of a single instrument from a song, those approaches struggle to keep the rest of the track unchanged. This indicates that single stem models are not suited to the needs of real world artists.

Test-time optimization methods as well do not require training a model from scratch. For instance, textual inversion has been applied for diffusion [25] and autoregressive [26] models, where given a pretrained frozen text-to-music model and a batch of audio that share similarities (e.g. same artist, style or instruments), a "pseudoword" in the text embedding space representing these similarities is obtained by doing a gradient descent on the "pseudoword" by optimizing the loss of the model. Still, these inversion methods often result in artefacts in the generated audio.

However, discrete models based on quantized autoencoders lack of flexibility for editing. The autoregressive ones can regenerate the end of a song by using its beginning as a prompt but they cannot perform inpainting or stem editing which is crucial when one wants to modify specific sections of music without altering the entire piece.

To enable an autoregressive model to perform editing, instruction tuning can be done. For instance, in Instruct-MusicGen [27], the authors fine-tune a pretrained MusicGen model with a source separation dataset and instructions in order to perform adding, removing, extracting and replacing instruments. This method is limited to 5 seconds generation and we observe that the task of stem editing often fails to keep the remaining stems unchanged because of the non separation of the instruments in the streams of the model.

C. Multi-stem music generation and editing

In SingSong [6], the authors perform vocal to accompaniment generation by conditioning an autoregressive MusicLM [3] model with the coarse tokens of the compressed vocals stem. In StemGen [7], the authors sequentially generate music stem by stem with a masked model transformer. The drawbacks of their model is that it needs a first stem as an input and then generates music stem by stem which is compute intensive. In Jen-1 Composer [8] as well as in Multi-source diffusion models [9] (MSDM), the authors use a source separated dataset to train a diffusion model that outputs 4 stems in parallel. Jen-1 composer is a latent diffusion model whereas the second one is a diffusion model in the waveform

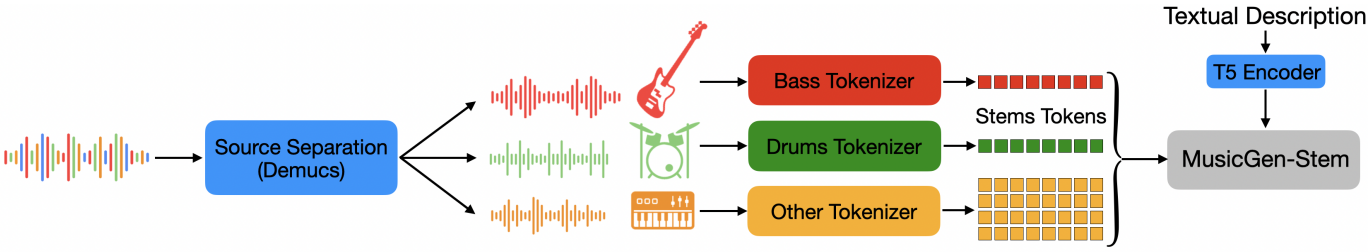

Fig. 2: Training pipeline. Given a song paired with its textual description, we process the song by using the source separation model Demucs and tokenize each stem with specific compression models. There is one stream of token for the *bass* as well as the *drums* and 4 streams of tokens for the *other* instruments. Then, these tokens as well as the encoded textual description are fed into MusicGen-Stem's autoregressive transformer which is trained with a cross-entropy loss.

space. In [28], the authors train a latent diffusion model on a source separated dataset with a large set of instruments in an iterative manner. The only open-source model is MSDM [9].

III. Method

In this section, we provide a detailed description of MusicGen-Stem. We start by describing the compression models. Next, we describe the auto-regressive sequential model. Lastly, we present the editing method. Fig. 2 describes the training pipeline.

A. Compression models

For each stem (*drums, bass, other*), we train a compression model similar to EnCodec [13] that compresses 32kHz mono music into tokens at a rate of 50Hz. For the *drums* and the *bass* stem, we obtain good reconstruction quality with only a single quantization level. For the *other* stem, we trained a model with 4 RVQ streams. Each of these specialized models is trained on our internal source separation dataset which consists of 3,000 professionally recorded songs.

B. Modeling and data preparation

We train an autoregressive transformer for the task of text-to-music generation using cross entropy loss on 30 seconds audio segments at 50Hz. The 3 different stems are tokenized thanks to the 3 different compression models and their streams are concatenated and modelled in parallel. We use the medium (1.5B) architecture of MusicGen's transformer [1] and apply its delay pattern to the tokens. In our setup *bass, drums* and the first RVQ *other* stream are "coarse" tokens and in sync. Thus we only apply a delay on the 3 residual streams of the *other* stem (see Fig. 3). Given the fact that we do not have a big dataset of labeled stem music, we train our model on the same data as in [1] but we removed the songs that had vocals (15% of our data) and use the last version of Demucs [10] to separate all the songs into 3 stems (*bass, drums* and *other*).

C. Editing

At each training step we either train our model to perform text-to-music generation or editing with a 0.5 probability. To train the editing task, we take a sequence of 25 seconds (1,250 tokens), downsample it by a factor 5 (i.e. 10Hz) and

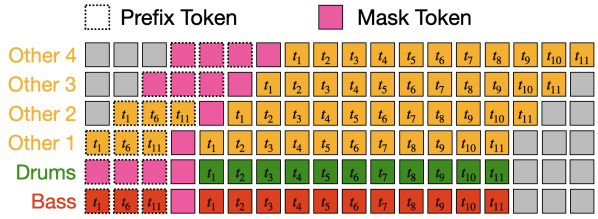

Fig. 3: Training the editing task. Here the *drums* and the 2 last streams of the *other* stem are masked. The cross-entropy loss is computed on the tokens on the right of the masked tokens.

use these 250 tokens as a prefix for the model. Then, we randomly sample 1 or 2 stems and mask the associated tokens in the prefix. If the *other* stem is selected to be masked, we randomly choose to mask the streams in $\{4\}, \{4,3\}, \{4,3,2\}$, or $\{4,3,2,1\}$, forcing the model to learn to generate the details (the streams 2, 3, 4) of the *other* stem given its first streams. We can see an example of a prefix on Fig. 3 where the *drums* is masked as well as the streams 3 and 4 of the *other* stem.

At inference time, we can 1) edit a song generated by the model (it is then already tokenized) 2) take an existing song separate its stems with Demucs and tokenize them 3) tokenize single stem music to be able to generate new stems. Then, we downsample to 10Hz this tokens sequence, we mask the desired stem and ask the model to continuous the generation in an autoregressive manner. During the autoregressive generation, we have the choice to force the unmasked streams to be exactly the same and only generate the masked streams or to let the model generate all of the streams. In the second case, we obtain a variation of the original unmasked stems (the model uses the downsampled prefix to reconstruct the stem). As well, the textual prompt let us control the generated new stems.

IV. Experiments

A. Training details

MusicGen-Stem is trained for 400K steps with a batch size of 128. The data used for training include the internal MMI dataset that contains 10k high quality songs, ShutterStock and Pond5. We filtered the dataset to remove all songs containing vocals, resulting in a total of 17K hours of instrumental only data. We use AdamW optimizer with a learning rate of 1e-4.

B. Metrics

We evaluate the proposed method against state-of-the-art music generation models, considering both music generation and stem-editing setups. All the objective evaluations are performed on an internal test set of 534 songs for which we used Demucs to separate the stems.

Music generation: We use established objective and subjective metrics from the literature. Specifically the objective metrics used are: Frechet Audio Distance [29], the KL-divergence based metric introduced in [1] and the CLAP score [30] for text-to-music. For FAD we use the official TensorFlow implementation and pre-trained VGGish model. For subjective evaluations we follow the protocol proposed in MusicGen [1] this consists in two studies, one for the overall quality of the samples (OVL) and one for the relation to text (REL).

Music editing: To evaluate the editing performances of our model we perform two objective evaluations. The first one evaluates whether the rhythm matches between the original song and the new stem (BEAT). To do so, we use the beat tracking algorithm from `madmom` [31] both on the original song and on the generated stem. Then, we report the F-measure calculated with `mir_eval` [32] using as reference the beats from the original song. To evaluate the harmonic matching (HAR) between the *bass* stem and the *other* stem we use Chordino[1] to extract the chords played in the *other* stem and use Pesto [33] to estimate the notes from the *bass* line, we only keep the pitches predicted with a confidence greater than 0.75. We then compute the ratio of time steps where the *bass* is playing a chord tone note [34]. For both metrics we zero out stems when the loudness was lower than -35dB.

In addition, we also conducted three subjective assessments of the editing process. In each assessment, we replaced one of the stems of the original song with a generated one. To ensure raters clearly ear the difference, we boost the generated stem to match the loudness of the rest of the track. We then set the overall loudness of this mix to -14dB. Participants are told which instrument differs and are asked to rate the overall quality of the resulting songs. Every subjective study includes 40 samples that are each rated by at least 4 participants.

C. Text-conditioned and unconditional music generation

In this section, we benchmark two families of generative models for music: text conditioned models and stem-level models. Note that only MusicGen-Stem fits into both categories since MSDM does not handle text conditioning and MusicGen operates at the mixture level. To evaluate MSDM [9] we use the official implementation[2]. Since the original model (PT) is trained on the limited Slakh2100 dataset [35], we include a version of this model trained on our dataset (RT).

Objective and subjective metrics presented in TABLE I suggest that MusicGen-Stem is on par with its predecessor on text-conditioned music generation. In the unconditional setup, our results suggest that MusicGen and MusicGen-Stem perform on par. OVL scores shows that MSDM RT produces good quality outputs. However this model mostly generates similar songs, specifically ambient tracks with silent *drums* and *bass*. This limited diversity is reflected in a FAD score over 14.

[1]https://github.com/ohollo/chord-extractor

[2]https://github.com/gladia-research-group/multi-source-diffusion-models

Model	FAD ↓	CLAP ↑	KLD ↓	REL ↑	OVL ↑
Ground Truth	×	0.40	×	93.4 ±0.7	93.6 ±0.5
MusicGen*	0.75	0.37	**0.59**	84.4 ±1.0	86.7 ±0.8
MusicGen-Stem*	**0.70**	**0.38**	0.60	**85.4** ±0.7	**87.0** ±0.8
MusicGen	2.13	×	1.02	×	**85.0** ±0.7
MSDM RT	14.05	×	1.19	×	84.7 ±0.8
MSDM PT	7.61	×	1.48	×	80.9 ±1.0
MusicGen-Stem	2.15	×	1.04	×	83.8 ±0.9

TABLE I: Comparisons of the different music generation models first in a text conditioned setup and then in an unconditional setup. Use of text conditioning is indicated with *.

	HAR ↑		BEAT ↑			OVL ↑		
Edited stem	bass	other	bass	drums	other	bass	drums	other
Ground Truth	72%	72%	0.52	0.87	0.55	93.9 ±0.7	93.4 ±0.7	93.5 ±0.6
MSDM RT	48%	47%	0.28	0.18	**0.45**	72.9 ±1.6	54.0 ±2.0	54.7 ±1.6
MSDM PT	31%	41%	0.03	0.20	0.04	65.0 ±2.0	54.6 ±2.4	42.4 ±2.7
Instruct-MusicGen	N/A	N/A	N/A	N/A	N/A	83.9 ±0.9	51.4 ±1.1	64.4 ±1.1
MusicGen-Stem*	**66%**	**68%**	0.42	**0.69**	0.41	86.5 ±0.8	**86.8** ±0.9	**75.8** ±1.7
MusicGen-Stem	**66%**	67%	**0.46**	0.67	**0.45**	**86.7**±0.9	86.4 ±0.8	72.6 ±1.2

TABLE II: Performances of the models on stem editing task. Use of text conditioning is indicated with *.

D. Text-conditioned and unconditional music editing

We evaluate MusicGen-Stem on single stem music editing. The model is used to generate a coherent third stem in the context of two given stems. We compare it to both versions of MSDM and Instruct-MusicGen. Since the latter regenerates everything at the mixture level, it does not keep the input stems unchanged which prevents us to compute objective metrics.

Results from Table II indicate that MusicGen-Stem outperforms all evaluated baselines in stem editing, regardless of whether text conditioning is applied. Our model consistently generates stems that are more coherent with the overall track, both in terms of rhythm and pitch. Subjective evaluations further validate the superior editing performance of our model. While Instruct-MusicGen shows promising results in bass performance, it is constrained to 5-second audio clips and occasionally alters the song significantly.

V. Conclusion

We introduce a model that is capable of generating music conditioned on either text or instrument stems. MusicGen-Stem reaches comparable performance to the evaluated baselines when considering text-to-music generation, while allowing stem editing. This makes it possible for musicians to iterate on their creations by being able to keep some parts at the instrument level. While MusicGen-Stem is a step towards better control in music generation, it is still limited to 3 stems due to the lack of high quality dataset containing more than the classic *bass*, *drums* and *other*. For future work we intend to increase the capacity of the *bass* tokenizer that tends to create artefacts for higher pitch notes. We also want to have better control on the *other* stem generation with refined conditioning like instrument embedding.

REFERENCES

[1] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez, "Simple and controllable music generation," in *Neurips*, 2023.

[2] Zach Evans, CJ Carr, Josiah Taylor, Scott H. Hawley, and Jordi Pons, "Fast timing-conditioned latent audio diffusion," 2024.

[3] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank, "Musiclm: Generating music from text," 2023.

[4] Or Tal, Alon Ziv, Itai Gat, Felix Kreuk, and Yossi Adi, "Joint audio and symbolic conditioning for temporally controlled text-to-music generation," in *ISMIR*, 2024.

[5] Shih-Lun Wu, Chris Donahue, Shinji Watanabe, and Nicholas J Bryan, "Music controlnet: Multiple time-varying controls for music generation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2692–2703, 2024.

[6] Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli, Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, and Jesse Engel, "Singsong: Generating musical accompaniments from singing," *ArXiv*, 2023.

[7] Julian D. Parker, Janne Spijkervet, Katerina Kosta, Furkan Yesiler, Boris Kuznetsov, Ju-Chiang Wang, Matt Avent, Jitong Chen, and Duc Le, "Stemgen: A music generation model that listens," in *ICASSP*, 2024.

[8] Yao Yao, Peike Li, Boyu Chen, and Alex Wang, "Jen-1 composer: A unified framework for high-fidelity multi-track music generation," *ArXiv*, 2023.

[9] Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà, "Multi-source diffusion models for simultaneous music generation and separation," in *ICLR*, 2024.

[10] Simon Rouard, Francisco Massa, and Alexandre Défossez, "Hybrid transformers for music source separation," in *ICASSP*, 2023.

[11] 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," 2016.

[12] 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*, vol. 30, 2022.

[13] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi, "High fidelity neural audio compression," *TMLR*, 2022.

[14] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar, "High-fidelity audio compression with improved rvqgan," in *NeurIPS*, 2023.

[15] Jean-Marie Lemercier, Simon Rouard, Jade Copet, Yossi Adi, and Alexandre Défossez, "An independence-promoting loss for music generation with language models," *arXiv preprint arXiv:2406.02315*, 2024.

[16] Alon Ziv, Itai Gat, Gael Le Lan, Tal Remez, Felix Kreuk, Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi, "Masked audio generation using a single non-autoregressive transformer," in *ICLR*, 2024.

[17] Hugo Flores Garcia, Prem Seetharaman, Rithesh Kumar, and Bryan Pardo, "Vampnet: Music generation via masked acoustic token modeling," *ArXiv*, 2023.

[18] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," 2022.

[19] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[20] Ke Chen, Yusong Wu, Haohe Liu, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies," 2023.

[21] Haohe Liu, Qiao Tian, Yiitan Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and MarkD . Plumbley, "Audioldm 2: Learning holistic audio generation with self-supervised pretraining," 2023.

[22] Hila Manor and Tomer Michaeli, "Zero-shot unsupervised and text-based audio editing using ddpm inversion," in *ICML*, 2024.

[23] Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J. Bryan, "Ditto: Diffusion inference-time t-optimization for music generation," *ArXiv*, 2024.

[24] Yixiao Zhang, Yukara Ikemiya, Gus Xia, Naoki Murata, Marco A. Martínez-Ramírez, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon, "Musicmagus: Zero-shot text-to-music editing via diffusion models," 2024.

[25] Manos Plitsis, Theodoros Kouzelis, Georgios Paraskevopoulos, Vassilis Katsouros, and Yannis Panagakis, "Investigating personalization methods in text to music generation," in *ICASSP*, 2024.

[26] Simon Rouard, Yossi Adi, Jade Copet, Axel Roebel, and Alexandre Défossez, "Audio conditioning for music generation via discrete bottleneck features," in *ISMIR*, 2024.

[27] Yixiao Zhang, Yukara Ikemiya, Woosung Choi, Naoki Murata, Marco A. Martínez-Ramírez, Liwei Lin, Gus Xia, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon, "Instruct-musicgen: Unlocking text-to-music editing for music language models via instruction tuning," 2024.

[28] Javier Nistal, Marco Pasini, Cyran Aouameur, Maarten Grachten, and Stefan Lattner, "Diff-a-riff: Musical accompaniment co-creation via latent diffusion models," 2024.

[29] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi, "Fréchet audio distance: A metric for evaluating music enhancement algorithms," 2019.

[30] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang, "Clap: Learning audio concepts from natural language supervision," 2022.

[31] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.

[32] Colin Raffel, Brian Mcfee, Eric Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel Ellis, "mir_eval: A transparent implementation of common mir metrics," in *Proceedings - 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 10 2014.

[33] Alain Riou, Stefan Lattner, Gaëtan Hadjeres, and Geoffroy Peeters, "Pesto: Pitch estimation with self-supervised transposition-equivariant objective," in *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023*. 2023, International Society for Music Information Retrieval.

[34] Vincent Persichetti, *Twentieth-Century Harmony*, W. W. Norton & Company, 1961.

[35] Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux, "Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity," in *ISMIR*, 2019.