InstructME: An Instruction Guided Music Edit Framework with Latent Diffusion Models

Bing Han¹, Junyu Dai², Weituo Hao², Xinyan He², Dong Guo², Jitong Chen², Yuxuan Wang², Yanmin Qian¹ and Xuchen Song²

¹Auditory Cognition and Computational Acoustics Lab, Shanghai Jiao Tong University
²ByteDance
xuchensong895@gmail.com

Abstract
Music editing primarily entails the modification of instrument tracks or remixing in the whole, which offers a novel reinterpretation of the original piece through a series of operations. These music processing methods hold immense potential across various applications but demand substantial expertise. Prior methodologies, although effective for image and audio modifications, falter when directly applied to music. This is attributed to music’s distinctive data nature, where such methods can inadvertently compromise the intrinsic harmony and coherence of music. In this paper, we develop InstructME, an Instruction guided Music Editing and remixing framework based on latent diffusion models. Our framework fortifies the U-Net with multi-scale aggregation in order to maintain consistency before and after editing. In addition, we introduce chord progression matrix as condition information and incorporate it in the semantic space to improve melodic harmony while editing. For accommodating extended musical pieces, InstructME employs a chunk transformer, enabling it to discern long-term temporal dependencies within music sequences. We tested InstructME in instrument editing, remixing, and multi-round editing. Both subjective and objective evaluations indicate that our proposed method significantly surpasses preceding systems in music quality, text relevance and harmony. Demo samples are available at https://musicedit.github.io/

1 Introduction
Music editing involves performing basic manipulations on musical compositions, including such atomic operations as the inclusion or exclusion of instrumental tracks and the adjustment of pitches in specific segments. On top of these atomic operations, remixing can be understood as an advanced version of music editing that mixes various atomic operations with style and genre considered [Fagerjord, 2010]. Both atomic operations and remix can be handled by using a text-based generative model. In music editing, the text would be natural language-based editing instructions, such as “adding a guitar track”, “replacing a piano track with a violin”, etc. Models from the text-generated image domain seem to be adaptable to the music editing scenario. However, unlike image generation, models need to pay attention to the music harmony in addition to understanding the text and generating it. For example, when introducing a guitar track, care is taken to harmonize its rhythm, chord progression, and melodic motifs with the original audio framework, thus ensuring that overall consistency and coherence are maintained. Therefore, for successful music editing, the model should be able to: (i) understand editing instructions and generate music stems; (ii) ensure the compatibility of the part being processed with the original music source.

Lately, a multitude of endeavours pertaining to text-based image or audio manipulation [Hertz et al., 2022; Lugmayr et al., 2022; Meng et al., 2021; Wang et al., 2024] have attracted considerable attention due to their noteworthy performance within their respective domains. However, the distinct data properties and generative prerequisites inherent to the domain of music preclude the direct applicability of these methods to the sphere of music editing. In image editing, it is feasible to maintain consistency over the residual regions by employing masking techniques, thereby confining attention solely to the objects to be generated. However, this underlying principle proves inapplicable to the domain of musical data, as shown in Figure 1 the interwoven nature of individual tracks across
both temporal and frequency domains prevents the straightforward implementation of such an approach. Perhaps the most similar method to ours is [Wang et al., 2024] for audio editing. However, the method is mainly applied to the editing of sound effects. Unlike music tracks, the individual sound effects are independent of each other, so there is no need to consider whether the sound effects are in harmony or not.

In order to bridge the gap between text-based generative models and music editing tasks, we propose InstructME, an instruction-guided music editing framework based on latent diffusion models. For simplicity, we limit music editing operations to adding, removing, extracting, replacing, and remixing. As shown in Figure 2, InstructME takes text instructions and source music as input, and outputs the target music accordingly. To maintain the consistency of the music before and after editing, we utilize the multi-scale aggregation strategy and incorporate the chord progression matrix into the semantic space [Kwon et al., 2022; Jeong et al., 2023] during the source music encoding process to ensure harmony. During training, we employ the chunk transformer to model long-term temporal dependencies of music data in a segmented chunk-wise manner and train the model on collected 417 hours of music data. For testing, we evaluate the model in terms of three aspects: music quality, text relevance and harmony. Experimental results of public and private datasets demonstrate that InstructME outperforms the previous system.

Our key contributions can be summarized as:

- To the best of our knowledge, we propose the first instruction guided music editing framework applicable for both atomic and advanced operations.
- We point out the special problem of consistency and harmony in music editing domain and develop multi-scale aggregation and chord condition via chunk transformer to solve it.
- We propose quantitative evaluation metrics for music editing tasks in terms of quality, relevance and harmony.
- Our proposed method InstructME surpasses previous systems through thorough subjective and objective tests.

2 Related Work

2.1 Text Guided Generation

Generating a new version of accompaniment for a track directly with targeted properties (e.g., genre, mood, instrument) adhering is a viable approach to accomplish the objectives of editing or remixing. Recent studies [Huang et al., 2023b; Liu et al., 2023a; Agostinelli et al., 2023; Schneider et al., 2023; Huang et al., 2023a; Lam et al., 2023; Copet et al., 2023] have already succeeded in generating plausible music that reflects key music properties (e.g., genre, mood, etc) that are depicted in a given text. However, there is no guarantee for them to generate tracks that are harmonious with a given track while keeping the given one or specified part of it unchanged. Another work [Donahue et al., 2023] proposed a generative model, which trained over instrumental vocals, generating coherent instrumental music to accompany input vocals. But it has no way for users to control the generation process, not to mention interactive editing, which is important for an intelligent editing tool as it applies feedback from users to make a more preferable output as in [Holz, 2023].

2.2 Audio Editing and Music Remixing

[Huang et al., 2023b; Liu et al., 2023a] propose zero-shot audio editing by utilizing pre-trained text-to-audio latent diffusion models, which seem flexible but not accurate enough for the editing process. Moreover, there is no guarantee for those audio generation models that are trained with general purposes to achieve a good editing effect in the editing specialized usage scenario. Due to this, AUDIT [Wang et al., 2024] proposed a general audio editing model based on a latent diffusion and denoising process guided by instructions. Certainly, as previously stated in the introduction section, this framework necessitates certain enhancements to effectively cater to music-related tasks.

For remixing, although the text-guided generative systems mentioned above can also perform generation conditions on a given recording [Liu et al., 2023a; Lam et al., 2023], or more specifically, melodies [Agostinelli et al., 2023; Copet et al., 2023], the generated music can only preserve the tune of the conditional melodies, the original tracks, such as vocal, will not directly feature in the output music. This kind of conditional-generated music is traditionally known as music covers, not remixes. Likewise, past studies [Yang et al., 2022; Yang and Lerch, 2020b; Wierstorfer et al., 2017] have attempted to apply neural networks to the task of music remixing. These methods, which are often incorporated with source separation models, primarily viewed music remixing as a task of adjusting the gain of individual instrument sources of an audio mixture. But music remixing is not just limited to manipulating the gain of different sources of the recording itself, it can also involve incorporating other materials to create something new [Waysdorf, 2021].

3 Methodology

In this section, we will provide an overview of the InstructME architecture and the process of instruction-based music editing, as illustrated in Figure 3. Additionally, we will explain strategies aimed at improving editing consistency and harmony, as well as approaches to achieving more sophisticated music editing operations.
3.1 Instruction to Music Editing

InstructME accepts music audio $x_s$ and editing instructions $y$ as input, and produces new audio $x$ that adheres to the given instructions. We utilize text and audio encoders to transform the data into a latent representation. For each text instruction $y$, a pretrained T5 [Raffel et al., 2020] converts it into sequence of embeddings $T(y) \in \mathbb{R}^{L \times D}$, similar to [Wang et al., 2024]. For each audio segment $x_s \in \mathbb{R}^{T \times 1}$, a variational auto-encoder (VAE) transforms the waveform into a 2D latent embedding $z_s \in \mathbb{R}^{L \times C}$. Using text and audio embeddings as conditions, a diffusion process [Song et al., 2020; Ho et al., 2020] produces embeddings of new audio samples, which the VAE decoder then converts to audio waveforms.

The VAE used by InstructME consists of an encoder $E$, a decoder $D$ and a discriminator with stacked convolutional blocks. The decoder reconstructs the waveform $\hat{x}$ from the latent space $z$ and there is no vocoder like [Wang et al., 2024; Liu et al., 2023a]. The discriminator was used to enhance the sound quality of generated audio through adversarial training. We provide the model and training details in the Appendix.

Diffusion Model

Diffusion model contains two processes. The forward process is a standard Gaussian noise injection process. At time step $t$,

$$q(z_t | z_0) = \mathcal{N}(z_t; \sqrt{1 - \alpha_t}z_0, (1 - \alpha_t)\epsilon)$$  \hspace{1cm} (1)

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{t=1}^{T} \alpha_t$ are scheduling hyperparameters. In the reverse process, we employ a time-conditional U-Net $\epsilon_{\theta}$ [Ronneberger et al., 2015; Rombach et al., 2022] as the denoise model backbone. At time step $t$, conditioning on embeddings of text $T(y)$ and source music $z_s$, this denoise model attempts to restore the original latent $z_0$ of target music from noisy $z_t$. For model optimization, we use reweighted bound [Ho et al., 2020; Rombach et al., 2022] as objective function:

$$\mathcal{L}_{\text{DM}} = \mathbb{E}_{\epsilon, t, z_0} \| \epsilon - \epsilon_{\theta}(t, T(y), z_s, z_t) \|^2_2$$  \hspace{1cm} (2)

with $t$ uniformly sampled from $[1, T]$ during the training. In the end, we pass $z_0$ through the decoder $D$ to obtain the waveform music.

The U-Net layers utilize transformers with self and cross attention as building blocks. In the down sampler layers, source audio embeddings and generated embeddings merge into the input of the self-attention layer. Cross attention is employed for text conditions in each down-up sampler layer, and for chord conditions in the bottleneck layer.

Efficient Diffusion

The self-attention of lengthy music sequences is computationally expensive. To alleviate this problem, we employ the chunk transformer to model long-term temporal dependencies in a chunk-wise manner, which is different from [Zha et al., 2021] in both architecture and motivation. Outlined in Figure 3 (Right), the process involves three steps: segmentation of $T$-frame embeddings into $K$-frame chunks with 50% overlap, individual chunk processing through a transformer layer, and fusion to merge overlapping output chunks into $T$ frames by addition.

At each layer of the chunk transformer, a token from a $K$-frame chunk can observe $3K$ neighboring frames. By stacking multiple layers of chunk transformer, the U-Net acquires an expansive receptive field, enabling effective modeling of long-term dependencies. Compared with oracle transformer’s complexity $O(T^3)$, chunk transformer has lower computational cost $O(2 \times \frac{T}{K} \times K^2) = O(TK)$. In addition to faster inference and lower memory consumption, the chunk-wise modeling approach decreases the model’s reliance on sequence length, learning invariant representations. This minimizes performance degradation caused by duration differences in training and sampling.
3.2 Improving Consistency and Harmony
To make the diffusion model more suitable for music editing tasks, we propose an enhanced U-Net with several modifications including multi-scale aggregation and chord condition.

Multi-Scale Aggregation
Contrary to the music generation tasks [Huang et al., 2023a], music editing tasks require the preservation of certain content and properties from the original music. In order to maintain coherence between the original and edited music, AUDIT [Wang et al., 2024] directly concatenates the source music channel $z_s$ with the target music channel $z_t$ at the U-Net’s input. It leans heavily on the invariance of some low-level and local music features, which might pose challenges or limitations when applied to more complex music manipulation tasks. To more effectively capture the high-level characteristics of the source music, we introduce a multi-scale aggregation (MSA) strategy as depicted in Figure 3 (Left). The source music embeddings $z_s$ are input to a multi-layer convolution encoder, yielding feature maps with varying resolutions for the corresponding U-Net layers. This strategy has been proven effective in high-resolution image generation [Karras et al., 2020].

Chord-Conditional
The Chord progression is a key element in defining a piece’s musical harmony. We adopt a chord progression recognition model $\mathcal{C}$ [Cheuk et al., 2022] to extract the chord probability embedding $p$ of the source music and then emphasize it explicitly during the denoise process. [Kwon et al., 2022; Jeong et al., 2023] discover the semantic latent space in the bottleneck of diffusion has nice properties to accommodate semantic image manipulation. Inspired by them, we incorporate the chord progression representation $p \in \mathbb{R}^{d_p \times T_p}$ in the bottleneck feature map $h \in \mathbb{R}^{d_h \times T_h}$ of U-Net with cross-attention mechanism [Vaswani et al., 2017]. With chord progression condition extracted by $\mathcal{C}$, in the bottleneck layer of U-Net, the objective function in Equation 2 can be rewritten:

$$L_{CDM} = \mathbb{E}_{r, t, z_0} \| t - \epsilon_\theta(t, p_s, z_s, z_t) \|_2^2$$

(3)

where $p_s$ denotes chord progression matrix of source music $x_s$, encoded by extractor $\mathcal{C}$.

3.3 Towards Advanced Music Editing - Remix
For diffusion models, there exist two primary strategies for achieving controllable generation. One of these is classifier guidance (CG) [Dhariwal and Nichol, 2021; Liu et al., 2023b], which utilizes a classifier during the sampling process and mixes its input gradient of the log probability with the score estimate of diffusion model. It is flexible and controllable, but tends to suffer a performance degradation [Ho and Salimans, 2022]. Another approach, named classifier-free guidance (CFG) [Ho and Salimans, 2022; Nichol et al., 2021; Ramesh et al., 2022; Saharia et al., 2022], achieves the same effect through training a conditional diffusion model directly without a guidance classifier. This method performs better but requires a large amount of data with diverse text descriptions, which is difficult for our InstructME trained with source-target paired data. In this work, to attain a tradeoff between quality and controllability, we adopt both classifier and classifier-free guidance to achieve the controllable editing of Remix operations.

We specify instrument and genre tags with CFG by incorporating these tags into text commands to train the conditional diffusion models. During the training, we discard our text condition $y$ randomly with a certain probability $p_{CFG}$ following [Liu et al., 2023a; Wang et al., 2024]. Then, in the sampling, we can estimate the noise $\epsilon_\theta(t, \mathcal{T}(y), p_s, z_s, z_t)$ with a linear combination of the conditional and unconditional score estimates:

$$\epsilon_\theta(t, \mathcal{T}(y), p_s, z_s, z_t) = (1 - w)\epsilon_\theta(t, p_s, z_s, z_t) + we_\theta(t, \mathcal{T}(y), p_s, z_s, z_t)$$

(4)

where $w$ can determine the strength of guidance.

To achieve finer-grained semantic control with weakly-associated, free-form text annotations, we apply classifier guidance during sampling with a pre-trained MuLan [Huang et al., 2022], which can project the music audio and its corresponding text description into the same embedding space. The guidance function we use is:

$$F(x_t, y) = \| E_L(y) - E_M(x_t) \|_2^2$$

(5)

where $E_L(\cdot)$ and $E_M(\cdot)$ denote the language and music encoders respectively. Then, by adding the gradient on estimated $x_t$, we can guide the generation

$$\hat{x}_t = x_t + s\nabla_{x_t} F(x_t, y)$$

(6)

with factor $s$ to control the guidance scale.

4 Experiments Setup
4.1 Dataset
We collected 417 hours of music audio. Each audio file consists of multiple instrumental tracks. We resampled audios to 24kHz sample rate and divided them into non-overlapping 10-second clips. For each audio clip, we select pairs of versions with varying instrument compositions and generate a text instruction based on the instrument differences. We use the clips generated before to prepare the triplet data $<text instruction, source music, target music>$ including remixing (1 Million), adding (0.3M) and replacement (0.3M), extracting (0.2M) and removing (0.2M) respectively. These music triplet data are referred to as the ‘in-house data’. We show our detailed data processing methods in Appendix.

Evaluation Data
We evaluate the models on both in-domain data and out-domain data. (1) In-domain data: We split the in-house data randomly into two parts and use one subset to generate triplet data for evaluating the models. (2) Out-domain data: To demonstrate the robustness of the system, we also evaluate the models on the Synthesized Lakh (S lakh) Dataset [Manilow et al., 2019] which is a dataset of multi-track audio and has no overlap with the training data.

4.2 Evaluation Metric
Music is sounds that are artificially organized in relation to the sensational moments, with complex interplay and multi-layered perceptual impact between pitch, intensity, rhythm.
and timbre. Defining a single suitable metric to fully evaluate music is challenging, [Agostinelli et al., 2023; Huang et al., 2023a] focus evaluation on signal quality and semantics, whereas [Lv et al., 2023; Ren et al., 2020; Yang and Lerch, 2020a] propose more direct evaluation approach based on musicality indicators, in order to achieve a more comprehensive evaluation of music, we proposed the following metrics to objectively evaluate the performance of edited music in three aspects:

Music Quality : We use the fréchet audio distance (FAD)\(^1\) [Kilgour et al., 2019] to measure the quality between edited music and target music, the audio classification model is implemented with VGGish [Hershey et al., 2017].

Text Relevance : We define the instruction accuracy (IA) metric to indicate the relevance of the text-music pair, the proposed editing tasks are all related to music tags such as instrument, mood and genre, so we calculate instruction accuracy according to the edited music tags and input command while tags are recognized with tagging models implemented with [Lu et al., 2021].

Harmony : We use three metrics for harmony evaluation,

- **Chord Recognition Accuracy (CRA)**: Chord Recognition Accuracy measures the harmony coherence between edited music and target music. We acquire the chord progression sequences of both source and target music in the initial step, while the chord progression recognition model is implemented by [Cheuk et al., 2022]. Then the alignment of these sequences is computed to determine the chord recognition accuracy.

- **Pitch Class Histogram (PCH)**\(^2\): The pitch class histogram is a pitch content representation that is octave-independent. We calculate the distribution of pitches classes according to this histogram.

- **Inter-Onset Interval (IOI)**\(^2\): Inter-onset interval refers to the time between two note onsets within a bar. In our case, regarding PCH and IOI, we further compute the averaging overlapped area of their distributions to quantize the musical harmony in terms of pitch and onset aspects.

For objective evaluation, we generate 800 triplet data randomly for each music editing task and evaluate them with these objective metrics.

As for subjective evaluation, same as baseline AUDIT [Wang et al., 2024], we conduct overall testing following the standard Mean Opinion Score (MOS) evaluation procedure, which is widely used in measuring audio generation tasks [Huang et al., 2023b; Schneider et al., 2023].

### 5 Results and Analysis

#### 5.1 Objective Evaluation Results

We compare against AUDIT [Wang et al., 2024] trained on the same data and in the same VAE latent space, as our baseline system in all experiments. As shown in Table 1, InstructME outperforms AUDIT in terms of music quality, text relevance and harmony. Specifically, operations such as extract and remove are tasks with definite answers, and these tasks emphasize the precision of generation. Alternatively, operations such as add, replace, and remix are tasks with indeterminate answers, and these tasks are more creatively oriented. These two types of tasks require the model to achieve stable and diverse output based on an accurate understanding of textual instructions. From Table 1, it is observed that InstructME improves musical quality by 5.84%, text relevance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Task</th>
<th>FAD(_{VGG}(↓))</th>
<th>Instruction Acc. (↑)</th>
<th>Chord Rec. Acc. (↑)</th>
<th>Pitch His. (↑)</th>
<th>IO Interval (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house</td>
<td>AUDIT</td>
<td>Extract</td>
<td>1.67</td>
<td>0.39</td>
<td>0.82</td>
<td>0.62</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remove</td>
<td>1.73</td>
<td>0.65</td>
<td>0.86</td>
<td>0.64</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add</td>
<td>1.25</td>
<td>0.73</td>
<td>0.72</td>
<td>0.64</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace</td>
<td>1.50</td>
<td>0.62</td>
<td>0.83</td>
<td>0.63</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>1.54</td>
<td>0.60</td>
<td>0.81</td>
<td>0.63</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td>InstructME</td>
<td>Extract</td>
<td>1.54</td>
<td>0.56</td>
<td>0.86</td>
<td>0.69</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remove</td>
<td>1.68</td>
<td>0.80</td>
<td>0.88</td>
<td>0.72</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add</td>
<td>1.22</td>
<td>0.73</td>
<td>0.75</td>
<td>0.72</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace</td>
<td>1.39</td>
<td>0.62</td>
<td>0.86</td>
<td>0.71</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>1.45</td>
<td>0.68</td>
<td>0.84</td>
<td>0.71</td>
<td>0.67</td>
</tr>
<tr>
<td>Slakh</td>
<td>AUDIT</td>
<td>Extract</td>
<td>4.91</td>
<td>0.52</td>
<td>0.66</td>
<td>0.62</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remove</td>
<td>1.92</td>
<td>0.57</td>
<td>0.57</td>
<td>0.61</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add</td>
<td>3.11</td>
<td>0.87</td>
<td>0.58</td>
<td>0.63</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace</td>
<td>4.08</td>
<td>0.78</td>
<td>0.55</td>
<td>0.62</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>3.50</td>
<td>0.68</td>
<td>0.59</td>
<td>0.62</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>InstructME</td>
<td>Extract</td>
<td>5.04</td>
<td>0.66</td>
<td>0.71</td>
<td>0.70</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remove</td>
<td>1.87</td>
<td>0.79</td>
<td>0.65</td>
<td>0.70</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add</td>
<td>3.15</td>
<td>0.87</td>
<td>0.66</td>
<td>0.74</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace</td>
<td>3.97</td>
<td>0.83</td>
<td>0.65</td>
<td>0.74</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>3.50</td>
<td>0.79</td>
<td>0.67</td>
<td>0.72</td>
<td>0.69</td>
</tr>
</tbody>
</table>

\(^1\)https://github.com/gudgud96/frechet-audio-distance
\(^2\)https://github.com/RichardYang40148/mgeval

Table 1: Objective Evaluation Results of different edit tasks on In-house and Slakh datasets. Avg. is the average result of several edit tasks including extract, remove, add and replace. FAD reflects the music quality, Instruction Acc., and Chord Rec. Acc., pitch His. and IO Interval can measure the harmony of edited music.
by 13.33% and harmony up to 26.42% compared to AUDIT on In-house dataset. These results demonstrate that our approach provides rich generated content in addition to capturing the difference between textual instructions.

<table>
<thead>
<tr>
<th>Model</th>
<th>FAD</th>
<th>IA</th>
<th>CRA</th>
<th>PCH</th>
<th>IOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT</td>
<td>0.49</td>
<td>0.69</td>
<td>0.59</td>
<td>0.60</td>
<td>0.46</td>
</tr>
<tr>
<td>InstructME</td>
<td>0.45</td>
<td>0.73</td>
<td>0.70</td>
<td>0.72</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table 2: Objective results of remixing on In-house dataset.

To study the generalization ability of InstructME, we also test it on the public available dataset Slakh [Manilow et al., 2019] which is more challenging in maintaining harmony by including unseen chord progressions. Table 1 shows that InstructME achieves comparable performance with AUDIT on musical quality but better results on text relevance and harmony. Particularly for chord recognition accuracy, our method surpasses the previous method by 13.56%.

As a novel and unique task proposed in this paper, we also evaluated the performance of InstructME and AUDIT on remix task. As Table 2 indicates, compared to AUDIT, InstructME achieves a significant improvement in harmony related metrics(CRA by 18.6%, PCH by 20% and IOI by 39%), and also achieves better results in quality and text relevance.

5.2 Subjective Evaluation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Duration</th>
<th>Quality</th>
<th>Relevance</th>
<th>Harmony</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT</td>
<td>10s</td>
<td>2.79 ± 0.15</td>
<td>2.94 ± 0.08</td>
<td>3.01 ± 0.17</td>
</tr>
<tr>
<td></td>
<td>30s</td>
<td>2.33 ± 0.19</td>
<td>2.23 ± 0.13</td>
<td>2.19 ± 0.21</td>
</tr>
<tr>
<td></td>
<td>60s</td>
<td>2.24 ± 0.22</td>
<td>2.09 ± 0.15</td>
<td>2.05 ± 0.20</td>
</tr>
<tr>
<td>InstructME</td>
<td>10s</td>
<td>3.35 ± 0.13</td>
<td>3.59 ± 0.10</td>
<td>3.54 ± 0.14</td>
</tr>
<tr>
<td></td>
<td>30s</td>
<td>2.62 ± 0.20</td>
<td>3.05 ± 0.16</td>
<td>2.63 ± 0.23</td>
</tr>
<tr>
<td></td>
<td>60s</td>
<td>2.62 ± 0.24</td>
<td>2.93 ± 0.17</td>
<td>2.48 ± 0.21</td>
</tr>
</tbody>
</table>

Table 3: Subjective results of editing on music of different duration.

We also conduct a subjective evaluation by outsourcing 10 testing samples(10s clip) per editing task for each labor. Mean Opinion Score(MOS) of scale 5 is used to compare the musical quality, text relevance and harmony of two methods. To be more representative of real-world application scenarios, we also respectively generate 30-second and 60-second audio results, leveraging the chunk transformer’s insensitivity to output length. A subjective evaluation of these generated long-duration music was also performed. We wrap all results in Table 3 and the MOS scores show the superiority of our method over the baseline.

5.3 Case Study

Chord Condition and Multi-Scale Aggregation

We perform ablation experiments to study the impact of the chord condition and multi-scale aggregation. The objective evaluation results of training InstructME without chord condition and multi-scale aggregation are listed in Table 4 respectively. The absence of chord conditioning is demonstrated to lead to a deterioration in harmony-related metrics such as CRA, PCH, and IOI. This observation underscores the critical role of the chord conditioning mechanism in holding the harmonicity of music editing. Moreover, the notable decline in FAD, subsequent to the deactivation of Multi-Scale Aggregation within the U-Net architecture, indicates its significant contribution to preserving the audio quality of music during the editing process.

<table>
<thead>
<tr>
<th>Metric</th>
<th>InstructME</th>
<th>w/o Chord Condition</th>
<th>w/o MSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAD(↓)</td>
<td>1.45</td>
<td>1.46(↑0.01)</td>
<td>1.53(↑0.08)</td>
</tr>
<tr>
<td>IA(↑)</td>
<td>0.68</td>
<td>0.63(↓0.05)</td>
<td>0.66(↓0.02)</td>
</tr>
<tr>
<td>CRA(↑)</td>
<td>0.84</td>
<td>0.81(↓0.03)</td>
<td>0.83(↓0.01)</td>
</tr>
<tr>
<td>PCH(↑)</td>
<td>0.71</td>
<td>0.64(↓0.07)</td>
<td>0.72(↑0.01)</td>
</tr>
<tr>
<td>IOI(↑)</td>
<td>0.67</td>
<td>0.53(↓0.14)</td>
<td>0.67(↓0.00)</td>
</tr>
</tbody>
</table>

Table 4: Impact of chord condition and multi-scale aggregation strategies. MSA denotes multi-scale aggregation here. Mean value over all edit operations for each metric is provided.

Consistency and Harmony

In Figure 4(b), we present an illustrative study to elucidate aspects of consistency and harmony. For example, we take “Add acoustic guitar” as the textual instruction. The waveforms in Figure 4(b)(1) indicate that the beat timings before and after editing are meticulously synchronized, which demonstrates InstructME’s capacity to maintain temporal consistency through the editing process. The temporal consistency is also observed in the corresponding spectrograms in Figure 4(b)(2), which exhibit energy spikes at identical times. Subsequently, we proceed to extract and compare the chord progression matrices from the source and generated musical segments as portrayed in Figure 4(b)(3). The intensity of the color is indicative of the predicted chord probability. Upon scrutinizing the probability patterns between the source and generated music, it is concluded that our method is able to preserve the musical harmony of the source. The last row in Figure 4(b) is the pitch matrix. In this representation, the uppermost pair of white stripes corresponds to the pitch pertaining to the piano and drum, respectively. Remarkably, the mere variations of these two lines between source and generated music, it is concluded that our method is able to preserve the musical harmony of the source. The last row in Figure 4(b) is the pitch matrix. In this representation, the uppermost pair of white stripes corresponds to the pitch pertaining to the piano and drum, respectively. Remarkably, the mere variations of these two lines between source and generated music, it is concluded that our method is able to preserve the musical harmony of the source. The last row in Figure 4(b) is the pitch matrix. In this representation, the uppermost pair of white stripes corresponds to the pitch pertaining to the piano and drum, respectively.
Figure 4: (a): Visualization of different editing tasks with three samples. All music segments are shown by spectrograms. (b): Comparison between source and edited music from four perspectives: (1) waveform (2) spectrogram (3) chord matrix (4) pitch matrix. The instruction command in the example is “add acoustic guitar”.

Figure 5: Line chart of objective evaluation results of three-round editing. Compared to AUDIT, InstructME shows a smaller decrease in metrics related to harmony and text relevance, while also exhibiting a smaller degradation in music quality (FAD).

remix demands diversity because of different creative interpretations of the same sound source. However, the removal operation requires stability as the model should generate accurate and unique results. We also conducted visualization explorations for different tasks, as depicted in Figure 4(a). Each row contains the source music, the ground truth and three samples generated by InstructME. For tasks requiring precision, our model consistently generates results congruent with the ground truth. For creativity-oriented tasks, the visual representation illustrates the diversity present in the spectrogram of the sampled music segments. The subjective results of diversity and stability are listed in Table 5, which underscore the capacity of InstructME to conceive and construe novel compositions derived from existing audio sources.

We strongly encourage readers to learn about the performance of our model through demos at https://musicedit.github.io/.

6 Conclusion

In this work, we introduce InstructME, a music editing and remixing framework based on latent diffusion models. For InstructME, we enhance the U-Net with multi-scale aggregation and chord condition to improve the harmony and consistency of edited music, and introduce chunk transformer to extend the long-term music generation capabilities. To evaluate the efficacy of music editing results, we establish several quantitative metrics and conduct experimental trials to validate them. Our findings indicate that the proposed InstructME outperforms the baselines in both subjective and objective experiments, which shows that our InstructME can effectively edit source music based on simple editing instructions, while preserving certain musical components and generating harmonious results that align with the semantic information conveyed in the instructions.

Acknowledgements

Work was done during Bing Han’s internship at ByteDance. All authors conceive the paper’s methodology and experiments. Bing Han and Junyu Dai contributed equally to this work. Yanmin Qian is the corresponding author. Bing Han and Yanmin Qian were supported in part by China NSFC projects under Grants 62122050 and 62071288, in part by Shanghai Municipal Science and Technology Commission Project under Grant 2021SHZDZX0102.
References


[Huang et al., 2023b] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Lining Liu, Mingze Li, Zhenhui Ye, Jinglei Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. arXiv preprint arXiv:2301.12661, 2023.


[Lv et al., 2023] Ang Lv, Xu Tan, Peiling Lu, Wei Ye, Shikun Zhang, Jiang Bian, and Rui Yan. Getmusic: Generating any music tracks with a unified representation and


