Guiding Audio Editing with Audio Language Model

Zitong Lan **Yiduo Hao** **Mingmin Zhao**
University of Pennsylvania

Abstract

Audio editing is increasingly important in immersive applications such as VR/AR, virtual conferencing, and sound design. While diffusion-based models have enabled language-driven audio editing, existing methods rely on predefined instruction formats and are limited to mono-channel audio. In this work, we introduce SmartDJ, a novel framework for stereo audio editing that combines the reasoning capabilities of Audio Language Models (ALMs) with the generative power of latent diffusion. Given a high-level prompt, SmartDJ decomposes it into a sequence of atomic editing steps, which are executed sequentially by a conditional diffusion model trained to manipulate stereo audio. We also develop a scalable data synthesis pipeline that generates training samples consisting of a high-level instruction, a sequence of atomic edits, and the corresponding audio at each step of the editing process. Experiments show that SmartDJ outperforms prior methods in perceptual quality, spatial coherence, and alignment with complex user instructions.

1 Introduction

Audio editing is central to shaping a listener's spatial and semantic perception of an environment, with its growing importance in VR/AR, gaming, virtual conferencing, and post-production sound design. Recent advances in diffusion-based generative models have enabled both text-to-audio generation [10, 11, 20, 22, 25, 39, 40] and language-driven audio editing [28, 37, 55, 61]. However, existing editing approaches typically rely on narrowly structured prompts or predefined templates (e.g., *add the sound of waves, remove the sound of car engines*) and cannot interpret or execute high-level and sometimes open-ended user instructions. Moreover, existing methods operate on mono-channel audio and can not model or manipulate spatial properties such as directionality (e.g. *change the sound source from left to right*), which are crucial for perceptual realism.

In practice, users often express their editing goals through short, high-level, and sometimes ambiguous instructions, as crafting detailed, step-by-step edits is tedious and unintuitive [34]. For example, a user might type a prompt such as *"have this audio in a sunny forest"* when editing a clip. Fulfilling this instruction requires nuanced reasoning across multiple editing operations, such as removing rain or thunder sounds, omitting unrelated elements like ocean waves, and enhancing subtle ambient cues like rustling leaves or distant bird calls. As shown on the left of Fig. 1, existing text-driven audio editors struggle with such prompts. They are unable to interpret high-level instructions and often produce semantically inconsistent results that fail to align with the original audio context.

In this work, we propose SmartDJ, the first framework to leverage large language models (LLMs) for intelligent audio editing guided by high-level user instructions. This approach is motivated by the recent success of LLMs in multimodal grounding and reasoning [5, 17, 31, 41, 35]. As illustrated on the right of Fig. 1, SmartDJ incorporates an Audio Language Model (ALM) that takes both the original audio and a high-level instruction as input. The ALM reasons over the input and decomposes the instruction into a sequence of atomic editing steps, each corresponding to a predefined atomic operation – such as adding or removing a sound event, or modifying the volume or spatial direction of an existing one. These steps are then executed sequentially by a conditional Latent Diffusion Model (LDM), which operates in the latent space of a variational autoencoder trained on stereo audio. This step-by-step editing process enables precise control over sound events, spatial positioning, and

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: The First Workshop on Generative and Protective AI for Content Creation.

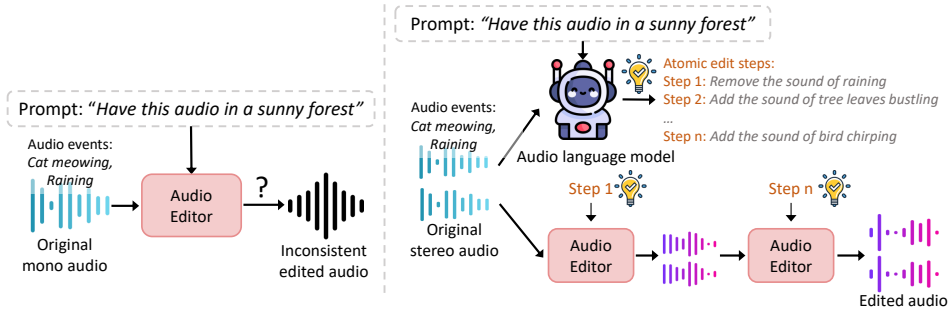


Figure 1: *Left*: Existing diffusion-based audio editors cannot interpret high-level editing instructions, producing inconsistent outputs. *Right*: SmartDJ leverages an Audio Language Model to decompose it into atomic editing steps, guiding the diffusion model to produce high-quality edited audio.

volume adjustment, all conditioned on natural language prompts. As a result, SmartDJ generates high-fidelity, spatially immersive audio edits that preserve key elements of the original scene while incorporating new content aligned with the user's intent.

To train and evaluate SmartDJ, we develop a scalable dataset generation pipeline that constructs *editable audio scenes* using a sound event library and an off-the-shelf LLM. For each data point, we begin by randomly sampling a set of labeled sound event. We then prompt the LLM with carefully designed instructions to generate both a high-level user editing command and a corresponding sequence of atomic editing steps. The LLM also assigns initial volume and spatial direction to each event. With basic audio processing, we render the spatial properties of each event and mix them via superposition to construct the scene's original audio. Crucially, since each sound source is *independently editable*, we can simulate the effect of each atomic edit by directly modifying event-level parameters, without altering other audio events. For example, to implement *"turn up the dog bark"*, we increase the volume of the dog bark event; to implement *"move the footsteps to the left"*, we adjust the spatial direction of the event accordingly. This allows us to generate the full editing trajectory by progressively updating sound parameters and re-composing the audio in the scene. These processes result in a large dataset containing high-level editing instructions, step-by-step atomic edit actions, and corresponding audio outputs at each step of the editing process.

Experimental results show that SmartDJ delivers superior editing quality and better alignment with high-level user instructions from both subjective metrics and human evaluations. Our latent diffusion model for audio editing also outperforms existing baselines on single-step editing tasks. Ablation studies show that the audio language model can effectively reason about and decompose high-level user instructions, enabling the latent diffusion model to perform expressive audio editing.

In summary, our main contributions are as follows:

- We introduce an audio editing framework SmartDJ that leverages ALM to interpret high-level instructions and generate detailed editing steps executed by a latent diffusion model.
- We design a scalable data generation pipeline for stereo audio editing with high-level instructions.
- We conduct extensive experiments and user studies with different baseline methods and demonstrate that SmartDJ has the highest editing quality for both objective and subjective metrics.

2 Related work

Audio generation and editing. With the current advances in deep generative models, lots of methods have achieved high-quality audio generation from text and multi-modal conditions [4, 10, 20, 25, 39, 40, 55]. Recently, spatial audio generation has attracted more attention [10, 22, 42, 52]. Parallel to these generation efforts, text-guided audio editing also emerged as a powerful tool for modifying existing audio recordings. Audit [55] introduced an end-to-end diffusion model conditioned on both the input audio and simple, structured text commands, but its reliance on fixed editing templates limits flexibility to interpret high-level user prompts. WavCraft [37] uses GPT-api to parse user instructions, yet it expects fully specified prompts (e.g., "extract baby crying from the audio" or "apply a low-pass filter to the wave crashing sound"). Recent works [28, 44, 61] adapt image-editing techniques to monaural audio. They still require users to provide precise token-level manipulation to complete the edit, struggle with high-level instructions, and offer no support for stereo or spatial

audio editing. No existing work can interpret high-level user input to complete the audio editing and they remain confined to monaural outputs and are ill-suited for immersive spatial scenarios.

Multimodal Large Language Model. Large language models (LLMs) are remarkable in natural language processing tasks. Provided with multimodal inputs, such as images and audio, multimodal LLMs (MLLMs) [9, 41, 35, 54, 32, 18] demonstrate exceptional performance across a wide range of downstream visual-language and audio-language tasks. In the vision domain, LLaVA [41] enables LLMs to achieve general-purpose visual and language understanding by fine-tuning on a multimodal instruction-following dataset. In the audio domain, LTU [19] and Audio Flamingo [32, 18] enhance LLMs with the ability to process non-speech sounds and non-verbal speech. With strong capabilities of MLLMs, researchers introduced them into the field of visual content generation [58, 60, 57, 30, 13, 26], world modeling [56, 14, 43], and embodied AI [8, 36, 48, 53]. In image generation and editing, various works [30, 13, 26] use VLM to guide diffusion models. However, in the audio domain, existing methods have not exploited the reasoning capabilities of audio language models.

Diffusion-based Image Editing. In contrast to text-to-image generation, image editing focuses on altering specific elements or attributes within an image while preserving the contents of the remaining image. Diffusion models have been widely used in image editing tasks [7, 21, 27] by altering the inversion process, which produces a latent representation that can reconstruct the image through the generative process. SDEdit [46] first adds noise to the source image, and then subsequently denoises the image through the SDE to produce the target image. P2P [21] adjusts the cross-attention features according to the difference between the source and target captions to generate the target images. Based on this, IP2P [1] finetuned a diffusion model on edit image triplets to enable image editing with simple natural language instructions. Following works [16, 27] further scale up the dataset to support more capable and generalized models. Furthermore, some works [13, 26] explore the usage of vision-language models to guide diffusion models for image editing tasks.

3 Method

3.1 Problem Definition and Notations

Let a_0 denote the original audio waveform, which contains multiple audio events (e.g., *cat meowing*, *rainfall*), as shown in Fig. 1. A high-level editing instruction $\mathcal{P}$ specifies a desired transformation of the audio scene, for example: *"make it sound like a quiet morning in a sunny forest"*. Since $\mathcal{P}$ is abstract and potentially ambiguous, it first needs to be decomposed into a sequence of atomic editing steps $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. Each step s_i either modifies an existing audio event in a_0 or introduces a new event required to fulfill $\mathcal{P}$. We denote the audio after applying step s_i as a_i, where a_0 is the original input and a_n is the final edited audio clip.

Specifically, each step s_i either modifies an existing audio event or introduces a new event required to satisfy $\mathcal{P}$. The atomic editing operations considered in this work are:

- **Add:** Mix a new sound event into the scene (e.g., inject bird chirps).
- **Remove:** Delete or suppress an existing sound event (e.g., remove car engine noise).
- **Extract:** Isolate a particular sound event from the original scene while removing background.
- **Turn volume up/down:** Adjust the volume of a specific event.
- **Change direction:** Modify the spatial location of an event.

For example, to transform the scene into a *sunny forest soundscape*, the atomic edit steps could be:

- *Remove the sound of rain* — inconsistent with a sunny scenario.
- *Add gentle leaf rustling* — enhances the outdoor atmosphere.
- *Turn up the bird chirp* — strengthens the perception of a lively forest.
- *Change the sound direction of the bird chirp to the right* — adds spatial diversity and realism.

The final goal is to produce a target edited audio clip a_n by applying the sequence of edits $\mathcal{S}$ to a_0. Importantly, this editing formulation must preclude shortcut solutions that ignore the original input (e.g., re-generating an entirely new clip from scratch). Instead, the edited audio a_n must preserve all unedited content from a_0 while achieving the requested audio scene transformation.

3.2 SmartDJ Framework

SmartDJ consists of an Audio Language Model (ALM) and a Latent Diffusion Model (LDM). We leverage the ALM to interpret high-level instructions and generate a sequence of atomic editing steps. The LDM then executes these edit steps sequentially to transform the original audio.

Specifically, as illustrated in Fig. 2, the ALM takes the original audio clip a_0 and the high-level editing instruction $\mathcal{P}$ as input, and outputs a sequence of atomic editing steps $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. These editing steps are then executed sequentially by the LDM, producing intermediate results $a_1, a_2, \ldots, a_n$, where a_n is the final edited audio. The overall process is formulated as:

$$\{s_1, s_2, ..., s_n\} = \text{ALM}(a_0; \mathcal{P}) \quad (1)$$

$$a_i = \text{LDM}(a_{i-1}; s_i), i = 1, 2, ..., n \quad (2)$$

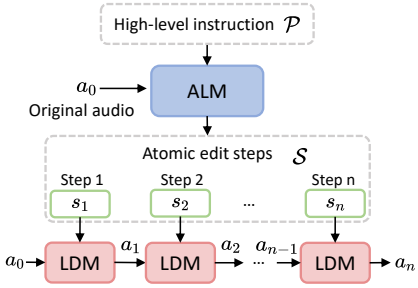


Figure 2: SmartDJ framework overview

3.3 Audio Language Model for Atomic Editing Steps Generation

The Audio Language Model (ALM) takes as input the original audio clip and the high-level editing instruction and generates a sequence of atomic editing steps. As shown at the top of Fig. 3, we first encode a_0 using a pretrained audio encoder (i.e., CLAP [59]) to obtain an audio embedding z_a, which is injected into the ALM via adapter layers. In parallel, the instruction $\mathcal{P}$ is tokenized and encoded as a sequence of embeddings $(p_1, p_2, \ldots, p_k)$, which serve as the textual context for the ALM.

Our ALM is trained in an auto-regressive manner to generate the token sequence corresponding to the atomic editing steps $\mathcal{S}$, by minimizing the following objective:

$$\mathcal{L}_{\text{ALM}} = -\sum_{t=1}^{l} \log P_\theta(r'_t = r_t \mid z_a, r_{1:t-1}, p_{1:k}), \quad (3)$$

where r and r' are the ground truth and predicted text tokens for the atomic editing steps $\mathcal{S}$, l is the length of the tokens, and θ denotes the model parameters. To enable efficient fine-tuning, we freeze the parameters of the CLAP audio encoder, apply Low-Rank Adaptation (LoRA) [24] to a small subset of the LLM layers[18], and fully fine-tune the adapter layers.

3.4 Sequential Stereo Audio Editing with Latent Diffusion Model

The Latent Diffusion Model (LDM) in our framework performs audio editing conditioned on the atomic editing steps $\mathcal{S}$. To support this, we adopt a latent diffusion architecture [20, 50] and extend it to enable editing of stereo audio with spatial effects.

Stereo Audio VAE. Given a stereo audio signal $a \in \mathbb{R}^{2\times L}$, where L is the number of time-domain samples and the leading dimension corresponds to the two audio channels (left and right). The audio Variational Autoencoder (VAE) encodes a into a latent representation $\hat{a} \in \mathbb{R}^{C\times L'}$, where C and L' denote the number of latent channels and the temporal length of the latent sequence. Similar to

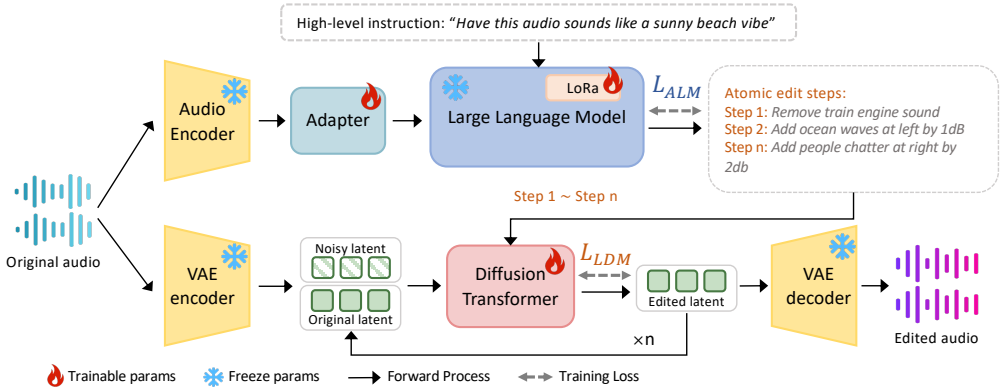


Figure 3: SmartDJ framework. Our method incorporates an audio language model that understands both the original audio and the high-level instructions to generate detailed atomic edit steps. These atomic steps are then fed into a diffusion-based audio editor to edit the audio sequentially.

DAC [33] and Stable Audio Open [11], our audio VAE is based on a 1D-CNN autoencoder with a continuous VAE bottleneck and snake activation functions. We train the VAE on AudioSet [15], augmented with simulated binaural effects to ensure high-quality stereo compression and reconstruction. The resulting latent $\hat{a}$ has $C = 128$ and $L' = L/480$, yielding a compression ratio of $7.5\times$.

Latent Diffusion Model. Our diffusion model conditions on both the text description s_i at the i-th editing step and the latent representation of the audio from the previous step, $\hat{a}_{i-1}$, to generate the updated latent $\hat{a}_i$. We use the FLAN-T5 [6] text encoder $E_{\text{text}}()$ to convert s_i to text embeddings. At each editing step, we initialize a randomly noised latent $\hat{a}'_i \in \mathbb{R}^{C\times L'}$, which is concatenated with $\hat{a}_{i-1}$ to form the input $[\hat{a}_{i-1}; \hat{a}'_i] \in \mathbb{R}^{2C\times L'}$ to the diffusion model. The model is conditioned on $E_{\text{text}}(s_i)$ via cross-attention layers, and the diffusion timestep t is incorporated through a modified AdaLN module [20] to reduce model parameters. We implement a Diffusion Transformer (DiT) that learns to denoise the latent $\hat{a}'_i$ across multiple timesteps by predicting the added noise. Let ϵ denote the true added Gaussian noise, and let $\epsilon_\theta(\cdot)$ be the predicted noise output by the model. The training objective is to minimize the following denoising loss:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),t,s_i,\hat{a}_{i-1},\hat{a}'_i} \left\|\epsilon - \epsilon_\theta(t, E_{\text{text}}(s_i), [\hat{a}_{i-1}; \hat{a}'_i])\right\|_2 . \tag{4}$$

During inference, we use DDIM sampling [51] with classifier-free guidance (CFG), which has proven effective for text-guided generation and editing [23].

3.5 Complex Audio Editing Dataset Curation

Since no public dataset captures complex audio editing conditioned on high-level instructions, we develop a scalable data generation pipeline using large-scale audio databases and off-the-shelf LLMs, as illustrated in Fig. 4. For each data point, we first randomly sample K single-event audio clips from public datasets, each labeled with tags such as {"*car engine*", "*church bell ring*", "*goat bleat*", ...}. We feed these labels into GPT-4o and prompt it to generate a high-level editing instruction $\mathcal{P}$ that transforms the original mix into a new scene (e.g., *"Make this sound like a countryside morning"* or *"Make it sound like a busy train station on a sunny afternoon"*). We then prompt GPT-4o to decompose $\mathcal{P}$ into a sequence of atomic edits $\mathcal{S} = \{s_1, s_2, ..., s_n\}$, including both *add* operations and modifications to existing events (e.g., *remove, turn up/down, change sound direction*).

To generate edit audio pairs, we first synthesize the initial audio a_0 by superposing the K audio clips using LLM-assigned volumes and spatial directions. Spatial effects are rendered with direction-dependent phase and amplitude on two channels. For each atomic edit s_i, we proceed as follows:

- If s_i modifies an existing event, we update its volume level or sound direction.
- If s_i is an *add* operation, we retrieve a new clip from the database with a matching label.

Since each sampled audio event is *independently editable*, an edit step s_i that modifies an existing event can be converted into event-level parameter adjustments, without altering any other sources in the mixture a_{i-1}. For example, to simulate *"remove the car engine sound"* (Fig. 4), we set the car engine clip's volume to zero when generating a_1; to simulate *"add rooster crowing at right"*, we retrieve a rooster clip, apply the specified spatial effect, and superpose it on a_1 to obtain a_2; to

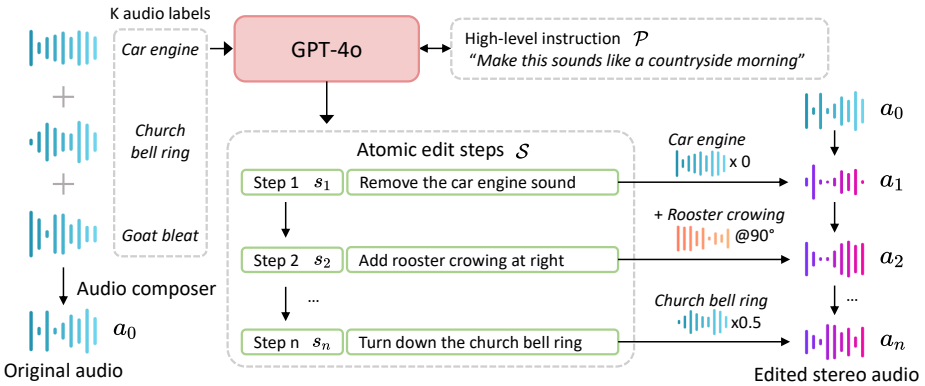


Figure 4: Dataset synthesize pipeline. We first sample audio clips from databases with text labels and compose them into original audio a_0; These text labels are then fed into GPT-4o, which is prompted to generate a high-level instruction $\mathcal{P}$ and the corresponding atomic editing steps $\mathcal{S}$. We compose the target audio $a_1, a_2, ..., a_n$ following the atomic steps sequentially with rule-based editing.

simulate *"turn down the dog bark"*, we reduce the volume of the corresponding clip. This allows us to generate a complete editing trajectory $a_1, a_2, ..., a_n$ by progressively updating event-level parameters and re-composing the audio scene. Our resulting dataset contains high-level editing instructions, atomic edit sequences, and the audio for the full editing trajectory.

4 Experiment

4.1 Experiment setup

Dataset. We use a combination of datasets including AudioCaps [29], VGGSound [2], FSD50k [12], ESC50 [49], and WavCaps [45]. We adopt a series of dataset cleaning pipelines following previous work [20, 52, 55] by filtering events with noisy data labels or low clap scores. Each audio is trimmed or padded into 10 seconds with a sampling rate of 24K. We sample 2-5 audio events and use GPT-4o to create 50k training pairs and 1k evaluation pairs of complex audio editing data to train our audio language model and evaluate the whole complex editing pipeline. We present the keyword of the diverse high-level instructions in Fig 5a. In addition, we expand the size of single-step editing data pairs (s_t, a_{t-1}, a_t) to 0.5M, where each step is an operation of *add, remove, extract, turn up/down, change sound direction* to train our LDM audio editor. Fig 5b shows the proportion of each edit action. Please find more details in the appendix.

Metrics. To evaluate edited audio quality and diversity, we use common metrics in audio generation and editing [39, 55], including Fréchet Distance (FD), Kullback-Leibler divergence (KL), Fréchet Audio Distance (FAD), Inception Score (IS), and Log-Spectral Distance (LSD). We also adopt the CLAP score to measure the semantic similarities between the edited audio and the text prompt. In terms of spatial audio attributions, we calculate GCC MSE (GCC) based on Generalized Cross-Correlation with Phase Transform (GCC-PHAT), and use StereoCRW [3] to produce stereo audio features to evaluate CRW MSE (CRW) and Fréchet Stereo Audio Distance (FSAD).

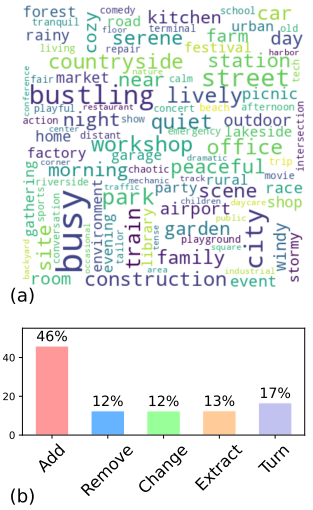


Figure 5: a) Key words in the high-level instructions. b) Proportion of single-step actions.

Baselines. To evaluate the complex audio editing task, we first train an end-to-end version of Audit [55] that directly predicts the final-step audio conditioned on the original audio and high-level instruction in one step without ALM. We extend the mono-channel Audit to our binaural setting by stacking the left and right channels of the mel-spectrograms. We also evaluate common zero-shot editing methods based on the ALM's outputs to perform multi-step sequential editing. The zero-shot methods include SDEdit [46], DDIM Inversion [47], ZETA [44], and AudioEditor [28], where we swap in current audio generation backbones (Details in Appendix B.1). To evaluate sequential editing with a trainable editor, we use an Audit baseline trained on single-step audio editing data. These baseline models are also evaluated on single-step editing task.

Implementation details. Our ALM model is initialized from Audio Flamingo 2 [17] with 3B parameters. Our LDM uses velocity prediction with Zero-SNR, and CFG rescaling technique [38] to adjust the magnitude. Please find more details in the Appendix B.2.

4.2 Results

Results on the complex audio editing task. We first show several inference examples from our ALM module in Fig. 6. The ALM-generated atomic editing steps accurately align both the original audio contents and the high-level editing instructions. The generated steps include actions to remove audio events that are semantically misaligned with the target audio scene. For example, it correctly removes audio event *engine roar* when transferring audio scene into a *busy family home* vibe. It also removes *people whistle* and adds *pages turning* to enhance the immersion of being in an *old library*.

We present the results of the complex audio editing task in Table 1. The first row is the end-to-end Audit baseline that is directly trained on high-level instructions and the final target audio, showing the worst performance overall. Since no prior method can interpret the complex instructions, we use the same set of ALM-generated atomic steps to guide all audio editing models in the multi-step evaluation, including the baselines and our method. We compare the edited audios from each method with 1k reference audios. Our method achieves the lowest metric in FD, FAD, LSD, and comparably

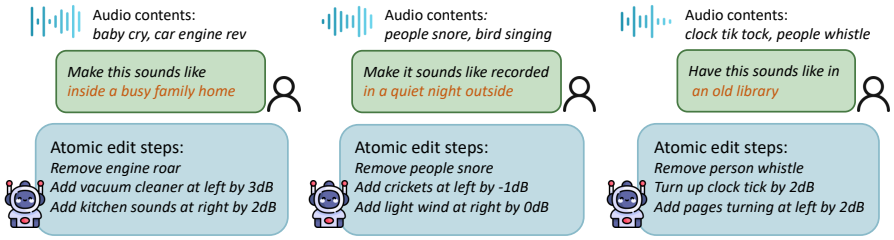


Figure 6: Examples of ALM's output detailed steps. Our ALM module identifies events in the original audio clips and reasons on the given high-level instruction to produce aligned editing steps.

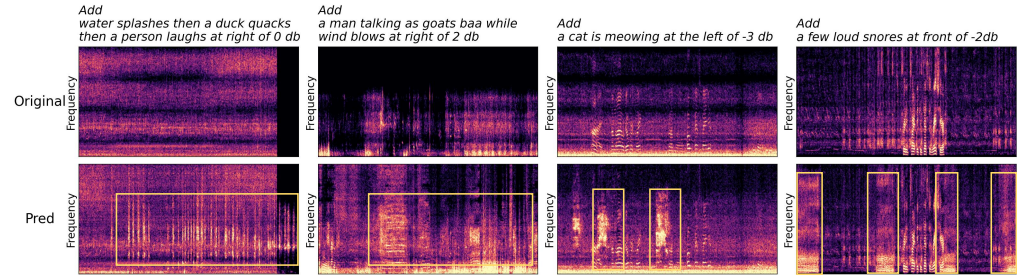


Figure 7: Examples of *Add* operations. Our method can *add* various new audio contents to the original audios. Yellow box highlights the added contents.

low in KL, indicating the smallest discrepancy from the reference audios. It also delivers a high IS metric, showing strong audio quality and diversity. In addition, the highest CLAP score demonstrates the best semantic alignment between the edited audio and the complex text prompts.

Framework	Method	Training	FD ↓	FAD ↓	KL↓	LSD↓	IS ↑	CLAP↑
w/o ALM	Audit [55]	✓	39.6	10.07	3.13	1.96	3.29	0.125
w/ ALM	SDEdit [46]	✗	27.3	3.73	3.26	2.25	6.66	0.188
	DDIM [47]	✗	34.3	9.49	4.07	2.23	3.97	0.076
	ZETA [44]	✗	28.8	3.75	2.93	2.24	7.72	0.224
	AE [28]	✗	27.6	5.02	3.22	2.11	**8.91**	0.211
	Audit [55]	✓	29.4	5.71	**2.81**	1.51	3.95	0.197
	SmartDJ (Ours)	✓	**14.7**	**1.53**	2.85	**1.42**	8.36	**0.238**

Table 1: Quantitative results of the whole pipeline from high-level instructions to audio editing. AE denotes AudioEditor and DDIM denotes DDIM Inversion.

Results on single-step audio editing tasks. We present the results of individual single-step audio editing operations. Table 2a shows results on the *add* task, where new audio contents need to be added to the original one. Our method consistently outperforms baseline methods in the edited audio, including better similarities to the ideal target audio (lowest FD, FAD and KL), and higher quality and diversity shown by the highest IS. Furthermore, stronger spatial metrics (GCC, CRW, and FSAD) also indicate that the stereo audio characteristics are preserved better by SmartDJ.

Table 2a also shows the performance on *remove* and *extract* tasks. The results clearly indicate that our approach delivers the best performance in aligning with the ground truth edited audio across both operations. Furthermore, in Table 2b, we show the results on *turn up/down* and *change sound direction* tasks. Training-free methods are excluded from the evaluations since the original backbones do not provide detailed control over these features. Again, our method shows stronger performance over Audit, which demonstrates SmartDJ has better fine-grained manipulation in audio event properties.

Some examples of spectrogram visualization are shown in Fig 7. More qualitative comparisons can be found in Appendix C.2.

Human evaluations. We evaluate the subjective preference via a user study. We provide users with data pairs consisting of the original audio, the editing prompt, SmartDJ edited result, and edited results from a random competing baseline. We ask the user to select the one that has higher audio quality, the one that has better alignment with the text or spatial instructions, and aligns with the original audio. We conducted extensive evaluations with 19 participants and 20 (10 complex audio editing, 10 single-step editing) data pairs per participant. We separate the evaluation for complex audio editing and single-step editing in Fig 8. In both tasks, SmartDJ is much preferred over all

Method	Add								Remove/Extract						
	FD ↓	FAD ↓	KL ↓	LSD ↓	IS ↑	GCC ↓	CRW ↓	FSAD ↓	FD↓	FAD↓	KL↓	LSD↓	GCC↓	CRW↓	FSAD↓
SDEdit	33.0	3.92	2.59	2.01	4.71	143.3	131.4	0.42	46.6	4.64	2.38	1.89	159.5	157.9	0.45
DDIM	36.9	6.28	2.64	2.02	4.57	131.2	116.1	0.11	53.9	5.69	2.75	1.85	159.6	138.5	0.29
ZETA	37.6	3.64	2.46	1.71	5.04	143.3	127.6	0.52	40.6	3.78	1.92	1.99	161.7	155.9	0.43
AE	30.5	3.66	2.13	1.84	5.51	133.5	145.8	0.55	47.1	3.65	2.43	2.01	164.0	165.6	0.58
Audit	36.5	4.49	1.95	1.42	4.37	145.7	136.3	0.31	54.5	7.56	2.12	**1.65**	189.2	204.5	1.07
SmartDJ	**22.7**	**1.82**	**1.39**	**1.35**	**5.96**	**76.5**	**41.3**	**0.03**	**26.0**	**2.57**	**1.03**	1.71	**15.9**	**5.5**	**0.02**

(a) Audio editing operation *add* and *remove/extract*

Method	Turn Up/Down							Change						
	FD↓	FAD↓	KL↓	LSD↓	GCC↓	CRW↓	FSAD↓	FD↓	FAD↓	KL↓	LSD↓	GCC↓	CRW↓	FSAD↓
Audit	47.1	5.6	1.51	1.04	136.8	139.0	0.89	42.5	4.9	1.34	1.02	170.3	163.5	0.99
SmartDJ	**11.8**	**1.0**	**0.27**	**1.01**	**23.3**	**2.86**	**0.01**	**13.0**	**0.88**	**0.33**	**1.00**	**59.6**	**36.6**	**0.02**

(b) Audio editing operation *turn up/down*, *change sound direction*

Table 2: Quantitative results on all individual audio editing operations.

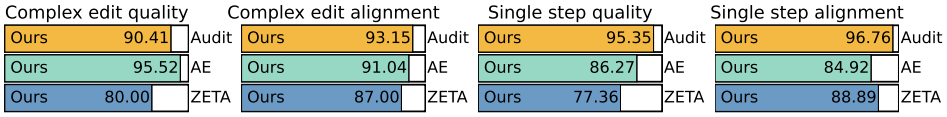


Figure 8: User study results. In terms of audio quality or text/audio alignment, our method is consistently preferred over baselines in both the complex editing task and single-step tasks.

competing methods. Our method delivers the best audio quality and the best alignment with both the complex edit instruction and the single-step instruction from user perception.

4.3 Ablation studies

Audio quality over multi-round editing. Since the complex audio editing task involves a sequence of editing, unchanged content must remain intact after multiple editing steps. We design a "round-trip" edit experiment: we perform the operations "add the sound of $\mathcal{A}$" and "remove the sound of $\mathcal{A}$" on an audio clip for five rounds, with $\mathcal{A}$ a pseudo audio label. For an ideal audio editor, this round-trip operation should exactly reconstruct the original audio. We measure the log spectrogram distance between each round's edited output with the original audio clip in Fig 9. Across all rounds, SmartDJ consistently achieves the lowest LSD, indicating the smallest drift from the original content. This demonstrates that our method effectively preserves the unedited audio content under repeated editing actions

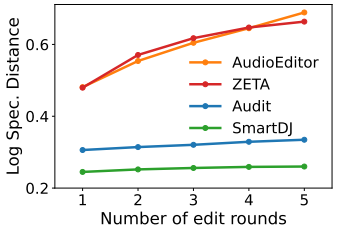


Figure 9: Similarity with original audio after multi-rounds.

Effectiveness of ALM. To evaluate the contribution of ALM in handling complex audio editing tasks, we conduct an ablation by removing the ALM module and training a variant of SmartDJ end-to-end. In this baseline, the LDM is directly conditioned on the high-level instruction and the original audio clip to predict the final edited output. As shown in Table 3, SmartDJ performs significantly worse without the ALM module, highlighting the importance of ALM's intermediate reasoning capabilities. By decomposing complex instructions into a sequence of interpretable atomic edit steps, ALM enables the model to produce semantically coherent edits aligned with the complex instruction and the original audio.

SmartDJ	FD↓	FAD↓	KL↓	LSD↓	IS↑	CLAP↑
w/o ALM	23.6	3.14	2.91	1.84	4.63	0.137
w/ ALM	**14.7**	**1.53**	**2.85**	**1.42**	**8.36**	**0.238**

Table 3: Ablation on the ALM module.

5 Discussion

Conclusion. SmartDJ is the first framework for complex instruction guided stereo audio editing that utilizes the reasoning capability of audio language models. Our approach produces atomic editing steps and executes them sequentially to achieve stereo audio transformations. Evaluations on both subjective audio metrics and human perceptual studies demonstrate that SmartDJ outperforms prior methods, and preserves spatial fidelity in complex scenes.

Limitations. Supporting a new task-specific editing operation on the LDM requires retraining the diffusion model. However, these edits can usually be achieved through a combination of our proposed atomic steps. Besides, our two-stage pipeline depends on a standalone ALM to translate high-level

user instructions into detailed steps. A future direction is an end-to-end joint training strategy that combines reasoning with audio editing.

Acknowledgments

We thank the members of the WAVES Lab at the University of Pennsylvania for their valuable feedback. We are grateful to the anonymous reviewers for their insightful comments and suggestions.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023.

[2] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 721–725. IEEE, 2020.

[3] Ziyang Chen, David F Fouhey, and Andrew Owens. Sound localization by self-supervised time delay estimation. In *European Conference on Computer Vision*, pages 489–508. Springer, 2022.

[4] Ziyang Chen, Prem Seetharaman, Bryan Russell, Oriol Nieto, David Bourgin, Andrew Owens, and Justin Salamon. Video-guided foley sound generation with multimodal controls. *arXiv preprint arXiv:2411.17698*, 2024.

[5] Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv preprint arXiv:2311.07919*, 2023.

[6] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[7] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance, 2022.

[8] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.

[9] Jean-Baptiste Alayrac et al. Flamingo: a visual language model for few-shot learning, 2022.

[10] Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. In *Forty-first International Conference on Machine Learning*, 2024.

[11] Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. *arXiv preprint arXiv:2407.14358*, 2024.

[12] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021.

[13] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models, 2024.

[14] Zhiqi Ge, Hongzhe Huang, Mingze Zhou, Juncheng Li, Guoming Wang, Siliang Tang, and Yueting Zhuang. WorldGPT: Empowering LLM as multimodal world model. In *ACM Multimedia 2024*, 2024.

[15] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 776–780. IEEE, 2017.

[16] Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Han Hu, Dong Chen, and Baining Guo. Instructdiffusion: A generalist modeling interface for vision tasks, 2023.

[17] Sreyan Ghosh, Zhifeng Kong, Sonal Kumar, S Sakshi, Jaehyeon Kim, Wei Ping, Rafael Valle, Dinesh Manocha, and Bryan Catanzaro. Audio flamingo 2: An audio-language model with long-audio understanding and expert reasoning abilities. *arXiv preprint arXiv:2503.03983*, 2025.

[18] Sreyan Ghosh, Zhifeng Kong, Sonal Kumar, S Sakshi, Jaehyeon Kim, Wei Ping, Rafael Valle, Dinesh Manocha, and Bryan Catanzaro. Audio flamingo 2: An audio-language model with long-audio understanding and expert reasoning abilities, 2025.

[19] Yuan Gong, Hongyin Luo, Alexander H Liu, Leonid Karlinsky, and James Glass. Listen, think, and understand. *arXiv preprint arXiv:2305.10790*, 2023.

[20] Jiarui Hai, Yong Xu, Hao Zhang, Chenxing Li, Helin Wang, Mounya Elhilali, and Dong Yu. Ezaudio: Enhancing text-to-audio generation with efficient diffusion transformer. *arXiv preprint arXiv:2409.10819*, 2024.

[21] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. 2022.

[22] Mojtaba Heydari, Mehrez Souden, Bruno Conejo, and Joshua Atkins. Immersediffusion: A generative spatial audio latent diffusion model. *arXiv preprint arXiv:2410.14945*, 2024.

[23] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[24] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[25] Jiawei Huang, Yi Ren, Rongjie Huang, Dongchao Yang, Zhenhui Ye, Chen Zhang, Jinglin Liu, Xiang Yin, Zejun Ma, and Zhou Zhao. Make-an-audio 2: Temporal-enhanced text-to-audio generation. *arXiv preprint arXiv:2305.18474*, 2023.

[26] Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, and Ying Shan. Smartedit: Exploring complex instruction-based image editing with multimodal large language models, 2023.

[27] Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Yuyin Zhou, and Cihang Xie. Hq-edit: A high-quality dataset for instruction-based image editing. *arXiv preprint arXiv:2404.09990*, 2024.

[28] Yuhang Jia, Yang Chen, Jinghua Zhao, Shiwan Zhao, Wenjia Zeng, Yong Chen, and Yong Qin. Audioeditor: A training-free diffusion-based audio editing framework. *arXiv preprint arXiv:2409.12466*, 2024.

[29] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. Audiocaps: Generating captions for audios in the wild. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 119–132, 2019.

[30] Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. Generating images with multimodal language models, 2023.

[31] Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio flamingo: A novel audio language model with few-shot learning and dialogue abilities. *arXiv preprint arXiv:2402.01831*, 2024.

[32] Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio flamingo: A novel audio language model with few-shot learning and dialogue abilities, 2024.

[33] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36:27980–27993, 2023.

[34] Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589*, 2023.

[35] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. Multimodal foundation models: From specialists to general-purpose assistants, 2023.

[36] Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation, 2023.

[37] Jinhua Liang, Huan Zhang, Haohe Liu, Yin Cao, Qiuqiang Kong, Xubo Liu, Wenwu Wang, Mark D Plumbley, Huy Phan, and Emmanouil Benetos. Wavcraft: Audio editing and generation with large language models. *arXiv preprint arXiv:2403.09527*, 2024.

[38] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5404–5411, 2024.

[39] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

[40] Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[41] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[42] Huadai Liu, Tianyi Luo, Qikai Jiang, Kaicheng Luo, Peiwen Sun, Jialei Wan, Rongjie Huang, Qian Chen, Wen Wang, Xiangtai Li, et al. Omniaudio: Generating spatial audio from 360-degree video. *arXiv preprint arXiv:2504.14906*, 2025.

[43] Xinji Mai, Zeng Tao, Junxiong Lin, Haoran Wang, Yang Chang, Yanlan Kang, Yan Wang, and Wenqiang Zhang. From efficient multimodal models to world models: A survey, 2024.

[44] Hila Manor and Tomer Michaeli. Zero-shot unsupervised and text-based audio editing using ddpm inversion. *arXiv preprint arXiv:2402.10009*, 2024.

[45] Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D Plumbley, Yuexian Zou, and Wenwu Wang. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[46] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022.

[47] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models, 2022.

[48] Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. EmbodiedGPT: Vision-language pre-training via embodied chain of thought. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[49] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[51] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[52] Peiwen Sun, Sitong Cheng, Xiangtai Li, Zhen Ye, Huadai Liu, Honggang Zhang, Wei Xue, and Yike Guo. Both ears wide open: Towards language-driven spatial audio generation. *arXiv preprint arXiv:2410.10676*, 2024.

[53] Andrew Szot, Bogdan Mazoure, Harsh Agrawal, R Devon Hjelm, Zsolt Kira, and Alexander T Toshev. Grounding multimodal large language models in actions. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[54] Qwen Team. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[55] Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, et al. Audit: Audio editing by following instructions with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:71340–71357, 2023.

[56] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye HAO, and Mingsheng Long. ivideoGPT: Interactive videoGPTs are scalable world models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[57] Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Zhe Chen, Wenhai Wang, Xizhou Zhu, Lewei Lu, Tong Lu, Ping Luo, Yu Qiao, and Jifeng Dai. VisionLLM v2: An end-to-end generalist multimodal large language model for hundreds of vision-language tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[58] Jianzong Wu, Xiangtai Li, Chenyang Si, Shangchen Zhou, Jingkang Yang, Jiangning Zhang, Yining Li, Kai Chen, Yunhai Tong, Ziwei Liu, and Chen Change Loy. Towards language-driven video inpainting via multimodal large language models, 2024.

[59] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[60] Bin Xia, Shiyin Wang, Yingfan Tao, Yitong Wang, and Jiaya Jia. Llmga: Multimodal large language model based generation assistant. *arXiv preprint arXiv:2311.16500*, 2023.

[61] Manjie Xu, Chenxing Li, Dan Su, Wei Liang, Dong Yu, et al. Prompt-guided precise audio editing with diffusion models. *arXiv preprint arXiv:2406.04350*, 2024.

A Dataset curation process

A.1 Dataset preparation

We construct our training corpus by merging several publicly-available audio dataset, including VGG-Sound, AudioCaps, WavCaps, ESC-50, and FSD50K. Since some of these sets provide audio captions rather than discrete audio label, we first convert every caption to audio labels with GPT-4o-mini API. We only retain single-labeled audio clip and any clip whose caption maps to multiple events is discarded. A CLAP model scores the semantic correspondence between the audio and its new label. Samples with a similarity score below 0.3 are filtered out. The remaining clips from all sources are finally mixed into one large dataset that we use for subsequent data curation.

A.2 Atomic edit actions

We explain the details on creating the single-step atomic edit data pairs. For this single-step audio editing, we have an original audio a_{i-1}, a single atomic edit operation s_i. Base on atomic edit operation s_i, we can generate the edited audio a_i.

Add. Assume the original audio is a mix of audio content $A+B+C$. To *add* a new content into this original audio, we sample a new audio content D from the database and mix the D with the original audio $A+B+C+D$. The atomic template is *"Add the sound of {dog barking} at {right} with {3} db"*. The contents inside the { } can be changed to other sound events or sound attributions. We support various sound direction (left, front and right) and dynamic volume adjustment.

Remove. Given an original mix $A+B+C$, let B be the undesired source. We suppress B so the output becomes $A+C$. THe atomic template is Atomic template: *"Remove the sound of {bird chirping} {at right}"*. The directional phrase in braces is optional. If there are similar audio contents in the same clip, the spatial features enables to manipulate it precisely.

Extract. Starting from the same mix $A+B+C$, we isolate one target source A and mute everything else, yielding only A. The atomic template is *"Extract the sound of {speaking} {at the right}"*. The direction is optional.

Turn up/down To change loudness of a specific source B, we scale it by α, where $\alpha > 1$ is for "up" and $0 < \alpha < 1$ is for "down". The resulted audio clip is $A + \alpha B + C$. The atomic template is *"Turn {up / down} the sound of {engine rev} by {2} dB"*. We also support a dynamic range of volume adjustment.

Change sound direction. We alter only the spatial cues of a specific source C, producing an edited version C' while leaving the other tracks untouched: $A+B+C'$. The atomic template used is *"Change the sound of {baby crying} {from front} to {right}"*. The "from" clause could also be omitted.

A.3 Complex audio editing dataset curation

In the dataset curation process, we first sample 2-5 audio labels in the database, with LLM randomly assigned volume and sound directions. We then call the GPT-4o batch API with sound sources and attributions. For each API call, we provide 15 sets of sound sources. Through API call, each set of sound sources will return a single data pair containing high-level editing instruction and the corresponding atomic editing steps. We follow these atomic editing steps to manually generate the step-by-step target edited audio with the rules in A.2. We generate 50K data pairs for complex audio editing for training. We generate 1K data pairs from AudioCaps test/validation set for evaluation.

Starting from these 50K complex audio editing pairs, we collect the generated corresponding step-by-step atomic actions and produce about 200K single step audio editing pairs (s_i, a_{i-1}, a_i). We further scale up the this dataset size to 500K to train the LDM audio editor and we also generate another 1K extra audio data pairs to evaluate the performance of single step audio editing. This scaled-up single step audio editing dataset keeps some instructions with audio captions, improving the LDM's robustness to different audio contents in the atomic edit instructions.

We provide the details of our *Base Prompt* for dataset curation as follow:

Prompt

You are an expert in spatial audio editing and sound design.

Your task is to generate complex audio editing instructions based on a given list of sound sources (labels). The sound sources will be provided as a list of full sentences (as strings), not character lists. Treat each sentence as a single atomic sound unit. Do not tokenize or split the sound sources into characters.

For example, if you are given: "a baby crying and a man talking; a bird is chirping; dog barking". You should consider "a baby crying and a man talking" as a complete sound source. "a bird is chirping" is another complete sound source and "dog barking" is another sound source. You then generate the step-by-step audio editing instructions based on the given complex instructions.

Task: you need to first brainstorm a complex audio editing instruction

- Imagine a realistic and creative soundscape editing for the given audios.
- You are not limited to the provided audio contents for the editing instruction. And the complex editing instructions should be brief.
- The complex editing instructions could be a soundscape transformation. For example:
  - - Make this sound like it was recorded in a bookstore
  - - Make this sound like a busy coffee shop
  - - Make this sound like a train station
  - - Make this sound like a forest at night
  - - Make this sound like a beach
  - - Make this sound like a sunny day
  - - Craft this sound to feel like a park
  - - Make this audio sound like a quiet farm
  - - Make this audio sound like a firework show
- Your generated complex instructions can be broader than these provided examples. Use your imaginations!
- But remember to keep them brief, the complex editing instructions ideally should not contain the actual sound sources.
- Then, based on the given sound event, give me the detailed editing instructions.

You then generate detailed editing instructions based on the complex audio editing instruction.

- Ensure the editing makes sense (e.g., no waves in a desert, no sheep indoors, rustling leaves in the forest, seas waves in the beach, no raining in the sunny day).
- You are encouraged to remove the original audio contents, but you are required to maintain at least one sound source, not removing all of them.
- Do not split or partially reference a sound source when applying operations.
- Use a combination of simple operations (but NOT necessarily all of them). For example:
  - - Add (e.g., thunderstorm, cat meowing) (for the add operation, you should add up to two additional sources that best align with the complex editing instructions)
  - - Remove (For remove, you should remove at least one sound sources (up to two sound sources), but you must keep at least one sound source!)
  - - Turn up/down (e.g., turn up/turn down the sound of xxx by xxx db (between 0 and 6db))
  - - Change sound direction.
- Each "target" in remove/turn up/turn down/change must exactly match one of the provided "sound sources"
- For the "add" operation:
  - - The "target" must clearly describe the new sound being added (e.g., "crowd chatter", "rain", "footsteps on gravel").
  - - The "effect" should specify the volume and direction (e.g., "at front by 4dB").
  - - Do NOT use "none", "null", or placeholder values as the target. The target must always be a descriptive label of the added sound.
  - - The added sound should not duplicate any original sound source.
- The same operation can be repeated for multiple targets.
- When doing add, you can also have volume and direction attributes
- When editing existing sound sources, you can also have mixed attributes in terms of the volume and sound directions.

- You must ensure each step logically contributes to the final transformation.

Return the output in the following structured JSON format:

```
{
    "sound sources": ["...", "..."], here you should put the sound sources you are given
    "complex editing instruction": "...",
    "atomic editing steps": [
        {"operation": "add", "target": "...", "effect": "at xxx(left, front, right) by xxxdB"},
        {"operation": "remove", "target": "...", "effect": "None"},
        {"operation": "turn up/turn down", "target": "...", "effect": "xxxdB"},
        {"operation": "change", "target": "...", "effect": "to xxx (left, right, or front)"},
    ]
}
```

Do NOT break down sound sources into individual words or characters. Sound sources are complete strings and must remain so in the JSON.

B Implementation Details

B.1 Baselines implementation

We present the following baseline methods to evaluate the complex audio editing task. One of the baselines is an end-to-end version of Audit without using ALM. All other baseline methods perform multi-step sequential editing with ALM's atomic editing step outputs.

End-to-End Audit. We first train an end-to-end version of Audit that directly predicts the final-step edited audio a_n conditioned on the original audio a_0 and high-level instruction $\mathcal{P}$ in one step without ALM. We extend the mono-channel Audit to our binaural setting, where we stack the left and right channels of the mel-spectrograms as the model inputs. Follow the original implementation, we convert 10s of audio into mel-spectrograms with a size of 80×624. using a hop size of 256, a window size of 1024, and mel-bins of size 80. We use the same model configurations in the original paper.

SDEdit. SDEdit is a zero-shot method that does not require training a new audio editing model. It uses an off-the-shelf text-to-audio (TTA) generation model, which we use Stable-Audio-Open, as it supports binaural audio generation. We use the default 200 total diffusion steps and start the reverse process from a timestep of 100. We use a classifier-free-guidance (CFG) scale of 7.5. The target caption is composed by concatenating the individual event captions after each editing step.

DDIM Inversion. Similar to SDEdit, we also use Stable-Audio-Open as the TTA generation model. We use the default 200 total diffusion steps and start the reverse process from a timestep of 100. We use a CFG scale of 7.5 for both the target and the source. The source text prompt and the target text prompt are composed by concatenating the individual event captions before and after each editing step, respectively.

ZETA. Similar to SDEdit and DDIM Inversion, we also use Stable-Audio-Open as the TTA generation model. We use the default 200 total diffusion steps and start the reverse process from a timestep of 100. We use a CFG scale of 7.5 for the target and a CFG scale of 1 for the source. The source text prompt and the target text prompt are composed by concatenating the individual event captions before and after each editing step, respectively.

AudioEditor. Specifically in AudioEditor, we replace Affusion with the spatial generator SpatialSonic in BEWO to support binaural editing. We use the default 100 total editing steps with 5 iterations per step to update text embedding for null-text inversion. For the addition task, we use the default punishment ratio (alpha) of -0.001. For the remove or extract task, we use the default punishment ratio (alpha) of 1.

Audit with ALM. To evaluate sequential editing with a trainable editor, we use an Audit baseline trained on single-step audio editing data with input audio a_{i-1}, atomic instruction s_i, and output edited audio a_i. The model and data configurations are the same with the end-to-end Audit baseline variant.

B.2 SmartDJ implementation

ALM. Our ALM module is initialized from Audio Flamingo 2. This model contains an AF-CLAP audio encoder module to encode the mono channel audio. The input 10 seconds audio is first resampled to 16KHz and transformed into a dense audio features $z_a \in \mathbb{R}^{64\times2048}$. This audio encoder is followed by a representation transformation layers that expand the model capacity. This module has three self-attention layers to the audio feature representation, each with 8 heads and an dimension of 2048. Following this, gated cross-attention layers are used to condition audio representations on the LLM. The LLM uses Qwen2.5-3B, a decoder-only casual LLM with 3B parameters, 36 hidden layers, and 16 attention heads. During training, we keep both the AF-CLAP and LLM frozen during training. The audio representation transformation layers are fully optimized. We apply LoRA only to the gated cross-attention layers with rank of 16. We fine-tune ALM for 20 epochs with a batch size of 24.

DiT architecture We conduct experiments at 24KHz audio sample rate. The waveform latent representation operates at 50Hz and consists of 128 channels. Our DiT model has 24 blocks, each with a channel of 1024, 16 attention heads, and an FFN (Feed-Forward Network) dimension of 4096, resulting in total of 597M parameters. During training, we begin with a 50K linear warm-up phase and the result is training with a cosine learning rate decay. In the inference, we use 100 diffusion

steps and a guidance scale of 4, with guidance rescale of 0.8. our LDM uses velocity prediction with Zero-SNR, and CFG rescaling technique [38] to adjust the magnitude of the predicted velocity and avoid over-exposure. It is trained on single-step editing data with a batch size of 256 of 500k iterations. 10% text is replaced with empty strings to enable unconditional modeling. The learning rates for the ALM and LDM training is 1e-5 and 5e-5 with AdamW. All experiments are conducted with four NVIDIA L40S GPUs.

C Additional results

In this section, we provide more qualitative results on the ALM-inferenced editing steps and the LDM edited audio results.

C.1 Results on ALM-inferenced atomic editing steps

We show multiple examples on ALM-inferenced atomic editing steps. In each example, the "audio labels" are the ground truth audio labels for reference and they are not provided to the ALM module. Only the corresponding audios are provided to the ALM. We then input the high level instruction to the ALM and it output the atomic editing steps.

Prompt

Example 1: Make this sound like a quiet afternoon in a garden Audio labels: clock tick, bird chirp, wind

High-level instructions: Make this sound like a quiet afternoon in a garden

Atomic editing steps:

- Remove the sound of clock tick
- Turn up the the sound of bird tweet by 3dB
- Add the sound of gentle breeze at front by 2dB

Prompt

Example 2: Make this sound like a busy city street Audio labels: engine rev, church bell ring

High-level instructions: Make this sound like a busy city street

Atomic editing steps:

- Remove the sound of bell ring
- Turn down the sound of engine rev by 2dB
- Add the sound of distant siren at left by 2dB
- Add the sound of traffic noise at front by 3dB

Prompt

Example 3: Make this sound like a bustling park on a sunny day. Audio labels: children scream, insect buzz, bird call, chainsaw run

High-level instructions: Make this sound like a bustling park on a sunny day.

Atomic editing steps:

- Remove the sound of chainsaw run
- Add the sound of laughter at left by 3dB
- Turn down children scream by 2dB
- Change the sound of bird call to front

Prompt

Example 4: Make this sound like a serene beach Audio labels: water waves, whistling, music playing

High-level instructions: Make this sound like a serene beach

Atomic editing steps:

- Remove the sound of whistling
- Turn up the sound of wave crash by 4dB
- Add the sound of seagulls calling at front by 3dB

Prompt

Example 5: Make this sound like a busy office Audio labels: typewriter tap, power tool drill

High-level instructions: Make this sound like a busy office

Atomic editing steps:

- Remove the sound of drilling
- Turn up the sound of typewriter type by 2dB
- Add the sound of phone ringing at right by 3dB

Prompt

Example 6: Make this sound like a military base Audio labels: fire truck siren, gunshot, vehicle move

High-level instructions: Make this sound like a military base

Atomic editing steps:

- Remove the sound of fire truck siren
- Turn up the sound of gunshot by 3dB
- Add the sound of helicopter blades at right by 2dB

Prompt

Example 7: Make this sound like a calm night in the countryside Audio labels: snoring, machine run, wind blow

High-level instructions: Make this sound like a calm night in the countryside

Atomic editing steps:

- Remove the sound of machine run
- Turn down the sound of wind blow by 2dB
- Add the sound of owl hooting at right by 2dB

Prompt

Example 8: Make this sound like a bustling urban street Audio labels: horn honk, cat, siren blare, male laugh

High-level instructions: Make this sound like a bustling urban street

Atomic editing steps:

- Remove the sound of cat
- Turn down the sound of siren blare by 2dB
- Add the sound of people chatter at front by 3dB
- Add the sound of traffic noise at left by 2dB

C.2 Results on atomic editing steps

We provide more atomic editing results on *add* in Fig 10. As shown by the comparison with the original audio and the edited one, while baseline method tends to replace the original clips and completely generate a new audio clip, SmartDJ can successfully keep the original contents and *add* new sound events into it.

We show more editing examples on *remove* and *extract* in Fig 11 and 12. Compared with baseline method, SmartDJ can effectively either remove unwanted or extract wanted audio contents. The edited results show good alignment with the ground truth edited audios.

Figure 13 presents additional qualitative results for the *change sound direction* task. The y axis in each figure is the sound direction heatmap. SmartDJ consistently relocates the source to the

requested spatial direction, and its outputs align well with the ground truth edited audios. By contrast, Audit can not alter spatial effect, showing the limitations of using spectrogram audio encoder for direction-aware editing.

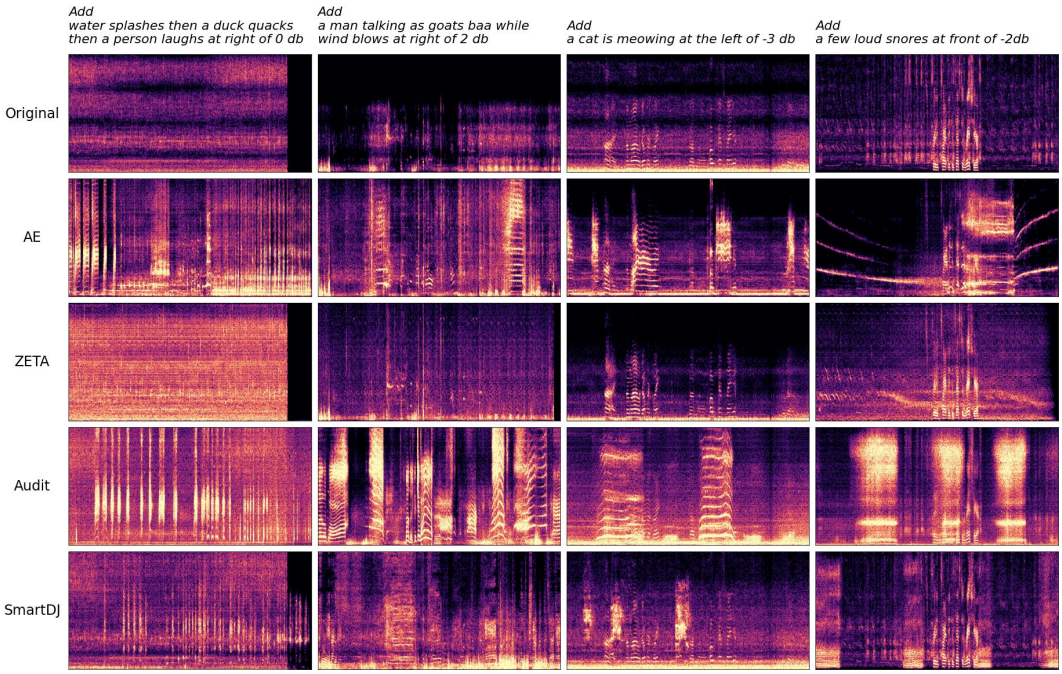


Figure 10: Examples on *add* operation. The top row is the original audio and the rest rows are the edited results. Only SmartDJ can keep the original audio clips while add new audio contents.

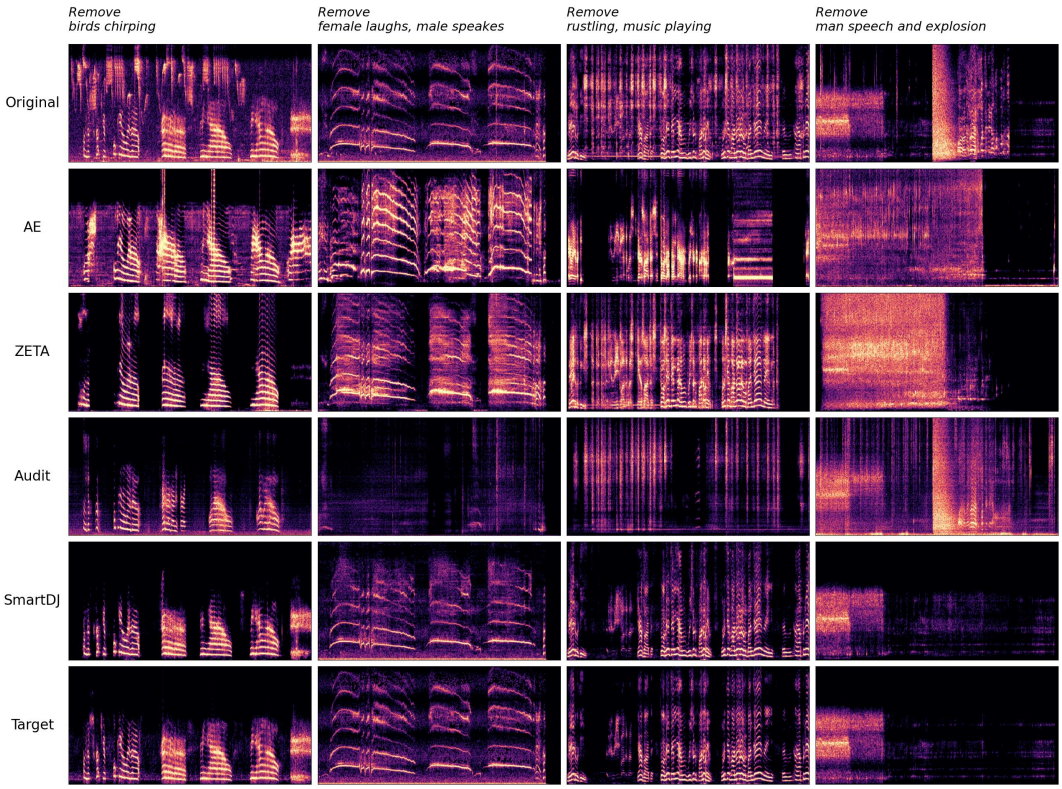


Figure 11: Examples on *remove* operation. The top row is the original audio and the bottom row is the target audio. Only SmartDJ can completely remove unwanted audios parts and keep the remaining part unchanged.

C.3 Human Subjective Studies Details

We provide more details on the subjective user study. We provide users with data pairs consisting of the original audio, the editing instruction, the SmartDJ edited result, and the edited results from a random competing baseline. In the case of single-step audio editing tasks *remove, extract, turn up/down, change sound direction* where there exists a ground-truth editing solution, we also provide it as the reference audio as shown in Fig. 14b. We conduct evaluations with 19 participants and 20 (10 complex audio editing, 10 single-step editing) data pairs per participant.

In the 10 complex audio editing question pairs, we ask the user to select between the SmartDJ edited audio and the edited results from a random competing baseline (randomly sampled from AudioEditor, ZETA and Audit) according to the following three questions:

Question list for complex audio editing user study

- Between Audio 1 and Audio 2, which one do you feel aligns with the original audio and the editing instruction better?
- Between Audio 1 and Audio 2, which one do you feel has the better audio quality?
- Between Audio 1 and Audio 2, which one do you feel better preserves some of the original audio's contents?

In the single-step editing for *add* operation (3 pairs in total), we compare SmartDJ with a randomly sampled method from three baselines (AudioEditor, ZETA and Audit). We ask the user to answer four questions, with one additional question listed below:

Additional question for *add* operation user study

- Between Audio 1 and Audio 2, which one do you feel the added spatial effect aligns with the text instruction?

For the single-step *remove* and *extract* tasks (4 pairs) we compare SmartDJ with a randomly chosen baseline model. For *turn-up/turn-down* and *change direction* (3 pairs) we benchmark only against

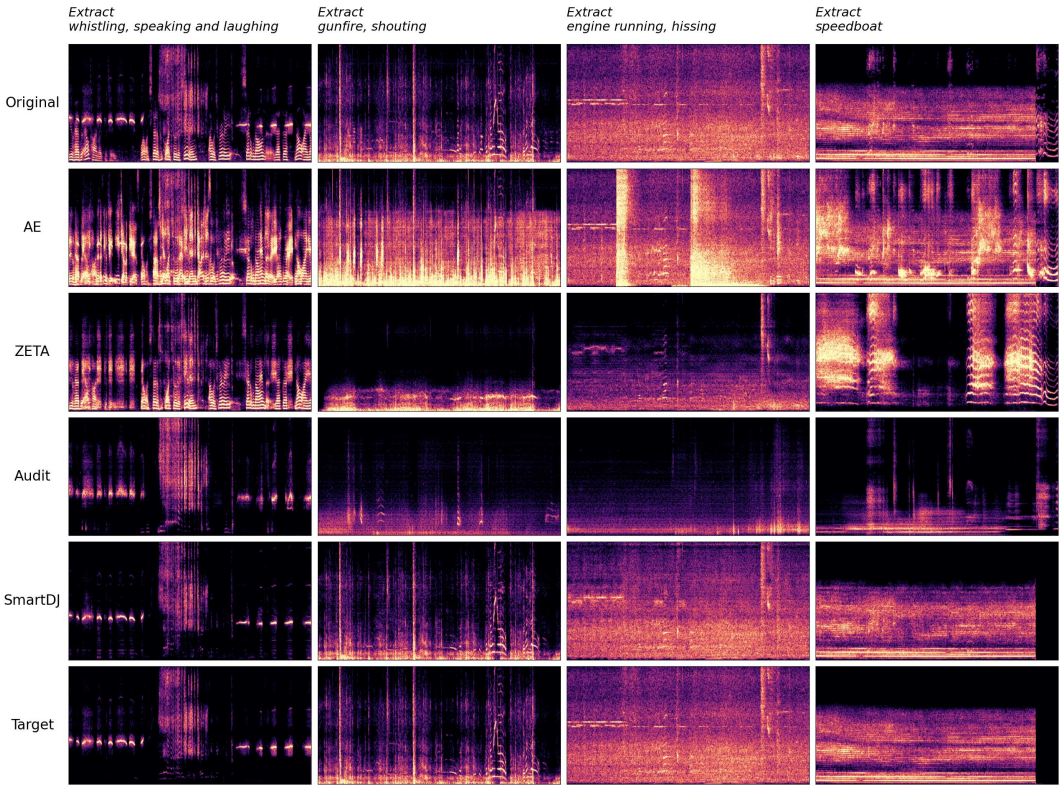


Figure 12: Examples on *extract* operation. The top row is the original audio and the bottom row is the target audio. Only SmartDJ can extract wanted audios that are clean and of high quality.

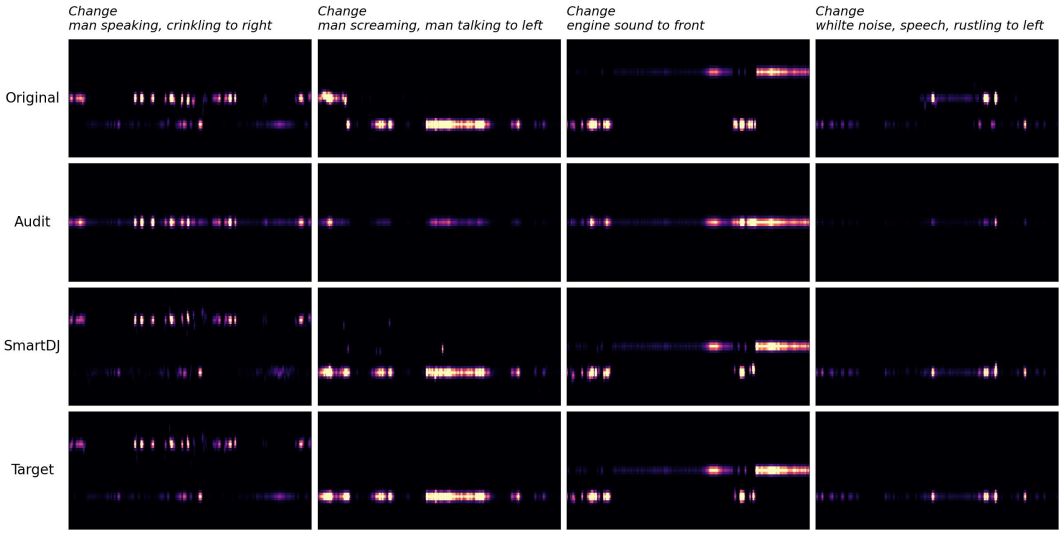
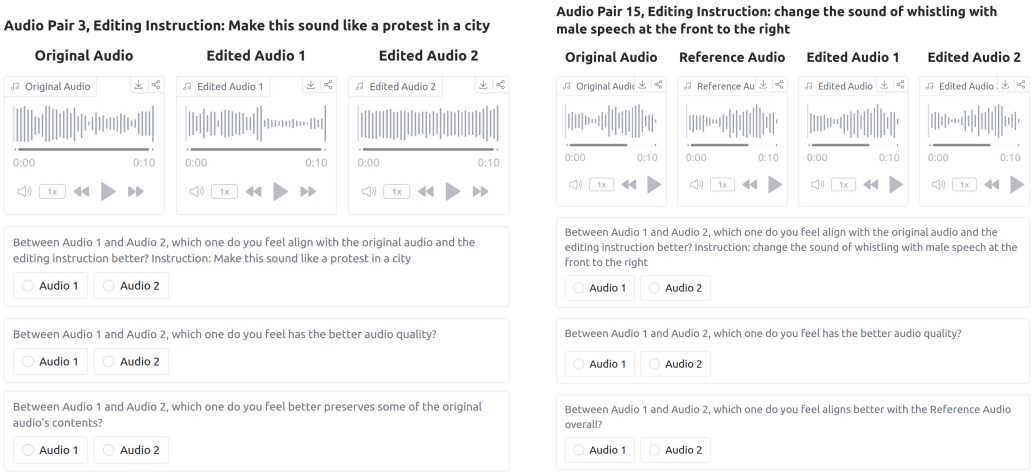


Figure 13: Examples on *change* operation. The top row is the original audio and the bottom row is the target audio. Only SmartDJ can perfectly edit the sound directions that are matching closely with the target.

AUDIT, since the other baselines cannot perform these operations. Each pair includes a ground-truth reference, and listeners answer the three evaluation questions listed below.

Questions list for the *Remove, Extract, Turn up/down, Change sound direction* operation user study

- Between Audio 1 and Audio 2, which one do you feel aligns with the original audio and the editing instruction better?
- Between Audio 1 and Audio 2, which one do you feel has the better audio quality?



(a) User interface for complex audio editing task

(b) User interface for change direction editing task

Figure 14: The audio pairs, questions, and user interfaces for different audio editing tasks

- Between Audio 1 and Audio 2, which one do you feel aligns better with the Reference Audio overall?

Across all tasks, SmartDJ is consistently preferred over all baseline methods. For complex audio editing task, SmartDJ receives at least 80% of user votes over the baselines for audio quality, and at least 87% for alignment with the high-level editing instruction and original audio. This shows that SmartDJ faithfully performs the requested scene transformation while preserving the key elements of the original audio. In the single-step editing task, our method receives more than 77% of user votes for audio quality, and 84% for alignment with the single-step editing text prompt, spatial description, and original or reference audio (when applicable). These results demonstrate that SmartDJ achieves the highest user preference across both quality and alignment metrics, outperforming all competing methods.

D Social impact

SmartDJ can broaden access to high-quality spatial sound design for creators who lack studio resources, lowering the barrier for indie games, VR storytelling, language-learning apps, and accessibility tools for the blind. Educators can synthesize realistic acoustic scenes for teaching. In virtual meeting platforms, our method can be used to suppress or enhance ambiance to reduce listener fatigue. On the other hand, the same technology could fabricate deceptive ambiance, e.g. fake emergency sirens. To mitigate this issue, one could embed inaudible watermarks in edited audio segment enabling provenance tracing.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state our contributions in the abstract and introduction and conduct extensive experiments to support our claim.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide the limitations in the Discussion session.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [No]

Justification: Our work does not involve any mathematical assumption.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We clearly detailed our methods and parameters for the training of the network. The code and the dataset will be open-sourced once the paper is accepted.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Our code and dataset will be open-sourced after the review process.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have all the training and testing details, including data splits, hyperparameters and the description of the used optimizer.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We exclude error bars because it would be too computationally expensive.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have all details in the experiment setup in the main tex and the supplementary results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We adhere to the NeurIPS Code of Ethics and preserve anonymity.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We mention potential societal impacts in the appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We discuss the safeguards against the misuse of our audio language models and editing models in the supplementary materials.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the corresponding paper when needed and follow the corresponding license when using public datasets and the license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We discuss the model architecture details, training details, dataset composition process in the paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Our paper finetuned an audio language model to interpret the complex editing instructions. LLMs are also used to create the dataset, as declared in the paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.