WAVCRAFT: AUDIO EDITING AND GENERATION WITH LARGE LANGUAGE MODELS

Jinhua Liang[1], **Huan Zhang**[1], **Haohe Liu**[2], **Yin Cao**[3], **Qiuqiang Kong**[4], **Xubo Liu**[2], **Wenwu Wang**[2], **Mark D. Plumbley**[2], **Huy Phan**[5]*, **Emmanouil Benetos**[1,6]
[1] Centre for Digital Music (C4DM), Queen Mary University of London, UK
[2] Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK
[3] Xi'an Jiaotong Liverpool University [4] The Chinese University of Hong Kong
[5] Amazon, Cambridge, MA, USA [6] The Alan Turing Institute, UK
jinhua.liang@qmul.ac.uk

ABSTRACT

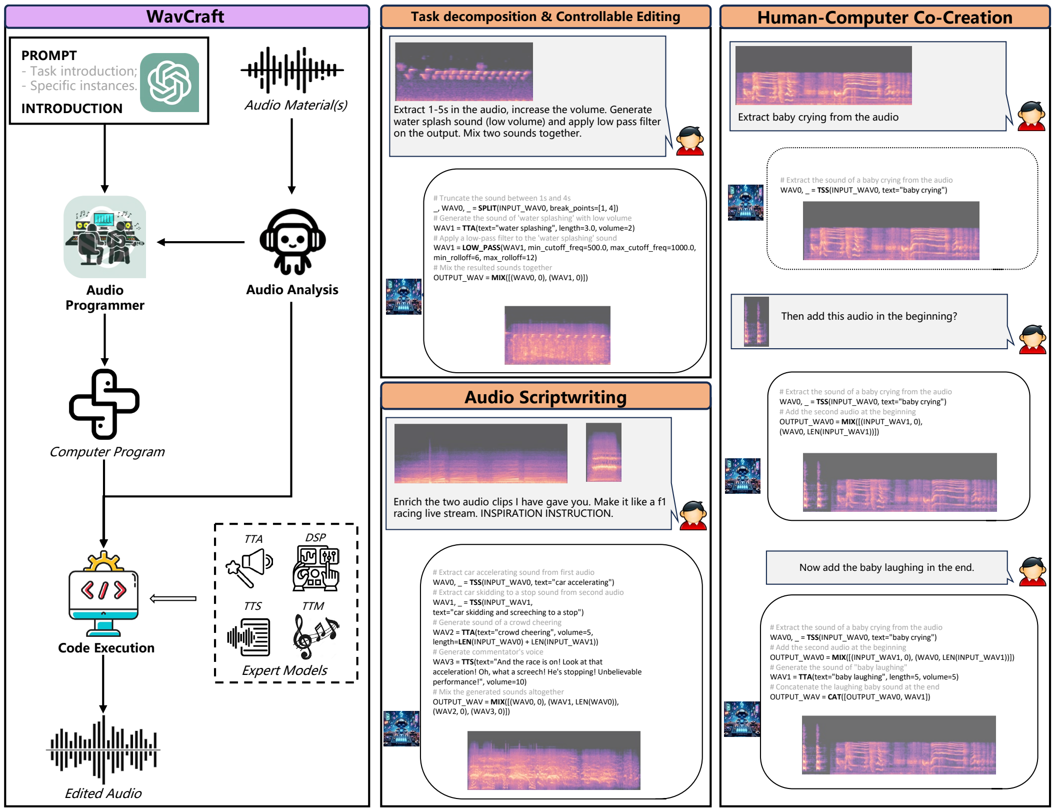
We introduce WavCraft, a collective system that leverages large language models (LLMs) to connect diverse task-specific models for audio content creation and editing. Specifically, WavCraft describes the content of raw audio materials in natural language and prompts the LLM conditioned on audio descriptions and user requests. WavCraft leverages the in-context learning ability of the LLM to decomposes users' instructions into several tasks and tackle each task collaboratively with the particular module. Through task decomposition along with a set of task-specific models, WavCraft follows the input instruction to create or edit audio content with more details and rationales, facilitating user control. In addition, WavCraft is able to cooperate with users via dialogue interaction and even produce the audio content without explicit user commands. Experiments demonstrate that WavCraft yields a better performance than existing methods, especially when adjusting the local regions of audio clips. Moreover, WavCraft can follow complex instructions to edit and create audio content on the top of input recordings, facilitating audio producers in a broader range of applications. Our implementation and demos are available at this https://github.com/JinhuaLiang/WavCraft.

1 INTRODUCTION

Large language models (LLMs) such as ChatGPT (OpenAI, 2023) have remarkably promoted the development of artificial intelligence-generated content (AIGC). Driven by large-scale pre-training on massive high-quality textual tokens and reinforcement learning from human feedback (RLHF), LLMs demonstrate advanced capacity in language analysis, rationale, and interaction. While LLMs have attracted increasing amount of attention on topics such as chain-of-thought (Wei et al., 2022), interpretability (Zhao et al., 2024), and in-context learning (Wei et al., 2023), they are limited to textual data and fail to engage with a broader range of AIGC tasks.

AI-empowered agents have been devised to tackle more practical applications by equipping LLMs with task-specific modules (Qian et al., 2023). These agents (Shen et al., 2023; Huang et al., 2024) use the LLM to interpret a user query to some basic tasks and call task-specific modules (namely expert models) with an appropriate order. By using a modular approach, AI-driven agents are capable of solving intricate tasks without the requirement of additional training. In the audio domain, WavJourney (Liu et al., 2023d) proposed an AI-driven agent that synthesises an audio clip by connecting speech, audio, and music generative models. An audio script is created based on user instructions and compiled into an executable computer program. The computer program then invokes various audio generative models to synthesize a recording. Despite its success, the current audio agents cannot use audio clips as input, hindering themselves from a broader range of audio generation applications. Considering the collaborative ability of LLMs and the real-world need for multimodal interactive creation, a natural question arises: *can we improve audio agents with the ability of audio analysis and transformation?*

*The work does not relate to H.P.'s position at Amazon.



Extract 1-5s in the audio, increase the volume. Generate water splash sound (low volume) and apply low pass filter on the output. Mix two sounds together.

```
# Truncate the sound between 1s and 4s
_, WAV0, _ = SPLIT(INPUT_WAV0, break_points=[1, 4])
# Generate the sound of 'water splashing' with low volume
WAV1 = TTA(text="water splashing", length=3.0, volume=2)
# Apply a low-pass filter to the 'water splashing' sound
WAV1 = LOW_PASS(WAV1, min_cutoff_freq=500.0, max_cutoff_freq=1000.0,
min_rolloff=6, max_rolloff=12)
# Mix the resulted sounds together
OUTPUT_WAV = MIX([(WAV0, 0), (WAV1, 0)])
```

Enrich the two audio clips I have gave you. Make it like a f1 racing live stream. INSPIRATION INSTRUCTION.

```
# Extract car accelerating sound from first audio
WAV0, _ = TSS(INPUT_WAV0, text="car accelerating")
# Extract car skidding to a stop sound from second audio
WAV1, _ = TSS(INPUT_WAV1,
text="car skidding and screeching to a stop")
# Generate sound of a crowd cheering
WAV2 = TTA(text="crowd cheering", volume=5,
length=LEN(INPUT_WAV0) + LEN(INPUT_WAV1))
# Generate commentator's voice
WAV3 = TTS(text="And the race is on! Look at that
acceleration! Oh, what a screech! He's stopping! Unbelievable
performance!", volume=10)
# Mix the generated sounds altogether
OUTPUT_WAV = MIX([(WAV0, 0), (WAV1, LEN(WAV0)),
(WAV2, 0), (WAV3, 0)])
```

Extract baby crying from the audio

```
# Extract the sound of a baby crying from the audio
WAV0, _ = TSS(INPUT_WAV0, text="baby crying")
```

Then add this audio in the beginning?

```
# Extract the sound of a baby crying from the audio
WAV0, _ = TSS(INPUT_WAV0, text="baby crying")
# Add the second audio at the beginning
OUTPUT_WAV0 = MIX([(INPUT_WAV1, 0),
(WAV0, LEN(INPUT_WAV1))])
```

Now add the baby laughing in the end.

```
# Extract the sound of a baby crying from the audio
WAV0, _ = TSS(INPUT_WAV0, text="baby crying")
# Add the second audio at the beginning
OUTPUT_WAV0 = MIX([(INPUT_WAV1, 0), (WAV0, LEN(INPUT_WAV1))])
# Generate the sound of "baby laughing"
WAV1 = TTA(text="baby laughing", length=5, volume=5)
# Concatenate the laughing baby sound at the end
OUTPUT_WAV = CAT([OUTPUT_WAV0, WAV1])
```

Figure 1: WavCraft overview. WavCraft processes the content of input audio clips and prompts the LLM to generate the code conditioned on user query and audio content. The generated code is then implemented as a computer program empowered by a set of expert models. WacCraft is capable to tackle cases involving: 1) task decomposition; 2) controllable editing; 3) human-computer co-creation; and 4) audio scriptwriting.

In this work, we introduce WavCraft, an AI-empowered audio agent that leverages LLMs, together with a variety of expert models, to edit and generate audio content based on human instructions and available audio materials. Given a few audio references, WavCraft analyses the content of audio references with an audio analysis module to produce audio description. The audio description and the user query are wrapped up with a pre-defined instruction template and directed to an audio programmer module. The audio programmer applies an LLM to break down a complex audio content creation task into several basic ones, and generates an executable program to invoke modules like audio expert models, DSP functions, or logic operations. Following this modular approach, WavCraft is able to assemble a variety of audio creation tools with great flexibility. The overall framework of WavCraft and its featured use cases are shown in Figure 1.

Features of WavCraft include: 1) Adjustable: WavCraft can take available audio clips as raw materials and create audio content based on both user instruction and input audio. Compared with existing audio agents (Liu et al., 2023d), WavCraft is capable of a broader range of audio content creation, such as sound infilling and removal; 2) Modular: WavCraft can break down a comprehensive instruction into several basic audio tasks and thus handle a wide range of audio content generation tasks. In addition, the decompositional framework of WavCraft presents an explicit pathway of content creation, enhancing the explainability in the eyes of users; 3) Interactive: Exploiting the language analysis ability of the LLM, WavCraft interacts with users in multiple dialogues. During multi-round co-creation, the generated audio clips stay consistent with each other; 4) Creative: Based on the analysis of audio content and user's blueprint, WavCraft leverages the LLM to narrate a story, infers instructions for expert models, and creates the audio content that fulfill the storyline. We refer to such ability of WavCraft to generate audio content without explicit user instruction as *audio scriptwriting*.

The contributions of this paper are summarised as follows:

- An LLM-driven audio agent named WavCraft is proposed to create or edit audio content based on user instructions and available audio clips.
- By coordinating various generative models, WavCraft produces audio content in a controllable manner. Our experiments demonstrate that WavCraft achieves better performance on audio generation and editing compared to current models.
- Additional experiments are conducted to evaluate WavCraft's ability of audio scriptwriting where models should manipulate audio content without explicit user commands. We hope this will facilitate the process of audio production.

2 Related work

Language models for multi-modal tasks. Language models such as ChatGPT (OpenAI, 2023) and LLaMA (Touvron et al., 2023) have achieved considerable progress in natural language processing. These models, featuring billions of parameters, are trained with massive high-quality training data to handle a variety of text-related tasks with a single model. To extend the open-world knowledge to more domains, subsequent works (Alayrac et al., 2022; Li et al., 2023) have aligned audio, video, and/or image to text and proposed multi-modal foundational models in their respective domains. While multi-modal language models achieved state-of-the-art performance in downstream tasks, most of them are restricted to common text-guided generation tasks, such as text-to-audio synthesis (Liu et al., 2023a), where the model is not required to analyse complex user instructions.

LLM-based agents. More recently, LLM-based agents have attracted an increasing amount of attention to tackle challenging, intricate applications by integrating language models with a set of task-specific models. Toolformer (Schick et al., 2023) was trained to decide when and where an API should be called and how to assemble outputs from different APIs. HuggingGPT (Shen et al., 2023) applied ChatGPT (OpenAI, 2023) as a controller to allocate existing neural networks in Huggingface (Shen et al., 2023). HuggingGPT is thus capable of solving diverse AI tasks across natural language, visual, and audio domains. Meanwhile, ViperGPT (Surís et al., 2023), VisProg (Gupta & Kembhavi, 2023), and RVP (Ge et al., 2023) have demonstrated the promise of visual agents on image/video analysis tasks. LLaVA-Plus (Liu et al., 2023b) extended the input query to visual domain by replacing LLM with visual language model (VLM). In the audio domain, AudioGPT (Huang et al., 2024) connected multiple audio neural networks and used ChatGPT to classify the user query into a predefined task. WavJourney (Liu et al., 2023d) used an LLM to screenwrite the audio scripts and then generate audio clips by calling diverse audio generative models. Although considerable progress has been made in previous works to extend open-world knowledge in language models to multiple modalities, few of them can be prompted by non-text inputs, restricting their ability to many practical applications, especially audio editing.

Audio Creation and Editing. Audio creation and editing are challenging parts of generative AI since they require models to not only understand the audio content but also modify the audio conditioned on input instructions. With the development of deep learning, generative models have demonstrated remarkable capacities to synthesise speech (Wang et al., 2023), audio (Liu et al., 2023a; Kreuk et al., 2023; Borsos et al., 2023; Vyas et al., 2023), and music (Copet et al., 2023; Agostinelli et al., 2023). Existing audio generation methods are mainly dedicated to synthesising audio conditioned on different types of prompts, such as text description, voice style, and music melody. However, these methods are trained to generate audio from scratch and thus are ill-suited to editing tasks on existing audio. Recently, AUDIT (Wang et al.) was proposed to learn an end-to-end diffusion model to modify the audio content based on both text instructions and input audio. While AUDIT is capable of various basic editing tasks, including adding, removal, replacement, super-resolution, and infilling, it suffers from two drawbacks: 1) it does not perform well in complex editing tasks that combines these basic tasks; and 2) it cannot perform local changes on designated audio regions, limiting its application in real-world scenario.

Table 2: List of the audio APIs and their implementation used by WavCraft.

Task	**Input**	**Output**	**API Name**	**Model name**
task-specific models for audio manipulation				
Text-to-Audio	Text	Audio	TTA	AudioGen (Kreuk et al., 2023)
Text-to-Speech	Text	Audio	TTS	Bark [4]
Text-guided Source Separation	Text, Audio	Audio	TSS	AudioSep (Liu et al., 2023c)
Extract	Audio	Audio	EXTRACT	AudioSep (Liu et al., 2023c)
Text-to-Music	Text	Audio	TTM	MusicGen (Copet et al., 2023)
Super resolution	Audio	Audio	SR	AudioSR (Liu et al., 2024)
Drop	Text	Audio	DROP	AudioSep (Liu et al., 2023c)
Inpaint	Audio	Audio	INPANT	AudioLDM (Liu et al., 2023a)
basic audio processing functions				
Mix	Audio	Audio	MIX	`numpy.add`
Length	Audio	Text	LEN	`len`
Concatenate	Audio	Audio	CAT	`numpy.concatenate`
Clip	Audio	Audio	CLIP	`numpy.ndarray`
Adjust Volume	Audio	Audio	ADJUST_VOL	`torchaudio.Vol`
Low Pass	Audio	Audio	LOW_PASS	`audiomentations.LowPassFilter`
High Pass	Audio	Audio	HIGH_PASS	`audiomentations.HighPassFilter`
Room Simulate	Audio	Audio	ROOM_SIM	`audiomentations.RoomSimulaor`
Impulse Response	Audio	Audio	ADD_RIR	`audiomentations.ApplyImpulseResponse`

3 WavCraft

3.1 Overall framework

WavCraft is an LLM-driven system equipped with a set of task-specific audio networks, capable of audio editing and creation. The overall framework of WavCraft can be found in Figure 1, highlighting three core designs: 1) *audio analysis*: WavCraft initially describes the content of audio clips using natural language; 2) *task decomposition*: given an user query and input audio descriptions, WavCraft formulates a set of instructions from a predefined template by prompting ChatGPT (OpenAI, 2023) directly; 3) *code execution*: WavCraft calls the APIs of expert audio models to execute the generated computer program. We will detail these core designs in the following:

Audio analysis. Complex audio editing requires models to modify audio clips based on user queries and the content of input recordings. Therefore, WavCraft applies the audio analysis module to describe input audio clips in natural language. We apply an audio question and answering model to describe sounds using a template question "write an audio caption to describe the sound". Please note that WavCraft can integrate any models for audio question and answering (Liang et al., 2023; Deshmukh et al., 2023), or even an audio captioning model (Mei et al., 2023), as the audio analysis module.

Task decomposition. The audio programmer module in WavCraft then drives the LLM to generate an executable script conditioned on both the user query and the content of input audio clips. Specifically, WavCraft fills a pre-defined template with the input query and audio description and then directs the instructions to the LLM (see more in the Appendix A). Compared with other AI programmers (Gupta & Kembhavi, 2023), WavCraft generates not only the code but also the comment for each line and the audio script. We found that these comments and the audio script facilitate the audio programmer module to generate code step by step, leading to high-quality output and explainable operations.

Code execution. WavCraft executes the generated scripts by calling a set of audio expert models. Table 2 lists the APIs constituting WavCraft. WavCraft consists of various publicly-available expert models: AudioGen (Kreuk et al., 2023) was used for text-to-audio generation; MusicGen (Copet et al., 2023) was adopted to music synthesis due to its high-fidelity performance based on text and/or melody. For text-to-speech generation, we use Bark[5], a state-of-the-art model that generates matched speech conditioned on the tone, pitch, emotion, and prosody of a given voice preset. For

[5]https://huggingface.co/spaces/suno/bark

text-guided source separation, AudioSep (Liu et al., 2023c) is used to separate targeted sound tracks conditioned on language queries. AudioSR (Liu et al., 2024) and AudioLDM (Liu et al., 2023a) are used for super-resolution and audio infilling, respectively. In addition, a series of DSP modules are introduced as well. We implement the DSP modules by using torchaudio (Yang et al., 2022) and audiomentations [6]. It is noteworthy that these task-specific modules can be easily replaced with the alternative architectures.

3.2 FEATURES

Empowered by LLMs, WavCraft is capable of intricate audio editing and creation tasks. WavCraft highlighted four advanced features:

Modular operations. WavCraft decomposes a user instruction into several basic tasks and thus is capable of more complex editing applications in an explainable manner.

Controllable editing. WavCraft translates user requests into executable lines such that it can edit the targeted attributes while keeping the rest unchanged.

Human-AI co-creation. WavCraft leverages large language models to edit audio in a interactive manner, facilitating human producers to create audio content through multi-round refinement. Moreover, WavCraft generates the audio script and comment lines to explain the process of audio content creation. This chain-of-thought method improves the interpretability and transparency of WavCraft.

Audio scriptwriting. Beyond audio content generation under explicit user guidance, WavCraft can produce the sounds in a creative approach, following a high-level outline. We name this ability to devise a plot itself and then manipulate the audio content as *audio scriptwriting*. To make an audio drama, WavCraft creates a script conditioned on input audio, together with the outline, and then sonifies the script with a variety of expert models.

4 TASKS

WavCraft provides a flexible framework that can address a diverse range of audio generative and editing tasks. We evaluate WavCraft on five basic tasks, involving adding, removal, replacement, super-resolution, and infilling. We also assess the advanced features of WavCraft on complex tasks, such as controllable editing and audio scriptwriting.

Adding. Given two audio clips A and B, the model is required to output a mixture M by combining A and B. Suppose C_A is the caption (i.e., text description) of A, an example of the instruction can be "Add C_A in the background of C_B".

Removal. Given a mixture M and one of its sound tracks A, the model is required to output a new audio clip B by removing A from M. Suppose C_A and C_M are the captions of A, respectively, an example of the instruction can be "Remove C_A from C_M".

Replacement. Given an audio clip A, a mixture M and one of its sound tracks B, the model is required to output a new audio clip C by replacing B with A in the same time slot. An example of the instruction can be "Replace C_B with C_A".

Super-resolution. Given a low-resolution audio clip A, the model is required to output a new audio clip A' with a higher sampling rate. An example of the instruction can be "Increase resolution of A".

Audio infilling. Given an audio clip A where some parts are masked, the model is required to complete the audio by filling the masked areas. An example of the instruction can be "Inpaint A".

In addition to the basic tasks, we also evaluate the advanced features of WavCraft through a case study.

[6]https://github.com/iver56/audiomentations

Table 3: Objective evaluation results on five different editing tasks.

Task	AUDIT (Wang et al.)				WavCraft			
	FAD ↓	IS ↑	KL ↓	LSD ↓	FAD ↓	IS ↑	KL ↓	LSD ↓
Add	9.27	3.87	3.00	1.95	0.63	6.05	1.45	1.59
Removal	17.57	3.27	4.40	3.46	3.48	6.38	1.72	2.07
Replacement	10.24	2.86	3.10	2.55	0.72	6.09	2.16	1.77
Infilling	12.61	3.88	2.86	3.40	3.31	6.37	1.00	2.10
Super-resolution	13.68	2.62	4.25	2.50	5.98	5.96	1.26	1.93

Table 4: Objective evaluation results on the AudioCaps evaluation set.

Model	FAD ↓	KL ↓	IS ↑
AudioLDM (Liu et al., 2023a)	4.65	1.89	7.91
WavJourney(Liu et al., 2023d)	3.38	**1.53**	7.94
WavCraft	**2.95**	1.68	**8.07**

5 Experiments

5.1 Experiments setup

To build up WavCraft, we used the GPT-4 model (OpenAI, 2023) as audio programming module and LTU (Gong et al., 2024) for audio analysis. We applied the publicly available models as audio expert models (shown in Table 2). We use 16 kHz sampling rate throughout the pipeline of audio editing and generation in line with the sampling rate of many integrated generative models (Kreuk et al., 2023; Copet et al., 2023). For the volume control of the generated audio content, we adopt the Loudness Unit Full Scale (LUFS) standard (International Telecommunication Union, 2020).

We evaluated WavCraft on audio editing and generation tasks separately. We compared WavCraft with AUDIT (Wang et al.), an state-of-the-art audio editing model, on diverse downstream tasks. For text-to-audio generation, we used AudioLDM (Liu et al., 2023a) and WavJourney (Liu et al., 2023d) for comparison. While WavJourney is also an LLM-based agent for audio content generation, it cannot take waveforms as inputs. We evaluate editing and text-to-audio generation task on AudioCaps (Kim et al., 2019) datasets. For editing tasks, we synthesis the database in according to 4 while directly using the evaluation split for the text-to-audio generation task. Please note that we will refer to the synthesised audio samples as the ground truth in the following sessions.

5.2 Evaluation metrics

For objective evaluation, we follows the evaluation protocols of existing audio generative models (Liu et al., 2023a;d; Wang et al.) to calculate several measurements: Frechet Audio Distance (FAD), Kullback-Leibler Divergence (KL), Inception Score (IS) and Log Spectral Distance (LSD) for evaluation. FAD measures the Frechet distance between reference and generated audio distributions of the embeddings extracted by a pre-trained VGGish model (Gemmeke et al., 2017). KL computes the similarity between logit distributions of two audio groups by using an audio tagging model, namely Patch-out Transformer (Koutini et al., 2022). IS reflects the variety and diversity of the generated audio group. Log Spectral Distance (LSD) calculates the distance between frequency spectrograms of output samples and target samples. A lower score of FAD, KL, or LSD indicates a better audio fidelity while a higher IS indicates a more diverse audio group (and thus more desirable for generated audio). Subjective evaluation were carried out by Amazon Mechanical Turk [7]. We offered raters with detailed instructions and illustrative examples to ensure a thorough evaluation process. Each audio sample was rated from one to five by a minimum of 15 different raters. We collected the feedback from rates and calculate mean opinion score (MOS) to reflect the overall quality of audio samples. Likewise, we assess the ability of audio scriptwriting, rated from one to fice, from five different aspects: audio-text relevance, audio coherence, naturalness, engagement, and

[7]https://requester.mturk.com

creativity. To extend AUDIT to the audio scriptwriting task, we use chatgpt to convert the instruction to several basic tasks and call AUDIT to execute tasks recursively. We refer to the extended AUDIT AUDIT+ in this paper.

5.3 Objective evaluation

Here we evaluate the performance of WavCraft on audio editing and text-to-audio generation tasks separately.

Evaluation on audio editing tasks. Table 3 shows the objective results of AUDIT and the proposed WavCraft. The WavCraft achieves a better performance than AUDIT in all objective evaluation across different tasks. Compared with AUDIT, WavCraft is 8.97, 14.09, 9.52 lower in FAD on the add, removal, and replacement tasks, respectively. On the audio infilling task, WavCraft achieves the FAD score of 3.31 while having the LSD score of 1.93.

Evaluation on text-to-audio generation. Table 4 shows the objective results of our WavCraft and the compared methods. WavCraft achieves the best FAD and IS score among the three evaluted models on the AudioCaps evaluation dataset. WavCraft also yields the KL score of 1.68, close to the performance of WavJourney (Liu et al., 2023d).

5.4 Subjective evaluation

Figure 2 shows the overall subjective evaluation results by comparing the performance of the ground truth, AUDIT, and our WavCraft. MOS value of WavCraft is very close to that of the ground truth while outperform the AUDIT's by a large margin.

Figure 3 compares the performance of the ground truth, AUDIT, and the proposed WavCraft in terms of frequency, time, and volume. WavCraft achieves the best results on frequency and time control compared to the AUDIT and even the ground truth. We hypothesise this is partly because the raw audio material used by the ground truth is less perceptually significant than the audio segments generated by WavCraft. In addition, WavCraft achieved a better MOS value compared to AUDIT in terms of volume.

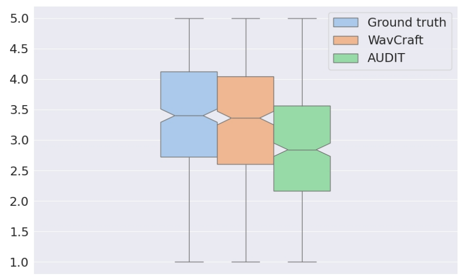


Figure 2: Overall subjective evaluation on audio editing quality by comparing the performance of the ground truth, AUDIT, and the proposed WavCraft.

Figure 4 shows the performance of WavCraft and AUDIT+ in terms of audio-text relevance, audio coherence, naturalness, engagement, and creativity. The WavCraft yielded a better scores than AUDIT+ from all aspects.

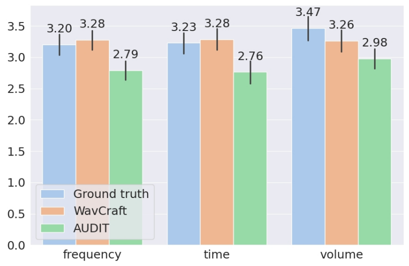


Figure 3: Subjective evaluation on the quality of edited audio obtained from the ground truth, AUDIT, and the proposed WavCraft in terms of frequency, temporal, and volume scale.

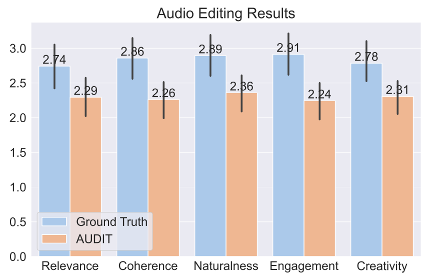


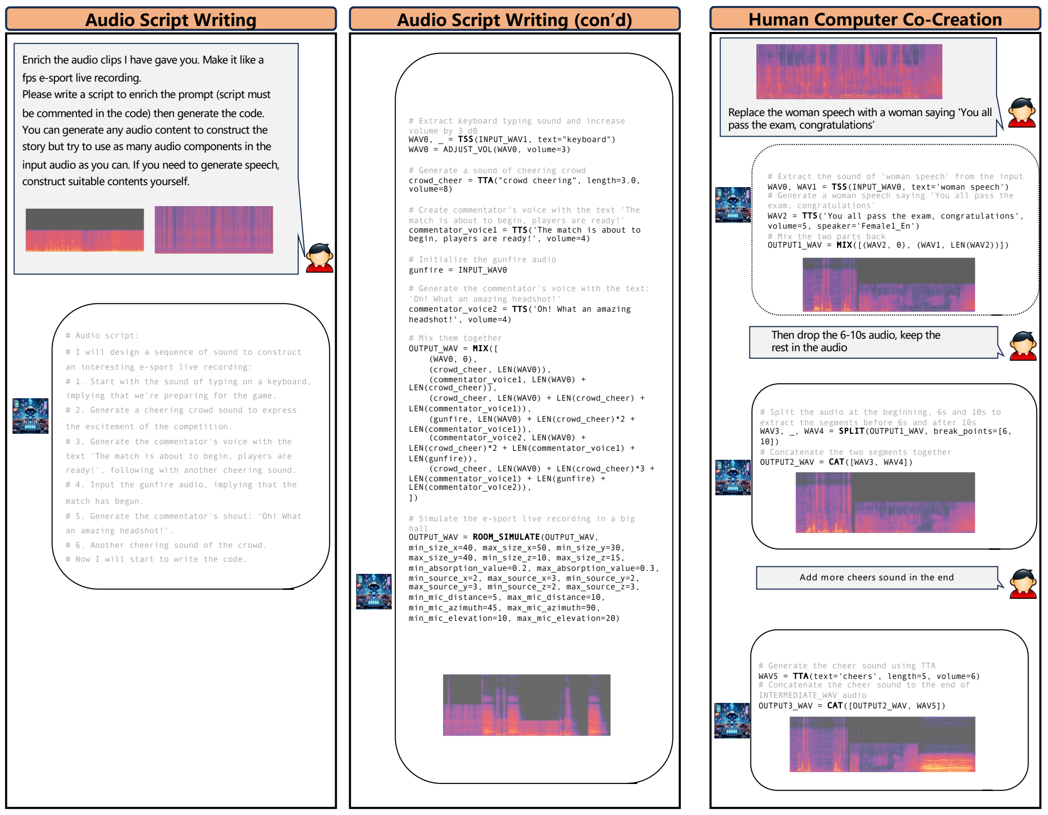
Figure 4: Comparing the ability of audio storytelling between AUDIT+ and the proposed WavCraft in terms of audio-text relevance, audio coherence, naturalness, engagement, and creativity.

5.5 CASE STUDY

Leveraging LLM's ability of natural language processing, WavCraft decomposes user's requests into basic applications and thus is capable of the tasks beyond the common tasks described in Sect. 5.3. As shown in Figure 5, we hereby discuss the advanced features of the proposed WavCraft by studying two cases:

Case study 1: Audio scriptwriting. WavCraft takes two raw audio materials (i.e., the beginning and end of an audio titled "a fan heading to a soccer match field") and a user instruction as inputs. It first analyses the content of input audio and write an audio script conditioned on both the user instruction and audio descriptions. The script is written in the format of python coding and applied to allocate diverse modules, such as target source separation, text-to-audio generation models, and the room simulator, for audio editing. The output of activated modules is mixed together in line with the generated audio script. To the best of our knowledge, WavCraft is the only audio agent capable of such complex editing task without an explicit user command.

Case study 2: Human-AI co-creation. We illustrate how WavCraft interacts with an user during the process of audio production. WavCraft starts with a basic replacement editing task: replacing the female speech in the audio with another female speaking. After the user went through the generated recording, namely `OUTPUT1_WAV`, the user further instructed the system to remove the audio content between 6-10s. WavCraft keeps track of the current conversation and generate `OUTPUT2_WAV` based on the new user instruction and the previous executation lines. Likewise, WavCraft continued to generate `OUTPUT3_WAV` in response to user's request about "Add more cheers sound in the end" while taking into consideration the previous instructions. Throughout the process of audio production, in addition to the generated audio, WavCraft also provide users with the executable code together with their comments. We hope this could improve the explainability of WavCraft operations.



Enrich the audio clips I have gave you. Make it like a fps e-sport live recording.
Please write a script to enrich the prompt (script must be commented in the code) then generate the code. You can generate any audio content to construct the story but try to use as many audio components in the input audio as you can. If you need to generate speech, construct suitable contents yourself.

```
# Audio script:
# I will design a sequence of sound to construct
an interesting e-sport live recording:
# 1. Start with the sound of typing on a keyboard,
implying that we're preparing for the game.
# 2. Generate a cheering crowd sound to express
the excitement of the competition.
# 3. Generate the commentator's voice with the
text 'The match is about to begin, players are
ready!', following with another cheering sound.
# 4. Input the gunfire audio, implying that the
match has begun.
# 5. Generate the commentator's shout: 'Oh! What
an amazing headshot!'.
# 6. Another cheering sound of the crowd.
# Now I will start to write the code.
```

```
# Extract keyboard typing sound and increase
volume by 3 dB
WAV0, _ = TSS(INPUT_WAV1, text="keyboard")
WAV0 = ADJUST_VOL(WAV0, volume=3)

# Generate a sound of cheering crowd
crowd_cheer = TTA("crowd cheering", length=3.0,
volume=8)

# Create commentator's voice with the text 'The
match is about to begin, players are ready!'
commentator_voice1 = TTS('The match is about to
begin, players are ready!', volume=4)

# Initialize the gunfire audio
gunfire = INPUT_WAV0

# Generate the commentator's voice with the text:
'Oh! What an amazing headshot!'
commentator_voice2 = TTS('Oh! What an amazing
headshot!', volume=4)

# Mix them together
OUTPUT_WAV = MIX([
    (WAV0, 0),
    (crowd_cheer, LEN(WAV0)),
    (commentator_voice1, LEN(WAV0) +
LEN(crowd_cheer)),
    (crowd_cheer, LEN(WAV0) + LEN(crowd_cheer) +
LEN(commentator_voice1)),
    (gunfire, LEN(WAV0) + LEN(crowd_cheer)*2 +
LEN(commentator_voice1)),
    (commentator_voice2, LEN(WAV0) +
LEN(crowd_cheer)*2 + LEN(commentator_voice1) +
LEN(gunfire)),
    (crowd_cheer, LEN(WAV0) + LEN(crowd_cheer)*3 +
LEN(commentator_voice1) + LEN(gunfire) +
LEN(commentator_voice2)),
])

# Simulate the e-sport live recording in a big
hall
OUTPUT_WAV = ROOM_SIMULATE(OUTPUT_WAV,
min_size_x=40, max_size_x=50, min_size_y=30,
max_size_y=40, min_size_z=10, max_size_z=15,
min_absorption_value=0.2, max_absorption_value=0.3,
min_source_x=2, max_source_x=3, min_source_y=2,
max_source_y=3, min_source_z=2, max_source_z=3,
min_mic_distance=5, max_mic_distance=10,
min_mic_azimuth=45, max_mic_azimuth=90,
min_mic_elevation=10, max_mic_elevation=20)
```

Replace the woman speech with a woman saying 'You all pass the exam, congratulations'

```
# Extract the sound of 'woman speech' from the input
WAV0, WAV1 = TSS(INPUT_WAV0, text='woman speech')
# Generate a woman speech saying 'You all pass the
exam, congratulations'
WAV2 = TTS('You all pass the exam, congratulations',
volume=5, speaker='Female1_En')
# Mix the two parts back
OUTPUT1_WAV = MIX([(WAV2, 0), (WAV1, LEN(WAV2))])
```

Then drop the 6-10s audio, keep the rest in the audio

```
# Split the audio at the beginning, 6s and 10s to
extract the segments before 6s and after 10s
WAV3, _, WAV4 = SPLIT(OUTPUT1_WAV, break_points=[6,
10])
# Concatenate the two segments together
OUTPUT2_WAV = CAT([WAV3, WAV4])
```

Add more cheers sound in the end

```
# Generate the cheer sound using TTA
WAV5 = TTA(text='cheers', length=5, volume=6)
# Concatenate the cheer sound to the end of
INTERMEDIATE_WAV audio
OUTPUT3_WAV = CAT([OUTPUT2_WAV, WAV5])
```

Figure 5: Case studies on audio scriptwriting and human-AI co-creation.

6 LIMITATIONS

Despite WavCraft demonstrating a desirable ability of audio editing and generation, there still exists some limitations: 1) *Audio analysis*: While WavCraft involves an audio analysis module to describe the raw materials, the performance of existing audio analysis models is limited, hindering WavCraft from precisely describing input audio with temporal relationship. 2) *Inference cost*: WavCraft needs to call diverse APIs to solve a complex task, which introduces time costs during inference. Reducing the inference time will facilitate a more seamless human-AI co-creation, meeting the requirement of more practical applications.

7 CONCLUSION

This work presented WavCraft, an agent system that integrates the LLM with diverse audio expert models, to create audio content conditioned on input audio clips and user queries. WavCraft decomposes an intricate editing work into individual basic audio tasks, after comprehending users' queries and the content of given recordings. The output of basic tasks is then assembled under the instructions formulated by audio programming module, contributing to the final output. Case studies conducted on several real-world scenarios have demonstrated the potential of WavCraft in audio production applications. WavCraft shows the feasibility of AIGC in the audio domain in a transparent, interpretable, and interactive manner.

ACKNOWLEDGEMENTS

This work is supported by the Engineering and Physical Sciences Research Council [grant number EP/T518086/1]. E. Benetos is supported by a RAEng/Leverhulme Trust Research Fellowship [grant number LTRF2223-19-106]. H. Zhang is supported by the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, funded by UK Research and Innovation [grant number EP/S022694/1]. This work is also partly supported by Engineering and Physical Sciences Research Council (EPSRC) Grant EP/T019751/1 "AI for Sound (AI4S)", and a PhD scholarship from the Centre for Vision, Speech and Signal Processing (CVSSP) at the University of Surrey and BBC R&D. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

REFERENCES

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. MusicLM: Generating music from text, January 2023. URL `http://arxiv.org/abs/2301.11325`. arXiv:2301.11325 [cs, eess].

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: A visual language model for few-shot learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2523–2533, 2023. doi: 10.1109/TASLP.2023.3288409.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=jtiQ26sCJi`.

Soham Deshmukh, Benjamin Elizalde, Rita Singh, and Huaming Wang. Pengi: An audio language model for audio tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=gJLAfO4KUq`.

Jiaxin Ge, Sanjay Subramanian, Baifeng Shi, Roei Herzig, and Trevor Darrell. Recursive visual programming, December 2023. URL `http://arxiv.org/abs/2312.02249`. arXiv:2312.02249 [cs].

Jort F. Gemmeke, Daniel P. W. 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)*, pp. 776–780, New Orleans, LA, March 2017. IEEE. ISBN 978-1-5090-4117-6. doi: 10.1109/ICASSP.2017.7952261.

Yuan Gong, Hongyin Luo, Alexander H. Liu, Leonid Karlinsky, and James R. Glass. Listen, think, and understand. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=nBZBPXdJlC`.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, June 2023.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, Yi Ren, Yuexian Zou, Zhou Zhao, and Shinji Watanabe. Audiogpt: Understanding and generating speech, music, sound, and talking head. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21):23802–23804, Mar. 2024. doi: 10.1609/aaai.v38i21.30570. URL `https://ojs.aaai.org/index.php/AAAI/article/view/30570`.

International Telecommunication Union. ITU-R BS.1770-4: Algorithms to measure audio programme loudness and true-peak audio level, 2020. URL `https://www.itu.int/rec/R-REC-BS.1770`.

Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. AudioCaps: Generating captions for audios in the wild. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 119–132, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1011. URL `https://aclanthology.org/N19-1011`.

Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. In *Proc. Interspeech 2022*, pp. 2753–2757, 2022. doi: 10.21437/Interspeech.2022-227.

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. AudioGen: Textually guided audio generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=CYK7RfcOzQ4`.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Jinhua Liang, Xubo Liu, Wenwu Wang, Mark D. Plumbley, Huy Phan, and Emmanouil Benetos. Acoustic prompt tuning: Empowering large language models with audition capabilities, November 2023. URL `http://arxiv.org/abs/2312.00249`. arXiv:2312.00249 [eess].

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. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21450–21474. PMLR, July 2023a. URL `https://proceedings.mlr.press/v202/liu23f.html`.

Haohe Liu, Ke Chen, Qiao Tian, Wenwu Wang, and Mark D. Plumbley. Audiosr: Versatile audio super-resolution at scale. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1076–1080, 2024. doi: 10.1109/ICASSP48485.2024.10447246.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang, Jianfeng Gao, and Chunyuan Li. LLaVA-Plus: Learning to use tools for creating multimodal agents, November 2023b. URL `http://arxiv.org/abs/2311.05437`. arXiv:2311.05437 [cs].

Xubo Liu, Qiuqiang Kong, Yan Zhao, Haohe Liu, Yi Yuan, Yuzhuo Liu, Rui Xia, Yuxuan Wang, Mark D. Plumbley, and Wenwu Wang. Separate anything you describe, August 2023c. URL `http://arxiv.org/abs/2308.05037`. arXiv:2308.05037 [cs, eess].

Xubo Liu, Zhongkai Zhu, Haohe Liu, Yi Yuan, Meng Cui, Qiushi Huang, Jinhua Liang, Yin Cao, Qiuqiang Kong, Mark D. Plumbley, and Wenwu Wang. WavJourney: Compositional audio creation with large language models, July 2023d. URL `http://arxiv.org/abs/2307.14335`. arXiv:2307.14335 [cs, eess].

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, March 2023. arXiv:2303.17395.

OpenAI. GPT-4 technical report, March 2023.

Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Dang, Jiahao Li, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development, December 2023. URL `http://arxiv.org/abs/2307.07924`. arXiv:2307.07924 [cs].

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=Yacmpz84TH`.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with chatGPT and its friends in hugging face. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=yHdTscY6Ci`.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, October 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, July 2023. arXiv:2307.09288.

Apoorv Vyas, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins, William Ngan, Jeff Wang, Ivan Cruz, Bapi Akula, Akinniyi Akinyemi, Brian Ellis, Rashel Moritz, Yael Yungster, Alice Rakotoarison, Liang Tan, Chris Summers, Carleigh Wood, Joshua Lane, Mary Williamson, and Wei-Ning Hsu. Audiobox: Unified audio generation with natural language prompts, 2023.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers, January 2023. URL `http://arxiv.org/abs/2301.02111`. arXiv:2301.02111 [cs, eess].

Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, and sheng zhao. AUDIT: Audio editing by following instructions with latent diffusion models. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 71340–71357. Curran Associates, Inc. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/e1b619a9e241606a23eb21767f16cf81-Paper-Conference.pdf`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently, March 2023. URL `http://arxiv.org/abs/2303.03846`. arXiv:2303.03846 [cs].

Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Artyom Astafurov, Caroline Chen, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jeff Hwang, Ji Chen, Peter Goldsborough, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, and Vincent Quenneville-Bélair. Torchaudio: Building blocks for audio and speech processing. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6982–6986, May 2022. doi: 10.1109/ICASSP43922.2022.9747236. URL `https://ieeexplore.ieee.org/document/9747236?denied=`. ISSN: 2379-190X.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, January 2024. ISSN 2157-6904. doi: 10.1145/3639372. URL `https://dl.acm.org/doi/10.1145/3639372`. Just Accepted.

A Prompt template.

Table 5 and Table 6 list the instruction prepended to the user query for the first and following round dialogs, respectively. We exploits the LLM's ability of in-context learning to allocate diverse audio modules by mimicing previous examples. In the follow-up instruction, we guide LLM to pay attention to the current instruction together with the those of previous dialogs. Therefore, WavCraft can perform audio manipulation by following current request while keeping track of the previous instructions.

Table 5: WavCraft's predifined instruction prepended to the user query.

INPUT INSTRUCTION
You are an professional audio editor. Try to follow the instruction I give using several predefined tools:
LEN(wav) # returns the duration of ‘wav‘ in seconds
MIX(wavs: list[tuple]) # returns the mixture of the input ‘wavs‘
CAT(wavs: list) # returns the concatenated wav using input ‘wavs‘
SPLIT(wav, break_points=list[float]) # returns the split wavs using ‘break_points‘
ADJUST_VOL(wav, volume: int) # returns the adjusted wav by ‘volume‘
TTA(text: str, length: float, volume: int) # returns a generated audio conditioned on ‘text‘
TTM(text: str, melody, length: float, volume: int) # returns a generated music conditioned on ‘text‘ and (optional) ‘melody‘
TTS(text: str, volume: int) # returns a generated speech conditioned on ‘text‘ and ‘speaker‘. ‘speaker‘ should be in ['Male1_En', 'Male2_En', 'Female1_En', 'Female2_En']
SR(wav, ddim_steps: int, guidance_scale: float, seed: int) # Returns a wav upsampled to 48kHz
TSS(wav, text: str) # returns foreground and background wav conditioned on ‘text‘
ADD_NOISE(wav, min_snr_db: float, max_snr_db: float)
returns a generated audio mixed with gaussian noise
LOW_PASS(wav, min_cutoff_freq: float, max_cutoff_freq: float, min_rolloff: int, max_rolloff: int)
returns a generated audio processed by low pass filter
HIGH_PASS(wav, min_cutoff_freq: float, max_cutoff_freq: float, min_rolloff: int, max_rolloff: int)
returns a generated audio processed by high pass filter
ADD_RIR(wav, ir) # returns a generated audio mixed with a given room impulse response
ROOM_SIMULATE(wav, min_size_x: float, max_size_x: float, min_size_y: float, max_size_y: float, min_size_z: float, max_size_z: float, min_absorption_value: float, max_absorption_value: float, min_source_x: float, max_source_x: float, min_source_y: float, max_source_y: float, min_source_z: float, max_source_z: float, min_mic_distance: float, max_mic_distance: float, min_mic_azimuth: float, max_mic_azimuth: float, min_mic_elevation: float, max_mic_elevation: float) # returns a synthesized audio by mixing the input ‘wav‘ with a room-specific synthesized impulse response
INPAINT(wav, text: str, onset: float, offset: float, duration: float)
returns a fixed audio where the part between ‘onset‘ and ‘offset‘ has been inpainted
I will give you several examples:
Instruction:
Increase the volume of child speech by 5 dB, decrease the volume of drum by 3 dB, drop the sound of machine sound.
Code:
Separate the sound of 'child speech' from the mixture and return both 'child speech' and the background sounds
WAV0, WAV1 = TSS(INPUT_WAV0, text="child speech")
Separate the sound of 'drum' from the mixture and return both 'drum' and the background sounds
WAV2, WAV3 = TSS(WAV1, text="drum")
Drop the sound of 'machine sound' from the mixture
_, WAV3 = TSS(WAV3, text="machine sound")
Increace the volume of "child speech" by 5dB
WAV0 = ADJUST_VOL(WAV0, volume=5)
Decrease the volume of 'drum' by 5dB
WAV2 = ADJUST_VOL(WAV2, volume=-3)
Mix the resulted sounds together
OUTPUT_WAV = MIX([(WAV0, 0), (WAV2, 0), (WAV3, 0)])

INPUT INSTRUCTION (con'd)

Instruction:
Extract 1-5s of the first audio with a low-pass filter to simulate the sound coming from inside a building. Replace male speech with dog barking in the second audio. Upsample the mix.
Code:

```
# Truncate the sound between 1s and 5 s
_, WAV0, _ = SPLIT(INPUT_WAV0, break_points=[1, 5])
# Add a low-pass filter
WAV0 = LOW_PASS(WAV0, min_cutoff_freq=300.0, max_cutoff_freq=800.0,
min_rolloff=6, max_rolloff=12)
# Extract the sound of 'male speech' from the truncated sound
WAV1, WAV2 = TSS(INPUT_WAV1, text="male speech")
# Generate the sound of 'dog barking' with the same length with the sound of 'male speech'
WAV3 = TTA(text="dog barking", length=LEN(WAV1), volume=4)
# Combine the sounds by mixing them together
MIXTURE_WAV = MIX([(WAV3, 0), (WAV2, 0), (WAV0, 0)])
# Perform super-resolution on the mixture of sounds
OUTPUT_WAV = SR(MIXTURE_WAV)
```

Instruction:
Isolate train sound in the input audio, apply a high-pass filter and increase the volume by 3 dB. Repeat it five times to simulate a longer train passing.
Code:

```
# Extract the sound of a train from the audio
WAV0, _ = TSS(INPUT_WAV0, text="train")
# Apply a high-pass filter to reduce low-frequency noise
FILTERED_WAV0 = HIGH_PASS(WAV0, min_cutoff_freq=500.0, max_cutoff_freq=1000.0,
min_rolloff=6, max_rolloff=12)
# Increase the volume by 3 dB
FILTERED_WAV0 = ADJUST_VOL(FILTERED_WAV0, volume=3)
# Concatenate the filtered train sound three times
OUTPUT_WAV = CAT([FILTERED_WAV0] * 5)
```

Instruction:
Extract the hammer sound from the first audio, and truncate it from the start towards 2 second. Remove the sound of baby crying in the second audio, and then decrease the volume by 1 dB. Mix two audio together, and the second sound should begin from 1 second. Add a reverb effect to the mixture sound using the third audio.
Code:

```
# Extract the hammer sound from the first audio
WAV0, _ = TSS(INPUT_WAV0, text="hammer")
# Truncate from the start towards 2 second
WAV0, _ = SPLIT(WAV0, break_points=[2])
# Drop the sound of baby crying in the second audio
_, WAV1 = TSS(INPUT_WAV1, text="baby crying")
# Decrease the volume by 1 dB
WAV1 = ADJUST_VOL(WAV1, volume=-1)
# Mix the ouput sounds together
MIXED_WAV = MIX([(WAV0, 0), (WAV1, 1)])
# Add a reverb effect using room impulse response
OUTPUT_WAV = ADD_RIR(MIXED_WAV, ir=INPUT_WAV2)
```

INPUT INSTRUCTION (con'd)

Instruction:
Inpaint the first audio between 2s and 5s with the text "a car passing by with rain falling". Generate a 10s long jazz music piece with the second audio as melody, then mix it with the sound of rain from the first, starting at 3s into the jazz music.
Code:

```
# Inpaint the first audio between 2s and 5s with the text "a car passing by with rain falling"
WAV0 = INPAINT(INPUT_WAV0, text="a car passing by with rain falling", onset=2,
offset=5, duration=LEN(INPUT_WAV0))
# Generate a 10-second jazz music piece
WAV1 = TTM(text="jazz", melody=INPUT_WAV1, length=10.0, volume=5)
# Extract the sound of rain from the audio file
WAV0, _ = TSS(WAV0, text="rain")
# Mix the jazz music with the rain sound, starting the rain at 3 seconds
OUTPUT_WAV = MIX([(WAV0, 0), (WAV1, 3)])
```

Instruction:
Remove wind sound from an outdoor recording. Generate a 5-second saxophone music with happy mood followed by "Bravo". Mix the generated sound with the outdoor recording and simulate the mixture in a small room with high absorption.
Code:

```
# Drop the sound of wind from the original recording
_, WAV0 = TSS(INPUT_WAV0, text="wind")
# Generate a 5-second saxophone music with happy mood followed by a male speech
"Bravo".
WAV1 = TTM(text="happy saxophone", length=5.0, volume=4)
# Generate a speech "Bravo"
WAV2 = TTS("Bravo", volume=5)
# Concatenate the generated sound together
CONCAT_WAV = CAT([WAV1, WAV2])
# Mix the generated sound with the background sound
MIXED_WAV = MIX((WAV0, 0), (CONCAT_WAV, 0))
# Simulate the recording in a small room with high absorption
OUTPUT_WAV = ROOM_SIMULATE(MIXED_WAV, min_size_x=3, max_size_x=4,
min_size_y=3, max_size_y=4, min_size_z=2.5, max_size_z=3, min_absorption_value=0.7,
max_absorption_value=0.9, min_source_x=1, max_source_x=1.5, min_source_y=1,
max_source_y=1.5, min_source_z=1, max_source_z=1.5, min_mic_distance=1,
max_mic_distance=1.5, min_mic_azimuth=45, max_mic_azimuth=90,
min_mic_elevation=20, max_mic_elevation=30)
```

Table 6: WavCraft's follow-up instruction to make sure the consistency of generated audio within the multi-round dialog between human and AI.

FOLLOW-UP INSTRUCTION
Regenerate the code by appending the new instruction to the previous instructions. The code must start with the provided audio (e.g., INPUT_WAV0) and cannot take the output from previous phase (i.e., 'OUTPUT_WAV') as a known input. The new instruction is: