Listen, Chat, and Edit on Edge: Text-Guided Soundscape Modification for Real-Time Auditory Experience

Siavash Shams Pranav Kumar Kota

Abstract

Often time we are in a situation where we would like to listen to only specific sounds and eliminate noise in our environment. To solve this issue, we employ the Listen, Chat, Edit methods on the edge to edit sound in our environment. The work uses a multimodal sound mixture editor that lets users edit a sound sample with user-provided text instructions. The user is allowed to provide instruction from an open vocabulary which are interpreted by large language model to create a semantic filter for editing the sound mixture. The system generates a new edited audio by applying this semantic filter and attenuating/enhancing the desired components. The focus of of work is implementing the system on the edge for a variety of applications. An audio demo of the original work is available at: https://listenchatedit.github.io/demo.

1. Introduction

Imagine the you are in a cocktail party. While you are engaged in a conversation with, say, two people, there are many sounds surrounding you. From a soft melody played by a guitar, to cars passing by, to a person talking loudly, many sounds dull the conversation. Now, imagine if you able to use a smart sound editor and say ”Can you remove the noise of the cars passing by and reduce the sound of the excited male speaker?”. You can now easily follow the conversations of those you are talking with.

This problem is the famous cocktail party problem (Haykin & Chen, 2005). Prior studies have been conducted to propose solutions. It was observed that tradition hearind kits could enhance hearing experience in a noisy environment (Kates, 2008), but fail to target specific sound sources. More studies targeted sound sources based on a neural auditory attention signal (Van Eyndhoven et al., 2016), sound word labels (Kilgour et al., 2022). The primary issue with these is that they focus on isolating single sources and fail to effectively handle multiple sources. They also do not provide a user-friendly experience that lets the models be used in an intuitive way. The work ”Listen, Chat, and Edit” (LCE) (Jiang et al., 2024), build on these issues to provide a solution that provides a versatile method to modify multiple sources in a sound mixture, aided by a user-friendly experience guided by natural language. LCE allows a user to chat with the system in natural language and mention the modification desired for each sound source. This is decoded into an embedding understood by a sound mixture model, by an LLM, which is then used to generate a new audio mixture free of the undesired sounds.

Why are we interested in deploying this system on the Jetson Nano? As seen in the earlier discussion, situations that motivate such a system occur as part of our daily life. To be able to provide a solution that is accessible to users in a convenient way, it is logical that the system is deployed on mobile edge devices. This comes with a significant issue of handling large model sizes. Due to limited storage space in edge devices and compute limitations too, it becomes close to impossible to be able to deploy large systems such as an LLM, a component of LCE, onto these devices. Our implementation suggests an integrated architectures that leverages the cloud to perform the decoding of text prompts with the LLM used, and then relay information expected by the sound mixture model to edit the audio sample.

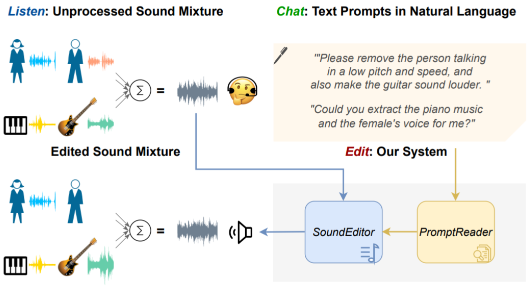
Listen: Unprocessed Sound Mixture
Chat: Text Prompts in Natural Language
""Please remove the person talking in a low pitch and speed, and also make the guitar sound louder. "
"Could you extract the piano music and the female's voice for me?"
Edited Sound Mixture
Edit: Our System
SoundEditor
PromptReader

Figure 1. LCE System

2. Listen, Chat, and Edit

In this section we introduce the sub-modules of Listen, Chat, and Edit (LCE). LCE includes two models: a *PromptReader* to read text prompts and a *SoundEditor* to edit sound mixtures. A graphical illustration of it in details is depicted in

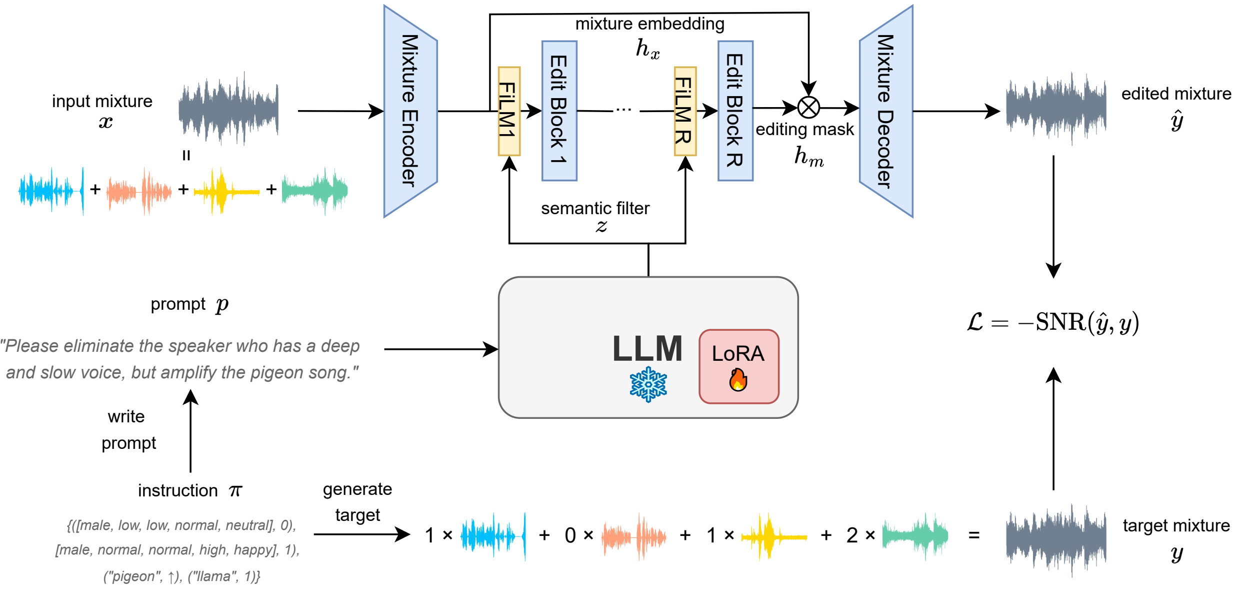

Figure 2. The components and the training paradigm of LCE.

Figure 2.

2.1. Prompt Reader

The *PromptReader*, a language model, is employed to function as the inverse of the text prompt encoding. The generated output is a D-dimensional embedding $z \in \mathbb{R}^D$ from a text prompt p. p encodes information about the source and actions in pi, and $p = \mathcal{H}(\pi)$, where $\mathcal{H}$ is a natural language encoding of the prompt information in π.

$$z = PromptReader(p) \tag{1}$$

The LLM used to decode the prompt into an embedding was LLaMA2. (Touvron et al., 2023)

2.2. Sound Editor

The *SoundEditor*, an acoustic model, edits the sound mixture x based on the semantic filter z as depicted:

$$\hat{y} = SoundEditor(x, z) \tag{2}$$

The architecture of source separation models was selected for the *SoundEditor* due to their capability to isolate individual sources within a sound mixture. It is critical to note that the *SoundEditor* does not perform explicit separation of all sources for individual modification and subsequent aggregation. Rather, the model operates analogously to a sound extraction model, directly producing a single sound that combines all edited sources.

The design principles of the time-domain source separation network Conv-TasNet (Luo & Mesgarani, 2019) was adopted. This models integrates a learnable linear encoder $\mathcal{E} : \mathbb{R}^{1\times T} \to \mathbb{R}^{C\times L}$ that maps the input waveform x with T samples into a C-dimensional latent representation h_x, and a learnable linear decoder $\mathcal{D} : \mathbb{R}^{C\times L} \to \mathbb{R}^{1\times T}$ that converts the output sound from latent space $h_{\hat{y}}$ back to the waveform $\hat{y}$.

$$h_x = \mathcal{E}(x), \quad \hat{y} = \mathcal{D}(h_{\hat{y}}) \tag{3}$$

A mask network denoted as $\mathcal{M}$ is situated between the encoder and decoder, estimating a single editing mask $h_m \in \mathbb{R}^{C\times L}$ that modulates samples in h_x according to z through element-wise multiplication:

$$h_m = \mathcal{M}(x, z), \quad h_{\hat{y}} = h_m \otimes h_x \tag{4}$$

Each of the R editing blocks within $\mathcal{M}$, consistent in structure, utilizes either a temporal convolutional network for Conv-TasNet.

The influence of the semantic filter z on each editing block is mediated through Feature-wise Linear Modulation (FiLM) (Perez et al., 2018). The computation of the text-guided feature $\tilde{h}_i$ before entering the ith block is given by:

$$\tilde{h}_i = \gamma_i h_i + \beta_i, \quad 0 \leq i \leq R - 1 \tag{5}$$

$$\gamma_i = \mathcal{F}_i(z), \quad \beta_i = \mathcal{G}_i(z) \tag{6}$$

In this setup, $\mathcal{F}_i : \mathbb{R}^D \to \mathbb{R}^C$ and $\mathcal{G}_i : \mathbb{R}^D \to \mathbb{R}^C$ function as two-layer perceptrons that project the semantic filter z into the dimensionality of h_i. Following this, γ_i and β_i are elongated along the temporal dimension to adjust h_i, thereby guaranteeing that the influence of the text prompt remains uniform over the duration of the audio signal. The architecture of Conv-Tasnet is shown in 3

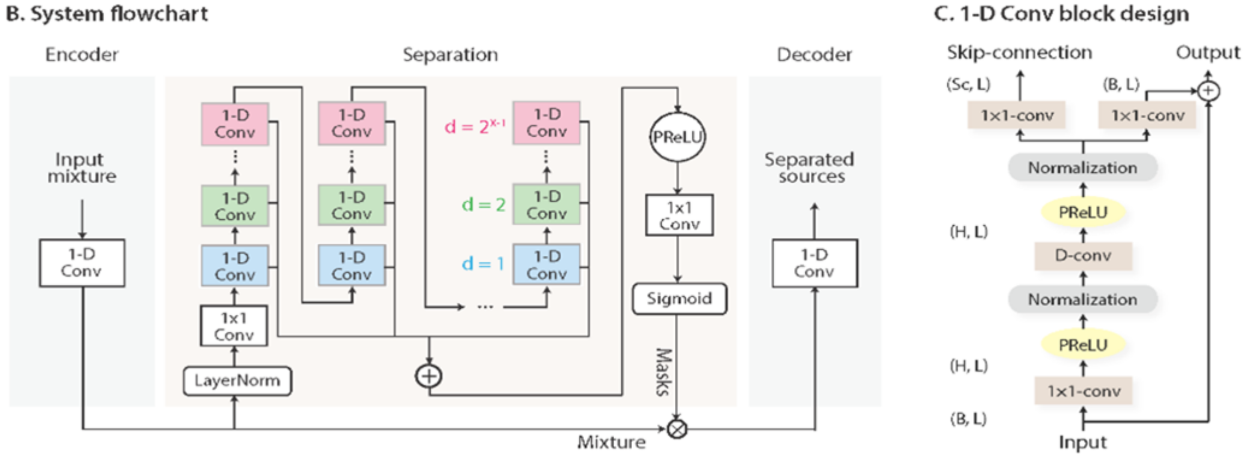

Figure 3. Sound editing module architecture

2.3. Training Objective

Every training sample consists of a triplet (x, y, p), which encompasses an input mixture x, a target mixture y, and a text prompt p. The components of the mixture $\{s_1, s_2, ..., s_N\}$ and the symbolic instruction π that guide the formation of y and p remain concealed from the system. Given x and p, LCE computes an estimated mixture $\hat{y}$, in accordance with Equations 1 and 2. The entire model is optimized concurrently by enhancing the signal-to-noise ratio (SNR) between the target and the predicted mixture:

$$\mathrm{SNR}(\hat{y}, y) = 10 \log_{10}\left(\frac{\|y\|^2}{\|y - \hat{y}\|^2}\right) \tag{7}$$

3. Dataset

The authors of the work generated their own dataset of sound-mixtures and text prompts due to the absence of a natural language instruction-prompted sound editing datasets. Some key features are noted below:

Sound Sources

Audio sources were taken from VGGSound (Chen et al., 2020) (training/in-domain evaluation) and FSD50k (Fonseca et al., 2021) (for zero-shot evaluation). Speech sources were taken from TextrolSpeech (Ji et al., 2023). All sources were sampled to 16kHz, cropped, padded to 5 seconds and split into a train, val, test set.

Sound Mixtures

Two speech and two audio sources of 100k, 10k, 5k mixtures from TextrolSpeech and VGGSound with were generated for training, validation and testing. These four-source mixtures contained a combination of categories that simulated acoustic scenes commonly encounter in daily life. From FSD50k, 5000 mixtures of 2 speech, 2 audio, 2 speech + 1 audio, 1 speech + 2 audio mixtures were generated for zero-shot evaluation.

Editing Tasks For every mixture, we first evenly assigned a Tasks were evenyl assigned for every mixture and then a particular instruction π for a task t. Performance of each task was computed by averaging the editing performance of all mixtures related to that task.

GPT-3.5 Turbo was used to write a text prompt five times from the assigned instruction π for each mixture. To ensure quality and diversity of text prompts, GPT was asked to try imperative or interrogative sentences and substitute synonyms for class labels and style keywords (*low*, *high*, etc.). To ensure that the prompts were natural, it was asked to mention a source only if specifically instructed to if the action included *keeping*. Speaker was described by a random subset of attributes and not all of them.

4. Experiments

4.1. Setup and Implementation

An investigation was carried out to assess the capabilities of a hybrid acoustic processing system that integrates a separator module deployed on a Jetson Nano with a large language model (LLM) hosted on the cloud. A robust connection was established between these hardware components to facilitate seamless interaction and data exchange.

4.2. Test Scenarios and Input Data

The system was evaluated using a variety of sound mixtures comprising human speech overlaid with diverse background noises such as helicopter sounds, dog barks, bell tones, and music. These scenarios are designed to mimic complex auditory environments where selective sound editing is crucial.

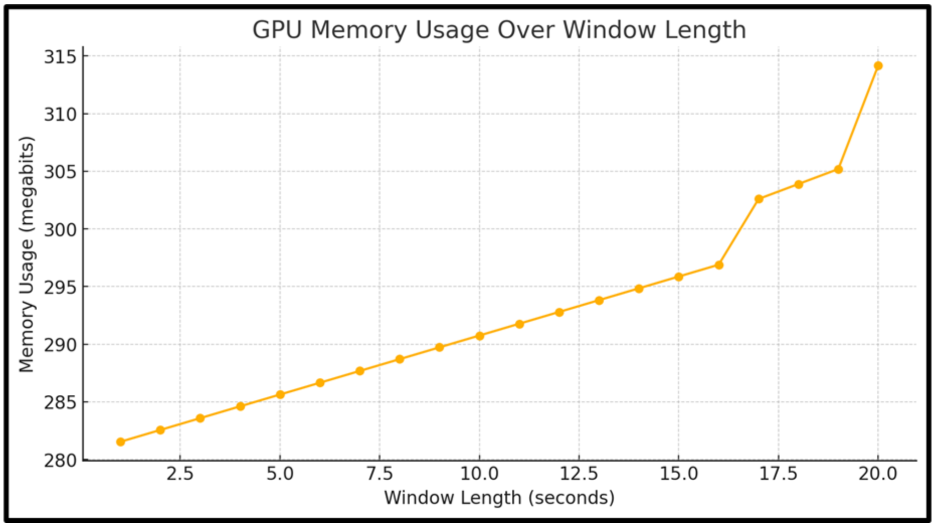

Figure 4. GPU memory usage during inference against input window length

4.3. Model Operations and Results

The model was tasked with executing several sound editing operations including the removal of all talkers to isolate non-speech sounds, amplification of specific talkers to enhance speech clarity in noisy environments, and reduction of various background noises. The overall performance of the system was satisfactory, aligning well with the quality demonstrated in the main paper's examples. However, challenges were observed, such as the system's tendency to mistakenly remove human talkers when prompted to eliminate all non-human sounds, a discrepancy from the expected outcomes demonstrated.

4.4. Demonstrations and Observations

These results were presented in a class demonstration, showcasing the system's real-time processing capabilities. For further analysis and review, both the original and edited audio mixtures were stored on a drive.

4.5. Performance Metrics

Detailed profiling was conducted to evaluate the computational efficiency of the system. This included measuring the time taken by each module within the separator and the LLM for completing inference processes, and monitoring GPU memory usage as it correlates with different audio window lengths. These metrics are essential for understanding and optimizing the system's performance under various operational contexts. Table 1 shows the processing time analysis of system components. Figure 4 shows the GPU memory consumption on Jetson Nano per window length of the input mixture in second.

Table 1. Processing Time Analysis of System Components

Module	Submodules	Processing Time (s)
Sound Editing Module	Encoder	0.042 ± 0.024
	MaskNet	4.520 ± 0.551
	Decoder	0.011 ± 0.016
LLM	LLaMA2	3.260 ± 0.149

5. Conclusion

This study has successfully demonstrated the deployment of *Listen, Chat, and Edit* (LCE), a text-guided sound mixture-to-mixture editing framework, on the Jetson Nano. This achievement addresses the cocktail party problem through a novel approach, allowing users to specify edits through natural language instructions in a real-world, resource-constrained environment.

Deploying LCE on the Jetson Nano has showcased the feasibility of running advanced sound editing models on edge devices, which traditionally lacked the capacity to handle complex deep learning tasks due to hardware limitations. The implementation on the Jetson Nano not only brings the power of sophisticated sound editing closer to everyday users but also significantly reduces latency and dependency on cloud services, enhancing privacy and accessibility.

The unique architecture of LCE, which integrates large language models for interpreting user prompts and editing sounds based on semantic descriptions, has been adapted to operate efficiently within the computational constraints of the Jetson Nano. This adaptation involved optimizing both the model's performance and resource usage, maintaining high-quality sound editing capabilities while managing the device's memory and processing power effectively.

Our experimental evaluations have confirmed that LCE maintains a high level of performance on the Jetson Nano, with satisfactory signal-to-noise improvements and robust handling of diverse sound editing tasks. Challenges such as occasional misidentification of sound sources were observed and are highlighted as areas for further improvement. These insights are crucial for refining the model's accuracy and expanding its utility in more dynamic environments.

References

Chen, H., Xie, W., Vedaldi, A., and Zisserman, A. VGGSound: A large-scale audio-visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 721–725. IEEE, 2020.

Fonseca, E., Favory, X., Pons, J., Font, F., and Serra, X. FSD50K: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021.

Haykin, S. and Chen, Z. The cocktail party problem. *Neural computation*, 17(9):1875–1902, 2005.

Ji, S., Zuo, J., Fang, M., Jiang, Z., Chen, F., Duan, X., Huai, B., and Zhao, Z. TextrolSpeech: A text style control speech corpus with codec language text-to-speech models. *arXiv preprint arXiv:2308.14430*, 2023.

Jiang, X., Han, C., Li, Y. A., and Mesgarani, N. Listen, chat, and edit: Text-guided soundscape modification for enhanced auditory experience, 2024.

Kates, J. M. *Digital hearing aids*. Plural publishing, 2008.

Kilgour, K., Gfeller, B., Huang, Q., Jansen, A., Wisdom, S., and Tagliasacchi, M. Text-driven separation of arbitrary sounds. *arXiv preprint arXiv:2204.05738*, 2022.

Luo, Y. and Mesgarani, N. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, August 2019. ISSN 2329-9304. doi: 10.1109/taslp.2019.2915167. URL http://dx.doi.org/10.1109/TASLP.2019.2915167.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Van Eyndhoven, S., Francart, T., and Bertrand, A. Eeg-informed attended speaker extraction from recorded speech mixtures with application in neuro-steered hearing prostheses. *IEEE Transactions on Biomedical Engineering*, 64(5):1045–1056, 2016.

A. Pub/Sub

A system developed to facilitate communication between the LLM and the cloud was the Pub/Sub service provided by Google Cloud Platform.

A.1. About

Pub-Sub is a message delivery service enabled by Google Cloud Platform. Topics contain information relevant to a specific information; publishers publish to topics. Subscribers load data from subscriptions linked to a specific topic. In our system for data transfer, a topic is setup for sending prompt from local machine to the cloud. A subscriber reads incoming data, detects change, and computes an embedding of the new prompt. This embedding is then published to another topic from which the local machine subscribes to get the required embedding.

A.2. Challenges

This method was not as straightforward as expected. Due to the streaming nature of data, there was evident asynchronicity in how the prompts were received and sent. Sometimes certain prompts wouldn't reflect immediately in the system and would take a while to reach the end of the prompt to embedding pipeline. This is not desirable as it interferes with user experience. To alleviate this issue, further system design choices must be made that will help in setting up fault-tolerance mechanisms.

A.3. System design and operation of the service

The source code for this section is available in the GitHub repository, in a folder called ”pubsub”. The user runs the ‘process_prompt.py‘ file in a remote cloud instance where the LLM is loaded. This script monitors the ”Prompt Input” topic and when it receives a new prompt, it saves it inside a local file and processes it into an embedding. The generated embedding is appropriately encoded and sent to the ”Embedding Output” topic. To send prompts, the user runs the ‘publish_prompt.py‘ file, and to receive the generated embedding, the ‘subscribe_emebdding.py‘ file is run.

The experiments with the system provided insights into the difficulty involved in transmission of data between models deployed on different compute nodes. This motivates studies into efficient edge AI and fault-tolerant data streaming for AI applications.

An illustrative figure showing what's happening to the spectrogram of the input mixture is shown in 5

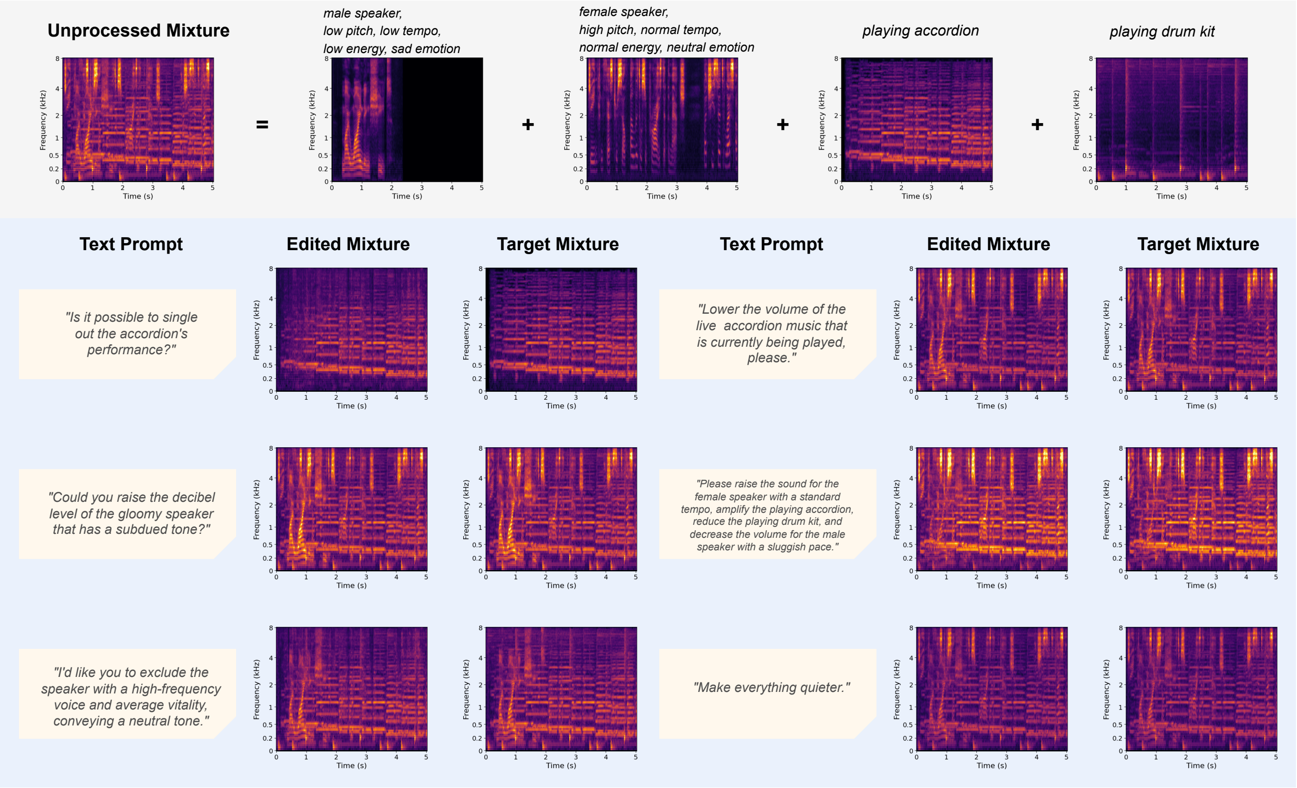

Figure 5. Example A: sound mixture editing on 2 Speech + 2 Audio (VGGSound) mixture.

Table 2. Contributions by Students

Student Name	Contributions
SS	Setting up sound editing module on Jetson Nano: dealt with compatibility issues of packages and refactored the code for Conv-Tasnet in PyTorch to make it compatible (100%), implementation and execution profiling and inference timing measurements (100%), setting up LLM-cloud and Jetson Nano interaction system (100%), writing the report (50%).
PKK	Setting up LLM on GCP VM (100%).Setting up communication between the Jetson Nano and the Cloud using pub/sub (100%) (Involved experimentation with features of pub/sub to ensure seamless communication. Identified issues with this service for this application). Writing report (50%).