The Cadenza Challenge – Task 1

(Author names removed for reviewer anonymity)

Abstract

The Cadenza project aims to better define what music personalised for someone with a hearing loss should sound like and explore the latest in machine learning to create improved listening experiences.

This experiment is based on the MUSDB18-HQ database, which includes 86 training songs and 14 validation songs. We have implemented two processes to derive the results; 1) DEMUCS and 2) Spleeter, producing two sets of results. The primary objective of both processes is to perform two stages of operation on the input audio stems. The first stage of demixing is performed after studying various models like Open-UnMix, Conv-Tasnet, and waveunet, leading to the choice of Hybrid Demucs and Spleeter to complete the audio source separation. For the second stage of remixing the primary focus is on applying compression, equalization and filters to facilitate remixing. We have used multiband compression to adjust the four control parameters of threshold, ratio, attack, and release. Also, we applied side-chain compression to control the dynamics of particular stems. In addition, we evaluated the output signal by the Hearing Aid Audio Quality Index (HAAQI) metric.

At the same time, the team incorporates the listener's unique hearing characteristics into the built model, providing personalised audio enhancements based on specific hearing profiles to improve the listening experience for people with different hearing abilities.

GitHub(demucs):https://github.com/skystriker233/E012-CAD1

GitHub(spleeter): https://github.com/sc22d2s/E016-CAD1

Index Terms: Hybrid Demucs, Spleeter, Multiband compression, Side-chain compression.

1. Introduction

The Cadenza Challenge is about improving the perceived audio quality of recorded music for people with a hearing loss. The Cadenza Challenge is to first demix stereo tracks into a VDBO (vocal, drums, bass and other) representation. This then allows for a personalised remixing for the listener that may support improved audio quality perception.

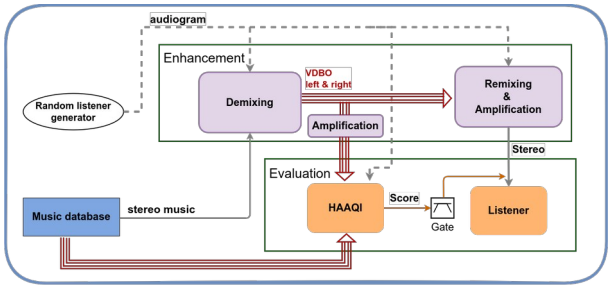

Figure 1: The baseline for the headphone listening scenario.

1.1. Demixing Stage

The first stage of the music enhancement process involves demixing the music using either the Hybrid Demucs model or the Spleeter model. By applying one of the models, the music is separated into its target stems, including vocals, drums, bass, and other stereo components.

1.2. Hybrid Demucs model

The Hybrid Demucs model utilizes a time-domain-based approach, specifically a wave-to-wave model that incorporates a U-Net structure and a bidirectional LSTM.

1.3. Spleeter model

Spleeter is Deezer's source separation library with pretrained models written in Python and uses Tensorflow.

1.4. Remixing Stage

The remixing stage focuses on further improving the audio applying various Digital Signal Processing (DSP) algorithms:

1.5. Butterworth filter

One of the key DSP algorithms used in the remixing stage is the Butterworth filter. This type of signal processing filter was chosen as it is designed to have a frequency response that remains as flat as possible within the passband. By applying the Butterworth filter, the audio's frequency components are adjusted to achieve a more balanced and even sound.

1.6. Multiband Compression

This method leverages multiband compression for audio balance and consistency by independently managing specific frequency bands. Utilising the four primary control parameters - threshold, ratio, attack, and release, it fine-tunes the highs and lows of each instrument or the total mix for a streamlined and refined audio output.

1.7. Sidechain Compression

Sidechain a.k.a. parallel compression is a technique that uses a signal from one instrument or voice to trigger a compressor on another signal. The side-chaining signal is used to determine the threshold of the compressor and thus control its effect on the signal to be compressed.

2. Design and Implementation

2.1. System information

- HPC: We utilised High Performance Computing to seamlessly process the large datasets and sophisticated audio separation models. We used the MobaXTerm interface to establish an SSH connection to our University's ARC4 systems.
- GPU: Our system was run on GPUs via ARC as well as on Google Colab for preliminary study and exploration. Using P4000 and V100 GPUs helped us achieve faster computational speeds.

- Cloud machine: We leveraged PaperSpace cloud services to create a virtual machine and run our enhancement process using their P4000 GPUs.

2.2. Model architecture

2.3. Description of the demixing stage

This section describes the demixing stage where Hybrid Demucs and Spleeter have been used for audio separation, along with individualized hearing corrections to each separated stream. The personalized hearing corrections are based on listener characteristics defined by an audiogram. These steps successfully personalize the audio and produce a remixed version of the song tailored to the individual's unique hearing profile.

Exploring Hybrid Demucs demixing:

In the Hybrid Demucs model, we built the Hybrid Demucs pipeline, formats the waveform into chunks of the expected size, and loops through the chunks (with overlap) and feeds them into the pipeline. We then collect the output blocks and combines them based on how they overlap.

Exploring Spleeter demixing:

Spleeter uses a deep learning model trained to perform source separation. The model has been pre-trained on a large dataset to identify and differentiate between different signals like vocals or bass, and to separate these from a mixed audio signal.

The 'spleeter:4stems' model is used for this purpose, which is specifically a U-Net model trained to separate audio into four stems.

Key specifications of Spleeter used in this task:

- The Separator object is initialized with the 'spleeter:4stems' configuration. This tells Spleeter to separate the audio into four stems: vocals, bass, drums, and other instruments.
- The AudioAdapter is a Spleeter class that provides a consistent interface for loading and saving audio files.
- audio_loader.load() is used to load the audio file into a NumPy array, resampling it to a sample rate of 44100Hz in the process.
- separator.separate(waveform) is where the final demixing happens. It takes the loaded waveform and separates it into the four stems.

In the Spleeter model, to process audio signals we use libraries such as *soundfile* to read and write audio files, and *numpy* for mathematical operations. In addition, we ran into computational limitations due to the high order of filters and large audio files. To deal with these issues, we used techniques such as down and re-sampling to reduce data size without compromising quality.

In summary, based on the characteristics of different listeners (defined by their audiograms), we applied a unique set of corrections to each stem. The corrections are applied using the NAL-R method implemented in the Clarity Toolset (an open-source library for hearing aid algorithm research). The corrections are applied to match the gain-frequency response to each listener's hearing abilities.

2.4. Description of the remixing stage

Keeping the balance of different instruments and making vocals more prominent are the focus of the remixing section. We have created a simple intelligent system which can access the audiograms and analyse the audiograms for individual listeners. According to different situations, the system will apply different methods to do remixing with stems.

Exploring Hybrid Demucs remixing:

For listeners who are not classified as having severe hearing loss, the compressor code will not be applied. Table 1 elaborates the hearing levels we assigned based on corresponding audiometric values in decibels. If a listener is with a moderately severe or severe hearing loss at certain frequencies, the system will decrease the level of bass, drums and other. At the same time, the multiband compressor will be applied to make vocals appear more prominent.

A multiband compressor is implemented based on the compressor which is provided in the baseline system. The multiband compressor includes a Butterworth filter which is used to isolate certain frequency ranges of the stem, so that the compressor only compresses these specific ranges. Since it was not practical for us to use a compressor on each stem due to the high time consumption when processing a large amount of audio samples, the vocals is chosen as the side chain to process the ‘other’ stems which can create more space for vocals. As ‘other’ may include keyboard and guitar, they might compete with vocals. However, it would have been preferable for us to use compressors on all stems and work on smaller frequency bands. The structure and flow of this process is demonstrated in Figure 2.

Level of hearing loss	Audiometric range (dB)	Classification
0	0 – 19 dB	No hearing loss
1	20 – 34 dB	Slight hearing loss
2	35 – 49 dB	Moderate hearing loss
3	50 – 64 dB	Severe hearing loss
4	Above 65 dB	Profound hearing loss

Table 1: Hearing loss classification

Exploring Spleeter remixing:

The remixing process in this experiment involves the application of specific listener characteristics to the demixed (source separated) audio stems, and then recombining these modified stems back into a single audio signal (mixture).

The remixing process is tailored to the hearing characteristics of each listener. This means that the resulting audio is not a simple recombination of the separated stems but is instead a customized version of the audio that should ideally sound better to the specific listener’s audiograms.

Key specifications of Spleeter used in this task:

- We apply a Butterworth bandpass filter to the remixed signal to filter out frequencies below 250 Hz and above 18500 Hz. This is done to focus on this specific frequency range.
- A 'FlacEncoder' object is created and used to save the processed audio data in the FLAC format. This is done to ensure audio quality is not degraded during the compression process.
- NALR is applied and maximum absolute value of each audio channel is saved to a .txt file. Audio data is normalized to the range of the int16 data type. This is a form of dynamic compression which ensures the volume of the audio stays within a certain range, preventing it from being too quiet or too loud.
- Resampling also takes place in this section where the audio is resampled to 32000Hz. Once all the filters and compressors are applied, we have our demixed, remixed, and enhanced signals.

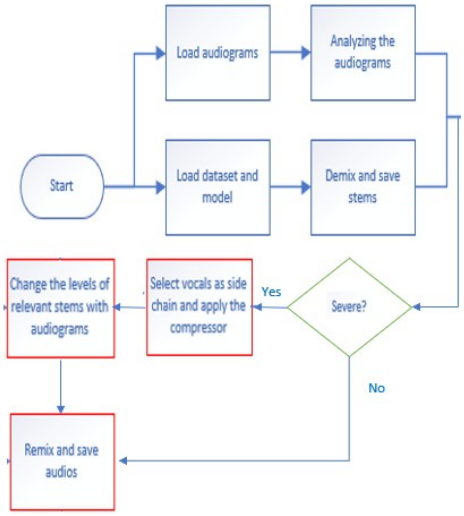

Figure 2: Diagram for remixing section.

3. Results

The result of our experiment provides us with two sets of audio signals and evaluation metrics as our submission has two audio source separation models (Demucs and Spleeter). We observed the differences between both models in all stages of the experiment (see Table 2 for summary). We were able to produce enhanced audio signals based on individual audiograms of listeners. Since we implemented the Demucs baseline provided by Cadenza, our HAAQI scores remain the same as existing baseline scores for Demucs. The enhanced signals, evaluation metrics and a detailed technical report comprised of an overall result of this experiment.

Metric	Demucs	Spleeter
Audio Analysis	Individual stem files (VDBO) generated using Demucs had higher clarity and quality	Spleeter produced clean audio stems, but we observed that in our case the clarity was lesser than Demucs
Computational Time	Running the enhancement process using Demucs took ~ 20 hours on a P400 GPU	Executing the process through Spleeter took ~6 hours using V100 GPU
Code Flexibility	We observed that Demucs had a few constraints while modifying or combining segments	Spleeter on the other hand had a much easier code flexibility providing scope for customization
Evaluation Metrics	The HAAQI scores generated by Demucs were higher than the scored for Spleeter separated audio stems	

Table 2: Comparing separation models

Since we made use of two models (Hybrid Demucs and Spleeter) in the demix stage, we will provide two submissions for the Cadenza challenge (Task 1). Therefore, we submitted the two results under the folders E012 and E016. The resultant signals at the end of the experiment have been detailed below

The demixed signals

-The VDBO (vocal, drums, bass, other) demixed signals for both left and right.

-Predefined 30-second segment.

-16-bit

-24 kHz sampling rate

-Compressed using the lossless FLAC compressor

The remixed stereo signals

-Predefined 15-second segment.

-16-bit

-32 kHz sampling rate

-Compressed using the lossless FLAC compressor

4. Discussion

During this research and implementation, we encountered several intensive discussions covering an array of topics related to audio source separation and audio signal enhancement. A lot of our knowledge base was supported by literature study which have been highlighted as references.

We extensively studied and conducted several experiments related to personalized audio separation. We established that handling such large-scale audio data along with complex and sophisticated audio processing models such as Demucs and Spleeter would require the access to a High-Performance Computing (HPC) and configuring an appropriate computational environment. Utilizing HPC facilities is critical given the high computational demands of training complex models and setting up this environment was a primary focus in our discussions. While performing exploratory analysis, we utilised Graphical Processing Units (GPUs) during evaluation stages, owing to their capacity for parallel processing which allowed us to efficiently handle complex machine learning models.

We also dissected the field of audio separation and personalization, considering various algorithmic approaches and evaluating their efficacy against our research objectives. The key factors that allowed us to arrive at a decision to choose Demucs and Spleeter revolved around model performance, accuracy, computational complexity as well as the quality of results they provide.

Establishing an initial strategy for audio separation was pivotal to ensure the differentiation of multiple sound sources within an audio stream. As the process evolved, this strategy underwent several iterations of refinement based on all the real-world, theoretical, and analytical knowledge we gathered during our study.

We delved into the comparison between standard and multiband compressors, elucidating the advantages of multiband compression in providing refined audio control over different frequency bands. We also agreed to utilize parallel compression for dynamic control of instrument or voice signals, thereby enhancing audio quality. As part of our auditory analysis, we decided to employ the Audacity software to scrutinize mixed music. We had a listening test on a smaller dataset in a studio. After evaluating these audios, it was discovered that increasing the volume of vocals may not be a wise method to make vocals prominent. Actually, it can result in an overall increase in the loudness of remixed audios. Therefore, we decided to lower the volume of other instruments to present vocals in the middle frequencies instead of adjusting the loudness of vocals.

A crucial aspect of our technical dialogue involved the integration of distinct code segments into a cohesive whole, with an emphasis on designing a code structure that is efficient, maintainable, and scalable. We were able to collate multiple segments of code and choose the best parameters, models and options which were mindful of computational complexities, running time, cost efficiency and most importantly results that fitted the objectives of our project the best.

5. Conclusion

To summarize, we applied the Hybrid Demucs model and the Spleeter model to demix the music and applied multiband compressor, side chain compressor and filters to remix the separated audio. As a result, we generated two results of the remixed stereo signals. We modified the enhanced the demixing, enhancement and remixing whereas the evaluation section was left untouched as regulations for the task recommended. The final result of this experiment aligned with the initial aim we had set out with, and we were able to produce audio signal stems and remixed music for the Cadenza Challenge task 1, for listeners with hearing loss to evaluate our stems based on perceived quality over headphones.

6. References

[1] Défossez, Alexandre, et al. “Music Source Separation in the Waveform Domain.” ArXiv:1911.13254 [Cs, Eess, Stat], 28 Apr. 2021, arxiv.org/abs/1911.13254.

[2] Sawata, Ryosuke, et al. “All for One and One for All: Improving Music Separation by Bridging Networks.” ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6 June 2021, https://doi.org/10.1109/icassp39728.2021.9414044. Accessed 10 July 2023.

[3] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, Yuki Mitsufuji. Open-Unmix - A Reference Implementation for Music Source Separation. Journal of Open Source Software, 2019, The Journal of Open Source Software, 4 (41), pp.1667. ff10.21105/joss.01667ff. ffhal-02293689f.

[4] Stoller, Daniel, et al. “Wave-U-Net: A Multi-Scale Neural Network for End-To-End Audio Source Separation.” ArXiv.org, 2018, arxiv.org/abs/1806.03185.

[5] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, Rachel Bittner. MUSDB18 - a corpus for music separation. 2017, ff10.5281/zenodo.1117371ff. ffhal-02190845.

[6] Stylianos Ioannis Mimilakis, et al. A Recurrent Encoder-Decoder Approach with Skip-Filtering Connections for Monaural Singing Voice Separation. 1 Sept. 2017, https://doi.org/10.1109/mlsp.2017.8168117. Accessed 1 June 2023.

[7] John W.B. Hershey, et al. “Deep Clustering: Discriminative Embeddings for Segmentation and Separation.” ArXiv (Cornell University), 20 Mar. 2016, https://doi.org/10.1109/icassp.2016.7471631. Accessed 21 Apr. 2023.

[8] Martínez-Ramírez, Marco A., et al. “Automatic Music Mixing with Deep Learning and Out-of-Domain Data.” Arxiv.org, 24 Aug. 2022, arxiv.org/abs/2208.11428, https://doi.org/10.48550/arXiv.2208.11428.

[9] Mitsufuji, Yuki, et al. “Music Demixing Challenge 2021.” Frontiers in Signal Processing, vol. 1, 28 Jan. 2022, p. 808395, arxiv.org/abs/2108.13559, https://doi.org/10.3389/frsip.2021.808395. Accessed 23 July 2023.

[10] Cohen-Hadria, Alice, et al. “Improving Singing Voice Separation Using Deep U-Net and Wave-U-Net with Data Augmentation.” ArXiv.org, 4 Mar. 2019, arxiv.org/abs/1903.01415. Accessed 23 July 2023.

[11] Défossez, Alexandre. “Hybrid Spectrogram and Waveform Source Separation.” ArXiv.org, 30 Aug. 2022, arxiv.org/abs/2111.03600. Accessed 23 July 2023.

[12] Cheeks, Joseph. IMPORTANCE of SPACE and HOW to CREATE IT USING SIDE-CHAIN COMPRESSION. 2017.

[13] Hennequin, Romain & Khlif, Anis & Voituret, Felix & Moussallam, Manuel. (2020). Spleeter: a fast and efficient music source separation tool with pre-trained models. Journal of Open Source Software. 5. 2154. 10.21105/joss.02154