CMPE491 – Senior Project I, Spring 2021

High-Level Design Report

Voiceolation

Emir Yılmaz

Kemalcan Güner

Yunus Emre Günen

Supervisor: Gökçe Nur Yılmaz

Jury Members: Aslı Gençtav, Venera Adanova

Contents

1. Introduction

1.1. Purpose of The System

Voiceolation is a music source separator that extracts vocals from songs. Either companies or individuals may need it for music information retrieval (MIR) and karaoke, may use it for their personal usage like mixing or editing a song.

Most people can produce music professionally or amateurishly at their homes, thanks to getting easy-to-use DAWs; their plugins, tutorials increased, diversified and DAWs became widespread. They - whether professional or not - require clean vocal tracks to use in their music projects. Some tracks' vocal channels are unobtainable because of issues with record labels like copyright, lost files etc. or simply the song is being too old for having stems for vocals. Let's assume you want to mash-up 2 songs you like - that is very popular nowadays. Unfortunately, they do not have any stems available because of mentioned reasons. That's where *Voiceolation* comes in to extract desired instruments, especially vocals. Instruments can be reproduced on DAWs, although it depends on music knowledge, it is hard to imitate perfectly in comparison with the having stems. On the other hand, vocals can not be produced with the same method. It should be recorded again with the same singer -it can be impossible, if not it will be grueling- or a different singer -it will not be the same.

As for corporations, some well-known companies such as Facebook and Deezer are trying to expand MIR scope by researching source separation techniques. These stems can be used for big data applications like genre classification, singer identification or vocal stems can be used for generating transcripts for a song or speech.

Imagine you are in a crowded party, everyone talking to each other and music blasting through your surroundings. Your brain is able to focus on a single conversation and ignore the other sounds. This phenomenon is called the cocktail party effect. Our system, like other audio source separation projects, will try to accomplish what our brains do.

Music source separators have become very common after the rising popularity of machine learning and neural networks. In the past, there were limited amounts of labeled datasets for training as they are difficult to label and require professional annotators. While the world is trying to overcome these hardships and gain great momentum towards the perfect source separator methods, they still need to improve, also datasets need to expand to be comprehensive that include various genres, languages, etc.

1.2. Design Goals

1.2.1. Reliability

Even the simplest sounds in real life are combinations of different frequencies, for example when you look at the waveform of the flute sound signal you won't see simple sine waves, it will be complicated. In order to extract this sound from a noisy environment, simple phase cancellation or brick-wall filters will not suffice because of the varying frequencies.

This is where we depend on neural networks for doing the classifications, making predictions, and learning the patterns of vocals/instruments for accurate isolating. When a signal is examined in the frequency-domain, it is so compatible with image processing methods. That's why we will use image processing and so on U-Net, because image processing is so advanced and dependable over waveform applications for now.

Moreover, the results of the project will be comparable with other projects in the benchmark using the same dataset[1].

1.2.2. Performance

We will be working on spectrograms while training the neural network. Because we will be focusing on vocals which are between 80-1000 Hz approximately (Goddard Blythe, 2017), we can crop the top half of the spectrogram with a low-pass filter which will result in a smaller working area which leads to higher performance for neural network training.

[1] Music Source Separation on MUSDB18
https://paperswithcode.com/sota/music-source-separation-on-musdb18

1.2.3. Usability

The project will be presented in the website to use for everyone who needs vocal separation to whatever and has basic computer skills. The website shall be easy to use.

1.2.4. Universality

If MUSDB18 were a larger and more comprehensive dataset, although one of the most diverse and largest, the project would be more eligible for all types of songs. However, after the model is created by using MUSDB18, the model may improve if other datasets were used and/or found.

1.2.5. Legality

Because of copyright issues, users must be agreed on that using *Voiceolation* is fully their responsibility. We do not recommend users to use *Voiceolation* for songs that may lead to legal issues. Thus, if they want to use *Voiceolation*, they must agree to the Terms of Service which does not conflict with DMCA. The user must know that *Voiceolation* has no responsibility for DMCA issues, it is their responsibility. Users know these and accept the Terms of Service, then they are able to use *Voiceolation*.

1.2.6. Security

We will not ask, hold or store any kind of user information. We do not save uploaded sound files because it can also contain private information. The file will be processed, and then will be deleted from the server. The processed version will be available for only limited time which is enough for listening and downloading. We will not share any kind of user data to third parties.

1.3. Definitions, Acronyms, and Abbreviations

Term	Definitions, acronyms, and abbreviations
stem	A stem is a group of audio sources mixed together, for example a vocal stem consists of vocal record and/or vocal chops.
DAW	Digital Audio Workstation[2]
MIR	Music Information Retrieval[3]
CNN	Convolutional Neural Network[4]
MUSDB18	"The musdb18 is a dataset of 150 full length music tracks (~10h duration) of different genres along with their isolated drums, bass, vocals and other stems."[5]
U-Net	"The U-Net is convolutional network architecture for fast and precise segmentation of images."[6]
STFT	Short Time Fourier Transform[7]
DMCA	Digital Millennium Copyright Act[8]

[2] Digital audio workstation - Wikipedia
[3] http://www.eecs.qmul.ac.uk/legacy/easaier/files/technical/retrieval/musicretrieval.pdf
[4] Convolutional neural network - Wikipedia
[5] https://sigsep.github.io/datasets/musdb.html
[6] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
[7] Short-time Fourier transform - Wikipedia
[8] DMCA Protection & Takedown Services

1.4. Overview

The project aims to extract/isolate vocals from any song. It is really common to encounter music tracks with inaccessible vocal stems, tracks due to various issues. Extracting vocals - in general, source separation - is not only used for remixing or editing but also used for music information retrieval (MIR) in order to define and brand music genres. There are similar softwares available, though most of them have flaws due to insufficient datasets, only focusing on acoustic music etc. Furthermore, our goal is even if we can not surpass already existing projects, we believe we might look at this subject with a different angle.

2. Current Software Architecture

Our current system is without the neural network architecture, as we leave the implementation of the model to the next course. Our priority was creating the spectrogram from audio files to be able to feed the neural network, since the proposed neural network model uses image processing.

We will use Python for everything since Python has advanced machine learning libraries, as for signal processing MATLAB is useful but Python does not fall behind on this subject with its extensive libraries like scipy and librosa. We will use these libraries along with a few helpers like numpy which excels at arrays.

For generating the spectrogram, a sound file is needed which is librosa or scipy can load as numpy ndarray along with sample rates. Sample rate is 44.1 KHz for MUSDB18 dataset.

As mentioned in performance subheading of design goals, we apply a low-pass filter. We used scipy.signal module for filtering in time-domain. We used the Butterworth filter type because it has minimal ripple in the passband also called ‘maximally flat magnitude filter’.

In order to generate a spectrogram we need to compute discrete Fourier transforms which will transform time domain to frequency domain over short overlapping windows. This process results in time-frequency domain and is called ‘short-time fourier transform(STFT)’. Default values of frame (window) size and hop size are 2048 and 512 samples respectively, which is default for librosa.stft.

Before plotting the result and observing the spectrogram a simple but effective modification should be done to frequency representation of the spectrogram. Lot of instruments and sounds, especially human voice, are at the lower end of human hearing range (20Hz- 20000Hz), because of that a linear representation of frequency spectrum's amplitude responses will be stacked on the bottom where all of the sounds exist. In order to solve this problem, the frequency scale of the spectrogram should be logarithmic. This allows more spread and detailed display on the lower side of the spectrogram where the vocal is located. Below are the spectrograms of the same music with logarithmic and linear frequency scales.

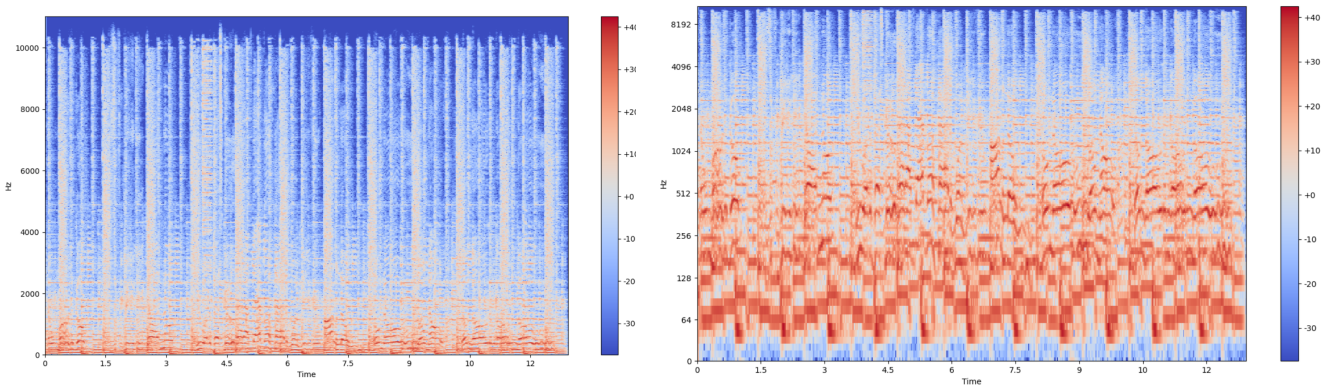

Figure 1: Spectrograms of the same music with linear(left) and logarithmic(right) frequency scales, created by the current software of the project.

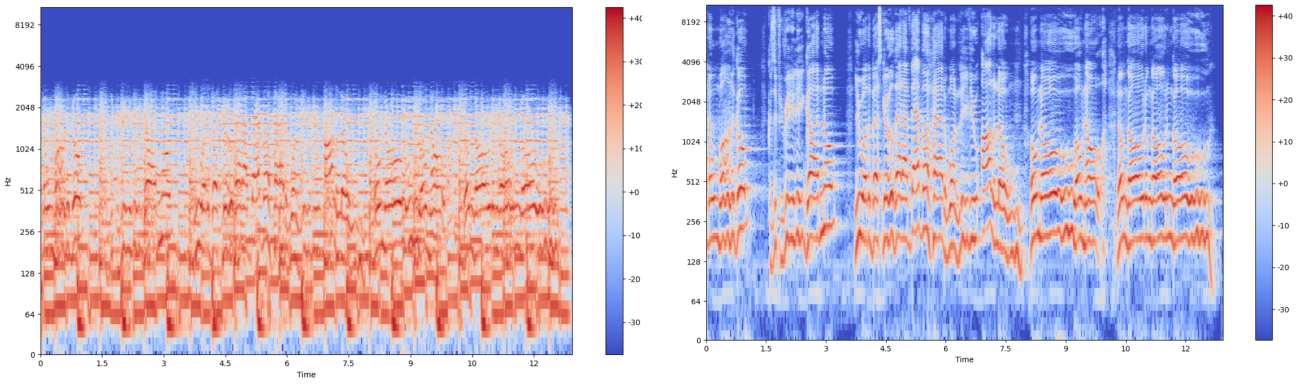

Figure 2: Complete mixture of a song (left). Vocal stem of same song(right)

These spectrogram comparisons indicate that even with a low-pass filter with 2048 Hz cutoff frequency no meaningful vocal data is lost. Because vocal frequency falls into 80-1000 Hz.

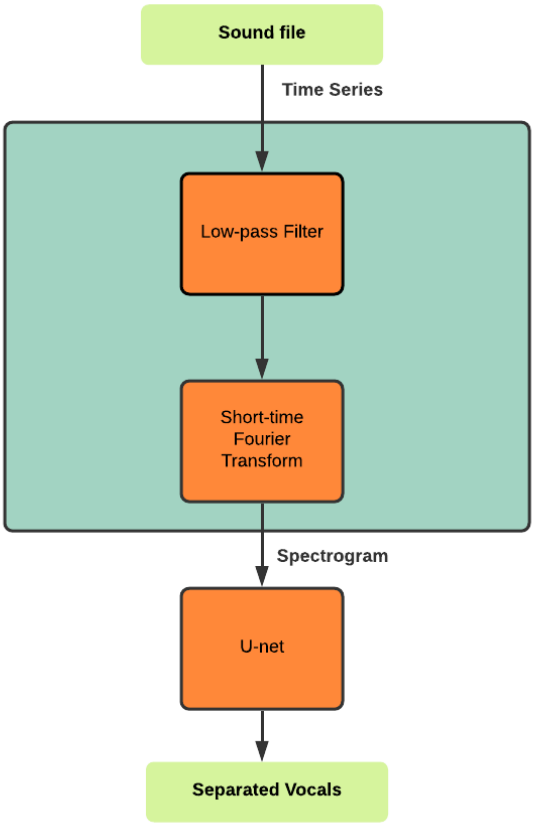

Figure 3: Block diagram representation of the planned architecture in basic manner.

3. Proposed Software Architecture

3.1. Overview

The core of *Voiceolation* software shall be a neural network. We will create a U-Net based CNN, but we cannot give detailed architectural information since our focus on neural networks will be the latter half of the project, so we mostly mentioned the website architecture that we will create in the last stage, after the training of the network.

In the proposed software architecture part we described analysis of subsystem decomposition in part 3.2. It is about relations between subsystems. In part 3.3, we described hardware software relationship. Part 3.4 and 3.5 are generally about data management and security. 3.7 is about the usage of the web site like how to start, end processes. All parts give detailed information that we used in *Voiceolation*.

3.2. Subsystem Decomposition

Our system has two main objectives as client side and server side. Client side consists of a web site which has upload-download pages. The only aim is building a bridge between a user and the server. Web site takes sound files as an input and sends it to the neural network. On that, we have two pre-processes elements: low pass filtering and Fourier transform. After these steps the input sound file turns to spectrogram. Then, that spectrogram goes into the U-Net phase. After some processes, the server gets output files and represents them on the web site to the client.

3.3. Hardware/Software Mapping

Voiceolation just needs a computer, mobile phone, tablet etc. which is able to open a web browser and so on our website. Also, a stable internet connection between user and the system for uploading and downloading .

Browser takes input data from the user and sends it to the GPU which is in the cloud. In the cloud, data is processed in the neural network. After the process, output data is sent back to the user via browser. Both sending and taking back data we use HTTP protocol.

3.4. Persistent Data Management

Our application does not hold any data of users, we do not ask e-mail, ID, password or anything. Users upload their sound files, *Voiceolation* processes these sound files, and gives 2 output files. After the output data, *Voiceolation* deletes all user input and output data.

The only data we will store is the MUSDB18 dataset -which will be irrelevant and will not be on the server after training of the neural network- and the neural network model.

3.5. Access Control and Security

We do not have access control, because *Voiceolation* does not have a login function. We do not check ID, password or email. The only thing we consider is Terms of Service. If a user accepts 'Terms of Service', s/he can use the application freely. We do not share any user data or user's uploaded tracks with third party applications.

3.6. Global Software Control

As mentioned previously, *Voiceolation* is based on client-server architecture. Whenever a user makes a request, the system starts to work on it. After making some processes, the system gets output files. Then these files are sent as a response by the system to the user. We have fundamentally 3 main phases. The first one is the request phase which starts with clicking the upload sound file by the user. The second step is the processing phase. This phase takes input files and works on it. Here we use U-Net to get results. After the processing phase ends, the response phase starts. System needs to send responses back to the user. Response data contain two outputs vocal and instrument sound files.

3.7. Boundary Conditions

3.7.1. Initialization

After clicking the website, the user may see a clear interface. We do not show any login or register page. The user only needs to accept "Terms of Service". After accepting the user can see the upload file screen. If the user does not accept s/he can not do any processes on the web page.

3.7.2. Termination

Because *Voiceolation* runs on a web page, the user may close the tab or web browser. S/he does not need to do any kind of log off operation, just needs to click the 'Close' button.

3.7.3. Failure

Any kind of interruption causes the re-upload of the files. Because we do not hold user's data (such as e-mail, ID or uploaded file). After the interruption (if any) users need to accept "Terms of Service" again and re-upload the sound files.

4. Subsystem Services

4.1. Web Application

4.1.1. User Interface

When a user enters the web page, s/he sees Terms of Services rules and a button to accept it. If the user accepts the terms of service s/he may continue to use the website. Otherwise, s/he can not do any process. The user sees 'Upload File' and 'Browse' (for local files) buttons. Now the user is able to upload a sound file from his/her desktop by clicking the 'browse' button as seen in Figure 4.

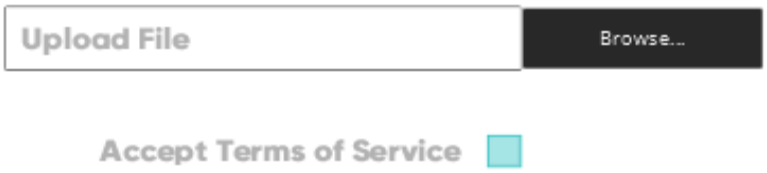

Figure 4: Audio file upload screen with simple representation

After uploading the file, the system starts to process and give results as two different options. One of the options is a Vocal sound file, another one is Instrumental sound file. The user may listen to output files on the web page by clicking the ‘play’ button as seen in Figure 5. Also, the user may download the sound files by clicking download button/s.

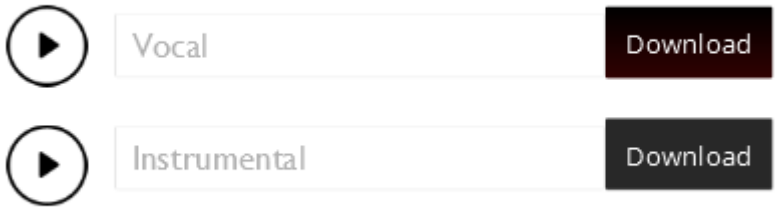

Figure 5: Listening and downloading the processed audio files screen with simple representation

4.1.2. Server

On the server side of the web application, there will not be a literal database and its management. It will get the file and create new sound files by processing it. After the separation process the input file will be deleted also the separated new sound files will be online just for a limited time then they will be deleted also.

4.1.3. Process

The given file will be processed by a trained neural network. The network gives two separated sound files -one of them is vocal and the other is instrumental tracks- to the server to present to the client.

4.2. Neural Network

The input file that came from the client, forwarding the network by server. The file will be a parameter to the program, first it will be preprocessed then goes to the neural network as input. Then, output will be sent to the client by the server.

4.2.1. Preprocess

4.2.1.1. Low-Pass Filter

The low pass filter is applied to the sound file to reduce the working area of the network for discarding overhead, faster and more reliable vocal separation. Filter applied on the time-domain by using the signal module from scipy.

4.2.1.2. Fourier Transform

The base architecture (U-Net) uses image processing, so the spectrogram must be created. First, STFT is applied to the ndarray coming from the low-pass filter by using librosa. Then, the spectrogram is created by using pyplot module from matplotlib and display module from librosa.

4.2.2. Network

U-Net is a CNN which is used for biomedical image segmentation. It has a lot of variants and can be adopted into spectrogram analysis. Our trained model will predict the vocal from the spectrogram and return the isolated vocal as a spectrogram to postprocess.

4.2.3. Postprocess

After the image processing operations in the time-frequency domain, software needs to convert back the processed data to the user as a time-domain signal. Hence, the system applies an inverse STFT to the spectrogram and writes out a sound file to the user for download.

5. References

1) Choi, W., Kim, M., Chung, J., & Jung, S. (2021, June). LaSAFT: Latent Source Attentive Frequency Transformation for Conditioned Source Separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 171-175). IEEE. https://arxiv.org/abs/2010.11631v2
2) Goddard Blythe, S. (2017, March 17). *Appendix 2: Frequency Range of Vocals and Musical Instruments*. Wiley Online Library. https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119164746.app2
3) Kumar, A. (2020, May 20). *Audio Source Separation with Deep Learning - Towards Data Science*. Towards Data Science. https://towardsdatascience.com/audio-source-separation-with-deep-learning-e7250e8926f7
4) *scipy.signal.butter — SciPy v1.6.3 Reference Guide*. (2021). SciPy v1.6.3 Reference Guide. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html
5) Valerio Velardo - The Sound of AI. (2020, September 10). *How to Extract Spectrograms from Audio with Python* [Video]. YouTube. https://www.youtube.com/watch?v=3gzI4Z2OFgY&t=927s