CMPE492 – Senior Project II, Fall 2021

Low-Level Design Report

Voiceolation

Emir Yılmaz

Kemalcan Güner

Yunus Emre Günen

Supervisor: Gökçe Nur Yılmaz

Jury Members: Aslı Gençtav, Venera Adanova

Contents

1. Introduction

Music is one of the oldest art forms of human history. Even literature was carried on with music and oral poems until the invention of writing. In other words, humanity has been dealing with music in some way since its existence. Today, the global music industry has approximately 22 billion dollars in revenue. Mixing, mastering, and editing sounds is essential and indispensable for today's music industry. While editing music files, we may need to separate music sources due to various difficulties such as copyright issues, lost files, being too old to have a stem, etc. Also, the existence of songs with inaccessible vocal parts is more common than expected.

Thanks to new algorithms, methods, and professional fields -such as machine learning, neural network-, separating music source has become common. There are few software that do this job on the market. But not most of them give us reliable results due to their algorithms or the way they coded. Also, they might not be open source or free to use. These programs are allowed to limited people.

We realized there was a need for this field and decided to build a vocal source separator. *Voiceolation* is a music source separator that extracts vocals from songs. *Voiceolation* is not only used for remixing or editing but also used for music information retrieval (MIR) in order to define and brand music genres. The music information retrieval has become a more important part for some well known companies like Spotify, Facebook, and Deezer. They already have some projects on music separation techniques to expand MIR views for example, genre classification, identify vocals and language, etc.

1.1 Trade-offs

1.1.1 Reliability vs. Cost

There is a direct proportionality between the size of the dataset and the effectiveness of the neural network. A bigger dataset means a more powerful neural network. There are millions of songs in this world but in order to train them for our model they need to be labeled. To create a source separation dataset, you need STEMs for each song which is not easy to acquire. As a result, more data for a dataset means more memory, money, and time. We used an already prepared dataset which is MUSDB18 instead of building our own dataset from scratch.

1.1.2 Security vs. Self-upgradability

As we mentioned over, more data means more reliable results for our project. While users use our project, they upload their song files and get vocal and instruments files. That means we would gain more and more datasets after each user usage. However, because of user security, we do not keep their data. We do not keep either uploaded music files nor result files which are vocal and instrumental song files

1.1.3 Performance vs. Accuracy

As we mentioned in previous reports, we cut off signals that are above a certain frequency range. The reason is human vocals are around 80-1000 Hz and we do not need to work on higher frequencies. So, we eliminate the unnecessary frequencies between 2048Hz and 20000 Hz with a low-pass filter. However, there can still be meaningful vocal sounds above our cut-off threshold because of production methods such as using vocoders, vocaloids, autotune, reverb, and harmony. Basically, we trade off an unlikely amount of over 2048 Hz vocals to get the more processable and clean spectrogram.

1.2 Interface Documentation Guidelines

In this document, we use a standard naming method which is upper camel for class names. Then, we have descriptions for classes. After class description, we have methods with name and short mission information.

class Class Name
Description of the class.
Methods
Method: To state what the method does.

1.3 Engineering Standards

We use UML guidelines for general purpose modeling object oriented and IEEE standards for projects of general computer engineering.

1.4 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.
CNN	Convolutional Neural Network[1]
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."[2]
U-Net	"The U-Net is convolutional network architecture for fast and precise segmentation of images."[3]
STFT	Short Time Fourier Transform[4]

2. Packages

2.1 Training Model Package

Voiceolation will be a neural network project, so we will write and train a network model. This model will be used for vocal source separation.

2.1.1 Dataset

To train an artificial intelligence, there should be a dataset. We will use MUSDB18, it has mainly 5 labeled stems -as isolated drums, bass, vocals, and others part- of every song in it. For our goal that is vocal separation via image segmentation, the vocals and other stems are essential. The neural network will be trained spectrograms of songs in the dataset by using their library (musdb) and preprocessor package.

[1] Convolutional neural network - Wikipedia
[2] https://sigsep.github.io/datasets/musdb.html
[3] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
[4] Short-time Fourier transform - Wikipedia

2.1.2 Preprocessor Package

The preprocessor package will generate a spectrogram of every song in the dataset by applying a low-pass filter. Vocals are between 80-1000 Hz approximately (Goddard Blythe, 2017), after observing the spectrograms we found some outlier vocals on higher frequencies. So, we decided on 2048 Hz as the threshold of the filter. We will use the Butterworth filter also known as 'maximally flat magnitude filter' to get as smooth a spectrogram as possible. Also, in order to generate a spectrogram, we need Fourier transforms, 'short-time Fourier transform(STFT)' will be used on the preprocessor package.

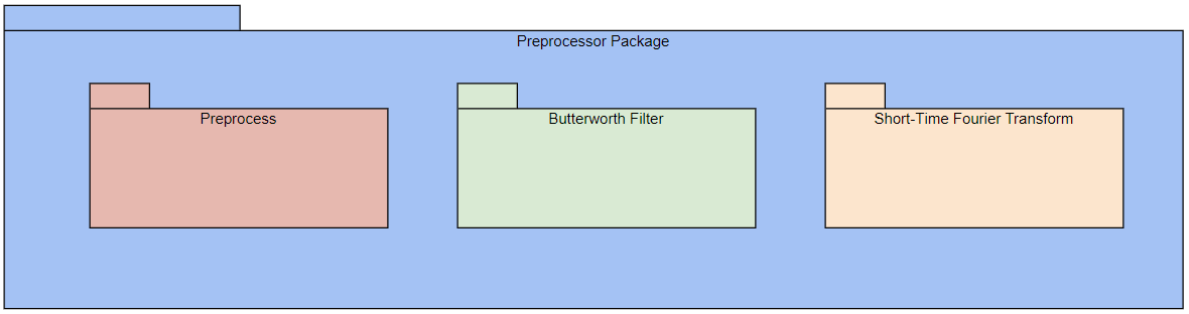

Figure 1: Preprocessor Package Diagram

2.1.3 Neural Network Package

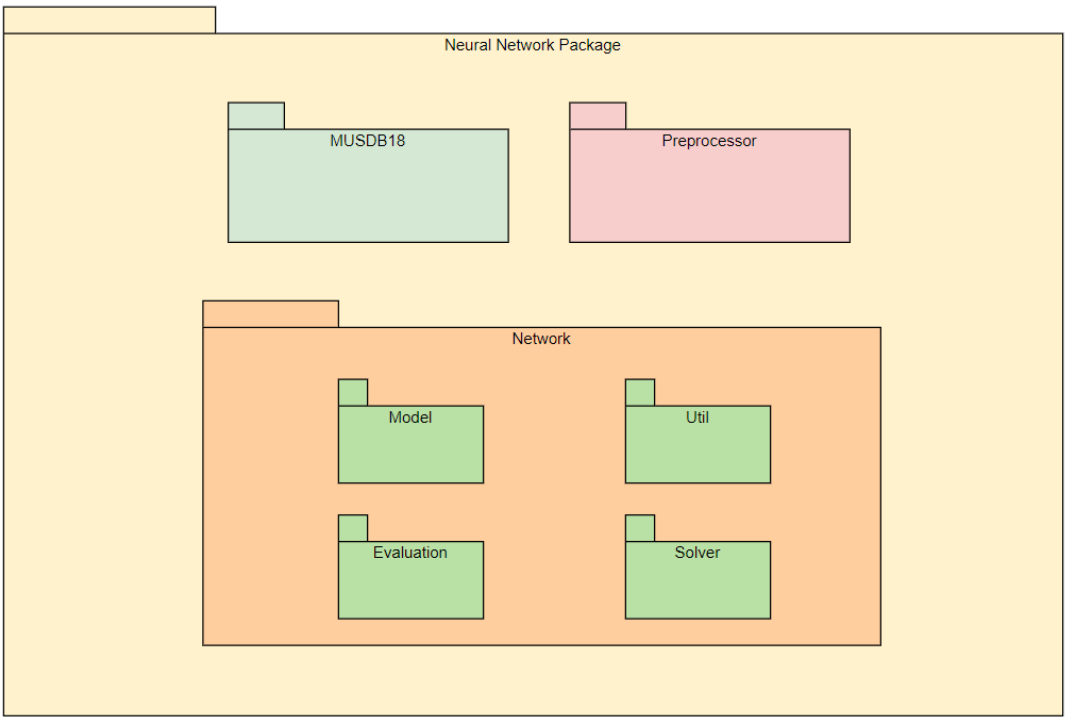

Figure 2: Neural Network Package Diagram

The preprocessor-generated spectrograms of the songs in the dataset (MUSDB18) are used for training and modeling the neural network. Our CNN architecture will be U-Net or akin to U-Net architecture. The U-Net architecture will be seen below.

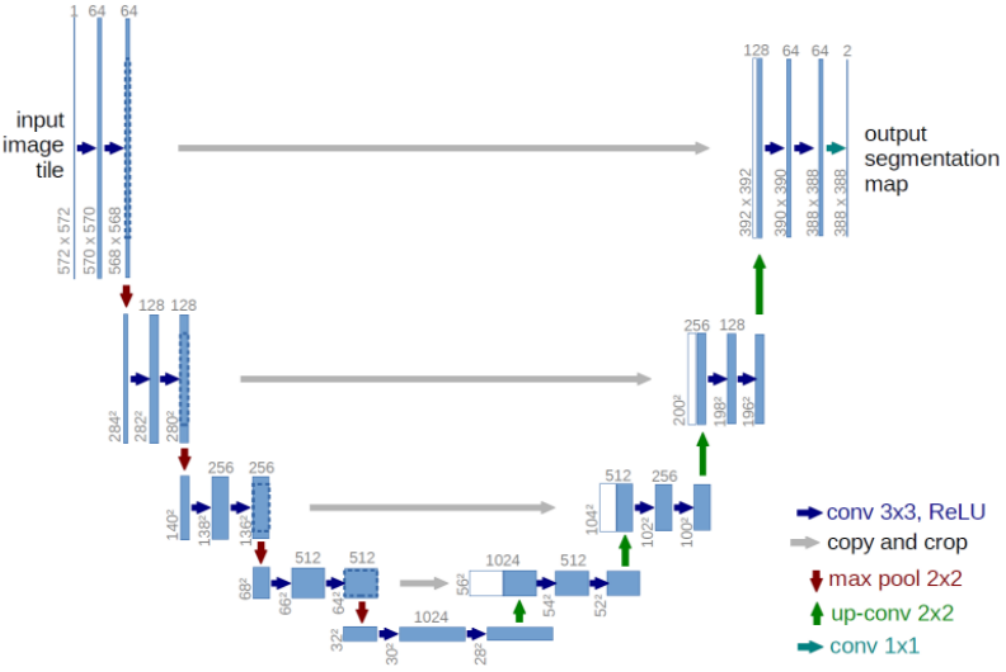

Figure 3: U-Net Architecture

2.2 Application Package

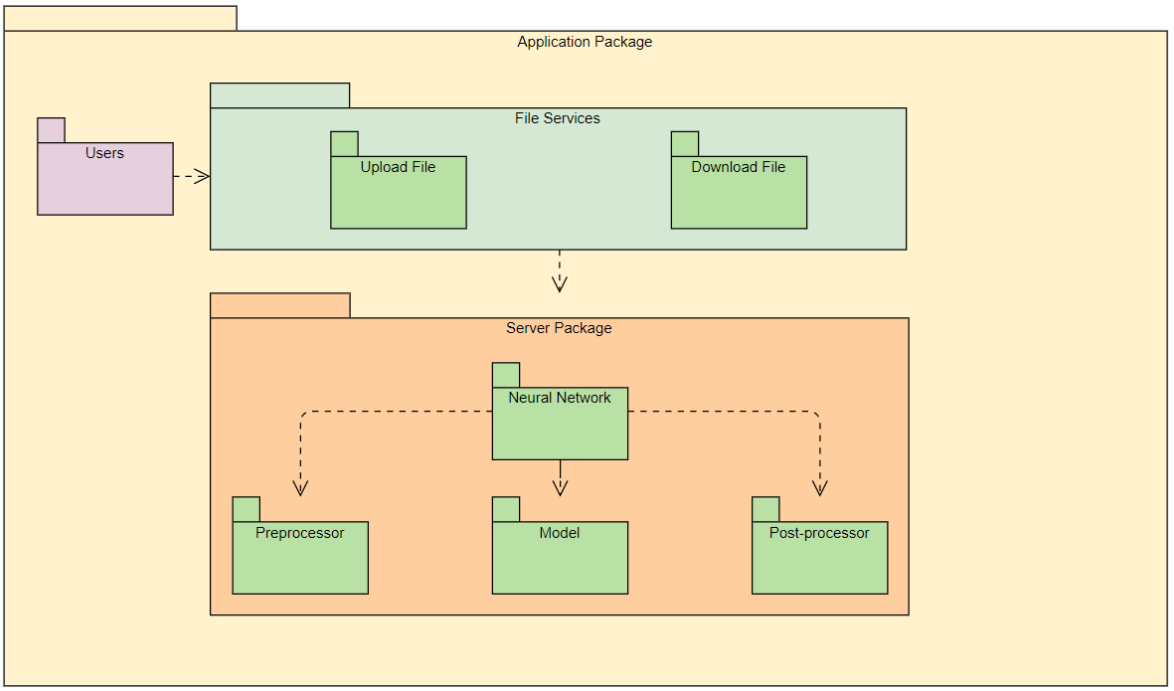

Figure 4: Application Package Diagram

2.2.1 File Services Package

It will handle sound files: gets a music file from the user, forwards to the server package for vocal separation; and gets the sound file that consists of isolated vocals from the server, forwards to the user for downloading.

2.2.2 Server Package

Like in the neural network of the training part, the server will isolate the vocal and present a sound file to the file services. This time, the model will be prepared thanks to training.The post-processor will convert the spectrogram output of the neural networks to the audio file and send it to the file services.

3. Class Interfaces

Due to not fully completed neural network architecture, we did not add methods to 3.4, 3.5, and 3.6 as they are to be decided.

3.1 Preprocessor

Class: Preprocessor
This class is used for loading audio files as floating time series.
Methods
get samplerate: this method gets samplerate of the audio file which is fixed in musdb dataset but we keep it flexible just in case *get ndarray:* this method gets n dimensional array of audio file *load musicfile:* loads sound file from file explorer or/and client side of website

3.2 Butterworth Filter

Class: Butterworth Filter
This class is used for applying a low-pass filter to audio.
Methods
set cutoff frequency: this method sets the threshold for filter, it will keep the lower frequencies and suspend the higher frequencies because it will be setted as low-pass filter *set sampling frequency:* the sampling frequency that we will use will be modified by this method, it will transform into nyquist frequency and normalized to fit scipy's digital butterworth filter *set order:* sets order of the filter high orders

3.3 Short Time Fourier Transform

Class: Short Time Fourier Transform
This class creates a time-frequency domain that represents the audio signal by taking discrete fourier transform over short time.
Methods
getData: this method gets filtered audio signal data as n dimensional array *set framesize:* this method sets the framesize for stft, default is 2048 *set hopsize:* this method sets the hopsize for stft, default is framesize/4

3.4 Model

Class: Model
This class contains a U-Net model for training. It is a convolutional neural network. This class will have U-Net and training environment parameters.

3.5 Evaluation

Class: Evaluation
This class is for testing the model's accuracy for unseen but labeled data in order to improve it for future runs.

3.6 Solver

Class: Solver
This class is for improving the loss by using a stochastic gradient descent based optimizer. We will use Adam optimizer as it is the best option in most deep learning cases. Class will have evaluation environment parameters and result displayer method.

3.7 Post-Process

Class: Postprocess
This class is for reverting the spectrogram back to audio signal via reverse short time fourier transform and writing it to a sound file.
Methods
inverse_stft: this method reads the processed spectrogram (ndarrays) and turns it back to a floating time series. *write file*: takes the time series in order to write it out to a sound file.

4. References

1. Choi, W. (2020, October 22). *A pytorch implementation of the paper: "Lasaft: Latent source attentive frequency transformation for conditioned source separation"*. GitHub. Retrieved October 21, 2021, from https://github.com/ws-choi/Conditioned-Source-Separation-LaSAFT.
2. Global Music Report 2021. (2021, March 23). *IFPI Issues Global Music Report 2021*. Retrieved October 21, 2021, from https://www.ifpi.org/ifpi-issues-annual-global-music-report-2021/.
3. Jun Hyun, L. (2018, June 18). *Pytorch implementation of U-Net, R2U-net, attention U-Net, and attention R2U-net.* GitHub. Retrieved October 21, 2021, from https://github.com/LeeJunHyun/Image_Segmentation.
4. McFee, B., Metsai, A., McVicar, M., Balke, S., Thomé, C., Raffel, C., Zalkow, F., Malek, A., Dana, Lee, K., Nieto, O., Ellis, D., Mason, J., Battenberg, E., Seyfarth, S., Yamamoto, R., Viktorandreevichmorozov, Choi, K., Moore, J., … Thassilo. (2021, May 25). *Librosa/librosa: 0.8.1RC2*. Zenodo. Retrieved October 21, 2021, from https://zenodo.org/record/4792298#.YXEU6chfhPY.
5. Meseguer-Brocal, G. (2019, November 2). *Control mechanisms to the U-net architecture for doing multiple source separation instruments*. GitHub. Retrieved October 21, 2021, from https://github.com/gabolsgabs/cunet.
6. *MUSDB18*. SigSep. (n.d.). Retrieved October 21, 2021, from https://sigsep.github.io/datasets/musdb.html.
7. *Numpy and scipy documentation*. Numpy and Scipy Documentation - Numpy and Scipy documentation. (n.d.). Retrieved October 21, 2021, from https://docs.scipy.org/doc/.
8. Ronneberger, O. (2015, May 18). U-Net: Convolutional Networks for Biomedical Image Segmentation. Retrieved October 21, 2021, from https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/.
9. Sun, J. (2015, November 9). *Python Lowpass Filter*. Gist GitHub. Retrieved October 21, 2021, from https://gist.github.com/junzis/e06eca03747fc194e322.
10. *What is Package Diagram?* What is package diagram? (n.d.). Retrieved October 21, 2021, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/.