CMPE492 – Senior Project II, Fall 2021

Final Report

Voiceolation

Ahmad Amireh

Emir Yılmaz

Kemalcan Güner

Yunus Emre Günen

Supervisor: Gökçe Nur Yılmaz

Jury Members: Aslı Gençtav, Venera Adanova

Table of Contents

1. Introduction

Music is regarded as one of the oldest art forms of human history, dating back to thousands of years before the modern age. It's composed and performed for various purposes, ranging from poems and operas to religious or ceremonial purposes, or as an entertainment product, in the form of songs or video clips, published for the marketplace.

The art of music and musical performance turned from the old-fashioned ways to one of the leading business forms of modern history, with a market share worth approximately 22 billion dollars. Mixing, mastering and editing musical notes and sounds are an essential and indispensable tool for today's music industry, due to the heavy demand for music in the marketplace.

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. With the help of modern algorithms, machine learning and neural networks, separating music sources has become a common and required practice in the music industry. This is where the need for a music source separation software arises from; hence our project idea - Voiceolation.

Music source separation (MSS) is the process of separating a music piece (song) into its individual sources, such as vocals and accompaniment. Neural network based methods have recently been used to tackle the MSS problem, and can be categorized into spectrogram and time-domain based methods. Nevertheless, there is still a lack of research that surrounds using complementary information of spectrogram and time-domain inputs for MSS.

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. 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[1]
MIR	Music Information Retrieval[2]
CNN	Convolutional Neural Network[3]
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."[4]
U-Net	"The U-Net is convolutional network architecture for fast and precise segmentation of images."[5]
STFT	Short Time Fourier Transform[6]
DMCA	Digital Millennium Copyright Act[7]

2. Final Architecture and Design Details

2.1 Overview

In this section, the system's architecture and design features will be explained in detail. Firstly, observing the fragmentation of the system, then looking over the software/hardware mapping and finally data management details.

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

2.2 Subsystem Decomposition

The system includes x modules. Model, Train, Prediction, and Utilities that consists of pre and post processes.

Model

Default U-Net architecture with modifications of filter size, strides and image size. Because of the change in image size, extra convolution and deconvolution layers are added. So in the end, our training model architecture has 12 convolutional layers with kernel size = (5x5) and strid = 2 as their activation functions as ReLU.

First half of these layers are in the contracting path, which is a common convolutional neural network process, and the second half is called the expansive path and layers consist of deconvolutions or convolution transpose in order to achieve upsampling of the feature map.

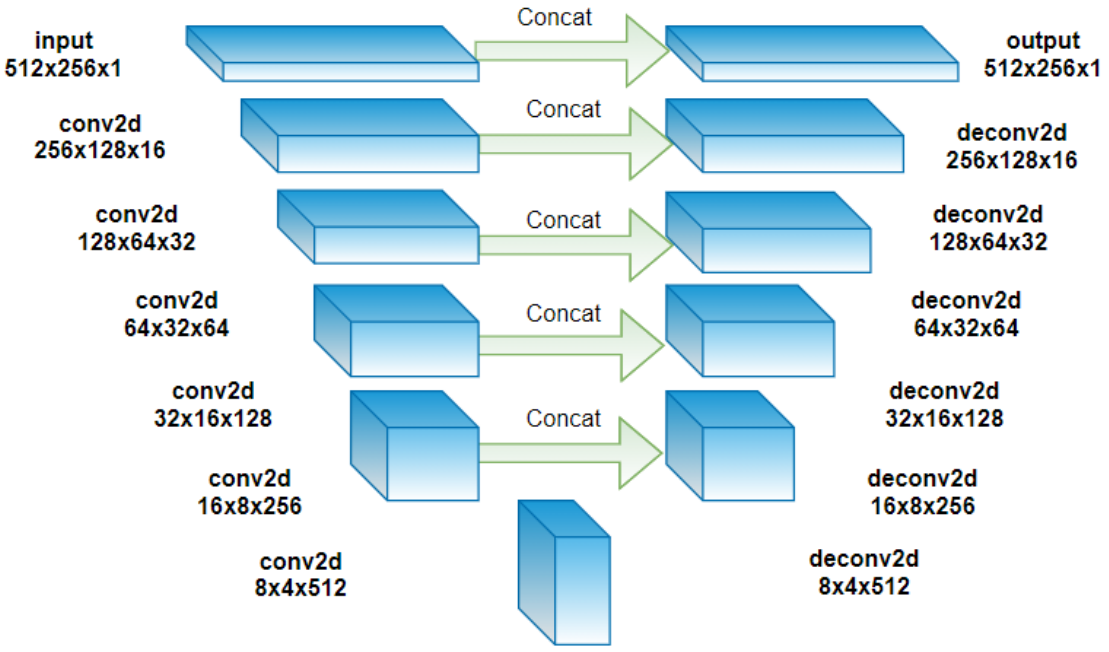

Figure 1: Model architecture.

Processing Dataset

In order to train U-Net we need to convert audio files to spectrogram images. Before that, the dataset (MUSDB18) is resampled to lower sample rates (to 16000 from 44100) in order to increase the speed of computations. Then .wav files are converted to spectrograms via librosa.stft() and saved as .npz (compressed NumPy files). This also resulted in the 45 GB dataset being reduced to 2.7GB.

Train

Due to lack of memory in the GPU we needed to change the training system to handle memory allocations. So we are picking 2 to 3 songs at a time and training them with low epochs. It is cumulatively over 100 epochs because of the looping nature of our training system. If we try to load all the dataset into the memory it will not be enough.

We also decided to save the model as .h5 and .json to the drive with every epoch. This allows us security because we are sometimes still getting memory errors and if an error prevents us from completing the training we can just use the model from the previous epoch and continue where we left afterwards. When compiling the model we use Adam as optimizer and mean squared error as loss function.

Predict

This is the trained version of our model and takes an input, applies the preprocess methods that we also use for training then predicts the vocals of said song. It can show its spectrogram and save it as a .wav file.

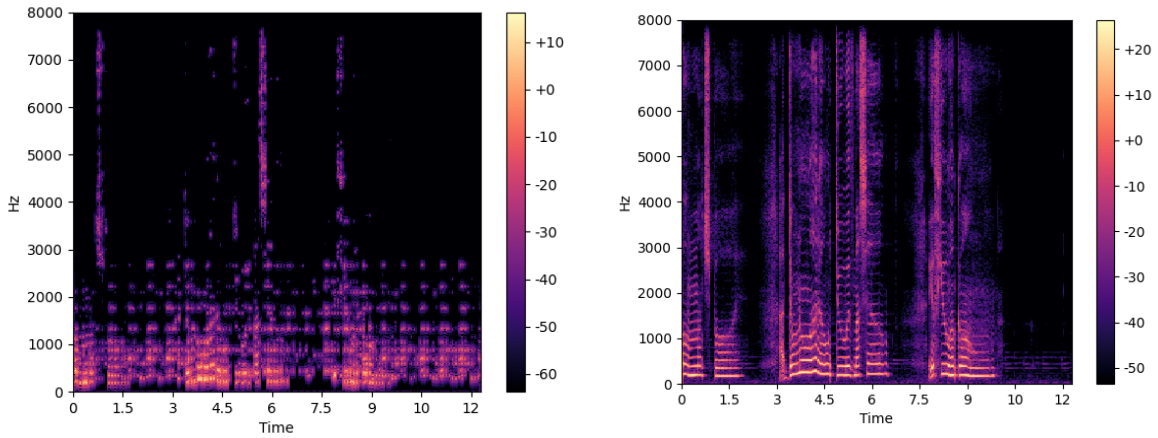

Figure 2: Prediction spectrogram (left) and target spectrogram (right) examples.

Test

As we mentioned in previous reports about musdb18 dataset. This dataset contains 50 tracks to test the result.

2.3 Hardware/Software Mapping

After preparing the dataset, we train the model on the GPU to train it faster. In order to train a model with GPU in tensorflow you need a CUDA supported NVIDIA Graphics card which Google Colab provides. We also had a laptop with NVIDIA GeForce GTX 960M graphics card to explore our software both locally and online.

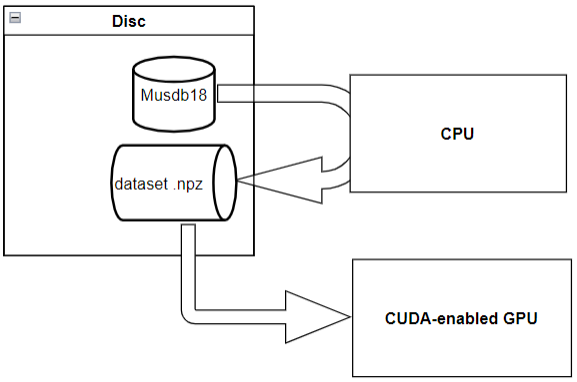

Figure 3: Basic presentation of mapping.

3. Engineering Solutions

3.1. Global Aspect

MIR is researched by colossal global companies like spotify, facebook, deezer and every minor development around the world is affecting the sector. While there are a lot of different MIR applications, Source separation is a relatively new topic and it is gaining a lot of interactions between researchers. We are proposing a different perspective that trains the model swiftly by some preprocessing methods like downsampling and filters.

3.2 Economical Aspect

Big companies that use advanced source separation technologies, generally are not open-source and most of them do not sell the said software to normal people. They are using it to improve their main softwares. So in an economical context, our project will be open-source and free to use by everyone.

3.3 Societal Aspect

Music is a way of expressing your emotions. Humans use this way of expression to communicate with other humans. You can make karaoke songs with Voiceolation and sing over the instrumentals. Or the opposite, you can isolate your favorite vocals from any instrument and play guitar as an accompaniment. Also, with pandemic and everyone being at home, a lot of people tried to become musicians or make music as a hobby. One of the easiest ways to make music is using a DAW on computers without needing an instrument. However, you need vocal samples or instruments to make music or remix on computer and Voiceolation can provide stems.

3.4 Ethics and Professional Responsibilities

We are aware of DMCA and copyright issues. In the testing phase of Voiceolation, we can use copyrighted music legally, but are not allowed to share the results of these songs [1]. In order to reduce confusions, we will use open-source non-copyrighted datasets. In every possible way, we want to use also the most popular copyrighted songs to get real-life results.

Because of copyright issues, users must agree that using Voiceolation is fully their responsibility. We do not recommend users to use Voiceolation for songs that may lead to legal issues. The user must know that Voiceolation has no responsibility for DMCA issues, it is their responsibility.

We respect users' privacy. We do not ask, hold or store any kind of user information. In addition, we do not save uploaded sound files because they can also contain private information. We use the sound file only for processing and sharing the result of separation for the user and delete when the user is done.

3.5 Public safety

Voiceolation does not collect or store any personal information about its users, nor does it sell or exchange data with third-parties. It is also out of reach of malicious parties or malwares because no information about users is retained or displayed.

3.6. Judgements and Impacts to Various Contexts

Impact Context	Impact Level (out of 10)	Judgement
Safety	5	Does not inquire any private user data to operate, does not share data to any third parties. However, it requires the user to upload a file, and offers a downloadable result.
Global	2	The service is provided for free, but can be promoted through ad campaigns to attract more users.
Economic	0	Free but requires internet connection.
Environmental	0	None

4. Technologies and Tools

4.1 Tensorflow and Keras

Tensorflow is a very popular open-source machine learning library. You can write neural network layers, activation functions, loss functions and other things with TensorFlow and Keras, also an open-source software library used as an interface for TensorFlow.

4.2 Google Colab

Google colab is a Jupyter Notebook environment but working on Google servers and Cloud. Colab provides GPU RAM and disk to its users and we used their GPUs as well as our GPUs to train our model.

4.3 Python

Python is a popular programming language that has a lot of libraries. Nowadays, when the subject is AI or Machine Learning it is the first one that comes to people's mind because of the variety of libraries. We also used lot of libraries like Numpy for multidimensional arrays, librosa for audio manipulation and fourier transform, and the aforementioned NN libraries Tensorflow and Keras

4.4 CUDA and NVIDIA

When we tried to train our model in our computers. It will only train in the CPU in default TensorFlow settings. In order to use your GPU for faster training, you need to have a CUDA enabled NVIDIA graphics card. CUDA is an API that allows to use graphics cards in our software

5. Testing Details

5.1 Neural Network Testing

Theoretically, training an artificial neural network is accomplished by finding the optimal values for the weights and biases with tests. Training is a process that can be defined as train-test-valid procedures in supervised systems. Neural networks and testing are already a unified concept since a neural network model trains itself by comparing the outcome with labeled data. There are other external evaluation methods for testing the system output such as SDR (Source-to-Distortion Ratio), SIR (Source-to-Interference Ratio), and SAR (Source-to-Artifact Ratio) in the case of audio source separation. The SDR approach evaluates the ratio of the actual source audio to errors such as noise, artifacts, and so on. The findings are expressed in decibels (dB), and the greater the number, the better. On MUSDB18[8] and at SiSEC

[8] MUSDB18 Benchmark: https://paperswithcode.com/sota/music-source-separation-on-musdb18

2018[9], we will use these numbers for benchmarking and comparing our results and other related activities.

There is also an optional test called 'overfit testing' that may be used to prevent overfitting while training neural networks with the dataset. Callback functions can be used to interfere with or check any stage of the training procedure. The EarlyStopping function in Keras, for example, can be used to avoid overfitting. The validation and training values are monitored by this function. When validation loss begins to climb after a certain number of epochs, it indicates the onset of overfitting and indicates that we will need to cease training to avoid overfitting. As a result, our test case would be to see if validation loss rises with time.

5.2 Bottom-Up Integration Testing

From the bottom to the top, we'll test every object and function. For example, we wrote a preprocess function that calls other functions such as short-time fourier transform (STFT) and Butterworth low-pass filter functions; consequently, test the callee functions first, then the preprocess function. After that, we'll test their integration, and so on at each phase. One of the most significant benefits of bottom-up integration testing is that it allows us to test multiple subsystems at once. Because we know which subprograms and modules are integrated or working up to our testing state in bottom-up integration testing, we can localize an error when we find it.

5.3 Unit Testing

Every unit, such as objects and subprograms, shall be tested independently as much as possible. For example, at the very end of the process, we will utilize the write_soundfile function. We will, however, test it independently from the vocal separation by just sending time series and seeing if it can convert them to sound files. We prefer unit testing because when we encounter an error, we don’t have to dive deep in the code again to fix it.

[9] SiSEC MUS 2018 Evaluation Results: https://sisec18.unmix.app#/results/vocals/SDR

5.4 Performance Testing

The system will be subjected to volume testing. Since our dataset is sizable, we need to examine whether our data structures can handle overflows or other extreme situations. For instance, if a user uploads high quality music file with .FLAC extension, we may have trouble converting it to .wav or even applying short-time fourier transform due to their sizes being larger than samples utilized in the training model.

Furthermore, we intend to create a website with a simple UI for public usage, and efficiency is critical. As a result, we'll put our source separation concept to the test in terms of performance and speed. Our goal is to separate the vocals of a typical song in under a minute.

5.5 Documentation Testing

Since we are using a lot of libraries and dependencies, a requirement file and technical documentation are required to keep our software maintainable and consistent. We will write requirements.txt and/or a pipfile to maintain libraries and configure the project after we get a stable version of it. Also, whenever there is a change, we will update these files.

6. Maintenance Plan and Details

Maintenance will be carried out in response to user feedback mainly, which may be collected in a form of email sent by the user. Since the service is provided online through a website, it retains the neural network model. That, alongside user feedback will be the main focus on which the service will be improved in terms of accuracy and performance. The communication will be established through the contact section on our website.

General improvements and bug fixes shall be delivered in three-month periods whereas larger and major updates to the performance will be released more or less once a year depending on the feedback collected and the behind-the-scenes improvements carried by the team.

7. Other Project Elements

Teamwork Details and New Knowledge Acquired, and Learning Strategies Used are all detailed in this part.

7.4 Teamwork Details

7.4.1 Contributing and functioning effectively on the team

Our group was divided into subgroups, each with its own set of tasks. The subgroups communicated amongst themselves, and we tried to communicate with everyone in the team on a regular basis to summarize our progress. Having subgroups, rather than a single large group, helped us keep track of tasks more effectively, and contributing was easier since assignments could be allocated more simply, and we could spend more time focusing on our tasks rather than continually reporting to a large group of people.

Additionally, even though each team member had their own assignment, if one completed theirs, they would assist the other team members in completing theirs, ensuring equal teamwork and a collaborative work atmosphere. For this purpose, the project maintained two different folders in Google Drive. One of them was used to keep track of the project's progress and write reports. The other contained Google Colab files for neural network model training. We could easily track our progress and share them with the supervisors and jury members thanks to these repositories and files.

7.4.2 Helping creating a collaborative and inclusive environment

Prior to beginning the project, each group member discussed their previous experience with relevant technology as well as the aspects of the project they would like to work on. We were able to focus on our particular capabilities and divide the work amongst ourselves in a way that emphasized our abilities as a result of this discussion. Some of the group members were taking a machine class alongside working for this project, so knowledge of machine learning modeling was gained which helped with performing certain tasks.

7.4.3 Taking lead role and sharing leadership on the team

Although our group did not have an appointed leader, some members were more involved in planning meetings, organizing report forms, and reminding others of deadlines. Overall, subgroups distributed work among themselves as they saw fit, then reported to one another. Everyone had a hand in each of the reports. However, some members of our group took more initiative in regulating teamwork while the rest of the group followed suit.

7.4.4 Meeting objectives

Initially, we wanted to develop a program that separates music sources from vocals via a website. Our plan was to use a neural network model. In our first meetings, we spent some time getting samples from music files to send to the neural network. In later meetings, most of the time of our project, we gathered and decided on deficiencies and requirements of the neural network implementation part.

7.5 New Knowledge Acquired and Applied

First of all we learnt what music is. We are not talking about our daily music that we listen to. We mean music file organization and presentation. How to manipulate those files and use as we wish was our second achievement. Then we wanted to learn from published papers. We learnt how to reach, read and analyze scientific papers. After getting information from those papers. We started to find a dataset that is core to our project. We did some research and after some meetings, we decided to use the MUSDB18 dataset to train our neural network. We studied Python as a programming language. Our code is based on Python. Finally, we found some resources to study neural network. We checked a couple online lectures. As we mentioned before we used U-Net architecture with modifications of filter size, strides and image size but we had to adapt to our project.

4. Conclusion and Future Work

In fact, the main aim of this project was to separate music files with an easy to use interface. We used neural network architecture to do this. We needed to train the system with as much data as possible to get desirable results. In order to do this we got samples from music files. We trained our neural network with those samples but due to high CPU requirements, we are not able to train with the whole dataset. However, we are sending partial samples.

As a future work, we want to train our system with more data if we can get higher quality computers. Our primary future work is to improve our system and results. Secondly, we planned to have an easy to use interface for users. However, because of time constriction, we could not complete this mission. Lastly, we will improve our system and website according to feedback that we take via the website.

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
6. 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/.
7. 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.
8. 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.