TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

MINOR PROJECT MID-TERM REPORT

ON

AI-CAPELLA: VOCAL ISOLATION AND SEPARATION USING CNN MODELS

SUBMITTED BY:

NISCHAL PANTHI(PUL077BCT053)

SUBHAM SHRESTHA (PUL077BCT082)

UTSAV MANANDHAR (PUL077BCT093)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

Feburary, 2024

Acknowledgments

First and foremost,We would like to express our heartfelt gratitude to everyone who helped us write this report. We wish to extend our utmost gratitude to, IC Chair, Asst. Prof. Bibha Sthapit and Asst.Prof. Santosh Giri, who guided and supervised for the writing of this proposal.

We would like to express our sincerest gratitude to Institute of Engineering, Tribhuvan University for the inclusion of minor project in the course of Bachelor's in Computer Engineering. We are also thankful to Department of Electronics and Computer Engi- neering at Pulchowk Campus for their guidance and support throughout this project. Their valuable insights and feedback were instrumental in helping us to complete this report to the best of our abilities. Finally, we would like to acknowledge the support and encouragement of our family and friends, who have always believed in us and supported us in some tough decision along the way. Thank you all for your contributions to this project.

Sincerely,

Subham Shrestha(077BCT082)

Utsav Manandhar (077BCT093)

Nishchal Panthi(077BCT053)

Abstract

This document details the progress report of our 6th semester minor project on isolation of vocals from audio tracks using Artificial Intelligence. The purpose of this report is to give a brief overview of the project along with its original objectives, and how those objectives have been achieved or changed as of now. The report outlines our experimental setup that is being used currently. The report also contains a detailed list of our progress so far, what we still have to do, as well as what problems we face and expect to face and how we plan to tackle them.

Contents

List of Figures

List of Abbreviations

CNN	Convolutional Neural Networks
AI/ML	Artificial Intelligence and Machine Learning
Demucs	Deep Extractor for Music Sources
ADC	Analog to Digital Converter
STFT	Short Time Fourier Transform
MFCC	Mel-Frequency Cepstral Coefficient
2D	2 Dimension
ReLU	Rectified Linear Unit
UI	User Interface
RAM	Random Access Memory

1. Introduction

1.1 Background

In the ever-evolving landscape of artificial intelligence and machine learning, the intersection of technology and creativity has opened up new possibilities for innovation. Our proposal introduces "Ai-capella: a Voice Isolation and Separation Project," an ambitious project using AI/ML technologies, specifically Convolutional Neural Networks (CNN), to automate the process of separating vocals from mixed audio recordings, primarily songs.

Ai-capella focuses on addressing a long-standing challenge within the music industry i.e automatic vocal isolation. Composers, vocal coaches, and musicians often grapple with the tedious and time-consuming task of manually dissecting different frequencies within a track to isolate vocals. Our project seeks to revolutionize this process by creating a user-friendly application that seamlessly separates vocal components from mixed audio, requiring minimal effort from the user.

1.2 Problem statements

In the field of audio processing, the manual extraction of vocals from mixed audio recordings poses a difficult challenge for users. The current methodology requires depth analysis and separation of different frequencies within a song, resulting in a time-consuming and labor-intensive process. This challenge hinders workflows, and decreasing efficiency.

To address this issue, our project, Ai-capella, aims to develop a solution utilizing artificial intelligence and machine learning technologies. By automating the process of vocal isolation through Convolutional Neural Networks (CNNs), we seek to empower users with a user-friendly application that significantly reduces the effort and time required to extract vocals from mixed audio recordings. The problem is not limited to professional industry, but it expands to other region including a casual listener or audio editor.

1.3 Objectives

- Design and implement a CNN-based architecture capable of learning and distinguishing vocal components from mixed audio signals
- Create a user-friendly application that allows users to upload mixed audio for vocal isolation and separation.
- Investigate and integrate data sets to improve the model performance

1.4 Scope

The purpose of this project is to decompose a music audio signal into its vocal and backing track components. Once decomposed, clean vocal or backing signals are useful for other MIR tasks such as singer identification, lyric transcription, and karaoke application[1]. The scope includes developing a user-friendly interface, implementing real-time processing capabilities, integrating diverse datasets for improved model performance, and deploying the application on a reliable platform. This project can be applicable to various users from casual to professional users, who are keen to the field of audio and songs. The best application can be by professional who can use it to analyze vocals or it can be used to generate and decompose any sort of songs without tedious hassle.

2. Literature Review

The system deals with two major fields of study: Audio Processing and Artificial Intelligence. The field of Audio Signal Processing is an extensively studied field in computer science. While the concept of artificial intelligence and machine learning is less developed, numerous advancements have been made in the past decade that have caused a rise in applications utilising AI.

2.1 Related work

Several notable projects and studies in the field of audio processing and source separation are:

1. Spleeter by Deezer[2]: Deezer's Spleeter is a pre-trained deep learning model designed for source separation, including vocal extraction. It has been widely recognized for its effectiveness in isolating vocals and accompaniment from music tracks.

Figure 2.1: Spleeter AI

2. Demucs (Deep Extractor for Music Sources) by Facebook AI[3]: Demucs is a deep learning-based system developed by Facebook AI that demonstrates exceptional performance in source separation, particularly in isolating vocals from music recordings. Its architecture aligns with the goals of AI-capella.

3. Open-Unmix[4]: Open-Unmix is an open-source initiative that provides pre-trained models for music source separation, allowing users to extract vocals or other instrument tracks from mixed audio recordings. It serves as a valuable resource for understanding and implementing similar methodologies.

2.2 Related theory

Sound is a mechanical wave that propagates through a medium, typically air. It is created by a vibrating source that produces compressions and rarefactions in the surrounding air particles. These pressure variations travel in wave-like patterns, carrying energy from the source to the surrounding environment. Raw sound is an analog signal and unfit for mathematical treatment. Hence, they are first converted to digital signals using an Analog to Digital Converter(ADC).

Digital signals are discrete and represented as sequences of numbers. Each number corresponds to the amplitude of the signal at a specific point in time. These sequences can be manipulated mathematically. Digital signals are represented by key numerical features that are suitable for mathematical treatment for machine learning.

2.2.1 Frequency content

Ordinary sound is comprised of numerous frequencies of varying amplitudes superimposed on one another. By transforming a given digital signal from it's time domain to it's frequency domain, the individual frequency content of the sound can be determined and studied.

The **Fourier Transform** is a mathematical tool used in signal processing a signal to be represented in terms of its constituent frequencies, providing valuable information about the different sinusoidal components that make up the signal.

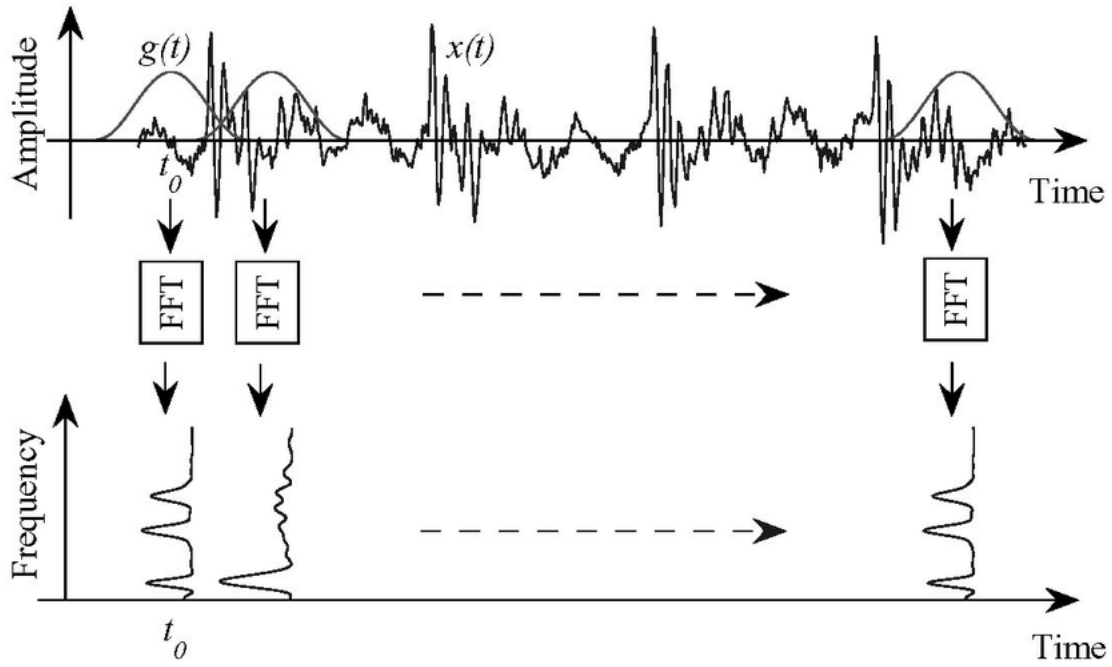

Figure 2.2: Short Time Fourier Transform

In practice, the **Short Time Fourier Transform (STFT)** proves useful for our purpose of analysing short discrete sections of an audio file.

2.2.2 Spectrogram

A spectrogram is a visual representation of the frequency content of a signal as it varies with time. It is a three-dimensional plot, where the x-axis represents time, the y-axis represents frequency, and the color intensity represents the magnitude or power of the frequencies at a given point in time. Spectrograms help in identifying various features of a signal, such as dominant frequencies, harmonics, and changes in frequency over time.

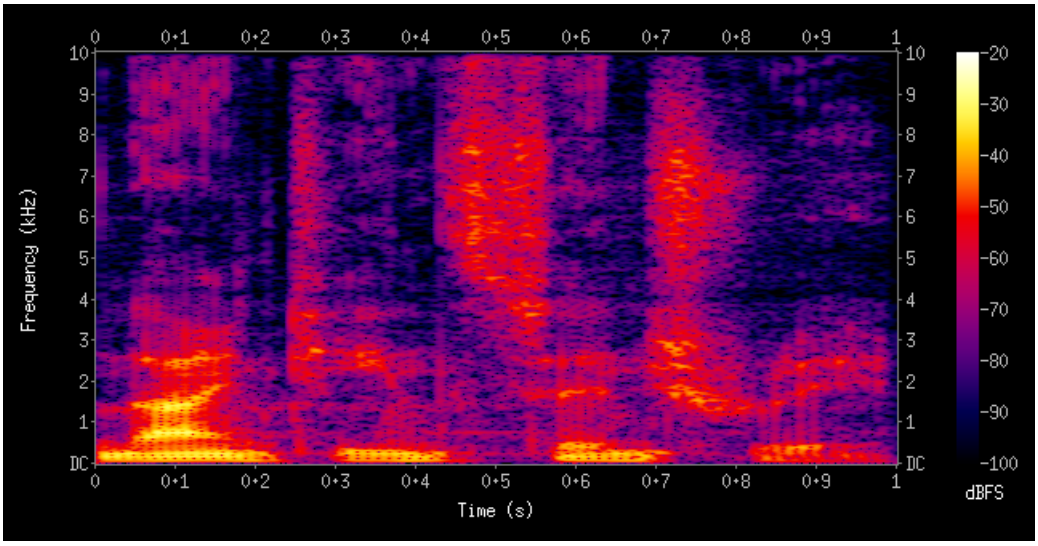

Figure 2.3: Spectrogram of the spoken words "nineteenth century"

2.2.3 U-Net

U-Nets [1] are a very popular Convolutional Neural Network(CNN) architecture for music source separation systems. The popular source separation system Spleeter by Deezer uses this network architecture[2].

U-Nets input a spectrogram and perform a series of 2D convolutions, each of which producing an encoding of a smaller and smaller representation of the input. The small representation at the center is then scaled back up by decoding with the same number of 2D deconvolutional layers (sometimes called transpose convolution), each of which corresponds to the shape of one of the convolutional encoding layers. Each of the encoding layers is concatenated to the corresponding decoding layers. Because the U-Net is convolutional, it must process a spectrogram that has a fixed shape. In other words, an audio signal must be broken up into spectrograms with the same number of time and frequency dimensions.

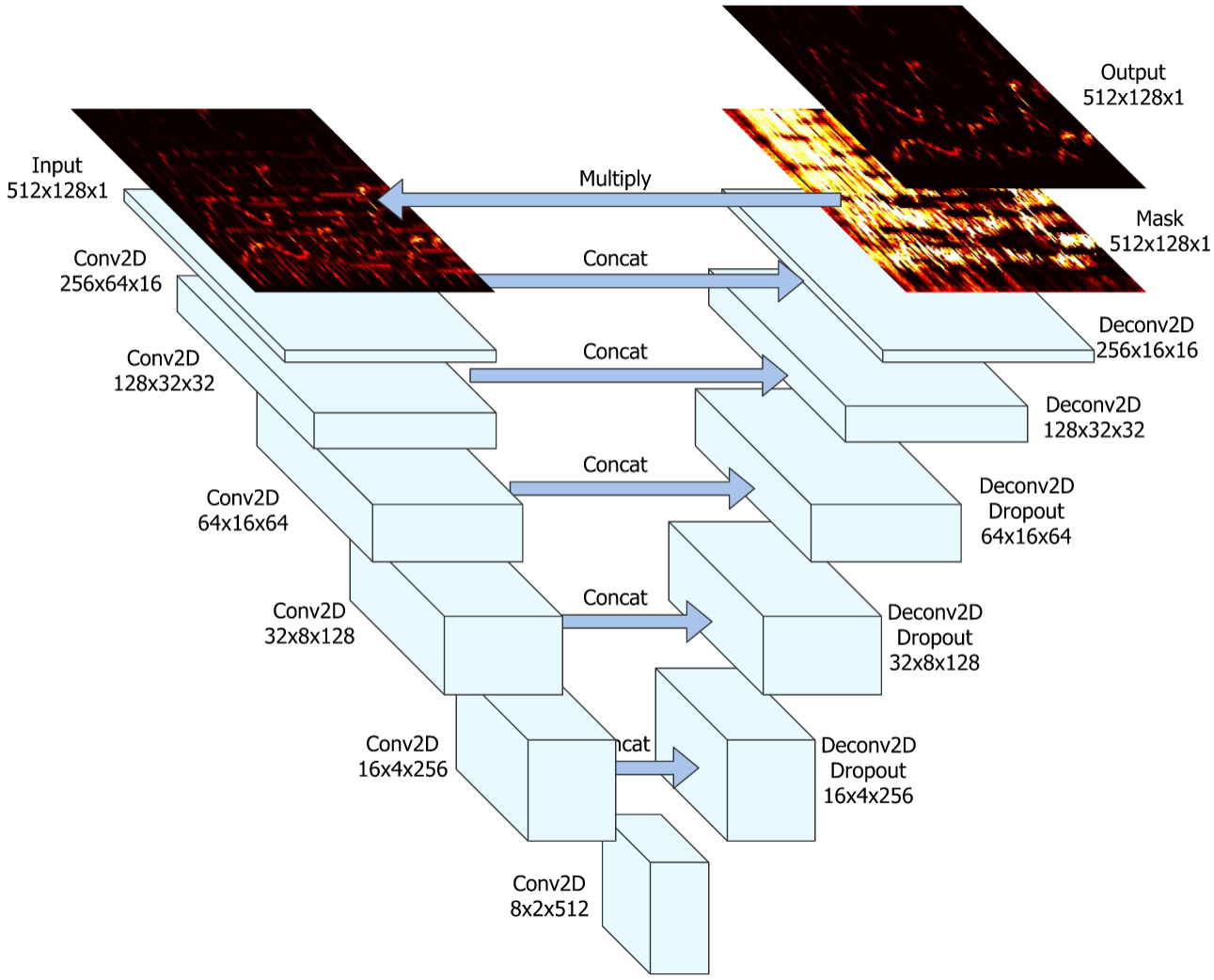

Figure 2.4: The U-Net Architecture

3. Methodology

3.0.1 Feasibility Analysis and Requirements Study

For the project we have used readily available tools and packages in the python programming language that are specialised in audio processing. One key library we are using is Librosa, a library that allows us to load and convert audio files into data that is suitable for mathematical treatment and easy to work with using artificial intelligence. It's readily available tools for feature extraction of various qualities of audio made it an ideal candidate for our application.

As stated in our proposal we found Convolution Neural Networks (CNNs) to be the best model for our purpose and hence chose to proceed with it. CNNs(U-Net Architecture) are primarily used to handle images and audio data is quite similar to graphical data.

The requirements for this project are outlined as follows:

Functional Requirements

- Vocal isolation: The software should be able to convert the given audio track and return the vocal isolated track into listenable form.
- User Interface (UI) : The app must feature an intuitive and user-friendly interface for seamless navigation. It should require minimal technical knowledge from the side of the user and include convenient song listening tools.
- Sharing and Saving: Enable users to download and share their vocal isolated tracks for future use.

Non Functional requirements

- Performance: The application should process and return the isolated track in a reasonable amount of time.
- Efficiency : The output track must have little to no traces of the instruments of the original track
- Accuracy : The returned audio track must be free from any artefacts that were not present in the original audio.

3.0.2 Data Collection

The required dataset can be classified into 3 categories,namely training set,testing set and validation set as our project is based on supervised machine learning. For project,dataset is available at [5],which is a dataset of 100 full lengths music tracks of different styles along with their isolated drums, bass, vocals and others stems. It is not uncommon for artists to release instrumental versions of tracks along with the original mix[1].

We have download and stored the raw audio files locally and converted the data into appropriate npz files which can be accessed via google drive.

3.0.3 Implementation

We have implemented the project in the large part by closely following the paper[1].The model is implemented using U-Net Architecture of Convolutional Neural Network. We plan to test the model with any readily available song clip and validate its output with available open-source application like Spleeter[2] on basis of accuracy and efficiency.

3.0.4 Tools and Libraries

The final product has utilised various pre-built libraries and programs for the purpose of audio feature extraction and manipulation as well as tools to implement machine learning algorithms. We have used utility libraries such as NumPY for manipulation of matrices, matplotlib for data visualisation and pandas for data storing and formatting.

Librosa provides a wide range of tools for tasks such as feature extraction, signal processing, and music information retrieval. TensorFlow is an open-source machine learning framework that provides a comprehensive set of tools, libraries, and community resources to support various machine learning tasks, including classification, regression, clustering, and more.

4. Experimental Setup

In this project, we trained the model on a dataset comprising numerical representations of songs, with inputs representing various song features and outputs corresponding to vocal elements. We initially train the model on a smaller subset of the data obtained from [5]. By limiting our dataset to 5000 samples from 111563 samples, it allowed us to create a prototype with available RAM space and create an overall outline for the system pipeline. The experimental setup is not concerned with the UI and is interfaced primarily through the source code.

Each training input-output pair consists of segments lasting about 9 seconds in each chunk.Feature extraction is done through the aforementioned music libraries. Subsequently, the extracted features of this data are passed through the neural network for training, enabling the model to learn patterns and relationships between the input features and corresponding vocal outputs. This approach facilitates the creation of a model capable of predicting vocal elements in songs based on their numerical representations. For the experimental setup, we have utilized a machine with the following specifications:

- Hardware: A virtual machine Configuration,Google Colab, with a minimum of 8GB RAM and a multicore processor to handle the computational demands of training a Convolutional Neural Network.

- Software: The project is carried out using Python , TensorFlow for implementing the CNN model, and Jupyter Notebooks for audio management and Google Colab for training.

This setup ensures a robust environment for conducting experiments, training the model, and evaluating its performance.

5. System design

The user engages with the system solely through the user interface, initiating the process by uploading a song of their choice in the .wav file format. Following the upload, feature extraction is conducted on the song, transforming it into chunk spectograms compatible with the model. The pre-processing module employed for this task takes a standard song as input, breaks it into manageable chunks (512x128 size), and conducts feature extraction on these segments.

The model is trained and tested using the audio dataset. The trained model processes this data, generating output that is later reintegrated into a complete song. Subsequently, the numerical features are converted back into audio. The user, in turn, receives a downloadable audio file featuring the isolated vocals, providing a tailored and personalized rendition of their chosen song. In short,the architecture consist of the following components:

- Input Module: This module allows users to upload mixed audio recordings for processing. It will be the User Interface for the application(web application).
- Preprocessing Module: Responsible for preparing spectogram of the audio data for input into the CNN model, including feature extraction using Librosa.
- CNN Model(U-Net Architecture): The heart of the system, trained to recognize and separate vocal components from mixed audio.
- Output Module: Provides users with the desired output file.

A visual representation of the system architecture is depicted in Figure 5.1.

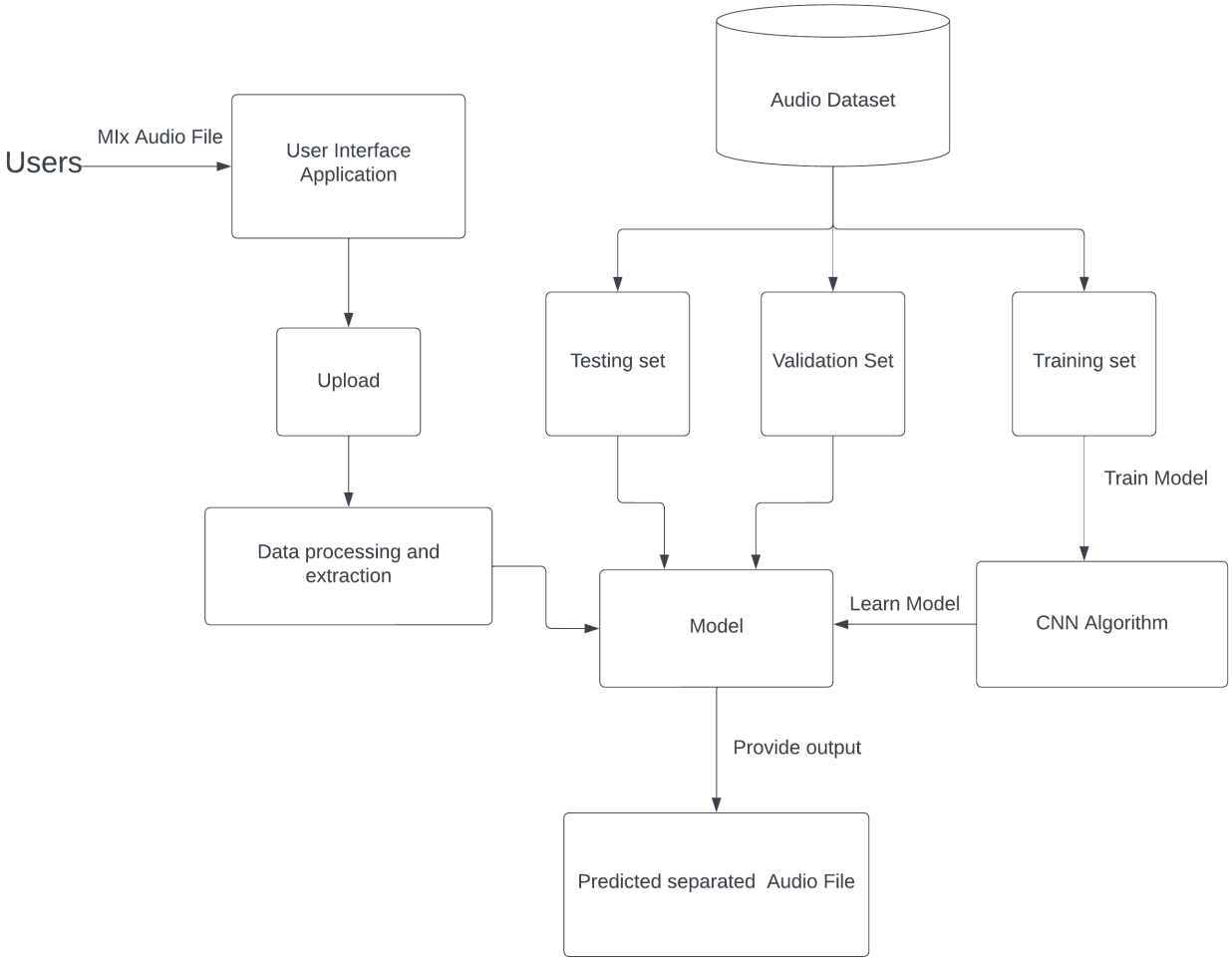

Figure 5.1: System Architecture Diagram

6. Task Completed

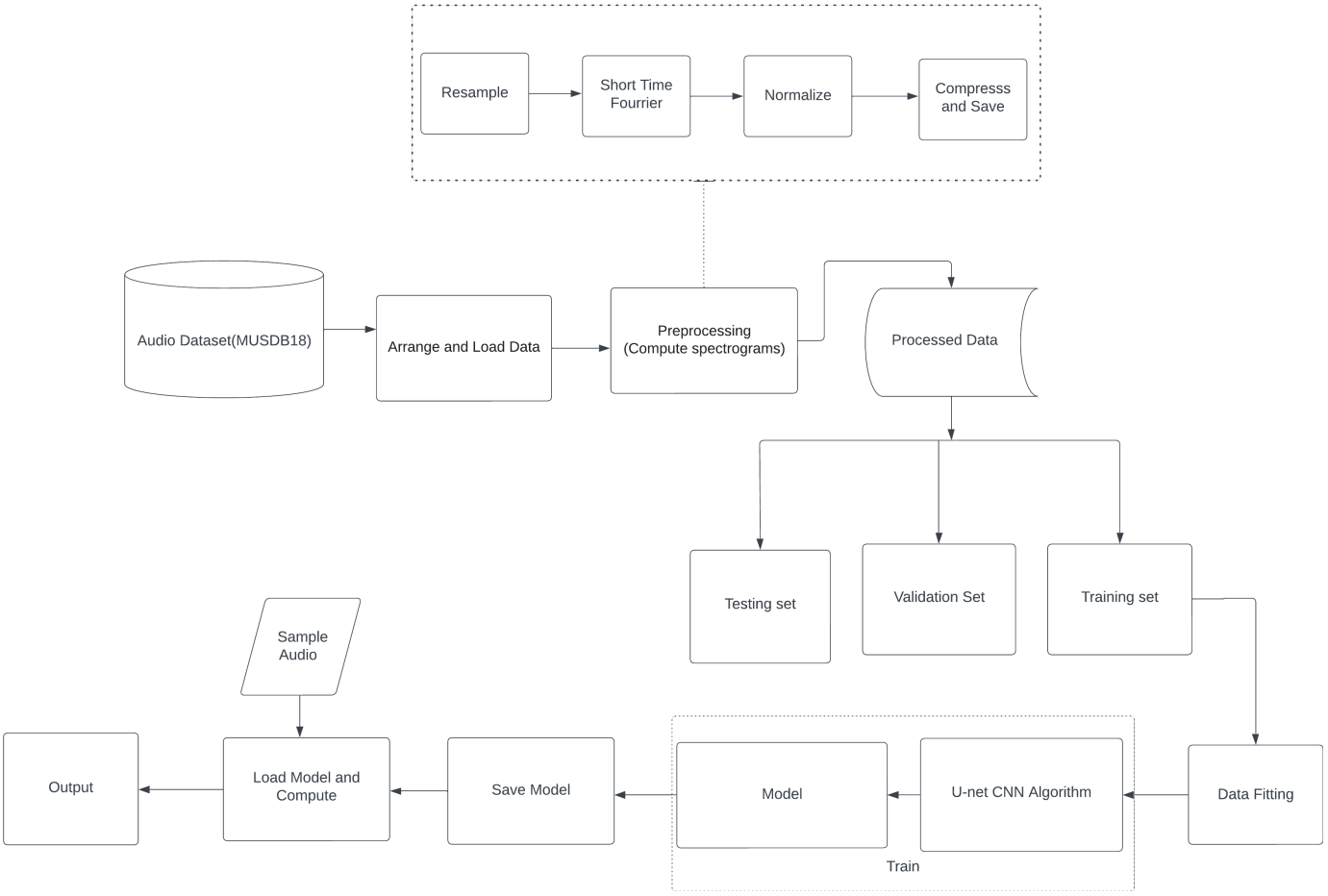

Figure 6.1: Task Block Diagram

- Feasibility Study: Explored available tools and packages for audio processing, selected Librosa for its feature extraction capabilities, and determined Convolutional Neural Networks (CNNs), specifically the U-Net architecture, as the suitable model for the project.

- Requirements Analysis: Identified functional and non-functional requirements, including vocal isolation, user interface (UI) development, sharing and saving features, performance, efficiency, and accuracy. Data Collection: Acquired a dataset consisting of 100 full-length music tracks along with their isolated drums, bass, vocals, and other stems from [source].

- Implementation: Designed and implemented the project using Python, based on the U-Net architecture for CNNs. Utilized libraries such as Librosa for audio feature extraction, TensorFlow for machine learning, NumPy for matrix manipulation, Matplotlib for data visualization.

  – Data Collection And Arrangement : We have downloaded data obtained from [5]and stored the raw audio files locally. Then we arranged the data into a proper file system .We converted the data into suitable npz files by using python scripting which can be accessed via google drive for use in Google Colab.

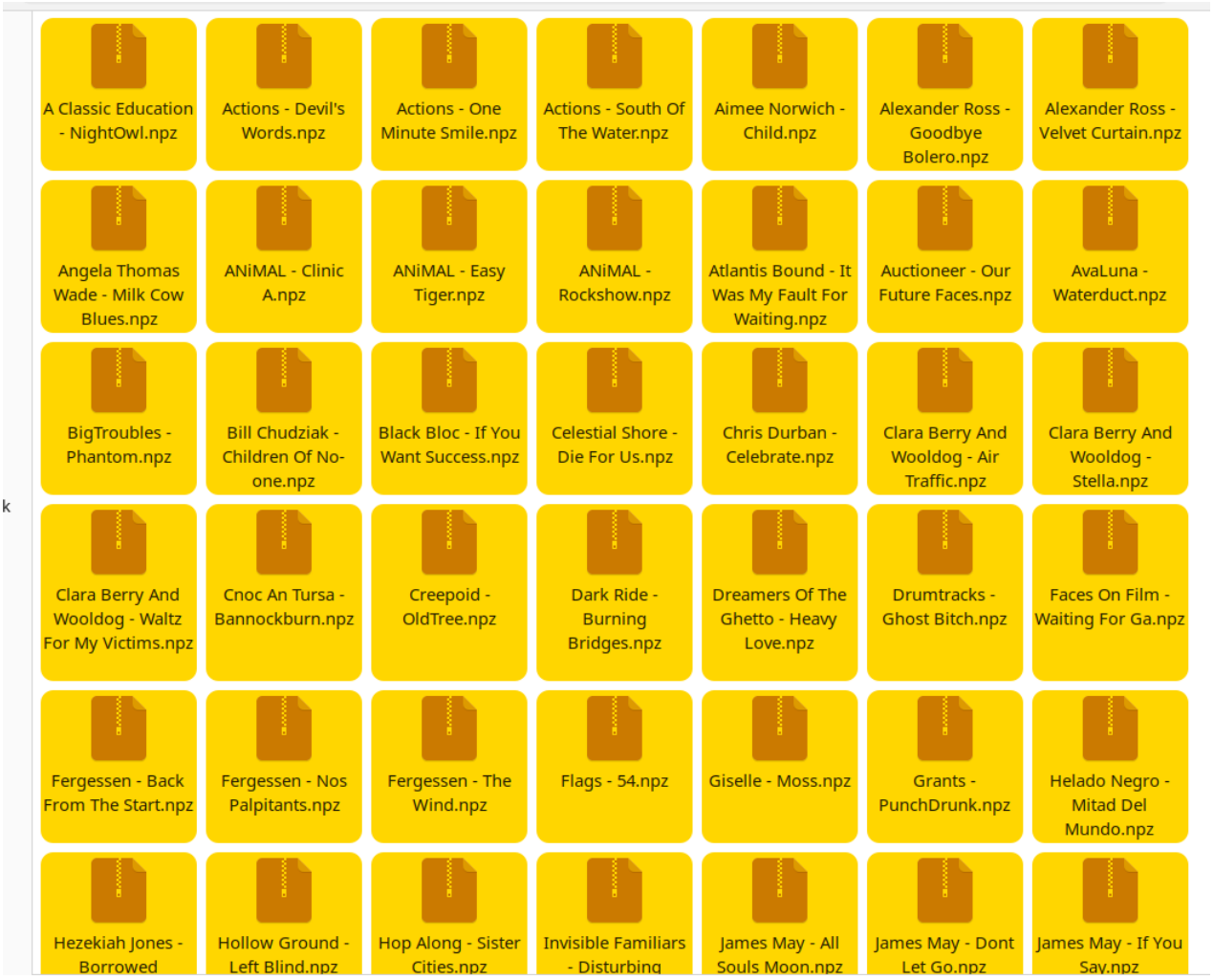

Figure 6.2: Data stored in local machine

– Data Pre-Processing : The raw data is in the form of audio files, specifically .in the .wav format. We extracted the audio features of these files using librosa and first down sampled then,applied Short Term Fourier Transform on this data to obtain spectrograms which are the sample data we will use to train our model.Also,the spectrtograms are normalized for easier computation.

```
def SaveSpectrogram(y_mix, y_inst,y_vocal, filename, orig_sr=44100, target_sr=SR) :
    y_mix = librosa.resample(y_mix,orig_sr=orig_sr,target_sr=target_sr)
    y_vocal = librosa.resample(y_vocal,orig_sr=orig_sr,target_sr=target_sr)
    y_inst = librosa.resample(y_inst,orig_sr=orig_sr,target_sr=target_sr)

    S_mix = np.abs(librosa.stft(y_mix,n_fft=window_size,hop_length=hop_length)).astype(np.float32)
    S_inst = np.abs(librosa.stft(y_inst,n_fft=window_size,hop_length=hop_length)).astype(np.float32)
    S_vocal = np.abs(librosa.stft(y_vocal,n_fft=window_size,hop_length=hop_length)).astype(np.float32)

    norm = S_mix.max()
    S_mix /= norm
    S_inst /= norm
    S_vocal /= norm

    np.savez(os.path.join('./Spectrogram',filename+'.npz'),mix=S_mix,inst=S_inst ,vocal=S_vocal)
```

Figure 6.3: Code for preprocessing of audio

– Data Fitting : The songs in the data are of varying lengths but our model only takes in inputs of fixed length. So , we tailored the data by dividing it into appropriately sized chunks of length 128. A single chunk with the given sample rate and frame size equates to roughly an 8.1 second audio file.

```
x_test = []
num_chunks = test_song_mag.shape[1]//128
for i in range(num_chunks):
    chunk = test_song_mag[:512, i * 128 : (i + 1) * 128, np.newaxis]
    x_test.append(chunk)

x_test = np.asarray(x_test, dtype = np.float32)
y_test = trained_model.predict(x_test)
print(y_test.max())
```

Figure 6.4: Code snippet for audio fitiing

Model Design : We defined the CNN model by following a U-net architecture[1]. The initial hyper-parameters are given standard values and will be optimized in the future.

Model Training : For training the model we used a small fraction of the processed data and passed it into the model. The model obtained was then saved for future testing. Predictions can be made using this model by passing appropriately processed audio files.

Audio Post Processing : The predictions made by the model are in the same format as the inputs i.e. arrays that constitute magnitudes of the spectrogram. In order to reconstruct the original audio have treated it with its phases and apply Inverse Short Term Fourier Transform (ISTFT) to obtain the output in time domain. Finally this output is converted back into a .wav file using the Soundfile python library

7. Task Remaining

We have completed almost major components and functionality in our project, some of the remaining tasks are:

- User Interface Development: We aim to develop interface for the application, allowing users to upload audio tracks, initiate the vocal isolation process, and download/share the isolated tracks.For the UI part , we are planning to develop a basic web application to host our model and provide the desired output.

- Evaluation:We are conducting thorough evaluation of the system's performance and efficiency in isolating vocals from audio tracks. This evaluation will involve testing the system with a variety of audio inputs and assessing the quality of the isolated vocal tracks from other available models.

- Performance Optimization: We are optimizing the performance of the system to ensure timely processing and minimal resource utilization.Proper tuning of hyperparameter is continuing currently in order to optimize the output .

References

[1] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde. *Singing voice separation with deep U-Net convolutional networks*, pages 746–749. Suzhou, China, October 2017.

[2] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154, 2020. Deezer Research.

[3] Alexandre Défossez. Hybrid spectrogram and waveform source separation. In *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[4] F.R. Stoter, S. Uhlich, A. Liutkus, and Y. Mitsufuji. Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*, 2019.

[5] Antoine Liutkus, Fabian-Robert Stter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In Petr Tichavský, Massoud Babaie-Zadeh, Olivier J.J. Michel, and Nadège Thirion-Moreau, editors, *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, pages 323–332, Cham, 2017. Springer International Publishing.