Web application for animal audio noise reduction using the ORCA-CLEAN model

Autor:

Néstor Rafael Calvo Ariza

Informe de trabajo de grado para optar por el título de: Ingeniero Electrónico

Tutor

Prof. Dr.-Ing. Juan Rafael Orozco Arroyave

Universidad de Antioquia
Facultad de Ingeniería
Departamento de Ingeniería Electrónica y Telecomunicaciones
Medellín
2021

Cita	(Calvo Ariza, 2021)
Referencia **Estilo APA 7 (2020)**	Calvo Ariza, N. R. (2021). *Web application for animal audio noise reduction using the ORCA-CLEAN model* Trabajo de grado. Universidad de Antioquia, Medellín UdeA.

Grupo de Investigación GITA UdeA.

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.
Decano/Director: Jesús Francisco Vargas Bonilla.
Jefe departamento: Augusto Enrique Salazar Jiménez.

Content table

Abstract

Audio analysis is a topic of study that has gained momentum in the last decade, the growing information as well as the improvement in computational power has allowed more and more academic and industrial sectors to perform studies of audio signals which previously went unnoticed. With this type of analysis certain drawbacks arise, one of them is that in many cases the recording conditions will not be optimal to obtain a sample with "clean" information, because external factors affect or introduce noise to the sample. As a solution to this problem, multiple algorithms have been developed for audio cleaning, some of them require manual work that can be exhausting depending on the size and quantity of audios, and on the other hand there are techniques that use predictive models created with Machine or Deep Learning to perform the cleaning process in an automated way. Although these last techniques have solved the problem of doing this work manually, many of them are not user-friendly and require the user to have knowledge of the model created in order to make changes and experiment at ease, thus reducing the number of people who can make use of this technology.

In this work a web application was created which allows to make use of a Deep Learning model called ORCA-CLEAN [23], created to perform audio cleaning for whales. and couple it in such a way that the user can perform audio cleaning without having knowledge of the model and just making use of his mouse and keyboard. The user can select multiple regions in the audio spectrogram in order to apply different types of parameters and make comparisons, as well as listen to the resulting audio(s) after applying the cleaning process. Finally, the user can download a zip folder containing images of the spectrograms of the regions before and after cleaning, as well as the cleaned audio(s).

Introduction

The animal world hides many secrets, and we, as human beings, are constantly advancing to decipher them, understand animals is one of these enigmas. For the past years, researchers have been focusing their attention on the way that animals use sounds to communicate between them. The majority of the animals use sounds that are incomprehensible to the human ear but for them, these sounds can be used as a method to detect or warn for a predator or prey, to alert others to stay away from their territory, to mate [1], among others. Advancements in machine learning [2],[3] and audio analysis [4],[5] applied in animal sounds facilitate the study of long hours of passive acoustic recordings of animals in the wild. Within the advantages of passive recordings are sounds and animal behavior unaffected by the human's presence; nevertheless, this leads to recordings with a high amount of environmental and animal sounds combined due to the absence of control.

This noise variety requires the use of different denoising techniques. In the past, methods based on low-pass [6] and high-pass [7] filters have been successfully applied in several studies of bird classification. Due to the high performance of deep learning techniques, recent studies use deep autoencoders [8], Deep Neural Networks (DNNs) [9][10], a combination of both [11] and Recurrent Neural Networks (RNNs) [12] to clean and enhance audio signals. However, the main concerns with these models are the low transferability and the need for clean/denoised ground-truth samples for the training stage, which in most of the cases is unavailable. ORCA-CLEAN is the first deep denoising approach that does not require clean ground-truth samples [13]. Adding that to the robustness and generalizability of the model makes it an excellent choice to be used for denoising sounds from different animal species. This study aims to create a web application. It will be allocated on a cloud instance in AWS. The ORCA-CLEAN model and the web application will allow other researchers to select frequency ranges and denoise multiple audios. Additionally, the web application will let the user segment the recordings to apply different types of denoising and adapt other features, giving a more feasible way of denoising their recordings with less programming knowledge.

The creation of a web application would provide people, researchers, and companies interested in the study of the animal kingdom with a tool that brings together concepts and strategies for audio segmentation, which is a process that can

sometimes be long and tedious. On the other hand, engineering and technological developments must be focused on facilitating and creating bridges between different disciplines. With this web application, we seek to apply knowledge in audio analysis, machine learning, and software development to build an application that allows audio segmentation of different species of animals for future research.

1. Objectives

1.1. General Objective

To develop a web application that includes the ORCA-CLEAN model for audio-denoising by using Django, React, and Apache tools to allow users without any programming background to clean multiple audios. In addition, the development must grant other research areas like biology an easy-to-use tool to create and interact with different scenarios.

1.2. Specific Objective

- To parameterize the ORCA-CLEAN model to allow users to tweak the different parameters before performing the audio-denoising, creating different scenarios and responses depending on the selected values.
- To design and implement the user interface and all the modules responsible for the different functionalities of the web application such as visualization, audio segmentation, parametrization, audio-denoising, among others, based on the scenarios and responses previously defined and using Django, React, and Apache tools.
- To validate the web application by performing unit tests on the different modules and presenting the web application to experts from the Pattern Recognition Lab in the Friedrich-Alexander-Universität Erlangen-Nürnberg to obtain feedback from them and fix possible problems.

2. Theorical Background

2.1. Backend

For every web application, there is a backend that focuses on everything “behind the scene”, the backend is the part of the application in charge of the continuous communication between the server and the application database. Its main focus is the logical part of the application, which contains all the features and basic functionalities that the application needs for its use, without the backend the applications will be an empty case where you cannot interact with anything. The flow of data that comes from the frontend is analyzed by the backend, which is capable of understanding the request, who sent it, what is needed to fulfill the request, where can be found the files and how to send the data back to the frontend so the user can see it. There are multiple options for a backend, each one of them with its benefits and disadvantages, for this work the framework Django will be used as a backend, Django [20] uses Python which will help us with a better and faster connection of the machine learning models.

2.2. HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems and it has been on the World-Wide Web since 1990 [26]. HTTP not only defines the structure of the message to allow the communication between two points but also give the user methods to perform basic petitions, the most used ones are:

- GET: This method is used to retrieve data and nothing else, if the request is pointing to a data-producing process the response has to be the produced data.
- POST: This method is used as a request to process the data that is sent in the POST request, this data most of the time comes in form of files, JSON, or form data.
- PUT: This method requests the server to store the information in the Uniform Resource Identifier (URI) that it’s sending in the request.

These methods are applied to a resource that is indicated through a URI. Each request contains headers, this consists of a name that is followed by a colon (":"), and then the value [26], these headers bring information such as the type of message, the length of the message, authorization credentials, among others.

After the request is processed the entity must respond with the status of the request to maintain communication, this response comes with a status code that identifies the type of response, these codes can be classified by the first number: Information 1xx, Successful 2xx, Redirection 3xx, Client Error 4xxx, Server Error 5xx.

2.3. API

The Application Programming Interface (API) is a set of rules that state how two devices must communicate, APIs primarily were used for the exchange of information between two or more programs [27], later on, this idea was used on the Internet to access data using the HTTP protocol and now provides the user ways to efficiently use programming queries to send and retrieve information.

The same way as HTTP, API uses URI as the identifiers, Figure 1 shows how a URI is structured and what are the main components:

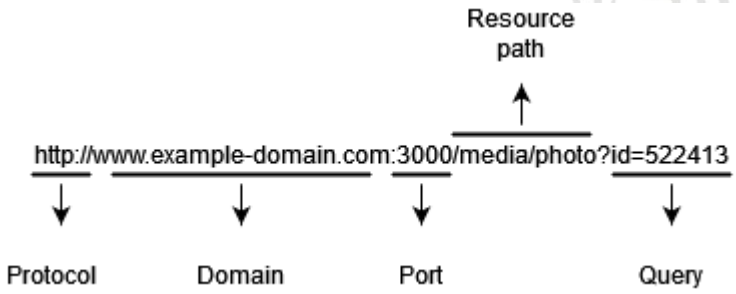

Figure 1. URI Structure

APIs provide great flexibility due to the fact that the data is not tied to the request methods, APIs also allow us a new layer of security by being the intermediary between the client and the server

2.4. Frontend

The frontend is the part of the web application in charge of displaying the information delivered by the backend to the user, it also fulfills the function of collecting information such as words, clicks, files, etc. The frontend is the visual part of an application and although it is an application more focused on the functional part than the visual part, it is important to have a frontend not only to facilitate the use of the application but to allow users to navigate through it correctly.

To create a good user experience, it is important to choose an appropriate a good architecture, this will not only facilitate future scalability but also will be faster and more stable, different frameworks and libraries are used in the front end, for this work we will be using React Js, which are JavaScript libraries for building user interfaces, JavaScript will allow us a better interaction with the user by creating a real-time communication. React Js it's a very popular and well-known library for the creation of front-end among developers, the ongoing growth in the number of framework and web applications that uses React Js for the front-end shows the stability and versatility of the libraries.

2.5. Time-Frequency representation

Since the purpose of this work is to make modifications to audios, it must be taken into account that these have a representation in both time and frequency. The collective use of both representations allows multiple types of analysis and modifications to the audio, for this reason, it is necessary to have a way to perform audio analysis in both the time domain and frequency domain. The Discrete Fourier Transform (DFT), as its name indicates, allows us to perform time series transforms of a signal the same way as a Fourier integral transform or a Fourier series transform. It is a mapping operation that allows the representation of a time signal in the frequency domain.

The discrete Fourier transform uses discrete signals, which are characterized by belonging to a domain n of integers. Discrete signals typically come from the discretization of an analog signal. The signal is transformed into a sequence of numbers once discretized, allowing its mathematical manipulation [30]. This is

important to mention because to perform a computational time-frequency analysis the process signals require to be a discrete signal.

When sampling the analog signal, to ensure a correct representation of the signal, the samples must be selected with a frequency at last two times of the maximum frequency of the waveform, also known as Sampling Theorem [17] [18], this is important to ensure enough points of the signals in order to do a correct reverse transformation without losing to much information.

$$\underline{X}_k = \sum_{m=0}^{N-1} x(m)\, W^{mk}\ , \qquad k = 0, \dots, N-1,$$

$$W = e^{-i2\pi/N}, i = \sqrt{-1}$$

Equation 1. Discrete Fourier Transform. [19]

The analysis of a signal in the frequency domain eases the convolutional operation, which is reduced to a multiplication of the signals in the frequency domain. This makes the Discrete Fourier Transform useful in spectrum analysis, digital signal filters, data compression, among other uses.

The DFT has a computational complexity of O (N ^{2}), meaning that for long signals the number of CPU's operations increase rapidly, to reduce these computational times the Fast Fourier Transform (FFT) algorithm is used to calculate the DFT in a more efficient way. This algorithm reduces the computational complexity of the operation from O (N ^{2}) to O (N log N) [28].

The FFT reduce the operation time by reusing values calculated previously in some parts of the operation, to have a better understanding on how the FFT reuse previous values first we have to express the Equation 1 as matrixes as shown in Equation 2.

$$[X(k)] = [W^{nk}][x_0(n)]$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix}$$

Equation 2. Matrix representation

After replacing $W^0 = 1$, and factoring the W matrix:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^2 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix}$$

Equation 3. Matrix factoring

With this new matrix

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix}$$

$$with \quad \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^2 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix}$$

Equation 4. Matrix separation

This factoring method to calculate the FFT is called the Cooley-Turkey FFT [29] algorithm, this algorithm can factor $N\ x\ N$ matrix in a way that the new factored matrix has the property of minimizing the number of operations, in the past equations this reduction is accomplished by the introduction of the multiple zero terms in the operation. This algorithm can also be seen graphically in Figure 2.

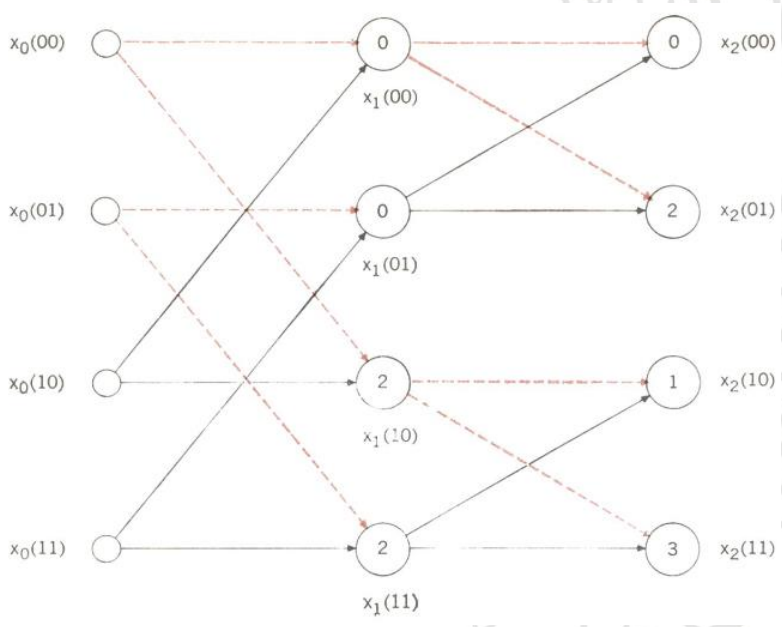

Figure 2. Graphical representation of matrix factoring. [a]

[a] Taken from Brigham, E. O., & Morrow, R. E. (1967). The fast Fourier transform. IEEE Spectrum, 4(12), 63–70. doi:10.1109/mspec.1967.5217220

2.6. Noise Reduction

Noise reduction is a technique used in order to reduce and control de amount of noise in a signal. Noise is considered a sound that does not contain any useful information but alters the important signal [21]. The definition of noise varies in every research field, in this work we will be focusing on Additive Noise, which is the noise that comes from different sources, either natural or artificial. Noise can introduce changes to the signals, that is why it is important to study it and find techniques to reduce it, these changes can affect the listener's perception or a machine's processing of the signal.

The randomness of a noise signal makes it hard to tackle as a problem, this is why even today it is still not clear how to describe this problem with a unique mathematical model, so rather than doing that, researchers had come with different approaches depending on the context, for audio enhancement, new analysis has been made in the past few years but for this work, we will be using an approach that was originally developed for image known as Noise2Noise [22] (N2N), which is transferred to the field of audio signals and extended by using an automatic machine-generated binary mask as extra, this approach was used in previous works [23], where the model was trained with various distributions of synthetic noises in order to create a noisy signal, additionally, real-world noise such as boat sounds, water noise, among others is used to intensify and stimulate other sounds. For this work, we will be integrating this model into the web application to perform sound enhancement in the audios requested by the user.

2.7. Segmentation

Segmentation is a process that aims to divide a digital audio signal into different segments, each of them contains information of the audio, this process helps to improve the analysis and the preprocess of the data, removing useless parts of the signal [15]. Most of the time, audio signals contain background noise that can deteriorate the quality of the audio and the information contained in it, in addition to this, there are segments of silence in the audio which do not provide any information. After the segmentation, we will have parts of the audio that are silent, that contains

information and other parts that could have information but are too noisy, for the last one noise reduction techniques are applied in order to clean this audio signals.

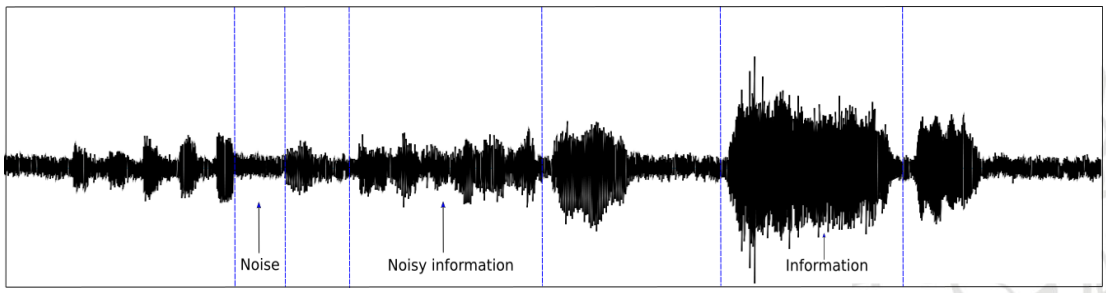

Figure 3. Segmentation process

Segmentation is performed cutting the signal into overlapping frames of the audio as shown in Figure 3, a window function of N frames is applied to tapper audio signal and to ensure its continuity, the window function has a time-shift between 10% and 20% of the window length [16], this implies an overlapping of the windows, this will avoid irregularities in the borders. In this work, the web application will allow the user a variety of options to perform segmentations of the audio in order to experiment and generate different audio signals for their posterior analysis.

2.8. Convolutional Neural Network

Convolutional neural networks are a type of network which has been growing in popularity in recent years as a very good network to perform image analysis or computer vision since it obtained such good results in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012. CNNs have shown great potential for all kinds of fields whenever image analysis is employed due to their structure [31]. Convolutional networks are composed of multiple blocks which maintain a similar structure to each other, these blocks have a convolutional layer, a pooling layer and can even have a fully connected layer.

2.8.1. Convolutional layer

The convolution is a linear operation which is used to perform the product of two signals or functions. In the case of a CNN, the product of the input signal with a kernel which is constantly being recalculated in each iteration is performed. The convolution

performs the element-wise multiplication generating a tensor at the output called feature map. Due to the nature of the element-to-element product, the output of a convolution is a feature map with dimensions smaller than the input dimensions. In many occasions it is desired to keep the same dimensions, for this a process called zero padding is performed. The zero padding consists of adding 0 to the input, so that the network can calculate the convolution and keep the input dimensions as shown in Figure 4.

$$s(t) = (x * w)(t) = \sum_{m=-\infty}^{\infty} x(a)\, \mathrm{w(t-a)}$$

Equation 5. 1D convolution

In Equation 5 we can observe the mathematical equation of the convolution for a discrete system in the case of a 1D data array. The first argument is the input signal while the second argument is the kernel that will produce the feature map. In the 2D case a 2D kernel must be used and the equation would be Equation 6, where I is the (m, n)-dimensional image and K is the new 2D kernel. [32]

$$S(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n)$$

Equation 6. 2D Image convolution with a kernel

It is important to emphasize that CNN does not require the use of manually extracted features but it is a network that needs to be trained with a large number of samples in order to estimate the best parameters.

2.8.2. Pooling layer

The pooling layer is a layer that allows a reduction of the dimensions of the resulting feature map after a convolutional layer, this layer is used in order to introduce distortion and reduction in the number of parameters to be learned by the network. This layer can be considered as a layer that subsamples the input, since a loss of information is generated with the reduction of the dimensions. [32]

The most popular operation in the pooling layer is max pooling, which consists of choosing a filter of size N x N (N will depend on the size chosen for the pooling layer) and choosing the maximum value within this filter, thus reducing the dimension of the feature map by a factor of N.

Another widely used pooling operation is global average pooling, this operation is usually performed before a fully connected layer and consists of taking the feature map or parts of the feature map, and reducing it to a 1x1 vector by averaging the values. This operation when performed before the fully connected layer allows to reduce the number of parameters to be learned by the network and also allows to make use of input images of different sizes. [31]

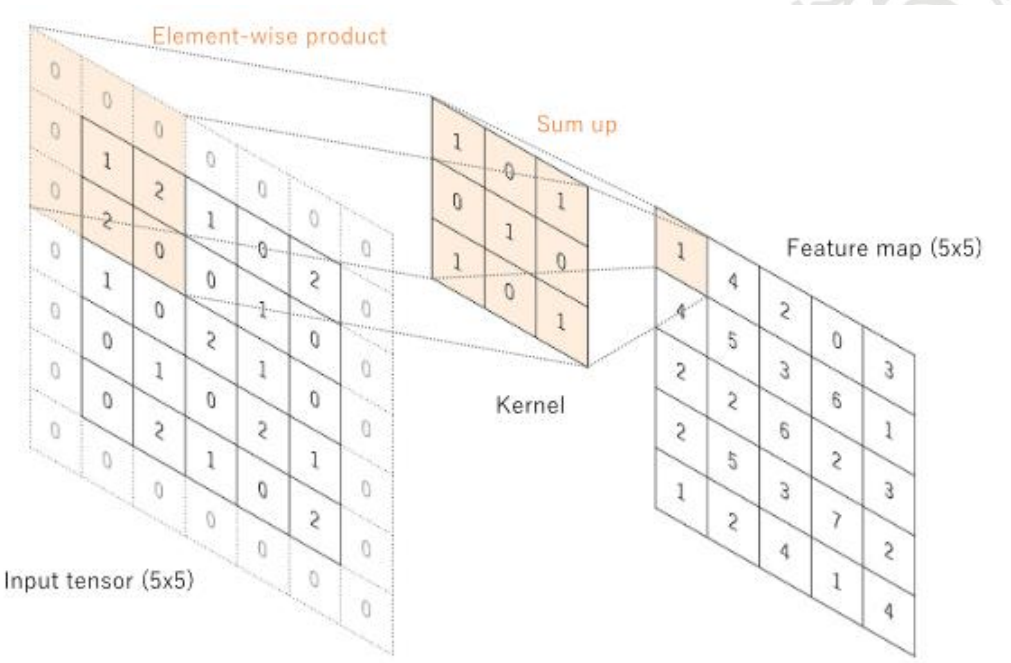

Figure 4. Convolutional neural network with zero padding. [b]

[b] Taken from Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, *9*(4), 611-629.

2.9. ORCA-CLEAN

ORCA-CLEAN is the first work presented as a fully-automated deep denoising approach for bioacoustics that doesn't require any clean ground-truth [23]. Due to the lack of clean bioacoustics datasets, ORCA-CLEAN implements a new approach that was originally intended to be used as image restoration known as Noise2Noise [22]. ORCA-CLEAN transfer the idea of Noise2Noise to the bioacoustics by introducing noise to the spectrogram image. The network can be trained to restore the image to the original one even if the original one is a nosy image. As an addition, the model also enhances the sounds of the orcas using a machine-generated binary mask, this binary mask

works as a filter that allows the network to increase or highlight those segments where the sound belongs to a killer whale and reduce any other type of sound.

Before feeding the data into the model it must be preprocessed, ORCA-CLEAN contains multiple layers of preprocessing, the first stage of preprocessing consists of converting the audio to mono audio and resampling it to a frequency rate of 44.1kHZ. Subsequently, the spectrograms are generated, which will be used by the network both for training and for its prediction process, the creation of these spectrograms is performed using a Short Time Fourier Transform (STFT). Among the parameters that are needed are the size of the FFT which is set by default at 100ms, and the hop-value for the window which is set by default at 10ms, the type of window to be used is a parameter which will be selected by the user when the audio parameterization process is being performed.

Lastly, an intensity within the range of -6dB to +3dB and a pitch within the range of 0.5 to 1.5 is randomly set. Once the audio is processed, other preprocessing operations are performed such as histogram equalizations, maximum/median filtering, morphological operations, among others.

ORCA-CLEAN during the process of noise addition to the spectrograms created uses real-world underwater noise as noise signals such as boat noise, water noise, among others. Subsequently each spectrogram is compressed in the frequency domain, this compressor has default values, but parameterizations were made so that the user can choose the type of compression, the maximum frequency and the minimum frequency. All the different noise additions that are added to the spectrogram can be seen in Figure 5 [23].

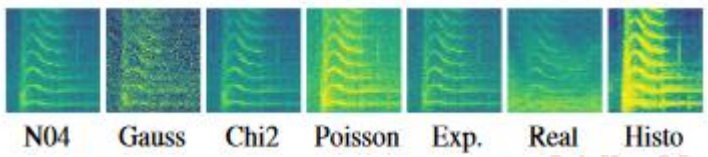

Figure 5. Noise types introduced to the spectrograms (N04 is the original signal)

At last, these spectrograms are fed through a network which is based on the U-Net architecture [32] (see Figure 6), the previously modified spectrogram are introduced to the network and it is expected to have the original file at the output, in this way there is a ground-true value to compare the performance of the network and make modifications in order to improve its prediction capacity.

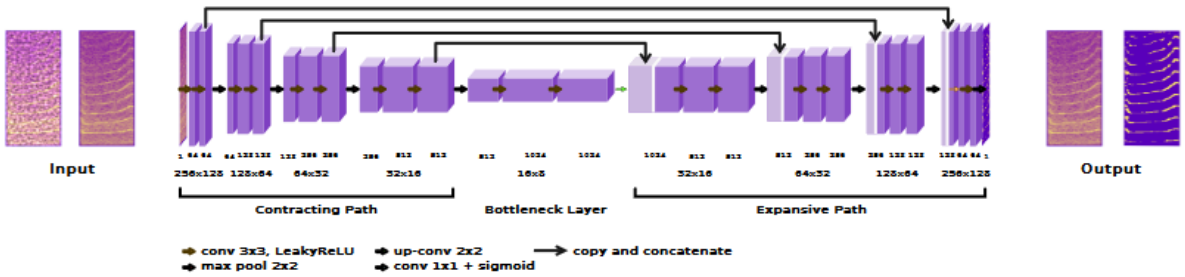

Figure 6 ORCA-CLEAN - Deep denoising network architecture [c]

[c] Taken from Bergler, C., Schmitt, M., Maier, A., Smeele, S., Barth, V., & Nöth, E. (2020). ORCA-CLEAN: A Deep Denoising Toolkit for Killer Whale Communication. Proc. Interspeech 2020, 1136-1140.

The network initially has multiple convolutional layers followed by polling layers which perform a reduction of the input size, then feeding through a bottleneck layer and at last, use of a set of layers for the expansion of the dimensions until having the same dimensions of the input as shown in Figure 6.

3. Methodology

This work is divided into three main parts which in the end are merged to form the final application. For this application, we built the FrontEnd in JavaScript more specifically using the React Js framework, on the other hand, the BackEnd of the application was developed with Python more specifically using the Django framework, finally the ORCA-CLEAN model that will be used to perform the cleaning of audios was developed in the Python language, the choice of backend language was made to be able to use in a better way the model with the backend.

Each of these parts performs relevant tasks during the whole process and their separation grants the possibility to identify and isolate errors in a more efficient way, at the same time separating the FrontEnd development from the BackEnd development allows to perform unit tests in a simpler way.

3.1. FrontEnd

The FrontEnd was created using components which are small pieces of code independent of each other that allow performing different functions, some of them are already in the React main package because they are widely used but React Js allows the creation of your components, this way of structuring the page allows:

isolate problems or bugs in the application and identify more effectively errors in the application because each component is independent of each other. On the other hand, some pages group multiple components and display them in order for the user to make use of them.

The pages used for the application as well as its components are listed below:

3.1.1. Home

The Home page or the start page is the first thing that the user observes when opening the web application, this page contains a component in the navigation bar that allows the user to access multiple points of the application, contains another component for the description of the functionality of the application and how its use facilitates the cleaning and segmentation of audio, also shows examples of spectrogram which allow a visual comparison of how an audio segment looks before and after the cleaning process. The images shown are brought from the BackEnd through a GET request.

3.1.2. Denoise

The denoise page contains a component that allows the user to upload the audio file that it's going to be analyzed, the component allows to choose an audio file (.wav) that is on the user's computer and upload it for further analysis by clicking on the Upload button, in case a file is not found a warning message will appear on the screen, if the file is valid the Next button is activated and allows the user to continue with the process. In this component, the user is also given a random unique identification number that allows keeping an identification of the user even when closing the page, in this way no repeated information is stored in the server.

3.1.3. Parameters

The parameters page is a simple page that allows the user to interact with the audio that has just been uploaded to the server, the main idea of this page is to create a simple interface making use of multiple components that allow the user:

- **Playing the audio:** The user has a Play/Pause button which allows the user to play and pause the audio and interact with it at any time.

- **Segment the audio:** By clicking and dragging the user can create multiple regions in the same audio, these regions or segments will allow the user to make modifications in a selective and different way for each of the regions, the user can create as many regions as desired, the only condition is to create regions with a size greater than 25ms, this in order to be able to create windows.

- **Audio parameterization:** Each audio segment contains a start and end value and the user can choose the minimum and maximum frequency of operation as well as the frequency filtering technique and the type of window to be used during the denoising process.

The user can create and delete as many regions as desired, at least one region needs to be created to do the analyze, the data chosen by the user are stored in the localStorage and then sent to the server for audio processing.

3.1.4. Process

The process page is considered the final page of the process, at the top of this page, there is a loading component that gives the user a visual indication that the BackEnd is working on processing the audio(s). Therefore, to avoid losing communication with the FrontEnd, no more requests are allowed until the BackEnd finishes processing the audio. Once the audios are processed, the BackEnd completes the request made by the frontend and allows the frontend to make the next request which consists of bringing the modified audios as well as the images of the spectrograms for each one of them before and after the cleaning process.

Finally, the user has the option to download all the modifications as well as the audios in a compressed file for later use. After downloading the file, the user can perform the same process again for another audio.

All the previous pages make calls to the backend to request or store information in a constant way, for it they make use of the HTTP Requests which allow making GET, POST, and other requests to an address.

3.2. Backend

The BackEnd was developed in Django and was created in such a way that it works as an API which performs an action and returns a value or a group of values depending on the request made, the multiple endpoints that were created in the API are:

- **/upload:** Endpoint that allows through a GET to diagnose the server and its responsiveness, on the other hand it also allows to perform a POST in order to create a model in the database depending on the values that come from the request (this model stores the audio in a specific path).

- **/parameters:** Endpoint that is called when the front end encounters the "parameters" page, this endpoint receives GET requests and allows to return an audio that is stored on the server in the user's folder, this returned audio is the one that is mounted on the component that allows audio playback.

- **process:** Endpoint that allows both GET and POST requests, each of them performing different tasks. This endpoint is used in the process page to obtain the information (GET request) of the JSON created which contains the information of the segmentation made by the user, in the same way if you make use of the POST request this endpoint will perform the creation of the folders and the JSON file if they are not already on the server, in case they are already there it stores the user's information in its respective JSON.

- **/processPDF:** This last endpoint allows to make requests using the token or each user to obtain the PDFs and the processed audios, this endpoint returns the information which will be later shown to the user in the final part.

3.3. ORCA-CLEAN Model

A parameterization of the ORCA-CLEAN model was made in order to be coupled with the backend and thus allow predictions, this parameterization allows Django to use the model as a function and thus create threads to run the function without interrupting the ability to handle calls from different users, so the application can continue to receive requests from multiple users.

The parameterization also allows using the model with parameters different from the default parameters, allowing the user to choose different types of values. The parameters selected for the model are:

- **start:** Start point in seconds (s) for the region or the audio that is going to be processed.

- **stop:** Stop point in seconds (s) for the region or the audio that is going to be processed.

- **sequence_len:** Contains the length of the sequence or region to be analyzed, this value is calculated depending on the values of the "start "and "stop" parameters chosen by the user.

- **min_frequency**: Allows the user to select the value for the minimal frequency that is going to be analyzed, this later on restrict the audio frequency bands and the minimal value in the Y-axis for the spectrogram. By default, this value is set to 200Hz.

- **max_frequency:** The same way as the “min_frequency” parameter, the max_frequency allows the user to select the maximum frequency to analyze, this value restrict the maximum value in the Y-axis for the spectrogram. By default, this value is set to 18000Hz

- **freq_compression:** Allows the user to choose between Linear, MEL or MFCC method to compress the frequency of the audio. By default, the value is set to Linear

- **window_type**: Allows the user to choose between Hamm, Blackman, Hamming and Kaiser windows. By default, the value is set to Hamm

Other parameters such as sample rate and number of frequency bins are present in the model but are set by default to avoid issues in the audio processing section.

4. Results

As a result of this work a web application was built, this web application is hosted in AWS with the following link: http://audiodenoisingfront.s3-website-us-east-1.amazonaws.com/ and the application allows the user to upload audios, segment them and later on perform denoising without having any previous knowledge.

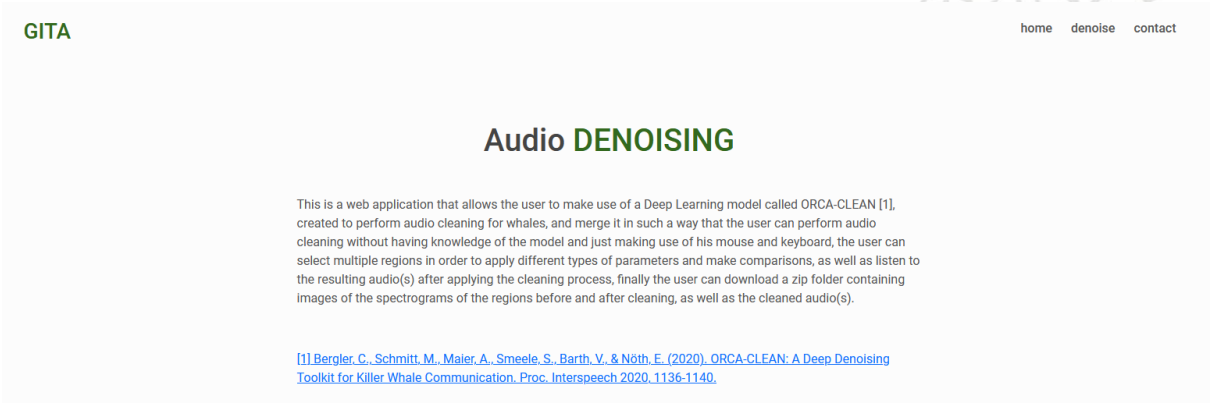

Figure 7. Home page and navbar

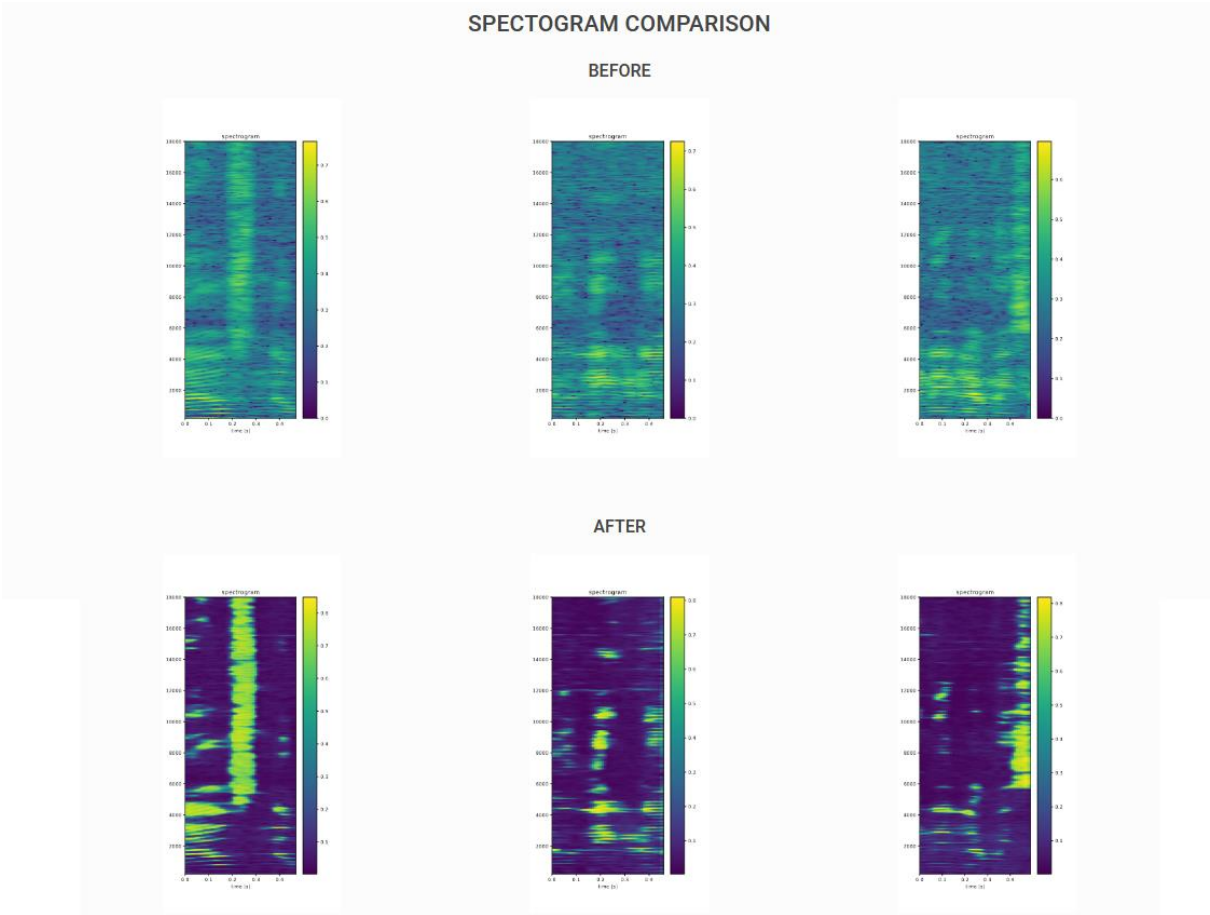

Figure 8. Spectrogram comparison

Figure 7 show the “Home” page for the web application, a small text that describes the main purpose of the application and the cite to the Deep Learning model can be found below the title, on the top we can found the name of the research group and the main buttons to navigate the application.

The Home button takes the user back to the main page, the contact button opens an email window that can be used to send emails towards me in case of having doubts with the application and the denoise button can be a click when the user is ready to start the denoising process.

Bellow the main text we can find a graphical example of denoising as shown in Figure 8, and how different the spectrograms are when we clean noisy audio. These same images can be downloaded at the end when the user finishes the process.

We can notice that in the first three images, there are green colors all over the image, this is noise due to the constant intensity through the frequency spectrum, on the other hand, we can notice that after the denoising process the audio-only has the relevant information, we can notice this because most of the image now is dark blue, this is the color for the lowest intensity (close to 0).

When the user clicks the denoise button on the navbar we go to the next page shown in Figure 9.

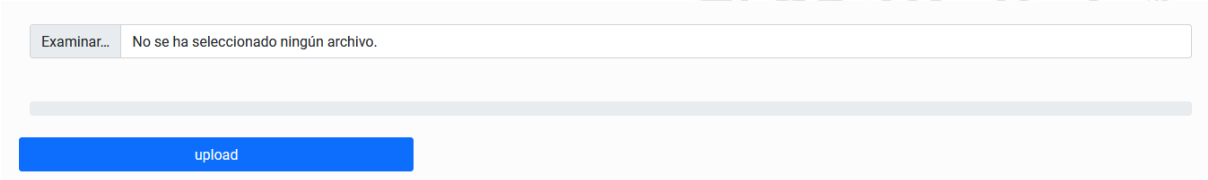

Figure 9. Component to upload audio

On this page the user can found the component to upload the audio, this component allows only audio files to avoid and check that there is a valid file before uploading, if the user selects a file and uploads the component activates the “Next” button and grant the user permission to go forward in the process as shown in Figure 10.

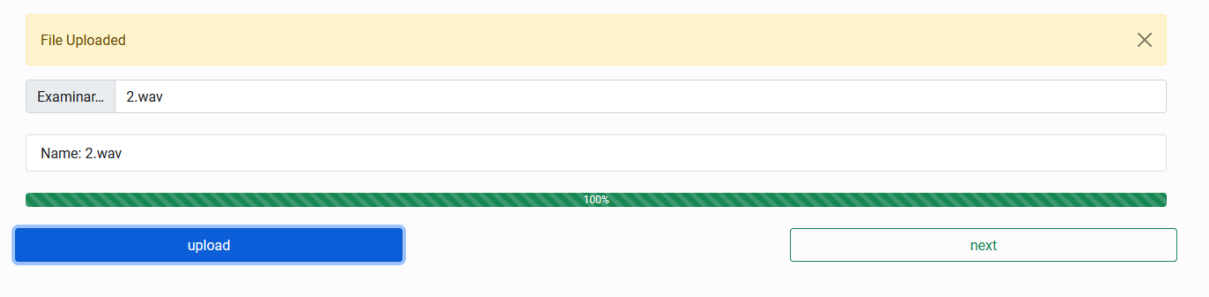

Figure 10. Component after file uploaded

After the user clicks the “Next” button, the web application goes to the “Parameters” page, in this page the user will have a preview of the recently uploaded audio, the user can play and pause the audio, and the most important part, the user will be able to segment or create regions by selecting parts of the audio as shown in Figure 11.

Figure 11. Audio segmentation

If the user clicks next the audio will be denoise only in that region selected and the parameters will be the default one, in case that the user will like to change parameters, the user can click the region and a form will appear where the user can change one or all parameters if wanted as shown in Figure 12.

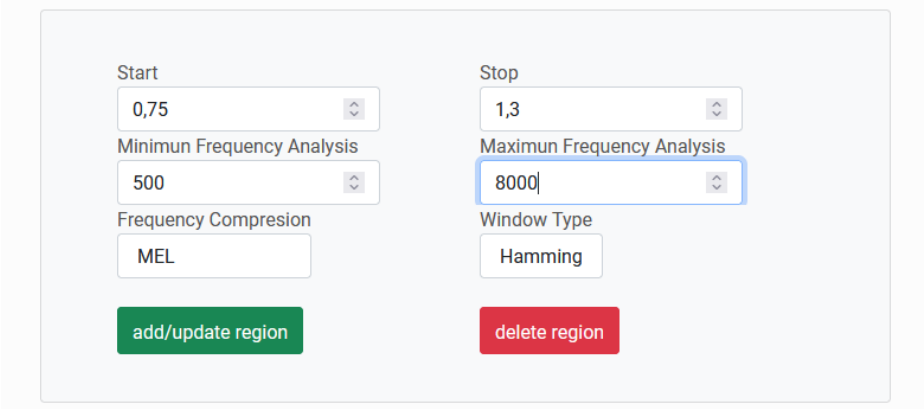

Figure 12. Form to tweak denoising parameters

As we can see, minimum frequency, maximum frequency, frequency compression, and window type are different from the default values, once the values are set the user can click the button “add/update region” to store these new values in the region if the user is not satisfied or the user wants to remove the region the user can click the button “delete region”.

In case everything is set the button Next in Figure 10 will send the regions and the form data through a POST request to the API, the API will preprocess the audio and then will run the prediction algorithm that uses the model to create the cleaned audios and the PDFs with the spectrogram.

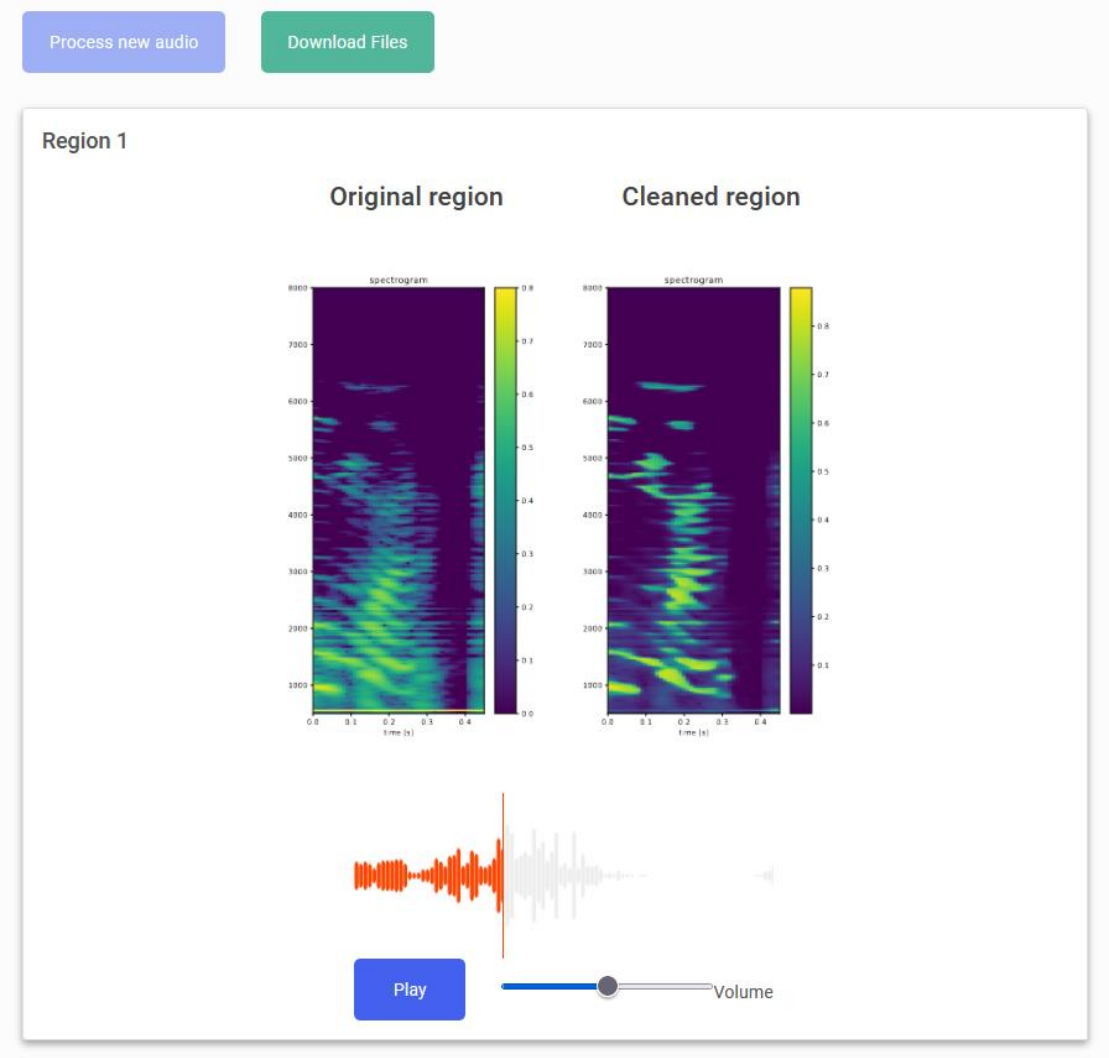

Figure 13. Results visualization

The FrontEnd will wait until the audios are denoised to maintain the same request and ensure that all the processes are finished correctly, while the BackEnd is cleaning the

audios the FrontEnd will display a loading icon so the user can have a visual confirmation that the audios are being processed.

When the audios are finished the web application will show the button to download the files (audio and PDFs) as a ZIP file, will also give a visualization card of the original and the cleaned region, and will give the user the possibility to listen to the cleaned audio before processing new audio as shown in Figure 13.

The user can have as many regions as the user needs, the only main constrain is that the region must have a length higher than $25\ ms$ to be processed.

The application not only focus on the denoising of the audio but it's also important to take into account the performance of the app, as we can notice in Figure 14 the application requires a total of $0.07\ s$ without cache to load the web page, this is due to the possibility of reacting to load components at its information at different moments, the application first load the static parts and when the data is received from the BackEnd this is store in the components and the page is re-render.

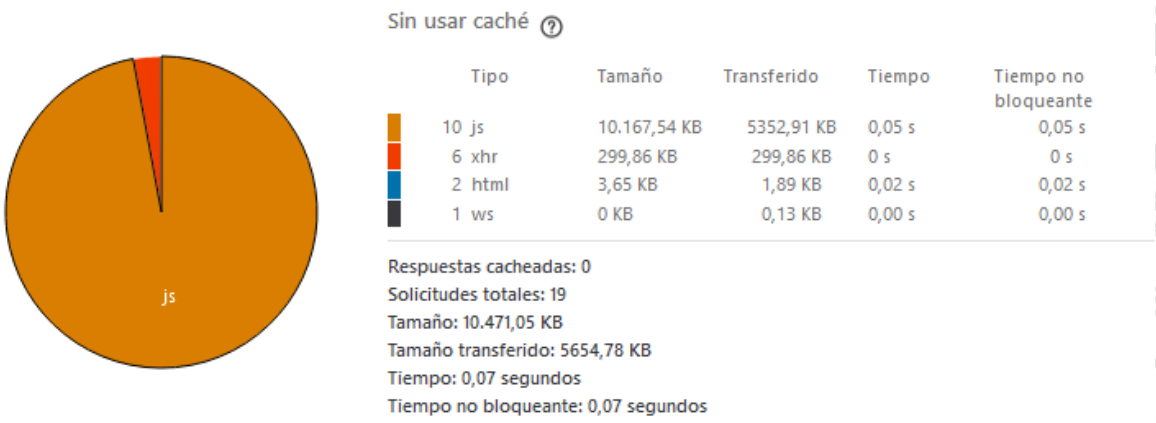

Figure 14. Performance of app without cache

5. Conclusion

In this work we present the whole process of creating a web application that can be used for audio cleaning and segmentation, to solve the problem we use ORCA-CLEAN, which is a deep denoising network designed for denoising of killer whale (Orcinus Orca) underwater recordings, not requiring any clean ground-truth samples, to improve the interpretation and analysis of bioacoustics signals by biologists and various machine learning algorithms, even though the model was trained to denoise killer whale, other experiments show the transferability, robustness, and generalization of the model, allowing the use for human speech.

Thanks to Django as a BackEnd language, the creation of the structure for the web application was more efficient, reducing the difficulty to merge the model with the BackEnd since both of them are written in Python, Django also allows the BackEnd to receive and handle multiple petitions from different users without the need to create a subprocess, reducing the difficult and the possibility of mistakes by working with multi-threads programs. Django also allows the possibility to create REST APIs with this we can separate the preprocess and handle of the audio from the sending and receiving information part of the application.

For the user interface the use of React Js allows the creation of a component base structure, reducing loading times (0.07 s without cache) and allowing the possibility of scalability in the future, in case of the response from the back when the audios are being denoised we don't have an estimate because depends on multiple factors such as the number on concurrent petitions, size of the audio, number of regions, size of regions, among others.

As future work is necessary to keep increasing the application usability, new models can be added to the web application to perform the comparison between then, metrics and ranking can be included and thanks to the use of components adding or removing parts of the applications can be done more easily and safely without damaging the structure of the web application. Adding more parameters to the model can be added but this requires a deeper knowledge of the model created to increase the parameters.

6. References

[1] Michelsen, A. (1992). Hearing and sound communication in small animals: evolutionary adaptations to the laws of physics. In *The evolutionary biology of hearing* (pp. 61-77). Springer, New York, NY.

[2] Piczak, K. J. (2015, October). ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia* (pp. 1015-1018).

[3] Fristrup, K. M., & Watkins, W. A. (1993). *Marine Animal Sound Classification*. WOODS HOLE OCEANOGRAPHIC INSTITUTION MA.

[4] Elemans, C. P., Heeck, K., & Muller, M. (2008). Spectrogram analysis of animal sound production. *Bioacoustics*, *18*(2), 183-212.

[5] Staddon, J. E. R., McGeorge, L. W., Bruce, R. A., & Klein, F. F. (1978). A simple method for the rapid analysis of animal sounds. *Zeitschrift für Tierpsychologie*, *48*(3), 306-330.

[6] Bardeli, R., Wolff, D., Kurth, F., Koch, M., Tauchert, K. H., & Frommolt, K. H. (2010). Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, *31*(12), 1524-1534.

[7] Priyadarshani, N., Marsland, S., Castro, I., & Punchihewa, A. (2016). Birdsong denoising using wavelets. *PloS one*, *11*(1), e0146790.

[8] Lu, X., Tsao, Y., Matsuda, S., & Hori, C. (2013, August). Speech enhancement based on deep denoising autoencoder. In *Interspeech* (Vol. 2013, pp. 436-440).

[9] Xu, Y., Du, J., Dai, L. R., & Lee, C. H. (2013). An experimental study on speech enhancement based on deep neural networks. *IEEE Signal processing letters*, *21*(1), 65-68.

[10] Kim, M. (2017, March). Collaborative deep learning for speech enhancement: A run-time model selection method using autoencoders. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 76-80). IEEE.

[11] Mimura, M., Sakai, S., & Kawahara, T. (2016, September). Joint Optimization of Denoising Autoencoder and DNN Acoustic Model Based on Multi-Target Learning for Noisy Speech Recognition. In *Interspeech* (pp. 3803-3807).

[12] Weninger, F., Erdogan, H., Watanabe, S., Vincent, E., Le Roux, J., Hershey, J. R., & Schuller, B. (2015). *Speech Enhancement with LSTM Recurrent Neural Networks and its Application to Noise-Robust ASR. Lecture Notes in Computer Science, 91–99.* doi:10.1007/978-3-319-22482-4_11

[13] Bergler, C., Schmitt, M., Maier, A., Smeele, S., Barth, V., & Nöth, E. (2020). ORCA-CLEAN: A Deep Denoising Toolkit for Killer Whale Communication. *Proc. Interspeech 2020*, 1136-1140.

[14] Yeager, N. J., & McGrath, R. E. (1996). *Web server technology*. Morgan Kaufmann.

[15] Theodorou, T., Mporas, I., & Fakotakis, N. (2014). An overview of automatic audio segmentation. International Journal of Information Technology and Computer Science (IJITCS), 6(11), 1.

[16] Foote, J. (2000, July). Automatic audio segmentation using a measure of audio novelty. In 2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532) (Vol. 1, pp. 452-455). IEEE.

[17] Nyquist, H. (1928). Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, *47*(2), 617-644.

[18] Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, *37*(1), 10-21.

[19] Nussbaumer, H. J. (1981). The fast Fourier transform. In *Fast Fourier Transform and Convolution Algorithms* (pp. 80-111). Springer, Berlin, Heidelberg.

[20] Forcier, J., Bissex, P., & Chun, W. J. (2008). *Python web development with Django*. Addison-Wesley Professional.

[21] Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). *Noise reduction in speech processing* (Vol. 2). Springer Science & Business Media.

[22] Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., & Aila, T. (2018). Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*.

[23] Bergler, C., Schmitt, M., Maier, A., Smeele, S., Barth, V., & Nöth, E. (2020). ORCA-CLEAN: A Deep Denoising Toolkit for Killer Whale Communication. *Proc. Interspeech 2020*, 1136-1140.

[24] Islam, Nazrul & John,. (2014). The Impact of Waiting Time Distributions on QoE of Task-Based Web Browsing Sessions.

[25] Smith, J. O. (2011). *Spectral audio signal processing*. W3K.

[26] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext transfer protocol–HTTP/1.1.

[27] Ofoeda, J., Boateng, R., & Effah, J. (2019). Application programming interface (API) research: A review of the past to inform the future. *International Journal of Enterprise Information Systems (IJEIS)*, *15*(3), 76-95.

[28] Brigham, E. O., & Morrow, R. E. (1967). The fast Fourier transform. IEEE Spectrum, 4(12), 63–70. doi:10.1109/mspec.1967.5217220

[29] Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, *19*(90), 297-301.

[30] Kuo, S. M., & Gan, W. S. (2005). *Digital signal processors: architectures, implementations, and applications*. Prentice Hall.

[31] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, *9*(4), 611-629.

[32] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.