People’s Democratic Republic of Algeria

University of BLIDA 1

Faculty of Sciences

Department of Informatics

MASTER THESIS

Option : Ingénierie des Logiciels

DESIGN OF A WORKBENCH FOR AUDIO ANALYSIS

By:

LAZAR Asma

YAICHE ACHOUR Yousra

Board of jury:

Mrs. BENYAHIA	President
Ms. YEKHLEF Hadjer	Supervisor
Ms. DAOUD	Examiner

2020/2021

Abstract

Acoustic Scene Classification (ASC) is the task to identify audio recordings into one of several predefined acoustic scene classes. There is a huge amount of systems that have been developed to tackle the ASC problem using different machine learning-based techniques. Therefore, building a program that combines multiple techniques has become a necessary need to properly assess the effects of these techniques on the performance of the ASC systems and compare between it in an easy way. Motivated by these needs, we have designed in our work an acoustic scene classification-based workbench that consists of variety of techniques and methods related to this task. The goal of this work is to allow the users to apply the techniques they are desiring to use on a chosen dataset, make changes in the parameters fast, create big tests, and visualize the results of these tests. To accomplish this task, we have created a workbench that is composed of modules consisting various tools for each module. Specifically, we have implemented 5 modules concerning the ASC task stages which are respectively: Data loading and visualization, for importing a Dataset and get an overview of it. Data transformation, for extracting relevant features from the dataset. Data augmentation, for increasing data size. Training, for train the employed Deep Neural Networks on the obtained dataset and evaluate its performance. And lastly Prediction, for evaluating the performance of the trained models on making an accurate class prediction of a chosen audio recording. Thereafter, for executing our workbench to see how it performs, we have used a modified TAU Urban Acoustic Scenes 2019 dataset that consist of 7192 audio recordings with 10 various classes on 5 different cities for each, and tested it on different existing techniques while recording the obtained accuracy on a table of results.

Keywords: Acoustic scene classification, Workbench, Machine learning, Data Augmentation, Deep Neural Network.

Résumé

La classification des scènes acoustiques (ASC) est la tâche d'identifier les enregistrements audio dans l'une des classes de scènes acoustiques prédéfinies. Il existe une énorme quantité de systèmes qui ont été développés pour résoudre le problème ASC en utilisant différentes techniques basées sur l'apprentissage automatique. Par conséquent, la construction d'un programme combinant plusieurs techniques est devenue un besoin nécessaire pour évaluer correctement les effets de ces techniques sur les performances des systèmes ASC et les comparer facilement. Motivés par ces besoins, nous avons conçu dans notre travail un workbench basé sur l'ASC qui consiste en une variété de techniques et de méthodes liées à cette tâche. Le but de ce travail est de permettre aux utilisateurs d'appliquer les techniques qu'ils souhaitent utiliser sur un ensemble de données choisi, de modifier rapidement les paramètres, de créer de gros tests et de visualiser les résultats de ces tests. Pour accomplir cette tâche, nous avons créé un workbench composé de modules consistés de divers outils pour chaque module. Concrètement, nous avons mis en place 5 modules concernant les étapes du cadre ASC qui sont respectivement : Chargement et visualisation des données, pour importer une base de données et en avoir une vue d'ensemble. Transformation des données, pour extraire les caractéristiques pertinentes de la base de données. Augmentation des données, pour augmenter la taille des données. Training pour former les réseaux de neurones profonds utilisés sur l'ensemble de données obtenu et évaluer ses performances. Et enfin Prédiction, pour évaluer les performances des modèles formés pour faire une prédiction de classe précise d'un enregistrement audio choisi. Par la suite, pour exécuter notre workbench pour voir comment il fonctionne, nous avons utilisé un ensemble de données TAU Urban Acoustic Scenes 2019 modifié qui se compose de 7192 enregistrements audio avec 10 classes différentes sur 5 villes différentes pour chacun, et l'avons testé sur différentes techniques existantes lors de l'enregistrement de l'exactitude obtenue sur un tableau de résultats.

Mots-clés : Classification de scènes acoustiques, Workbench, Apprentissage automatique, Augmentation des données, Réseaux de neurones profonds.

الملخص

تصنيف المشهد الصوتي (ASC) هو مهمة تحديد التسجيلات الصوتية في واحدة من عدة فئات للمشاهد الصوتية المحددة مسبقًا. هناك عدد هائل من الأنظمة التي تم تطويرها لمعالجة مشكلة ASC باستخدام تقنيات مختلفة قائمة على التعلم الآلي. لذلك ، أصبح بناء برنامج يجمع بين تقنيات متعددة حاجة ضرورية لتقييم آثار هذه التقنيات بشكل صحيح على أداء أنظمة ASC والمقارنة بينها بطريقة سهلة. بدافع من هذه الاحتياجات ، صممنا في عملنا طاولة عمل قائمة على تصنيف المشهد الصوتي تتكون من مجموعة متنوعة من التقنيات والأساليب المتعلقة بهذه المهمة. الهدف من هذا العمل هو السماح للمستخدمين بتطبيق التقنيات التي يرغبون في استخدامها على مجموعة بيانات مختارة ، وإجراء تغييرات في المعلمات بسرعة ، وإنشاء اختبارات كبيرة ، وتصور نتائج هذه الاختبارات. لإنجاز هذه المهمة ، قمنا بإنشاء طاولة عمل تتكون من وحدات تحتوي على أدوات مختلفة لكل وحدة. على وجه التحديد ، قمنا بتنفيذ 5 وحدات تتعلق بمراحل اطار عمل ASC وهي على التوالي: تحميل البيانات والتصور لتحميل مجموعة بيانات والحصول على نظرة عامة عليها . تحويل البيانات لاستخراج الميزات ذات الصلة من مجموعة البيانات. زيادة البيانات لزيادة حجم البيانات. التدريب لتدريب الشبكات العصبية العميقة المستخدمة على مجموعة البيانات التي تم الحصول عليها وتقييم أدائها. وأخيرًا ، التنبؤ لتقييم أداء النماذج المدربة على إجراء تنبؤ صفي دقيق لتسجيل صوتي مختار. بعد ذلك ، لتنفيذ طاولة العمل الخاصة بنا لمعرفة كيفية أدائها ، استخدمنا مجموعة بيانات TAU Urban Acoustic Scenes 2019 المعدلة التي تتكون من 7192 تسجيلًا صوتيًا مع 10 فئات مختلفة في 5 مدن مختلفة لكل منها ، اختبرناها على التقنيات الموجودة مع تسجيل نتائج الدقة المتحصل عليها في جدول نتائج.

الكلمات المفتاحية: تصنيف المشهد الصوتي, طاولة العمل, التعلم الالي, زيادة البيانات, الشبكات العصبية العميقة.

Acknowledgements

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout our work.

We are grateful for all the people who have helped and support us directly or indirectly in the realization of our thesis.

In the first place, we would like to thank our supervisor Ms. YEKHLEF Hadjer for accepting us as her thesis students and providing us the chance to conduct this very interesting work and for her guidance throughout the researches and implementation work. The wise counsels and tips were useful during this project, and we are sure they will prove useful in the rest of our career.

Secondly, Special thanks goes to Ms.DIFFALLAH Zhor for helping us with her invaluable ideas and wisdom, and to the examiners for the time they spent on carefully reviewing this thesis.

Finally, we want to dedicate our work to our beloved families for being there for us, supporting us and motivating us to give our best for the completion of this work.

Content

List of Figures

List of Tables

List of acronyms

A

AASP: Audio and Acoustic Signal Processing

ADC: Analog to Digital Conversion\ converters

AI: Artificial Intelligence

ANN: artificial neural network

API: Application Programming Interface

ASC: Acoustic Scene Classification

C

CNN: Convolutional Neural Network

D

DFT: Discrete Fourier Transform

DCASE: Detection and Classification of Acoustic Scenes and Events

DA: Data Augmentation

DNN: deep neural network

F

FT: Fourier Transform

FNN: Feedforward neural network

FC: Fully Connected layer

G

GMM: Gaussian mixture model

GUI: Graphical User Interface

H

HTK: Hidden Markov model Toolkit

M

MFCC: mel frequency cepstrum coefficient

MFC: mel-frequency cepstrum

ML: Machine Learning

MLP: multilayer perceptron

N

NN: Neural Network

NLL: Negative Log-Likelihood Loss

R

ReLu: Rectified Linear Unit

RNN: recurrent neural network

S

STFT: Short-Time Fourier Transform

SVM: support vector machines

SSL: Semi Supervised Learning

V

VGG: Visual Geometry Group

INTRODUCTION

Sounds carry a large amount of information about everyday environments. This information can indicate the sound sources, surrounding environment, music genres, possible dangers or even the emotions of a speaker; thus, playing a crucial part in our daily communication and interaction with the world. Humans can classify sounds all the time without conscious effort. For example, recognizing different indoor and outdoor acoustic environments from recorded acoustic signals and telling the difference between it, these are tasks that we don't consider very difficult. Therefore, developing methods to automatically extract this information has a huge potential in relevant applications, such as autonomous cars or home assistants. Such systems are known as **audio scene recognition** systems and have been attracting a continuously growing attention during the past years [1]. An audio scene is described as a collection of sound events on top of some ambient noise. Given a predefined set of tags where each describes a different audio scene (i.e. Airport, Public Park, Metro, etc.) and an audio clip coming from a particular audio scene, **Acoustic Scene Classification** (ASC) is the automatic assignment of one single tag to describe the content of the audio clip. Its applications range from fields like multimedia, bioacoustics monitoring, intruder detection in wildlife areas and audio surveillance, and environmental sounds, [2] to noise mitigation enabled by smart acoustic sensor networks.

Furthermore, Acoustic scene classification systems can process the sound received through various major stages with the help of multiple algorithms and methods. Generally, there are three different stages that are attached to the classification of sound; **pre-processing** of the audio signal, specific spectral **feature extraction**, and finally the **classification** of the audio signal [1]. Audio pre-processing enhance certain characteristics of the signal by reducing the effects of noise or by emphasizing the target sounds in the signal [3]**.** In the feature extraction phase, a 2D time-frequency representation of audio data is obtained via multiple signal processing-based methods for extracting relevant features from the audio signal. The most well-known features for audio classification are mel spectrogram, log frequency spectrogram and mel frequency cepstrum coefficient (MFCC). The last stage of ASC is the classification of the audio signal using the extracted features in the classification model. Common model used for ASC task is CNN as it works with 2D time frequency representation as inputs. Classic ASC systems were based on feature-engineering approaches, where the research effort was mainly

aimed at developing meaningful features and using them to feed classical classifiers. However, state of art ASC systems focuses on deep learning based algorithms to learn the associations between class labels and audio recordings within a dataset.

Great number of acoustic scene classification systems have been developed using a plethora of audio processing methods and machine learning paradigms. However, there is no single system that works in most cases with multiple audio processing techniques to properly assess the effects of these techniques on the performance of these systems and facilitate for the users to test a method without having to write a full code.

Therefore, it became of paramount importance to build a dedicated **workbench** that consists of different tools for acoustic scene classification task for comparing sound analysis systems easily. A workbench is defined as a specialized table used to work with various tools needed for a certain job. In software, a workbench is a graphical interface for users to perform a specified task using the necessary techniques involved in the workbench. It is good practice to compare ideas and techniques with multiple other techniques, if the tools for these other techniques are easy to obtain and use, the comparison becomes easier. Motivated by these needs, we have designed an acoustic scene classification-based workbench that combines variety of techniques and methods related to this task.

The process of using the workbench can be broken into different stages correlated to the ASC pipeline, and we can characterize it as follows:

- Obtain the dataset which is a set of audio scene recordings defined with its class.
- Transform the dataset into a format that the model can use. This step is concerning the feature extraction process.
- Increase the size of the dataset. Regarding the problems that could occur due to the needs of the model such as overfitting problem, some methods can be often applied on the transformed dataset. Those methods can require a restructuring of the data. Among these methods, we mention Data Augmentation techniques.
- Train the model with the dataset. This step can be repeated several times with different parameters to obtain the best results.
- Gather results. To use the results, they have to be extracted from the output of the model and be saved as checkpoints.

- Use the results. This brings the results together, possible from different executions of the model, or different models. A prediction can then be made.

The goal of this work is to design and implement a workbench that can be used to optimize the techniques that create the predictions. This workbench should allow the users to make changes in the parameters fast, create big tests, and visualize the results of these tests. The results of different tests are gathered automatically, and the users can look at this data from different angles. Using these views of the results, the users can continue the search for a better prediction. The use of this program offers multiple advantages. The first is that the creation of tests is easier. Instead of manually making changes or writing tools to do so. The second advantage is that the program is extendible. New modifications or prediction techniques can be added fairly easily, and these new blocks can cooperate with all existing parts to create new predictions.

Thesis organization:

This thesis is categorized into four chapters. In **Chapter 1**, we cover some fundamentals behind sound signals. Specifically, we describe the various representations of sound signals and the most frequently used feature extraction techniques in the audio analysis task. In **Chapter 2**, we introduce the acoustic scene classification task, providing a brief description of machine learning with its pipeline, state-of-the-art deep learning architectures used in our work along with the different techniques to improve its performances. In **Chapter 3**, we introduce our workbench design and architecture to gain an understanding of its performance, we also extend the objective that was stated for the workbench. Lastly, in **Chapter 4**, we define the dataset employed for executing the workbench along with the setups and libraries used in building it. We afford also the design of the graphical interface of the workbench as well as the results that occur in every step of using it.

CHAPTER 1: Basics Of An Audio Signal Processing

1.1 Introduction

The growing amount of audio and music data on the Internet and in user databases leads to an increasing need for intelligent browsing, retrieving, and processing of this data with automated methods. Audio signal processing, aims at extracting audio properties (or features) directly from the audio signal to support these tasks. Knowledge of these properties allows us to improve the interaction of humans or machines with digital audio signals. It enables new ways of assessing, processing, and visualizing music.

The central objective in audio processing is the extraction of information from audio signals and to represent the signal with a minimum number of bits while achieving transparent signal reproduction.

Chapter 1 starts by introducing some definitions and offers a short reiteration of the most important tools of digital signal processing for the analysis of audio signals.

1.2 Audio signals

An audio signal is the representation of a sound which is a variation in **air pressure** over **time**. It is generated by vibration of particles in a medium received by our ears and converted into some information by neural actions of our brain. If the signal perceived is meaningful and follows some symmetrical pattern then our brain considers it as sound, else if it is meaningless and doesn't have any symmetry it is considered as noise which causes uneasiness. Formally sound waves are defined as the propagation of a disturbance of particles through any medium without permanent displacement of particles themselves involving changes in the particle characteristics such as pressure, displacement, and velocity [4]**.**

1.3 Types of audio signals

Signals may be grossly categorized into two types of signals depending upon their statistical features. As it is shown in **Figure 1.1**:

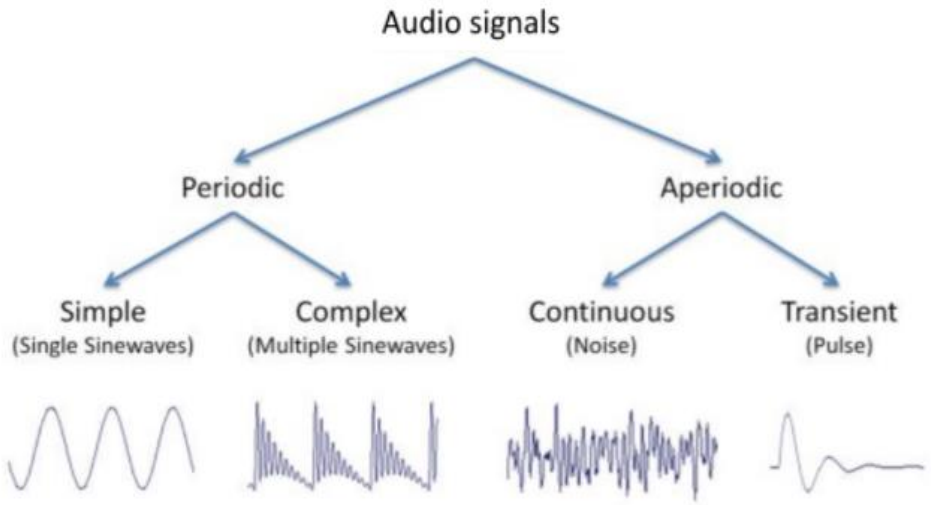

Figure 1.1: Types of audio signals. *[5]*

1.3.1 Periodic signal

Many of the sounds we hear are periodic. When these sounds occur, vibrations repeat the same motion over and over again. One repetition of the vibration is referred to as a **cycle**. Each cycle occurs over a constant duration of time referred to as a **period**. Conceptually, a period is the time it takes for a signal to complete one cycle. A signal's period is represented with the Greek letter *tau* τ; where:

$$\boldsymbol{T} = \frac{time}{1\ cycle} \tag{1.1}$$

1.3.2 Aperiodic signal

Many real-world signals, such as environmental noise signals, and speech and music signals are aperiodic. Aperiodic sounds are sounds that do not repeat [6]**.** Their values are random variables, i.e. signals take random values at various times and, therefore, cannot be modeled deterministically using math functions. They can however be characterized statistically. There are two ways of imagining how a non-repeated sound would be:

- **A transient sound** is one that occurs just once, briefly (For example, a popping balloon, dropping a pen, or synthesizing one cycle of a sine wave)
- **A noise sound** is one where the pressure at each moment is random and does not depend on the previous moment (For example, a tire leaking air or static on a TV).

1.4 Characteristics of audio signal

Audio signals may be characterized by different parameters in order to know all the information of that signal to have the ability to manipulate it. The various important properties which characterize an audio signal are:

- **Amplitude:** The amplitude of a sound wave can be defined as the loudness or the amount of maximum displacement of vibrating particles of the medium from their mean position when the sound is produced. Loudness is directly proportional to the amplitude of the sound. If the amplitude of a sound wave is large, then the loudness of sound will be more. If the amplitude is small, then the sound will be feeble. [7]

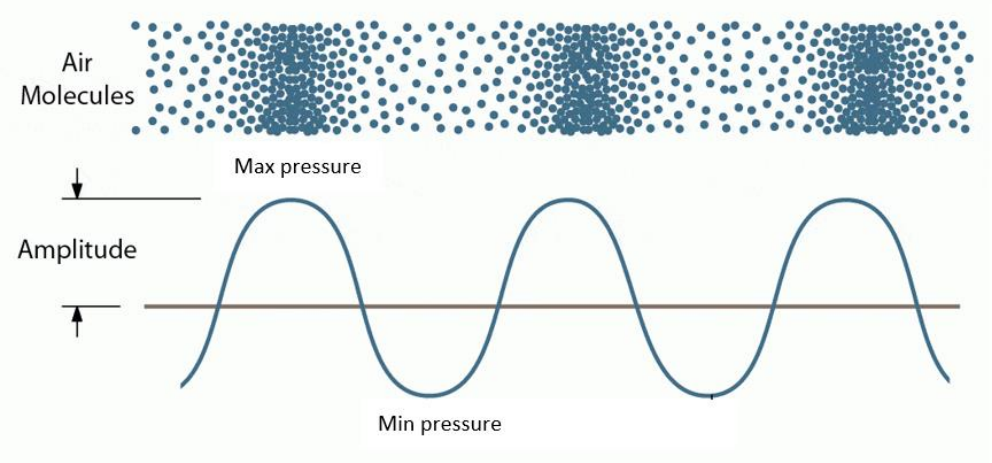

Figure 1.2: A representation of difference in amplitude. *[8]*

- **Frequency:** The frequency is defined as the number of oscillations per second. Its unit is hertz and is denoted by Hz. The frequency of a wave in general means how frequently the particles of a medium vibrate when a wave moves through the medium. [7] The generally established audio frequency range is 20 Hz to 20,000 Hz, which corresponds to the lower and upper limits of human hearing. [5]

- **Cycle:** One complete oscillation of a sound wave is called a **Cycle**. Cycles are also referred to as vibrations. [7]

- **Period:** Is a particular characteristic of periodic signals, it's the time required to complete one cycle of vibration. [7]

1.5 Representation of audio signal

An audio signal is a signal that contains information in the audible frequency range. **Audio representation** refers to the extraction of audio signal properties, which are representative of the audio signal composition and audio signal behavior over time [9]**.**

The signal is defined for all times t and is therefore a continuous signal *x {t)* which is called an analog signal. In order to process the audio signal in a digital way, the analog signal must be converted to digital. A digital signal is a signal that represents data as a sequence of discrete values [10]**.** The procedure of digitalizing provide us a representation of a sound in **a time-domain** or what is known as a "**waveform**". **Figure 1.3** is a representation of that waveform.

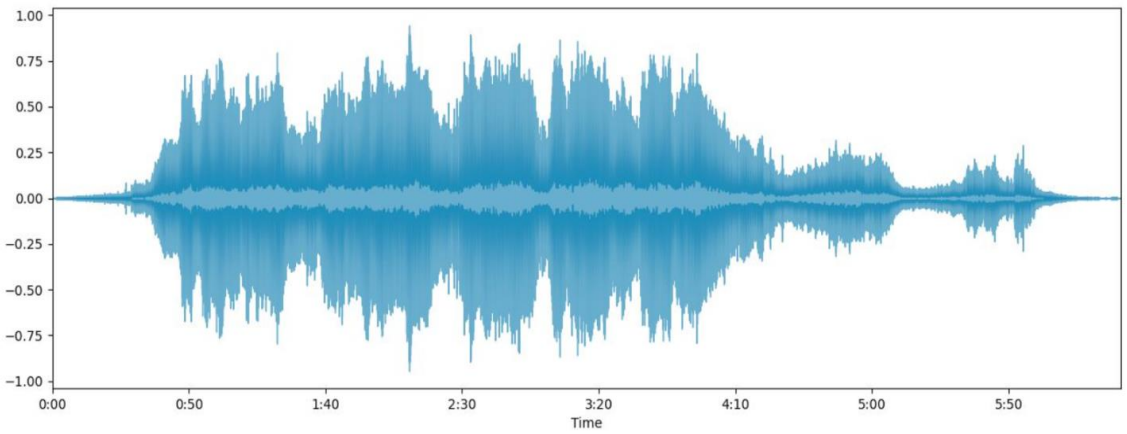

Figure 1.3: A representation of a Waveform (time-domain signal). [5]

1.6 Analog to Digital Signal Conversion (ADC)

Analog-to-Digital converters (ADC) translate analog signals into a digital representation of that signal. This digital representation can then be processed, manipulated, computed, transmitted or stored [11]**.** An ADC carries out two processes, **sampling** and **quantization**:

1.6.1 Sampling

The discretization in time is done by sampling the signal at specific times. The distance Ts in seconds between sampling points is equidistant in the context of audio signals and can be derived directly from the sample rate $f\ /s$ by:

$$Ts\ =\ 1/\ f\ s \tag{1.2}$$

Figure 1.4 visualizes the process of sampling by plotting a continuous signal and a sampled representation of it, sampled at a sample rate of $f\ /s\ =\ 700\ Hz$. An important property of sampled signals is that the sampled representation is ambiguous as exemplified in **Figure 1.5**. This example shows that different input signals may lead to the same series of samples.

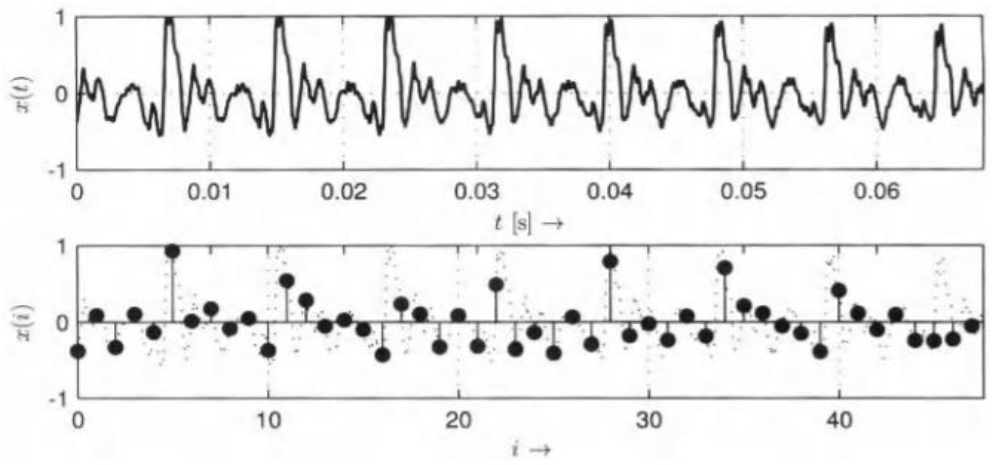

Figure 1.4: Continuous audio signal and corresponding sample values at a sample rate of $f\ /s\ =\ 700\ Hz$.

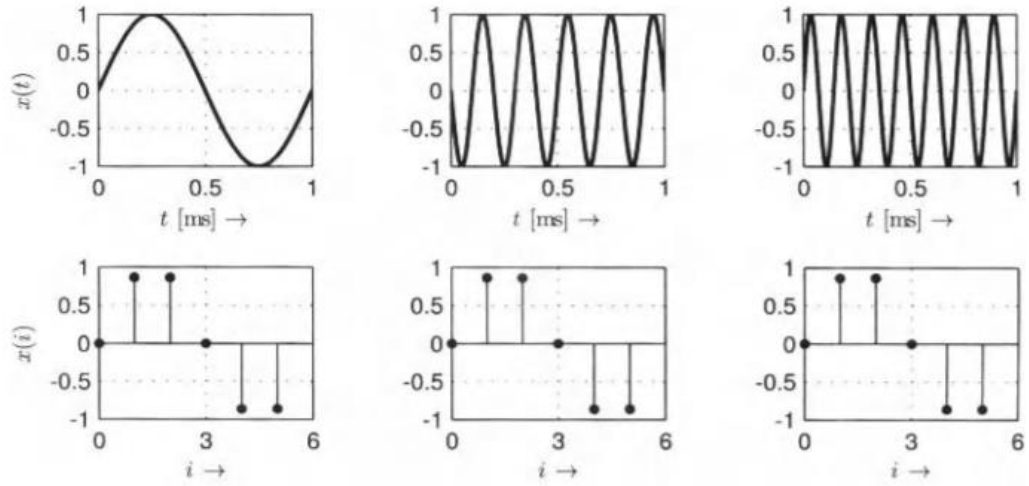

Figure 1.5: Continuous (top) and sampled (below) sinusoidal signals at a sample rate is $f/s = 6\,kHz$.

A sampled signal can only be reconstructed without loss of information if the sample rate f/s is higher than twice the highest frequency f_{max} in the sampled audio signal.

$$f/s > 2f_{max} \tag{1.3}$$

1.6.2 Quantization

In order to discretize the signal amplitudes the signal is quantized. Quantization means that each amplitude of the signal is rounded to a predefined set of allowed amplitude values, as it is shown in **Figure 1.6**. Usually, the input amplitude range between the maximum and the minimum amplitude is assumed to be symmetric around 0 and is divided into M steps. If M is a power of 2, the amplitude of each quantized sample can easily be represented with a binary code of word length (*w*) through the encoding step where:

$$w = log2(M) \tag{1.4}$$

1.6.3 Encoding

It converts each quantized sample $x_q(nt_s)$ into "*b*" bits code word**.** [12] The number of bits determines the number of quantization levels. An N-bit AD converter has 2N quantization levels and outputs binary words of length N (that is, it outputs

N-bit values for every sample). [13] **Figure 1.6** shows the pipeline of digitalizing process.

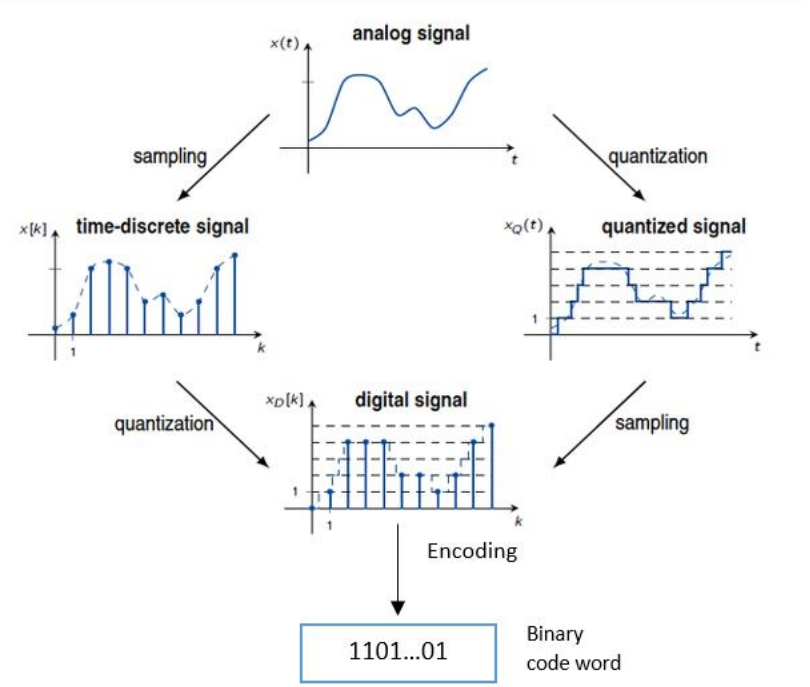

Figure 1.6: Digitalization process steps (sampling, quantization and encoding). *[14]*

1.7 Fourier Transform

The Fourier Transform (FT) is a mathematical technique that transforms a time-domain signal *x (t)*, to a frequency-domain *X (ω)* or what is known as "Spectrum" -as it is shown on **Figure 1.7**- in order to have a better comprehension of the sound signal and to identify or localize sound events. The Fourier representation of a signal shows the spectral composition of the signal [15]**.** It contains a list of sinusoidal functions, identified by frequency, and each sinusoid has an associated amplitude and phase. The phase of a signal is the start location of the sinusoid relative to some specific zero [16]**.** The discrete Fourier transform (DFT) is the most direct way to apply the Fourier transform. It is given by the equation:

$$X(w) = \int_{-\infty}^{+\infty} x(t)e^{-jwt}dt \qquad (1.5)$$

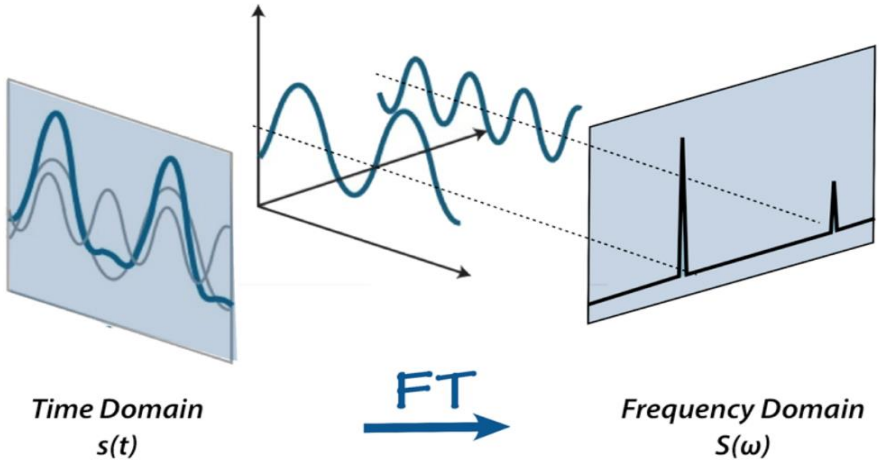

Figure 1.7: Transformation of time-domain signal into frequency-domain signal using Fourier Transform. *[17]*

1.8 Short-Time Fourier Transform

Spectral analysis of no stationary signals cannot be accommodated by the classical Fourier transform since the signal has time-varying characteristics. Instead, a time-frequency transformation is required.

Short-time Fourier transform (STFT) is a series of Fourier transforms of a windowed signal. It provides the time-localized frequency information for situations in which frequency components of a signal vary over time, whereas the standard Fourier transform provides the frequency information averaged over the entire signal time interval [18]**.** The procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length called "**Frames**" and then compute the Fourier transform separately. This reveals the Fourier spectrum on each Frame. Visualization of STFT is often realized via its spectrogram on **Figure 1.9**, which is an intensity plot of STFT magnitude over time**.**

However, As a consequence of dividing the signal, there is often a discontinuity or break in the signal at the frame boundaries. This introduces spectral components into the transform that are not present in the original signal i.e. spectral leakage. And to solve this discontinuity, a procedure called **windowing** is a must.

Windowing Is the process of eliminating samples by scaling the amplitude of the signal to zero at both endpoints of each frame to generate a periodic signal. When these windowing functions are applied to a signal, it is clear that some information near the frame boundaries is

lost. For this reason, a further improvement to the STFT is to **overlap the frames.** As shown in **Figure 1.8.**

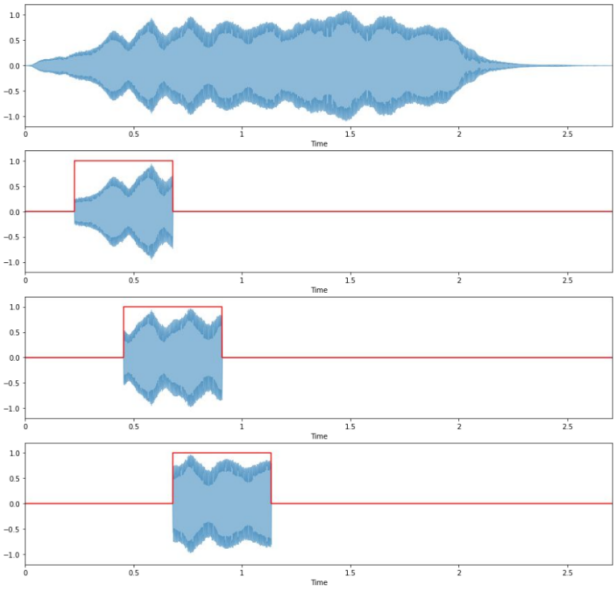

Figure 1.8: An example of overlapping windows. *[19]*

1.9 Audio pre-processing

The raw audio data is frequently pre-processed before computing instantaneous features from the data. The motivation for this preprocessing step is to reduce the amount of audio data to be analyzed by omitting unnecessary information or to minimize the impact of unwanted information on the extracted features. The role of this step is to enhance certain characteristics of the signal for further analysis [20].

1.10 Audio features

In order to represent the information contained in the audio (stream) in a compact way that focuses on the task of interest, a parameterized form is usually chosen. These parameters usually describe the properties of the audio in a highly information-reduced form, and usually describe the properties of the audio at a relatively low rate, such as the average energy or pitch over a long period of time. Because different intelligent audio analysis tasks are usually best represented by different such "functions".

The model cannot directly understand the data provided by the audio, so for converting it into an understandable format, feature extraction is employed. This is a process of interpreting most of the data in an understandable way. Classification, prediction, and recommendation algorithms require feature extraction.

Generally, there are two types of audio characteristics: **physical features** and **perceptual features**. Physical features refer to mathematical measurement values directly calculated from sound waves, such as energy function, frequency spectrum, cepstrum coefficient, fundamental frequency, etc. Perceptual features are subjective terms related to human perception of sound, including loudness, brightness, pitch, timbre, rhythm, etc...

1.10.1 Physical features

- **Spectrogram:** A spectrogram is a visual way to show the change in signal strength or loudness over time at various frequencies present in a specific waveform. Not only can you see how much energy is between 2 Hz and 10 Hz, but you can also see how the energy level changes over time. The spectrogram is basically a two-dimensional graph, and the third dimension is represented by colors as illustrated in **Figure 1.9**. The method of reading the spectrogram is simple and easy, and the time runs from left to right along the horizontal axis. The vertical axis represents frequency, which can also be considered as pitch or tone, with the lowest frequency at the bottom and the highest frequency at the top. The amplitude (or energy or "loudness") of a specific frequency at a specific time is represented by the third dimension. The dark color corresponds to low amplitude, and the brighter the color is to yellow, it corresponds to gradual increase (or loudness) amplitude.

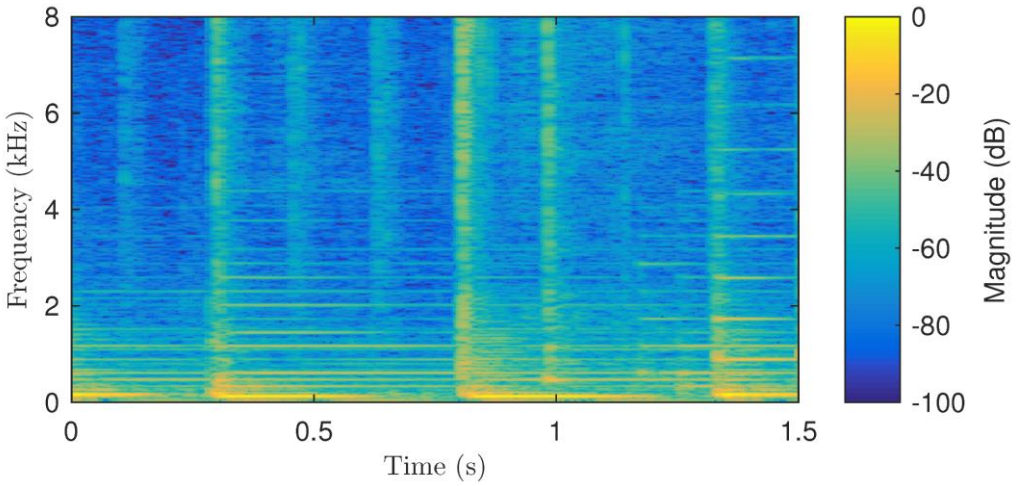

Figure 1.9: A representation of a spectrogram. *[21]*

- **Mel-Spectrogram:** The Mel spectrogram is a spectrogram that visualizes sound on the Mel scale. The Mel scale is the logarithmic transformation of the signal frequency [22]**.** The core idea of this conversion is to treat equidistant sounds on the Mel scale as equal to human perception. It is actually harder for humans to distinguish higher frequencies, and easier to distinguish lower frequencies. This is the basis of Mel Scale in machine learning audio applications because it mimics our own perception of sound. Mel-Spectrogram is a sound function. Many top teams in DCASE 2017 use Mel spectrograms and believe that they are most suitable for acoustic scene classification. The mel spectrogram is obtained by applying the mel filter bank, using the HTK method. In the HTK method, Hz is converted to mel using the following formula:

$$m = 1127.log(1 + \frac{f}{700}) \quad (1.6)$$

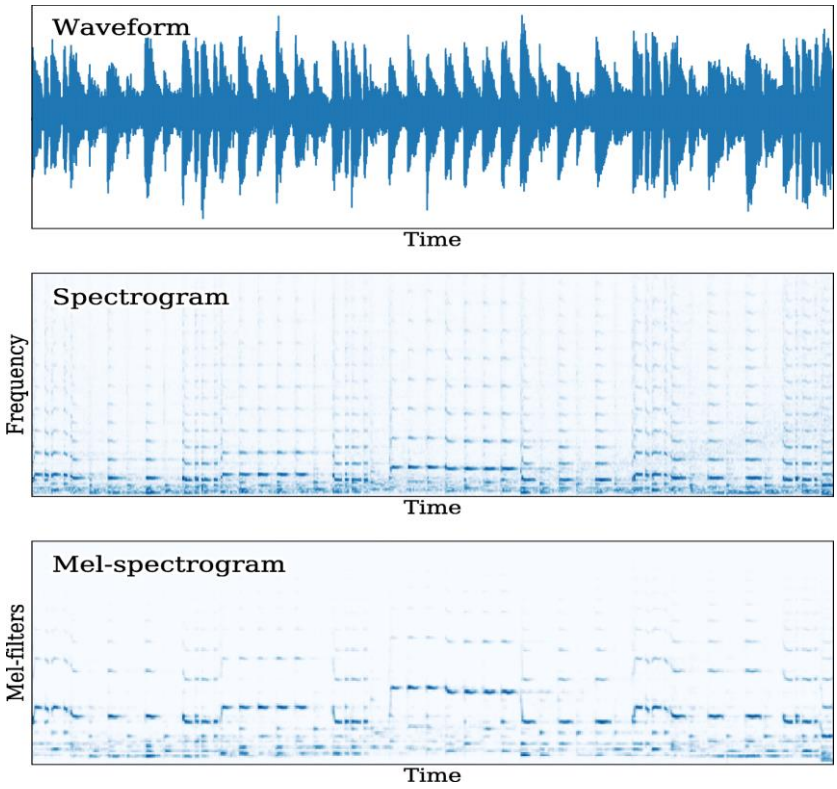

Figure 1.10: Comparison between a time-domain signal, spectrogram and mel-spectrogram of a 9s speech segment. *[23]*

- **Mel-Frequency Cepstral Coefficients (MFCC):** The mel frequency cepstral coefficients (**MFCCs**) of a signal are a small set of **features** (usually about 10-20) which concisely describe the overall shape of a spectral envelope [24]**.** They are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel-scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of sound. In MIR, it is often used to describe timbre.

  In sound processing, the **mel-frequency cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

  The basic procedure to develop MFCCs is the following:

  - Convert from Hertz to Mel Scale.
  - Take logarithm of Mel representation of audio.
  - Take logarithmic magnitude and use Discrete Cosine Transformation.

- This result creates a spectrum over Mel frequencies as opposed to time, thus creating MFCCs.

- **Log-Frequency Spectrogram:** Also referred to as “constant-q power spectrogram”, it’s obtained from a spectrogram by converting the linear frequency axis (measured in Hertz) into a logarithmic axis (measured in pitches). It is generated as a mapping applied to a STFT representation. Each bin in the Log-frequency spectrogram is formed as a weighting of corresponding frequency bins from the original spectrogram [25]**.**

1.10.2 Perceptual features

- **Loudness:** Generally the loudness of a sound is related to the amplitude of the sound wave, a wave with bigger variations in pressure generally sounds louder. Changes in loudness in music are called Dynamics.

- **Pitch:** Pitch is the quality of sound which makes some sounds seem "higher" or "lower" than others. Pitch is determined by the number of vibrations produced during a given time period. The vibration rate of a sound is called its frequency the higher the frequency the higher the pitch.

- **Timbre:** Timbre is a French word that means "tone color". It is the quality of sound which allows us to distinguish between different sound sources producing sound at the same pitch and loudness. The vibration of sound waves is quite complex; most sounds vibrate at several frequencies simultaneously. The additional frequencies are called overtones or harmonics. The relative strength of these overtones helps determine a sound's timbre.

1.11 Conclusion

The purpose of this chapter was to present an overview on the topic of audio signal in order to gain an understanding of the basic concepts, to gain a knowhow of the audio signal processing techniques being used. Thus, we have seen different types of audio, its characteristics, the way we can represent it and its various features which will provide us the ability to use machine learning techniques for our work.

CHAPTER 2: Acoustic Scene Classification task using Machine Learning Approaches

2.1 Introduction

Sounds carry a large amount of information about surrounding circumstances and physical events [26]. Developing signal processing methods to extract this information automatically has huge potential in several applications, such as information retrieval [27], mobile devices, robots, etc. Acoustic scene classification (ASC) [28], one of such ongoing research subjects. Its aim is to classify audio recordings in a public area, into predefined acoustic scene classes. If a computer can recognize acoustic scenes, it can be applied to various fields, such as robotic navigation [29], context-aware services [30], surveillance [31], etc.

In the previous chapter we introduced the basics of an audio signal, its types and most substantial characteristics that distinguishes it, we also presented fundamentals about the different sound information and its representations in a specific form that can be comprehended by the machine.

Our main focus in this chapter revolves around the classification task in machine learning with use of the supervised learning method. Hence, in section 2.2 we define the ASC task, in section 2.3 we introduce the machine learning concept along with its various categories and pipeline. In the following section we present the machine learning subfield **Deep Learning** for the ASC task and some of its approaches and techniques.

2.2 Acoustic scene classification task

In recent years, acoustic scene classification (ASC) has attracted widespread attention in the Audio and Acoustic Signal Processing (AASP) community [32] [33]. The ASC problem was first proposed by Sawhney and Maes [34]. This task is an essential part of auditory scene analysis and involves describing an entire recorded acoustic signal using a pre-defined semantic description like “office room” or “public place”. Those semantic entities are denoted as **acoustic scenes** and the task of recognizing them as **acoustic scene classification** (ASC) [35]. Acoustic Scene Classification is an audio classification problem that carries broad interest due to the

development of context-aware devices and applications. It is a straightforward multi-class supervised classification problem in which the categories for classification are labels describing the acoustic scene. Acoustic scene classification aims to classify a test recording sound into predefined classes that characterizes the environment in which it was recorded [36]. Thus, for this purpose state-of-the-art ASC systems that are based on machine learning approaches have been shown to outperform humans on this task [37]. Therefore, they are applied in numerous application scenarios such as context-aware wearables and hearables, hearing aids, health care, security surveillance, wild-life monitoring in nature habitats, smart cities, IoT, and autonomous navigation.

2.3Introduction to Machine Learning

Machine learning is a subfield of artificial intelligence that gets computers to act without being explicitly programmed. It provides systems the ability to automatically learn and improve from experience without human intervention or assistance and adjust actions accordingly. In machine learning, rather than teaching a computer an enormous list of rules to unravel the matter, we provide it with a model with which it can evaluate examples, and a small set of instructions to switch the model when it makes an error. We expect that, over time, a well-suited model would be ready to solve the matter extremely accurately [38]**.**

Machine learning techniques can roughly be divided into two large classes: Supervised and Unsupervised learning, while one more class is often added Semi-Supervised learning.

2.3.1 Unsupervised Learning

Unsupervised learning studies how systems can deduce a function to explain a hidden structure from unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to estimate a possible solution [38]. So this learning allows the model to work on its own to discover patterns and information that was previously undetected [39]. Unsupervised learning is more difficult when compared with supervised learning, since we do not have the ground truth to evaluate the results [40]. The most common type of unsupervised ML algorithm is clustering.

2.3.2 Supervised Learning

In contrast to unsupervised learning, supervised learning is defined by its use of labeled datasets to train algorithms that classify data or predict outcomes accurately [41]**.** This learning

employs a training set for teaching the models to yield the specified output. Usually, the algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized in order to modify the model accordingly [41]**.** This learning process is predicated on the comparison of computed output and expected output.

Supervised learning are often separated into two types of problems, classification and regression:

- **Classification**: refers to the process of classifying something according to an established criteria. In machine learning, classification uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined [42]**.** The final output in the classification task is a label. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbors, and random forest [41]**.**

- **Regression**: not like classification which predicts discrete outcomes Regression techniques predict continuous outcomes. For example, the cost of an item, the temperature, the simplest algorithm that can be used is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

2.3.3 Semi-Supervised Learning

Semi-supervised learning (SSL) is a type of Machine Learning (ML) technique. It is half-way between supervised and unsupervised. The main objective of semi supervised learning is to overcome the drawbacks of both supervised and unsupervised learning. SSL has been proposed by the research community, which can learn with a small amount of training data and can label the unknown (or) test data. It builds a model with few labeled patterns as training data and treats the rest of the patterns as test data [40]**.**

2.3.4 Machine Learning Pipeline

In the machine learning applications, a pipeline (a series of complex data processing steps) processes a labeled training dataset and produces a machine learning model. The model then has to be deployed into a deployment platform where it answers prediction queries in real-time. To properly preprocess the prediction queries, typically the pipeline has to be deployed

alongside the model [43]**.** Machine learning-based algorithms for ASC have a typical processing steps illustrated in the following **Figure**:

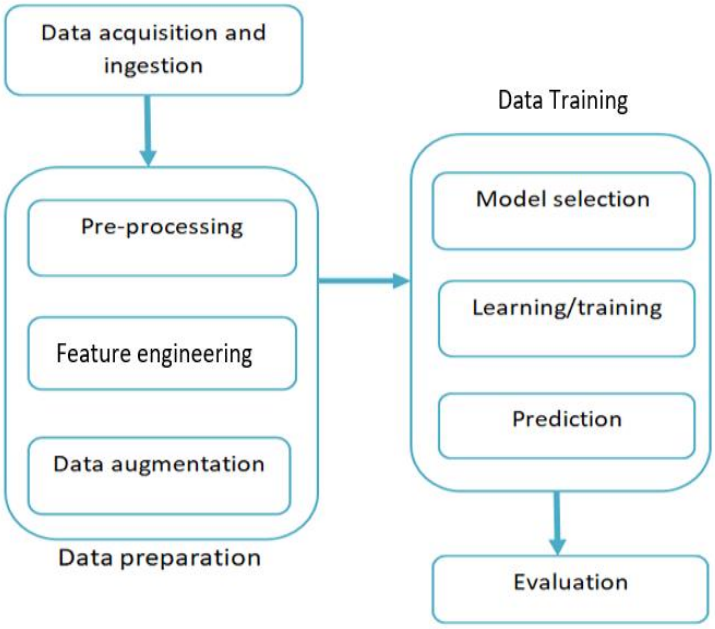

Figure 2.1: Machine Learning Pipeline. *[44]*

1. Data acquisition and ingestion

Data acquisition is the process of collecting as much data as possible through different ways. The Data might be in different formats: texts, images, audios… and gathered all in the form of a **dataset**. This data will be used in the creation of an artificial intelligent system to ensure its good performance. The Data can be collected in three methods:

· **Creating a new dataset:** It is done by recording real-world sounds as they are present in real life having the control over the data content. However the recording should be done many times to cover max of acoustic variability, and that is by changing some factors such as time and place which is time consuming and financially draining [45].

· **An existing dataset:** There are several free datasets available for different machine learning tasks like: classification. As for the audio scene classification problem there is the Detection and Classification of Acoustic Scene and Events (DCASE) datasets which are

provided for the different audio centered machine learning tasks. And other datasets as NYU Urban Sound, and Freefield … [46].

· **Data simulation:** This method allows the creation of different sounds by combining and mixing isolated sounds token with the control of its complexity and background noises, in order to get examples of complex sounds. But, it is a limited method because the results can be unrealistic [47].

On the other hand, data ingestion is a process by which data is obtained and imported from one or more sources to a destination (system) where it can be stored and further analyzed.

2. Data Preparation

Before data can be analyzed and fed to the learning algorithm, it must be organized into an appropriate form. Data preparation is typically an iterative process of manipulating and organizing raw data, which is often unstructured and messy, into a more structured and useful form that is ready for further analysis [48]. The whole preparation process consists of a series of major activities (or tasks) as shown in **Figure 2.2**, including **data pre-processing**, **transformation of features**, and **data augmentation.**

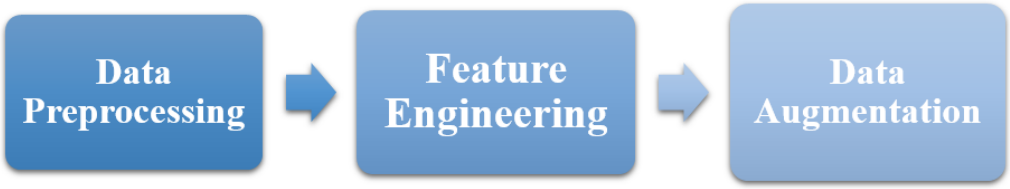

Figure 2.2: Data Preparation task's pipeline.

A. Data Pre-processing

Data preprocessing is considered as one of the most important phases in ML [49]. There is a hierarchy of problems that are often encountered in data pre-processing, such as corrupts or inaccurate records, missing values, redundant data. So this phase implies cleaning the raw data by solving those problems with numerous approaches that exist for addressing these hurdles depending on the type of problem at hand. Several filtering approaches are used as pre-processing for ASC algorithms [44]. For example, a common pre-processing technique that is

used to compensate for the average spectral shape is called **pre-emphasis**. It emphasizes higher frequencies in an audio in order to overcome the problem of underrepresentation of frequencies in the log-frequency spectrogram [50]**.** This technique can be applied using a filter on the signal to amplify the high frequencies using the following equation:

$$y[n] = \frac{(x[n] - alpha * x[n-1])}{(1 - alpha)} \quad (2.1)$$

Where *x* is the signal and alpha is proportion of the output signal that comes from the first derivative and its typical values for are 0.95 or 0.9 [51]**.**

The pre-emphasis filter is useful to balance the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower frequencies.

B. Features Engineering

It is the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model. It is a crucial step in the machine learning pipeline, because the right features can ease the difficulty of modeling, and therefore enable the pipeline to output results of higher quality [52]**.** Moreover, Feature extraction is one of the most important stages in the acoustic scene classification task. It extracts appropriate parameters of the audio signal that inform about its most significant traits aiming at making the most of its particular characteristics. Common features for the ASC task that have been widely used are **mel-spectrogram** and **constant-q power spectrogram** introduced in the **section 1.10**.

C. Data Augmentation

For acoustic scene classification, the labeled training data is limited in the public dataset, thus we will have a small dataset, and with that we will be prone to overfitting. A classification model trained on such a small data set may not work well in practice, due to the mismatch between the training and test data distribution. However, it is possible to reduce the over-fitting of the model and provide generalization of the test distribution by minimizing the proximity risk in order to enhance the generalization ability of the acoustic scene classification model. Therefore, one of the ways of dealing with the lack of data problem and expanding the size of the data set is **Data Augmentation (DA).**

3. Data Training

A. Model Selection

The choice of the learning algorithm is a critical step in the machine learning process. Usually, the process of selecting a suitable machine learning model for a particular problem is done in an iterative manner. First, an input dataset must be transformed from a domain specific format to features which are predictive of the field of interest. Once the features have been engineered, users must pick a learning setting which is appropriate to their problem as classification or recommendation. Next, users must pick an appropriate model, such as support vector machines (SVM), or any flavor of neural networks (NNs). Each model family has a number of hyper-parameters, such as regularization degree, learning rate, number of neurons, etc. which must be tuned to achieve optimal results. Finally, users must pick a software package that can train their model, configure one or more machines to execute the training and evaluate the model's quality. However, it can be challenging to make the right choice when selecting a model based on intuition or randomness and/or leave hyper-parameters set to default. This approach will usually yield sub-optimal results. Therefore, the model can be selected in an automatic way and simultaneously set its hyper-parameters to optimize the performance [53].

The procedure of selecting a model: Given a set of learning algorithms A and a limited amount of training data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. The goal of model selection is to determine the algorithm $A^* \in A$ with optimal generalization performance. Generalization performance is estimated by splitting D into disjoint training and validation sets $D_{train}^{(i)}$ and $D_{valid}^{(i)}$ learning functions fi by applying A^* to $D_{train}^{(i)}$, and evaluating the predictive performance of these functions on $D_{valid}^{(i)}$. This allows for the model selection problem to be written as:

$$A^* \in Argmin \frac{1}{k} \sum_{i=1}^{k} L\left(A, D_{train}^{(i)}, D_{valid}^{(i)}\right) \tag{2.2}$$

Where $L\left(A, D_{train}^{(i)}, D_{valid}^{(i)}\right)$ is the loss achieved by A when trained on $D_{train}^{(i)}$ and evaluated on $D_{valid}^{(i)}$ [54].

B. Learning/Training

The process of training an ML model involves providing an ML algorithm (the learning algorithm) with training data to learn from. The training data must contain the correct answer,

which is known as a **target** or **target attribute**. The training data is obtained from applying a technique on the dataset that splits it into sets of training and testing. This technique is called **Data validation.** Data validation is a ML technique that should be done before training the model to prevent it from overfit. There is various methods for the data validation:

- **Train/validation/test split:** It is the most basic method of data validation, the typical ratio of the sets is 80/10/10 where there is enough training data. After training the model with the training set, we validate the results and tune the hyperparameters with the validation set till we reach a satisfactory performance metric. Then, we move on to testing the model with the test set to predict and evaluate the performance.

- **Cross validation:** Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation. Other forms of cross-validation are special cases of k-fold cross-validation or involve repeated rounds of k-fold cross-validation [55]**.**

In K-fold cross-validation, the dataset is broken into *k* number of folds as shown in **Figure 2.3**, wherein one fold or the validation fold will be used as the test set and the rest will be used as the training sets. The main advantage of this technique is that with it, every data sample gets to be tested exactly once and is used in training k-1 times; and hence the bias is reduced. Another advantage is that it works well on small and large data sets, because it ensures that every observation from the original dataset has the chance of appearing in training and test sets.

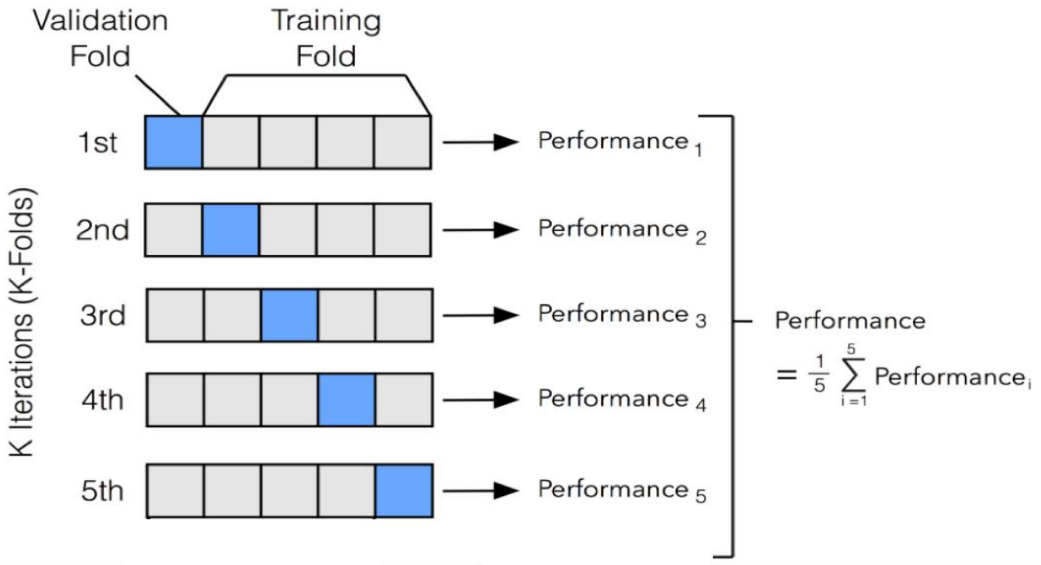

Figure 2.3: An example of 5-folds Cross validation procedure. *[56]*

The training model is used to run the input data (training set) through the algorithm to correlate the processed output against the sample target. The result from this correlation is used to modify the model. This iterative process is called "**model fitting**" and is done by adjusting the parameters of the model as the weights and biases accordingly with learning [57].

C. Prediction

The learning algorithm finds patterns in the training data that map the input data attributes to the target and it outputs an ML model that captures these patterns [58]**.** These patterns can be used later to get predictions on new data for which the target is unknown.

4. Evaluation and Deployment

Evaluation allows us to test our model against data that has never been used for training. In machine learning, model evaluation is referred to as the process where the performance of a fully trained model is evaluated on a testing set. The testing set consisting of a set of testing samples should be separated from the both training and validation sets, but it should follow the same probability distribution as the training set. Each testing sample has a known value of the target. Based on the comparison of the model's predicted value, y^i [59]**.**

For classification problems, the evaluation of the optimal solution during the training stage can be defined based on a **confusion matrix** as shown in **Table 2.1**. The row of the table

represents the predicted class, while the column represents the true class. From this confusion matrix, *tp* and *tn* denote the number of positive and negative instances that are correctly classified. Meanwhile, *fp* and *fn* denote the number of misclassified negative and positive instances, respectively.

Deployment of machine learning models, or simply, putting models into production, means making models available, so that they can receive data and return their predictions. It allows taking **full advantage of the model** we built [60]**.** For machine learning models, effective deployment is crucial to have real-world benefit and impact. Deployment required us to solve several new problems, covering technical implementation, governance of the system, the cost to use the system, and trust in the system [61]**.**

Table 2.1: Confusion Matrix Representation.

	Actual positive class	**Actual negative class**
Predicted positive class	True positive (*tp*)	False negative (*fn*)
Predicted negative class	False positive (*fp*)	True negative (*tn*)

2.4 Deep Learning for ASC

Deep learning was conceptualized by Geoffrey Hinton in the 1980s [62]. It is a subset of machine learning based on artificial neural networks which is concerned with algorithms inspired by the structure and function of the brain. Its algorithms are typically applied to extremely complicated domains such as images, audio sequences and text, and on AI tasks such as ASC or recognizing speech or recognizing objects which the simple machine learning algorithms cannot solve. Deep learning replaces not only the shallow models at the end of traditional machine learning pipelines, but also the labor-intensive process of feature engineering which is the reason for deep learning success in perceptual tasks. Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labeled training examples. Deep learning methods involve artificial neural networks, in which computations are structured in terms of interconnected

groups of artificial **neurons**, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables [42]**.**

Artificial neural networks are composed of layers of nodes (neurons). A layer of neurons consists of some neurons operating in parallel. The neurons in each layer are supposed to work at the same time but independently. Each node is designed to behave similarly to a neuron in the brain. The first layer of a neural network is called the **input layer**, followed by **hidden layers**, then finally the **output layer**; as shown in **Figure 2.4**. Each node in the neural net performs some sort of calculation, which is passed on to other nodes deeper in the neural network.

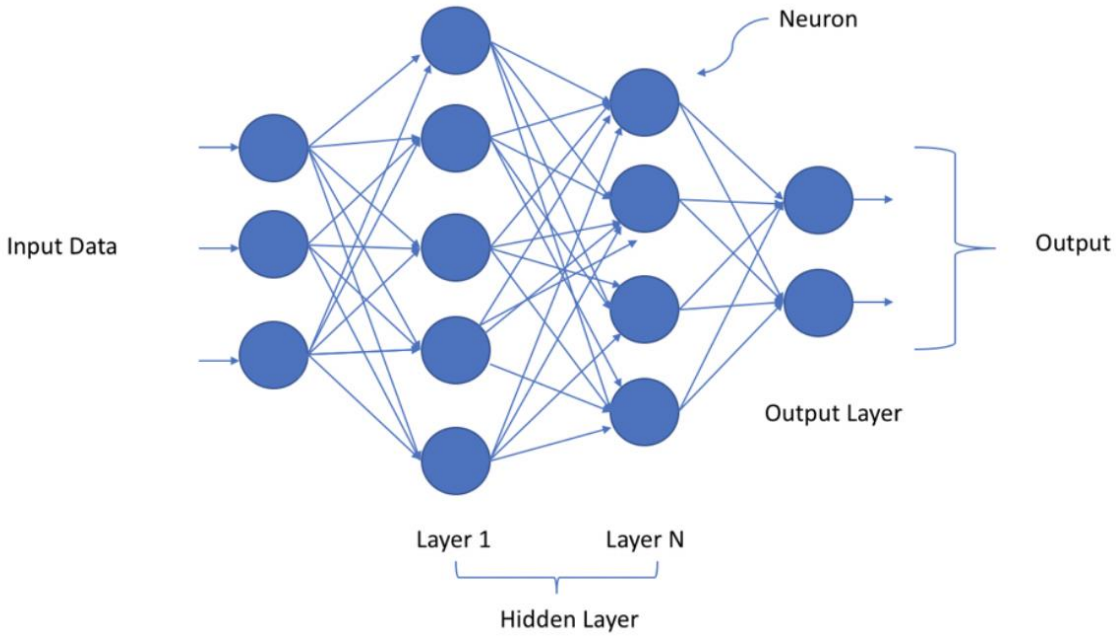

Figure 2.4: A representation of a typical artificial neural network. *[63]*

The artificial neuron takes a vector of input features $x_1, x_2, \dots, x_m$, and each of them is multiplied by a specific weight, $w_{k1}, w_{k2}, \dots, w_{km}$. The weighted inputs are summed together, and a constant value called *bias* (b_k) is added to them to produce the net input of the neuron, as shown in Figure 2.5. The network's input is then passed through an activation function φ *(.)* to produce the output y_k which is then transmitted to other neurons.

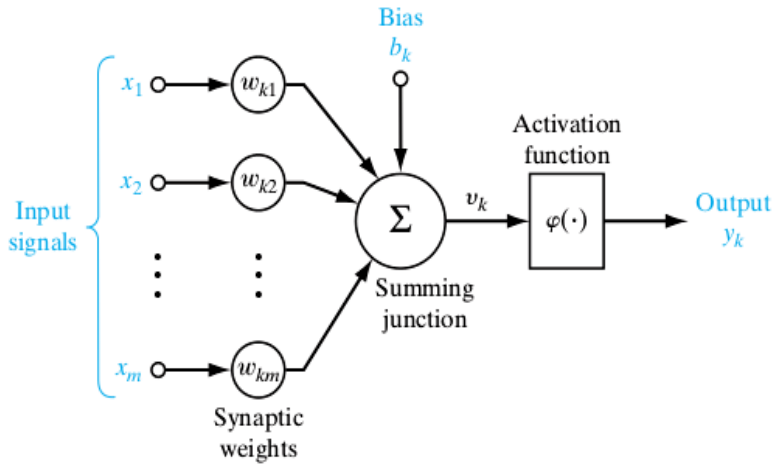

Figure 2.5: Schematic representation of an artificial neuron. *[64]*

The activation function is chosen by the designer, but w_i weight represents the strength of the connection between units and b are adjusted by some learning rule during the training process of the neural network. A **weight** decides how much influence the input will have on the output. **Biases**, which are constant, are an additional input into the next layer that will always have the value of 1. The bias unit guarantees that even when all the inputs are zeros there will still be an activation in the neuron. There are different activation functions that you can use in a neural net, and the one which is used more commonly is ***Rectified Linear Unit (ReLU) function***.

ReLU is a non-saturated activation function, which not only effectively improves the training speed of the model, but also better controls the problem of gradient disappearance and gradient explosion, it is easy to train a deeper network [65]. The form of ReLU function is given by:

$$ReLU(x) = max\,(0, x\,) \tag{2.3}$$

2.4.1 Feed Forward Neural Networks

The Feedforward neural network (FNN) is the simplest type of artificial neural network which has lots of applications in machine learning. It was the first type of neural network ever created before the convolutional or recurrent neural nets. These networks of models are called

feedforward because the information only travels forward in the neural network, through the input nodes then through the hidden layers (single or many layers) and finally through the output nodes. Feedforward neural networks are used in several ASC algorithms [44]. For example, Bisot et al used an FNN architecture to concatenate features from an NMF decomposition and a constant-Q transform of the audio signal [66]**.** Whereas, Takahashi combined an FNN with multiple Gaussian mixture model (GMM) classifiers to model the individual acoustic scenes [67]**.** The learning process in FNNs consists of a forward pass and a backward pass.

- In the forward pass outputs are calculated and compared with desired outputs. The function used to measure the mathematical distance between these output's values is called the "**Loss function**", and the result produced by this function is called "**loss**" or "**error**". During training, the aim is to minimize this error between the predicted value vs the actual value. Loss functions vary with the type of problem we are trying to solve (Regression problem or classification problem). In classification, the most common loss function used is Cross entropy loss function, it is used when adjusting model weights during training. It measures the degree of inequality between two probability distributions. Each predicted class probability is compared to the actual class desired output 0 or 1 and a score/loss is calculated that penalizes the probability based on how far it is from the actual expected value. The penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences tending to 0.
  Cross entropy is defined as:

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes} \tag{2.4}$$

  Where t_i is the truth label and p_i is the softmax probability for the i^{th} class.

  The cross entropy function can be categorized into 2 sub-function: **The Binary cross entropy loss** used for classification problems that have two classes, and **the categorical cross-entropy loss** which is as binary cross entropy loss but expanded to multiple classes. Another loss function for the classification problem is the **NLL Loss function.** It is useful to train a classification problem with C classes. The Negative Log-Likelihood Loss function (NLL) is applied only on models with the softmax function as an output activation layer. It uses a negative connotation since the probabilities (or likelihoods) vary between

zero and one, and the logarithms of values in this range are negative. In the end, the loss value becomes positive.

- In the backward pass, this error is used to alter the weights in the network in order to decrease the error. The most common but not exclusive way to conduct this process is with the optimization algorithm **gradient descent** and **backpropagation** [68]**. Gradient descent** is an optimization algorithm used for minimizing the loss function in various machine learning algorithms. It is basically used for updating the parameters of the neural network [69], whereas **Backpropagation** offers an efficient, algorithmic formulation for computing error derivatives with respect to the weights of a neural network. As such, it allows us to implement gradient descent for neural network training without explicitly hard-coding derivatives. In simple terms, after each forward pass through a network, backpropagation performs a backward pass while adjusting the model's parameters (weights and biases) [70]**.**

Forward and backward passes are repeated until convergence [71]**.**

2.4.2 Deep Neural Networks

A deep neural network (DNN) is an artificial neural network with multiple hidden layers of units between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships [72]. The extra layers enable composition of features from lower layers, giving the potential of modeling complex data with fewer units than a similarly performing shallow network [73]**.**

2.4.3 Convolutional Neural Networks (CNN)

Convolutional neural network (CNN) is a **deep neural network** algorithm that can absorb the input image, assign importance to various aspects and objects in the image and be able to distinguish each other. Compared with other classification algorithms, the preprocessing required in CNN is much lower. In the original method, the filters are manually designed, and with sufficient training, CNN can learn these filters/features. Thus, it is the most popular choice among researchers and application engineers due to its ability to solve both problems of **feature extraction** and **classification** simultaneously within a single computational structure [74]**.**

CNN's typical architecture represents a multi-layer artificial neural network (ANN) that is structured by kernel layers and ending with a classification layer. **Figure 2.6** shows the architecture of CNN [75]**.**

Moreover, CNN has shown remarkable results in many different areas [76] [77] such as Acoustic scene classification tasks. It can be applied to audio by transforming the audio of the waveform into an image of time-frequency representation of the signal (Spectrogram) so it can capture the time and frequency evolution of the signal [78]. In some cases, CNN is used in combination with multiple technologies, such as multilayer perceptron (MLP), recurrent neural network (RNN), support vector machine (SVM), Gaussian mixture models (GMM) And i-vectors.

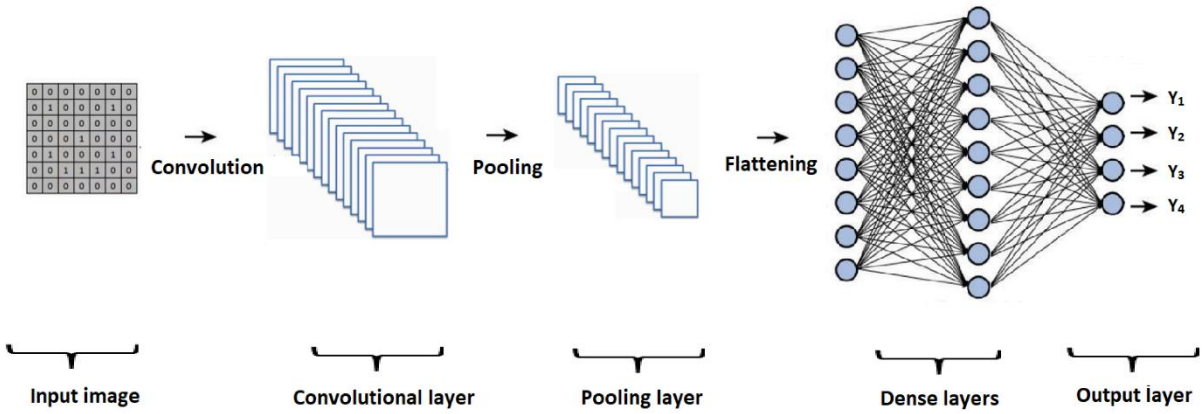

Figure 2.6: An architecture of a Convolutional Neural Network. *[79]*

As illustrated in **Figure 2.6**, CNNs consist of various sequential layers that build the model architecture some are repeated n times; these layers are:

A. Input Layer

The input layer is the input of the whole CNN. In the neural network of image processing, it generally represents the pixel matrix of the image. In ASC tasks, several works for automatic audio classification have successfully proposed to feed CNNs with raw 1D audio signals [80]**,** but, most state-of-the-art approaches use 2D time-frequency representations as a suitable input representation [81]**.**

B. Convolutional Layer

The convolutional layer is the first layer in a CNN architecture; this layer performs feature extraction from the input volume. The first convolutional layer is responsible for

capturing low-level features, such as edges, colors, gradient directions, etc. By adding layers, the architecture can also adapt to advanced features, so that we have a comprehensive understanding of the number of images in the dataset. The convolution layer is also known as (CONV) uses filters that perform convolution operations by convolving a local region of the input volume (receptive field) to filters of the same size, thereafter. The resulting output volume is called **feature map** or **activation map**. Then the filter slides over the next receptive field of the same input image by a defined stride and again the convolution between the new receptive field and the same filter is computed. Doing this for the whole input image provides the input for the next layer [82]**.** The convolution layer can be fine-tuned with respect to the following filter's hyperparameters:

1. **The stride:** It is the number of pixels with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially [83]**.**

2. **Filter size**: **filter** also known as **kernel** is a feature detector that aims to learn different portions of an input image. The filter slides left to right across the image from top to bottom and that moving filter is looking for, say, a dark edge. Each time a match is found, it is mapped out onto an output image.

    - **Filter size** refers to the dimensions of the filter or kernel in the CONV. It is an integer or list of 2 integers, specifying the height and width of the 2D convolution window.

3. **Number of filters:** also referred to as depth of output or number of output channels. It corresponds to the number of filters we would like to use for learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color [84].

4. **Padding parameter:** Is the amount of pixels added to each side of the boundaries of the input image when it is being processed by the kernel of a CNN. It is often used to keep the output image at the same width and height without decreasing the filter size, by adding padding to the original image with zeros and making a convolution slice through the image [83]**.**

Moreover, the output size of the feature map is given by the formula:

$$o = \frac{I-F+2P}{S} + 1 \tag{2.5}$$

Where I is the length of the input volume size, F is the length of the filter, P is the amount of padding and S is the stride.

After each CONV layer, a non-linear activation layer is applied to improve classification and the learning capabilities of the network. Common activation functions are the non-saturating ReLU function.

C. Pooling Layer

Pooling layers (POOL) are commonly present between two CONV layers. It is required to perform non-linear down sampling operations aimed at reducing the spatial size of the representation while simultaneously decreasing the number of parameters as well as the possibility of overfitting, and the computational complexity of the network. There are two types of Pooling: Max Pooling and Average Pooling.

Max Pooling returns the maximum value from the part of the image covered by the kernel as shown in **Figure 2.7**.

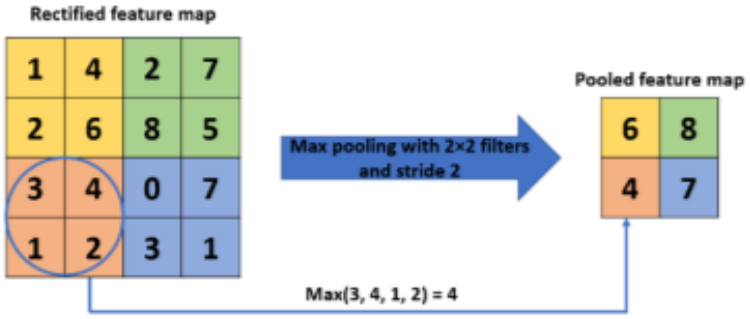

Figure 2.7: Example of Max-Pooling operation. *[85]*

On the other hand, **Figure 2.8** shows the Average Pooling which returns the average of all values from the part of the image covered by the kernel. Max Pooling can also act as a noise suppressor, it completely discards noisy activations, and also performs noise reduction and dimensionality reduction. Whereas, average pooling only performs dimensionality reduction as

a noise suppression mechanism. Therefore, it can be said that the max pool is better than the average pool [86].

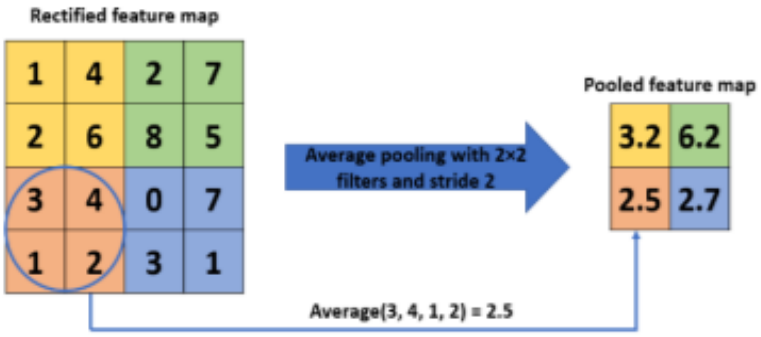

Figure 2.8: Example of Average pooling operation. *[85]*

D. Fully-Connected Layer (FC)

Fully-connected layers or Dense layers are usually the last hidden layers in a CNN architecture. It operates on a flattened input which are all the activations in the previous layer flattened with the **Flatten function** that converts the pooled feature map to a 1-dimensional array (a vector), where each input is connected to all neurons. FC is similar to the arrangement of neurons in a traditional neural network. Therefore, each node connects directly to the one level up and down. The fully connected layers can be used to optimize objectives such as class scores.

E. Output layer

The output CLASS layer performs the final classification in which the output of the last FC layer is fed into a softmax classifier. **SoftMax** is a commonly used activation function for the CLASS layer.

The softmax step can be seen as a generalized logistic function that takes as input a vector of scores $x \in \mathrm{R}n$ and outputs a vector of output probability $p \in \mathrm{R}n$ through a softmax function at the end of the architecture. It is defined as follows:

$$p = (p_1 \ldots p_n) \quad \text{Where} \quad p_i = \frac{xi}{\sum_{J=1}^{n} e^{Ji}} \quad \text{in the layer J} \tag{2.6}$$

1. VGG16

It is also called OxfordNet. It is a convolutional neural network architecture named after the Visual Geometry Group from Oxford [87]**.** The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. The most unique thing about VGG16 is that instead of having a large number of hyper-parameters it focuses on having convolution layers of 3x3 filter with stride 1 and always uses the same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC (fully connected layers) followed by a softmax for output.

2. AlexNet

AlexNet is a fundamental, simple, and effective deep convolutional neural network architecture [88]. It was released in 2012 by Alex Krizhevsky. Its architecture consists of eight layers: five convolutional layers and three fully connected layers, of which the maximum pooling operation is performed after three convolutional layers [89]**.**

In AlexNet, dropout is used to reduce the degree of over-fitting; in the training process of the model, neurons are stopped with a certain probability, thus the dependence on local nodes is reduced, and the generalization ability of the model is improved. The neurons which are "dropped out" do not contribute to the forward pass and do not participate in backpropagation. Furthermore, AlexNet uses ReLU as the activation function in the seven layers instead of the traditional sigmoid and tanh functions. This function is known for its effectiveness in accelerating the training time.

3. DenseNet

Densely Connected Convolutional Networks [90] (DenseNets) are the next step on the way to keep increasing the depth of deep convolutional networks. The DenseNets proposed in [HLMW17] employ the composition of three operations as weight layers. It simplifies the connectivity pattern between layers and requires fewer parameters than an equivalent traditional CNN, as there is no need to learn redundant feature maps. The Dense Convolutional Network (DenseNet) connects each layer to every other layer in a feed-forward fashion [91] **Figure 2.9** illustrates this layout schematically. Thus, for each layer, the feature maps of all the preceding layers are used as inputs, and its own feature maps are used as input for each subsequent layer. Hence, for L layers, there are $L(L+1)/2$ direct connections [92]**,** but do not sum the output

feature maps of the layer with the incoming feature maps instead it concatenates them [93]**.** Therefore, the number of inputs are quite high especially for later layers in the network. 1×1 convolution can be introduced as a bottleneck layer before each 3×3 convolution to reduce the number of input feature-maps, and thus to improve computational efficiency.

To facilitate both down-sampling in the architecture and feature concatenation DenseNets are divided into multiple densely connected DenseBlocks, where the dimensions of the feature maps remains constant within a block, but the number of filters changes between them**.** These layers between them are called Transition Layers and take care of the downsampling applying a batch normalization, a 1x1 convolution and a 2x2 pooling layers.

DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

However, one big advantage of DenseNets is their improved flow of information and gradients throughout the network, which makes them easy to train. Each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision [94]. This helps training of deeper network architectures. Further, we also observe that dense connections have a regularizing effect, which reduces overfitting on tasks with smaller training set sizes [95]**.**

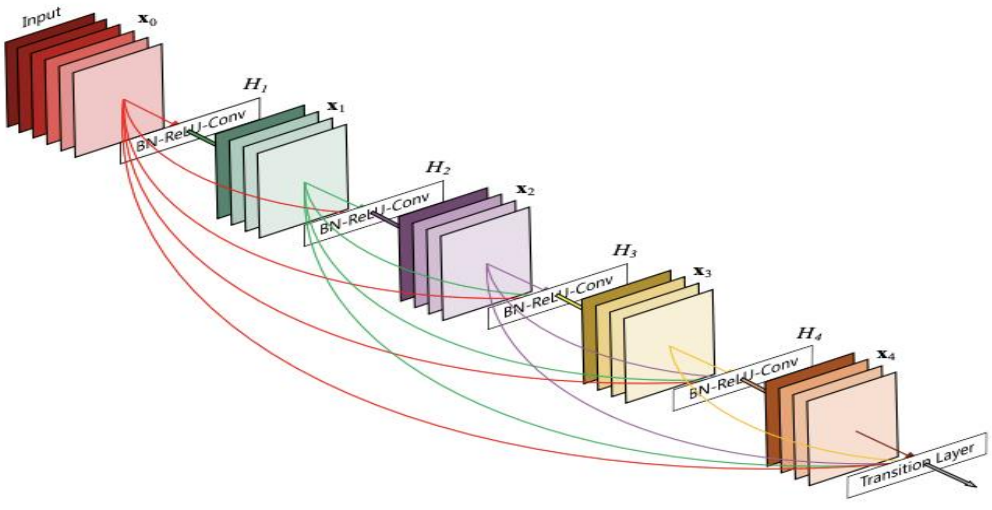

Figure 2.9: Illustration of DenseNet's architecture. *[96]*

2.4.4 Techniques for improving CNN performance

While we develop the **Convolutional Neural Networks** (CNN) to classify the images, it is often observed that the model starts overfitting when we try to improve the accuracy. Hence we list down the following techniques also called **regularization** techniques which would improve the model performance without overfitting the model on the training data.

1. Dropout

Deep learning architectures have a natural tendency to over-fitting especially when there is a little training data. Dropout is a simple but effective way to alleviate this problem [97]. **Dropout** is a technique where randomly selected neurons are ignored during training [98]**.** It means randomly switching some percentage of neurons of the network. When the neurons are switched off the incoming and outgoing connection to those neurons is also switched off. This is done to enhance the learning of the model. Dropouts are usually advised not to use after the convolution layers, therefore they are mostly used after the dense layers of the network [99]**.** For the acoustic scene classification task, the testing audio segment could be totally different from the used training audio segments due to the presence of background noise. Thus Dropout should be adopted to improve its robustness to generalize to variants of testing segments.

2. Data Augmentation

To prevent the problem of overfitting during training due to the small and enhance the robustness of our models. As we mentioned in the Machine Learning Pipeline section, data augmentation is a widely used method to overcome that problem. For image data a large variety of transformations can be used for augmentation. These include for instance random resized cropping, rotations, and flipping. However, not all of them make sense in the case of spectrograms. Thus, the common data augmentation techniques employed in the ASC task are:

A. SpecAugment

SpecAugment is an augmentation method that operates on the log mel spectrogram of the input audio, rather than the raw audio itself. It is a simple method that acts directly on the spectrogram as if it were an image, and does not require any additional data. Therefore we can

apply SpecAugment online during training. SpecAugment consists of three kinds of deformations of the spectrogram that will produce the illustration shown in **Figure 2.10**.

- **Time warping**: it is a deformation of the time-series in the time direction. Given a log mel spectrogram with τ time steps, we view it as an image where the time axis is horizontal and the frequency axis is vertical. A random point along the horizontal line passing through the center of the image within the time steps (W, τ − W) is to be warped either to the left or right by a distance W chosen from a uniform distribution from 0 to the time warp parameter W along that line. We fix six anchor points on the boundary—the four corners and the mid-points of the vertical edges.

The other two augmentations, inspired by "Cutout", proposed in computer vision [100]**:**

- **Time masking:** it masks the t consecutive time steps [t0, t0 + t), where t is first selected from 0 to the uniform distribution of the time mask parameter T, and t0 is selected from [0, τ-t) [101]**.**

- **Frequency masking:** it makes f continuous mel frequency channels [f0, f0 + f) masked, where f is first selected from 0 to the uniform distribution of the frequency masking parameter F, and f0 is selected from [0, ν − f), where ν is the number of mel channels [101].

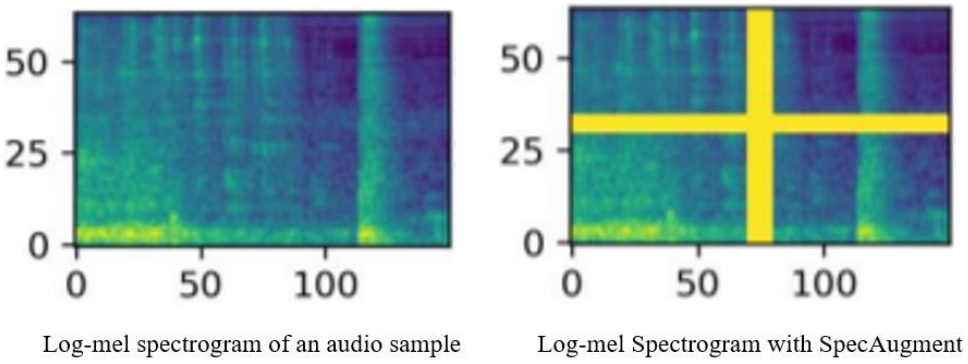

Figure 2.10: An example of Log-mel spectrogram before and after application of SpecAugment technique. *[102]*

B. Audio Deformation

A common strategy adopted to increase the quantity of training data, Audio deformation is a method that has been widely used as a data augmentation approach in signal processing; it artificially modifies the raw sound signal to a different tone or harmony of sound but preserve the semantic meaning of the signal. The different techniques of audio deformation are:

· **Time Stretching (TS)**: It slows down or speeds up the audio sample (while keeping the pitch unchanged).

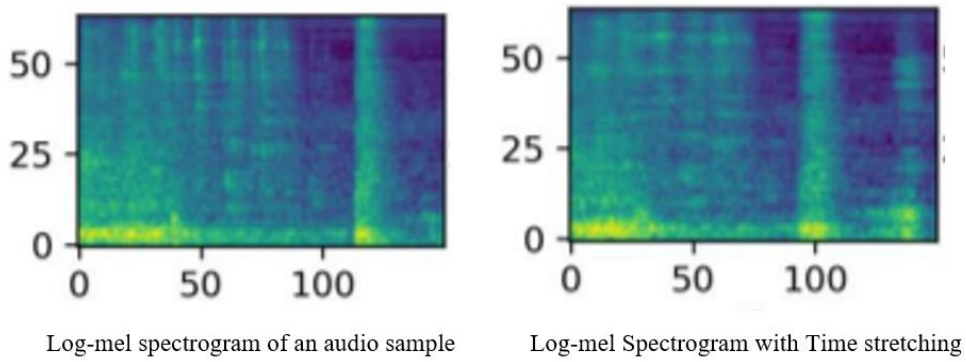

Figure 2.11: An example of Log-mel spectrogram before and after application of Time Stretching technique. *[102]*

· **Pitch Shifting (PS2):** It raises or lowers the pitch of the audio sample (while keeping the duration unchanged).

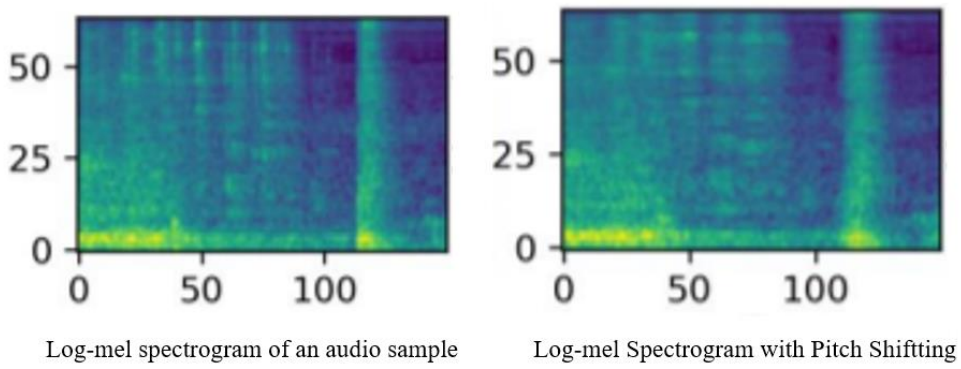

Figure 2.12: An example of Log-mel spectrogram before and after application of Pitch Shifting technique. *[102]*

- **Dynamic Range Compression (DRC)**: It compresses the dynamic range of the sample using 4 parameterizations.

- **Background Noise (BG):** It mixes the sample with another recording containing background sounds from different types of acoustic scenes.

C. MixUp

Mixup is a popular regularization technique for training deep neural networks that can improve generalization and increase adversarial robustness. It perturbs input training data in the direction of other randomly-chosen instances in the training set [103]**.** Mixup generates weighted combinations of random image pairs from the training data. Given two images and their ground truth labels: (xi,yi),(xj,yj), a synthetic training example (x^,y^) is generated as:

$$x^{\wedge} = \lambda xi + (1 - \lambda)xj \tag{2.7}$$

$$y^{\wedge} = \lambda yi + (1 - \lambda)\, yj \tag{2.8}$$

Where *xi* and *xj* are two randomly sampled input points, and *yi* and *yj* are their associated one-hot encoded labels.

Mixup has two ways of working: one is to use the synthetic sample as a new sample. The other is to use the synthetic sample as a linear combination of the original sample. The former view shows that mixup uses linear interpolation of different classes of samples to generate new samples. Different linear interpolation will produce different new samples, which makes the neural network have more opportunities to sample new data to avoid overfitting.

Mixup trains a neural network on convex combinations of pairs of examples and their labels. By doing so, mixup regularizes the neural network to favor simple linear behavior in-between training examples [104]. It is proven to be a better suited augmentation method for audio-based inputs than previously mentioned transformations, due to the temporal nature of audio signals and the physical meaning of the spectrum over the horizontal and vertical axes [105].

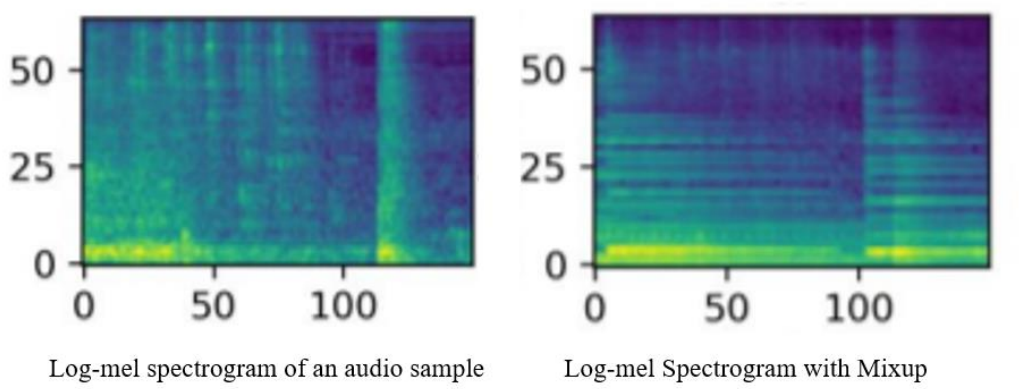

Figure 2.13: An example of Log-mel spectrogram before and after application of Pitch Shifting technique. *[102]*

3. Batch Normalization

Batch normalization is a layer that allows every layer of the network to do learning more independently [106]. It is used to normalize the output of the previous layers [107]. Using batch normalization, the learning becomes efficient. Also, it can be used as regularization to avoid overfitting of the model [108]. It can be used at several points in between the layers of the model

but it is often placed just after defining the sequential model and after the convolution and pooling layers.

2.5 Conclusion

In this chapter we introduced the machine learning process and its various categories as well as the numerous steps of this process's pipeline and techniques for the ASC task. We have also presented the subfield of machine learning which is known as deep learning, including some neural network-based architectures that are widely used for ASC such as Convolutional neural network's architectures.

CHAPTER 3: Design of a Workbench For Acoustic Scene Classification

3.1 Introduction

In the previous chapters, we introduced audio signals and a variety of its characteristics. We defined the fundamental concepts behind sound manipulation, and the definition of **Acoustic Scene Classification** task. We have also presented notions of **Machine Learning**, including its numerous representations and processing techniques as well as an overview of the machine learning pipeline for acoustic scene classification. We have then introduced **Deep Learning,** the subfield of machine learning along with its approaches. Also we provided an overview of the various types of neural networks such as **Convolutional Neural Network** and its different architectures which is an important aspect in our experiment. In this chapter, we present the description of our ASC system. To gain an understanding of the functions that the workbench should perform or facilitate, a workbench description is given. This description will introduce the components that are to be included, and is used as a use-case to identify different parts of the workbench.

3.2 System Architecture

The aim is to build a workbench that helps the users to exploit their audio dataset using different learning techniques for the task of classifying acoustic scenes. Then, allows them to evaluate and compare the achieved accuracy and loss for each technique, with offering the user the possibility of visualizing the data after each change made by the altering of the various settings of data transformation, data augmentation. The workbench is composed of several modules. **Figure 3.1** shows the structure of these modules. It contains 5 modules corresponding to each stage of the Acoustic scene classification task. The user-interface for these modules allows the user to make changes in the parameters used during the execution.

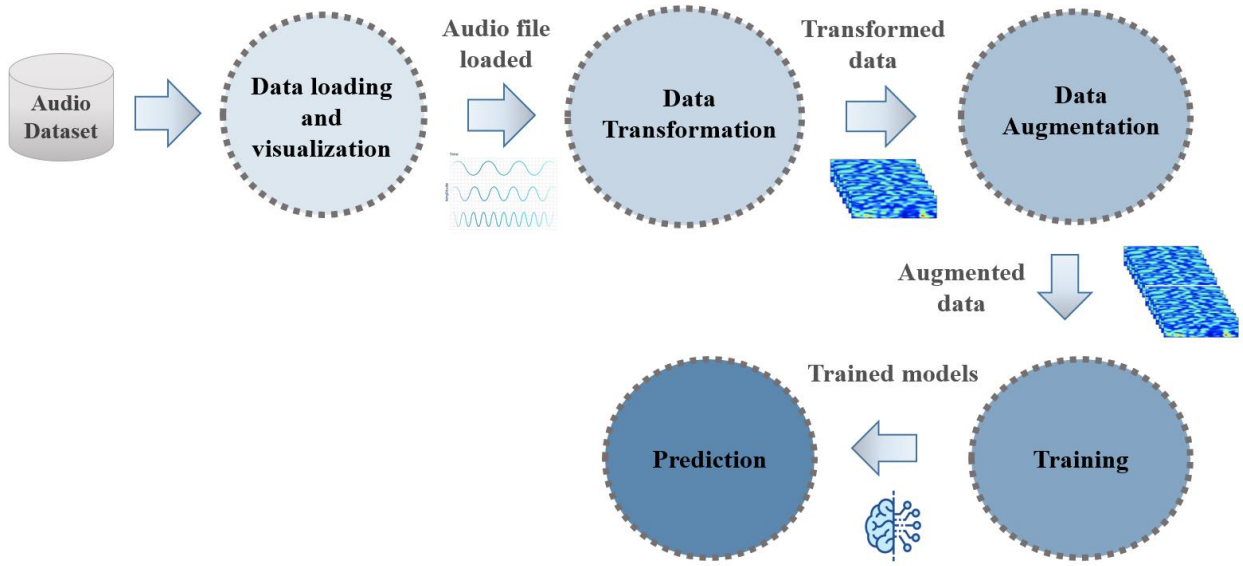

Figure 3.1: The Workbench's Modules.

To better understand the procedures that the system performs, a description of the workbench is provided in this section. This description introduces the workbench's modules and components that are included, as well as the terminology that is able to be used. This description is used as a user guide to identify the different parts of the workbench.

First, the workbench has a form of a graphical user interface, which provides procedures with different options for the users to manipulate and display the final results. The parts that manipulate the foremost interactions with the user are the parts called functions. Each module consists of several functions for every task, such as **"loading a dataset", "plotting the results**"... etc. The modules communicate with one another via the workbench framework. This framework provides to store data which will be used by modules for the latter aim. The interactions between modules provides a data structure to manage the information that each module can use.

Because of the modularity structure, additional features are often added later without having to rewrite the workbench. It should even be possible for others to increase the workbench with relative ease by adding other functionality and extending the workbench without modifying all the code due to the modular structure, and all in a very easy way.

The general method to work with the workbench is defined by moving through the modules. The first module **Data Loading and Visualization** performs the data acquisition. Afterward, the features extraction part is processable in the **Data Transformation** module

where it will be applied on the imported data. Next, in the third step, the transformed data is going to be augmented in the **Data Augmentation** module. This will increase the size of the dataset that will be later fed to the learning model for the training purpose. The training of the model step is processed in the **Training** module. It provides at its end numerous results such as the accuracy and loss that the model achieved. The last step of the process is the **Prediction** module. It is related to the training module as it uses the trained models to make a prediction and visualize the results of its performance.

3.3 Data loading and Visualization module

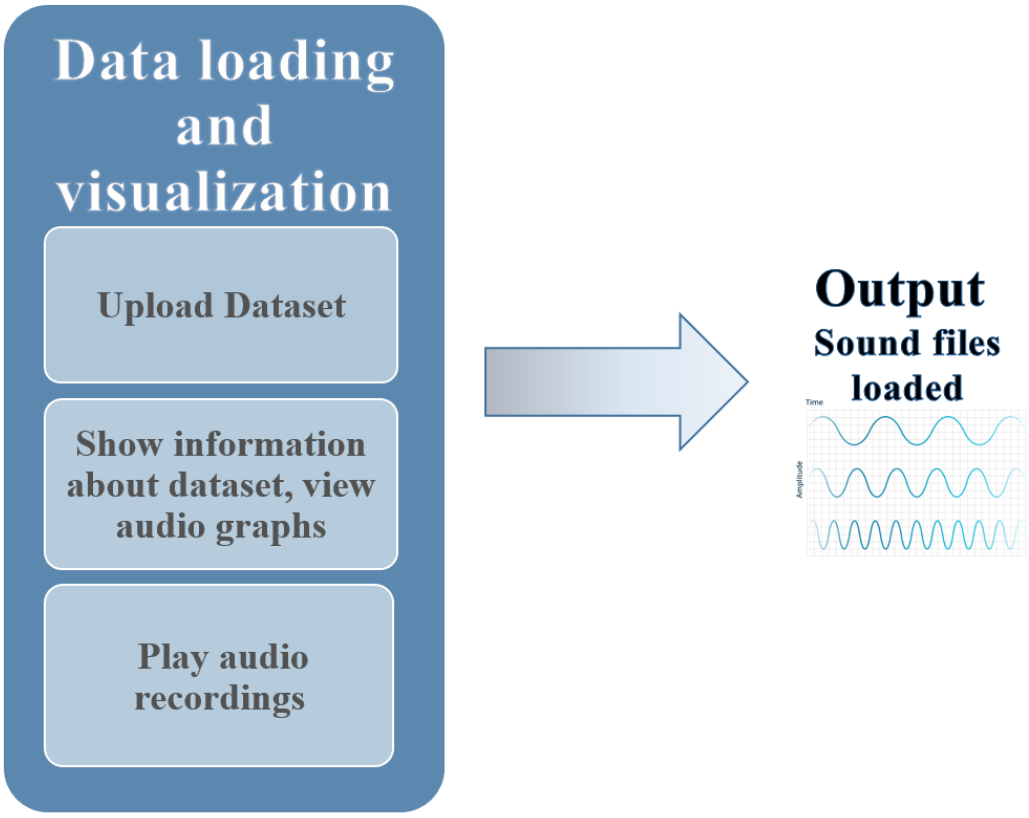

Figure 3.2: Data loading and Visualization module process.

Figure 3.2 illustrates the different functions that can be done in this module. The first step is to specify a dataset to import into the system. The user can choose to import either a one development dataset or a training and testing set by browsing the path of the file of this data. The dataset should be a set of audio recordings, which the recordings are defined with their

name and so the class label of the audio (the scene). Afterward this data will be viewed and tabulated to provide the user with basic tools to better understand and gain an insight in this data. Therefore, the user could possibly have an overview on some information about it such as number of samples and the max and min duration among the audio recordings, and so the ability of playing a recording or even plotting the time domain and frequency spectrum for any audio chosen from the dataset.

3.4 Data Transformation module

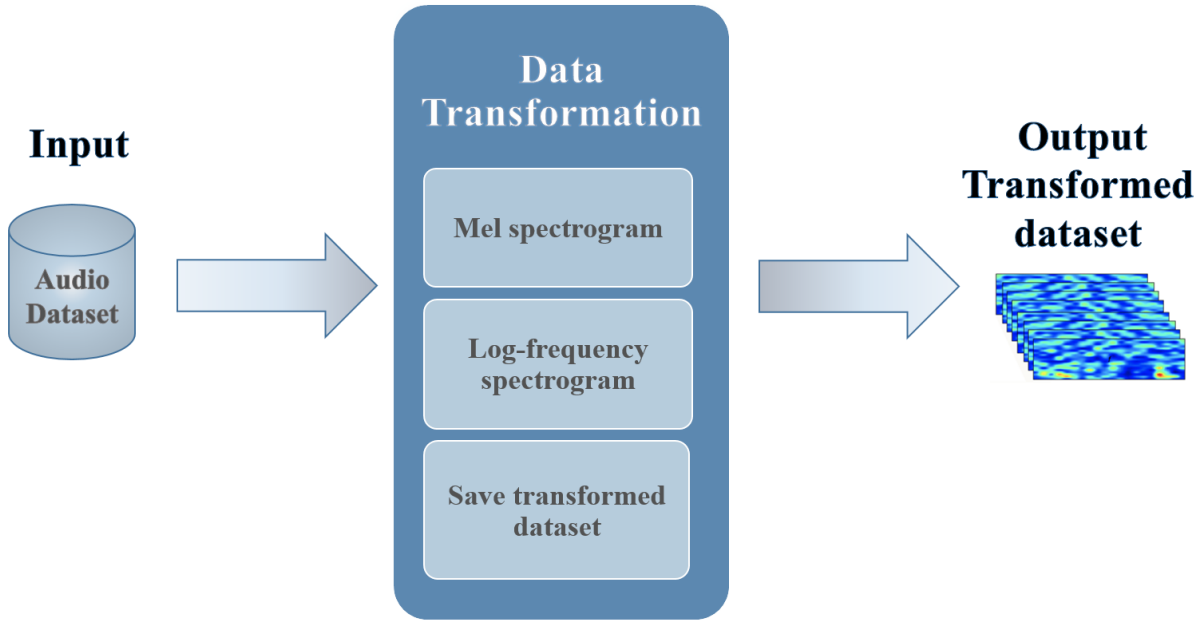

Figure 3.3: Data Transformation module process.

The next module is to extract the characteristics from the data by transforming the audios into a spectrogram in order to be processed by the subsequent steps. The process of the conversion is divided into two choices depending upon the selection of the user.

- The first one is executed if the user chooses to use the creation of transformed data, the row data is going to be converted into a mel spectrogram or a log frequency spectrogram depending on the user choice as well for the purpose of extracting features from that data. After that there are few parameters that the user has to set for the conversion process and it is related to the type of spectrogram that the user has chosen. Thence, the system provides

the user different options to check up the modifications of conversion on the dataset that have been done with the given parameters. One of these options is the ability of visualizing a spectrogram of a selected audio from the recording list. Whereas, the other option is saving the transformed dataset as a file of spectrogram representations after choosing the path of this file. In that case this transformed dataset can be used directly later through the second choice of this module. Therefore, the dataset that has been converted is passed to subsequent steps, for modules that require the transformed data like input.

- The second choice is executed if the user chooses to use an already transformed dataset so he uploads it into the workbench. The source of this dataset is a regular file on the hard disk containing a set of spectrogram representations. Before that, the user also has to specify the type of the converted data whether it is Mel spectrogram or log Frequency spectrogram representations. At the end of this process, the converted data is going to be transmitted to subsequent steps to be utilized in the data augmentation part.

3.5 Data Augmentation module

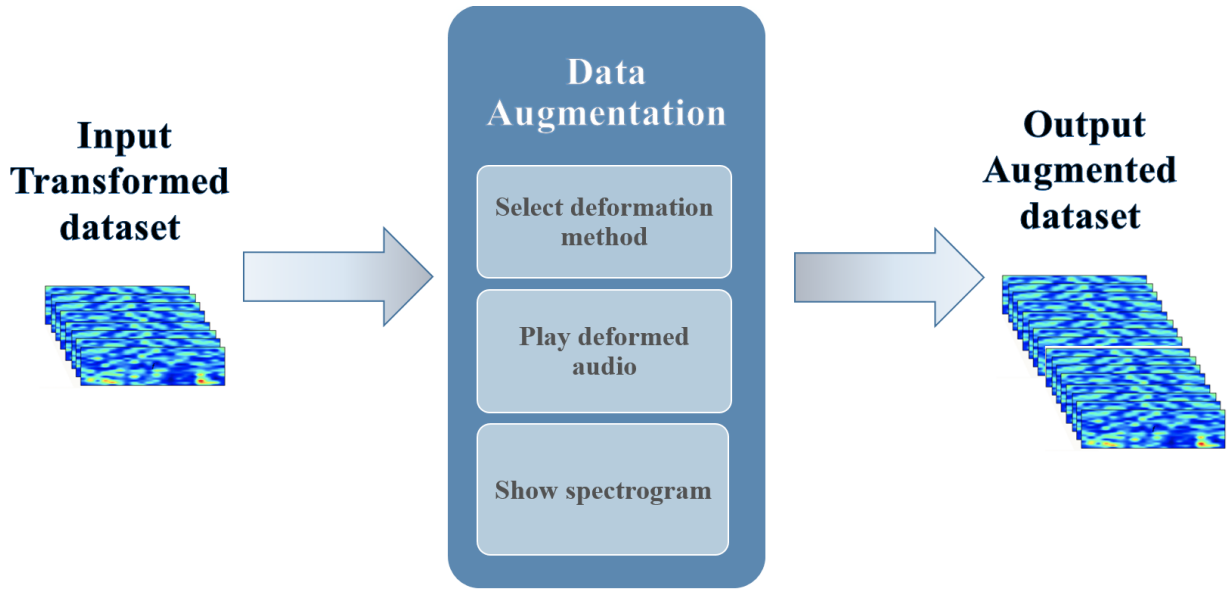

Figure 3.4: Data Augmentation module process.

In order to provide a better performance of the model that requires a large dataset for training to be sufficient in learning the problem, and to prevent an overfitting problem, the application of one or more deformations on the set of transformed data must be done. Hence, the workbench provides this function in the form of a module that includes various options

illustrated in **Figure 3.4**. To start the process of data augmentation, the row dataset along with the previously transformed one is imported into this module. The dataset is displayed as a list of recordings. Then as a first step to fulfill this process, the techniques of DA must be selected from those which exist in the workbench such as pitch shifting, time stretching and frequency masking. Each one of these methods has its own parameters that should be set by the user. The chosen methods will be then applied on the dataset providing the user abilities for observing the results of these deformations by showing the spectrums of a selected audio recording from the list, or allowing him to play and hear the changed audio that is selected from the audio list by the user as well. And as a last step for this module, the user is able to save the parameters he entered to be easily used in the next execution of this module. At the end, this module produces an augmented dataset that will be fed to the model in the next module.

3.6 Training module

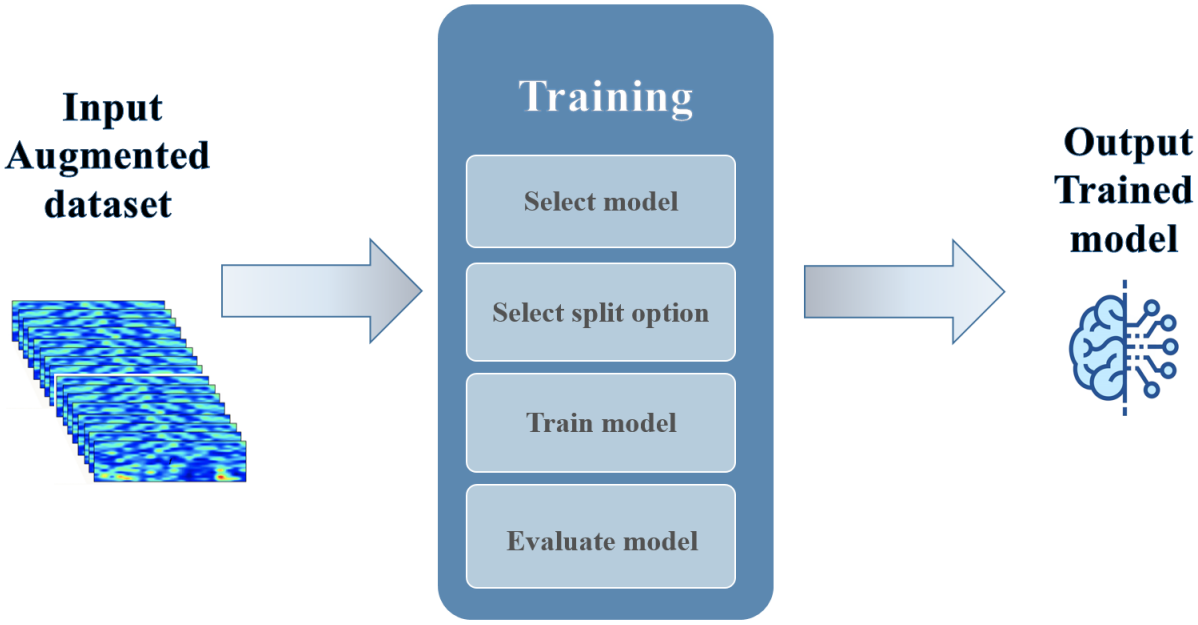

Figure 3.5: Training module process.

Now that the new dataset is ready to be exploited as an input to the model, moving to the training module which consists of several functions as shown in **Figure 3.5**. The user can choose in this module one of the existing models along with setting some of its hyper-parameters. The first step that must be done by the user in this module is setting the parameters related to CNNs architectures since the workbench works with CNNs as models, those parameters include:

Setting the kernel size, selection of an optimizer to change the attributes of the neural network such as weights and learning rate to reduce the losses, selection of a loss function to compute the error for a single training example. The number of training examples should be set also as a batch size parameter and so the number of epochs.

After setting all the parameters, the next step is to choose one model from the list. The workbench adopts four CNN architectures, three of them are pre-trained CNN including AlexNet, VGG16 and DenseNet. The architecture for each one is shown in the tables below respectively, **Table 3.1**, **Table 3.2**, **and Table 3.3**. The fourth CNN architecture is imported as a file that the user has in his computer. This module also allows the user to have an overview of the architecture for the chosen model.

The next step is to choose one splitting method to apply on the dataset in order to divide it into a training set and testing set so that it can be fed to the neural network. The methods available in the workbench are: split by percentage where the user must specify a percentage number of the training set, and k-fold cross validation where the user enters the number of folds *k*. Afterward, in the training step the model starts the learning process on the training set given while calculating the loss for each epoch and displays this progress in a text area.
Moreover, the training module provides an option for the user to add another training process by entering the number of epochs desired and start running again. This option improves the performance of the trained model by making it learn more.

The last step in the current module is evaluating the model performance. This can be accomplished by observing several results such as training accuracy and loss and validation accuracy and loss achieved at the end of training; those results are plotted in a graph form of the estimated accuracy/loss over the epoch number. In the case of K-folds cross validation, the accuracy and loss plots are evaluated for each fold severally.

Table 3.1: AlexNet Architecture.

AlexNet

Input 227x227x3
Conv1 55x55x64 (stride 4)(padding 2)+ReLu
Max Pooling 3x3 stride 2 padding 0
Conv2 27x27x192 (stride 1) (padding 2)+ReLu
Max Pooling 3x3 stride 2 padding 0
Conv3 13x13x384 (stride 1)(padding 1)+ReLu
conv4 13x13x256(stride 1)(padding 1)+ReLu
conv5 13x13x256 (stride 1)(padding 1)+ReLu
Max Pooling 3x3 stride 2 padding 0
FC (out_features 512)+ReLu
Dropout 0.5
FC (out_features 512)+ReLu
Dropout 0.5
FC (out_features 3)
Output Softmax

Table 3.2: VGG16 Architecture.

VGG16
Input 224x224x3
2x Conv 224x224x64 (stride 1)(padding 1)+ReLu
Max Pooling 2x2 stride 2 padding 0
2 Conv 112x112x128 (stride 1)(padding 1)+ReLu
Max Pooling 2x2 stride 2 padding 0
3 Conv 56x56x256 (stride 1)(padding 1)+ReLu
Max Pooling 2x2 stride 2 padding 0
3 Conv 28x28x512 (stride 1) (padding 1)+ReLu
Max Pooling 2x2 stride 2 padding 0
3 Conv 14x14x512 (stride 1)(padding 1)+ReLu
Max Pooling 2x2 stride 2 padding 0
FC (out_features 512)+ ReLu
Dropout 0.5
FC (out_features 512)+ ReLu
Dropout 0.5
FC (out_features 3)
Output Softmax

Table 3.3: DenseNet Architecture.

DenseNet
Input 224x224x3
Conv1 224x224x3 stride 2+ReLu +BN
Maxpool 2x2 stride 2
Dense block 1, 6 57x57x96
Transition_layer,1 57x57x384
Dense block 2, 12 29x29x192
Transition_layer,1 29x29x768
Dense block 3, 36 15x15x384
Transition_layer,1 15x15x2112
Dense block 4, 24 8x8x1056
BN + ReLu
Global Average Pooling 2d 8x8
2D FC , Softmax

3.7 Prediction module process

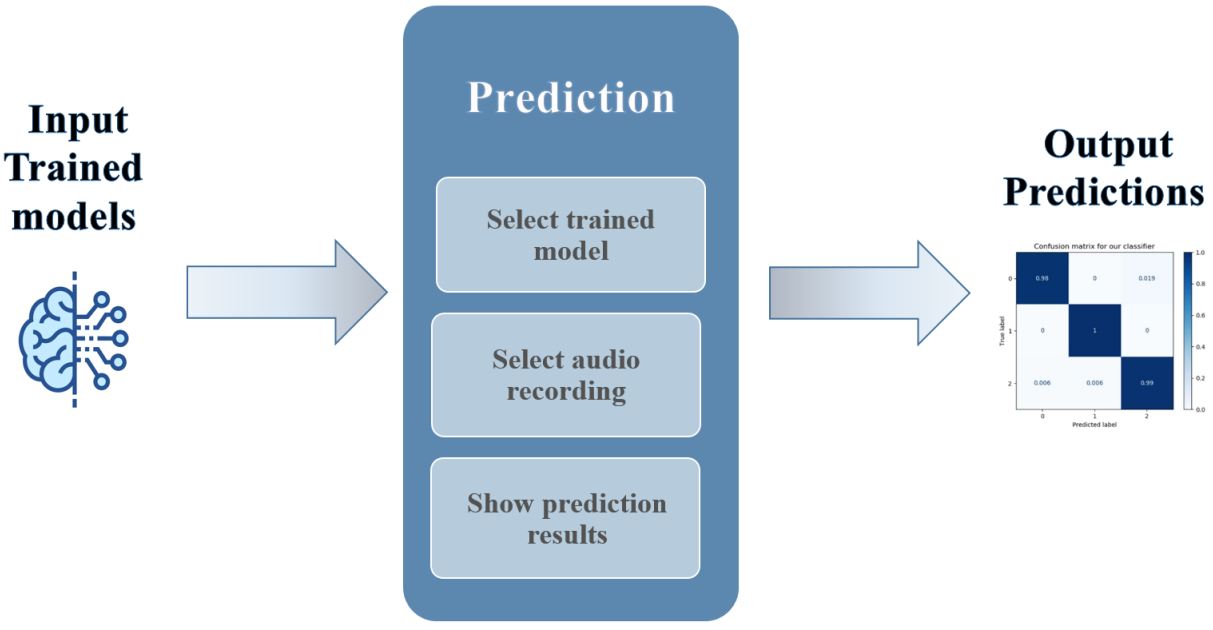

Figure 3.6: Prediction module process.

In order to make a prediction about the class of any chosen recording, and to check out the performance of the trained model and evaluate it, the workbench contains a Prediction

module as a last step for the user to explore. In this module, a list of all the trained models is exhibited for the user to select which one he is willing to employ in this process. Then, by choosing the audio recording from the dataset file, the model chosen starts the prediction process by applying the patterns associated with past outcomes which are obtained from the learning process to the new audio. When the prediction of the audio class is done, the workbench displays the corresponding results which are: the achieved accuracy from this process as well as the probabilities estimated for the 5 first classes predicted with providing a graph plot of these classes and probabilities.

The workbench also grants the user the ability to see the performance of the model through consulting the confusion matrix and a table of the achieved accuracy for all classes.

3.8 Conclusion

In this chapter, we introduced the workbench designed for this work, and presented a brief description for the different modules that constitute this workbench, and so the numerous techniques that can be used in it. In the next chapter, we will address the setup used in building our workbench and show its graphical interface. We will also present the dataset used to train our ASC system and discuss the result obtained with it.

CHAPTER 4: Implementation Of The Workbench

4.1 Introduction

We saw in the previous chapter the numerous modules constituent of our workbench as we described the adopted major steps and functions in it for the ASC task. In this chapter, we define the dataset employed for executing the workbench along with the setups and libraries used in building it. We also present an overview of the workbench's interface accompanied by screenshots of it that illustrates the way of using its modules and the results obtained in each module.

4.2 Data acquisition

Dataset: In our experiment we used the TAU Urban Acoustic Scenes 2019 dataset (35.6GB), which consists of recordings from the following 10 various acoustic scenes as classes: airport, indoor shopping mall, metro station, Pedestrian Street, public square, street with medium level of traffic, travelling by tram, travelling by bus, travelling by underground metro, and urban park. Those recordings were in 10 different cities for each class. The whole dataset contains 40 hours of audio, balanced between classes and of 14400 segments in total divided into 144 segments per city per acoustic scene class. For each recording location there are 5-6 minutes of audio. The original recordings were split into segments with a length of 10 seconds that are provided in individual files.

However, to get a better performance for our system, we have done some modifications to this dataset by reducing its size by removing the recordings of all the scenes from 5 cities out of 10 so that the execution would take a shorter time. Therefore, our new dataset contains 5 cities which are: Lisbon, London, Barcelona, Helsinki and Lyon. And the total number of segments is 7192.

The recording process: The recordings were done with (Soundman OKM II Klassik/studio A3, electret binaural microphone and a Zoom F8 audio recorder using 48kHz sampling rate and 24 bit resolution which guarantees a high audio quality. The microphones are like headphones, being worn in the ears so that the recorded audio is very similar to the sound that reaches the human auditory system of the person wearing the equipment.

4.3 Libraries

1. **NumPy:** In the Python world, NumPy (Numeric Python) arrays are the standard representation of numeric data and allow efficient implementation of numeric calculations in a high-level language. The performance of NumPy is usually improved through three techniques: computing vectorization, avoiding copying data in memory, and minimizing the amount of operations [109]**.** The NumPy package provides basic routines for manipulating large arrays and matrices of numeric data [110]**.**

2. **SciPy:** The SciPy (Scientific Python) package extends the functionality of NumPy with a substantial collection of useful algorithms, like minimization, Fourier transformation, regression, and other applied mathematical techniques [110]**.**

3. **Pandas:** It is a Python library of rich data structures and tools for working with structured data sets common to statistics, finance, social sciences, and many other fields. The library provides integrated, intuitive routines for performing common data manipulations and analysis on such data sets [111]. It aims to be the foundational layer for the future of statistical computing in Python. Pandas serves as a strong complement to the existing scientific Python stack while implementing and improving upon the kinds of data manipulation tools found in other statistical programming languages such as R. [111]**.**

4. **Scikit-learn:** Scikit-learn (Sklearn) is the most useful and powerful library in Python machine learning. It provides a series of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction Consistent interface in Python. This library mainly written in Python is built Based on NumPy, SciPy and Matplotlib [112]**.**

5. **Matplotlib:** Matplotlib is a library for making 2D plots of arrays in Python [113]. It can be used in a Pythonic, object oriented way. Although Matplotlib is written primarily in pure Python, it makes heavy use of NumPy and other extension code to provide good performance even for large arrays [113]. Matplotlib is designed with the goal of creating simple plots with just a few commands, or just one with many options for customizing the color, thickness and style of lines, the size, shape, and annotation of axes, the composition of a figure with multiple plots, and other features needed for the

presentation of publication-quality graphics [114]**.** In our task we used it for plotting the 2D representations of the audio signal.

6. **PyQt5:** It is a library that lets you use the Qt GUI framework from Python. Qt itself is written in C++ [115]. By using it from Python, you can build applications much more quickly while not sacrificing much of the speed of C++ [115]. Pyqt5 is the Graphical User interface widget toolkit that is known as one of the most powerful and popular python interfaces. It is a combination of the Qt module and python programming [116]. It is a free open-source widget toolkit that can be implemented on cross-platform application development.

   Qt Designer: Qt Designer is Qt's tool for designing and building graphical user interfaces (GUIs) from Qt components [117]. It is a program supplied with the Qt package already installed which allows us to draw our windows visually and connect it to the code. And moreover, Qt Designer allows us to modify the properties of widgets, to use layouts, and make the connection between signals and slots.

   The main advantage of using qt designer is to save time and avoid the tasks of repetitive writing of the window generation code.

7. **PyTorch:** PyTorch is a library for Python programs that facilitates building deep learning projects [118]. It emphasizes flexibility and allows deep learning models to be expressed in idiomatic Python through it is clear syntax, streamlined API, and easy debugging which make it an excellent choice for introducing deep learning. PyTorch provides a core data structure -the tensor- which is a multidimensional array that shares many similarities with NumPy arrays [119]. Around that foundation, PyTorch comes with features to perform accelerated mathematical operations on dedicated hardware [120], which makes it convenient to design neural network architectures and train them on individual machines or parallel computing resources [121]**.**
8. **Librosa:** It is a powerful Python library built to work with a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems [122]**.** Librosa is used for uploading an audio file into the system, retrieving various features from it such as: pitches, amplitudes, tempo and so on. The features we have used this package to extract in our experiment are log-frequency spectrograms and mel-spectrograms.

4.4 Implementation Setup

Our workbench graphical interface consists of a collection of items and widgets like text boxes, buttons, labels and multimedia elements like audio player, created with Qt 5.12 version of Qt Designer. The graphical interface is connected directly with the code of the workbench specifying the functionality of each item. The code was written on Spyder Notebook with python 3.7. The implementation and testing of the workbench requires a powerful pc to improve its performance and get a better showing. Therefore, we have conducted these tasks on a windows 10 operating system and a pc that has the following hardware characteristics:

- An AMD Ryzen 5 3600 Wraith Stealth processor with a Max Boost frequency of 4.2 GHz, 6 core, 12 threads and a Wraith Stealth cpu cooler. It provide us a fast and smooth performance.
- The B450 Tomahawk AM4 ATX Motherboard from MSI with four memory slots that can hold up to 64GB of DDR4 RAM running at up to 3466 MHz .
- 8 Gb of RAM
- NVIDIA GeForce GTX 1650 GPU based on Turing architecture Built-in 128-bit GDDR6 memory interface with 4GB buffer memory and 8gbps memory speed.

4.5 Workbench graphical interface

In this section we will have an overview of our workbench GUI and we provide an explanation for each existing module.

4.5.1 Dataset loading and Visualization

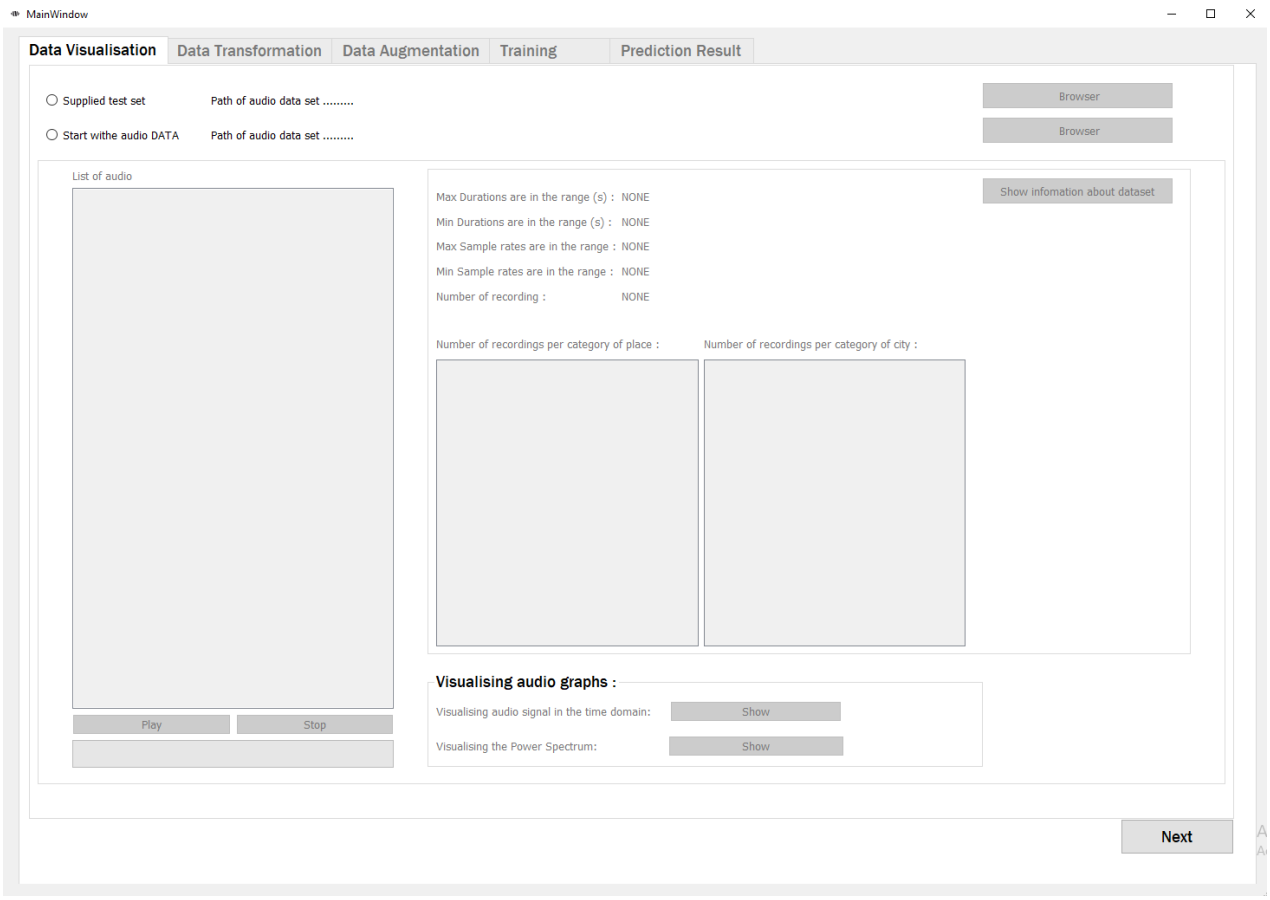

Figure 4.1: Workbench's main window.

To begin, once the user opens the workbench all the packages and libraries needed in the whole process will be imported and accessible; our system does not need an internet connexion to work.

Figure 4.1 shows the main window of the graphical user interface, it contains 5 tabs on the top corresponding to each module. First tab is of data visualization module. This tab presents a panel on the top of it for uploading the dataset willing to exploit. It consists of two options, first one is for importing a splitted dataset (training and testing sets), by selecting the radio button "**Supplied test set**" with a button for browsing its path labeled "**Browser**". The second option is by selecting the radio button "**start with audio data**" for importing one development dataset with the second "**Browser**" button that browse the path of that dataset. The selected item should be a folder of audio recordings. Then that path will appear in the label beside the radio button selected in the same panel.

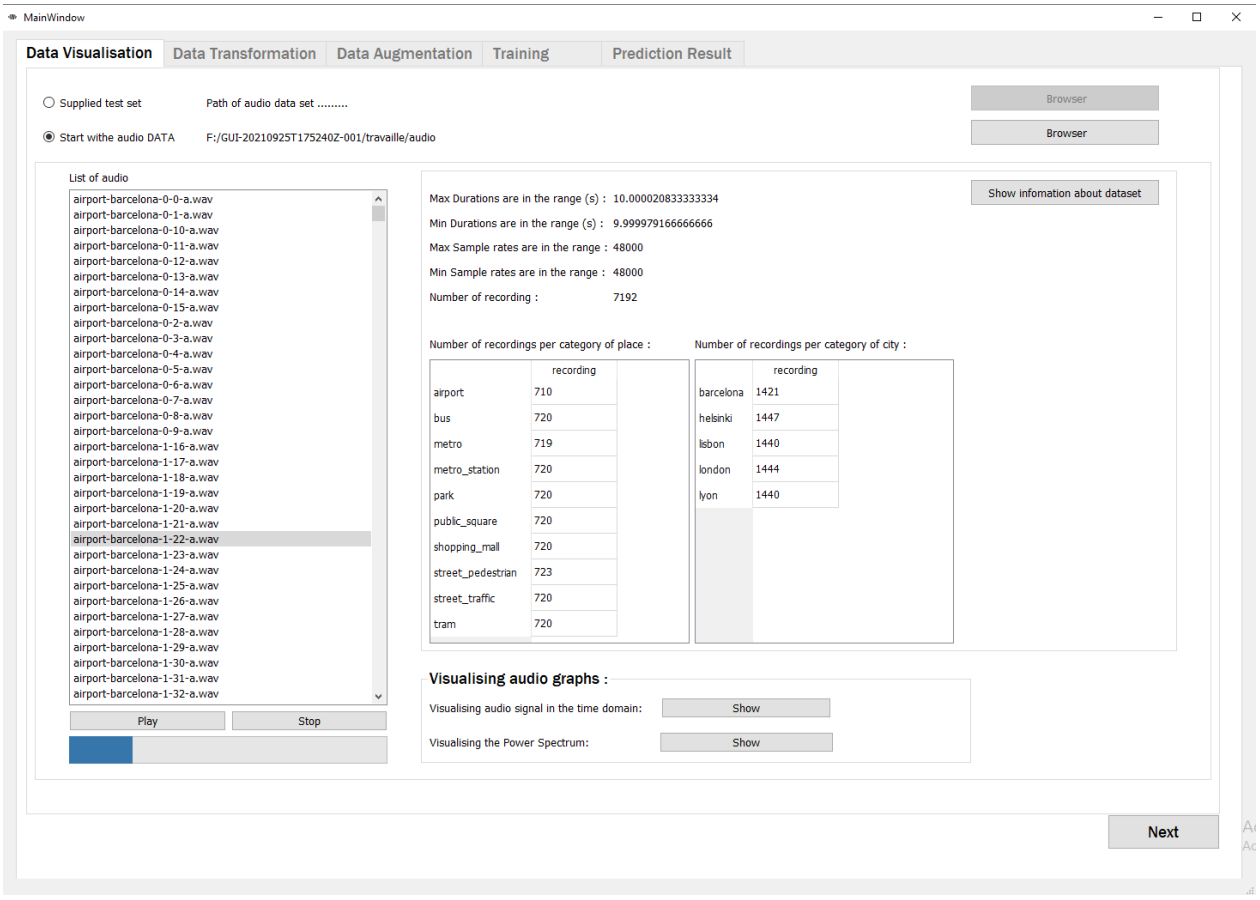

Figure 4.2: Tab 1 "Data Visualisation".

Having loaded the file, the screen will be as shown in **Figure 4.2.** The panel at the bottom displays information on the currently imported dataset. Once the user click on the "**Show information about dataset**" button, a playable list of audio is displayed in the left side which allows the user to listen to any recording from the dataset by selecting it and pressing the "**Play**" button. The user can stop the player by clicking the "**Pause**" button. Also, he can control the volume of the sound through the horizontal slider beside the buttons. Whereas, in the right side of the panel appears some information about the dataset:

Max durations of all recordings and **Min durations of all recordings** which are computed by our system and return the maximum or minimum value over the specified axis (here we set the "duration" axis) in the dataframe.

Max and Min sample rate, it is the number of samples of audio recorded every second and is also calculated in the same method as the max and min durations.

And lastly, the **total number of recordings** in the dataset, the system returns the number of rows and columns in the dataset.

Two tables appear under these information representing the number of recordings per category, one table is categorized by cities and the other table by scenes. This helps to compare between the ratios of classes and to see how the dataset is divided per each category.

On the other hand, the user can see the time domain and the frequency domain plots of a selected audio from the list by pressing the "**Show**" buttons for each plot in the "**visualizing audio graphs**" panel, the plots will be displayed in new small window as the **Figure 4.3** illustrate. It is possible to display several plots of different recordings at the same time to compare and see the differences between them.

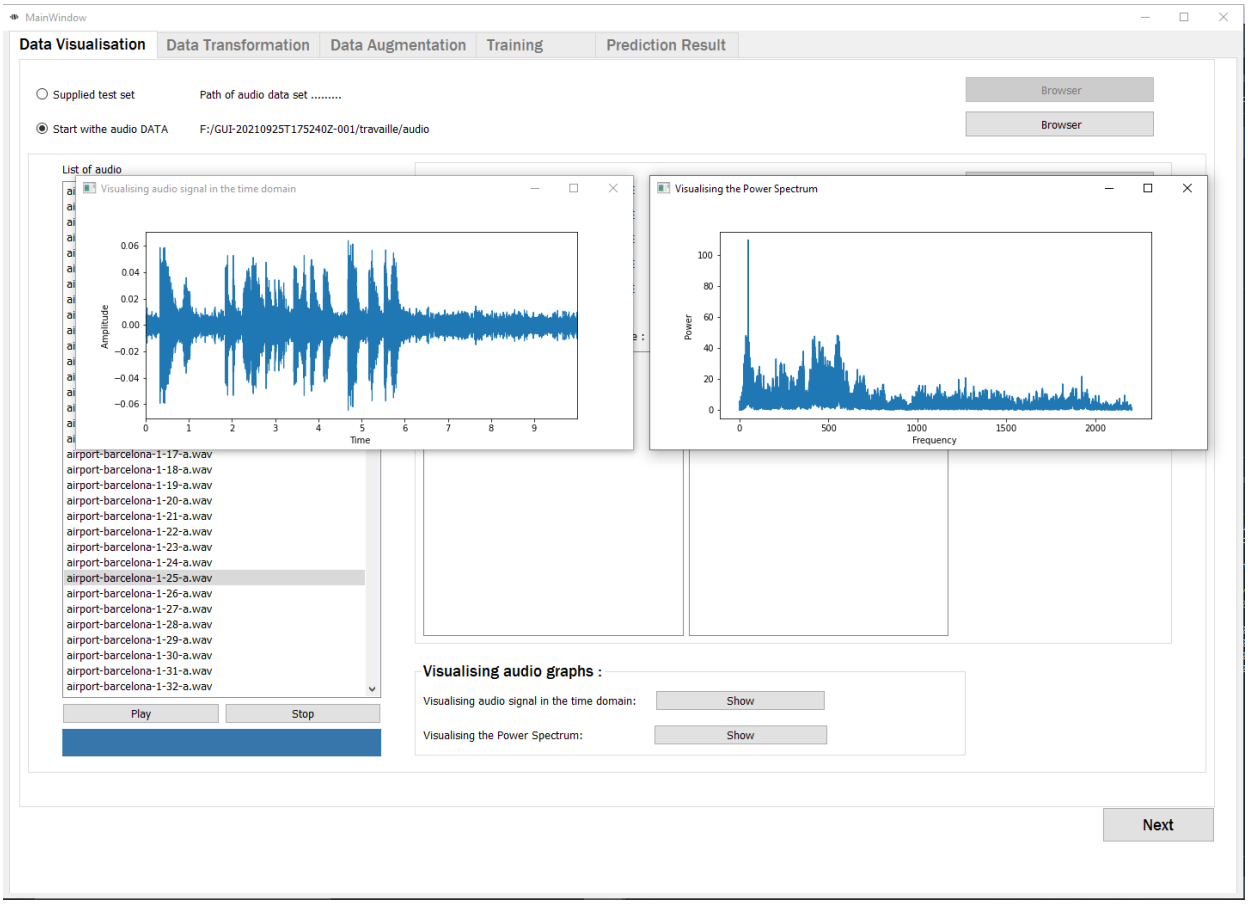

Figure 4.3: Visualizing audio graphs (Time-domain and Frequency domain).

Then lastly, clicking the "next" button will lead to the second tab "data transform" that aims to manipulate the dataset's recordings and extract its metadata.

4.5.2 Data Transformation

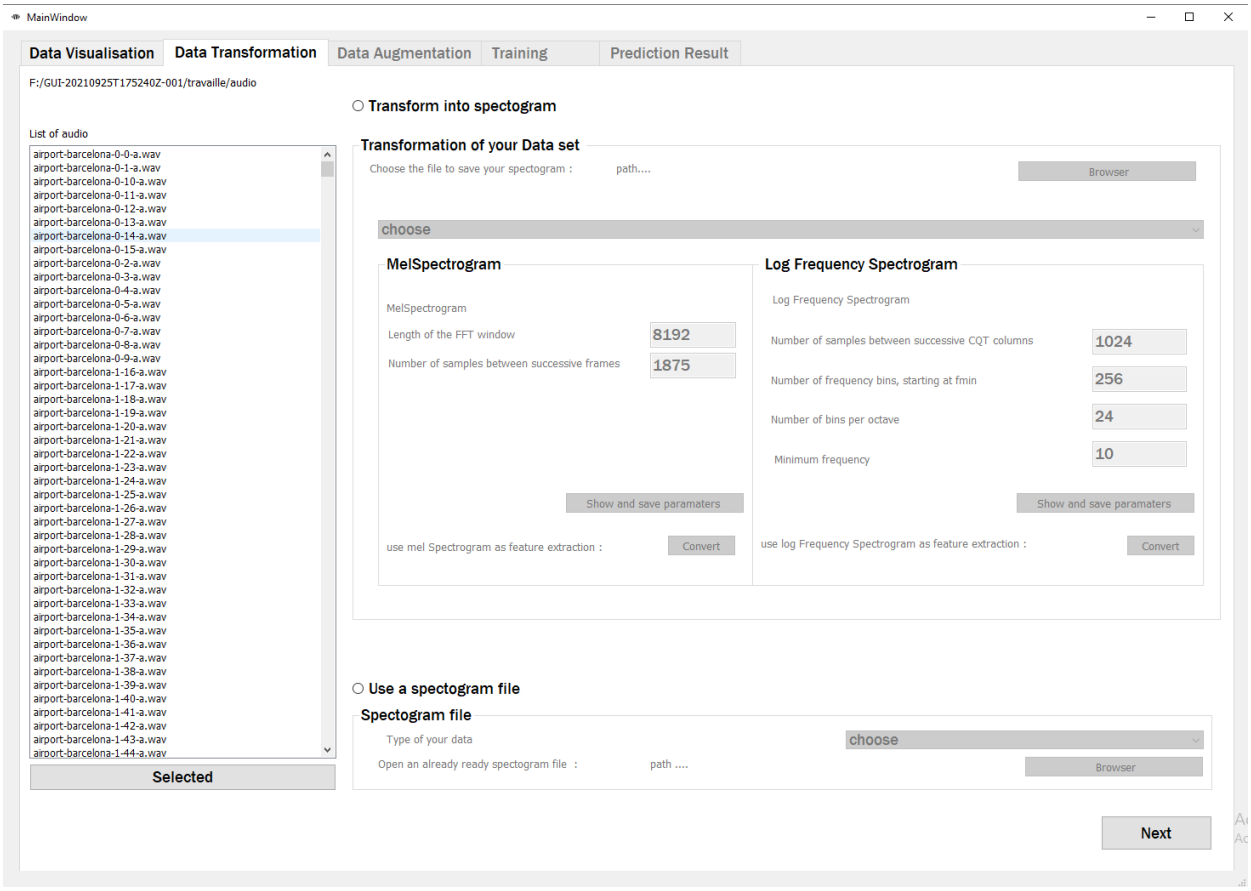

Figure 4.4: Tab 2 "Data Transformation".

After the user loads the data into the system, it is now ready to be analyzed within the "Data Transformation" Tab.

This tab is composed of two adjacent panels as shown in the **figure 4.4**. The left panel resembles the one in the first tab which displays the playable list of recordings for the purpose of selecting an audio which the user wants to see its feature's representation later after the transformation. The right panel presents the functionality of the features extraction process. It consists of two options in the form of two radio buttons. The selection of one of them will enable the functions inside the group-box under it. The first one on the top "**Transform into spectrogram**" provides the different functions to convert the raw data into the chosen feature forms. It contains on the top a browsable button "**Browser**" for choosing a file in which the features will be saved. The file's path will appear beside the button right after. Under it, there is a drop-down box of the existing features in our system to work with. The list has two choices: **Mel spectrogram** and **Log-frequency spectrogram**. The user can select one of them to be used and convert all recordings into it. After selecting one, he gets the ability of altering various parameters related to the spectrogram chosen by enabling the edit option for those parameters.

If the user chooses a Mel-spectrogram, the screen shown will be as in **Figure 4.5**. The parameters he must enter are:

- **Length of the window:** it is the size of the window in samples, meaning how many samples are in one window, thus the sum of all window lengths equal to the total number of samples of the audio signal segment. And moreover, that total number of samples if divided on this window length will give **the number of frames**. Number of frames are how many frames/segments the system should divide the signal into for computing then the Fourier transform of each frame separately to get the frequency spectrum over time.
- **Hop-length:** it is the number of samples between successive overlapped frames in the windowing procedure of the STFT approach, to prevent the loss of information of the signal.

The system uses those parameters in the building of a mel-spectrogram.

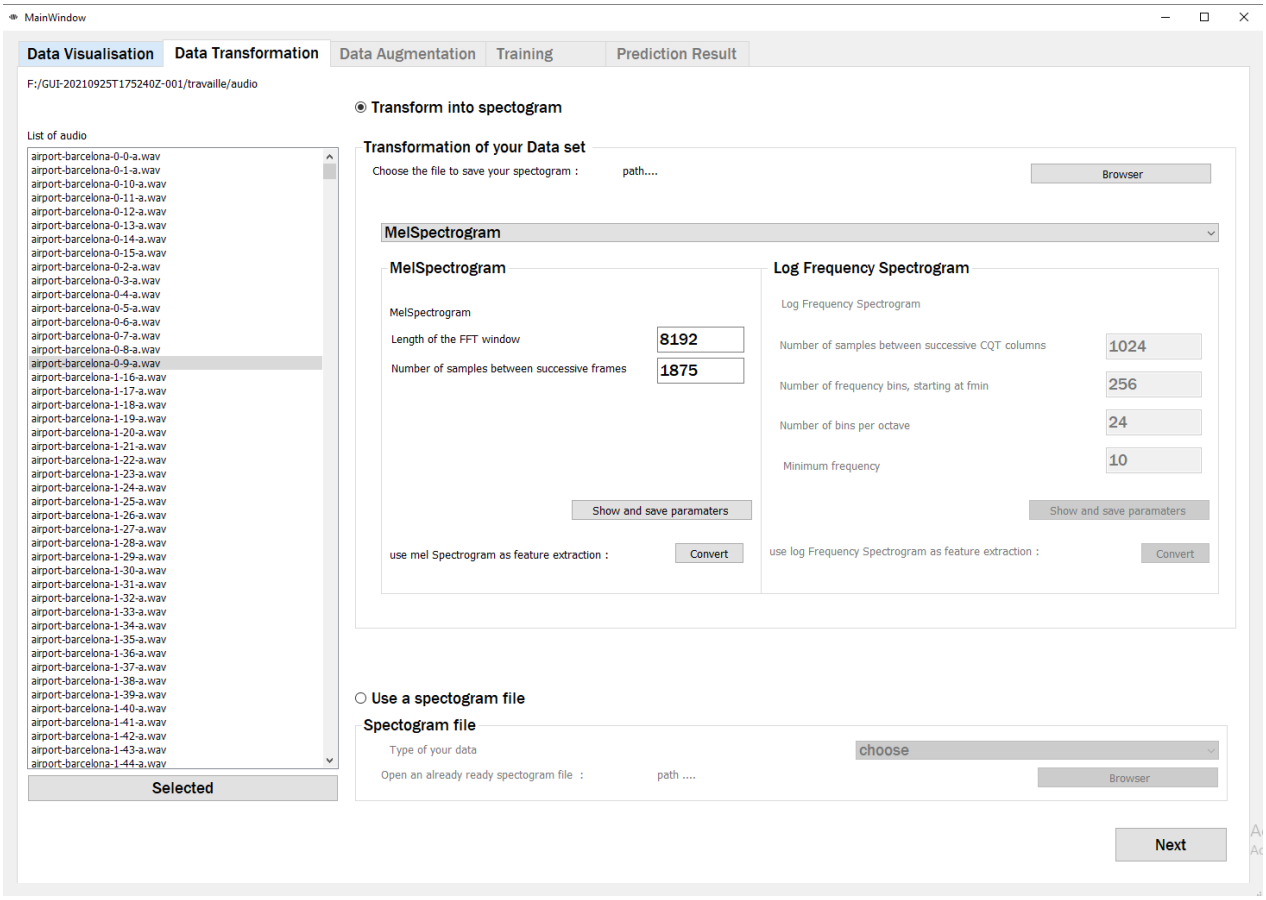

Figure 4.5: Transformation to Mel spectrogram option.

While, if the user chooses the log-frequency spectrogram, its producing function requires several parameters, the ones which the user must enter are as shown in **Figure 4.6**:

- **Hop-length** as of the mel-spectrogram.
- **N_bins (starting at f-min):** Number of frequency bins, starting at f-min.
- **Bins per octave:** Number of bins per octave.
- **F-min:** is the Minimum frequency. Defaults to C1 ~= 32.70 Hz.

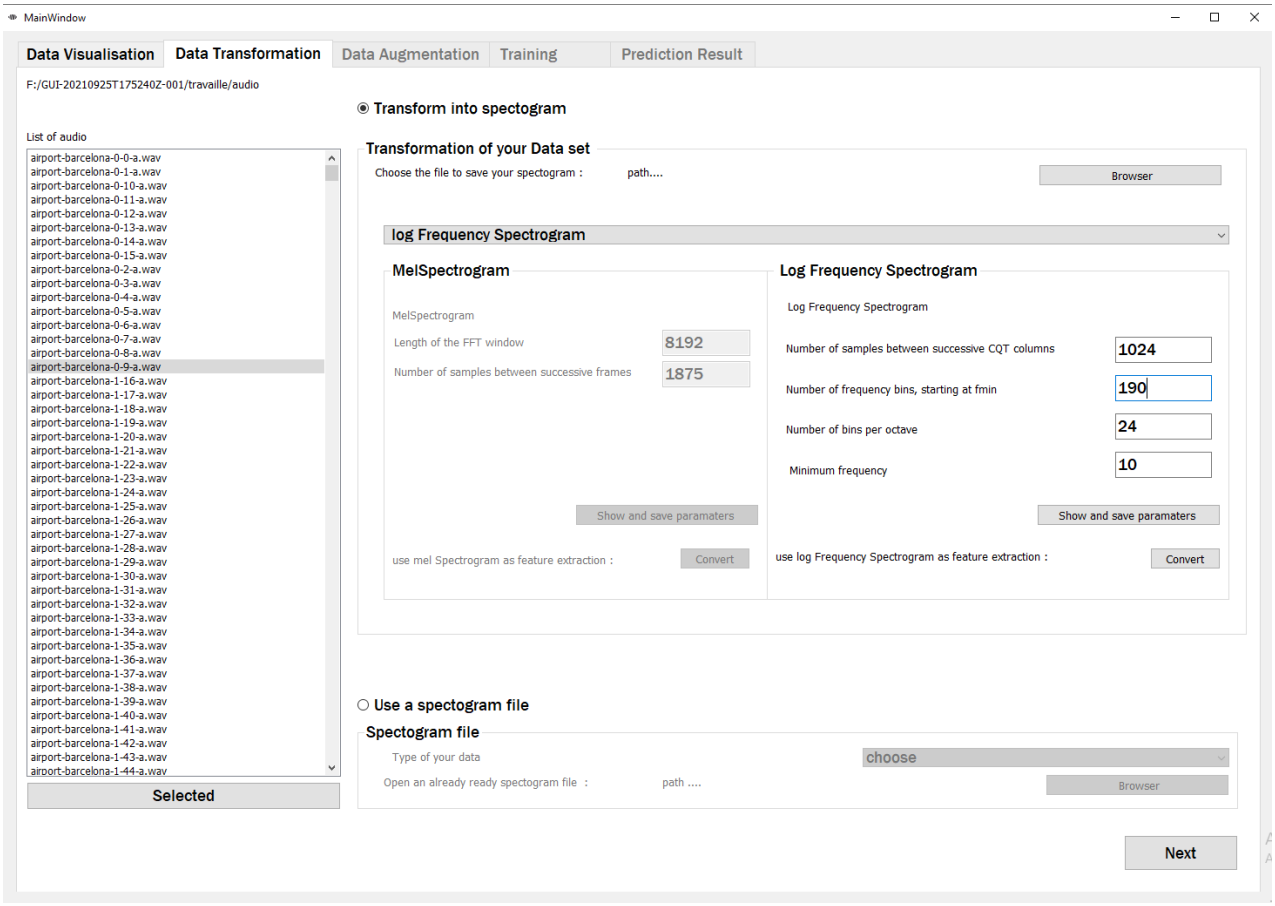

Figure 4.6: Transformation to Log-frequency spectrogram option.

After that, the user will be able to display in a new window the spectrogram plot for any selected recording from the list of audio by clicking on the "**Show and save parameters**" buttons corresponding to the selected spectrogram type. **Figure 4.7** and **Figure 4.8** show an example of the Mel-spectrogram and Log-frequency spectrogram for a chosen audio respectively. Then by clicking on the "**Convert**" button, the system conducts the conversion process and produces the spectrograms of the recordings. Afterward those spectrograms of data recordings will be saved in the chosen file path.

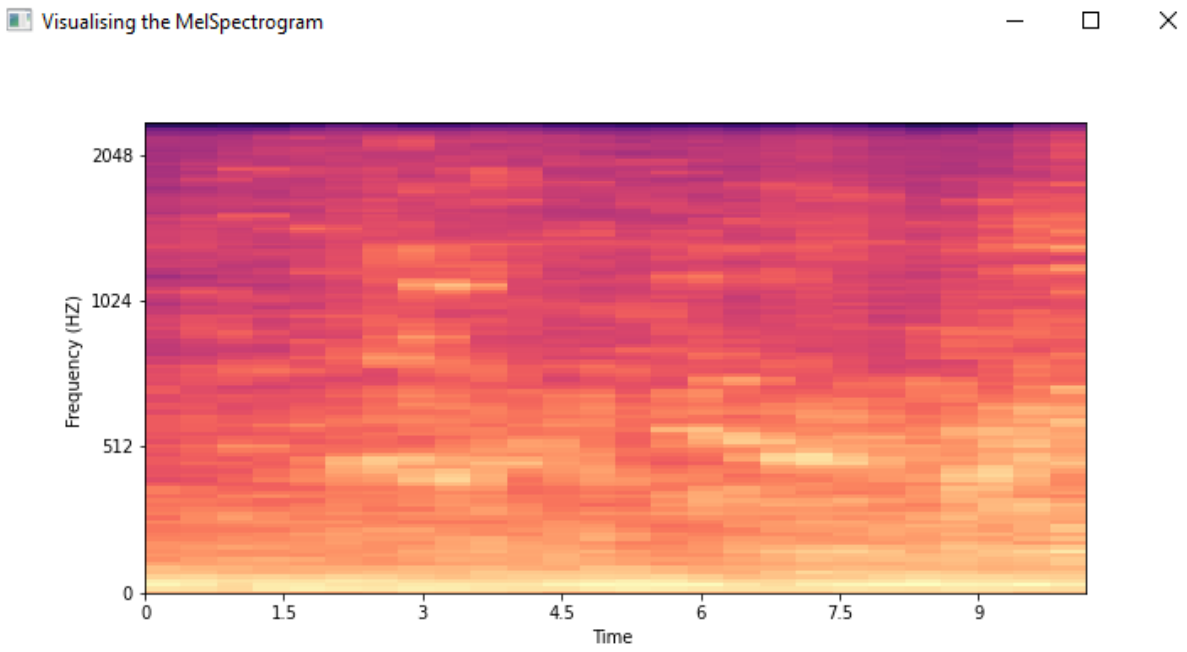

Figure 4.7: Mel spectrogram for a Barcelona airport recording.

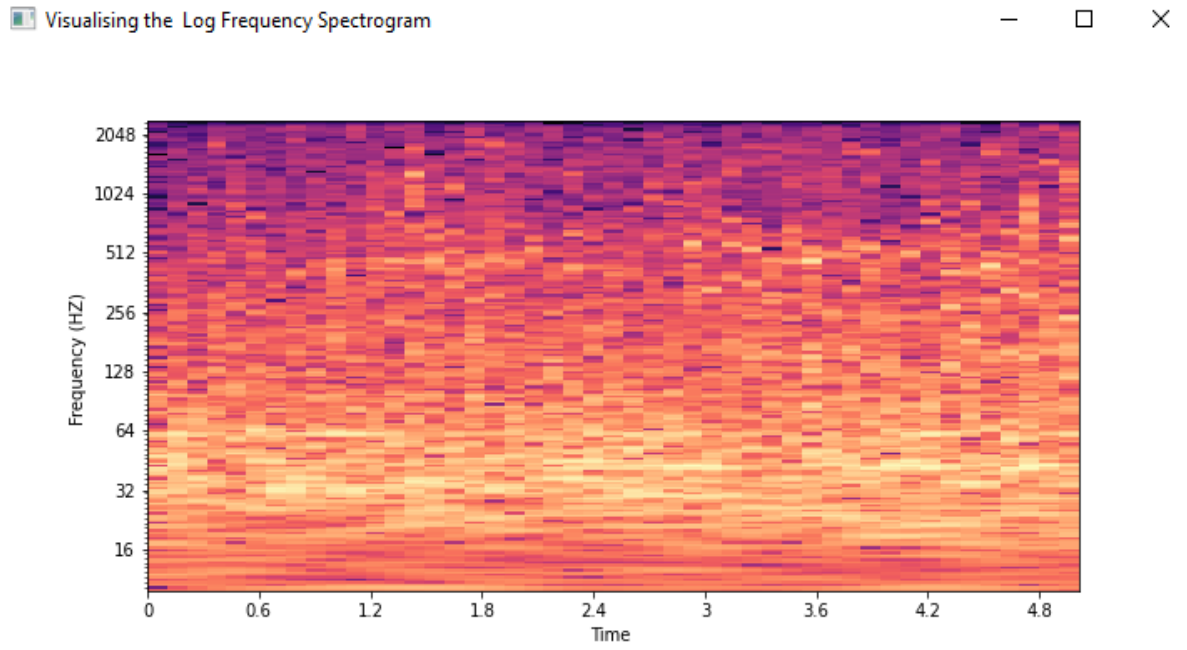

Figure 4.8: Log-frequency spectrogram for a Barcelona airport recording.

Another way to fulfill this stage of feature extraction is to load an already saved spectrogram's file. To do this, the user must select the second radio button "**Use a spectrogram file**", then choose the type of spectrogram he has from the drop-down box in the bottom as

shown in **Figure 4.9**, after making the choice, it is now ready to import the file by clicking on the "**File- browser**" button and choose its path.

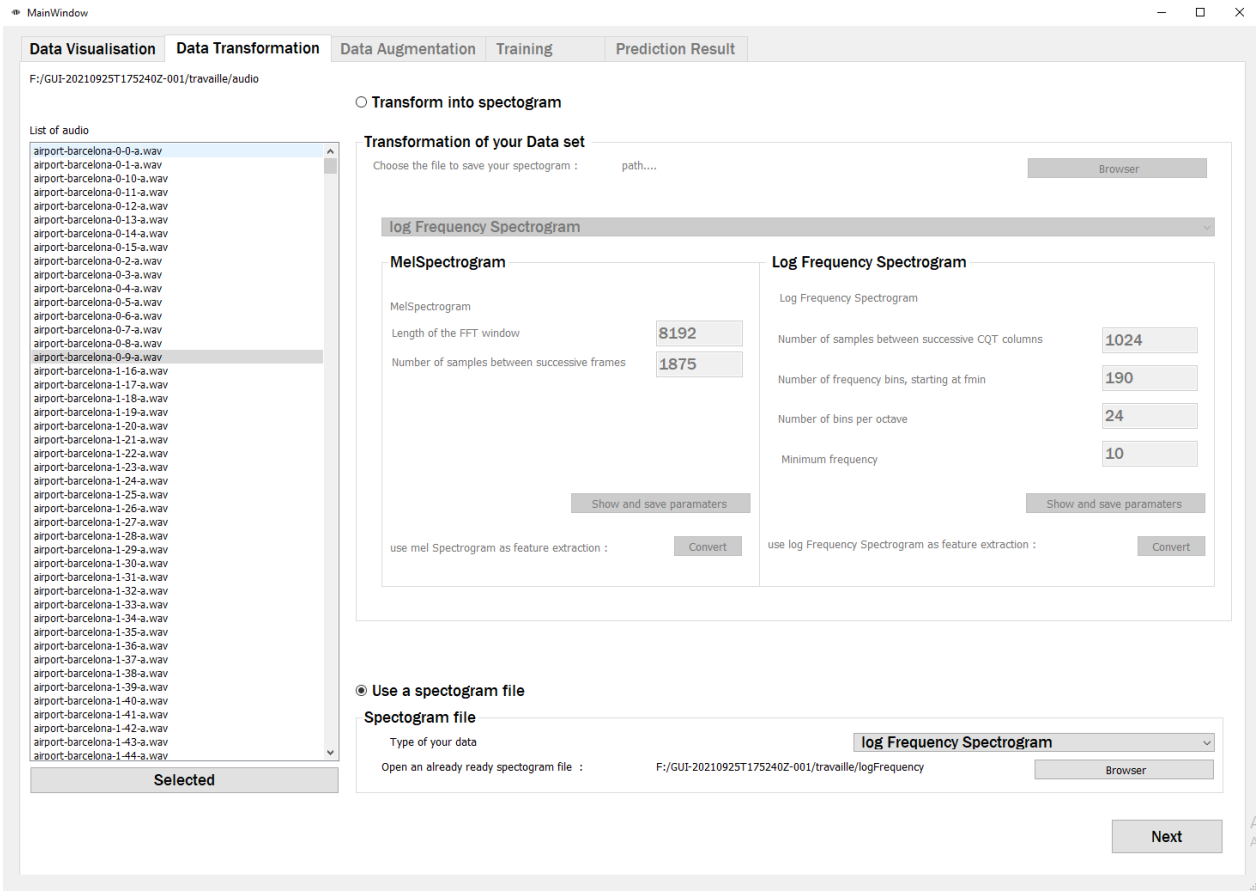

Figure 4.9: Import a spectrogram file option.

The "**Next**" button at the end leads the user to the next tab and next stage of the process under the name of data augmentation.

4.5.3 Data Augmentation

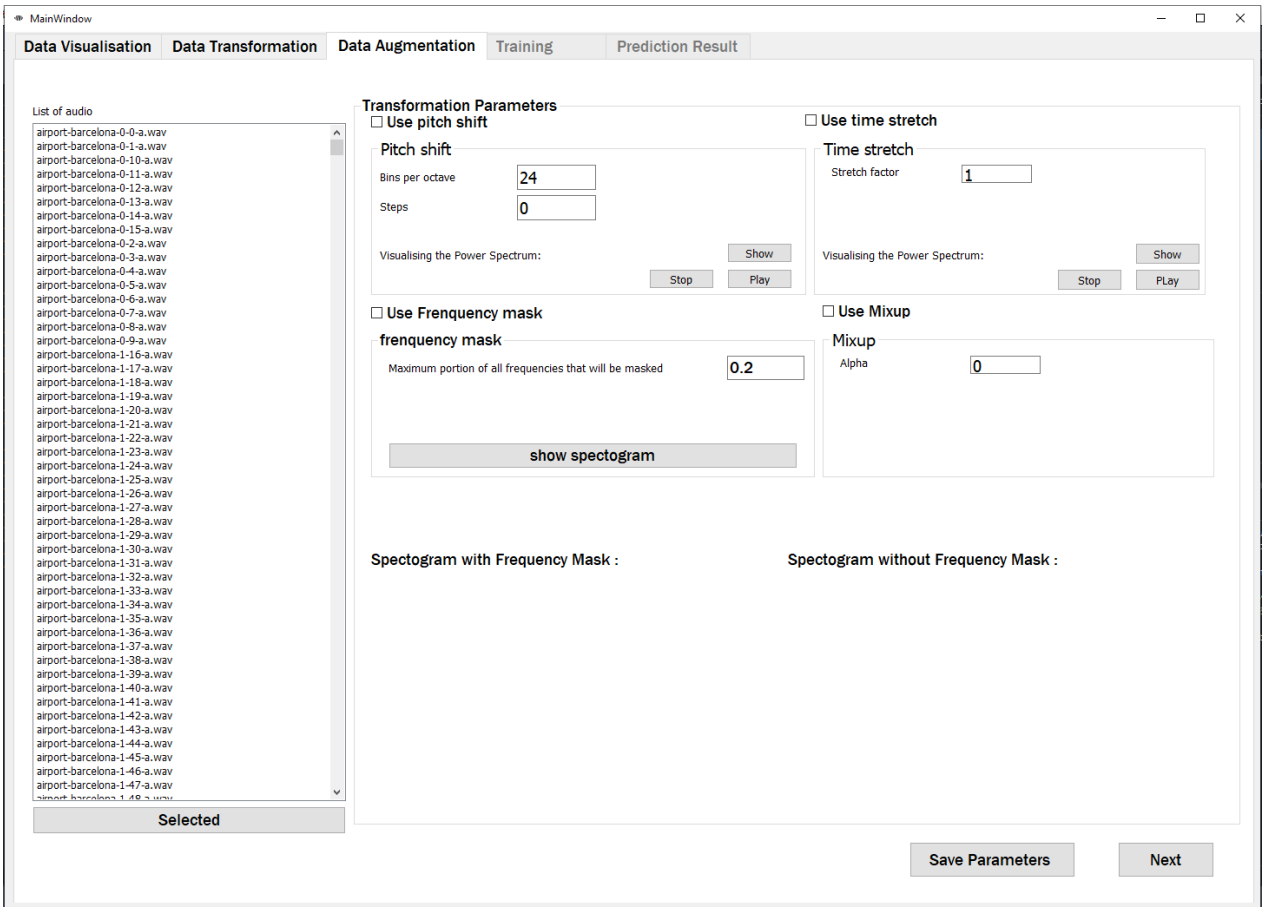

Figure 4.10: Tab 3 "Data Augmentation".

Data augmentation is a common approach that has been implemented in order to increase the training data quantity for Convolution Neural Networks in signal processing. In order to provide a better performance of our models that require a large dataset for training to be sufficient in learning the problem, and to prevent an overfitting problem, the application of one or more data augmentation techniques to a collection of annotated training samples which result in new additional training data must be done. There are many techniques of D.A as we introduced in the second chapter. However, in our system we implemented four algorithms, as shown in **Figure 4.10.** In this section the user is able to make multiple choices of the data augmentation techniques he desires to apply. The first two methods are audio deformation methods to convert the raw audio signal data to a different tone of sound. Those methods are:

Pitch shifting: which raises or lowers the pitch of the audio sample without altering the duration. Where the parameters that the user can alter in this technique are the number of **Bins per octave** which is generally in most of cases set to 12 because there are 12 pitches in an octave. And the second parameter is the **number of steps** to move the pitch either up or down.

Both parameter's type should be an integer number. Since our system requires the number of octave to shift the pitch, the insert of those parameters helps it to calculate that number by applying the following equation:

$$Number\ of\ octaves = \frac{number\ of\ steps}{number\ of\ bins\ per\ octave} \quad (4.1)$$

Time Stretching: it slows down or speeds up the audio sample without changing the pitch. The user can change the factor of stretching which is a float number by entering it in the text box located in the "**Time stretching**" panel. If factor > 1 the signal is sped up, while if factor < 1 the signal is slowed down.

The new spectrum of the selected audio can be displayed as a small window shown in **Figure 4.11** after applying those methods on it by clicking on the "**Show**" button for each one of the two. Whereas, the buttons "**Play**" allows the user to play the converted audio and "**Stop**" for stop it.

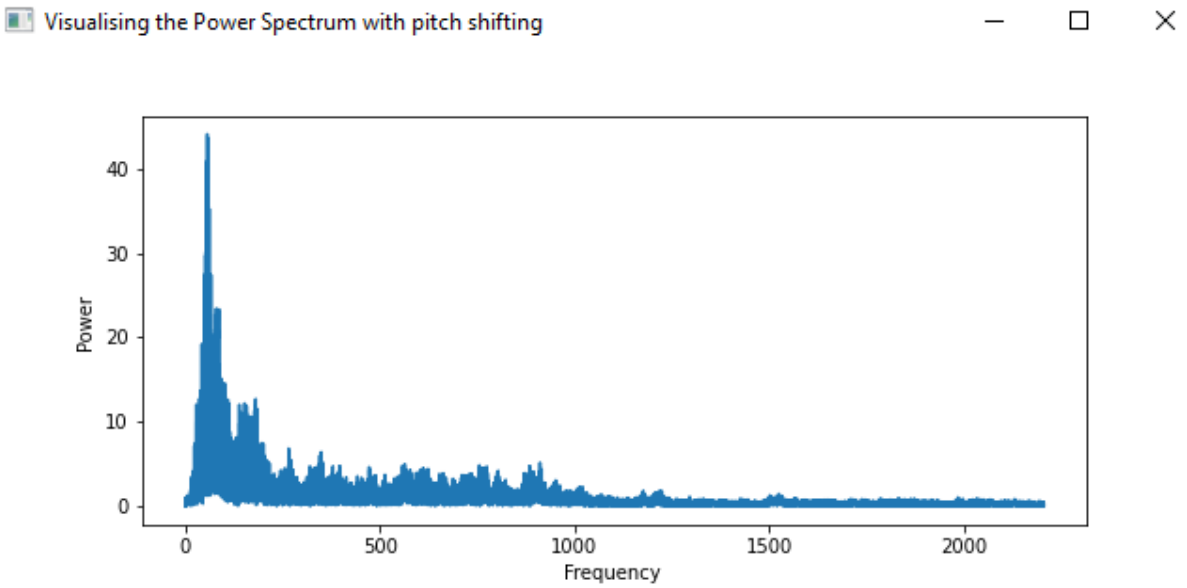

Figure 4.11: Spectrums of audio recording after pitch shifting deformation.

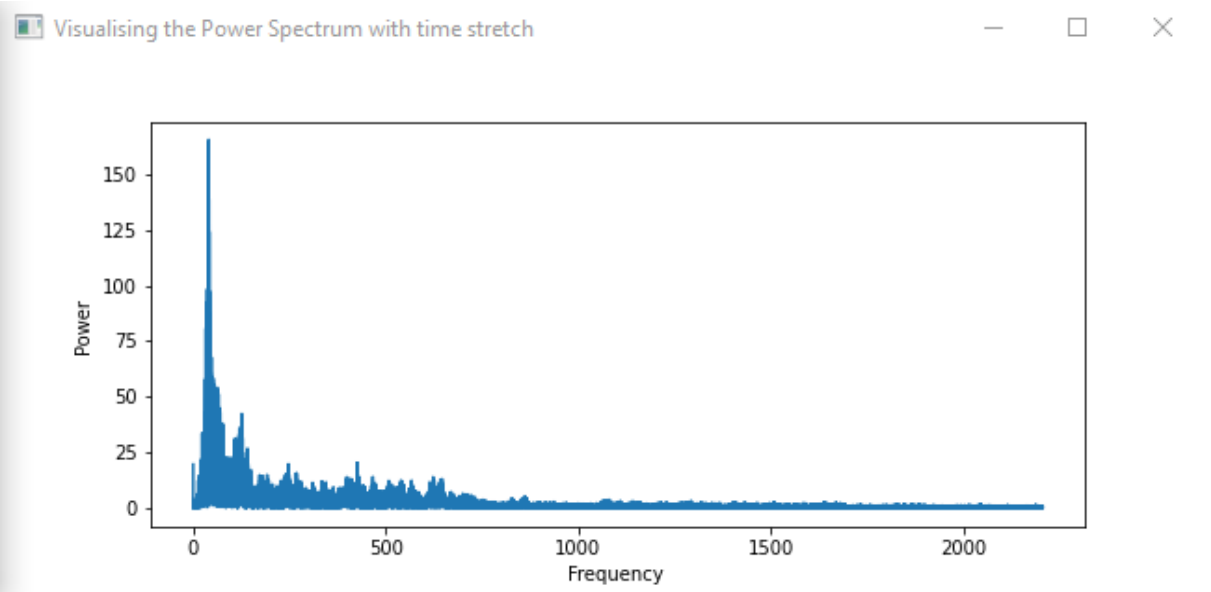

Figure 4.12: Spectrums of audio recording after time stretching deformation.

Our third data augmentation protocol is **Frequency masking**. It works directly on spectrograms. In this technique the user has the control over one parameter which is the maximum portion of all frequencies that will be masked. Also he can observe in the bottom of the window the spectrograms of the selected recording from the list of audio before and after frequency masking by clicking on the "**Show spectrogram**" button as shown in **Figure 4.12**.

As for the fourth technique that can be used in this approach is the Mixup technique, where the parameter required alpha **(α)** is the hyper-parameter that controls the strength of the interpolation between pairs of images and the associated smoothing of the training labels.

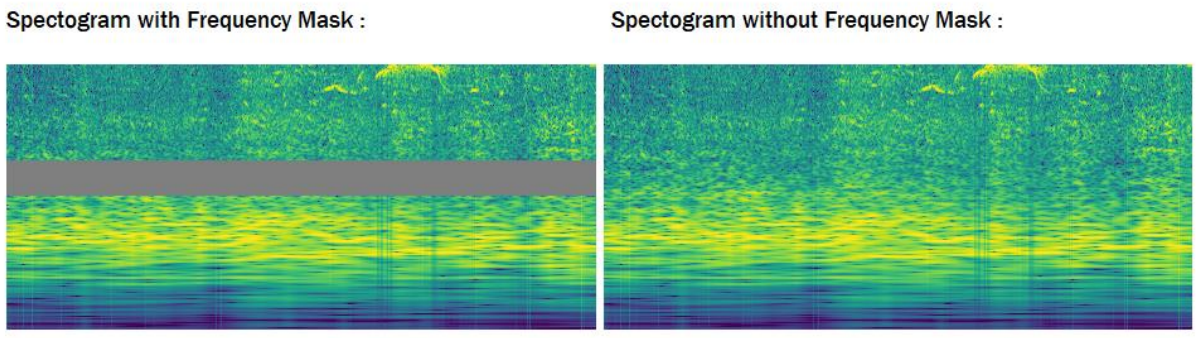

Figure 4.13: Spectrogram with and without frequency masking.

The new augmented dataset obtained from this process is then transferred to the next module of Training to be fed to the model by clicking on the "**Next**" button.

4.5.4 Training

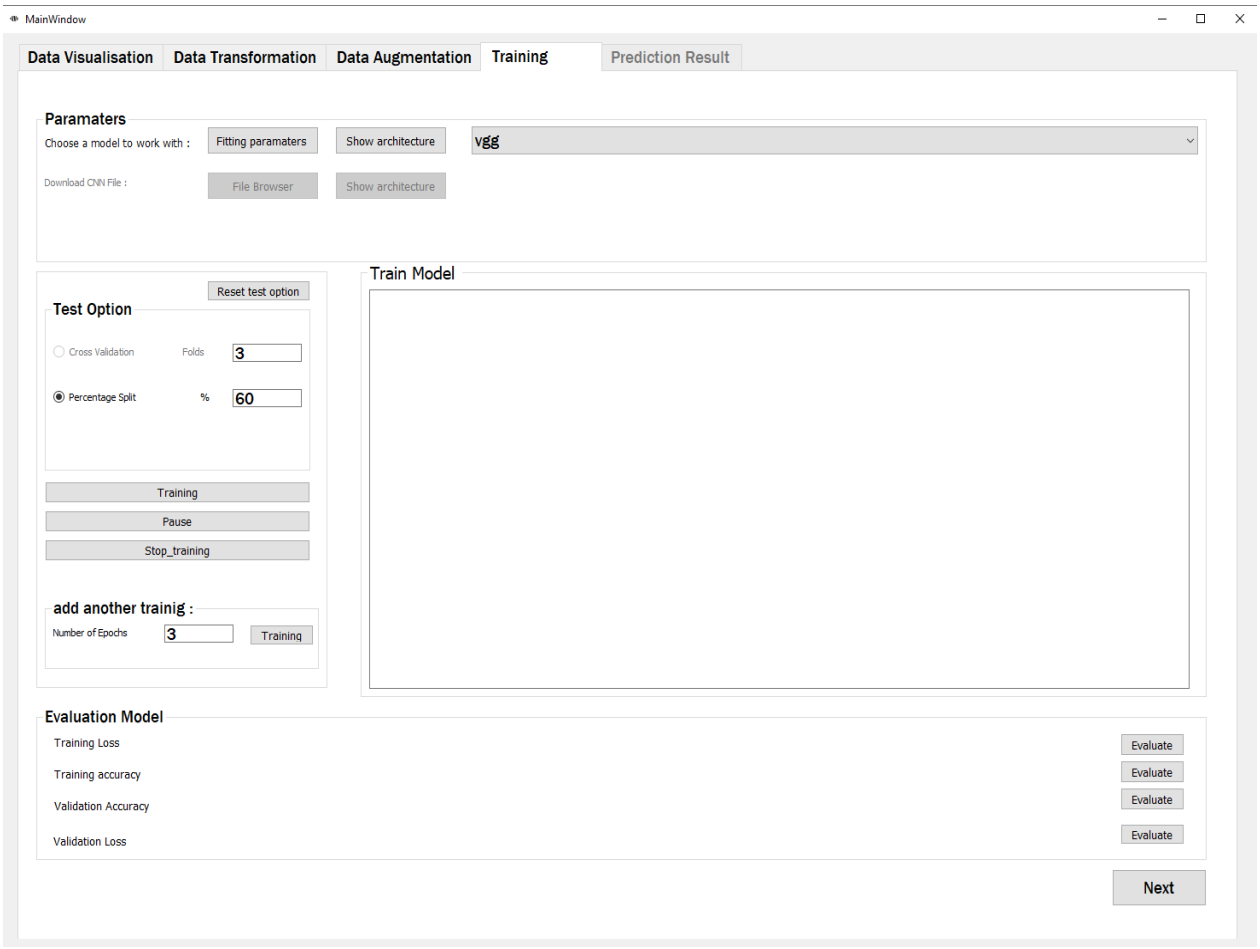

Figure 4.14: Tab 6 “Training”.

Now that the new dataset is ready to be employed as an input to the model. Moving to the training step shown its interface in **Figure 4.13**. Where in the top panel, the user has to select a model he is desiring to employ from the drop-down box shown in **Figure 4.14**. He has two options for that purpose. The first one is selecting a pre-trained model from the list. Whereas the second option is importing a CNN file that he already has in his computer.

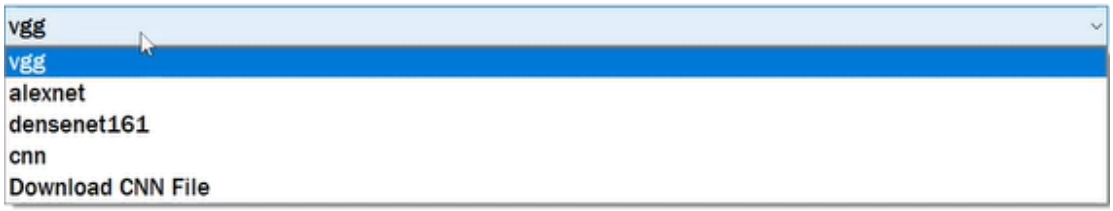

Figure 4.15: List of models to choose.

For both options the user has the ability to:

First, see the model's architecture and observe all types of layers composing it (conv/pooling and FC), its input dimensions and the various hyper-parameters for that layer; by clicking on "**Show architecture**" button. The architecture is displayed in a new small window in a form of lines where each line refers to each layer and its parameters such as the stride, filter size, number of filters and the padding parameter. **Figure 4.15** shows the window of "VGG16" architecture that appears.

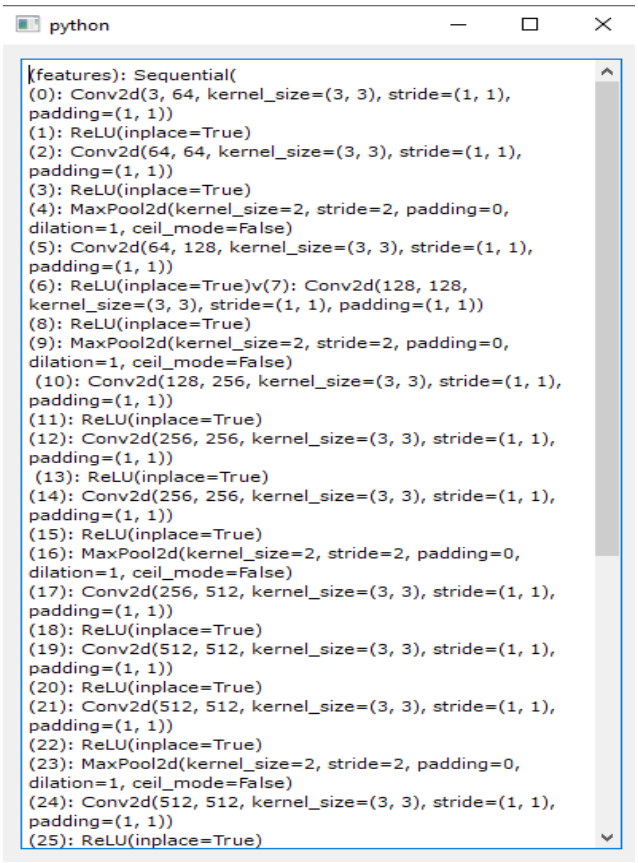

Figure 4.16: Architecture window.

Secondly, the user should do some alterations in the parameters of the model architecture through the button "**Fitting parameters**" which will display a new window containing the tunable parameters. There appear 3 types of parameters as shown in **Figure 4.16**:

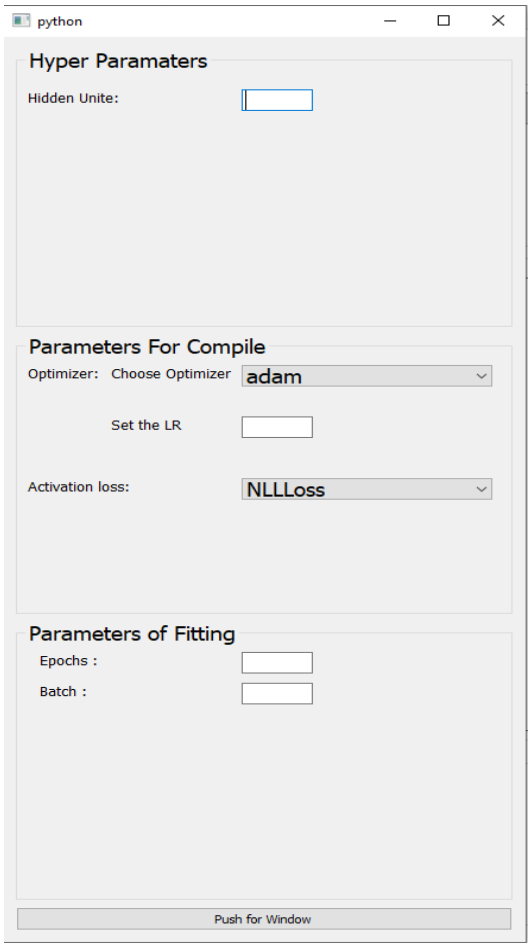

Figure 4.17: Model's parameter to set.

1. **Hyper-parameters:** in this type we added one parameter which is the hidden units. It corresponds to the output of a single filter at a single particular x/y offset in the input volume.

2. **Parameter for compile:** in this section there are 2 parameters. The first one is the optimizer algorithm to use for changing the attributes of the neural network such as weights and learning rate to reduce the losses. Thus, the user can choose which optimizer to use through a drop-down box shown in **Figure 4. .** We have adopted two different state-of-the-art gradient descent-based optimizers in our experiment, namely:

    A. **Adaptive Momentum (Adam):** Adam is known to be the fastest algorithm to converge to minima. The method is really efficient when working with large problems involving a lot of data or parameters. It requires less memory and is efficient.

B. **Stochastic Gradient Descent SGD**: SGD algorithm can only be used for shallow networks.

For the optimizer, the user have to enter its parameter which is the learning rate (LR) or step size, This parameter tells the optimizer how far to move the weights in the direction opposite of the gradient for a mini-batch. The training should start from a relatively large learning rate because, in the beginning, random weights are far from optimal, and then the learning rate can decrease during training to allow more fine-grained weight updates. LR=0.001 shows a relative good performance to start with.

The second parameter in this section is Activation loss where the user can choose one loss function among the existing ones in the drop-down box. The loss functions we adopted in our system are **NLL Loss** and **Cross entropy loss**.

3. **Parameters of fitting:** in this section there are 2 parameters as well to set:

Batch size: the higher the batch size, the more memory space our system will need. **Number of epochs:** is the number of forward pass and backward pass of all the training examples.

After setting all the parameters needed, the user must select a type of splitting from the ones in the "**Test option**" panel:

A. **Cross validation:** our system adopts the k-fold cross validation technique where the parameter the user must enter is the folds number, after that the system splits the data set into folds wherein one fold will be for the testing and the rest are for training. And so on, this splitting phase is done k times with different testing sets each time. K-fold cross-validation provides a way to improve the estimated performance of a machine learning model. That means the result is expected to be more accurate.

B. **Percentage split:** This option allows the user to choose the percentage of the training set he desires to train on, and the remaining percentage will be for the testing set.

Note that these options can be selected only if the user imported a one development dataset in the first module (Data loading and visualization).

To start the execution of the model's training process and see what the deep learner does in each epoch in the training, the user has to click on the "**Training**" button to invoke the model

chosen and get the screen that looks like **Figure 4.17.** This screen shows the progression of the accuracy achieved along with the confusion matrix in each epoch severally, or folds if the split option was “cross validation”.

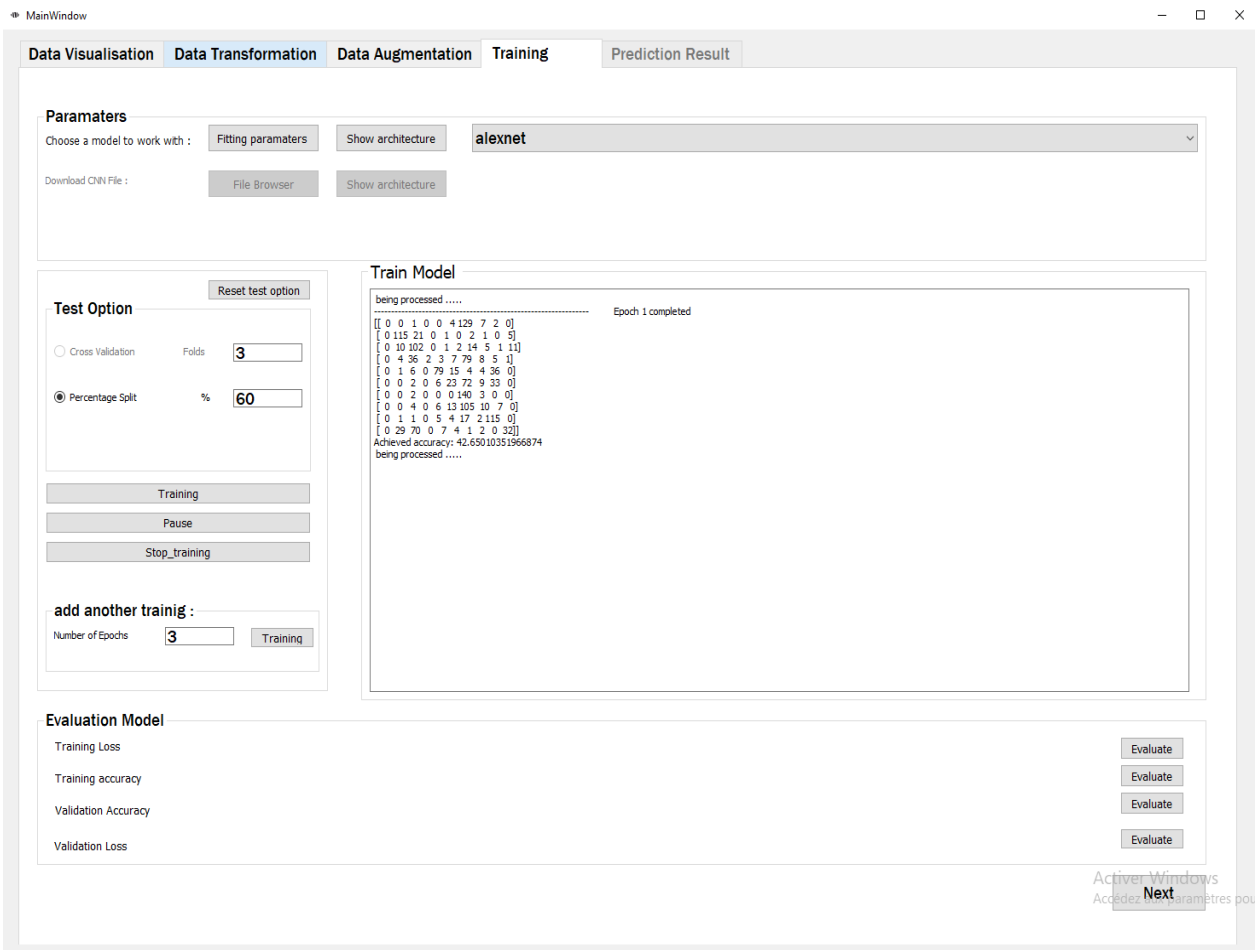

Figure 4.18: Training process.

Furthermore, to stop the training process at a certain point, our system provides the ability in that case through the “**Stop_training**” button. The trained model then will be saved as checkpoints. On the other hand, if the user wants to pause the training for a while and resume it later, he can click on the “**Pause**” button which will be switched right away to “**Restart**” button to start the training again.

Also, after the end of training process or when stopping the training, the user can add another round of training of the model by setting the number of epochs he desire in the text box of the “**Add another training**” panel, and then click on “**Training**” button.

The proposed model produces state-of-the-art results for acoustic scene classification. After the loss is calculated with the chosen loss function, the running loss is calculated, further,

the epoch loss is scored as well as the accuracy**.** And thence to evaluate the final results that the model produced. There are four buttons in the panel of "**Model Evaluation**". Each one of them displays a plot of the specified result in a window that pops up. The available plots are of:

1. **Training accuracy:** the accuracy obtained from using the training set after the split. It is calculated by dividing the number of correct predictions by the number of total predictions.

2. **Validation accuracy:** the accuracy the model calculates on the testing set that is not used for training.

3. **Training loss:** measured during each epoch and it indicates how well the model is fitting the training data. It is calculated with the loss function

4. **Validation loss:** measured after each epoch and it indicates how well the model fits new data.

Figure 4.18 shows an example of training accuracy plot for the "VGG16" model. Whereas **Figure 4.19** is a plot of training loss for the same model.

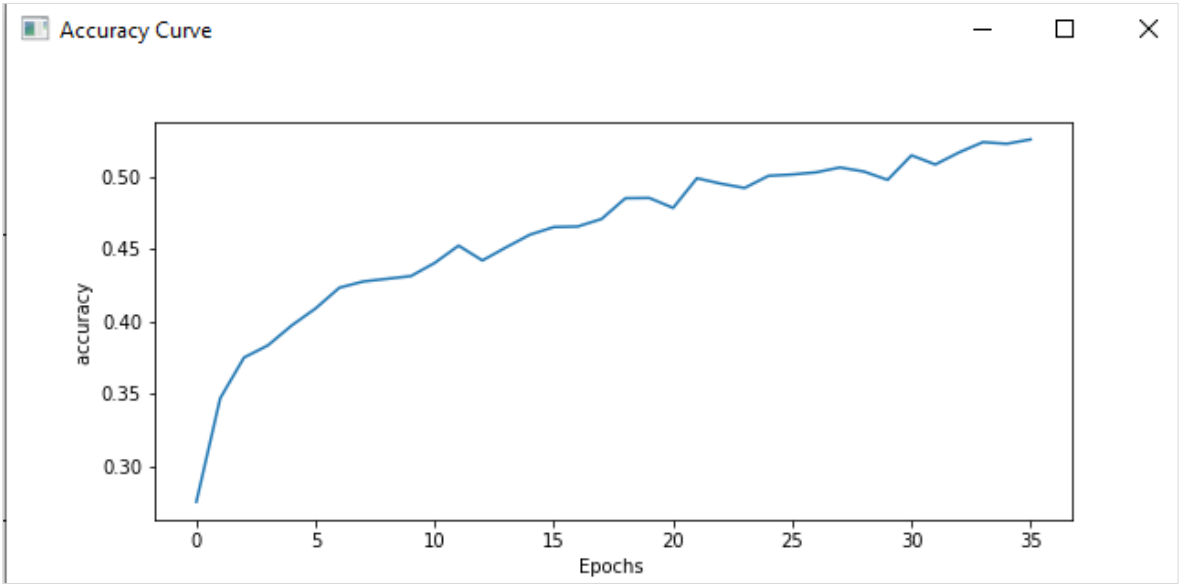

Figure 4.19: Training accuracy plot of the VGG16 model.

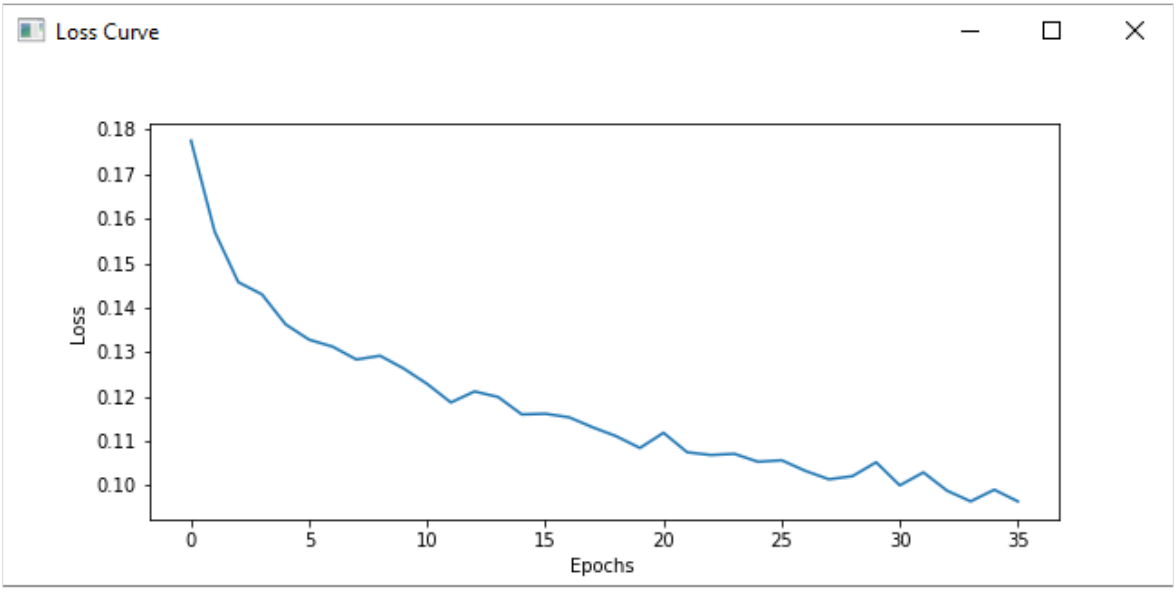

Figure 4.20: Training loss plot of the VGG 16 model

Moreover, the trained model is saved as checkpoints. By clicking on “Next” button the user can move to the next tab and the last one of the workbench which is the “Prediction” tab.

4.5.5 Prediction

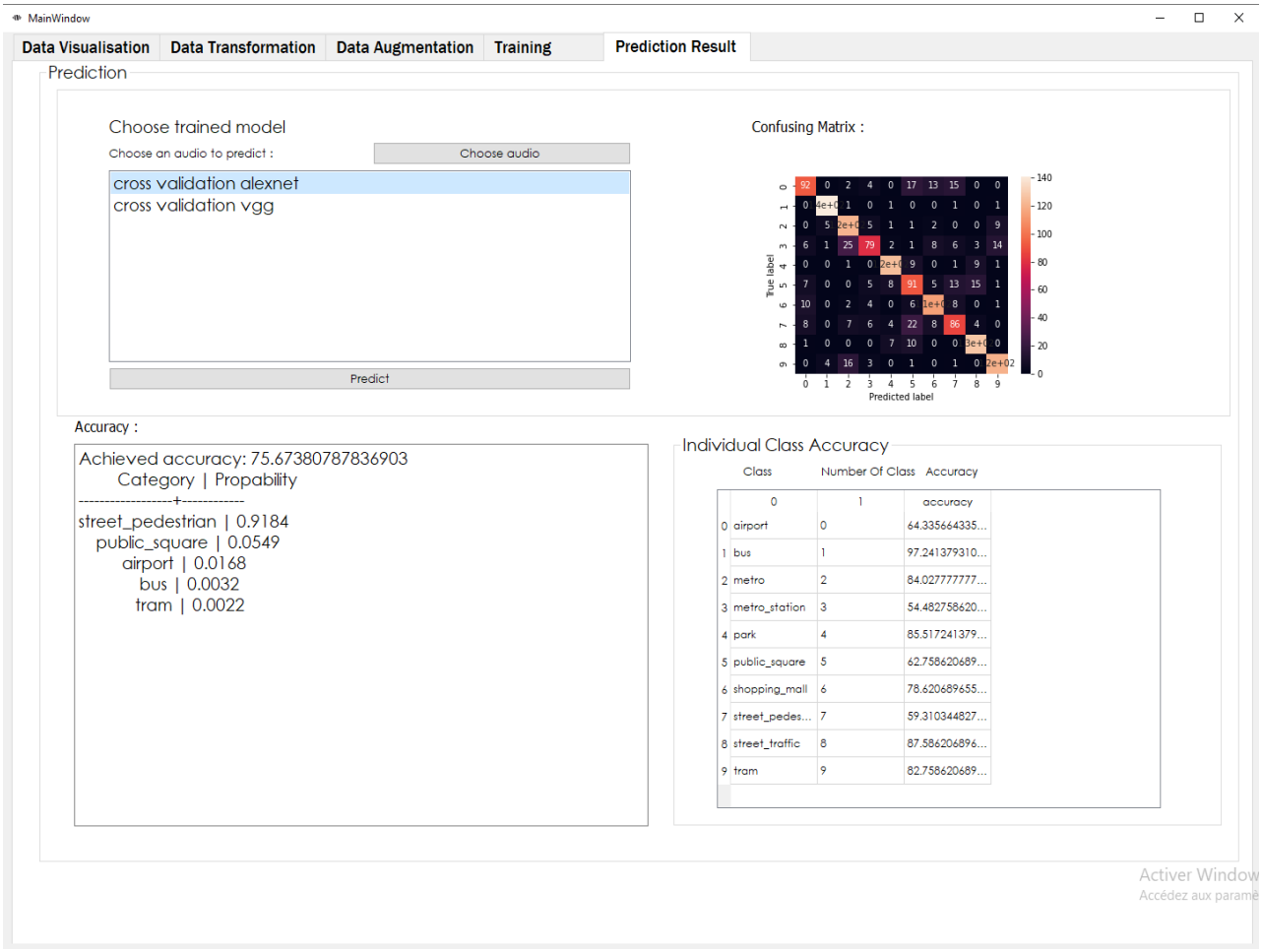

Figure 4.21: Tab 5 "Prediction".

Last tab of our workbench concerning the prediction module. It allows the user to get a prediction of the class label of an audio recording he chooses along with the accuracy achieved and the probability percentage using a model that is trained previously.

First, the user must choose an audio recording from the dataset file by clicking on the "**Choose audio**" browsing button on the "**Choose trained model**". Then select the model from the list that appears of the trained models to invoke it as shown in **Figure 4.20**. The system adds a new model to the list whenever the user trains a model in the previous module and define it under the name of the split method used + the name of model. Then lastly, the user clicks on the "**Predict**" button to start the prediction process. At this point, the system feeds the audio to the trained model which is saved as a checkpoints and returns the labels of the data loaded based upon the trained data obtained from the model. The different results produced by the system will be displayed in the screen. **Figure 4.20** Represent the prediction results for a street pedestrain audio recording. Where the right side of the panel presents the prediction accuracy achieved for assessing the performance of the model. And a table of probabilities for the

predicted 5 distinct classes that the model assigns, that part is exhibited the text area. Whereas the left side is for the confusion matrix, to have an overview and thoroughly observe the model behavior over all the classes, each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. Furthermore, to better understand the result obtained from the confusion matrix in a clear way, the user gets also a table in the "**Individual class accuracy**" panel that illustrates the accuracy of the model per all classes to see which classes the model can recognize more accurately.

Another window that pops-up as well with those prediction results is shown in **Figure 4.21**. This window offers a plotted chart of the probabilities table.

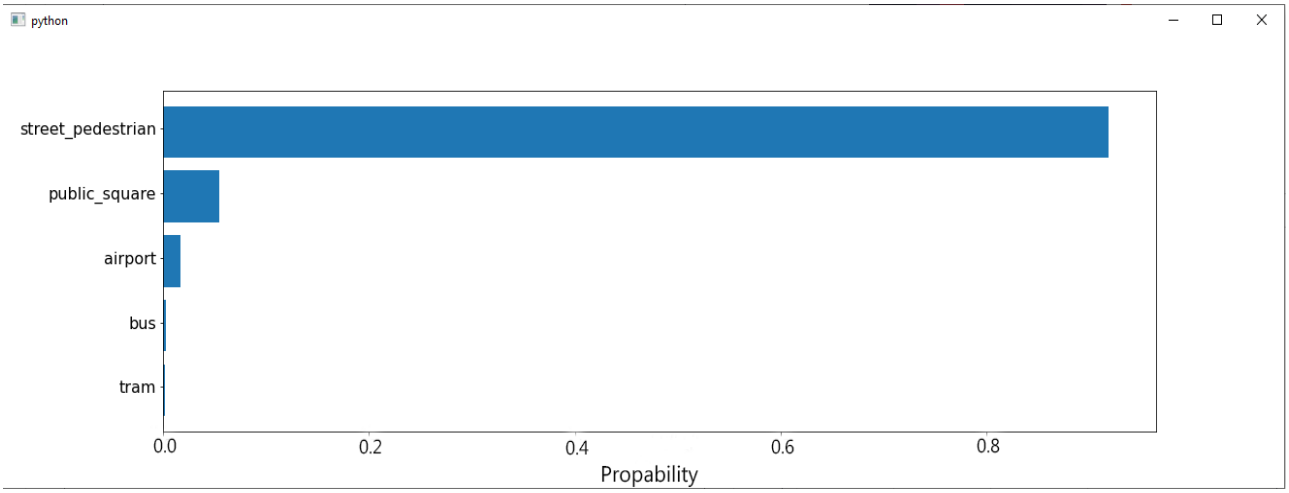

Figure 4.22: Chart of the probabilities.

4.6 Systems Evaluation

We conducted in this section various tests of systems built with the workbench on the dataset we introduced in section 4.2, while varying in each system a technique or a parameter values and recorded the obtained accuracy in order to assess the influence of these techniques. In these systems, we have set as features parameters the following values:

Table 4.1: Features parameters.

Mel Spectrogram	
Parameter	**Value**
Window length	8192
Hop-length	1875
Log-frequency Spectrogram	

Hop-length	1024
Number of frequency bins	256
Bins per octave	24
Minimum frequency	10

Whereas the unchanged hyper-parameters we have set to train all those systems are:

Table 4.2: Hyper-parameters.

Parameter	Value
Hidden units	1024
Number of epochs	40
Batch size	16

In **Table 4.3**, we characterized the systems we tested, and evaluated the accuracy achieved by each one, first we varied the techniques and parameters using the AlexNet model to see the impact of those techniques on the systems. Then, we changed the model to compare between the performances of the models used.

Table 4.3: Systems characterization and performances.

Number	Model	Feature	Data Augmentation	Split option	Fitting parameters	Accuracy
1	AlexNet	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	Percentage split (60%)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	70.51%
2	AlexNet	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	K-fold (2 folds)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	71.63%
3	AlexNet	Mel-spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	Percentage split (60%)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	68.70%

4	AlexNet	Log-frequency spectrogram	Mixup (alpha= 0.7)	K-fold (2 folds)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	70.15%
5	AlexNet	Log-frequency spectrogram	Mixup (alpha= 0.7)	Percentage split (60%)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	70.05%
6	AlexNet	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2) -Mixup (alpha= 0.7)	K-fold (2 folds)	Optimizer: Adam LR= 0.003 Activation Loss: NLLLoss	60.5%
7	AlexNet	Log-frequency spectrogram	-Mixup (alpha= 0.7) -Pitch shifting(0.5 octave) - Time stretching(0.25 steps)	K-fold (2 folds)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	72.8%
8	AlexNet	Log-frequency spectrogram	-Mixup (alpha= 0.7) -Pitch shifting(0.5 octave) - Time stretching(0.25 steps)	K-fold (2 folds)	Optimizer: SGD LR= 0.001 Activation Loss: NLLLoss	64.92%
9	AlexNet	Log-frequency spectrogram	-Mixup (alpha= 0.7) -Pitch shifting(0.5 octave) - Time stretching(0.25 steps)	K-fold (2 folds)	Optimizer: Adam LR= 0.001 Activation Loss: CrossEntropyLoss	69.21%
10	VGG16	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	Percentage split (60%)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	70.87%
11	Imported CNN (modified AlexNet)	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	Percentage split (60%)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	72.6%

12	Imported CNN (modified AlexNet)	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	K-fold (2 folds)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	73.8%
13	DenseNet	Log-frequency spectrogram	-Pitch shifting(0.5 octave) - Time stretching(0.25 steps) - Frequency masking(0.2)	K-fold (3 folds)	Optimizer: Adam LR= 0.001 Activation Loss: NLLLoss	62.20%

From this experiment, and with observing the results in the table, we can see clearly that the 7th system have the best accuracy among the others. Comparing the techniques used in this system with the rest, we have concluded that Log-frequency spectrogram as audio feature representation has shown a better result than the Mel-spectrogram with the AlexNet model. And using mixup with both pitch shifting and time stretching as data augmentation methods has granted a best performance of the AlexNet model. Moreover, K-fold cross validation was the best option to split the data. And as for fitting parameters, using Adam optimizer and NLLLoss function has shown a preferable accuracy of the system over SGD optimizer and Cross Entropy Loss function.

Furthermore, for the model choice, we deduced from our experiment that AlexNet has a high and stable accuracy and a finer performance compared the other existing models which got relatively a lower accuracy rate.

4.7 Conclusion

In this chapter we defined the different setups we have used to conduct our experiment and to build our ASC workbench. We have presented the dataset used to train our systems as well as the libraries utilized to perform its functions. We also presented the graphical interface of the workbench and the sequential steps of using it, with a brief description for the different functions of each module of the workbench. And lastly, we executed our system with various techniques for every module and recorded in a table the results obtained.

CONCLUSION

In this thesis, we dealt with the sound analysis approach. Although analysis of audio signals covers other research areas such as automatic speech recognition, we restricted ourselves to the Acoustic Scene Classification task in the context of this thesis. We opted to include the state-of-the-art notions that are necessary for understanding the audio signal and its various representations with offering a short reiteration of the most important frequently used techniques of digital signal processing for the analysis of audio signals. Our main goal in this trial was to design an open source acoustic scene classification workbench able to recognize and classify acoustic scenes using different machine learning-based techniques for the purpose of creating accurate predictions. The implemented workbench's design is modular; it consists of processing blocks which concerns the stages of classification task. A block can represent different functionality where the users is allowed to make changes in the parameters fast, create big tests, and visualize the results of these tests.

The execution of our workbench have been conducted using a modified TAU Urban Acoustic Scenes 2019 dataset; an audio recordings database recorded from real world sound in a way similar to the sound that reaches the human auditory system. It consists of 10 various recording location as labels in 5 different cities. Totaling 7192 segments and 35.5 GB in wav format. This dataset was loaded into our workbench for testing the techniques implemented. We used 2 different 2D representation to extract the data features, first one is the Mel spectrogram and the other is Log-frequency spectrogram. And to prevent the occurrence an overffiting, we provided 4 methods of the data augmentation regularization, which increases the size of the dataset by making changes on the original data while keeping its characteristics. Those methods are respectively: pitch shifting, time stretching, frequency masking, and mixup. Lastly, as deep learning-based model, we have added diverse CNN models that can be trained on sets of images (2D representation in our case). The models are namely: VGG16, AlexNet, DenseNet, and another CNN architecture could be imported. To evaluate the training and testing of the model we choose to plot the accuracy and the loss achieved after each training process. And to consult the performance of the model after the prediction, we opted to use the confusion matrix along with a table of the accuracy achieved per each class.

In our work, we built an extendible workbench that can be updated frequently by other developers. That can be done by adding new machine learning-based techniques to provide the users with plenty of options and state of art tools for feature extraction and machine learning as paradigms well as tools for deep learning algorithms. One of the tools that can often be implemented is pre-processing techniques for extracting and removing the noises from the audio dataset for the classification such as Denoising AutoEncoder (DAE) methods. Other tools that can be added to the workbench is audio feature extraction methods to choose as MFCC. And moreover, different deep learning models for audio features classification along with the other evaluation metrics may be appended to the program for training it through the workbench in an easy way, such as ResNet, LeNet-5 and Inception-v3. Or even add more of its hyperparameters to tune such as filter size, stride…etc; for the CNN-based architectures.

REFERENCES

[1] "I. Lezhenin, N. Bogach, and E. Pyshkin, Urban Sound Classification using Long Short-Term Memory Neural Network. 2019.," [Online].

[2] "Sharma, Jivitesh & Granmo, Ole-Christoffer & Goodwin, Morten. (2019). Environment Sound Classification using Multiple Feature Channels and Deep Convolutional Neural Networks.," [Online].

[3] "T. Virtanen, M. D. Plumbley, and D. Ellis, Computational analysis of sound scenes and events. Springer International Publishing, 2017.," [Online].

[4] F. Thomas, Discret Time speech processing, 2006.

[5] ""Sound and Waveforms." YouTube, uploaded by Valerio Velardo - The Sound of AI, 29 june 2020,www.youtube.com/watch?v=bnHHVo3j124&list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0&index=2," [Online].

[6] "SPA 3011: Introduction to Speech Science," 2020. [Online].

[7] A. F. P. Sound, "Byjus," 2021. [Online]. Available: https://byjus.com/physics/amplitude-frequency-period-sound/.

[8] anonymous, "what-is-sound," The Soundproofing Company, 2019. [Online]. Available: https://www.soundproofingcompany.com/soundproofing_101/what-is-sound.

[9] Lu L., Hanjalic A. (2009) Audio Representation. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA..

[10] Proakis, John G.; Manolakis, Dimitris G. (2007-01-01). Digital Signal Processing. Pearson Prentice Hall. ISBN 9780131873742. Archived from the original on 2016-05-20. Retrieved 2015-09-22..

[11] Doug Mercer,"Chapter 20: Analog to Digital Conversion", 2021, university/courses/electronics/text/chapter-20.txt.

[12] A. L. K. S. Faraj, Chapter 3 (Quantization & Encoding) part (2). 10.13140/RG.2.2.35436.10888., Al-Israa University College, 2020.

[13] anonymous, EC312_Lesson_20_Analog_to_Digital_Course_Notes, 2020.

[14] "JavaTPoint, difference-between-analog-signals-and-digital-signals," 2011-2021. [Online]. Available: https://www.javatpoint.com/difference-between-analog-signals-and-digital-signals.

[15] American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)(2017)"Fuzzy Logic Based Segmentation for Myanmar Continuous Speech Recognition System" Volume 00, No 1, pp 00-00.

[16] David Gerhard, "Audio Signal Classification:History and Current Techniques"Technical Report TR-CS 2003-07 November, 2003.

[17] "Isaac's science blog, Fourier transform," 13 08 17. [Online]. Available: https://isaacscienceblog.com/2017/08/13/fourier-transform/.

[18] Nasser Kehtarnavaz, in Digital Signal Processing System Design (Second Edition), 2008.

[19] ""Short-Time Fourier Transform Explained Easily." YouTube, uploaded by Valerio Velardo - The Sound of AI, 6 sep 2020, https://www.youtube.com/watch?v=-Yxj3yfvY-4&list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0&index=15," [Online].

[20] T. Virtanen, M. D. Plumbley, and D. Ellis, Computational analysis of sound scenes and events. Springer International Publishing, 2017..

[21] "Damskägg, E.-P.; Välimäki, V. Audio Time Stretching Using Fuzzy Classification of Spectral Bins. Appl. Sci. 2017, 7, 1293. https://doi.org/10.3390/app7121293," [Online].

[22] "Adam Sabra, Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients, towards data science," 17 February 2021. [Online].

[23] D. &. L.-D. A. &. T. D. &. G.-R. J. de Benito, "Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset.," *EURASIP Journal on Audio, Speech, and Music Processing*, 2019.

[24] "Sanket Doshi, "Music Feature Extraction in Python", Dec 30, 2018," [Online].

[25] F. Jin, F. Sattar and S. Krishnan, "Log-frequency spectrogram for respiratory sound monitoring," 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 597-600, doi: 10.1109/ICASSP.2012.6287954.

[26] "DCASE," 2019. [Online]. Available: http://dcase.community/challenge2019/..

[27] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, “Semantic annotation and retrieval of music and sound effects,” IEEE Transactions on Audio, Speech and Language Processing, vol. 16, no. 2, pp. 467-476, Feb 2008.

[28] A. Mesaros, T. Heittola, and T. Virtanen, “A multi-device dataset for urban acoustic scene classification,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018), November 2018, pp.9-13.

[29] S. Chu, S. Narayanan, C. . J. Kuo, and M. J. Mataric, “Where am i? scene recognition for mobile robots using audio features,” in 2006 IEEE International Conference on Multimedia and Expo, July 2006, pp. 885888..

[30] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi et.al, “Audiobased context recognition,” IEEE Transactions on Audio, Speech, and Language Processing, vol. 14, no. 1, pp. 321 329, Jan 2006.

[31] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, “Audio analysis for surveillance applications,” in IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005, Oct 2005, pp. 158161.

[32] Zhang, L.; Shi, Z.; Han, J. Pyramidal Temporal Pooling With Discriminative Mapping for Audio Classification. IEEE/ACM Trans. Audio Speech Lang. Process. 2020, 28, 770–784. [CrossRef].

[33] Abeber, J. A Review of Deep Learning Based Methods for Acoustic Scene Classification. Appl. Sci. 2020, 10, 2020..

[34] N. Sawhney and P. Maes, “Situational awareness from environmental sounds,” URL: http://web. media. mit. edu/˜ nitin/papers/Env Snds/EnvSnds. html, 1997.

[35] Computational Analysis of Sound Scenes and Events; Virtanen, T., Plumbley, M.D., Ellis, D., Eds.; Springer International Publishing: Berlin, Germany, 2018; doi:10.1007/978-3-319-63450-0. [CrossRef].

[36] Available online: http://dcase.community/challenge2020/task-acoustic-scene-classification (accessed on 2 January 2021)..

[37] Mesaros, A.; Heittola, T.; Virtanen, T. Assessment of Human and Machine Performance in Acoustic Scene Classification: DCASE 2016 Case Study. In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, 2017.

[38] D. Sharma and N. Kumar, “A Review on Machine Learning Algorithms, Tasks and Applications,” Int. J. Adv. Res. Comput. Eng. Technol., vol. 6, no. 10, pp. 1323–2278, Oct. 2017.

[39] S. Das, A. dey, A. Pal, and N. Roy, “Applications of Artificial Intelligence in Machine Learning: Review and Prospect,” Int. J. Comput. Appl., vol. 115, pp. 31–41, Apr. 2015, doi: 10.5120/20182-2402..

[40] Y. C A Padmanabha Reddy, P. Viswanath, and B. Eswara Reddy, “Semi-supervised learning: a brief review,” Int. J. Eng. Technol., vol. 7, no. 1.8, p. 81, 2018, doi: 10.14419/ijet.v7i1.8.9977..

[41] "IBM Cloud Education, “Supervised Learning”, August 2020, IBM," [Online].

[42] P. Dönmez, “Introduction to Machine Learning, 2nd ed., by Ethem Alpaydın. Cambridge, MA: The MIT Press2010. ISBN: 978-0-262-01243-0. $54/£ 39.95 + 584 pages.,” Nat. Lang. Eng., vol. 19, no. 2, pp. 285–288, 2013, doi: 10.1017/s1351324912000290..

[43] "Derakhshan, Behrouz,Markl, Volker,2019/03/29,“Continuous Deployment of Machine Learning Pipelines”,10.5441/002/edbt.2019.35," [Online].

[44] "Jakob Abeßer, “A Review of Deep Learning Based Methods for Acoustic Scene Classification”, Appl. Sci. 2020, 10(6), 2020; https://doi.org/10.3390/app10062020," [Online].

[45] A. Mesaros, T. Heittola, and D. Ellis, “Datasets and Evaluation,” in Computational Analysis of Sound Scenes and Events, 2018, pp. 147–179..

[46] A. Rakotomamonjy and G. Gasso, “Histogram of Gradients of Time–Frequency Representations for Audio Scene Classification,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 23, no. 1, pp. 142–153, 2015, doi: 10.1109/TASLP.2014.2375575..

[47] G. Lafay, M. Lagrange, E. Benetos, M. Rossignol, and A. Roebel, “A Morphological Model for Simulating Acoustic Scenes and Its Application to Sound Event Detection,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 24, pp. 1854–1864, Oct. 2016, doi: 10., oct 2016.

[48] Zahraa S. Abdallah, Monash University Lan Du, Monash University Geoffrey I. Webb, Monash University, “Encyclopedia of Machine Learning and Data Mining”, Springer, 2017.

[49] D. Pyle, S. Editor, and D. D. Cerra, Data Preparation for Data Mining, Editorial. San Francisco, CA 94104-3205 USA: Morgan Kaufmann Publishers, Inc., 1999..

[50] "Tom Bäckström, “Introduction to Speech Processing,” Aalto University Wiki, 2019.," [Online].

[51] "Haytham Fayek, Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and what's In-Between, 2016," [Online].

[52] "Alice Zheng , Amanda Casari ; Ed: Rachel Roumeliotis , Jeff Bleiel ; Prod: Kristen Brown ;2018," [Online].

[53] Laredo, David & Qin, Yulin & Schütze, Oliver & Sun, J. Q.. (2019). Automatic Model Selection for Neural Networks..

[54] Frank Hutter, Lars Kotthoff, Joaquin Vanschoren, The Springer Series on Challenges in Machine Learning," Automated Machine Learning Methods, Systems, Challenges" ISBN 978-3-030-05317-8 ISBN 978-3-030-05318-5, 2019.

[55] "PAYAM REFAEILZADEH, LEI TANG, HUAN LIU Arizona State University, « Cross-validation », 2008," [Online].

[56] S. Harikant, "K Fold Cross Validation Technique.," 15 06 2021. [Online]. Available: https://inblog.in/K-Fold-Cross-Validation-Technique-NCaSQ8Kmfh.

[57] "["model-training", Oden Technologies, 2020," [Online].

[58] Aishwarya , Akshatha , Anusha , Shishira , Deepa Mahadev; "Engagement Detection with Autism Spectrum Disorder using Machine Learning", Volume: 07 Issue: 07 | July 2020.

[59] Wang H., Zheng H. (2013) Model Testing, Machine Learning. In: Dubitzky W., Wolkenhauer O., Cho KH., Yokota H. (eds) Encyclopedia of Systems Biology. Springer, New York, NY.

[60] "Udemy course, "deployment-of-machine-learning-models"," 10 2021. [Online].

[61] Ackermann, Klaus & Haynes, Lauren & Ghani, Rayid & Walsh, Joe & De Unanue, Adolfo & Naveed, Hareem & Navarrete Rivera, Andrea & Lee, Sun-Joo & Bennett, Jason & Defoe, Michael & Cody, Crystal. (2018). Deploying Machine Learning Models for Public Policy, A Framework.

[62] "Nick McCullum, "Deep Learning Neural Networks Explained in Plain English", freecodecamp, 2020," [Online].

[63] "Jojo John Moolayil,"A Layman's Guide to Deep Neural Networks", towards data science,2019," [Online].

[64] "Nayak, Deepak & Mahapatra, Amitav & Mishra, Pranati. (2013). A Survey on Rainfall Prediction using Artificial Neural Network. International Journal of Computer Applications. 72. 32-40. 10.5120/12580-9217.," [Online].

[65] "Hinton G E. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair[C]// International Conference on International Conference on Machine Learning. Omnipress, 2010," [Online].

[66] Bisot, V.; Serizel, R.; Essid, S.; Richard, G. Nonnegative Feature Learning Methods for Acoustic Scene Classification. In Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), Munich, Germany, 16–17 November 2017..

[67] G. Takahashi, T. Yamada, N. Ono and Makino, S. Performance Evaluation of Acoustic Scene Classification using DNN-GMM and Frame-Concatenated Acoustic Features. In Proceedings of the 9th Asia-Pacific Signal and Information Processing Association Annual Summit and conference (APSIPA), Honolulu, HI, USA,, 2-15 November 2018, pp. pp. 1739-1743.

[68] "Amazon Machine Learning: Developer Guide, Training ML Models, Amazon Web Services, Inc. and/or its affiliates," 9 4 2015. [Online].

[69] "Niklas Donges, “Gradient Descent: An Introduction to 1 of Machine Learning’s Most Popular Algorithms”, Builtin, 2021," [Online].

[70] "Simeon Kostadinov, “Understanding Backpropagation Algorithm”, towards data science, 2019," [Online].

[71] "M. Cilimkovic, “Neural Networks and Back Propagation Algorithm,” Fett.Tu-Sofia.Bg, pp. 3–7, 2010, [Online]. Available: http://fett.tu-sofia.bg/et/2006/ET2006 BOOK 1/Circuits and Systems/173 Paper-V_Skorpil.pdf.," [Online].

[72] "J. Schmidhuber, “Deep Learning in Neural Networks: An Overview” http://arxiv.org/abs/1404.7828, 2014," [Online].

[73] "Bengio, Yoshua (2009). “Learning Deep Architectures for AI” (PDF). Foundations and Trends in Machine Learning 2 (1).," [Online].

[74] Sergi Perez-Castanos, Javier Naranjo-Alcázar, Pedro Zuccarello, Maximo Cobos and Francesc J. Ferri, “Ensemble Methods and Input Alternatives for Acoustic Scene Classification Using Convolutional Neural Networks”, URSI 2020, Malaga.

[75] "Rebai I, BenAyed Y, Mahdi W and Lorré J P 2017 Procedia Computer Science 112 316-22".

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, 2012, pp. pp. 1097-1105.

[77] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar, “Applications of convolutional neural networks,” International Journal of Computer Science and Information Technologies, vol. 7, no. 5, pp. 2206–2215, 2016..

[78] Lee, Yerin, Soyoung Lim, and Il-Youp Kwak. 2021. "CNN-Based Acoustic Scene Classification System" Electronics 10, no. 4: 371. https://doi.org/10.3390/electronics10040371.

[79] V. Maeda-Gutiérrez, C. Galván-Tejada, L. Zanella-Calzada, J. Celaya-Padilla, J. Galván-Tejada, H. Gamboa-Rosales, H. Luna-García, R. Magallanes-Quintanar, C. Guerrero Méndez and C. Olvera-Olvera, "Comparison of Convolutional Neural Network Architectures for Classification of Tomato Plant Diseases.," no. Applied Sciences 10, no. 4: 1245. https://doi.org/10.3390/app10041245, 2019.

[80] J. Lee, J. Park, K. Kim, and J. Nam, “Sample Cnn: End-to-end deep convolutional neural networks using very small filters for music classification,” Applied Sciences, vol. 8, no. 1, p. 150, 2018.

[81] Y. Su, K. Zhang, J. Wang, and K. Madani, “Environment sound classification using a two-stream CNN based on decision-level fusion,” Sensors, vol. 19, no. 7, p. 1733, 2019..

[82] Loris Nanni, Gianluca Maguolo, Michelangelo Paci, “Data augmentation approaches for improving animal audio classification”, Ecological Informatics, Volume 57, 2020,ISSN 1574-9541.

[83] "Chris Kevin, “Feature Maps”, medium, 2018," [Online].

[84] "CS231n: Convolutional Neural Networks for Visual Recognition, “Convolutional Neural Networks (CNNs / ConvNets)”," 2021. [Online].

[85] "Hossein Gholamalinezhad , Hossein Khosravi,”Pooling Methods in Deep Neural Networks, a Review”," 17 10 2020. [Online].

[86] "Sumit Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way”,towards data science, 2018," [Online].

[87] Aggarwal, Arshiya & Das, Nisheet & Sreedevi, Indu. (2019). Attention-guided deep convolutional neural networks for skin cancer classification. 1-6. 10.1109/IPTA.2019.8936100..

[88] X. Han, Y. Zhong, L. Cao, and L. Zhang, “Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification,” Remote Sens., vol. 9, no. 8, 2017, doi: 10.3390/rs9080848..

[89] V. Maeda-Gutiérrez et al., “Comparison of convolutional neural network architectures for classification of tomato plant diseases,” Appl. Sci., vol. 10, no. 4, 2020, doi: 10.3390/app10041245..

[90] "G. Huang, Z. Liu and L. van der Maaten, “Densely Connected Convolutional Networks,” 2018," [Online].

[91] Huang, Gao & Liu, Zhuang & van der Maaten, Laurens & Weinberger, Kilian. (2017). Densely Connected Convolutional Networks. 10.1109/CVPR.2017.243..

[92] "anonymous, “DenseNet Architecture Explained with PyTorch Implementation from TorchVision", amaarora.github.io, 2020," [Online].

[93] "Pablo Ruiz,” Understanding And Visualizing Densenets" , towards data science, 2018," [Online].

[94] "C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, “Deeply Supervised nets”. In AISTATS, 2015".

[95] Huang, Gao & Liu, Zhuang & van der Maaten, Laurens & Weinberger, Kilian. (2017). Densely Connected Convolutional Networks. 10.1109/CVPR.2017.243..

[96] Ho, Nhat & Nguyen, Tan & Patel, Ankit & Anandkumar, Animashree & Jordan, Michael & Baraniuk, Richard. A Bayesian Perspective of Convolutional Neural Networks through a Deconvolutional Generative Model., 2021.

[97] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for lvcsr using rectified linear units and dropout,” in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2013, pp. 8609–8613..

[98] J. Brownlee, "“Dropout Regularization in Deep Learning Models With Keras” ,machine learning mastery,," 2016. [Online].

[99] anonymous, " “Everything You Should Know About Dropouts And Batch Normalization In CNN”, analytics India mag," 2020. [Online].

[100] "T. DeVries and G. Taylor, “Improved Regularization of Convolutional Neural Networks with Cutout,” in arXiv, 2017".

[101] "Park, Daniel & Chan, William & Zhang, Yu & Chiu, Chung-Cheng & Zoph, Barret & Cubuk, Ekin & Le, Quoc. (2019). SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. 2613-2617. 10.21437/Interspeech.2019-2680".

[102] Shengyun Wei1 , Shun Zou , Feifan Liao and weimin lang, “A Comparison on Data Augmentation Methods Based on Deep Learning for Audio Classification”,CISAI 2019,doi:10.1088/1742-6596/1453/1/012085.

[103] Kristjan Greenewald, Anming Gu, Mikhail Yurochkin, Justin Solomon, Edward Chien; “k-Mixup Regularization for Deep Learning via Optimal Transport”, [v1] Sat, 5 Jun 2021.

[104] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz, “mixup: Beyond Empirical Risk Minimization”, Apr 2018 , version, v2.

[105] S. Wei, S. Zou, F. Liao, and W. Lang, “A Comparison on Data Augmentation Methods Based on Deep Learning for Audio Classification,” J. Phys. Conf. Ser., vol. 1453, no. 1, 2020, doi: 10.1088/1742-6596/1453/1/012085.

[106] "WHAT IS THE NORMALISATION PROCESS?," [Online]. Available: https://sociology-tips.com/library/lecture/read/12382-what-is-the-normalisation-process.

[107] B. Davis, "What is layer normalization?," 1 june 2021. [Online]. Available: https://www.mvorganizing.org/what-is-layer-normalization/.

[108] A. Christopher, "Batch Normalization," 16 june 2021. [Online]. Available: https://medium.com/nerd-for-tech/batch-normalization-51e32053f20.

[109] van der Walt, Stéfan & Colbert, S. & Varoquaux, Gael. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering. 13. 22 - 30. 10.1109/MCSE.2011.37..

[110] "M. Scott Shell, “An introduction to Numpy and Scipy”, 2019," [Online].

[111] "Mckinney, Wes. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics. Python High Performance Science Computer.," [Online].

[112] "scikit_learn_tutorial.pdf, Scikit-Learn, kiran panigrahi," 2007-2021. [Online].

[113] D. D. E. F. M. D. a. t. M. d. t. John Hunter, "Matplotlib pdf," 13 August 2021. [Online]. Available: https://matplotlib.org/stable/.

[114] "John Hunter, Darren Dale, Eric Firing, Michael Droettboom, Matplotlib, Release 2.0.0," 10 05 2017. [Online].

[115] "pyqt5-tutorial," 2021. [Online]. Available: https://build-system.fman.io/pyqt5-tutorial.

[116] M. A. Hakim, "top-5-ways-to-build-a-python-desktop-app-in-2021," 13 february 2021. [Online].

[117] "Qt Designer Manual," 2020. [Online]. Available: https://doc.qt.io/archives/qt-4.8/designer-manual.html.

[118] K. Tran, "What is PyTorch?," 6 april 2020. [Online]. Available: https://towardsdatascience.com/what-is-pytorch-a84e4559f0e3.

[119] "Learning PyTorch : Introduction," 3 october 2020. [Online]. Available: https://medium.com/@pixelbegetLab.

[120] L. A. A. T. V. ELI STEVENS, Deep Learning With PyTorch [First ed.], 2020, p. 450.

[121] ELI STEVENS, LUCA ANTIGA, AND THOMAS VIEHMANN FOREWORD BY SOUMITH CHINTALA, Deep Learning with PyTorch, ©2020 by Manning Publications Co..

[122] "Dishant Parikh, Saurabh Sachdev, Improving the efficiency of spectral features extraction by structuring the audio files, arXiv:2010.03136v1 [cs.SD] 7 Oct 2020".