Università
Ca'Foscari
Venezia

Bachelor's Degree
in Informatics

Final Thesis

A study on Equalization Curve Detection in Audio Tape Digitization process using Artificial Intelligence

Supervisor
Ch. Prof. Ilaria Prosdocimi

Graduand
Matteo Spanio
Matricolation number
877485

Academic Year
2021/2022

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to all those who have contributed to the successful completion of my thesis in computer science.

First and foremost, I would like to thank my family for their constant support and encouragement throughout my academic journey. Their unwavering belief in me has been a source of motivation and inspiration that has kept me going during the challenging times.

I am also deeply grateful to the research group of the CSC laboratory for their warm welcome and support. In particular, I would like to thank Nadir Dalla Pozza for his guidance and assistance in navigating the intricate world of MPAI. Without his help, I would have been lost and unable to overcome the obstacles that I encountered.

I would also like to express my gratitude to Niccolò Pretto for his supervision of my work. His guidance and expertise have been invaluable throughout the entire research process.

Finally, I would like to extend a special thank you to Andrea Pietracaprina and Sergio Canazza, who first introduced me to the CSC laboratory. Without their help, I would not have had the opportunity to work alongside such talented and dedicated researchers.

Once again, I extend my sincere thanks to all those who have supported me on this journey. Your contributions have been invaluable, and I will always be grateful for your help and encouragement.

ABSTRACT

In recent decades, archives have seen a rapid change in the media used to store sound information, and many of these media are rich in obsolete material that risks becoming unusable due to aging. Therefore, it is necessary to digitize sound documents in order to make them durable over time. However, during the digitization process, errors such as applying an incorrect equalization curve or playing back the tape at the wrong speed can lead to the acquisition of inauthentic material.

This work focuses on studying the detection of possible errors due to incorrect equalization curve settings and tape playback speed during the transfer of material from analog to digital, verifying if and how it is possible to detect them using methods specific to Artificial Intelligence (clustering and classification).

The results of this research demonstrate that these algorithms may offer good precision in detecting errors and have the potential to automate the verification process, ensuring the preservation of valid information for a longer period of time, but before they can be used in a real-world scenario, they must be further improved.

TABLE OF CONTENTS:

> I want to make one minor quibble about the idea that there's Mathematics behind anything: there is no math behind anything. Math is one of the languages that we use to describe the world, computational languages are also languages that we use to describe the world.
>
> —Allen Downey, Scipy Conference, 2015

This thesis is written as a narrative that aims to investigate the possibility of recognizing specific anomalies of magnetic tapes that are created during the reproduction phase through machine learning, and to study how to integrate the models into a software workflow. The need for a standard that brings order in the current scenario of software that makes use of AI is presented in Chapter 1, while Chapter 2 describes the possibilities of representing a signal in a digital way by presenting the particular case of the preservation of magnetic tapes. In addition to representing the tapes digitally, metadata linked to the specific medium must also be included. Chapters 3 and 4 build on the knowledge presented in the previous chapters to conduct experiments using machine learning algorithms for clustering and classification.

The analysis section contains the results of the experiments carried out to test the models. The results are presented in the form of tables and graphs, which are explained in the text. The python notebooks used to generate the results are available in the project repository on GitLab, where also a usage guide is available and the API documentation. An html version of the thesis with integrated notebooks and API guide is hosted at https://matteospanio.gitlab.io/mpai-audio-analyser.

Conventions used

Bibliographic references are indicated by a specific number in square brackets like [7], which is then mapped in the bibliography. If the reference is inside a paragraph and is the first time it appears in the text, the number in square brackets is preceded by the author name as this Downey [7].

The references to figures, tables and code listings are indicated by the number of the object:

- this is a reference to a table, Table 3.11,
- this is a reference to a figure, Fig. 3.6,
- this is a reference to a code section, Listing 4.1

Inter-chapter references are composed by the title of the section referred and the page number specification between round braces, like this: *MPAI* (page 3).

The target reader of this thesis is expected to have prior knowledge of machine learning algorithms for clustering and classification, while aspects of signal processing and audio analysis are briefly introduced in the text to provide context.

CHAPTER

ONE

MPAI

> The nice thing about standards is that you have so many to choose from.
>
> —Andrew S. Tanenbaum

In recent decades, artificial intelligence (AI) has emerged as a transformative technology with a wide range of applications in various industries. The use of AI in everyday-use software has become increasingly common, and its impact is felt across a wide spectrum of human activities. AI technologies have been used to improve the efficiency of industrial processes, enhance the accuracy of medical diagnoses, optimize the performance of financial markets, and even help us to make better decisions in our personal lives. However, the rapid proliferation of AI-based software applications has led to a range of challenges that must be addressed if the full potential of this technology is to be realized.

One of the key challenges that arise from the proliferation of AI-based software is the lack of a consistent and standardized development methodology. Each software project tends to establish its own development methodology, which creates a lot of confusion on the general scene for interoperability between various software. In fact, the development of AI-based software is a complex and challenging process that requires a deep understanding of the underlying technologies and a rigorous approach to the development process. This often leads to a situation where each software project has to start from scratch in establishing its own development methodology, which can lead to significant delays, increased costs, and reduced interoperability between different software applications.

The lack of standards is not only a problem for software developers but also for end-users who are often unaware of the underlying technology and its limitations. End-users expect AI-based software to be reliable, accurate, and safe, but without a standardized development methodology, there is no guarantee that these expectations will be met[1] . This creates a significant risk for end-users who may unwittingly rely on AI-based software applications that are not well-designed or well-tested, leading to potential safety hazards, loss of privacy, and other negative outcomes. To address these challenges, there is a need for a standard to establish some firm points in the development of AI-bound software. A standard would help to provide a clear and consistent framework for the development of AI-based software applications, ensuring that they are reliable, accurate, and safe for end-users. It would also help to reduce the time and costs associated

[1] Unfortunately, another very common problem is the fact that there is still a lot of misinformation regarding the world of AI. Too often, there is a tendency to “humanize” machines, expecting them to be able to produce judgments autonomously, to process thoughts. One aspect to consider would be to adequately inform the general public, in this way a proper use of the tools provided by experts in the field would be spread, generating greater overall satisfaction and awareness of the tools being used.

with the development process by providing a common set of guidelines that can be used across different software projects. In addition, a standard would help to promote interoperability between different software applications, allowing for greater collaboration and innovation in the AI industry.

Overall, the need for a shared view of things in the development of AI-bound software is clear, it would be a concrete improvement in a world where the coming years are expected to be pervaded by the growth of smart assistants, self-driving vehicles and who knows what else. By addressing these challenges, a standard would help to unlock the full potential of AI-based software and ensure that it is used in a way that benefits society as a whole.

The MPAI project has been proposed as a solution to this problem, with the aim of developing AI-enabled data coding standards[2] . The MPAI - *Moving Picture, Audio and Data Coding by Artificial Intelligence* - project proposes a range of standards for products, applications and services that make central use of Artificial Intelligence. The project comprehends many areas affected by AI as health data coding, context-based audio enhancement, connected autonomous vehicles, AI-enhanced video coding, multimodal conversation, server-based predictive multiplayer gaming, and many more. The MPAI standards are designed to be modular and flexible, and are based on entities called AIFs (AI-Frameworks) that are themselves constituted by AIMs (AI-Modules). Each AIM is designed to perform a specific AI task. The AIF provides a framework for organizing and combining these AIMs to perform more complex tasks. MPAI provides the semantic and syntax of input and output data between each AIM and the AIF, defining software pipelines based on data sharing and exchange[3] .

In conclusion, the MPAI project is a significant step forward in the development of AI-bound software standards. Its modular and flexible approach provides a powerful framework for developing customized solutions for a wide range of applications.

As it stands to reason, MPAI therefore establishes a series of technical specifications which must then be implemented and become part of a common ecosystem as specified below:

- MPAI provides Technical, Conformance and Performance specifications
- These are implemented
- MPAI verifies implementations via its Performance Assessors
- The MPAI Store incorporates the verified implementation and distributes it to end users.

Specifications are issued in the form of AI Framework (AIF) like the one in Fig. 1.1 and those AIF are composed by many basic processing entities called AI Modules (AIM).

[2] MPAI was born in September 2020 from the ashes of the MPEG project, which was closed in June of the same year. The idea behind the establishment of MPAI is that the experience of MPEG has seen the birth of standards that have revolutionized the audio and video industry of the last thirty years, allowing easy intercommunication between various producers. With the advent of AI, we are witnessing rapid changes accompanied by general chaos in the background. MPAI's objective is to bring order to this scenario by proposing a common way of addressing current issues.

[3] The concept is really similar to a pipeline: the standard defines what can enter and what can exit the pipeline, but it does not specify how the data is processed (a implementation guide is provided but it is not mandatory). This is left to the discretion of the implementer, who can choose the best algorithm for the task at hand.

[4] This figure refers to the MPAI-AIF v1. At the time of writing, the MPAI-AIF v2 is under development and it should introduce a new layer all over the AIF adding security support to MPAI-AIF v1. Since the discussion of the MPAI-AIF specifications is outside the scope of this section and version 2 does not differ much from version 1, the figure illustrates the MPAI-AIF v1 for the sake of simplicity.

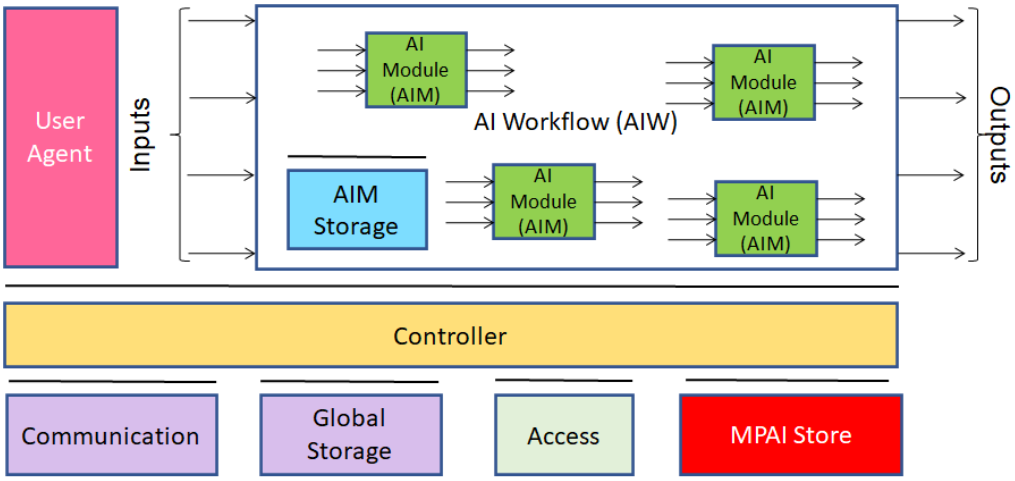

Fig. 1.1: MPAI-AIF Reference Model[4]

MPAI has many lines of standard development, but, at the moment, only few of them are under active development. Table 1.1 shows the current status of the MPAI standardization lines.

Table 1.1: MPAI Standardization Lines

Standard	Status	Description
MPAI-AIF	Active	Artificial Intelligence Framework
MPAI-AIH	Active	Artificial Intelligence for Health data
MPAI-ARA		Avatar Representation and Animation
MPAI-CAE	Active	Context-based Audio Enhancement
MPAI-CAV		Connected Autonomous Vehicles
MPAI-CUI	Active	Compression and Understanding of Industrial Data
MPAI-EEV		End-to-End video coding
MPAI-EVC	Active	AI-Enhanced Video Coding
MPAI-GSA		Integrative Genomic/Sensor Analysis
MPAI-MCS		Mixed-Reality Collaborative Spaces
MPAI-MMC	Active	Multimodal Conversation
MPAI-NNW		Neural Network Watermarking
MPAI-OSD		Visual object scene description
MPAI-SPG	Active	Server-based Predictive Multiplayer Gaming
MPAI-XRV		XR Venues

Each of these lines has its own use cases, which are the applications that the standard is intended to support. Of course, due to the dimension of the project, different research groups are working on different lines.

1.1 Context-based Audio Enhancement

The Context-based Audio Enhancement (MPAI-CAE) is a collection of 4 use cases that share common characteristic concerning the improvement of the user experience for audio-related applications. [4]

The 4 use cases considered are:

1. Emotion Enhanced Speech (EES). In the field of speech synthesis the next step is to make the speech sound more natural including the emotional aspects of the speech. The implementations of this standard can be used to create virtual assistants that can express emotions and feelings in a more natural way.
2. Audio Recording Preservation (ARP). The conversion from analog to digital audio is a process that can introduce artifacts in the audio signal and must take in consideration also some aspects of the physical support where can be found annotations and other information. This standard provides a structured way to preserve the original audio signal and the information related to it.
3. Speech Restoration System (SRS). The purpose of this use case is to restore damaged Audio Segments containing speech from only one speaker (the audio can be fully or partially damaged).
4. Enhanced Audioconference Experience (EAE). This use case improve auditory experience in an audioconference, in fact often the undertandability of the speech in a conference can be compromised by the presence of background noise or a not optimal environment. This standard provides a way to improve the audio quality of the conference.

1.2 Centro di Sonologia Computazionale

Over the past two decades, Centro di Sonologia Computazionale (CSC), the Sound and Music Computing laboratory of the Department of Information Engineering at the University of Padova, has been actively engaged in research on the preservation of historical audio documents. Given the multifaceted challenges associated with this task, a multidisciplinary approach has been adopted to fully leverage the vast potential of this documentary heritage. The methodology developed by CSC over the years has focused on both active preservation of historical audio documents and enabling access to them, with particular emphasis on analog magnetic tapes. The efficacy of this methodology has been tested and validated through various international projects undertaken in partnership with esteemed European audio archives, including the Paul-Sacher-Stiftung in Basel, the Fondazione Arena di Verona, the Historical Archive of the Teatro Regio di Parma, and the Luigi Nono Archive in Venice. [2]

Unlike passive preservation, which pertains to safeguarding the material structure of the documents, active preservation aims to preserve their content in digital form. This involves digitizing the tapes and ensuring safe transfer of identical copies from one digital carrier to another. Several factors must be considered during the digitization process, including the material structure of the object, which encompasses the physical-chemical components, technology, production system (acoustic, electroacoustic, magnetic), and audio and playback format (such as speed and equalization). Additionally, the primary information, which is the recorded audio signal, and

secondary informations, such as notes on the box, noise signals characterizing the recording system, alterations of the carrier (corruptions, splices, signs), and other metadata, including the history of the document transmission (storage, duplication), must be preserved. All these metadata must be stored alongside the digital audio in preservation copies, which are organized data sets that group all the information (data and metadata) represented by the source document and are stored and maintained in several directories of the archive data center. [14]

The proposed methodology seeks to enhance the reliability and scholarly suitability of digital preservation copies by taking the process a step further. Specifically, the software tools developed emphasize the "textual" aspects of a sound document, treating the A/D transfer as a philological operation of *restitutio textus*. This approach is particularly important in the realm of electroacoustic music on analog magnetic tapes. During A/D transfer of an audio document, digitization errors related to speed, equalization, and track numbers can occur, but the loss of useful ancillary information can also result in the creation of document "witnesses" with limited philological value. These document witnesses are non-identical digital audio documents with variants or differences in comparison to the original analog tape. Although they may represent a rough approximation of the original, these variants generate "noise" in the textual critic's task, rendering them imperfect and of poor quality. Therefore, the proposed methodology seeks to address these issues and produce more reliable and accurate preservation copies by focusing on the philological aspects of sound documents.

The methodology developed by CSC is based on the following principles:

1. convert analog magnetic tapes to digital audio files;
2. capture a video recording of the playback head of the tape recorder;
3. listen to the audio recording and take notes on the presence of anomalies or irregularities in the audio signal;
4. analyze the video recording to detect and locate the presence of irregularities in the audio signal;
5. collect and store the metadata related to the audio document.

It may be pertinent to inquire whether it is truly essential to automate the process of digital acquisition of tapes, given that an expert operator is expected to be capable of appropriately configuring the playback parameters for individual tapes, based on their known equalization curves and speeds. However, the obstacle lies in the fact that tape rewinding is frequently a protracted operation, and if this task is carried out in a consecutive manner, there exists a potential for errors to occur owing to fatigue or distraction. Additionally, the complexity of the problem exceeds superficial appearances since a single tape may contain multiple recordings of disparate materials, each with their own unique velocity and equalization curve.[5]

[5] It is common practice to reuse the same tape for multiple, disconnected instances in order to optimize its usage, and it may even be used by different individuals, rendering the content highly variable. For instance, it is not uncommon for a tape to be utilized for recording only a few minutes of audio, leaving the majority of the reel unused. Subsequently, on separate occasions, the tape may be reused, possibly played in reverse to avoid having to rewind it to the first available moment. Additionally, when only limited materials were available and the sole tape had only a few minutes of available recording space, the playback speed was set at a lower level to extend the available tape space. This set of recording practices results in a highly delicate and intricate digitization process for tapes, which requires meticulous attention to detail. Automating this process thus enables an expedited workflow while applying necessary corrections, where needed, in a precise and reproducible manner.

1.2.1 CSC Methodology as a Standard for Audio Preservation

The methodology developed by the Centro di Sonologia Computazionale (CSC) for the preservation of historical audio documents has been recognized for its effectiveness and reliability. As a result, it has been adopted by the MPAI-CAE Audio Recording Preservation (ARP) use case for implementation in the laboratory. This implementation will provide a structured approach to preserve historical audio documents, ensuring their reliability and scholarly suitability while also enabling access to them.

The standard includes various stages of processing, such as digitization of the analog audio signal, detection of irregularities, restoration of audio files, and packaging of the final output.

Specifically, given the following inputs:

- A Preservation Audio File, which is a digitized copy of the original audio recording.
- A Preservation Audio-Visual File produced by a camera that records the playback head of the magnetic tape recorde.

The ARP-AIF produces Preservation Master Files (a copy of the input) and Access Copy Files, which contain the processed and restored audio signal. A detailed description of the ARP-AIF can be found in Fig. 1.2.

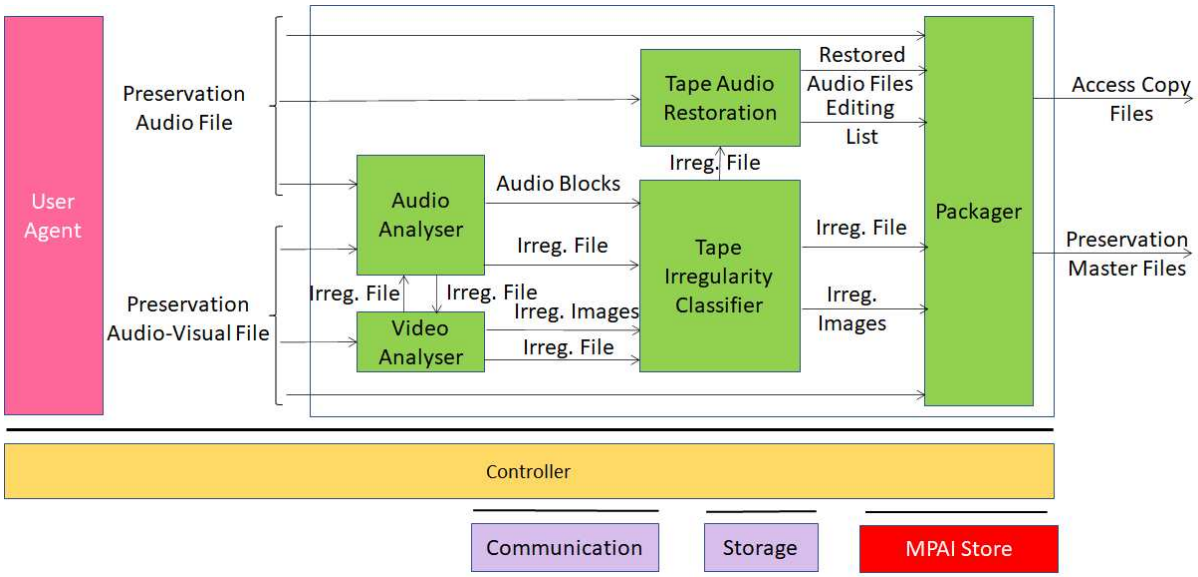

Fig. 1.2: ARP, AI Framework

One of the key components of the MPAI-CAE ARP standard is the Audio Analyser, which is responsible for detecting irregularities in the digitized audio signal and extracting the corresponding audio files. The Audio Analyser performs this task by comparing the digitized audio signal with a reference signal to identify any deviations in terms of speed or equalization.

In this thesis, the main focus is the implementation of the Audio Analyser, and in particular, it has been taken in consideration the use of machine learning algorithms to determine the original equalization and speed of the audio tape. The input of the system is the digitized audio signal. The output of the system is a set of parameters that describe the equalization and speed of the original recording, which are used to restore the audio files to their original quality in next AIMs.

CHAPTER

TWO

MANAGING DIGITAL AUDIO

> Digital audio is like painting with numbers.
>
> —Kim Cascone

According to physics, sound is a traveling vibration, i.e. a wave that moves through a medium such as air. The sound wave transfers energy from particle to particle until it is finally "received" by our ears and perceived by our brain. The two fundamental parameters to describe a sound are the amplitude (also known as volume) and the frequency (the measure of the number of oscillations of the wave per unit of time). [15]

Recent technological growth has led to a marked improvement in the speed and density performance of circuits and memories, making it possible to digitally represent large amounts of data, including acoustic signals. Specifically, the digitization of sound has led to a series of transformations since the 1980s, which have affected both professionals and music users. From the introduction on the market of the first CD for commercial use in 1982 to today, we have witnessed the birth (and end) of numerous digital media (Digital Audio Tape, MiniDisc, USB, DVD, HDD, SSD, Cloud). [3]

This has allowed the treatment and numerical processing of digital signals to take on a clear preponderance over the analog one: a sequence of numbers representing the amplitude of the signal at precise and discrete instants of time is much more precise and reliable than an approximation captured on a magnetic tape. [9] Therefore, it was necessary to design systems capable of converting analog sound into a succession of values that describe the various parameters of sound such as pitch, intensity, and timbre. Observing the Cartesian representation of a sound wave in the time domain, it can be seen that the ordinate axis describes the amplitude (intensity), and the abscissa axis highlights the frequency at which the wave moves (pitch). The conversion of sound from analog to digital, therefore, takes place on the two aforementioned levels: we will speak of *sampling* for the frequency and *quantization* for the amplitude. The timbre is instead strictly related to the representation of the sound in the frequency domain, and it can usually be analyzed by applying a Fourier Transform to the digitized signal.

2.1 Sampling

Sampling is the discretization of the analog signal over time. [9]

In other words, sampling is used to convert the time-varying continuous signal into a discrete sequence of real numbers. [8] The interval between two successive discrete samples is the sampling period (T). We will speak of sampling rate ($sf = \frac{1}{T}$) as an attribute describing the sampling process.

Since the representation of the signal is given by the variation of the amplitude over time, sampling corresponds to the periodic identification of the presence of the signal on the abscissa axis. In detail, to be able to correctly describe a signal, it is necessary to sample at least one point in the positive phase and one in the negative phase, otherwise we would incur the loss of essential values for recreating the wave, and in the interpolation phase we would create signals not originally present or others would be lost.[1]

Obviously the shorter the time intervals between one sample and the next, the more similar to the original analog the sampled sound will be. At the limit, for infinitely short time intervals, the digital signal will correspond to the real one. Generally speaking, it has been observed that to avoid the loss of information, the sampling must take at least two samples for the partial maximum frequency present in the signal ($sf = 2 \times freq_{max}$).[2]

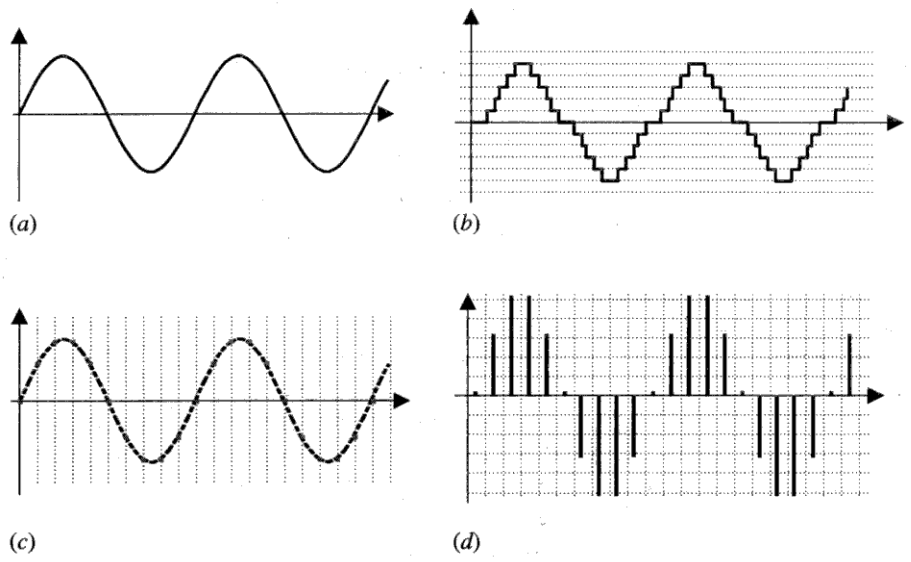

Fig. 2.1: a Analogic signal. b Quantization. c Sampling. d Quantized and sampled.

[1] The so-called phenomenon of aliasing or subsampling.

[2] This rule has been defined by the *Nyquist-Shannon sampling theorem* in 1949, even if it would be more correct to call it the Whittaker-Nyquist-Kotelnikov-Shannon (WNKS) theorem, according to the chronological order of those who have proved increasingly generalized versions of it . In any case, it follows that the standard sampling frequency is 44100 Hz, i.e. double 22050 Hz, which is the maximum frequency audible to the human ear (Lombardo and Valle [9])

2.2 Quantization

Quantization is a critical step in the digital signal processing of audio signals. It involves discretizing a continuous-time signal, such as an audio waveform, by mapping each sample value to the nearest value in a set of discrete values int the y-axis. The number of bits used to represent each quantized sample determines the resolution of the quantization process. The higher the number of bits used, the more accurate the approximation of the original signal will be.

In digital audio, the most common bit depth for quantization is 16 bits per sample, which provides 2^{16} possible levels of amplitude for each sample, in a range that varies from -2^{15} to 2^{15} (1 bit used for the sign). This level of precision is sufficient for most audio applications, including music recording and playback. However, higher bit depths are used for specialized applications, such as audio mastering, where the goal is to preserve the maximum amount of dynamic range and detail in the original recording (standard bit encoding are: 16, 24 and 32).

Quantization can introduce errors into the audio signal, known as *quantization noise*. This noise is the difference between the original signal value and its quantized approximation. The amount of quantization noise depends on the number of bits used for quantization, with higher bit depths resulting in lower levels of quantization noise. [9] It is worth noting that the choice of the number of bits used for quantization can have a significant impact on the overall audio quality. If the bit depth is too low, the quantization error can introduce audible artifacts, such as distortion or noise. On the other hand, using a high bit depth can increase the size of audio files and require more processing power to handle the data.

The *Signal-to-Quantization-Noise-Ratio* (SQNR) is a measure of the quality of the signal based on the *quantization noise* that has been introduced during the digitization process: it measures the ration between the original signal power and the max amplitude that can be achieved by the quantization noise.

2.3 Pulse Code Modulation

The final step of digitization, which incorporates the processes of quantization and sampling, is the generation of the code associated with the sample.

Digital audio signals are commonly encoded using PCM, with the linear pulse code modulation (LPCM) being the most widely used form. In LPCM, the continuous analog signal is first sampled at a specific rate, resulting in a discrete set of samples. These samples are then quantized into a series of numerical values that can be represented using a fixed number of bits, usually ranging from 8 to 32 bits per sample. The number of bits used per sample determines the dynamic range of the digital signal, with a higher number of bits resulting in a greater dynamic range and better signal-to-noise ratio.

To illustrate how PCM encoding works at a bit level, consider the following example. Suppose we have a PCM-encoded audio signal sampled at a rate of 44.1 kHz and quantized with 16 bits per sample. This means that for every second of audio, there are 44,100 sample points, with each sample represented by 16 bits. Each sample value represents the amplitude of the audio signal at that particular moment in time.

For instance, let's take a sample of the PCM-encoded audio signal at a specific time point, which has a value of 0110001011010101 in binary, or 16309 in decimal. This value represents the amplitude of the audio signal at that time point, with higher values indicating a louder sound and lower values indicating a quieter sound. The PCM-encoded signal can be decoded back into an analog signal by reversing the quantization and sampling process. In Fig. 2.2 you can see the graphical representation of PCM: the Roman numerals indicate the succession of samples, and for each sample (of 3 bits) there is a binary number which represents its amplitude.

PCM encoding is used in many digital audio formats, including the Waveform Audio File Format (.wav) and Audio Interchange File Format (.aiff). These formats use PCM encoding with various bit depths and sampling rates to store and transmit digital audio data. PCM encoding is also used in digital communication systems, such as digital telephone networks and digital television broadcasting. [3]

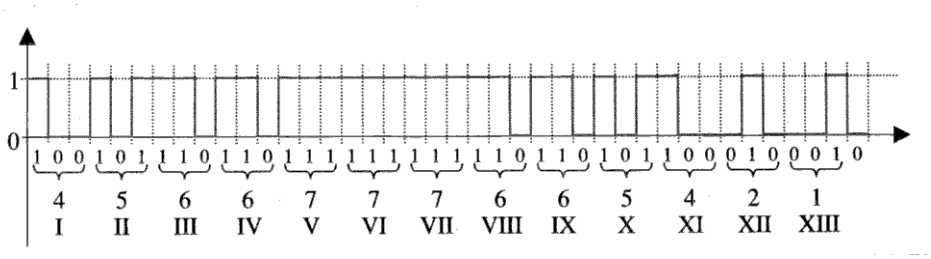

Fig. 2.2: Pulse Code Modulation

2.4 The frequency domain

In addition to representing sound in the time domain, it is also possible to represent its properties in the frequency domain. This system was studied by Charles Fourier, who defined the process of converting from time domain representation to frequency domain representation, which is called the Fourier Transform (FT). When the starting signal is in digital format, the Discrete Fourier Transform (DFT) can be applied. The underlying idea behind the DFT is that the spectrum is sampled in frequency just as the digital waveform is sampled in time.

Mathematically speaking, the relationship between N samples in the time domain $x_0, x_1, ..., x_{N-1}$ and N complex numbers of the Discrete Fourier Transform $X_0, X_1, ..., X_{N-1}$ is described by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-ik\frac{2\pi}{N}n} \quad k = 0, 1, ..., N-1 \tag{2.1}$$

where i is the imaginary unit, $e^{\frac{2\pi}{N}}$ is the N -th root of unity, and k is the frequency index. [18]

Essentially, the complex numbers X_k represent the amplitude and phase of different sinusoidal components of the input signal x_n. The calculation of the sum requires $O(N^2)$ arithmetic operations, and to optimize the algorithm's performance, the Fast Fourier Transform (FFT) was developed [5], which calculates the same result with a number of operations of $O(Nlog(N))$, making the summation calculation much faster. In fact, the DFT and FFT algorithms are an essential element in digital signal analysis. [9]

2.5 A case study: Audio Tapes

Fig. 2.3: Studer A807 Reel to Reel Tape Recorder.

The invention of magnetic tape for audio recordings dates back to 1928, when Fritz Pfleumer, a German inventor, created this innovative technology. Reel-to-reel audio tape recordings quickly became the primary recording format used by professional recording studios until the late 1980s, resulting in numerous sound archives preserving a large number of audio tapes.

However, like any analogue carrier, magnetic tape is also subject to physical degradation, which can be slowed down but not entirely prevented. Therefore, digitization becomes essential to prevent degradation that can render the information inaccessible. Additionally, magnetic tape is closely linked to its playback device: the reel-to-reel tape recorder. Before pressing the play button, the machine must be configured correctly to play back the recordings on the tape, and any error may cause audio alteration and the loss of the preservation copy's authenticity. [10]

Two main parameters must be configured for optimal playback: reel replay speed and equalization.

Regarding the equalization parameter, during the recording process, the source signal is modified by applying equalization that alters the frequency response (application of a pre-emphasis curve) to maximize the Signal-to-Noise Ratio (SNR) of the recorded signal. This alteration must be compensated for during the reading of the tape by juxtaposing an inverse curve (post-emphasis curve) to obtain the original audio signal. The main standards adopted are CCIR, also known as IEC1, mostly used in Europe, and NAB, alternatively called IEC2, mostly adopted in the USA. It is worth noting that curves of the same standard may differ according to the reel speed.

As per standard definitions, the equalizations are the product of two curves:

$$N(DB) = 10\log\left(1 + \frac{1}{4\pi^2 f^2 t_2^2}\right) - 10\log\left(1 + 4\pi^2 f^2 t_1^2\right) \tag{2.2}$$

where f represents the frequency in Hz, and t_1 and t_2 are time constants measured in seconds. In CCIR, the curve shape is solely dependent on the constant t_1, while t_2 is set to infinity. Moreover, the time constants alter concerning the tape speed. The equalization and tape speed must be appropriately set to achieve a flat frequency response. [14]

Often, carriers do not indicate the speed and equalization standards, and lack of documentation may necessitate the operator to depend on auditory cues. However, previous experiments have demonstrated that this approach is prone to errors. [14] To avoid subjectivity and potential errors that may compromise the preservation copy's accuracy, a software tool that can identify the correct equalization is the solution. This not only aids operators in the digitization process but also benefits musicologists who can verify the correctness of an unknown provenance digitized copy and rectify any mistakes if necessary.

CHAPTER

THREE

DATA ANALYSIS

> Without data, you're just another person with an opinion.
>
> —W. Edwards Deming

Given the abstractions that allow representing the physical reality of signals in the digital world, it is now possible to analyze how they are integrated into the data analysis process and the implementation of AI modules within the AIF ARP. As previously mentioned, the central points of investigation in this thesis are:

- the project for implementing the Audio Analyser AIM within the AIF ARP;
- the study, through clustering and classification, of the possibilities of automating the recognition of errors resulting from incorrect application of speed or equalization filters during the acquisition of an audio tape.

In this chapter, we are going to evaluate the possibilities and capabilities of automating the detection of such errors based on studies that have been previously conducted on the topic in Micheloni *et al.* [10] and Pretto *et al.* [14].

One of the fundamental issues is the type of data that needs to be processed. So, before explaining in detail the procedures used and the results, it is necessary to study a methodology for sound analysis and understand which representation is most suitable for our purpose. In Rodà *et al.* [16], a scheme is provided, whose main parts are shown in Fig. 3.1, which illustrates the different phases of sound analysis. A characteristic aspect of this type of data is *segmentation*, which involves breaking the original signal into multiple parts. In fact, considering audio files with a duration of 10 minutes rather than fragments of 10 milliseconds can strongly influence the results of the investigation[1] , especially if the signal being considered is highly variable over time. Therefore, in addition to the sampling frequency, it is necessary to establish the frequency at which the signal is analyzed or to establish terms by which to divide the signal into parts in order to obtain homogeneous events. In any case, the final result is an approximation of the signal, as each segment must be considered as a "snapshot" at a specific moment of the input, which is assumed to be constant[2].

[1] The first option is called *long-time* analysis, often used to analyze the structure of an entire piece of music or for signal that don't vary much over time. In contrast *short-time* analysis is use where the signal varies quite often. In addition considering a 10 minute audio file as a whole it would be interpreted as a single sample, while the same file splitted in 10ms chunks would generate 60000 samples; since in data analysis the sample size matters a lot, it is important to consider the right size of the sample in relation to the quantity of data available.

[2] For example, if one considers voice recording, the assumption that the signal is constant is justified by the time it takes for the body to move the larynx, mouth, and all the other muscles and organs involved in the process, whose changes are not faster than a few hundred milliseconds, so the signal can be considered constant for a period of time of about 100-200 milliseconds.

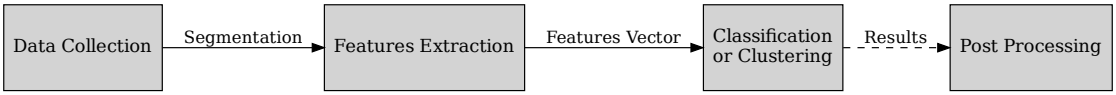

Fig. 3.1: Scheme for sound analysis.

The second really peculiar aspect of audio data are their features, there are many and each one has its own characteristics. In [16] can be found a detailed description of the most common features used in audio analysis, but in this thesis we are going to focus only on *Mel Frequency Cepstral Coefficients* (MFCCs) which have been used in the studies mentioned above. Just for general knowledge it is important to say that there exist two main types of features: *spectral* and *temporal*. The first type is based on the frequency spectrum of the signal, while the second is based on the time domain. In the case of the spectral features, the most common are the MFCCs.

The MFCCs relies on a chain of transformations of the input signal: first, the signal is transformed into the frequency domain, then the frequency spectrum is transformed into a Mel scale, and finally, the Mel spectrum is transformed into a cepstral domain. Initially, the signal is converted in the frequency domain via DFT (see *The frequency domain* (page 12)), then the frequency spectrum is converted into the Mel scale, which is a logarithmic scale of frequency correspondant to the response of humans' ears to frequency[3] :

$$\text{mel}(f) = \begin{cases} f & \text{if } f \leqslant 1000 \text{ Hz} \\ 2595 \log_{10}\left(1 + \frac{f}{700}\right) & \text{if } f > 1000 \text{ Hz} \end{cases} \tag{3.1}$$

this operation changes the unit of measure of the frequency by applying a filter-bank to the signal[4] , where each filter is centered on a specific frequency and has a width that is proportional to the distance from the center frequency. So from a smooth spectrum, the signal is transformed into a series of peaks, each one corresponding to a specific frequency. At this point would be interesting to separate the Mel spectrum into multiple components: that's the concept of *cepstrum*[5]. This is a general method to divide the oscillatory-harmonic components (the base signal) of a sound from its timbric components (the filter-bank). If we think of a signal in the frequency domain as the multiplication of this two components it can be mathematically expressed as $y(n) = x(n) \cdot h(n)$ where the spectrum y is the resultant from x, the oscillatory-harmonic component, and h, the timbric component. [16] A further step can be done due to the logarithmic representation of the spectrum, which, thanks to the property of logarithms $\log(a \cdot b) = \log(a) + \log(b)$, consents to pass from a multiplication to a sum, then the discrete time cosine transform (DCT) is

[3] Long story short: in the occidental music theory system the distance between two octaves is always the double in frequency of the lower note ($A_3 = 220$ Hz, $A_4 = 440$ Hz, $A_5 = 880$ Hz), but the space between each octave is always divided linearly in twelve semitones and the human ear perceives these distances as equally distributed. The Mel scale is a logarithmic scale that tries to reproduce the same perception of the human ear, it applies a filter-bank to the signal and then it applies a logarithmic scale to the result. The Mel scale is used in audio analysis because it is more similar to the human perception of sound than the linear scale.

[4] A filter-bank is a traditional method for analyzing a signal's spectrum. It involves using a set of band-pass filters placed at regular intervals along the frequency axis to obtain the log-energies at the output of each filter. This provides a general idea of the signal's spectral shape and helps to minimize any harmonic structure that may exist. [12]

[5] This strange name is derived from inverting the first four letter of *spectrum*: this name reflects that it is the application of a transformation (usually the inverse FFT) to the spectrum, also the unit measure passes from frequency (Hz) to *quefrency* (seconds).

applied to the result[6] :

$$\text{DCT}(\log(|Y[k]|)) = \text{DCT}(\log(|X[k]|)) + \text{DCT}(\log(|H[k]|)) \tag{3.2}$$

where k is the k-th component of the filter-bank.

As result of this operations the direct formula to extract the i-th MFCC is:

$$C_i = \sum_{k=1}^{N} Y_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right] \quad i = 1, 2, \ldots, M \tag{3.3}$$

where $Y_1, \ldots, Y_N$ are the log-energy outputs of a mel-spaced filter-bank and N is the total number of channels in the filter-bank.

In [10], it has been shown that certain portions of audio are more significant than others. In particular, it has been noted that different types of silence (i.e., with different signal intensity) have a good descriptive capability of the considered signal to identify the correct playback speed and equalization curve. Table 3.1 illustrates the three classes of noise that have been identified and their intensity in decibels[7].

Table 3.1: Noise classes

Class	Intensity (dB)	Description
A	-50 to -63	Noise in the middle of a recording, i.e., silence between spoken words
B	-63 to -69	Noise of the recording head without any specific input signal
C	-69 to -72	Noise coming from sections of pristine tape

By extrapolating sections of silence with a length of 500 milliseconds[8] , it was proven that MFCCs contain sufficient information to train classifiers with accuracy very close to 100%. While in the study by [14] manually extracting sections of silence was carried out, in this case, a software was created that is capable of automatically identifying sections of silence and extracting MFCCs (with the idea of then integrating it directly into the AIF ARP Audio Analyser).

[6] The DCT is specific for the MFCCs calculation, the general method to evaluate the cepstrum is based on the inverse FFT instead.

[7] Actually, the intensity value is not exactly absolute, but it could slightly vary depending on the bit depth of the audio file, which can introduce quantization errors. However, the difference is negligible and the values reported in the table are correct for the majority of the cases except for the class C, in fact higher bit representaition of the signal have a higher dynamic range and can reach values under -72 dB (the minimum noise intensity at 16 bit is -96 dB, while at 24 bit is -144 dB).

[8] The length of the silence sections was chosen based on the results of the study in [10] and [14]. In those studies different lengths of silence sections were tested and it was found that longer sections did not enrich the results.

3.1 Datasets

In [10] the main dataset was artificially generated in laboratory. It was composed by samples that covered all the combinations of correct and incorrect filter chains that could occur during the digitization of audio tapes (including only 7.5 and 15 ips speed tapes).

In the current study the starting dataset is the same but includes also 3.75 ips speed. So all possible combinations for both reading and writing operations are: 3.75 ips with NAB equalization, 7.5 ips with CCIR or NAB equalization and 15 ips with CCIR or NAB equalization (it is not possible to record or reproduce an audio tape at 3.75 ips with CCIR equalization).

The research so has been done in two phases: first, recreate (and confirm) the study in [10] on the extended dataset (referred as *Pretto* or *Pretto dataset* in the rest of the thesis), and on second instance, test the trained models on a new dataset (referred as *Berio-Nono*) generated from real world data.

Once the audio tapes were defined, the data were collected from the digital files using the method described in the section above[1] (500 ms of silence samples), and then their first 13 MFCCs were computed. Table 3.2 shows the structure of the datasets. All kinds of noises were extracted from the audio files to see if any of them carried more information than the others. Of course, a greater amount of data allows for more coverage of the feature space, but it also increases the computational cost of the training process.

Table 3.2: Dataset features

Feature	type	description
label	string	A string that specifies the equalization curve and speed used in the recording and reproduction processes[2] (e.g. 3C_7N)
noise_type	string	A, B or C
MFCC1-13	float	The first 13 MFCCs of the sample

See also:

The feature extraction process for both dataset can be recreated executing the notebook `data_extraction.ipynb` at the thesis' repository , and can be summarized as follows:

1. execute the script `noise-extractor` on the audio files to generate a set of samples of 500ms each labeled with the type of noise contained in the sample (A, B or C);
2. extract the first 13 MFCCs from each sample;
3. save the results as a csv file containing the MFCCs, the noise type and a label that specifies the equalization curve and speed used in the recording and reproduction processes.

[1] A detailed description of how the software to extract silence can be found at *An Audio Analyser implementation* (page 33).

[2] The labels have been composed by a number that specifies the writing speed (3 for 3.75 ips, 7 for 7.5 ips and 15 for 15 ips), a letter that specifies the writing equalization standard used (C for CCIR and N for NAB), an underscore character, a number for the reading speed (analogue as writing) and a letter for the reading equalization standard used. For example the label `3N_7N` means that the sample has been recorded at 3.75 ips with NAB equalization curve and reproduced at 7.5 ips with NAB equalization curve.

3.1.1 Pretto

As already mentioned, this dataset consists of a collection of features extracted from a tape created ad hoc at different speeds and with different pre/post-emphasis curves (CCIR or NAB) for both the writing and reading processes. As explained in *Managing digital audio* (page 9), a tape can be recorded and reproduced with different equalization curves and speeds (3.75, 7.5, 15 ips). That's why this dataset was generated, as it contains all possible main combinations of speeds and equalization curves for the writing and reading processes of the tape. In particular, the tape is composed of a set of audio extracts, each lasting 10 seconds, and the interval between two extracts is composed of a 1-second beep sound, 2 seconds of silence, and another 1-second beep. The audio extracts have been taken from different sources and have been combined as shown in Table 3.3.

Table 3.3: Audio tape composition

Author	Title	Duration (s)
Taylor Swift	*Shake It Off*	10
The Weeknd	*Save Your Tears*	10
Richard Wagner	*Ride of the Valkyries*	10
Carl Orff	*Carmina Burana - O Fortuna*	10
Queen	*We Will Rock You*	10
Eagles	*Hotel California*	10
Bruno Maderna	*Continuo*	10
Bruno Maderna	*Syntaxis*	10
Luciano Berio	*Différences*	10
Bruno Maderna	*Musica su Due Dimensioni*	10
CLIPS project	*LP1f19bZ*	10
CLIPS project	*LP4m19bZ*	10
CLIPS project	*LP1f20bZ*	10
CLIPS project	*LP4m20bZ*	10
CLIPS project	*LP4m18bZ*	10

The eclectic intent is evident: different kinds of music have really different MFCCs (in fact, they are often used for automatic music genre recognition), so the dataset comprises classical, pop, and modern music and also speech recordings (CLIPS project has a public corpus of speech recordings).

Fig. 3.2 shows the distribution of the dataset taking into consideration different subsets based on noise type. It is noticeable that the dataset is not balanced: tapes recorded at a higher speed tend to have more noise samples when reproduced at a lower speed[3] . This imbalance could influence the training process. In fact, we will have a heterogeneous distribution of coverage of the feature space. For example, there are only 6 instances of noise A recorded at 7.5 ips with CCIR equalization curve and reproduced at 15 ips with NAB equalization curve, while there are 181 instances with the opposite label (7.5 ips CCIR recorded and 15 ips NAB reproduced). Unless

[3] Of course this isn't so inespected: reproducing a tape with a lower speed than the one used for recording it, the tape will play slower resulting to have longer noise section of the same intensity as the original ones. It is like stretching the tape: the speed of the tape is slower, the pitch is lower (the frequency has been altered), but the intensity of the sound cannot be alterated since it is interested only by the y axis.

the class distribution tends to stick to the same point of the feature space, this could lead to poor generalization of the model.

Fig. 3.2: The dataset distribution by noise type, y labels regard the recording processes, x labels regard the reproduction processes.

Table 3.4 shows a summary of the dataset with some statistics: labels are 25 because each speed can be recorded and reproduced with two different equalization curves except for 3.75 ips, so each tape can be recorded or played with one of 5 configurations: 3.75 ips NAB, 7.5 ips NAB, 7.5 ips CCIR, 15 ips NAB, 15 ips CCIR.

Table 3.4: Pretto Dataset summary

Feature	value
Samples	9058
Noise types	3 (A, B, C)
Equalization curves	2 (NAB, CCIR)
Writing speeds	3 (3.75, 7.5, 15 ips)
Reading speeds	3 (3.75, 7.5, 15 ips)
Number of labels	25
Samples of noise type A	2075
Samples of noise type B	5050
Samples of noise type C	1933

3.1.2 Berio-Nono

This dataset is composed by the digitization of real case scenarios tapes of unpublished works by Luciano Berio and Luigi Nono, in which the tape has been recorded and reproduced with the same equalization curve and speed. Once obtained the digital files, they have been used as input for the pipeline described above. The samples extraction procedure generated 12202 samples, and the features extraction process generated 12202 samples of 13 MFCCs each. The features in the dataframe are the same as in Table 3.2.

Fig. 3.3 shows the distribution of the dataset, which is unbalanced, but no evident pattern can be inferred from the distribution since the files were generated from a variable number of different equalized tapes: 4 tapes were recorded at 7.5 ips in NAB format, 6 tapes were recorded at 7.5 ips in CCIR format, 5 tapes were recorded at 15 ips in NAB format, and 3 tapes were recorded at 15 ips in CCIR format. The tapes considered for this dataset had a much longer duration (about half an hour each) compared to the Pretto dataset, which had audio tracks of only 10 seconds.

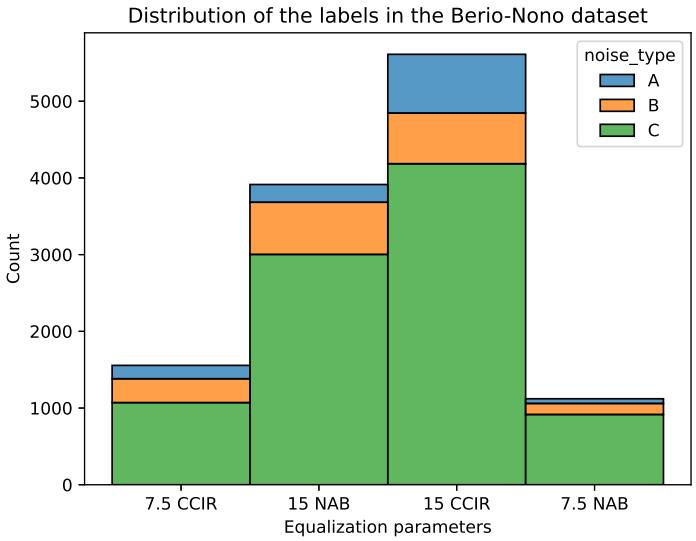

Fig. 3.3: The dataset distribution by noise type, each column represents a different equalization curve and speed used in the recording and reproduction processes.

Table 3.5 shows a summary of the dataset with some statistics. The labels are 4 because only the correct equalization curves and speeds have been applied in the acquisition step, so each tape can be recorded or played with one of the following 4 configurations: 7.5 ips NAB, 7.5 ips CCIR, 15 ips NAB, 15 ips CCIR. In this case, the extraction software found a noticeable quantity of C silence. It is not uncommon to find a great amount of unused tape; however, in this situation, it emerges that the characterization of the noise is missing a 4th kind of noise: when the acquisition process starts, there are a few seconds where the tape is not moving, and neither is the pristine tape. Therefore, the recorder is capturing noise from the environment, usually lower than $-72dB$. In this case, this noise from the 4th kind was included in the C noise type, omitting the lower limit for C noise.

Table 3.5: Berio-Nono Dataset summary

Feature	value
Samples	12202
Noise types	3 (A, B, C)
Equalization curves	2 (CCIR or NAB)
Writing speeds	2 (7.5, 15 ips)
Reading speeds	2 (7.5, 15 ips)
Number of labels	4
Samples of noise type A	1231
Samples of noise type B	1796
Samples of noise type C	9175

3.1.3 Comparision

The main goal of the entire analysis is to be able to recognize the pre/post-emphasis equalization curves applied to the tape. On the basis of the data we have, the most interesting experiment is therefore to create a classifier trained on the Pretto dataset, i.e. the one built ad hoc and which includes all possible cases, to recognize the examples taken from real cases of the Berio-Nono dataset. To do this, it makes sense to visualize how the data is distributed in space. It is clear that, given the premises, the Berio-Nono dataset should cover (approximately) a subset of the space of the other dataset, but this is not the case. Different factors could the cause of this results:

- the different nature of the tapes in the two datasets, while in Pretto there are short music and speech samples (10 seconds), in Berio-Nono the musical events last for minutes
- the machine used to acquire the tapes has to be recalibrated periodically, it could be that the conditions of the machine were not the same during the acquisiotion of the samples

Anyway it is interesting to observe that the noise classes are well recognizable along the x axis, this means that, even if the Berio-Nono dataset is shifted along the y axis in respect to the Pretto dataset, there are evident analogies between the two datasets, while there are groups of samples really far from the others which are probably outliers (a further analysis is needed to confirm this hypothesis, but it is not the purpose of this work).

Principal component analysis

The plots have been generated transforming the 13 MFCCs into 2D using the principal component analysis (PCA) algorithm. The PCA algorithm is a linear dimensionality reduction technique that uses singular value decomposition of the data to project it to a lower dimensional space [11]. At each step, the algorithm choise the axis that maximizes the variance of the projected data, and then it projects the data onto that axis. The algorithm is repeated until the desired number of dimensions is reached. This method gives the possibility to visualize complex distributions in a 2D space, anyway this kind of visualization could lead to wrong considerations due to the fact that we are cutting part of the informations.

Fig. 3.4: The features space of the two datasets, each point represents a sample, the color represents the noise type and the shape represents the dataset (Pretto or Berio-Nono).

3.2 Clustering

Note: Clustering results can be reproduced executing the `equalization_clustering` notebook in the `notebooks` section of the thesis' repository, where a detailed written and graphic explanation of the code for the process is provided.

To evaluate the quality of the data and the feasibility of the classification task, the dataset has been clustered using the K-Means and Hierarchical clustering algorithms as in [10].

As a measure of the quality of the clustering, the V-measure has been used. The V-measure is a measure of the similarity between two data clusterings, and it is defined as the harmonic mean between the homogeneity and completeness of the clustering. Homogeneity is the ratio between the number of pairs of samples that are in the same cluster in both data clusterings and the total number of pairs of samples that are in the same cluster in the first data clustering. Completeness is the ratio between the number of pairs of samples that are in the same cluster in both data clusterings and the total number of pairs of samples that are in the same cluster in the second data clustering. [11]

About the cluster scoring

There are many metrics to evaluate the quality of a clustering, anyway not always are all applicable to the specific clustering task. For example, the Davies-Bouldin index is a metric that evaluates the quality of a clustering by measuring the distance between clusters and the distance between points in the same cluster. This metric is not applicable to the equalization classification task since the clusters are often overlayed. In addition is useful to observe that in this specific case the ground truth is know in advance, so all the metrics that take advantage of this information are a good choice. Lastly the V-measure returns a value in the range $[0, 1]$ which make really easy to treat the results as if they were the accuracy of a classifier. For a more detailed explanation of the metrics see this article.

Since the data classes are overlayed the clustering and algorithm evalutation have been performed on different subsets of the dataset structured as shown in Table 3.6. For each subset has been specified the original dataset, the clustering task, the speeds used and the number of clusters[1] , i. e. if the speed is "7.5 ips, 15 ips", it means that all the considered classes have been extracted from recorded and played back tapes at 7.5 ips or 15 ips, regardless of the equalization (i.e., `7N_7N`, `7C_7C`, `7C_7N`, `7N_7C`, `15N_15N`, `15C_15C`, `15C_15N`, `15N_15C`).

The subsets A and B were constructed based on the datasets built in [10], where classification and clustering were only performed on data with the recording speed equal to the playback speed. The study demonstrated that, in general, machine learning algorithms achieve similar performance in identifying equalization curves at a single speed. This thesis aimed to go beyond that by introducing the E dataset, which combines samples from both subsets A and B, yielding encouraging results. Furthermore, the clustering of recording speeds is addressed by subsets C, D, and G, a problem not tackled in either [10] or [14].

[1] Both K-Means and Hierarchical clustering algorithms require the number of clusters as input, in this case the number of classes to be clustered.

There isn't a subset of Berio-Nono dataset to evaluate the clustering of equalization classes (wrong/right equalization) because the tapes have been recorded and played back at the same speed.

Table 3.6: Clustering Datasets

Dataset	Original dataset	Clusters	Speeds	# Clusters
A	Pretto	Right/Wrong equalization	7.5 ips	2
B	Pretto	Right/Wrong equalization	15 ips	2
C	Pretto	Speed	7.5 ips, 15 ips	2
D	Pretto	Speed	3.75, 7.5 ips, 15 ips	3
E	Pretto	Right/Wrong equalization	7.5 ips, 15 ips	2
F	Berio-Nono	Speed and equalization	7.5 ips, 15 ips	4
G	Berio-Nono	Speed	7.5 ips, 15 ips	2

The results of K-Means and Hierarchical clustering are shown respectively in Table 3.7 and Table 3.8. The scores higher than 0.5 are highlighted in bold.

Table 3.7: K-means Clustering results

Dataset	Noise A	Noise B	Noise C	All
A	0.2	0.086	**0.589**	0.166
B	**0.747**	0.413	0.046	0.301
C	0.09	0.015	0.145	0.008
D	0.172	0.177	0.123	0.084
E	0.377	0.215	**0.527**	0.194
F	0.176	0.108	0.22	0.156
G	0.056	0.008	0.043	0.024

Table 3.8: Hierarchical Clustering results

Dataset	Noise A	Noise B	Noise C	All
A	0.079	0.178	**0.972**	0.338
B	**0.948**	**0.567**	0.304	**0.514**
C	0.27	0.119	0.116	0.05
D	0.243	**0.535**	0.222	0.094
E	0.453	0.395	**0.760**	0.268
F	0.233	0.145	0.233	0.163
G	0.099	0.03	0.043	0.025

Subsets A, B, E have been used to evaluate the clustering of equalization classes (Fig. 3.5). Overall the clusters resulting by analyzing only tapes recorded and played at the same speed (dataset A and B) gave better results, from this it can be inferred that MFCCs describe in a quite good manner the difference between correctly equalized and non-correctly equalized tapes. The situation changes when clustering of wrong or correct equalization is performed on different speeds

(dataset E[2]), in this case the results are not so good. This latter fact is much more evident in the clusterization of subset F, where the clusters should highlight both the equalization and the speed of the tapes.

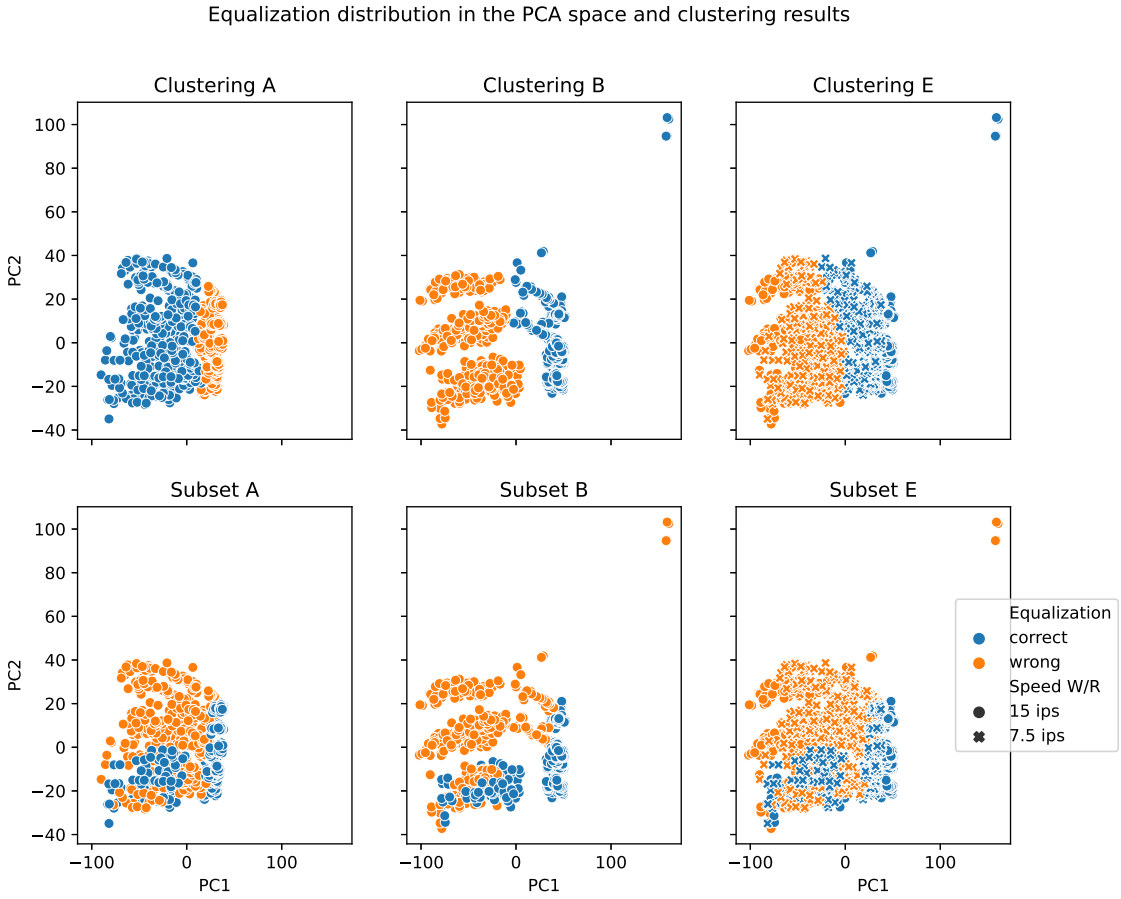

Fig. 3.5: Clusters distribution from dataset A and B. While considering only one speed at a time (7.5 ips or 15) the clustering performance is better, the dataset union generates instead more confusion.

The clustering results are not homogeneous over the different datasets, but, looking for trends, it can be said that, in general, Hierarchical clustering overperforms K-Means and when one algorithm gives an acceptable result also the other one does. Another fact is that good results came across different kind of noises, but the same noise also gave really bad results for other datasets, e. g. Noise C for dataset A and B: in the first one the score was 0.972 with Hierarchical clustering, while in the second one it was 0.304 with the same algorithm. From this fact can be inferred that for a effective classification of the tapes it can be helpful to use different kind of noise, and when an algorithm gives a bad result, it can be useful to evaluate it over another noise.

Another fact is that the results were bad where speed clustering was involved, this is a confirmation over previous studies, but should make us think about the possibility to use other features to cluster the tape's speed. MFCCs are a good choice for equalization clustering, but they are not

[2] Before the actual subset E, composed only by tapes recorded and played back at the same speed (7.5 and 15 ips), a preliminary study has been performed on the dataset union, i. e. all the tapes recorded and played back at different speeds, where the results were much worse than the ones obtained in dataset E. Due to drastically unuseful results the dataset E has then been resized to its actual shape.

so good for speed clustering, so it would be interesting to find other features that can be used to cluster the tapes based on their speed.

3.3 Classification

Note: Classification results can be reproduced by executing the `equalization_classification` notebook in the `notebooks` section of the thesis' repository, where a detailed written and graphic explanation of the code for the process is provided.

As for clustering analysis, the classification task has also been performed on different subsets of the datasets. The classification has been carried out using the K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees (DT) and Random Forest (RF) algorithms. The first three methods have already been used in previous investigations, while in this study, the Random Forest is introduced.

To evaluate the performance of the classification algorithms, the accuracy score has been used, which is the ratio of the number of correctly classified samples to the total number of samples.

About the accuracy scoring

The disadvantage of using the accuracy score is that it is insensible to heavily unbalanced dataset. For example, if in a classification task, 99% of the data belongs to class A, and 1% of the data belongs to class B, if the classifier puts all B istances in A, the accuracy will return a score of 0.99 (almost perfect classification) but, in reality, the classifier put everithing in only one class (so it is completely useless).

The classification has always been performed tuning the hyperparameters of the model using a grid search with cross-validation at 5 folds on 80% of the selected dataset. Once the parameters have been tuned, the model's performance is tested over the remaining 20% of the data.

In summary, the datasets are listed in Table 3.9. Since those datasets are used to train classification models, they are all subsets of the Pretto dataset, which comprehends more cases of wrong and correct equalizations and speeds.

Table 3.9: Classification Datasets

Dataset	Speeds	Equalization	Classes
H	7.5 ips	Mixed and correct	4
I	15 ips	Mixed and correct	4
J	7.5 ips	correct	2
K	15 ips	correct	2
L	7.5, 15 ips	Mixed and correct	2

The results obtained from perform classification over the 20% test data of each dataset subset are reported in Table 3.10, Table 3.11, Table 3.12 and Table 3.13.

The classification results show that all the models have very good performance on the test data, with accuracy scores ranging from 0.91 to 1.0 depending on the dataset and the algorithm used. The K-Nearest Neighbors and Support Vector Machines algorithms generally have the best performance, with accuracy scores of 0.98 or higher for all datasets except for Dataset H with KNN, where the accuracy score is 0.94. The Decision Tree and Random Forest algorithms have slightly lower performance than KNN and SVM, but still have accuracy scores above 0.9 for all datasets.

Table 3.10: KNN Classification results

Dataset	Noise A	Noise B	Noise C	All
H	0.94	0.91	1.0	0.99
I	1.0	1.0	1.0	0.98
J	0.88	0.96	1.0	1.0
K	1.0	1.0	1.0	1.0
L	1.0	0.99	1.0	1.0

Table 3.11: SVM Classification results

Dataset	Noise A	Noise B	Noise C	All
H	0.88	0.93	1.0	0.99
I	1.0	1.0	1.0	0.99
J	1.0	1.0	1.0	1.0
K	1.0	1.0	1.0	1.0
L	1.0	1.0	1.0	1.0

Table 3.12: Decision Tree Classification results

Dataset	Noise A	Noise B	Noise C	All
H	0.79	0.71	0.98	0.91
I	0.95	0.92	0.96	0.93
J	1.0	0.87	1.0	0.94
K	0.95	1.0	1.0	0.95
L	0.99	0.93	0.98	0.97

Table 3.13: Random Forest Classification results

Dataset	Noise A	Noise B	Noise C	All
H	0.88	0.87	0.99	0.97
I	1.0	0.98	0.96	0.98
J	0.88	0.91	1.0	1.0
K	1.0	0.94	1.0	0.99
L	1.0	1.0	0.99	1.0

3.3.1 A step further

The studies by [10] and [14] have been confirmed, and also the introduction of speed classification (subset L) gave an exciting result, allowing us to make a step further and test the classification task on wider datasets:

- study the classification performance on the entire Pretto dataset, giving as input all the possible combinations of speed and equalization curves (25 classes)
- test the effectiveness of the trained classifiers on Berio-Nono dataset

In this case, based on the previous results, it was decided to use only the KNN and RF algorithms since the former provided very similar results to those obtained with SVM and has a much lower training time, while the latter proved to be clearly more effective than a single decision tree and still has a relatively fast training time.

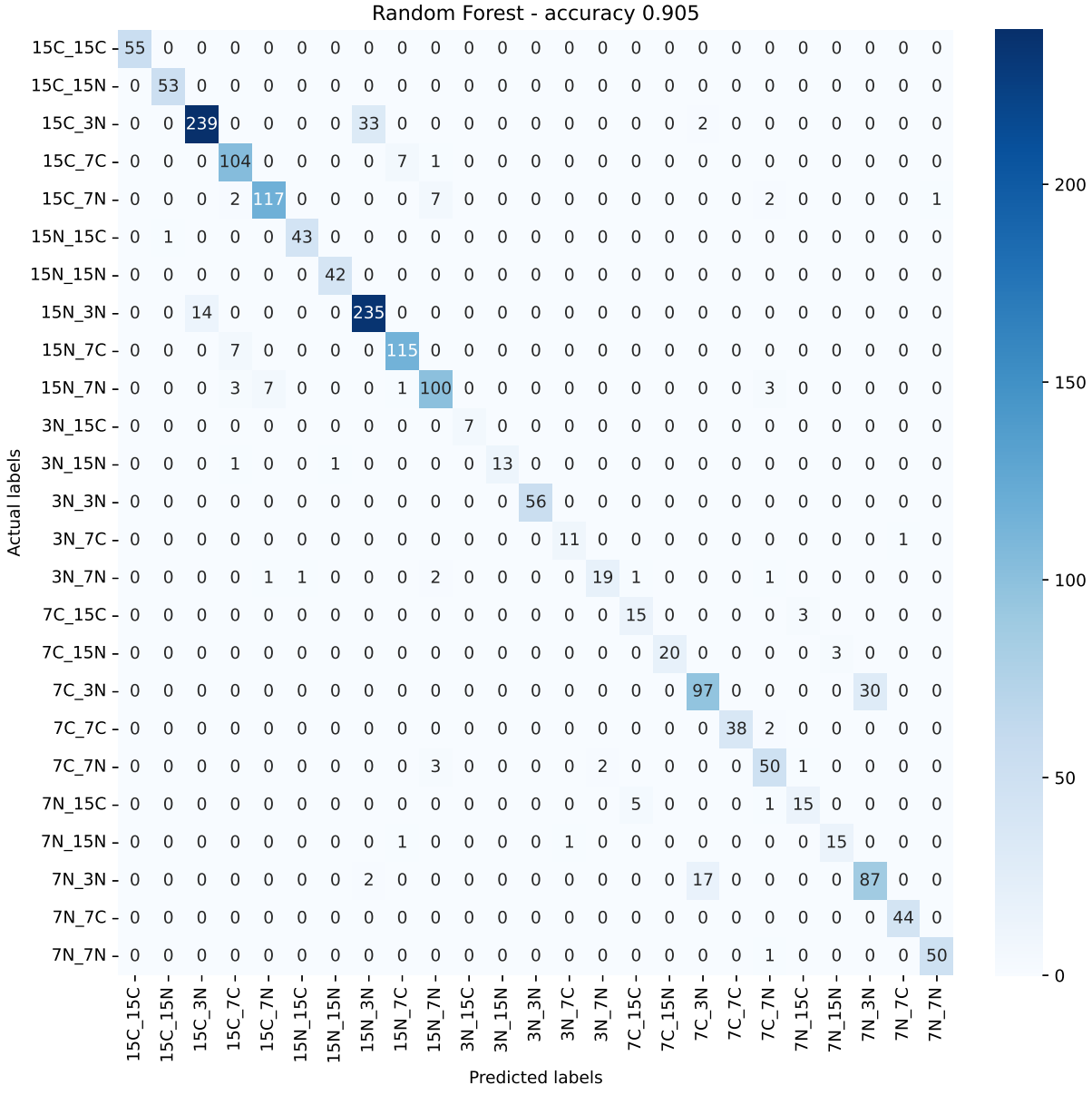

Fig. 3.6: Confusion matrix of the classification of the Pretto dataset for Random Forest algorithm.

The results of the classification of the 25 possible combinations of tape speed and equalization curves are once again very promising. Despite a 4-point difference in the third decimal place in terms of accuracy, both models are capable of correctly classifying most of the samples, even for the less represented classes in the dataset. The confusion matrix of the classification of the Pretto dataset for the Random Forest algorithm is reported in Fig. 3.6. The classification's accuracy has decreased by about 10% compared to the results on the subsets of the dataset, but this is still a very good result, considering that the dataset is much larger and that the classes are not balanced.

The next step was to use the models trained on the entire Pretto dataset to classify the Berio-Nono data. In this case, the process was reversed: first, the model trained on the entire Pretto dataset (using a Random Forest) was evaluated. Then, due to its inefficiency, the possible classes were limited by considering the subsets shown in Table 3.9 (in this case, returning to SVMs), and models were evaluated on individual silence classes (only silences A, B, or C). The confusion matrix of the classification of the Berio-Nono dataset using a Random Forest trained on the Pretto dataset is reported in Fig. 3.7. The results were completely wrong. However, it can be seen that the equalization curves are not recognized in any way, while the model seems to roughly recognize the correct speeds (although much less accurately than the models seen previously).

A better result has been obtained using subsets of the Pretto dataset, but of course, limiting the data also means limiting the possibilities of making mistakes. Classifying 25 categories of data results in 25^2 possibilities between input and output, while simply dealing with the equalizations of a speed (e.g. subset H') is equivalent to choosing between 4^2 possible combinations. It goes without saying that limiting the scope of action can increase accuracy, but it also reduces usefulness.

Table 3.14: accuracy test on Berio-Nono dataset using models trained on Pretto subsets

Train set	Algorithm	Accuracy Score
whole Pretto	Random Forest	0.01
H	SVM	0.12
I	SVM	0.01
J	SVM	0.42
K	SVM	0.58
L	SVM	0.74

Overall, all algorithms have performed really well on the training/validation set, but testing them on a different dataset did not yield any results. Even though the datasets' structures and the analysis objects are the same, a model trained on one dataset cannot be used to classify the other one. This fact underlines the insufficient amount of data for the analysis. As seen in Fig. 3.4, the samples belonging to different datasets are very far apart from each other in space, making it very difficult to accurately classify Berio-Nono samples with models trained on Pretto. The result might be surprising as both cases involve magnetic tapes with different equalization and speed curves, but evidently some parameters were not taken into consideration during data acquisition or analysis.

It is important to note that combining the datasets together and training a model on the whole

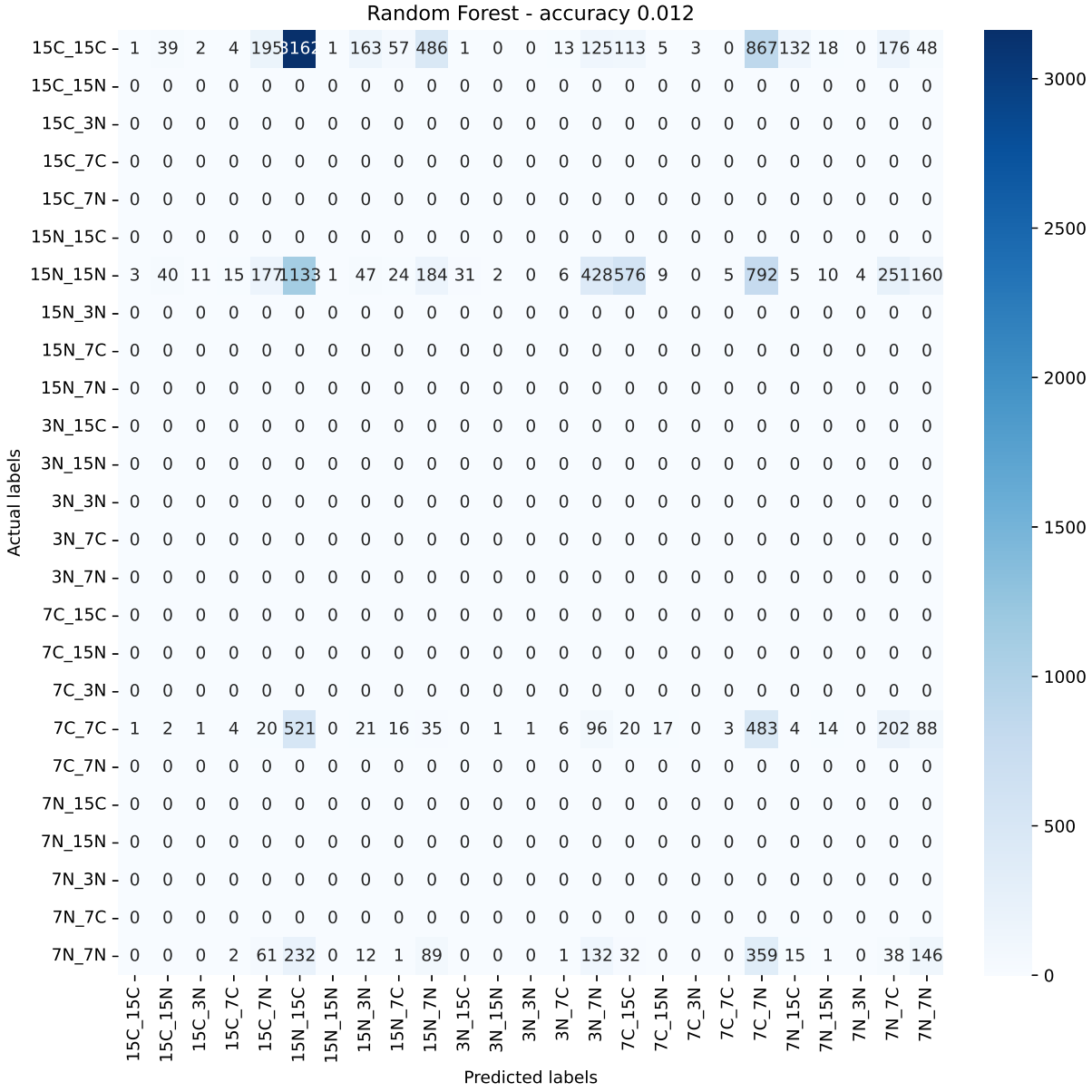

Fig. 3.7: Confusion matrix of the classification of the Berio-Nono dataset using a Random Forest trained on the Pretto dataset.

dataset gives really good results, as those given by the algorithm on a single dataset. In Fig. 3.8, the confusion matrix of the classification of the tapes from both the Pretto and the Berio-Nono datasets is reported. Even though the data is not balanced, the model is able to classify the tapes with good accuracy, revealing once again that a good direction to follow for further analysis could be to enlarge the dataset.

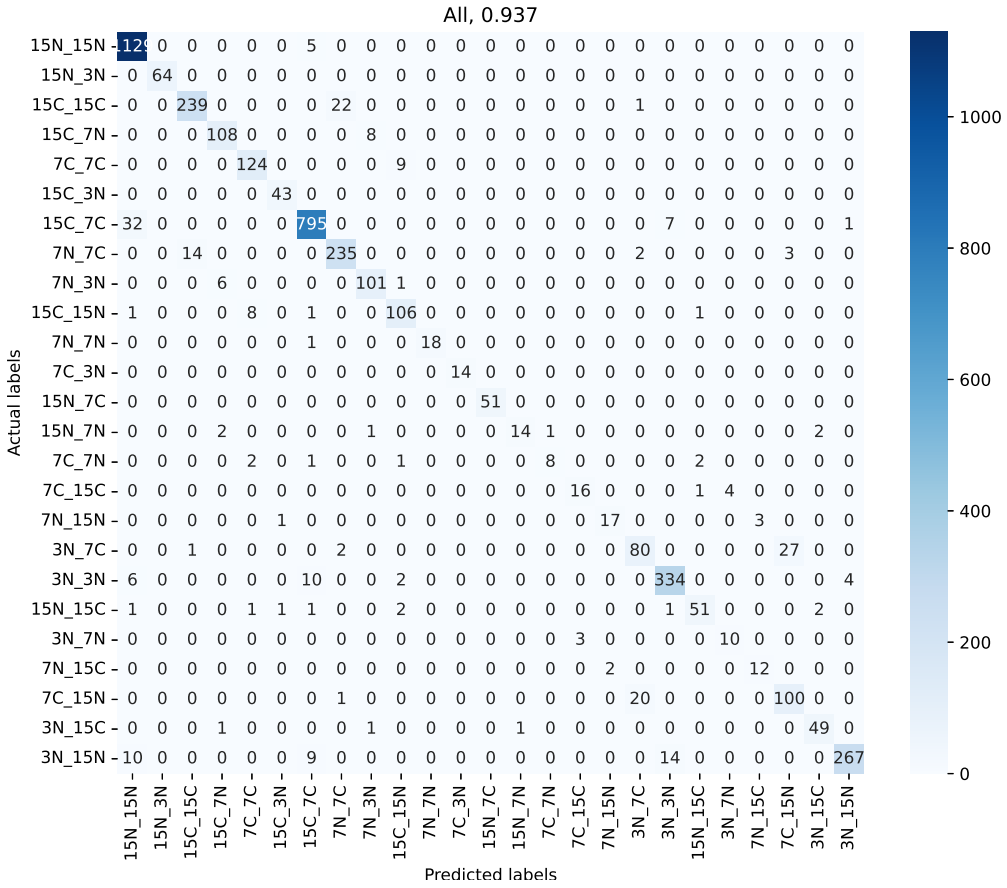

Fig. 3.8: Confusion matrix of the classification of the union of the Pretto and the Berio-Nono datasets.

It would be interesting to conduct an analysis taking into consideration additional tapes and see the performance of this model on them. If the model is able to classify correctly, then it could be thought that the data collected is sufficient. Otherwise, it will be necessary to increase the dataset size until all points in the feature space are considered.

CHAPTER

FOUR

AN AUDIO ANALYSER IMPLEMENTATION

> The purpose of software engineering is to control complexity, not to create it.
>
> —Pamela Zave

Even if the equalization curves classification didn't give excellent results in the experiments illustrated at *Data Analysis* (page 15), the Audio Analyser AIM is still a very important component of the preservation system. The Audio Analyser AIM is the first step of the preservation process and it is responsible for the detection of irregularities in the audio signal, and the modularity across the MPAI-AIF model allows us to implement the Audio Analyser even without a properly working classifier.

Starting from the implementation of the Audio Analyser, based on the technical specifications provided in [4], the module must be able to[1] :

1. calculate the temporal offset between the audio signal and the video signal[2] ;
2. detect Irregularities in the audio signal;
3. assign a unique ID to each irregularity;
4. receive an Irregularity File from the Video Analyser AIM and send the identified irregularities to the Video Analyser;
5. extract an Audio File corresponding to each irregularity (both those found in point 1 and those received in point 3);
6. send the Audio Files and the Irregularity File to the Tape Irregularity Classifier AIM.

Fig. 4.1 provides a general overview of the Audio Analyser.

The operation that this research focuses on is, however, the detection of irregularities in the audio signal (verification of the equalization curve and the playback speed of the tape), the extraction of the corresponding Audio File, and the creation of the Irregularity File for the Video Analyzer. The modules developed for this research don't take in consideration the calculation of the offset between audio and video signals and don't receive the Irregularity File from the Video Analyser.

[1] The following steps have not to be strictly followed in this order, neither they have to respect this separation in the implementation.

[2] Since the operation of starting and stopping the playback of the tape and the video recording is subject to latencies due to the hardware used and is not always engineered in the same way, the time offset between the audio signal and the video signal can be highly variable.

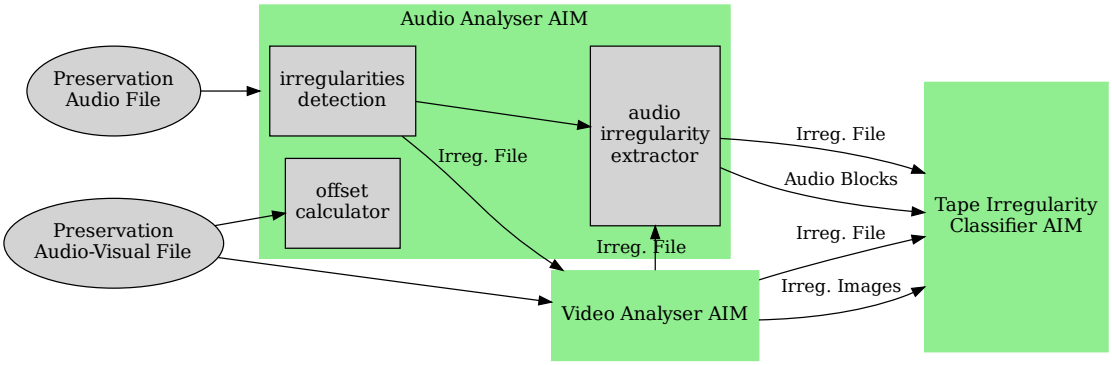

Fig. 4.1: The Audio Analyser AIM pipeline.

In this case, the implementation of the Audio Analyser has been divided into two parts: the first part identifies the silence portions in the signal and extracts them into files, while the second part classifies the pre/post equalization curves and the tape playback speed for each extracted file. The results of the classification are then saved in an IrregularityFile.

4.1 Find the Irregularities

Before diving into the technical analysis of the software, it is necessary to establish its fundamental requirements. The software is expected to take an audio file or a folder of audio files as input and perform the following tasks: 1) extract silence portions from the input file(s), 2) split multi-channel files into single-channel files, 3) save the extracted silence portions in a folder with an appropriate naming convention, and 4) create a log file in .json format containing information about the extracted silence portions.

Given the requirements, it's trivial to define the main steps of the software:

Listing 4.1: the main steps of the software in pseudo-code

```
function extractNoiseSingleThreaded:
    Read the input files
    for each file in input:
        for each channel in the file:
            extract the silence portions from the channel
            save the silence portions in a folder with an opportune name
    save a json file containing information about the extracted silence
↪portions
```

since input files are independent from each other, the software can be easily parallelized. The following pseudo code spawns a thread for each input file:

Listing 4.2: the main steps of the software in pseudo-code (parallelized)

```
function extractNoiseParallelized:
    Read the input files
    for each file in input:
        spawn a thread calling extractNoise on the file
    wait for all threads to finish
    save a json file containing information about the extracted silence
↪portions

function extractNoise:
    for each channel in the file:
        extract the silence portions from the channel
        save the silence portions in a folder with an opportune name
```

The audio portions of interest are the usual A, B, and C class silences (see *Data Analysis* (page 15)), which are identified as audio signal portions with a noise level below a certain threshold value. The duration of these portions is fixed at 500 milliseconds.

To identify signal portions with power below the established threshold, a linear scan of the input file is performed with a window of 500 milliseconds. If the maximum power of the signal contained in the window is lower than the threshold, then the window is considered a portion of audio signal of interest, the average power of the signal is calculated, and the silence class to which it belongs is determined. Otherwise, the window is discarded and the scan continues, moving the seek point to the next sample after the last peak above the set threshold has been identified.

4.2 Classify the Irregularities

The second part of the software is responsible for the classification of the extracted silence portions. The classification is performed by the classifier obtained by training with both Pretto and Berio-Nono datasets to have a better coverage of the features space.

Listing 4.3: the main steps of the irregularity classification in pseudo-code

```
function classifyIrregularities:
    for each AudioBlock:
        extract its first 13 MFCCs
        classify the AudioBlock with the pre-trained classifier
        map the class to the corresponding IrregularityType
```

For efficiency reasons, the MFCCs are saved to a DataFrame and then the classification is performed only once on the entire DataFrame, which is much faster than classifying each AudioBlock separately.

The result of the entire pipeline is an IrregularityFile, which contains the classification of each individual block of silence extracted from the input audio file. The file is structured as follows:

Listing 4.4: the structure of the IrregularityFile


```
{
    "Irregularities": [
        {
            "IrregularityID": "00786a08-9020-4a3a-a4ce-6feaec768f3d",
            "Source": "a",
            "TimeLabel": "00:03:05.462",
            "AudioBlockURI": "./AudioBlocks/C_0_17804362_17852362.wav"
        },
        {
            "IrregularityID": "332fdeea-c545-4bb5-afe1-fbbe08fd8207",
            "Source": "a",
            "TimeLabel": "00:03:05.962",
            "AudioBlockURI": "./AudioBlocks/C_0_17852362_17900362.wav"
        },
        {
            "IrregularityID": "ae159ddb-116d-49bb-a9d9-5a1aa3a13c91",
            "Source": "a",
            "TimeLabel": "00:03:06.462",
            "IrregularityType": "ssv",
            "AudioBlockURI": "./AudioBlocks/C_0_17900362_17948362.wav",
            "IrregularityProperties": {
                "ReadingSpeedStandard": 7.5,
                "ReadingEqualisationStandard": "IEC1",
                "WritingSpeedStandard": 3.75,
                "WritingEqualisationStandard": "IEC2"
            }
        },
        ...
    ]
}
```


as can be clearly seen from the structure of the JSON file, each portion of silence is classified, and various types of irregularities can occur even within a single audio file. The following modules after the audio analyzer are responsible for handling this information and making decisions based on it. For example, if the type of irregularity changes constantly from a certain point onwards, it can be inferred that the tape contains multiple recordings. On the other hand, if irregularities occur sporadically, it can be inferred that the tape contains only one recording and that the rare differences are due to classification errors[3] .

[3] Clearly, the following modules do not only perform this task, but the one exemplified is the most evident that can be appreciated from the output of the audio analyzer.

CHAPTER

FIVE

THE RESEARCH FUTURE

> The only true wisdom is in knowing you know nothing.
>
> —Socrates

As often happens in research, this work is part of a wider investigation. In the early stages, efforts were made to confirm previous research findings, but then, the ability of the starting data to predict similar but unobserved data was examined. The Pretto dataset was used due to its composition, with the hypothesis that it could cover all possible cases to be tested. Unfortunately, this initial hypothesis was incorrect.

The analysis revealed that the Pretto dataset was inadequate to cover all possible cases, prompting exploration of additional avenues to gather more data. One direction that could be taken is to investigate whether data augmentation techniques could be used to expand the existing dataset. Techniques such as audio stretching, pitch shifting, and noise injection are commonly employed to increase the size of datasets in computer vision and speech recognition applications. These techniques generate additional audio samples that can be used to train machine learning models, leading to a more diverse audio dataset that better represents the range of audio signals in the real world.

Another potential factor identified as a possible cause of this insuccess was the analog recorder used to acquire the tapes. The data was acquired at different times from the same machine, and it is possible that it was not in the same calibration condition, leading to variations in the recorded data. These variations could explain why the distribution of the data was dissimilar and why the data was unable to cover all possible cases.

In light of these findings, it has been realized the need to go back to the drawing board and rethink the analysis approach. It may be that the Mel Frequency Cepstrum Coefficients used to describe the audio signals are not sufficient to capture all the nuances of the audio data, other audio features could be used to better describe the audio signals.

To address these issues, further research was recommended into alternative audio features that could be used to describe the audio signals, and other machine learning models could be tested to better predict the audio data. Additional audio data should also be collected to supplement the existing dataset.

In conclusion, while the initial hypothesis of this work did not hold true, several avenues for future research were revealed that could lead to better predictions of audio data. By exploring data augmentation techniques and alternative audio features, and by using a different machine learning model, the accuracy of predictions may be improved. Collecting more audio data and

ensuring that the recorder is well calibrated could help to generate a more comprehensive and representative dataset of real-world audio signals.

BIBLIOGRAPHY

[1] Alessandro Artusi, Andrea Basso, Marina Bosi, Leonardo Chiariglione, Miran Choi, Fabiano Columbano, M.B. Coteli, Nadir Dalla Pozza, Roberto Dini, Michelangelo Guarise, Huseyin Hacihabiboglu, Roberto Iacoviello, Chuanmin Jia, Jisu Kang, Panos Kudumakis, Valeria Lazzaroli, Marco Mazzaglia, Guido Perboli, and Mark Seligman. *Towards Pervasive and Trustworthy Artificial Intelligence: How standards can put a great technology at the service of humankind*. MPAI Community, 12 2021. URL: https://mpai.community/the-mpai-book-2021/.

[2] Sergio Canazza and Giovanni De Poli. Four decades of music research, creation, and education at padua's centro di sonologia computazionale. *Computer Music Journal*, 43:58–80, 2019.

[3] Alessandro Cipriani and Maurizio Giri. *Musica elettronica e sound Design*. ConTempoNet, 2013.

[4] MPAI community. Technical specification — context-based audio enhancement (mpai-cae) v2. 2023.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduzione agli algoritmi e strutture dati*. McGraw-Hill, 2010.

[6] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. doi:10.1109/TASSP.1980.1163420.

[7] Allen B. Downey. *Think DPS, Digital signal processing in Python*. Green Tea Press, 2014.

[8] Theodoros Giannakopoulos. Audio handling basics: process audio files in command-line or python. 2020. URL: https://hackernoon.com/audio-handling-basics-how-to-process-audio-files-using-python-cli-jo283u3y.

[9] Vincenzo Lombardo and Andrea Valle. *Audio e multimedia*. Apogeo, 2008.

[10] Edoardo Micheloni, Niccolò Pretto, and Sergio Canazza. A step toward ai tools for quality control and musicological analysis of digitized analogue recordings: recognition of audio tape equalizations. In *AI*CH@AI*IA*. 2017.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] Giovanni De Poli and Paolo Prandoni. Sonological models for timbre characterization. *Journal of New Music Research*, 26(2):170–197, 1997. URL: https://doi.org/10.1080/09298219708570724, arXiv:https://doi.org/10.1080/09298219708570724, doi:10.1080/09298219708570724.

[13] Niccolò Pretto, Nadir Dalla Pozza, Alberto Padoan, Anthony Chmiel, Kurt Werner, Alessandra Micalizzi, Emery Schubert, Antonio Roda, Simone Milani, and Sergio Canazza. A workflow and digital filters for correcting speed and equalization errors on digitized audio open-reel magnetic tapes. *Journal of the Audio Engineering Society*, 70:495–509, 06 2022. doi:10.17743/jaes.2022.0009.

[14] Niccoló Pretto, Carlo Fantozzi, Edoardo Micheloni, Valentina Burini, and Sergio Canazza. Computing Methodologies Supporting the Preservation of Electroacoustic Music from Analog Magnetic Tape. *Computer Music Journal*, 42(4):59–74, 12 2018. URL: https://doi.org/10.1162/comj\T1\textbackslash{}_a\T1\textbackslash{}_00487, arXiv:https://direct.mit.edu/comj/article-pdf/42/4/59/2005103/comj_a_00487.pdf, doi:10.1162/comj_a_00487.

[15] Pietro Righini. *L'acustica per il musicista: Fondamenti fisici della musica.* Ricordi, 2009.

[16] Antonio Rodà, Nicola Orio, Luca Mion, and Giovanni De Poli. From audio to content. 2012.

[17] Wikipedia. Audio bit depth — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Audio_bit_depth, 2023.

[18] Wikipedia. Trasformata discreta di Fourier — Wikipedia, the free encyclopedia. https://it.wikipedia.org/wiki/Trasformata_discreta_di_Fourier, 2023.

[19] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.