UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

MASTER DEGREE IN COMPUTER ENGINEERING

A new derivative-based model for the automatic detection of time-reversed audio in the MPAI/IEEE-CAE ARP international standard

Supervisor:
Sergio Canazza

Student:
Fabio Zanini

Co-supervisors:
Matteo Spanio
Alessandro Russo

ACCADEMIC YEAR: 2023/2024
Degree date: 23 October 2024

Abstract

This work introduces a novel, envelope derivative-based method to detect reverse audio sections present in an audio document. The proposed method was born to be incorporated in the MPAI/IEEE-CAE ARP standard. The Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Context-based Audio Enhancement (CAE) Audio Recording Preservation (ARP) standard provides the technical specifications for a comprehensive framework for digitizing and preserving analog audio, specifically focusing on documents recorded on open-reel tapes. The primary objective of this project was that of developing a method to automatically identify segments of audio recorded in reverse, in order to use the algorithm during the digitization process of open-reel magnetic tapes. Leveraging advanced derivative-based signal processing algorithms, the system enhances its capability to detect such reversed sections, thereby reducing errors during the analog-to-digital (A/D) conversion. This feature not only aids in identifying and correcting digitization errors but also improves the efficiency of large-scale audio document digitization projects. The system's performance has been evaluated using a diverse dataset encompassing various musical genres and digitized tapes, demonstrating its effectiveness across different types of audio content.

Sommario

In questo lavoro viene presentao un nuovo metodo, basato sulla derivata dell'inviluppo di un segnale, per rilevare le sezioni rovesciate presenti in un documento audio. Il metodo proposto è stato creato per essere integrato nello standard MPAI/IEEE-CAE ARP. Lo standard Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Context-based Audio Enhancement (CAE) Audio Recording Preservation (ARP) fornisce le specifice tecniche per un framework completo per la digitalizzazione e la conservazione di documenti audio analogici, concentrandosi nello specifico su documenti registrati su nastri magnetici a bobina aperta. L'obiettivo principale di questo progetto è stato quello di sviluppare un metodo per identificare automaticamente porzioni di audio registrate al contrario, in modo da utilizzare l'algoritmo durante il processo di digitalizzazione dei nastri magnetici a bobina aperta. Utilizzando avanzati algoritmi di elaborazione di segnali basati su derivate, il sistema incrementa le sue capacità di identificare tali sezioni rovesciate, riducendo di conseguenza errori che possono occorrere durante la conversione da analogico a digitale (A/D). Questa caratteristica non solo aiuta nell'identificare e nel correggere errori di digitalizzazione ma migliora anche l'efficienza dei progetti di digitalizzazione su larga scala di documenti audio. Le prestazioni dell'algoritmo sono state valutate utilizzando un dataset diversificato contenente vari generi musicali e nastri digitalizzati, dimostrando così l'efficacia dell'algoritmo su diversi tipi di contenuti audio.

Contents

1 Introduction

The preservation of analog audio documents is a fundamental process for the survival of the information stored in them. In the past two centuries, all kinds of informations were stored on analog carriers of various types. These carriers were used all over the world and for all kinds of purpose, and contain an enormous amount of cultural heritage. Because of this amount of information stored on those old and fragile carriers, there is the need to transfer the information to more modern types of carriers. For this purpose, digitization is a fundamental step to ensure the survival and accessibility of this important cultural heritage, which would otherwise run the risk of being lost and disappearing, due to the obsolescence of original carriers and analog devices. For this digitization process, it is crucial to develop and design accurate methodologies to minimize the information loss and to avoid introducing irreversible errors in the digitized audio document [1]. The amount of information to digitize and the fast deterioration of which analog carriers suffer, gave birth to the need for fast and effective digitization methodologies and tools. Because of this, in recent years, mass digitization efforts have increased. These works involve the processing of big amounts of materials, which often are unique copies, within reduced time frames. This gives birth to a performance and temporal pressure, which stresses the operators and increases the likelihood of inadvertently introducing errors during routine digitization operations. This errors can be due to distraction of the operator or to misinterpretation of the original audio document. This types of errors can also be increased by the diversified nature of the documents: recordings contain information on their artistic and cultural existence that goes beyond the audio signal itself. It is then crucial to also pay attention to preserve all the content-independent information represented by the container, the signs on the carrier, the accompanying material, and so on.

The digitization process and, with it, the safeguard of audiovisual documents are obstruct by multiple factors. Above all these factors, there is the huge amount of human and economical resources required by the digitization campaigns. About this need, it is also important to specify that the human resources need to be trained, and this training can be difficult, specialized and expensive. Because of this, many archives are lacking methodological and technological tools to safeguard properly their documents. It is now clear that there is a need for software tools for automatize this digitization processes.

The Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Context-based Audio Enhancement (CAE) [2], adopted by the IEEE Standard Association as IEEE 3302-2022 [3], Audio Recording Preservation (ARP) standard satisfies this need, defining precise guidelines and automatic tools for the preservation of analog audio documents. The guidelines defined by this standard originate from the methodology originally developed at the Centro di Sonologia Computazionale (CSC) of the University of Padua (see section 1.1).

The methodologies of the MPAI/IEEE-CAE ARP international standard specifically address the preservation of audio documents recorded on open-reel magnetic tapes. The digitization process of these materials is prone to several common errors that can alter the original content of the carrier, impact the final result of the process and significantly affect the quality of the digital output. Among this errors, one of the most common is the misinterpretation, and so the wrong choice in the digitization process, of the original tape speed, which can also be more than one inside the same tape. Another aspect that can be wrongly chosen is the equalization curves of the conversions from analog to digital and vice versa. The channel configurations (mono, stereo, dual mono, or quadraphonic) can also be a cause of errors in the digitization of the carrier. When dealing with open-reel magnetic tapes, there is indeed the need of using specific gear to read (and digitize) specific channels configurations. Because of this, misinterpreting the configurations of the channel and using the wrong tape heads for the reading, can bring to an unfaithful result which, in the case of extremely fragile carriers, can be fatal. Another point of crucial importance is the detection of audio sections recorded in reverse. The recording of sections in reverse could be due to a vast amount of reasons, from practical to artistic, and this lead to a vast amount of possible scenarios: there can be reversed sections which last a couple of seconds (or even milliseconds), sections that last some minutes and also reversed and non-reversed sections which overlap or alternate.

All these issues can result in tonal alterations and audio playback that is not faithful to the original recording, which is to avoid, given that the aim of the mass digitization process is that of maintaining the original information contained on the carriers.

A deep and trained knowledge of the original recording techniques and formats can help avoid some of these mistakes but in real-world scenarios, time restrictions and the lack of correctly functioning equipment can bring the operators to try and make some compromises, which can make the digitization process even less straightforward. One of these compromises is the reading of all the tracks from a two-sided tape in one pass [4], which can bring to the presence of untracked time-reverse sections in the digital audio document. In this context, modern information technologies provide operators with valuable tools to optimize the digitization process and automatically detect the presence of errors, one of which is presented in this work.

The MPAI/IEEE-CAE ARP Audio Analyzer module (see Fig. 3) combines AI techniques with traditional analytical algorithms to automatically detect common digitization errors, such as misinterpretations of the original equalization and speed variations. In the initial release of the CAE-ARP reference software, the Audio Analyzer module did not include the capability to automatically detect time-reversed audio sections, a feature mandated by the MPAI CAE-ARP standard. This work presents a new algorithm that employs a derivative-based approach for detecting reversed audio sections, integrated within the ARP Audio Analyzer module.

The work presented in this thesis was described in the paper [5], accepted for publication for the 157th AES convention (2024 October 8-10, New York, NY, USA).

1.1 The Centro di Sonologia Computazionale (CSC)

The Centro di Sonologia Computazionale (CSC) of the University of Padua was founded in 1979, after, since the late 1950s, a group of researchers and musicians had been working in Padua, Italy, on computer music.

Since it was founded, CSC has been an important research center in the field of computer music and, over the years, the researchers of the center addressed several topics. These topics include sound processing, expressiveness, multi modal interaction and musical cultural heritage. Along with this, theoretical and scientific achievements have been tested and applied also to music production, and more than two hundred musical pieces have been made at CSC ([6], [7] [8]).

The research activity in the field of electronic music increased the interest in investigating methodologies for the preservation of analog recordings produced at the CSC during the years, in order to save this documents from the deterioration process they are subject to. This research work at the CSC led to the development of a methodology that focuses on the importance of metadata and contextual information. This methodology was supported by a software, the Preservation Software Kit (PSKit) [9]. This software was developed to assist and automatize the creation of the preservation copy of audio documents stored on analog carriers. Using this software, a big amount of information was collected and, for each audio archive, this tool allowed to identify the most common issues and the usage practices.

The first experiences in the field of preservation of audio documents were carried out on the internal archive of electronic music of the centre. This archive is formed by records made by Italian and international composers over the past 40 years. Because of this, the music genre of electronic music was one of main interest during the first years of activity, with the developing of specific skills within this field. The developed methodology has been improved working on various collections of this genre, facing the most common typologies and issues that may affect them. Later on, the digitization process focused on other types of audio archives containing speech recordings of historical and cultural value and music in a more broad sense, with other genres. Each new audio archive helped the development of a more general methodology that could be used in all the cases. The various audio documents of different genres, in fact, differ in many aspects: a tape recorder in a professional studio is very different from a tape recorder with a portable device in an untreated audio environment (in these cases, audio quality has been considered less important than mobility and extended recording time). The electronic music collections are often formed for the most part of sketches and unpublished materials, used then in the final cut of the musical works. The opera and contemporary music collections are made mainly of live recordings of concerts in theaters, mostly on open reel tapes with semiprofessional equipment and tools. The pop-folk music collections contain field recordings of popular Italian songs recorded with portable devices in live settings. The speech collections are composed mainly of audio cassettes of various contents, such as interviews. These collections are often characterized by the lack of written records. The digitization of these collections play a role of great importance both from a cultural and a historical point of view.

2 Analog tapes digitization and restoration

Before the advent of computers and the increase of computer performances, all the informations about music and sound were stored on analog carriers of various types: magnetic tapes, vinyl discs, audio cassettes, and so on. All these types of carriers have one thing in common: they suffer from physical degradation of the carrier and, also, fast obsolescence of the machines used to read the carriers.

This physical degradation process can be slowed down, paying attention when handling the carriers and storing the carriers in temperature and humidity controlled storages, but can't be stopped. Also, the amount of money that needs to be spent for this preservation process is not negligible.

Therefore, the survival of the information contained in the document is possible only renouncing to its materiality, through a constant transfer of the information onto new carriers. Also, the process of transferring the information from one carrier to a new one can give birth to electronic, procedural and operative errors. In addition to these errors, the cultural influence of the time in which the transfer process is made can impact in the way the information is transferred: the operator could remove some particularities of the signal that he thinks are errors, but in reality are not. Because of this, musicological and historical competences are essential for the correct transferring and cataloguing of the information contained in audio documents.

The preservation process, as described accurately by the IASA-TC 05 international guidelines [10], can be divided in passive and active preservation. Passive preservation aims to preserve the document, keeping it on the same carrier. Active preservation, instead, involves data transfer onto new media. Passive preservation is further divided into indirect preservation, if we act on the storage conditions and on the handling procedures, and direct preservation, if the carrier is treated in order to stabilize its physical status.

Until the very end of the 20th century, digital recording techniques were not used, because there was a rapid change in the adopted technologies, and so rapid obsolescence of hardware, digital format and storage media. There was also a lack of consensus regarding sample rate, bit depth, and record format for sound archiving. In addition to these two problems, the stability and durability of storage media was questionable, and also the performances of computers (and so the definition and the quality of digital audio files) were far from similar to today's standards. This led the experts to keep transferring audio documents from analog carriers to other analog carriers.

With the beginning of the 21th century, the audio preservation community introduced the concept "preserve the content, not the carrier", which led to the idea of digitizing the audio content and making it available using digital libraries technology ("distribution is preservation"). With this new paradigm, the process of transferring audio documents from one analog carrier to another analog carrier was no more the standard. The new standard, instead, was that of transferring the informations contained in an analog carrier to a digital file, and then storing the analog carrier, knowing that a safe digital copy was stored in a database. In this way, obsolescence of hardware and deterioration of carriers is no more a problem. However, this new standard led to a series of ethical and philological problems.

In the process of digitizing a document from an analog carrier, multiple choices have to be made (the choice of removing background noise, and so on), which can lead to completely different results.

“To Save History, Not Rewrite It”. The philosophical approach *save history, not rewrite it* is stated in a guide [11] commissioned by UNESCO. This approach aims at analyzing what the original carrier represents, technically and artistically, and to understand what the various aims of the re-recording may be. It is stated that the original equipment should be used only in the case in which it is wanted to reconstruct the sound as it was originally heard. Otherwise, the aim of the re-recording process is to obtain the signal’s best quality, limiting the audio processing to the minimum. Signal alterations can be intentional, like recording, equalization and noise reduction systems, and unintentional, which are divided in alterations caused by imperfections in the original recording technique and in alterations caused by misalignment of the recording equipment (wrong speed, misalignment of the recording in magnetic tape, and so on). Different re-recording strategies derive from the decision of whether or not to compensate for these alterations. We can define two strategies and results: the recording as it was heard in its time (type A); the recording as it has been produced, equalized for intentional recording equalizations, compensated for eventual errors and replayed on modern equipment (type B). The compensations needed by type B require knowledge which is external to the audio signal. Because of this, there is an interpretation factor, given by the fact that historical knowledge is required alongside technical knowledge. Most of this knowledge is retrievable from the history of audio technology, while other informations are experimentally inferable. This re-recording process is then objective and represents the optimal level for the defining of the standard for the preservation copy. A third type of re-recording technique, the type C, aims at obtaining ”the recording as produced, but with additional compensation for recording imperfections caused by the recording technique of the time” [12]. These equalization compensations correct the non-linear frequency response caused by historical equipment and eliminate rumble, needle noise and tape hiss. These operations must be rigorously documented by the restorer, because they are outside the standard operational criteria.

Another crucial aspect is that the metadata contained on the carriers itself or the information on the status of the carrier cannot be lost. This is because using these informations it is possible to make some of the decisions required in the digitization process. So, there was the need for a standard methodology for the preservation and digitization of audio documents [13].

In the recent years, at the Centro di Sonologia Computazionale (CSC), a lot of studies about this problems were carried out, and, together with standard methodologies, some software tools for this task were developed [14]. These methodologies originate from the standards defined by iasa in the IASA-TC 04 guidelines [15]. From these guidelines, the CSC developed specific methodologies due to the fact that the work was often specifically focused on computer music. Over the years, these methodologies defined all the steps of the preservation process, from the documentation of the metadata present on the audio carrier (by means of photos of the carrier and the carrier holder) to the documentation of the physical conditions of the carrier itself (by means of textual descriptions and video recordings of, in the case of analog tapes, the tape being read) to the actual transfer of information to a digital file, have been addressed in a way so that no information is lost. All these informations are stored in a dedicated database and all the digitization process is documented and recorded. Also, all types of alterations to the original detected audio signal are documented, and an unaltered copy of the signal is also stored.

2.1 The preservation process

The preservation copy of an audio document is a dataset that contains all the information contained in the source document. The process that is used to obtain the preservation copy of an audio document is divided in three main blocks: before playback, playback and after playback. The output of each procedure of the blocks is either data or a report. The process is the following [9]:

1. *Preparation of the carrier*
    (a) Physical documentation
        i. Photographic documentation
        ii. Scanned images
        iii. Data validation
    (b) Visual inspection
    (c) Chemical analysis
    (d) Optimization of the carrier
2. *Signal transfer*
    (a) Analysis of the recording format/parameters
    (b) System setup
        i. Replay equipment (e.g. tape recorder)
        ii. Remediation equipment (e.g. converter, acquisition software, etc.)
    (c) Monitoring
    (d) Data validation
    (e) Archival of the source carrier
3. *Data processing and archival*
    (a) Metadata extraction
    (b) Completion of the preservative copy

Preparation of the carrier. In the phase of the preparation of the carrier, photographs of the document and its box are made. In this way, its state of conservation is documented and eventual note and marks present on it are recorded, in order to retrieve important informations which can be useful to playback the tape at the right speed and with the right channel configuration. A visual inspection of the tape is performed in order to discover eventual issues or chemical and physical syndromes of the tape. In case it is necessary, also a chemical analysis is performed. The optimization of the carrier consist in a thermal treatment of the tape, used to correct the Sticky-Shed Soft Binder Syndrome, and the cleaning of the tape, used to remove dust and other unwanted stuff from its surface. Some other optimization processes can be the manual restoration and reparation of old splices and the addition of the leader tape [16].

Signal transfer. The signal transfer phase contains the A/D conversion. Before it, an analysis of the recording formats and parameters is made and the setup of all the replay, remediation and monitoring equipment is made. The monitoring of the whole process is useful to avoid various errors. At the CSC the audio documents are digitized at 24 Bits and 96 kHz, and the tape is filmed during all the digitization process. The video is useful to track marks, notes and other types of marks present on the tape. The video is also useful to detect eventual misalignments of the tape on the tape head, which can cause drops in volumes and other errors. After the digitization, the original document is stored in safe storage conditions.

Data processing and archival. For this step a specific methodology was developed, starting from the IASA-TC 03 guidelines [1] [17], in which all the recommended formats and data management processes are listed. This last phase of the process consist in the data processing and the creation of the preservation copy and of the access copy. The preservation copy contains a high-quality digital audio file, without any restoration. A separate audio file for each channel is stored and multiple acquisitions are made in the cases in which the tape contains audio segment with multiple different speeds. The preservation copy also includes photos and videos, scanned images of the documentation present together with the tape, and checksums. Some metadata, containing brand of the tape, reel diameter, channel configuration, recording speed and so on, are stored in a dedicated database and stored in a .pdf document. This document is included in the preservation copy. The access copy, instead, is saved in a compressed format and can be restored if necessary.

The output of the process is the preservation copy and, after the remediation process, the original document and carrier's conditions should be better than the starting ones. The original documents should always be kept and stored for future use and purposes which are unknown at the moment. If all the process was performed correctly and paying enough attention, the preservation copy will be accurate, reliable and philologically authentic.

This much needed attention to the details gives birth to a process which can be very time consuming and, also, for which there is the need of skilled operators.

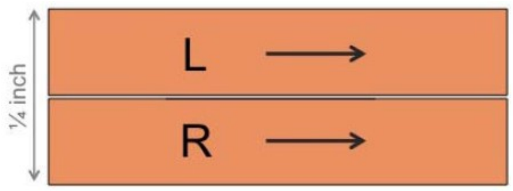
The duration of the process and the cognitive load needed for it can be decreased using algorithms and software tools designed to automatize repetitive tasks. In this way, the process can be a lot faster and, because of this, in the same amount of time, a larger number of documents can be digitized with respect to the standard process. Because the amount of time for the digitization of an audio document decreases, the overall cost for the digitization also decreases. In addition to this, given that the process becomes easier and more automatic, less cognitive load is needed and the operator will be less tired and make less errors.

This need for software tools that automatize the digitization process inspired this work.

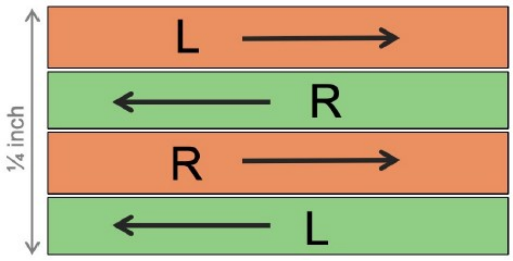
2.2 Reading Tapes Backwards

Based on the composition of the tape and on how a tape recorder works, a magnetic tape can be recorded in multiple track configurations. The track configuration to use when recording the tape is chosen based on different kinds of needs: cost saving, need for quality, and so on. The cost saving need can be satisfied by using stereo or quadraphonic configurations to record multiple tracks on the same tape segment: on the right channel, for example, there can be recorded a track and on the left channel another different track. Also, some other costs can be saved decreasing the tape speed of the recording process. If, instead, the need of the author is to have the maximum quality possible, the way to achieve this is to record a single track in mono configuration, on the widest tape possible and at the fastest speed possible.

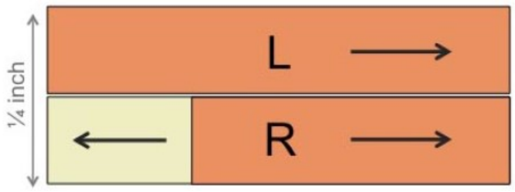
In Fig. 1 we can see some examples of track configurations for magnetic tapes.



(a) One-sided tape: half-track stereo



(b) Two-sided tape: quarter-track stereo



(c) Two-sided tape, non-standard

Figure 1: Examples of track configurations. Source: [18]

For reading or recording these different types of configurations, we need different setups of the tape recorder. These setups differ for tape speed and tape head: we need to select the right speed, according to which speed the tape was recorded with, and the right tape head to mach the tape's track configuration. In fact, if we have a stereo tape, we need a stereo tape head.

The advantage of working with digital signals is that we can manipulate the signal in various ways: we can change the speed digitally, after the reading, we can revert the signal in the time domain, and so on. Reversing a signal head to tail digitally is the equivalent of reading it backwards. From this possibility of reverting a digital track in the time domain derives a possible solution to speed up the process required for the digitization of the magnetic tapes. In fact, we could extract all tracks on a two-sided (or four-sided) tape in one pass, regardless of the intended direction, and then we could subsequently reverse the signal in our digital workstation. Doing this process, we could avoid having to re-read multiple times the same tape to digitize the various tracks in the right direction. In this way, the total digitization time needed for each tape could be reduced by 50%. The time saving advantage is not only to consider profit-wise. In fact, saving time during the digitization process of one project allows the operators to move quicker onto the digitization of another project. In this way, the amount of documents digitized in the same time will be a lot higher, and the history and cultural heritage preserved will be bigger. A part from the time and cost saving aspect, another reason for which this process could be really useful is that it allows to play only once those old and fragile tapes that could only resist a single pass into the tape recorder. In this way, it is possible to avoid damaging too much the tape and possibly loosing some important cultural heritage and information.

This possible solution was examined in the paper [19]. In it, many different metrics were used to measure the differences between the forward signal and the reverted backward signal. These metrics were mainly time domain metrics, because they highlight the differences related to impulse noises or transient responses. The most used metric was the Euclidean distance between two signals: these two signals represent two different readings of the same tape, made with the reel-to-reel tape recorder. Using this metric, it was found that the distance between the signals read with the same direction is significantly minor than the distance between signals read with opposite direction and then digitally time-reversed. These results show that this process introduces measurable alterations in the signal that are greater than the differences inherent in the playback process with reel-to-reel recorders. However, this result does not imply that this strategy should be avoided: the differences are measured at signal level and not necessarily at hearing level. Because of this, this result means that the decision of whether digitizing tapes with this process should be made considering these differences, also taking into account how important it is to save time in the specific case.

A part from the results of the experiment, there are some disadvantages of this strategy that need to be considered:

1. when digitizing multiple tracks with different directions at the same time, the monitoring of the signal transfer can only be performed for some tracks at a time, or needs special setups and equipment;

2. the digital processing needed to find the backward parts of the tracks and to revert them in the correct direction requires specific tools, software and competences and also non-negligible time expense; together with these needs, some errors could be introduced during the digital processing of the files.

Digitizing multiple tapes at the same time that contain signals in only one direction is not the only scenario in which we could find ourselves in. There could also be situations in which a single tape contains multiple signals in different time directions. This could be due to authors trying different things with magnetic tapes to express their artistic views or also for cost savings manners: using the same long magnetic tape to record different things was a very common practice to reduce the costs of recordings. Because of these various needs, it is very common to have to deal with tapes that contain a high number of different recordings, with different speeds and directions.

The large amount of possible configurations, speeds and directions that a tape can contain, creates a myriad of possible outcomes and results that can be recorded on a single magnetic tape. This myriad of different scenarios increases the attention needed during the digitization process and, consequently, the time needed for it. It is then crucial to have some automatic tools to recognize if a portion of tape contains a backward signal or a forward signal, to avoid the need for the operator to listen to all the tape recording, which can last a lot of minutes, or even hours.

In the following sections we will present an algorithm for the automatic detection of reverse signals, which could be used as a solution to this problem and could help automatize the digitization process, to minimize errors and time costs.

3 MPAI

MPAI (Moving Picture, Audio and Data Coding by Artificial Intelligence) is an international unaffiliated non-profit organization developing standards for Artificial Intelligence (AI)-based data coding [16]. It was established in September 2020 from Leonardo Chiariglione, former founder of the MPEG project, together with other former components of MPEG. MPAI is governed by a union of legal entities and members coming from academic departments who contributes to the development of standards for the efficient use of data. MPAI aims at defining and developing standards for audio coding, mainly using AI, to promote the efficient use of data, by standardizing coding technologies, and at facilitating the integration of these data coding components into already complete systems [20]. MPAI integrates state-of-the-art tools, like AI, to address the evolving scenario of data-related applications.

To achieve its objectives, MPAI adopted a process that allows it to reach out to the people in need of a solution and to the people possessing technologies that can satisfy the need of the first group. The process, in fact, was thought to be community-driven, to allow to connect the needs of the various groups of people, to let MPAI members participate in the development of standards, and to allow the members who have specific permissions (principal members) to develop guidelines. The standards of MPAI are implemented using the AI Framework (AIF) infrastructure, described in the MPAI/IEEE-AIF 3301-2022 Standard [21]. The AI Frameworks implementation enables the execution of AI Workflows (AIW), which consist of the aggregation of processing elements known as AI Modules (AIM). The MPAI standard defines the functionality and the input and output data formats of an AIM, to ensure interoperability and also the capability to substitute an AIM or AIW implementation seamlessly with others that offer the same functionality. In this way, systems developed using this process can adapt to different technologies or improvements in AI implementations without requiring re-engineering or disruption of existing functionalities.

MPAI has developed guidelines to test the conformance of AIMs and AIFs and to assess their performances: in the scope of MPAI, performance is a synonym for reliability, robustness and fairness [20]. MPAI can count various standards, working at different levels of complexity: MPAI-AIF, MPAI-CAV (Connected Autonomous Vehicle), MPAI-GME (Governance of the MPAI Ecosystem), MPAI-MMC (multimodal conversation), MPAI-NNW (Neural Network Watermarking), MPAI-PAF (Portable Avatar Format), MPAI-CAE (context-based audio enhancement standard), MPAI-CUI (Compression and Understanding of Industrial Data), MPAI-HMC (Human and Machine Communication), MPAI-MMM (MPAI Metaverse Model), MPAI-OSD (Object and Scene Description), MPAI-PRF (AI Module Profiles) and other projects in development phase [22].

As it is probably known, standards evolve with time and the number of use cases and their names can change. Because of this, in the following paragraphs we describe the version 1 of each standard. However, for some standards, new versions came out and, therefore, some names or some notions about the number of use cases may be slightly outdated.

MPAI-AIF. The MPAI-AIF standard aims at providing a framework that allows easy interconnection and interoperability of AI and data processing technologies when they are part of modules with standard interfaces called AIMs [23]. The data processing technologies can be implemented both in hardware and in software. AIF uses the components-based development (CBD) philosophy, to enable the reuse of independent components, the AIMs, into the systems. The AIMs utilize standard interfaces to communicate with one another. These interfaces specify the services that other components can use and how they should use them. In this way, the specific implementation of each AIM can be unknown. So, basically, the MPAI/IEEE-AIF: is component-based; defines interfaces between components; is secure and the components work in safe zones; supports mixed hardware-software implementations; supports distributed and local execution environments; supports machine learning functionalities; supports the operation of proximity-based distributed AIFs.

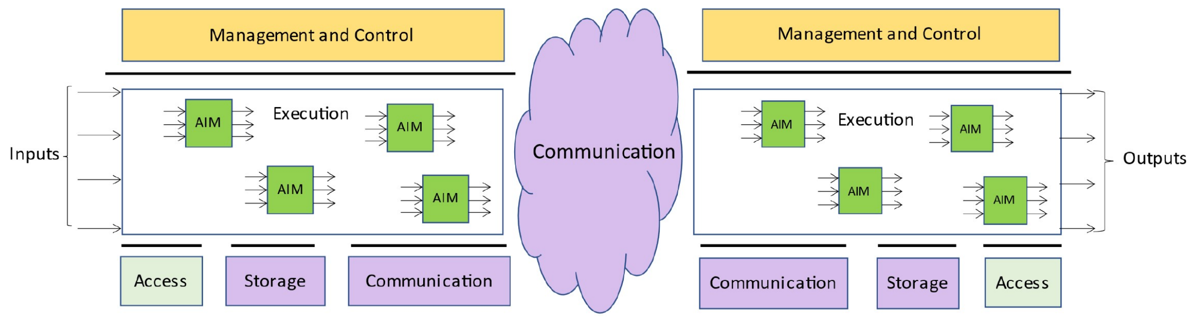


Figure 2: AI Framework schema. Source: [24]

MPAI-CAV. MPAI-CAV specifies the subsystems and components of a Connected Autonomous Vehicle (CAV) that enable the industry to accelerate the development of explainable, efficient and affordable CAVs. This is achieved developing standards designed to enhance interoperability of CAV subsystems and components [25].

MPAI-GME. MPAI-GME specifies the operation of the Ecosystem to enable [26]:

1. developers and implementers to develop components and solutions;
2. performance assessors to assess the performance of an implementation;
3. the MPAI store to test implementations for conformance and post implementations to the MPAI store website together with the results of performance assessment;
4. end users to download implementations and report experience scores.

MPAI-MMC. MPAI-MMC aims at using AI to emulate human to human conversation in completeness and intensity, to improve human-machine conversation [27]. This standard includes five use cases [20]:

- *Conversation with emotion (CWE)*: the human side of the dialogue includes speech, video and text, and the machine responds with synthesized voice, text and animated face video;

- *Multimodal question answering (MQA)*: a human asks for informations about a displayed object in natural language and the machine responds with synthesized voice;
- *Unidirectional speech translation (UST)*: a sentence said by a human is translated by a machine using a synthesized voice that mimics the speech features of the original speaker. In this case the translation is unidirectional;
- *Bidirectional speech translation (BST)*: as before, a sentence said by a human is translated by a machine synthesizing a voice that mimics the speaker's voice. However, in this case the translation is bidirectional;
- *One-to-many speech translation (MST)*: as before, the aim is to translate sentences said by a human using synthesized voice that mimics the speaker's voice. In this case the translation is one-to-many.

MPAI-NNW. The MPAI-NNW standard provides the means to measure, for a given size of the watermarking payload, the ability of the watermark [28]:

1. *inserter* to inject a payload without deteriorating the NN performance;
2. *detector* to recognize the presence and *decoder* to successfully retrieve the payload of the inserted watermark;
3. *inserter* to inject a payload and *detector/decoder* to detect/decode a payload from a watermarked model or from any of its inferences at a measured computational cost.

MPAI-PAF. The MPAI-PAF standard specifies [29]:

1. The Portable Avatar Format and related Data Formats allowing a sender to enable a receiver to decode and render an Avatar as intended by the sender;
2. The Personal Status Display Composite AI Module allowing conversion of a Text and a Personal Status into a Portable Avatar;
3. The AI Workflows and AI Modules composing the Avatar-Based Videoconference Use Case also using Data Types from other MPAI Technical Specifications.

MPAI-CAE. The MPAI-CAE standard utilize the AIF general architecture to enhance user experience across various audio applications, including entertainment, communication, teleconferencing, gaming, post-production and preservation/restoration [30] [24]. CAE applies context-enhanced information to the input audio content to deliver the audio output using the most appropriate protocol. Four distinct MPAI-CAE use cases are defined [20]:

- *Audio Recording Preservation (ARP)*: provides a workflow for managing the process to preserve and access open-reel audio tapes;

- *Emotion-Enhanced Speech (EES)*: implements a user-friendly system control interface that generates speech with various levels of emotion;
- *Speech Restoration System (SRS)*: aims at restoring damaged parts of a speech using neural networks speech models to synthesize it;
- *Enhanced Audio-Conference Experience (EAE)*: improves the auditory experience in an audio conference scenario.

MPAI-CUI. MPAI-CUI includes the AI-based Company Performance Prediction (CPP) use case enabling the prediction of company performance in a given prediction horizon measured in terms of Default Probability, Adequacy Index of Organisational Model, and Business Continuity Index, using its Governance, Financial and Risk data [31].

MPAI-HMC. MPAI-HMC enables new forms of communication between an Entity (a human present or represented in a real or virtual space, or a machine represented in a virtual space as a speaking avatar, acting in a context) and another Entity using text, speech, face, gesture, and the audio-visual scene in which it is embedded, much as humans do. It integrates a wide range of technologies available from existing MPAI standards [32].

MPAI-MMM. MPAI-MMM aims at developing Technical Reports and Technical Specifications supporting Metaverse Interoperability [33].

MPAI-OSD. MPAI-OSD specifies technologies to represent the spatial information of Audio and Visual Objects and Scenes for use across MPAI Technical Specifications [34].

MPAI-PRF. MPAI-PRF specifies a method to signal the profile of an AI Module defined by collections of Attributes (input data, output data and functionality) that uniquely characterize an AI Module instance. In other words, this standard provides a mechanism that unambiguously signals which characteristics of an AIM (Attributes) are supported by the AIM [35].

3.1 MPAI/IEEE-CAE ARP

MPAI/IEEE-CAE ARP covers several different areas of the Sound and Music Community (SMC). In this section we describe the first version of the ARP use case. From the release of the first version, the ARP use case has been further improved and the CSC has been constantly developing new solutions and tools to increase the potential of ARP. For example, a recent work by Lunardon [36] was aimed at developing an AI-based tool for the automatic detection of speed variations in digitized tapes. This and other previous works improved the capabilities of the standard. However, as it often is, there is a discrepancy between what the current version of the standard declares and what are the actual capabilities of the tools. Because of this, the informations presented in this section are related only to the first version of the ARP use case.

The Audio Recording Preservation (ARP) use case represents an important example in the preservation of open-reel analog audio tapes [24]. ARP uses AI and other technologies to extract informations from digitized audio recordings stored, as expected, on open-reel tapes. Its input is the audio of the digitized tape and a video of the tape running through the reel-to-reel recorder, at the level of the magnetic head. The architecture of ARP comprehend five AIMs (see Fig.3): *Audio Analyser, Video Analyser, Tape Irregularity Classifier, Tape Audio Restoration* and *Packager*. The outputs of the ARP workflow are the preservation master, a digital copy for long-term preservation and the related access copy, which is restored, if necessary, and generally stored in a compressed format to allow a correct playback of the digitized audio content [16]. The video recording of the tape being read allows the use of AI algorithms for the automatic detection of various types of irregularities on the tape's surface. This improves speed and precision in the process of selecting and extracting data for the primary storage and usage of it.

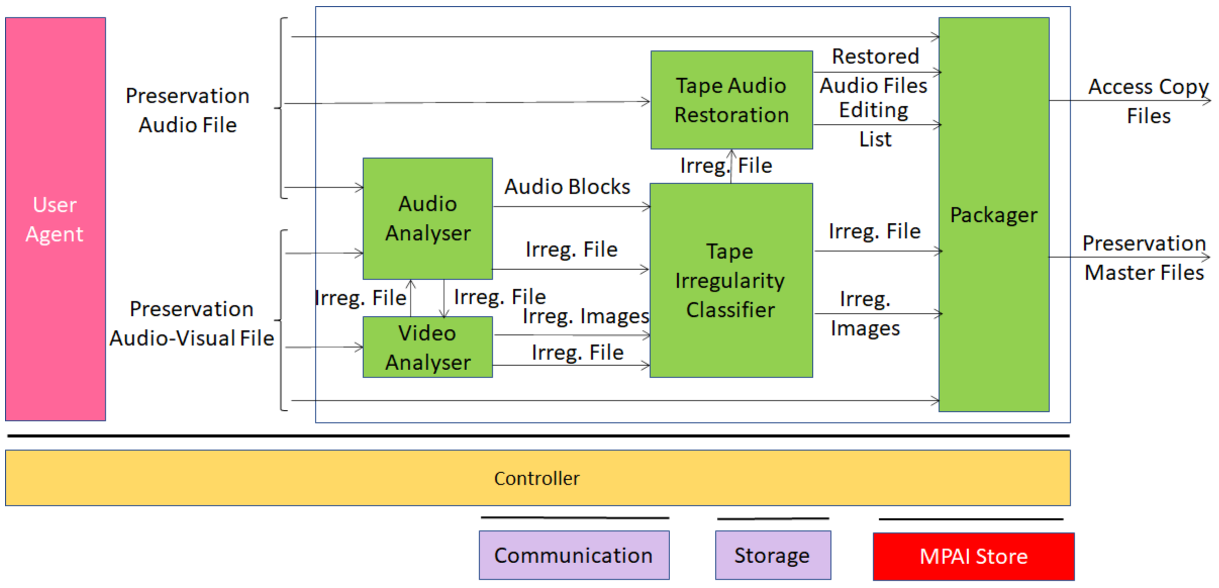


Figure 3: MPAI-CAE ARP AI Workflow

Audio Analyzer and Video Analyzer. As described carefully in [16], the first thing that the Audio Analyzer AIM does is to perform spectral analysis on digitized audio files, to detect irregularities such as misinterpretations of tape speed and recording equalization. Additionally, the MPAI/IEEE-CAE standard specifies that the digitization of open-reel tapes should be accompanied by the video recording of the A/D transfer (Preservation Audio-Visual File). The video is recorded with the camera pointing at the playback head of the open-reel recorder, and the video files also includes a low-quality audio track extracted from one of the outputs of the open-reel recorder, to allow the video and the high-quality audio files (Preservation Audio File) to be synced via cross-correlation. The Video Analyzer uses computer vision algorithms to detect irregularities on the tape, and provides irregularity images, to which the Video Analyzer assigns a unique ID. The Audio Analyzer extracts fragments of the audio recording where the signal is lower than -50 dB and classifies them using machine learning algorithms to detect the correct equalization curve. To detect irregularities in the selection of the equalization curves, a Deep Neural Network (DNN) was selected [37].

The Audio Analyzer relies on analyzing variations in the audio spectrogram images to detect the audio playback speed. Also, it uses Convolutional Neural Network (CNN) models in the speed detection stage. Irregularities are then identified based on the recognition of the equalization curves and the recording speeds done in the previous stages. The Irregularity File, containing all the detected irregularities, together with the offset between the Preservation Audio File and the audio track of the Preservation Audio Visual File, are then sent by the Audio Analyzer to the Video Analyzer. The irregularities detected by the Audio Analyzer and the Video Analyzer modules are synchronized and then Audio Blocks, Irregularities Images, and the corresponding Irregularity Files are sent to the Tape Irregularity Classifier module.

Tape Irregularity Classifier. The Irregularity Files and the corresponding Images and Audio Files sent by the Audio and Video Analyzer modules are received by the Tape Irregularity Classifier. This module classifies the Irregularity Images and, using Neural Networks, selects the relevant ones. It then sends the Irregularity Files and the related Irregularity Images to the Packager.

Tape Audio Restoration. Open-reel audio tapes can be recorded at different speeds, from 30 ips (76.2 cm/s) to 0.9375 ips (2.38 cm/s). Higher speeds are typically adopted by professional studio machines, while low-quality portable machines usually feature lower speed values to increment the recording length for smaller tapes. Another variable parameter is equalization. The equalization curve is in fact used during the recording for extending the dynamic range and improving the recorded signal's Signal to Noise Ratio (SNR) (pre-emphasis curve). The inverse post-emphasis curve is then applied during playback to restore the previously obtained flat frequency response. Given the difficulty of finding correctly operating analog machines with a proper speed configuration and equalization settings, as often required by audio funds, sometimes operators have to perform the digitization process adapting to the functioning of the machines they have at their disposal. Another common situation is when the operator doesn't detect changes in speed occurring during the recording. In these cases, errors that cannot be undone may be introduced and therefore there is the necessity of correcting these errors. The Tape Audio Restoration module has exactly this purpose: to correct errors concerning changes in speed and those related to the application of wrong equalization curves. After this process, the Tape Audio Restoration module sends the Restored Audio Files and an Editing List to the Packager.

Packager. The Packager module collects the Preservation Audio File, the Restored Audio Files, the Editing List, the Irregularity File, the Irregularity Images and the Preservation Audio Visual File and produces the Preservation Master Files and the Access Copies. These files are the output of the ARP AI Workflow. The Preservation Master File includes the Preservation Audio File, the Preservation Audio-Visual file with the low-quality-audio replaced with the Audio of the Preservation Audio File synchronized with the video, the set of Irregularity Images in a .zip file and the Irregularity File listing all the Irregularities detected. The Access Copy includes an Editing List, the Irregularity File and the corresponding set of Irregularity Images, the audio information on the tape and the Restored Audio

Files, saved in a compressed format which is suitable for access to the audio information, but not for long-term preservation.

The current implementation of the ARP standard reference software does not yet include a technology capable of automatically detecting time-reversed audio sections in tape recordings. This feature is increasingly necessary to minimize the risk of introducing errors, especially in massive digitization projects, since, at the current state, operators have to rely on external tools or manual methods to handle reverse audio detection, which, as explained in the previous section, can be extremely time-consuming and add further complexity to the workflow. In the next section we introduce the novel ARP approach utilized to automatically detect time-reversed portions of the digitized audio signal.

4 Envelope Derivative-Based Reverse Detection

4.1 Empirical analysis of first audio signals

The first step in the development of the reverse detection algorithm was the empirical analysis of some audio signals, coming from digitized audio tapes. The analysis was made on 3 different tapes, digitized in both directions with a sample rate of 96 kHz and a bit depth of 24 bit.

The analysis was performed on a python notebook, using the *scipy.io* library [38] to import the signals. The *matplotlib.pyplot* library [39] was also used to create and plot the time-amplitude graphs representing the signals, or portions of them.

The first step of the analysis was to find a set of significant peaks of the signal, and observe the general behaviour and trend of the signal. The peaks were found using the *scipy.signal.find_peaks* function, which was used to find both positive (above 0 amplitude) and negative (under 0 amplitude) peaks. This function was used with a set of parameters empirically founded, which were useful to find only a relevant subset of the peaks of the signal and to avoid taking all the signal's peaks, which would be very performance heavy:

```
signal.find_peaks(signal_data, height = 1e8, distance = 10000)
```

To find the negative peaks of the signal, the phase of the signal was inverted (the signal was mirrored with respect to the time axis) and the positions of the positive peaks on the new signal were calculated and stored.

The signal was then plotted highlighting the peaks and, looking at the graph, it could be noticed that the signal wave was more prominent on the negative amplitude quadrant of the graph for forward read tapes, and the opposite for the backward read tapes. To show this, the highest and lowest peaks were selected (where the lowest peak is the highest peak of the phase-inverted signal) and compared, and it was found that the lowest peak had bigger absolute value than the highest peak, in case of tapes read forward; in case of tapes read backward, the peak with highest absolute value was the positive peak.

This characteristic was at first sight interesting, but was later discarded because it was probably due to a phase issue on the tape machine, and it couldn't be used as a recognizing factor. In fact, it was probably a problem of the one machine in our lab and more calibrated machine shouldn't have this issue.

After discarding this solution, the analysis was carried on taking a portion of the signal around its maximum peak. To avoid taking a random number of samples around the max, the documentation of *librosa.stft* [40] was checked and it was found that, for audio files containing speech (which was the case, for now), a good amount of milliseconds that is sufficiently representative of the signal is 23 ms. Doing a simple proportion $96000 \div 1000 = x \div 23$, it was discovered that, in order to take 23 ms of signal, the amount of samples to take was $96 \times 23 = 2028$ samples.

Because of this, the portion of the signal corresponding to 1014 samples before the absolute max and 1014 samples after the absolute max was taken, which in math notation can be wrote as

$N_{1014}(\max_x f(x))$.

After having calculated the window to analyze, the prominence and the width of the peaks were calculated, but no useful results were found.

Plotting the selected window and looking at the graph, it was seen that the peaks formed a downhill trend: at first, there was a sudden, highest peak; then, a series of slowly decreasing peaks. To calculate and show in some way the downhill trend, the peaks' positions with respect to their width was analyzed, but that didn't gave useful results. Another try was to use the peaks' heights to show the trend. In theory, in fact, if the trend was downhill, the peak to the right of the max should be higher than the peak to the left of the max. In the portion of signal that was initially taken, this process worked and gave confirming results.

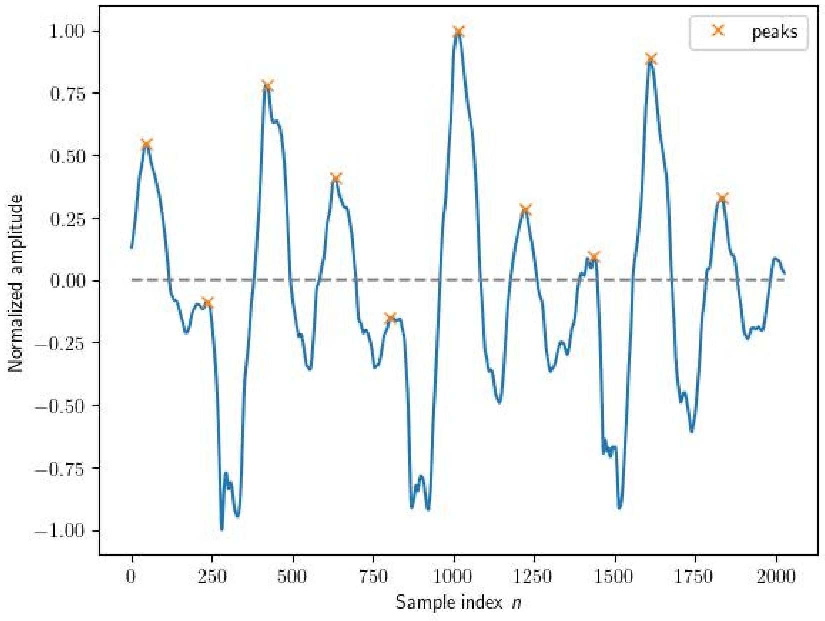


Figure 4: Selected window with highlighted peaks.

To have a broader view of the signal and to try to confirm this method, 10 windows across the signal were taken. To take the windows (of 2028 samples each), it was calculated how many peaks it was needed to skip between a selected peak and the one after that. After this step, the windows were created taking, as before, 1014 samples before the center $(p_0, p_1, ...)$ and 1014 after it: $N_{1014}(p_i) | i \in \mathbb{N} \wedge i < 10$. The window were then plotted and, for every one of them, the uphill/downhill trend was calculated as before.

This process lead to pretty consistent results for the first forward signal, both analytically and visually, looking at the graphs.

Also for the first signal read in reverse, the trend seemed to be downhill. Taking the 10 windows, as with the forward signal, the trend was not consistent and clear: 5 windows had downhill trend and 5 had uphill trend.

This method was applied also on the other four files, and the results kept to be inconsistent.

This results showed that the approach of the downhill/uphill trend was not a consistent approach. Because of this, also this method was discarded.

The final approach was that of analyzing the trend of the peaks of the signal, to highlight its different behaviours, when read in the right direction or time-reversed. The approach is described in the following sections.

4.2 Investigation of the ADSR envelope

The adopted approach for the detection of time-reversed audio segments relies on the analysis of the shape of the time-envelope of the signal.

A well-established model for describing the shape of the time-envelope of audio signals is the ADSR (Attack, Decay, Sustain, Release) envelope (see Fig. 5). This envelope divides the signal in 4 segments/phases:

1. the *attack* phase consists of the time taken by the signal for its initial run-up from silence to its peak level;
2. the *decay* phase consists of the time taken by the signal to decrease from the attack level to the sustain level;
3. the *sustain* is the level maintained by the signal for a period of time;
4. the *release* phase describes the time taken by the signal to decay from the sustain level to silence.

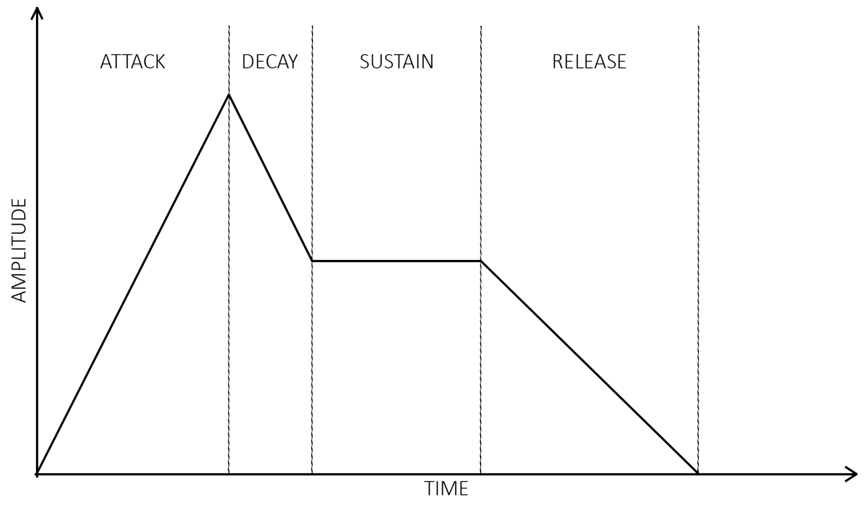


Figure 5: ADSR envelope.

From the empirical observation of audio signals, it is possible to see that the attack phase is typically shorter than the release phase. Inspecting the time-envelope of the audio signal in digitized files, these observations are confirmed in the forward-read audio signals. However, for the backward audio signals, the attacks are typically longer than the releases (see for example Fig. 7). In our model, this particularity is the key-factor to recognize if an audio signal is reversed with respect to time.

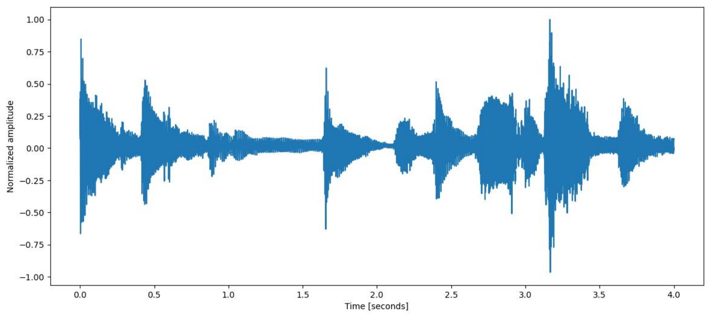


Figure 6: Audio signal amplitude as a function of time.

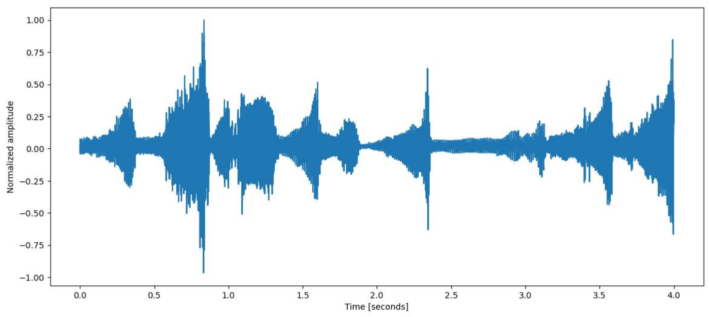


Figure 7: Example of time-reversed audio signal.

By evaluating the length of the attacks and the releases and comparing them, it is then possible to determine whether the audio signal was recorded in reverse. The basic assumption behind this method is that a forward signal is more likely to have fast attacks and slow releases. This can be due to various reasons, among which the most common are reverberation or natural long decays of sounds [41].

Therefore, by comparing the total duration of attack and overall decay phases, the signal playback direction can be determined.

But how can we locate the attack and the release phases?

Let's begin by saying that, in our model, we consider the decay and the release phases as a unique phase. Doing this, we don't consider the sustain as a determining phase for our detection. This is because the way the attack and the release (considered as decay+release) phases are detected is by inspecting the envelope derivative. When the envelope derivative is positive, in fact, the audio signal in analysis is increasing in amplitude. This increase of the amplitude of the signal can be associated to the attack phase of it. Similarly, when the envelope derivative is negative, the audio signal is decreasing in amplitude. The decreasing in amplitude of the signal corresponds to its decay and release phases. Therefore, we can consider attack phases the segments of the signal in which the envelope derivative is positive, and release phases the segments of the signal in which the envelope derivative is negative.

Finally, because of this characteristic, the reverse detection method described in detail in the following sections is based on the computation and analysis of the first derivative of the signal's temporal envelope. This will allow for automatically distinguish between signals recorded forward or backward.

Why not ML In recent years the use of Machine Learning algorithms and Artificial Intelligence has spread across all the scientific disciplines and also across all the aspects of our lives: AI is used in healthcare systems, in educational tools, in smartphones, and so on. In all the fields in which AI algorithms are applied, audio signal processing is the one on which we focused in this work. Audio signal processing algorithms are typically concerned with the analysis of signals, the extraction of the signals properties, the prediction of the signal's behaviour, the correlation between signals, the recognition of patterns in signals, and so on. In terms of signal analysis and classification, in the last decades audio signal processing has significantly grown. In recent years it also have been proven that many existing common issues in this field can be solved using audio signal processing techniques with the integration of modern machine learning (ML) algorithms [42].

MPAI is no different from this. Indeed, as the name suggests, MPAI aims at developing standards for Artificial Intelligence (AI)-based data coding. MPAI uses AI because it is the technology with the greatest potential to transform data: speech can be transformed into text, the number of bits required to represent an image or a video can be reduced, anomalies in data can be detected, and so on. AI is then highly adopted by MPAI because data has value only if its format is known and can be changed to another format that has more value. MPAI, in fact, addresses the issue of data conversion from a format to another by using the same principles and technologies to achieve a lot of different operations and data conversions [16].

The method described in this work does in fact use AI: all modern coding languages and tools are heavily based on the use of artificial intelligence. However, our method doesn't rely on the use of ML algorithms. In other words, our algorithm is an analytic algorithm which uses libraries and tools that have been developed relying on AI to achieve useful results and better performances, but does not rely on newly developed neural networks or classification/regression machine learning algorithms. The analytic solution that will be described in detail later in the work was in fact the most straightforward solution that came to our minds. Also, developing this solution, we found that there weren't parts in which we needed to use ML algorithms or to develop a NN (Neural Network) to have the work done. All the solution came together using analytic tools and methods which rely on mathematics and ”classic” algorithms. The solution also proved to be correct and efficient, so we decided that it didn't require machine learning tools. However, it is not to be excluded that a solution that uses ML can be developed. This solution could also be better than the proposed method and, maybe, have better computational costs and performances. This solution could also be simpler than the solution provided in this work.

The proposed method will still be suited to be added to the MPAI standard. This is because our method, although it doesn't use newly developed machine learning algorithms, is still based on the use of languages and tools which are deeply relying on AI. Also, another important point to mention is that the main aim of MPAI is not that of using AI or ML, but it's that of developing new and useful standards, without focusing on the way these standards are developed (if they use or don't use AI or ML).

4.3 Envelope Computation

As previously anticipated, the reverse detection algorithm presented in this work is able to recognize if an audio file is forward or backward, which means that it is reverse with respect to time.

The algorithm takes as input the audio file to be analyzed. This audio file could be any possible audio file: a digitized analog tape, a piece of music recorded with a smartphone, a piece of music created digitally, and so on.

The algorithm also takes in input some other parameters, some of which are optional, that will be listed and explained later in this work.

In Fig. 8 a block diagram of the functioning of the algorithm is presented. This diagram highlights the main steps of the process, which will now be explained.

The first thing that the algorithm does is to detect if the path passed as input is a folder of files or a single file. In the case the path represents a single file, the file is read and the computation begins.

```
if path.is_file() and (path.suffix == ".wav"):
        print("\nInput is a single file.\n")
        # use reverse_detection_noise_removed_fft_precise to check if file is
        # forward or backward
        info = reverse_detection_noise_removed_fft_precise(
            path,
            noise_thresh,
            window_width_mult,
            min_silence_length,
            filter_freq,
            with_filter,
            pre_post_cut,
        )
        print("\n")
        for i in info:
            print(f"{path.name} : {i}")
```

Otherwise, if the path passed as input represents a folder containing a series of audio files, the computation is performed for every one of them. In the case the input is a folder, a function *display_and_count* is defined, which performs the computation on every one of the files contained in the folder.

```
    else:
        print("\nInput is a folder.")
        files = list(path.glob("*.wav"))

        def display_and_count(f: Path):
            print(f"\nFile name: {f.name}")
            return reverse_detection_noise_removed_fft_precise(
                f,
                noise_thresh,
                window_width_mult,
                min_silence_length,
                filter_freq,
                with_filter,
                pre_post_cut,
            )
```

After this step, the function *reverse_detection_noise_removed_fft_precise* has been called, for one or more audio files. This function is basically the implementation of the reverse detection algorithm proposed in this work.

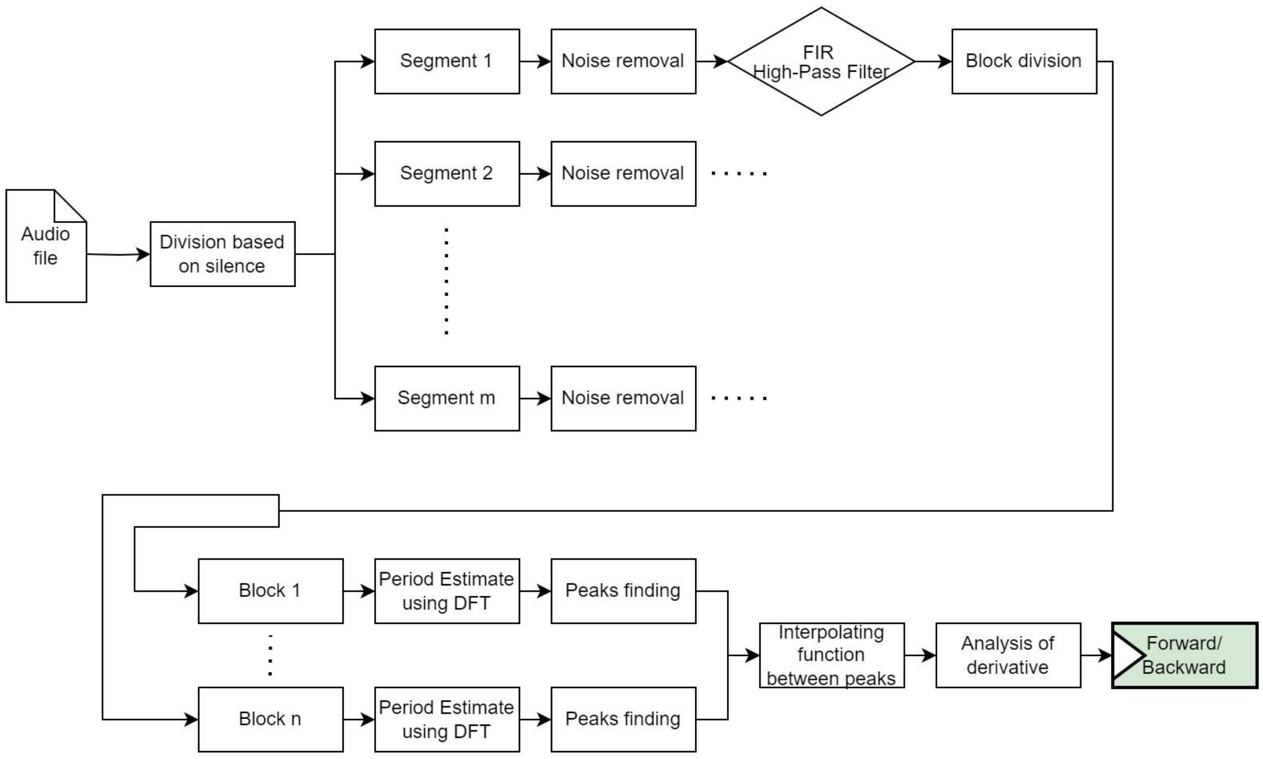


Figure 8: Block diagram of the reverse detection algorithm.

The first thing that the function, and so the algorithm, does is to read the file using the *scipy.io.wavfile.read()* function and to read and store the metadata of the file. The metadata of the file consists of:

- the bit rate;
- the number of channels;
- the sample rate;
- the duration of the audio file in seconds.

This metadata is stored in a Dataclass called AudioMetadata.

```python
@dataclass(frozen=True)
class AudioMetadata:
    bit_rate: BitRate
    channels: int
    sample_rate: int
    duration: float

    @classmethod
    def from_file(cls, filename: PathLike | str) -> "AudioMetadata":
        """Create an instance of AudioMetadata from a file."""
        with wave.open(str(filename), "rb") as wave_file:
            bitrate: BitRate = wave_file.getsampwidth() * 8  # type: ignore
            channels = wave_file.getnchannels()
            samplerate = wave_file.getframerate()
            length = wave_file.getnframes() / samplerate

        return cls(bitrate, channels, samplerate, length)

    def __repr__(self) -> str:  # noqa: D105
        return f"""AudioMetadata(
    bit_rate:    {self.bit_rate},
    channels:    {self.channels},
    sample_rate: {self.sample_rate},
    duration:    {self.duration}s
)"""
```

After storing the metadata of the audio file, the algorithm checks the number of channels to understand if the file contains a mono or stereo or quadraphonic audio signal. If the file contains more than one channel, the function analyzes each channel separately.

As previously anticipated, the presented method utilizes the derivative of the signal temporal envelope to divide the attack and release phases of the signal. To study the derivative of the signal envelope, however, it is essential to accurately describe the envelope as a function of the audio signal waveform. Audio signals are typically quasi-periodic, which means that even thought they aren't strictly periodic, they present a pattern of recurrence, but with a component of unpredictability. The quasi-periodicity of the audio signals allows to compute the envelope function by interpolating the maxima of the signal amplitude values for each period.

While examining signal variations over time, sections of silence can pose challenges in the computation of its envelope. However, extended segments of silence could also simply indicate a break between multiple recordings. Because of this, the algorithm needs to comprehend a silence removal step, to remove all the silence parts between different recordings and to divide the original file based on them. In this way, if the file contained multiple recordings divided by meaningful portions of silence, these are analyzed separately.

The algorithm checks for audio portions of the signal below a predefined threshold and subdivides the original audio signal into smaller segments, where the audio signal segmentation is based on the duration of the silence.

For computing this division, it is necessary to compute a set of relevant peaks to utilize to find the portions of silence. The peaks are computed using some parameters found empirically. After finding these peaks, only the peaks above the predefined silence threshold are kept and stored. The silence threshold τ, measured in dB, is determined by input parameters, but default parameters are also provided for convenience. In this case, the default silence threshold is $\tau = -50$ dB, as proposed in [13], which is a standard threshold for noise in the middle of a recording (i.e., silence between spoken words).

```python
def find_peaks_over_threshold(
    audio_segment: NDArray,
    bitrate: int,
    threshold: int = -50,
) -> NDArray[np.int64]:
    # Take peaks of signal
    peaks, _ = signal.find_peaks(audio_segment, height=257, distance=100)

    # Find only peaks of signal higher than the threshold
    peaks = peaks[amplitude2dbfs(audio_segment[peaks], bitrate) > threshold]
    return peaks
```

Since the silence threshold is expressed in dB, it is necessary to convert the amplitude values of the peaks to a dBFS value. For this task, the function *amplitude2dbfs* is used, which basically is the implementation of the following formula:

$$db = 20 \times log_{10}\left(\frac{\text{PCM}}{2^{bit\_depth-1}}\right)$$

```python
def amplitude2dbfs(
    signal: Iterable[int],
    bit_depth: int,
) -> NDArray[np.float64]:
    min_val = {
        16: -98.09,
        24: -146.26,
        32: -194.42,
    }

    if bit_depth == 24:
        # scipy uses 32 bit depth for 24 bit depth audio
        bit_depth = 32

    if signal == 0:
        return np.float64(min_val[bit_depth])  # type: ignore

    return 20 * np.log10(abs(signal / 2 ** (bit_depth - 1)))
```

After finding all the peaks above the silence threshold τ, the function *remove_silence* is called, which basically removes the silence segments from the audio signal given in input. The function checks for audio portions of the signal below the predefined threshold (using the peaks over the threshold, computed using the *find_peaks_over_threshold* function) and subdivides the original audio signal into smaller segments, where the audio signal segmentation is based on the duration of the silence. One parameter, a part from the signal and the peaks, is necessary for the functioning of the algorithm: the minimum silence duration (*min_silence_len*). The minimum silence duration represents the minimum quantity, in milliseconds, of silence that there has to be between two audio segments, to consider the two segments as separate signals. Also in this case, there is a default minimum silence duration, which is 2 seconds.

The *remove_silence* function takes the signal passed as input and returns an array containing [*start_sample*, *end_sample*] tuples of all portions of signals distanced from one another by, at least, *min_silence_len* milliseconds of silence.

```python
def remove_silence(
    audio_segment: NDArray,
    peaks: NDArray[np.int_],
    samplerate: int,
    min_silence_len: int = 2000,
) -> list[tuple[int, int]]:
    sig_len = len(audio_segment)

    signals_indexes = []

    # if len of signal is less then min_silence_len, we don't serach for silence and
    # simply return a tuple [0, sig_len]
    if sig_len < min_silence_len:
        signals_indexes.append((0, sig_len))
        return signals_indexes

    # now I have only peaks higher than the threshold
    # I need to calculate the space between peaks, to check if it is longer than
    # min_silence_len

    # First, convert min_silence_len from milliseconds to samplerate
    min_silence_len_ms = (samplerate * min_silence_len) / 1000

    start_sig = peaks[0]
    end_sig = peaks[0]
    for index in range(len(peaks) - 1):
        # calc distance (in samples) between two peaks
        dist = peaks[index + 1] - peaks[index]

        # if distance is higher than min_silence_len_ms we have
        # a portion of silence we want to exclude
        if dist >= min_silence_len_ms:
            # the end of the portion of signal beofre silence is peaks[index]
            end_sig = peaks[index]

            # portion of signal befoe detected silence goes from start_sig to
    end_sig
            signals_indexes.append((start_sig, end_sig))

            # new start of next portion of signal is peaks[index + 1]
            start_sig = peaks[index + 1]
        elif index == len(peaks) - 2:
            # if i reached the last index,
            # I need to close the current signal segment
            # that I'm inspecting
            end_sig = peaks[index + 1]

            signals_indexes.append((start_sig, end_sig))

    # Now I have an array full of [start sample, end sample] of actual segments,
    # and I return it
    return signals_indexes
```

Basically, the function counts the space between the peaks over the noise threshold, found with the *find_peaks_over_threshold* function, and checks if this space is greater than the minimum silence duration parameter. In this case, the function splits the two signals storing the start and the end of each signal in the [*start_sample*, *end_sample*] tuples. The array returned by the function is then used to get the start and the end of each different audio segment (which correspond to a different recording), to analyze them separately.

After segmenting the signal, an optional noise removal stage eliminates portions of the signal with levels below -40 dB. The portions are removed from the beginning and from the ending of the signal. Basically, using the peaks founded over the silence threshold τ, the function selects the first peak higher than -40 dB and removes all the signal segment before it. The same is done for the end of the signal: the function takes the last peak higher than -40 dB and removes all the signal after that. This pre-and-post cut option was founded to be useful in some particular cases, like with the audio files used for the first empirical analyses.

The optional pre-and-post cut function is activated by an input boolean parameter of the function: the *pre_post_cut_option*.

```python
if pre_post_cut_option:
    peaks = peaks[
        amplitude2dbfs(sig_data[peaks], meta.bit_rate) > NOISE_THRESHOLD
    ]

    if len(peaks) != 0:
        last_peak_index = len(peaks) - 1
        sig_data = sig_data[peaks[0] : peaks[last_peak_index]]
sig_data[amplitude2dbfs(sig_data, meta.bit_rate) < noise_threshold] = 0
```

After this optional function, the function presents another optional stage. This stage, activated by the parameter *filter_option*, encompasses a finite impulse response (FIR) high-pass filter. This filter has linear phase and cutoff frequency set by default at 800 Hz. The cutoff frequency can also be specified by an input parameter: the *filter_frequency*.

This filter, when applied, removes all the frequency range from 0 Hz to the cutoff frequency specified, or the default frequency of 800 Hz. By default, this option is off, but in the cases in which the frequency spectrum of the signal tends to be overcrowded, it could be useful to better identify the transients of the signal. In this case, in fact, when the frequency spectrum is full, the transients of the signal tend to be less clear and less recognizable. This lead to a difficulty in the analysis of those transients and, consequently, in worse performances of the algorithm.

```python
if filter_option:
    print("\nFilter cuts from 0 Hz to", filter_frequency, "Hz")
    f = filter_frequency / 1000
    with_filter = signal.firwin(5, f, pass_zero=False)
    sig_data = signal.convolve(sig_data, with_filter)
```

The segment in analysis is then partitioned into windows. The length of these windows is determined by an input parameter: the *samplerate_mult*. This parameter represents the multiplier used to calculate the width of the windows as:

$$length\_window = samplerate \times samplerate\_mult.$$

Also in this case there is a default value for the multiplier, $samplerate\_mult = 0.25$, which is the equivalent of calculating the windows length as $length\_window = \frac{samplerate}{4}$.

```python
# calculate the width of the windows on which we calculate the fft
length_window = math.floor(meta.sample_rate * samplerate_mult)
# I take more windows, in order to cover all the signal,
# and for every window, I calculate DFT and take the foundamental peaks
iterations_to_do = math.ceil(len(sig_data) / length_window)
```

After windowing the segment, each window is analyzed separately. For each window, the Discrete Fourier Transform (DFT) is calculated. The DFT is used to calculate the fundamental frequency of the window. This frequency is obtained by taking the first half of the DFT graph, that is the frequency spectrum of the window, and calculating the component with the maximum magnitude: the frequency is the position (the index in the spectrum) of the calculated maximum. Then, using the fundamental frequency, we can obtain the period of the window by

$$period\_window = \frac{1}{fundamental\_freq}.$$

```python
# For the window I calculate the DFT
window_data_fft = fft.fft(window_data)

# Take half of the fft
window_data_fft = window_data_fft[
    : math.floor(len(window_data_fft) / 2)
]

# With the DFT I find the fundamental frequency (and so, period = 1/f)
if len(window_data_fft) == 0:
    fundamental_freq = 1
else:
    max_freq_val = max(window_data_fft)
    for index in range(len(window_data_fft)):
        if window_data_fft[index] == max_freq_val:
            fundamental_freq = index
if fundamental_freq == 0:
    new_window_data_fft = window_data_fft[1:]
    if len(new_window_data_fft) == 0:
        fundamental_freq = 1
    else:
        new_max_freq_val = max(new_window_data_fft)
        for index in range(len(new_window_data_fft)):
            if new_window_data_fft[index] == new_max_freq_val:
                fundamental_freq = index
if fundamental_freq != 0:
    period_window = 1 / fundamental_freq
```

The calculated period is stored in seconds. The next step is to convert it from seconds to number of samples. This is done using the proportion:

$$1 : samplerate = period\_window : length\_period\_samples.$$

```python
# I have the period in seconds, so I need to convert it in number of samples
length_period_samples = math.floor(meta.sample_rate * period_window)
```

After finding the period and calculating the width of the period in samples, this period width is used as the distance parameter of the *scipy.signal.find peaks* function. In this way, a positive peak per period is selected.

```python
# In order to use the period to find the fundamental peaks of each
# window, I put the period in the distance parameter of find_peaks
if ind == 0:
    fundam_peaks, _ = signal.find_peaks(
        window_data,
        height=0,
        distance=length_period_samples,
    )
else:
    fundam_peaks_window, _ = signal.find_peaks(
        window_data,
        height=0,
        distance=length_period_samples,
    )

    # Shift the start of the founded peaks
    fundam_peaks_window = fundam_peaks_window + start

    # Add founded peaks to array containing all the peaks
    fundam_peaks = np.concatenate(
        (fundam_peaks, fundam_peaks_window),
    )
start = start + length_window
```

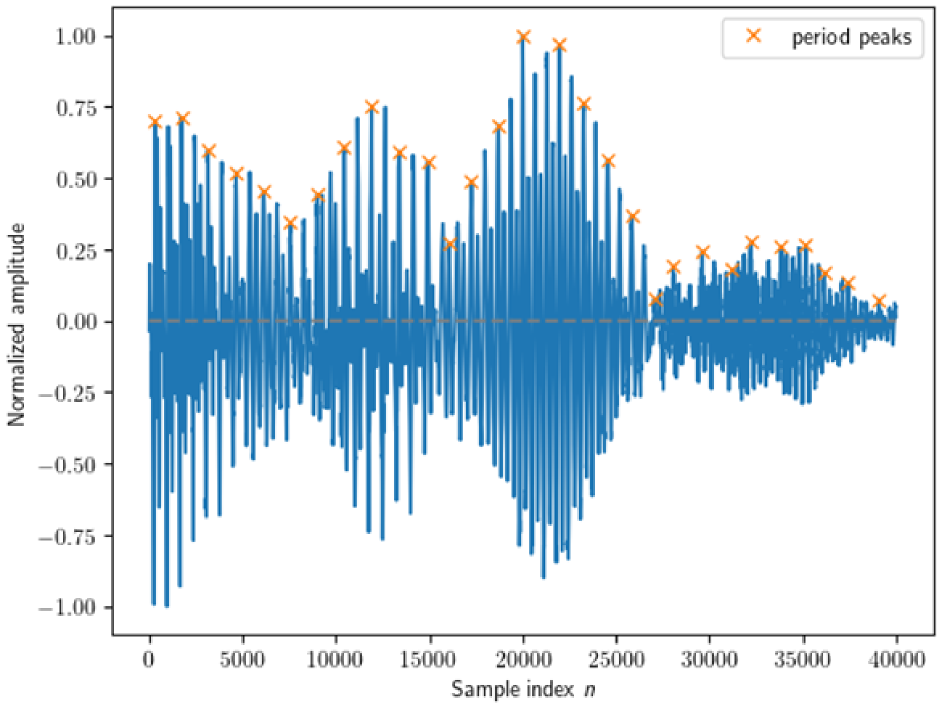


Figure 9: Audio signal segment showing peaks.

All the founded peaks are then stored in the *fundam_peaks* array. These peaks, all together, form a clear representation of the time-envelope of the signal segment.

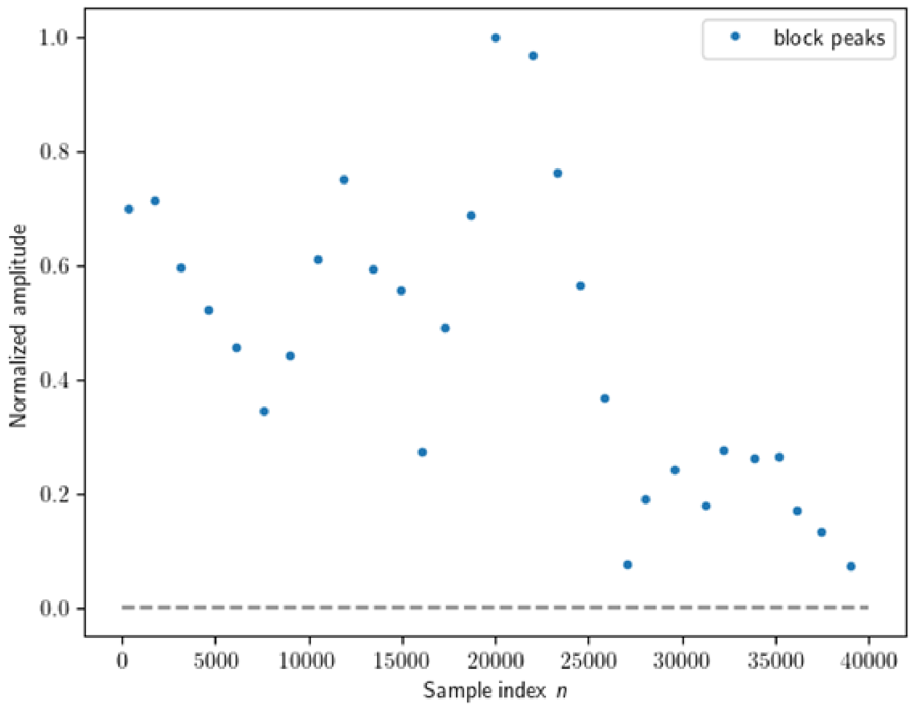


Figure 10: Segment peaks.

The proper representation of the signal envelope $E(t)$ is then derived by linearly interpolating the peaks throughout the signal (see Fig. 11).

The interpolating function is calculated with the *calc_interpolate* function. This function takes as input the signal and the array of the peaks indexes, and calculates the interpolating function that connects the peaks using the *scipy.interpolate.interp1d* function.

```python
interp_f, xnew_int = calc_interpolate(sig_data, fundam_peaks)
```

```python
def calc_interpolate(
    data: NDArray,
    peaks: NDArray,
) -> tuple[interpolate.interp1d, NDArray]:
    f = interpolate.interp1d(peaks, data[peaks])
    xnew = np.arange(peaks[0], peaks[len(peaks) - 1], 1)
    return f, xnew
```

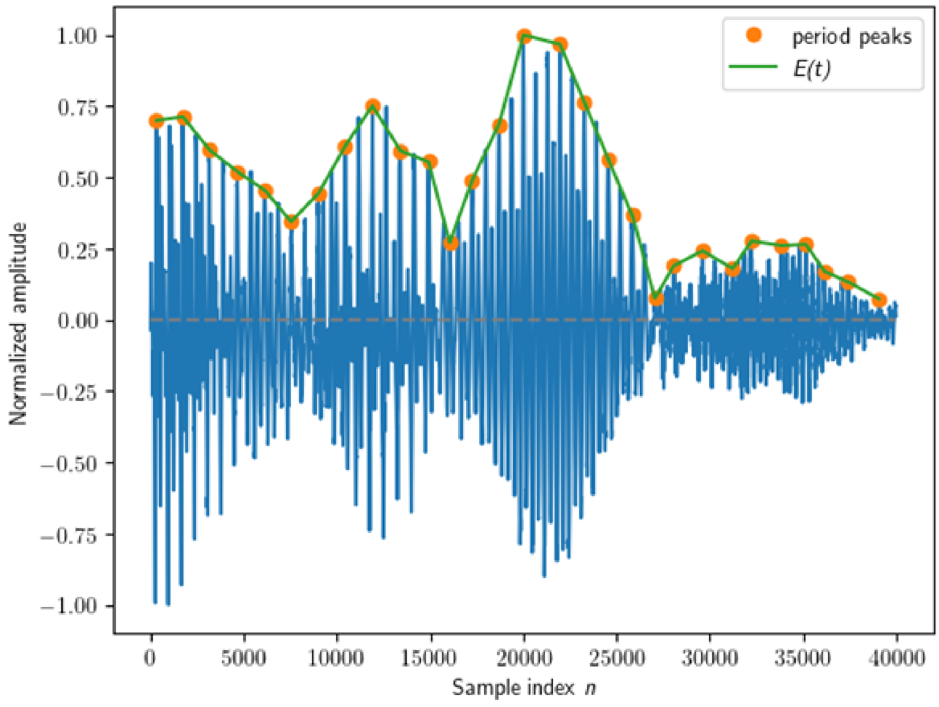


Figure 11: Signal envelope $E(t)$.

4.4 Derivative Computation and Analysis

A numerical approximation of the derivative $E'(t)$ at time t_i of the computed envelope $E(t)$ is then calculated as follows:

$$E'(t_i) \approx \frac{E(t_{i+1}) - E(t_{i-1})}{t_{i+1} - t_{i-1}} \tag{1}$$

It is easy to recognize that when $E'(t) > 0$, the time-envelope of the signal is increasing in amplitude, and so we are in the attack region. Conversely, when $E'(t) < 0$, the time-envelope of the signal is decreasing in amplitude, and so we are in a decay or release region. As previously explained, we designate the sum of the decay and release times as the overall decay phase. In general, we expect that a time reversed signal will have longer attacks and shorter overall decays.

Let's consider the time vector:

$$\mathbf{t}_{E'^{+}} = t_0, \ldots, t_k. \tag{2}$$

The elements of this vector are times where $E'(t) > 0$.

Conversely, the time vector:

$$\mathbf{t}_{E'^{-}} = t_0, \ldots, t_j \tag{3}$$

has elements where $E'(t) < 0$.

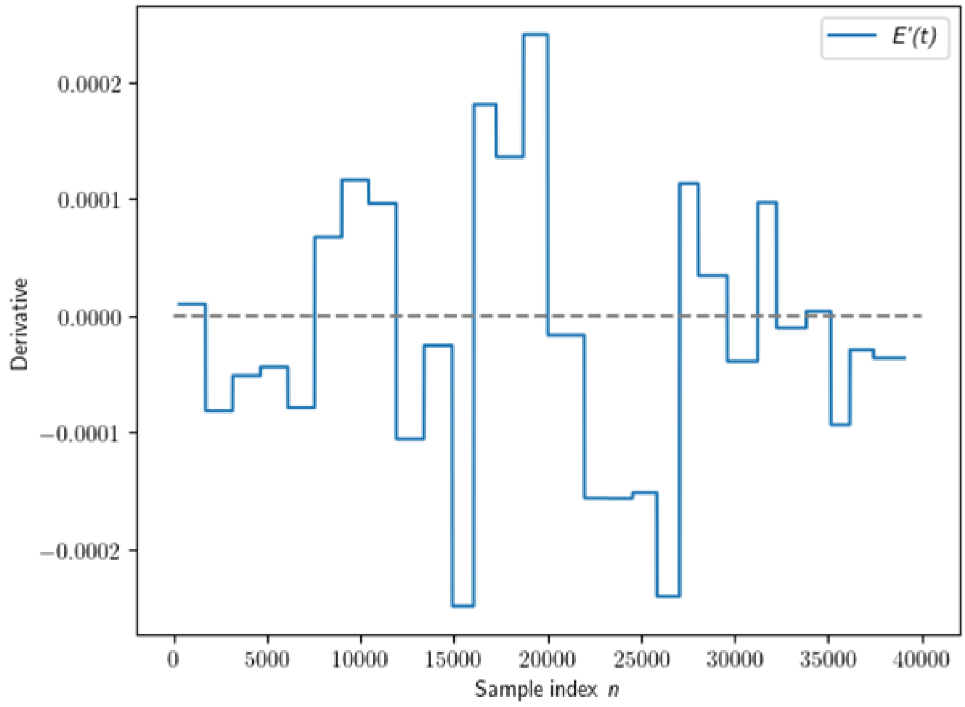


Figure 12: Graph of the envelope derivative $E'(t)$.

The cardinality of the time vectors $\mathbf{t}_{E'+}$ and $\mathbf{t}_{E'-}$, calculated as

$$T_{E'+} = |\mathbf{t}_{E'+}| \tag{4}$$

$$T_{E'-} = |\mathbf{t}_{E'-}|, \tag{5}$$

represent an indication of the length of the overall attack and decay phases:

- $T_{E'+}$ represents the overall duration of the attack phases;
- $T_{E'-}$ represents the overall duration of the decay/release phases.

By comparing $T_{E'+}$ and $T_{E'-}$, and remembering that for a forward signal the overall duration of the attack phases is less than the overall duration of the decay phases, the signal playback direction can be determined as follows:

$$T_{E'+} < T_{E'-} \implies \text{forward signal} \tag{6}$$

$$T_{E'+} > T_{E'-} \implies \text{reverse signal.} \tag{7}$$

The derivative of the signal envelope is computed in the *calc_der* function, using the *numpy.gradient* function [43]. The function receives as parameters the interpolating function computed before and its related x axis, and returns the derivative of the interpolating function.

```
def calc_der(interp: interpolate.interp1d, x_axis: NDArray) -> np.ndarray:
    derivative = np.gradient(interp(x_axis), x_axis)
    return derivative
```

The detection of the signal playback direction is computed in the *forward_check* function. This function takes as input the derivative function, computed with the *calc_der* function, and the signal segment, and counts the positive and negative values of the derivative. Counting the positive and negative values of the derivative, the function checks if the signal is forward or backward: if the number of positive values of the derivative is less than the number of negative values (faster attacks), the signal is forward; otherwise, the signal is backward.

Along with the forward or reverse label, the algorithm also calculates the score function S for the correctness of the result as:

$$S = \frac{\max\{T_{E'+}, T_{E'-}\}}{T_{E'+} + T_{E'-}} \times 100 \tag{8}$$

The score values represent a measure of the difference between the durations of the attack phases and the overall decay phases. This score function assigns values of correctness between 50% and 100%. Results higher than 50% are always preferable, while results closer to 50% should be considered less favorable. This is because values near 50% indicate that the durations of the attack phases and the overall decay phases are similar, increasing the likelihood of errors. In fact, if the durations are similar, it means that the probability that the algorithm miscalculates the signal time-envelope and, so, the forward result, is higher. Results above 55% are considered good because they indicate a significant difference between the durations of the attack and overall decay phases. In this case, the transients and the signal envelope are detected and calculated precisely by the algorithm, so the likelihood of errors is lower. Values higher than 90% also highlight the probability of errors in the computation, because the difference in the duration of the attack and decay phases has low probability of being this big. The difference in the duration of the phases, in fact, can be big, but not too big, because it would mean that probably the algorithm took a segment of the signal too small, containing only the attack or the decay phase of the transient.

Summarizing, the score function gives results based on the following assumption: the greater the difference in duration between the attack and decay phases, the more reliable the algorithm's result can be considered.

The function *forward_check* takes as input parameters the derivative of the signal envelope and the signal itself, and returns a tuple containing the result of forwardness (ReadingDirection) and the percentage value of correctness.

```python
def forward_check(derivative: NDArray, data: NDArray) -> tuple[ReadingDirection,
    float]:
    data = data[0 : derivative.size]

    above_0_mask = (derivative > 0) & (data != 0)
    under_0_mask = (derivative < 0) & (data != 0)

    above_0_count = np.sum(above_0_mask)
    under_0_count = np.sum(under_0_mask)

    if above_0_count + under_0_count == 0:
        perc = 0
    else:
        perc = max(above_0_count, under_0_count) / (above_0_count + under_0_count) *
 100
    direction = (
        ReadingDirection.BACKWARD
        if above_0_count > under_0_count
        else ReadingDirection.FORWARD
    )

    return direction, perc
```

4.5 Other technical considerations on the developed solution

Command line interface The algorithm has been developed as a Python package, called *reverse_detection*, using a command line interface with the following arguments and options:

1. **path**: the command line interface (cli) argument for the *filename* parameter of the *reverse_detection_noise_removed_fft_precise* (*rdnrfp*) function. It represents the path of the file (or folder) containing the signal (or signals) to analyze;

```python
@click.argument(
    "path",
    type=click.Path(
        exists=True,
        path_type=Path,
        dir_okay=True,
        file_okay=True,
    ),
)
```

2. **with-filter**: the cli option for the *filter_option* parameter of the *rdnrfp* function. It represents the option to apply the FIR high-pass filter with cut-off frequency of 800 Hz;

```python
@click.option(
    "-f",
    "--with-filter",
    help="Apply a finite impulse response (FIR) high-pass"
    " filter with linear phase and cutoff frequency set at 800"
    " Hz. Applying this filter aids in identifying transients in"
    " the audio signal.",
    is_flag=True,
)
```

3. **filter-freq**: the cli option for the *filter_frequency* parameter of the *rdnrfp* function. It represents the option to change the cut-off frequency of the FIR high-pass filter;

```
@click.option(
    "-ff",
    "--filter-freq",
    help="The cutoff frequency (in Hz) of the high-pass filter."
    " If the high-pass filter is enabled, the filter cuts frequencies from"
    " 0Hz to the frequency specified by this parameter.",
    default=800,
    type=click.IntRange(max=999, min=0),
)
```

4. **pre-post-cut**: the cli option for the *pre_post_cut_option* parameter of the *rdnrfp* function. It represents the option to apply the pre-and-post cut phase, to cut noise under -40 dB at the start and at the end of the signal;

```
@click.option(
    "-ppc",
    "--pre-post-cut",
    help="Cut noise at -40dB before and after the signal (before analyzing it)."
    " Eliminates portions of the signal with levels below -40 dB,"
    " before and after the actual start of the signal "
    " (the signal is analyzed starting from the first peak with"
    " amplitude higher than -40dB to the last peak with amplitude higher than -40dB)
    .",
    is_flag=True,
)
```

5. **noise-thresh**: the cli option for the *noise_threshold* parameter of the *rdnrfp* function. It represents the option to change the noise threshold τ utilized by the function, set by default to $\tau = -50$ dB;

```
@click.option(
    "-t",
    "--noise-thresh",
    help="The threshold (in dBFS) under which the signal is considered noise"
    " to be removed. The default value for this parameter is -50dB, which is"
    " the value for noise in the middle of a recording, i.e., silence between"
    " spoken words, cited in a scientific paper from Pretto et al.",
    default=-50,
    type=click.IntRange(max=0),
)
```

6. **window-width-mult**: the cli option for the *samplerate_mult* parameter of the *rdnrfp* function. It represents the option to specify a custom sample rate multiplier for the defining of the windows length on which the function calculates the DFT;

```
@click.option(
    "-w",
    "--window-width-mult",
    help="The window width used to perform fft expressed as "
    "multiple of the sample rate. "
    "0.5 means samplerate/2, 0.25 means samplerate/4, ... ."
    " This parameters expresses the width of the various windows on which"
    " the algorithm calculates the fft to find the period of the signal.",
    default=0.25,
    type=click.FloatRange(max=1),
)
```

7. **min-silence-length**: the cli option for the *min_silence_len* parameter of the *rdnrfp* function. It represents the option to change the minimum silence duration needed between two segments to consider them as different recordings, which is set by default to 2 seconds;

```
@click.option(
    "-l",
    "--min-silence-length",
    help="The minimum length of silence (in ms) that there has to be between two "
    "segments, to consider the two segments as separte signals.",
    type=click.IntRange(min=1),
    default=2000,
)
```

Custom types and dataclasses The developed solution utilizes some custom data types and some dataclasses to represent specific types of information and to aggregate those informations relative to the same object.

The first custom type is the class *ReadingDirection*. This class is an enumeration class used to represent the information relative to the direction of the reading. The class, in fact, is an enumeration of two labels: the FORWARD label, used to represent the fact that a signal is in the right forward direction with respect to time, and the BACKWARD label, used if the signal is time-reversed.

```
class ReadingDirection(str, Enum):
    FORWARD = "FORWARD"
    BACKWARD = "BACKWARD"
```

The second is the class *AudioMetadata*. This class is a dataclass used to store the metadata informations of an audio file. The dataclass stores the information about:

1. the bit rate of the audio file (*bit_rate*);

2. the number of audio channel of the file (*channels*);

3. the sample rate of the file (*sample_rate*);

4. the duration, in seconds, of the file (*duration*).

The *AudioMetadata* dataclass also presents a method: the *from_file* method. This method is used to create an instance of the *AudioMetadata* class, starting from the *PathLike* path of a file. Basically, the method uses the *wave.open()* [44] function to obtain a *Wave_read* object, and then the specific methods of this object are used to retrieve the informations about the audio file.

```python
class AudioMetadata:
    bit_rate: BitRate
    channels: int
    sample_rate: int
    duration: float

    @classmethod
    def from_file(cls, filename: PathLike | str) -> "AudioMetadata":
        """Create an instance of AudioMetadata from a file."""
        with wave.open(str(filename), "rb") as wave_file:
            bitrate: BitRate = wave_file.getsampwidth() * 8  # type: ignore
            channels = wave_file.getnchannels()
            samplerate = wave_file.getframerate()
            length = wave_file.getnframes() / samplerate

        return cls(bitrate, channels, samplerate, length)

    def __repr__(self) -> str:  # noqa: D105
        return f"""AudioMetadata(
    bit_rate:    {self.bit_rate},
    channels:    {self.channels},
    sample_rate: {self.sample_rate},
    duration:    {self.duration}s
)"""
```

The third is the dataclass *SignalInfo*. This dataclass is used to store the final results of the algorithm relative to an audio segment. The informations stored in this class are:

1. the channel's number of the analyzed segment (*channel*);
2. the start time of the audio segment, in milliseconds (*start*);
3. the end time of the audio segment, in milliseconds (*end*);
4. the detected reading direction, stored as a *ReadingDirection* object, of the signal (which basically is the main result of the algorithm) (*direction*);
5. the confidence in the correctness of the detected reading direction, calculated using the score function S (Formula 8) (*correctness*).

```python
class SignalInfo:
    channel: int
    start: float
    end: float
    direction: ReadingDirection
    correctness: float

    def __str__(self) -> str:  # noqa: D105
        return (
            f"channel {self.channel} from {self.start:.2f} ms to {self.end:.2f} ms"
            f" : {self.direction.value}"
            f" [Confidence: {self.correctness:.2f} %]"
        )
```

Visualization functions The developed solution also contain a module *visualization.py* where some plot functions can be found. Specifically, the plot functions developed and present in the *visualization.py* module are the following:

1. *plot_signal*, which takes as input a signal and plots it;

```python
def plot_signal(data: typing.NDArray) -> None:
    plt.plot(data)
    plt.xlabel("Sample index")
    plt.ylabel("Amplitude")
    print("\n### Plotting graph ###")
    plt.show()
```

2. *plot_peaks*, which takes as input a signal and a set of peaks indexes and plots the signal highlighting the peaks;

```python
def plot_peaks(data: typing.NDArray, peaks: typing.NDArray) -> None:
    plt.plot(data)
    plt.plot(peaks, data[peaks], "x", label="Peaks")
    plt.legend()
    plt.xlabel("Sample index")
    plt.ylabel("Amplitude")
    print("\n### Plotting peaks ###")
    plt.show()
```

3. *plot_peaks_only*, which takes as input a signal and a set of peaks indexes and plots only the peaks;

```python
def plot_peaks_only(data: typing.NDArray, peaks: typing.NDArray) -> None:
    plt.plot(peaks, data[peaks], "x", label="Peaks")
    plt.legend()
    plt.xlabel("Sample index")
    plt.ylabel("Amplitude")
    print("\n### Plotting only peaks ###")
    plt.show()
```

4. *plot_interpolate*, which takes as input a signal, an array of peaks indexes, an interpolating function and its related x axis, and plots the signal highlighting the peaks and the interpolating function;

```python
def plot_interpolate(
    data: typing.NDArray,
    peaks: typing.NDArray,
    int_fun: interpolate.interp1d,
    x_axis: typing.NDArray,
) -> None:
    ynew = int_fun(x_axis)  # use interpolation function passed as parameter
    plt.plot(data)
    plt.plot(peaks, data[peaks], "o", x_axis, ynew, "-")
    plt.plot(np.zeros_like(data), "--", color="gray")
    plt.legend()
    plt.xlabel("Sample index")
    plt.ylabel("Amplitude")
    print("### Plotting interpolating function ###")
    plt.show()
```

5. *plot_derivative*, which takes as input a signal, a derivative function and its related x axis, and plots the derivative function;

```python
def plot_derivative(
    data: typing.NDArray,
    der: np.ndarray,
    xnew: typing.NDArray,
) -> None:
    plt.plot(xnew, der, "-")
    plt.plot(np.zeros_like(data), "--", color="gray")
    plt.legend()
    plt.xlabel("Sample index")
    plt.ylabel("Derivative value")
    print("### Plotting derivative ###")
    plt.show()
```

6. *plot_graph_fft*, which takes as input the DFT of a signal and plots it.

```python
def plot_graph_fft(data: typing.NDArray) -> None:
    plt.plot(data)
    plt.xlabel("Frequency")
    plt.ylabel("Magnitude")
    print("\n### Plotting graph ###")
    plt.show()
```

All the plot functions utilize the *matplotlib.pyplot* library.

5 Detection Assessment

To test the performances of the algorithm and to evaluate if the developed solution was presenting errors, various tests were carried out. The tests consisted of running the algorithm on various datasets, composed by various files with various content types.

The first test was the empirical analysis of the first 6 audio files, as we explained in the section 4.1. Those files were used during the developing of the solution as test files and, as explained before, they consisted of the digitization of 3 magnetic tapes in both directions. The tapes were digitized with a sample rate of 96 kHz and a bit depth of 24 bit.

After the first analysis, the algorithm was tested on a dataset based on the GTZAN dataset [45]. The dataset contains 1099 audio files, divided in genres of music and speech recordings. The dataset extended the GTZAN dataset with a custom set of a hundred speech recordings in different languages. Each audio sample is coded in PCM format with a sample rate of 22050 Hz, 16-bit precision and a duration of 30 seconds. The dataset is divided into a train and a test set; the first contains 704 audio files, while the second 395 audio files, all recorded in both directions. The algorithm was tested on the whole dataset, which was divided in forward and backward (time-reversed) files and for genres:

1. blues;
2. classical;
3. country;
4. disco;
5. hip hop;
6. jazz;
7. metal;
8. pop;
9. reggae;
10. rock;
11. speech recordings.

5.1 Experimental Results

The testing phase was first performed using the following parameters on the train dataset:

- noise threshold τ set to the default value of $\tau = -50$ dB;
- finite impulse response high-pass filter NOT applied (as by default);
- pre-and-post cut option NOT applied (as by default);

- window width multiplier set to 0.5, instead of the default 0.25;
- minimum silence length set to the default value of 2 seconds.

The first results can be seen in Table 1. The performances of the algorithm showed by these results were already pretty good. The overall total percentage of correctness, in fact, was 94.3892% and the results on the various genres were satisfactory for nearly all the genres.

The same performance testing was made also using the same parameters as before, but with different window width multipliers: 0.25 and 0.125. The results for these tests can be found, respectively, in Table 2 and in Table 3.

After looking at these results, the window multiplier of 0.25 was chosen as the default one, as it was the overall best performing.

Another performance test was made with the chosen multiplier of 0.25 and also applying the FIR high-pass filter and the pre-post cut option. The results of this test can be found in Table 4.

Also, a closer look at the performances of the algorithm for all the different genres was taken.

Blues For blues music, the results were always very good, with a 100% correctness with a sample rate multiplier of 0.25 and no filter and pre-post cut. Here, instead, when applying the filter and the pre-post cut, the performances slightly dropped, with one error over 128 total answers. This was because the audio files of this genre presented clear audio transients and a not too full frequency spectrum. Also, the presence of defined drums attacks and defined attacks and decay of other instruments helped the algorithm with the analysis. Because of this, for this genre it was chosen not to apply the filter and the pre-post cut, to obtain the best performances. Doing this, the final percentage of correct answers obtained was 100%.

Classical For classical music, the performances were always worse than those of blues music. To try to find the issue that caused the performances to drop, the signal waves were inspected and the files were listened. Doing this analysis, it was found out that most of the classical music pieces present in those files were lacking clear and defined transients and, also, were presenting long attacks and short decays also in the forward files. This was due to specific techniques of the playing of the musicians, which obviously led the algorithm to make errors cataloguing the direction of the signals. The lack of defined and clear transients together with the presence of slow attacks and fast decays in forward signals, made the performances of the algorithm drop. However, applying the FIR high-pass filter and the pre-post cut option, the performances increased and the algorithm was able to obtain a 78.125% percentage of correctness. This was due to the fact that the filter helped with the defining of the transients and, consequently, helped the algorithm in those cases in which the audio transients were not clear and defined. Because of this, it was chosen to apply the filter and the pre-post cut when dealing with signals containing classical music.

Country With country music, the performances of the algorithm were better than with classical music, but still worse than with blues music. Inspecting the signal waves of this genre, it was found out that they tended to be overcrowded in the frequency range. Because of this, the algorithm struggled to

Noise threshold τ	**FIR high-pass filter**	**Cutoff frequency of filter**
-50 dB	NO	-

Pre-post cut	**Window width multiplier**	**Minimum silence length**
NO	0.5	2000 ms

Genre	**Correct answers**	**Wrong answers**
Blues forward	63	1
Blues backward	64	0
Blues total	127	1
Classical forward	50	14
Classical backward	37	27
Classical total	87	41
Country forward	59	5
Country backward	63	1
Country total	122	6
Disco forward	63	1
Disco backward	63	1
Disco total	126	2
Hip hop forward	64	0
Hip hop backward	63	1
Hip hop total	127	1
Jazz forward	62	2
Jazz backward	63	1
Jazz total	125	3
Metal forward	61	3
Metal backward	64	0
Metal total	125	3
Pop forward	61	3
Pop backward	64	0
Pop total	125	3
Reggae forward	64	0
Reggae backward	63	1
Reggae total	127	1
Rock forward	60	4
Rock backward	61	3
Rock total	121	7
Voice forward	59	5
Voice backward	58	6
Voice total	117	11

Percentage of total correct answers	94.3892%

Table 1: Correct answers given by the algorithm, divided by genre, when using a window width multiplier of 0.5

isolate clearly the attack and decay phases of the signal. Therefore, using the FIR high-pass filter to remove some parts of the frequency spectrum and to make it less full of informations, and using also the pre-post cut option, the algorithm obtained a 100% percentage of correctness. Because of this, when dealing with country music, it was chosen to apply the FIR high-pass filter and the pre-post cut option.

Noise threshold τ	**FIR high-pass filter**	**Cutoff frequency of filter**
-50 dB	NO	-

Pre-post cut	**Window width multiplier**	**Minimum silence length**
NO	0.25	2000 ms

Genre	**Correct answers**	**Wrong answers**
Blues forward	64	0
Blues backward	64	0
Blues total	128	0
Classical forward	49	15
Classical backward	46	18
Classical total	95	33
Country forward	60	4
Country backward	62	2
Country total	122	6
Disco forward	64	0
Disco backward	64	0
Disco total	128	0
Hip hop forward	64	0
Hip hop backward	64	0
Hip hop total	128	0
Jazz forward	64	0
Jazz backward	63	1
Jazz total	127	1
Metal forward	57	7
Metal backward	57	7
Metal total	114	14
Pop forward	64	0
Pop backward	64	0
Pop total	128	0
Reggae forward	64	0
Reggae backward	64	0
Reggae total	128	0
Rock forward	60	4
Rock backward	62	2
Rock total	122	6
Voice forward	61	3
Voice backward	59	5
Voice total	120	8
Percentage of total correct answers		95.1705%

Table 2: Correct answers given by the algorithm, divided by genre, when using a window width multiplier of 0.25

Disco With disco music, as with blues music, the results given by the algorithm were always very good. The best results were obtained without the high-pass filter and the pre-post cut, and with this configuration the algorithm gave a result of 100% of correctness. This is because the disco music signals were characterized, as in blues music, by well defined transients, as it is expected thinking at the very defined beats characterizing disco music, and well distributed frequencies in the frequency

Noise threshold τ	FIR high-pass filter	Cutoff frequency of filter
-50 dB	NO	-

Pre-post cut	Window width multiplier	Minimum silence length
NO	0.125	2000 ms

Genre	Correct answers	Wrong answers
Blues forward	64	0
Blues backward	63	1
Blues total	127	1
Classical forward	48	16
Classical backward	45	19
Classical total	93	35
Country forward	60	4
Country backward	64	0
Country total	124	4
Disco forward	61	3
Disco backward	64	0
Disco total	125	3
Hip hop forward	62	2
Hip hop backward	60	4
Hip hop total	122	6
Jazz forward	63	1
Jazz backward	63	1
Jazz total	126	2
Metal forward	49	15
Metal backward	53	11
Metal total	102	26
Pop forward	63	1
Pop backward	64	0
Pop total	127	1
Reggae forward	64	0
Reggae backward	63	1
Reggae total	127	1
Rock forward	64	0
Rock backward	64	0
Rock total	128	0
Voice forward	63	1
Voice backward	61	3
Voice total	124	4
Percentage of total correct answers		94.1051%

Table 3: Correct answers given by the algorithm, divided by genre, when using a window width multiplier of 0.125

spectrum. Because of this, the configuration used with this genre doesn't use the FIR high-pass filter and the pre-post cut option.

Hip hop For hip hop, the reasoning is the same as with disco music. This genre, in fact, typically presents well defined beats that help to identify clearly the transients of the signal. Moreover, this type

Noise threshold τ	FIR high-pass filter	Cutoff frequency of filter
-50 dB	YES	800 Hz

Pre-post cut	Window width multiplier	Minimum silence length
YES	0.25	2000 ms

Genre	Correct answers	Wrong answers
Blues forward	64	0
Blues backward	63	1
Blues total	127	1
Classical forward	47	17
Classical backward	53	11
Classical total	100	28
Country forward	64	0
Country backward	64	0
Country total	128	0
Disco forward	64	0
Disco backward	64	0
Disco total	128	0
Hip hop forward	63	1
Hip hop backward	60	4
Hip hop total	123	5
Jazz forward	64	0
Jazz backward	63	1
Jazz total	127	1
Metal forward	60	4
Metal backward	64	0
Metal total	124	4
Pop forward	61	3
Pop backward	60	4
Pop total	121	7
Reggae forward	63	1
Reggae backward	63	1
Reggae total	126	2
Rock forward	63	1
Rock backward	63	1
Rock total	126	2
Voice forward	63	1
Voice backward	61	3
Voice total	124	4

Percentage of total correct answers	96.0938%

Table 4: Correct answers given by the algorithm, divided by genre, when using a window width multiplier of 0.25 and applying the FIR high-pass filter and the pre-post cut option

of music is often characterized by the presence of few instruments and a well defined voice. Because of this, the frequency spectrum is not too full and it is easier for the algorithm to isolate the attack and decay phases. Therefore, for this genre it was chosen to avoid applying the high-pass filter and the pre-post cut option and, with this configuration, the algorithm obtained a 100% percentage of correct answers.

Jazz Also in the case of jazz music, the results were always pretty good. This is due to the fact that this music presents clear and defined transients and well distributed frequencies in the frequency spectrum. Because of this, the use of the FIR high-pass filter and the pre-post cut are not essential and, without them, the algorithm is able to detect and divide the attack and decay phases really well, obtaining a percentage of correct answers of 99.21875%.

Metal With metal music, the algorithm struggled to perform good when used without applying the high-pass filter and the pre-post cut. Inspecting the signal waves, it was found that the signals were typically characterized by an overcrowded frequency spectrum and by not well defined audio transients. The confusion given by these two characteristics is even increased by the presence of very small tails of the signal, or even the absence of them. This could be due to the absence of reverb in the recordings and also to the presence of a large number of instruments that tend to fill the space in which the tails and, more in general, the decay phases would be. Therefore, the high-pass filter helped reducing the amount of information present in the signal, helping the algorithm to identify clearly the transients. It was then decided to use the FIR high-pass filter and the pre-post cut option and, doing this, the algorithm obtained a percentage of correct answers of 96.875%.

Pop With pop music, the results were always very good. This is due to the fact that, as with blues, disco and hip hop music, the audio signal of this genre presented clear and defined audio transients and also well distributed frequencies in the frequency spectrum. Because of this, the best results were obtained without the high-pass filter and the pre-post cut option and, with this configuration, the algorithm obtained a 100% percentage of correct answers.

Reggae In the case of reggae music, the observations on the audio signals were the same as with blues music: these signals are characterized by defined audio transients and well distributed frequencies in the frequency spectrum. Therefore, in the case of reggae music, the high-pass filter and the pre-post cut were not applied and, doing this, the algorithm scored a 100% percentage of correct answers.

Rock The case of rock music was very similar to that of metal music. Also in this case, the algorithm without the high-pass filter performed not very good. This is because, also in this case, the signals often presented not well defined transients and a frequency spectrum that tended to appear overcrowded, but not as much as with metal music. It was then decided to apply the FIR high-pass filter and the pre-post cut in the case of rock music and, doing this, the algorithm obtained a percentage of correctness of 97.65625%.

Voice In the case of audio signals containing speech recordings, the performances of the algorithm without the filter applied were not very good. This was because these types of signals sometimes presented not well defined audio transients and decay phases. This was the case with some particular languages like, for example, Greek. In this case, in fact, the way that the language is spoken, doesn't leave space for the decay phases of the sounds and, moreover, the cadence and the rhythm of the spoken Greek can remind of a reverse recording. By applying the high-pass filter and the pre-post cut option, the algorithm obtained better performances, scoring a final best of 96.875% of correct answers.

In Fig. 13 are reported some cake plots representing the percentage of correct answers of the algorithm for each genre. As we already discussed before and as it is clear by looking at the plots, the best results are obtained on blues, country, disco, hip hop, pop and reggae music, whereas the worst results are obtained on classical music, even though they are still decent results, as the percentage of correct answers of the algorithm is 78.125%.

As explained before, along with the result of the detected direction, the algorithm also returns a score value, calculated with the Formula 8. This measure represents the probability of correctness of the algorithm's results. The percentages of correct answers and the overall mean and standard deviation for the scoring function are stored for each genre and are reported in Table 5.

Genre	Percentage of correct answers	Overall mean and standard deviation of the scoring function
Blues	100%	55.086 ± 2.857
Classical	78.125%	51.271 ± 1.096
Country	100%	54.541 ± 1.928
Disco	100%	56.438 ± 3.019
Hip hop	100%	55.582 ± 2.395
Jazz	99.21875%	54.136 ± 2.215
Metal	96.875%	53.009 ± 2.202
Pop	100%	55.954 ± 2.744
Reggae	100%	57.694 ± 3.218
Rock	97.65625%	53.969 ± 2.561
Voice	96.875%	52.261 ± 1.288

Table 5: Percentage of correct answers and average accuracy of answers, divided by genre.

The last column of Table 5, which indicates the mean and standard deviation for each genre, shows the average accuracy of the answers, calculated with the score function.

As explained before, values of accuracy closer to 50% are not good, since it means that there is not much difference between the duration of the attack and the decay and release phases. For this reason, the probability that the result is wrong is higher because higher is the probability that the algorithm miscounted the duration of the transient's phases. Instead, values higher than 55% can be considered good results. Higher values of average accuracy should, therefore, correspond to a higher percentage of correct answers.

Fig. 14, which reports for each genre the average value of the scoring function, and Fig. 15, which presents a more comprehensive overview of the data using boxplots, give a more visual representation of the results of the scoring function. The box plot in Fig. 14 shows the distribution of the values of the score function divided by genre where the whiskers are plotted following the standard convention described in [46].

Looking at these graphs, we can state that, as expected, genres with high percentage of correct

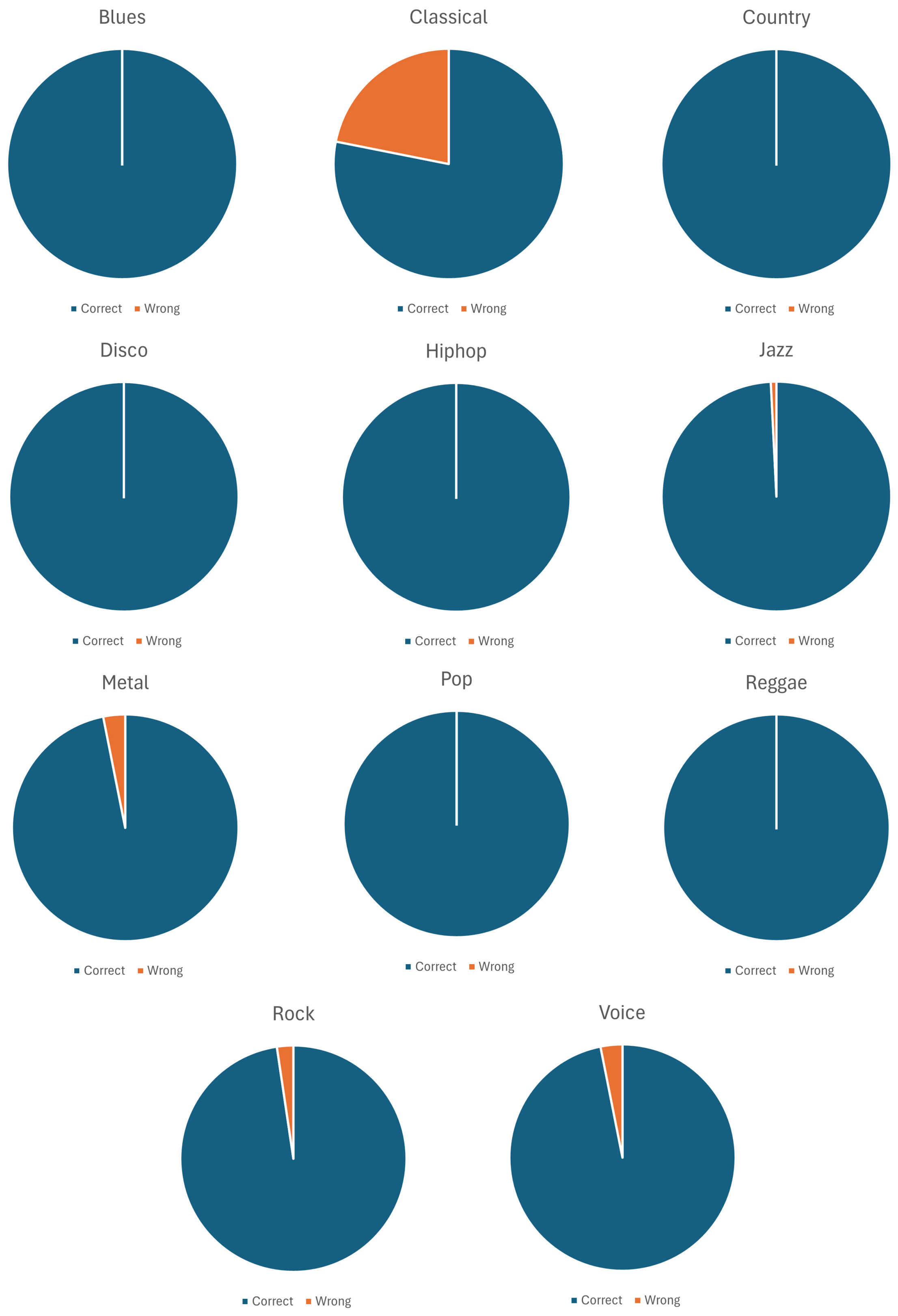


Figure 13: Cake plots representing the percentage of correctness of the algorithm, divided by genre.

answers also have high average accuracy value of the scoring function, validating our evaluation methods. So, we can state that higher values of score function correspond to higher percentage of correct answers. This outcome was expected, since our algorithm tends to give incorrect answers with poorly defined transients and, more in general, in the presence of ambiguities in the attack and

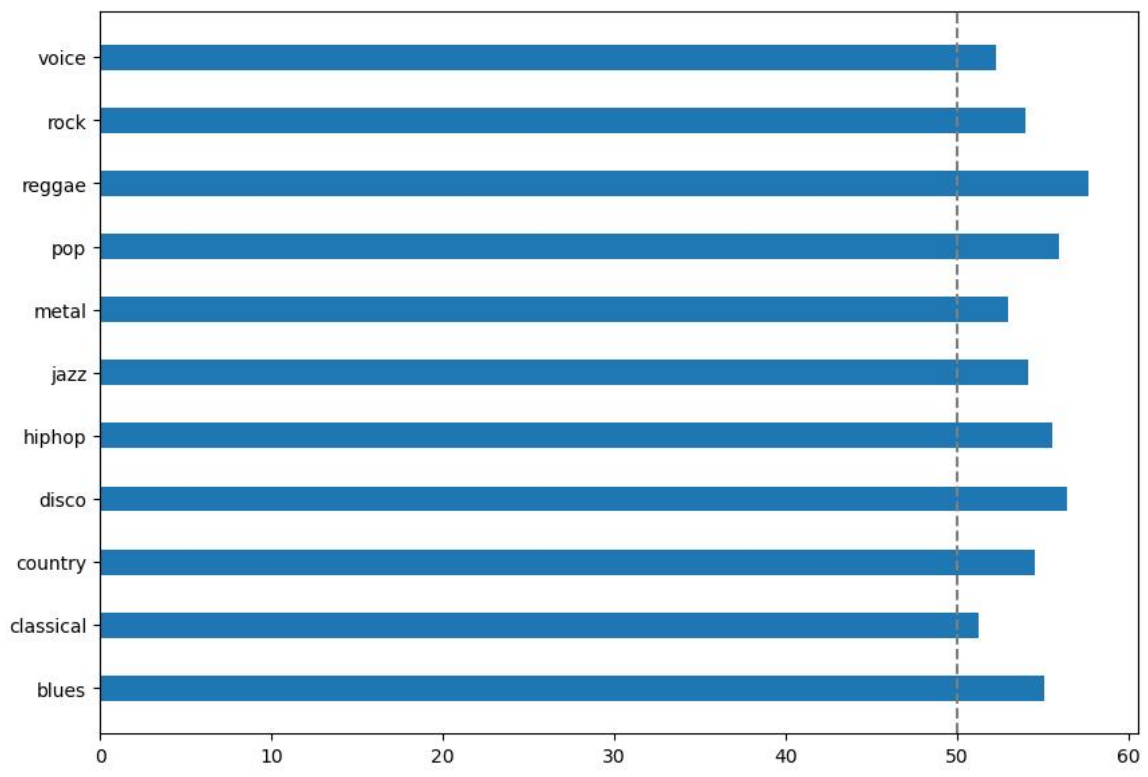


Figure 14: Average score value for each genre.

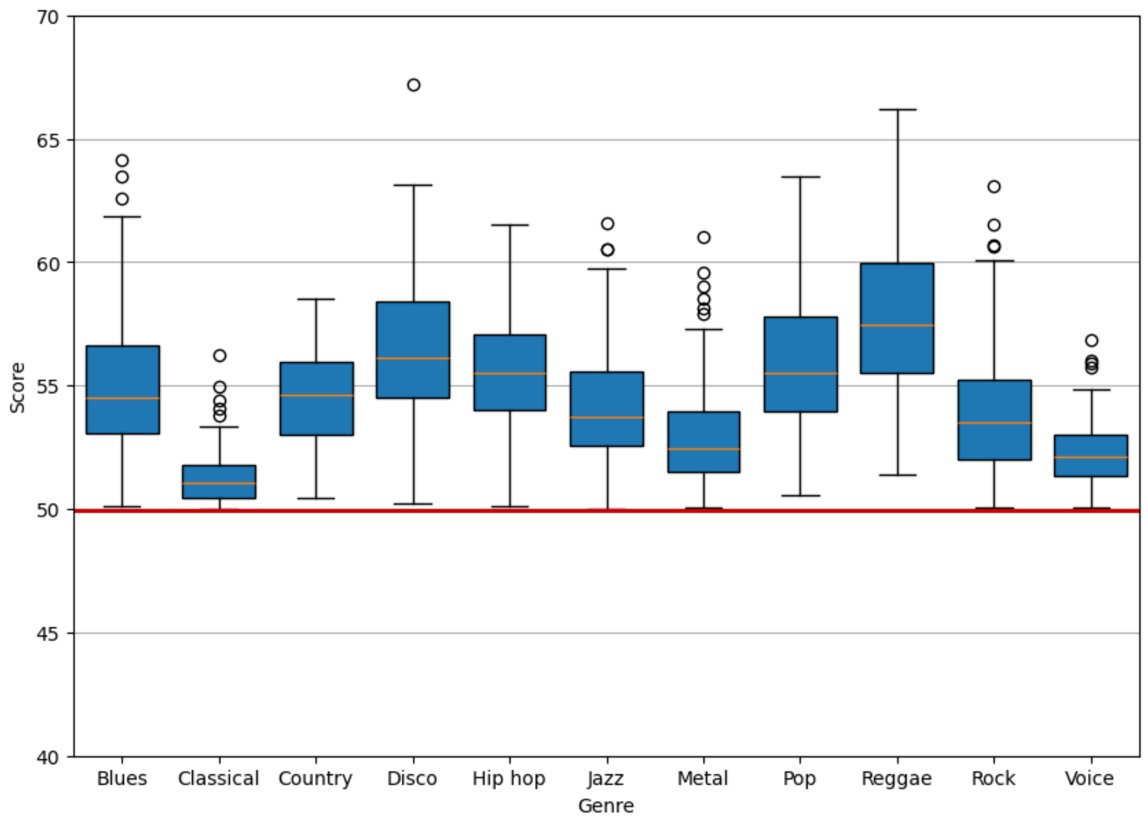


Figure 15: Score values S (see Formula 8) for different genres; values above 50% (red line) indicate more reliable results.

release phases.

From these plots we can also see that the best score-wise performing genre is reggae music, which can be said to be the genre where the algorithm gives the overall best performances. In this case, the accuracy results are consistently distributed above the 55% threshold, meaning that the algorithm gave results with high values of confidence that correspond to the higher probability of correctness. The genre where the algorithm gives the overall worst performances, instead, is classical music, where the score values consistently hovered near the 50% threshold, indicating a lower probability of correctness.

In general, we can state that the algorithm shows good results across all genres, but struggles a bit with classical music. This is due to the fact that some classical music pieces lacked distinct attack and release phases, which made it challenging for the algorithm to divide the different phases and to accurately determine the audio direction. Furthermore, some classical music audio signals presented longer attacks and shorter releases, even being recorded in the correct direction: this obviously confused the algorithm.

5.2 Assessment Using Audio Archive Material

To evaluate the performances of the algorithm in the realm of the preservation of audio documents, which was the reason the algorithm was developed in the first place, some tests on audio archive materials were carried on. The tests were performed on a series of digitized tapes of various genres and containing various recorded contents to achieve a comprehensive evaluation.

Over the past decade, the CSC has been deeply involved in the preservation and digitization of audio archives, with a particular focus on tape music. In this medium, works were frequently recorded on tapes using a range of channel configurations, speeds and directions. Due to the heterogeneity of the recorded content, these tapes provide a particularly interesting case study for testing our algorithm.

The assessment was performed on a dataset of 8 fragments digitized from 6 tapes. The original tapes that were digitized and analyzed were belonging to the archives of two well-known Italian composers of the 20th century: Luciano Berio and Luigi Nono. These two composers were some of the most important composers in the field of electronic and classical research music. Their work was of unique importance in the developing of modern music. Because of this, the test was made on these really representative documents, to evaluate the performances of the algorithm when dealing with real pieces of music and recordings that later helped to shape contemporary music and culture. Specifically, the selected tapes contained pieces of electroacoustic tape music. This type of music was not present in the genre division of the GTZAN dataset. However, electroacoustic music is a mixture of nearly all the different music genres [47]. Because of this, the assessment performed on these tapes was an assessment on audio material which is very different from the GTZAN dataset used for the previous performance assessment and, at the same time, very comprehensive of all the different characteristics of the different music genres. Therefore, the test was more complex than those performed before, given the complexity and diversity of the tapes on which it was performed.

These files were sampled at 96 kHz with 24-bit precision, and have different duration: from a couple of minutes to over an hour. Some more detailed informations about the files can be found in

Table 6.

Fragment	File name	Tape	Author	Playback speed	Duration
1	BERIO224a_15ips.wav	Tape 1	L. Berio	15 ips	00:14:49
2	BERIO224a_75ips.wav	Tape 1	L. Berio	7.5 ips	00:29:29
3	BERIO224b_15ips.wav	Tape 2	L. Berio	15 ips	00:14:47
4	LNONO042a_712ips.wav	Tape 3	L. Nono	7.5 ips	01:06:15
5	LNONO042b_15ips.wav	Tape 4	L. Nono	15 ips	00:01:36
6	LNONO070a_15ips.wav	Tape 5	L. Nono	15 ips	00:16:53
7	LNONO070a_334ips.wav	Tape 5	L. Nono	3.75 ips	00:03:09
8	LNONO070b_712ips.wav	Tape 6	L. Nono	7.5 ips	00:33:33

Table 6: Informations on the utilized audio archive material.

These digitized audio documents contain various types of content: speech recordings, vocal sounds and classical pieces of music.

These files were also chosen because they contain both forward and time-reversed signals within the same file. This made it possible to analyze the performances of the algorithm in the division and cataloguing of the various signals contained in a single file. However, in some cases, signals recorded in different directions were not separated by a significant amount of silence. Therefore, in these cases the algorithm struggled to correctly identify and separate the various different audio signals, causing several errors considering forward and backward signals as a single segment.

To mitigate this issue, different parameters for silence duration and noise threshold were tested, to try to find the parameters that suited the best each different audio file. However, when signals were closely intertwined one with the other, separating them in the correct way proved to be extremely challenging and, sometimes, impossible.

In Table 7 a summary of the algorithm's results on each different file is reported.

Fragment	Original Tape	Author	Playback speed	Correct answers with silence duration threshold at 500ms	Correct answers with silence duration threshold at 1000ms	Correct answers with silence duration threshold at 2000ms
1	Tape 1	L. Berio	15 ips	50%	75%	50%
2	Tape 1	L. Berio	7.5 ips	50%	66.66%	33.33%
3	Tape 2	L. Berio	15 ips	66.66%	66.66%	50%
4	Tape 3	L. Nono	7.5 ips	100%	50%	66.66%
5	Tape 4	L. Nono	15 ips	50%	50%	50%
6	Tape 5	L. Nono	15 ips	100%	66.66%	100%
7	Tape 5	L. Nono	3.75 ips	50%	100%	50%
8	Tape 6	L. Nono	7.5 ips	50%	75%	50%

Table 7: Assessment results on archive material.

The table, along with the reference about which file the results correspond to, contains information on how many sections of the input signal the algorithm labeled correctly as forward or backward.

The algorithm was tested on these files using different parameters for the noise threshold and the silence duration, depending on the content of each single file. During this process, the noise threshold τ was modified between -50 dB and -63 dB, according to the indications given in the paper [13] for the threshold for noise due to recording head without any specific input signal. At the same time, also the minimum silence duration was modified: we assigned to this parameters values from 0.5 seconds to 2 seconds.

To give an overview of the performances of the algorithm, changing the silence duration threshold, Table 7 and Fig. 16 contain the algorithm's results for $\tau = -63$ dB and for different values of the silence duration threshold. As we can see, a part from two exceptions (Berio_01 and Nono_05), the trend is that the algorithm performances drop when the minimum silence duration threshold is increased. This could be expected, because it represents the fact that in the files at our disposal often the various signals are separated by portions of silence that are shorter than 2 seconds. Because of this, using a minimum silence duration threshold of 2 seconds causes the algorithm to make errors dividing the different signals present in a digitized tape: if the threshold is longer than the silence portion between two signals, the algorithm takes the two signals as a single signal and gives to both the same detected direction.

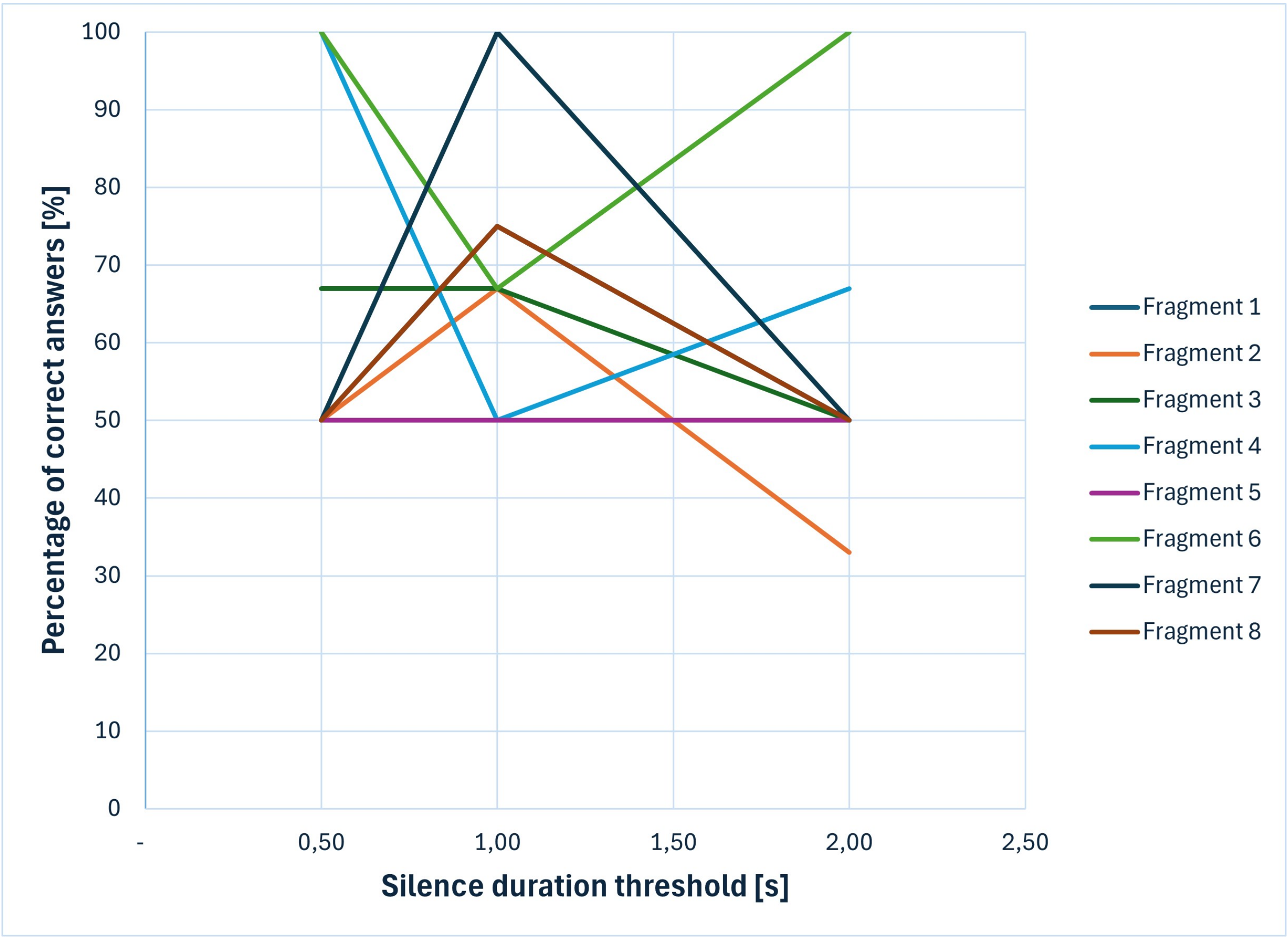


Figure 16: Percentage of correct answers in archive material as the silence duration parameter varies.

Table 8 also reports some additional notes on the algorithm's performance, to make more clear the reasons behind the occurred errors. In some cases, as said, the algorithm struggles to label correctly some sections of the input signals. This can be due to various reasons, one of which is the absence of significant portions of silence or noise between the different signals.

Fragment	Original Tape	Author	Notes
1	Tape 1	L. Berio	No clear division between forward and backward signals: the separation is not always successful
2	Tape 1	L. Berio	
3	Tape 2	L. Berio	
4	Tape 3	L. Nono	
5	Tape 4	L. Nono	
6	Tape 5	L. Nono	Poor identification of the sections
7	Tape 5	L. Nono	Excellent job in dividing sections and guessing the correct audio temporal direction
8	Tape 6	L. Nono	Identified sections tend to be too large

Table 8: Assessment results on archive material.

Because of these characteristics, we found out that the best way to use the algorithm with archival documents with significantly different characteristics is to heuristically find the parameters that work best with the single file, or the group of similar files, and to use those parameters when dealing with similar files.

Therefore, Table 7, in the columns referring to the percentage of correct answers, shows how many sections the algorithm labeled correctly, among those it found in the file. Because of this, even if the tape is the same, the algorithm can find a different number of signals and, in the table, we reported how many of them the algorithm labeled correctly as forward or backward. This is the case with tape 1 by L. Berio and tape 5 by L. Nono, which were digitized at different speeds and stored in different files. Because they were digitized at different speeds, the duration of the portions of silence between the different signals is different. Therefore, the algorithm separates differently the signals based on the silence duration threshold, and performs differently based on which threshold works better with the different files.

Looking at the results of this test, we can say that the algorithm parameters need to be fine tuned for each different group of similar files. This fine tuning is needed to allow the algorithm to properly segment the different sections present in the file. Due to the inherent variability in archival audio materials, it is in fact necessary for the operator to experiment with different parameter settings, leveraging his expertise and considering the specific characteristics of the audio being processed, to optimize the results. In some cases, however, the absence of significant portions of silence between the signals recorded on the original document, makes it difficult or even impossible for the algorithm to divide correctly the different sections.

A part from the segmentation process, we can say that the algorithm performs really well in identifying the direction of the signals. When the sections are correctly divided, in fact, the results are consistently accurate.

6 Conclusions

In this work, we presented and described a novel reverse detection approach based on the evaluation of audio envelope derivatives, which showed promising results in automating the detection of time-reversed audio segments in preserved audio documents.

This new method was developed with the aim of simplifying the process of digitization of analog audio documents, in particular open-reel magnetic tapes. The developed algorithm, in fact, is useful to automatically detect segments of audio recorded in reverse with respect to time. It is then possible, using the developed solution, to speed up the digitization process and, also, to avoid the introduction of unwanted errors, which would also be really difficult to detect.

This automatic audio reverse detection function was developed with the idea of integrating this tool in the MPAI/IEEE-CAE ARP international standard. In this context, in fact, the algorithm is an extremely useful tool to speed up the digitization workflow, minimizing at the same time the risk of introducing errors into the process.

The described method has been implemented in an algorithm which should be used during the digitization process. The algorithm could be used to avoid the need of listening to hours of recordings and to automatically find and correct reversed portions of audio. In addition to this, the algorithm would be also a useful tool in the case of the digitization of multiple audio documents at the same time. It is, in fact, possible to record multiple documents at the same time, some of which are read backwards. The algorithm would then be useful to automatically detect which documents are backward and, subsequently, digitally reverse them.

The idea for the algorithm came from the empirical observation of the ADSR (Attack, Decay, Sustain, Release) envelope of some audio signals. During this phase, it was noted that forward signals presented shorter attack phases and longer decay and release phases (due to the presence of reverberations and other characteristics of audio signals). Backward signals, instead, presented long attacks and shorter decay and release phases. Analyzing the duration of these phases, it was then possible to detect the direction of the audio signals.

The algorithm was therefore developed based on the study of the derivative of the signal envelope, calculating the total duration of the attack phases of the signal, when the derivative is positive, and the total duration of the decay and release phases, when the derivative is negative. Doing this and counting the number of positive and negative values of the derivative, the algorithm is able to determine the direction of the signal.

The proposed solution was developed as a python package and then its performances were evaluated on two different datasets. The first dataset contained a series of short audio files, divided by music genre. The second contained a series of digitized analog tapes from two important Italian composers: Luciano Berio and Luigi Nono.

The performance assessments indicate that our algorithm's performance varies based on the musical genre of the file. Specifically, the algorithm performs really well when processing audio signals containing sounds with clear and well-defined transients (e.g., drums or vocals), especially when these sounds present fast attacks and longer releases. The best results, in fact, were obtained with genres such as Blues, Reggae and Disco, which achieved 100% correctness and average accuracy of

55% to 57%. Conversely, the poorest results of the algorithm were obtained with classical music and, in particular, with those recordings that feature sounds with slow attacks and extended sustains (e.g. strings), even being recorded in the right forward direction: this feature obviously challenge accurate recognition. Because of this, there is the need for future works aiming at improving the performances of the algorithm in these cases.

The algorithm showed some limitations also in those cases in which the different recordings contained in a single file were not separated by any portion of silence. In these cases, the algorithm struggled to correctly divide the various segments and, when failed doing this, assigned the same direction to different segments with different recording directions. Although this situation is not very common in real-world scenarios, future research is needed to investigate possible implementations for improving the algorithm's efficiency in similar cases.

In conclusion, the proposed method, which is based on the evaluation of audio envelope derivatives, proved to be a possible solution for automating the detection of time-reversed audio segments in preserved audio documents. This would be useful to speed up the digitization process and to avoid the introduction of errors. The evaluation of the performances of the algorithm also demonstrates the validity of the proposed solution, but it also highlighted some of its weaknesses.

Some future work is then necessary to improve the audio file segmentation and to increase the performances of the algorithm in presence of sounds with less clearly defined transients. However, the proposed method is still a valid solution, given that, even in those critical cases, the performances were always reasonably good, with a percentage of correct answers consistently higher than 75%.

References

[1] Will Prentice and Lars Gaustad., eds. *IASA Technical Committee, The Safeguarding of the Audiovisual Heritage: Ethics, Principles and Preservation Strategy, Version 4, IASA-TC 03. International Association of Sound and Audiovisual Archives.* `www.iasa-web.org/tc04/audio-preservation`, 2017.

[2] MPAI. *Technical Specification MPAI Context-based Audio Enhancement (MPAI-CAE)" V.1.4.* 2022.

[3] "IEEE Standard Adoption of Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Technical Specification Context-based Audio Enhanced (CAE) Version 1.4". In: *IEEE Std 3302-2022* (2023), pp. 1–94. DOI: `10.1109/IEEESTD.2023.10112597`.

[4] Federica Bressan, Valentina Burini, Edoardo Micheloni, Antonio Rodà, Richard L. Hess, and Sergio Canazza. "Reading Tapes Backwards: A Legitimate Approach to Saving Time and Money in Digitization Projects?" In: *Applied Sciences* 11.15 (2021). ISSN: 2076-3417. DOI: `10.3390/app11157092`. URL: `https://www.mdpi.com/2076-3417/11/15/7092`.

[5] Marina Bosi, Fabio Zanini, Matteo Spanio, Alessandro Russo, and Sergio Canazza. "A novel derivative-based approach for the automatic detection of time-reversed audio in the MPAI/IEEE-CAE ARP international standard". In: *Proceedings of the AES 157th Convention 2024 October 8-10, New York, NY, USA*. Accepted for publication. 2024.

[6] Sergio Canazza, Giovanni De Poli, and Alvise Vidolin. "Gesture, Music and Computer: The Centro di Sonologia Computazionale at Padova University, a 50-Year History". In: *Sensors* 22.9 (2022). ISSN: 1424-8220. DOI: `10.3390/s22093465`. URL: `https://www.mdpi.com/1424-8220/22/9/3465`.

[7] S. Canazza and G. De Poli. "Four decades of music research, creation, and education at Padua's Centro di Sonologia Computazionale". English. In: *Computer Music Journal* 43.4 (2020), pp. 58–80.

[8] F. Bressan and S. Canazza. "A Systemic Approach to the Preservation of Audio Documents: Methodology and Software Tools". In: *Journal of Electrical and Computer Engineering* 2013 (2013), 21 pages.

[9] Federica Bressan and Sergio Canazza. "A systemic approach to the preservation of audio documents: Methodology and software tools". In: *Journal of Electrical and Computer Engineering* 2013 (2013), pp. 5–5.

[10] iasa. *IASA-TC 05*. `https://www.iasa-web.org/tc05/handling-storage-audio-video-carriers` [Accessed: Sep. 11 2024]. 2014.

[11] George Boston. *Safeguarding the documentary heritage: a guide to standards, recommended practices and reference literature related to the preservation of documents of all kinds*. United Nations Educational, Scientific and Cultural Organization, 1998.

[12] Dietrich Schüller. “The ethics of preservation, restoration, and re-issues of historical sound recordings”. In: *Journal of the audio engineering society* 39.12 (1991), pp. 1014–1017.

[13] Niccolò Pretto, Carlo Fantozzi, Edoardo Micheloni, Valentina Burini, and Sergio Canazza. “Computing methodologies supporting the preservation of electroacoustic music from analog magnetic tape”. In: *Computer Music Journal* 42.4 (2019), pp. 59–74.

[14] Niccolo Pretto, Alessandro Russo, Federica Bressan, Valentina Burini, Antonio Roda, Sergio Canazza, et al. “Active preservation of analogue audio documents: A summary of the last seven years of digitization at CSC”. In: *Proceedings of the 17th Sound and Music Computing Conference, SMC20*. 2020, pp. 394–398.

[15] iasa. *IASA-TC 04*. `https://www.iasa-web.org/tc04/audio-preservation` [Accessed: Sep. 11 2024]. 2009.

[16] Marina Bosi, Sergio Canazza, Alessandro Russo, Niccolo Pretto, and Leonardo Chiariglione. “An MPAI/IEEE International Standard for Audio: Overview of CAE Audio Recording Preservation (ARP) Technology”. In: *Audio Engineering Society Conference: 2023 AES International Conference on Audio Archiving, Preservation & Restoration*. June 2023. URL: `http://www.aes.org/e-lib/browse.cfm?elib=22136`.

[17] iasa. *IASA-TC 03*. `https://www.iasa-web.org/tc03/ethics-principles-preservation-strategyhttps://www.iasa-web.org/tc05/handling-storage-audio-video-carriers` [Accessed: Sep. 11 2024]. 2017.

[18] Federica Bressan and Richard L Hess. “NON-STANDARD TRACK CONFIGURATION IN HISTORICAL AUDIO RECORDINGS”. In: *Fontes Artis Musicae* 67.3 (2020), pp. 229–252.

[19] Federica Bressan, Valentina Burini, Edoardo Micheloni, Antonio Rodà, Richard L Hess, and Sergio Canazza. “Reading Tapes Backwards: A Legitimate Approach to Saving Time and Money in Digitization Projects?” In: *Applied Sciences* 11.15 (2021), p. 7092.

[20] Andrea Basso, Paolo Ribeca, Marina Bosi, Niccolo’ Pretto, Gerard Chollet, Michelangelo Guarise, Miran Choi, Fathy Yassa, and Roberto Iacoviello. “AI-based Media Coding Standards”. In: *SMPTE Motion Imaging Journal* 131.4 (2022), pp. 10–20. URL: `doi:10.5594/JMI.2022.3160793`.

[21] MPAI. *Technical Specification: Artificial Intelligence Framework (MPAI-AIF) V1*. 2022.

[22] MPAI Community. *MPAI standards*. `https://mpai.community/standards/` [Accessed: Jul. 22 2024]. 2024.

[23] MPAI Community. *Artificial Intelligence Framework (MPAI-AIF)*. `https://mpai.community/standards/mpai-aif/` [Accessed: Jul. 22 2024]. 2024.

[24] Marina Bosi, Niccolò Pretto, Michelangelo Guarise, and Sergio Canazza. “SOUND AND MUSIC COMPUTING USING AI: DESIGNING A STANDARD”. In: *Proceedings of the 18th Sound and Music Computing Conference 2021, SMC’21, Virtual Conference*. 2021.

[25] MPAI Community. *Connected Autonomous Vehicle (MPAI-CAV)*. `https://mpai.community/standards/mpai-cav/` [Accessed: Jul. 24 2024]. 2024.

[26] MPAI Community. *Governance of the MPAI Ecosystem (MPAI-GME)*. https://mpai.community/standards/mpai-gme/ [Accessed: Jul. 24 2024]. 2024.

[27] MPAI Community. *Multimodal Conversation (MPAI-MMC)*. https://mpai.community/standards/mpai-mmc/ [Accessed: Jul. 22 2024]. 2024.

[28] MPAI Community. *Neural Network Watermarking (MPAI-NNW)*. https://mpai.community/standards/mpai-nnw/ [Accessed: Jul. 24 2024]. 2024.

[29] MPAI Community. *Portable Avatar Format (MPAI-PAF)*. https://mpai.community/standards/mpai-paf/ [Accessed: Jul. 24 2024]. 2024.

[30] MPAI Community. *Context-based Audio Enhancement (MPAI-CAE)*. https://mpai.community/standards/mpai-cae/ [Accessed: Jul. 22 2024]. 2024.

[31] MPAI Community. *Compression and Understanding of Industrial Data (MPAI-CUI)*. https://mpai.community/standards/mpai-cui/ [Accessed: Jul. 24 2024]. 2024.

[32] MPAI Community. *Human and Machine Communication (MPAI-HMC)*. https://mpai.community/standards/mpai-hmc/ [Accessed: Jul. 24 2024]. 2024.

[33] MPAI Community. *MPAI Metaverse Model (MPAI-MMM)*. https://mpai.community/standards/mpai-mmm/ [Accessed: Jul. 24 2024]. 2024.

[34] MPAI Community. *Object and Scene Description (MPAI-OSD)*. https://mpai.community/standards/mpai-osd/ [Accessed: Jul. 24 2024]. 2024.

[35] MPAI Community. *AI Module Profiles (MPAI-PRF)*. https://mpai.community/standards/mpai-prf/ [Accessed: Jul. 24 2024]. 2024.

[36] LORENZO LUNARDON. “Implementation of an AI-based model for detecting speed variations by means of spectrogram images for new magnetic tapes preservation strategies”. In: ().

[37] Ajay Shrestha and Ausif Mahmood. “Review of Deep Learning Algorithms and Architectures”. In: *IEEE Access* 7 (2019), pp. 53040–53065. DOI: 10.1109/ACCESS.2019.2912200.

[38] The SciPy community. *scipy.io API reference*. https://docs.scipy.org/doc/scipy/reference/io.html [Accessed: Jul. 26 2024]. 2024.

[39] Matplotlib development team. *matplotlib.pyplot API reference*. https://matplotlib.org/stable/api/pyplot_summary.html [Accessed: Jul. 26 2024]. 2012.

[40] Librosa development team. *librosa.stft API reference*. https://librosa.org/doc/latest/generated/librosa.stft.html [Accessed: Jul. 26 2024]. 2023.

[41] Tapio Lokki, Jukka Pätynen, Sakari Tervo, Samuel Siltanen, and Lauri Savioja. “Engaging concert hall acoustics is made up of temporal envelope preserving reflections”. In: *The Journal of the Acoustical Society of America* 129.6 (May 2011), EL223–EL228. ISSN: 0001-4966. DOI: 10.1121/1.3579145. eprint: https://pubs.aip.org/asa/jasa/article-pdf/129/6/EL223/13645437/el223_1_online.pdf. URL: https://doi.org/10.1121/1.3579145.

[42] Garima Sharma, Kartikeyan Umapathy, and Sridhar Krishnan. “Trends in audio signal feature extraction methods”. In: *Applied Acoustics* 158 (2020), p. 107020.

[43] NumPy Developers. *numpy.gradient API reference*. `https://numpy.org/doc/stable/reference/generated/numpy.gradient.html` [Accessed: Aug. 05 2024]. 2024.

[44] The Python Standard Library. *wave.open API reference*. `https://docs.python.org/3/library/wave.html` [Accessed: Aug. 07 2024]. 2024.

[45] George Tzanetakis, Georg Essl, and Perry Cook. *Automatic Musical Genre Classification Of Audio Signals*. 2001. URL: `http://ismir2001.ismir.net/pdf/tzanetakis.pdf`.

[46] M. Hubert and E. Vandervieren. “An adjusted boxplot for skewed distributions”. In: *Computational Statistics & Data Analysis* 52.12 (2008), pp. 5186–5201. ISSN: 0167-9473. DOI: `https://doi.org/10.1016/j.csda.2007.11.008`. URL: `https://www.sciencedirect.com/science/article/pii/S0167947307004434`.

[47] Sergio Canazza and Alvise Vidolin. “Introduction: Preserving electroacoustic music”. In: *Journal of New Music Research* 30.4 (2001), pp. 289–293.

Ringraziamenti

Al Prof. Canazza. Grazie per questo lavoro insieme e per le opportunità che mi ha concesso. Farò tesoro di tutte le nozioni che mi ha dato e della passione che mi ha trasmesso.

Ai miei co-relatori Matteo Spanio e Alessandro Russo. Grazie per la supervisione, i consigli e per il grande supporto che mi avete dato durante questo percorso. Avete arricchito le mie conoscenze e le mie abilità, ma anche la mia passione per questo ambito e per questo lavoro.

Ai miei genitori, Roberta e Stefano. Grazie dell'infinito amore e supporto che mi dimostrate ogni giorno. Siete stati, siete e sarete sempre esempio di vita.

Ai miei fratelli, Marco e Davide. Grazie per i consigli, le dritte e le risate. Oltre ad essere miei fratelli, siete miei grandi amici.

A Chiara. Grazie per l'amore, la pazienza e il supporto enorme che mi hai dato in questo percorso di studi e, più in generale, nella vita in questi anni. Sei la mia migliore amica. Se sono arrivato a questo obiettivo è anche merito tuo.

Ai CNS. Grazie per le risate e la spensieratezza che sapete darmi. Siete fonte di leggerezza e so che ci sarete sempre.

A Daniele. Grazie per essere stato così presente in questo percorso. Che fosse per una puntata di X Factor, per una partita (brutta) del Milan o per una semplice pizza, ci sei sempre stato.

Ai miei compagni di corso, nonché grandi amici. Grazie per tutto l'aiuto che mi avete dato e per aver fatto volare così velocemente questi 5 anni.

Ai miei coinquilini. Grazie per aver trasformato un semplice appartamento in una casa accogliente. Siete stati dei grandi compagni di viaggio.

Ai tutti i miei amici. Grazie per tutte le risate e per i momenti insieme. Mi sono sempre sentito parte di qualcosa.

A tutti voi, grazie.

[illegible]