University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Automated Testing of Medical Equipment using Sensor-based Spectral Analysis

Jano Sven Vukadinovic
Trimmis, Graubünden
Student ID: 19-175-371

Supervisor: Jan von der Assen, Chao Feng
Date of Submission: May 17, 2023

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich, 

Signature of student

Zusammenfassung

Die globale Pandemie, die durch das SARS-CoV-2-Virus ausgelöst wurde, hat die entscheidende Rolle von lebenserhaltenden Beatmungsgeräten in den Fokus gerückt und zu einer gesteigerten Wahrnehmung dieses Themas in der breiten Öffentlichkeit geführt. Vor dem Hintergrund der hohen Nachfrage nach Beatmungsgeräten ist es von entscheidender Bedeutung, sicherzustellen, dass alle Aspekte dieser Geräte rigoros und effizient getestet werden. Eines dieser Merkmale ist die Fähigkeit, Alarme auszulösen, sobald sich der Zustand des Geräts verändert, Aufmerksamkeit erforderlich ist oder ein unerwarteter Ausfall eintritt. Die Bedeutung solcher Alarme wird ersichtlich, wenn man bedenkt, dass sie potenziell dazu beitragen können, das Leben eines Patienten zu retten. Diese Alarme entsprechen internationalen Standards. Das Ziel dieses Projekts ist die Entwicklung eines Prototyps zur Evaluierung verschiedener Alarmtypen und die Sicherstellung, dass sie internationalen Standards und Herstelleranforderungen entsprechen. Des Weiteren wird im Rahmen des Projekts die Effektivität des Prototyps hinsichtlich der Erreichung dieser Ziele evaluiert.

Abstract

The global pandemic caused by the SARS-CoV-2 virus has brought to the fore the crucial role of life-supporting ventilators, prompting a heightened awareness of this issue among the general public. Given the high demand for ventilators, it is imperative to ensure that all aspects of these devices are rigorously and efficiently tested. One such feature is the ability to trigger alarms when the condition of the device changes, requires attention, or an unexpected failure has occurred. The significance of such alarms becomes evident when one considers that they can potentially save a patient's life. These alarms are in accordance with internationally accepted standards. The objective of this project is to develop a prototype for evaluating different types of alarms and ensuring that they comply with international standards and manufacturer requirements. Furthermore, the project will investigate the effectiveness of the prototype in meeting these objectives.

Acknowledgments

I would like to begin by expressing my gratitude to my supervisor, Jan von der Assen, for his unwavering support and constructive feedback throughout the project.

I would like to express my gratitude to Prof. Dr. Burkhard Stiller for providing me the privilege of writing my thesis at the Communication Systems Group (CSG) of the University of Zurich.

I am grateful to Hamilton Medical AG and my immediate superior, Marco Costa, for affording me the chance to devise a testing system for life-saving ventilators.

I would like to express my gratitude to my wife, Lara Vukadinovic, for her invaluable support during the course of this thesis. Her contribution was of the utmost importance.

I would also like to express my gratitude to my brother-in-law, Spencer Huete, for taking the time to proofread this thesis.

Contents

Chapter 1

Introduction

This first chapter elucidates the underlying motivation of the project and describes the tasks required to achieve the objective. It also provides an overview of the different chapters and what they entail.

1.1 Motivation

In the field of medical equipment, ventilators are of paramount importance for the care of patients, particularly in the management of respiratory diseases. The functionality and reliability of audible alarms have a significant impact on the effectiveness of these devices. Alarms are employed to notify healthcare personnel of potential issues with the device, and it is therefore crucial to ensure that they are audible and effective.

The past has demonstrated that some companies have neglected careful testing of ventilators [1]. This underscores the importance of testing all aspects of a ventilator, including alarms. Failure to properly test a ventilator can result in the loss of approval to sell the device, this can be devastating to a company, given the resources expended to develop and manufacture such devices. Moreover, the lack of comprehensive testing of medical devices has the potential to significantly impact patient lives.

The primary motivation for this testing prototype is to enhance patient safety. Ventilators are essential for patients who are unable to breathe independently. A reliable alarm system can promptly notify healthcare professionals of potential issues such as mechanical failures, disconnections, or various other errors. This could potentially save lives by ensuring a prompt response to such incidents. Therefore, it is of critical importance that these alarms function properly.

1.2 Description of Work

The objective of this thesis is to design and develop a system for real-time audio capture, analysis, and to test its correctness. The audio to be tested is that of the three different

priority alarms that a ventilator is required to have. In order to achieve the goal of implementing this prototype, this thesis is divided into a number of distinct sections.

The initial section comprises a comprehensive literature review with the objective of determining the scientific foundation of the subject of alarm recognition and testing. The tools and methods employed in the aforementioned papers are then utilized as a foundation for the acquisition of background knowledge relevant to the prototype. Once the a background knowledge has been acquired, it is employed in the design of the solution's architecture. After the architecture has been designed, the prototype is implemented. Finally, after the solution has been successfully implemented, it is evaluated based on a series of case studies.

1.3 Thesis Outline

The following is a structured overview of the thesis. Chapter 2 presents the principles and essential concepts that are fundamental to comprehending audio analysis and spectral analysis as a subset of it. Chapter 3 provides an overview of the current state of the science and a brief history of the tools used to analyze sound. The chapter also includes a table that presents a comprehensive overview of all the scientific theses that were examined during this stage. Chapter 4 explains the architectural design of two proposed solutions and defines the system requirements. The two implementations are then described in detail in Chapter 5, along with an explanation of how the solutions developed, the difficulties that arose, and how they were resolved. The sixth chapter presents a detailed evaluation of the two proposed solutions. The final chapter presents a proposal for future work based on the solutions presented in this thesis and provides a summary of the thesis.

Chapter 2

Background

The purpose of this section is to provide an understanding of the underlying knowledge and tools that are essential for this thesis. This section is intended to familiarize the reader with the field of audio and its processing.

2.1 Sound

Sound is energy traveling through a medium, such as air and is defined as "an airborne version of vibration" [2, p. 23]. When sound travels through a medium, it moves the atoms in the air out of place creating a longitudinal wave in its path [3]. This means that the particles vibrate parallel to the direction of the wave.

2.1.1 Digital Audio

In nature, sound is continuous motion. This implies that it contains an infinite number of signals within a given time window. In order for sound to be processed and analyzed, it must be converted into discrete values, or data, because no computer can handle infinite values in an array. The process of digitizing audio begins with the use of a microphone to convert sound waves into their electrical counterpart. This electrical signal is then digitized by an analog-to-digital converter [4], which creates the digital representation through a process called 'sampling'. Sampling is the process of taking a sample of an audio source at certain intervals of time [5].

The spacing of the sample intervals determines the frequency of the audio signal. To illustrate this concept more clearly, consider the following example:

We have a sample rate of $X_{sr} = 0.005$s. This implies that every X_{sr} a sample is taken for digitization. We now take the inverse of this sample rate $I_{sr} = 1/0.005$s. This frequency is measured in Hertz (Hz). Consequently, with a sample rate of 0.005s, the frequency is 200 Hz.

One of the most significant issues in the context of sampling audio is the question of which frequency should be selected for the digital audio version in order to approximate the analog version. This question is addressed by the Nyquist-Shannon theorem. This theorem states that the rate of sampling should be at least double the signal bandwidth [6].

2.1.2 Quantization

The subsequent stage in the conversion of analog audio signals to digital ones is quantization. Thus far in the context of sampling, only the aspect of timing has been considered. Quantization, however, concerns the amplitude values. Amplitude is defined as the strength of a sound wave, which is typically correlated with the loudness or volume of sound and is measured in decibels (dB). The decibel level of everyday sounds ranges from 30 dB, which is the threshold for a whisper, to 60 dB, which is the level of a normal conversation. A level of 120 dB is equivalent to the sound of a police siren. Quantization is the process of converting the discrete amplitude values to analog ones. This is achieved by allocating each value that was sampled to the closest predefined available digital set [7].

The resolution of the digital set, or number of bits used, is defined by the number of bits. The higher the bit depth or number of bits, the better the quality of the digital audio. Usually, the sound has a bit depth of either 16, 24 or 32 [8]. The process of quantization involves converting a continuous value to a discrete value by rounding. This introduces quantization noise into the sample [9]. This noise is reduced by increasing the bit depth.

2.2 Sound Analysis

Once the analog sound has been converted to digital audio, it is then ready to be subjected to analysis. The following section presents an overview of the most commonly utilized methods and tools for audio analysis.

2.2.1 Audio Waveform

The initial step in the process of audio analysis is typically the creation of a visual representation. The waveform is a graphical representation of the pattern of sound in the time domain [10]. This visualization enables the observation of specific events in time, such as the loudness of the sound or any irregularities. Additionally, patterns in the audio stream can be discerned.

2.3 Frequency Spectrum

An additional method for representing audio is through a frequency spectrum. This frequency domain can be obtained through the computation of the discrete Fourier transform

(DFT). The representation describes the individual frequencies that comprise the signal and their respective strength [11]. This becomes particularly intriguing when examining specific windows within an audio sequence, as it allows for the visualization of individual frequencies at a fixed point.

2.4 Spectrogram

The combination of time and frequency domain is referred to as a spectrogram. The audio signal is divided into smaller segments, and a Fourier transform is applied to each segment [12]. The resulting plot illustrates the temporal evolution of the frequency. It is a highly informative representation, providing insights into the relationships between time, frequency, and amplitude.

2.5 Spectral Analysis

Spectral analysis represents a technique that is employed across a diverse array of disciplines. Its objective is to gain an understanding of the frequency content of a signal or system [13]. This objective is achieved by first decomposing the signal into its constituent frequencies and then conducting a subsequent analysis of their magnitudes and phases.

2.5.1 Fourier Transform

The Fourier Transform is a mathematical tool that decomposes a signal into its transferable frequencies. It takes a signal in the time domain and transforms it to represent it in the frequency domain [14]. In other words, the model takes as input the audio signal and the amplitude spectrum, and gives as output the pure frequencies that make up the signal.

2.5.2 Fast Fourier Transform

The Fast Fourier Transform (FFT) is an algorithm that is used to compute the Discrete Fourier Transform (DFT) of a sequence or its inverse in an efficient manner [15]. In contrast to the brute-force approach of the DFT, which has a time complexity of O(N2), the FFT reduces the complexity to O(N log N), making it significantly faster for practical implementations [16].

2.5.3 Power Spectral Density

The Power Spectral Density (PSD) is a measure of the power of a signal as a function of frequency. It provides insights into the distribution of power across different frequency components of a signal, revealing information about its frequency content and characteristics.

2.5.4 Wavelet Transform

Wavelet Transform is a mathematical tool for analyzing signals, similar to the Fourier Transform but with certain advantages, particularly in capturing both frequency and time information simultaneously. Unlike the Fourier Transform, which uses sinusoids as basis functions, the Wavelet Transform employs wavelets, which are short-lived waveforms localized in both frequency and time domains.

The Wavelet Transform is an effective method of decomposing a signal into various frequency components at varying resolutions. This enables the analysis of transient features and non-stationary signals, which are often challenging to analyse using traditional techniques. This makes it particularly useful in applications such as signal denoising, compression, and feature extraction from time-varying signals [17].

2.6 Ventilators

Ventilators are medical devices designed to support patients who have difficulty breathing or are unable to breathe on their own. They function to deliver oxygen to the lungs and remove carbon dioxide from the body. Ventilators play a critical role in a variety of medical settings. These include intensive care units (ICUs), operating rooms, and emergency departments.

These devices consist of a control system and some mechanical components. The control system is responsible for regulating the airflow, pressure and volume of oxygen delivered to match the patient's needs. The mechanical components include a breathing circuit that connects the ventilator to the patient's airway and a set of valves and sensors that monitor and control the airflow [18].

2.6.1 Alarms

Ventilators are equipped with a variety of safety features to ensure patient safety, such as alarms for high or low airway pressure, disconnection, and power failure. They require careful monitoring by trained healthcare professionals to adjust settings and respond quickly to changes in the patient's condition, which can save lives.

These alarms are in compliance with the IEC 60601-1-8 international standard. Such alarms assist in the identification and resolution of potential hazards that could compromise patient care. These alarms notify healthcare professionals of any issues or irregularities in the patient's ventilation parameters or in the functionality of the ventilator itself. In order to ensure the optimal functioning of ventilators, it is essential to implement three distinct alarm systems: The alarms are classified as High Priority, Medium Priority, and Low Priority. The aforementioned alarms are triggered in accordance with the severity of the reported error. The requisite specifications for IEC 60601-1-8 are presented in Table 2.1.

Characteristics	High Priority Alarm	Medium Priority Alarm	Low Priority Alarm
Number of pulses	10	3	1 or 2
Repeating	every 2.5s to 15.0s	every 2.5s to 30.0s	no repeat
Frequency Range	150Hz-1000Hz	150Hz-1000Hz	150Hz-1000Hz

Table 2.1: IEC 60601-1-8

Chapter 3

Related Work

This chapter presents a discussion of various use cases where spectral analysis or other tools have been employed to identify patterns in audio. This chapter concludes with a summary of the various solution approaches, accompanied by a table that provides a concise overview of these approaches.

3.1 Related Research Papers

In the course of conducting research for this thesis, IEEE Xplore and Google Scholar were employed as primary sources for determining the current state of scientific knowledge, in conjunction with testing audio with spectral analysis. During this ”research phase,” keywords were utilized to identify related materials. Keywords such as ”spectral analysis,” ”sound analysis,” or ”sound detection” proved to be invaluable in acquiring an understanding of the current scientific landscape on the topic of sound analysis of alarms.

The findings of the current state of research show that numerous studies have been conducted in the analysis of sound alarms. Further, the state shows that different solution approaches may be classified together. The model-based approach and the non-model-based approach are two distinct methodologies. Each of these solutions has its own set of advantages and disadvantages, which depend on the specific use case. In the following sub-chapters existing literature is reviewed and key concepts and theories used are discussed.

3.2 Beginning of Audio Detection Sound Alarms

The earliest alarm detection system with correlation to the theme of this thesis was found in [19]. In this research paper, two different solutions are discussed. One is model-based using neural networks borrowed from speech recognition, while the latter is a non-model based on sinusoidal modeling where prominent spectral features are extracted and analyzed. This analysis allows for background noise to be reduced. However, in said

paper, only alarms, such as telephone rings, sirens, and bells, are tested. Both solutions discussed had a high amount of missing alarms, with the neural net system having a slightly lower error rate. The high error rate can be attributed to the fact that general alarms were tested, and that for higher accuracy, more alarm examples should have been used for training the machine learning models.

3.3 Model-Based Systems

The model-based solution, which employs a technique utilized in speech recognition, is also presented in [20] with significantly enhanced accuracy and a markedly reduced response time. A system was developed to facilitate the transmission of audible alarms to individuals who are deaf or hard of hearing. In this paper, a database with prerecorded alarm sounds was created. Different realistic scenarios were recorded with a low-budget microphone at a sampling frequency of 16kHz and a signal amplitude of 16bit and stored using the WAVE container format [21]. After this initial stage, features were extracted using a tool called Hidden Markov Model Toolkit (HTK) and converted into sequences of feature vectors. In the next step, patterns were matched, meaning the previously extracted vectors were assigned to a membership class. This classification was done through the usage of an artificial neural network. Before the alarms were recognized, the performance was enhanced by increasing the observation window. This allowed the recognition system to make decisions on a much wider range of spectral features. Through this last step, performance was optimized, and false alarms were reduced.

3.4 None Model-Based Systems

Non-model based systems do not use a previously trained model, but rather they learn heuristically from the data itself and do not suffer from model bias [22]. Two none-model-based solutions were introduced in [23]. One of the solutions used a Root Mean Square (RMS). This approach monitors or detects changes in the level of sound. Through this method an intensity profile of the average power is extracted and used to identify changes. Such changes may be interpreted as alarms. The second approach uses the Power of the Normalized Auto-correlation Function (PNA). This function can detect periodic patterns in the audio signal. It does this by measuring the similarity of a signal with a delayed version of itself. Both solutions have their advantages and disadvantages depending on the setting and the alarm being tested for. The author suggested combining both approaches to achieve a more promising, optimized, and accurate solution in future work.

Another algorithm used to detect sound alarms in cockpit voice recordings (CVR) was shown in [24]. The CVR records are also known as the black box of the aircraft and record all sound in the cockpit during a flight. This implies that not only are the voices of the pilots recorded, but also the sounds of the alarms. The background alarms were identified through a four-step process.

The first step entailed building templates for every kind of alarm. This was done by using STFT and analyzing its spectrogram for features of prerecorded material. The next step involved detecting the pitches of frames. This step aimed to identify substantial feature frequencies of given frame sizes by applying 2 equations. These equations made sure that these frames incorporated local maximum amplitude and were amongst the most powerful pitches. This step was conducted to facilitate the comparison with the template values in the first step, which was done in the third step. However, up to this point, only spectral properties were considered, and and impact of noises was neglected, which resulted in a high roughness of the data. This was encountered in the fourth step. In this last step, isolated classification results were eliminated through a middle-valued filter, and the temporal properties were verified by applying the top hat and bottom hat transformations. This solution acquired a high accuracy and high detection rate in recognizing alarms. However, it was only effective for prerecorded alarms and not for real-time audio input.

Detection of alarm sound was also achieved in [25], where an algorithm was introduced to automatically detect sound alarms in hearing protection devices. Part of this approach was to evaluate the sound amplitude periodicity in a given bandwidth. This system first used a band-pass filter to filter out unwanted frequencies that are not associated with the alarms that are trying to be detected. After passing through the filter, the envelope detector generated a signal with its amplitude features over time. This step basically creates a shape of the signal, which includes a summary of different amplitudes of the signal. The next step was to use the Auto-correlation Computing Block function similar to [23]. This function allows to recognize repeating patterns. In the final stage of the process, a detection algorithm was developed to integrate the results of the preceding three steps and apply decision rules to determine whether the input signal should be classified as an alarm.

3.5 Part-Based Systems

In [26], a part-based model is presented for detecting sirens in traffic noise. PB-Model consists of individual parts that have a defined yet also configurable configuration. This flexibility is used to detect sirens. The spectrogram is used as an image and the flexibility of PBM training allows spectral and temporal analysis. This solution was compared with a HMM approach and concluded that a PBM solution performed better.

3.6 Combination of Models

A combination of a model-based and a none model-based is introduced in [27]. The paper discusses statistical models which take advantage of spectral and temporal characteristics of different medical alarms found in a neonatal intensive care unit environment. The author states that through this solution, the baseline performance is improved by 60%. Nevertheless, the error rate still remains high. This can be attributed to the exposure to loud noises which are heard in the NICU.

3.7 Simple Algorithm

The spectro-temporal properties are also made use of in [28]. This solution proposed an algorithm that entails two steps. In the first step, single alarm tones are extracted by analyzing their frequency content. The recording is preprocessed by using a low-pass filter, allowing only specific frequencies to pass through – this is also known as down-sampling. The recording is then windowed, and the short time intervals are analyzed. Then, an algorithm is used to evaluate the DFT coefficient corresponding to specific frequency bins. These bins allow possible frequencies to be captured of alarm tones. After specific detection criteria are met, an output is created in the form of a log. In the second step, the log is refined to ensure accuracy and reliability. The final log results in a sequence of specific alarms. Astonishingly, the accuracy achieved by this solution was 99%, and only a minimal amount of false alarms were detected.

3.8 Further Solutions

In [29], an electronic stethoscope was used to analyze characteristics of abnormal breathing sounds. In [30], CNN and STFT are combined to analyze breathing sounds recorded by a stethoscope to detect abnormalities. CNN and FFT were also used in [31] to detect COVID-19 patients based on of breath, cough and sound.

In [32], FFT is used for Haar-based coefficients. To detect illegal deforestation. This algorithm takes a spectrogram of an audio signal and transfers it to a set of coefficients, which capture features of the signal. Usually, this method is used in object recognition. The author of this paper uses it to detect specific frequency patterns. The goal is to detect frequency patterns which are identical to those of an actively cutting chainsaw.

3.9 Discussion of Solutions

As seen by the various solutions and propositions, there are many variations used to detect patterns in an audio signal. Still, some universal principles are observed. Such as the usage of filters and the extractions of temporal and spectral properties. The objective of this thesis is to identify the most effective algorithms and principles from various sources in order to develop an optimal methodology for testing sound alarms. This methodology must be able to distinguish between the three different priority alarms on a ventilation system.

It is evident that the majority of solutions employ machine learning for pattern learning and pattern recognition. It would be of interest to contrast a solution that does not employ machine learning with one that incorporates it. A substantial proportion of the literature indicate that the majority of solutions employ pre-recorded data for extraction, rather than real-time data. This presents a challenge, as the objective of this system is to

Table 3.1: Comparison of Related Work.False Rate (FR), Missing Rate (MR)

Paper	**Year**	**Solution**	**Accuracy**	**Data**
[19]	2001	ANN, FTT	200% FR	General Alarms
[20]	2006	ANN, FTT, MFCC	99%	Sirens
[23]	2012	RMS, PNA	80%	Alarm Database
[24]	2009	STFT	99%	CVR Recordings
[25]	2013	ALG		Pulsed, Sirens, Alternating
[26]	2013	PBM, HMM		Police Sirens
[27]	2018	DFT	32% MR	NICU Recordings
[28]	2022	ML	99%	NICU Recordings
[30]	2023	CNN, STFT	85.27%	Lung Sounds
[32]	2018	FFT, Haar Wavelet	97.28%	Chainsaw Sounds
this work	2024	FFT, STFT, CNN	98% and 75%	Ventilator Alarms

create a system that is capable of analyzing real-time data, identifying the relevant alarm, and determining whether an alarm has been triggered correctly or not.

The scientific status also indicates that no system has yet been developed to test or predict alarms for ventilators. This thesis seeks to address this gap in the literature.

Chapter 4

Architecture

This chapter presents the design and architectural framework of the proposed solution. The objective of this section is to provide the reader with a comprehensive understanding of the solution without providing exhaustive details. The following chapter delves into the decision behind the design of the proposed audio analysis system. This architecture proposal is then used as the basis for the implementation of the prototype discussed in the next chapter.

The objective is to validate and test audio alarms by creating a device (testbox) with a microphone and software that is connected to the Internet. The device is equipped with an Ethernet interface that enables users to connect to a message broker in order to test audio alarms. The user is able to send a request to the device, which will then respond.

4.1 Algorithmic Analysis Component Architecture

The architecture of the sound analysis system is shown in Figure 4.1 The initial stage of the system is to capture the data, which, in this case, is the sound alarm waves. The next step is to process the data. In this step, the captured audio signal is processed by filtering and windowing. Once this pre-processing is complete, the temporal and spectral characteristics are extracted using an FFT transformation. These properties are then stored in a buffer and are used to judge whether a sound is playing or not. Finally, an algorithm checks which alarms have been heard in the past, selects the correct alarm based on this information, and writes the information to a log file.

4.2 Machine Learning Analysis Component Architecture

The architecture of the sound machine learning analysis system is depicted in Figure 4.1. In a manner analogous to that described in the preceding section, audio is recorded. Following the capturing phase, the data is preprocessed in order to ensure its compatibility with the CNN machine learning model. This is done by taking a segment of the capture

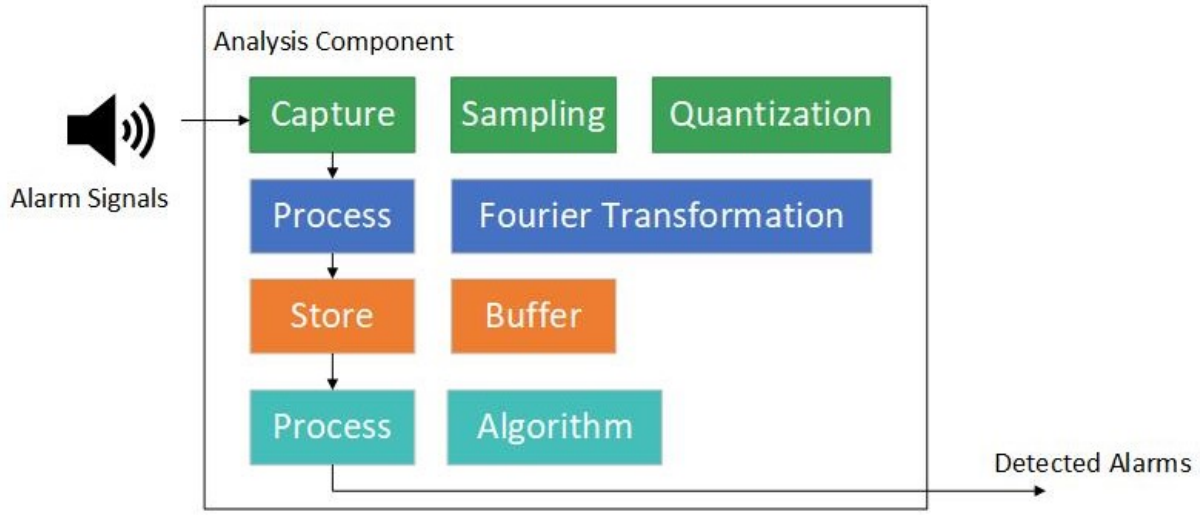

Figure 4.1: Audio Analysis Component

audio using STFT and creating a spectrogram. The previously trained model is then utilized to make an estimation about which alarm has been triggered.

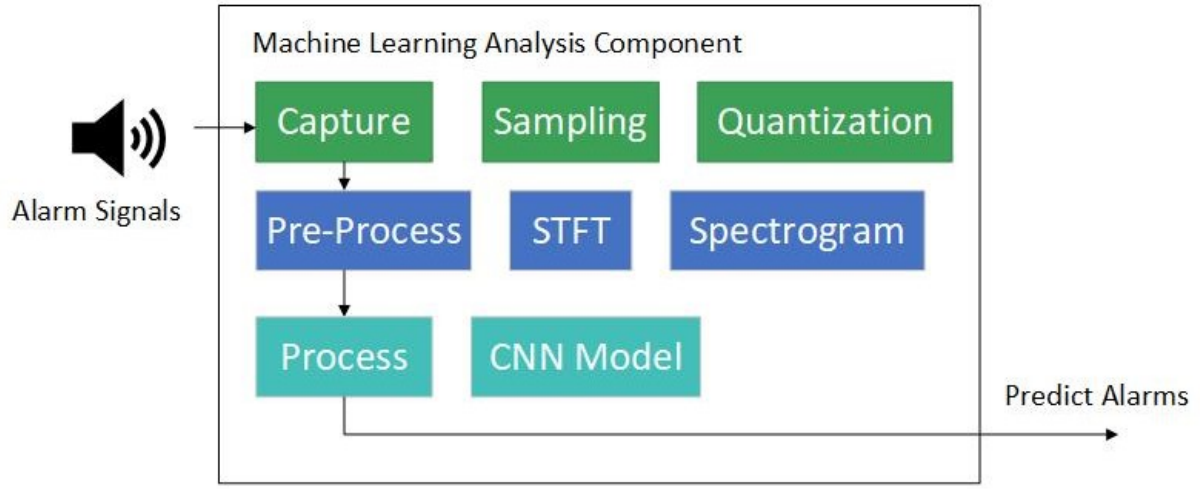

Figure 4.2: Machine Learning Audio Analysis Component

4.3 Hardware and Software Components

The hardware and software components are shown in Figure 4.3. The user interacts with a front-end platform and specifies what types of alarms need to be tested and on what ventilators. This sends a message to a backend server application, which is connected to the testbox device via a message broker. The testbox has an interface that handles

requests and responses, once a request has been made it sends a response back to the backend application. The backend stores interactions with the testbox device in an SQL database as a history log. The backend application sends feedback back to the user at the front end to explain whether a test was successful or ultimately failed.

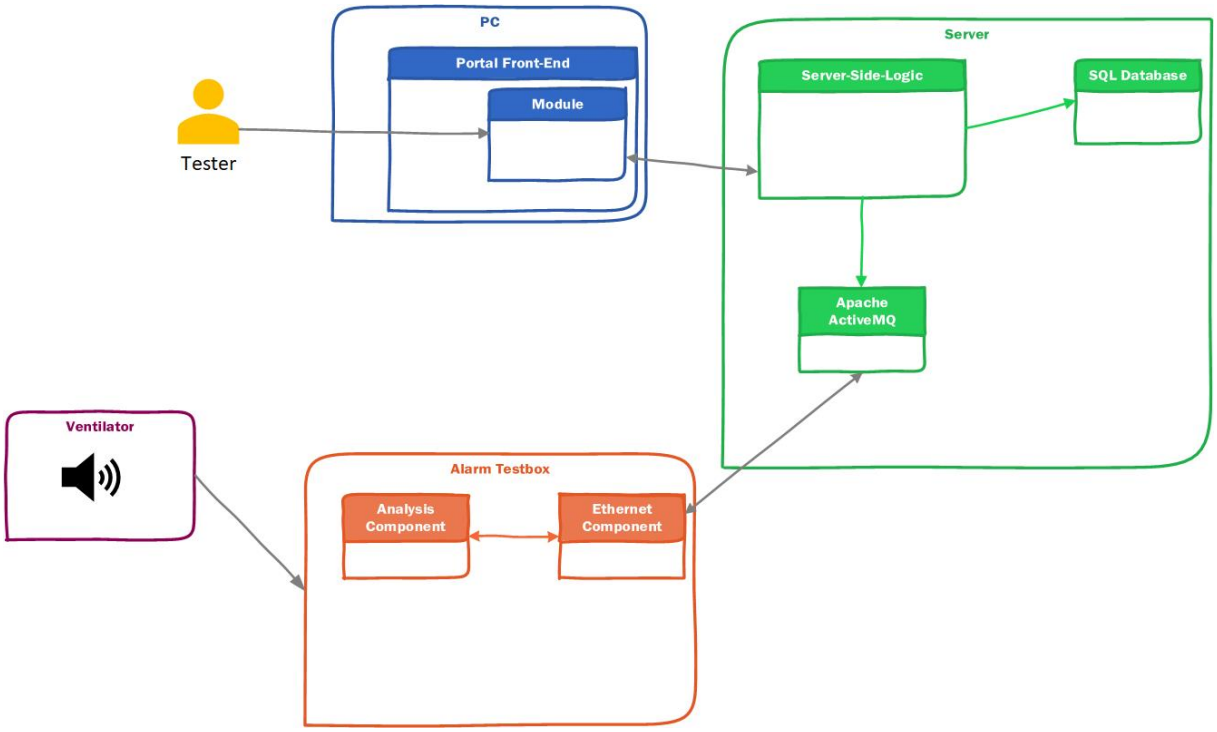

Figure 4.3: Hardware and Software Components

4.4 Requirements

This section explains the system requirements for the testbox. The interface requirements for the testbox interface are explained in the subsequent chapter.

The following functional and non-functional requirements are considered for the prototypical implementation of the testbox:

R1) Real-time Audio Processing – To ensure instant analysis of incoming audio streams, the proposed solution must be able to process audio inputs in real-time with no noticeable latency.

R2) Modularity – Design the solution with a modular architecture. This makes it easy to integrate testing for new types of alarms from different ventilators. This requirement makes it easier to add more functionality for future enhancements.

R3) Performance Optimization – The solution ensure optimal performance, even with constant audio input, by implementing efficient algorithms and using parallel processing techniques.

R4) Low Resource Consumption – The solution ensures that the software operates with minimal resource consumption, including CPU and memory usage, to avoid performance issues on the device running the solution.

R5) Error Handling and Logging – Implement robust error handling and logging capabilities to track and report any problems encountered during the audio analysis process, ensuring reliability and troubleshooting capabilities.

R6) IEC 60601-1-8 Standard – The proposed solution should be capable of verifying and validating three audible alarms, in accordance with the international standard.

These requirements will guide the implementation of the testbox, as well as serving as an objective to evaluate the solution. The necessity for these requirements is demonstrated in order to guarantee that the prototype performs the essential tasks correctly and meets the user needs.

Chapter 5

Design and Implementation

The objective of this chapter is to provide an understanding of the development of the prototype of the proposed sound alarm system according to the design and architecture explained in the previous chapter. The various software and hardware components are discussed in order to create the system, with its requirements discussed in the previous chapter. Finally, challenges that arose during the development cycle are discussed, along with the countermeasures that were imposed to address them.

5.1 Data Collection

The first step in the implementation was to record the different alarms in different settings. This was done using a Sandberg Streamer USB Clip Microphone. The ventilator used for the recordings was a Hamilton C1. A sampling frequency of 44.1kHz and a signal amplitude of 16 bit was chosen for the recordings. This was done to achieve a compromise between quality and file size. The alarms were manually triggered and recorded in a one-minute WAV container format file with the audio sample encoded in 16-bit Signed Integer PCM. Pulse code modulation (PCM) is a time-domain encoding of an audio waveform as a series of amplitudes [33]. A time-amplitude plot was employed to analyze the aforementioned one-minute files, and the individual alarms were subsequently transformed into a two-second WAVE container format file. The three alarms were recorded in different scenarios, with varying degrees of background noise as seen in Table 5.1. A Python script was utilized to record and store the files on a local machine in a designated folder using the SoundDevice [34] and Scipy [35] libraries.

Scenario	Quantity of Recordings
Quiet Environment	10
Speaking Background	10
Music Background	10

Table 5.1: Alarm Scenarios

5.2 Data Visualization

Once the data had been recorded, the subsequent step was to analyze it visually. This allowed for a more comprehensive and intuitive understanding of the information, including identifying patterns and trends. The WAV files were read and plotted in order to observe the amplitude of the flow of time, utilizing the Matplotlib [36] library in Python. A script was developed that accepts WAV files representing different alarm types as input and generates a visual representation of the time series as output. Figure 5.1 illustrates three distinct alarm types, which serve as examples for the different recordings. Subsequently, an FFT was applied to the audio files using the Scipy library, resulting in a spectral representation of the frequencies present and their respective magnitudes. This illustrates which frequencies are present during the alarm tone. This approach facilitated a more profound comprehension of the nature of an alarm in terms of data. Figure **??** depicts the spectral plot of three distinct alarms.

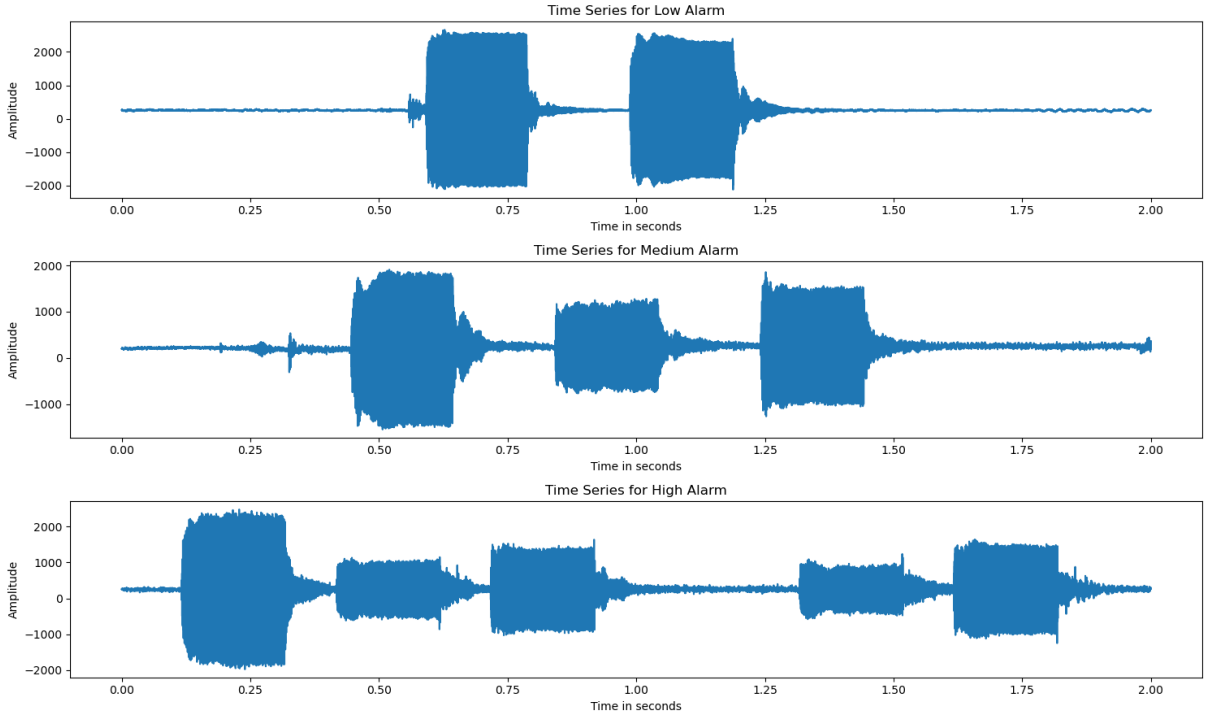

Figure 5.1: Timeseries of Alarms

5.3 Programming Language

The subsequent stage of the process involved identifying the most appropriate programming language. Given the substantial volume of data processed by the system, it was necessary to select a programming language that would provide a wide variety of tools and libraries. Given the plethora of available libraries and the intuitive syntax, the programming language Python was selected for the development of the alarm testing system [37].

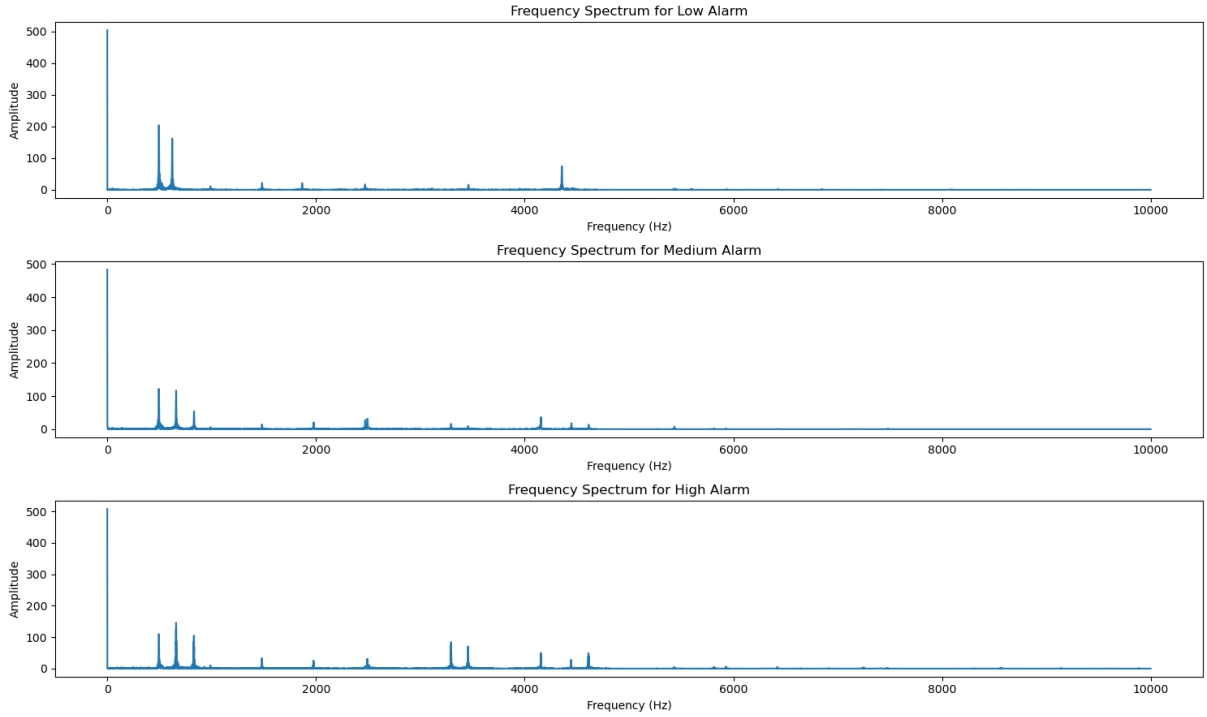

Figure 5.2: Spectral Properties of Alarms

5.4 Algorithmic Solution

The data collection and visualization processes constituted a fundamental basis for the implementation of the sequential solution. In the initial phase of implementing this version of the solution, the previously recorded audio files were utilized instead of real-time audio. This approach facilitated the manipulation of the data. The audio file was loaded into the script using 'load' function from the Librosa library [38], which loaded the file as a floating point time series. The time series was then split into segments and an FFT was performed on each of them. Here, several iterations were performed to find an appropriate segment length and to test which one was best suited for the alarms. In the end, a segment length of 0.006 seconds was chosen, meaning that one second of data had 265 different segments. The individual segments were then used by an algorithm , which evaluated the length of the sound and frequency to ascertain whether or not an alarm sounded. If an alarm was heard, the custom 'AlarmBuffer' data structure shown in 5.1 was filled. If the requirements for one of the alarms were met, an entry was written to a log file as shown in 5.2 .

```
from collections import deque

class AlarmBuffer:
    def __init__(self, max_length):
        self.buffer = deque(maxlen=max_length)

    def add_alarm(self, alarm):
        self.buffer.append(alarm)

    def clear_buffer(self):
```

```
        self.buffer.clear()

    def count_items(self):
        return len(self.buffer)
```

Listing 5.1: Alarm Buffer Data Structure

```

import logging

logging.basicConfig(filename='alarm_log.txt', level=logging.INFO, format
    ='%(asctime)s - %(message)s')

def write_to_log(message):
    logging.info(message)
```

Listing 5.2: Logging Alarms

With the non-real-time version successfully implemented, the next task was to refactor the code for testing incoming audio in real time. Instead of loading a pre-recorded audio file, the SoundDevice library is used to call a callback function every number of blocks of 1136 data points. These blocks are then analyzed in the same way as the non-real-time solution to see if an alarm tone is heard. This approach allowed the solution to record audio in real time, analyze it, and log the result.

5.5 CNN-Based Classifier Solution

A further solution was then implemented based on a convolutional neural network (CNN) using the Tensorflow [39] library. The initial objective was to create the training model, which entailed the development of a number of key components, including the input data, the training data set, and the training parameters. The input data were the prerecorded audio alarms. Subsequently, the data was concatenated into a single list and labeled to distinguish the different alarms as seen in 5.3. The data was then processed by converting the sample rate from 41 kHz to 16 kHz in order to reduce the volume of data. Additionally, it was ensured that all audio files had the same shape, with the length either being cut to two seconds or filled with zero values.

```
import tensorflow as tf

labels_high = tf.data.Dataset.from_tensor_slices(tf.fill([len(alarm_h)],
    0))  # Label '0' for High Alarm
labels_medium = tf.data.Dataset.from_tensor_slices(tf.fill([len(alarm_m)
    ], 1))  # Label '1' for Medium Alarms
labels_low = tf.data.Dataset.from_tensor_slices(tf.fill([len(alarm_l)],
    2))  # Label '2' for Low Alarms
label_none = tf.data.Dataset.from_tensor_slices(tf.fill([len(alarm_l)],
    3)) # Label '3' for No Alarms
dataset_high = tf.data.Dataset.zip((alarm_h, labels_high))
dataset_medium = tf.data.Dataset.zip((alarm_m, labels_medium))
dataset_low = tf.data.Dataset.zip((alarm_l, labels_low))
dataset_none = tf.data.Dataset.zip((alarm_l, labels_low))
```

```
11
12 full_dataset = dataset_high.concatenate(dataset_medium).concatenate(
       dataset_low) #List of all Data
```

Listing 5.3: Creation of Tensorflow Dataset

Once the data had been cleaned, a spectrogram was created with a specific frame length and frame step to transform the time domain signal into a time-frequency domain over time. This was achieved by computing the short-time Fourier transform (STFT) of the audio signal, with the absolute value calculated and an extra dimension added, which is required for training the model. The visual perception of spectrograms is exemplified in Figure 5.3 and the preprocessing function is seen in 5.5. Once the complete dataset has undergone the requisite preprocessing, it is prepared for the model through caching, shuffling, batching, and prefetching, as detailed in 5.5. The last step was to define the parameters for the training model 5.6.

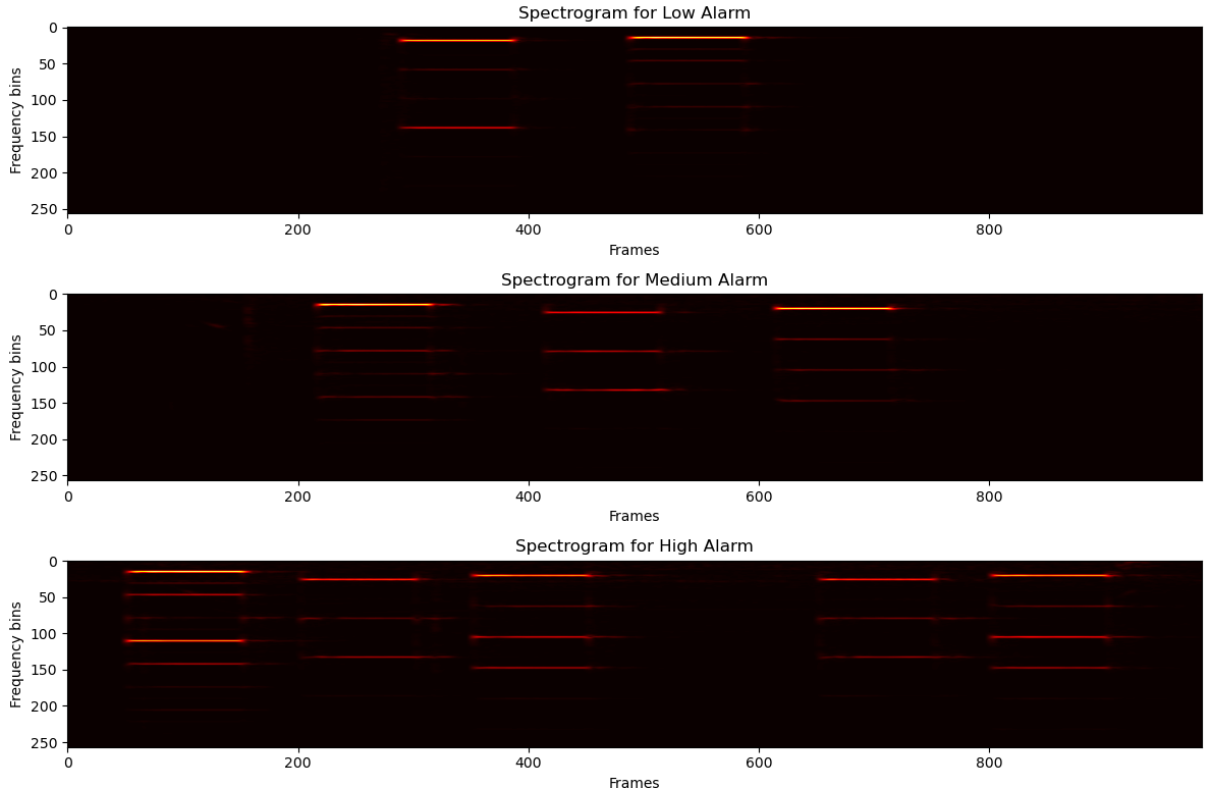

Figure 5.3: Spectrograms of Alarms

First the model was initialized as a 'Sequential' [40] model. The fundamental simplicity of the architectural design was the primary factor in selecting this model. This is because, in our case, one input should directly be matched to one output. This implies that the data flows in a unidirectional manner, traversing a single, linear pathway from the input layer to the output layer, without any branching, merging, or parallel pathways. The subsequent stage involved the addition of the layers to the CNN. A two-dimensional convolutional layer was incorporated to extract features from the input images, in this case, the spectrograms. The parameter filter of the initial layer was selected as 16, a common standard for capturing fundamental features such as edges and textures without

overwhelming computational resources. The filter size of 3x3 was selected as it is sufficiently small to permit the detection of fine details and patterns in the input image, while the computational load is relatively simple and efficient. The activation of the function 'relu' or rectified linear unit [41] introduces non-linearity into the model, which enables the model to learn more complex functions while maintaining its simplicity and achieving high performance. Following the initial layer, a second layer with identical parameters is incorporated, as stacking multiple layers allows the CNN to learn more complex features. The flattening of the layer is necessary because the resulting convolutional output is a 3D tensor [42], which must be flattened into a vector in order to be fed into the next layer. This process simplifies the complex structure into a simpler form. The next layer added is a dense layer [43], which is commonly used in CNN and has the advantage of being able to capture complex patterns and is well suited for image classification. The last layer in our case is the output layer with 4 final neurons corresponding to the 4 labels to be predicted. Activating the 'softmax' [44] parameter activates the function to output a probabilistic distribution over the classes. After the layer was added, the training phase began with the parameters shown in Figure 5. The chosen optimizer was the Adam class which is computationally efficient, has low memory requirements, and works on a wide range of problems with little need for hyperparameter tuning. Once the model had completed its learning cycle, it could be tested on the "test" dataset and saved for future use.

```

full_dataset = full_dataset.cache() #caches data_set in memory for
    quicker iterations
full_dataset = full_dataset.shuffle(buffer_size=1000) #randomizes the
    data, prevents the model of learning any unintended order
full_dataset = full_dataset.batch(8) #groups Data for more efficient
    processing, can be parallelized
full_dataset = full_dataset.prefetch(4) #4 batches will be prepared in
    advance for improved efficiency

# Splitting the dataset into 'training' sets and 'test' sets
train = full_dataset.take(X) #takes the first X batches for training
test = full_dataset.skip(X).take(1) #skips X batches and takes last for
    testing
```

Listing 5.4: Dataset Preperation for Model Training

```
import tensorflow as tf

def load_wav(file_path):
    file_contents = tf.io.read_file(filename) #reads the file
    tf_wav_file,sample_rate = tf.audio.decode_wav(file_contents) #
    decodes the raw WAV file and makes sure it it is converted to mono
    sample_rate = tf.cast(sample_rate,dtype=tf.int64) #required tf
    format sample rate
    tf_fwav_file= tfio.audio.resample(tf_wav_file, rate_in=sample_rate,
    rate_out=16000) #resampling the audio file from 41kHz to 16 kHz
    tf_wav_file = tf.squeeze(tf_wav_file, axis=-1) #removes unnecessary
    dimensions
    return tf_wav_file #return the Tensorflow Data Model

def preprocess_function(file_path):
    tf_wav_file = load_wav(file_path)
```

```
    tf_wav_file= tf_wav_file[:32000] #trim the file to make sure it is
    not more than 32000 samples long or 2 seconds
    zero_padding = tf.zeros([320000]-tf.shap(tf_wav_file, dtype =tf.
    float32)) #add zeroes if less than 32000 samples
    spectrogram = tf.signal.stft(tf_wav_file,frame_length=512,frame_step
    =32)  #STFT is applied with frame_len adn frame_step
    spectrogram = tf.abs(spectogram) #only magnituted is reained
    spectrogram = tf.expand_dims(spectogram, axis=2) #dimension added
    for compatibility with CNN
    return spectrogram
```

Listing 5.5: Preprocessing for CNN Model

```
alarm_model = Sequential()
alarm_model.add(Conv2D(16, (3, 3), activation='relu', input_shape=(985,
    257, 1)))
alarm_model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(4, activation='softmax'))
```

Listing 5.6: CNN Model Parameters

Like the first solution, this one needed to be real-time. So, the model was saved and loaded into another script. This script 5.7 takes the real-time data, segments it, and preprocesses the segment to make sure it fits the data the model was trained on. With this segment, the model makes a prediction of the spectrogram it took as input and writes its prediction to a log that is read by the interface.

```
stream = sd.InputStream(callback=callback, channels=1, samplerate=16000,
    blocksize=1000)
with stream:
    print("Starting stream...")
```

Listing 5.7: Real-Time Refactor

5.6 Interface

The next requirement is to connect the test equipment to a test system via an interface. The proposed solution is to make HTTPS requests through the Apache Active MQ message broker. The protocol suggests the use of MQTT because of its lightweight, low-bandwidth capabilities [45]. The interface parses the log generated by the test script and returns the solution. The Interface requirements are seen in Table 5.2. There is a request queue and a response queue. When a request is made, the test device receives a JSON [46] data. Figure 5.8 shows an example JSON request to check if the test device is connected, and Figure 5.9 shows a response to the request.

Furthermore, the interface may receive a request to test an alarm. This request includes a timestamp and an integer value, which indicates that the script will subsequently verify whether any alarms have been triggered in the last amount of seconds as indicated by the integer value. Subsequently, a response is transmitted to the tester indicating whether an

alarm has been triggered or not. To illustrate, consider the following example of a request and response, as depicted in Figure 5.10 and Figure 5.11 .

Resource	Action	Description	Request Body	Response
connection	GET	Check Connection	no body	204 No Content
high	GET	Test High Alarm	TimeDTO	200 OK + Body
medium	GET	Test Medium Alarm	TimeDTO	200 OK + Body
low	GET	Test Low Alarm	TimeDTO	200 OK + Body

Table 5.2: Test Device Interface Requirement

```
{
    "head": {
        "id":"00001234",
        "action":"GET",
        "resource":"/rest/api/v1/connect"
    }
}
```

Listing 5.8: Request Connection JSON

```
{
    "head": {
        "id": "00001234",
        "value": "204"
    }
}
```

Listing 5.9: Response Connection JSON

```
{
    "head": {
        "id": "00001235",
        "code":"GET",
        "resource":"/rest/api/v1/high"
        }
    },
   "body": {
       "timeDTO": {
            "current": "2024-05-16 08:56:21,416",
            "within":"5"
        }
     }
}
```

Listing 5.10: Request Alarm JSON

```
{
    "head": {
        "id": "00001235",
        "code":"200",
        "resource":"/rest/api/v1/high"
        }
    },
```

```
    "body": {
        "value": "true"
      }
}
```

Listing 5.11: Responset Alarm JSON

5.7 Troubleshooting

Several issues arose during the development process. One of the initial challenges was the necessity of ensuring that the test device operated in real time. The objective was to continuously record audio and simultaneously analyze everything heard. The issue was resolved by implementing a buffer that is continuously being filled with data, which is then periodically extracted.

Another issue pertained to the method of transmitting the data once the alarm had been successfully identified. As the test device was continuously recording audio and analyzing it, the solution involved writing a log with the pertinent information and the interface parsing the log to transmit the data, which was requested through the message broker. The log script is executing in a separate thread from the analysis script as seen in Figure 5.12.

```
{
import multiprocessing

def audio_analysis_script():
    exec(open('alarm_analysis.py').read())

def interface_script():
    exec(open('interface.py').read())

if __name__ == '__main__':
    # Create process for audio analysis script
    process_1 = multiprocessing.Process(target=audio_analysis_script)

    # Create process for interface script
    process_2 = multiprocessing.Process(target=interface_scrip)

    process_1.start()
    process_2.start()
```

Listing 5.12: Responset Alarm JSON

Another challenge was identifying the optimal model for the machine-learning solution. A variety of models were considered, including Hidden Markov Models, Support Vector Machines, and Transformers. All of these models have their respective advantages and disadvantages, and they are employed in different related papers, as evidenced in the chapter on related work. The CNN model was selected as the optimal choice due to its capacity to automatically extract pertinent features from raw input data, particularly in the context of alarm features. This capability renders it an effective tool for image classification, which is the process by which the audio alarm is transformed. Additionally, it exhibits high accuracy and is highly robust, making it suitable for real-world applications [47].

Once all obstacles had been overcome, the two distinct solutions were successfully implemented. The next step involved testing and evaluating them in accordance with the requirements stated in the previous chapter. This will be discussed in the next chapter.

Chapter 6

Evaluation

This chapter presents a comparative analysis of the test device through three case studies. The objective is to evaluate the accuracy and reliability of the testing device under different scenarios. After the comparison, the solutions are evaluated against the requirements from the Architecture chapter provided in this thesis.

6.1 Testing Scenarios

Four distinct testing scenarios are presented for the analysis of the efficacy of the test device. The first scenario presents a typical quiet test environment. There is no background noise and the alarms are the only sounds audible. The initial scenario should reflect a typical alarm test environment, with the sole focus on the alarm or sound to be analyzed. The second scenario is accompanied by background music. The second scenario is designed to test the robustness of the solution. The key question is how the data would change if background noise were introduced. In the third scenario, the presence of background noise is indicated by the fact that humans are speaking. Background music is a constant background noise, whereas speech is not. Therefore, this speech scenario was added because speech is not constant, but has pauses in between. The three scenarios are presented in Table 6.1.

Testing Scenarios
Quiet Scenario
Speaking Scenario
Music Scenario

Table 6.1: Scenarios

6.2 Test Environment

The testing environment was identical for both solutions. The first solution is an algorithmic solution that counts segments of specific frequencies and provides an output depending on the schema it follows. The second solution employs a classifier-based (CNN) model to predict which alarm is heard. Both solutions employ a Fourier Transformation to effectively perform a spectral analysis.

A Hamilton C1 was brought into the room, and a Sandberg Streamer USB Clip Microphone was affixed to the ventilator, as illustrated in Figure 6.1. The microphone was connected to a personal computer running the testing scripts. For all use cases, the three different alarms were manually triggered 100 times to ascertain whether the script provided the correct output. For the background sound and music scenario an UE Megaboom was placed 2 meters away from the microphone. This was done to simulate the real world scenario. Pop nusic or a podcast was then played via Bluetooth to simulate background noise at around 60-70 dB. To assess the effectiveness of the solution, the output log was evaluated.

Figure 6.1: The Testing Environment

6.3 Performance Metrics

When working with sound data, it can be challenging to accurately assess the quality of the measurements due to the variability in the test environment and the microphone's performance. In this thesis, a low-cost microphone was utilized, as also utilized in [20]. To assess the accuracy of the three different alarms, they were triggered manually 100 times per alarm per use case per solution. The performance of the single solution was determined by averaging the results of the three alarms across the three use cases. This was done by checking the log file for missing alarms and false alarms.

6.4 Results

First, the algorithmic solution was evaluated. After going through all the case studies for all the different alarms, the data showed the following output as shown in the Table 6.2. The data indicate that the low alarm is functioning effectively. In all test cases, only one alarm was not recognized. As the complexity of the sound alarm increases, the likelihood of error also rises. This is due to the fact that the alarm distinguishes between alarms by sharing similar features. Consequently, the solution encounters difficulties in distinguishing between the high and low alarms, particularly when more background noise is introduced. To further evaluate the data, the true positive rate, true negative rate, false positive rate, and false negative rate were calculated, as well as the f1 score. This was done in order to achieve a balance between precision and recall[48]. This representation is seen in Table 6.3

Table 6.2: Performance Algorithmic Solution

Alarm Type	Case Study	AT	AR	ANR	FA
Low Alarm	Quiet	100	100	0	0
Low Alarm	Music	100	100	0	0
Low Alarm	Speech	100	99	1	0
Medium Alarm	Quiet	100	98	2	0
Medium Alarm	Music	100	95	5	1
Medium Alarm	Speech	100	99	1	0
High Alarm	Quiet	100	90	10	5
High Alarm	Music	100	89	11	7
High Alarm	Speech	100	85	15	15

Alarms Triggered (AT), Alarms Recognized (AR),
Alarms Not Recognized (ANR), False Alarms (FA)

After testing the first solution, the second model-based solution was tested on the same scenarios. The data collected for the solution is shown in Figure 6.4. The data shows that the lower alarm is the alarm with the most detected alarms. The greater the number of background noise signals present in the environment, the less accurate the model's ability to predict the alarm. Furthermore, the data indicates that only approximately half of the high alarm conditions are being identified. To gain a more comprehensive understanding

Table 6.3: Metrics Algorithmic Solution

Alarm Type	Scenarios	TPR	FNR	FPR	TNR	F1
Low Alarm	Quiet	1	0	-	-	1
Low Alarm	Music	1	0	-	-	1
Low Alarm	Speech	0.99	0.01	-	-	0.995
Medium Alarm	Quiet	0.98	0.02	-	-	0.9899
Medium Alarm	Music	0.95	0.05	1	0	0.9694
Medium Alarm	Speech	0.99	0.01	-	-	0.995
High Alarm	Quiet	0.9	0.1	1	0	0.923
High Alarm	Music	0.89	0.11	1	0	0.9082
High Alarm	Speech	0.85	0.15	1	0	0.85
Total	Total	0.95	0.05	1	0	0.959

True Positive Rate (TPR), True Negative Rate (TNR),
False Positive Rate (FPR), False Negative Rate (FNR)

of the data, it was analyzed through a confusion matrix [49] in order to calculate the F1 score.

Table 6.4: Performance CNN Model Solution

Alarm Type	Case Study	AT	AR	ANR	FA
Low Alarm	Quiet	100	90	10	0
Low Alarm	Music	100	88	12	1
Low Alarm	Speech	100	87	13	5
Medium Alarm	Quiet	100	91	9	5
Medium Alarm	Music	100	80	20	6
Medium Alarm	Speech	100	78	22	8
High Alarm	Quiet	100	60	40	20
High Alarm	Music	100	50	50	24
High Alarm	Speech	100	45	55	27

Alarms Triggered (AT), Alarms Recognized (AR),
Alarms Not Recognized (ANR), False Alarms (FA)

6.4.1 Requirements

After testing the effectiveness and robustness of the solutions, the next step was to evaluate the solutions based on the requirements stated in Chapter 4 of this thesis. Both solutions were compared against the six requirements.

The initial requirement was that the solution should have real-time capabilities. This requirement has been met by both solutions, as sound is continuously being recorded and analysed in real time. The next requirement was to develop a modular system, which means dividing a system into separate modules or components to make it easier to implement possible future enhancements and to make it easier to change code without having to make changes to the entire system [50]. The solutions are constructed with a modular design, with each component serving a specific function. This design facilitates the

Table 6.5: Metrics CNN Model Solution

Alarm Type	Scenarios	TPR	FNR	FPR	TNR	F1
Low Alarm	Quiet	0.9	0.1	-	-	0.9
Low Alarm	Music	0.88	0.12	1	0	0.9312
Low Alarm	Speech	0.87	0.13	1	0	0.9062
Medium Alarm	Quiet	0.91	0.09	1	0	0.9286
Medium Alarm	Music	0.8	0.2	1	0	0.8602
Medium Alarm	Speech	0.78	0.22	1	0	0.8387
High Alarm	Quiet	0.6	0.4	1	0	0.6417
High Alarm	Music	0.5	0.5	1	0	0.5747
High Alarm	Speech	0.45	0.55	1	0	0.5233
Total	Total	0.7433	0.25667	1	0	0.8036

True Positive Rate (TPR), True Negative Rate (TNR),
False Positive Rate (FPR), False Negative Rate (FNR)

maintenance and scaling of the system in the event that such actions would be necessary in the future. The interface script is separated from the analyzing script, which enhances the overall stability and reliability of the system.

A performance test was conducted to assess the efficiency of the script. The Psutil [51] library was utilized for this purpose. The script was observed to consume between 65 and 70 megabytes of memory. The required memory for the CNN model solution is between 100 and 110 megabytes. These results indicate that the two scripts require minimal resources.

The subsequent requirement was for the device to implement error and logging capabilities. While both solution do log the alarms that are heard, it does not address the issue of robust error handling. This could be a topic for further investigation in a future iteration.

The final requirement was to test the international standard IEC 60601-1-8 to ascertain whether the solution in question complies with it. The algorithmic solution successfully fulfills the testing requirements for the low alarm, as it does not require the alarm to be repeated. However, the medium and high alarms are required to be repeated at certain intervals, which are not tested. This could also be a task for a future addition of this feature. In contrast, the CNN model is more suited to predicting alarms than to testing the attributes of the alarm itself. Table 6.6 provides an overview of the requirements associated with the two solutions.

Requirement	Pass/Fail Solution Algorithmic	Pass/Fail Solution CNN Model
R1	PASS	PASS
R2	PASS	PASS
R3	PASS	PASS
R4	PASS	PASS
R5	FAIL	FAIL
R6	Partial PASS	FAIL

Table 6.6: Requirement Algorithmic Solution

6.5 Interpretation and Discussion of the Results

The data clearly shows that the algorithmic solution without using a model is better at detecting alarms and is a robuster solution. The CNN model could be trained with more data and go through more layers to improve its alarm detection capabilities.

Both solutions are capable of handling less complex low-priority alarms with greater efficacy. However, as the complexity of the background is increased, the accuracy of the solution is also reduced.

In order to evaluate both solutions, the F1 score was deemed to be an invaluable tool for providing a balanced and realistic evaluation of a model's performance. Given that the algorithmic solution achieved an F1 score of 0.95 and the CNN model an F1 score of 0.80, the comparison leads to the conclusions that the algorithmic solution outperforms the CNN model.

Chapter 7

Conclusion and Future Work

The objective of this thesis was to develop a testing apparatus for the evaluation of sound alarms on medical equipment. Two solutions were then presented and compared to each other. One solution entailed an algorithmic proposal, while the other involved a machine learning approach using a CNN model.

The initial chapter of this thesis outlined the underlying motivation for the project. It then explained the work that had to be accomplished and the proposed methodology for achieving this goal. In the second chapter, a theoretical foundation was established for the analysis of sound data and the necessary tools to accomplish this task. Additionally, the characteristics of ventilators and the three alarm types to be tested were described.

Chapter 3 then proceeded to conduct a comprehensive examination of the current scientific state of research and the methodologies employed in the creation of systems that are capable of detecting specific sound tones or alarms. Several overarching tools were highlighted as being of particular importance, including the use of Fourier transformation and the use of neural networks. The related work papers could be split into two sections: on the one hand, model-based solutions and, on the other, non-model-based solutions. This chapter laid the foundation for the development of the proposed solutions' architecture.

Chapter 4 explained the architecture of the two proposed solutions. It first described the sound analysis component of the two solutions. It also presented a proposal of how the entire software and hardware architecture could look like for the test system. At the end of the chapter functional and non-functional requirements were introduced.

In Chapter 5, there was a change from theory to practical application by explaining the implementation of the 2 solutions were explained along with the motivation of how certain challenges were faced and how they were countered. In this chapter, first the algorithmic non-ML solution was explained and then a solution using a CNN model was explained in detail. Both solutions were implemented in a first iteration using pre-recorded data and then refactored to work with real-time data.

Chapter 6 evaluates both solutions using three different case studies and pre-defined requirements. The data showed that both solutions were able to detect alarms, but the

solution using the CNN model had an accuracy of only 75%, while the algorithmic solution had an accuracy of 95%. The data indicated that the algorithmic solution achieved an F1 score of 0.95, while the CNN model attained an F1 score of 0.80.

7.1 Future Work

The CNN model of this thesis was trained with a relatively small set of training data and a limited number of layers. It would be of interest to train the model with more data and add layers with specific hyper-parameters to ascertain whether the accuracy of the testing device would increase.

The experiment was conducted and the solutions were implemented with the use of a single microphone, which resulted in a mono recording. Additionally, a low-budget microphone was utilized in this project. It would be beneficial to employ multiple microphones to record stereo sound and utilize a high-end budget microphone to conduct the testing for more data.

This thesis focused on the detection and testing of sound alarms for ventilators. However, the field of alarm systems is much broader than that. It would be of interest to apply the models and solutions developed in this thesis to other areas of alarm in the healthcare field, such as patient monitors, infusion pumps, cardiac monitors, or anesthesia machines.

Bibliography

[1] A. Park, *Philips recalls handful of hospital ventilators equipped with substandard power circuits*, https://www.fiercebiotech.com/medtech/philips-recalls-handful-ventilators-equipped-substandard-power-management-circuits Last Visit: 10.01.2024.

[2] J. Watkinson, *Introduction to Digital Audio 2nd Edition*. Focal Press, 2002.

[3] A. Zola, *Definition sound wave*, https://www.techtarget.com/whatis/definition/sound-wave Last Visit: 10.01.2024.

[4] -, *Analog-to-digital converters basics*, https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters Last Visit: 10.04.2024.

[5] Adobe, *Sample rates and audio sampling: A guide for beginners*, https://www.adobe.com/uk/creativecloud/video/discover/audio-sampling.html Last Visit: 15.01.2024.

[6] D. Lavry, *Sampling theory for digital audio*, https://lavryengineering.com/pdfs/lavry-sampling-theory.pdf Last Visit: 20.01.2024.

[7] -, *Quantizing*, https://cmtext.indiana.edu/digital_audio/chapter5_quantize.php Last Visit: 11.04.2024.

[8] -, *Digital audio basics: Audio sample rate and bit depth*, https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth Last Visit: 11.04.2024.

[9] "Copyright", in *The Essential Guide to Image Processing*, A. Bovik, Ed., Boston: Academic Press, 2009, p. iv, ISBN: 978-0-12-374457-9. DOI: https://doi.org/10.1016/B978-0-12-374457-9.00033-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123744579000330.

[10] -, *Term: Waveform (sound)*, https://www.digitizationguidelines.gov/term.php?term=waveformsound Last Visit: 11.04.2024.

[11] -, *Practical introduction to frequency-domain analysis*, https://ch.mathworks.com/help/signal/ug/practical-introduction-to-frequency-domain-analysis.html Last Visit: 15.04.2024.

[12] J. V. Kamp, *Philips recalls handful of hospital ventilators equipped with substandard power circuits*, https://www.fiercebiotech.com/medtech/philips-recalls-handful-ventilators-equipped-substandard-power-management-circuits Last Visit: 26.04.2024.

[13] F. Blateyron, *What is spectral analysis?*, https://www.digitalsurf.com/blog/what-is-spectral-analysis/s Last Visit: 30.04.2024.

[14] -, *Fourier transforms*, https://www.thefouriertransform.com/ Last Visit: 27.04.2024.

[15] -, *Fast fourier transformation fft - basics*, https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft Last Visit: 12.04.2024.

[16] S. E. Cleve Moler, *Faster finite fourier transforms matlab*, https://ch.mathworks.com/company/technical-articles/faster-finite-fourier-transforms-matlab.html Last Visit: 11.04.2024.

[17] S. Ramakrishnan, *Introductory chapter: Wavelet theory and modern applications*, https://www.intechopen.com/chapters/1181336 Last Visit: 1.05.2024.

[18] -, *What is a ventilator?*, https://www.nhlbi.nih.gov/health/ventilator Last Visit: 16.04.2024.

[19] D. P. W. Ellis, "Detecting alarm sounds", 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:642215.

[20] F. Beritelli, S. Casale, A. Russo, and S. Serrano, "An automatic emergency signal recognition system for the hearing impaired", in *2006 IEEE 12th Digital Signal Processing Workshop and 4th IEEE Signal Processing Education Workshop*, 2006, pp. 179–182. DOI: 10.1109/DSPWS.2006.265438.

[21] -, *Wave pcm soundfile format*, http://soundfile.sapp.org/doc/WaveFormat/ Last Visit: 15.04.2024.

[22] R. Kalra, *The difference between model-based and model-free reinforcement learning*, https://medium.com/@kalra.rakshit/the-difference-between-model-based-and-model-free-reinforcement-learning-9499af3770db Last Visit: 27.04.2024.

[23] R. A. Lutfi and I. Heo, "Automated detection of alarm sounds", *The Journal of the Acoustical Society of America*, vol. 132, no. 2, EL125–EL128, Jul. 2012, ISSN: 0001-4966. DOI: 10.1121/1.4734555. eprint: https://pubs.aip.org/asa/jasa/article-pdf/132/2/EL125/15303724/el125_1_online.pdf. [Online]. Available: https://doi.org/10.1121/1.4734555.

[24] X. Xiao, H. Yao, and C. Guo, "Automatic detection of alarm sounds in cockpit voice recordings", in *2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009)*, 2009, pp. 599–602. DOI: 10.1109/CASE.2009.88.

[25] M.-A. Carbonneau, N. Lezzoum, J. Voix, and G. Gagnon, "Detection of alarms and warning signals on an digital in-ear device", *International Journal of Industrial Ergonomics*, vol. 43, no. 6, pp. 503–511, 2013, Noise: Assessment and Control, ISSN: 0169-8141. DOI: https://doi.org/10.1016/j.ergon.2012.07.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169814112000625.

[26] J. Schröder, S. Goetze, V. Grützmacher, and J. Anemüller, "Automatic acoustic siren detection in traffic noise by part-based models", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 493–497. DOI: 10.1109/ICASSP.2013.6637696.

[27] G. Raboshchuk, C. Nadeu, P. Jančovič, *et al.*, "A knowledge-based approach to automatic detection of equipment alarm sounds in a neonatal intensive care unit environment", *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, pp. 1–10, 2018. DOI: 10.1109/JTEHM.2017.2781224.

[28] S. Spagnol, T. G. Goos, I. Reiss, and E. Özcan, "An algorithm for automatic acoustic alarm recognition in the neonatal intensive care unit", in *2022 7th International Conference on Frontiers of Signal Processing (ICFSP)*, 2022, pp. 59–63. DOI: 10.1109/ICFSP55781.2022.9924684.

[29] Y. A. Amrulloh and L. M. H. Maulidin, "Spectral analysis of abnormal breath sounds in childhood pneumonia", in *2018 International Symposium on Electronics and Smart Devices (ISESD)*, 2018, pp. 1–5. DOI: 10.1109/ISESD.2018.8605486.

[30] R. Phettom, N. Theera-Umpon, and S. Auephanwiriyakul, "Automatic identification of abnormal lung sounds using time-frequency analysis and convolutional neural network", in *2023 15th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2023, pp. 1–6. DOI: 10.1109/ICITEE59582.2023.10317776.

[31] A. Ekiz and K. Kaplan, "Covid-19 detection from cough, breath, and speech sounds with short-time fourier transform and a cnn model", in *2023 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2023, pp. 1–5. DOI: 10.1109/ASYU58738.2023.10296675.

[32] A. Gaiţă, G. Nicolae, A. Rădoi, and C. Burileanu, "Chainsaw sound detection based on spectral haar coeffluents", in *2018 International Symposium ELMAR*, 2018, pp. 139–142. DOI: 10.23919/ELMAR.2018.8534594.

[33] -, *Pcm*, https://wiki.multimedia.cx/index.php/PCM Last Visit: 5.05.2024.

[34] -, *Play and record sound with python*, https://python-sounddevice.readthedocs.io/en/0.4.6/ Last Visit: 7.05.2024.

[35] -, *Input and output (scipy.io)*, https://docs.scipy.org/doc/scipy/reference/io.html Last Visit: 7.05.2024.

[36] -, *Matplotlib (3.8.4)*, https://pypi.org/project/matplotlib/ Last Visit: 8.05.2024.

[37] C. Ginsberg, *Top 5 programming languages for data analysts*, https://www.nobledesktop.com/classes-near-me/blog/top-programming-languages-for-data-analysts Last Visit: 15.04.2024.

[38] -, *Librosa load*, https://librosa.org/doc/main/generated/librosa.load.html Last Visit: 18.04.2024.

[39] -, *Tensorflow*, https://www.tensorflow.org/ Last Visit: 8.05.2024.

[40] -, *Sequential model*, https://keras.io/guides/sequential_model/ Last Visit: 8.05.2024.

[41] B. Krishnamurthy, *An introduction to the relu activation function*, https://builtin.com/machine-learning/relu-activation-function Last Visit: 10.05.2024.

[42] -, *What is a tensor?*, https://www.doitpoms.ac.uk/tlplib/tensors/what_is_tensor.php Last Visit: 10.05.2024.

[43] Baeldung, *The concepts of dense and sparse in the context of neural networks*, https://www.baeldung.com/cs/neural-networks-dense-sparse Last Visit: 10.05.2024.

[44] S. Saxena, *Introduction to softmax for neural networks*, https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/ Last Visit: 10.05.2024.

[45] -, *What is mqtt?*, https://aws.amazon.com/what-is/mqtt/ Last Visit: 20.04.2024.

[46] -, *Introducing json*, https://www.json.org/json-en.html Last Visit: 10.05.2024.

[47] -, *Tamanna*, https://medium.com/@tam.tamanna18/exploring-convolutional-neural-networks-architecture-steps-use-cases-and-pros-and-cons-b0d3b7d46c71 Last Visit: 25.04.2024.

[48] R. Kundu, *F1 score in machine learning: Intro and calculation*, https://www.v7labs.com/blog/f1-score-guide Last Visit: 11.05.2024.

[49] S. Narkhede, *Understanding confusion matrix*, https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 Last Visit: 12.05.2024.

[50] -, *Introducing json*, https://www.springboottutorial.com/modularity-non-functional-requirement-in-microservices Last Visit: 10.05.2024.

[51] -, *Psutil 5.9.8*, https://pypi.org/project/psutil/ Last Visit: 12.05.2024.

Abbreviations

ANN	Artificial neural network
ALG	Algorithmic
CNN	Convolutional Neural Network
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
HTK	Hidden Markov Model Toolkit
HMM	Hidden Markov Model
PCM	Pulse Code Modulation
PSD	Power Spectral Density
PNA	Power of the Normalized Auto-correlation Function
PBM	Part-Based Models
STFT	Short-Time Fourier Transform
RMS	Root Mean Square
WAV	Waveform Audio File Format

List of Figures

List of Tables

Appendix A

Contents of the CD

1. This thesis as PDF
2. This thesis as LATEX source in a .zip file
3. Midterm presentation slides as PDF
4. The source code of this thesis
5. The datasets used for the model, in a directory called datasets