People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Ibn Khaldoun Tiaret
Faculty of Applied Science
Department of Electrical engineering

Development and design of an intelligent system for poultry diseases prediction

Dissertation Submitted in Partial Fulfillment for the Requirement of the Master Degree in Automatic and Industrial computing

Submitted by :

Mohamed Amine HAMAME

Supervised by :

Pr. Tayeb ALLAOUI

Board of Examiners:

Chairman: Dr. Hamid BOUMEDIENNE
Examiners: Dr. Houari BENABID
Dr. Amine BENZERROUK
Dr. Mohamed BEY

Academic year

2022/2023

Dedication

*First and foremost, my gratitude is owed to **ALLAH**, lord of the world,for giving me health and energy to accomplish this humble work, and to whom I should owe everything.*

My deepest hearty gratitude goes to my parents - my precious mother may Allah keep her safe and my dear father Henni may Allah bless him. Also, my special thanks to my Quran's teacher Chaib Benchaib - for educating me and for their unconditional support during all the stages of my life.

My heartfelt thanks are addressed to my dear brothers, my sister, my friends and precious EUREKA club members who supported me along the past five years of my studies journey.

*To all those mentioned, I would say a hearty **"Thank you"***

Acknowledgment

I particularly want tothank my supervisor Pr. ALLAOUI Tayeb for his supervision and advices during the implementation of this project.

I sincerely thank the members of the jury for having accepted to be part of the committee of examiners.

I would also like to express our thanks to all our teachers, who have contributed to our training throughout our years of study.

Our thanks are also addressed to the members of the laboratory of energy engineering and computer engineering-L2GEGI-, University of TIARET.

All appreciations go to examiners for their time to correct our work. My special thanks go to Dr. Mohamed BEY , Dr. Houari BENABID, Dr. Amine BENZERROUK and Dr. Hamid BOUMEDIENNE.

I will not forget my gratitude to all my previous teachers.

I would like in my turn to acknowledge warmly all the many people near and far who contributed to our work: teachers, colleagues, participants of our research, my colleagues at workplace whomI should be thankful for his training and support.

TABEL OF CONTENT

TABEL OF CONTENT

ABBREVIATIONS

ML : Machine Learning.
DL : Deep Learning.
AI : Artificial Intelligence.
SVM : Support Vector Machine.
CNN : Convolutional Neural Network.
RNN : Recurrent Neural Network.
MFCC : Mel Frequency Cepstral Coefficients.
FFT : Fast Fourier Transform.
RMS : Root Mean Square.
STFT : Short-Time Fourier Transform.
LPC : Linear Predictive Coding.
HMM : Hidden Markov Model.
PCA : Principal Component Analysis.
GMM : Gaussian Mixture Model.
LDA : Linear Discriminant Analysis
ANN : Artificial Neural Network
DNN : Deep Neural Network
LSTM : Long Short-Term Memory
GRU : Gated Recurrent Unit
F1-score - F1 Measure or F-Score
TP : True Positive
TN : True Negative
FP : False Positive
FN : False Negative
PR : Precision-Recall
EER : Equal Error Rate
ROC : Rate of Change
AR : Autoregressive
DCT : Discrete Cosine Transform
LBP : Local Binary Patterns
NMF : Non-Negative Matrix Factorization
ROI : Region of Interest
VAD : Voice Activity Detection
PSD : Power Spectral Density

List of figures

General Introduction

General Introduction

Artificial intelligence (AI) has emerged as a powerful tool for solving complex problems in various domains, and animal husbandry is no exception. With the emergence of new and complex diseases, the need for effective disease management and control in animal husbandry has become increasingly critical. In recent years, there has been a growing interest in the development of intelligent systems that can aid in predicting the occurrence of animal diseases using AI and signal processing techniques [1].

Animal diseases have a significant impact on animal health and welfare, as well as the productivity and profitability of animal farming. Early detection and diagnosis of diseases are crucial for effective disease management and control. However, traditional methods of disease detection and diagnosis, such as physical examinations and laboratory tests, can be time-consuming, expensive, and invasive. Moreover, these methods may not be suitable for monitoring large populations of animals in real-time [2].

AI has the potential to revolutionize disease management and control in animal husbandry. AI techniques, such as image analysis, natural language processing, and sound classification, can analyze large volumes of data quickly and accurately, enabling early detection and diagnosis of diseases. Sound classification, in particular, has gained increasing attention as a non-invasive and cost-effective method for detecting and monitoring animal health. Sound classification techniques can analyze audio recordings of animal vocalizations to identify changes in pitch, frequency, or other acoustic features that may indicate the presence of disease [3].

Signal processing is an important component of intelligent systems for animal disease prediction. Sound classification, in particular, requires a combination of signal processing, feature extraction, and machine learning algorithms to analyze audio recordings of animal vocalizations. Signal processing techniques, such as filtering, feature extraction, and pattern recognition, can be used to preprocess the data and extract relevant features for classification.

In this dissertation, we present a novel system for predicting poultry diseases based on AI and signal processing techniques. The system is designed to analyze audio recordings of poultry vocalizations to detect abnormal sounds that may indicate the presence of disease. The system uses a combination of signal processing, feature extraction, and machine learning algorithms to predict the likelihood of a disease outbreak. The system's performance is

evaluated using data collected from a poultry farm, and the results demonstrate the potential of the system for predicting and managing poultry diseases.

The dissertation is organized as follows: Chapter 1 provides an overview of the importance of AI in animal husbandry, with a focus on disease management and control. Chapter 2 discusses the importance of signal processing in sound classification, highlighting its potential applications in animal health monitoring. Finally, Chapter 3 presents the conception of the system using AI and signal processing techniques, including data preprocessing, feature extraction, and machine learning algorithms. The system's performance is evaluated using data collected from a poultry farm, and the results demonstrate the potential of the system for predicting and managing poultry diseases.

Overall, the development of an intelligent system for predicting poultry diseases based on AI and signal processing techniques can significantly improve disease management and control in the poultry industry, leading to increased productivity and profitability, as well as improved animal welfare.

CHAPTER ONE

ARTIFICIAL INTELLIGENCE AND ANIMAL

CHAPTER ONE : ARTIFICIAL INTELLIGENCE AND ANIMAL HUSBANDRY

Introduction

The term "artificial intelligence was coined more than 60 years ago, but only recently have we realized the full benefits of artificial intelligence, machine learning, and deep learning in our daily lives [4].

Most of us already use intelligent machines to learn, recognize sounds, make decisions, solve problems, and make recommendations on everything from the routes we drive to the movies we watch to the clothes we buy. We have smartphones in our pockets, intelligent personal assistants on our workbenches, robots in our factories and self-driving cars on our highways. This is just for starters.

Artificial intelligence, Deep Learning / Machine Learning Systems are having a major impact on the aerospace industry, too. With the technologies mentioned above, flying is becoming safer, more comfortable, and more predictive and outcome based. Airlines improve schedule performance, use less fuel and create a better passenger experience. Airports are more efficient and easier for travelers to navigate. Ground crews turn flights around faster and dispatch operations are getting more efficient and autonomy based. Airlines are able to use the learning systems to derive better segment strategies and charge according to the relevance and value. And aircraft maintenance is easier, faster, prescriptive and more precise. [1].

Like many other industries and use cases, AI in animal husbandry can improve environmental management, increase the animal's quality of life, improve resource allocation, and reduce costs. Farmers in animal husbandry are looking for new-age tools to improve animal welfare, increase efficiency, and create better production. [3].

AI can also be used in poultry farms, where poultry drones can detect nutritional deficiencies and mitigate bird diseases. Robots can help feed birds, collect eggs and remove droppings. AI-driven sound recognition systems, as what we are trying to do on this research, use machine learning to decipher vocalizations and identify warning signs of distress. AI monitoring can also detect patterns of poultry weight gain, allowing farmers to pinpoint unhealthy livestock. [5].

Finally we can say that AI helps humanity to progress and develop more and more, it enables businesses, governments, and communities to create a high-performing ecosystem

that can service the entire planet. Its significant impact on human lives is resolving some of society's most pressing issues.

1.1 AI Definition

The prospect of creating intelligent computers has fascinated many people for as long as

computers have been around and, as we shall see in the historic overview, the first hints in the direction of Artificial Intelligence date even before that. But what do we mean by Artificial Intelligence, if even the term intelligence itself is difficult to define?

The precise definition and meaning of the word intelligence, and even more so of Artificial Intelligence, is the subject of much discussion and has caused a lot of confusion [6]. One dictionary alone, for example, gives four definitions of Artificial Intelligence:

An area of study in the field of computer science. Artificial intelligence is concerned with the development of computers able to engage in human-like thought processes such as learning, reasoning, and self-correction.

- The concept that machines can be improved to assume some capabilities normally thought to be like human intelligence such as learning, adapting, self-correction, etc [6].
- The extension of human intelligence through the use of computers, as in times past physical power was extended through the use of mechanical tools.
- In a restricted sense, the study of techniques to use computers more effectively by improved programming techniques. [6]

The definitions have also changed in the course of time, due to the rapid developments. Definitions that are more recent speak of “imitating intelligent human behavior,” which is already a much stronger definition.

For some time now, the Artificial Intelligence community has been trying to imitate intelligent behavior with computer programs. This is not an easy task because a computer program must be able to do many different things in order to be called intelligent.

- Instead of looking at a general definition of Artificial Intelligence, one can also restrict

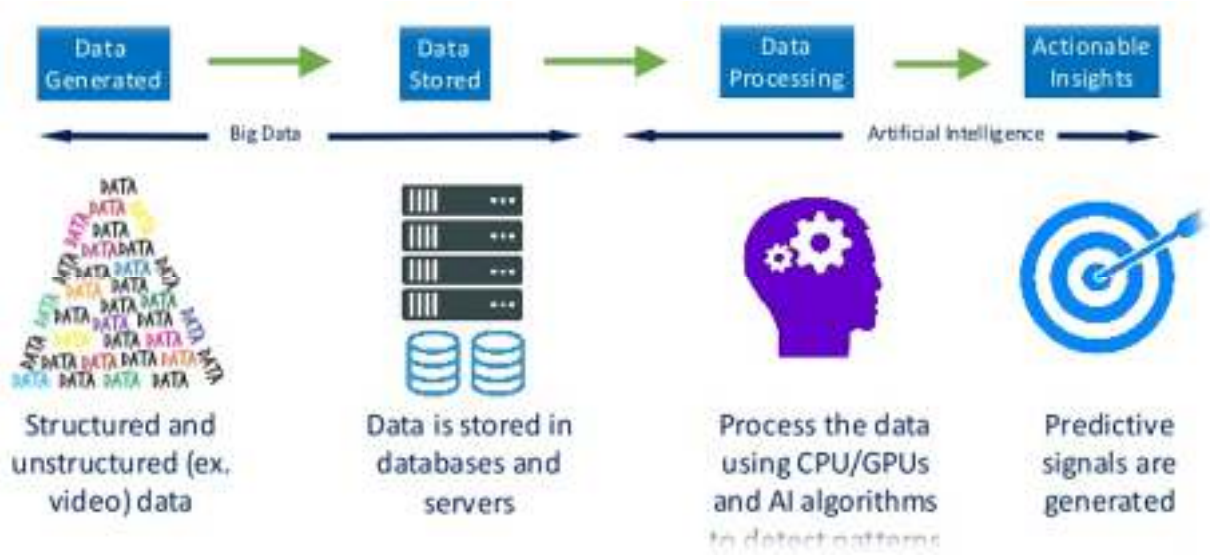
  oneself to the definition of artificially intelligent systems. There are many definitions around, but most of them can be classified into the following four categories [7]: Systems that think like humans.
- Systems that act like humans.
- Systems that think rationally.
- Systems that act rationally.

1.2 Artificial Intelligence Process

Artificial Intelligence (AI) is a broad field that encompasses various processes and techniques for building intelligent machines capable of performing tasks that typically require human intelligence. Here is the general process AI [8]:

1. Define the Problem: The first step in building an AI system is to define the problem you want to solve. This involves understanding the requirements of the problem and determining the feasibility of building an AI solution [8]:.
2. Gather Data: AI systems require large amounts of data to learn and make decisions. You will need to gather data relevant to the problem you are trying to solve. This data can come from a variety of sources, such as sensors, databases, and external APIs.
3. Preprocess Data: Once you have gathered the data, you need to preprocess it to make it suitable for use in an AI system. This can involve cleaning, transforming, and normalizing the data [8]:.
4. Choose Algorithms: Next, you need to choose the algorithms you will use to train your AI model. The choice of algorithms depends on the type of problem you are trying to solve and the data you have available [8]:.
5. Train Model: You will use the data and algorithms to train your AI model. This involves feeding the data into the model and adjusting the parameters of the algorithms until the model accurately predicts the outcomes [8]:.
6. Evaluate Model: Once we have trained our model, we need to evaluate its performance on new data. This involves testing the model on a separate set of data that it has not seen before [8].

7. Deploy Model: If the model performs well on the test data, you can deploy it into production. This involves integrating it into your existing systems and making it available for use.



FigureI. 01 : Artificial intelligence Process [9]

1.4 Artificial intelligent subfields

AI encompasses numerous subfields, each representing a distinct area of scientific research. Among them, six major subfields can be identified [10]:

1. Machine Learning: Algorithms and models enabling machines to learn from data [10].

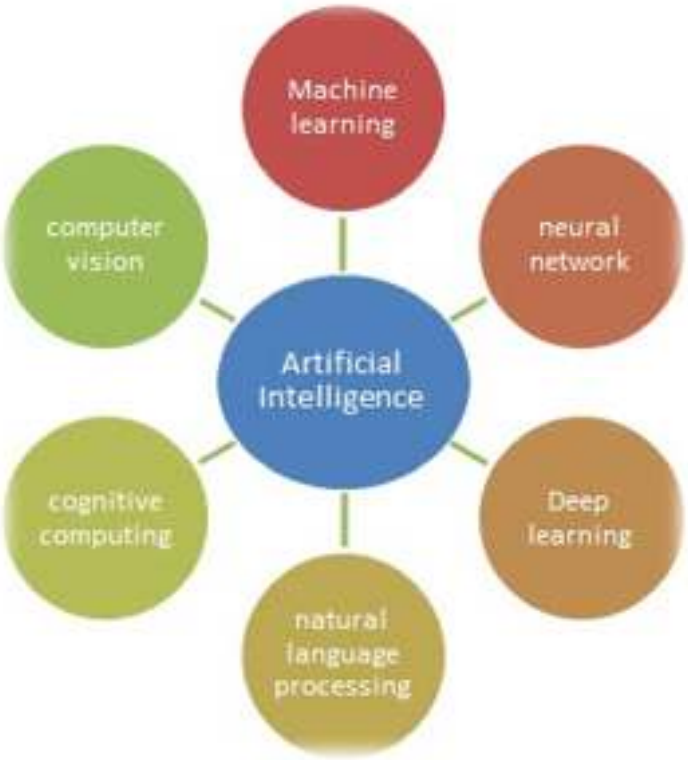
2. Natural Language Processing (NLP): Interaction between computers and human language [10].

3. Computer Vision: Understanding and interpreting visual information [10].

4. Neural network: Computational models inspired by the brain that learns from data and makes predictions or decisions. Combining AI with mechanical engineering for intelligent machines [10].

5. Deep learning: Machine learning technique that uses multi-layered neural networks to learn complex patterns from data [10].

6. Cognitive computing: Combines artificial intelligence and data analytics to simulate human cognitive processes and enhance decision-making [10].



FigureI. 02 : Artificial intelligence Subfildes [11].

1.4.1 Machine Learning

Machine learning (ML) is a field devoted to understanding and building methods that let machines "learn" – that is, methods that leverage data to improve computer performance on some set of tasks [12].

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so [13]

Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks [13].

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers, but not all machine learning is statistical learning.

The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning [13].

Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain.

In its application across business problems, machine learning is also referred to as predictive analytics.

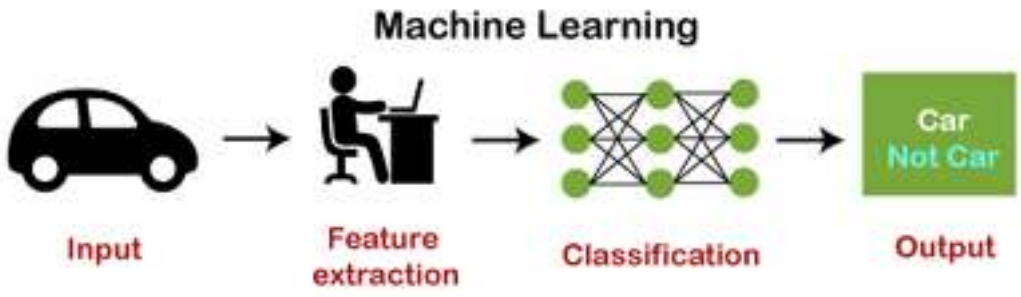


Figure I. **03: Machin Learning Process** [14].

1.4.2. Deep Learning

Deep learning is part of a broader family of machine learning methods, which is based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised [15].

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, convolutional neural networks and transformers have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance [15].

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, artificial neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog [15].

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a non-polynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation that is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability [15].

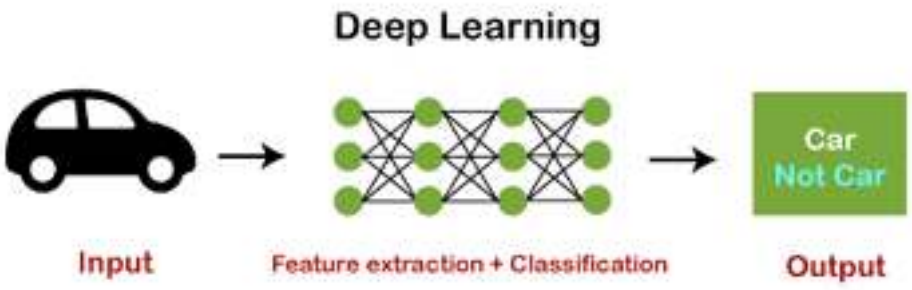


Figure I. 04: Deep Learning Process [14]

1.4.3. Deference between machine learning and deep learning

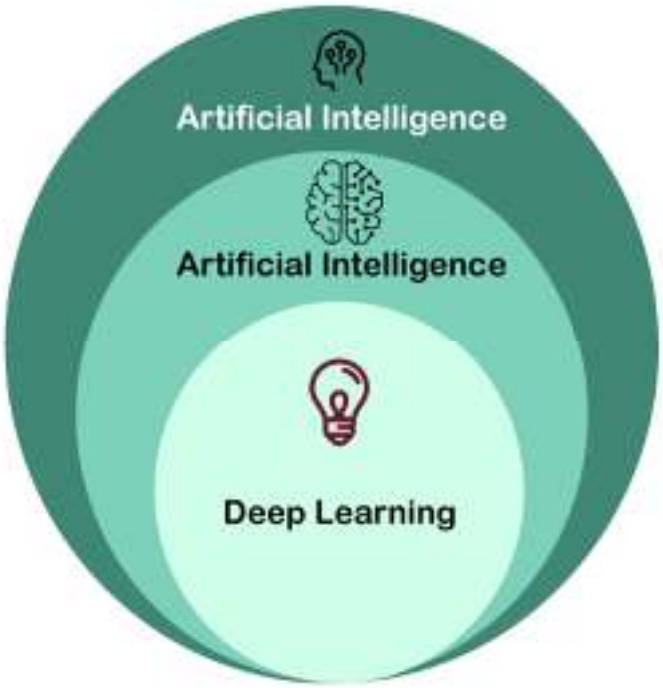


Figure I. **05: Deep and Machine learning** [16].

Machine learning and deep learning are both subsets of artificial intelligence, but they differ in their approach and complexity. Here are some of the key differences between machine learning and deep learning [17].

1. Architecture: Machine learning models are generally simpler and have a smaller number of layers. Deep learning models, on the other hand, are more complex and can have multiple layers [17]..
2. Data requirements: Machine learning models work well with structured data and require a smaller amount of training data to be effective. Deep learning models require large amounts of unstructured data, such as images or text, to train effectively [17]..
3. Feature extraction: Machine learning models require the engineer to manually extract features from the data, whereas deep learning models can automatically learn features from the data [17]..

4. Interpretability: Machine learning models are often more interpretable, meaning it's easier to understand how the model is making decisions. Deep learning models are generally less interpretable due to their complexity [17]..
5. Performance: Deep learning models can outperform machine learning models in tasks such as image recognition, speech recognition, and natural language processing [17]..

Overall, the main difference between machine learning and deep learning is the complexity of the models and the amount of data required for training. Machine learning is a good choice for simpler problems with structured data, while deep learning is more appropriate for complex problems with unstructured data [17].

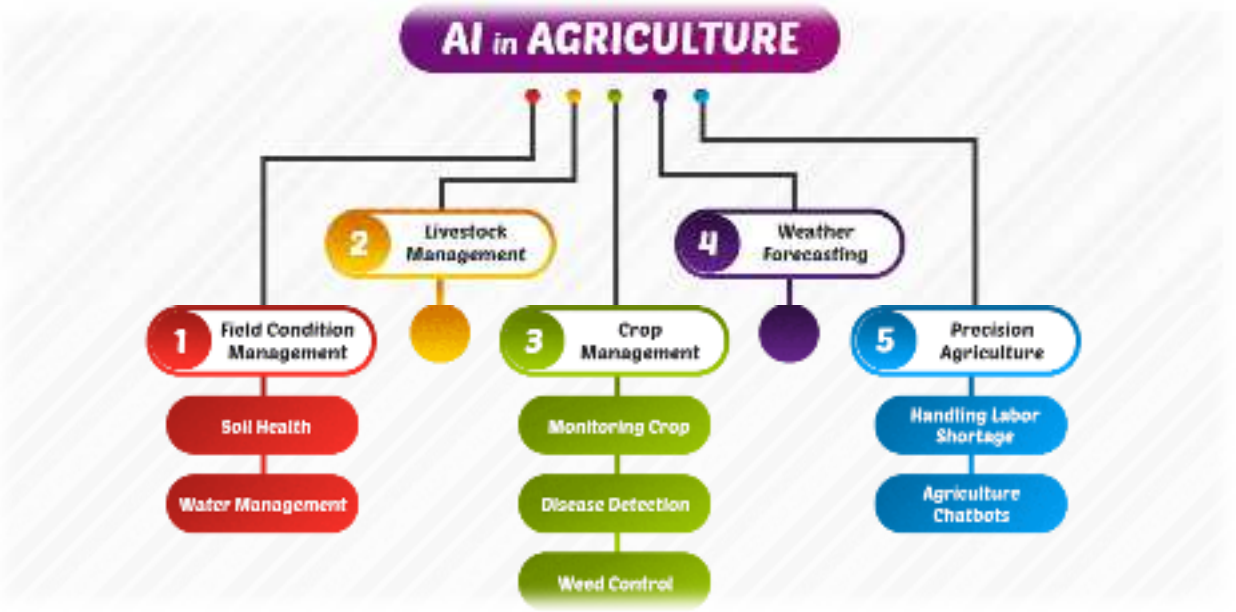
1.4 Artificial intelligence Applications

Artificial Intelligence (AI) has become an essential technology in various fields, enabling machines to perform tasks that would typically require human intelligence. AI has numerous applications, including data analysis, decision-making, and automation. This study provides an overview of the applications of AI in different fields [18].

- Healthcare: One of the most significant applications of AI is in the field of healthcare. AI technology is used for medical diagnosis, drug discovery, and patient care. AI algorithms can analyze large amounts of medical data, including electronic health records (EHRs), medical images, and genomics data, to identify patterns and predict outcomes [18].
- Finance: Another application of AI is in the financial industry. AI algorithms can be used for fraud detection, risk assessment, and portfolio optimization. AI can analyze vast amounts of financial data to identify patterns and predict market trends [18].
- Transportation: AI technology is also used in the transportation industry. AI algorithms can optimize traffic flow, reduce congestion, and improve safety. Self-driving cars and drones are examples of AI-based transportation technologies [18].
- Manufacturing: AI is also being used in the manufacturing industry. AI algorithms can optimize production processes, reduce waste, and improve product quality. AI can analyze sensor data from machines to predict equipment failures and schedule maintenance [18].

As we are talking in this study about prediction of poultry diseases we must talk about AI in the field of Animal husbandry, this science which consist of breeding, caring, and managing animals for food, fiber, and other purposes. In recent years, there has been a growing interest in the application of AI technology in animal husbandry. AI can be used for various purposes in this field, including monitoring animal health, optimizing feeding strategies, and improving breeding outcomes [18].

Figure I. 06: **Artificial intelligence In Agriculture** [19].



1.5.1 Applications of AI in Animal Husbandry:

1. Monitoring Animal Health: AI can be used to monitor the health of animals and detect any abnormalities early on. For instance, sensors can be attached to animals to track their vital signs and behavior, and AI algorithms can analyze this data to detect any signs of illness or distress [20].

2. Optimizing Feeding Strategies: AI can also be used to optimize feeding strategies for animals. AI algorithms can analyze data on animal weight, feed intake, and growth rates to determine the optimal feeding schedule and ration [20].

3. Improving Breeding Outcomes: AI can be used to improve breeding outcomes in animal husbandry. For instance, AI algorithms can analyze data on animal genetics, behavior, and health to determine the best breeding matches and optimize breeding strategies [20].

1.5.1.2 The limitations of AI in animal husbandry:

1 Data Bias: One of the biggest limitations of AI in animal husbandry is the potential for bias in the data used to train the AI models. Biased data can lead to inaccurate predictions and decisions, which can have negative impacts on animal welfare and productivity [20].

2 Limited Data Availability: In some cases, there may be limited data available for AI applications in animal husbandry. This can be due to factors such as the cost of data collection or the limited availability of certain types of data [14].

3 Ethical Considerations: The use of AI in animal husbandry raises ethical considerations, particularly around issues such as animal welfare and the potential for reduced human involvement in decision-making [20].

4 Technical Limitations: AI models are only as effective as the data they are trained on, and they may not always be able to account for complex factors that can influence animal health and productivity [20].

5 Cost: Implementing AI technologies in animal husbandry can be costly, particularly for smaller-scale operations that may not have the resources to invest in expensive hardware or software [20].

It is important to note that while AI can offer many benefits in animal husbandry, it is not a one-size-fits-all solution. Careful consideration must be given to the specific needs and limitations of individual operations, and AI should be used in conjunction with other management strategies to optimize animal welfare and productivity [20]..

1.6 Artificial Intelligence Limits

While artificial intelligence (AI) has numerous benefits and various potential applications, it's crucial to bear in mind that it is not a cure-all solution. AI models' efficiency relies heavily on the quality of the data they are trained on and the instructions they are given to follow. Just like any other technology, it's important to be mindful of the potential constraints and ethical ramifications of AI systems. Therefore, a thoughtful evaluation should be conducted before implementing AI technologies in various industries at general this are ai limits [21].

1. Data Dependence: AI systems rely heavily on data, and the quality and quantity of data available can limit the effectiveness of these systems. In some cases, there may not be

enough data to train an AI system to perform a certain task, or the data available may be biased or incomplete [21].

2. Lack of Creativity: AI systems are programmed to operate within a specific set of rules and parameters, and they do not have the same creative capacity as human beings. This can limit their ability to handle complex or novel situations [21].

3. Limited Understanding of Context: AI systems are trained on specific data sets and may not always understand the larger context in which the data exists. This can lead to errors or biased decision-making [21].

4. Ethical Concerns: As AI systems become more advanced, there are concerns about their impact on privacy, security, and fairness. For example, AI algorithms may perpetuate social biases or be used to manipulate individuals or groups [21].

5. Technical Limitations: AI systems are limited by the capabilities of the hardware and software on which they run. This can include limitations in processing power, memory, and bandwidth [21].

Conclusion

In conclusion, artificial intelligence (AI) is a rapidly evolving field that involves the development of intelligent machines capable of performing tasks that would typically require human intelligence. Deep learning, a subset of AI, involves training artificial neural networks to identify and classify patterns in data, leading to breakthroughs in areas such as image recognition, natural language processing, and speech recognition.

The process of AI involves using algorithms and statistical models to analyze data, identify patterns, and make predictions or decisions based on that information. AI has a broad range of potential applications, including healthcare, finance, transportation, and entertainment.

In the context of animal husbandry, AI technologies have significant potential to improve the industry by enhancing animal welfare, increasing production efficiency, and reducing labor costs. AI applications in animal husbandry include monitoring animal behavior, predicting disease outbreaks, and optimizing feed and water usage.

However, despite its potential benefits, AI also has its limits. These include the difficulty of creating truly autonomous systems, the need for vast amounts of high-quality data, and the potential for unintended consequences or bias in decision-making. Therefore,

it is essential to carefully consider the limitations and ethical implications of AI systems and put appropriate safeguards in place to ensure their responsible use.

This study utilizes AI and deep learning techniques in the animal husbandry industry, specifically for predicting poultry diseases. As highlighted in this chapter, data collection is a critical aspect of AI, and in our case, as we are working on cough detection, we need to have a good understanding of signal processing and sound classification and their integration into AI models.

CHAPTER TWO : SIGNAL PROCESSING AND SOUND CLASSIFICATION ON AI MODELS

Introduction

Artificial Intelligence (AI) has revolutionized the way we interact with technology, and its applications span across various fields, including speech recognition, music analysis, and environmental sound monitoring. In recent years, the use of AI models for sound classification has gained significant attention, and signal processing techniques have played a vital role in the initial stages of the process. The success of AI models heavily depends on the quality of the input data, which in turn relies on the proper signal processing techniques [22].

Signal processing techniques involve the use of algorithms to manipulate and transform signals, such as audio, video, or image data. These techniques can enhance the signal-to-noise ratio, filter out unwanted noise, and extract essential features that can help AI models classify the sounds accurately. The signal processing techniques used in AI models vary depending on the application, but generally, they involve a combination of pre-processing, feature extraction, and classification.

In this context, this chapter aims to review the latest signal processing and sound classification techniques used in AI models, with a particular emphasis on their applications in various fields, challenges, and future directions.

2.2 Signal Processing Definition

Signal processing is the systematic manipulation of signals to extract useful information or transform them into a desired form. It involves the analysis, synthesis, and modification of signals, which can be in the form of sound, images, or other types of data. The ultimate goal of signal processing is to extract useful information from signals and make them more understandable or usable for further processing or analysis [22].

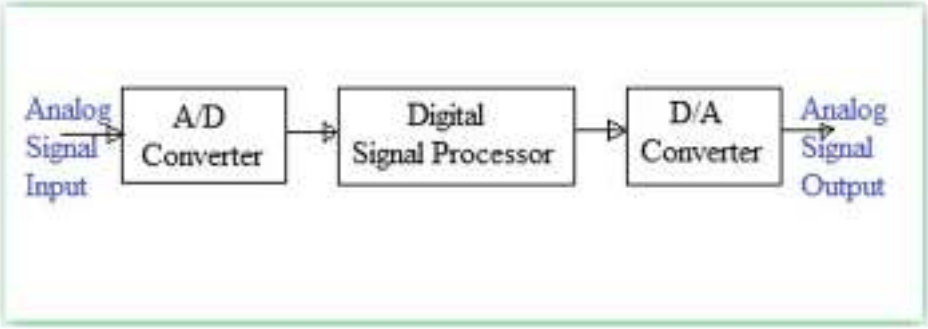


Figure II 01: Signal Processing operation [23]

2.2.1 Audio Signal Processing

Sound signal processing is a branch of signal processing that focuses on the analysis and manipulation of sound signals. It involves the use of various techniques and algorithms to transform the raw sound signals into a more meaningful representation that can be used for further analysis or processing. These techniques may include filtering, equalization, noise reduction, compression, and feature extraction, among others. Sound signal processing has various applications, including speech recognition, music analysis, acoustic monitoring, and audio communication, among others [23].

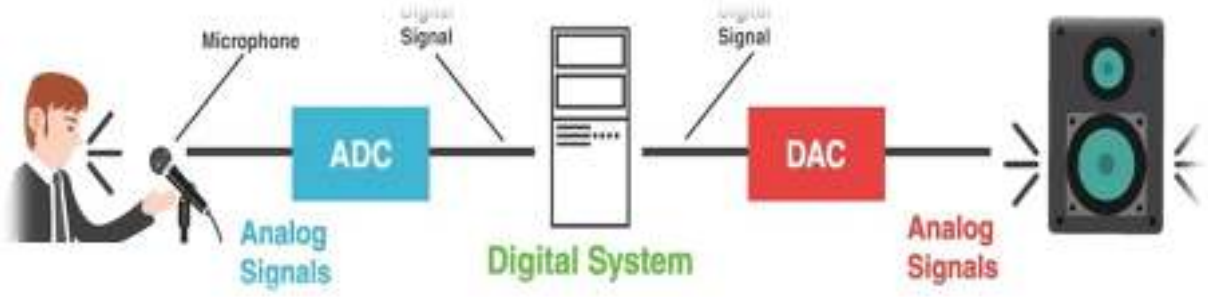


Figure II 02: Sound Signal Processing [24]

2.2.2 Steps of Sound Signal Processing

The steps of sound signal processing before using it in AI models typically include [24]:

1. Acquisition: In this step, the sound signal is acquired using a microphone or other sensing device. The analog signal is then converted to a digital signal for further processing [24].
2. Pre-processing: The acquired signal is often pre-processed to filter out noise, eliminate unwanted frequencies, and enhance the signal-to-noise ratio. Pre-processing techniques include filtering, noise reduction, and signal enhancement [24].
3. Feature extraction: In this step, the relevant features of the sound signal are extracted to form a feature vector. These features can include spectral features such as the frequency, amplitude, and phase, as well as statistical features such as the mean and variance [24].
4. Feature selection: The feature vector may contain many features that are not relevant to the classification task at hand. In this step, the most relevant features are selected using feature selection techniques such as correlation analysis, principal component analysis, and mutual information [24].
5. Classification: Finally, the selected features are fed into a machine learning algorithm for classification. The algorithm learns to classify the sound signal based on the extracted features and a set of training data. Common machine learning algorithms used for sound classification include support vector machines, decision trees, and neural networks [24].

These steps are critical in ensuring the accuracy of sound classification in AI models. Proper sound signal processing techniques can enhance the signal-to-noise ratio, filter out unwanted noise, and extract essential features that can help the AI models classify the sounds accurately [24].

2.2.1.2 Audio Signal processing technics

Signal processing techniques involve a wide range of methods used to modify or analyze signals, such as sound, image, or video. Some of the commonly used signal processing techniques in sound signal processing include [25]:

1. Pre-processing techniques: This involves removing unwanted noise, filtering the signal to enhance the signal-to-noise ratio, and normalizing the data to ensure consistency in the input data [25].

2. Feature extraction techniques: This involves extracting relevant features from the pre-processed signal, such as pitch, frequency, and amplitude, which can be used to train the AI model [25].

3. Time-frequency analysis: This technique allows for the representation of sound signals in both the time and frequency domains. By using techniques such as Short-Time Fourier Transform (STFT) or Wavelet Transform, the AI model can analyze the temporal and spectral characteristics of the sound signal [25].

4. Spectral analysis: This technique involves analyzing the spectral content of the sound signal. By using techniques such as Fast Fourier Transform (FFT), the AI model can analyze the frequency components of the signal [25].

5. Deep learning techniques: Deep learning algorithms such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can be used for sound classification, where the AI model learns to identify patterns in the pre-processed sound signals [25].

When choosing a signal processing technique for sound classification, it is important to consider the type of sound signals being analyzed and the computational resources available. Deep learning techniques may require large amounts of data and computational resources, while simpler techniques such as spectral analysis may be more computationally efficient but may not capture all the relevant features of the sound signal and this is why we are choosing deep learning for this study [25].

2.3 Sound classification

Sound classification is the process of assigning predefined categories or labels to a sound signal based on its acoustic characteristics. The aim of sound classification is to automatically distinguish different sound types, such as speech, music, and environmental sounds. To achieve accurate sound classification, various signal processing techniques are used to extract features that can effectively represent the unique characteristics of different sound types. Machine learning algorithms, such as neural networks and support vector machines, are then applied to these features to classify the sounds into their respective

categories. Sound classification has various applications, including speech recognition, music genre classification, and environmental sound monitoring [26].

2.4 Importance of Signal Processing in Sound Classification

Signal processing is critical in sound classification as it provides a means of extracting useful information from raw audio data. Sound signals captured by microphones often contain noise and unwanted artifacts that can interfere with accurate classification. Signal processing techniques are used to reduce noise and enhance the desired features of the sound signal, making it easier for the AI models to classify the sounds accurately [27].

Additionally, sound signals have complex structures and contain a vast amount of information that needs to be analyzed by AI models. Signal processing techniques can help to extract essential features from the sound signal, reducing the amount of data that needs to be analyzed. This process is known as feature extraction, and it is a critical step in sound classification [27] .

Proper signal processing techniques can also improve the signal-to-noise ratio, making it easier for AI models to differentiate between similar sounds. For example, in speech recognition, signal processing techniques are used to filter out unwanted background noise and enhance the speech signal, making it easier for the AI model to recognize spoken words accurately [27].

In summary, signal processing is essential in sound classification as it helps to enhance the quality of the sound signal, extract essential features, and improve the signal-to-noise ratio, making it easier for AI models to classify the sounds accurately [27].

Conclusion

In conclusion, the use of AI models for sound classification in poultry disease detection has shown promising results. However, the success of these models heavily depends on the quality of input data, which can be enhanced by proper signal processing techniques. The initial stages of AI models require signal processing techniques to pre-process raw data, filter out unwanted noise, and extract essential features that can help the models classify the sounds accurately. Signal processing plays a vital role in the effectiveness of AI models for sound classification, as it can significantly enhance the signal-to-noise ratio and improve the accuracy of disease detection. Overall, the integration of signal processing and sound classification in AI models for poultry disease detection has the potential to improve the

efficiency and effectiveness of disease In this study, we applied signal processing and sound classification techniques to an AI model aimed at detecting anomalies in poultry vocalizations.se monitoring and control, leading to better animal welfare and economic outcomes,

CHAPTER THREE : REALIZATION

Introduction

The realization chapter is the culmination of our project, where we bring our sound classification model to life on a Raspberry Pi. This chapter focuses on the practical implementation of our model, integrating it with hardware components, and creating a real-time sound classification system. By following the steps outlined in this chapter, we will have a fully functional system capable of identifying normal and abnormal sounds.

We begin by discussing the initial steps of model conception, including dataset collection, preprocessing, and model training. We carefully design and train our model to achieve high accuracy and robust performance in sound classification tasks.

Next, we delve into the hardware setup, which involves configuring the Raspberry Pi as the central processing unit. We connect a microphone to capture audio input, an LCD screen to display classification results, and a buzzer to provide audible feedback for abnormal sounds.

Software configuration is an essential aspect of our system. We explore the choice of operating system for the Raspberry Pi and guide you through the necessary installations. This includes setting up libraries for audio processing and model loading.

The heart of our realization phase is the real-time sound classification. We cover the preprocessing steps for capturing and preparing audio data, as well as loading our trained model onto the Raspberry Pi. We demonstrate how the system can process incoming audio streams and make accurate predictions in real-time.

To enhance user experience, we integrate an LCD screen to visually display classification results, allowing easy identification of normal and abnormal sounds. We also incorporate a buzzer that produces audible feedback for abnormal sounds, providing an additional layer of information.

Testing and evaluation are crucial to validate the performance of our system. We define test scenarios using different sound samples and employ evaluation metrics to measure system accuracy and efficiency. The results are analyzed to gain insights into system strengths and areas for improvement.

In conclusion, the realization chapter transforms our sound classification model from theory to practice. By following the steps outlined, you will create a real-time sound classification system on a Raspberry Pi. This chapter provides a comprehensive guide to

the implementation process and equips you with the knowledge to adapt the system to your specific needs.

3.2 AI Model Conception:

The foundation of our real-time sound classification system lies in the construction of a robust and accurate model. In this section, we will outline the key steps involved in building our model for sound classification.

3.2.1 Dataset Building:

In order to build an accurate and diverse sound classification model, we faced the challenge of collecting a sufficient amount of high-quality sound data. Collecting a comprehensive dataset manually proved to be a time-consuming and challenging task, as it required configuring specialized microphones and recording sounds in various real-world environments.

To overcome this obstacle, we leveraged an existing dataset called Google AudioSet. is a large-scale collection of audio samples from a wide range of sources, covering a vast array of sounds. This dataset provided a valuable starting point for our sound classification model.

However, the Google AudioSet dataset alone did not fully meet our requirements. It lacked specific sound samples that were relevant to our use case and did not cover all the necessary abnormal sound categories. Therefore, we employed data augmentation techniques to enhance the dataset's diversity and balance.

Data augmentation involved applying various transformations to the existing audio samples, such as pitch shifting, time stretching, noise addition, and amplitude scaling. These techniques helped simulate different variations of the original sounds and created a more comprehensive dataset for training our model. By augmenting the dataset, we were able to capture a broader range of sound variations and improve the model's ability to generalize and classify abnormal sounds accurately.

The combination of the Google AudioSet dataset and data augmentation techniques enabled us to overcome the limitations of manually collecting sound data and provided us with a robust and diverse dataset for training our sound classification model.

By utilizing these resources and techniques, we ensured that our model had access to a wide variety of normal and abnormal sound samples, enabling it to learn and generalize effectively in real-world scenarios.

3.2.1.1 Dataset Augmentation

To do the data augmentation we used two technics are:

1. Mixup Data Augmentation: is a data augmentation technique commonly used in sound classification tasks. It involves mixing pairs of audio samples from the dataset to create new training examples. The process combines two audio samples by taking a weighted sum of their waveforms and their corresponding labels. This creates a blended audio sample that lies somewhere between the two original samples [28].

By applying Mixup data augmentation, we introduce additional variability and diversity into the training data. It helps the model learn to generalize better by exposing it to a broader range of sound combinations and labels. Mixup encourages the model to focus on relevant audio features rather than relying solely on specific instances in the training set. This regularization technique aids in reducing overfitting and improving the model's performance on unseen data [28].

2. SpecAugment: is a popular data augmentation technique used specifically for audio and speech recognition tasks. It operates in the spectrogram domain, which is a visual representation of the audio signal's frequency content over time. SpecAugment applies random transformations to the spectrogram, introducing localized modifications [29].

The three main transformations used in SpecAugment are [29]. :

- Time Masking: This involves masking consecutive time steps in the spectrogram by setting them to zero. It helps the model focus on different temporal segments of the audio and enhances its robustness to temporal variations [29].
- Frequency Masking: This technique masks a random set of frequency bands in the spectrogram. By doing so, it encourages the model to attend to different frequency components and improves its ability to handle variations in the frequency domain [29].

- Time Warping: Time warping randomly warps the spectrogram along the time axis, introducing slight temporal distortions. This further increases the model's robustness to temporal variations in the input audio [29].

SpecAugment helps in preventing overfitting and enhances the model's ability to generalize by introducing variations in both the time and frequency domains. By applying SpecAugment, we improve the model's robustness to different types of noise, background variations, and temporal distortions that may occur in real-world audio recordings [29]..

By combining Mixup data augmentation and SpecAugment techniques, we enhance the diversity and variability of the training data. This enables the model to learn more robust and generalized representations of both normal and abnormal sounds, improving its classification performance on unseen audio samples.

In order to applicate this techniques using python programming language:

1. Mixup Data Augmentation:

```
import os
import torchaudio
import torch
import torch.nn.functional as F
import torchaudio.transforms as T
import shutil
# Set the directory where the audio files are located
directory = '/content/drive/MyDrive/normalpoultrysound/'
# Define the output directory to save the mixed files
output_directory = '/content/drive/MyDrive/MixedFiles/'
# Create the output directory if it doesn't exist
os.makedirs(output_directory, exist_ok=True)
# List all audio files in the directory
file_names = os.listdir(directory)
# Resampling parameters
desired_sample_rate = 16000
desired_channels = 1
# Initialize mixed audio
mixed_audio = None
max_length = 0
# Loop through the audio files and build mixed files
for i, file_name in enumerate(file_names):
    # Load audio file
    file_path = os.path.join(directory, file_name)
    waveform, sample_rate = torchaudio.load(file_path)
    # Resample audio
    if sample_rate != desired_sample_rate:
        waveform = T.Resample(sample_rate, desired_sample_rate)(waveform)
    # Convert to mono if needed
    if waveform.shape[0] > 1 and desired_channels == 1:
        waveform = torch.mean(waveform, dim=0, keepdim=True)
    # Update the maximum length
    max_length = max(max_length, waveform.shape[-1])
    # Perform mixing
    if mixed_audio is None:
        mixed_audio = waveform
    else:
        mixed_audio = F.pad(mixed_audio, (0, max_length - mixed_audio.shape[-1]))
        waveform = F.pad(waveform, (0, max_length - waveform.shape[-1]))
        mixed_audio += waveform
    # Save the mixed audio
    output_file_path = os.path.join(output_directory, f'mixed_audio{i}.wav')
    torchaudio.save(output_file_path, mixed_audio, sample_rate=desired_sample_rate)
    print(f"Mixed audio {i+1} saved to:", output_file_path)
```

When we execute this code, it will load the audio files, resample them if necessary, convert them to mono, mix them together, and save the mixed audio files in the specified output directory. The output file names will be in the format "mixed_audio{i}.wav", where {i} is the index of the mixed audio file.

We applicate this technic (the same code) on the two types of data normal, and abnormal.

2. SpecAugmen

```
import os
import torchaudio
import torch
import torchaudio.transforms as T
# Set the directory where the audio files are located
directory = '/content/drive/MyDrive/Normal/'
# Set the output directory to save augmented files
output_directory = '/content/drive/MyDrive/sick/'
# List all audio files in the directory
file_names = os.listdir(directory)
# Set SpecAugment parameters
time_warping_para = 80
frequency_masking_para = 27
time_masking_para = 70
num_time_masks = 2
num_frequency_masks = 2
# Loop through the audio files
for file_name in file_names:
    # Load audio file
    file_path = os.path.join(directory, file_name)
    waveform, sample_rate = torchaudio.load(file_path)
    # Apply STFT
    spec = torch.stft(waveform, n_fft=400, hop_length=160,
win_length=400, window=torch.hann_window(400), center=True,
pad_mode='reflect', normalized=False, return_complex=True)
    # Apply SpecAugment
    num_bins, num_frames = spec.shape[-2], spec.shape[-1]
    warped_masked_spec = spec.clone()
    # Time warping
    if time_warping_para > 0:
        time_warp_factor = torch.randint(-time_warping_para,
time_warping_para + 1, (1,))
        src_idx = torch.arange(num_frames)
        tgt_idx = torch.clamp(src_idx + time_warp_factor, 0, num_frames
- 1)
        warped_masked_spec = warped_masked_spec[:, :, tgt_idx]
    # Frequency masking
    for _ in range(num_frequency_masks):
        f = torch.randint(0, frequency_masking_para + 1, (1,))
        f0 = torch.randint(0, num_bins - f.item() + 1, (1,))
        warped_masked_spec[:, f0:f0 + f, :] = 0
    # Time masking
    for _ in range(num_time_masks):
        t = torch.randint(0, min(time_masking_para + 1, num_frames),
(1,))
        t0 = torch.randint(0, num_frames - t.item() + 1, (1,))
        warped_masked_spec[:, :, t0:t0 + t] = 0
    # Apply inverse STFT to obtain augmented waveform
    augmented_waveform = torch.istft(warped_masked_spec, n_fft=400,
hop_length=160, win_length=400, window=torch.hann_window(400),
center=True, normalized=False, length=waveform.shape[-1])
    # Save augmented audio
    output_file_path = os.path.join(output_directory, file_name)
    torchaudio.save(output_file_path, augmented_waveform,
sample_rate=sample_rate)
    print("Augmented audio saved to:", output_file_path)
```

By executing this code, each audio file in the specified directory will undergo SpecAugment techniques, resulting in augmented versions of the audio files. These augmented files will be saved in the output directory with the same file names and format as the input audio files.

3.2.1.2 Data Preparing:

In this step of data preparation, we are combining the normal and abnormal voice files from their respective directories into a single dataset directory. This step is performed to have all the data in one location for easier access and further processing.

We do that using this code on python

```
import os
import shutil

# Set the paths to the normal and abnormal voice directories
normal_voice_dir = '/content/drive/MyDrive/normal/'
abnormal_voice_dir = '/content/drive/MyDrive/abnormal/'

# Set the path to the target directory for the combined dataset
dataset_dir = '/content/drive/MyDrive/predata/'

# Create the target directory if it doesn't exist
os.makedirs(dataset_dir, exist_ok=True)

# Function to copy files from source directory to target directory
def copy_files(source_dir, target_dir):
    files = os.listdir(source_dir)
    for file in files:
        source_path = os.path.join(source_dir, file)
        target_path = os.path.join(target_dir, file)
        shutil.copyfile(source_path, target_path)

# Copy normal voice files to the dataset directory
copy_files(normal_voice_dir, dataset_dir)

# Copy abnormal voice files to the dataset directory
copy_files(abnormal_voice_dir, dataset_dir)

print("Data preparation completed.")
```

This code ensures that all the necessary data is available in a single directory, which simplifies further processing steps such as data loading and training.

3.2.2 Data Preprocessing

In this part of the code, we are implementing the preprocessing stage for our audio data. The goal of this preprocessing is to transform the raw audio signals into a format that is suitable for further analysis or machine learning tasks.

The source code

```
import os
import numpy as np
import librosa
from google.colab import drive
# Mount Google Drive
drive.mount('/content/drive', force_remount=True)
# Folder paths
folder_path = '/content/drive/MyDrive/sik'  # Path to the folder in
Google Drive
output_folder = '/content/drive/MyDrive/sick'  # Output folder for
preprocessed data
# Preprocessing steps
n_mels = 20  # Number of Mel filterbanks
hop_length = 512  # Number of samples between successive frames
max_length = 1000  # Maximum length in frames
# Iterate over audio files in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.wav'):  # Adjust file extension if necessary
        # Load audio file from Google Drive
        file_path = os.path.join(folder_path, filename)
        audio, sr = librosa.load(file_path)
        # Preprocessing steps
        mel_spectrogram = librosa.feature.melspectrogram(y=audio,
sr=sr, n_mels=n_mels, hop_length=hop_length)
        log_mel_spectrogram = librosa.amplitude_to_db(mel_spectrogram,
ref=np.max)
        # Pad or trim spectrogram to a fixed length
        if log_mel_spectrogram.shape[1] < max_length:
            log_mel_spectrogram = np.pad(log_mel_spectrogram,
                                         ((0, 0), (0, max_length -
log_mel_spectrogram.shape[1])))
        # Save preprocessed data to Google Drive
        output_path = os.path.join(output_folder,
filename.replace('.wav', '.npy'))
        np.save(output_path, log_mel_spectrogram)
```

The main actions carried out in this code snippet are as follows:

1. **Loading audio**: The code iterates over the audio files in a specified folder and loads each file using the `librosa.load` function. This step extracts the audio waveform and the sample rate from each file.

2. **Mel spectrogram computation**: Using the loaded audio waveform, the code calculates the Mel spectrogram representation using the `librosa.feature.melspectrogram` function. The Mel spectrogram captures the distribution of frequencies in the audio signal over time.

3. **Logarithmic transformation**: The code applies a logarithmic transformation to the spectrogram using the `librosa.amplitude_to_db` function. This transformation converts the amplitude values to decibels (dB), enhancing the perceptual representation of the spectrogram.

4. **Padding or trimming**: To ensure a consistent length for the spectrograms, the code checks the shape of each log-mel spectrogram and pads it with zeros or trims it if necessary. This ensures that all spectrograms have the same number of frames.

5. **Saving preprocessed data**: The preprocessed log-mel spectrograms are saved as NumPy arrays in an output folder. Each preprocessed file corresponds to an audio file, and the filename is modified to have the `.npy` extension.

By executing this code, we are processing the audio data by computing the Mel spectrogram, applying a logarithmic transformation, standardizing the spectrogram length, and saving the preprocessed data. These preprocessing steps prepare the audio data for subsequent analysis, such as training a machine learning model for sound classification or anomaly detection.

3.2.2.1 Data Splitting

The split allows us to separate the data into distinct subsets for training and testing. The training data is used to train the model, while the testing data is used to evaluate the model's performance. This split helps assess how well the model generalizes to unseen data.

We applicate this step using the follow code:

```
import os
import shutil
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
# Set the path to the dataset directory
dataset_dir = '/content/drive/MyDrive/npyprdata/'
# List all the audio file paths in the dataset directory
file_paths = [os.path.join(dataset_dir, file) for file in
os.listdir(dataset_dir)]
# Create the dataset instance
dataset = AudioDataset(file_paths)
# Path to the folder containing the preprocessed data
data_folder = dataset_dir
# List all the preprocessed data files in the folder
file_names = os.listdir(data_folder)
# Get the full file paths
file_paths = [os.path.join(data_folder, file) for file in file_names]
data = []  # List to store the preprocessed data (Mel spectrograms)
labels = []  # List to store the corresponding labels
    # Load the preprocessed data
    preprocessed_data = np.load(file_path)
    # Get the label based on the file path or filename
    label = get_label(file_path)
       .append(label)
# Split the data into training and testing sets
train_data, test_data, train_labels, test_labels =
train_test_split(data, labels, test_size=0.2, random_state=42)
```

In this code snippet, data represents the preprocessed data and labals represents the corresponding labels. The train test split function is called with the following arguments:

- `data`: The preprocessed data array.
- `labels`: The corresponding labels array.
- `test_size`: The proportion of the data to be allocated for testing. In this case, it is set to 0.2, which means that 20% of the data will be used for testing.
- `random_state`: An optional parameter that sets the random seed for reproducibility. It ensures that the same random split is obtained each time the code is executed.

After executing this code, we will have four arrays:

- `train_data`: The training data, which is a subset of the original data.
- `test_data`: The testing data, which is the remaining subset of the original data.
- `train_labels`: The labels corresponding to the training data.
- `test_labels`: The labels corresponding to the testing data.

These arrays can then be used for training and evaluating machine learning model.

3.2.2.2 Label Extraction Process

The label extraction process is important in machine learning tasks because it assigns a specific category or class to each input sample. In this case, the labels represent whether the audio file is classified as normal or abnormal.

- Code of Label Extraction Process :

Iterate over the preprocessed data files and labels def get_label(audio_path):

Extract the label from the audio path or filename

Check if the audio path contains the 'trainnormal' directory or the 'normal_' substring

if '/trainnormal/' in audio_path or 'normal_.npy' in os.path.basename(audio_path):

return 0

Check if the audio path contains the 'abnrml' directory or the 'abnormal_' substring

elif '/trainabnormal/' in audio_path or 'abnormal_.npy' in os.path.basename(audio_path):

return 1

else: return -1

- Code explication :

The function `get_label(audio_path)` is responsible for extracting the label from the audio file path or filename. It checks if the audio path contains certain directory names or specific substrings in the filename to determine the label. In this case, it assigns the label `0` for normal audio files and `1` for abnormal audio files. If the audio path or filename does not match any expected pattern, it assigns the label `-1` as a default value.

This function is typically used during the data preprocessing step to assign labels to the corresponding audio data. It helps organize and categorize the data based on their characteristics, which is essential for training a supervised learning model.

The model creation step, on the other hand, focuses on defining the architecture and parameters of the machine learning model itself, such as selecting the model type (e.g., logistic regression, neural network, etc.), specifying the number of layers .

3.2.3 AI Model Type's Choice

We choice to use a logistic regression model for our task this choice depends on several factors, including the nature of our data and the problem we are trying to solve. Here are some reasons why logistic regression might be a suitable choice [30] :

1. **Binary Classification:** Logistic regression is a popular choice for binary classification problems where the goal is to predict one of two possible outcomes. In our case, we are classifying audio data into two categories: normal and abnormal. Logistic regression is well-suited for such problems as it provides a probabilistic interpretation and produces class probabilities [30] .

2. **Efficiency:** Logistic regression is a relatively simple and computationally efficient model compared to more complex models like neural networks. It can handle large datasets and high-dimensional feature spaces without significant computational overhead [30].

3. **No Assumptions of Linearity:** Although logistic regression is a linear model, it can capture non-linear relationships between the features and the target variable by using non-linear transformations or interactions. This flexibility allows logistic regression to handle a wide range of data patterns [30].

4. **Good Performance with Sufficient Data:** Logistic regression can perform well when we have a small amount of labeled training data. Even if our dataset is not large enough, logistic regression can provide good accuracy and generalization.

However, it's important to note that the suitability of the logistic regression model depends on the specifics of our dataset and problem. It's always a deepens to practice to consider alternative models, such as decision trees, support vector machines, or neural

networks, and evaluate their performance to choose the best model for our specific task we used this model to be simple and effective as possible [30] .

3.2.4 Model Creation

The model creation step involves defining the architecture and parameters of the machine learning model that will be used to learn from the training data and make predictions. The model is responsible for capturing and representing the underlying patterns and relationships present in the data.

The Used code :

```
from sklearn.linear_model import LogisticRegression

# Create a logistic regression model instance
 model = LogisticRegression()
```

3.2.5 Model Training

This step is a crucial part of the machine learning pipeline where the model learns from the labeled training data to make accurate predictions on new, unseen data. During the training process, the model adjusts its internal parameters based on the patterns and relationships present in the training data.

We used this code:

```
# Fit the model to the training data
model.fit(train_data, train_labels)
```

the model is trained on the training data using the `fit` method. This process involves optimizing the model parameters based on the input data and corresponding labels.Once the model is trained, it can be used for making predictions on new, unseen data.

3.2.6 Model Evaluation

Evaluation is a crucial step in assessing the performance of a machine learning model. It involves measuring the model's ability to make accurate predictions on unseen data and understanding its strengths and weaknesses. The evaluation process provides insights into how well the model generalizes to new instances and helps determine if the model is suitable for its intended purpose.

CHAPTER THREE : REALIZATION

There are various evaluation metrics and techniques used to assess the performance of a model, depending on the type of problem and the nature of the data. Some commonly used evaluation metrics include accuracy, precision, recall, F1-score, and confusion matrix.

In our case we calculate accuracy and confusion matrix and classification

- **Code of Evaluation**

```
# Use the trained model to make predictions on the test data
predictions = model.predict(test_data)

# Evaluate the model's performance
accuracy = model.score(test_data, test_labels)
print("Accuracy:", accuracy)

# Calculate additional evaluation metrics
from sklearn.metrics import confusion_matrix, classification_report

# Calculate confusion matrix
cm = confusion_matrix(test_labels, predictions)
print("Confusion Matrix:")
print(cm)

# Calculate classification report
report = classification_report(test_labels, predictions)
print("Classification Report:")
print(report)
```

- Code Explication :

1. `predictions = model.predict(test_data)`: This line uses the trained model (`model`) to make predictions on the test data (`test_data`). It applies the learned pattern from the training phase to classify each sample in the test data and assigns a predicted label to each sample. The predicted labels are stored in the `predictions` array.

2. `accuracy = model.score(test_data, test_labels)`: This line calculates the accuracy of the model on the test data. The `score` method of the model calculates the mean accuracy by comparing the predicted labels (`predictions`) with the true labels (`test_labels`). The accuracy represents the proportion of correctly classified samples in the test data.

3. `print("Accuracy:", accuracy)`: This line prints the accuracy of the model on the test data.

4. `from sklearn.metrics import confusion_matrix, classification_report`: This line imports the `confusion_matrix` and `classification_report` functions from the `sklearn.metrics` module. These functions are used to evaluate the performance of the classification model further.

5. `cm = confusion_matrix(test_labels, predictions)`: This line calculates the confusion matrix based on the true labels (`test_labels`) and the predicted labels (`predictions`). The confusion matrix provides a summary of the model's predictions by showing the number of true positive, true negative, false positive, and false negative predictions.

6. `print("Confusion Matrix:")` and `print(cm)`: These lines print the confusion matrix, providing a visual representation of the model's performance.

7. `report = classification_report(test_labels, predictions)`: This line calculates the classification report, which includes various metrics such as precision, recall, F1-score, and support for each class. The report provides a comprehensive evaluation of the model's performance on each class.

8. `print("Classification Report:")` and `print(report)`: These lines print the classification report, providing a detailed evaluation of the model's performance.

3.2.6.1 Evaluation Results

Accuracy: 0.90

Confusion Matrix:
[[27 3]
[2 18]]

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.90	0.92	30
1	0.86	0.90	0.88	20
accuracy			0.90	50
macro avg	0.89	0.90	0.90	50

weighted avg	**0.90**	**0.90**	**0.90**	**50**

3.2.6.2 Evaluation Results analyze

- Accuracy: The overall accuracy of the model is 0.90, which means that it correctly classified 90% of the samples in the test data.
- Confusion Matrix: The confusion matrix provides information about the model's predictions for each class. In this case, the matrix is as follows:

[[27 3] [2 18]]

- True Positive (TP): The model predicted 27 samples correctly as class 0.
- False Positive (FP): The model predicted 3 samples as class 1, but they actually belong to class 0.
- False Negative (FN): The model predicted 2 samples as class 0, but they actually belong to class 1.
- True Negative (TN): The model predicted 18 samples correctly as class 1.

Classification Report: The classification report provides several metrics for each class:

- Precision: Precision represents the proportion of correctly predicted positive samples out of the total predicted positive samples. For class 0, the precision is 0.93, indicating that 93% of the samples predicted as class 0 were actually class 0. For class 1, the precision is 0.86, indicating that 86% of the samples predicted as class 1 were actually class 1.
- Recall: Recall (also known as sensitivity or true positive rate) represents the proportion of correctly predicted positive samples out of the total actual positive samples. For class 0, the recall is 0.90, indicating that the model correctly identified 90% of the actual class 0 samples. For class 1, the recall is also 0.90, indicating that the model correctly identified 90% of the actual class 1 samples.
- F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. For class 0, the F1-score is 0.92, and for class 1, the F1-score is 0.88.
- Support: Support represents the number of samples in each class. In this case, there are 30 samples for class 0 and 20 samples for class 1.

- Weighted Average: The weighted average calculates the average metrics (precision, recall, F1-score) considering the support of each class. The weighted average precision, recall, and F1-score are all 0.90 in this case, indicating the overall performance of the model.

In summary, the model achieved an accuracy of 90% and performed well in terms of precision, recall, and F1-score for both classes. The classification report and confusion matrix provide insights into the model's performance for each class, allowing us to assess its effectiveness in differentiating between the classes.

3.2.7 Model Saving

Saving the model allows us to persist it to disk, so we can later load it and use it for making predictions on new data without having to retrain the model

We used this code:

```
import joblib

# Save the trained model
model_path = '/content/drive/MyDrive/model.pkl'
joblib.dump(model, model_path)
```

In this code, the `joblib.dump()` function is used to save the `model` object to the specified `model_path`. The `model.pkl` file will be created at the specified location.

3.3 Software Tools:

3.3.1 Python Language

There are several reasons why Python is was our choice for working on this project:

1. **Ease of Use**: Python is known for its simplicity and readability, making it easy to learn and write code. It has a clean and intuitive syntax that resembles English, which reduces the learning curve for beginners and facilitates collaboration among team members [31].

2. **Vast Ecosystem of Libraries**: Python has a rich ecosystem of libraries and frameworks specifically designed for machine learning and data science. Libraries such as NumPy, Pandas, Matplotlib, scikit-learn, and TensorFlow provide powerful tools for

data manipulation, analysis, visualization, and building machine learning models. This extensive collection of libraries saves time and effort by providing pre-implemented functions and algorithms [31].

3. **Great Community Support**: Python has a large and active community of developers and data scientists. This community contributes to the development of open-source libraries, shares knowledge through online forums and communities, and provides support to newcomers. The availability of resources, tutorials, and code examples makes it easier to find solutions to problems and learn from others[31].

4. **Cross-Platform Compatibility**: Python is a cross-platform language, meaning that code written in Python can run on various operating systems, including Windows, macOS, and Linux. This makes it flexible and ensures that the code can be easily deployed on different platforms without major modifications [31].

5. **Integration with Other Languages and Tools**: Python can be easily integrated with other programming languages and tools. For example, Python can be used for data preprocessing and model development, and then the trained models can be seamlessly integrated into production systems written in other languages. Additionally, Python can interact with popular databases, web frameworks, and APIs, allowing for easy integration with other components of a project [31].

6. **Machine Learning and Data Science Community**: Python has emerged as a dominant language in the field of machine learning and data science. Many research papers, tutorials, and resources are available in Python, and most machine learning frameworks and tools provide Python APIs. This strong presence in the machine learning community makes Python a natural choice for working on such projects [31].

7. **Versatility**: Python is a versatile language that can be used for a wide range of tasks beyond machine learning. It is widely used in web development, scripting, scientific computing, and automation, making it a valuable skill to have in various domains.

Overall, Python's simplicity, extensive library ecosystem, community support, cross-platform compatibility, and versatility make it an excellent choice for working on machine learning and data science projects. Its ease of use and powerful tools enable developers and data scientists to prototype, experiment, and deploy machine learning models efficiently[31].

3.3.1.1 Used Python's libraries

- `os`: It is a Python built-in library for interacting with the operating system. It provides functions for working with file paths, directories, and other operating system-related tasks [32].

- `shutil`: It is a Python built-in library that provides high-level file operations. In this code, it is used to copy files from one directory to another [32].

- `numpy` (imported as `np`): It is a popular library for numerical computing in Python. It provides support for multi-dimensional arrays and various mathematical operations. It is used in this code for manipulating and storing numerical data [32].

- `sklearn` (Scikit-learn): It is a powerful machine learning library in Python. It provides a wide range of tools and algorithms for tasks such as classification, regression, clustering, and model evaluation. In this code, it is used for data splitting (`train_test_split`), logistic regression model creation, and evaluation metrics (`confusion_matrix`, `classification_report`) [32].

- `joblib`: It is a library for efficient serialization of Python objects to disk. It is used in this code to save the trained model to a file [32].

- `librosa`: It is a popular library for audio and music signal processing in Python. It provides various functions for loading audio files, extracting audio features, and performing audio analysis. In this project, `librosa` is used for loading audio files, computing mel spectrograms (`librosa.feature.melspectrogram`), and converting amplitudes to decibels (`librosa.amplitude_to_db`) [32].

- `torchaudio`: It is a library for audio processing and deep learning with PyTorch. It provides audio I/O functionality, transformations, and dataset handling for working with audio data in PyTorch. In this project, `torchaudio` is used for loading and saving audio files (`torchaudio.load`, `torchaudio.save`), as well as for applying the SpecAugment technique to the audio spectrograms [32]..

3.3.2 Google colab

Google Colab is an online platform provided by Google that allows users to write, run, and collaborate on Python code using Jupyter notebooks [33]. It provides a cloud-based

computing environment that offers several advantages for machine learning and data science projects we choose to work on google colab in place of a python IDE for many reasons are:

1. **Free GPU and TPU**: Google Colab provides free access to GPU (Graphics Processing Unit) and TPU (Tensor Processing Unit) resources. This is particularly beneficial for computationally intensive tasks, such as training deep learning models, as it significantly reduces the training time compared to running the code on a CPU [33].

2. **Interactive Coding Environment**: Colab notebooks provide an interactive coding environment where you can write and execute code in individual cells. This allows for easy experimentation and iterative development, as you can run code cells independently and modify them as needed [33].

3. **Pre-installed Libraries**: Colab comes pre-installed with popular Python libraries, including NumPy, Pandas, Matplotlib, and scikit-learn, among others. This saves time and effort in setting up the environment and installing necessary dependencies [33].

4. **Integration with Google Drive**: Colab integrates seamlessly with Google Drive, allowing to access and store files directly from Google Drive. This is useful for loading and saving data, models, and other files required for your project [33].

5. **Collaborative Features**: Colab supports real-time collaboration, enabling multiple users to work on the same notebook simultaneously. This facilitates teamwork and knowledge sharing, as team members can view and contribute to the code and analysis in real-time.

6. **Hardware and Memory Management**: Colab handles the underlying infrastructure and resource management, including memory allocation and disk space. This frees users from the burden of managing hardware and allows them to focus on the code and analysis [33].

7. **Notebook Persistence**: Colab notebooks are automatically saved to Google Drive, ensuring that we work is preserved even if we close the browser or lose the connection. This eliminates the risk of losing code and analysis progress [33].

8. **Access to Additional Tools**: Colab provides access to additional tools and resources, such as Google Cloud services, GitHub integration, and various APIs. These tools expand the capabilities of our project and enable integration with other platforms and services [33].

Overall, Google Colab offers a convenient and powerful environment for developing and running machine learning projects. It provides access to computational resources, simplifies the setup process, and promotes collaboration, making it a popular choice for data scientists, researchers, and developers working on machine learning tasks [33].

3.4 Hardware Setup

3.4.1 Raspberry Pi 4 Board

In this project we used raspberry Pi 4, single-board computer (SBC) that follows the open-source hardware philosophy. It is equipped with a system-on-a-chip (SoC) that integrates a central processing unit (CPU), memory, graphics processing unit (GPU), and various input/output (I/O) interfaces on a single board [33].

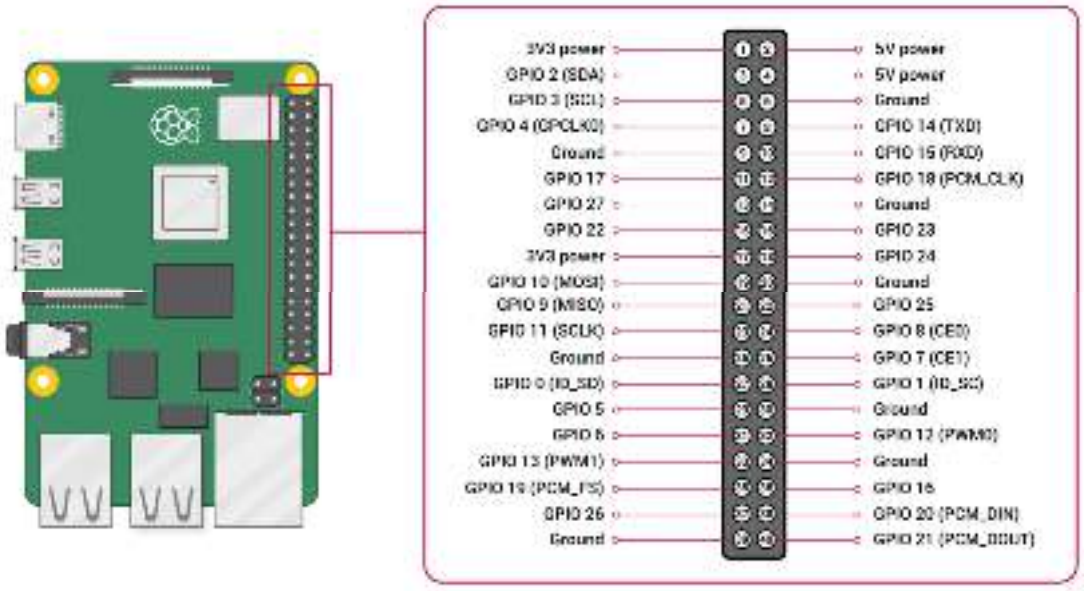


Figure III 01: Raspberry Pi 4 Pins [24]

3.4.1.1 Reasons of choosing Raspberry Pi board for this project

1. Compact and Portable: Raspberry Pi is a small and portable device, making it easy to integrate into sound classification systems. Its compact size allows for flexibility in placement and deployment, whether it's embedded within a larger system or used as a standalone device [33].

2. Sufficient Processing Power: Raspberry Pi 4, with its quad-core CPU and improved processing capabilities, provides sufficient power to handle sound classification tasks. It can efficiently process audio data, perform real-time feature extraction, and execute machine learning algorithms for classification [33].

3. GPIO and I/O Interfaces: Raspberry Pi boards offer General-Purpose Input/Output (GPIO) pins and various I/O interfaces, allowing easy connectivity with external devices such as microphones, sensors, LCD screens, and buzzers. This facilitates the integration of audio input/output components necessary for sound classification systems [33].

4. Support for Libraries and Tools: Raspberry Pi supports a wide range of libraries and tools used for audio processing, machine learning, and data analysis. Libraries like librosa, torchaudio, and scikit-learn provide functionalities for audio feature extraction, preprocessing, and model training. Additionally, popular machine learning frameworks like TensorFlow and PyTorch can be installed and utilized on Raspberry Pi [33].

5. Cost-Effective Solution: Raspberry Pi offers a cost-effective solution for sound classification systems. It is considerably more affordable compared to high-end computing devices, making it accessible for hobbyists, researchers, and educational institutions. The lower cost allows for scalability and wider adoption of sound classification technology.

6. Customization and Flexibility: Raspberry Pi provides a high level of customization and flexibility. Users can configure the software and hardware components according to their specific requirements. They have the freedom to develop and adapt the sound classification system based on their application needs, allowing for tailored solutions [33] .

Figure III 02: Raspberry Pi 4 Components [34]

3.4.2 ISD1820 Microphone Sensor

The ISD1820 is a sound recording module featuring an on-board microphone and various playback functions [35].

Figure III 03 : ISD1820 Microphone Sensor

3.4.2.1 Principle of Works of ISD1820 Microphone Sensor:

1. Sound Sensing: The ISD1820 module incorporates a built-in microphone that captures sound waves from the environment. When there is a sound present, the microphone converts the sound waves into electrical signals [35].

2. Analog-to-Digital Conversion: The module also includes an analog-to-digital converter (ADC) that converts the analog electrical signals from the microphone into digital data. This conversion process enables the Raspberry Pi to receive and process the audio data in a format suitable for sound classification [35].

3. Digital Output Signal: The ISD1820 module provides a digital output signal (OUT pin) that indicates the presence of sound. This signal can be connected to a GPIO pin on the Raspberry Pi, allowing it to monitor the state of the output signal. When sound is detected, the GPIO pin will go HIGH, indicating the presence of sound [35].

3.4.2.2 Advantages of Using the ISD1820 Microphone Sensor for Sound Classification Projects:

The ISD1820 microphone sensor offers several advantages that make it a suitable choice for sound classification projects:

1. Simplicity and Ease of Integration: The ISD1820 module is designed to be user-friendly and easy to integrate with the Raspberry Pi. It comes as a complete module, requiring minimal external components and simple wiring connections. This simplicity facilitates its use, particularly for beginners and hobbyists [36].

2. Real-time Sound Detection: With the ISD1820 sensor, sound detection is performed in real-time. The module continuously captures audio signals and provides a digital output signal when sound is detected. This real-time detection capability enables prompt action or initiation of sound classification processes based on the presence of sound [36].

3. Cost-effectiveness: The ISD1820 microphone sensor is an affordable option that is widely available in the market. Its cost-effectiveness makes it a practical choice for hobbyist projects and those with budget constraints [36].

4. Compatibility with the Raspberry Pi: The ISD1820 module is designed to be compatible with the Raspberry Pi, ensuring seamless integration with the GPIO pins of the Raspberry Pi. This compatibility simplifies the connection and communication between the sensor and the Raspberry Pi [36].

5. Versatility: While its primary function is sound detection, the ISD1820 microphone sensor offers additional functionalities such as voice recording and sound playback. This versatility allows for potential expansion of the sound classification project's capabilities if desired [36].

Considering its simplicity, real-time sound detection, affordability, compatibility with the Raspberry Pi, and versatility, the ISD1820 microphone sensor is considered a suitable choice for sound classification projects. It provides the necessary functionality to capture and process audio data, enabling accurate and timely sound classification while being accessible to users with varying levels of expertise [36].

3.4.3 Buzzer Device

A buzzer is an electronic device that produces an audible sound or tone when an electric current is applied to it .

Figure III 04: Buzzer Device

3.4.3.1 The principle of function of a buzzer

Involves the conversion of electrical energy into sound energy. Buzzer components usually consist of a coil and a diaphragm. When an electrical current passes through the coil, it creates a magnetic field that causes the diaphragm to vibrate. These vibrations produce sound waves, resulting in an audible sound or tone [37].

The frequency and intensity of the sound produced by the buzzer can be controlled by varying the electrical current applied to the coil. Buzzer components are designed to generate specific frequencies or tones, such as continuous buzz, beeping patterns, or musical notes, depending on the application [37].

In summary, the principle of function for a buzzer involves the electromagnetic vibration of a diaphragm when an electrical current passes through a coil, resulting in the production of sound waves and audible tones [37].

In Our project we used only one simple sound when the captured sound is abnormal.

3.4.4 LCD L1652

LCD (Liquid Crystal Display) is a flat-panel display technology that uses liquid crystals to produce visual output. It consists of multiple layers, including a liquid crystal layer sandwiched between two transparent electrodes and a backlight or sidelight source.

Figure III 05: LCD L5216

3.4.4.1 The principle of function of an LCD

The functioning principle of an LCD involves manipulating the light properties of liquid crystals to generate images or text. When an electric current is applied to the liquid crystal molecules, they align to control the passage of light. The liquid crystal layer acts as a light valve, selectively allowing or blocking light from the backlight source based on the electrical signals received [38].

LCD displays are known for their thin profile, low power consumption, and excellent image quality. They can display information in alphanumeric characters or graphical formats, making them suitable for various applications, including electronic devices, appliances, computer monitors, and signage [38].

The display content on an LCD can be updated by sending appropriate electrical signals to control the liquid crystal alignment. This enables the display to show different text, images, or graphics based on the input received [38].

In summary, an LCD operates by controlling the properties of liquid crystals to modulate the passage of light, resulting in the display of visual content. It offers a versatile and energy-efficient solution for displaying information in various electronic devices and applicationsv [38].

3.5 Realization:

3.5.1 Model Implementation to Raspberry Pi 4

To implement our model to the raspberry pi board we passed by this steps

- Copy the model to raspberry using USB.
- Load the saved model using this command “ model = joblib.load('model.pkl')”

3.5.2 Circuit Conception

To integrate the concept a real time sound classification system using the Raspberry Pi 4 board, a circuit needs to be constructed. The components involved in this setup include the Raspberry Pi, ISD1820 sound module, buzzer, and LCD display. The Raspberry Pi is the central processing unit where our AI model is implemented and connects to the ISD1820 module for sound input. The buzzer is connected to the Raspberry Pi to provide audible alerts when abnormal sounds are detected. The LCD display is used to visually indicate the status of the sound classification. Proper connections between the Raspberry Pi and the components, such as power, ground, and data pins, need to be established.

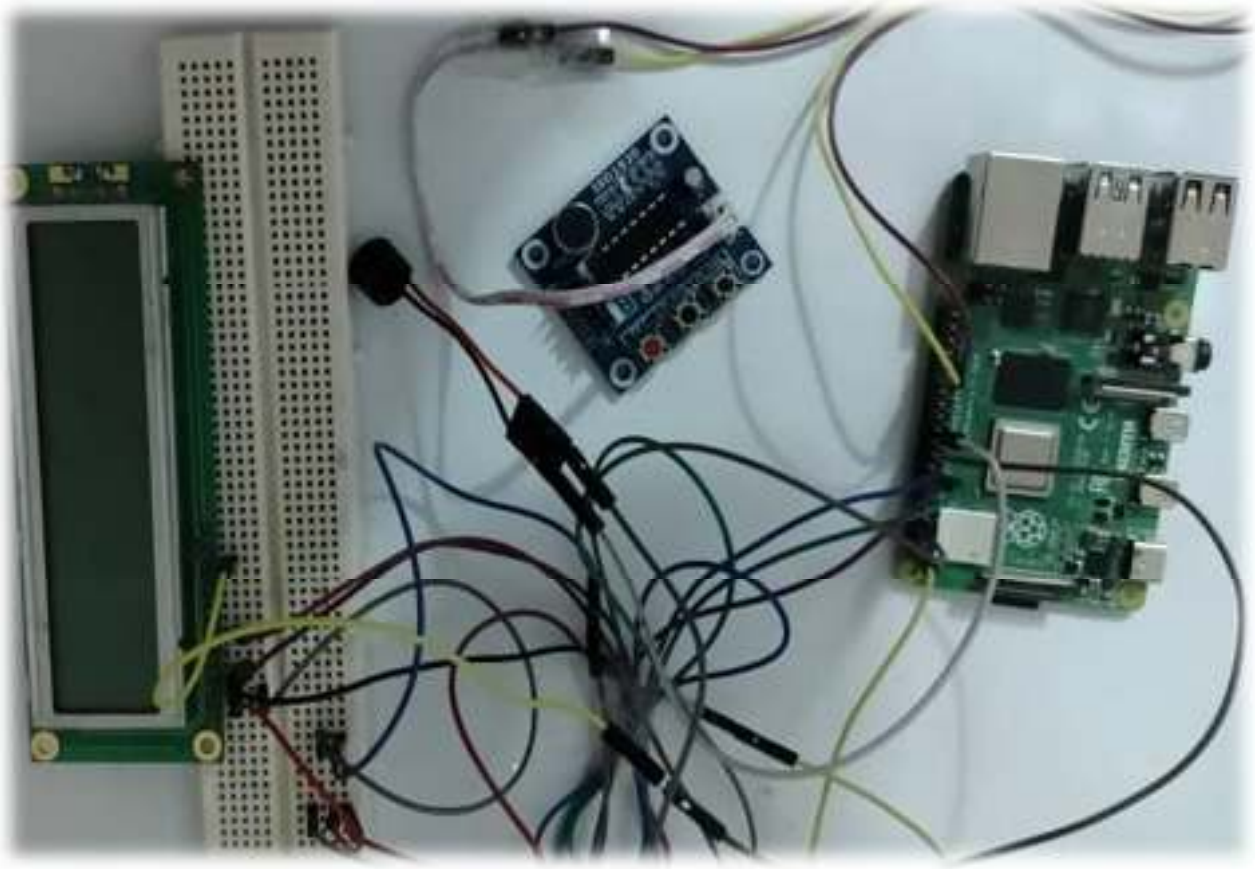


Figure III 06 : Circuit Assembly

3.5.3.1 Circuit Assembly Explication

Raspberry Pi 4 Connections:

1. Connect the 5V pin on the Raspberry Pi 4 to the VCC pin on the ISD1820.
2. Connect a GND pin on the Raspberry Pi 4 to the GND pin on the ISD1820.
3. Connect a GPIO pin on the Raspberry Pi 4 to the DATA pin on the ISD1820.
4. Connect the GPIO pin on the Raspberry Pi 4 to the positive terminal of the buzzer module.
5. Connect the negative terminal of the buzzer module to a GND pin on the Raspberry Pi 4.

ISD1820 Connections:

1. Connect the REC pin on the ISD1820 to GND to enable recording mode.
2. Connect the SP+ and SP- pins on the ISD1820 to a speaker or an amplifier for sound output.

LCD Connections:

1. Connect the VCC pin on the LCD to the 5V pin on the Raspberry Pi 4.
2. Connect the GND pin on the LCD to the GND pin on the Raspberry Pi 4.
3. Connect the SDA pin on the LCD to the SDA GPIO pin on the Raspberry Pi 4.
4. Connect the SCL pin on the LCD to the SCL GPIO pin on the Raspberry Pi 4.

3.5.4 Circuit Programming

Here the code that we used to connect raspberry pi 4 board with the buzzer, LCD and ISD1820 sensor:

```
import RPi.GPIO as GPIO
import time
import joblib
import Adafruit_CharLCD as LCD
# Set up GPIO pins
ISD_PIN = 17
BUZZER_PIN = 27
# Set up LCD pins
LCD_RS = 26
LCD_EN = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11
LCD_COLUMNS = 16
LCD_ROWS = 2
# Load the trained model
model = joblib.load("model.pkl")

# Set up GPIO mode and pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(ISD_PIN, GPIO.IN)
GPIO.setup(BUZZER_PIN, GPIO.OUT)
# Set up LCD
```

```
lcd = LCD.Adafruit_CharLCD(
    LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7,
    LCD_COLUMNS, LCD_ROWS
)
# Function to classify sound and control buzzer
def classify_sound():
    while True:
        sound = GPIO.input(ISD_PIN)
            # Perform sound classification using the loaded model
        prediction = model.predict([sound])[0]
            # Control the buzzer based on the classification result
        if prediction == 0:
            # Normal sound
            GPIO.output(BUZZER_PIN, GPIO.LOW)
            lcd.clear()
            lcd.message('Normal')
        else:
            # Abnormal sound
            GPIO.output(BUZZER_PIN, GPIO.HIGH)
            lcd.clear()
            lcd.message('Not Normal')
                time.sleep(0.1)  # Adjust sleep duration as needed
try:
    # Run the sound classification loop
    classify_sound()

except KeyboardInterrupt:
    # Clean up GPIO pins on keyboard interrupt
    GPIO.cleanup()
```

- Code explication :

  - Import the required libraries: RPi.GPIO, time, joblib, and Adafruit_CharLCD.
  - Set up the GPIO pins for the ISD1820 and buzzer.
  - Set up the LCD pins for the Adafruit_CharLCD.
  - Load the trained model using joblib.
  - Set up the GPIO mode and pins using GPIO.setmode and GPIO.setup.
  - Set up the LCD using Adafruit_CharLCD.
  - Define a function to classify sound and control the buzzer.
  - In a loop, read the sound input from the ISD1820 pin.

- Use the loaded model to classify the sound.
- Control the buzzer and display the result on the LCD based on the classification.
- Add a small delay using time.sleep.
- Run the sound classification loop.

Conclusion:

In this chapter, we have successfully implemented a real-time sound classification system using Raspberry Pi. We began by conceiving and training our sound classification model, ensuring its accuracy and robustness. We then moved on to setting up the necessary hardware components, including the Raspberry Pi as the central processing unit, the ISD1820 microphone sensor for sound input, the LCD display for visual feedback, and the buzzer for audible feedback. We configured the software environment, installing libraries such as RPi.GPIO and Adafruit_CharLCD, and loaded the trained model into the Raspberry Pi.

The heart of our realization phase was the real-time sound classification process. We captured and preprocessed audio data, extracting relevant features and normalizing the data for optimal performance. We utilized the loaded model to classify the sound, distinguishing between normal and abnormal sounds. The LCD display provided visual feedback, displaying "Normal" or "Not Normal" based on the classification result, and the buzzer produced audible feedback when abnormal sounds were detected.

We thoroughly tested and evaluated our system's performance using different sound samples, employing evaluation metrics to measure accuracy and efficiency. The results demonstrated the effectiveness of our real-time sound classification system in accurately identifying abnormal sounds.

GENERAL CONCLUSION & PERSPECTIVE

GENERAL CONCLUSION & PERSPECTIVES

The poultry industry is a critical sector in the global food supply chain, providing a significant source of protein through the production of poultry products. Ensuring the health and well-being of poultry flocks is essential for maintaining high-quality products and maximizing productivity. Our project, focused on real-time sound classification for poultry health monitoring, has the potential to make a significant impact on the poultry industry.

By developing a sound classification system capable of distinguishing between normal and abnormal sounds in poultry, we provide a non-invasive and efficient method for monitoring the health and welfare of the birds. Abnormal sounds, such as coughing, wheezing, or distress calls, can be indicative of underlying health issues or environmental stressors that require immediate attention. Through the use of machine learning algorithms and the integration of our system with Raspberry Pi and associated hardware components, we have created a practical and accessible solution for poultry farmers and producers.

The implementation of our system in the poultry industry can bring several benefits. Firstly, it enables early detection of health problems in poultry flocks, allowing for timely intervention and treatment. This can prevent the spread of diseases, minimize mortality rates, and improve overall flock health and productivity. Additionally, the system's real-time monitoring capabilities provide continuous and comprehensive data on the well-being of the birds, enabling proactive measures to optimize their living conditions and reduce stress factors.

The impact of our project on the poultry industry goes beyond health monitoring. By automating the sound classification process, poultry farmers can save time and resources that would otherwise be spent on manual monitoring or periodic veterinary inspections. The system's ability to provide immediate feedback through visual displays and audible alerts enhances the efficiency of farm management and decision-making. It empowers poultry farmers with actionable insights and supports their efforts in maintaining healthy and thriving flocks.

Furthermore, our project promotes the adoption of advanced technologies in the poultry industry, fostering innovation and progress. By demonstrating the feasibility and benefits of sound classification for poultry health monitoring, we encourage the industry to explore and embrace similar technological solutions. This can lead to further advancements in animal welfare practices, disease prevention, and overall farm efficiency.

GENERAL CONCLUSION & PERSPECTIVES

Like every research project, there are some perspectives we think that it is an obligation to be integrated in future solutions:

1. Model Improvement: Continuously refining and optimizing the sound classification model can lead to higher accuracy and better performance. This can involve exploring different machine learning algorithms, feature engineering techniques, and data augmentation methods to enhance the model's ability to classify poultry sounds accurately.
2. Feature Extraction: Experimenting with different audio feature extraction techniques can provide additional insights into the characteristics of poultry sounds that are indicative of specific health conditions. Exploring advanced signal processing algorithms or deep learning architectures for feature extraction may further improve the system's performance.
3. Multi-Class Classification: Expanding the classification system to include multiple classes of poultry sounds can enable the detection of a wider range of health conditions or behaviors. This could involve training the model on a more diverse dataset that covers various sound patterns associated with different diseases or stressors.
4. Cloud Integration: Integrating the system with cloud-based services can offer additional capabilities, such as centralized data storage, advanced analytics, and remote monitoring. Cloud integration can enable access to historical data, collaborative analysis, and the ability to scale the system across multiple locations.
5. User Interface Enhancements: Improving the user interface of the system can enhance usability and accessibility. This can involve developing a graphical user interface (GUI) or a web-based dashboard that provides real-time visualizations, historical data analysis, and customizable alerts for poultry farmers.
6. Integration with Other Sensors: Combining sound classification with other sensor data, such as temperature, humidity, or motion sensors, can provide a more comprehensive view of poultry health. Integrating multiple sensor inputs can enable more accurate and robust health monitoring, detecting early signs of illness or stress from different sources.

7. Data Analysis and Insights: Investing in data analysis techniques, such as machine learning algorithms or statistical models, can help uncover hidden patterns and correlations within the collected sound and sensor data. Extracting valuable insights can provide deeper understanding of poultry health trends, leading to more informed decision-making for farmers.

In conclusion, our project's impact on the poultry industry lies in its ability to revolutionize poultry health monitoring through real-time sound classification. By leveraging machine learning, Raspberry Pi, and associated hardware components, we provide a practical and effective solution for early detection of health issues in poultry flocks. This can lead to improved flock health, increased productivity, and enhanced farm management practices. Through our project, we contribute to the ongoing advancements in the poultry industry and promote the adoption of technology for sustainable and responsible poultry farming.

ملخص

تواجه تربية الدواجن تحديات كبيرة في الحفاظ على صحة الطيور ورفاهيتها، خاصة مع ظهور أمراض جديدة ومعقدة. يعد التنبؤ بهذه الأمراض وتشخيصها أمرا بالغ الأهمية للإدارة والسيطرة الفعالة. في السنوات الأخيرة، كان هناك اهتمام متزايد بتطوير أنظمة ذكية يمكن أن تساعد في التنبؤ بحدوث هذه الأمراض باستخدام التصنيف السليم.

تقدم هذه الدراسة تطوير وتصميم نظام ذكي للتنبؤ بأمراض الدواجن يعتمد فقط على التصنيف السليم. يقوم النظام بتحليل التسجيلات الصوتية لأصوات الدواجن للكشف عن الأصوات غير الطبيعية التي قد تشير إلى وجود المرض. ويهدف النظام أيضا إلى تقديم توصيات بشأن تدابير المكافحة المناسبة التي يتعين اتخاذها في حالة تفشي المرض.

يستخدم النظام المقترح مزيجا من معالجة الإشارات واستخراج الميزات وخوارزميات التصنيف للتنبؤ باحتمالية تفشي المرض بناء على التسجيلات الصوتية. تم تقييم أداء النظام باستخدام البيانات التي تم جمعها من مزرعة دواجن، وأظهرت النتائج أن النظام حقق دقة عالية في التنبؤ بحدوث الأمراض باستخدام التصنيف السليم.

بشكل عام، يمكن أن يؤدي تطوير نظام ذكي للتنبؤ بأمراض الدواجن بناء على التصنيف السليم إلى تحسين إدارة الأمراض ومكافحتها بشكل كبير في تربية الدواجن. ويمكن للنظام أن يساعد في الكشف المبكر عن الأمراض، والحد من خطر انتقال العدوى وتقليل التأثير على إنتاج الدواجن.

الكلمات المفتاحية: الذكاء الاصطناعي، معالجة الإشارات، التعلم الآلي، الصوت، معالجة الإشارات الصوتية.

Abstract

The poultry industry faces significant challenges in maintaining the health and welfare of birds, with the emergence of new and complex diseases. Predicting and diagnosing these diseases is crucial for effective management and control. In recent years, there has been a growing interest in the development of intelligent systems that can aid in predicting the occurrence of poultry diseases using sound classification.

This study presents the development and design of an intelligent system for predicting poultry diseases based solely on sound classification. The system analyzes audio recordings of poultry vocalizations to detect abnormal sounds that may indicate the presence of disease. The system is also designed to provide recommendations on the appropriate control measures to be taken in the event of an outbreak.

The proposed system uses a combination of signal processing, feature extraction, and classification algorithms to predict the likelihood of a disease outbreak based on sound recordings. The system's performance was evaluated using data collected from a poultry farm, and the results show that the system achieved high accuracy in predicting the occurrence of diseases using sound classification.

Overall, the development of an intelligent system for predicting poultry diseases based on sound classification can significantly improve disease management and control in the poultry industry. The system can assist in early detection of diseases, reducing the risk of transmission and minimizing the impact on poultry production.

Keywords: Artificial intelligence, signal processing, Machine Learning, voice, Sound signals processing

BIBLIOGAPHY

Bibliography

1. New Horizons in Life Science (pp.1-13)Publisher: VITAL BIOTECH PUBLICATION
2. The impacts of livestock diseases and their control on growth and development Processes that are pro-poor Philos Trans R Soc Lond B Biol Sci. 2009 Sep 27 .
3. Kumari, M. and Dhawal, K. 2021. Application of Artificial Intelligence (AI) in Animal Husbandry. Vigyan Varta 2(2): 27-29
4. https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/ 03/04/2023
5. Sampath Boopathi, Sri Harsha Arigela, Ramakrishnan Raman, C. Indhumathi, V. Kavitha, Bhuvan Chandra Bhatt, "Prominent Rule Control-based Internet of Things: Poultry Farm Management System", *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, pp.1-6, 2022.
6. The New International Webster's Comprehensive Dictionary of the English Language, Encyclopedic Edition.
7. https://www.unr.edu/cse/undergraduates/prospective-students/what-are-intelligent-systems 03/04/2023
8. https://www.javatpoint.com/types-of-artificial-intelligence 03/04/2023
9. https://www.sas.com/AI-Process 03/04/2023
10. https://devopedia.org/artificial-intelligence 01/06/2023
11. Pennachin, C.; Goertzel, B. (2007). "Contemporary Approaches to Artificial General Intelligence". Artificial General Intelligence. Cognitive Technologies
12. https://www.softwaretestinghelp.com/what-is-artificial-intelligence/ 01/06/2023
13. Alpaydin, Ethem (2010). Introduction to Machine Learning. MIT Press. p. 9. ISBN 978-0-262-01243-0. Archived
14. https://www.researchgate.net/publication/355050755_Applications_of_Deep_Learning_Techniques_for_Pedestrian_Detection_in_Smart_Environments_A_Comprehensive_Study
15. https://www.researchgate.net/publication/328285467_Deep_Learning_A_Review
16. https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide 01/06/2023
17. Professor Frank Kelly CBE FRS "The online information environment"
18. https://www.javatpoint.com/artificial-intelligence-in-agriculture 03/06/2023
19. https://www.sciencedirect.com/science/article/pii/S2589721722000095 03/06/2023
20. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7936489/ 03/06/2023
21. https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf
22. Kelley, Troy Dickerson, Kelly A Review of Artificial Intelligence (AI) Algorithms for Sound Classification: Implications for Human-Robot Interaction (HRI)
23. Anastassiou, D. (2001). "Genomic signal processing". IEEE Signal Processing Magazine

24. Reynolds, John M. (2011). An Introduction to Applied and Environmental Geophysics
25. https://www.allaboutcircuits.com/technical-articles/an-introduction-to-digital-signal-processing/ 01/06/2023
26. "Machine Audition: Principles, Algorithms and Systems" 03/06/2023
27. towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step 03/06/2023
28. https://www.sciencedirect.com/science/article/pii/S2666827021001031
29. https://arxiv.org/abs/1710.09412v2
30. https://arxiv.org/abs/1904.08779
31. T. Viarbitskaya and A. Dobrucki, "Audio processing with using Python language science libraries," *2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Poznan, Poland, 2018, pp. 350-354, doi: 10.23919/SPA.2018.8563430.
32. https://docs.python.org/fr/3/library/ 03/06/2023
33. https://ledatascientist.com/google-colab-le-guide-ultime/ 03/06/2023
34. https://www.raspberrypi.com/documentation/computers/raspberry-pi.html 03/06/2023
35. https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/ 03/06/2023
36. ISD1820 Voice Recorder Module User GuideRev 1.0, Oct 2012
37. https://www.elprocus.com/buzzer-working-applications/ 03/06/2023
38. Tarun Agrwal construction and principle of LCD display , Oct 2016