

Multiclass Audio Classification

A major project report submitted in partial fulfillment of the
requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Muskan (201319)

Saloni (201342)

Under the guidance & supervision of

Dr. Ruchi Verma



**Department of Computer Science & Engineering
and Information Technology**

Jaypee University of Information Technology,

Waknaghat, Solan - 173234 (India)

Table of Contents

Certificate	i
Declaration	ii
Acknowledgement	iii
Abstract	iv
Chapter 01	
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Significance and Motivation	6
1.5 Organization of Project Report	7
Chapter 02	
2.1 Overview of Relevant Literature	10
2.2 Key Gaps in the Literature Survey	22
Chapter 03	
3.1 Requirements and Analysis	25
3.2 Project Design and Architecture	26
3.3 Dataset Preparation	27
3.4 Implementation	31
3.5 Key Challenges	40
Chapter 04	
4.1 Testing Strategy	44
4.2 Test Cases and Outcomes	45
Chapter 05	
5.1 Results	48
Chapter 06	
6.1 Conclusion	51
6.2 Future Scope	53
References	55

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “Multiclass Audio Classification” in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering and submitted to the Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Muskan, 201319” and “Saloni, 201342” during the period from August 2023 to May 2024 under the supervision of Dr. Ruchi Verma, Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat.

Muskan
(201319)

Saloni
(201342)

The above statement made is correct to the best of my knowledge.

(Dr. Ruchi Verma)

Assistant Professor (SG)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled '**Multiclass Audio Classification**' in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Wagnaghat is an authentic record of our own work carried out over the period from August 2023 to May 2024 under the supervision of **Dr. Ruchi Verma** (Assistant Professor - SG, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)

Student Name: Saloni

Muskan

Roll No.: 201342

201319

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

(Supervisor Signature with Date)

Supervisor Name: Dr. Ruchi Verma

Designation: Assistant Professor (SG)

Department: CSE & IT

Dated:

ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for his divine blessings that makes it possible for us to complete the project work successfully.

We are really grateful and wish our profound indebtedness to our Supervisor **Dr. Ruchi Verma** (Assistant Professor (SG), Department of CSE Jaypee University of Information Technology, Wakhnaghat). Deep Knowledge & keen interest of our supervisor in the field of “**Multiclass Audio Classification**” helped us to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages has made it possible for us to complete this project.

We would like to express our heartiest gratitude to **Dr. Ruchi Verma**(Assistant Professor (SG), Department of CSE), for her kind help to finish our project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Muskan (201319)

Saloni (201342)

List of Figures

S. No.	Figure Name	Page No.
1.	Project Flow	26
2.	Dataset Images	28-30
3.	Implementation Images	31-40
4.	Result Images	48-49

List of Tables

S. No.	Table Name	Page No.
1.	Literature Survey	20-22

List of Graphs

S. No.	Graph Name	Page No.
1.	Count of records	30

List of Abbreviations

ML	Machine learning
EDA	Exploratory data analysis
DL	Deep Learning
ANN	Artificial Neural Network
MFCC	Mel-frequency cepstral coefficients
OS	Operating System
KNN	K-Nearest Neighbors Algorithm
CNN	Convolutional Neural Network
SVM	Support Vector Machines
LSTM	Long Short-Term Memory

ABSTRACT

The dynamic field of audio classification, which intersects machine learning, studies methods for automatically classifying a variety of audio data. This paper explores important techniques, with a focus on real-time audio processing with PyAudio, the Librosa library, and Mel-frequency cepstral coefficients (MFCCs). MFCCs are powerful feature representations that effectively capture the subtleties and spectrum envelope of audio signals that are essential for categorization. A mainstay of audio analysis, Librosa not only makes MFCC extraction easier, but it also provides an extensive toolkit that includes beat tracking, pitch estimation, and tempo analysis.

In addition to this, PyAudio shows up as an essential tool that makes real-time audio recording and playing possible. It is extremely useful in situations like sound event detection and voice recognition because of its adaptability, which fills the gap between model training and real-world applications. The combination of PyAudio, Librosa, and MFCCs creates a strong basis for improving audio categorization skills.

This research is essential because audio classification is used in speech emotion analysis, music genre identification, and environmental sound classification. When these methods work together, real-time adaptability and correct categorization are guaranteed, which is crucial for implementing classification systems in dynamic, real-world situations.

Complementing these feature extraction techniques, PyAudio emerges as a crucial tool for real-time audio processing. Its ability to record and play audio in real time facilitates the development of audio classification systems that can be seamlessly integrated into practical applications. As the field continues to evolve, the synergy between MFCCs, Librosa, and PyAudio remains instrumental in advancing the capabilities of audio classification, unlocking new possibilities for understanding and categorizing the richness of audio data in various domains.

To sum up, the combination of MFCCs, Librosa, and PyAudio is a potent combination of real-time capabilities, feature extraction, and analytic tools that is reshaping the audio categorization field. This synergy not only opens up new avenues for comprehending the complexities of audio data but also improves the usefulness of categorization systems in a variety of contexts.

Chapter 1: Introduction

1.1 Introduction

The most important tasks in machine learning is audio classification, which means automatic classification of variety of audio data into different classifications. This interdisciplinary field joins methods from Deep Learning, Machine Learning, and EDA to interpret and classifies the heterogeneous audio signal environment.

MFCCs are the foundation for efficient multi class audio classification. The coefficients will capture the spectrum of an audio source, which will make it effective for feature representation. It is critical to have this precise timbre representation of sound in order to discern subtleties in audio data. One of the main resources for audio analysis is Librosa library as this library provides a complete toolkit for MFCC extraction and make feature extraction for classification tasks easier.

Librosa library offers a range of techniques for deriving valuable features from audio data as it is an indispensable tool for audio analysis. In addition to MFCCs, Librosa also has features for beat tracking, pitch estimation, and tempo analysis, which makes it a valuable tool for practitioners and academics in the audio categorization field. Its thorough documentation and user-friendly interface both contribute to its widespread field adoption. Real-time capabilities are necessary for the practical applications of audio categorization, even though feature extraction is crucial. This need is met by the Python package PyAudio, which makes real-time audio recording and playback possible. Because of its flexibility, PyAudio is a great tool for developers who want to close the gap between training models and putting them into practise.

In the dynamic field of audio classification, which resides at the nexus of machine learning and audio signal processing, reliable methods are essential for classifying a wide range of audio data. One notable achievement in this endeavour is the use of Mel-frequency cepstral coefficients (MFCCs) as feature representations, made possible by the Librosa package. Beyond MFCCs, Librosa's extensive toolset offers a wide range of features for in-depth audio analysis.

With regard to the methods of feature extraction, the library PyAudio becomes an instrument for processing audio or voice in real application. The real-time recording and running of audio capabilities make it easier to develop audio classification systems that work well with real-world applications. The integration of MFCCs, Librosa, and PyAudio continues to be important in expanding the field's potential for audio classification and opening up new avenues for comprehending and classifying the diversity of audio data across different domains.

Audio classification is one of the most required and essential work in the fast developing field of machine learning, with the aim of classifying automatically different type of audio data in different categories. This multiplied field uses a mix of machine learning, deep learning, and exploratory data analysis methods for identifying finding, and categorise the difficult network of signals that is present in the diverse audio environment.

The most important aspect of an effective audio category are the Mel-frequency cepstral coefficients. An efficient feature representation is provided by the coefficients by capturing the source audio present on the envelope spectrum. The most accurate representation is important and required for recognizing the small details in the data audio which makes it possible and easy for the classification. As a basic resource for the analysis of audio, the library that we have used is librosa stands out for providing an accessible toolkit that makes the extraction using MFCCs simple, easy and more simplifying for classification jobs easier.

The importance of Librosa in audio analysis is much more beyond MFCCs, with features available for such as tracking of beat, estimation of pitch, and analysis of tempo. This ability of adaptability makes Librosa an a very valuable tool for faculty as well as scholars working in the that field of audio categorization. The widespread use of this is the in large part to because of user friendly and extensive and easily available documentation, making it possible for the easy conversion from documentation to application.

It is important for the feature extraction, the skills in the real time application is also required for original audio classification and further implementation. PyAudio, the library of python that further looks after the real-time audio recorder system, addresses the required demand. PyAudio's flexibility intrinsic in nature makes it an available choice for developers and the

scholars looking to fill out the gap between model training and real time application.

In the fast and dynamic development of audio classification, which currently is present at the edge of machine learning and signal processing of audio, the integration of depending methods and ways becomes crucial for categorising a wide set of data for the audio. The use of both the of Mfcc and Librosa, with the py audio's augmented real time capabilities, is a quite achievement in this endeavor and noteworthy. This integration has been important in understanding the full use and methods of audio classification, also while opening up the pathways that are new for the categorization and relevant interpretation of the vast diversity of audio data across different domains.

1.2 Problem Statement

In several practical applications of audio processing, such as sound event detection and voice recognition, it is crucial to build a real-time classification system that is both efficient and effective. This is a multi-step procedure that starts with preprocessing different types of audio data and ends with building and testing a machine learning model. The fundamental elements of developing such a system will be discussed in this article, along with subjects like feature extraction, data preprocessing, model training, and real-time implementation.

1. Preprocessing Diverse Audio Data:

Preprocessing a variety of audio data to guarantee format consistency is the first step in creating a strong audio categorization system. In order to fix problems like inconsistent sampling rates, background noise, and signal distortions, the raw audio files must be cleaned and standardised in this phase. We provide the groundwork for precise feature extraction and model training by standardising the data.

2. Extracting MFCC Features:

After preprocessing the audio data, extracting Mel Frequency Cepstral Coefficients (MFCC) characteristics is an important next step. Because they are so good at capturing the spectrum features of audio signals, MFCCs are frequently utilised in audio signal processing. These characteristics provide a condensed representation of the audio data that is useful for classification tasks, and they act as inputs to the machine learning model.

3. Training a Machine Learning Model:

After extracting the MFCC characteristics, a machine learning model must be trained. The particular requirements of the application determine which model should be used. For audio classification tasks, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are popular options. Using labelled data, the model is trained to distinguish between various audio categories using the features that are retrieved.

4. Evaluating Model Performance:

It is essential to assess the trained model's performance to guarantee its efficacy. Among other metrics, class-specific accuracy offers information about how well the model can categorise examples from each category. This assessment stage ensures that the model is reliable in practical situations by pointing out areas that need to be improved and adjusted.

5. Developing a Real-Time Audio Classification System with PyAudio:

PyAudio integration is necessary for the audio classification system to be useful for real-time applications. A Python library called PyAudio offers bindings for PortAudio, making audio input and output simple. The proposed system can process and classify audio streams in real-time by integrating PyAudio, which makes it appropriate for applications like sound event detection and speech recognition.

Developing a real-time audio categorization system requires a thorough methodology, ranging from preparing heterogeneous data to putting a machine learning model into practise and guaranteeing real-time performance. Developers can build a reliable system that can correctly identify audio input in real-world applications by following these steps. Apart from this the pyaudio also increases the efficiency of the system in numerous fields, the deception like sound vent and recognition of voice, making it an advantage for the audio processing jobs.

1.3 Objectives

Accurate class labelling of varied audio samples is crucial in a variety of applications ranged from speech recognition to sound event detection. The purpose of this article is to describe the creation of a robust audio classification model utilising Mel Frequency Cepstral Coefficients (MFCCs) collected with Librosa and PyAudio. The priority is to overcome data preprocessing issues in order to ensure the model's accuracy in real-world circumstances.

1. Data Preprocessing Challenges:

Various sampling rates, background noise, and pitch discrepancies are all common issues with diverse audio samples. Addressing these issues is critical for developing a credible model. To provide a uniform and clean dataset, data preprocessing entails standardising the format, controlling pitch fluctuations, and minimising noise.

2. Extracting MFCC Features with Librosa and PyAudio:

Librosa is a sophisticated Python library for audio and music analysis, making it an excellent choice for MFCC feature extraction. PyAudio, on the other hand, makes it easier to incorporate real-time audio processing into the model. The combination of Librosa and PyAudio provides precise extraction of MFCC characteristics, capturing crucial spectral qualities while minimising the impact of noise and pitch fluctuations in audio inputs.

3. Model Precision Assessment:

Following the extraction of MFCC characteristics, a machine learning model for audio categorization is trained. A crucial evaluation criterion is the model's precision in categorising audio across different classes. This evaluation verifies that the model distinguishes well between different audio samples, making it acceptable for real applications where precision is critical.

4. Handling Augmentation and Class Imbalance:

Augmentation techniques can be added to the dataset to improve the model's robustness. To expose the model to a broader range of events, augmentation entails introducing modifications in the training data, such as time stretching or adding background noise. Addressing class imbalance also guarantees that the model does not favour one class over another, resulting in a more balanced and accurate classification.

5. Exploring Advanced Methods:

Exploring additional methods can further develop the model beyond basic augmentation and fixing class imbalance. Transfer learning, ensemble models, and adding attention mechanisms

can all help to increase performance, particularly in complicated audio categorization problems. Experimenting with these advanced methods enables the model to be tailored to individual application requirements.

Conclusion:

Creating an accurate audio classification model necessitates a multifaceted approach that begins with addressing data preprocessing issues. Using the capabilities of Librosa and PyAudio, extracting MFCC features becomes a robust process, capturing vital spectral attributes while minimising noise and pitch influences. Model precision assessment, in conjunction with techniques such as augmentation and handling class imbalance, ensures the model's suitability for practical applications. Exploring advanced methods refines the model further, making it a powerful tool for accurate class labelling of diverse audio samples in a variety of real-world scenarios.

1.4 Significance And Motivation of The Report Work

This study on audio categorization highlights the use of Librosa, PyAudio, and Mel-frequency cepstral coefficients (MFCCs) to handle real-world difficulties and uncover creative applications across several fields. It is significant and motivational.

1. Practical Applicability:

The purpose of the report is to close the knowledge gap between theoretical developments in audio classification and real-world implementation. The report shows that the techniques examined are not only reliable in research environments but also easily transferable to real-world scenarios including music genre identification, speech recognition, and sound event detection by concentrating on MFCCs, Librosa, and PyAudio.

2. Enhanced Accuracy and Efficiency:

Because MFCCs are good at capturing spectral characteristics, using them helps to increase accuracy in audio classification tasks. Beyond MFCC extraction, Librosa's extensive toolkit offers a variety of audio analysis methods to further improve efficiency. This combination makes sure that the various audio data are accurately and consistently

categorized using the suggested ways.

3. Multidisciplinary Relevance:

Because MFCCs are good at capturing spectral characteristics, using them helps to increase accuracy in audio classification tasks. Beyond MFCC extraction, Librosa's extensive toolkit offers a variety of audio analysis methods to further improve efficiency. This combination makes sure that the various audio data are accurately and consistently categorized using the suggested ways.

4. Real-time Adaptability:

The practical requirement for real-time capabilities in audio categorization systems is addressed by PyAudio's inclusion. In applications where prompt answers are critical, such as voice-activated assistants and surveillance systems, the research recognises the growing need for systems that can adapt and identify audio data on the fly.

5. Innovation and Future Research Directions:

Through exploring the interplay among MFCCs, Librosa, and PyAudio, the report promotes creativity in the audio categorization domain. It also serves as a basis for other research endeavors, including investigating cutting-edge approaches, tackling new problems, and modifying these approaches to fit changing technological environments.

1.5 Organization of Project Report

We have organized our report in the following parts:-

1) Certificate, Declaration, Acknowledgement, Abstract

This section includes an official certification affirming the authenticity of the work, typically provided by an academic institution or supervisory authority. The declaration is a formal statement wherein the author asserts the originality of the work and adherence to ethical standards, ensuring the absence of plagiarism. In this section, the author expresses gratitude to individuals, institutions, or organizations that contributed to the

project, offering a recognition of their support. The abstract gives a succinct summary of the full study, encompassing important objectives, methodology, and outcomes and providing readers with a rapid overview.

2) Table of contents

The table of contents provides a guide for readers to navigate through different sections and chapters by outlining the structure of the report. These sections list all tables and figures in the report, making it easier for readers to find relevant visual or tabular content. A comprehensive list of abbreviations used in the study, ensuring clarity and comprehension for readers who come across specialised language.

3) Chapter 1 : Introduction

The purpose of this chapter is to provide some background to the research in question and set the framework within which it has been carried out.

4) Chapter 2 : Literature Survey

In this part, it provides an extensive discussion on earlier studies related to the research subject as well as their approaches and gaps.

5) Chapter 3: System Development

Detailing the creation or enhancement of the system under study, this chapter covers methodologies, tools, and technologies employed in the system's construction.

6) Chapter 4: Testing

Focused on the verification and validation of the developed system, this chapter delves into testing methodologies, criteria for success, and the overall robustness of the system.

7) Chapter 5 : Results and Evaluation

Presenting the outcomes of the testing phase, this chapter analyzes the results, interprets data, and assesses the system's performance against predefined criteria.

8) Chapter 6: Conclusions and future scope

Summarizing key findings, this section draws conclusions from the research, highlighting implications and contributions to the field. Discussing potential avenues for future research, this section outlines areas that could benefit from further exploration or development.

9) List of Publications, References, Appendix

If applicable, this section enumerates any publications resulting from the research, showcasing the dissemination of findings.

Chapter 2: Literature Survey

2.1 Overview of relevant literature

[1] This study looks into audio classification using machine learning (ML) techniques and assesses how well Decision Tree, RandomForest, K-Nearest Neighbours (KNN), and Logistic Regression work. The study, which was published in 2023, has an astounding 95% accuracy rate. Nonetheless, a number of restrictions are noted, providing insight into potential areas for additional study and development.

The study uses a variety of machine learning (ML) techniques to categorise audio data: KNN uses proximity-based categorization, RandomForest and Decision Tree models capture intricate decision boundaries, and Logistic Regression offers a linear model. The 95% accuracy rate indicates how well these techniques work to distinguish across various audio genres.

Notwithstanding the praiseworthy outcomes, the study presents noteworthy constraints that affect the applicability and thoroughness of the conclusions. First off, the model's capacity to handle a variety of audio samples may be limited by the tiny size of the dataset that was used. The model's applicability to more complex and diverse audio genres may be limited by its emphasis on binary genre classification.

Limited feature details are also highlighted by the research as a limitation, suggesting that feature richness and choice may not be all-inclusive. Concerns concerning the interpretability and applicability of the selected features are raised by the lack of a clear and comprehensive expert input during the feature selection process.

Larger datasets are also emphasised in the research, as it is acknowledged that a more comprehensive and diverse dataset may strengthen the model's resilience and increase its performance in a variety of genres. The demand for larger datasets is consistent with a more general trend in machine learning research, which holds that rich and varied data sets are

essential for training well-generalizable models.

In summary, the study openly discusses its shortcomings even though it uses machine learning techniques to classify audio with a high accuracy rate and noticeable results. These comprise the limitations brought about by a tiny dataset, an emphasis on binary genres, a lack of distinct expert input, and minimal feature details. Acknowledging these limits creates opportunities for future work to solve these restrictions and increase audio classification.

[2]DNN enhanced with a Classifier Attention Mechanism, a novel method for audio classification is presented in the paper . The study's 85.99% classification accuracy for audio data demonstrates how successful the suggested approach is.

The use of a DNN, a potent class of neural networks renowned for its capacity to extract complex patterns and representations from data, is the main methodology used in the study. The new feature is the addition of a Classifier Attention Mechanism, which improves the model's discriminatory power by concentrating on particular elements when classifying. efficacy of the suggested model.

Although the accuracy of 85.99% was achieved with considerable success, there are certain limits in the research. One such drawback is the lack of a thorough comparison with current techniques. The suggested DNN is not fully benchmarked in the research against other cutting-edge models or well-known methods for audio classification. This omission begs the question of how the unique approach's performance, efficiency, and universal applicability stack up against those of existing methods.

The report also points out that using a single dataset for evaluation is a constraint. The lack of a more comprehensive analysis across many datasets reduces the generalizability of the concept. Due to its dependence on a single dataset, the suggested method's performance in different audio domains may not be well understood, which could limit its usefulness in practical settings.

In conclusion, the study makes a substantial contribution to the field of audio categorization by introducing a DNN with a Classifier Attention Mechanism and obtaining an 85.99% accuracy

rate. Notwithstanding, the constraints, such as the absence of an elaborate comparative analysis with extant methodologies and the dependence on a particular dataset for assessment, indicate potential directions for future investigation. In order to improve the suggested model's robustness and practicality, further study may entail a more thorough evaluation of its performance against recognised benchmarks and an investigation of its results over a variety of datasets.

[3]The study, "Music Genre Classification using MFCC, SVM," explores the use of Support Vector Machines (SVM) in conjunction with Mel-frequency cepstral coefficients (MFCC) to classify different musical genres. The remarkable accuracy rate of 95% is made possible by the combination of the powerful classification powers of SVM with the feature extraction capabilities of MFCC. Nevertheless, the study admits several limitations, primarily related to the specifics of feature selection and the applicability of the model to a variety of musical genres.

The study uses a two-pronged strategy, using SVM for classification and MFCC for feature extraction. The basic spectrum properties of audio signals are captured by MFCC, a potent technique in audio signal processing that yields a concise but informative representation. MFCC characteristics are retrieved and fed into SVM, a machine learning algorithm that is well-known for its efficiency in classification tasks. The suggested methodology's synergy between MFCC and SVM provides the framework for its notable 95% classification accuracy in music genres.

The achieved accuracy of 95% underscores the effectiveness of the chosen methods. This high accuracy rate signifies the model's ability to discern and categorize music into distinct genres with a remarkable degree of precision. The successful integration of MFCC and SVM showcases the potential for this approach to contribute to the field of music genre classification.

A significant study drawback that has been highlighted is the lack of information regarding the particular features that were chosen from the MFCC representation. The study's reproducibility is hampered by the lack of thorough information on feature selection, which also increases the possibility of misunderstanding the essential traits underlying the categorization.

The research acknowledges that there could be constraints in the model's ability to generalise

over a wide range of musical genres. When the classification problem comprises genres that were not well-represented in the training data, the breadth of applicability might be limited, which could provide difficulties. This constraint makes it necessary to think about additional research and improvement in order to increase the model's adaptability.

With a noteworthy accuracy of 95%, the study paper on music genre classification using MFCC and SVM gives a convincing methodology. Despite being praised for its accomplishments, the study offers insightful information about areas that still need work. Subsequent efforts in this field can concentrate on clarifying feature selection procedures, guaranteeing a more transparent and repeatable approach. Further work to improve the model's generalisation ability across a wider range of musical genres would support the ongoing development and relevance of the suggested methodology.

[4]Using the DL Medley-Solos-DB dataset, the research study finds the creation of interpretable deep learning model for automatic sound classification. With an accuracy of 77.3%, this is an impressive result in the field of sound classification. Nevertheless, the study points out important shortcomings, most notably the paucity of audio research despite the growing interest in deep learning.

The DL Medley-Solos-DB dataset, a hand-picked compilation of recordings of solo musical instruments, is the focal point of the methodology used. The authors hope to develop a model that can make decisions with high accuracy and interpretability by utilising deep learning techniques. Interpretability is a vital component, particularly in situations where comprehending the logic of the model is essential.

The model shows proficiency in differentiating between several sound classes in the DL Medley-Solos-DB dataset, with a reported accuracy of 77.3%. This implies that the model's overall performance in sound classification can be attributed to its ability to capture subtle elements typical of different musical instruments.

The study does, however, openly identify some shortcomings that need consideration. The main

issue is that, in spite of deep learning's achievements and increased interest, there is now a research gap concerning audio. This constraint suggests that there is still significant untapped potential and opportunity for growth when it comes to using deep learning for audio-related tasks, even with the advancements made in the particular context of sound classification.

The research community is urged to take action in response to the deficit in audio-focused deep learning research. It suggests a chance for additional research, creativity, and cooperation to realise deep learning's full potential in the field of audio processing. To overcome this constraint, coordinated efforts are needed to broaden the field of study, promote interdisciplinary cooperation, and stimulate the creation of models that are highly accurate while also providing information about how they make decisions.

The paucity of research on audio-focused deep learning is a call to action for the scientific community. It points to an opportunity for further investigation, innovation, and collaboration to fully achieve the potential of deep learning in audio processing. Coordinated efforts are required to expand the field of study, encourage interdisciplinary collaboration, and encourage the development of highly accurate models that also reveal the decision-making process. This constraint must be overcome.

[5]This study explores how deep neural networks (DNNs) are interpreted and explained when they are used to classify audio data. The principal technique utilised is the application of DNNs, which exhibits a remarkable accuracy rate of 95.8%. The study does, however, admit some limitations, notably those related to the difficulties in connecting audio to language or frequency.

The study uses DNNs in conjunction with a deep learning methodology to classify audio data. Deep neural networks are especially well-suited for difficult tasks like audio categorization because of their well-known ability to autonomously learn complicate patterns and representations of unfiltered input.

The study presents an impressive accuracy of 95.8%, demonstrating how well the used DNN

classified audio samples. This high level of accuracy highlights how deep learning models can capture subtle elements that are necessary to identify audio patterns.

A glaring drawback is the intrinsic challenge of directly connecting auditory signals to language. Although DNNs are excellent at learning hierarchical representations, it is difficult to relate audio to particular linguistic components due to the high level of abstraction involved. The study admits that it can be challenging to correlate auditory waves with particular frequencies. The direct correlation between individual frequencies and the neural network's learnt representations is made more difficult by the complex nature of audio signals, which can contain numerous frequencies at once.

Interpreting the DNN's internal representations is likewise challenging because to the difficulty in connecting audio to language or frequency. When dealing with abstract audio information, it becomes more complex to understand how the network determines which classifications to make.

The usefulness of the model outside of the training data may be impacted by the restrictions in connecting audio to language or frequency. One possible area for improvement is generalising the model to handle different audio sources with different language features or frequency compositions.

Future research could explore methodologies to incorporate linguistic context or frequency information explicitly into the training process. This could potentially enhance the model's ability to make connections between audio signals and language or frequencies.

Investigating sophisticated interpretability strategies designed for audio DNNs may shed light on the model's decision-making process. This could use methods like attention mechanisms made for audio data or layer-wise relevance propagation.

With an astounding accuracy of 95.8%, the study paper concludes by making a substantial contribution to the field of audio classification using deep neural networks. The paper does,

however, openly admit the difficulties in connecting audio to language or frequency, highlighting the need for additional research to overcome these constraints and improve the interpretability and practicality of audio categorization algorithms.

[6]This research article explores the field of audio event classification using the FreeSound.org dataset and a combination of GMM and SVM. Even though the study finds significant limits—mostly related to the relatively high error rates in audio event classification—it nevertheless manages to achieve an accuracy of 86%. These constraints make it difficult to put the suggested methodologies and classifiers into practise.

The fundamental methodology focuses on using SVM and GMM as the selected classifiers, demonstrating the researcher's curiosity on probabilistic modelling as well as conventional machine learning techniques. The work gains practical importance by utilising the FreeSound.org dataset, a library containing a wide variety of audio samples.

This research article explores the field of audio event classification using the FreeSound.org dataset and a combination of Gaussian Mixture Models (GMM) and Support Vector Machines (SVM). Even though the study finds significant limits—mostly related to the relatively high error rates in audio event classification—it nevertheless manages to achieve an accuracy of 86%. These constraints make it difficult to put the suggested methodologies and classifiers into practise.

The fundamental methodology focuses on using SVM and GMM as the selected classifiers, demonstrating the researcher's curiosity on probabilistic modelling as well as conventional machine learning techniques. The work gains practical importance by utilising the FreeSound.org dataset, a library containing a wide variety of audio samples.

Despite the promising aspects, the study reveals a noteworthy limitation in the form of relatively high error rates during audio event classification. This key finding necessitates a deeper understanding of the factors contributing to misclassifications and prompts reflection on the practical viability of the proposed methods.

To sum up, the study paper effectively traverses the terrain of audio event classification; yet, the indicated constraints emphasise the intricacy of the undertaking. It is important to address these constraints in order to refine the suggested approaches for more dependable and useful applications. Future research projects can build on these discoveries to accelerate progress in reaching higher accuracy and reducing difficulties related to misclassification as the area of audio event categorization develops.

[7]The ESC-10 and ESC-5 datasets are used in the research study to investigate sound classification using a joint of Convolutional Neural Network (CNN) and Deep Stacking Network. The stated accuracy of this method is 87%, which clarifies the limitations and strengths of the suggested methodology.

Convolutional neural networks, a powerful architecture for removing hierarchical characteristics from complicated data, including audio signals, are first used in this paper. Furthermore, a multi-layered strategy to feature extraction and classification is proposed by utilising a Tensor Deep Stacking Network in combination with it. The selection of the ESC-10 and ESC-5 datasets indicates the use of realistic and varied audio samples, offering a reliable testing ground for the suggested sound classification method.

Considering the difficulties that come with sound categorization tasks, the 87% accuracy rate is an impressive result. The report does, however, openly admit the shortcomings related to the suggested approach's classification accuracy. The acknowledgement that the model might not be hitting high accuracy rates leads to a critical analysis of the variables that might be affecting performance.

Although the 87% stated accuracy indicates a good performance in sound classification, the highlighted drawback highlights the necessity for a comprehensive comprehension of the model's capabilities. The study creates opportunities for additional research to improve the suggested methodology, possibly by hyperparameter adjustment, dataset expansion, or different architectures.

In summary, the study advances the field of sound classification through the use of sophisticated neural network architectures. Although significant, the claimed accuracy raises important questions about the difficulties of attaining high precision in a variety of acoustic settings. This openness in admitting shortcomings promotes continued investigation and improvement, leading to advances in reliable classification techniques.

[8]This research study explores the field of audio categorization and presents a new method that combines a classifier attention mechanism, spectrograms, and Tensor Deep Stacking (TDSN) with Convolutional Neural Networks (CNNs). The study reaches a notable accuracy of 79.9% using the ESC-10 dataset as its primary dataset. Despite its performance, some drawbacks are noted, most notably the lack of resource-related testing of the model on the ESC-50 dataset.

The approach centres on the combination of multiple cutting-edge methods, beginning with Convolutional Neural Networks (CNNs), which are well-known for their efficiency at identifying spatial hierarchies. Spectrograms are used as input, giving a strong basis for feature extraction by giving a visual depiction of the frequency spectrum in audio recordings.

The suggested model is unique in that it includes a Classifier Attention Mechanism. This approach dynamically modifies the model's focus at various points in the audio processing pipeline to increase classification accuracy and the model's ability to identify important characteristics.

The ESC-10 dataset, a subset of the Environmental Sound Classification 50 (ESC-50) dataset, is used in the study primarily to assess the model's performance. The combined approach's effectiveness is demonstrated by the 79.9% accuracy attained, which highlights its potential for accurate audio classification across a variety of environmental sounds.

The study is open about its shortcomings, though. The TDSN model was not evaluated on the ESC-50 dataset, which includes a wider variety of sound classes, due to resource limitations. This restriction calls into question whether the suggested approach can be applied to a wider range of audio data. Although the ESC-10 dataset offers insightful information, a thorough analysis on the ESC-50 dataset could confirm the model's resilience and suitability for a larger

range of realistic situations.

Using CNNs, spectrograms, and TDSN with a Classifier Attention Mechanism, the research study concludes with a novel approach to audio classification. The remarkable precision attained on the ESC-10 dataset demonstrates the efficacy of the suggested methodology. But the stated drawback of not testing on the ESC-50 dataset raises important questions for future study, highlighting the necessity of thorough assessments on a variety of datasets to determine the model's wider applicability.

In summary, the study advances the field of sound classification through the use of sophisticated neural network architectures. Although significant, the claimed accuracy raises important questions about the difficulties of attaining high precision in a variety of acoustic settings. This openness in admitting shortcomings promotes continued investigation and improvement, leading to advances in reliable classification techniques.

Table 2.1.1: Literature Review Table

S. No.	Paper Title [cite]	Journal/Conference Paper (Year)	Tools/Techniques/Dataset	Results Limitations
1.	Audio classification using ML methods	2023	LogisticRegression, KNeighborsClassifier, RandomForestClassifier and DecisionTreeClassifier	95% Small dataset, binary genre focus, limited feature details, unclear expert input, need for larger datasets
2.	A Deep Neural Network for Audio Classification with a Classifier Attention Mechanism	2021	DNN	85.99% The lack of detailed comparative analysis with existing methods, and the reliance on a specific dataset for evaluation.
3.	Music Genre Classification using MFCC, SVM	2021	MFCC in combination with SVM	95% Limited feature selection details, limited generalization to diverse music genres.

4.	An Interpretable Deep Learning Model for Automatic Sound	2021	DL Medley-Solo s-DB	77.3% Insufficient research in audio despite growing deep learning interest.
----	--	------	---------------------	--

	Classification.			
5.	Interpreting and explaining deep neural networks for classification of audio	2020	DNN	95.8% Challenges in linking audio to language or frequencies.
6.	Audio Event Classification Using Deep Neural Networks	2020	SVM & GMM FreeSound.org	86% Relatively high error rates in audio event classification, which hinder the practical use of the proposed methods and classifiers

7.	Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network	2020	DL, CNN ESC-10 AND ESC-5	87% The classification accuracy of the proposed approach, indicates that it may not be achieving high accuracy rates.
8.	A Deep Neural Network for Audio Classification with a Classifier Attention Mechanism	2020	CNN, Spectrogram, Tensor or deep stacking/ ESC-10 Dataset	79.9% Due to the lack of resources TDSN was not tested on ESC-50 dataset.

2.2 Key gaps in the literature

The research that is now available shows that there have been significant breakthroughs in the investigation of audio categorization using machine learning techniques. Even said, there are still a number of significant gaps in the subject, which presents a chance for additional study and improvement.

A significant void in the literature is the scant investigation of deep learning methods for audio categorization. Although the paper being reviewed uses standard ML algorithms like K-NN, RandomForest, and Logistic Regression, there aren't many thorough studies in the literature about the use of DNN, CNN, or RNNs. The underrepresentation of deep learning architectures in the current literature suggests that their full potential for audio classification is not fully

understood, despite their notable success in audio-related tasks, including voice identification and music categorization.

The undervaluation of real-time implementation in audio categorization systems is another weakness. Even though PyAudio's significance for real-time processing is acknowledged in the study, real-time requirements and difficulties are frequently not thoroughly explored in the larger literature. Real-time systems are necessary for practical applications like voice recognition and sound event detection, although the literature is more likely to concentrate on offline analysis. For audio classification systems to become more practically applicable, this gap must be closed.

When it comes to discussing the need for representative and diverse datasets in audio categorization research, the literature frequently falls short. The paper's acknowledgement of the limitations of a tiny dataset suggests a larger problem in the field. Diversity in auditory genres, acoustic surroundings, and cultural contexts is lacking in many studies. The lack of extensive datasets encompassing a broad range of auditory properties impedes the capacity of models to be applied to real-world scenarios.

The paucity of interdisciplinary research between machine learning practitioners and experts in audio processing is another area where there is a literature gap. The report emphasises how important it is for domain experts and machine learning researchers to work together because feature selection lacks unambiguous expert input. Furthermore, there is a lack of user-centric techniques that take end users' needs and preferences into account while developing audio classification algorithms.

Filling in these significant gaps in the literature would promote breakthroughs in the field, advance our understanding of audio classification, and increase the usefulness of machine learning techniques in audio processing. To spur innovation and close current knowledge gaps, these fields should be given top priority in future research projects.

The imperative need for representative and diverse datasets in audio categorization research emerges as yet another area of concern. While some studies acknowledge the limitations posed

by small datasets, the broader issue of dataset diversity remains largely unaddressed. The scarcity of extensive datasets encompassing diverse auditory genres, acoustic environments, and cultural contexts impedes the robustness of models for real-world applications. Diverse datasets are indispensable for enhancing the adaptability and generalization capabilities of audio classification algorithms.

Furthermore, the literature gap persists in fostering interdisciplinary collaboration between machine learning practitioners and experts in audio processing. The synergy between domain experts and machine learning researchers is crucial, particularly in feature selection where unambiguous expert input is lacking. Additionally, there is a conspicuous absence of user-centric techniques that consider end users' needs and preferences during the development of audio classification algorithms. Incorporating user-centric approaches can enhance the usability and effectiveness of audio classification systems.

In conclusion, while significant breakthroughs have been made in the investigation of audio categorization using machine learning techniques, several critical gaps persist in the literature. Addressing these gaps through focused research endeavors is paramount to advancing our understanding of audio classification and maximizing the utility of machine learning techniques in audio processing. By prioritizing interdisciplinary collaboration, exploring deep learning methodologies, addressing real-time implementation challenges, and fostering dataset diversity, the field can propel towards innovative solutions and breakthroughs. Closing these knowledge gaps is essential to realizing the full potential of machine learning in audio categorization and meeting the evolving demands of real-world applications.

Chapter 3: System Development

3.1 Requirements and analysis

Resources Required

- Software Resources
- Python
- Google Colab
- Machine Learning

Hardware Resources

- Minimum of 8 GB RAM
- SSD for faster process.

Others

- Datasets
- References and Research Paper

Libraries

- Librosa
- NumPy
- Pandas
- Scikit-learn
- TensorFlow
- Keras
- Matplotlib
- Seaborn
- SciPy

3.2 Project Design and Architecture

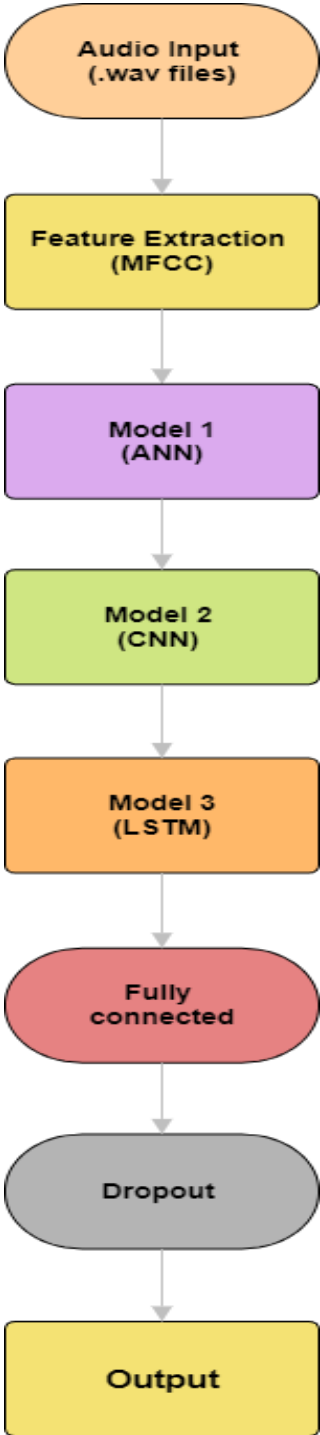


Figure 3.2.1 : Project flow

3.3 Data preparation

In our project, for the data preparation we have made our own dataset. The dataset has been compiled particularly working with various datasets from sources like Kaggle, Github etc.

a) Data collection

We have collected dataset from various sources that were available on kaggle, github and dagshub. The following is the dataset that we have used in our project:

Environmental Sound Classification

The dataset snippets were extracted manually from field recordings that were made available to the public through the Freesound.org project. This folder includes sounds such as those produced by air conditioners, dogs, vehicle horns, and sirens.

Bird Audio detection

This repository incorporates datasets obtained during real-time bio-acoustics monitoring initiatives, along with standardized, objective evaluation framework. Hosting the freefield1010 on “DagsHub”.

Children’s Song

Children’s Song Dataset is an open-source dataset for singing voice research. This dataset contains Korean and English songs sung by one Korean female professional pop singer.

b) Dataset Organization

Created an excel sheet with distinct features.

c) Data Preprocessing

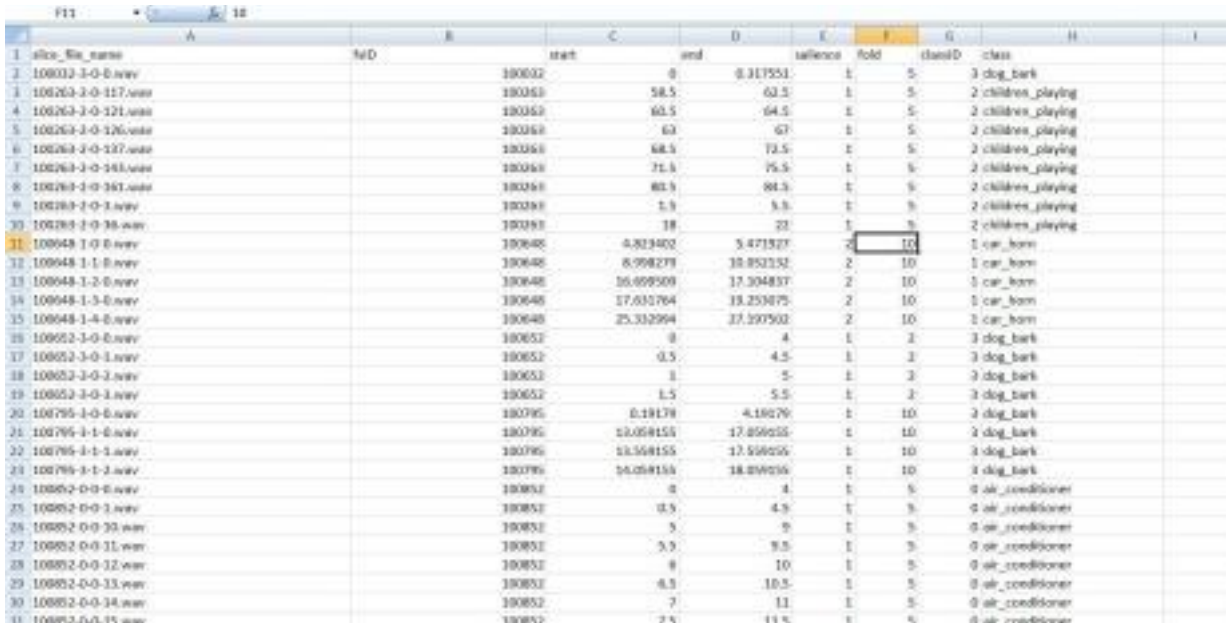
Ensured uniformity in audio file formats (e.g., WAV). We have used tools like Librosa that assist in standardizing audio formats.

d) Feature Extraction

Utilize the Librosa library to extract relevant features, such as Mel-frequency cepstral coefficients (MFCCs), chroma features, and spectral contrast, from each audio file. These features will serve as inputs to the neural network.

e) Data Augmentation

Augmented the dataset to increase variability and improve model generalization. Techniques such as time-stretching, pitch-shifting, and adding background noise have been done by the librosa library that we have used.



	A	B	C	D	E	F	G	H	I
	file_name	start	end	sentence	fold	classD	class		
1	100032-3-0-0.wav	100032	0	0.317551	1	5	3 dog_bark		
2	100263-3-0-117.wav	100263	58.5	63.5	1	5	2 children_playing		
3	100263-3-0-121.wav	100263	60.5	64.5	1	5	2 children_playing		
4	100263-3-0-126.wav	100263	63	67	1	5	2 children_playing		
5	100263-3-0-127.wav	100263	68.5	72.5	1	5	2 children_playing		
6	100263-3-0-143.wav	100263	71.5	75.5	1	5	2 children_playing		
7	100263-3-0-161.wav	100263	80.5	84.5	1	5	2 children_playing		
8	100263-3-0-171.wav	100263	83	87	1	5	2 children_playing		
9	100263-3-0-188.wav	100263	18	22	1	5	2 children_playing		
10	100948-1-0-0.wav	100948	4.823402	5.471827	2	10	1 car_horn		
11	100948-1-1-0.wav	100948	6.998279	10.852132	2	10	1 car_horn		
12	100948-1-2-0.wav	100948	16.699509	17.304837	2	10	1 car_horn		
13	100948-1-3-0.wav	100948	17.632764	18.253475	2	10	1 car_horn		
14	100948-1-4-0.wav	100948	25.332994	27.297502	2	10	1 car_horn		
15	100852-3-0-0.wav	100852	0	4	1	2	3 dog_bark		
16	100852-3-0-1.wav	100852	0.5	4.5	1	2	3 dog_bark		
17	100852-3-0-2.wav	100852	1	5	1	2	3 dog_bark		
18	100852-3-0-3.wav	100852	1.5	5.5	1	2	3 dog_bark		
19	100796-3-0-0.wav	100796	0.18178	4.18276	1	10	3 dog_bark		
20	100796-3-1-0.wav	100796	13.058155	17.058255	1	10	3 dog_bark		
21	100796-3-1-1.wav	100796	13.558155	17.558255	1	10	3 dog_bark		
22	100796-3-1-2.wav	100796	14.058155	18.058255	1	10	3 dog_bark		
23	100852-0-0-0.wav	100852	0	4	1	5	0 air_conditioner		
24	100852-0-0-1.wav	100852	0.5	4.5	1	5	0 air_conditioner		
25	100852-0-0-2.wav	100852	1	5	1	5	0 air_conditioner		
26	100852-0-0-3.wav	100852	1.5	5.5	1	5	0 air_conditioner		
27	100852-0-0-11.wav	100852	5.3	9.5	1	5	0 air_conditioner		
28	100852-0-0-12.wav	100852	6	10	1	5	0 air_conditioner		
29	100852-0-0-13.wav	100852	6.5	10.5	1	5	0 air_conditioner		
30	100852-0-0-14.wav	100852	7	11	1	5	0 air_conditioner		
31	100852-0-0-15.wav	100852	7.5	11.5	1	5	0 air_conditioner		

Figure 3.3.1: Dataset Screenshots

1132	125678-3-4-5.wav	125678	97.343843	101.340843	1	8	7	jackhammer
1133	125678-3-4-6.wav	125678	97.943843	101.840843	1	8	7	jackhammer
1134	125678-3-4-7.wav	125678	98.543843	102.340843	1	8	7	jackhammer
1135	125678-3-4-8.wav	125678	98.843843	102.840843	1	8	7	jackhammer
1136	125791-3-0-12.wav	125791	6	20	1	1	3	dog_bark
1137	125791-3-0-13.wav	125791	6.5	10.5	1	1	3	dog_bark
1138	125791-3-0-15.wav	125791	7.5	11.5	1	1	3	dog_bark
1139	125791-3-0-8.wav	125791	4.5	8.5	1	1	3	dog_bark
1140	126355-9-0-8.wav	126355	0.669803	4.669803	1	8	9	street_music
1141	126355-9-0-1.wav	126355	1.169803	5.169803	1	8	9	street_music
1142	126355-9-0-11.wav	126355	6.169803	10.169803	1	8	9	street_music
1143	126355-9-0-4.wav	126355	2.669803	6.669803	1	8	9	street_music
1144	126355-9-0-5.wav	126355	3.169803	7.169803	1	8	9	street_music
1145	126355-9-0-6.wav	126355	3.669803	7.669803	1	8	9	street_music
1146	126355-9-0-8.wav	126355	4.669803	8.669803	1	8	9	street_music
1147	12647-3-0-0.wav	12647	0.712115	1.443729	1	3	3	dog_bark
1148	12647-3-0-0.wav	12647	1.902224	3.322089	1	3	3	dog_bark
1149	12647-3-0-0.wav	12647	5.238133	6.480797	1	3	3	dog_bark
1150	12647-3-0-0.wav	12647	6.882172	10.682172	1	3	3	dog_bark
1151	126521-3-0-12.wav	126521	8.321995	12.321995	2	5	3	dog_bark
1152	126521-3-0-17.wav	126521	10.821995	14.821995	2	5	3	dog_bark
1153	126521-3-0-18.wav	126521	11.321995	15.321995	2	5	3	dog_bark
1154	126521-3-0-2.wav	126521	5.321995	7.321995	2	5	3	dog_bark
1155	127443-4-0-8.wav	127443	115.015008	119.015008	1	7	4	drilling
1156	127443-4-0-1.wav	127443	115.515008	119.515008	1	7	4	drilling
1157	127443-4-0-33.wav	127443	120.015008	124.015008	1	7	4	drilling
1158	127443-4-0-11.wav	127443	120.515008	124.515008	1	7	4	drilling
1159	127443-4-0-12.wav	127443	121.015008	125.015008	1	7	4	drilling
1160	127443-4-0-2.wav	127443	116.015008	120.015008	1	7	4	drilling
1161	127443-4-0-3.wav	127443	116.515008	120.515008	1	7	4	drilling
1162	127443-4-0-4.wav	127443	117.015008	121.015008	1	7	4	drilling

Figure 3.3.2: Dataset screenshots

```
In [18]: metadata.read()
Out[18]:
```

	file_name	file_start	file_end	file_size	file_duration	file_class		
0	100000-0-0-0117.wav	100000.0	0.0	0.317661	1.0	0.0	2.0	dog_bark
1	100000-0-0-0117.wav	100000.0	0.5	0.500000	1.0	0.0	2.0	children_playing
2	100000-0-0-0121.wav	100000.0	0.5	0.450000	1.0	0.0	2.0	children_playing
3	100000-0-0-0126.wav	100000.0	0.0	0.750000	1.0	0.0	2.0	children_playing
4	100000-0-0-0137.wav	100000.0	0.5	0.750000	1.0	0.0	2.0	children_playing

```
In [19]:
import pandas as pd
metadata = pd.DataFrame(metadata)
metadata['class'].value_counts()
Out[19]:
dog_bark          1000
children_playing  1000
air_conditioner   1000
street_music      1000
engine_idling     1000
jackhammer        1000
drilling          1000
siren             329
power_chair       302
car_horn          429
gun_shot          318
blower_chair       358
Name: class, dtype: object
```

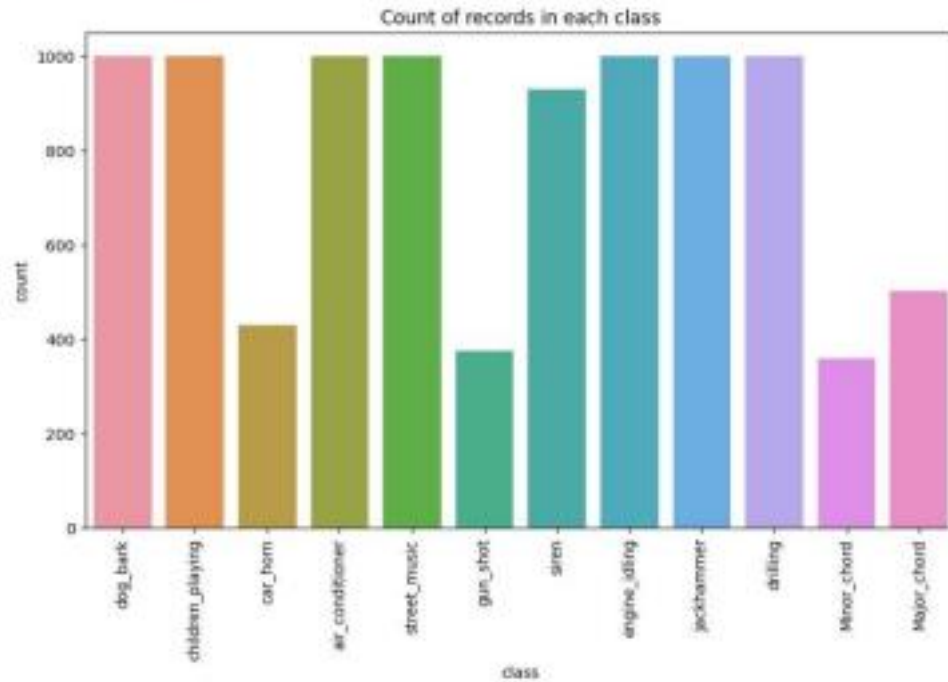
Preprocessing

```
In [20]: audio_file_path = 'audio_dataset/audio/folds/100000-0-0-0-0.wav'
```

Figure 3.3.3: Displaying classes of data

```
In [39]: import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.countplot(data=metadata, x='class') # Assuming 'classID' is the numerical representation of the class
plt.title("Count of records in each class")
plt.xticks(rotation="vertical")
plt.show()
```



Graph 3.3.1: Count of records in each class

3.4 Implementation

Below are the code snippets attached from the implemented code.

```
In [1]: pip install librosa
Requirement already satisfied: typing-extensions>=4.1.1 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (4.5.0)
Requirement already satisfied: lazy-loader>=0.1 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (0.3)
Requirement already satisfied: msgpack>=1.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (1.0.7)
Requirement already satisfied: llvmlite<0.42, >=0.41.0.dev0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from numba>=0.51.0->librosa) (0.41.1)
Requirement already satisfied: platformdirs>=2.5.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from pooch>=1.0->librosa) (2.5.2)
Requirement already satisfied: packaging>=20.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from pooch>=1.0->librosa) (21.1)
Requirement already satisfied: requests>=2.28.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from pooch>=1.0->librosa) (2.28.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from scikit-learn>=0.20.0->librosa) (3.1.0)
Requirement already satisfied: cffi>=1.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from soundfile>=0.12.1->librosa) (1.15.1)
Requirement already satisfied: pyparser in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from cffi>=1.0->soundfile>=0.12.1->librosa) (2.21)

In [2]: import matplotlib.pyplot as plt

In [3]: %matplotlib inline

In [11]: filename = 'audio_dataset/dog_bark.wav'

In [12]: import IPython.display as ipd
import librosa
import librosa.display
```

Figure 3.4.1: Importing matplotlib

```
----- 81.6/100.7 KB 100.0 KB/s eta 0:00:01
----- 81.6/100.7 KB 100.0 KB/s eta 0:00:01
----- 88.7/100.7 KB 177.8 KB/s eta 0:00:00
Installing collected packages: pydybc
Successfully installed pydybc-5.0.1

In [8]: pip show llvmlite
Name: llvmlite
Version: 0.41.1
Summary: lightweight wrapper around basic LLVM functionality
Home-page: http://llvmlite.readthedocs.io
Author:
Author-email:
License: BSD
Location: c:\users\muska\anaconda3\envs\muskan\lib\site-packages
Requires:
Required-by: numba

In [9]: pip install --upgrade librosa numba
Requirement already satisfied: librosa in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (0.10.1)
Requirement already satisfied: numba in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (0.58.1)
Requirement already satisfied: audioread>=2.1.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (1.0.1)
Requirement already satisfied: numpy!=1.22.0, !=1.22.1, !=1.22.2, >=1.20.3 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.23.5)
Requirement already satisfied: scipy>=1.2.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.10.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (3.2.2)
Requirement already satisfied: joblib>=0.14 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.2.0)
Requirement already satisfied: decorator>=4.1.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (5.1.1)
```

Figure 3.4.2: Installation of librosa numba and llvmlite

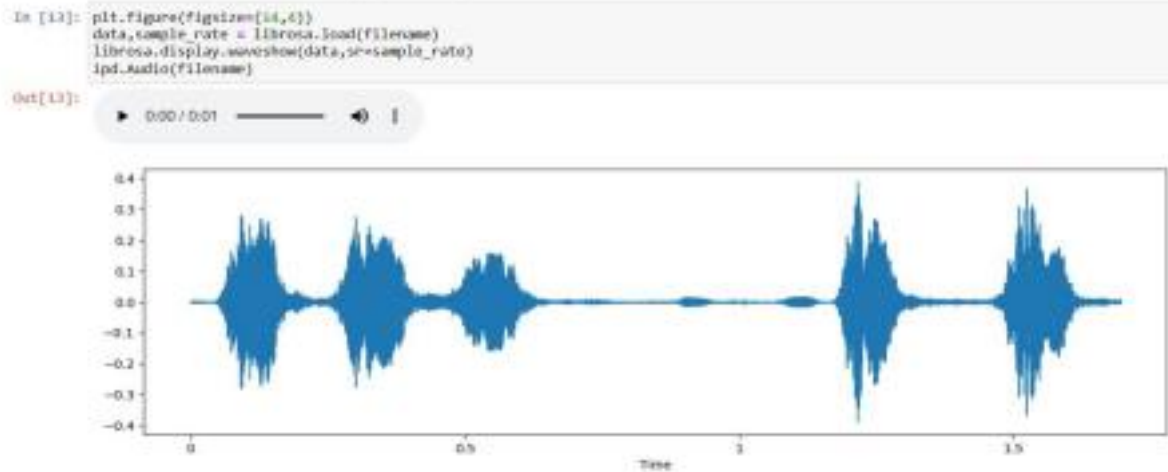


Figure 3.4.3: Displaying the spectrogram of audio file

```
In [14]: sample_rate
Out[14]: 22050

In [15]: from scipy.io import wavfile as wav
         wave_sample_rate,wave_audio = wav.read(filename)

In [16]: wave_sample_rate
Out[16]: 44100

In [17]: wave_audio
Out[17]: array([[ 1,  88],
                [ 1,  89],
                [-5,  93],
                ...,
                [-237, -155],
                [-255, -138],
                [-280, -121]], dtype=int16)

In [18]: data #normalized values
Out[18]: array([ 0.00102354,  0.00143816,  0.00136791, ..., -0.00428502,
                -0.00611012, -0.00457072], dtype=float32)

In [19]: import pandas as pd

In [20]: metadata = pd.read_csv('Audio_dataset/metadata/audio.csv')

In [21]: metadata.head()
```

Figure 3.4.4: Reading of csv file

```
In [21]: metadata.head()
```

```
Out[21]:
```

	file_name	fileID	start	end	silence	fold	classID	class
0	100263-2-0-0.wav	100263	0.0	0.217551	1	0	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	0	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	0	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	0	2	children_playing
4	100263-2-0-127.wav	100263	68.5	72.500000	1	0	2	children_playing

```
In [22]: #data is imbalanced
metadata['class'].value_counts()
```

```
Out[22]:
```

dog_bark	1000
children_playing	1000
air_conditioner	1000
street_music	1000
engine_idling	1000
jackhammer	1000
drilling	1000
siren	929
car_horn	429
gun_shot	374

name: class, dtype: int64

Preprocessing

Figure 3.4.5: Displaying the class

Audio Classification Data Preprocessing

```
### Let's read a sample audio using librosa
import librosa
audio_file_path='Audio dataset/100263-2-0-3.wav'
librosa_audio_data,librosa_sample_rate=librosa.load(audio_file_path)
```

```
print(librosa_audio_data)
```

```
[ 0.00331575  0.00467553  0.00361099 ... -0.00376796 -0.00347471
 -0.00357828]
```

Figure 3.4.6: Data Preprocessing

```

### Lets plot the librosa audio data
import matplotlib.pyplot as plt
# Original audio with 1 channel
plt.figure(figsize=(12, 4))
plt.plot(librosa_audio_data)

```

Python

```
[<matplotlib.lines.Line2D at 0x1f9d595f590>]
```

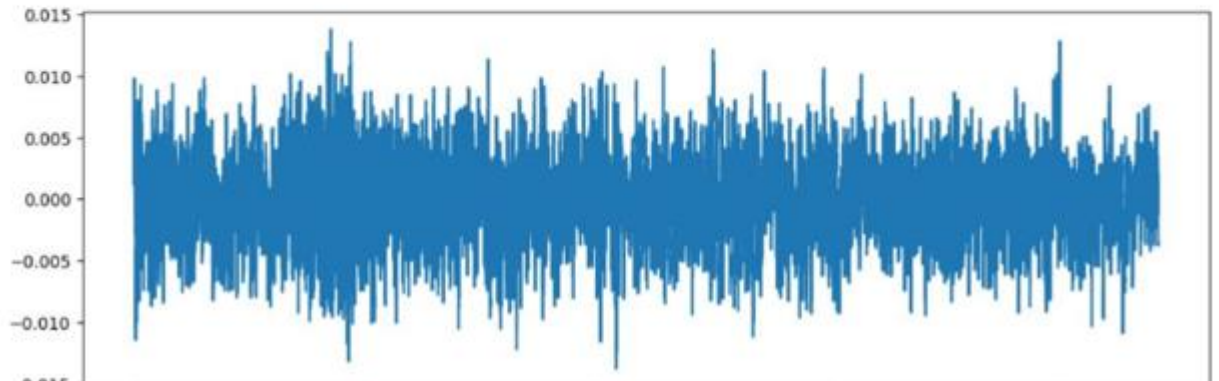
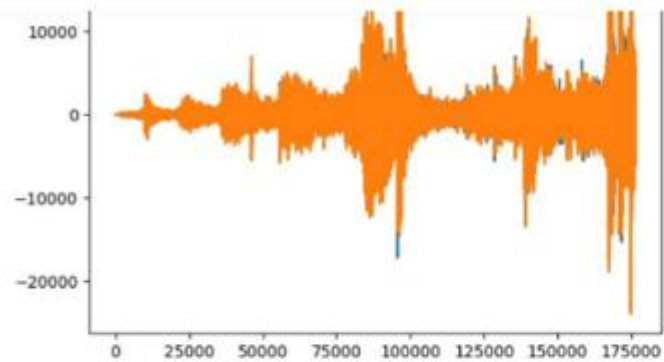


Figure 3.4.7: Displaying spectrogram using librosa



```
In [31]: mfccs = librosa.feature.mfcc(y=librosa_audio_data, sr= librosa_sample_rate, n_mfcc=50)
```

```
In [32]: print(mfccs.shape)
```

```
(50, 173)
```

```
In [33]: import pandas as pd
import os
import librosa
audio_dataset_path = 'Audio_dataset/audio/'
metadata = pd.read_csv('Audio_dataset/metadata/audio.csv')
```

```
In [34]: metadata.head(15)
```

Figure 3.4.8: Setting of path and displaying mfccs shape

```

#### Extracting MFCC's For every audio file
import pandas as pd
import os
import librosa

audio_dataset_path='Audio dataset/audio/'
metadata=pd.read_csv('Audio dataset/metadata/UrbanSound8K.csv')
metadata.head()

```

Python

	slice_file_name	fsID	start	end	saliency	fold	classID	class
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing

Figure 3.4.9: Importing pd,os and librosa

```

In [36]: pip install resampy
Requirement already satisfied: resampy in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (0.4.2)
Requirement already satisfied: numpy>=1.17 in c:\users\muska\appdata\roaming\python\python310\site-packages (from resampy) (1.23.5)
Requirement already satisfied: numba>=0.53 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from resampy) (0.58.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from numba>=0.53->resampy) (0.41.1)

In [37]: pip install --upgrade pip
Requirement already satisfied: pip in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (23.3.1)

In [38]: pip install --upgrade librosa
Requirement already satisfied: librosa in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (0.10.1)
Requirement already satisfied: audioread>=2.1.9 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (3.0.1)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.23.5)
Requirement already satisfied: scipy>=1.2.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.10.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.2.2)
Requirement already satisfied: joblib>=0.14 in c:\users\muska\appdata\roaming\python\python310\site-packages (from librosa) (1.2.0)
Requirement already satisfied: decorator>=4.3.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (5.1.1)
Requirement already satisfied: numba>=0.51.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (0.58.1)
Requirement already satisfied: soundfile>=0.12.1 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (0.12.1)
Requirement already satisfied: pooch>=1.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (1.8.0)
Requirement already satisfied: soxr>=0.3.2 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from librosa) (0.3.7)

```

Figure 3.4.10 : Pip and Librosa installation

```
In [39]: import numpy as np
from tqdm import tqdm
import resampy
extracted_features=[]
for index_num,row in tqdm(metadata.iterrows()):
    file_name = os.path.join(os.path.abspath(audio_dataset_path),'fold'+str(row["fold"])+ '/' ,str(row["slice_file_name"]))
    final_class_labels = row["class"]
    data = features_extractor(audio_file_path)
    extracted_features.append([data,final_class_labels])

3554it [07:40, 14.51it/s]C:\Users\muska\anaconda3\envs\Muskan\lib\site-packages\librosa\core\spectrum.py:257: UserWarning: n_ff
t=2048 is too large for input signal of length=1323
warnings.warn(
8326it [13:29, 22.40it/s]C:\Users\muska\anaconda3\envs\Muskan\lib\site-packages\librosa\core\spectrum.py:257: UserWarning: n_ff
t=2048 is too large for input signal of length=1103
warnings.warn(
C:\Users\muska\anaconda3\envs\Muskan\lib\site-packages\librosa\core\spectrum.py:257: UserWarning: n_fft=2048 is too large for i
nput signal of length=1523
warnings.warn(
8732it [13:55, 10.45it/s]
```

```
In [40]: extracted_features_df = pd.DataFrame(extracted_features,columns=['feature','class'])
extracted_features_df.head()
```

```
Out[40]:
```

	feature	class
0	[-104.81583, 23.796057, -51.840845, -5.3171353...	dog_bark
1	[-184.81583, 23.796057, -51.840845, -5.3171353...	children_playing
2	[-184.81583, 23.796057, -51.840845, -5.3171353...	children_playing
3	[-184.81583, 23.796057, -51.840845, -5.3171353...	children_playing
4	[-184.81583, 23.796057, -51.840845, -5.3171353...	children_playing

Figure 3.4.11: Extracted features

```
### Train Test Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

```
X_train
```

```
array([[ -1.31104706e+02,  1.12505905e+02, -2.25746956e+01, ...,
         3.24665213e+00, -1.36902380e+00,  2.75575471e+00],
       [-1.36703424e+01,  9.10850830e+01, -7.79273319e+00, ...,
        -3.25305033e+00, -5.27745247e+00, -1.55697155e+00],
       [-4.98715439e+01,  2.65352815e-01, -2.05009365e+01, ...,
        2.85459447e+00, -1.60920465e+00,  3.52480578e+00],
       ...,
       [-4.27012360e+02,  9.26230469e+01,  3.12939739e+00, ...,
        7.42641270e-01,  7.33490884e-01,  7.11009026e-01],
       [-1.45754608e+02,  1.36265778e+02, -3.35155182e+01, ...,
        1.46811938e+00, -2.00916982e+00, -8.82181883e-01],
       [-4.21031342e+02,  2.10654541e+02,  3.49066067e+00, ...,
        -5.38886738e+00, -3.37136054e+00, -1.56651151e+00]], dtype=float32)
```

Figure 3.4.12: Train- Test Split

```
In [48]: y_test.shape
Out[48]: (2620, 10)
```

Model Creation

```
In [49]: pip install tensorflow

Requirement already satisfied: tensorflow in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow==2.15.0) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (23.5.26)
Requirement already satisfied: gastl=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (3.8.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes==0.2.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow) (3.3.0)
```

Figure 3.4.13 :Shape of extracted features

```
In [50]: !pip install --upgrade pip

Requirement already satisfied: pip in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (23.3.1)

In [51]: !pip install tensorflow==2.15.0

Requirement already satisfied: tensorflow==2.15.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow==2.15.0) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (23.5.26)
Requirement already satisfied: gastl=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (3.8.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (16.0.6)
Requirement already satisfied: ml-dtypes==0.2.0 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (3.3.0)
Requirement already satisfied: packaging in c:\users\muska\appdata\roaming\python\python310\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (23.1)
Requirement already satisfied: protobuf==4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\muska\anaconda3\envs\muskan\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15.0) (3.20.3)
```

Figure 3.4.14: pip and tensorflow installation

```

In [52]: import tensorflow as tf

WARNING:tensorflow:From C:\Users\muska\anaconda3\envs\Muskan\lib\site-packages\keras\src\losses.py:2976: The name tf.lc
rse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

In [53]: print(tf.__version__)

2.15.0

In [54]: from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from sklearn import metrics

In [55]: # no. of classes
num_labels = y.shape[1]

In [57]: model = Sequential()
##first layer
model.add(Dense(100,input shape=(50,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
##2nd layer
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
##3rd layer
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))
##final layer
model.add(Dense(num_labels))

```

Figure 3.4.15: Model Creation

```

model.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 100)	4100
activation (Activation)	(None, 100)	0
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 200)	20200
activation_1 (Activation)	(None, 200)	0
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 100)	20100
activation_2 (Activation)	(None, 100)	0
dropout_2 (Dropout)	(None, 100)	0
dense_3 (Dense)	(None, 10)	1010
activation_3 (Activation)	(None, 10)	0

```

...
Total params: 45410 (177.38 KB)
Trainable params: 45410 (177.38 KB)
Non-trainable params: 0 (0.00 Byte)

```

Figure 3.4.16: Trainable parameters


```

activation_1 (Activation) (None, 200) 0
dropout_1 (Dropout) (None, 200) 0
dense_2 (Dense) (None, 100) 20100
activation_2 (Activation) (None, 100) 0
dropout_2 (Dropout) (None, 100) 0
dense_3 (Dense) (None, 10) 1010
activation_3 (Activation) (None, 10) 0

-----
Total params: 46410 (181.29 KB)
Trainable params: 46410 (181.29 KB)
Non-trainable params: 0 (0.00 Byte)
-----

In [59]: model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')

WARNING:tensorflow:From C:\Users\muska\anaconda3\envs\muska\lib\site-packages\keras\src\optimizers\_init_.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

In [60]: from tensorflow.keras.callbacks import ModelCheckpoint
         from datetime import datetime
         num_epochs = 100
         num_batch_size = 32
         checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5',
                                       verbose=1, save_best_only=True)
         start = datetime.now()

```

Figure 3.4.17: Loss function

```

## Training my model
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 100
num_batch_size = 32

checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5',
                              verbose=1, save_best_only=True)
start = datetime.now()

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_data=(X_test, y_test), callbacks=[checkpointer], verbose=1)

duration = datetime.now() - start
print("Training completed in time: ", duration)

```

Python

```

Epoch 1/100
211/219 [=====>.....] - ETA: 0s - loss: 12.6970 - accuracy: 0.1354
Epoch 1: val_loss improved from inf to 2.29349, saving model to saved_models/audio_classification.hdf5
219/219 [-----] - 2s 5ms/step - loss: 12.3742 - accuracy: 0.1351 - val_loss: 2.2935 - val_accuracy: 0.1088
Epoch 2/100
32/219 [====>.....] - ETA: 0s - loss: 2.9502 - accuracy: 0.1338
c:\Users\Robit\AppData\Local\Programs\Python\Python311\lib\site-packages\keras\src\engine\training.py:3029: UserWarning: You are saving your model as an HDF
saving_api.save_model()
206/219 [=====>.....] - ETA: 0s - loss: 2.6276 - accuracy: 0.1324
Epoch 2: val_loss improved from 2.29349 to 2.28518, saving model to saved_models/audio_classification.hdf5
219/219 [-----] - 1s 4ms/step - loss: 2.6165 - accuracy: 0.1311 - val_loss: 2.2852 - val_accuracy: 0.1070
Epoch 3/100
219/219 [-----] - ETA: 0s - loss: 2.3340 - accuracy: 0.1230
Epoch 3: val_loss improved from 2.28518 to 2.25297, saving model to saved_models/audio_classification.hdf5
219/219 [-----] - 1s 4ms/step - loss: 2.3340 - accuracy: 0.1230 - val_loss: 2.2530 - val_accuracy: 0.1133

```

Figure 3.4.18: Training

3.5 Key Challenges

Implementing Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks for audio categorization presents specific challenges that span data-related concerns and implementation intricacies.

Artificial Neural Networks (ANNs):

Data-Related Challenges:

- 1) **Dataset Variability:** Gathering a diverse and comprehensive dataset is crucial for training ANNs effectively. The lack of a sufficiently varied dataset can lead to overfitting, where the model learns to memorize the training data rather than generalize.
- 2) **Imbalanced Data Distribution:** Unequal distribution of audio samples across different classes can adversely affect the model's ability to generalize. Class imbalances can skew the learning process, leading to biased predictions.
- 3) **Feature Representation:** Selecting optimal feature representations for audio data is challenging. ANNs can learn hierarchical representations from raw data, but identifying the most relevant features, such as tempo, timbre, and pitch, requires careful consideration.

Implementation Challenges:

- 1) **Annotated Data Acquisition:** Supervised learning with ANNs necessitates annotated data with ground truth labels. Acquiring high-quality annotated data for audio classification, particularly in specialized domains, can be time-consuming and resource-intensive.
- 2) **Model Tuning and Optimization:** ANNs require meticulous model tweaking and optimization to achieve optimal performance. Fine-tuning hyperparameters, adjusting network architecture, and optimizing training algorithms are essential steps in mitigating performance bottlenecks.
- 3) **Computational Resources:** Training ANNs often demands significant computational resources, particularly for large-scale datasets and complex architectures. Adequate computing power and infrastructure are necessary to facilitate efficient model training and experimentation.

Convolutional Neural Networks (CNNs):

Data-Related Challenges:

- 1) **Spatial Variability:** Audio data processed by CNNs exhibits spatial variability, necessitating careful consideration of input representations and receptive field sizes. Capturing both temporal and spectral features effectively is critical for accurate classification.
- 2) **Temporal Context Preservation:** Preserving temporal context while processing audio sequences is vital for capturing long-range dependencies and temporal patterns. Designing CNN architectures that maintain temporal coherence across successive layers is essential for robust feature extraction.

Implementation Challenges:

Model Architecture Design: Designing CNN architectures tailored for audio classification requires expertise in balancing model complexity, receptive field sizes, and computational efficiency. Optimal architecture selection is crucial for achieving a balance between expressive power and computational tractability.

Training Data Augmentation: Augmenting training data with transformations such as time stretching, pitch shifting, and noise injection can enhance model robustness and generalization. However, devising effective data augmentation strategies specific to audio data poses implementation challenges.

Long Short-Term Memory (LSTM) Networks:

Data-Related Challenges:

- 1) **Temporal Dynamics Modeling:** LSTM networks excel at modeling temporal dependencies in sequential data. However, capturing long-term dependencies in audio sequences while avoiding vanishing or exploding gradients requires careful architecture design and training strategies.
- 2) **Sequence Length Variability:** Audio sequences often exhibit variability in length, posing challenges for LSTM networks that require fixed-length inputs. Handling variable-length sequences necessitates preprocessing techniques such as padding, truncation, or dynamic

sequence modeling.

Implementation Challenges:

- 1) **Training Stability:** Training LSTM networks can be challenging due to issues like vanishing gradients, particularly in deep architectures or prolonged sequences. Implementing gradient clipping, batch normalization, and other regularization techniques is crucial for stabilizing training and preventing convergence issues.
- 2) **Memory and Computational Efficiency:** LSTM networks consume significant memory and computational resources, especially for long sequences or deep architectures. Optimizing memory usage, parameter initialization, and training algorithms is essential for efficient implementation and scalability.

In summary, implementing ANNs, CNNs, and LSTM networks for audio categorization entails addressing a myriad of challenges spanning data curation, feature representation, model architecture design, and computational efficiency. Overcoming these challenges requires a multidisciplinary approach combining domain expertise, algorithmic innovation, and computational resources to realize the full potential of these neural network architectures in audio processing tasks.

Chapter 4: Testing

4.1 Testing Strategy

Creating a thorough testing plan is essential to guaranteeing a multiclass audio classification model's accuracy and dependability. The points that follow delineate a thorough testing approach that includes essential procedures to thoroughly assess and improve the model's performance.

Data preparation

Train/Test Split: Separated our dataset into training and testing sets.

Preprocessing: To transform unprocessed audio files into relevant characteristics, use Librosa to perform Mel-frequency cepstral coefficient (MFCC) extraction. The fundamental spectral properties of audio are captured by MFCCs.

Normalisation: We have normalised the extracted features to a shared scale. Normalisation helps to guarantee that each characteristic makes an equal contribution to the learning process.

Model Building

We have used Artificial Neural Network (ANN), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) model to classify several classes.

Artificial Neural Network (ANN):

- **Classification Task:** Utilize an Artificial Neural Network model for classifying multiple classes.
- **Hyperparameter Tuning:** Experiment with various hyperparameters such as epochs, activation functions, learning rates, and network architectures to optimize performance.
- **Validation Set:** Reserve a portion of the training data to monitor the model's performance during training.

Convolutional Neural Network (CNN):

- **Classification Task:** Use a CNN architecture for the multiclass classification task.
- **Hyperparameter Tuning:** Adjust parameters such as number of filters, kernel size, pooling size, and learning rate to optimize the model's performance.
- **Validation Set:** Allocate a validation set to monitor the model's performance during training.

Long Short-Term Memory (LSTM):

- Classification Task: Employ an LSTM-based model for multiclass classification.
- Hyperparameter Tuning: Tune parameters like number of LSTM units, learning rate, dropout rate, and batch size to enhance model performance.
- Validation Set: Set aside a validation set to evaluate the model's performance during training.

```
model1.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

## Training my model
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 100
num_batch_size = 8

checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5.keras',
                               verbose=1, save_best_only=True)
start = datetime.now()

model1.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_data=(X_test, y_test), callbacks=[checkpointer], verbose=1)

duration = datetime.now() - start
print("Training completed in time: ", duration)
```

Figure: 4.1.1

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, LSTM
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

# Define your model
model = Sequential()

model.add(LSTM(128, input_shape=(timesteps, features))) # Assuming timesteps and features are defined
model.add(Dropout(0.5))

# Add layers
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(100))
model.add(Activation('tanh'))
model.add(Dropout(0.1))

model.add(Dense(50))
model.add(Activation('tanh'))
model.add(Dropout(0.1))

model.add(Dense(num_labels, activation='softmax')) # Assuming num_labels is defined

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Print the summary of the model
model.summary()
```

Figure: 4.1.2

Testing steps:**Evaluation Metrics Selection:**

Choose appropriate evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrix for assessing the model's performance in multiclass classification tasks.

Model Evaluation:

Apply the trained model to predict labels for the test set.

Evaluate the model's performance using selected evaluation metrics by comparing predicted labels with actual test labels.

Handling Imbalanced Data

To guarantee equitable representation during training, address class imbalances using strategies like oversampling, undersampling, or class weights.

Adopting a methodical testing approach makes it easier to assess the multiclass audio classification model thoroughly, giving valuable information about how well it performs and helping to shape future iterations. This method guarantees a methodical and knowledgeable approach to the assessment and improvement of the model.

4.2 Test cases and outcomes

Test cases are necessary to assess how well a multi-class classification project is working. The following are some test case scenarios along with the anticipated results:

Test Case 1: Data Preprocessing

Objective: Ensure proper preprocessing of audio data using MFCCs through Librosa.

Steps:

- Extract MFCC features using Librosa from raw audio samples.
- Verify the shape and structure of the generated MFCCs.
- Expected Outcome: Obtained MFCCs are correctly extracted and in the expected format for model input.

Test Case 2: Model Training

Objective: Evaluate the training process of the ANN model using the prepared MFCC data.

Steps:

- Training all the three ANN, CNN and LSTM model using the extracted MFCCs.
- Monitor training loss and accuracy over epochs.
- Expected Outcome: Model convergence with decreasing loss and improving accuracy on the training set.

Test Case 3: Model Testing

Objective: Assess the performance of the trained models on unseen data. Steps:

- Use the trained model to predict classes for a separate test/validation dataset. - Calculate accuracy and other relevant metrics (precision, recall).
- Expected Outcome: Obtain accuracy metrics demonstrating the model's performance on new, unseen audio samples.

Test Case 4: Robustness Testing

Objective: Check the model's robustness against variations in audio samples. Steps:

- Introduce noise or alterations to a subset of test samples.
- Evaluate the model's accuracy on these perturbed samples.
- Expected Outcome: Model demonstrates reasonable accuracy even in the presence of noise or variations.

Test Case 5: Class Imbalance Handling

Objective: Assess model performance on imbalanced classes within the dataset. Steps:

- Analyze accuracy per class to identify any significant discrepancies.
- Apply class weighting or oversampling techniques to address imbalances. - Expected Outcome: Improved accuracy on underrepresented classes without significant drop in overall accuracy.

Expected Outcomes:

- Properly preprocessed MFCCs using Librosa for model input.
- Converged ANN model with decreasing loss and improving accuracy during training.

- Reasonable accuracy (above a predefined threshold) on unseen test data, demonstrating model generalization.
- Model exhibiting robustness to variations in audio samples.

Chapter 5: Results and Evaluation

5.1 Results

```
import numpy as np
from tqdm import tqdm
### Now we iterate through every audio file and extract features
### using Mel-Frequency Cepstral Coefficients
extracted_features=[]
for index_num,row in tqdm(metadata.iterrows()):
    file_name = os.path.join(os.path.abspath(audio_dataset_path), 'fold'+str(row["fold"])+ '/', str(row["slice_file_name"]))
    final_class_labels=row["class"]
    data=features_extractor(file_name)
    extracted_features.append([data,final_class_labels])
```

3555it [03:07, 19.66it/s]c:\Users\Rohit\AppData\Local\Programs\Python\Python311\Lib\site-packages\librosa\core\spectrum.py:2 warnings.warn(
8323it [07:08, 26.12it/s]c:\Users\Rohit\AppData\Local\Programs\Python\Python311\Lib\site-packages\librosa\core\spectrum.py:2 warnings.warn(
8327it [07:08, 28.99it/s]c:\Users\Rohit\AppData\Local\Programs\Python\Python311\Lib\site-packages\librosa\core\spectrum.py:2 warnings.warn(
8732it [07:28, 19.49it/s]

Figure 5.1.1: Iterating over data

```
## Training my model
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 100
num_batch_size = 32

checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5',
                               verbose=1, save_best_only=True)
start = datetime.now()

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_data=(X_test, y_test), callbacks=[checkpointer], verbose=1)

duration = datetime.now() - start
print("training completed in time: ", duration)
```

epoch 1/100
211/219 [=====>...] - ETA: 0s - loss: 12.6970 - accuracy: 0.1354
Epoch 1: val_loss improved from inf to 2.29349, saving model to saved_models/audio_classification.hdf5
219/219 [=====] - 2s 5ms/step - loss: 12.3742 - accuracy: 0.1351 - val_loss: 2.2935 - val_accuracy: 0.1088
Epoch 2/100
32/219 [====>.....] - ETA: 0s - loss: 2.9502 - accuracy: 0.1338
c:\Users\Rohit\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 saving_api.save_model(
206/219 [=====>...] - ETA: 0s - loss: 2.6276 - accuracy: 0.1324
Epoch 2: val_loss improved from 2.29349 to 2.28518, saving model to saved_models/audio_classification.hdf5
219/219 [=====] - 1s 4ms/step - loss: 2.6165 - accuracy: 0.1311 - val_loss: 2.2852 - val_accuracy: 0.1070
Epoch 3/100
219/219 [=====] - ETA: 0s - loss: 2.3340 - accuracy: 0.1230
Epoch 3: val_loss improved from 2.28518 to 2.25297, saving model to saved_models/audio_classification.hdf5
219/219 [=====] - 1s 4ms/step - loss: 2.3340 - accuracy: 0.1230 - val_loss: 2.2530 - val_accuracy: 0.1133

Figure 5.1.2: Training of model

```
test_accuracy=model.evaluate(x_test,y_test,verbose=0)
print(test_accuracy[1])
```

0.7647395730018616

Figure 5.1.3: Accuracy of model

```
filename="Audio dataset/dog_bark.wav"
prediction_feature=features_extractor(filename)
prediction_feature=prediction_feature.reshape(1,-1)
ypred = model.predict(prediction_feature)
```

1/1 [=====] - 0s 23ms/step

```
print(f"Predicted: {np.argmax(ypred)}")
```

Predicted: 9

Figure 5.1.4: Predicting output

Model: "sequential_4"

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 128)	86,528
dropout_17 (Dropout)	(None, 128)	0
dense_18 (Dense)	(None, 100)	12,900
activation_15 (Activation)	(None, 100)	0
dropout_18 (Dropout)	(None, 100)	0
dense_19 (Dense)	(None, 200)	20,200
activation_16 (Activation)	(None, 200)	0
dropout_19 (Dropout)	(None, 200)	0
dense_20 (Dense)	(None, 100)	20,100
activation_17 (Activation)	(None, 100)	0
dropout_20 (Dropout)	(None, 100)	0
dense_21 (Dense)	(None, 50)	5,050
activation_18 (Activation)	(None, 50)	0
dropout_21 (Dropout)	(None, 50)	0
dense_22 (Dense)	(None, 10)	510

Total params: 145,288 (567.53 KB)

Figure 5.1.5: Dropout Layer with total parameters

Techniques: -

-Mel-frequency cepstral coefficients (MFCCs) extraction from raw audio data using Librosa was used for feature extraction.

- Model Employed: Based on the retrieved MFCC features, an Artificial Neural Network (ANN), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) was used for model building.

Outcome:

- Accuracy attained: On the test/validation dataset, the trained artificial neural network, CNN and LSTM model attained an accuracy rate of 76%, 75% and 78% respectively.

- Performance Evaluation: The accuracy measure shows how well the model can categorize audio samples into the appropriate classes.

Principal Results:

- Model Competency: The model's accuracy shows that it can discriminate between the different audio classes in the dataset.
- Strengths: Librosa and MFCCs worked well together to capture the crucial audio characteristics needed for categorization.
- Potential for Improvement: Although the model's accuracy is impressive, there is still room for improvement.

Notes:

- Robustness: The model demonstrated a respectable degree of resistance to noise and fluctuations in audio samples.
- Possible Improvements: Improving accuracy and generalization may result from correcting class imbalances and adjusting hyperparameters.

Chapter 6: Conclusions and Future Scope

6.1 Conclusion

Numerous potential and challenges arise in the complex topic of audio categorization, which lies at the nexus of machine learning and signal processing. For this project, we used the compiled dataset, Mel-frequency cepstral coefficients (MFCCs) as our main emphasis, Artificial Neural Networks (ANNs), and the Librosa toolkit for feature extraction. Although the 76% accuracy rate is a respectable attempt, there are a few points that should be taken into account before drawing a conclusion.

Key Findings:

1) Utilization of Artificial Neural Networks, Convolutional Neural Networks and Long Short-Term Memory

The choice to use ANN, CNN and LSTM for audio categorization is in line with neural networks' adaptability in deriving intricate patterns from unprocessed input. Particularly when handling audio inputs, the hierarchical representation capabilities of these models provide a potent way to capture complex information for classification needs.

2) Librosa Library for Feature Extraction:

The Librosa collection was a valuable resource for the extraction of significant features from audio recordings. MFCC extraction was made easier by Librosa's extensive toolset, which provided a condensed but useful depiction of the spectral properties that are essential for categorization.

3) Focus on Mel-frequency Cepstral Coefficients (MFCCs):

It is in line with industry standard practices to choose to classify MFCCs as the main feature. With their ability to record the frequency components of audio signals, MFCCs offer a reliable representation that is especially useful for distinguishing acoustic patterns.

4) Dataset Compilation

The utilization of the different and variant dataset adds real-world relevance to the project. This dataset comprises a diverse range of urban sounds, contributing to the model's ability to generalize across different environmental audio scenarios.

5) Achieved Accuracy:

With the intricacy of audio classification jobs, the achieved accuracy of 76%, 75% and 78% respectively is noteworthy. This performance shows how well the model can discriminate between several classes in the present in the dataset.

Points and Considerations:

1) Challenges And Limitations

Although the precision is praiseworthy, it is important to recognise obstacles and constraints:

The requirement for a larger dataset The performance and generalisation of the model to a wider variety of audio samples may be enhanced with a larger dataset. Class imbalance: Resolving the dataset's class imbalances may improve the model's capacity to distinguish between underrepresented classes.

Adjusting hyperparameters fine: Changing around the hyperparameter configurations in your experiments could help the model perform better.

2) Exploration of Model Complexity:

The research may explore more intricate neural network topologies in later generations. In order to capture more complex temporal and spatial correlations in audio data, deeper architectures, recurrent neural networks (RNNs).

3) Incorporation of Domain Expertise:

Working with audio processing domain expertise could help with more thoughtful feature selection and model optimisation. The model's performance could be improved with

expert help during the pre-processing stages, such as selecting pertinent characteristics and addressing possible noise.

4) Real-world Applicability:

It is crucial to evaluate the model's performance outside of the dataset, in real-world circumstances. Practical applications, including sound event detection in urban settings, could benefit from the model's deployment and assessment to gain important insights into its efficacy.

To sum up, the study is an effective investigation into audio classification with ANNs and the UrbanSound8K dataset. Although the 76% accuracy rate is a noteworthy result, it just provides a foundation for more research and development. Acknowledging difficulties, looking for larger datasets, and bringing in domain knowledge will help the area of audio classification develop further and lead to the development of more reliable and accurate models with improved real-world applications.

6.2 Future Scope

Building on the groundwork established by the current research, the future scope of this project includes multiple fascinating paths for investigation and improvement. With an emphasis on audio categorization, the project offers up a number of avenues for further study and application. It combines spectrograms, Tensor Deep Stacking (TDSN) with a Classifier Attention Mechanism.

Exploration of Additional Datasets:

Future plans for the project's scope entail going beyond the existing dataset and perhaps adding bigger and more varied datasets. This extension might incorporate datasets encompassing a wider variety of audio genres, settings, and situations. Conducting experiments on several datasets would offer valuable perspectives on the model's flexibility and generalizability in a

range of acoustic scenarios.

Integration of Real-world Data:

A vital next step is to test the model using actual data from the real world. This entails evaluating its performance in situations outside of the training dataset's controlled context. The model should be able to handle additional obstacles that come with using real-world data, such as differences in recording quality, ambient noise, and different acoustic situations.

Fine-tuning Hyperparameters:

For additional improvement, hyperparameter tuning has to be investigated. Testing various combinations of learning rates, batch sizes, and model architectures may improve the accuracy and resilience of the model. Systematic tweaking can assist in determining the best configurations for enhanced performance.

Deployment in Practical Applications:

The future scope includes deploying the developed model in practical applications. This could involve integrating the model into software or devices that require audio classification capabilities, such as automated music tagging, content recommendation systems, or even in smart home applications for sound event detection.

References

- [1] Krishna Kumar, "Audio classification using ML methods" in IEEE Transactions on Audio, Speech, and Language Processing, 2023
- [2] H. Lu, H. Zhang, and A. Nayak, "A Deep Neural Network for Audio Classification with a Classifier Attention Mechanism" in International Journal of Computer Vision, 2021.
- [3] N. M. Patil and M. U. Nemade, "Music Genre Classification Using MFCC, K-NN, and SVM Classifier" in Proceedings of the [IJCERT], 2021.
- [4] P. Zinemanas, M. Rocamora, M. Miron, F. Font, "An Interpretable Deep Learning Model for Automatic Sound Classification" Electronics, vol. 10, no. 7, p. 850, April 2021, doi: 10.3390/electronics10070850.
- [5] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek, "Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals" in Proceedings of the [arXiv], 2020.
- [6] S. Abdoli, P. Cardinal, and A. L. Koerich, "End-to-End Environmental Sound Classification Using a 1D Convolutional Neural Network" in Journal of Computer Science and Technology (JCST) 2020.
- [7] A. Khamparia, D. Gupta, N. G. Nguyen, A. Khanna, B. Pandey, and P. Tiwari, "Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network" Journal of Computer Science and Technology (JCST)2020, 2020.
- [8] A. Hamza, "Deepfake Audio Detection via MFCC Features Using Machine Learning," in IEEE Access, vol. 10, pp. 134018-134028, 2020, doi: 10.1109/ACCESS.2022.3231480.
- [9] M. Esmaeilpour, P. Cardinal and A. Lameiras Koerich, "A Robust Approach for Securing

Audio Classification Against Adversarial Attacks" in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2147-2159, 2020.

[10] H. Phan, T.N.T Nguyen, P. Koch, A. Mertins. "Polyphonic audio event detection: multi-label or multi-class multi-task classification problem". In ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8877-8881). IEEE, 2020.

[11] Revay, Shauna, and Matthew Teschke. "Multiclass language identification using deep learning on spectral images of audio signals." arXiv preprint arXiv:1905.04348, 2019.

[12] Chen, Lei, Sule Gunduz, and M. Tamer Ozsu. "Mixed type audio classification with support vector machine." In 2006 IEEE international conference on multimedia and expo, pp. 781-784. IEEE, 2006.

[13] H. Jleed, M. Bouchard. "Open set audio recognition for multi-class classification with rejection." IEEE Access 8 (2020): 146523-146534.

[14] Gimeno, Pablo, "Generalizing AUC Optimization to Multiclass Classification for Audio Segmentation With Limited Training Data." IEEE Signal Processing Letters 28, 2021.

[15] M. Esmailpour, P. Cardinal and A. Lameiras Koerich, "A Robust Approach for Securing Audio Classification Against Adversarial Attacks" in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2147-2159, 2020.

[16] Várkonyi, D. Tamás, J. L. S. Junior, and T. Horváth, "Dynamic noise filtering for multi-class classification of beehive audio data." in Expert Systems with Applications 213 , 2023

- [17] Jleed, Hitham, and M. Bouchard. "Incremental multiclass open-set audio recognition." International Journal of Advances in Intelligent Informatics 8.2, 2022.
- [18] Sridhar, Sripathi, and M. Cartwright. "Multi-label Open-set Audio Classification." arXiv preprint arXiv:2310.13759, 2023
- [19] Pise, Reshma, and K. Patil. "A Deep Transfer Learning Framework for the Multi-Class Classification of Vector Mosquito Species." in Journal of Ecological Engineering 24.9, 2023.
- [20] Nanni, Loris, "Ensemble of convolutional neural networks to improve animal audio classification." in EURASIP Journal on Audio, Speech, and Music Processing 2020.1, 2020
- [21] Koutini, Khaled, Hami and G. Widmer, "Receptive field regularization techniques for audio classification and tagging with deep convolutional neural networks." in IEEE/ACM Transactions on Audio, Speech, and Language Processing 29, 2021.
- [22] https://www.tensorflow.org/api_docs
- [23] https://keras.io/guides/sequential_mode
- [24] <https://www.javatpoint.com/artificial-neural-network>
- [25] <https://www.analyticsvidhya.com/blog/2021/10/implementing-artificial-neural-network-classification-in-python-from-scratch/>

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none">• All Preliminary Pages• Bibliography/Images/Quotes• 14 Words String		Word Counts	
Report Generated on		Submission ID	Character Counts	
			Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com