DETECTING LOGGING OF FOREST TREES USING SOUND EVENT DETECTION

A Capstone Project report submitted
in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING (CSE)

By

Boda Raju	**19K41A0594**
Jinukala Vamshi	**19K41A0510**
Mohammed Raamizuddin	**19K41A0517**
Bhonagiri Shreya	**19K41A05E9**
Jupally Yochitha	**19K41A05F7**

Under the guidance of

Sallauddin Mohmmad

Assistant Professor, Department of CSE.

SR Engineering College

Innovation . Creativity . Entrepreneurship

SR ENGINEERING COLLEGE

Ananthasagar, Warangal.

CERTIFICATE

This is to certify that this report entitled **"Detecting Logging of Forest Trees Using Sound Event Detection"** is the bonafied work carried out by **Boda Raju, Jinukala Vamshi, Mohammed Raamizuddin, Bhonagiri Shreya and Jupally Yochitha** as a Capstone Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING (CSE)** during academic year 2022-2023 under our guidance and supervision.

Sallauddin Mohmmad
Asst. Prof., Department of CSE,
S R Engineering College,
Ananthasagar, Warangal.

Dr. M. Sheshikala
Assoc. Prof. & Head (CSE),
S R Engineering College,
Ananthasagar, Warangal.

External Examiner

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Mr. Sallauddin Md, Asst.Prof.** as well as Head of the CSE Department **Dr. M.Sheshikala, Associate Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advice and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr. Sallauddin Md, Asst. Prof., and R.Ashok Asst. Prof.** for their encouragement and support.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Principal, **Dr. V. MAHESH,** for his continuous support and guidance in completing this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

Automated detection of certain acoustic signals in environments is becoming an emergent field. One such area explored is the detection of logging of trees in forests. In forests, tree-cutting activities are illegal but due to a shortage of manpower and other resources, governments are not very successful in curbing this menace. One strategy to prevent this is to identify the tree-cutting process as soon as possible so that prompt action may be taken to halt it. The simplest method of early detection of tree cutting is to regularly monitor the forest area either manually or using some automatic techniques. As tree cutting generates a lot of noise, it can be detected by regularly monitoring the acoustic signals inside the forest. The tree-cutting sounds are recorded using a microphone, and the sound of various events such as axe knocking, chainsaw sound, and natural sounds such as wind are also recorded in the system and unwanted sounds from this are eliminated using Machine Learning technology. This method is modular when compared to the visual detecting method since it requires a lot of resources and cannot be implemented in a very vast area, it is easy to produce and energy efficient as it relies on audio evidence and uses powerful Machine Learning algorithms. The system can be adapted to different forest features and can be used equally during the day and night. The system is built by collecting audio data of various sounds related to the logging of trees and important features from this audio data are extracted, which is then further used to train the machine learning and deep learning algorithms used to implement this system. The models trained are CNN, CRNN, BI-CRNN and ResNet-34. BI-CRNN gives the best accuracy of 95%. This system also holds the capability to extend its use beyond the chosen environment by training the model with suitable audio data for the needed environment.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACRONYM	ABREVIATION
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
MFCC	Mel-Frequency Cepstral Coefficients
STFT	Short-Term Fourier Transform

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW:

Forests play a critical part in the preservation of the earth's global biodiversity and ecological equilibrium. In general, forest cover is critical across the world and serves as a measure of the planet's overall health. It has been widely said that forests adequately filter the air, conserve watersheds, reduce erosion, improve water quality, and supply natural resources. Furthermore, forests help to mitigate global warming by absorbing a large amount of carbon dioxide, the principal greenhouse gas, and so help to safeguard the world from climate change. According to numerous estimates, around 1.6 billion people worldwide rely on forest settings for their livelihoods, and roughly 60 million indigenous people rely heavily on forests for their existence and subsistence. In India, forests cover a wide area of land of which the major part remains unexplored due to the landscapes they are present in this becomes an advantage to the people who want to exploit the natural resources present in it. As a result, increasing the efficacy of surveillance for unlawful fires and logging is required. On the other hand, because of a lack of human resources, environmental money, and other resources, onsite monitoring by staff patrols with on-ground control and observation towers is too expensive and time-consuming to offer capillary and widespread monitoring. Therefore, automated detection approaches are required.

Many variables influence the survival and sustainability of forests. Illegal logging is a major problem that can result in uncontrolled and irreversible deforestation. Furthermore, illegal logging occurs. Because forests maintain about 90% of terrestrial biodiversity, they pose the greatest danger to biodiversity. Furthermore, illegal logging endangers the viability of forest ecosystems and can lead to widespread deforestation, which significantly impacts the environment. Flash floods, landslides, drought, climate change, and global warming are the primary consequences of illegal logging. Illegal logging also reduces government income and may contribute to the growth in poverty. Illegal logging operations have an impact on forest-rich countries as well as numerous countries that import and use various wood-based goods from wood-producing countries. Because of the nature of the activity, it is sometimes hard to correctly determine the scope or volume of illegal logging. Illegal forest operations are projected to cost governments throughout the world between USD 10-15 billion in yearly income. Furthermore, it has been said that in the most vulnerable forest zones, more than half

of all logging activities are carried out illegally. Curbing this illegal logging of trees is needed to preserve biodiversity and reduce the opposability of a natural disaster.

The automated system which identifies these illegal activities should be built using a large amount of data, since the system should be constructed using machine learning and deep learning algorithms the data that is going to be used for training this algorithms should not be redundant, they should be easily distinguishable such that the features that will be used to train the models are highly accurate in classifying the sounds to a particular class. In our case the data that is going to be used should be in an audio format, the audio files can either be recorded manually or can be collected through an already available audio dataset over the internet. People who have already worked on a similar problem have chosen data that is already available in machine learning repositories like UCI and Kaggle, namely UrbanSound8k which consists of all the sounds that an urban area can have including domestic animal sounds but the data available is not completely related to our problem, it consists of multiple classes which are related to the environment. In our case, we need audio files which determine sounds such as chainsaw, axe knocking etc. The sounds which are related to trees will only become the positive labels of our system. Based on the sounds an accurately predicting model will be developed using various algorithms. The various classes that are going to be considered for this problem are chainsaw sounds, axe knocking sounds, wind sounds etc.

In the next phase, where the data has been collected and the audios have been identified as distinguishable, the data needs to be pre-processed. The audio files will be pre-processed by using some readily available tools to match the sample rate of all the audio files as one. Next, the files will be spilt into many parts of ten seconds each either manually or by using the audio tools. If the audio files are not spilt into smaller ones the features cannot be extracted because of the large audio file which creates very high-dimensional features over a very long period. After the above pre-processing steps the features are extracted from the audio files by using the librosa library available in python which has many feature extraction techniques such as Mel-frequency Cepstral Coefficients (MFCC’s), Spectral Density, Spectral Contrast, Tonnetz etc. Many people have used the above feature extraction techniques which have given good results over time, they have used it for many problem statements which require these feature extraction techniques for audio files. This phase is the most crucial phase for the development of the system which determines the vectors that will be used to train the models for developing the automated system. These generated features or

vectors can either be single-dimensional or multi-dimensional depending on the way or method that has been chosen by the system developers. This feature extraction phase sets the ground for the training of the models that predict the outcome.

After the feature extraction phase the extracted features from the audio files will be used for training the machine learning and deep learning models where the features extracted will be considered as the processed data whereas the audio files which were used to construct these features will be considered as raw data. The processed data either single-dimensional or multi-dimensional will be split into multiple parts which are training, testing and validation sets where each set determines its significance by its name, the training set will be used to train the model, the validation set will be used to validate the training of the models which tests the model while training itself whereas, the testing set will be used to test the model once the training has been completed successfully. Each set of the data is not redundant, no data of processed data will repeat in any set. Many people who have worked in this domain by using audio files to train the models have gone with the above specified approach, it is the most generalized approach as of now. The various models that can be used to train are Logistic regression, Decision trees, Random Forest, Neural Networks etc. For the specified problem the better model would be a customized neural network which will accurately predict the class of the audio being given or recorded.

In this article, we determine our contribution in the field of machine learning using acoustic signals, we introduce an acoustic surveillance-based methodology for detecting logging in a forest. The presented methodology is modular and since it relies on audio evidence, it can be adapted to different forest characteristics and can be operated equally well during day and night. We have used the same flow of the system development cycle mentioned in the above paragraphs, brief research has also been performed on the systems that have been developed in this field by other authors and scientists. The data has been collected by us manually by recording the sounds of the needed classes and by using the AudioSet framework of google which provides the sounds of the needed classes. The popular feature extraction techniques have been used to extract the features from the sounds and custom developed models have been used for training and developing the system. At the end the best accurate model will be developed for the determined problem. Through this work we hope to contribute and help develop a good model which will be accurate in identifying sounds and alerting the concerned authorities. The aim of this system will be modular and can be used for implementation in any domain.

1.2 MOTIVATION AND SCOPE OF PROJECT

Surveillance is one of the most promising and emerging areas of ML development. All around the world there are numerous scenarios where there is a need to detect forest anomalies early and efficiently. Recent advances in ML have led to a promising performance in many areas of its applications. The aim of this project is to use ML to detect the logging of trees in the forest effectively such that the authorities are informed on time, this eliminates the need of manpower for surveillance.

SCOPE:

- To detect the logging of trees using machine learning.
- To enable authorities to utilize Machine Learning as a part of their work for monitoring resources.
- To let the Authorities, know the anomalies in forest effectively and in time.

1.3 PROBLEM STATEMENT

Forest Authorities are facing challenges that are monitoring the resources effectively and efficiently. In Forests, tree cutting activities are illegal but due to shortage of manpower and other resources, governments are not very successful in curbing this menace. Monitoring the forest assets can be difficult using manpower since, there is a lot of ground to cover and monitoring them visually also requires a lot of equipment that is laid out in a large area and cannot be effective in the day as well as night. The issue of this level of illicit logging must be dealt with very seriously as it exhausts the forest assets. Authorities are trying to adapt to the new technologies available, they are trying to incorporate AI into their surveillance methods such that it can become more efficient and effective.

1.4 EXISTING METHODS

1.4.1 Illegal Tree logging Detection

The detection of illegal logging in forests has attracted great interest in the research community mainly due to the substantial effect it has on the environment, the economy, and society, and therefore many studies in the literature aim at automatically detecting logging in forests. Several techniques can be employed to detect illegal tree cutting. One method involves analysing satellite imagery to identify changes in the landscape, such as the disappearance of trees. Drones equipped with cameras can also be used to monitor forests and detect illegal tree-cutting by providing high-resolution images. Acoustic sensors can be placed in strategic locations throughout the forest to detect the sound of chainsaws or other cutting tools. Ground patrols can be utilized to monitor forests and detect illegal tree-cutting by sending personnel to walk through the forest and look for signs of illegal activity. A complete presentation of methods and works on environmental sound recognition can be found in [1] with many works and studies on illegal logging in various urban and forest environments in [2]. Most systems have relied on wireless sensor networks (WSNs) and have utilized sound sensors to detect the operation of chainsaws, as well as vibration sensors to specify the exact position where logging was taking place in a forest.

Below are few limitations found from the existing detection systems:

1. Limited Resources: One of the major limitations is limited resources. Illegal tree-cutting often occurs in remote areas that are difficult to access, making it difficult to deploy surveillance equipment or conduct regular patrols.
2. False Positives and Negatives: Another limitation is the possibility of false positives and false negatives. False positives occur when legitimate tree-cutting is mistaken for illegal activity, while false negatives occur when illegal activity goes undetected.
3. Limited Coverage: Most detection methods have limited coverage, which means they can only monitor a small area at a time. This makes it difficult to detect illegal tree-cutting in large forests or areas where monitoring is difficult.
4. Weather Conditions: Weather conditions such as heavy rain, fog, or snow can make it difficult to detect illegal tree-cutting using satellite imagery or drones.
5. Human Error: Human error is another limitation, as personnel monitoring the forest may miss signs of illegal activity or misinterpret data.
6. Cost: The cost of deploying and maintaining surveillance equipment and personnel can be high, making it difficult to implement large-scale monitoring programs.

1.4.2 Audio Classification

Audio classification involves categorizing audio signals based on their unique characteristics. Many methods can be used for audio classification, including machine learning techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Support Vector Machines (SVMs), as well as statistical models like Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs), and K-Nearest Neighbors (KNN). Deep Belief Networks (DBNs) are another type of deep neural network that can be used for audio classification. Each method has its own strengths and limitations, and the most appropriate method may vary depending on the specific application and available data. These techniques have many practical applications, from speech recognition to music genre classification, environmental sound classification, and more.

Below are few limitations found from existing classification system:

1. Limited Data: Audio classification models require a large amount of high-quality training data to be effective. However, it can be challenging to collect and label large datasets, particularly for less common audio classes.
2. Noise and Variability: Audio signals are often contaminated by background noise, reverberation, and other sources of variability, which can affect the accuracy of classification models.
3. Domain-Specificity: Audio classification models trained on one dataset or domain may not perform well on a different dataset or domain, making it necessary to retrain models for each new application.
4. Interpretability: It can be difficult to interpret the decisions made by audio classification models, particularly deep learning models, which may be seen as a "black box" approach.
5. Computational Complexity: Some audio classification models, particularly deep learning models, can be computationally expensive to train and require significant computing resources.
6. Bias: Audio classification models can be biased towards certain classes or features, which can result in unequal treatment of different groups or inaccurate predictions.

1.5 LITERATURE SURVEY

Jia-Ching Wang et al. [3] presented effective ambient sound identification system for home automation in this paper. Depending on the sound classes that are discovered, certain home automation services can be activated. To achieve they did it using two main methods: frame-based multiclass support vector machines and independent component analysis Mel-frequency cepstral coefficients for sound detection, respectively, and signal-to-noise ratio-aware subspace-based signal augmentation. They achieved an accuracy of 86.7% with this method. Siddharth Sigtia et al. [4] Compared different machine learning algorithms as a function of their computing cost in this study. Comparing the rate of performance deterioration that happens when different Automatic Environmental Sound Recognition algorithms are scaled down by a comparable number of computing operations is a corollary to this subject since one class of algorithms may be more resistant to downscaling than another. Finally, they found out that GMM provides a low computational cost.

Shrikanth Narayanan et al. [5] Focused on the identification of ambient noises in this study, with a special emphasis on feature extraction using the matching pursuit (MP) approach. When other audio characteristics, such as MFCCs, fall short in describing sounds, MP offers a solution. When it comes to background noise, they are more durable. The unique use of MP for feature extraction and its usage in unstructured audio processing is the paper's contribution. Ishitaq Ahmad et al. [6] described a method in this research that uses the hidden Markov model (HMM) approach for classification and the Mel frequency cepstral coefficients (MFCCs) methodology for feature extraction to identify drones from the noises made by their propellers. Two feature vector approaches (one utilising twenty-four MFCCs and the other using the proposed thirty-six MFCCs) are used in the feature extraction step, The HMM-based classifier is then trained using the extracted features giving an accuracy of 100% with drone sounds.

Geard Roma et al. [7] address the issue of feature aggregation for auditory scene recognition in unlabelled audio. They specify a fresh set of descriptors that may be derived from a time series of audio descriptors' similarity matrix using Recurrence Quantification Analysis (RQA). In the framework of the AASP D-CASE competition, their findings suggest that using RQA will improve the event detection accuracy when compared with the usual algorithms finding accuracy. Agnes Incze et al. [8] presents and evaluates a CNN system for categorising bird sounds using various setups and hyperparameters. A dataset obtained from

the Xeno-canto bird song sharing portal, which offers a sizable collection of labelled and categorised recordings, is used to fine-tune the MobileNet pre-trained CNN model. The accuracy decreases whenever the classes are increased, they have achieved an accuracy of 80% with two classes.

Qiang Yu et al. [9] provide a spike-based paradigm for the ESR problem from a more cerebral standpoint in this paper. Thier framework is a unifying system that consistently combines the three key functional components of fast learning, robust readout, and sparse encoding. Their findings demonstrate the benefits of multispike learning and serve as a selection guide for different spike-based advancements. Boon-Yaik Ooi et al. [10] did this research to assess how practical it is to listen for machine operations. It is done using sound recognition for operation status tracking. They specifically assess the efficiency of Mel Frequency Cepstral Coefficient (MFCC) to identify actual machine sound and infer the machine's operational condition. The trial findings, which indicate an accuracy of 95.4% and loss of 0.04% percent, are encouraging.

O.K. Toffa et al. [11] introduce a novel method for categorising ambient noises that combines audio data with a textural feature called a local binary pattern (LBP). ESC-10 and ESC-50 datasets were used to assess their system using traditional machine learning techniques such support vector machines (SVM), random forests, and k-nearest neighbour (kNN). Their best mixed model, which combines LBP features and audio descriptors, produces results that are at the cutting edge of ambient sound categorization, scoring 88.5% on ESC-10 and 64.6% on ESC50. Luka Vujosevic et al. [12] use image classification to address the sound classification issue, they associate sound files with their corresponding picture representations, namely the mel spectrogram, tonal centroid, spectral contrast, and chromagram, and train a CNN deep learning network using these image representations. Using 10-fold cross validation, the suggested approach obtains a mean accuracy of 73%. The experimental findings also demonstrate a significant accuracy advantage of the deep learning technique over fully connected NN 59% accuracy.

Mesaros et al. [13] focuses on the development of evaluation metrics for polyphonic sound event detection (SED) systems. The authors proposed two metrics to measure the performance of polyphonic SED systems, the error rate, and the F-measure. They tested these metrics on two datasets, which are the TUT Sound Events 2016 and the DCASE 2016. The results showed that the proposed metrics were effective in evaluating polyphonic SED

systems. T. Heittola et al. [14] describes the Tampere University of Technology (TUT) database for acoustic scene classification and sound event detection. The dataset consists of over 6,000 audio clips with 15-second duration, and they are divided into 15 classes of acoustic scenes and 11 classes of sound events. The authors provided baseline results for both acoustic scene classification and sound event detection using various feature extraction and classification methods.

E. Çakır et al. [15] proposed a convolutional recurrent neural network (CRNN) for polyphonic sound event detection. The authors used the TUT Sound Events 2016 dataset for training and evaluation. They compared the proposed method with other state-of-the-art methods, and the results showed that the CRNN outperformed the other methods in terms of the F-measure. Min Xu et al. [16] proposed a fusion scheme of visual and auditory modalities for event detection in sports videos. The authors used a dataset consisting of 28 hours of broadcast tennis videos. They extracted both visual and auditory features and fused them using a score-level fusion method. The results showed that the proposed method achieved better performance compared to using either modality alone.

Y. Zigel et al. [17] proposed a method for automatic fall detection of elderly people using floor vibrations and sound. The authors used a human-mimicking doll to simulate falls and recorded the corresponding signals. They proposed a classification system based on a machine learning approach that combines features from both the vibration and sound signals. The features were extracted from the signals and used to train a support vector machine (SVM) classifier. The classifier was able to detect falls with high accuracy and low false alarm rates. The authors evaluated their method on a dataset containing recordings of both falls and non-fall events. The results showed that the system achieved a high detection rate of falls with a low false alarm rate. The system also performed well in detecting falls of varying severity and under different conditions, such as different surface materials.

Dong Zhang et al. [18] proposed a semi-supervised adapted Hidden Markov Model (HMM) approach for unusual event detection. They introduced a system that learns a model from a few examples of the normal behavior of a scene and detects any rare or abnormal behavior. They evaluated their approach on two datasets, one with videos of office behavior, and another with traffic scenes. Their results show that their semi-supervised method can achieve a better performance in detecting abnormal events compared to fully supervised approaches. C. Clavel et al. [19] proposed an audio-based surveillance system that can detect

various types of events, such as breaking glass, voices, and gunshots. They used a spectral clustering approach to detect the events and evaluate their system on an annotated dataset with different sound events. The results show that their system achieves a high detection rate with low false-positive rates.

G. Valenzise et al. [20] proposed a system for detecting and localizing screams and gunshots in audio-surveillance systems. They used Mel-frequency cepstral coefficients (MFCCs) and Gaussian mixture models (GMMs) to extract the features and classify the events. They tested their system on a dataset that contains various types of sounds and evaluated the performance using the receiver operating characteristic (ROC) curve. The results show that their system can detect and localize the events with high accuracy. D. Stowell et al. [21] proposed a system for detecting and classifying acoustic scenes and events. They used a deep convolutional neural network (CNN) to learn the features from the audio signals and then used a support vector machine (SVM) for classification. They evaluated their system on two datasets, one for acoustic scenes and another for bird vocalizations. Their results show that their system outperforms previous state-of-the-art methods.

D. Ruinskiy et al. [22] proposed an algorithm for detecting and demarcating breath sounds in speech and song signals. They used a combination of time and frequency domain analysis to detect the breath sounds, and then applied morphological operations to demarcate them. They evaluated their algorithm on a dataset that contains various types of speech and song signals and compared their results with those of other methods. The results show that their algorithm outperforms previous state-of-the-art methods in detecting and demarcating breath sounds. In summary, the proposed algorithm provides a robust solution for breath sound detection and demarcation in speech and song signals, with high accuracy even in noisy environments.

Ref No.	Domain	Type of Data set	Feature Extraction	Models Used	Results (Accuracy)
[3]	Home	Home Sounds Events	ICA-transformed MFCC's, Perceptual Features	SVM	File: 83.5% Frame: 86.7%
[4]	Environment	Audio data environmental sounds	MFCC's, Spectral Centroid, Spectral Flatness, Spectral rolloff and zero crossing rate	GMM, SVM	GMM EER: 14.0% SVM EER: 13.5%
[5]	Environment	Environment Sound Events from audio data	Matching Pursuit (MP)	KNN, GMM	MP: 72.5% MFCC: 70.9% MP+MFCC: 90%
[6]	Environment, Drones	Drone Sounds, Environment Sounds	MFCC	GMM	Accuracy: 80%
[7]	Environment, Domestic	Auditory events, Environmental Sound	RQA, MIR, AR	HMM	Frame-based: 1.46 Event-based: 3.33 Class - based: 3.41
[8]	Birds, Environment	Bird CLEF	STFT, Colormap, Normalization	CNN	Accuracy: 74%
[9]	Human Speech	Speech Babble	KP Encoding Frontend, STFT	CNN, DNN	Accuracy: 90%

[10]	Machines	Machine Sound Events	FFT, MFCC	Euclidean distance function, DTW	Accuracy: 95.4%, Error rate: 0.04%
[11]	Environment	ESC-50	MFCC, GFCC, CQT	CNN, SVM	Accuracy: 88.5%
[12]	Urban Areas	UrbanSound8k	Mel-Spec, Tonnetz, Spectral Features	CNN	Accuracy: 73%
[13]	Street Environment	TUT Sound Events 2016	MFCC, CQT, HPCP, spectral features, and temporal features	SVM, RF, KNN	F-measure of 0.47 for SVM, 0.46 for RF, and 0.41 for KNN
[14]	Street Environment	TUT Acoustic Scenes 2016, TUT Sound Events 2016	MFCC, Mel-scaled spectrogram, and chroma features	GMM, SVM, and HMM	60.1% accuracy for acoustic scene classification and F-measure of 0.53 for sound event detection
[15]	Street Environment	TUT Sound Events 2017	Mel-scaled spectrogram and phase information	CRNN	F-measure of 0.52 for sound event detection
[16]	Sport Events	Audio-visual sports event dataset	MPEG-7 audio descriptors and visual features	MLP, SVM	87.6% accuracy
[17]	Fall Events	Self-created dataset of human mimicking doll falls	MFCC, filter-bank energies, and statistical features	GMM and SVM	100% accuracy for floor vibration and 93.5% accuracy for audio features

[18]	Unusual Events	Real-life dataset collected in meetings	MFCCs	HMM	78% Accuracy
[19]	Surveillance	Audio dataset of public spaces	Time-frequency representation of audio signals	HMM and SVM	75% Accuracy
[20]	Scream and Gunshots	Dataset of gunshot and scream sounds	Wavelet transform and local binary patterns	GMM and SVM	98% Accuracy
[21]	Acoustic Scenes	DCASE 2013 dataset	MFCCs and CQT	CNN	64% Accuracy
[22]	Singing and Breathing	Dataset of speech and song signals	MFCCs	HMM	96.7% Accuracy

Table 1.1 Survey Papers Comparison

The Previous table represents the comparative analysis of different papers that are published in the sound event detection domain, many authors have contributed to develop systems that help identify and distinguish audios or sounds that are particular to a scenario. The most used dataset by different authors is TUT sound events. This dataset consists of the sounds from the environment that contain different classes of sounds. The most used feature extraction technique is Mel-Frequency Cepstral Coefficients (MFCC) which uses Mel-Spectrogram to extract the crucial features from the audio wave form. The most used model according to the above comparative analysis is Convolutional Neural Network (CNN) and Hidden Markov models (HMM) which are a part of DL. CNN is mostly used on data that has high dimensionality and needs extra preprocessing before passing it to the layers on which the model is trained. The accuracy ranges between 40%-90% in this comparative study as the models used are different and dataset has high dimensionality.

CHAPTER 2

SOFTWARE TOOLS

2.1 PYTHON

The interactive, object-oriented, and high-level programming language Python is particularly well-liked. Python is a garbage-collected, dynamically typed programming language. Between 1985 and 1990, Guido van Rossum designed it. Python source code is also accessible under the GNU General Public License, much like Perl (GPL). Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high-level dynamic data types, and dynamic typing. There are interfaces to many system-calls and libraries.

It's applied for:

- software development
- server-side web development
- system programming
- mathematical problems.

Programming languages such as procedural, object-oriented, and functional are supported by Python. Python's design concept places a strong emphasis on code readability using deep indentation.

Why Python?

- Python is cross-platform compatible (Windows, Mac, Linux, Raspberry Pi, etc).
- Python is Interpreted - The interpreter processes Python as it is being used. The software does not need to be compiled before running. This is comparable to PHP and PERL and Python supports the Object-Oriented programming approach, which wraps code into objects.
- Python's syntax differs from various other programming languages in that it enables programmers to construct applications with lesser lines of code.

Main Concepts of Object-Oriented Programming (OOPs) are Class, Objects, Polymorphism, Encapsulation, Inheritance, Data Abstraction. AI researchers are fans of Python. Google TensorFlow, as well as other libraries (scikit-learn, Keras), establish a foundation for AI development because of the usability and flexibility it offers Python users. These libraries, and their availability, are critical because they enable developers to focus on growth and building.

2.1.1 Libraries

This section specifies the libraries that are used for the pre-processing of the data and the development of the model for classifying the images accordingly:

2.1.1.1 TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

2.1.1.2 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

2.1.1.3 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidM

2.2 JUPYTER NOTEBOOK

The most recent web-based interactive development environment for code, data, and notebooks is JupyterLab. Users may create and arrange workflows in data science, scientific computing, computational journalism, and machine learning using the interface's flexibility. A modular structure encourages expansions to increase and improve functionality. It is the first web application for producing and sharing computational documents. It offers an easy-to-use, efficient, document-focused interface. It is a server-client programme that enables web-based editing and running of notebook pages. The Jupyter Notebook App may be used locally, without an internet connection, or it can be deployed on a remote server and viewed online.

2.2.1 Kernel

The "computational engine" that runs the code in a Notebook document is called a notebook kernel. Python code is executed by the ipython kernel, which is mentioned in this manual. There are kernels for several more languages (official kernels). A Notebook document's related kernel is immediately launched when you open it. The kernel does the computation and generates the results when the notebook is run (either cell-by-cell or through the menu Cell -> Run All). The kernel may use a lot of CPU and Memory, depending on the type of computations. The Memory is not freed until the kernel is terminated, which you should be aware of.

2.2.2 Notebook Dashboard

When you start the Jupyter Notebook App, the feature that is displayed immediately is the Notebook Dashboard. The Notebook Dashboard is primarily used to handle the active kernels and open notebook papers.

A file manager's additional functionality, such as folder navigation and file renaming/delete, are also available in the Notebook Dashboard.

2.3 VISUAL STUDIO CODE

Working with Python in Visual Studio Code, using the Microsoft Python extension, is simple, fun, and productive. The extension makes VS Code an excellent Python editor and works on any operating system with a variety of Python interpreters. It leverages all of VS Code's power to provide auto complete and IntelliSense, linting, debugging, and unit testing, 13 along with the ability to easily switch between Python environments, including virtual and conda environments.

CHAPTER 3

PROJECT IMPLEMENTATION

3.1 DESCRIPTION

Computers can analyze audio in a very distinguishable range of frequencies, they can be used to classify sounds which can be overheard by us humans at that specific interval of time. Thus, they can be used to detect the logging of trees efficiently with an improving accuracy over time. The Proposed methodology is to build a system that can effectively do this task by using some acoustic sensors known as monitoring stations which will be used to analyze and detect these sounds. The monitoring stations can be put up according to the authorities, areas which are highly suspicious of illegal logging.

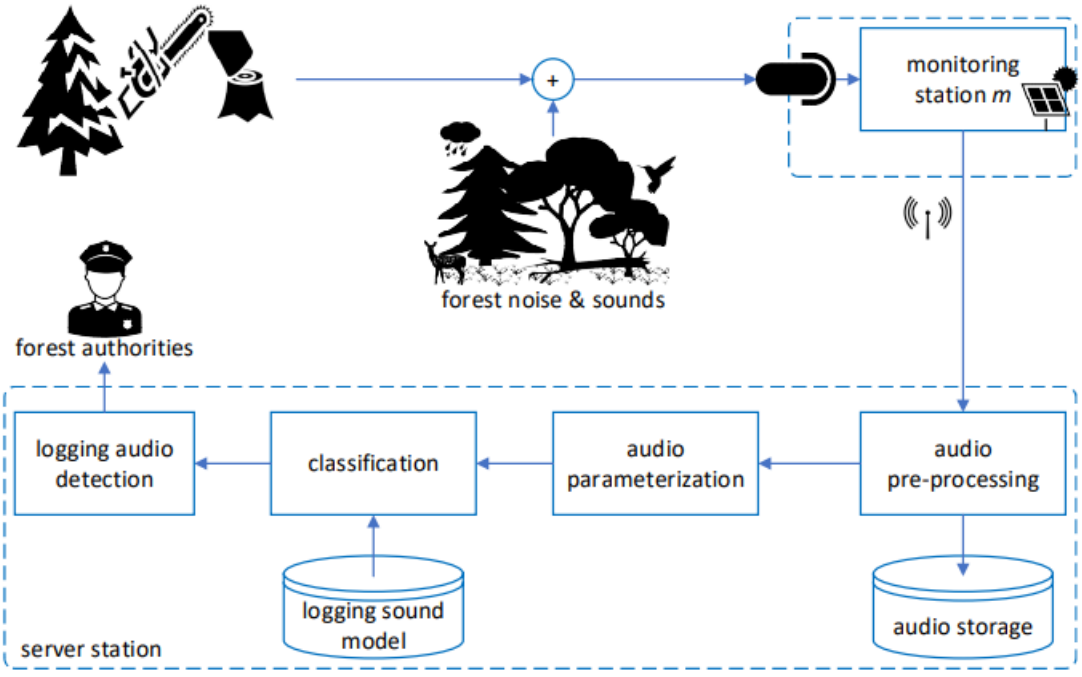


Fig.3.1 Block Diagram of the Concept.

As can be seen in Figure 3.1, the monitoring station has a microphone (which can be expanded to microphone array), a solar panel for energy autonomy, and an antenna for wireless communication with a base station (server) or a wired communication with it. The microphone captures sound events and the acquired audio samples are sent wirelessly or wired to a server for further processing. Any logging sound, at a distance that can be heard, is captured by the microphone, together with additive forest sounds and environmental noise also. At the server side, the captured audio signal, which is transmitted from monitoring station m, is preprocessed, and parameterized (Feature Extraction) before being analyzed by

machine learning methods for classification to detect logging sounds. The detection is performed using pretrained acoustic models for logging and the classification is binary, i.e., detection of logging sounds or not. Once a logging activity is detected, an alarm is activated to inform forest authorities. This can be done either by direct connection to a forest management/monitoring system and activation of the corresponding alarm or by an automatic phone call or text message to patrolling units.

3.2 DESIGN

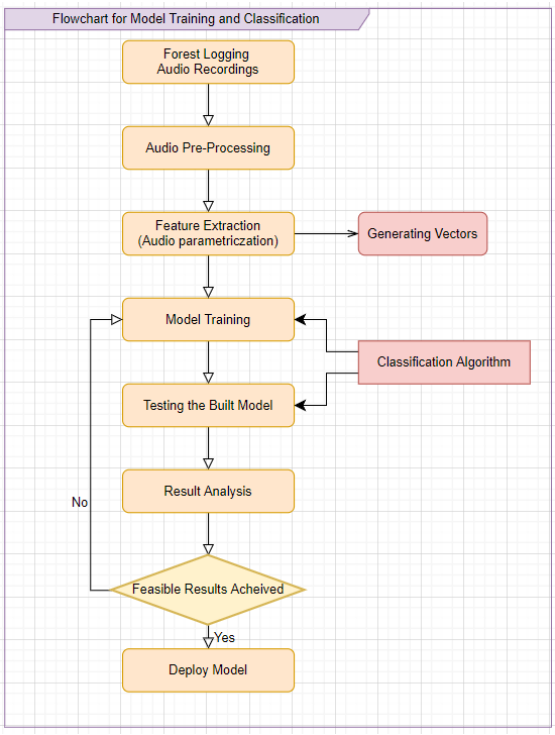


Fig.3.2 Flowchart for Model Training.

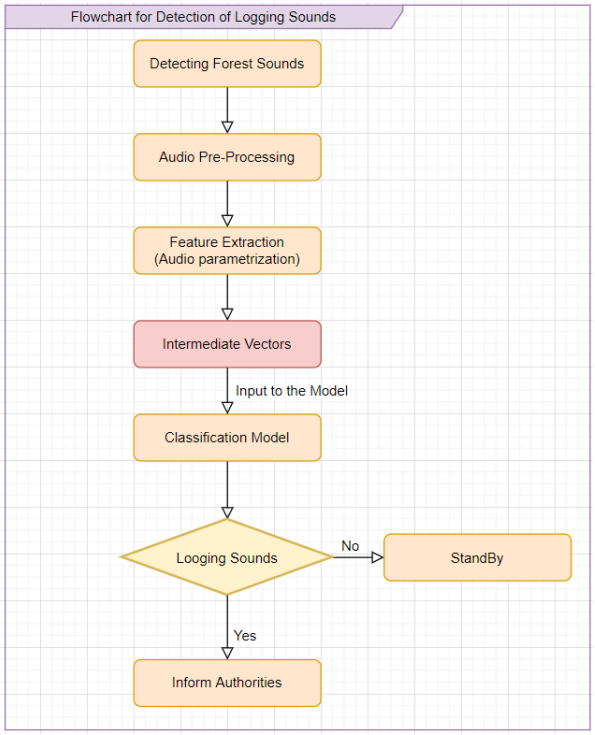


Fig.3.3 Flowchart for Logging Detection.

In Figure 3.2, it depicts the workflow of the Classification model training and testing. The first step in it would be to collect the data on which the model needs to be trained, in our case it is audio recordings of the logging sounds. These logging sounds will be preprocessed with some pre-determined techniques which can get the data ready from which we can extract crucial features and train the model with it, after which the model will be tested for feasible accuracy then deployed for real-time detection.

In Figure 3.3, it depicts the workflow of Detection of Logging Sounds, the first step in it would be to record the sounds occurring in the environment near, then preprocess the recorded sounds from the microphone and apply the feature extraction technique applied during the training phase and form the vectors which will be the input to the deployed model from the previous phase. The model will classify the sounds according to the classes

determined during training. If the sound is of logging, then the concerned authorities will be informed, and action will be taken.

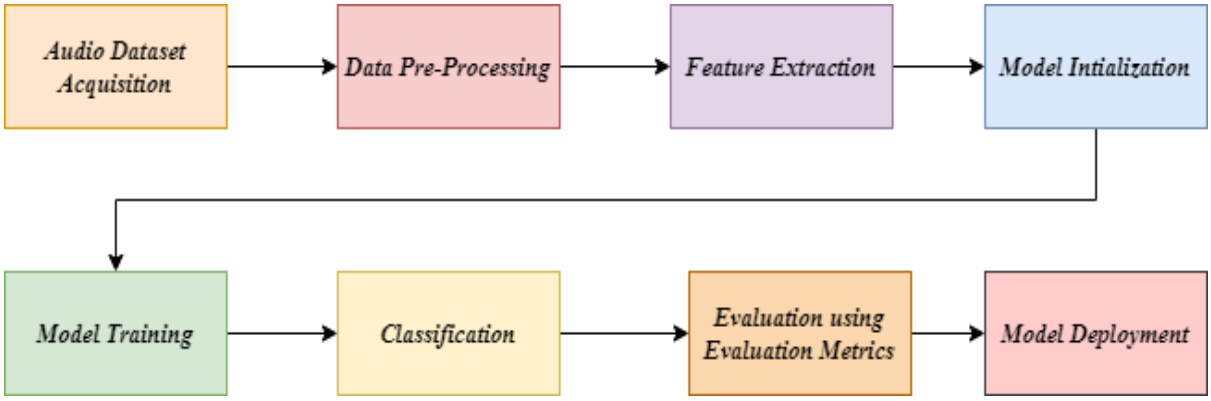


Fig.3.4 Workflow of the Audio classification system.

3.2.1 Dataset Acquisition

A dataset is defined as a collection of similar records. It is generally stored in tabular formats in which a column describes each parameter, and each row represents a separate record. These datasets can be used in artificial intelligence and data science to generate models for certain predictions. It can also be used for visualization and analysis of data for various purposes. But here in our application a dataset in a tabular format cannot be applied since the data is in waveform (i.e., Audio Signals) so, the data that we have is audio recordings of the sounds which can be classified as illegal logging of trees like cutting sounds, sawing sounds, falling sounds, and any data relating to the wood logging activity. As per our concern we have selected six classes for developing the classification model:

- Axe cutting Sounds.
- Chainsaw Sounds.
- Hand Saw Sounds.
- Forest Sounds.
- Rain & Thunder Sounds.
- Wind Sounds.

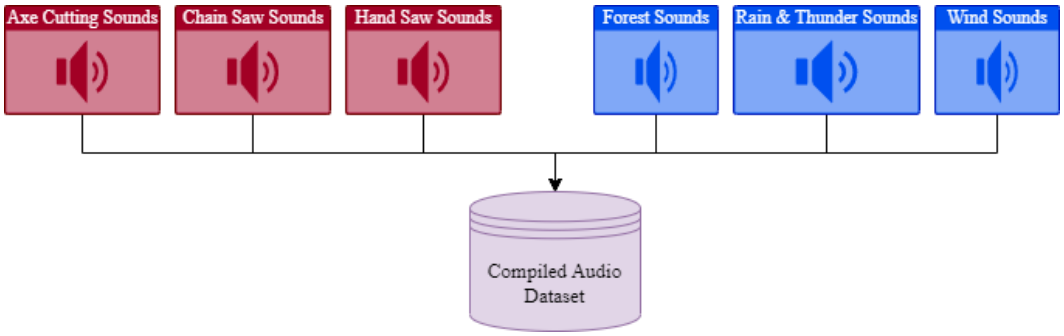
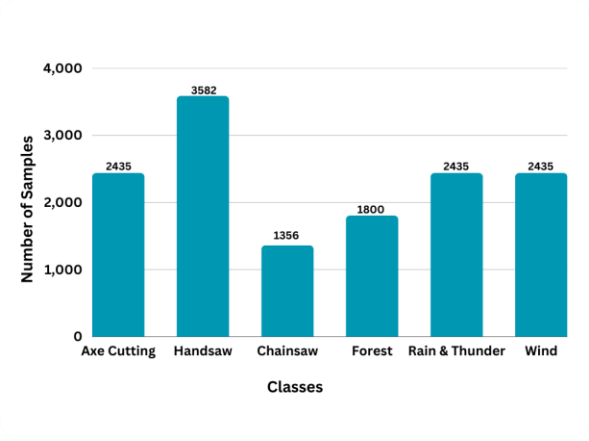


Fig.3.5 Compiled Dataset

Fig.3.5 represents how the compiled audio dataset is formed, these Sounds are collected from various resources like websites and videos form which the audio will be extracted. Some of the data is collected in real time. Since the collected data is noisy and imbalanced it needs to be balanced for training the model.



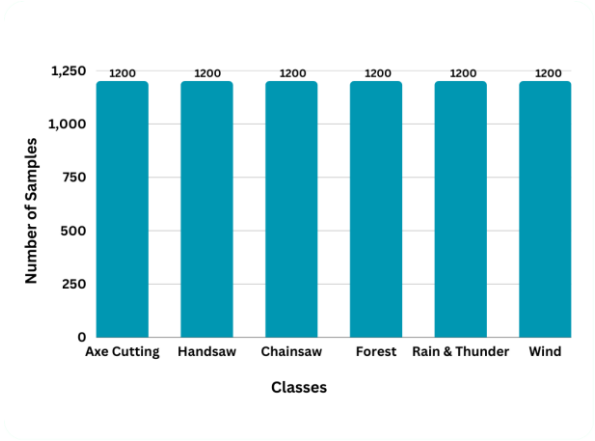


Fig.3.6 Imbalanced Dataset

Fig.3.7 Balanced Dataset

In Figure 3.6, it depicts the Imbalanced dataset and the number of samples in each class of the dataset the highest samples collected was for the handsaw class which were around 3600 samples and the lowest number of samples collected was for chain saw class which were around 1400 samples. This imbalanced dataset cannot be used for training the model since it will give bias results, a balanced dataset is preferred.

In Figure 3.7, it depicts the Balanced dataset and the number of samples in each class of the dataset, 1200 samples from each class of 10 seconds each have been selected to balance the dataset such that the model doesn't give bias results.

3.2.2 Data Pre-Processing

Pre-Processing the data is the process of cleaning the data and making it ready for the model to train. For numeric data, it can be normalization, feature engineering, removal of outliers etc. For images, techniques like resizing and normalization are used. It also involves splitting data for training and testing. The training data is used to train the model and testing data is then used to evaluate the performance of the model on different data. For the audio data the preprocessing techniques differ from the usual ones. These preprocessing techniques are in turn helpful for extracting features from the audio signals. The most popular techniques used for preprocessing are Short Term Fourier Transform (STFT) and Mel-Spectrogram. These two methods are helpful in representing the audio data in wave form (Frequency based images). This step also involves the splitting of audio signals into chunks of seconds each such that the model is trained rigorously.

3.2.3 Feature Extraction

Feature extraction is a process by which an initial set of data is reduced by identifying key features of the data for machine learning. It is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is many variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set. For the audio data the feature extraction techniques work hand in hand with the audio preprocessing techniques that are applied to the audio data. The most popular feature extraction techniques are Mel Frequency Cepstral Coefficients (MFCC), Linear Prediction Coefficients (LPC), Linear Prediction Cepstral Coefficients (LPCC), Line Spectral Frequencies (LSF), Discrete Wave Transform (DWT). These methods are helpful in extracting the crucial features form the audio signals that would help us detect the logging of trees. These methods take the processed audio data as input and give vectors as the output which will be used for training and testing purposes.

3.2.4 Model Initialization

The models implemented are dependent on the architecture of CNN, the general architecture of CNN is shown in the fig.3.8, the layers that must exist in a convolutional neural network for classification purposes are Convolution Layer, Pooling Layer, Flatten Layer, Dense Layer. The purpose and operation of these layers is discussed below.

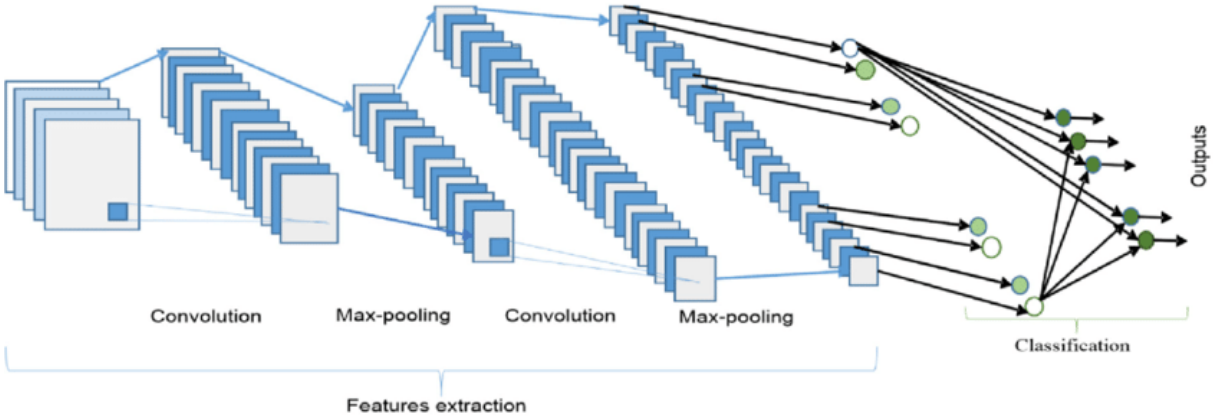


Fig.3.8 Convolutional Neural Network.

3.2.4.1 Convolution Layer

The convolution layer is used to create filters or feature maps for feature extraction from images. These feature maps indicate the strength of each pixel and thus play a crucial

role in image based deep learning applications. A sample [2,2] filter is shown with a [3,3] pixel image in fig.3.9.

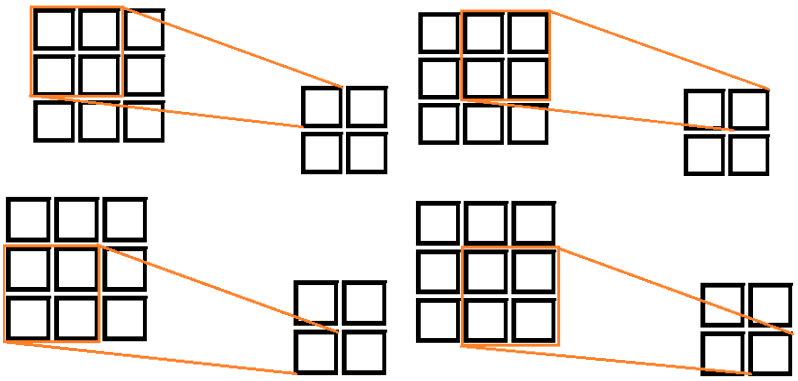


Fig.3.9 Convolution Filter

As seen in the Fig.3.9, the output image is a [2,2] pixel image. The formula to calculate the output image size is [(W−FW+2P)/(SH+1), (H-FH+2P)/(SV+1)] where W and H are width and height of input image, FW and FH are filter width and filter height, P is padding, and SH and SV are horizontal and vertical strides respectively. The value of a pixel in output image is calculated by adding the products of each pixel of filter with the corresponding pixel of image. The calculation is shown in fig.3.10.

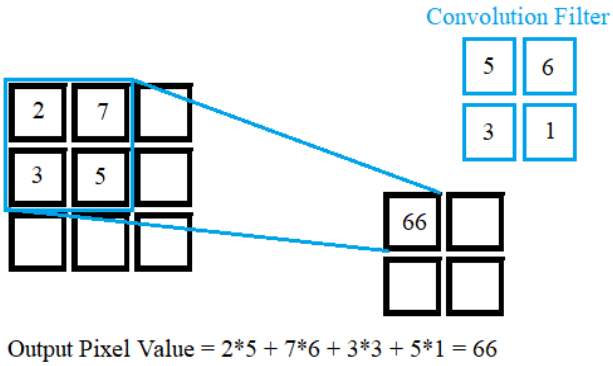


Fig.3.10 Convolution Filter Output Calculation

3.2.4.2 Pooling Layer

There are two types of pooling operations generally used in convolutional neural networks. Max Pooling and Average Pooling. It is used to reduce the size of an image i.e., reduce the number of pixels in the image. It is performed so that a smaller number of values are passed on to the next layer and the operations of further layers are done more quickly as less data is received. The Max Pooling operation extracts the maximum pixel value from each part of the image. A [2,2] Max Pooling operation on a [3,3] image can be seen in fig.3.11.

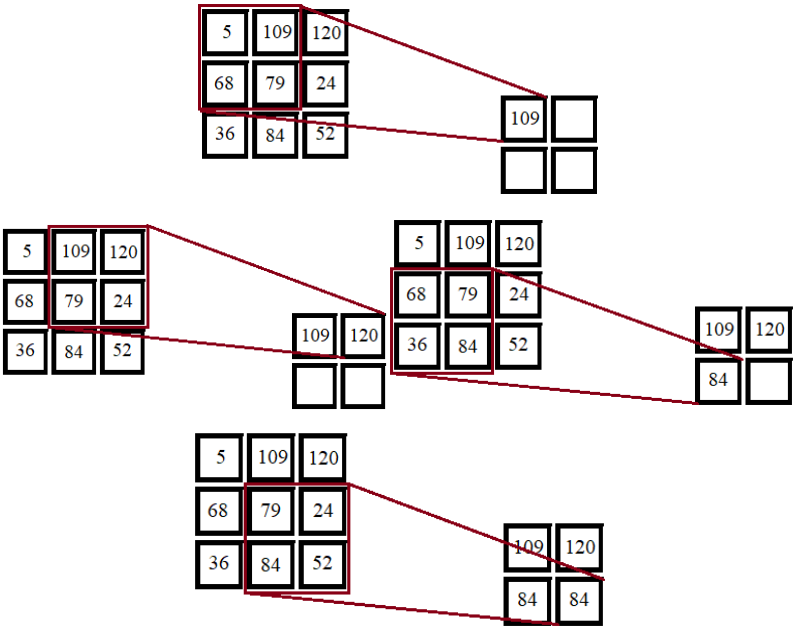


Fig.3.11 Max Pooling

3.2.4.3 Flatten Layer

The Flatten Layer is used to convert a multi-dimension image into a single dimension array to further use it in fully connected or dense layer. The flatten operation for a [2,2,1] image can be seen in fig.3.12.

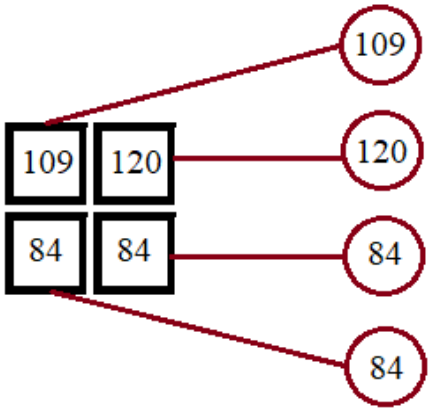


Fig.3.12 Flatten Layer

3.2.4.4 Dense Layer

Dense Layer is a basic layer of neurons in which each neuron receives input from all neurons in the preceding layer, thus the name. Dense Layer is used to classify images based on convolutional layer output. Each Layer of the Neural Network comprises neurons that compute the weighted average of their input, which is then processed through a non-linear function known as an "activation function." The result of this activation function is regarded

as the neuron's output. Similarly, the procedure is repeated for all neurons in all levels. The dense layer is depicted in the below fig.3.13.

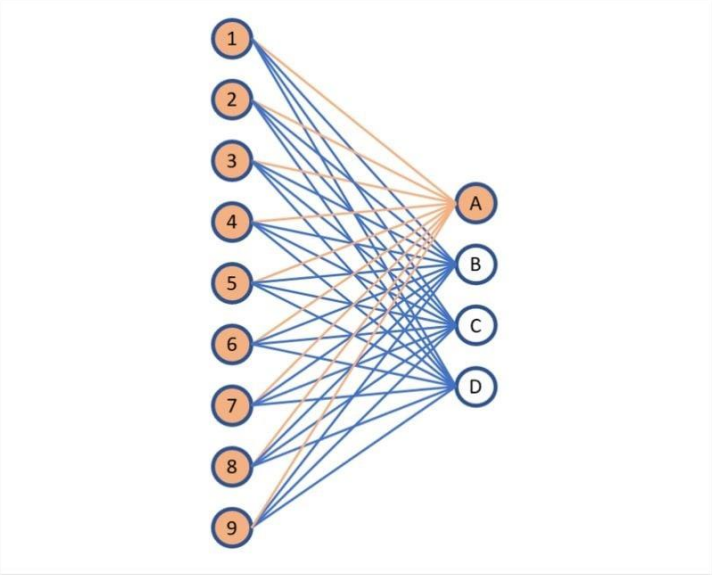


Fig.3.13 Dense Layer

3.2.4.5 Activation Functions

The activation functions are used to get a non-linearity in the output of a neuron. The most used activation functions in CNN are ReLU, Sigmoid and SoftMax. ReLU or Rectified Linear Unit activation function is generally applied after a convolutional layer or in hidden layers of a CNN model. The mathematical formula for ReLU activation function is [f(x) = max (0, x)]. The Sigmoid Activation function is used for binary classification and multi-label classification operations and is applied in the output layer. The formula for sigmoid activation function is [sig(x)=1/ (1+ e^-x)]. The SoftMax Activation is used for multi-class classification operation and is applied in the output layer of the CNN model. The formula for SoftMax activation function is as follows:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

3.2.5 Model Training

Giving the training data as input to the model for it to alter the parameters and make predictions is called model training. It is then evaluated on the testing data using accuracy score, confusion matrix, specificity etc. In our case the model will be trained using the vectors that are generated by the feature extraction techniques, these vectors can be passed to a Convolution Neural Network (CNN) to train it for the prediction and classification of the logging sounds.

3.2.6 Model Testing

In machine learning, model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. The testing set consisting of a set of testing samples should be separated from both training and validation sets, but it should follow the same probability distribution as the training set. Here, the testing dataset is used to test the performance of the fully trained model on the audio dataset and the performance analysis is done on it. If the model is giving Feasible results, then it is deployed in real time and the logging of trees can be detected effectively.

3.2.7 Classification Metrics

Classification metrics are tools used to evaluate the performance of a classification model by measuring how well it can correctly classify examples into their respective classes. There are various metrics that are commonly used to evaluate a model's performance, such as accuracy, precision, recall, F1-score, confusion matrix, and ROC curve. Accuracy measures the percentage of correctly classified examples out of all examples in the dataset, while precision measures the proportion of true positives out of all examples classified as positive. Recall measures the proportion of true positives out of all actual positive examples. The F1-score provides a balance between precision and recall by summarizing the overall performance of the model. A confusion matrix is a useful tool for visualizing the performance of the model and identifying areas where it may be making errors. Finally, an ROC curve is used to evaluate the performance of a binary classifier at different thresholds. The choice of classification metric depends on the specific application and desired trade-offs between different types of errors.

3.2.8 Model Deployment

In deep learning, the deployment of a classification model involves making the trained model available for use in a production environment. This process includes exporting the model in a format that can be used in the deployment environment, preprocessing the input data, integrating the model into the deployment environment, testing it to ensure accurate predictions, and monitoring its performance over time. There are several platforms and tools available for deploying deep learning models, such as TensorFlow Serving, Amazon SageMaker, and Microsoft Azure Machine Learning, which can be used to deploy models in a variety of environments. Successful deployment of a deep learning model can provide significant benefits for a range of applications, including image and speech recognition, natural language processing, and predictive analytics.

3.3 EXPERMIENTAL ANALYSIS

The Current implementation is based on the research and the best feature extraction techniques which provides feasible results, the implementation consists of pre-processing audio files, extracting features from them and executing different models to test the accuracy variances among them. For the implementation the dataset of six classes is considered.

3.3.1 Visualization of Audio data

Here, we analyze the audio data into visual forms by using the IPython.display.Audio module present in python programming language, below is the representation of audio classes that are taken into consideration in wave form.

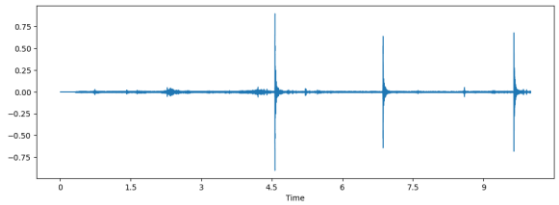


Fig.3.14 Axe cutting Sound.

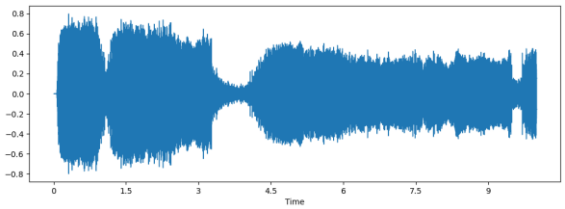


Fig.3.15 Chain Saw Sound.

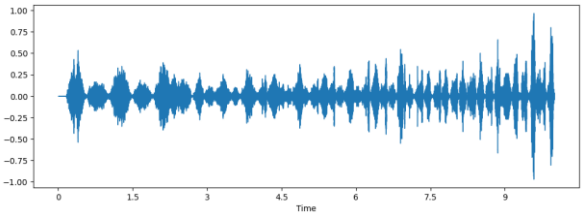


Fig.3.16 Hand Saw Sound.

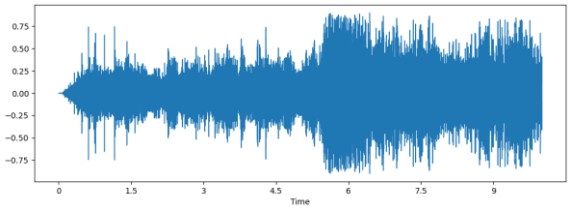


Fig.3.17 Rain & Thunder Sound.

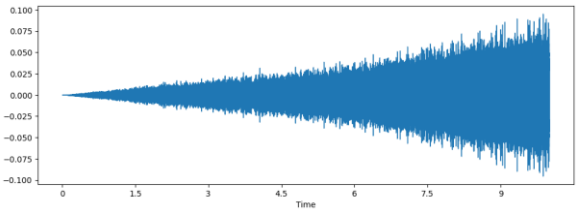


Fig.3.18 Wind Sound.

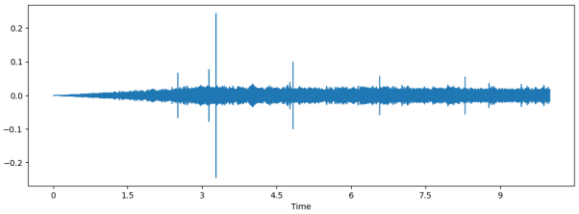


Fig.3.19 Forest Sound.

From the above six figures the difference between the six waveforms of the audio files is very high, the chainsaw produces a high frequency which is shown in figure 3.15 and the wind sounds produce a frequency on a very periodic level. This difference can be used and applied to the models which are going to be trained for these six classes and good accuracy can be achieved. The figures depict the frequency of the audio waves over a period at some points where the frequency is high or low. The next step in visualizing would be to analyze the meta-data of all the data files and store them in a .csv file which can be further used to extract crucial information at any time. The meta-data of the audio files is extracted

using the audio_metadata module present in python. This data provides information like sample rates and channels of the recorded audio.

3.3.2 Pre-Processing & Feature extraction

Here, we preprocess the audio data and convert them into vectors by using the librosa module present in python, below is the list of pre-processing and feature extraction techniques used for creating vectors of the audio data,

- Mel-frequency cepstral coefficients (MFCC).
- Short-term Fourier Transform (STFT).
- Chroma.
- Mel Spectrogram.
- Contrast.
- Tonnetz.

Below is a description of all the above feature extraction steps,

3.3.2.1 Mel-frequency cepstral coefficients (MFCC):

Mel-frequency cepstral coefficients (MFCC) are a feature extraction technique widely used in speech recognition and music analysis. MFCCs are based on the human auditory system's response to different frequency bands, where the frequency bands are represented in the Mel scale. The Mel scale is a non-linear scale that maps the frequency range of human hearing to a more perceptually relevant scale. The MFCC process involves several steps, including pre-emphasis, windowing, discrete Fourier transform (DFT), and logarithmic compression. The resulting features are a set of coefficients that represent the spectral envelope of the audio signal, which captures the most important information about the sound.

3.3.2.2 Short-term Fourier Transform (STFT):

The Short-term Fourier Transform (STFT) is a technique for analyzing the spectral content of a signal over time. STFT is obtained by dividing a signal into small, overlapping time segments and calculating the Fourier transform of each segment. The resulting spectrum for each time segment is then used to create a spectrogram, which is a 2D representation of the signal's frequency content over time. STFT is commonly used in audio processing, speech recognition, and music analysis. By analyzing the spectrogram, we can identify changes in the spectral content of a signal over time, which can help us to identify different sound events or features in the signal. STFT is widely used in audio processing due to its ability to capture both time and frequency information in a signal. It is a fundamental tool for analyzing and manipulating audio signals, including filtering, noise reduction, and feature extraction.

3.3.2.3 Chroma:

Chroma is a feature extraction technique that represents the tonal content of music. The chroma features are derived from the frequency spectrum of the audio signal by mapping the spectral content onto a 12-dimensional pitch class space, where each dimension represents one of the 12 pitch classes. Chroma features are commonly used in music analysis, including genre classification, chord recognition, and melody extraction. By analyzing the chroma features of a piece of music, we can identify its tonal characteristics and extract useful information about the underlying musical structure.

3.3.2.4 Mel Spectrogram:

Mel Spectrogram is a visual representation of the spectral content of an audio signal. Mel Spectrogram is obtained by applying the Mel filter bank to the magnitude spectrum obtained from the Short-term Fourier Transform (STFT) of the audio signal. Mel Spectrogram is commonly used in speech and music analysis, as it provides a compact representation of the spectral content of the audio signal. By analyzing the Mel Spectrogram, we can identify important features of the audio signal, such as formants in speech or musical notes in music.

3.3.2.5 Contrast:

Contrast is a feature extraction technique that highlights the differences between adjacent frequency bands in the audio signal. Contrast is obtained by calculating the difference between the spectral energy in adjacent frequency bands. Contrast features are commonly used in speech recognition and music analysis. By analyzing the contrast features of an audio signal, we can identify changes in spectral content that are important for recognizing different sound events.

3.3.2.6 Tonnetz:

Tonnetz is a feature extraction technique that represents the harmonic relationships between musical notes. Tonnetz is based on a network of nodes, where each node represents a musical note, and the edges represent the harmonic relationships between the notes. Tonnetz features are commonly used in music analysis, including chord recognition and harmony analysis. By analyzing the Tonnetz features of a piece of music, we can identify its underlying harmonic structure and extract useful information about its tonal relationships.

Each of the techniques mentioned above extracts a specific type of feature from the audio signal, such as the spectral envelope (MFCC), tonal content (Chroma), or harmonic relationships (Tonnetz). By combining these different feature extraction techniques, we can obtain a comprehensive set of features that capture a wide range of information about the

audio signal, which can improve the accuracy of machine learning models and other signal processing applications. Below is a representation of all the time domain graphs achieved by using the above pre-processing and feature extraction steps,

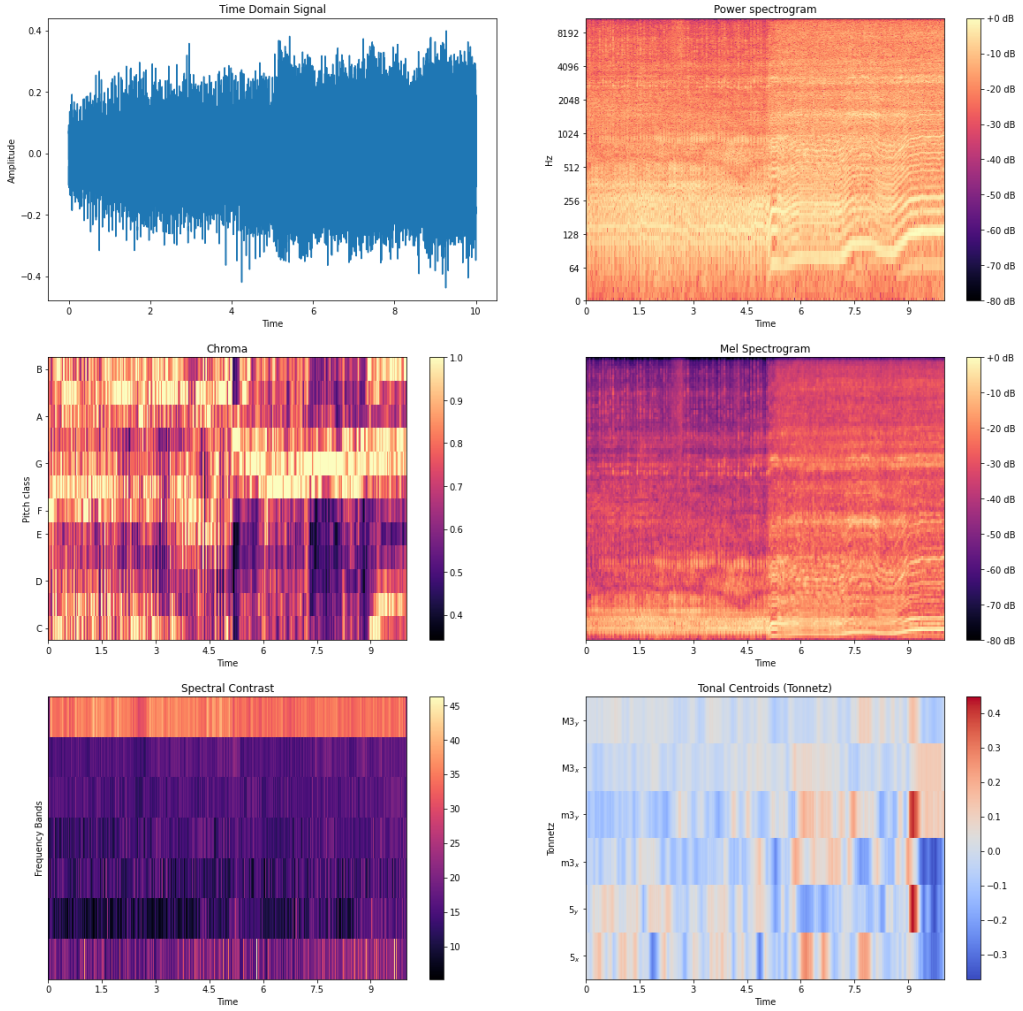


Fig.3.20 Representation of Pre-processing and Feature Extraction Techniques.

Actual Matrices	**Dimensions**
Sampling Rate	22050
Shape of Audio File	(220500,)
Shape of MFCC Matrix	(40, 431)
Shape of STFT	(1025, 431)
Shape of Chroma gram Matrix	(12, 431)
Shape of Mel Spectrogram Matrix	(128, 431)
Shape of Spectral Contrast Matrix	(7, 431)
Shape of Tonal Centroid Features Matrix	(6, 431)

Table.3.1 Feature extraction matrix dimensions.

The above table gives information regarding the dimensions of the feature matrices that are generated with the feature extraction techniques. Below is the description of the training features that are going to be utilized in the training of the model. These features are then put into a .csv file which becomes the audio feature set on which the models will be trained. Below is the representation of audio feature vectors extracted from the audio files.

Model Training Features	**Columns**
MFCC	40
Chromagram	12
Mel Spectrogram	128
Spectral Contrast	7
Tonal Contrast	6

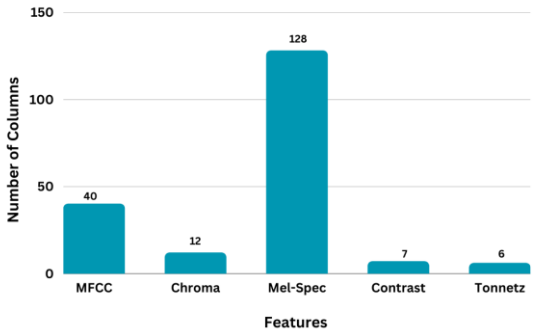


Table.3.2 length of Training Features.

The generated features are stored in a csv file where the number of columns is 193 and the number of rows is 7200 because the number of classes are six and in each class the number of samples are 1200. Below is a representation of the generated features csv file.

MFCC	……..	Chroma	……	Mel-Spec	……	Contrast	……	Tonnetz	**Class**
	……..		……..		……		……		
	……..		……..		……		……		
	……..		……..		……		……		
	……..		……..		……		……		

Table.3.3 Representation of CSV File.

The above table represents the CSV file format that is achieved after extracting all the features that are specified in the previous section, MFCC has 40 columns at the beginning of the file, chroma has 12 columns, Mel-Spec has 128 columns, spectral contrast has 7 columns and Tonal Contrast has 6 columns at the end. After all these columns a Class label column is also required to identify the features of a particular class, this helps us train the model according to those features of the class. Without labeled class names, the machine learning model will not be able to distinguish between different classes, and the resulting predictions will be random or meaningless. In some cases, unsupervised learning techniques can be used to identify patterns or clusters in the data without labeled class names.

3.3.3 Training Deep Learning Models

There are four algorithms used to train the model on the generated feature vectors in the previous section, the four algorithms are specified below,

- Convolutional Neural Network (CNN).
- Convolutional Recurrent Neural network (CRNN).
- Bidirectional Convolutional Recurrent Neural Network (Bi-CRNN).
- Residual Network (Res-Net 34).

These models are built using the Keras and TensorFlow module in python, the models are provided with the training set which is split from the data using the train test split method which splits the data into training and testing vectors. The model will be trained using the training set and the model will be tested using the testing set and accuracy will be calculated based on the results achieved by using the classification metrics.

Data Type	Percentage of Data
Training	85%
Testing	15%
Validation	15%

Table.3.4 Train test data split .

Training and testing are essential steps in developing a predictive model. When building a machine learning model, it is important to train the model on a portion of the available data and then test its performance on another portion of the data that the model has not seen before. The purpose of splitting the data into training and testing sets is to evaluate the performance of the model and to ensure that it generalizes well to new, unseen data. The training set is used to train the model, while the testing set is used to evaluate the model's performance on data that it has not seen before. Without splitting the data, the model would be trained on the same data that it is tested on, which can lead to overfitting. Overfitting occurs when the model fits too closely to the training data and does not generalize well to new data. In contrast, if the model is underfitting, it is not capturing the underlying patterns in the data and is not able to make accurate predictions. By splitting the data into training and testing sets, we can evaluate the model's performance on new, unseen data and ensure that it generalizes well. The above table.3.4 represents the train test data split percentages that is used for splitting the data into training, testing and validation sets for the model training and testing purpose, majority of the data is used for training which is 85% rest of the data is used for testing and validating the training of the model.

Below are the architectures of the deep learning models that are used for classifying the audio data,

3.3.3.1 Convolutional Neural Network (CNN)

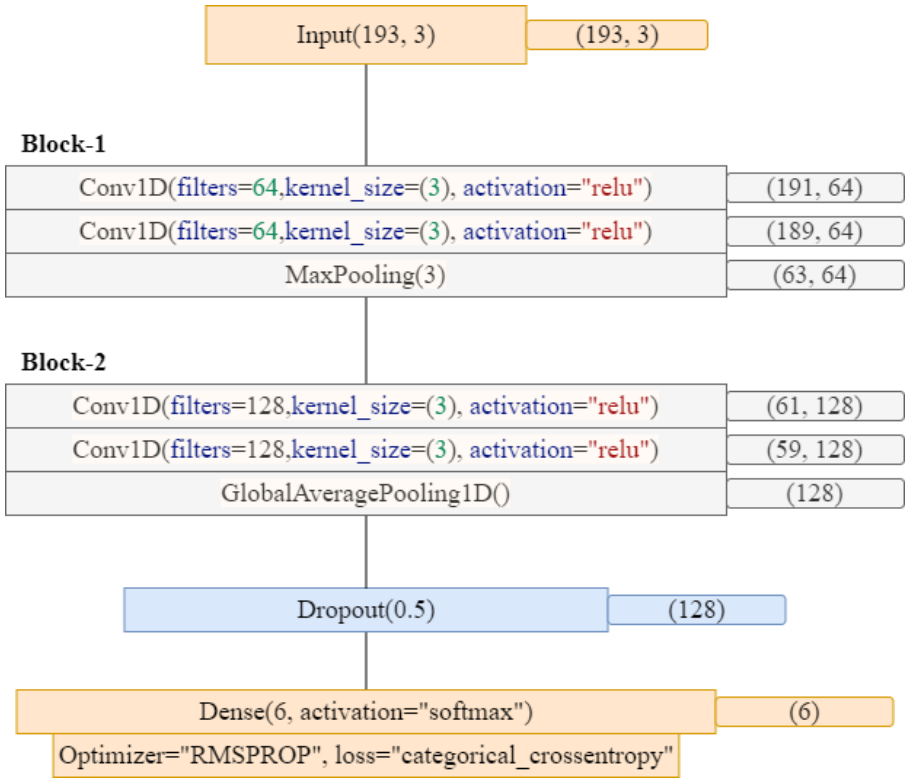


Fig.3.21 CNN Architecture.

In the above fig.3.21 the architecture of the CNN model is depicted, it contains a single input layer of size (193,3) which depicts the number of neurons in the input layer, then it consists of two blocks of convolutional layers, these convolutional layers are of single dimensions because the data we have is linear, total four CONV1D layers have been utilized, one Max Pooling layer and one Global Average Pooling layer has been used after all the feed forwarded data has been trained through the convolutional layers a dropout layer is used to drop the weights of layers then finally dense layer of six neurons used to classify the features into their respective classes. In the figure the output of each layer is also specified at the end. Here, Conv1D is used because it allows the network to learn local patterns in the input data by sliding a filter over the sequence and applying a mathematical operation to compute the output at each step. The filters are learned during the training process, which enables the network to extract relevant features from the input data that are useful for making accurate predictions. This layer is commonly used in deep learning applications for processing sequential data such as time series or natural language processing.

3.3.3.2 Convolutional Recurrent Neural Network (CRNN)

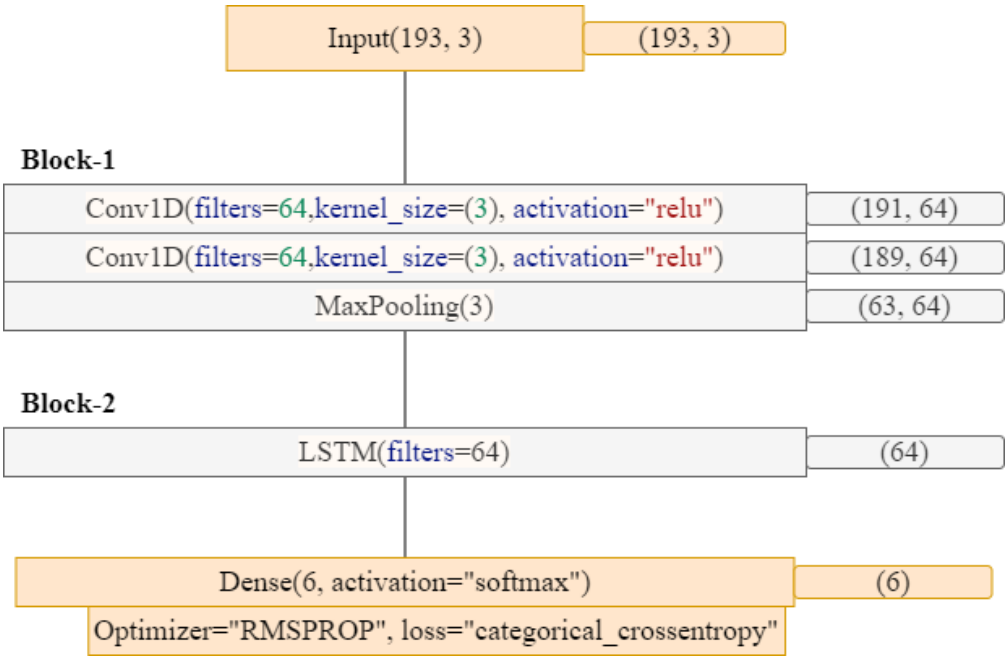


Fig.3.22 CRNN Architecture.

In the above fig.3.22 the architecture of the CRNN model is depicted, it contains a single input layer of size (193,3) which depicts the number of neurons in the input layer, then it consists of two blocks of convolutional layers and LSTM layer, these convolutional layers are of single dimensions because the data we have is linear, total two CONV1D layers have been utilized, one LSTM layer has been used after all the feed forwarded data has been trained through the convolutional layers then finally dense layer of six neurons used to classify the features into their respective classes. In the figure the output of each layer is also specified at the end. An LSTM layer is a type of recurrent neural network (RNN) layer that includes memory cells and gates designed to selectively store or discard information from previous time steps in a sequence. The LSTM layer typically takes as input a sequence of data and processes it in a recurrent manner. At each time step, the input is passed through multiple gates, which control the flow of information into and out of the memory cells. The input gate determines how much of the input should be stored in the memory cells, while the forget gate determines how much of the previous memory should be discarded. The output gate then determines how much of the current memory should be used to produce the output at the current time step. The LSTM layer is designed to address the problem of vanishing gradients in traditional RNNs, which can make it difficult for the network to learn long-term dependencies in the input data. By selectively storing or discarding information from

previous time steps, the LSTM layer can maintain long-term dependencies in the input sequence and make accurate predictions.

3.3.3.3 Bidirectional Convolutional Recurrent Neural Network (Bi-CRNN)

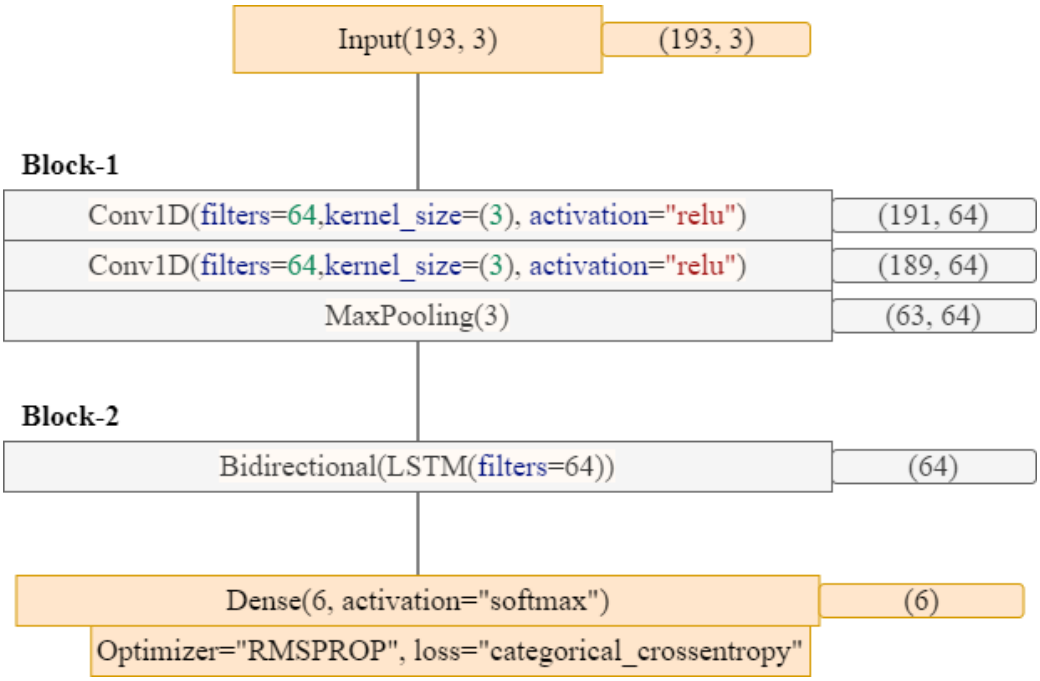


Fig.3.23 Bi-CRNN Architecture.

In the above fig.3.23 the architecture of the Bi-CRNN model is depicted, it contains a single input layer of size (193,3) which depicts the number of neurons in the input layer, then it consists of two blocks of convolutional layers and LSTM layer, these convolutional layers are of single dimensions because the data we have is linear, total two CONV1D layers have been utilized, one Bidirectional LSTM layer has been used after all the feed forwarded data has been trained through the convolutional layers then finally dense layer of six neurons used to classify the features into their respective classes. In the figure the output of each layer is also specified at the end. A Bidirectional LSTM (BiLSTM) layer is a type of LSTM layer that processes a sequence of data in two directions: from the beginning to the end and from the end to the beginning. In a standard LSTM layer, information flows only from past to future time steps, which means that the predictions are based only on the previous context. In contrast, a BiLSTM layer processes the sequence in both directions, allowing the network to capture both past and future context. The BiLSTM layer includes two separate LSTM layers that process the sequence in opposite directions. The output at each time step is then the concatenation of the output from the forward LSTM and the backward LSTM. The BiLSTM layer is particularly useful for tasks where the context in both directions is important, such as

speech recognition, where the context from both past and future time steps can be useful in identifying the current phoneme or word.

3.3.3.4 Residual Network (Res-Net 34)

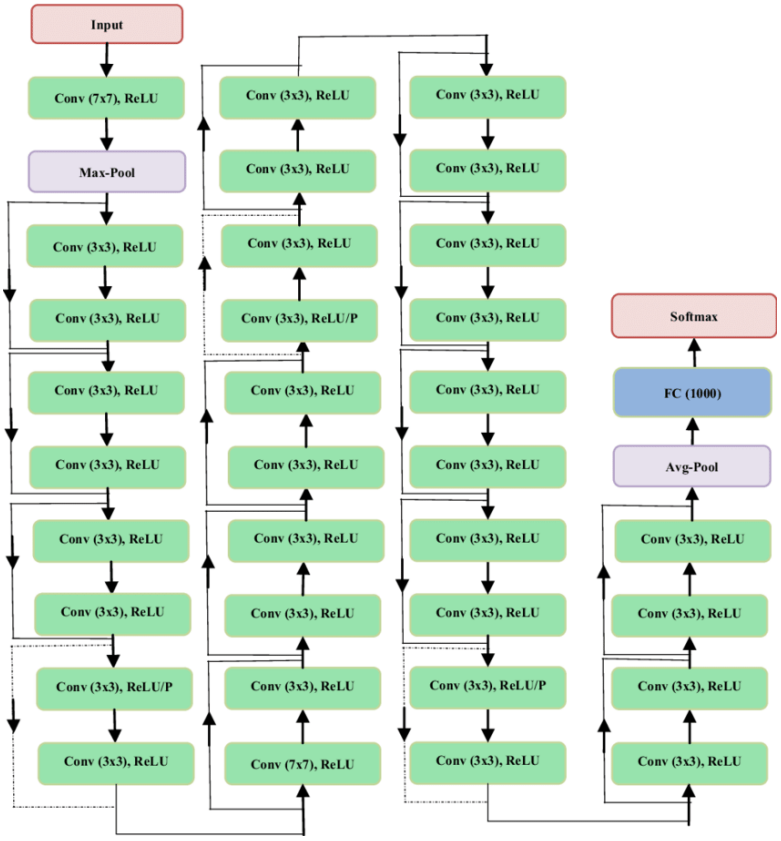


Fig.3.24 Res-Net 34 Architecture.

In the above fig.3.23 the architecture of the Res-Net 34 model is depicted, ResNet-34 is a deep neural network architecture that consists of 34 layers, including convolutional, pooling, fully connected, and skip connection layers. The ResNet architecture was introduced to address the problem of vanishing gradients in deep neural networks, which can make it difficult to train networks with many layers. The skip connection layers in ResNet allow the network to bypass one or more layers and pass information directly to a deeper layer, which helps to reduce the vanishing gradient problem and improve the overall performance of the network. In ResNet-34, the network includes 32 convolutional layers and 2 fully connected layers. The convolutional layers use 3x3 filters and are grouped into blocks, with each block consisting of multiple convolutional layers followed by a skip connection layer. The pooling layers are used to reduce the spatial dimensions of the feature maps, while the fully connected layers are used for classification or regression tasks. ResNet-34 has been widely used in many computer vision applications, including image classification, and has demonstrated state-of-the-art performance on a range of benchmark datasets.

CHAPTER 4

RESULT ANALYSIS

4.1 RESULT ANALYSIS

Below are the results of the executed models from the previous section. These results help us understand the prediction accuracy of the models and how well a model can classify the sound to their respective class.

4.1.1 CNN Results

	Precision	Recall	F1-Score	Support
0	0.98	0.94	0.96	176
1	0.98	0.95	0.97	200
2	0.99	1.00	1.00	173
3	0.98	1.00	0.99	192
4	0.94	0.86	0.90	167
5	0.83	0.95	0.89	172
Accuracy			**0.95**	1080
Macro avg	0.95	0.95	0.95	1080
Weighted avg	0.95	0.95	0.95	1080

Table.4.1 Classification report of CNN.

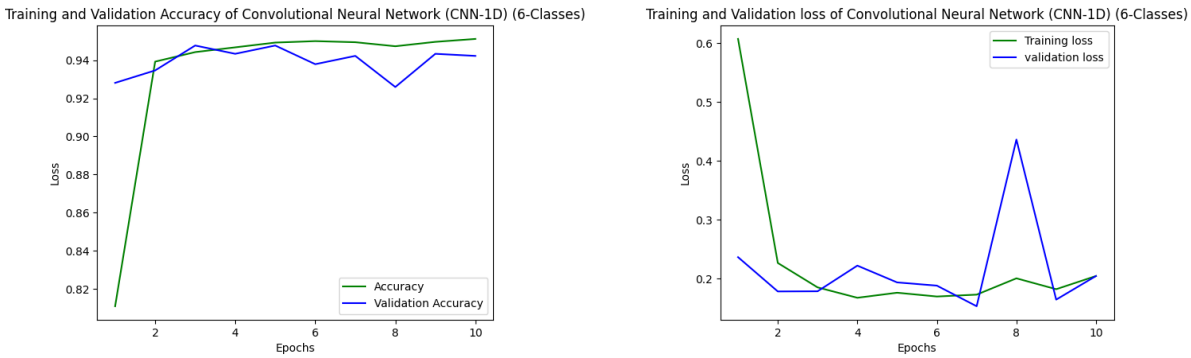


Fig.4.1 Training & Validation Accuracy, Loss of CNN.

The above table 4.1 depicts the classification report of the CNN model where all the classification metrics are considered for evaluating the model overall the model gives an accuracy of **0.95** on all the six classes data, the accuracy of all the other metrics is also specified in the table, in the fig.4.1 the training and validation accuracy, loss of the CNN model is shown in the graph, it is clearly depicted where the loss and accuracy are increasing and decreasing over the period of execution of **10 epochs** for training the model. The confusion matrices of the CNN model are given below in fig.4.3.

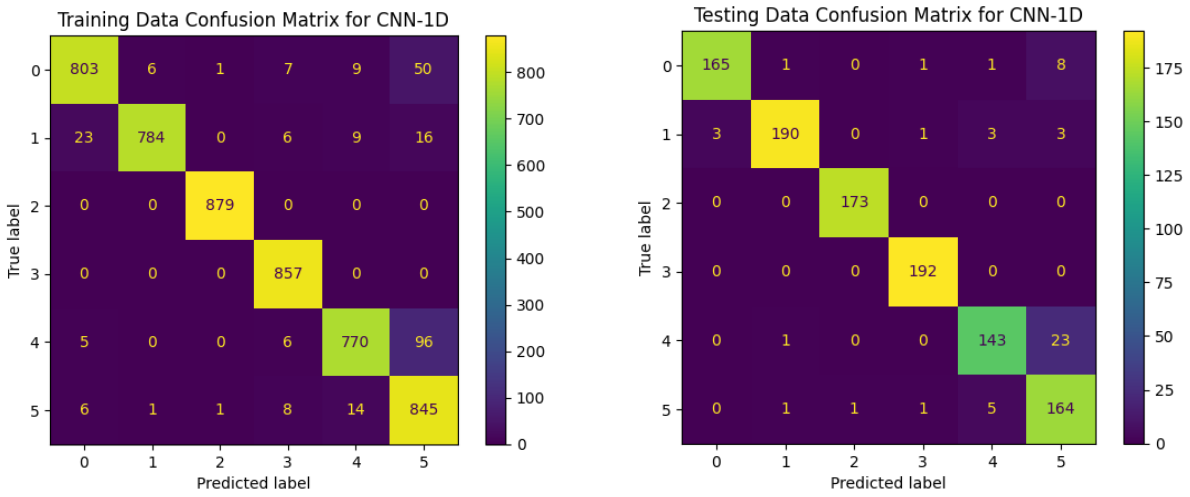


Fig.4.2 Training & Testing confusion matrix of CNN.

4.1.2 CRNN Results

	Precision	Recall	F1-Score	Support
0	0.88	0.93	0.90	176
1	0.99	0.96	0.97	200
2	0.99	0.92	0.95	173
3	0.99	1.00	0.99	192
4	0.87	0.96	0.91	167
5	0.92	0.85	0.89	172
Accuracy			**0.94**	1080
Macro avg	0.94	0.94	0.94	1080
Weighted avg	0.94	0.94	0.94	1080

Table.4.2 Classification report of CRNN.

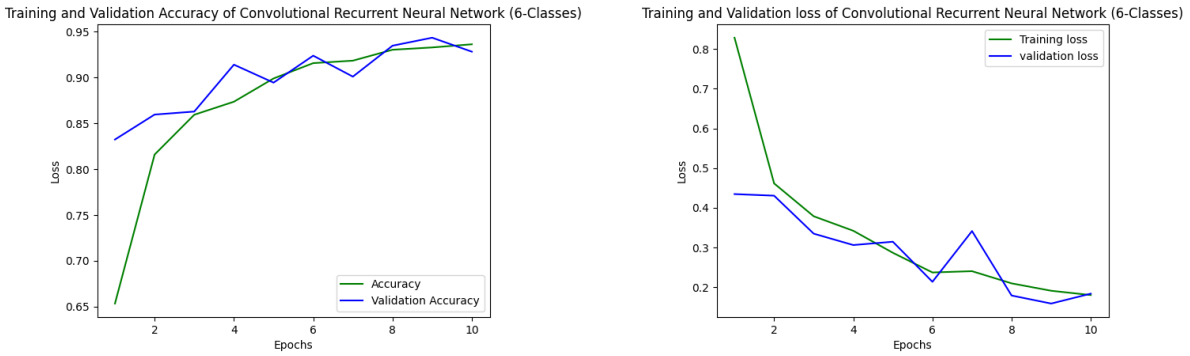


Fig.4.3 Training & Validation Accuracy, Loss of CRNN.

The above table 4.2 depicts the classification report of the CRNN model where all the classification metrics are considered for evaluating the model overall the model gives an accuracy of **0.94** on all the six classes data, the accuracy of all the other metrics is also

specified in the table, in the fig.4.2 the training and validation accuracy, loss of the CRNN model is shown in the graph, it is clearly depicted where the loss and accuracy are increasing and decreasing over the period of execution of **10 epochs** for training the model. The confusion matrices of the CRNN model are given below in fig.4.4.

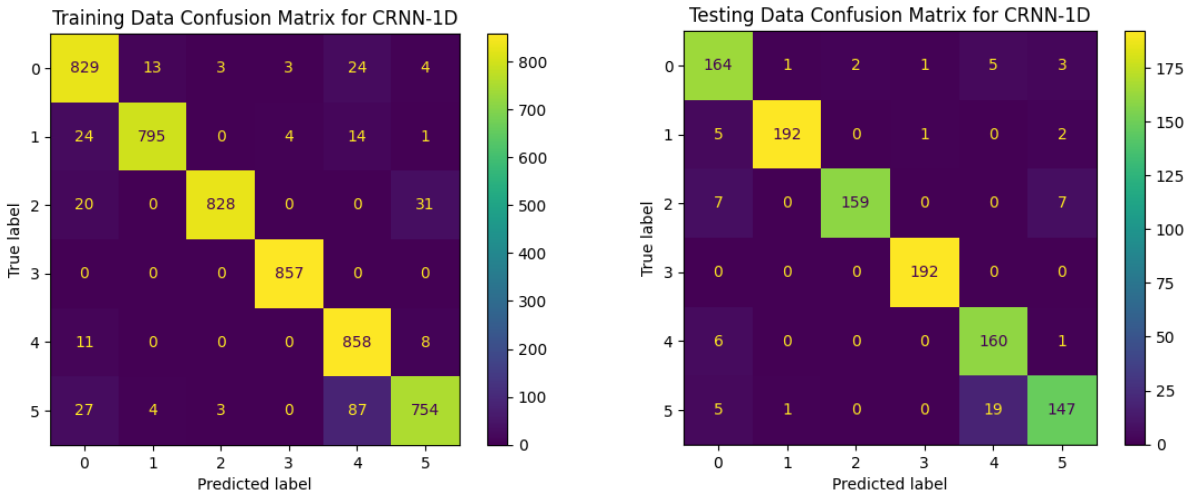


Fig.4.4 Training & Testing confusion matrix of CRNN.

4.1.3 BI-CRNN Results

	Precision	Recall	F1-Score	Support
0	0.94	0.95	0.95	176
1	0.96	0.97	0.96	200
2	1.00	1.00	1.00	173
3	0.94	1.00	0.97	192
4	0.93	0.91	0.92	167
5	0.94	0.87	0.91	172
Accuracy			**0.95**	1080
Macro avg	0.95	0.95	0.95	1080
Weighted avg	0.95	0.95	0.95	1080

Table.4.3 Classification report of BI-CRNN.

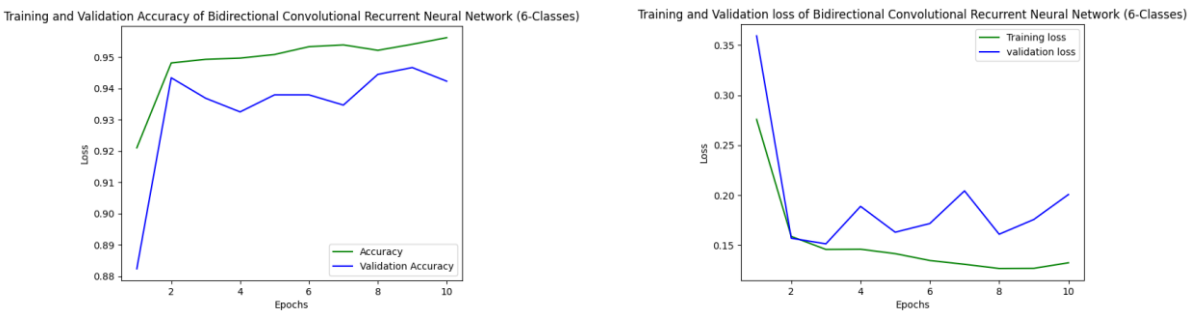


Fig.4.5 Training & Validation Accuracy, Loss of BI-CRNN.

The above table 4.3 depicts the classification report of the BI-CRNN model where all the classification metrics are considered for evaluating the model overall the model gives an accuracy of **0.95** on all the six classes data, the accuracy of all the other metrics is also specified in the table, in the fig.4.5 the training and validation accuracy, loss of the BI-CRNN model is shown in the graph, it is clearly depicted where the loss and accuracy are increasing and decreasing over the period of execution of **10 epochs** for training the model. The confusion matrices of the BI-CRNN model are given below in fig.4.6.

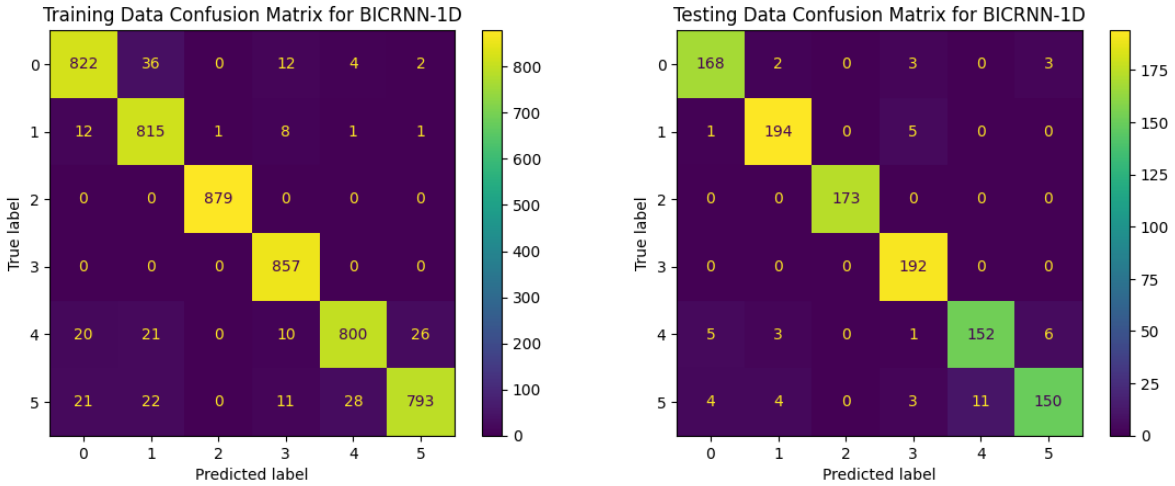


Fig.4.6 Training & Testing confusion matrix of BI-CRNN.

4.1.4 Res-Net 34 Results

	Precision	**Recall**	**F1-Score**	**Support**
0	0.88	0.91	0.90	183
1	0.99	0.88	0.93	192
2	1.00	1.00	1.00	165
3	0.99	1.00	0.99	177
4	0.79	0.98	0.88	187
5	0.99	0.82	0.90	176
Accuracy			**0.93**	1080
Macro avg	0.94	0.93	0.93	1080
Weighted avg	0.94	0.93	0.93	1080

Table.4.4 Classification report of Res-Net 34.

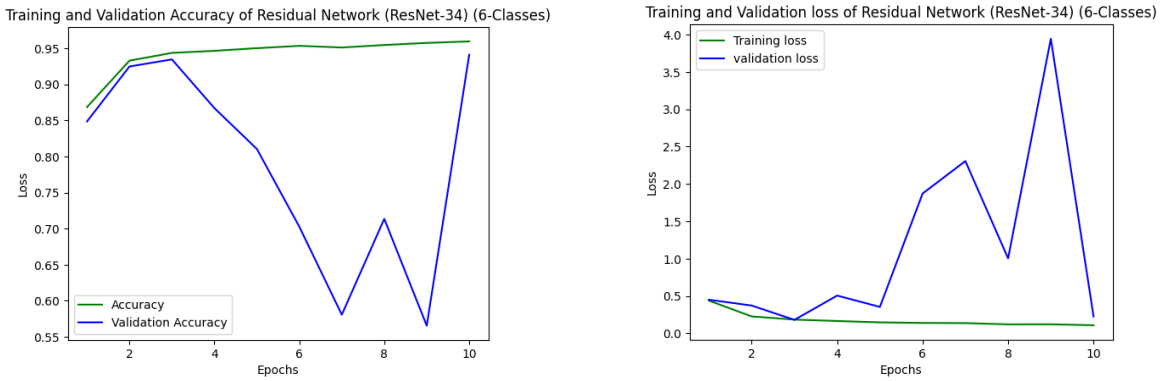


Fig.4.7 Training & Validation Accuracy, Loss of Res-Net 34.

The above table 4.4 depicts the classification report of the Res-Net 34 model where all the classification metrics are considered for evaluating the model overall the model gives an accuracy of **0.93** on all the six classes data, the accuracy of all the other metrics is also specified in the table, in the fig.4.5 the training and validation accuracy, loss of the Res-Net 34 model is shown in the graph, it is clearly depicted where the loss and accuracy are increasing and decreasing over the period of execution of **10 epochs** for training the model. The confusion matrices of the Res-Net 34 model are given below in fig.4.6.

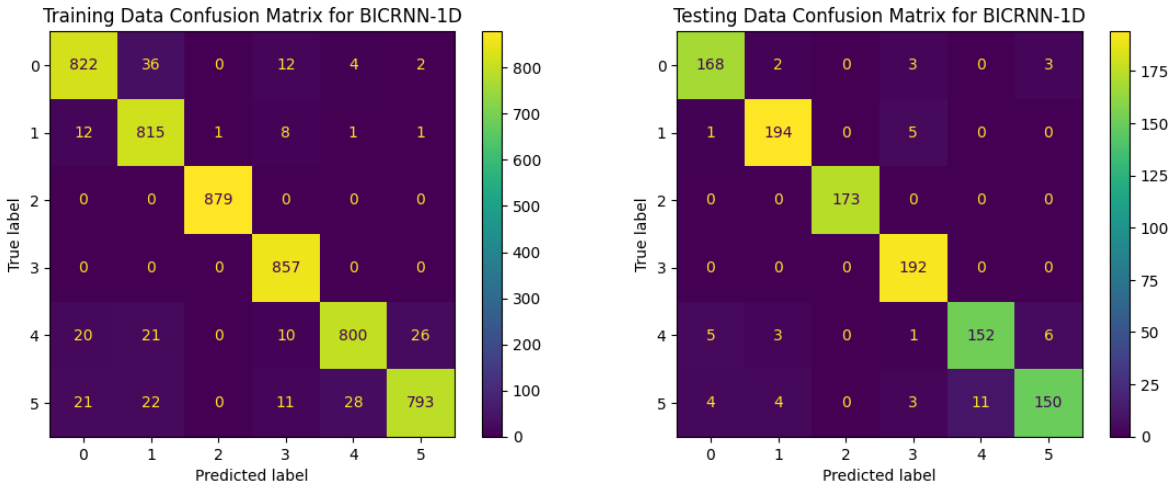


Fig.4.8 Training & Testing confusion matrix of Res-Net 34.

All the above four models provide significant results together where the accuracy ranges in between **0.93 to 0.95.** The four models are based on the convolutional layers overall and choose the most important features as their foundation to predict the classes of the audio data. When the models are compared all together then both CNN and BI-CRNN provide the highest accuracy of classification, each with accuracy of **0.95.** These results are helpful to analyze the results among the models and choose the best one from it to deploy it in the real time application of the system.

4.2 MODEL COMPARISION

The models are compared accordingly in the below table where the type of data is specified in the first column and the accuracies of each model i.e., CNN, CRNN, BI-CRNN and Res-Net 34 models are specified in the next columns for training as well as testing data. This comparison helps achieve the overall best model based on accuracy and loss.

Type	**CNN**		**CRNN**	
	Accuracy	**Loss**	**Accuracy**	**Loss**
Training	0.951	0.204	0.938	0.150
Testing	0.950	1.680	0.938	0.150
Validation	0.942	0.204	0.928	0.182

Table.4.5 Model Comparison Table-1.

Type	**BI-CRNN**		**RES-NET 34**	
	Accuracy	**Loss**	**Accuracy**	**Loss**
Training	**0.956**	**0.132**	0.959	0.108
Testing	**0.952**	**0.159**	0.930	0.214
Validation	**0.942**	**0.200**	0.941	0.225

Table.4.6 Model Comparison Table-2.

In the above two tables, all the four models are compared with their respective accuracy and losses in three types of sets of data which is split i.e., training, testing and validation sets, in all the four models the Bidirectional Convolutional Neural Network (BI-CRNN) provides the best result i.e., with highest accuracy and lowest loss in when compared to the other models. The BI-CRNN model has a marginally greater accuracy and a marginally lower loss compared to the other models, which suggests that the BI-CRNN model might perform better on this dataset. It's crucial to remember that high accuracy and low loss numbers do not always imply good performance on unknown data, and that these metrics should be understood in the context of the problem and dataset the model was trained on. To guarantee the model's vigorous and hypothesis, more analysis and testing are required on that specific dataset.

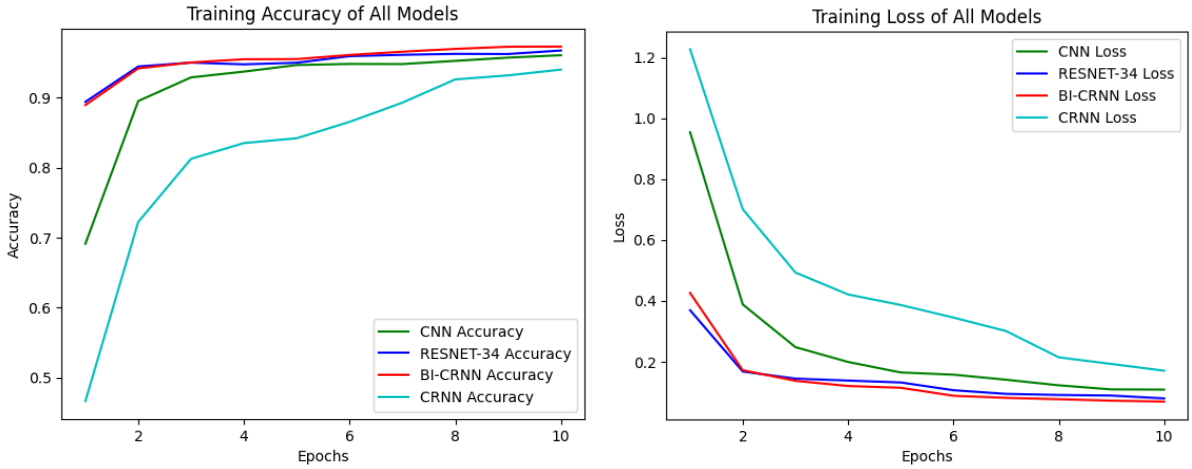


Fig.4.9 Training Accuracy and loss of all models.

The above figures represent the training accuracy and loss of all the four models compared together, the accuracy of the CRNN model is peaking the least in the period of 10 iterations. In the loss graph also the loss line of CRNN is the highest, whereas the BI-CRNN model provides the best accuracy and less loss when compared to all the other models over the period of execution.

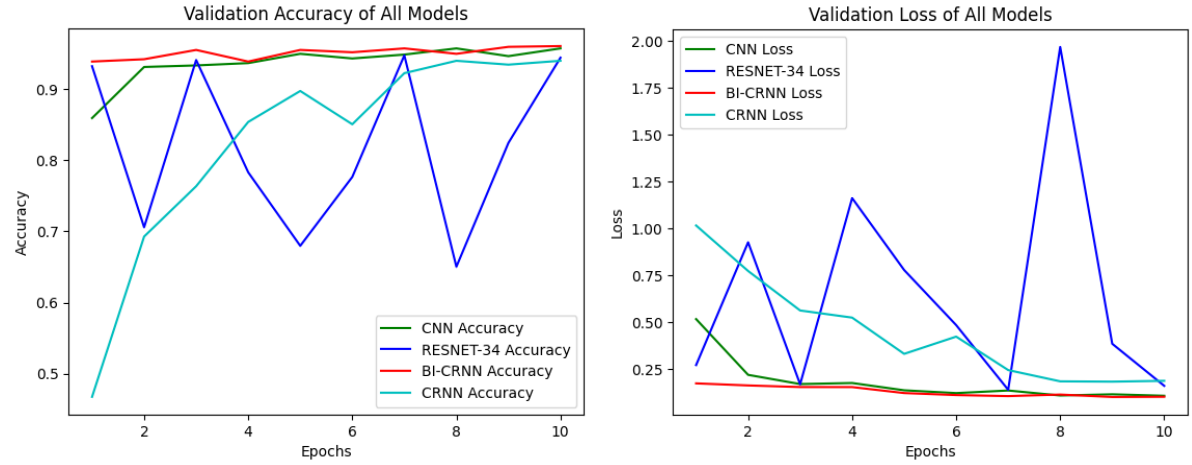


Fig.4.10 Validation Accuracy and loss of all models.

The above figures represent the training accuracy and loss of all the four models compared together, the accuracy of the CRNN model is peaking the least in the period of 10 iterations. In the loss graph also the loss line of CRNN is the highest, whereas the BI-CRNN model provides the best accuracy and less loss when compared to all the other models over the period of execution.

By observing all the above graphs, we can justify that the Bidirectional Convolutional Recurrent Neural Network (BI-CRNN) is the most stable model while training the model and executing the model in real time. Other models do provide good results overtime but by observing the graphs we see how unstable they are over the execution time.

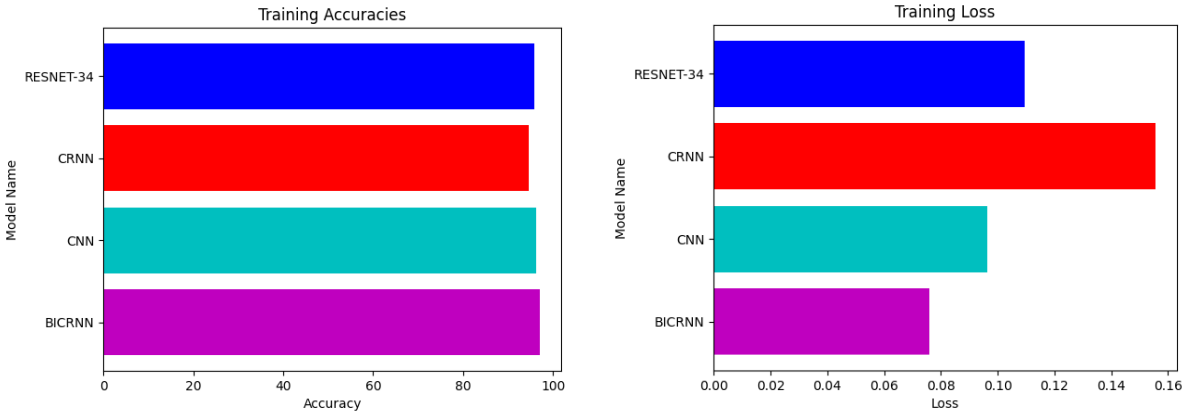


Fig.4.11 Comparing Training Accuracy and loss of all models.

The above graphs show the training accuracy and loss of all the four models compared together, the accuracy of the CRNN model is peaking the least in the period of overall comparison. In the loss graph also the loss line of CRNN is the highest, whereas the BI-CRNN model provides the best accuracy and less loss when compared to all the other models over the period of execution.

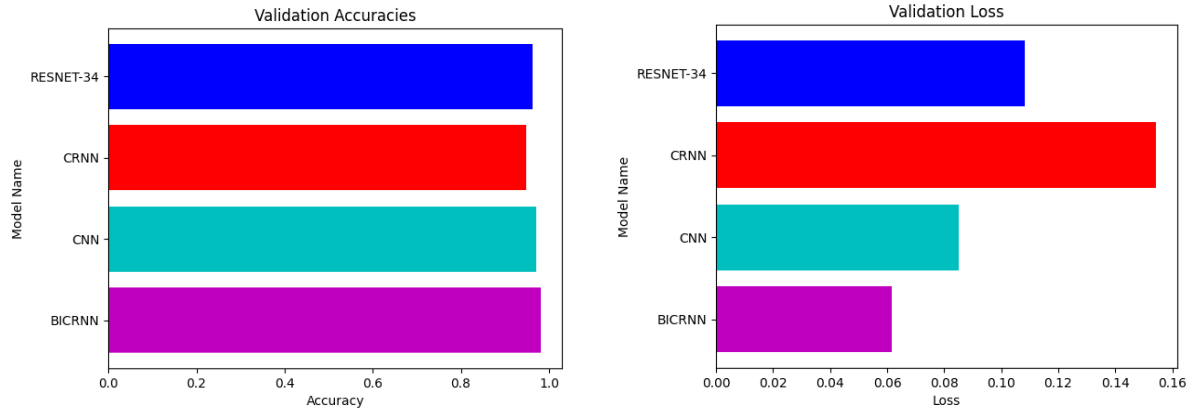


Fig.4.12 Comparing Validation Accuracy and loss of all models.

The above graphs show the validation accuracy and loss of all the four models compared together, the accuracy of the CRNN model is peaking the least in the period of overall comparison. In the loss graph also the loss line of CRNN is the highest, whereas the BI-CRNN model provides the best accuracy and less loss when compared to all the other models over the period of execution.

By observing all the above graphs, we can justify that the Bidirectional Convolutional Recurrent Neural Network (BI-CRNN) is the most accurate model while validating the model and executing the model in real time. Other models do provide good results overtime but by observing the graphs we see how minute the difference between the model accuracies are this minute difference can make the impact of determing the most generalzied model.

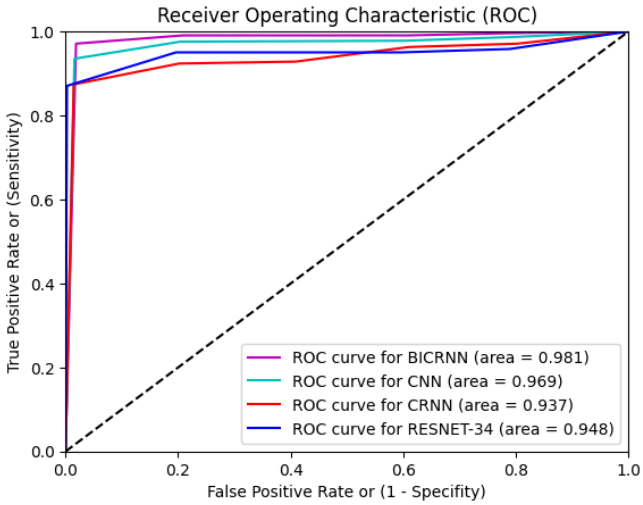


Fig.4.13 ROC Curve.

The above figure represents the ROC curve, the ROC (Receiver Operating Characteristic) curve is a graphical representation of the performance of a binary classification model. It is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

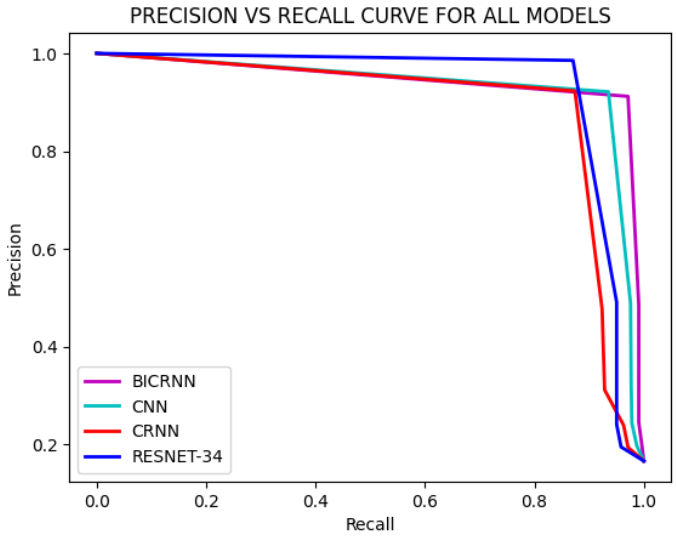


Fig.4.14 Precision vs Recall Curve.

The above figure represents the Precision and Recall curve for all the models, they are two important metrics for evaluating the performance of a classification model, particularly in imbalanced datasets. The precision-recall curve is a graphical representation of the trade-off between precision and recall for different classification thresholds. Precision is the ratio of true positive (TP) cases to the total number of positive (TP+FP) cases predicted by the model, while recall is the ratio of true positive (TP) cases to the total number of positive (TP+FN) cases in the dataset. All these comparisons between the models with different metrics help us determine the best model out of all the models trained.

CHAPTER 5
CONCLUSION

5.1 CONCLUSION

Here, a Concept for automatic detection of logging activity in forests using audio recordings was presented. The Concept used monitoring stations installed in the forest for audio recordings using microphones, and then acquired audio samples which were then processed and automatically classified into logging or not logging sounds. Multiple classification algorithms were tested, using well known and widely used audio descriptors during the feature extraction step, with the evaluation focusing on the cutting sound identification whether axe or chainsaw and tree falling sound identification during logging in the forests.

We deem that the presented concept greatly contributes as an affordable solution in the development of systems for monitoring forests and for preserving the sustainability of the environment, to reduce illegal deforestation and protect biodiversity. In the presented approach, CNN based models were proposed to recognize and classify six categories of audios. A Res-Net model was also trained to accomplish the classification task. Furthermore, the proposed architecture based on CNN is economically resilient at categorizing the audios. We acquired accuracy rates of 95.2 in the BI-CRNN model. The proposed work performs well for Detection logging of forest trees using sound event detection.

5.2 LIMITATIONS AND FUTURE SCOPE

To begin, the model was only investigated using six classes, hence our findings are limited to six audio classes. There are various other classes that also play a vital role in detecting the logging of trees effectively and classifying them, including the sounds which do not occur in the forest. In the future, the proposed approach can be applied to various other sound event detection domains from which the data obtained would be more accurate and less noisy.

Additionally, the method can effectively classify the six different kinds of audios. The proposed work performs well for detecting logging of trees. However, a little performance degradation was observed for audios with excess noise. In the future, we plan to cover this limitation. Moreover, the models developed can also be improved by applying techniques of localization where the precise area from where the sound is being generated is also found with accurate margin.

BIBLIOGRAPHY

[1] Chachada, Sachin, and C.-C. Jay Kuo. “Environmental Sound Recognition: A Survey.” APSIPA Transactions on Signal and Information Processing 3 (2014): e14. doi:10.1017/ATSIP.2014.12.

[2] Segarceanu, Svetlana & Olteanu, Elena & Suciu, George. (2020). Forest Monitoring Using Forest Sound Identification. 346-349. 10.1109/TSP49548.2020.9163433.

[3] Wang, Jia-Ching & Lee, Hsiao-Ping & Wang, Jhing-Fa & Lin, Cai-Bei. (2008). Robust Environmental Sound Recognition for Home Automation. Automation Science and Engineering, IEEE Transactions on. 5. 25 - 31. 10.1109/TASE.2007.911680.

[4] Siddharth Sigtia, Adam M. Stark, Sacha Krstulovic, Mark D. Plumbley, Siddharth Sigtia, Adam M. Stark, Sacha Krstulovic, and Mark D. Plumbley. 2016. Automatic Environmental Sound Recognition: Performance Versus Computational Cost. IEEE/ACM Trans. Audio, Speech, and Lang. Proc. 24, 11 (November 2016), 2096–2107. https://doi.org/10.1109/TASLP.2016.2592698

[5] S. Chu, S. Narayanan and C. -C. J. Kuo, "Environmental sound recognition using MP-based features," 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, 2008, pp. 1-4, doi: 10.1109/ICASSP.2008.4517531.

[6] L. Shi, I. Ahmad, Y. He and K. Chang, "Hidden Markov model-based drone sound recognition using MFCC technique in practical noisy environments," in Journal of Communications and Networks, vol. 20, no. 5, pp. 509-518, Oct. 2018, doi: 10.1109/JCN.2018.000075

[7] G. Roma, W. Nogueira and P. Herrera, "Recurrence quantification analysis features for environmental sound recognition," 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, USA, 2013, pp. 1-4, doi: 10.1109/WASPAA.2013.6701890.

[8] Á. Incze, H. -B. Jancsó, Z. Szilágyi, A. Farkas and C. Sulyok, "Bird Sound Recognition Using a Convolutional Neural Network," 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2018, pp. 000295-000300, doi: 10.1109/SISY.2018.8524677.

[9] Q. Yu, Y. Yao, L. Wang, H. Tang, J. Dang and K. C. Tan, "Robust Environmental Sound Recognition with Sparse Key-Point Encoding and Efficient Multispike Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 2, pp. 625-638, Feb. 2021, doi: 10.1109/TNNLS.2020.2978764.

[10] B. -Y. Ooi, J. J. -W. Lim, W. -K. Lee and S. Shirmohammadi, "Non-Intrusive Operation Status Tracking for Legacy Machines via Sound Recognition," 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Dubrovnik, Croatia, 2020, pp. 1-6, doi: 10.1109/I2MTC43012.2020.9129526.

[11] O. K. Toffa and M. Mignotte, "Environmental Sound Classification Using Local Binary Pattern and Audio Features Collaboration," in IEEE Transactions on Multimedia, vol. 23, pp. 3978-3985, 2021, doi: 10.1109/TMM.2020.3035275.

[12] L. Vujošević and S. Đukanović, "Deep learning-based classification of environmental sounds," 2021 25th International Conference on Information Technology (IT), Zabljak, Montenegro, 2021, pp. 1-4, doi: 10.1109/IT51528.2021.9390124.

[13] Mesaros, Annamaria, Toni Heittola, and Tuomas Virtanen. 2016. "Metrics for Polyphonic Sound Event Detection" Applied Sciences 6, no. 6: 162. https://doi.org/10.3390/app6060162

[14] A. Mesaros, T. Heittola and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," 2016 24th European Signal Processing Conference (EUSIPCO), Budapest, Hungary, 2016, pp. 1128-1132, doi: 10.1109/EUSIPCO.2016.7760424.

[15] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen and T. Virtanen, "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 6, pp. 1291-1303, June 2017, doi: 10.1109/TASLP.2017.2690575

[16] Min Xu, Ling-Yu Duan, Chang-Sheng Xu and Qi Tian, "A fusion scheme of visual and auditory modalities for event detection in sports video," 2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698), Baltimore, MD, USA, 2003, pp. I-333, doi: 10.1109/ICME.2003.1220922

[17] Y. Zigel, D. Litvak and I. Gannot*, "A Method for Automatic Fall Detection of Elderly People Using Floor Vibrations and Sound—Proof of Concept on Human Mimicking Doll

Falls," in IEEE Transactions on Biomedical Engineering, vol. 56, no. 12, pp. 2858-2867, Dec. 2009, doi: 10.1109/TBME.2009.2030171.

[18] Dong Zhang, D. Gatica-Perez, S. Bengio and I. McCowan, "Semi-supervised adapted HMMs for unusual event detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 611-618 vol. 1, doi: 10.1109/CVPR.2005.316

[19] C. Clavel, T. Ehrette and G. Richard, "Events Detection for an Audio-Based Surveillance System," 2005 IEEE International Conference on Multimedia and Expo, Amsterdam, Netherlands, 2005, pp. 1306-1309, doi: 10.1109/ICME.2005.1521669.

[20] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," 2007 IEEE Conference on Advanced Video and Signal Based Surveillance, London, UK, 2007, pp. 21-26, doi: 10.1109/AVSS.2007.4425280

[21] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events," in IEEE Transactions on Multimedia, vol. 17, no. 10, pp. 1733-1746, Oct. 2015, doi: 10.1109/TMM.2015.2428998.

[22] D. Ruinskiy and Y. Lavner, "An Effective Algorithm for Automatic Detection and Exact Demarcation of Breath Sounds in Speech and Song Signals," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 15, no. 3, pp. 838-850, March 2007, doi: 10.1109/TASL.2006.889750.