# Enhancing Audio Classification Through MFCC Feature Extraction and Data Augmentation with CNN and RNN Models

Karim Mohammed Rezaul[1], Md. Jewel[2], Md Shabiul Islam[3], Kazy Noor e Alam Siddiquee[4], Nick Barua[5], Muhammad Azizur Rahman[6], Mohammad Shan-A-Khuda[7], Rejwan Bin Sulaiman[8], Md Sadeque Imam Shaikh[9], Md Abrar Hamim[10], F.M Tanmoy[11], Afraz Ul Haque[12], Musarrat Saberin Nipun[13], Navid Dorudian[14], Amer Kareem[15], Ahmmed Khondokar Farid[16], Asma Mubarak[17], Tajnuva Jannat[18], Umme Fatema Tuj Asha[19]

Wrexham University, UK[1]
Centre for Applied Research in Software & IT (CARSIT), UK[2, 10, 11, 12, 18, 19]
Multimedia University, Malaysia[3]
State University of Bangladesh[4]
Kobe Institute of Computing, Japan[5]
Cardiff Metropolitan University, UK[6]
Leeds Beckett University, Uk[7]
Northumbria University London[8]
Coventry University London[9]
Brunel University London[13]
Brunel University London (BPC) [14, 17]
University of Bedfordshire, Uk[15]
Canterbury Christ Church University, Uk[16]

***Abstract*—Sound classification is a multifaceted task that necessitates the gathering and processing of vast quantities of data, as well as the construction of machine learning models that can accurately distinguish between various sounds. In our project, we implemented a novel methodology for classifying both musical instruments and environmental sounds, utilizing convolutional and recurrent neural networks. We used the Mel Frequency Cepstral Coefficient (MFCC) method to extract features from audio, which emulates the human auditory system and produces highly distinct features. Knowing how important data processing is, we implemented distinctive approaches, including a range of data augmentation and cleaning techniques, to achieve an optimized solution. The outcomes were noteworthy, as both the convolutional and recurrent neural network models achieved a commendable level of accuracy. As machine learning and deep learning continue to revolutionize image classification, it is high time to explore the development of adaptable models for audio classification. Despite the challenges associated with a small dataset, we successfully crafted our models using convolutional and recurrent neural networks. Overall, our strategy for sound classification bears significant implications for diverse domains, encompassing speech recognition, music production, and healthcare. We hold the belief that with further research and progress, our work can pave the way for breakthroughs in audio data classification and analysis.**

***Keywords*—Deep learning (artificial intelligence); data augmentation; audio segmentation; signal processing; frame blocking; fast fourier transform; discrete cosine transform; feature extraction; MFCC; CNN; RNN**

## I. Introduction

Deep learning techniques have enabled the classification of audio, which has numerous practical applications. This technology can be used to recommend music, categorize various musical instruments, recognize music genres, organize music collections, develop streaming services, differentiate between male and female speech, distinguish between different languages or accents, build speech recognition systems, and analyze audio recordings from surveillance equipment to detect sounds indicating a threat or emergency. Consequently, deep learning-based audio classification has become an essential tool that can be employed in diverse contexts to analyze and classify audio data.

Lately, there has been a notable surge in the utilization of Digital Signal Processing (DSP) for musical instrument processing and speech analysis. In addition, there is an increasing demand for online access to music data on the internet, and this has led to a rise in computational tools for development such as summarization, analysis, classification, and indexing. Music Information Retrieval (MIR) provides solutions for music-related tasks, including the subtask of sound classification of musical instruments, which involves identifying different musical instruments [1]. MIR is also used for a variety of applications, including beat tracking, beat recognition and separation, automatic music transcription, and polyphonic audio processing [2].

Instrumental music frequently contains insightful information regarding current events. Although automatic sound processing is thought to be the state of the art, robots are

still far behind humans in their ability to perceive and distinguish between wide varieties of sound events. More research is needed today to create a dependable system that can accurately identify a wide spectrum of audio, including different musical instruments [3].

Generally speaking, there are three sub-categories of audio classification tasks: music classification, acoustic scene classification, and speech recognition (acoustic model). Each of these activities includes various signal qualities, which causes changes in the audio data input aspects. Recent significant deep learning breakthroughs have made it possible to build a single audio or musical instrument model that is adaptable enough to handle diverse cross-domain tasks. The potential of a CRNN (Convolutional Recurrent Neural Network) model was harnessed by Adavanne et al. [4] for the detection of sound events, the classification of auditory birds [5], and the recognition of musical emotion [6]. The selection of parameters for representing time-frequency as input has a substantial impact on the efficacy of audio classification models for various tasks, according to recent breakthroughs in the field. Unfortunately, a large number of existing models use non-optimal filter bank size and type, time-frequency magnitude compression, and resolution. Particularly concerning the choice of 2D or 1D convolutional layers and the shape of the filter, these decisions have a significant impact on the model's architecture [7]. Waveform-based models, which directly handle unaltered input signals, offer an inventive way to get around these problems. Regarding the aforementioned problems, this strategy has promise. Notably, Schrauwen and Dieleman [8] recently showed the effectiveness of CNN models using raw waveforms as input for automatically tagging music, opening up new potential for enhancing audio categorization performance. This approach helps overcome the limitations of traditional audio classification methods by allowing the model to learn directly from the raw audio signals. Consequently, waveform-based models have the potential to significantly improve the accuracy and efficiency of audio classification in a wide range of applications.

Most of the previous research in music information retrieval (MIR) has focused on monophonic music [9], while this approach predominantly utilizes monophonic data for instrument classification. For speech identification, Sainath et al. [10] employed a convolutional long short-term memory deep neural network (CLDNN), whereas Dai et al. [11] used a deep convolutional neural network (DCNN) with residual connections to identify environmental sounds. The majority of the investigations employed frame-level filters with carefully designed first convolutional layers made up of extensive samples.

The method for extracting musical instrument features and categorizing instrumental audio in our study is based on these attributes. Fig. 1 shows the block diagram describing the procedure in detail for this operation.

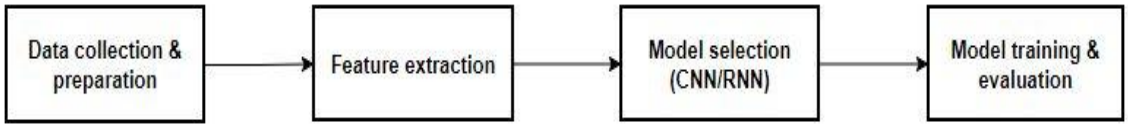


Fig. 1. Basics of audio tagging and feature extraction.

We arranged this paper as follows, in Section I, the introduction is given, in Section II, the related work is explained, in Section III, the feature extraction techniques are described, in Section IV, the dataset is explored, in Section V, the pre-processing steps are discussed, in Section VI, the model is built and explained, in Section VII, a comprehensive analyses of the results are conducted, and in Section VIII, conclusion is drawn.

## II. Related Work

The classification of musical instruments is a topic that is actively being researched, and many approaches have been suggested by scholars and it is clear from the literature study that more research is still needed in this area to get the best results with greater precision, especially when working with tiny datasets.

Meinard Müller, Daniel P. W. Ellis, Anssi Klapuri, and Gal Richard examined numerous signal processing techniques to categorize musical instruments in a notable work [12]. With a focus on musical signal processing, this article provides a thorough overview of numerous research fields. Among the methods investigated, the use of MFCCs (Mel Frequency Cepstral Coefficients) in the classification of musical instruments attracts a lot of interest and debate.

In a noteworthy study, Jadhav, P. S. [13] proposed a unique method for identifying musical instruments by fusing MFCCs with Timbral Associated Descriptors of Audio. They used a binary tree, SVM (Support Vector Machine), and k-nearest neighbor as part of their feature extraction technique. The research demonstrated an insightful investigation into improving musical instrument recognition by further examining and evaluating the identification accuracy attained through various combinations of classification algorithms and feature extraction methods.

D. G. Bhalke, C. B. Rama Rao, and D. S. Bormane [1] pioneered the use of FFT (Fractional Fourier Transform) in conjunction with MFCCs for categorizing musical instruments in a different work that has been discussed in the introduction section. They also used temporal traits like assault time, zero-crossing rates, decay time, and energy to their advantage to support their classification strategy. The method for calculating the zero-crossing rate, Eq. (1), was introduced in the study, offering important insights into the development of musical instrument categorization algorithms.

$$ZCR = \frac{1}{T}\sum_{0}^{T-1} \mid \mathrm{sgn}[x(y)] - \mathrm{sgn}[x(y-1)] \mid \qquad (1)$$

Here, T denotes the sample in each frame, while x(y) and x(y-1) denote the signals of the yth and (y-1) samples, respectively.

The Eq. (2) was utilized to compute the energy of the sound sample.

$$Energy = \sum_{n=0}^{T-1} (|\, m(n) \,|)^2 \quad (2)$$

The signal of the nth sample is represented by m(n), while T signifies the number of samples present in one frame.

Essid, S., Richard, G., & David, B. suggested that the sound samples feature be used for MFCCs [14]. They provided more information on how using the derivative of MFCCs over time can be used to exploit the Delta MFCC capabilities. The SVM technique was used to classify data. Spectral characteristics such as spectral centroid and spectral breadth were used. One mapping vs one SVM was used to train the data. M. Erdal Ozbek, Nalan Ozkurt, and F. Acar Savaci [15] classified musical instruments using wavelet decomposition up to the first three stages and wavelet ridges (Fig. 2). By using this method, three precise coefficients and one estimated coefficient are extracted, allowing for accurate classification.

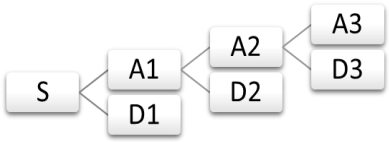


Fig. 2. Three Stages of wavelet decomposition.

Herein, the signal frame denoted as S, is calculated as the sum of the approximate coefficient (A) and the detailed coefficients (D) at levels 1 to 3, where D1, D2, and D3 correspond to the detailed coefficients at each respective level.

Farbod and Karthikeyan suggested using wavelet-dependent time scale information to categorize musical instruments [16]. In order to obtain the required qualities, they continuously took the wavelet transform signal frame and extracted features relating to bandwidth and temporal fluctuation.

A support vector machine (SVM) employing Mel-Frequency Cepstral Coefficients (MFCC) as feature vectors was suggested in a prior study for the classification of musical genres. While melody is crucial for understanding music data, it is not a suitable feature for classification. Music genres are strongly correlated with the timbre of music, which corresponds to the frequency characteristics of sound signals. Therefore, previous studies commonly use MFCC as feature vectors for music genre classification [17]. Another study combines audio and lyrics features to detect music emotions, using a synchronized dataset of chorus audio and lyrics. The audio features extracted include dynamics, rhythm, timbre, pitch, and tonality, while lyric features include psycholinguistic, stylistic, and statistical features. Weighting the audio and lyric features using a Naive Bayes probability value shows that the audio feature is dominant with an 80% weighting ratio [18].

The application of these diverse feature extraction techniques is evident in numerous research papers. An article discusses how cardiovascular diseases are a major cause of deaths worldwide and identifies the importance of detecting heart disease at an early stage. The article presents an approach for classifying heart audio samples using deep learning techniques and compares the results of various machine learning algorithms. The approach involves implementing existing segmentation techniques and feature engineering in the audio domain. The precision values indicate that the Hybrid CNN model performed best with a precision of 1 for artifact, 0.906 for normal, and 0.859 for murmur categories [19].

Another study shows that Vehicle classification is a crucial task in managing traffic and road infrastructure, with new challenges continuously emerging. Through classifier fusion techniques, the complementary nature of information has previously been used to enhance classifier performance. This hasn't been looked at in the context of a multi-modal categorization system that uses only neural networks. To increase performance, this study suggests a complementarity-based multi-modal vehicle categorization system. The system uses sets of Mel Frequency Cepstral Coefficients (MFCC) as the feature vectors for the audio modality to perform vehicle classification with two distinct modalities using Convolutional Neural Network (CNN) classifiers. At the decision level, the predictions from the base classifiers are combined to provide a final prediction, increasing accuracy. The study finds that in a fully neural network-based multi-modal system, decision-level fusion is an efficient method for enhancing vehicle classification accuracy [20].

## III. MFCC Feature Extraction

Generally, Automated Speech Recognition (ASR) systems require feature extraction from speech signals that are non-stationary in nature [21]. Feature extraction becomes difficult due to speech variability constraints such as differences between speakers, intonation, and changes in speech production. A good feature extraction technique should identify specific linguistic properties and discard irrelevant information such as background noise and emotion. Commonly used feature extraction techniques include Mel Frequency Cepstral Coefficients (MFCC), Linear Predictive Coefficients (LPC), Perceptual Linear Predictive (PLP) Coefficients, Discrete Wavelet Transform (DWT), and Principal Component Analysis (PCA).

Our study aims to demonstrate the process of sound feature extraction using the MFC technique, which is currently popular. The vocal tract's shape, including the shape of the tongue and teeth, filters spoken sounds and defines the sound made. Accurately determining the shape provides an accurate phoneme representation, which is reflected in the short-time power spectrum's envelope. MFCCs precisely represent this envelope. To classify instrumental music, identifying various audio signal components and eliminating background noise or dead space is the initial step.

The use of Mel Frequency Cepstral Coefficients (MFCCs) for audio feature extraction in various recognition applications has become commonplace in the present day. Because of their

effective classification accuracy in a clean environment, MFCCs have been around since Davis and Mermelstein first introduced them in the 1980s [22]. Fig. 3 shows the steps involved in the MFCC feature extraction approach, which has recently gained in popularity [23].

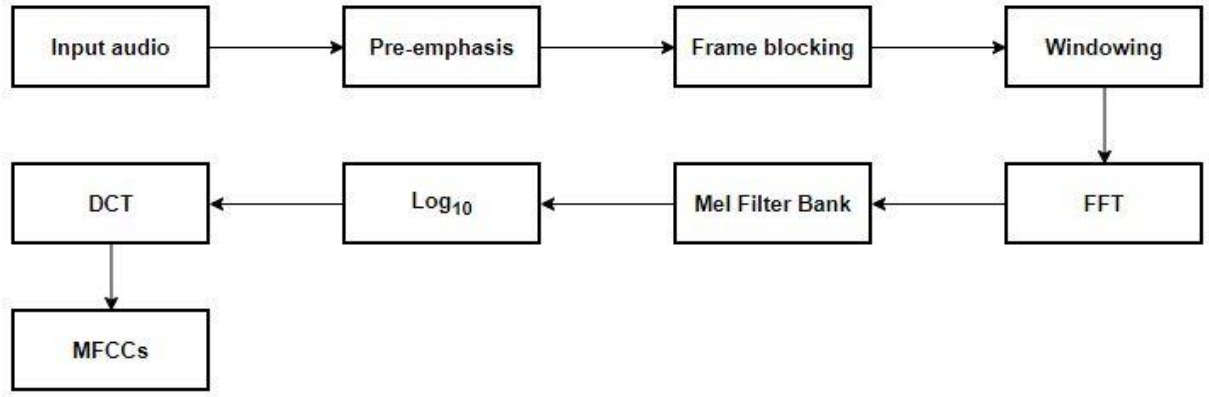


Fig. 3. MFCCs block diagram.

So, to extract features from a signal, it is common to partition it into short frames, estimate the power spectrum for each frame using periodogram analysis, apply a mel filterbank to the power spectra and calculate the energy within each filter, compute the logarithm of the filterbank energies, apply the discrete cosine transform (DCT) to the logarithmic energies, and then keep only the DCT coefficients while discarding the rest.

Further processing steps may include appending the frame energy and delta and delta-delta features to the feature vectors, as well as filtering the final features. Fig. 3 illustrates these proposed steps.

When implementing feature extraction, MFCCs are often preferred over other techniques due to their relative simplicity and robustness across various conditions [24]. MFCCs are designed to mimic the human auditory system. Steps of MFCCs (Fig. 3) are described below:

*1) Pre-emphasis*: To optimize an audio signal for subsequent processing, it is standard practice to apply a pre-emphasis filter. The purpose of this filter is to boost the energy levels of the higher frequencies, thereby emphasizing them in the overall signal.

Through the use of a first-order infinite impulse response (FIR) filter, pre-emphasis filtering can be carried out by conducting spectral flattening [25], [26]. The FIR filter used in this stage of the process is specifically represented by Eq. (3).

$$H(z) = 1 - az^{-1}, \quad 0.9 \leq a \leq 1.0 \tag{3}$$

By applying this filter, the audio signal's energy distribution is altered, with the higher frequency components becoming more prominent. This can help improve the signal-to-noise ratio and enhance overall signal quality, making it easier to extract useful information from the audio data.

*2) Frame blocking*: When analyzing time-varying signals like audio, it is vital to balance the need for signal accuracy with the practicalities of signal processing. This is particularly true when it comes to frame blocking, the process of dividing a signal into smaller segments, or frames, to facilitate analysis. If the signal is too long, its properties may be altered, while too-short frames can compromise the resolution of narrow-band components. To achieve an optimal balance between these considerations, audio signals are typically divided into frames of 20-30 milliseconds, with adjacent frames separated by M samples (where M<N). M is frequently set to 100, and N to 256.

For the analysis of time-varying signals whose characteristics are fixed over short time intervals, framing is needed. By segmenting the signal into smaller frames, spectral analysis can be conducted on individual segments, enabling a more precise analysis of the signal's characteristics.

*3) Windowing*: After dividing the audio signal into frames, the next step is to apply a window function to each frame. The Hamming window, which is provided by the equation, is a frequent option for this:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{4}$$

In this case, 'n' varies from 0 to N-1, with 'N' being the window length, pertaining to audio signals sampled at 16 kHz, a standard frame length of 25 ms is used, which translates to a frame length of 400 samples. The frame step is typically set to 10 ms (or 160 samples), which allows for overlapping between adjacent frames. The first frame starts at sample 0, followed by the next frame starting at sample 160, and so on until the end of the signal. In cases where the audio file cannot be evenly divided into frames, zero padding is used to make up the difference.

Fredric J. Harris [27] compares the various sorts of windows that are accessible in detail. Fig. 4 displays the Hamming window function's resulting plot (with 200 samples).

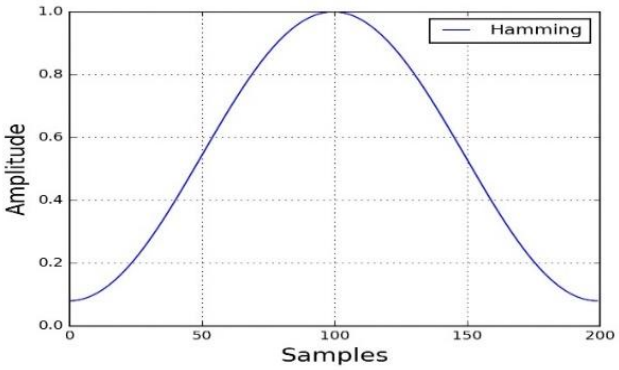


Fig. 4. Hamming window.

*4) FFT (Fast Fourier Transform)*: The Fourier transform changes a signal's time domain to its frequency domain, can be used to show a spectrum on a computer screen. A spectrum essentially depicts the frequency domain manifestation of the time-domain signal of an audio input [28].

Using mathematics, the discrete Fourier transform (DFT) converts a constrained sequence of uniformly spaced function samples into a sequence of equally spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The DFT can be written as [29], which transforms a sequence of N complex numbers into another sequence of complex numbers.

$$\hat{x}(k) = \sum_{n=0}^{N-1} x(n) e^{-2\pi i k n / N}$$

$$\text{for } k = 0,1,\ldots, N-1 \qquad (5)$$

Since DFT operates on a limited amount of data, it can be executed on computer devices using numerical algorithms or specialized hardware [30]. The effective Fast Fourier Transform (FFT) techniques are frequently used in these tasks. The terms "FFT" and "DFT" are frequently used interchangeably. The abbreviation "FFT" may have also been used to refer to the ambiguous word "Finite Fourier Transform" before its present use [31] [32].

Since the mathematical procedure known as the Fast Fourier Transform (FFT) allows for the transfer of signals from the time domain to the frequency domain, by applying the FFT to each frame, we can obtain the magnitude frequency of the signal. Thus, the output of the FFT process results in either a Periodogram or a Spectrum, as stated in reference [33]. So, this process is critical in signal processing and provides valuable insights into the characteristics of the signal.

Fig. 5 describes the application of FFT on saxophone signals, enabling the conversion of frequency information into a magnitude-based domain.

*5) Triangular bandpass filter*: A bandpass filter is an electronic filter that permits only a specific range of frequencies to pass through while suppressing or obstructing frequencies outside that range. It is engineered to transmit signals within a designated bandwidth while impeding signals that are beyond it. Bandpass filters are frequently employed in a wide range of applications such as audio processing, medical equipment, and wireless communication systems [34] [35]. These filters can be constructed using different methods, including passive RC filters, active filters, and digital filters. The center frequency, bandwidth, and quality factor are crucial design parameters that govern the filter's selectivity, gain, and noise performance.

To obtain a smooth magnitude spectrum for our research, which significantly shrunk the feature size, we used 20 triangle bandpass filters. We need to note that Hertz is represented by f in this context. The linear scale frequency is converted to Mel scale frequency using Eq. (6), which is defined as,

$$\text{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \qquad (6)$$

Spectral envelop extraction is achieved by utilizing triangular bandpass filters, as described in reference [36]. Mel frequency filters are constructed using triangular bandpass filters that are dispersed unevenly along the Mel frequency axis. In other words, the low-frequency axis has a higher density of filters, whereas the high-frequency area has a lower density of filters. References [37] and [38] are used to support this, which is shown in Fig. 6 (for 26 filters).

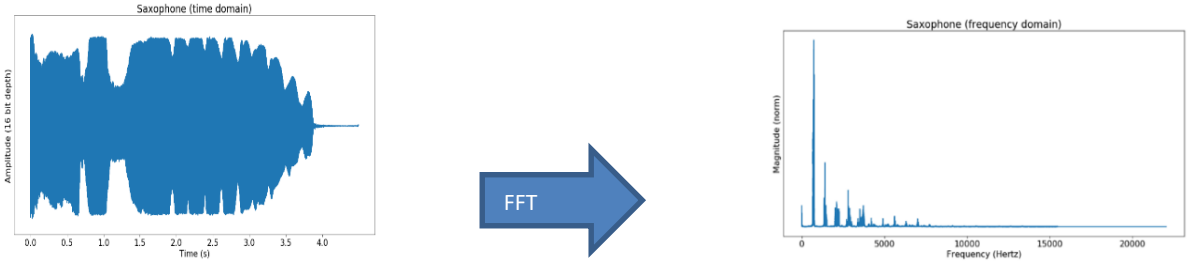


Fig. 5. FFT transformation.

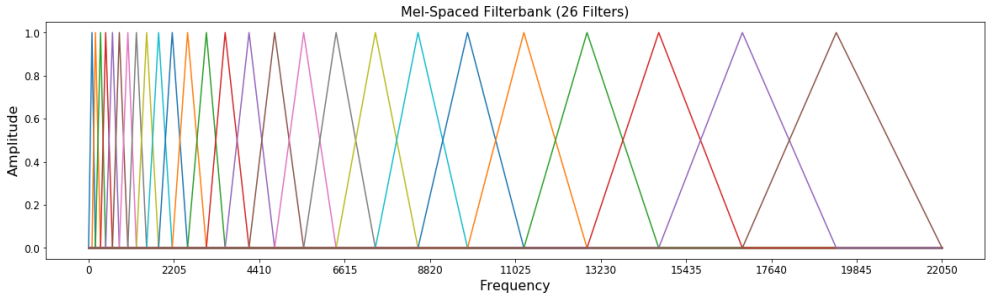


Fig. 6. Filter bank.

In order to create a filter bank, Eq. (6) is employed, which is visible in Fig. 6. The equation is defined as follows:

$$Hm(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)} & f(m-1) \le k \le f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \quad (7)$$

In this case, the letters "M" stand for the total number of filters used (26 in Fig. 6), while the letters "f()" stand for a list of M+2 Mel-spaced frequencies. The plots of the 26 filters cross over, each filter bank having a different pattern. The first filter starts at the first point, peaks at the second point, and then resets to zero at the third. The continuation of this pattern for succeeding filter banks results in an orderly evolution that improves the accuracy and thoroughness of our study.

Overall, a significant step in lowering feature size and obtaining spectral envelop extraction is the use of Mel frequency filters and triangular bandpass filters. These filters work by applying a non-uniform distribution along the Mel frequency axis, so more filters are found in the low-frequency zone and fewer in the high-frequency region. A filter bank that can be utilized for several purposes can be made using this process.

*6) The filter's energy logarithm:* The logarithm of filter energy is a commonly employed technique in audio classification, which entails computing the energy of an audio signal in particular frequency bands, followed by taking the logarithm of those energies.

This approach generally reduces the dimensionality of the feature space by transforming the raw energy values into logarithmic values, which are less sensitive to small fluctuations in signal amplitude. This is crucial since audio signals can exhibit a wide range of amplitudes, and the logarithmic transformation helps to normalize energy values across different signals. Moreover, the logarithm of filter energy can capture both high- and low-energy components of a signal, rendering it valuable for classification tasks such as music genre classification or speech recognition. The technique enables the extraction of pertinent features from an audio signal, such as energy distribution across different frequency bands, which can differentiate between various audio signals.

Our study uses Eq. (8) to calculate log-energy by adding the filtered components from each filter. This process offers insightful information about the data.

$$S(m) = \log_{10}[\sum_{K=0}^{N-1} |X(k)|^2 . Hm(k)] \quad 0 \le m \le M \quad (8)$$

We determine the log-energy, denoted as S(m), by taking the base-10 logarithm of the spectral magnitude weighted sum within the filter bank's channel. Specifically, the sum of the squared magnitudes of the discrete Fourier transform (DFT) coefficients in each frequency bin is multiplied by the corresponding filter weights (Hm(k)). This calculation is performed for each filter bank channel, resulting in a log-energy value for each bin per frame of filter.

Overall, the logarithm of filter energy is a potent tool in audio classification, often combined with other techniques such as Mel Frequency Cepstral Coefficients (MFCCs) to achieve high classification accuracy. Recent studies have shown that combining the logarithm of filter energy with deep learning approaches can significantly enhance the performance of audio classification systems [39] [40] [41].

*7) DCT (Discrete Cosine Transform):* A widely used mathematical technique for evaluating and processing various sorts of signals, including audio signals, is the discrete cosine transform (DCT). In the realm of audio classification, the DCT is frequently utilized to alter an audio signal from the time domain to the frequency domain, thereby making it possible to efficiently analyze and extract essential features.

The DCT mainly breaks down a signal into a collection of cosine functions of differing frequencies, with each function having its own amplitude. The output frequency coefficients represent the contribution of each frequency component to the original signal. These coefficients can then be deployed as features for audio classification.

Several variations of DCT are available, with DCT Type II being the most commonly used version, often referred to as the "standard" DCT. This version is employed in the widely popular audio compression format, MP3. So we can say that DCT is an immensely powerful tool for audio analysis and classification. By capturing vital frequency information that is not immediately evident in the time domain, the DCT greatly enhances the accuracy and efficiency of audio classification [42] [43].

We applied the Discrete Cosine Transform (DCT) to transform the Mel frequency domain, which characterizes the

logarithmic power spectrum of an audio signal, back into the time domain [44]. This crucial step yields Mel Cepstral Coefficients as the output. The Mel Frequency Cepstral Coefficient (MFCC) was the final preprocessing step, as described in a paper [45]. The MFCC version produces a more condensed image compared to the Mel filterbank. It achieves decorrelation of many energies from the prior energy band using the DCT, a compression method utilized for audio and image files. By converting higher frequencies to lower frequencies, the DCT principle compresses audio and image data, allowing different sounds to have distinct visual representations. This completes the data preprocessing stage.

## IV. Dataset Overview

Our final objective was to increase our model's accuracy, even when it was trained on a modestly sized dataset. This was accomplished using a portion of the Freesound Dataset Kaggle 2018 ("FSD Kaggle 2018") dataset, a considerably larger dataset than the one we utilized. The total dataset is many gigabytes in size and consists of forty-two audio classes. More information about the dataset can be found at [46] [47].

We also experimented with a seemingly more extensive dataset compared to FSD, namely ESC-50, having a sample rate of 44100 KHz and a substantial size of approximately 2 gigabytes. This dataset encompasses 50 distinct classes and comprises a total of 2000 audio files. To further amplify its scale, we applied augmentation techniques.

Upon our rigorous customization of the dataset, we have taken the initiative to share it on Kaggle, ensuring it serves as a substantial resource for future researchers. The concise FSD-Kaggle dataset as well as the CSV file can be found at https://www.kaggle.com/datasets/jewelmd/subset-of-fsd-kaggle-2018, while the augmented variant is available at this link: https://www.kaggle.com/datasets/jewelmd/augmented-esc-50-441-khz.

Through a combination of careful dataset selection and model construction, we were able to achieve our target of higher accuracy, even with a limited dataset. Our results demonstrate the effectiveness of our approach, as well as the importance of careful dataset selection and model construction in achieving accurate and reliable results.

### A. Contents of Our Dataset

From the Kaggle competition, we have specifically chosen ten diverse classes pertaining to musical instruments where we have 300 audio files (30 audio files per class). We will also be working with a CSV file that will help us associate the audio files' obscure names with the respective musical instrument classes. The ten instrument classes we are working with are 'Acoustic_guitar', 'Bass_drum', 'Cello', 'Clarinet', 'Double_bass', 'Flute', 'Hi-hat', 'Saxophone', 'Snare_drum', and 'Violin_or_fiddle'. We will employ advanced analytical techniques to classify these instruments based on the data we have gathered.

The ESC-50 dataset initially comprised 50 distinct classes, 40 files per class, total (40 x 50) 2000 audio files. Through augmentation, the dataset underwent a sixfold expansion, yielding a total of 12000 audio files. Detailed explanations of this augmentation process are provided in the dedicated Section IV on data augmentation.

### B. Employing Data Augmentation Techniques

In our endeavors with the ESC-50 dataset, we diligently applied data augmentation methods. Given our focus on audio data, we navigated through a plethora of techniques tailored specifically for enhancing this type of information. Audio data augmentation has emerged as a pivotal practice within the domain of machine learning, particularly in tasks pertaining to audio processing. This practice involves artificially amplifying the diversity of a dataset by subjecting original audio samples to an array of transformations. The overarching objective is to equip machine learning models with the capacity to adeptly handle a wider range of real-world scenarios. Notably, recent years have seen the advent of seminal research papers [48][49][50] that have propelled advancements in this domain. Below, we outline the detailed steps taken to implement data augmentation on the ESC-50 dataset in Fig. 7.

*1) Initialization and directory definitions*: In the initialization phase, necessary packages were imported. Following this, paths for both the original and augmented dataset directories were established. Then this module verifies if the augmented dataset directory already exists; if not, it creates it. This step ensures the availability of essential directories for seamless data processing.

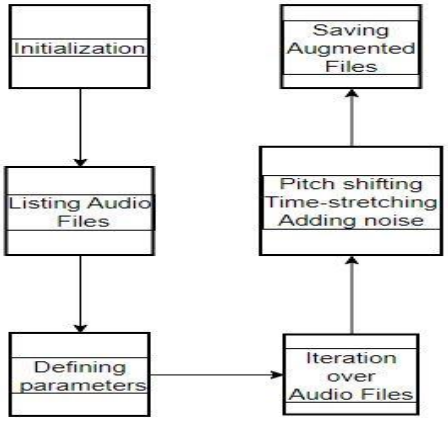


Fig. 7. Data augmentation steps.

*2) List audio files*: This step involves obtaining a list of audio files from the designated original dataset directory.

Parameter definitions

In this step, key parameters for data augmentation are established. These include pitch shift steps, time stretch factors, and noise levels, which are essential for modifying the audio data.

Below are the parameters we applied to the audio files, visible in Table I.

TABLE I. AUGMENTATION TYPES AND FACTORS

| Augmentation type | 1st factor | 2nd factor |
|---|---|---|
| Pitch shift step | -2 | 2 |
| Time stretch factor | 0.8 | 1.2 |
| Noise level | 0.001 | 0.01 |

*3)* Iteration over audio files: This step involves a loop that iterates through each audio file in the list. For each file, it performs two tasks: extracts the class label from the file name and loads the audio file.

*4) Data augmentation*: In the data augmentation phase, an empty list named augmented_audios is initialized to store modified versions of the audio. Pitch shifting is applied for each specified pitch shift step, and the augmented audio is appended to the list. Similarly, time stretching is implemented for designated factors, and the altered audio is added to augmented_audios. Additionally, background noise is introduced by generating random noise and combining it with the audio. This augmented audio is then included in the augmented_audios list, completing the data augmentation process.

*5) Saving augmented files*: In this step, each augmented audio file from the list augmented_audios undergoes a two-part process: first, a unique file name is generated, and then the augmented audio is saved to the designated augmented dataset directory. This ensures that the augmented versions are properly stored for future use.

We derived six additional files from a single audio recording. Initially, the ESC-50 dataset comprised 2000 audio files. After implementing data augmentation, this number multiplied to 12000 (2000 x 6). This expansion is due to the application of three distinct types of data augmentation, each with two contributing factors, resulting in a sixfold increase in dataset size.

*a) CSV file generation*: With the dataset update resulting in a total of 12000 audio files, it became imperative to also update our CSV file for training purposes. To accomplish this, we developed a script that generated a new CSV file containing all the newly created file names and their respective categories.

To implement the proposed approach for audio data classification, it is necessary to set up a folder (named 'wavfiles') to store all the raw audio files and a corresponding CSV file is required too. This CSV file should consist of at least two columns: one labeled 'filename' containing the names of the audio files, and the other labeled 'category' representing their respective classes.

For instance, if we have an audio file named 'Audio_file_001', it would be associated with the class 'Flute'. While the CSV file may contain additional columns like 'take' or 'length', our primary focus will be on these two columns.

## V. DATASET PRE-PROCESSING AND CLEANING

Our whole model, including the pre-processing phases, was conducted within a Python environment (version 3.7) before beginning the analysis. We carefully incorporated crucial libraries like "Python speech features," "Tqdm," "Librosa," and other necessary packages to enable a seamless analysis, establishing a solid platform for a thorough study of the data.

We carried out a thorough analysis of the distribution of all classes in our audio dataset (Fig. 8), which revealed a significant amount of dead space in the audio files. Eliminating these duplicate sections will greatly improve the quality and effectiveness of our study, producing more reliable and significant outcomes.

To prepare an audio dataset for classification, it is needed to remove any dead spots, i.e., the silent parts in the files. This process, known as cleaning, ensures that the data is of high quality and is free of any unnecessary noise. As depicted in Fig. 9, after cleaning the dataset and storing it in a separate directory, the distribution of classes has undergone a transformation, indicating the effectiveness of this approach in enhancing the quality of the data. We performed this cleaning process on all of our datasets, including FSD, ESC-50, and Augmented ESC-50. However, in the Figure, only FSD-Kaggle is depicted.

### A. Plotting and Cleaning

The first step in the procedure was to create a directory called "Clean" that would be used to store the cleaned audio recordings. We also initialized four dictionaries that were crucial to the task at hand: signals, FFT, Filter bank, and MFCCs. We chose to use 26 filters, 512 FFT, and a signal rate of 16000 for each dictionary. Additionally, with a 25 ms window size, we used the short-term Fourier transformation as well as an 1103 sample per second sampling frequency. The ideal number for our needs was 13, hence the MFCCs were programmed to have 13 Cepstral Coefficients. For both versions of the ESC-50 dataset, we employed a signal rate of 44100, as the data versions we utilized were formatted at this rate.

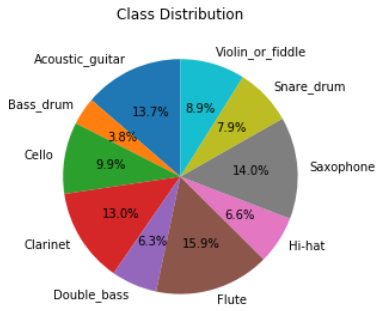


Fig. 8. Before cleaning.

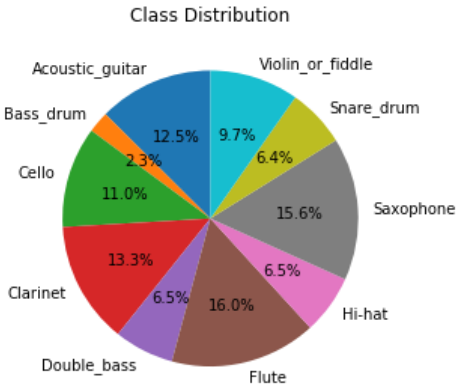


Fig. 9. After cleaning.

*1) Removing dead spots*: To optimize the quality of the audio signal, we executed a series of steps to prepare our data for effective training. The specific procedures are outlined in Table II.

The process aims to enhance audio data quality for analysis. It begins by smoothing amplitude representation with an envelope calculation. Sample rates are adjusted for compatibility (44100 Hz for ESC-50, 16000 Hz for FSD). A mask generated from the envelope function refines the signal, followed by filtering for data accuracy. Processed audio files are stored in a dedicated "clean" directory for organized analysis.

TABLE II. STEPS TAKEN FOR REMOVING DEAD SPOTS FROM AUDIO FILES

| Steps | Description |
|---|---|
| 1. Calculate Envelope | Utilized a window size of 0.1s and a frequency of 1 period/minute to obtain a smooth amplitude representation. |
| 2. Up/Down-Sampling | Adjusted the sample rate to 16000 Hz (for FSD), 44100 Hz (for ESC-50) for compatibility and signal refinement. |
| 3. Generate Mask | Utilized the envelope function to create a mask and applied it at a 0.005 rate after adjusting the sample rate. |
| 4. Apply Filter | Successfully removed redundant or erroneous data using a filter. |
| 5. Create "clean" Directory | Established a directory named "clean" to store processed audio files. |

For greater clarity, a specific example of the 'Flute' is presented, illustrating its appearance before being cleaned, which exhibited several dead spots. Following the removal of these dead spots from the audio file, a visual representation of the cleaned Flute can be observed in Fig. 10 and Fig. 11 as it is evident that there are a lot of dead spots.

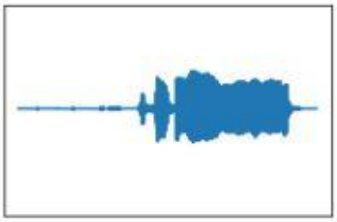

Fig. 10. Before removing dead space.

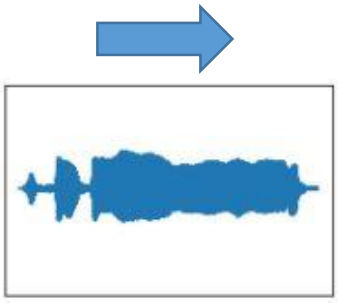

Fig. 11. After removing dead space.

## B. Exploratory Data Analysis (EDA)

As part of the preprocessing phase, we created an 'eda.py' file with the following functionalities:

This script is designed to conduct a comprehensive analysis of the audio dataset. It encompasses tasks such as feature extraction, generating visualizations, and potentially deriving insights into the characteristics of the audio files. The visualizations produced by this script serve as valuable aids for informing further analysis or gaining a deeper understanding of the dataset before proceeding with more advanced tasks like machine learning or signal processing.

To begin, we imported required libraries including os, tqdm, pandas, numpy, and matplotlib. Following that, we established several plotting functions to facilitate visual representation and analysis.

Plotting Functions:

*1) Plot_signals(signals)*: This function takes a dictionary of time series signals and plots them in a 2x5 grid, showing the waveforms of different audio samples, visible in Figure 12.

*2) Plot_fft(fft)*: This function takes a dictionary of Fourier Transforms and plots them in a 2x5 grid, displaying the frequency domain representation of different audio samples, visible in Fig. 13.

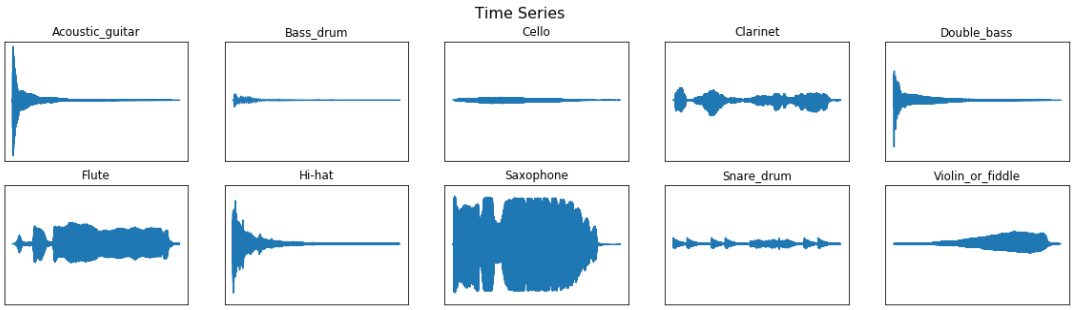


Fig. 12. Time series plot for clean data.

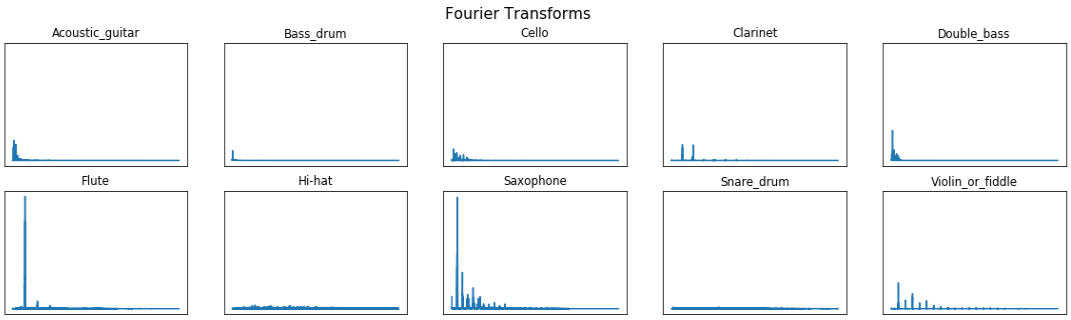


Fig. 13. FFT plot of clean data.

*3) Plot_fbank(fbank)*: This function takes a dictionary of Filter Bank Coefficients and displays them in a 2x5 grid as images, showing the distribution of frequency components, visible in Fig. 14.

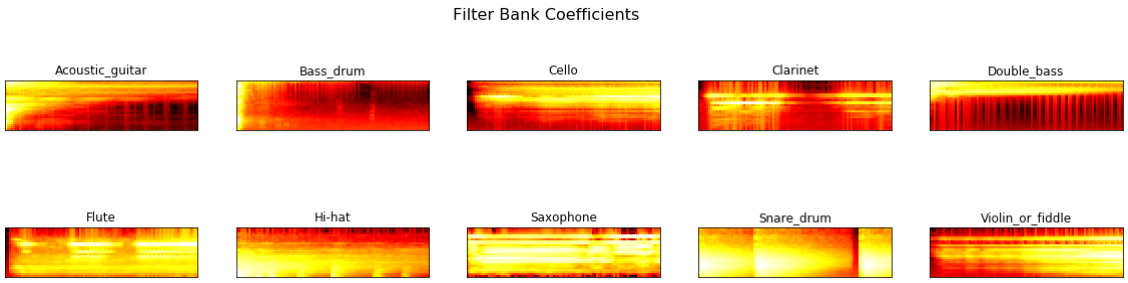


Fig. 14. Filter Bank plot of clean data.

*4) Plot_mfccs(mfccs)*: This function takes a dictionary of Mel Frequency Cepstrum Coefficients and displays them in a 2x5 grid as images, representing the features of audio signals, visible in Fig. 15.

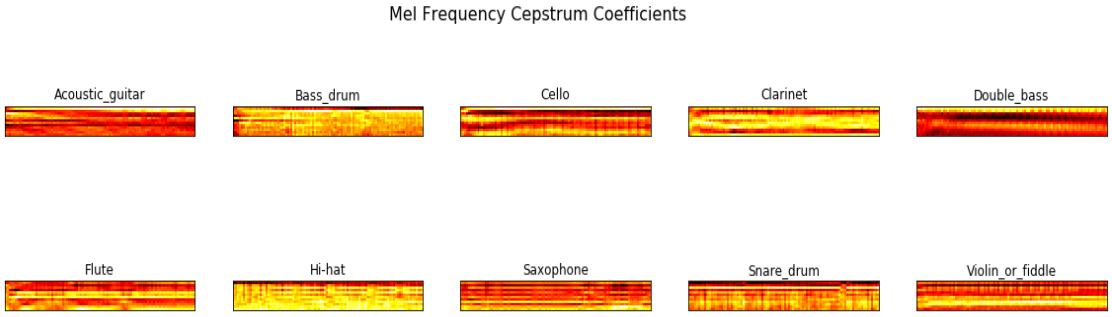


Fig. 15. MFCCs plot of clean data.

We applied these procedures to all three datasets: FSD-Kaggle, ESC-50, and Augmented ESC-50. However, in this demonstration, we are specifically showcasing the plotting for the FSD-Kaggle dataset.

Then we read the CSV file and created a DataFrame (df) to store the data. We also set the index of the DataFrame, which is likely a unique identifier for each audio file.

In the next stage to process the audio files, this script reads the WAV file located in the wavfiles/ directory and computes the sample rate and the signal. It then calculates the length of the audio file in seconds and stores it in the DataFrame under the column 'length'.

So, this script (eda.py) is designed to read and process a dataset of audio files, extracting features like signal length, and generating visualizations to help analyze the audio data. It leverages libraries like Pandas, NumPy, Matplotlib, and others for efficient data handling and visualization

## VI. Model Building

### A. Model Preparation

To enhance our model, we focused on managing class distribution and balance during training. With a specially designed function, we generated the input (X) and target (Y) matrices, randomly sampling one-second audio chunks, utilizing only a tenth of a second per sample. This data transformation enabled accurate prediction of the target variable Y, significantly improving our analysis.

We also paid close attention to the model's properties, including its sampling rate, window length, step size, and N FFT value. By taking a meticulous approach to model preparation, we were able to optimize our neural network's performance and accuracy. The procedures for building this model were completed by following Seth Adams' guidelines on audio classification [51]. So, we developed a separate script named 'cfg.py' to handle configuration settings. These settings are particularly pertinent to the processing of audio data. Within this script, we constructed a class named 'Config' with the specific purpose of managing these parameters. The 'Config' class not only provides predefined values for certain parameters but also allows for tailored adjustments when an instance of the class is created. The outlined configurations are detailed below in Table III:

TABLE III. Name and Value of the Parameter

| Parameter | Property | Default Value (customizable) |
|---|---|---|
| mode | A string, indicating the mode | 'conv' |
| nfilt | An integer, representing the number of filters | 26 |
| nfeat | An integer, specifying the number of features | 13 |
| nfft | An integer, representing the size of the Fast Fourier Transform (FFT) | 512 |
| rate | An integer, denoting the sample rate | 16000 |

Due to their customizable nature, we fine-tuned these values to align with our specific needs. For instance, we configured the mode to 'time' when training our model on RNN and adjusted the rate to 44100 Hz for the ESC-50 dataset. We also defined the step size here which is one-tenth of the sample rate. We used it for processing audio data.

### B. Convolutional Neural Network (CNN) Model

Audio classification can employ both 1D and 2D Convolutional Neural Networks, based on the input's data representation type. When the audio's time and frequency domains need to be analyzed, a 2D CNN is more appropriate. A 2D CNN was used for both ethnicity recognition and gender classification tasks in [52] but the feature maps extracted from the input images were combined and encoded into a 1D vector to facilitate classification. For tasks that involve the temporal structure of the audio, a 1D CNN is more suitable. Ultimately, the selection of the CNN architecture should be based on the audio data's specific characteristics and the classification task requirements.

During this phase of our project, we have successfully constructed several Convolutional Neural Network (CNN) models. The first step in building this model involved decoding the hot encoded Y matrix and converting it back to its original class form. We utilized the powerful Numpy Argmax function to accomplish this, which allowed us to map the encoded data back to its original column with ease.

The next step involved specifying the input shape for the convolutional layer, a critical aspect in ensuring the model's efficacy in detecting underlying data features. Key parameters, such as batch size, epochs, shuffling (enabled), class weighting (utilized Scikit-learn), and the reserved test data proportion, were defined. Following this, a sequence of convolutional and pooling layers was implemented to compress and stack the data over time, effectively reducing the dimensionality of high-dimensional input spaces. This process enabled the construction of a CNN architecture adept at capturing significant data features. The specific architecture of CNN models is given below in Table IV.

The ESC-50 CNN model consists of 10 layers, including Conv2D layers with varying filter numbers (16, 32, 64, 128) and 3x3 kernels with ReLU activation. It also features MaxPooling, Dropout (0.5), Flatten, and Dense layers with 128 and 64 units, each followed by ReLU activation. The output layer has 50 units with Softmax activation. The model utilizes the Adam optimizer and employs Categorical Crossentropy as the loss function.

TABLE IV. SEQUENTIAL CNN ARCHITECTURE

| ESC-50 – CNN model | Augmented ESC-50 – CNN model |
|---|---|
| **Number of Layers: 10** | **Number of Layers: 15** |
| **Layer Types and Details:**<br>- Conv2D (16 filters), (3x3), ReLU<br>- Conv2D (32 filters), (3x3), ReLU<br>- Conv2D (64 filters), (3x3), ReLU<br>- Conv2D (128 filters), (3x3), ReLU<br>- MaxPool2D<br>- Dropout (0.5)<br>- Flatten<br>- Dense (128 units), ReLU<br>- Dense (64 units), ReLU<br>- Output Dense (50 units), Softmax | **Layer Types and Details:**<br>- Conv2D (128 filters), (3x3), ReLU<br>- Batch Normalization<br>- MaxPool2D<br>- Conv2D (256 filters), (3x3), ReLU<br>- Batch Normalization<br>- MaxPool2D<br>- Conv2D (512 filters), (3x3), ReLU<br>- Batch Normalization<br>- MaxPool2D<br>- Flatten<br>- Dense (1024 units), ReLU<br>- Dropout (0.5)<br>- Dense (512 units), ReLU<br>- Dropout (0.5)<br>- Output Dense (50 units), Softmax |
| **Optimizer:** Adam | **Optimizer:** Adam (Learning Rate: 1e-3) |
| **Loss Function:** Categorical Crossentropy | **Loss Function:** Categorical Crossentropy |
| | **Additional Techniques:**<br>- Learning Rate Scheduling<br>- Early Stopping |

The Augmented ESC-50 CNN model has 15 layers, featuring Conv2D layers with varying filters, Batch Normalization, MaxPooling, Flatten, Dense layers, and Dropout for regularization. The output layer has 50 units with Softmax activation. The model uses the Adam optimizer (LR: 1e-3) and Categorical Crossentropy as the loss function. Additional techniques include Learning Rate Scheduling and Early Stopping.

Here are two CNN models designed for the ESC-50 and augmented ESC-50 datasets. The model used for the FSD-Kaggle dataset (with 10 classes) mirrors that of ESC-50, with the only difference being the last dense layer which has 10 units. We implemented various supplementary techniques as outlined below.

*1) Learning rate scheduling*: Learning Rate Scheduling dynamically adjusts the learning rate during training. We implemented a custom schedule using a function, lr_schedule(epoch), which scales the rate based on epoch thresholds (e.g., 10, 20, 30, 40).

*2) Early stopping*: Early Stopping prevents overfitting by monitoring validation loss and stopping training after a set number of epochs with no improvement (patience=10). Using restore_best_weights = True ensures the model retains the best state.

*3) Batch normalization*: Batch Normalization stabilizes training, speeds up convergence, and reduces overfitting by normalizing activations within each layer. This leads to better generalization and enables the use of higher learning rates, ultimately enhancing the model's performance.

### *C. RNN (Recurrent Neural Network) Model*

In our ongoing efforts to optimize machine learning algorithms, we developed RNN models to complement our existing CNN architecture. Unlike CNNs, RNNs employ LSTM (Long Short-Term Memory) units, which excel in processing sequential data due to their long-term memory capabilities. Our RNN model demonstrated exceptional proficiency in learning from such data. Our rigorous training and testing procedures guaranteed that the model would exhibit accuracy and adaptability to new datasets. More detailed descriptions of our RNN model can be found in the Table V:

The FSD-Kaggle RNN model (10 classes) comprises nine layers, including two LSTM layers with 128 units each. It incorporates a Dropout layer (rate: 0.5) for regularization, followed by TimeDistributed Dense layers with varying units (64, 32, 16, and 8) and ReLU activation. The model concludes with a Flatten layer and a Dense layer (10 units, softmax activation), tailored for multi-class classification. It is optimized using Adam with Categorical Crossentropy loss, suiting the classification task's requirements.

The RNN model for ESC-50 and Augmented ESC-50 datasets consists of 13 layers. It includes LSTM units, Batch Normalization, and Dropout layers. TimeDistributed Dense layers with ReLU activation are utilized, followed by Flatten and a final Dense layer for multi-class classification. The model uses Adam optimizer (LR: 0.001) and employs Categorical Crossentropy as the loss function. Additional techniques like Learning Rate Scheduling and Early Stopping are implemented for improved training performance and prevention of overfitting.

TABLE V. SEQUENTIAL RNN ARCHITECTURE

| **FSD-Kaggle**(10 classes) – **RNN model** | **ESC-50 & Augmented ESC-50 – RNN model** |
|---|---|
| **Number of Layers: 9** | **Number of Layers: 13** |
| **Layer Types and Details:**<br>- LSTM (128 units), return_sequences=True, input_shape=input_shape<br>- LSTM (128 units), return_sequences=True<br>- Dropout (0.5)<br>- TimeDistributed(Dense(64, activation='relu'))<br>- TimeDistributed(Dense(32, activation='relu'))<br>- TimeDistributed(Dense(16, activation='relu'))<br>- TimeDistributed(Dense(8, activation='relu'))<br>- Flatten<br>- Dense (10 units, softmax) | **Layer Types and Details:**<br>- LSTM (256 units), return_sequences=True, input_shape=input_shape<br>- Batch Normalization<br>- Dropout (0.3) or [0.2 for augmented]<br>- LSTM (256 units), return_sequences=True<br>- Batch Normalization<br>- Dropout (0.3)<br>- TimeDistributed(Dense(128, activation='relu'))<br>- Batch Normalization<br>- TimeDistributed(Dense(64, activation='relu'))<br>- Batch Normalization<br>- TimeDistributed(Dense(32, activation='relu'))<br>- Flatten<br>- Dense (50 units, softmax) |
| **Optimizer:** Adam | **Optimizer:** Adam (Learning Rate: 0.001) |
| **Loss Function:** Categorical Crossentropy | **Loss Function:** Categorical Crossentropy |
| | **Additional Techniques:**<br>- Learning Rate Scheduling<br>- Early Stopping |

ESC-50 and augmented ESC-50 models were similar, with dropout rates of 0.3 and 0.2 respectively. Additional techniques were applied with adjusted rates compared to the CNN model. Here, *Learning Rate Scheduling* is implemented with an initial constant rate for the first 10 epochs, followed by an exponential decrease. Early Stopping is employed to halt training if no improvement is detected over six consecutive epochs.

### *D. Comparison of ModeTraining Parameters*

For all models, the class weight is consistently set to 'Balanced'. Monitors are configured as 'val_acc' and 'val_accuracy' in 'max' mode, while both 'save_best_only' and 'save_weights_only' are uniformly set to 'True'. The main distinguishing factors emerge in the number of epochs, the allocation for validation split, and the incorporation of class weights alongside a learning rate scheduler, visible in Table VI.

TABLE VI. COMPARISON OF TRAINING CONFIGURATIONS

| **Parameter** | **FSD-Kaggle** | | **ESC-50** | | **Augmented ESC-50** | |
|---|---|---|---|---|---|---|
| | **CNN** | **RNN** | **CNN** | **RNN** | **CNN** | **RNN** |
| **Period** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Batch Size** | 32 | 32 | 32 | 32 | 32 | 32 |
| **Shuffle** | True | True | True | True | True | True |
| **Validation Split** | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 |
| **Epochs** | 15 | 15 | 100 | 30 | 50 | 30 |
| **Learning Rate Scheduler** | No | No | No | Yes | Yes | Yes |
| **Early Stopping** | No | No | No | Yes | Yes | Yes |
| **Total Files** | 300 | | 2000 | | 12000 | |

## VII. ANALYSES OF RESULTS

Within this section, we will assess the performance of multiple deep learning models, including Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). The CNN models harnessed convolutional and pooling layers to effectively capture underlying data features, resulting in impressive accuracy rates within relatively short training periods. On the other hand, RNN models, employing LSTM units for sequential data processing, required more time to train due to their computational complexity. After training across various epochs for each respective model, we achieved good accuracy levels. A comparative table (Table VII) detailing the performance of these diverse models is provided below.

TABLE VII. COMPARATIVE TABLE DETAILING THE PERFORMANCE OF THESE DIVERSE MODELS

| **Model** | **Architecture** | **Accuracy (%)** | | **Loss (%)** | |
|---|---|---|---|---|---|
| | | **Train** | **Test** | **Train** | **Test** |
| FSD-Kaggle (small dataset) | CNN | 96.26 | 96.52 | 10.03 | 9.99 |
| | RNN | 87.84 | 87.88 | 33.60 | 34.55 |
| ESC-50 | CNN | 86.33 | 88.47 | 45.02 | 40.86 |
| | RNN | 92.86 | 92.89 | 22.70 | 27.38 |
| Augmented ESC-50 | CNN | 71.25 | 76.20 | 104.41 | 84.84 |
| | RNN | 77.29 | 79.18 | 80.90 | 73.23 |

In our study, we observe that for smaller datasets, CNNs outperform RNNs. As dataset size increases, RNNs prove superior in learning underlying features. CNNs efficiently capture important data features but may struggle with sequential data. Conversely, RNNs, with LSTM units, excel in processing sequences, but at a higher computational cost and time investment compared to CNNs.

We also observed a performance decrease after applying data augmentation, possibly due to factors like over-augmentation and model sensitivity. Fine-tuning augmentation parameters and exploring alternative techniques may mitigate this. Future research could delve into optimizing augmentation strategies and model configurations for improved performance. In addition, the augmented RNN's accuracy of 79.18% implies a potential for increased robustness in real-world scenarios, given its training on a dataset comprising 12,000 audio files.

### A. *Result Visualization and Analysis*

This step involves visualizing the model's performance metrics, such as accuracy and loss, over the training process. By plotting these metrics, it provides a clear overview of how well the model is learning from the data. These visualizations help in understanding the effectiveness and progress of the training process.

*1) FSD-kaggle dataset*: A peek at the accuracy and loss curves reveals good convergence for both CNN and RNN models after running for 15 epochs. So, we got higher accuracy and lower loss (given in Table VII). We achieved around 96.52% and 87.88% accuracy during testing on the FSD-Kaggle dataset for CNN and RNN models respectively. On the other hand, we had a very lower loss too for both architectures. However, on the FSD-Kaggle dataset, the performance of CNN model was better than RNN model during training and testing on dataset. The visualizations are shown in Figures 16 through 19.

*2) ESC-50 dataset*: After running 100 epochs for the ESC-50 dataset on the CNN model and 50 epochs on the RNN model, we got 88.47% and 92.89% accuracy on a testing dataset of the CNN and RNN model respectively (given in Table VII). So, here the RNN model outperforms the CNN model. We can see from the graphs that there is a bit of instability at the initial phase of training for the ESC-50 RNN model but after running for 25 epochs it seems to be stable. The visualizations are shown in Fig. 20 through Fig. 23.

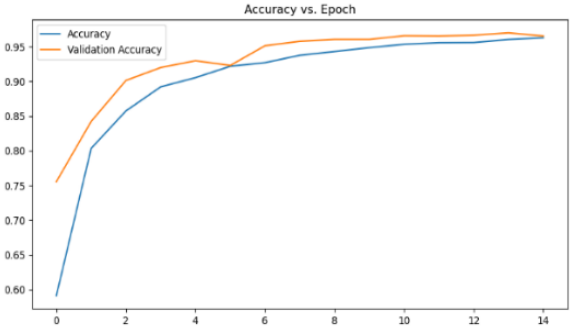


Fig. 16. Accuracy vs. epoch for FSD-Kaggle CNN model.

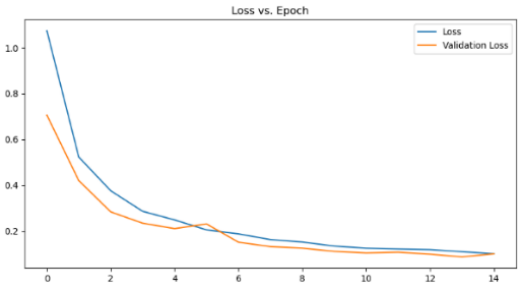


Fig. 17. Loss vs. epoch for FSD-Kaggle CNN model.

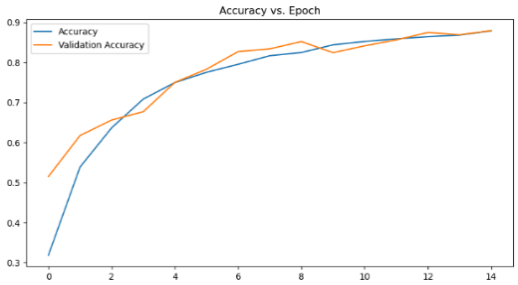


Fig. 18. Accuracy vs. epoch for FSD-Kaggle RNN model.

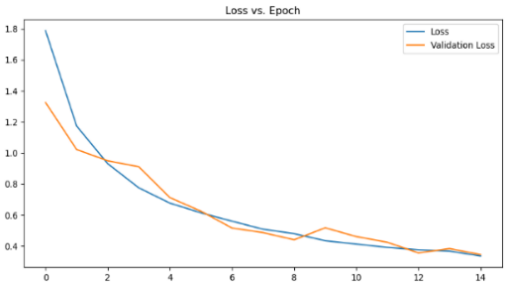


Fig. 19. Loss vs. epoch for FSD-Kaggle RNN model.

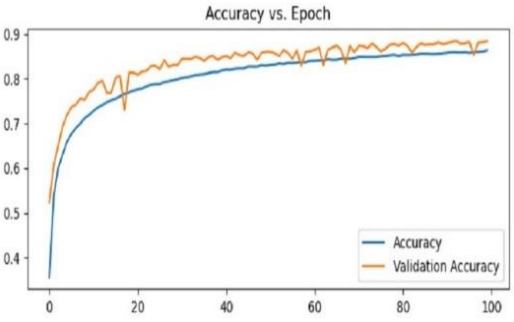


Fig. 20. Accuracy vs. epoch for ESC-50 CNN model.

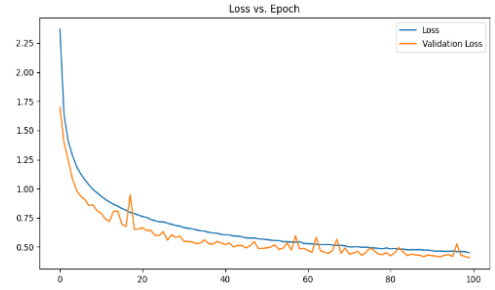


Fig. 21. Loss vs. epoch for ESC-50 CNN model.

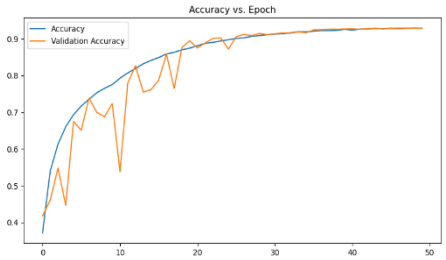


Fig. 22. Accuracy vs. epoch for ESC-50 RNN model.

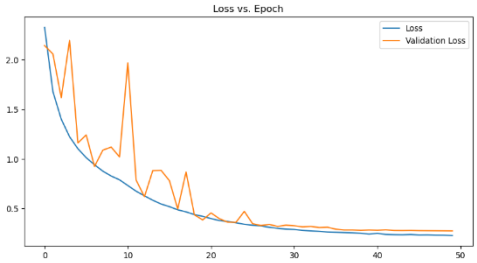


Fig. 23. Loss vs. epoch for ESC-50 RNN model.

*3) Augmented ESC-50 dataset*: From the accuracy and loss curves of CNN and RNN models on the Augmented ESC-50 dataset, it is clear that the RNN model achieved more accuracy and lower loss during training. We ran around 50 epochs for both of the models and got around 76% and 79% accuracy on testing for CNN and RNN models respectively (given in Table VII). During training, the RNN model took a bit longer time since there were LSTM layers. Probably the reason lies in the fact that LSTM cells can store more information over extended time periods. From the substantial amount of loss, it can be said that the model had difficulties adapting to the augmented features generated by the augmented dataset. This also suggests that RNN outperforms CNN on new data. The visualizations are shown in Fig. 24 through Fig. 27.

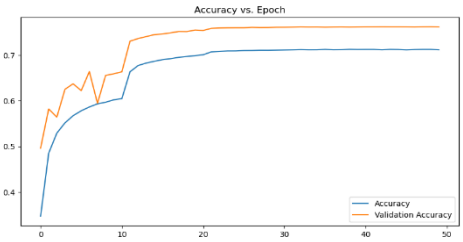


Fig. 24. Accuracy vs epoch for augmented ESC-50 CNN model.

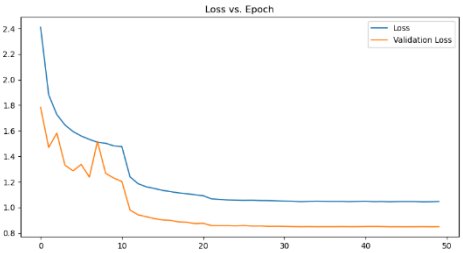


Fig. 25. Loss vs epoch for augmented ESC-50 CNN model.

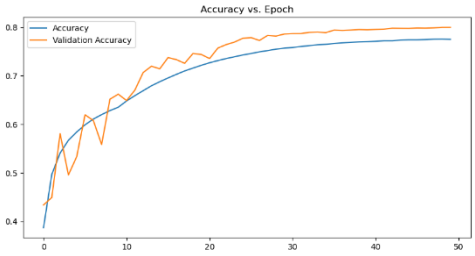


Fig. 26. Accuracy vs. epoch for augmented ESC-50 RNN model.

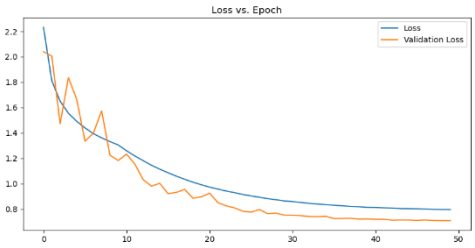


Fig. 27. Loss vs. epoch for augmented ESC-50 RNN model.

So, using different datasets, and testing various models based on CNN and RNN architectures, we have come to a conclusion where we can say that both CNN and RNN models can classify audio but the measurement of accuracy and the convergence of the graph curves depend on numerous factors including the complexity of the dataset and the ability of the model to extract several underlying features. We have also seen that for larger datasets RNN model outperformed the CNN model.

## VIII. Conclusions

This study delves into audio classification using CNN and RNN-LSTM models, exploring their performance across different dataset sizes. We found that CNNs excel with smaller datasets, efficiently capturing key features, while RNN-LSTM models better perform with larger datasets, revealing intricate underlying patterns. The impact of data augmentation was also examined, revealing a nuanced balance between augmentation and performance. While augmented models showed improved robustness, some experienced a minor accuracy reduction, highlighting the need for parameter fine-tuning. Our research contributes valuable insights for optimizing audio classification, paving the way for applications in diverse real-world scenarios. Future studies can build upon these findings to further refine these models' capabilities.

## References

[1] D. G. Bhalke, C. B. Rama Rao, D. S. Bormane, “Automatic musical instrument classification using fractional Fourier transform based- MFCC features and counter propagation neural network”, Journal of Intelligent Information Systems, 2016, Volume 46, Number 3, Page 425

[2] Monica S. Nagawade, Varsha R. Ratnaparkhe, “Musical Instrument Identification using MFCC”, 2017 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT), May 19-20, 2017, India.

[3] T. Virtanen, M. D. Plumbley, and D. Ellis, Computational Analysis of Sound Scenes and Events. Springer, 2018.

[4] Sharath Adavanne and Tuomas Virtanen. Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network. In DCASE Workshop, 2017.

[5] Sharath Adavanne, Konstantinos Drossos, Emre Çakır, and Tuomas Virtanen. Stacked convolutional and recurrent neural networks for bird audio detection. In EUSIPCO, 2017.

[6] Miroslav Malik, Sharath Adavanne, Konstantinos Drossos, Tuomas Virtanen, Dasa Ticha, and Roman Jarina. Stacked convolutional and recurrent neural networks for music emotion recognition. In Sound and Music Computing Conference (SMC), 2017.

[7] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In Content-Based Multimedia Indexing (CBMI) Workshop, pages 1–6, 2016.

[8] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In IEEE ICASSP, pages 6964–6968, 2014.

[9] K. Koutini, H. Eghbal-zadeh and G. Widmer, "Receptive Field Regularization Techniques for Audio Classification and Tagging With Deep Convolutional Neural Networks," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 1987-2000, 2021, doi: 10.1109/TASLP.2021.3082307.

[10] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw wave form cldnns. In Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[11] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms. In IEEE ICASSP, pages 421–425, 2017.

[12] Meinard Müller, Member, IEEE, Daniel P. W. Ellis, Senior Member, IEEE, Anssi Klapuri, Member, IEEE, and Gaël Richard, Senior Member, IEEE. "Signal processing for music analysis". IEEE Journal of selected topics in signal processing, VOL. 5, NO. 6, OCTOBER 2011

[13] Jadhav, P. S. (2015). Classification of Musical Instruments Sounds by Using MFCC and Timbral Audio Descriptors. International Journal of Research in Information Technology and Computing (IJRITCC), 3(7), 5001-5006. https://doi.org/10.17762/ijritcc.v3i7.4778

[14] Essid, S., Richard, G., & David, B. (2004). Efficient musical instrument recognition on solo performance music using basic features. AES 25th International Conference, London, United Kingdom, June 17-19, 2004

[15] M. Erdal Ozbek , Nalan Ozkurt and F. Acar Savaci, "Wavelet ridges for musical instrument classification",J Intell Inf Syst (2012) 38:241–256, DOI 10.1007/s10844-011-0152-9

[16] Farbod Foomany and Karthikeyan Umapathy, "Classification of music instruments using wavelet-based time-scale features", 2013 IEEE ICMEW.

[17] Y. KIKUCHI, N. AOKI and Y. DOBASHI, "A Study on Automatic Music Genre Classification Based on the Summarization of Music Data," 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 2020, pp. 705-708, doi: 10.1109/ICAIIC48513.2020.9065046.

[18] F. H. Rachman, R. Sarno and C. Fatichah, "Music Emotion Detection using Weighted of Audio and Lyric Features," 2020 6th Information Technology International Seminar (ITIS), Surabaya, Indonesia, 2020, pp. 229-233, doi: 10.1109/ITIS50118.2020.9321046.

[19] Taneja, Y. Gulati, T. Chugh, P. Joshi and N. Thakur, "Heart Audio Classification Using Deep Learning," 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2020, pp. 485-488, doi: 10.1109/ICMLA51294.2020.00082.

[20] V. Viswanath and B. P. Babu, "Vehicle Classification with Audio and Video Modalities Using CNN and Decision-Level Fusion," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), Bhimtal, India, 2020, pp. 482-486, doi: 10.1109/CICN49253.2020.9242556.

[21] S. Nivetha, "A Survey on Speech Feature Extraction and Classification Techniques," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 48-53, doi: 10.1109/ICICT48043.2020.9112582.

[22] Chauhan, P. M., & Desai, N. P. (2014). Mel Frequency Cepstral Coefficients (MFCC) based speaker identification in noisy environment using wiener filter. 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE).

[23] Karpagavalli S and Chandra E, " A Review on Automatic Speech Recognition Architecture and Approaches", International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.9, No.4, (2016), pp.393-404

[24] C. Poonkuzhali, R. Karthiprakash, S. Valarmathy and M. Kalamani, An Approach to feature selection algorithm based on Ant Colony Optimization for Automatic Speech Recognition, International journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 11(2), and 2013.

[25] Yuan Meng, Speech recognition on DSP: Algorithm optimization and performance analysis, The Chinese university of Hong Kong, July 2004, pp. 1-18.

[26] Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi, Voice recognition algorithm using MFCC & DTW techniques, Journal Of Computing, Volume 2, Issue 3, March 2010, ISSN 2151-9617, pp. 138-143.

[27] Fredric J. Harris, Mexber, IEEE, "On then Use of Windows for Harmonic Analysis with the Discrete Fourier Transform"in Proceeding of the IEEE, January 1978.

[28] Herman R. (2016), An Introduction to Fourier Analysis, Chapman and Hall/CRC, eBook ISBN 9781315367064.

[29] Andersson T. (2004). Audio Classification and Content Desicription. M Sc. Thesis, Department of Computer Science Electrical Engeerring, University of Techonology

[30] Terenzi A, Cecchi S, Sorcioni S and Piazza F. (2019), Features Extraction Applied to the Analysis of the Sounds Emitted by Honey Bees in a Beehive, in 2019 11 International Symposium on Image and Signal Processing and Analysis (ISPA), IEEE, pp. 03-08. https://doi.org/10.1109/ispa.2019.8868934

[31] Grama L and Rusu C. (2017), Audio Signal Classification Using Linear Predictive Coding and Random Forests, in 2017 International Conference on Speech Technology and Human-Computer Dialogue (SpeD), IEEE, pp. 1-9.

[32] Jasim, Wala'A & Jasim, Nibras & Abdual, Saba & Saddam, Saba & Jasem, Esra & Harfash, J. (2022). Wind Sounds Classification Using Different Audio Feature Extraction Techniques. Informatica. 45. 10.31449/inf.v45i7.3739.

[33] Parwinder Pal Singh, Pushpa Rani," An Approach to Extract Feature using MFCC", IOSR Journal of Engineering (IOSRJEN) ISSN (e): 2250-3021, ISSN (p): 2278-8719 Vol. 04, Issue 08 (August. 2014), ||V1|| PP 21-25

[34] Fadhel, M. A., & Salman, D. M. (2020). Design of Bandpass Filter using Modified Hairpin Resonator for ISM Applications. International Journal of Emerging Trends in Engineering Research, 8(6), 1769-1775.

[35] Challa, S., & Deenadayalan, E. (2019). Design of Microstrip Bandpass Filter for Wireless Applications. International Journal of Engineering and Advanced Technology, 8(4), 1262-1266.

[36] Siddhant C. Joshi, Dr. A.N.Cheeran, "MATLAB Based Feature Extraction Using Mel Frequency Cepstrum Coefficients for Automatic Speech Recognition", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 6, June 2014

[37] Yuan Meng, Speech recognition on DSP: Algorithm optimization and performance analysis, The Chinese university of Hong Kong, July 2004, pp. 1-18.

[38] Sirko Molau, Michael Pitz, Ralf Schl¨uter, and Hermann Ney, Computing Mel-frequency cepstral coefficients on the power spectrum, University of Technology, 52056 Aachen, Germany

[39] X. Zhang, X. He, and Y. Wang, "Logarithm Filter Energy-Based Audio Classification Using Convolutional Neural Networks," IEEE Access, vol. 8, pp. 28372-28379, 2020.

[40] Wang, L., Liu, Y., & Liu, J. (2021). Audio classification using logarithmic filter energy feature and convolutional neural network. Digital Signal Processing, 116, 103044. doi:10.1016/j.dsp.2021.103044

[41] Zhang, J., Li, Z., Li, X., & Jia, P. (2021). Audio classification based on multi-level feature fusion with logarithmic filter bank energies. Journal of Ambient Intelligence and Humanized Computing, 12(9), 10263-10272. doi:10.1007/s12652-020-02705-2

[42] Deep Convolutional Neural Networks with Discrete Cosine Transform for Audio Classification," by C. Bajaj, S. Saini, and S. Sharma. IEEE International Conference on Signal Processing and Communication (ICSC), 2021. DOI: 10.1109/ICSC51661.2021.9386934

[43] Discrete Cosine Transform-Based Audio Feature Extraction for Music Genre Classification," by A. Gomila and V. Perez-Marin. IEEE Access, vol. 9, pp. 37327-37337, 2021. DOI: 10.1109/ACCESS.2021.3060033

[44] Gaurav, Devanesamoni Shakina Deiv, Gopal Krishna Sharma, Mahua Bhattacharya, Development of Application Specific Continuous Speech Recognition System in Hindi, Journal of Signal and Information Processing, 2012, 3, pp. 394-401.

[45] M. S. Imran, A. F. Rahman, S. Tanvir, H. H. Kadir, J. Iqbal and M. Mostakim, "An Analysis of Audio Classification Techniques using Deep Learning Architectures," 2021 6th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 805-812, doi: 10.1109/ICICT50816.2021.9358774.

[46] M. S. Imran, A. F. Rahman, S. Tanvir, H. H. Kadir, J. Iqbal and M. Mostakim, "An Analysis of Audio Classification Techniques using Deep Learning Architectures," 2021 6th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 805-812, doi: 10.1109/ICICT50816.2021.9358774.

[47] Eduardo Fonseca1∗, Manoj Plakal2, Frederic Font1, Daniel P. W. Ellis2, Xavier Favory1, Jordi Pons1, Xavier Serra1, "General-purpose tagging of free sound audio with audio set labels: task description, dataset, and baseline", Detection and Classification of Acoustic Scenes and Events 2018.

[48] Khan, A. A., Khan, M. A., & Khan, M. A. (2022). Data augmentation and deep learning methods in sound classification: A systematic review. *arXiv preprint arXiv:2208.06099*.

[49] Wang, J., Liu, X., Sun, D., & Zhang, H. (2021). Sample mixed-based data augmentation for domestic audio tagging. *IEEE Access*, 9, 128119-128128.

[50] Sprengel, P. A., Li, S., & Wang, Z. (2023). Data augmentation on convolutional neural networks to classify mechanical noise. *Applied Acoustics*, 229, 108959.

[51] S. Adams, Audio-classification, https: //github.com/seth814/Audio-Classification, Apr. 2020

[52] M. Jewel, M. I. Hossain and T. H. Tonni, "Bengali Ethnicity Recognition and Gender Classification using CNN & Transfer Learning," *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India, 2019, pp. 390-396, doi: 10.1109/SMART46866.2019.9117549.