Urban Sound Classification using Convolutional Neural Network and Long Short Term Memory based on Multiple Features

by

Joy Krishan Das
17301218
Arka Ghosh
16201007
Abhijit Kumar Pal
16301148
Sumit Dutta
16301104

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
April 2020

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]	Arka Ghosh
Joy Krishan Das 17301218	Arka Ghosh 16201007
Abhijit kumar pal	[illegible]
Abhijit Kumar Pal 16301148	Sumit Dutta 16301104

Approval

The thesis titled "Urban Sound Classification using Convolutional Neural Network And Long Short Term Memory based on Multiple Features" submitted by

1. Joy Krishan Das (17301218)
2. Arka Ghosh (16201007)
3. Abhijit Kumar Pal (16301148)
4. Sumit Dutta (16301104)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on April 7, 2020.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dr. Amitabha Chakraborty
Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Rabiul

Dr. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Dr. Mahbub Alam Majumdar
Professor
Department of Computer Science and Engineering
BRAC University

Abstract

There are many sounds all around us and our brain can easily and clearly identify them. Furthermore, our brain processes the received sound signals continuously and provides us with relevant environmental knowledge. Although not up to the level of accuracy of the brain, there are some smart devices which can extract necessary information from an audio signal, with the help of different algorithms. And as the days pass by more, more research is being conducted to ensure that accuracy level of this information extraction increases. Over the years several models like the CNN, ANN, RCNN and many machine learning techniques have been adopted to classify sound accurately and these have shown promising results in the recent years in distinguishing spectra- temporal pictures. For our research purpose, we are using seven features which are Chromagram, Mel-spectrogram, Spectral contrast, Tonnetz, MFCC, Chroma_CENS and Chroma_cqt.We have employed two models for the classification process of audio signals which are LSTM and CNN and the dataset used for the research is the UrbanSound8K. The novelty of the research lies in showing that the LSTM shows a better result in classification accuracy compared to CNN, when the MFCC feature is used. Furthermore, we have augmented the UrbanSound8K dataset to ensure that the accuracy of the LSTM is higher than the CNN in case of both the original dataset as well as the augmented one. Moreover, we have tested the accuracy of the models based on the features used. This has been done by using each of the features separately on each of the models, in addition to the two forms of feature stacking that we have performed. The first form of feature stacking contains the features Chromagram, Mel-spectrogram, Spectral contrast, Tonnetz, MFCC, while the second form of feature stacking contains MFCC, Mel-spectrogram, Chroma_cqt and Chroma_stft. Likewise, we have stacked features using different combinations to expand our research.In such a way it was possible, with our LSTM model, to reach an accuracy of 98.80%, which is state-of-the-art performance.

Keywords: Sound Classification; Spectrograms; Urbansound8k; CNN; LSTM; LibROSA.

Acknowledgement

Firstly, we would like to thank the great almighty for whom our study was carried out without any big split.
Secondly, Dr. Amitabha Chakrabarty sir, our mentor and supervisor, for his kind assistance and guidance throughout the course of the research work. Without his encouragement and support from the very start of our research we could not have reached this stage in our thesis.
And to our families, as it would not have been possible without their full support. We are now on the brink of our graduation for their caring encouragement and devotion.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

Adam Adaptive Moment Estimation

ANN Artificial Neural Network

CENS Chroma Energy Normalized Satistics

CLDNN Compute Library for Deep Neural Network

CNN Convolutional Neural Network

ConvRBM Convolutional Boltzmann Machine

CQT Constant Q Transform

CRNN Convolutional Recurrent Neural Network

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DNN Deep Neural network

ESC Environmental Sound classification

GTSC Gammtone Spectral Coefficient

KLT Karhunen-Loeve Transform

LMCN Log-Mel spectrogram Convolutional Network

LPC Linear prediction coefficients

LSTM Long-Short Term Memory

MCN Mel spectrogram Convolutional Network

MFCC Mel-Frequency Cepstral Coefficient

RCNN Recurrent Convolutional Neural Network

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SAI	Stabilized auditory image
$STFT$	Short Time Fourier Transform
$TEO-GTSC$	TEO-based Gammatone Spectral Coefficient
$TSCNN-DS$	Two Stream Convolutional Neural Network- Dempster Shafer
WER	Word Error Rate

Chapter 1

Introduction

1.1 Background

Living in a world surrounded by different forms of sounds from different sources, our brain and auditory system is constantly identifying each sound that it hears, in its own way [11]. We humans are constantly in contact with the audio data that is in the environment around us. Starting from the sound of human voice during any form of interaction, to melodious music from different instruments, to the annoying traffic horns that we all hear, it is very evident that sound is an integral part of our everyday life. While it is very easy for the human auditory system to classify an audio, through very fast processing of the brain, it is not the same for the machines that we use.

Classifying audio or sound has been a major field of research for many years now and there have been many tried and tested methods with different models and features which has proven to be useful and accurate. The reason behind all these researches are due to the various fields where the application of ESC is very important. Classification of audio can range from fields like multimedia, bioacoustics monitoring, intruder detection in wildlife areas to audio surveillance and environmental sounds[40]. Notwithstanding, this is a difficult issue because of the complex nature of ecological sounds as far as dimensionality, diverse instrument of sound creation mixture of various sources, and absence of significant level structures typically in discourse and in numerous sorts of melodic sounds. These complex natures of sound, when coupled with different natural noises as well as the various forms of sound production mechanism makes it harder for classification of the sound to be done effectively.

There are three different stages that are attached with the classification of sound. These are pre-processing of the audio signal, specific feature extraction from that signal and finally the classification of the audio signal. Signal pre-processing partitions the input audio signal into various fragments which are utilized for extricating essential features. Feature extraction lessens the size of information and helps in the representation of the data as the feature vectors. Zero-crossing rate, spectral flux and centroid, chroma vector, MFCCs[47], STFT [47], CRP [28], and DWT are among the most well-known handcrafted features for audio classification. The different features help by reducing dimensionality as well as some noise from the audio signals that help in better classification of sound. For the sake of our approach to audio classification we have chosen the features chromagram, mel spectrogram, spectral contrast, tonnetz, MFCC, Chroma_CENS and Chroma_cqt. While we have tried and tested the accuracy of all these features with respect to each of our selected models, our main emphasis for comparison between the two models will be mainly based on the feature MFCC. The idea of MFCC is to convert time domain signals into frequency domain signals and use mel filters to mimic cochlea which is a part of our ear and has more filters at low frequency and less filters at high frequency. Thus it is safe to conclude that the feature MFCC and its characteristics are focused on the audibility of human hearing system, that can accommodate the dynamic nature of true-life sounds with the way that they are treated with feature vectors for classification. [47]

Once the steps regarding the feature extraction are all carried out, classification is required to be done using neural network models. There are a colossal number of potential techniques to construct ESC models utilizing distinctive sound or audio feature extraction procedures and AI or non-AI based models. The best ESC models comprise of one or then again increasingly standard sound element extraction methods and deep neural networks. SAI and LPC [28], DNN [45], Decision Tree Classifier, Random Forest [27], ANN[49] are some of the popular models amongst other classifier models for audio. Moreover, there are different audio datasets that are used to test and train the models for classification.

For our purpose of audio classification we are working with the Urbansound8K dataset and we are performing the classification via two different models which are CNN and LSTM respectively. This has been done to put forth a comparison among the two models, using the same features and the same dataset, to see which model gives a better accuracy result in classifying an input audio signal. The CNN models have capability to learn both from 1 dimensional as well as 2-dimensional data. CNN models are more or less similar to the Deep Neural Network models but the former has a convolution layer which deals with the raw data. The accuracy of the CNN model depends on the features that are being used as well as the number of layers of the model. A total of 8 layers have been used in our CNN model. These layers include two Conv2D layers, each of which are followed by a MaxPooling2D layer, and after that comes a flatten layer and then two fully connected dense layers, finishing off with a output layer. The first Conv2D layer has input shape consisting of 64 filters, a kernel size of 5 and stride set to 1. The node number for the first Conv2D layer is 64 and ReLu is used as an activation function for this layer. The first Conv2D layer is followed by the MaxPooling2D layer which is used in reducing the dimension of the input shape fed through the convolutional layer. Then comes the second Conv2D layer which is similar to the first one with regards to the kernel size, stride and activation function used but varies in filter size which in this case is 128. Furthermore, to reduce overfitting dropout of 0.3 is used in this layer, which is again followed by a MaxPooling2D layer. After that comes the flatten layer and followed by the two fully connected dense layers which have the dropout value and activation function in common while the difference remains with regards to the number of nodes being used, which is 256 for the first one and 512 for the second dense layer. Lastly, the output dense layer is placed with the number of nodes being

10 and the activation function used being softmax, as it helps in the classification of more than two classes.

The second model that we have used to classify sound is the LSTM.The LSTM and RNN networks have been the most popular models for a range of machine learning challenges in recent years. Thus many researchers are now interested in investigating the function and usefulness of many LSTM in the field of sound classification as well [23]. This model has recently become popular in the field of audio classification and we have added different layers in this model to make sure that the accuracy rate obtained is better. First of all it is a sequential model in which we have stacked two LSTM layers both of which have 128 number of hidden units and the first layer has the input shape which is (20,5). The second layer consists of dropout to reduce overfitting and both the layers have a return sequence parameter. After the two stacked LSTM layers we have used two Time Distributed Dense layers which have 256 and 512 nodes respectively and both have ReLu as activation function. In the second last layer we have used a flatten layer followed by the last output dense layer which uses softmax as activation function.

For the analysis of the audio data we have used LibROSA[16]. It is a very popular python package that uses soundfile and audioread to load audio files and uses some parameters across all the functions which are
1. Sampling rate = 22050
2. hop_length = 512, which identifies the number of samples between frames.
3. n_fft = 2048, which denotes the number of frames in STFT-like analyses.

1.2 Research Objective

We have set all our purposes for this research regarding the classification of an audio signal using both our models. The audio signal given as input can be within or outside the UrbanSound8K dataset, with which we have trained and tested both our models. Even if there is noise added in the background, our model should be able to identify the audio signal. Moreover, another objective is to show a comparison of the accuracy between the two models used, which are LSTM and CNN, in classifying an audio signal. Furthermore, how different features show differences in accuracy level when used in the two models, have also been investigated through our research. For this purpose we have also stacked different features together. MFCC, Mel-spectrogram, Tonnetz, spectral contrast and chromagram has been stacked in the first form of feature stacking, while in the second form we have stacked MFCC, Mel-spectrogram, chroma_cqt and chroma_stft together. All the features have been used separately as well as in the two forms of stackings mentioned, on both the models to check how does their accuracy fair with respect to the models.

1.3 Research Problems

In our thesis we are working with a form of image which is a spectrogram, and it was one of the preliminary steps for us that before feeding an audio signal as an input to the model, we had to convert it to a spectrogram. But this is different from image classification as it requires a different form of processing of the audio signals and it required a lot of study from our side. Furthermore, as mentioned in the background of this paper, there are different varieties of features that are extracted from each audio sample and each feature has different characteristics which makes it work in a different way on audio. Moreover, for audio classification there are a smaller number of datasets over the Internet. To be more specific, there are 3 datasets which are UrbanSound8K, ESC50 and ESC10, where ESC10 is a subset of its larger dataset ESC50. So this lack of data created a problem in our course of research. In addition to this, manual extraction of the features like MFCC, chroma features, mel-spectrogram, tonnetz, spectral contrast from the audio signal was a difficult task for us to carry out and also time consuming as well. That is why we took on the approach of extracting features from an audio file through LibROSA[16].

1.4 Contribution and Impact

This research has been aimed towards establishing the fact that between the two models, which are LSTM and CNN, the LSTM model with the MFCC feature can show better accuracy in classifying an audio signal compared to CNN. We have altered the dataset, using both augmented and original datasets, on both the models to make sure that the comparison is done on a basis of fair ground. In addition to this both the models can identify any given audio signal accurately, given that the audio signal has some noise added to it.

In our research we have also tried to contribute in bringing up the effect of different features on the two models in terms of classification accuracy. We have used different forms of stacking for this purpose to make sure that the contribution is impactful. Using seven features separately, which are chromagram, mel spectrogram, spectral contrast, tonnetz, MFCC, Chroma_CENS and Chroma_cqt, we have shown that different classifiers due to their difference in characteristics, show different accuracy for each of the models. On the other hand, stacking MFCC, Mel-spectrogram, Tonnetz, spectral contrast and chromagram in the first form of feature stacking, while in the second form we have stacked MFCC, Mel-spectrogram, chroma_cqt and chroma_stft together, we have tried to bring up the effect on accuracy due to stacking features as well.

Although there have been many researches conducted on the basis of audio classification, most of them have been aimed towards finding out the classification accuracy of a single model. And in researches where comparisons have been shown, it does not fully depict the fact that LSTM is better in classifying audio signals in comparison to the CNN. The features that we have used separately as well as stacked have not been done in the same way as ours, according to our knowledge.

To sum up, our research is a promising approach towards putting forth the fact that LSTM, with the mostly commonly used and effective feature MFCC, shows better results than CNN in audio classification. With further experiments and analysis the research can be carried out for unsupervised learning as well.

1.5 Scopes and Limitations

As mentioned earlier, our research has been conducted using the UrbanSound8k and although it is a standard dataset to be used for audio classification, there are certain limitations to this dataset which are imbalance class distribution and varying sampling rates. Apart from these the number of classes in this dataset is only 10 and we had the scope to work with the ESC50 dataset, which has 50 classes each containing 40 audio clips of 5 seconds in length and the same sampling points as UrbanSound8k.Due to our time constraints we had to train and test our model based on the UrbanSound8k.The UrbanSound8K has 10 classes, consisting of 8732 audio files[15].So, the scope to work with newly created audio dataset apart from these is also possible. Moreover, we have taken on the supervised learning approach for our both models and scopes to train and test the model via unsupervised learning is also plausible. For our thesis research, the resources that we got or had in hand were not the most efficient ones and at times it was very frustrating for us because we had to use our resources at home, which slowed down our progress. If provided the correct resources at the proper time, we could have overcome some of the limitations and the scopes of our research would have increased. Extraction of the spectrograms along with the pre processing of the audio signal took away a lot of time in our research which is due to our limitation of the knowledge about working mechanisms of the signals. So, we had to dedicate our time for the study purpose more than we hoped for.

1.6 Documentation Outline

Our research paper is structured in the following way. Section 2 of our research paper contains the literature review and discusses previous related works in the field of sound classification. After this in section 3, discussion regarding the data processing has been done in details, where information regarding audio exploratory, dataset that we have used, feature extraction and process of data augmentation is given. In section 4 of the paper, we discuss the experimental setup. Here, we go in details about the layers that we have used in both CNN and LSTM models along with representation of the model summary for each and lastly model information about model compilation has been provided. We provide discussions regarding the result and analysis in section 5, where we show comparison between the two models and also how different features affect the classification accuracy. Lastly, in section 6 the conclusion is provided which contains information regarding the challenges we faced and also the contributions and future works that can be done regarding our research.

Chapter 2

Literature Review and Related Work

The potential of CNN to learn various spectro-temporal patterns has made them perfectly suited for the classification of environmental sound. CNN were developed in the 1980s but recently introduced different object classification activities. In 2012 the work of Krizhevsky et al [8] has been able to create a turning point for specific visual acceptance of CNN. Since then it has been possible to significantly improve CNN in various pattern recognition tasks by replacing manually designed techniques. Different kinds of CNN models have been used in sound classification and categorization tasks in recent years as they have been able to exceed the standard techniques in different categorization tasks. Once adding a convolutional layer to some data, these layers extract features from local data patches and are often accompanied by a pooling method to pool features over neighboring layers. Models with convolutional layers can surpass models with extract features from raw datasets and show effectiveness in achieving state-of-the-art results.

K.J Piczak [17] proposed a deep model which consists of two convolutional layers along with max-pooling layers and two fully connected layers which is trained on a low level representation of audio data. In order to be consistent with other state-of-the-art, the model exceeds simple implementations based on MFCC which can classify 5.6% more accurately than convolutional methods. The model was tested in a cross validation scheme of 5 folds (ESC-10 and ESC-50) and 10 fold (UrbanSound8K) with a single regime fold used as a cross validation method. In all the cases, the models which are based on a neural network performed better compared to the respective implementations using manual engineered features, particularly when classifying events in different categories of the ESC-50 datasets (baseline accuracy: 44%, best network: 64.5%).

J. Salamon and J.P Bello [24] proposed a deep CNN architecture for environmental sound classification consisting of three convolutional layers interleaved with 2 max-pooling operations and followed by 2 fully connected layers. The also proposed to use the audio data augmentation techniques such as time stretching, adding noise, Dynamic Range Compression, pitch shifting to resolve the data scarcity problem and to explore the effect of various improvements on CNN architecture performance throughout with the UrbanSound8k dataset. Data augmentation is considered as a useful technique by enhancing the amount of training datasets without acquiring new data in case of building CNN model. The concept of data augmentation relies on duplicating the existing datasets with variation in order to give the CNN model to learn from more samples so that it can yield an improved accuracy rate. It has also been shown that increment of class-conditional data can further improve performance. As a consequence of the proposed augmentation, this proposed model significantly increases the efficiency by resulting in a mean accuracy of 79%.

J. Sharma, O. Granmo and M. Goodwin proposed a model consisting of as a DCNN along with several feature channels provided for the Environmental Classification Task (ESC) [40]. Compared to the other models, a deeper CNN (DCNN) made of 2D-separable time and domain convolutions has been employed in this model. There are max pooling layers in the model that separately test the duration and functionality of the domain. To further boost performance, different data augmentation techniques have been used in this model. This model is innovative in that it utilizes a variety of five feature channels. This proposed system has been able in achieving state-of-the art performance in all of the three different benchmark environment datasets used i.e. ESC-10 (95.75%) and ESC-50 (90.48%), UrbanSound8K (97.35%).

I. Lezhenin and N. Bogach proposed a model based on LSTM for urban sound classification because of its efficiency in time dependency learning [36]. LSTMs are RNNs that is used to map the sequence of input to the output through using the background information over long intervals. The model is trained over the magnitude of the mel-spectrogram derived from UrbanSound8k dataset. The proposed model is tested through 5 cross validation in contrast to the standard CNN. In this proposed, both of the models provide almost similar result while the accuracy of CNN model is 81.67% and the accuracy of the proposed LSTM model is 84.25%.

T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. proposed a model in which they have used the complementarity of CNN, LSTM and DNN through integrating them into one single architecture [18]. The complimentary modeling functions of CNNs, LSTMs and DNNs include CNN which are well suited for reducing variability in frequencies, LSTMs which are well known for its temporal modelers and DNNs which are suitable for mapping characteristics to a more separate space. The proposed model in this paper which is referred to as CLDNN has been explored to investigate a number of broad ranges of vocabulary activities from 200 to 2,000 hours. In comparison with LSTM which is the best out of three models, the CLDNN has been able to provide a comparatively 4-6% boost in WER.

An approach on ESC classification system from sub-spectrum segmentation was proposed in [38]. The effectiveness of features derived from the ambient sounds is heavily dependent upon the ESC output. In this paper, CRNN and score level fusion have been implemented jointly to improve the accuracy of classification. Comprehensive truncation schemes are tested to determine the optimum number and corresponding sub-spectrogram band set. This proposed model could accomplish 81.9% correct classification on the ESC-50 data set offering an increase of 9.1% improvement over conventional baseline approaches.

Another method of training a raw audio signal filter bank has been proposed in a very groundbreaking and successful unsupervised approach in [7]. An unsupervised generation system, ConvRBM was trained to generate raw audio waves. The model proposes that the mid-frequency filters look like four bases while gamma tones resemble in the low-frequency range. A CNN is used as a classifier together with ConvRBM filter bench and score-level fusion with Mel filter bank energies. This model has been able to achieve 86.5% on the ESC 50 dataset.

An approach for classifying environmental sounds which stands on Multi-temporal Convolutional Network in combination with multi-level features has been proposed in [33]. In order for this network architecture to be able to achieve multi-temporal resolution functionality, raw waveforms are used and a variety of independent CNNs are used in different layers with different filter sizes and stages. This proposed model has been conducted on two data sets respectively: ESC-50 and DCASE 2017. (ReLUs has been used to introduce nonlinear activation features in this proposed model. This proposed multi-temporal network with multi-level features has been able to achieve an average improvement of 3.0% and 2.0% on ESC-50 and DCASE 2017 respectively in contrast to single-temporal models.

Y. Tokozume and T. Harada proposed a novel end-to-end method which is based on the detection and classification of raw sound signals by CNN [26]. On all three benchmark baselines, their proposed model called EnvNet has achieved competitive performance. The authors employed a system called Before Class (BC) learning in the second version of EnvNet which is EnvNetv2. Two audio signals from various classes are randomly combined in the BC learning process. Then the CNN model provides feedback and is conditioned to generate the mixing ratio. The accuracy on the ESC-50 has been 5.1% higher than the accuracy obtained with the use of static log mel-CNN on their proposed model.

Such outstanding research work discussed above has helped us by giving plenty of insights on different environmental datasets and achieving high efficiency in environmental sound classification.

Chapter 3

Data Processing

3.1 Dataset

There are some of the datasets which are being previously used by researchers for both supervised and unsupervised learning. These datasets contain short clips of environmental sounds which are extracted from Freesound [2], a field recordings repository containing more than 160,000 audio clips. Some of the known datasets are UrbanSound8k, ESC-10, ESC-50 and ESC-US. All of these datasets contains minimum background noises and are manually annotated datasets except the dataset of ESC-US which is unlabelled and does not contain any metadata therefore it is suitable for procedures including unsupervised learning such as clustering,anomaly detection etc.The ESC-50 dataset holds 2000 audio recordings which divided into 50 classes and are grouped into 5 major categories. Whereas the dataset of ESC-10 is a subdivision of the larger dataset of ESC-50. Both of these dataset contain audio clips which are of approximately 5 seconds. Since these datasets of ESC are not large enough we have decided to work with UrbanSound8k since from our preliminary knowledge we were confident that our Deep learning models will give better results with large data.

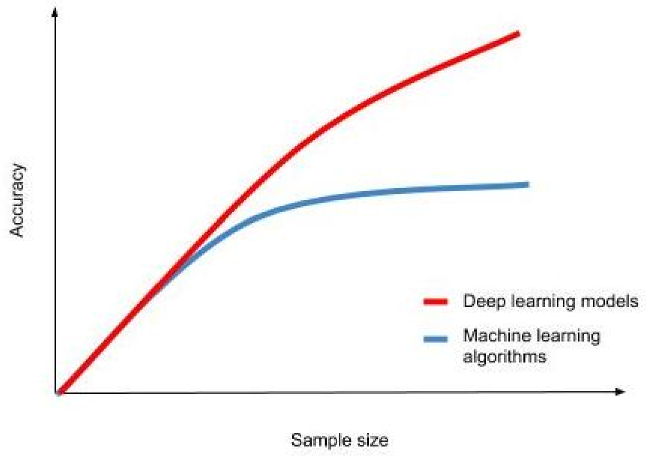


Figure 3.1: Comparison of performance of Deep and ML models w.r.t sample size

One of the main challenges of urban sound research persists in the categorization of audio data into specific classes as there is no urban taxonomy which would help in the labelling and creation of different datasets. In [15], an urban sound taxonomy is built which satisfies 3 rudimentary demands which include satisfying previously proposed taxonomies, detailing low-level sounds and consist of sounds which contribute to noise pollution in urban areas.

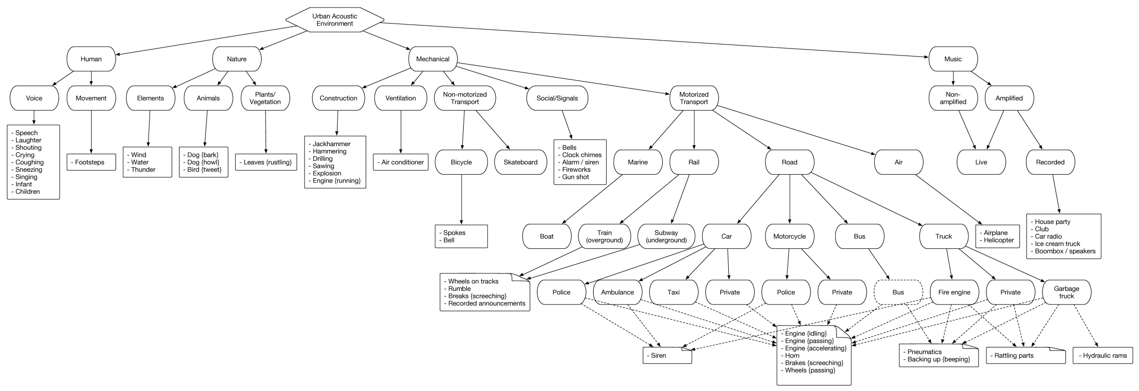


Figure 3.2: Urban Sound Taxonomy [15]

In addition, the UrbanSound8K dataset is built on the basis of the following taxonomy [15] which has duration of 27 hours. The dataset contains 8732 labeled slices of audio that have varying audio length and sampling rate. The 10 classes are air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, police sirens and street music. Most of the classes have approximately 1000 clips but the car horn and gun shot have less slices of 429 and 374 respectively. Hence, some of baseline machine learning algorithms carried out on the dataset in [15] reveal less accuracy for car horn and gun shot. However, the number of classes are less compared to other datasets as wav files were labelled manually. The csv file of UrbanSound8K contains 8 columns which are slice file name, fsID, start, end, salience, fold, classID and class names.

- Slice file name - Determines the name of the wav files in the folders.
- FsID - Each wav file is given a unique ID for identification
- Start and end - Labels the start and end time of every occurrence in the 27 hours of audio.
- Salience - Indicating whether the sound was in the background or foreground of the recordings.
- Fold - Specifies the folder to look into for each wav file. In each folder attributes are selected based on correlation to circumvent overfitting of training data.
- ClassID and class name - Each class is given a name along with an ID from 1-10.

In addition, the 10 folders are made with random assignment of the slices to avoid unnaturally high classification accuracies. In [15] a subsection of the larger dataset is created named as UrbanSound8K subset which can be also be used for several researches on sound source determination. The subset also contains 10 folds with comparatively less number of wav files in them. Finally,UrbanSound8k was selected to run our model as the dataset was both informative and large.

3.2 Audio Exploratory

The audio samples are in the .wav format and these continuous waves of sounds are digitised. They are converted into an array of digital values by sampling them at discrete interim. Such waves are one directional and can be used to represent a specific frequency or amplitude at given time instances. Therefore a specific sampling rate is used and these sample values can be assembled if the audio is needed to reconstruct. Such methods were used by Sony from 1979 in [5]. The choice of the sampling rate depends on the Nyquist-Shannon theorem [9] which recommends to take the sample rate double with respect to the maximum frequency that we need to produce. So the complete wave can only be recorded into an array without any noise or attenuation when we follow the Nyquist-Shannon theorem. However we chose LibROSA in [16] which is a python package audio and music processing by Brian McFee [37]. The LibROSA library has predefined functions which are commonly used throughout the audio processing.

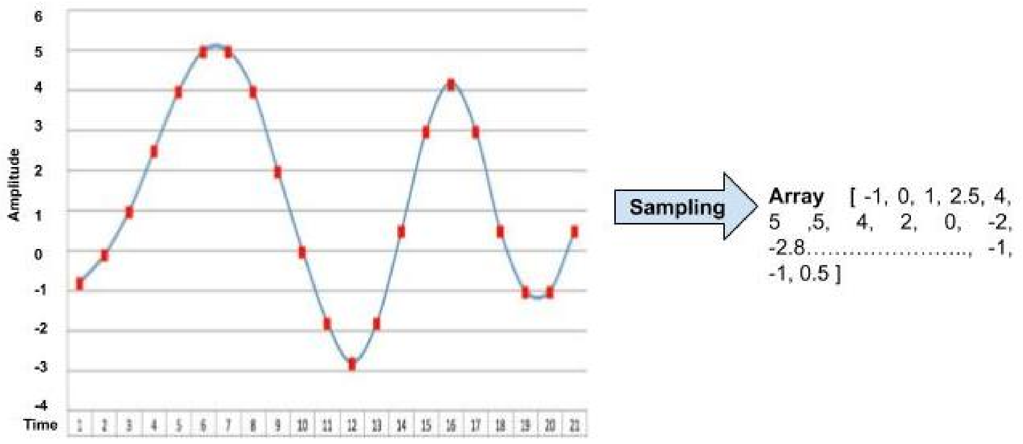


Figure 3.3: Resulting Array after sampling (digital values are represented in red).

The above picture 3.3 shows the resulting one dimensional numpy array after sampling where the bit depth of the sound defines how comprehensive the sample will be. But LibROSA uses a default sample rate of 22050 Hz. Therefore when we load the audio using *librosa.load()* function it combines the channel on both sides to make it a mono signal. Furthermore the default frame and hop lengths are 2048 and 512 samples respectively[16]. However the sample rate of 22050 Hz assists in keeping the size of the array small by decreasing the quality of audio but greatly reducing the training time. On the other hand, the load time is reduced by passing a parameter to *librosa.load()* function in [16] which is 'res_type'. It represents the resample type where we use 'kaiser_fast' to drastically bring down the load time. LibROSA also normalise the data so that the values represented in the array are between -1 and 1. After loading the audio and representing it in an array of values , various types of features are extracted from it.

3.3 Features extracted

3.3.1 MFCC

Mel frequency cepstral coefficients which are compact representations of the spectrum are typically used to automatically identify speech and it is also used as a primary feature in many research areas that include audio signals. In the 1980s, Davis and Mermelstein launched them and have been a cutting-edge feature since then [2]. There are five fundamental steps used to extract mfcc features from a one dimensional signal[14]. These are framing and blocking, windowing, fft, mel-scale, inverse-fft shown in figure 3.4

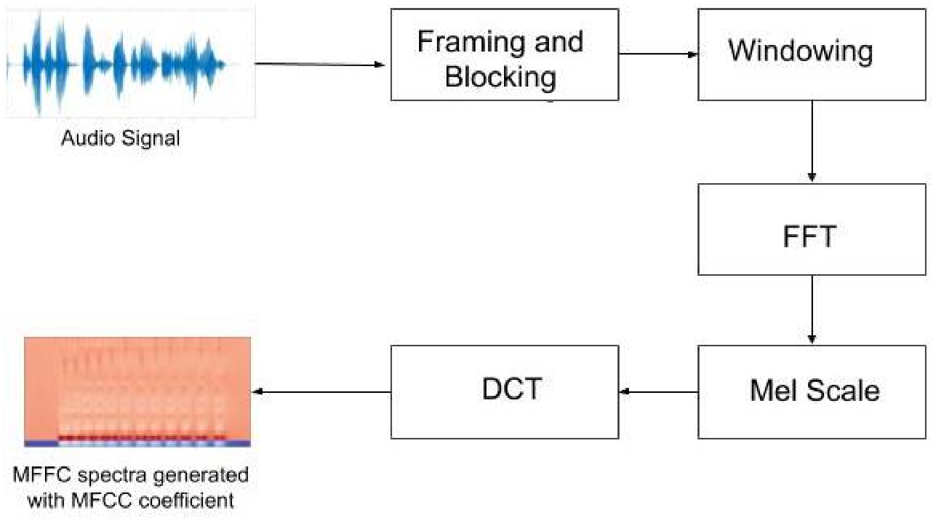


Figure 3.4: Steps to extract MFCCs from an audio signal

3.3.1.1 Framing and blocking

During this stage the continuous one dimensional signal is disrupted into N frames while the corresponding frames are separated by M samples where $M < N$ with N-samples overlapping the neighboring frames. According to many research works the standard value picked for N and M is 256 and 100 respectively [12] and the reason behind this is dividing the signal in such a way so that we have enough samples so that information is not lost. This process of breaking the given signal into frames continues until the entire sound clip is broken down into small frames.

3.3.1.2 Windowing

In this step the disruption at the starting and ending of the frame is minimized.If a window is defined by:

$$W(n), 0 \leq n \leq N-1 \tag{3.1}$$

In the above equation 3.1, Nm represents the quantity of samples within every frame. Therefore the output for the window process is represented by:

$$Y(n) = X(n)W(n), 0 \leq n \leq N-1 \tag{3.2}$$

Here Y(n) is basically the output signal after the multiplication of the input signal, $X(n)$, and window represented by $W(n)$. Furthermore there are various types of windows which are used to reduce distortion such as rectangular window, flat window and hamming window. But to extractMFCCs, hamming window is used which is usually represented by the equation given below:

$$W(n) = 0.54–0.46Cos(2\pi n/(n-1)), 0 \leq n \leq (N-1) \tag{3.3}$$

3.3.1.3 FFT

Using fft the spatial domain of each frame is converted to frequency domain. An NN-point FFT is done on each frame to find out the frequency spectrum. This process is also called STFT. In this NN-FFT, NN is 256 or 512 [42].

3.3.1.4 Mel-Scale

In this phase the above generated spectrums are mapped on the Mel scale in the aid of a triangular window which is commonly known as the triangular filter bank. This is basically used to know the current energy approximations at each spot.Mel scale thus allows spacing the specified filter and determining how much wider it will be as such filters get wider as the frequency increases. The equation used to convert the frequency to Mel from [10] is given below:

$$mf = 2595 * \log_{10}((f/700) + 1)) \tag{3.4}$$

Once the energy at each spot is estimated, the log of such energies is known as Mel spectrum which is later used to calculate the coefficients using DCT.

3.3.1.5 Discrete cosine transform

The resulting spectrum from the previous process has coefficient values which are highly correlated therefore DCT is applied to decorrelate these coefficients. The output after applying DCT is known as MFCC shown in fig 3.5.

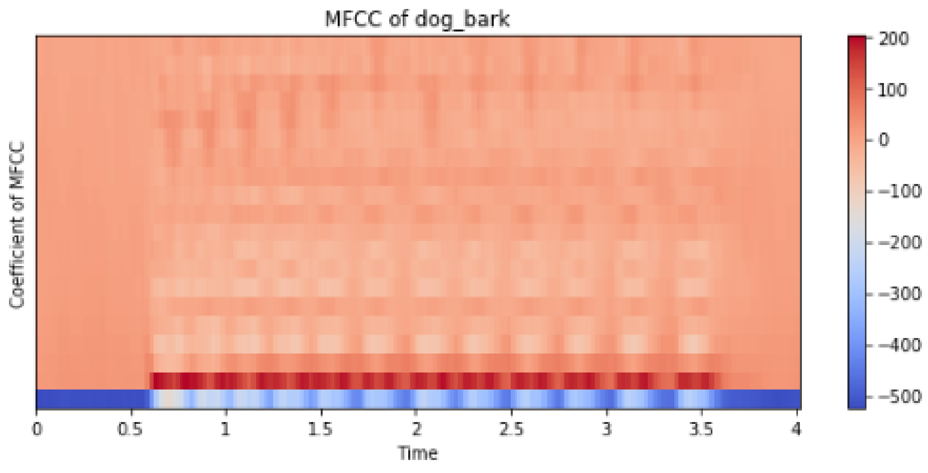


Figure 3.5: Mel Frequency Cepstral Coefficient Feature

3.3.2 Mel Spectrogram

A Mel Spectrogram is the collaboration of Mel scale and a spectrogram. Where mel scale represents the non-linear transformation of the frequency scale. This is done using overlapping triangular filters as discussed above. The steps required to generate Mel specs are somewhat similar to the steps required for MFCC coefficient [38]. Figure 3.6 shows the steps required to extract the Mel Spectrogram from a one directional audio signal in the fingure 3.5.

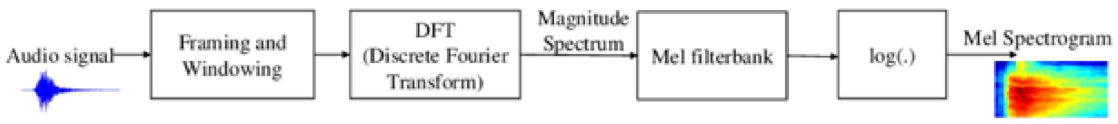


Figure 3.6: Steps required to extract Mel spectrogram.

The audio signal is first broken down into smaller frames and a hamming window is applied on each frame. Then DFT is applied to switch from time domain to the frequency domain.In the step of Mel filterbank, triangular filters are applied to extract the frequency bands. *Log*() is used at the final stage to generate the spectrum which consists of the spectral envelope and spectral details, that helps in the generation of the mel spectrogram [52] shown in the figure 3.7.

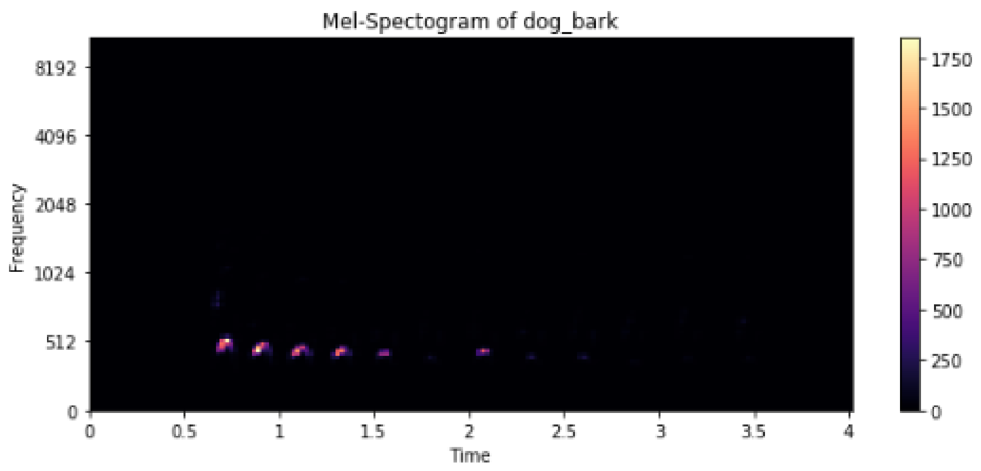


Figure 3.7: Mel spectrogram.

3.3.3 Chromagram

The Chromagram in [39] is another feature technique for extracting music audio signals. Chroma based characteristics are particularly useful for audio pitch analysis.The chroma representation tells us the intensity of each of the 12 distinct musical chroma of the octave at each time frame. They can be employed in the differentiation of the pitch class profiles between audio signals. We have used chroma cens, chroma STFT, chroma cqt as features for our model.

3.3.3.1 Chroma STFT

Chroma STFT in figure 3.8 used STFT to compute chroma features. Chromagram STFT represents information about classification of pitch and signal structure. It represents the spike with high values (as evident from color bar net to the graph) in low values (dark regions).

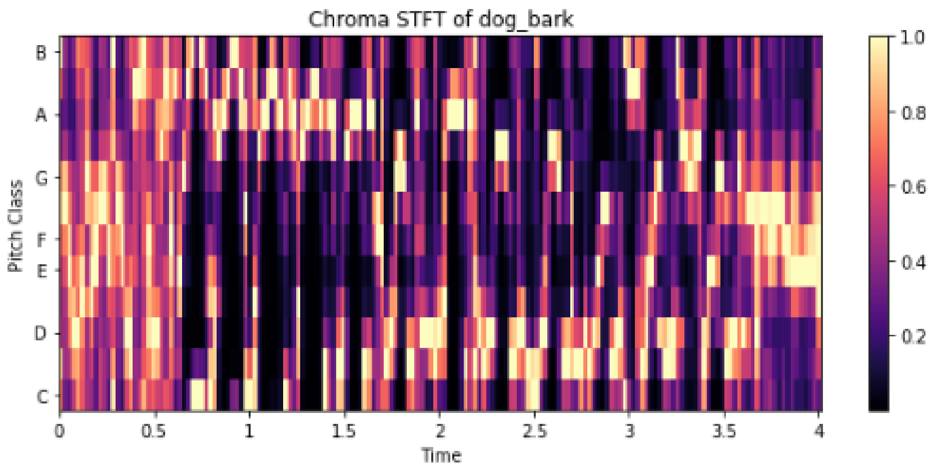


Figure 3.8: Chroma STFT

3.3.3.2 Chroma cqt

Another approach of chromagram is chroma constant-q transform in figure 3.9 which adjusts the STFT to logarithmically create space between the frequency bins. In addition, center frequency is modified with window function so that the width of each bin increases. It contains DFT, which uses a constant buffer size throughout all frequencies. This typically leads to a pretty consistent, fully continuous transform. However, the constant bin size for all frequencies leads to some problems when you map frequency on a logarithmic scale [3]. CQT seeks to solve this problem by increasing the buffer size for lower frequencies, and alleviate some of the computational strain caused by this by reducing the buffer size used for high frequencies.Although there are few similarities between the CQT and the Fourier transform which includes both being filter banks but the difference lies in the Fourier Transform has some core frequencies which are geometrically placed[1].

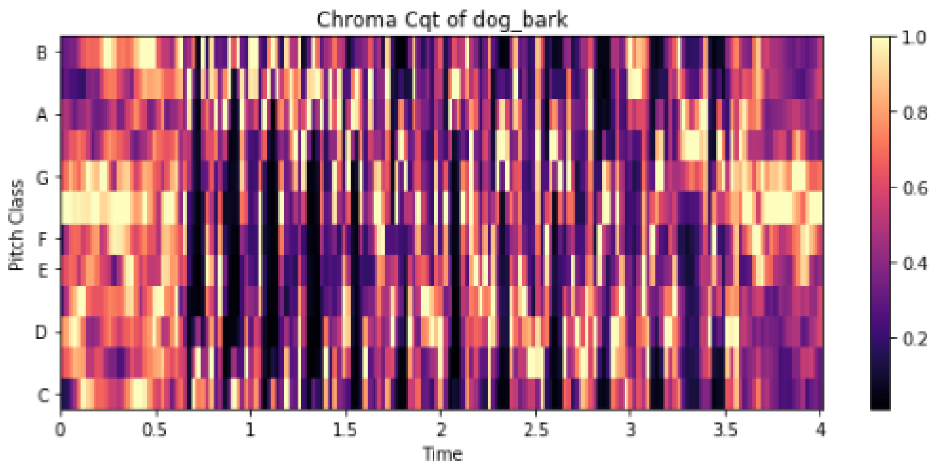


Figure 3.9: Chroma cqt

3.3.3.3 Chroma cens

With a further degree of complexity by looking at short-term statistics on chroma band energy distributions. Thus we can obtain CENS shown in figure 3.10.CENS is closely associated with the short-term harmonic quality of the fundamental audio signals, absorbing varying features, including rhythm, intonation, musicality, note classes and duration micro deviation. In addition, CENS functions can be interpreted effectively due to their low spatial resolution. The Chroma characteristics are centered on just the 12 pitch sounding properties that are familiar in the western music notation in which each Chroma variable demonstrates how intensity of an audio is dispersed through the dozen chroma bands around the transmitted signal frame [6].

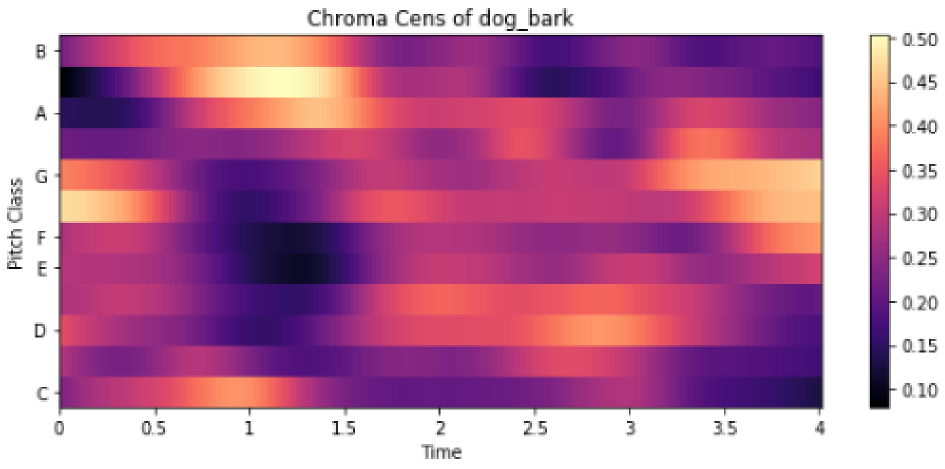


Figure 3.10: Chroma cens

3.3.4 Spectral Constrast

Another form of extraction technique that has been used in our proposed model is Spectral Contrast, which has been represented in the figure 3.11. Spectral Contrast takes into account the spectral peak, the spectral valley and the disparity between each sub band frequency. Spectral Contrast features show the spectral characteristics of an audio clip which represents the spectral distribution instead of average spectral envelope. Spectral Contrast uses octave base filters while MFCC uses Mel-Scale filters. Moreover, Spectral Contrast feature uses a K-L transform. In order to eliminate relativity, they are identical.

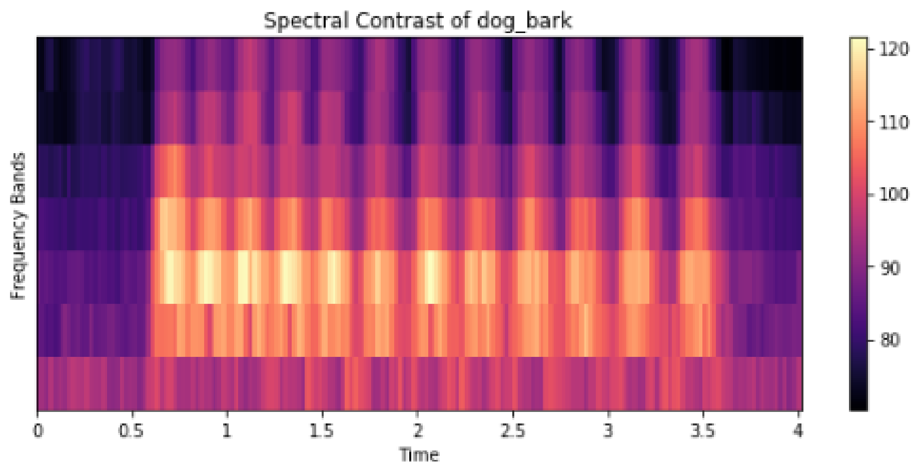


Figure 3.11: Spectral contrast

3.3.5 Tonnetz

The last audio extraction technique, shown in figure 12 we have worked with Tonnetz. Tonnetz computes the tonal centroid features by detecting the harmonic changes in musical audio clips. It is a well-known planar representation of pitch relations in an audio clip which is considered as an infinite planar. To reduce the efforts of transient frames in Tonnetz, the sequence of tonal centroid vectors are convolved with various kinds of Gaussian smoothing window.

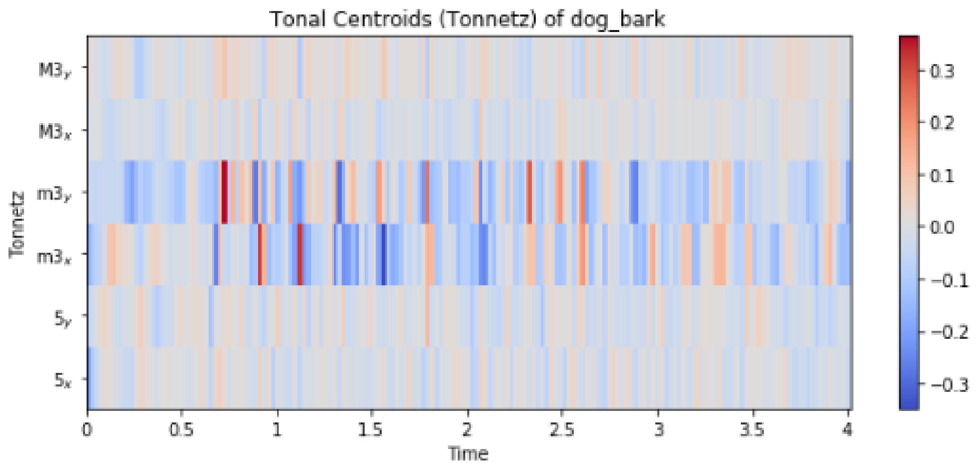


Figure 3.12: Tonnetz.

3.4 Data Augmentation

Neural network models such as CNN and LSTM rely particularly on the availability of large quantities of training data to acquire a nonlinear function from input to output that generates well and provides high classification accuracy for unseen information. In neural networks, there are a number of ways to deal with data scarcity issues. Data augmentation is a useful technique for building the CNN that can enhance the amount of training datasets without acquiring new data. The concept of data augmentation is very straightforward; duplicate the existing datasets with variation in order to give the models to learn from more samples so that it can yield an improved accuracy rate [29]. To evaluate our proposed model architecture for data augmentation we have been working with UrbanSound8k dataset. There are some augmentation techniques such as pitch shift, time streching which are effective in terms of data augmentation [24]. So we have adopted these techniques to augment the UrbanSound8K dataset of 8732 audio files. After augmentation we achieved a total of 78588 audio clips which were labelled accordingly. Since we have seen a number of data increases the performance of the deep learning models, we took fairly enough data to achieve a state of art performance using our model which is shown in the result part. As detailed below, we have experienced 2 different audio data augmentation techniques in which each deformation is added to the audio signal immediately before it is transformed into the input representation. The deformation augmentation sets are mentioned below:

3.4.1 Pitch Shift

The factors of -2, +2 are used to raise and lower the pitch(in semitones) of a audio sample in the dataset. Therefore using this technique we can create two novel samples from a single audio. Likewise we could create 17464 more samples using pitch shift.Figure 3.13 represents how the signal vary when the pitch is changed(the time duration remains same but the frequency of the audio changes).In comparison with the original signal(*blue*), the *orange* audio signal have less frequency since the pitch is reduced by a factor of 2. Whereas the *green* signal have higher frequency due to pitch increment by a factor of 2.

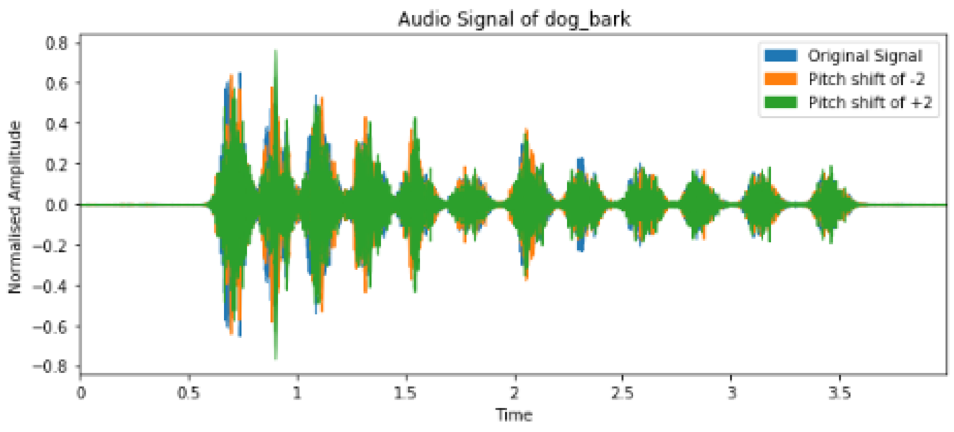


Figure 3.13: Audio Signal of Dog_bark after Pitch Shifting

3.4.2 Time Stretch

In this technique we slow down or speed up the sound clips with a rate of 0.9 and 1.1. Here the rate of 0.9 slows down the audio therefore the spikes of the *orange* signal in the figure 3.14 occur after few seconds compared to the original signal *blue*. On the other hand, the *green* audio signal happens to occur before the *blue* sample audio.In this way, we could generate more 17464 new audio clips for our augmented dataset.

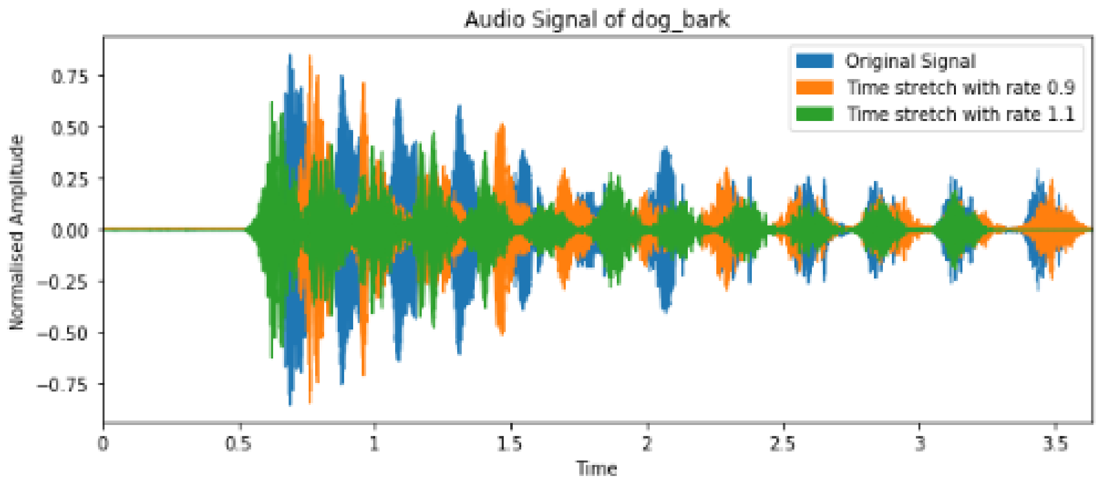


Figure 3.14: Audio Signal of Dog_bark after time-stretch

3.4.3 Pitch Shift along with Time Stretch

In this augmentation step, a sound clip is manipulated using both pitch shift and time stress to generate a total of 34928 novel audio clips.In figure 3.15 we can see that the *orange* and *green* signal's variation with respect to time and normalised amplitude when it is pitch shifted along with time stretch with the factors mentioned above. To illustrate this, a sample of dog_bark is shown below in the figure 3.15

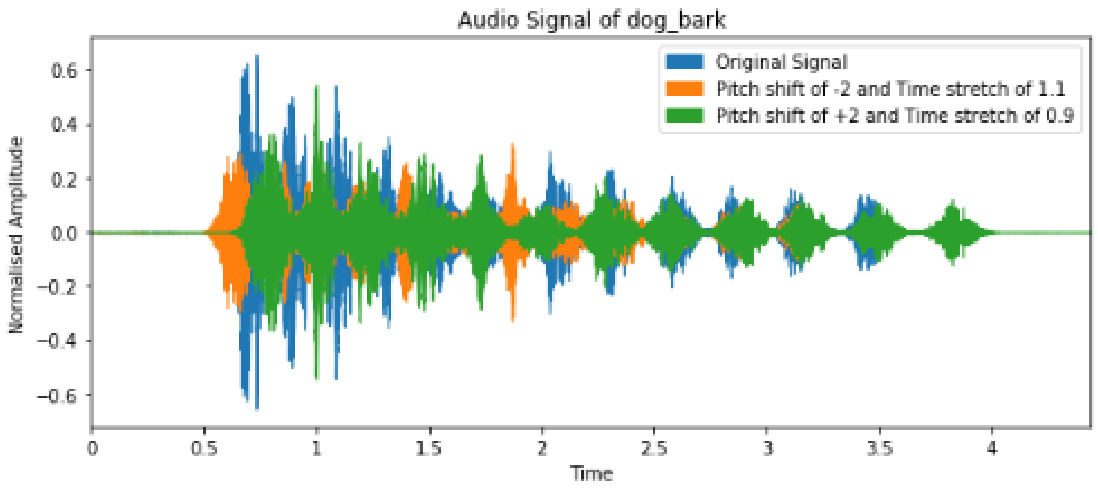


Figure 3.15: Audio Signal of Dog_bark after pitch shift along with time-stretch

Chapter 4

Model Architecture

4.1 CNN Model

In a typical CNN, there is a series of various kinds of layers which are combined together in the overall architecture. Every input passes through a number of Convolutional layers with filters, pooling layers, fully connected layers and an output layer which justifies the classification with a probabilistic value. The training of a CNN requires different kinds of decisions which needs to be taken in terms of both architectural patterns such as number of convolution and pooling layers, input data format, filter dimension etc. as well as hyper parameters such as learning rate, dropout probability, number of epochs, batch size etc [17]. Usually CNN produces good classifiers and performs well with classification tasks because of its extraction and classification components. Our proposed model is a sequential model consisting of two Conv2D layers followed by three dense layers among which the final layer being the output layer. Our proposed model operates by sliding the window of a filter over the input shape, matrix multiplication and then providing the stored result into a feature map.

4.1.1 Convolutional Layers

Both of the convolutional layers are used for detecting the feature. The first Conv2D layer receiving the input shape consists of 64 filters, a kernel size of 5 and stride set to 1. The node numbers in each layer is indicated by filter parameter. In this case, each layer in our model will increase from 64 to 128 in size. The parameter determines the kernel window's size which in our case is 5 that results in a 5x5 filter matrix. Stride tracks the operation of the filter around the input range. As stride is set to 1 in the first layer, the filter converges around the input volume by moving one unit at a time.This convolutional layer with 'same padding' operation produces an output of the same height and weight as the input. The activation function we have used for this layer is called ReLU which has got several benefits compared with conventional units such as effective gradient propagation and quicker computation rather than sigmoid units, thus maintaining, in spite of their simplicity, adequate discriminatory characteristics [15]. The first Conv2D layer is followed by MaxPooling2D layer which is used in reducing the dimension of the input shape fed through the convolutional layer.

The second Conv2D layer consists of 128 filters, a kernel or of size 5 and stride set to 1 along with ‘same padding operation’. The activation function that we will be using is ReLU which is same as the first Conv2D layer. The second Conv2D layer is followed by a MaxPooling2D layer and a dropout of 30%. CNN tends to have over-fitting naturally. The one effective way to tackle this is to introduce dropout. Using a dropout value of 30% in our model will remove nodes from each update cycle which in effect to a network can generalize widely and over fit the training data less likely. Thus, the averaging of architecture enforced by the dropout value of 30% will attempt to ensure that the proper classification response is produced by each hidden layer.

4.1.2 Flatten Layer

The flatten layer converts the output of the convolutional layers into a one-dimensional array to be inputted into the next hidden layers.

4.1.3 Dense Layers (Fully Connected Layers)

Two dense layers (Fully connected layers) and an output layer have been used for further processing of the model. Among the next three layers, the first two dense layers will have 256 and 512 nodes correspondingly. Dense layers refer to a linear operation that connects every input fed through the layers to each output by a weight [19]. The non-linear function which is followed in the first two dense layers is ReLU that simply switches all the negative activation’s to zero. This is proven to be effective in increasing the model’s non-linear properties and does not affect the overall network’s respective fields.

4.1.4 Output Layer

The output layer having activation function softmax will consist of 10 nodes in our model that refers to the possible classification numbers [43]. This added constraint allows converging quicker than it would otherwise. The model then predicts the choice with the highest probability.

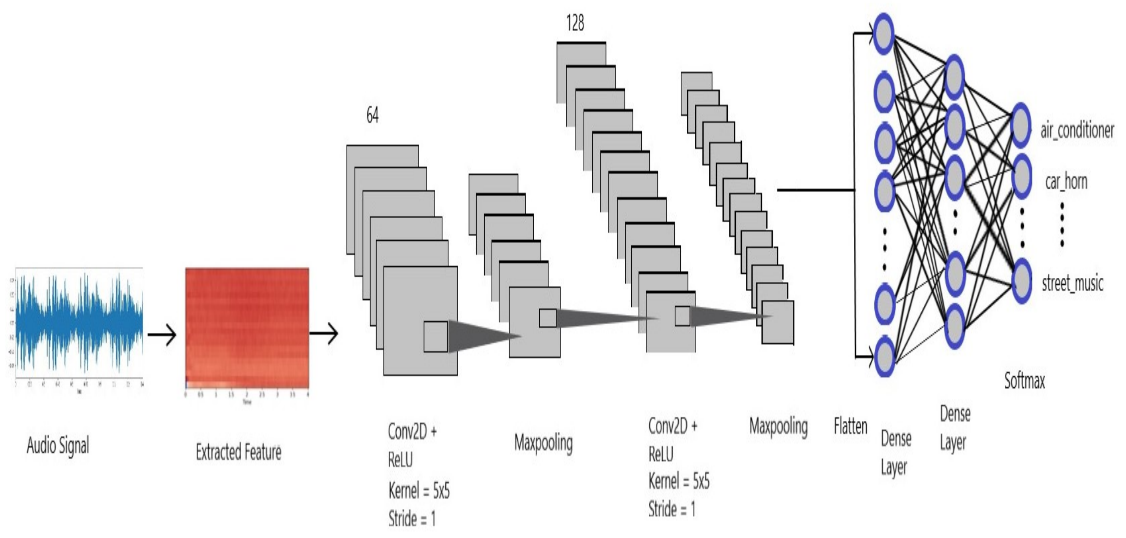
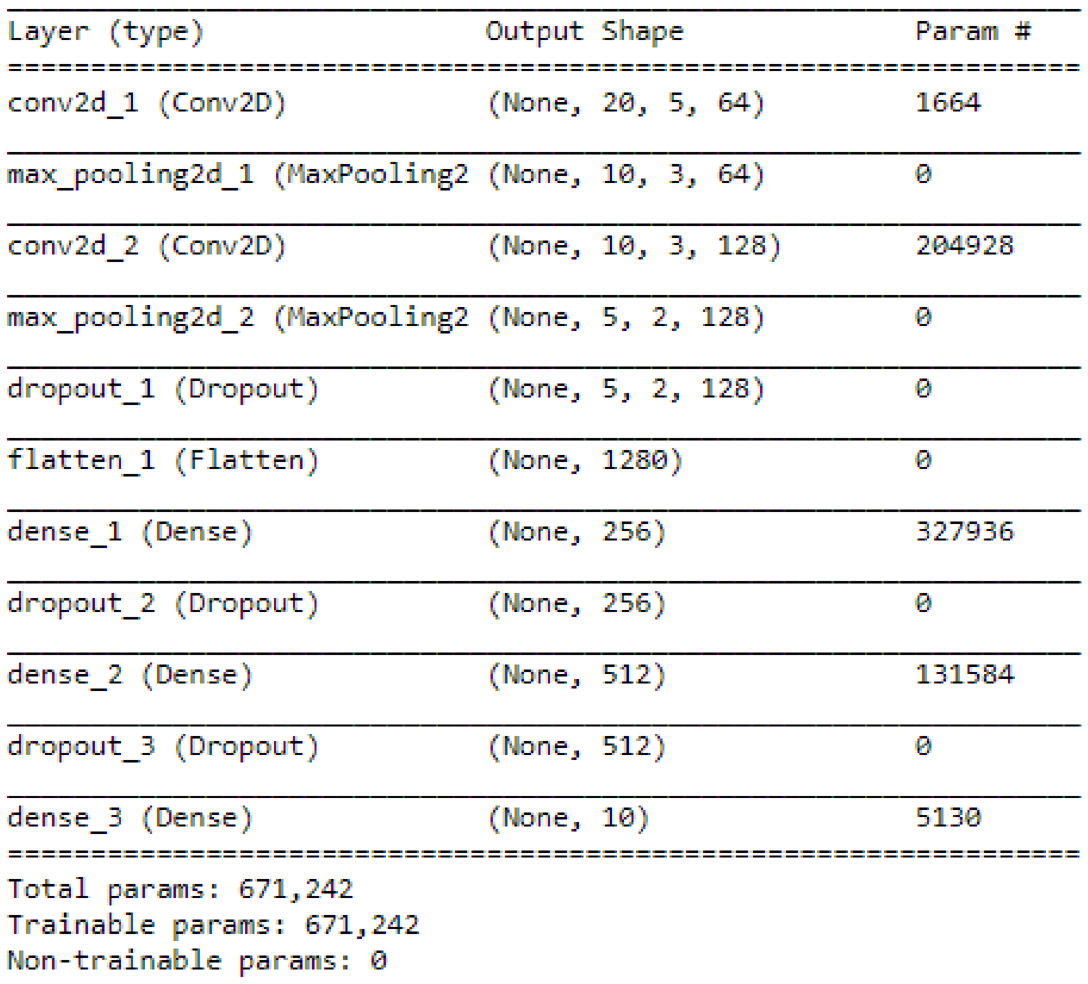


Figure 4.1: Architecture of our Convolutional Neural Network

4.1.5 Model Summary of CNN

The number of the total trainable parameters of the following model depends entirely on the dimension of the input shape and the size of the filters. The calculation of trainable parameters for the different layers are shown below with appropriate formulas:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 20, 5, 64)         1664
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 3, 64)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 3, 128)        204928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 2, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               327936
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 671,242
Trainable params: 671,242
Non-trainable params: 0
```

Figure 4.2: Model Summary Of CNN

Conv2d_1: The equation [30] of trainable parameters for the Convolutional Layers is

$$\text{Total Trainable Parameters} = ((n * m * l) + 1) * k)) \tag{4.1}$$

where,
$n * m$=filter size
l = number of feature map as input
K = number of the filters
As there is no feature map in the first convolutional network, so the no. of feature map as will be avoided here. The additional one is added as a bias term. Here, the width and height of the filter's shape is 5 and the numbers of the filters are 64.

$$\text{Total Trainable Parameter} = ((5 * 5) + 1) * 64) = 1664 \tag{4.2}$$

Maxpooling Layers: As there is no back propagation involved, these layers has got no trainable parameters.

Conv2d_2: In this Convolutional layer, height and weight of the filter's shape is 5 , the layer has 64 feature maps as input and the number of the total filters are 128. So according to the formula:

$$\text{Total Trainable Parameters} = ((5 * 5 * 64) + 1) * 128) = 204928 \tag{4.3}$$

Maxpooling Layers: As there is no back propagation involved, these layers has got no trainable parameters [48].

Flatten Layer: A flatten layer has no trainable parameter of it's own to learn as it is added to increase the overall learning parameters of the rest of the model.

Dense_1 (Fully Connected Layer): Each input has a separate weight linked to the output unit in fully connected layers. The number of the weight is $n * m$ where n denotes the inputs and m denotes the number of outputs. As, there's a bias in each output node additionally so the overall equation [30] for calculating the trainable parameter for the fully connected layers is $(n + 1) * m$. In the first dense layer, $n = 1280$ and $m = 256$.

$$\text{Total Trainable Parameters} = (1280 + 1) * 256 = 327936. \tag{4.4}$$

Dense_2: Here, $n = 256$ and $m = 512$ where n is the input and m is the number of the outputs. So, according to the formula:

$$\text{Total Trainable Parameters} = (256 + 1) * 512 = 131584. \tag{4.5}$$

Output Layer: As it is also a fully connected layer, so it has got $(n + 1) * m$ parameters where n denotes the number of the inputs and m denotes the number of the outputs.so,total trainable

$$\text{Total Trainable Parameters} = (512 + 1) * 10 = 5130. \tag{4.6}$$

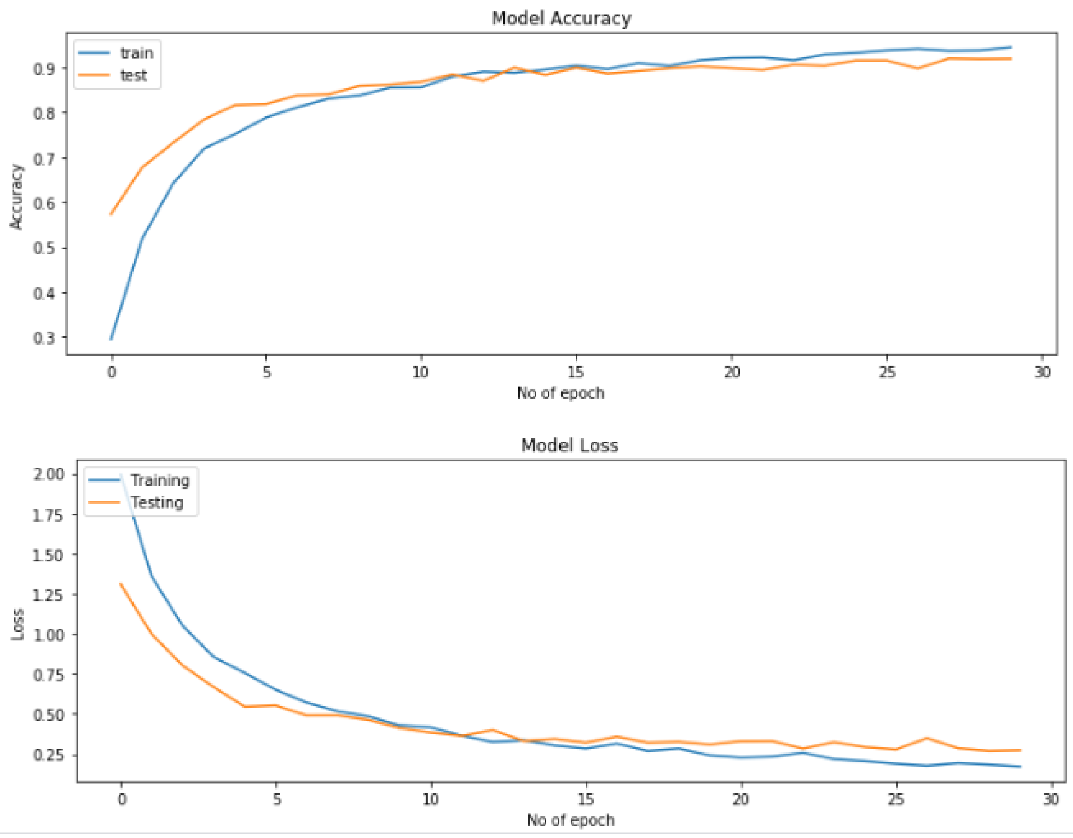


Figure 4.3: Model Accuracy and Loss for MFCC (CNN)

For training our CNN model, we have started with 30 epochs. The figure 4.3 displays the model accuracy and loss of CNN in the Y axis and number of epochs in the X axis. An epoch refers to the period corresponds to the full dataset over one cycle. The validation error (val_loss) of the model keeps improving until a certain stage is reached in the model. We have chosen 30 epochs for our model because the validation error (val_loss) stops improving somewhere around the 30th epoch. The entire dataset is passed forward and backward through the model in one epoch [41]. In the figure 4.3, we can see after the 25th epoch, the validation error (val_loss) has become horizontally linear and is not significantly improving. This is the reason why we have kept the epoch number to 30. We split the dataset into batch size of 50 because an epoch is too big to feed through the model at once. We have kept the batch size to 50 as having a low batch size can improve the generalization of our model. As the batch size is set to 50, it will keep taking 50 samples for training the model and keep doing the same in same sequence until all the samples have been propagated through the model.

4.2 LSTM Model

Audio classification with LSTM has become popular recently and LSTM has shown an impressive accuracy rate in the classification process. The model has different layers that are involved in the process of training as well as testing the model and for the sake of our approach towards the audio classification we have used two LSTM layers, followed by two Time Distributed Layer and a flatten layer and lastly a dense layer shown in the figure 4.4. Furthermore, different activation functions like ReLU and Softmax at different layers have been used. With regards to the model categorical cross entropy as a loss function, the metric accuracy and adam optimizer have been used as well.

Although seen as a part of RNN, LSTM actually overcomes shortcomings of the RNN itself [13]. LSTM model have blocks of memory which can be referred to as a set of subnets. A memory cell which is also known as cell state and three other gates are present in each block. These gates are: forget gate, input gate and output gate.

4.2.1 LSTM layers

The LSTM layer usually works by mapping input sequence X= (x1, x2, x3. . . ..xT) and provides Y= (y1, y2, y3. . . ...yT) which are denoted as output sequence [36]. This is done according to the equation given below [36] where the state of the memory cell is denoted by ct and the gate outputs are denoted by it, ft, ot which are the input, forget and output gates respectively. Tuning of the weights W and biases b are done in the learning phase so that the loss function is minimized.

$$i_t = sig(W_{xi}xt + W_{yi}y_{t-1} + b_i) \tag{4.7}$$

$$f_t = sig(W_{xf}xt + W_{yf}y_{t-1} + b_f) \tag{4.8}$$

$$C_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xo}xt + W_{yo}y_{t-1} + b_c) \tag{4.9}$$

$$o_t = sig(W_{xo}xt + W_{yo}Y_{t-1} + b_o) \tag{4.10}$$

$$y_t = O_t \odot tanh(C_t) \tag{4.11}$$

In our model we are stacking two LSTM layers. A LSTM layer can be defined by the dimension of the hidden states or cells along with the number of layers that are used. Each LSTM layer has hidden cells, as many as the number of time steps, which in our case is 20. Each of the 20 hidden cells contain 128 hidden units, and each hidden unit contains some information from the immediate previous hidden cell. An increase in the number hidden units used in each cell affects the training data causing it to overfit [35]. Furthermore, in both the LSTM layers return sequence has been set to "true" because to stack LSTM layers, because both the LSTM layers require to output a 3D array which will be used by the following Time Distributed Dense layers as input. In the first LSTM layer of our model input shape provided is (20,5), where 20 represents the timesteps which lets a LSTM layer know how many times it should repeat itself once it is applied to input. This means each of the input passes 20 times through each of the hidden cells and the 128 hidden units within it. The second layer of LSTM has a dropout of 0.3 in addition to the return sequence which is set to "true". A dropout of 0.3 reduces the effect of overfitting of the training data. The output of the two LSTM layers which is a 3D output and passed on to the time distributed dense layers for further processing.

4.2.2 Time distributed dense layer

Moving on to the time distributed layer which is a wrapper refers to every temporal portion of an input a dense layer. 3-dimensional input is what this layer expects and usually works with , and the index one dimension should be known as temporal [31] and the recurrent calculations of our model will be performed only in the timesteps dimension of LSTM, which is set to be 20. The size of input in the first layer is 128 and the number of neurons or nodes in the output layer of first time distributed dense layer is 256. For the second Time Distributed layer the size of input is 256 and the output layer contains 512 nodes. In both these layers ReLU have been used as an activation function if any negative input is provided, it will convert it to zero but in case of positive inputs the output remains the same as the input. It leads to better performance for our model as it deals with the vanishing gradient problem and allows our model to learn faster as well as perform in a better way. Furthermore, since our model is required to classify between ten classes, ReLU is the better option as it has the ability to classify between two or more classes. The output of these two layers contain the batch size, timesteps and input size. So the error function should be between the predicted label sequence and the actual label sequence. The 3D output is then passed on to the fifth layer which is the flatten layer.

4.2.3 Flatten Layer

The flatten layer reshapes the tensor to such a shape which is equal to the number of elements present in the tensor, in other words, it makes 1D array of the elements, in this case the 3D output from the time distributed dense layer which it takes in as input. After the flattening is done, we end up with a long vector of input data which is then passed onto the last layer which is the dense layer.

4.2.4 Dense Layer

Since the dense layer requires a 2D input and the flatten layer sends a 1D input vector, it refers to a multiplication of matrix vector assuming a batch size, which makes the input 2D. The values in the matrix are the trainable parameters which get updated during backpropagation [31].

$$u^T.W, W \in R^{nxm} \tag{4.12}$$

The output dense layer feeds all the output that it has fed from the previous flatten layer to all the neurons in the hidden layer which is 10 in our case, as it is the number of classes that we want to categorize our data into. We have used softmax activation function in the output dense layer, which converts its inputs – likely the logits, also known as the outputs of the last layer of our neural network when no activation function is applied yet – into a discrete probability distribution over the target classes. Softmax ensures that the criteria of probability distributions – being that probabilities are non-negative real valued numbers and that the sum of probabilities equals 1 – are satisfied. We have chosen the softmax activation function over other ones, for the last layer of classification because in our model there are 10 classes of audio and each of the input provided to the model can only belong to exactly one of the 10 classes, and thus it will contain 10 nodes.

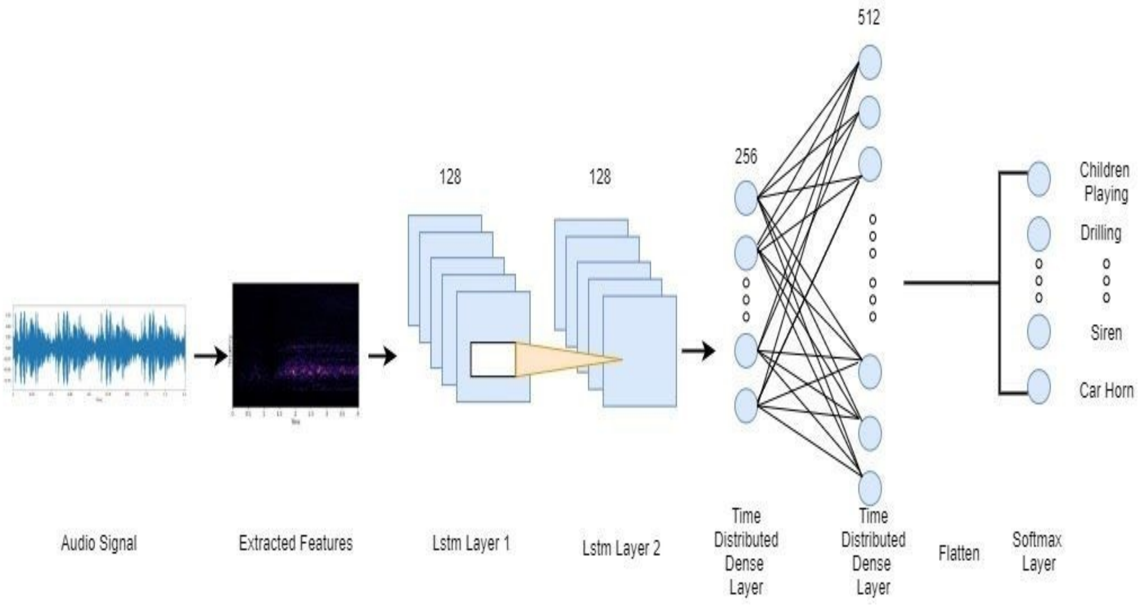
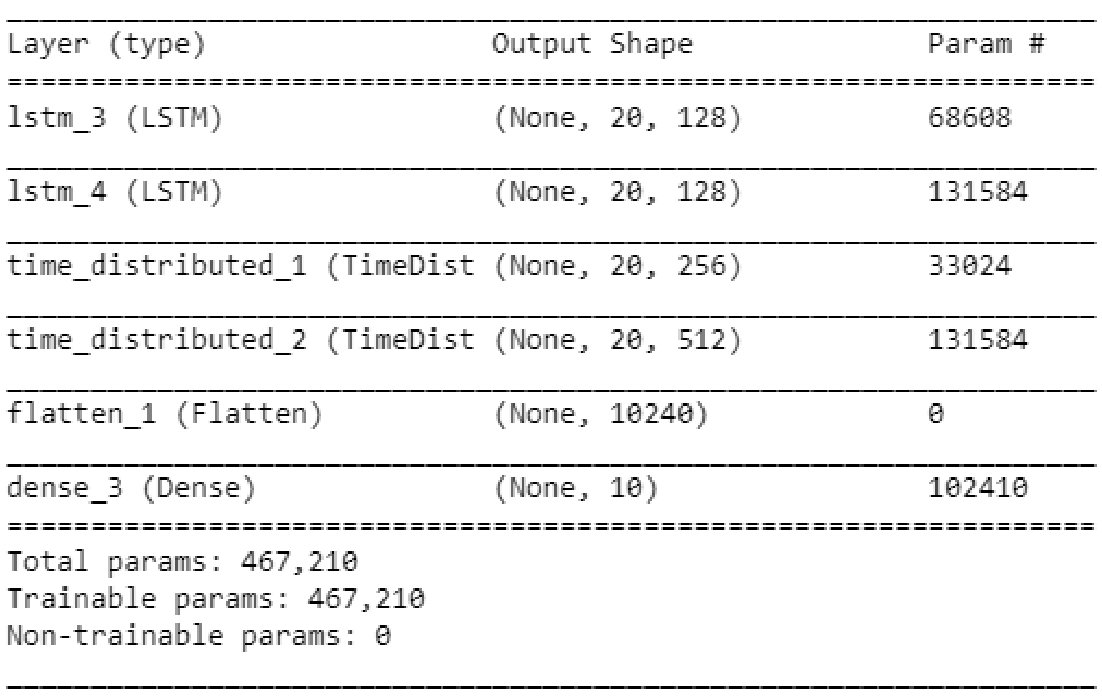


Figure 4.4: Architecure of our Lstm model

4.2.5 Model Summary of LSTM

In the figure 4.5 shown a model summary of lstm layers with Time Distributed Dense layer.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 20, 128)           68608
_________________________________________________________________
lstm_4 (LSTM)                (None, 20, 128)           131584
_________________________________________________________________
time_distributed_1 (TimeDist (None, 20, 256)           33024
_________________________________________________________________
time_distributed_2 (TimeDist (None, 20, 512)           131584
_________________________________________________________________
flatten_1 (Flatten)          (None, 10240)             0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                102410
=================================================================
Total params: 467,210
Trainable params: 467,210
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.5: Model Summary of LSTM.

The trainable parameters calculation for different layers vary according to the input size and the output size as well. Trainable parameter values for each layer are shown along with calculations below:

First LSTM layer: The formula for the calculation of trainable parameter for this layer is shown in equation 4.13.

$$\text{Total Trainable Parameters} = 4 * ((n + 1) * m + m^2) \tag{4.13}$$

where n = size_of_input and m = size_of_output.Now, in this layer the input size is 5 and output size is 128.

$$\text{Total Trainable Parameterss} = 4 * ((5 + 1) * 128 + 128^2) = 68608 \tag{4.14}$$

Second LSTM layer: In the second layer the size_of_input is 128 and size_of_output is also 128.So,total

$$\text{Total Trainable Parameters} = 4 * ((128 + 1) * 128 + 128^2) = 131584 \tag{4.15}$$

First Time Distributed Layer: The formula for the calculation of trainable parameter for this layer is shown in the figure 4.17.

$$\text{Total Trainable Parameters} = n * m + m \tag{4.16}$$

Where n = input_size and m = ouput_size.Therefore,

$$\text{Total Trainable Parameters} = 128 * 256 + 256 = 33024 \tag{4.17}$$

Second Time Distributed Layer: Here the size_of_input is 256 and the size_of_output is 512.Therefore,

$$\text{Total Trainable Parameters} = 256 * 512 + 512 = 131584 \tag{4.18}$$

Flatten layer: It has no learn able parameters thus the value is 0.

Output Dense layer: In this layer the parameter is calculated via equation4.17.So,

$$\text{Total Trainable Parameters} = 10240 * 10 + 10 = 102410. \tag{4.19}$$

So, in LSTM model total trainable

$$\text{Total Trainable Parameters} = 68608{+}131584{+}33024{+}131584{+}0{+}102410 = 467210 \tag{4.20}$$

Using all of the mentioned functions and criteria in the LSTM model and MFCC as the main feature, we have run an epoch of 30 which is a standard one considering that the accuracy that the model provided was approximately constant after 22-25 epochs and a batch size of 50 have been used just like in CNN. These parameters work well for the LSTM model as well and the accuracy of the model shows in the figures 4.6 and 4.7 clearly.

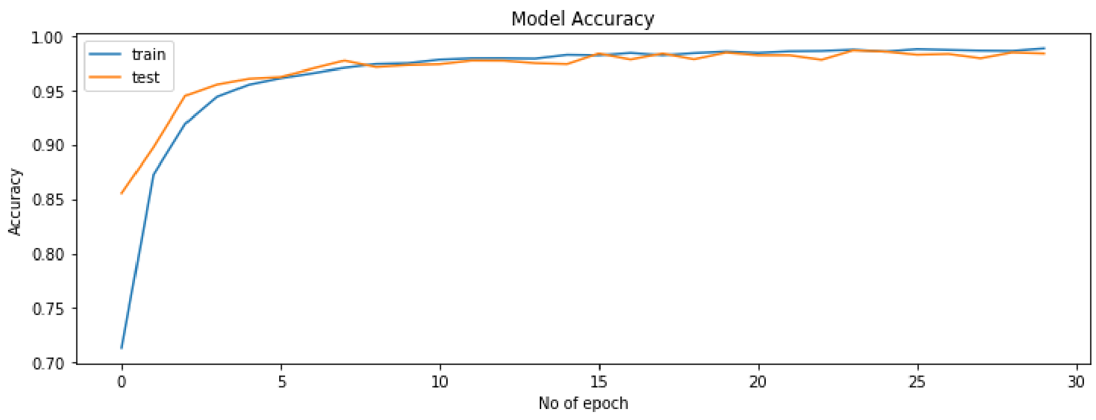


Figure 4.6: Model Accuracy of LSTM.

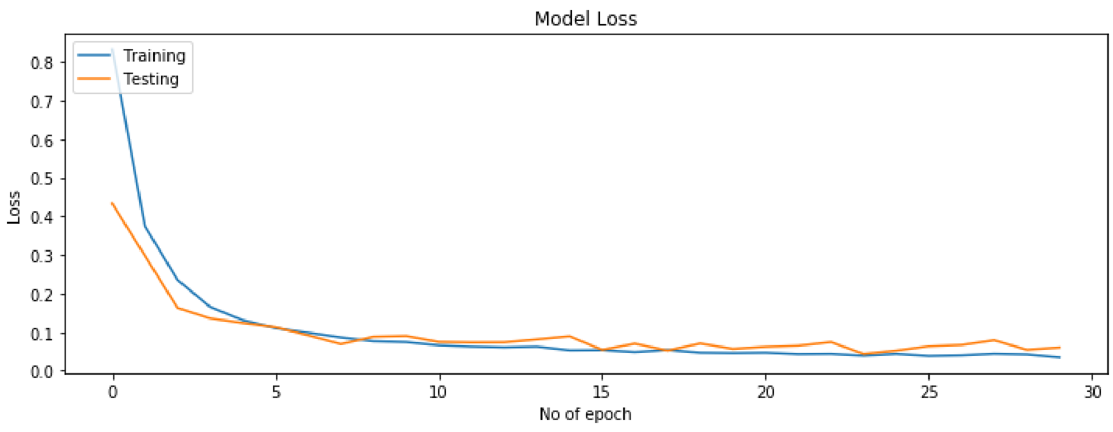


Figure 4.7: Model loss of LSTM.

The model accuracy in Figure 4.6 depicts the relationship of the accuracy of the training and testing data of the LSTM model with respect to the number of epochs. As we can see the accuracy of both the training and testing data increases as the number of epochs is increased. At epoch 0 the training data is approximately at 0.70 and the testing data is approximately at 0.85. With gradual increase in number of epochs we can see that the accuracy of both data increases. After 15-20 epochs we can see that the change in accuracy has almost become constant and fluctuates within the accuracy of 0.95 and 1.00. Furthermore, at a particular epoch if the accuracy is subtracted from the value 1, then the loss of training and testing data at that particular epoch can be calculated.

The Figure 4.7 of model loss depicts the relationship between the Loss in the Y-axis and Number of epochs in the X-axis. In the figure we can see that with the increase in the number of epochs the Loss of the model is decreasing in an exponential way for both the training and testing data. After 15-20 epochs it can be seen that the Loss has become almost constant for both the training and testing data at 0.0 and 0.1, and this continues until epochs. Initially the training data starts with a loss of about 0.8 and testing starts with a loss of about 0.4 at 0 epochs.

4.3 Model Compilation of CNN and LSTM

We will be using the following three parameters to compile both our model:

4.3.1 Loss Function

Among different loss functions, the loss function we will be using in both of our model is Categorical Crossentropy because of the increased number of classes in our model which determines the categorization of a single label. The distribution between predications (activations of output layers) and the true distribution will be compared through the Categorical Crossentropy where the likelihood of the true class is 1 or 0 for the other classes. A lower score is an indicator of better performance of the model. The equation of Categorical Crossentropy [50] is:

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * \log(\hat{y}_{ij})) \tag{4.21}$$

Here, ŷ is the predicted value.

4.3.2 Metric

The metric we will be using in both of our model is accuracy metric which is the fraction of our model's predictions. It will let us see the validation data accuracy score during the model training. Accuracy is generally obtained by the following equation from [51]:

$$Accuracy = \frac{No.of correct predictions}{No.of Total Predictions} \tag{4.22}$$

4.3.3 Optimizer

Here we will be using the optimizer Adam[46] that measures the adaptive learning rate for each parameter. This uses the adaptive models of learning rates to determine specific learning levels for each parameter. It uses the first and second gradient calculations to adjust the learning rate of the increasing neural weight. Adam prefers the flat surface error minima for which it is proven to be a good optimizer in many cases.The pseudo-code of Adam optimizer is provided in the appendix.

Chapter 5

Result and Analysis

5.1 Results Overview

UrbanSound8K dataset makes classification of audio easier because of its size and less number of classes.We have thoroughly tested our dataset on the model specified to show the importance of augmentation, stacking of features and size of the model.

Our approaches to the research experiment included identification of features that work effectively and produce high accuracy for both the models. In such scenarios, the MFCC was the best feature that showed the most accuracy for both the models compared to other features.

Our another approach to research experimentation was to stack different features together to find out how does the classification accuracy of the models change in accordance.On the other hand, stacking features increases the computational cost but some of the stacking approach has not really affected the performance by a great deal as predicted at first. The reason for this is that the difference between the array values are large and the spectrograms created after stacking do not follow a specific pattern for each class. But some of the stacking worked significantly well in boosting the performance of our model for example stacking MFCC and chroma STFT helped us to reach a validation accuracy of 98.8% shown in table 5.1 whereas the best performance till date was 97.52% in [40].

We have also shown that LSTM performed better compared to CNN when it comes to audio classification. This is simply because the network of cells in LSTM take input from the previous state ht-1 and present input xt which is used together to predict the next output. Since spectrograms like MFCC have approximately the same pattern through the time domain, it works better with such features. Some of very few researches have also shown that LSTM is very good model for data related to time series such as speech recognition, natural language recognition etc[37].

Finally, testing both these models with the original and augmented UrbanSound8k dataset as well as different features, we have been able to achieve state-of-the-art performance, which is 98.80%, with our LSTM model, shown in Table 5.3. Our CNN model also achieved a very impressive accuracy of 96.78%, which is shown in Table 5.2.

Model	***ClassificationAccuracy***(%)
EnvNet [25]	66.30
EnvNet + logmel-CNN [25]	71.10
EnvNetv2 [26]	76.60
EnvNetv2 + Strong Augment [26]	78.30
M18 [22]	71.68
PiczakCNN [17]	73.70
AlexNet [21]	92.00
GoogleNet [21]	93.00
SB-CNN [24]	79.00
CNN + Augment + Mixup [32]	83.70
GTSC + TEO-GTSC [20]	88.02
1D-CNN Random [34]	87.00
1D-CNN Gamma [34]	89.00
LMCNet [44]	95.20
MCNet [44]	95.30
TSCNN-DS [44]	97.20
ADCNN-5 [40]	97.52
Multiple Features with CNN and LSTM (Proposed)	**98.80**

Table 5.1: Proposed model vs previous state of art models.

The table 5.1 above shows our model outperforms all the previous models for the dataset of UrbanSound8K. Furthermore we achieved this state of art using LSTM and augmenting the dataset. As mentioned before, we have used time stretch and pitch shift function from [37] to expand the dataset. Augmentation plays a significant role in increasing the performance of both systems shown in Table 5.2 and Table 5.3. We have almost used all the features from made for audio and speech analysis to identify their importance and computational cost.Amongst all the features, we noticed that MFCC works best and works as a dominant feature when stacked with number of other features. Many researches[3] use MFCC as the only feature to classify sounds. MFCC single handedly gives the highest performance boost and stacking it with features like chroma and mel gives a slight increase in the performance which is also shown in table 5.2.

Implementation with CNN			
Features	Input Dimension	Testing Accuracy without Augmentation	Testing Accuracy With Augmentation
Mfcc	(20,5.1)	90.78	96.78
MelSpectogram	(20,5,1)	83.11	94.42
Chroma_CQT	(20,5,1)	64.68	69.41
Chroma_cens	(20,5,1)	31.37	32.04
Chroma_STFT	(20,5,1)	72.98	79.36
Mfcc+Melspectogram	(20,5,1)	89.63	94.97
Mfcc+Chroma_STFT	(20,5,1)	89.86	95.90
Mfcc+Chroma_CQT	(20,5,1)	88.32	96.02
MelSpectogram+chroma_STFT	(20,5,1)	82.25	95.02
MelSpectogram+chroma_CQT	(20,5,1)	79.91	93.17
Mfcc +MelSpectogram+chroma_STFT +Chroma_CQT	(20,5,1)	91.64	95.36
Mfcc +MelSpectogram+chroma_STFT +Tonnetz+Spectral_contrast(174 features)	(20,5,1)	86.09	94.99

Table 5.2: Performance of all the implementation using CNN

Implementation with LSTM			
Features	Input Dimension	Testing Accuracy without Augmentation	Testing Accuracy With Augmentation
Mfcc	(20,5)	93.30	98.23
MelSpectogram	(20,5)	81.85	96.25
Chroma_CQT	(20,5)	57.53	69.23
Chroma_cens	(20,5)	30.51	32.40
Chroma_STFT	(20,5)	65.48	78.92
Mfcc+Melspectogram	(20,5)	91.01	98.03
Mfcc+Chroma_STFT	(20,5)	93.76	**98.80**
Mfcc+Chroma_CQT	(20,5)	91.36	97.95
MelSpectogram+chroma_STFT	(20,5)	82.94	95.18
MelSpectogram+chroma_CQT	(20,5)	79.68	94.11
Mfcc +MelSpectogram+chroma_STFT +Chroma_CQT	(20,5)	91.76	98.34
Mfcc +MelSpectogram+chroma_STFT +Tonnetz+Spectral_contrast(174 features)	(20,5)	89.87	95.69

Table 5.3: Performance of all the implementation using LSTM

In the tables 5.2 and 5.3, we can see that MFCC works best for both the systems.Either used a single feature or being stacked with several other features, MFCC plays an important role to classify the sound correctly. MFCCs are frequency transformed and logarithmically scaled , moreover, it is universally recognised as the most effective feature for audio. The reason behind spectrograms created with other features looks distorted even for the sounds with similar patterns like drilling and jackhammer. But MFCC tries to keep the spectrogram following a single pattern throughout the time domain. For better illustration, we have used some figures we have extracted after stacking MFCC with unpatterened Chroma_stft which shown in figure 5.1

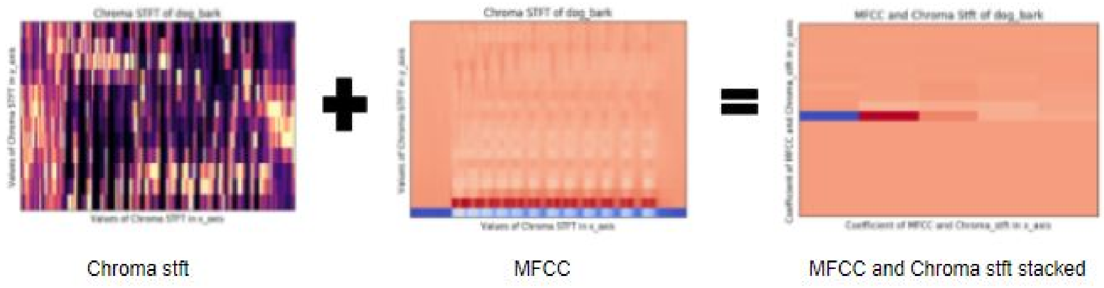


Figure 5.1: Resultant spectrogram from stacking.

5.2 Comparison between CNN and LSTM model

CNN is a very powerful technique for classification based on image, but since it has no memory built in it struggles to show better results while dealing with time series data. Instead of processing and working well with sequences of data, CNN is designed in way to exploit "spatial correlation" in data images and speeches. In scenarios where time series data or sequential data is involved, LSTM does a better job as it contains a feedback look that acts as a memory and thus the past inputs leave a footprint. As a result, any input that is fed into the LSTM model, which is sequential, like in our case, is classified with utmost accuracy. Furthermore, LSTM provides better accuracy than CNN due to its short-term sequence memory, which is not present in CNN. In particular, the backpropagation technique, which adjusts of weight each time to the degree of the derivative of the error to reduce total error on a set of training sequences, is adopted by LSTM networks[4]. The LSTM memory cell includes constant error backpropagation that allows LSTM to deal with data noise that is very useful for us, as the models have been applied with expanded datasets after augmentation using techniques of time stretch and pitch shift function from LibROSA[16]. In our research approach, we have used different features to observe how the classification accuracy varies for both the models. In case of MFCC, which is one of the most popular features used for classification, both models show high accuracy level because the coefficient of MFCC which are logarithmically scaled therefore large numerical values are represented in a compact way,in other words the pattern does not vary much, as the figure 3.5 depicts,This makes the MFCC a suitable feature for both the CNN and LSTM, so the accuracy of both models are 90% or above, but due the LSTM having the short-term memory sequence it can look back at the footprint left by the immediate MFCC coefficient and thus give a better accuracy than CNN.But that is not the case when it comes to the chroma features, where the pattern over the time domain is not constant and thus when the LSTM looks into its short-term sequence memory, most of the times it does not find a footprint made by the immediate input in the time domain. Then again, when we stacked different features together, we see a change in the pattern of the stacked features over the time domain, which is more suitable for the LSTM to classify.

In figure 5.2 and 5.3, we can see that LSTM outperforms CNN model in most cases when it comes to classification of audio. But CNN works slightly better with features like cqt, stft and mel. As the do not follow any pattern and CNN treats them as image generated from the audio signal. On the other hand, we can see features of Chromagram which represent the harmonic information[39] in simplified form of a audio signal, does not have a good performance in any of the model.Since chroma features vary alot throughtout the time domain, prediction from such spectrogram becomes difficult when it is treated as a image.Moreover, CENS features have the worst performance on our model because of temporal smoothing where the vectors are averaged and less sharp features are obtained.However, features like MFCC work best with LSTM. In the Figure 5.3 we have shown the significance of data augmentation which increases the testing accuracy by $\tilde{4}\%$. Our LSTM model have also achieved state of art performance with the help of data augmentation on a benchmark dataset of UrbanSound8k.

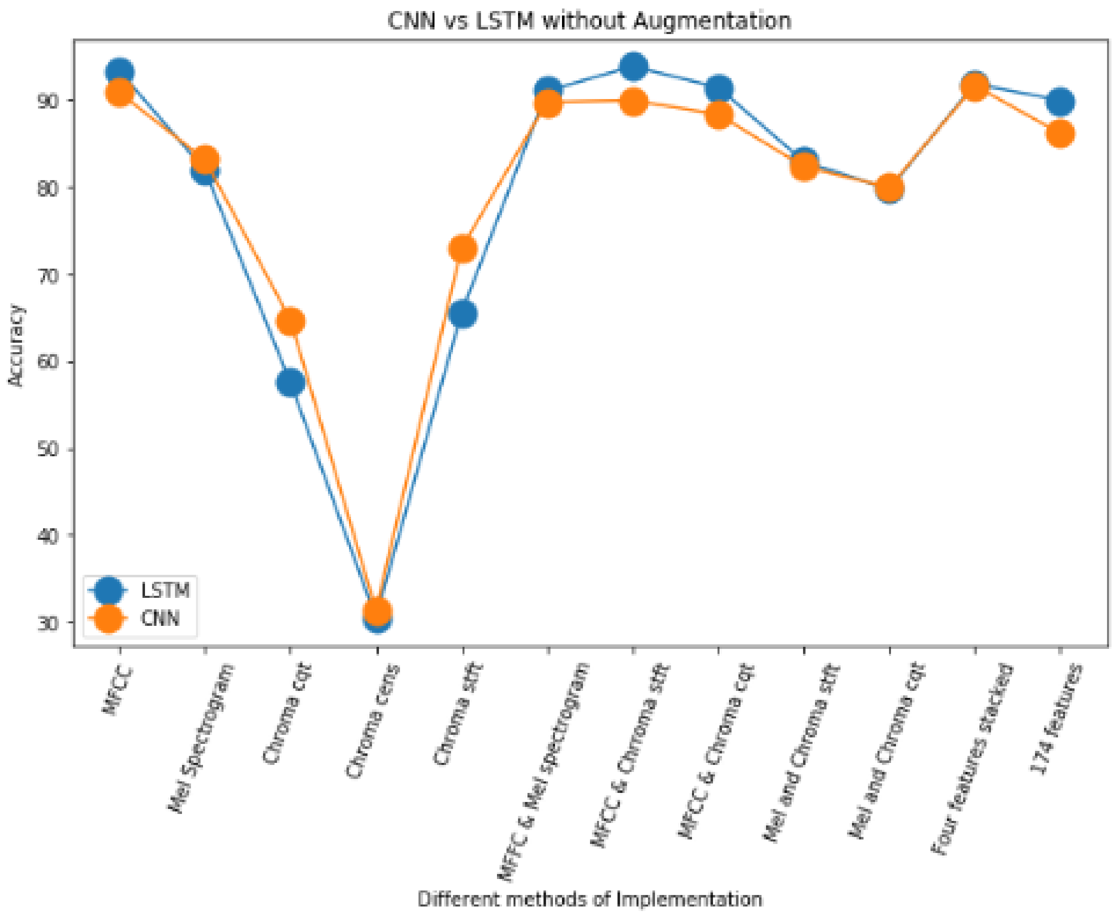


Figure 5.2: CNN VS LSTM without Augmentation.

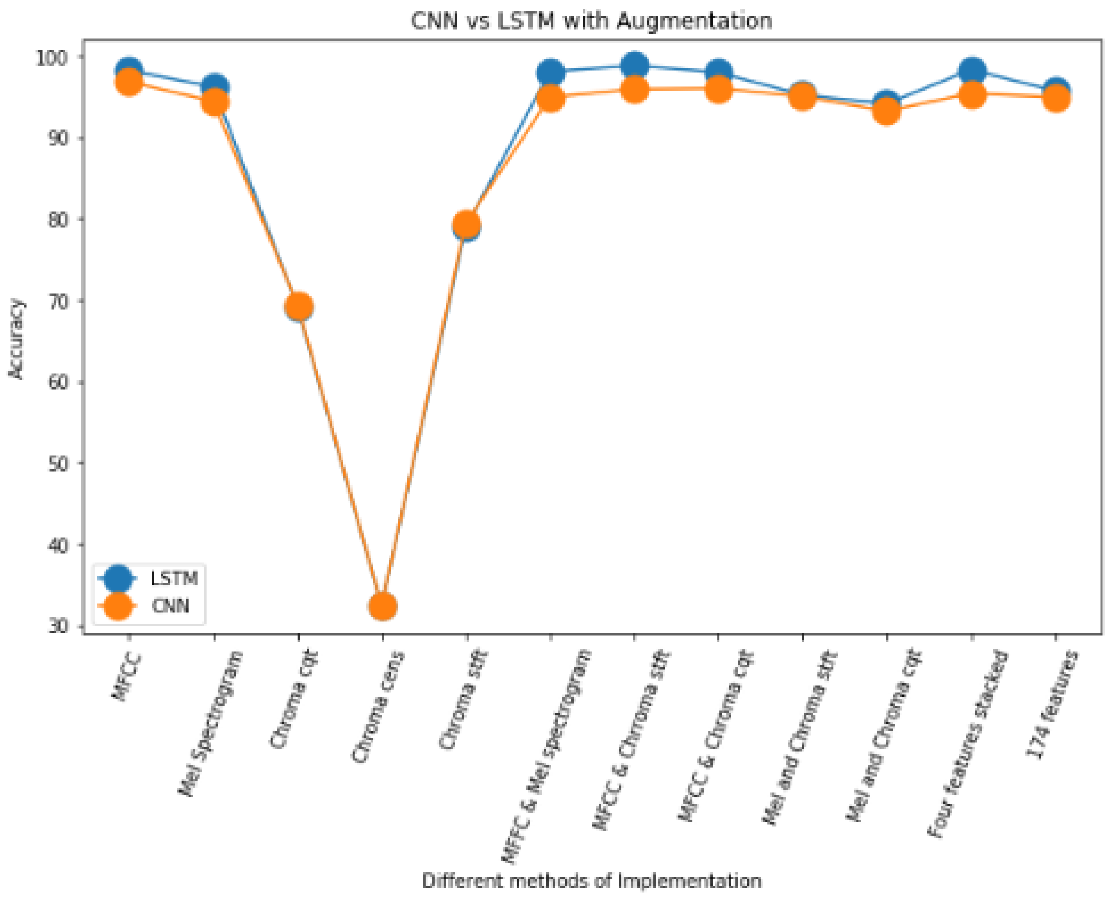
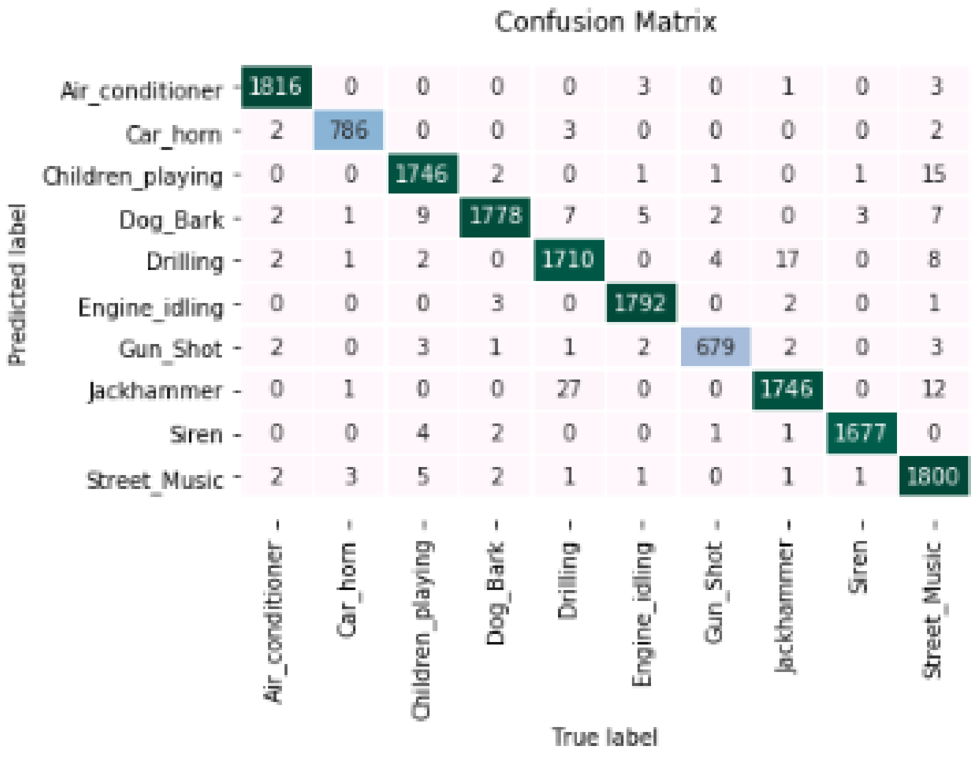


Figure 5.3: CNN VS LSTM with Augmentation.

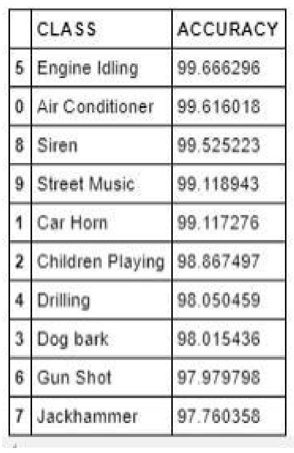
5.3 Analysis of state-of-the-art performance

In 5.4, we demonstrate the confusion matrix of our best performing approach which is LSTM using stacked features of MFCC and chroma stft. Since it outperformed all the previous models we declare our approach as state-of-the-art in audio classification. In the confusion matrix, we can see that out of 15718 of augmented testing data, our model has incorrectly classified only 186 testing samples reaching an accuracy of 98.8%. Furthermore, it performs really well for all the classes with 97.7% being the least accuracy for the class Jackhammer which is shown in the figure 5.5. From figure 5.4, we can notice that our approach works well in recognition of different groups (engine idling, air conditioner, siren, street music,car horn). Furthermore, we can also notice that jackhammer is misclassified as drilling and children playing is misclassified as street music most of the times because these samples sound similar.In total,our model have performed outstandingly for all the classes in the UrbanSound8k dataset with the aid of augmentation and stacking techniques



Predicted label \ True label	Air_conditioner	Car_horn	Children_playing	Dog_Bark	Drilling	Engine_idling	Gun_Shot	Jackhammer	Siren	Street_Music
Air_conditioner	1816	0	0	0	0	3	0	1	0	3
Car_horn	2	786	0	0	3	0	0	0	0	2
Children_playing	0	0	1746	2	0	1	1	0	1	15
Dog_Bark	2	1	9	1778	7	5	2	0	3	7
Drilling	2	1	2	0	1710	0	4	17	0	8
Engine_idling	0	0	0	3	0	1792	0	2	0	1
Gun_Shot	2	0	3	1	1	2	679	2	0	3
Jackhammer	0	1	0	0	27	0	0	1746	0	12
Siren	0	0	4	2	0	0	1	1	1677	0
Street_Music	2	3	5	2	1	1	0	1	1	1800

Figure 5.4: Confusion Matrix of our state-of-the-art system



	CLASS	ACCURACY
5	Engine Idling	99.666296
0	Air Conditioner	99.616018
8	Siren	99.525223
9	Street Music	99.118943
1	Car Horn	99.117276
2	Children Playing	98.867497
4	Drilling	98.050459
3	Dog bark	98.015436
6	Gun Shot	97.979798
7	Jackhammer	97.760358

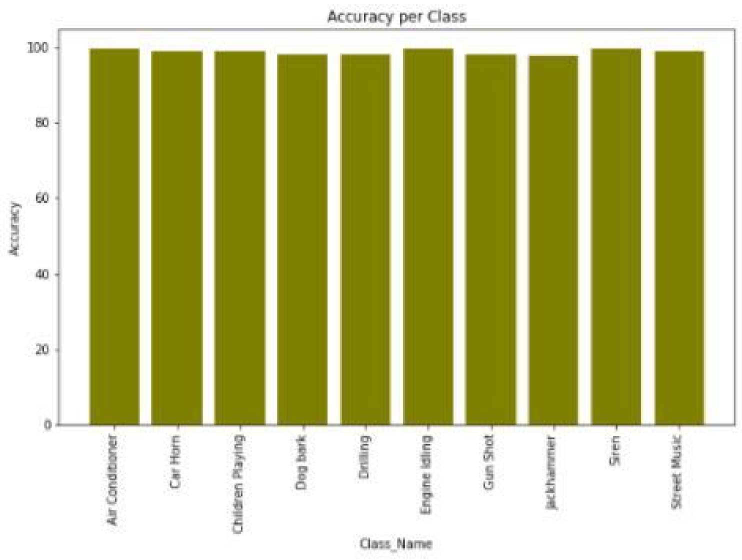


Figure 5.5: Accuracy of each class for our state-of-the-art system

5.4 Discussion

The integration of MFCC with other features gives a slight boost in the performance of our novel classifier which are LSTM and CNN. We combine many features like mel, mfcc, chroma cqt, chroma stft etc and use it as an input for our system. Each one of these features sets offers unique and selective details that enhance the classification precision. Our built models have satisfied the key objective of reaching state-of-the-art performance where LSTM have worked better in classification compared to CNN. Nevertheless, our CNN consisting of only 2 convolution layers have also managed to reach an accuracy of 96.78% shown in table 5.2. Finally, with all the results and statistics we can say both the models have performed remarkably when comes to audio classification.

Chapter 6

Conclusion

6.1 Research Overview

We are presenting an approach to sound classification, which consists of multiple features stacking and two different neural network models which are CNN and LSTM. Both the models have been trained and tested with original and augmented UrbanSound8K dataset. Different feature extraction methods have been applied both separately as well as in the form of two different stackings on both the models to test their accuracy levels. While one form of the stacking consisted of MFCC, Tonnetz, Mel-spectrogram, Chromagram and spectral contrast, the other contained MFCC, Mel-Spectrogram, Chroma_STFT and Chroma_Cqt. As the results of our two models show, each feature and the two forms of stacking contain some specifications and separate characteristics that either increases or decreases the classification accuracy of each of the models.

We have used two models for our research purpose, which are CNN and LSTM. While CNN has been used as an audio classifying model for a long time now and is very effective, the LSTM model has become recently popular showing high accuracy performance in this field. In our implementation, the CNN model contains a total of 8 layers. These layers include two Conv2D layers, each of which are followed by a MaxPooling2D layer, and after that comes a flatten layer and then two fully connected dense layers, finishing off with an output layer. ReLu as activation function has been used in the Conv2D layers while in the output layer softmax has been used. On the other hand the LSTM model consists of 6 layers. Two LSTM layers are followed by two Time Distributed dense layers, having activation function ReLu, and then a flatten layer comes, finishing off with a dense layer having softmax activation function. Furthermore, we have used LibROSA, which is an audio and music analysis python package built by Brian Mcfee, that uses some parameters like sampling rate, hop_length and n_fft to analyze the audio signals. Moreover, Adam optimizer, metric accuracy and categorical cross entropy has been used as a loss function.

As this research is aimed towards showing a valid comparison between the two models in terms of classification accuracy, we used a common and the mostly used feature which is MFCC to show the comparison. Furthermore, to ensure that the results obtained for accuracy are always true, we have used both original and augmented dataset on both the models. Lastly, for both the models we have adopted a supervised learning approach.

6.2 Research Challenges

We are presenting an approach to sound classification, which consists of multiple features stacking and two different neural network models which are CNN and LSTM. Both the models have been trained and tested with original and augmented UrbanSound8K dataset. Different feature extraction methods have been applied both separately as well as in the form of two different stackings on both the models to test their accuracy levels. While one form of the stacking consisted of MFCC, Tonnetz, Mel-spectrogram, Chromagram and spectral contrast, the other contained MFCC, Mel-Spectrogram, Chroma_STFT and Chroma_Cqt. As the results of our two models show, each feature and the two forms of stacking contain some specifications and separate characteristics that either increases or decreases the classification accuracy of each of the models.

We have used two models for our research purpose, which are CNN and LSTM. While CNN has been used as an audio classifying model for a long time now and is very effective, the LSTM model has become recently popular showing high accuracy performance in this field. In our implementation, the CNN model contains a total of 8 layers. These layers include two Conv2D layers, each of which are followed by a MaxPooling2D layer, and after that comes a flatten layer and then two fully connected dense layers, finishing off with an output layer. ReLu as activation function has been used in the Conv2D layers while in the output layer softmax has been used. On the other hand the LSTM model consists of 6 layers. Two LSTM layers are followed by two Time Distributed dense layers, having activation function ReLu, and then a flatten layer comes, finishing off with a dense layer having softmax activation function. Furthermore, we have used LibROSA, which is an audio and music analysis python package built by Brian Mcfee, that uses some parameters like sampling rate, hop_length and n_fft to analyze the audio signals. Moreover, Adam optimizer, metric accuracy and categorical cross entropy has been used as a loss function.

As this research is aimed towards showing a valid comparison between the two models in terms of classification accuracy, we used a common and the mostly used feature which is MFCC to show the comparison. Furthermore, to ensure that the results obtained for accuracy are always true, we have used both original and augmented dataset on both the models. Lastly, for both the models we have adopted a supervised learning approach.

6.3 Recommendation and Future Work

The main recommendation regarding this research will be to ensure that the same result regarding the comparison of the two models can be done using unsupervised learning as well. Furthermore, some parameters and architectural tuning can be done to ensure that the CNN model also shows state-of-the-art performance. Different features can also be used along with different forms of stacking to test the classification accuracy of the models.

Bibliography

[1] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform", *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978. DOI: 10.1109/proc.1978.10837.

[2] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[3] J. C. Brown, "Calculation of a constant q spectral transform", *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] M. Ridley and D. MacQueen, "Sampling plan optimization: A data review and sampling frequency evaluation process", *Bioremediation journal*, vol. 8, no. 3-4, pp. 167–175, 2004.

[6] M. Miiller, "Information retrieval for music and motion", *Springer, Berlin Heidelberg*, 2007.

[7] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] Z. Song, B. Liu, Y. Pang, C. Hou, and X. Li, "An improved nyquist–shannon irregular sampling theorem from local averages", *IEEE transactions on information theory*, vol. 58, no. 9, pp. 6093–6100, 2012.

[10] A. Abushakra and M. Faezipour, "Acoustic signal classification of breathing movements to virtually aid breath regulation", *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 2, pp. 493–500, 2013. DOI: 10.1109/jbhi.2013.2244901.

[11] S. Chachada and C.-C.J.Kuo, "Environmental sound recognition:a survey", vol. 3, pp. 1–9, Oct. 2013.

[12] S. Gupta, J. Jaafar, W. Fatimah, and A. Bansal, "Feature extraction using mfcc", 2013.

[13] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks", *arXiv preprint arXiv:1312.6199*, 2013.

[14] S. Magre, P. Janse, and R. Deshmukh, "A review on feature extraction and noise reduction technique", Feb. 2014.

[15] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research", in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044.

[16] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in python", in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[17] K. J. Piczak, "Environmental sound classification with convolutional neural networks", in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2015, pp. 1–6.

[18] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4580–4584.

[19] T. Elliot, J. Howard, R. Lisam, K. Fehling, A. d. Luca, and D. Haba, *Dense vs convolutional vs fully connected layers*, Nov. 2016. [Online]. Available: https://forums.fast.ai/t/dense-vs-convolutional-vs-fully-connected-layers/191.

[20] D. M. Agrawal, H. B. Sailor, M. H. Soni, and H. A. Patil, "Novel teo-based gammatone features for environmental sound classification", in *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, 2017, pp. 1809–1813.

[21] V. Boddapati, A. Petef, J. Rasmusson, and L. Lundberg, "Classifying environmental sounds using image recognition networks", *Procedia computer science*, vol. 112, pp. 2048–2056, 2017.

[22] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 421–425.

[23] Greff, Klaus, Srivastava, R. Kumar, Koutník, Steunebrink, Schmidhuber, and Jürgen, *Lstm: A search space odyssey*, Oct. 2017. [Online]. Available: https://arxiv.org/abs/1503.04069.

[24] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification", *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[25] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 2721–2725.

[26] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition", *arXiv preprint arXiv:1711.10282*, 2017.

[27] C.-Y. Yu, H. Liu, and Z.-M. Qi, "Sound event detection using deep random forest", in *Detection and Classification of Acoustic Scenes and Events*, 2017.

[28] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4774–4778.

[29] N. Davis and K. Suresh, "Environmental sound classification using deep convolutional neural networks and data augmentation", in *2018 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, 2018, pp. 41–45.

[30] Madhivarman, *How to calculate the number of parameters in the cnn?*, May 2018. [Online]. Available: https://medium.com/@iamvarman/how-to-calculate-the-number-of-parameters-in-the-cnn-5bd55364d7ca.

[31] J. Sang, S. Park, and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms", in *2018 26th European Signal Processing Conference (EUSIPCO)*, IEEE, 2018, pp. 2444–2448.

[32] Z. Zhang, S. Xu, S. Cao, and S. Zhang, "Deep convolutional neural network with mixup for environmental sound classification", in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, Springer, 2018, pp. 356–367.

[33] B. Zhu, K. Xu, D. Wang, L. Zhang, B. Li, and Y. Peng, "Environmental sound classification based on multi-temporal resolution cnn network", *CoRR, abs/1805.09752*, 2018.

[34] Abdoli, Sajjad, Patrick, Koerich, and A. Lameiras, *End-to-end environmental sound classification using a 1d convolutional neural network*, Apr. 2019. [Online]. Available: https://arxiv.org/abs/1904.08990.

[35] S. Agrawal, *Reading between the layers (lstm network)*, Feb. 2019. [Online]. Available: https://towardsdatascience.com/reading-between-the-layers-lstm-network-7956ad192e58?.

[36] I. Lezhenin, N. Bogach, and E. Pyshkin, "Urban sound classification using long short-term memory neural network", in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2019, pp. 57–60.

[37] M. Nguyen, *Illustrated guide to lstms and grus: A step by step explanation*, Jul. 2019. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[38] T. Qiao, S. Zhang, Z. Zhang, S. Cao, and S. Xu, "Sub-spectrogram segmentation for environmental sound classification via convolutional recurrent neural network and score level fusion", *arXiv preprint arXiv:1908.05863*, 2019.

[39] A. Shah, M. Kattel, A. Nepal, and D. Shrestha, "Chroma feature extraction", Jan. 2019.

[40] J. Sharma, O.-C. Granmo, and M. Goodwin, "Environment sound classification using multiple feature channels and deep convolutional neural networks", *arXiv preprint arXiv:1908.11219*, 2019.

[41] S. Sharma, *Epoch vs batch size vs iterations*, Mar. 2019. [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

[42] T. Singh, *Mfcc's made easy*, Jun. 2019. [Online]. Available: https://medium.com/@tanveer9812/mfccs-made-easy-7ef383006040.

[43] *Softmax layer*, May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[44] Y. Su, K. Zhang, J. Wang, and K. Madani, "Environment sound classification using a two-stream cnn based on decision-level fusion", *Sensors*, vol. 19, no. 7, p. 1733, 2019.

[45] E. Tzinis, S. Wisdom, J. R. Hershey, A. Jansen, and D. P. Ellis, "Improving universal sound separation using sound classification", *arXiv preprint arXiv:1911.07951*, 2019.

[46] J. Yuan and Y. Tian, "An intelligent fault diagnosis method using gru neural network towards sequential data in dynamic processes", *Processes*, vol. 7, p. 152, Mar. 2019. DOI: 10.3390/pr7030152.

[47] Esmaeilpour, Mohammad, P. Cardinal, and A. L. Koerich, "Unsupervised feature learning for environmental sound classification using weighted cycle-consistent generative adversarial network", *Applied Soft Computing 86, 105912*, 2020.

[48] R. Vasudev, *Understanding and calculating the number of parameters in convolution neural networks (cnns)*, Jan. 2020. [Online]. Available: https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d.

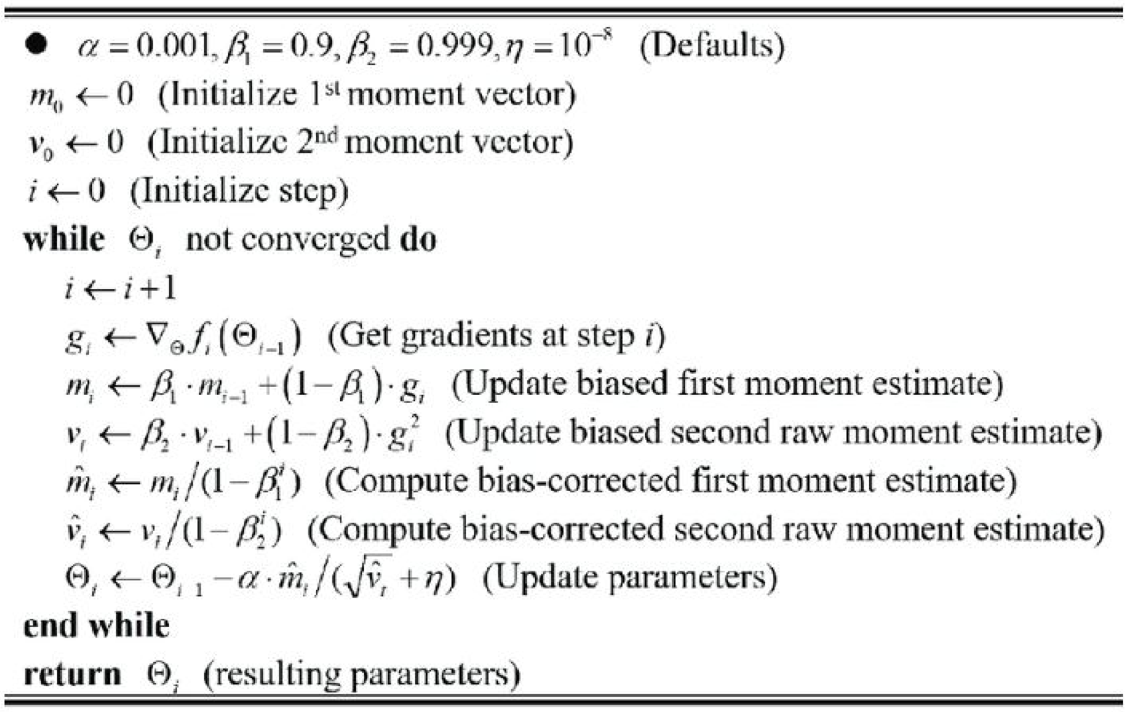
[49] A. Y. Yang and L. Cheng, "Two-step surface damage detection scheme using convolutional neural network and artificial neural neural", *arXiv preprint arXiv:2003.10760*, 2020.

[50] *Categorical crossentropy loss function: Peltarion platform.* [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.

[51] *Classification: Accuracy — machine learning crash course.* [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[52] K. Prahallad, *Topic: Spectrogram, cepstrum and mel-frequency analysis.* [Online]. Available: https://www.cs.brandeis.edu/~cs136a/CS136a_docs/KishorePrahallad_CMU_mfcc.pdf.

Adam Optimizer Pseudo-code



```
● $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \eta = 10^{-8}$ (Defaults)
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$i \leftarrow 0$ (Initialize step)
while $\Theta_i$ not converged do
    $i \leftarrow i + 1$
    $g_i \leftarrow \nabla_\Theta f_i(\Theta_{i-1})$ (Get gradients at step $i$)
    $m_i \leftarrow \beta_1 \cdot m_{i-1} + (1 - \beta_1) \cdot g_i$ (Update biased first moment estimate)
    $v_i \leftarrow \beta_2 \cdot v_{i-1} + (1 - \beta_2) \cdot g_i^2$ (Update biased second raw moment estimate)
    $\hat{m}_i \leftarrow m_i / (1 - \beta_1^i)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_i \leftarrow v_i / (1 - \beta_2^i)$ (Compute bias-corrected second raw moment estimate)
    $\Theta_i \leftarrow \Theta_{i\;1} - \alpha \cdot \hat{m}_i / (\sqrt{\hat{v}_i} + \eta)$ (Update parameters)
end while
return $\Theta_i$ (resulting parameters)
```

Figure 6.1: Adam optimization pseudocode