VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018

Project Report

on

"DEEP LEARNING BASED AUDIO TAGGING USING IMAGE PROCESSING"

Submitted in partial fulfilment of the requirements for VIII Semester

Bachelor of Engineering

in

COMPUTER SCIENCE AND ENGINEERING

For the Academic Year

2019-2020

BY

RAHUL SURESH	**1PE16CS188**
VENKATESH B N	**1PE16CS174**
RAM SINGH	**1PE16CS197**
SHRUTI DANAGAR	**1PE17CS426**

UNDER THE GUIDANCE OF

Dr. D C KIRAN

Associate Professor, Dept. of CSE, PESIT-BSC

Department of Computer Science and Engineering

PESIT - BANGALORE SOUTH CAMPUS

Hosur Road, Bengaluru - 560100

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018

Project phase-2 Report

on

"DEEP LEARNING BASED AUDIO TAGGING USING IMAGE PROCESSING"

Submitted in partial fulfilment of the requirements for the VIII Semester

Bachelor of Engineering in

COMPUTER SCIENCE AND ENGINEERING

For the Academic Year

2019-2020

BY

RAHUL SURESH	**1PE16CS188**
VENKATESH B N	**1PE16CS174**
RAM SINGH	**1PE16CS197**
SHRUTI DANAGAR	**1PE17CS426**

UNDER THE GUIDANCE OF

Dr. D C KIRAN

Associate Professor, Dept. of CSE, PESIT-BSC

Department of Computer Science and Engineering

PESIT - BANGALORE SOUTH CAMPUS

Hosur Road, Bengaluru - 560100

PESIT - BANGALORE SOUTH CAMPUS

HOSUR ROAD, BENGALURU - 560100

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "Deep Learning Based Audio Tagging Using Image Processing" carried out by Rahul Suresh, Venkatesh B N, Ram Singh, Shruti Danagar bearing USN's 1PE16CS188, 1PE16CS174, 1PE16CS197, 1PE17CS426 respectively in partial fulfillment for the award of Degree of Bachelors (Bachelors of Engineering) in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the year 2019-2020. It is certified that all corrections/ suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Signature of the Guide	Signature of the HOD	Signature of the Principal
Dr. D C Kiran	**Dr. Annapurna D**	**Dr. Subhash Kulkarni**
Associate Professor	**HOD, CSE**	**Principal, PESIT-BSC**

External Viva

Name of the Examiners **Signature with Date**

1.

2.

Declaration

We, Rahul Suresh (1PE16CS188), Venkatesh B N (1PE16CS174), Ram Singh (1PE16CS197) and Shruti Danagar (1PE17CS426) hereby declare that the dissertation entitled, 'Deep Learning Based Audio Tagging Using Image Processing', is an original work done by us under the guidance of Dr. D C Kiran, Associate Professor, Department of Computer Science and Engineering, PESIT-BSC, Bengaluru, is being submitted in partial fulfilment for the award of B.E. in Computer Science and Engineering, VTU of the requirements for 8th semester.

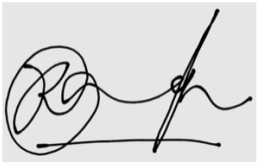
Date: 31-07-2020

Place: Bangalore

Signature of the candidate

Rahul Suresh 

Venkatesh B N 

Ram Singh 

Shruti Danagar 

Word Count: 7263

Plagiarism Percentage 18%

Matches

1 **World Wide Web Match**
View Link

2 **World Wide Web Match**
View Link

3 **World Wide Web Match**
View Link

4 **World Wide Web Match**
View Link

5 **World Wide Web Match**
View Link

6 **World Wide Web Match**
View Link

7 **World Wide Web Match**
View Link

8 **World Wide Web Match**
View Link

9 **World Wide Web Match**
View Link

10 **World Wide Web Match**
View Link

11 **World Wide Web Match**

Acknowledgements

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowned our effort with success.

We are indebted to our Guide, **Dr. D C Kiran**, Associate Professor, Department of Computer Science and Engineering, PESIT Bangalore South Campus, who has not only coordinated our work but also given suggestions from time to time.

We are also extremely grateful for our project coordinators , Assistant Professor **Prof. Shubaraj K.B.**, and Assistant Professor **Prof. Sheba Pari N.**, Department Of Computer Science And Engineering, PESIT- Bangalore South Campus, for their constant support and advice throughout the course of preparation of this document. support and advice throughout the course of preparation of this document.

We are greatly thankful to **Dr. Annapurna D**, Professor and HOD, Department of Computer Science and Engineering, PESIT Bangalore South Campus, for her able guidance, regular encouragement and assistance throughout this project.

We would like to express our immense gratitude to **Dr. Subhash Kulkarni**, Principal, PESIT Bangalore South Campus, for providing us with excellent infrastructure to complete our project work.

We gratefully acknowledge the help lent out to us by all faculty members of the Department of Computer Science and Engineering , PESIT Bangalore South Campus, at all difficult times. We would also take this opportunity to thank our college management for the facilities provided during the course of the project. Furthermore, we acknowledge the support and feedback of our parents and friends.

RAHUL SURESH
VENKATESH B N
RAM SINGH
SHRUTI DANAGAR

ABSTRACT

Audio tagging is the process of recognising, classifying and inferring descriptive labels from audio clips. This can be treated as a multi label classification task. Due to the significant amount of manual effort involved in tasks like annotation of sound collections, there is a need for audio preprocessing and preparation techniques which can be used to efficiently perform the task of audio tagging.

The main task addressed in this project is the research into audio preprocessing and deep learning based image processing techniques that can be used to recognise sounds, classify them, and apply tags of varying natures. In an effort to study the applications of audio tagging for real world problems, there is a need to develop a good general purpose audio tagging model. Post this, transfer learning can be performed using the developed model, on applications that require real time audio processing and classification.

Making use of different audio sample datasets, audio parameters and pre-processing techniques, along with appropriate deep neural network architectures, we aim to perform the multi-label classification task accurately and ideally perform content based recognition of real time sounds.

Keywords: Audio Analysis and pre-processing, Audio Tagging, Deep Learning, Multi-label classifier, Image processing, Deep neural networks

Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose

- The task explored in this research/project is the development of a general purpose audio tagging system capable of classifying sound collections for a good range of real-world environments.
- The need for such a system arises due to the fact that researchers are still trying to develop a reliable automatic audio tagging system.
- Through this project, we aim to research into the domains of audio signal processing as well as image processing, to find common ground in the task of audio tagging. These methods can help further research into the topic and all its applications.

1.2 Scope

- A general purpose Audio Tagging model can help tag audio clips with descriptive labels. This can help websites like Youtube, Freesound, and Flickr improve the metadata related to the large volumes of user-contributed audio they host.
- In the context of noise pollution monitoring, Audio tagging and related computational methods can help in the recognition and classification of urban sounds and acoustic events, to determine if they are in violation of allowed decibel levels.
- Research into the task of audio tagging can open doors to a wide variety of applications that require real time audio processing, recognition, and classification.

Chapter 2

Literature Survey

2.1 Ensemble of Convolutions Neural Networks for Weakly-Supervised Sound Event Detection Using Multiple Scale Input

This paper explores two types of Convolution Neural Networks in the task of detecting and tagging audio events, namely Global Input Model and Separation Input Model. An ensemble of the two combined structures is explored, and performance is evaluated on the DCASE2017 challenge dataset. Results showed that a combination of the two models performed well in the given task.[8]

2.2 General Audio Tagging with Ensembling Convolutional Neural Networks and Statistical Features

This paper explores a variety of Convolution Neural Networks such as Resnet and Densenet. Statistical features such as Skewness and Kurtosis are applied on the data. To solve the audio tagging problem, stacked generalization in multiple levels is explored to improve the accuracy and robustness. The system achieves a mean average precision of 0.958, outperforming the baseline system of 0.704.[11]

2.3 Unsupervised Feature Learning Based on Deep Models for Environmental Audio Tagging

This paper proposes an Unsupervised shrinking Deep Learning model to effectively handle acoustic labelling. Further, Symmetric and Asymmetric deep Denoising Auto-Encoder (syDAE or asyDAE) methods are used to generate new data driven

features from the DCASE 2016 evaluation set. A high high accuracy was observed because of the use of syDAE or asyDAE to obstruct and reconstruct the data features. As a result, the DNN is highly optimised to classify even unseen examples. The authors also suggest using a CNN along with the unsupervised DNN to further decrease the EER of the unseen samples.[12]

2.4 A Comparison on Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging

This paper comparitively studies the effect of different audio pre-processing techniques on the overall accuracy of the DNN model. Among several preprocessing techniques tested in this study, only logarithmic scaling of magnitude of the signal resulted in significant improvement. Log scaling refers to the compression of the time-frequency varying melspectogram magnitudes represented as $\log(X + \alpha)$ where α can be an arbitrary constant like epsilon or 1. The authors assure that this preprocessing technique can be used in most audio classification tasks to improve the accuracy gain of the model.[1]

2.5 Transfer Learning of Weakly Labelled Audio

This paper proposes a method to transfer the learning of a Recurrent Neural Network (RNN) trained on abundant labelled data, to classify less abundant weakly labelled data. The goal was to build an architecture for a general purpose neural network which could be effectively used for transfer learning. Three transfer learning models are compared with the fully trained RNN on abundant data. The proposed model uses two parallel lower-level LSTM layers, one of which will have weights learnt from the source task and then fixed, while the other will be trainable on the weakly labelled data. From the experimental results, it was found that the model had excellent classification accuracy, but still less than that of the fully trained RNN.[2]

2.6 General-purpose Audio Tagging from Noisy Labels Using Convolutional NeuralNetworks

This paper proposes a system that is an ensemble of Convolutional Neural Networks trained on log-scaled Mel Spectrograms. Different pre-processing and data

augmentation methods are used to improve the performance further. For example, each feature vector is partitioned into chunks of a fixed size, which resolves the problem of varying clip durations. The system achieves a mean average precision of 0.972, out performing the base line system 0.694 excellence performance.[6]

2.7 Content-based Auto-tagging of Audios using Deep Learning

This paper proposes a Convolutional Recurrent Neural Network used for the multi-label classification task of audio clip tag prediction, which outperformed a Fully Convolutional Neural Network for the same task. The authors also suggest that the AUC-ROC score serves as a good result measure for the comparison.[9]

2.8 A Joint Detection-Classification Model for Audio Tagging of Weakly Labelled Data

This paper explores the use of the Joint Detection-Classification (JDC) model to detect sounds in an audio clip, and classify the clip on the whole based on whether the detected sounds are informative or not. Experimental results showed that with a good amount of training data, the JDC suffers less from over fitting, as opposed to the Bag of Blocks model.[7]

Chapter 3

System Requirements Specification

3.1 Operating Environment

This section briefs about the hardware and software pre-requisites for the project development environment. However, the end goal of the project is for it to be deployed on any type of computing device, embedded system, sensor, edge device etc., that can meet the minimum operation requirements of signal processing and deep learning model testing. This is a matter for future research.

3.1.1 Hardware Requirements

- **Processor**: Dual-Core - Intel Core i5 (1.6 GHz)
- **RAM**: 8GB - 1600 MHz DDR3
- **Storage**: 128GB Flash Storage
- Other general hardware such as a mouse and keyboard for inputs and a monitor for display

3.1.2 Software Requirements

- **Operating System**: MacOS Sierra (version 10.12) and above / Ubuntu 14.04 and above / Windows 8 and above.
- **Development environment**: Kaggle Notebooks, Jupyter Notebooks
- **Programming language**: Python
- **Packages**: Numpy, Librosa, Tensorflow, Keras, OpenCV, Pandas, Matplotlib

Chapter 4

High level Design

4.1 System Design

The desired audio tagging system is one that has been trained on multiple audio samples of varying natures, and is capable of performing audio recognition, classification and tagging of any audio sample. Input to the system is a raw audio sample, which is then subject to certain pre-processing stages, before classification using the trained deep learning model.

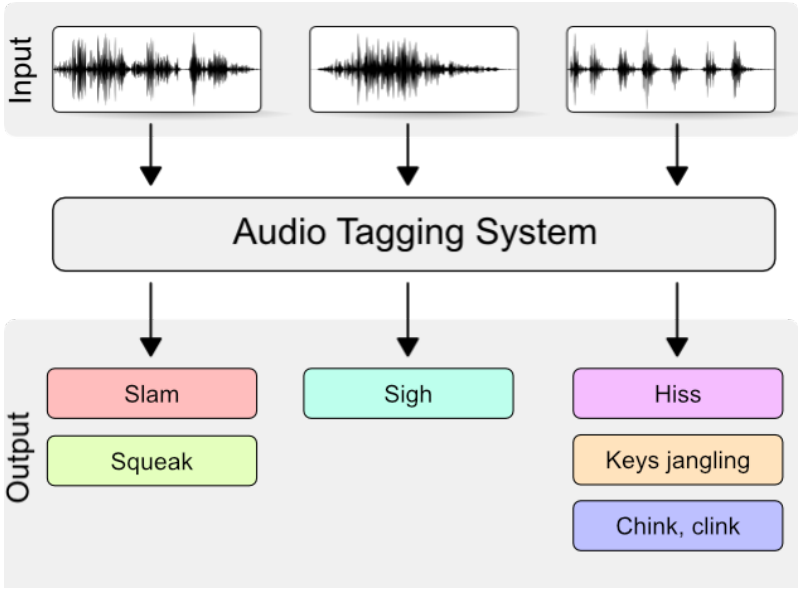


Figure 4.1: Desired tagging system[4]

Since the system is intended to be a general purpose one, it would be used for a variety of applications that require real time audio recognition and classification. However, since the idea of a general purpose tagging system is still far fetched, application specific audio samples would be required for improving the learning model's cognition. The system would also have to be loaded onto an application specific computing device, which meets the minimum processing power requirements to execute the pipeline steps mentioned in section 4.2.

4.2 Processing pipeline

The input to the audio tagging system is a raw audio sample, which needs to be converted from the time domain into the frequency domain. This frequency domain representation of the audio sample is in turn, used as the input image to the DL (deep learning) model for the audio recognition and multi label classification task. The end goal is to have the above mentioned computation be performed at the edge level, on hardware that is connected directly to the application specific sensors/devices.

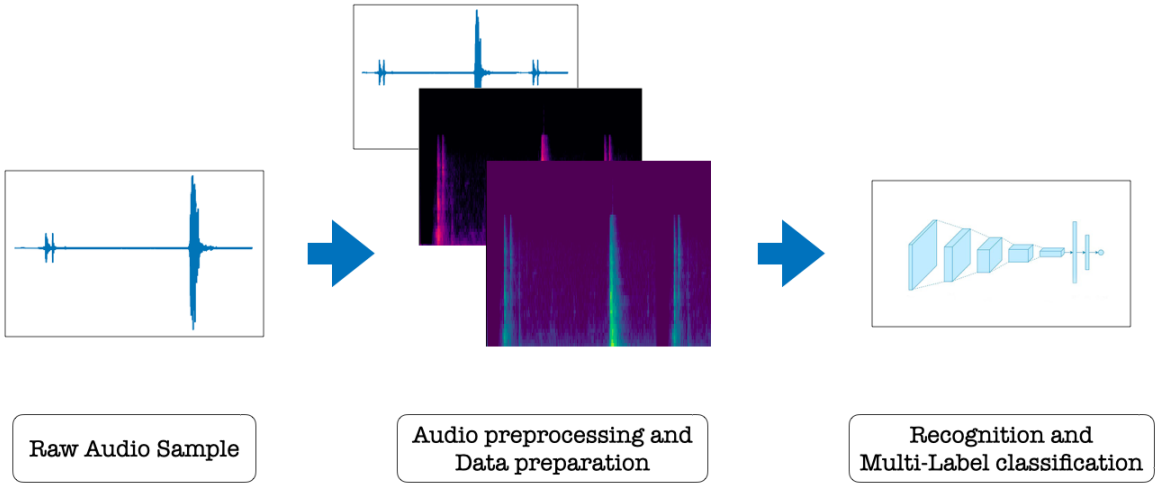


Figure 4.2: Processing pipeline

4.3 Audio Preprocessing

As a part of preprocessing the data to extract useful features and information, the audio samples are transformed into their frequency domain representations, specifically the Mel Spectrogram representation. These audio representations may undergo further resizing and dimensionality reduction that is specific to the deep learning model used.

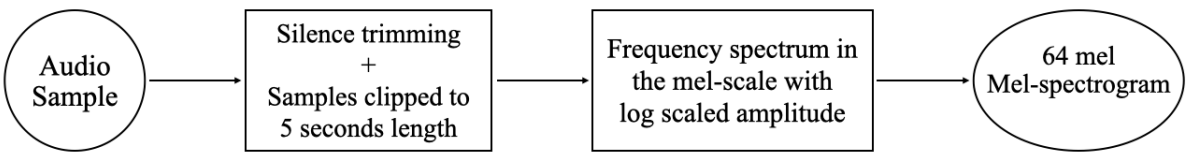


Figure 4.3: Preprocessing Steps

Chapter 5

Implementation

Implementation is the phase of the project when the hypothetical project plan is transformed into a working framework. It is where plans are evaluated, visions become reality, all leading upto the logical conclusion of the project.

5.1 Programming Language Selection

This project is implemented in Python due to it's many features such as:

- **Open Source**: The Python language is open-source and completely free to use, which means that it's source code is publicly available.
- **Object-Oriented Language**: One of the most important features of Python is its support of OOP (Object Oriented Programming). Python supports object oriented language, along with the concepts of classes, encapsulation, objects, etc.
- **High-Level Language**: Python programming language is a high-level one. When Python programs are written, there is no need to consider system architecture, manage the memory, etc.
- **Extensible feature**: Python is an extensible language. Some Python code can even be written and compiled in C/C++ languages.
- **Portability**: Python language is also a portable one. Python code can be run on Linux, Unix, Windows, Mac, etc., without the need for many changes in code.

Apart from these reasons, Python also provides support for many high level external libraries for tasks such as Image Processing, Signal Processing, Audio Analysis, Machine Learning, complex mathematical operations, etc. These benefits allow for the quick translation of theoretical concepts to practical trials, often what is required in research studies and projects.

5.2 Platform Selection

5.2.1 Ubuntu

Ubuntu is an open-source and free Debian OS based Linux distribution. It's released in three editions: Core, Desktop and Server. The Core edition is used for IoT (Internet of Things) devices and robots. All editions can run on any computer alone, or in virtual machines. Ubuntu is also a popular OS for cloud computing, with OpenStack support. Ubuntu being developed by Canonical, also gets the support of a community of other developers. This is done under the meritocratic governance model. It is released every 6 months, with long-term support releases (LTS) every 2 years. Security updates, along with support are provided by Canonical for each Ubuntu release, starting from the release date, until the designated end-of-life (EOL) date.

5.2.2 Kaggle notebooks

Kaggle Notebooks offers a cloud computational environment that permits reproducible and collaborative analysis. Kaggle supports two types of workspaces for computation: Scripts, which are files that execute everything as code sequentially; and Notebooks, which consist of a sequence of cells, where each cell is formatted in either Markdown (for writing text) or in a programming language of choice (for writing code). Some advantages of using Kaggle Notebooks as a data science workbench:

- Data sources can be easily added from thousands of publicly available datasets or even uploaded by the user.
- Output files from other Notebooks can also be used as data sources.
- Multiple data sources can be added to the Notebook's environment, allowing joining of multiple interesting datasets.

5.2.3 Python

Python programming language is based on object oriented programming, that has grabbed pervasiveness in light of its sensible syntax and conceivability. Python is said to be moderately easy to learn and advantageous, which implies its announcements can be interpreted in different working systems.

5.3 Libraries required

- **Numpy**: A library for Python, with support for expansive arrays and matrices (may be multi-dimensional in nature), in conjunction with a massive collection of high-level mathematical functions to operate on these arrays, matrices, etc.
- **Librosa**: A python package for audio/music analysis, providing the necessary building blocks to make MIR (music information retrieval) systems. It helps visualize audio signals in multiple ways and also helps perform different levels of feature extraction using different signal processing techniques.
- **Tensorflow**: A machine learning open source platform, providing a comprehensive and flexible ecosystem of community resources, libraries and tools, allowing researchers to deploy the best in Machine Learning, and developers to easily build, run and deploy Machine Learning (ML) powered applications.
- **Keras**: An open-source library written for Python, offering an industry-strength framework that can scale to large clusters of GPUs or even a complete TPU pod. Capable of running above TensorFlow, PlaidML, Theano, etc., it is designed to enable and power quick experimentation with neural networks, while also being modular, user-friendly and extensible.
- **OpenCV**: A library of functions aimed towards computer vision for real-time applications. Developed originally by Intel, later on supported by Willow Garage, then followed by Itseez, the library is free to use under a BSD open-source licens, and is also cross-platform.
- **Pandas**: A library written for Python - for data organisation, analysis and manipulation. Particularly, it offers a range of data structures, statistical functions and operations for manipulating numerical tables. It is free under a three-clause BSD license.
- **Matplotlib**: A Python library for 2D plotting, capable of producing high quality figures for data visualization and analysis.

5.4 Datasets explored

5.4.1 FSDKaggle2019 dataset

The FSDKaggle2019 dataset comprises of audio clips from the following sources.

- Freesound Dataset (FSD): a dataset being collected at MTG-UPF, supported by Freesound content organized with the AudioSet Ontology
- The soundtracks of a Flickr video pool, taken from the Yahoo Flickr Creative Commons 100M dataset (YFCC)

The audio data is labeled employing a vocabulary of 80 labels, all from Google's AudioSet Ontology, covering diverse topics: Guitar and other Musical instruments, Percussion, Water, Digestive, Respiratory sounds, Human voice, Human locomotion, Hands, Human group actions, Insect, Domestic animals, Glass, Liquid, Motor vehicle (road), Mechanisms, Doors, and a variety of Domestic sounds.[4]

1. FSD: The audio content from FSD has been manually labeled by humans following a data labeling process using the Freesound Annotator platform. Most labels have inter-annotator agreement but not all of them.
   - Number of clips/class: 75 except for a few cases (where there are less)
   - Total number of clips: 4970
   - Average number of labels/clip: 1.2
   - Total duration: 10.5 hours

   The duration of each audio clip ranges from 0.3 to 30 seconds.
2. YFCC: The YFCC soundtracks are labeled using automated heuristics, applied to the audio content and metadata of the initial Flickr clips. Hence, a considerable amount of label noise is often expected.
   - Number of clips/class: 300
   - Total number of clips: 19815
   - Average number of labels/clip: 1.2
   - Total duration: Approximately 80 hours

   The duration of the audio clips ranges from 1s to 15s, with many clips lasting 15 seconds.

5.4.2 ESC-50 dataset

The ESC-50 dataset consists of 2000 labeled environmental recordings equally balanced between 50 classes (40 clips per class). For convenience, they are grouped in 5 loosely defined major categories (10 classes per category):

- animal sounds
- natural soundscapes and water sounds
- human sounds (non-speech)
- interior/domestic sounds
- exterior/urban noises

The dataset provides exposure to a range of sound sources, including some very common sounds (laughter, cat meowing, dog barking), some that are distinct in nature (glass breaking, brushing teeth) and some where differences are more subtle (helicopter and airplane noise).[10]
One possible deficit of this dataset is the limited number of clips available per class. This can be attributed to the high cost of manual annotation and extraction, and the judgement to maintain strict balance between classes, despite limited availability of recordings for more exotic sound events.

5.4.3 Audio Cats and Dogs Dataset

The dataset contains many audio files for both the cat and dog classes :

- cat class having 164 WAV files, corresponding to 1323 seconds of audio.
- dog class having 113 WAV files, corresponding to 598 seconds of audio.

All the WAV files are of 16KHz audio and have variable length. This dataset has been extracted from the FREESOUND dataset. It is split into train/test sets, with test containing 96 randomly selected files from both cats and dogs (48 for cats and 48 for dogs). Small datasets usually lead to lower precision in estimates, lower power and they have a much larger risk of skewed results, so this dataset was not made use of.

5.5 Audio Preprocessing

5.5.1 Overview

- There are 2 common ways to represent sound:
  - **Time domain**: An audio signal examined in the time domain has the X-axis as time, and the Y-axis as the signal's changing amplitude with respect to time. Each sample represents variation in air pressure.
  - **Frequency domain**: Examining audio signals in the frequency domain is much more useful, with the X-axis as frequency, and the Y-axis indicating the amplitude for each frequency at each time stamp.

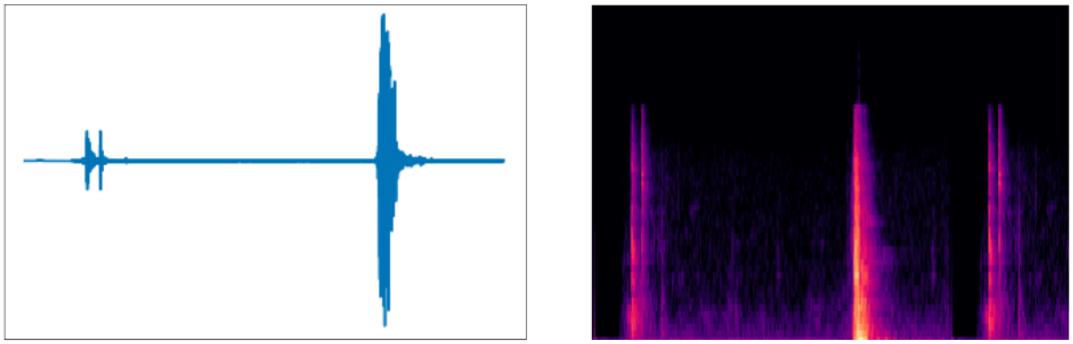


Figure 5.1: Time domain vs Frequency domain

- For our project, the features of interest are encoded in the frequency domain representation of the sound. A spectrogram portrays how frequency content of a signal changes over time. It can be calculated and generated from the time domain signal. The operation, or transformation, used to do that is known as the short time fourier transform (STFT).
- A variation of this frequency spectrum is the Mel Spectrogram. It is a spectrogram with the Y-axis as the Mel Scale. This Mel Spectrogram is what we chose as the frequency representation for the audio samples in consideration. Further these representations are used to generate the images for our image processing experiments in this research study.

5.5.2 Key terms

- **Sampling rate**: In the context of signal processing, sampling refers to the reduction of a continuous time signal, to that of a discrete time signal. Sampling rate/frequency defines the number of samples/second (or an alternative unit) taken from a continuous signal, in order to make the corresponding discrete/digital signal.
- **Fourier transform**: In mathematics, a Fourier transform (FT) is a mathematical transform which decomposes a function (often a function of time, or a signal) into its constituent frequencies, such as the expression of a musical chord with respect to volumes and frequencies of its constituent notes.
- **Short Time Fourier transform**: Short time fourier transforms (STFT) are sequences of fourier transforms on a signal that is windowed. While the standard fourier transform provides frequency information averaged over the entire signal, STFT provides the time-localized frequency information, for instances in which the frequency components of a signal are varied over time.
- **Spectrogram parameters**:
  - **Window**: A discrete-time signal can be sampled with a particular sampling frequency and sampling period. A window defines the width of each decomposed chunk in terms of samples.
  - **NFFT**: Since an FFT can be taken for each chunk, the NFFT tells you how many FFT points are desired to be computed per chunk. In other words, it is the length of the FFT window. A higher number of FFT points would give higher frequency resolution and thus showing fine-grained details along the frequency axis of the spectrogram.
  - **Hop Length**: The number of samples between successive frames, denoted as a positive integer
- **Mel-scale**: The Mel-scale attempts to mimic the non-linear perception of sound by the human ear, being more discriminative at lower frequency sounds, and less discriminative at higher frequency sounds. It is the result of the frequency scale being subject to a non-linear transformation. The Mel scale is made such that sounds of equal distance on the Mel Scale, also “sound” equal in distance from one another to humans.

5.5.3 Preprocessing Steps

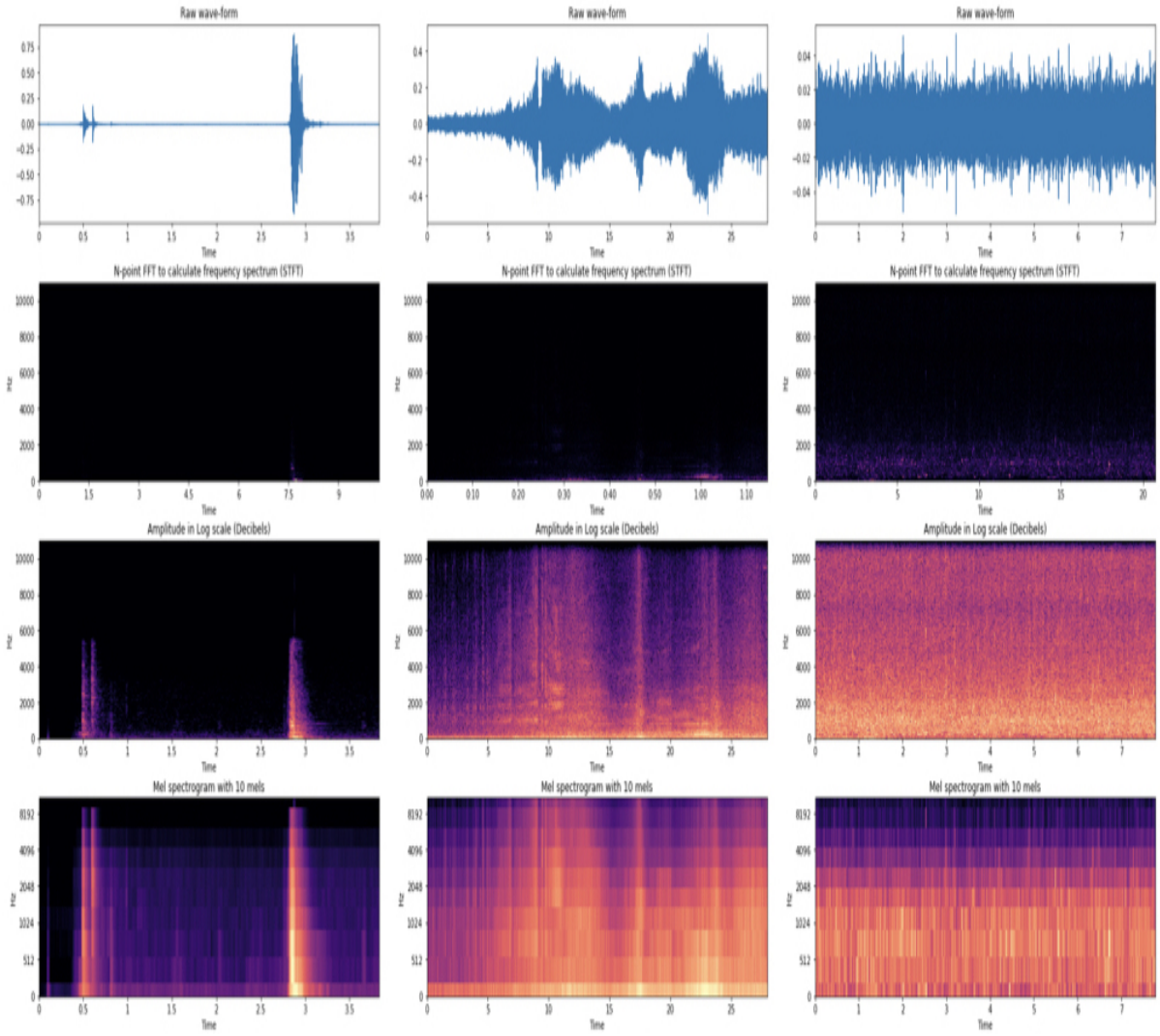


Figure 5.2: Audio Preprocessing

1. The entire dataset of audio samples is trimmed of silence at the starting and end of the clip. This is done to extract the more useful information for the classification process.

2. All audio samples are trimmed to a length of 5 seconds to create an even length audio dataset. Different types of data augmentation is done to handle those clips that are less than the 5 second limit length.

3. The samples are separated into timed windows, with the fourier transform being applied to each time window.

4. Adjustments are made to transform the frequency y-axis to log scale, and the

amplitude "color" axis to Decibels. This represents most sounds heard by humans, concentrated in very small frequency & amplitude ranges.

5. Mel Spectrograms are generated using functions provided by the Librosa library, partitioning the Hz scale into bins, and transforming each bin into a corresponding bin on the Mel Scale using overlapping filters.

5.5.4 Code snippet

```
### load audio file
x,sr = librosa.load(file, sr=44100)
### trim silences
x, _ = librosa.effects.trim(x)
### modify clips to equal 5 second length requirement
x = getUnifiedLength(x)
### generate spectrogram
spectrogram = librosa.feature.melspectrogram(
                    x,sr=specConfig.sampling_rate,
                    n_mels=specConfig.n_mels,
                    hop_length=specConfig.hop_length,
                    n_fft=specConfig.n_fft,
                    fmin=specConfig.fmin,
                    fmax=specConfig.fmax)
logmel = librosa.power_to_db(spectrogram, ref=np.max)
fig = plt.figure()
full_frame(10, 7)
### generate melspectrogram
librosa.display.specshow(
        logmel,
        sr=specConfig.sampling_rate,
        hop_length=specConfig.hop_length,
        x_axis='time',
        y_axis='mel')
### save figure
filename = filename.split('.')[0] + '.png'
plt.savefig(join(OUTPUT_IMG,filename))
```

5.6 Data Augmentation

5.6.1 Overview

Exploiting the fact that not all audio samples in the dataset were of same length, data augmentation could be used to account for the samples that were less than 5 seconds in length, and also increase the size of the dataset in a manner which didn't skew results. The data augmentation was done before the spectrogram generation stage.

All audio clips belonged to 2 categories in the audio preprocessing phase:

- **Longer than or equal to 5 seconds**: these signals were clipped to extract only the sample points corresponding to the first 5 seconds of the audio sample.
- **Shorter than 5 seconds**: these signals were modified using the numpy array padding function and duplicated using two padding modes:
  - **'constant'**: the sample array was padded with a constant value (0) to match a signal length of 5 seconds.
  - **'wrap'**: the sample array was padded with the wrap of the sample vector (repeated) along the x axis.

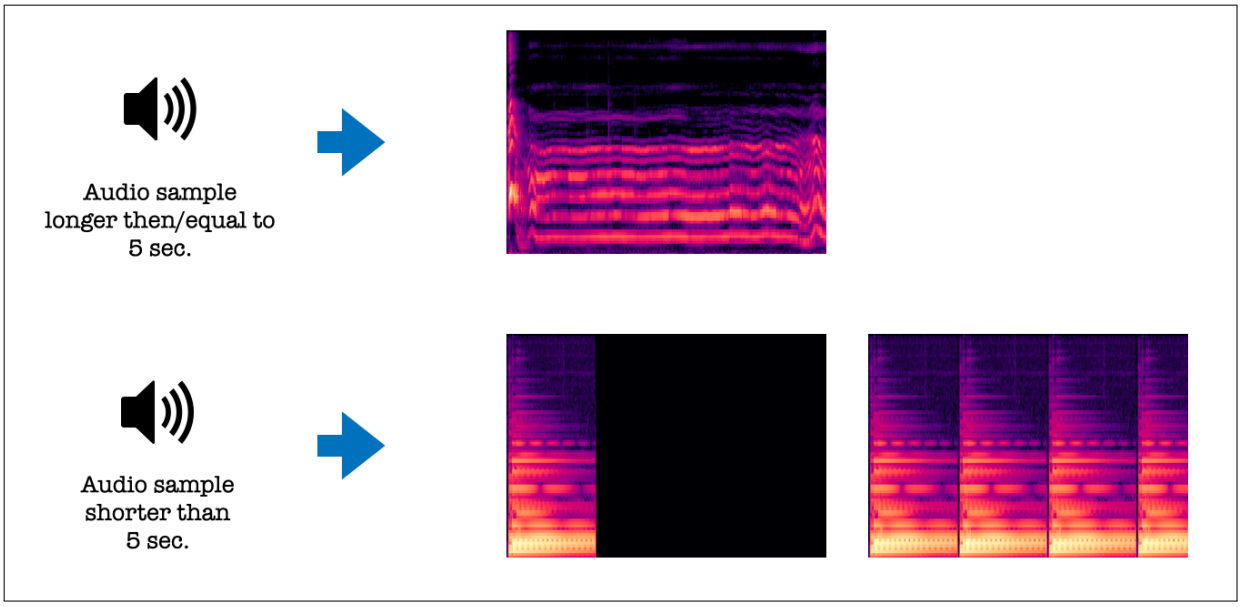


Figure 5.3: Data Augmentation

5.6.2 Code snippet

```
### for audio samples longer than 5 seconds
if len(y) > config.samples:
    y = y[ 0 : 0+config.samples ]

### for audio samples shorter than 5 seconds
else:
    y1 = np.pad(y, (0, config.samples - len(y)), 'constant')
    y2 = np.pad(y, (0, config.samples - len(y)), 'wrap')
```

5.7 Dataset preparation for model

5.7.1 Overview

The dataset preparation phase involves gathering, combining, structuring and organizing the data so it can be fed into the deep learning model. All the generated frequency representation images of the preprocessed audio is structured into folders based on their target labels for the model to perform training easily. These images are also resized and modified to meet the model specific input dimension requirements. The modification more specifically refers to the reduction in dimension of the image to a single channel - grayscale image.

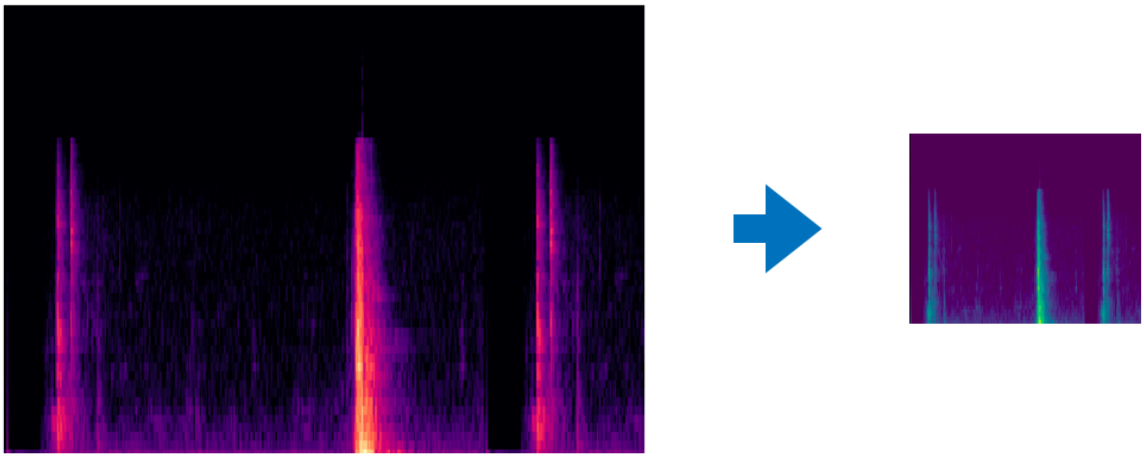


Figure 5.4: Data Preparation

Experimental results have shown that classification with grayscale images results in higher accuracy classification than with RGB images across the different

types of classifiers. There is also the added advantage of reduced computational cost. All this is done using functions provided by the OpenCV library for Python.

5.7.2 Code snippet

```
### for all spectrograms in generated dataset
for image in imagePaths:
    image = cv2.imread(image)

    ### Resize image
    image = cv2.resize(image, (IMAGE_DIMS[1], IMAGE_DIMS[0]))

    ### Convert to grayscale image
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    ### Add to input data array
    image = img_to_array(image)
    data.append(image)
```

5.8 Audio Classification using Spectrogram images

5.8.1 Overview

Deep learning provides very good methods for object recognition in images, making use of multiple artificial neural network layers, with each layer responsible for extracting features of the image in consideration. For this task, computers must first be taught what the target object looks like before it can begin recognition of the same. The more of the object seen by the computer, the better it gets at recognition. This is termed as supervised learning.
Apart from objects, these models are also capable of learning patterns corresponding to some target value, and differentiating them with other patterns, just as they do with objects. Audio files have a single time dimension, and while it is possible to operate on this raw audio waveform data, image classifiers can be used to classify spectrograms and mel-spectrograms. Since spectrograms represent the amplitudes of an audio sample for different frequencies over time, patterns in frequencies present in a spectrogram of a particular sound can be learnt and distinguished from those present in spectrograms of other audio samples.

5.8.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are popular models for Image Segmentation, Classification, Object Detection and many other image processing tasks. They are deep learning networks of processing layers used to reduce an image to its key features so that it can be more easily classified.

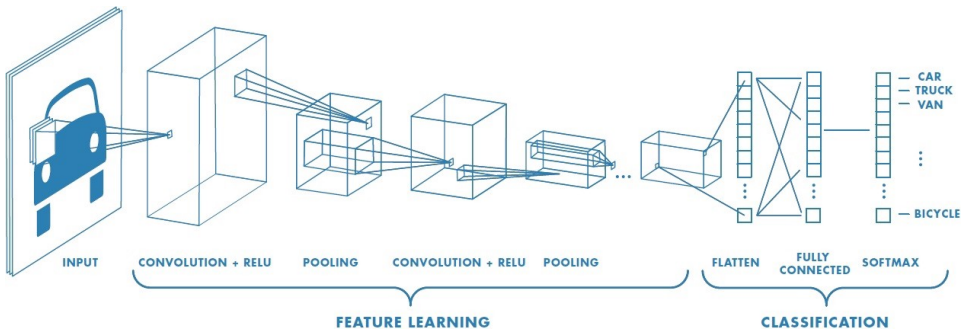


Figure 5.5: Convolutional Neural Networks

Building blocks of CNN's:

- **Convolution layer**: The fundamental component of the CNN architecture which performs feature extraction. It typically consists of combinations of linear and non-linear operations.
- **Convolution**: A special type of linear operation used for feature extraction, A small array of numbers (kernel) is applied across the input, which is an array of numbers (tensor).
- **Feature map**: An element-wise product between the input tensor and each constituent of the kernel, calculated at each tensor location, and then summed to obtain the output value in the corresponding output tensor position.
- **Pooling**: The operation providing a typical downsampling , which reduces the in-plane dimensionality of feature maps. Types of pooling operations include - Max Pooling and Global Average Pooling.
- **Fully connected layer**: Layers in which every input is in turn connected to every output by a corresponding learnable weight. It is used to map extracted features to final network outputs.
- **Last layer activation function**: A non-linear transformation done over the input. This is done before sending it to the next layer of neurons/finalizing it as output. An appropriate task-specific activation function needs to be selected and applied to the last FC (fully connected) layer.

5.9 Architectures explored

5.9.1 MobileNet

MobileNets are efficient models usually used for mobile & embedded-vision applications. They are based on a streamlined architecture that uses depthwise separable convolutions. This is in turn used to build light weight deep neural networks. This deals with not just the image's spatial dimension, but also the depth dimension (number of channels) as well. MobileNets are built to be smaller and faster using width multipliers and resolution multipliers, in the process trading off a reasonable amount of accuracy to reduce size and latency. The role of the width multiplier is to thin out a network at each layer uniformly. Applying the resolution multiplier to the input image, subsequently reduces the internal representation of every layer by the same multiplier.[5]

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 5.6: MobileNet Architecture

The MobileNet architecture provided in the Keras library was used as a baseline model in this research project. It provided evidence that light weight models performed better in the task of spectrogram based image recognition and classification.

Further, since many applications for require the audio tagging system being loaded on embedded systems, small computing devices, sensors, etc., the small MobileNet architecture proved to be a starting point for model performance comparison.

5.9.2 Smaller VGGNet

The VGG network architectures are characterized by their simplicity, but at the same time their large size due to their number of fully-connected layers and depth. They are very common in many image classification problems; however, smaller network architectures are more desirable.

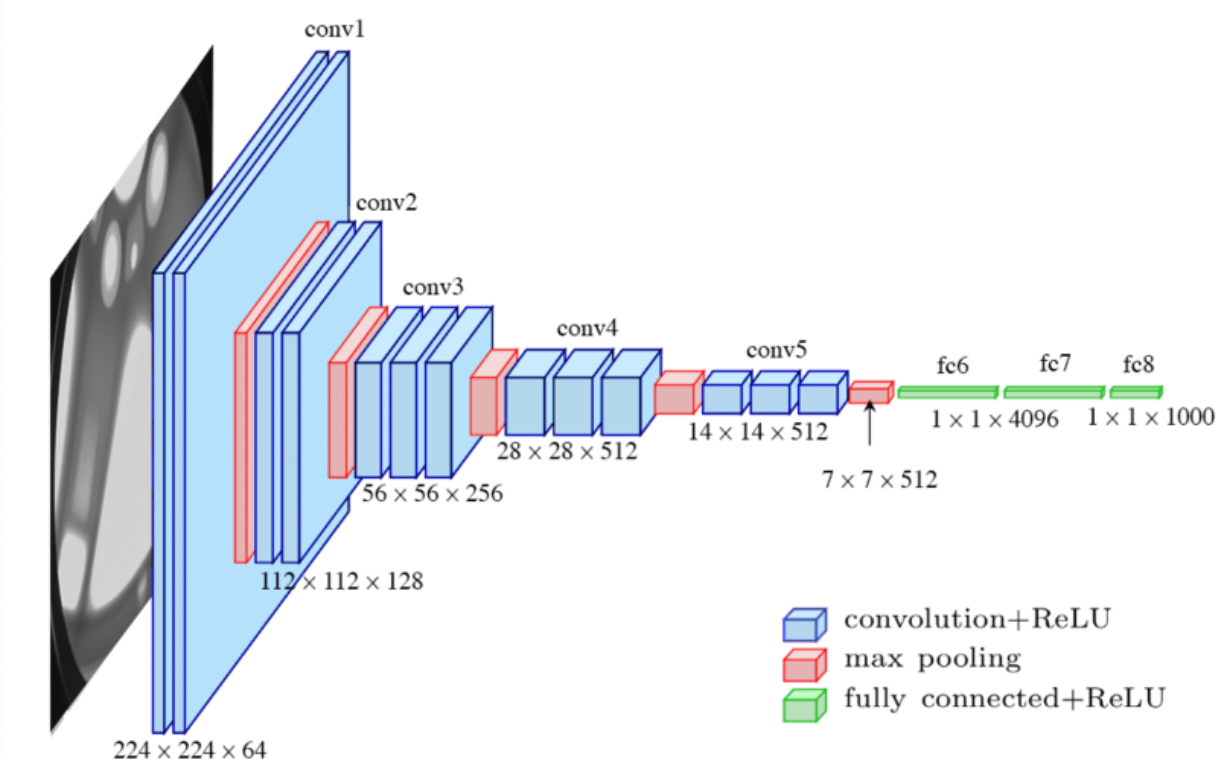


Figure 5.7: Standard VGGNet architecture[3]

The Keras library provides different variations of VGGNet architecture (VGG16 and VGG19) as in-built application functions. However, ever after considering the same operational conditions of the MobileNet, the VGGNet architecture provided poor results on the Audio tagging task. Hence a new custom model was created and experimented with, which was based on the VGGNet. This custom model (Smaller VGGNet) takes inspiration from the architecture of the VGGNet, but is a smaller version and yielded better results for applications of the audio tagging task, surpassing the results of that the baseline MobileNet model.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 150, 200, 32)      896
activation_1 (Activation)    (None, 150, 200, 32)      0
batch_normalization_1 (Batch (None, 150, 200, 32)      128
max_pooling2d_1 (MaxPooling2 (None, 50, 66, 32)        0
dropout_1 (Dropout)          (None, 50, 66, 32)        0
conv2d_2 (Conv2D)            (None, 50, 66, 64)        18496
activation_2 (Activation)    (None, 50, 66, 64)        0
batch_normalization_2 (Batch (None, 50, 66, 64)        256
conv2d_3 (Conv2D)            (None, 50, 66, 64)        36928
activation_3 (Activation)    (None, 50, 66, 64)        0
batch_normalization_3 (Batch (None, 50, 66, 64)        256
max_pooling2d_2 (MaxPooling2 (None, 25, 33, 64)        0
dropout_2 (Dropout)          (None, 25, 33, 64)        0
conv2d_4 (Conv2D)            (None, 25, 33, 128)       73856
activation_4 (Activation)    (None, 25, 33, 128)       0
batch_normalization_4 (Batch (None, 25, 33, 128)       512
conv2d_5 (Conv2D)            (None, 25, 33, 128)       147584
activation_5 (Activation)    (None, 25, 33, 128)       0
batch_normalization_5 (Batch (None, 25, 33, 128)       512
max_pooling2d_3 (MaxPooling2 (None, 12, 16, 128)       0
dropout_3 (Dropout)          (None, 12, 16, 128)       0
flatten_1 (Flatten)          (None, 24576)             0
dense_1 (Dense)              (None, 1024)              25166848
activation_6 (Activation)    (None, 1024)              0
batch_normalization_6 (Batch (None, 1024)              4096
dropout_4 (Dropout)          (None, 1024)              0
dense_2 (Dense)              (None, 12)                12300
activation_7 (Activation)    (None, 12)                0
=================================================================
Total params: 25,462,668
Trainable params: 25,459,788
Non-trainable params: 2,880
```

Figure 5.8: Smaller VGGNet architecture

5.10 Deep learning model - training and fitting

5.10.1 Phase 1 - MobileNet model and the Freesound dataset

The purpose of phase 1 of the project was to check if the selected techniques for the generalised audio tagging task proved to show results on further experimentation. Hence, for this phase, the pretrained Keras MobileNet architecture was considered as the baseline model for result comparisons. The training and fitting of the model was done on the Freesound dataset provided as a part of the Kaggle Audio Tagging Challenge 2019. This dataset contains around 9500 audio clips corresponding to 80 target labels. Post the data augmentation process, the number of images in the spectrogram image dataset corresponding to the FSD dataset was increased to around 16000 spectrogram images.

Importing dependencies and libraries : Some of the dependencies for running the project include libraries such as Keras, Numpy, OpenCV, etc. These libraries need to be imported into all the processing scripts to ensure the implementation pipeline works as expected.

```
from keras import metrics, optimizers, losses, applications
from keras.optimizers import Adam

from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer, LabelEncoder

import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
import copy
import pickle
```

Prepare dataset images for model input : The spectrogram dataset as is contains images that are too large to provided as input to the DL model. Hence, they must be prepared for the same using the OpenCV library for Python. These modifications were explained in section 5.6.

```
data = []
labels = []
for image_path in imagePaths:
    image = cv2.imread(image_pat)
    image = cv2.resize(image, (IMAGE_DIMS[1], IMAGE_DIMS[0]))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = img_to_array(image)
    data.append(image)
    l = label = image_path.split(os.path.sep)[-2]
    labels.append((l,))

data = np.array(data, dtype="float")
```

Encoding target labels : Since the target values in the dataset cannot be text content, there is a need to prepare it before the model can be run. This kind of categorical text data can be converted into model understandable numerical data using the Label Encoder class provided in the sklearn pre-processing library. LabelEncoder encodes labels with values that lie between 0 and n-1, where n is the number of distinct class labels.

```
labels = np.array(labels)
# Create an object of the label encoder class
le = LabelEncoder()
# Encode all labels
labels = le.fit_transform(labels)
```

Splitting into train and validation set : Selecting a proper model allows the generation of accurate results when making a prediction. To do that, there is a need to train the model by using a specific dataset. A validation set is also used to evaluate the performance of the model for different combinations of hyperparameter values and keep the best trained model. All this is done using a functionality provided in the

sklearn library known as the sklearn.model_selection.train_test_split(). This function splits arrays/matrices into random train & test subsets.

```
(train_X, valid_X, train_y, valid_y)
        = train_test_split( data,
                            labels,
                            test_size=0.2,
                            random_state=2018 )
```

Model creation : Keras Applications are deep learning models made available with pre-trained weights. These models can be used for feature extraction, prediction, classification, etc. The pre-trained weights are automatically downloaded when instantiating the model.

```
def top_3_accuracy(x, y):
    return metrics.sparse_top_k_categorical_accuracy(x,y,3)

def create_model():
    model = applications.mobilenet.MobileNet(
                input_shape = (IMAGE_DIMS[0],IMAGE_DIMS[1],1),
                classes = 80,
                weights=None )
    optimizer = optimizers.Adam(lr=LR)
    model.compile(
        optimizer = optimizer,
        loss = losses.sparse_categorical_crossentropy,
        metrics = [ metrics.sparse_categorical_accuracy,
                    top_3_accuracy ]
    )
    model.summary()
    return model

model = create_model()
```

Model fitting : Model fitting is a measure of how well a learning model generalizes to unseen data, which is similar to that on which it was trained. A well-fit model produces more accurate outcomes, while one that is overfitted tends to match the data too closely.

```
fit_results = model.fit(
    train_X, train_y,
    epochs = EPOCHS,
    batch_size = BS,
    validation_data = (valid_X, valid_y),
    verbose=1
)
```

5.10.2 Phase 2 - Smaller VGGNet model and the Animal sounds dataset

Phase 2 of the project corresponds to research of an audio tagging application - the recognition of animals using their respective sounds. A use case of this is in recognising wild animals that cause harm to farm lands, by substituting costly cameras with cheap audio sensors and recorders. For this use case, the audio preprocessing and data preparation steps were the same as before, however a different model was used for experimentation. This model was the Smaller VGGNet mentioned in section 5.8.2.
A small dataset was created by combining multiple animal audio clips from different sources, resulting in roughly 600 corresponding spectrogram images to be used for fitting the model. Some changes were also made in the implementation code with respect to :

Encoding target labels : In most cases of multi label classification, there is a need to encode multiple labels per instance. This can be done by using the Multi Label Binarizer class provided in the sklearn-preprocessing library. This transforms the text labels into a vector, encoding the class(es) present in the image.

```
labels = np.array(labels)
# Create an object of the multi label binarizer class
mlb = MultiLabelBinarizer()
# Binarize all labels
labels = mlb.fit_transform(labels)
```

Model creation :

```
def create_model():
    model = SmallerVGGNet.build(
        width=IMAGE_DIMS[1], height=IMAGE_DIMS[0],
        depth=IMAGE_DIMS[2], classes=len(mlb.classes_),
        finalActivation="sigmoid"
    )
    optimizer = Adam(lr=LR, decay= LR / EPOCHS)
    model.compile(
        loss="binary_crossentropy",
        optimizer=optimizer,
        metrics=["accuracy"]
    )
    model.summary()
    return model

model = create_model()
```

Chapter 6

Result Analysis

6.1 Phase 1 results

Phase 1 was described in section 5.9.1 as an effort to experiment with a generalized processing pipeline, in order to get baseline results for comparison on the audio tagging task. Although this phase involved using only the MobileNet model for experimentation, changes in the different techniques for audio preprocessing and dataset preparation directly affected the results. Some observations are as follows:

- Varying the length of the audio clips considered for the preprocessing showed better results for shorter clips.
- Changes in the spectrogram parameters affected the learning process due to the difference in spectrogram images generated.
- There were improvements in the validation accuracy post the introduction of data augmentation which resulted in the increase in dataset size.

6.1.1 MobileNet with 5 second Freesound Dataset samples

Firstly, the freesound audio clips were experimented upon by limiting the duration to 5 seconds length. No data augmentation was performed other than repeating the signal to match the length requirement. Some of the preprocessing and spectrogram parameters are as follows:

- sampling_rate = 44100
- hop_length = 512
- duration = 5
- n_mels = 64

- nfft = 1024

The resultant dataset used for model fitting was of the following dimensions:

- Training: Data = (9601, 214, 256, 1) and Labels = (9601,)
- Validation: Data = (2401, 214, 256, 1) and Labels = (2401,)

The model was allowed to run for 150 epochs, with a batch size of 64. However at around the halfway point itself it was noticed that both validation and training results showed no improvement with continued training. The maximum top-3 validation accuracy noticed was around 69%. The validation loss also began to increase with the number of epochs increasing. Since audio events are a combination of multiple sounds, the top 3 label classifications were taken into consideration as a suitable metric for evaluation.

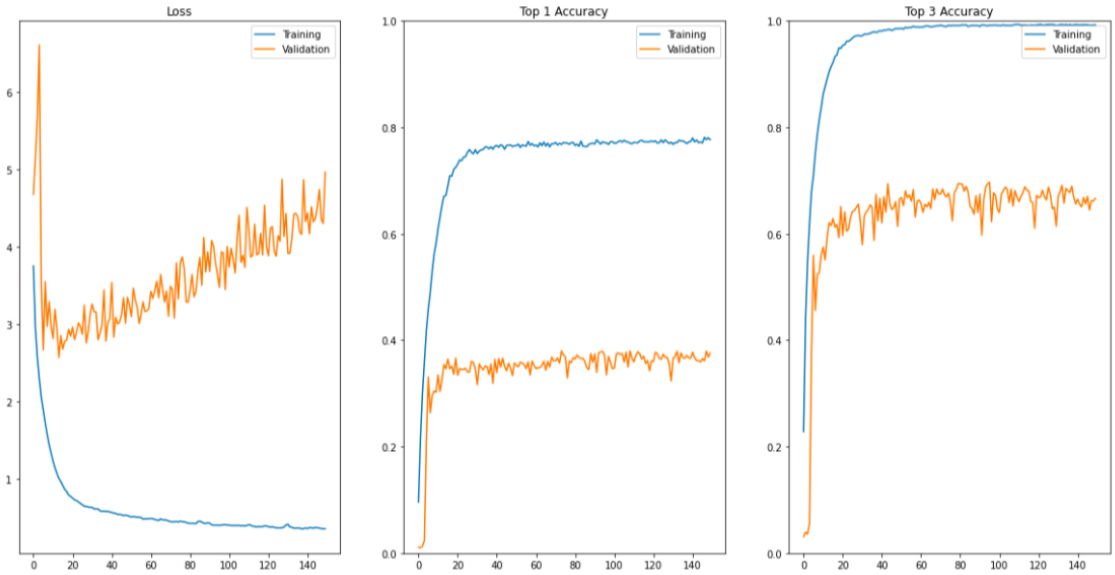


Figure 6.1: MobileNet with 5 second Freesound Dataset samples

6.1.2 MobileNet with 10 second Freesound Dataset samples

The freesound audio clips were experimented upon by limiting the duration to 10 seconds length. No data augmentation was performed other than repeating the signal to match the length requirement. Some experimentation was done using the color channel as well instead of converting the images to grascale. As a result the image dimensions had to be reduced to match computation specifications.

Some of the preprocessing and spectrogram parameters are as follows:

- sampling_rate = 22050
- hop_length = 256
- duration = 10
- n_mels = 96
- nfft = 512

The resultant dataset used for model fitting was of the following dimensions:

- Training: Data = (9601, 80, 200, 3) and Labels = (9601,)
- Validation: Data = (2401, 80, 200, 3) and Labels = (2401,)

After making changes to the dataset, changes were required to be made to the model as well. The model was allowed to run for 50 epochs, using a batch size of 32 images. Due to large image size and use of the color dimension, it was required to make the reduction in the epochs for efficient resource utilization on the kaggle platform. The maximum top-3 validation accuracy noticed in this experiment was around 58%.

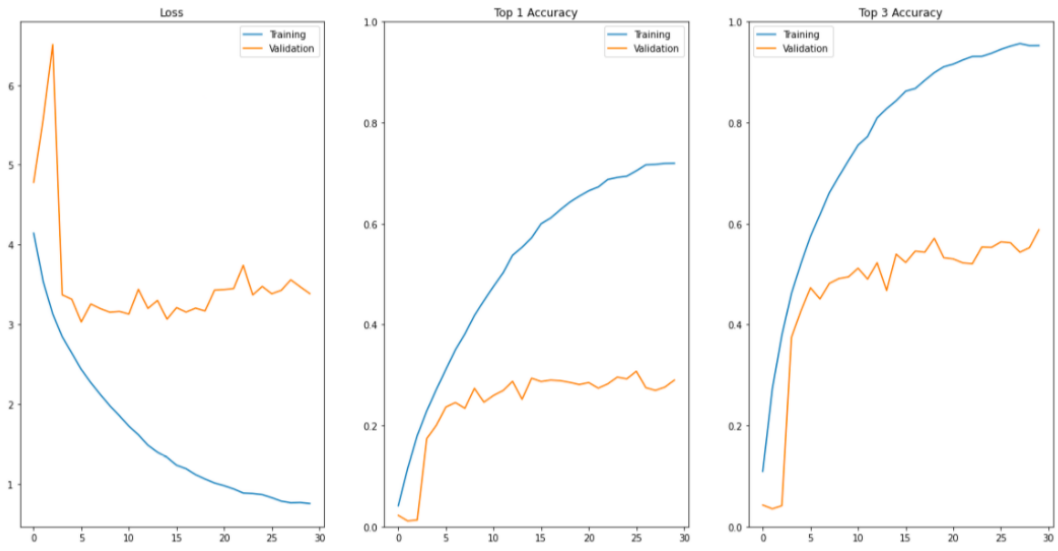


Figure 6.2: MobileNet with 10 second Freesound Dataset samples

6.1.3 MobileNet with 5 second augmented Freesound Dataset samples

Finally, the freesound audio clips were experimented upon by not only limiting the duration to 5 seconds length, but also performing the data augmentation steps mentioned in section 5.5. This resulted in a much larger dataset than before, and hence changes in other parameters were to be made.

The resultant dataset used for model fitting was of the following dimensions:

- Training: Data = (13432, 150, 200, 1) and Labels = (13432,)
- Validation: Data = (3359, 150, 200, 1) and Labels = (3359,)

Similar to the previous case, the model was allowed to run for 50 epochs, but this time a batch size of 70 was used. The maximum top-3 validation accuracy noticed in this experiment was around 75%, which was the best of all experimentation done so far. This proved that the data augmentation steps considered improved the recognition and classification results.

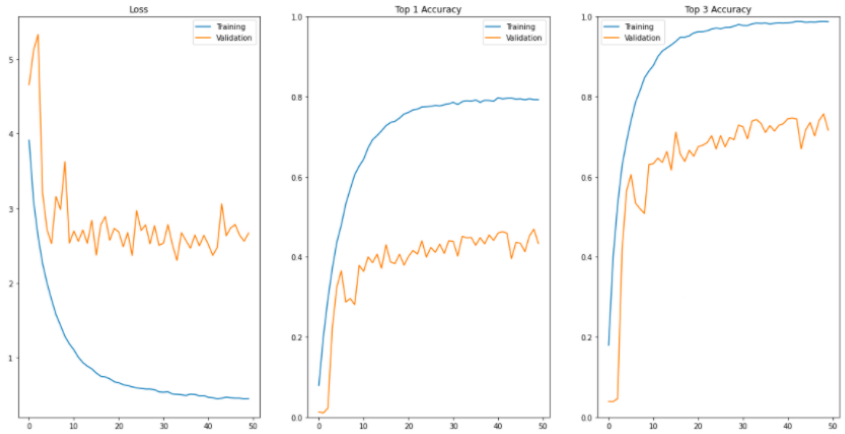


Figure 6.3: MobileNet with 5 second augmented Freesound Dataset samples

6.2 Phase 2 results

Phase 2 was described in section 5.9.2 as an effort to explore an application of the audio tagging task - to recognise and classify animal sounds. This phase involved a change in the use of model to the smaller VGGNet, however most of the proprocessing and data preparation techniques remains the same as previously mentioned. Although the dataset used was comparatively smaller than in the previous cases, hence more susceptible to overfitting, the methods used in this phase and their respective results proved that it was worth further researching on the same.

The resultant dataset used for model fitting was of the following dimensions:

- Training: Data = (466, 150, 200, 3) and Labels = (466,)
- Validation: Data = (117, 150, 200, 3) and Labels = (117,)

The experimentation involved a combination of previous trials, making use of the data augmentation methods to increase dataset size, and also color images for the training purposes. The model was run for a total of 100 epochs with a batch size equal to 30 images. The maximum validation accuracy observed was around 97%, but due to small sized dataset, there is good chance of the results being skewed. Nevertheless this opens doors to further research with bigger datasets, using the current processing pipeline.

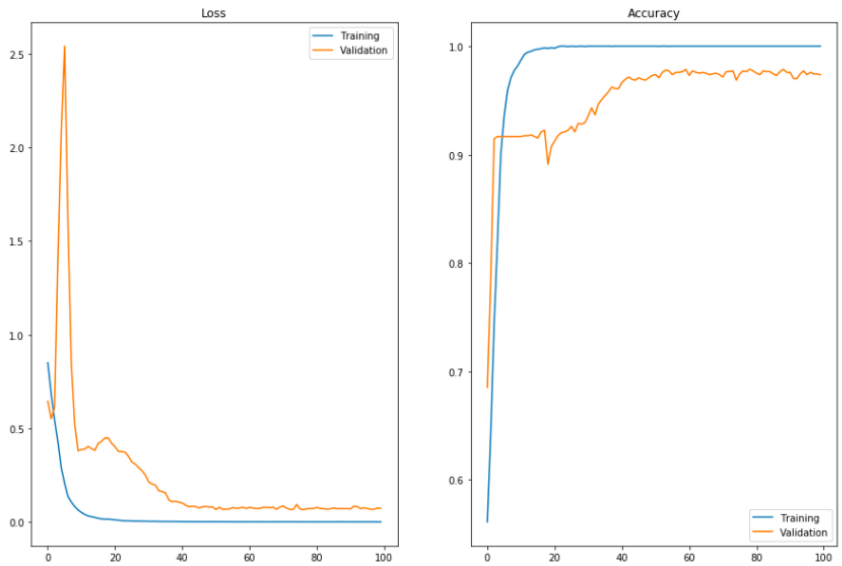


Figure 6.4: Smaller VGGNet with 5 second augmented Animal sound samples

Making use of a few external audio samples, we were able to test the model, observing that the model has learned to distinguish between animals based on the pipeline used.

```
image = cv2.imread('./images/Cow3.png')

# pre-process the image for classification
image = cv2.resize(image, (200, 150))
image = image.astype("float") / 255.0
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image.shape
```

```
(1, 150, 200, 3)
```

```
proba = model.predict(image)
top_3 = mlb.classes_[np.argsort(-proba, axis=1)[:, :3]]
top_3
```

```
array([['Cow', 'Dog', 'Chirping birds']], dtype=object)
```

Figure 6.5: Smaller VGGNet testing results

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

In this project, we successfully demonstrated the working of an automated audio-tagging system using image processing techniques. We explored different methods for converting the analog signal of sound to a digital image as well as showing the effectiveness of augmenting the dataset. We compared the architectures of different pretrained image processing networks. In addition to this, we noted the relevance of transfer learning and their effects on model performance. By observing the results, we proved that such a model can be used for the classification and analysis of sounds on the internet, thereby substituting conventional practices albeit with the same amount of computing resources.

7.2 Future Scope

- Different audio preprocessing techniques and deep learning/image processing models can be experimented with for the task of audio tagging.
- Larger datasets of audio samples can be built and explored for the improvement in results of the models explored, as well as help in further research.
- Transfer learning can be studied using the proposed audio tagging system, which can be used for different applications that require real time audio processing, recognition and classification.

References

[1] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1870–1874. IEEE, 2018.

[2] Aleksandr Diment and Tuomas Virtanen. Transfer learning of weakly labelled audio. In *2017 ieee workshop on applications of signal processing to audio and acoustics (waspaa)*, pages 6–10. IEEE, 2017.

[3] Max Ferguson, Ronay Ak, Yung-Tsun Tina Lee, and Kincho H Law. Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)*, pages 1726–1735. IEEE, 2017.

[4] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel PW Ellis, and Xavier Serra. Audio tagging with noisy labels and minimal supervision. *arXiv preprint arXiv:1906.02975*, 2019.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[6] Turab Iqbal, Qiuqiang Kong, Mark D Plumbley, and Wenwu Wang. General-purpose audio tagging from noisy labels using convolutional neural networks. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018*, pages 212–216. Tampere University of Technology, 2018.

[7] Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D Plumbley. A joint detection-classification model for audio tagging of weakly labelled data. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 641–645. IEEE, 2017.

[8] Donmoon Lee, Subin Lee, Yoonchang Han, and Kyogu Lee. Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input. *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[9] Rashmeet Kaur Nayyar, Sushmita Nair, Omkar Patil, Rasika Pawar, and Amruta Lolage. Content-based auto-tagging of audios using deep learning. In *2017 International Conference on Big Data, IoT and Data Science (BID)*, pages 30–36. IEEE, 2017.

[10] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[11] Kele Xu, Boqing Zhu, Qiuqiang Kong, Haibo Mi, Bo Ding, Dezhi Wang, and Huaimin Wang. General audio tagging with ensembling convolutional neural networks and statistical features. *The Journal of the Acoustical Society of America*, 145(6):EL521–EL527, 2019.

[12] Yong Xu, Qiang Huang, Wenwu Wang, Peter Foster, Siddharth Sigtia, Philip JB Jackson, and Mark D Plumbley. Unsupervised feature learning based on deep models for environmental audio tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1230–1241, 2017.

TEAM INFORMATION

NAME: Rahul Suresh

USN: 1PE16CS188

CONTACT: 8792898757

EMAIL: rahs98@gmail.com

ADDRESS: #39 Sitara, 2nd Cross, Vibhutipura, Bangalore-560037

NAME: Venkatesh B N

USN: 1PE16CS174

CONTACT: 9980171956

EMAIL: venkateshbn98@gmail.com

ADDRESS: 15/1, 2nd Main, Palace Guttahalli, Malleshwaram, Bengaluru-560003

NAME: Ram Singh

USN: 1PE16CS197

CONTACT: 8892183811

EMAIL: ramprincesingh@gmail.com

ADDRESS: Bhavani Road Hebbagodi, Bangalore-560099

NAME: Shruti Danagar

USN: 1PE17CS426

CONTACT: 8147492942

EMAIL: danagarshruti1122@gmail.com

ADDRESS: Ward no 5 saiyad darga galli mantur road mudhol, dist: Bagalakote 587313